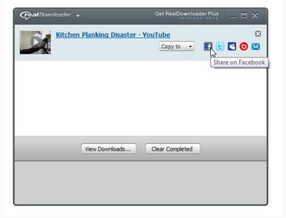
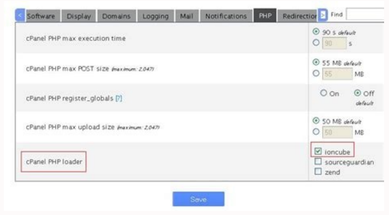
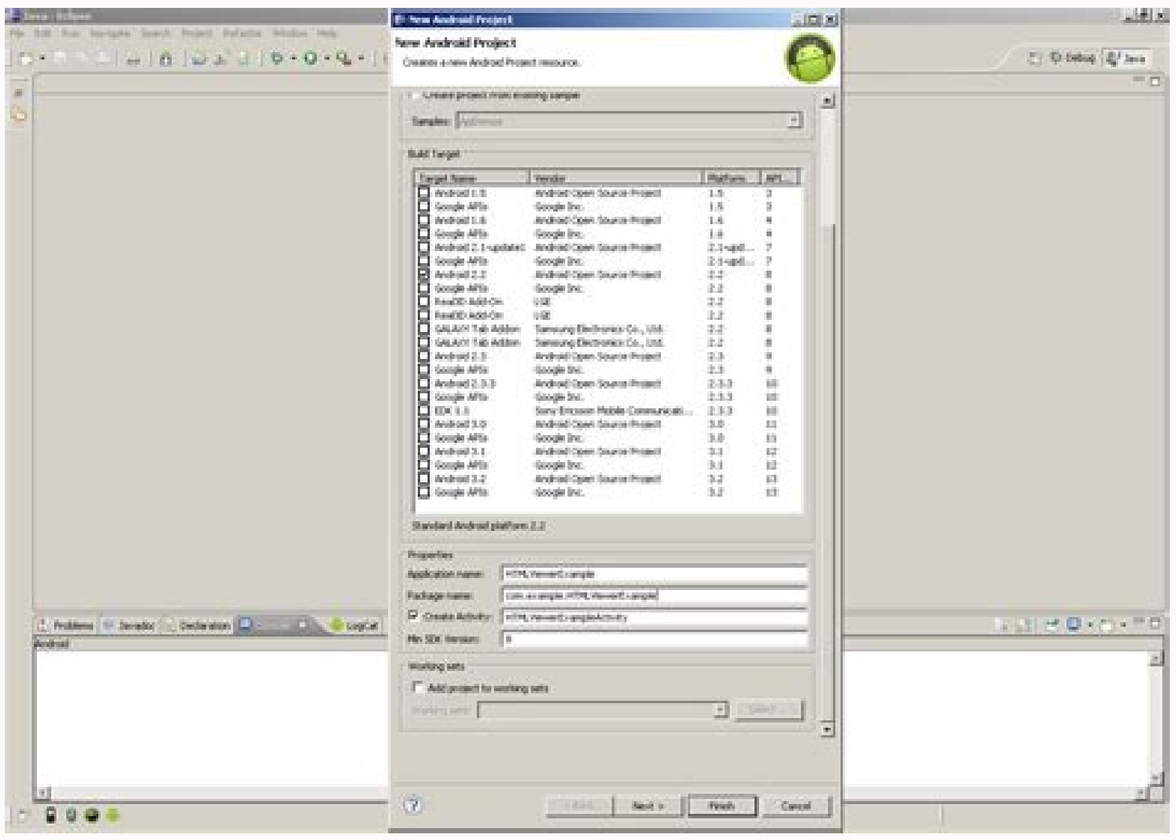
Super loadurl file android_asset www index html

Continue







Android_asset www index html error. Android_asset/index.html. File ///android_asset/www/index.html. File ///android_asset www index html solucion.

A View that displays web pages. In most cases, we recommend using a standard web browser, like Chrome, to deliver content to the user. To learn more about web browsers, read the guide on invoking a browser with an intent. WebView objects allow you to display web content as part of your activity layout, but lack some of the features of fully-developed browsers. A WebView is useful when you need increased control over the UI and advanced configuration options that will allow you to embed web pages in a specially-designed environment for your app. To learn more about WebView and alternatives for serving web content, read the documentation on Web-based content. interface WebView.FindListener Interface to listen for find results. class WebView.HitTestResult interface WebView.PictureListener This interface was deprecated in API level 12. This interface is now obsolete. class WebView.VisualStateCallback Callback interface supplied to WebView.postVisualStateCallback(long, WebView.VisualStateCallback) for receiving notifications about the visual state. class WebView.WebViewTransport Transportation object for returning WebView across thread boundaries. From class android.view.View android:accessibilityHeading Whether or not this view is a heading for accessibility purposes. android:accessibilityLiveRegion Indicates to accessibility services whether the user should be notified when this view changes. android:accessibilityPaneTitle The title this view should present to accessibility as a pane title. android:accessibilityTraversalAfter Sets the id of a view after which this one is visited in accessibility traversal. android:accessibilityTraversalBefore Sets the id of a view before which this one is visited in accessibility traversal. android:allowClickWhenDisabled Whether or not allow clicks on disabled view. android:alpha alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque). android:autoHandwritingEnabled Whether or not the auto handwriting initiation is enabled in this View. android:autofillHints Describes the content of a view so that a autofill service can fill in the appropriate data. android:autofilledHighlight Drawable to be drawn over the view to mark it as autofilled May be a reference to another resource, in the form "@[+][package:]type/name" or a theme attribute in the form "?[package:]type/name". android:background A drawable to use as the background. android:backgroundTint Tint to apply to the background. android:backgroundTintMode Blending mode used to apply the background tint. android:clickable Defines whether this view reacts to click events. android:clipToOutline Whether the View's Outline should be used to clip the contents of the View. android:contentDescription Defines text that briefly describes content of the view. android:contextClickable Defines whether this view reacts to context click events. android:defaultFocusHighlightEnabled Whether this View should use a default focus highlight when it gets focused but doesn't have R.attr.state_focused defined in its background. android:drawingCacheQuality Defines the quality of translucent drawing caches. android:duplicateParentState When this attribute is set to true, the view gets its drawable state (focused, pressed, etc.) from its direct parent rather than from itself. android:elevation base z depth of the view. android:fadeScrollbars Defines whether to fade out scrollbars when they are not in use. android:fadingEdgeLength Defines the length of the fading edges. android:filterTouchesWhenObscured Specifies whether to filter touches when the view's window is obscured by another visible window. android:fitsSystemWindows Boolean internal attribute to adjust view layout based on system windows such as the status bar. android:focusable Controls whether a view can take focus. android:focusableInTouchMode Boolean that controls whether a view can take focus while in touch mode. android:focusedByDefault Whether this view is a default-focus view. android:forceHasOverlappingRendering Whether this view has elements that may overlap when drawn. android:foreground Defines the drawable to draw over the content. android:foregroundGravity Defines the gravity to apply to the foreground drawable. android:foregroundTint Tint to apply to the foreground. android:foregroundTintMode Blending mode used to apply the foreground tint. android:hapticFeedbackEnabled Boolean that controls whether a view should have haptic feedback enabled for events such as long presses. android:id Supply an identifier name for this view, to later retrieve it with View.findViewById() or Activity.findViewById(). android:importantForAccessibility Describes whether or not this view is important for accessibility. android:importantForAutofill Hints the Android System whether the view node associated with this View should be included in a view structure used for autofill purposes. android:importantForContentCapture Hints the Android System whether the view node associated with this View should be use for content capture purposes. android:isScrollContainer Set this if the view will serve as a scrolling container, meaning that it can be resized to shrink its overall window so that there will be space for an input method. android:keepScreenOn Controls whether the view's window should keep the screen on while visible. android:keyboardNavigationCluster Whether this view is a root of a keyboard navigation cluster. android:layerType Specifies the type of layer backing this view. android:layoutDirection Defines the direction of layout drawing. android:longClickable Defines whether this view reacts to long click events. android:minHeight Defines the minimum height of the view. android:minWidth Defines the minimum width of the view. android:nextClusterForward Defines the next keyboard navigation cluster. android:nextFocusDown Defines the next view to give focus to when the next focus is View.FOCUS_DOWN If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed. android:nextFocusForward Defines the next view to give focus to when the next focus is View.FOCUS_FORWARD If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed. android:nextFocusLeft Defines the next view to give focus to when the next focus is View.FOCUS_LEFT. android:nextFocusRight Defines the next view to give focus to when the next focus is View.FOCUS_RIGHT If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed. android:nextFocusUp Defines the next view to give focus to when the next focus is View.FOCUS_UP If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a RuntimeException will result when the reference is accessed. android:onClick Name of the method in this View's context to invoke when the view is clicked. android:outlineAmbientShadowColor Sets the color of the ambient shadow that is drawn when the view has a positive Z or elevation value. android:outlineSpotShadowColor Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value. android:padding Sets the padding, in pixels, of all four edges. android:paddingBottom Sets the padding, in pixels, of the bottom edge; see R.attr.padding. android:paddingEnd Sets the padding, in pixels, of the end edge; see R.attr.padding. android:paddingHorizontal Sets the padding, in pixels, of the left and right edges; see R.attr.padding. android:paddingLeft Sets the padding, in pixels, of the left edge; see R.attr.padding. android:paddingRight Sets the padding, in pixels, of the right edge; see R.attr.padding. android:paddingStart Sets the padding, in pixels, of the start edge; see R.attr.padding. android:paddingTop Sets the padding, in pixels, of the top edge; see R.attr.padding. android:paddingVertical Sets the padding, in pixels, of the top and bottom edges; see R.attr.padding. android:preferKeepClear Sets a preference to keep the bounds of this view clear from floating windows above this view's window. android:requiresFadingEdge Defines which edges should be faded on scrolling. android:rotation rotation of the view, in degrees. android:rotationX rotation of the view around the x axis, in degrees. android:rotationY rotation of the view around the y axis, in degrees. android:saveEnabled If false, no state will be saved for this view when it is being frozen. android:scaleX scale of the view in the x direction. android:scaleY scale of the view in the y direction. android:screenReaderFocusable Whether this view should be treated as a focusable unit by screen reader accessibility tools. android:scrollIndicators Defines which scroll indicators should be displayed when the view can be scrolled. android:scrollX The initial horizontal scroll offset, in pixels. android:scrollY The initial vertical scroll offset, in pixels. android:scrollbarAlwaysDrawHorizontalTrack Defines whether the horizontal scrollbar track should always be drawn. android:scrollbarAlwaysDrawVerticalTrack Defines whether the vertical scrollbar track should always be drawn. android:scrollbarDefaultDelayBeforeFade Defines the delay in milliseconds that a scrollbar waits before fade out. android:scrollbarFadeDuration Defines the delay in milliseconds that a scrollbar takes to fade out. android:scrollbarSize Sets the width of vertical scrollbars and height of horizontal scrollbars. android:scrollbarStyle Controls the scrollbar style and position. android:scrollbarThumbHorizontal Defines the horizontal scrollbar thumb drawable. android:scrollbarThumbVertical Defines the vertical scrollbar thumb drawable. android:scrollbarTrackHorizontal Defines the horizontal scrollbar track drawable. android:scrollbarTrackVertical Defines the vertical scrollbar track drawable. android:scrollbars Defines which scrollbars should be displayed on scrolling or not. android:soundEffectsEnabled Boolean that controls whether a view should have sound effects enabled for events such as clicking and touching. android:stateListAnimator Sets the state-based animator for the View. android:tag Supply a tag for this view containing a String, to be retrieved later with View.getTag() or searched for with View.findViewWithTag(). android:textAlignment Defines the alignment of the text. android:textDirection Defines the direction of the text. android:theme Specifies a theme override for a view. android:tooltipText Defines text displayed in a small popup window on hover or long press. android:transformPivotX x location of the pivot point around which the view will rotate and scale. android:transformPivotY y location of the pivot point around which the view will rotate and scale. android:transitionName Names a View such that it can be identified for Transitions. android:translationX translation in x of the view. android:translationY translation in y of the view. android:translationZ translation in z of the view. android:visibility Controls the initial visibility of the view. From class android.view.ViewGroup int CLIP_TO_PADDING_MASK We clip to padding when FLAG_CLIP_TO_PADDING and FLAG_PADDING_NOT_NULL are set at the same time. int FOCUS_AFTER_DESCENDANTS This view will get focus only if none of its descendants want it. int FOCUS_BEFORE_DESCENDANTS This view will get focus before any of its descendants. int FOCUS_BLOCK_DESCENDANTS This view will block any of its descendants from getting focus, even if they are focusable. int LAYOUT_MODE_CLIP_BOUNDS This constant is a layoutMode. int LAYOUT_MODE_OPTICAL_BOUNDS This constant is a layoutMode. int PERSISTENT_ALL_CACHES This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int PERSISTENT_ANIMATION_CACHE This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int PERSISTENT_NO_CACHE This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int PERSISTENT_SCROLLING_CACHE This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. From class android.view.View int ACCESSIBILITY_LIVE_REGION_ASSERTIVE Live region mode specifying that accessibility services should interrupt ongoing speech to immediately announce changes to this view. int ACCESSIBILITY_LIVE_REGION_NONE Live region mode specifying that accessibility services should not automatically announce changes to this view. int ACCESSIBILITY_LIVE_REGION_POLITE Live region mode specifying that accessibility services should announce changes to this view. int AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS Flag requesting you to add views that are marked as not important for autofill (see setImportantForAutofill(int)) to a ViewStructure. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE Hint indicating that this view can be autofilled with a credit card expiration date. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY Hint indicating that this view can be autofilled with a credit card expiration day. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH Hint indicating that this view can be autofilled with a credit card expiration month. String AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR Hint indicating that this view can be autofilled with a credit card expiration year. String AUTOFILL_HINT_CREDIT_CARD_NUMBER Hint indicating that this view can be autofilled with a credit card number. String AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE Hint indicating that this view can be autofilled with a credit card security code. String AUTOFILL_HINT_EMAIL_ADDRESS Hint indicating that this view can be autofilled with an email address. String AUTOFILL_HINT_NAME Hint indicating that this view can be autofilled with a user's real name. String AUTOFILL_HINT_PASSWORD Hint indicating that this view can be autofilled with a password. String AUTOFILL_HINT_PHONE Hint indicating that this view can be autofilled with a phone number. String AUTOFILL_HINT_POSTAL_ADDRESS Hint indicating that this view can be autofilled with a postal address. String AUTOFILL_HINT_POSTAL_CODE Hint indicating that this view can be autofilled with a postal code. String AUTOFILL_HINT_USERNAME Hint indicating that this view can be autofilled with a username. int AUTOFILL_TYPE_DATE Autofill type for a field that contains a date, which is represented by a long representing the number of milliseconds since the standard base time known as "the epoch", namely January 1, 1970, 00:00:00 GMT (see Date.getTime(). int AUTOFILL_TYPE_LIST Autofill type for a selection list field, which is filled by an int representing the element index inside the list (starting at 0). int AUTOFILL_TYPE_NONE Autofill type for views that cannot be autofilled. int AUTOFILL_TYPE_TEXT Autofill type for a text field, which is filled by a CharSequence. int AUTOFILL_TYPE_TOGGLE Autofill type for a togglable field, which is filled by a boolean. int DRAG_FLAG_ACCESSIBILITY_ACTION Flag indicating that the drag was initiated with AccessibilityNodeInfo.AccessibilityAction#ACTION_DRAG_START. int DRAG_FLAG_GLOBAL Flag indicating that a drag can cross window boundaries. int DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION When this flag is used with DRAG_FLAG_GLOBAL_URI_READ and/or DRAG_FLAG_GLOBAL_URI_WRITE, the URI permission grant can be persisted across device reboots until explicitly revoked with Context.revokeUriPermission(Uri, int) Context.revokeUriPermission}. int DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION When this flag is used with DRAG_FLAG_GLOBAL_URI_READ and/or DRAG_FLAG_GLOBAL_URI_WRITE, the URI permission grant applies to any URI that is a prefix match against the original granted URI. int DRAG_FLAG_GLOBAL_URI_READ When this flag is used with DRAG_FLAG_GLOBAL, the drag recipient will be able to request read access to the content URI(s) contained in the ClipData object. int DRAG_FLAG_GLOBAL_URI_WRITE When this flag is used with DRAG_FLAG_GLOBAL, the drag recipient will be able to request write access to the content URI(s) contained in the ClipData object. int DRAG_FLAG_OPAQUE Flag indicating that the drag shadow will be opaque. int DRAWING_CACHE_QUALITY_AUTO This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int DRAWING_CACHE_QUALITY_HIGH This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int DRAWING_CACHE_QUALITY_LOW This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int FIND_VIEWS_WITH_CONTENT_DESCRIPTION Find find views that contain the specified content description. int FIND_VIEWS_WITH_TEXT Find views that render the specified text. int FOCUSABLE This view wants keystrokes. int FOCUSABLES_ALL View flag indicating whether addFocusables(java.util.ArrayList, int, int) should add all focusable Views regardless if they are focusable in touch mode. int FOCUSABLES_TOUCH_MODE View flag indicating whether addFocusables(java.util.ArrayList, int, int) should add only Views focusable in touch mode. int FOCUSABLE_AUTO This view determines focusability automatically. int FOCUS_BACKWARD Use with focusSearch(int). int FOCUS_DOWN Use with focusSearch(int). int FOCUS_FORWARD Use with focusSearch(int). int FOCUS_LEFT Use with focusSearch(int). int FOCUS_RIGHT Use with focusSearch(int). int FOCUS_UP Use with focusSearch(int). int GONE This view is invisible, and it doesn't take any space for layout purposes. int HAPTIC_FEEDBACK_ENABLED View flag indicating whether this view should have haptic feedback enabled for events such as long presses. int IMPORTANT_FOR_ACCESSIBILITY_AUTO Automatically determine whether a view is important for accessibility. int IMPORTANT_FOR_ACCESSIBILITY_NO The view is not important for accessibility. int IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS The view is not important for accessibility, nor are any of its descendant views. int IMPORTANT_FOR_ACCESSIBILITY_YES The view is important for accessibility. int IMPORTANT_FOR_AUTOFILL_AUTO Automatically determine whether a view is important for autofill. int IMPORTANT_FOR_AUTOFILL_NO The view is not important for autofill, but its children (if any) will be traversed. int IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS The view is not important for autofill, and its children (if any) will not be traversed. int IMPORTANT_FOR_AUTOFILL_YES The view is important for autofill, and its children (if any) will be traversed. int IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS The view is important for autofill, but its children (if any) will not be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_AUTO Automatically determine whether a view is important for content capture. int IMPORTANT_FOR_CONTENT_CAPTURE_NO The view is not important for content capture, but its children (if any) will be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_NO_EXCLUDE_DESCENDANTS The view is not important for content capture, and its children (if any) will not be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_YES The view is important for content capture, and its children (if any) will be traversed. int IMPORTANT_FOR_CONTENT_CAPTURE_YES_EXCLUDE_DESCENDANTS The view is important for content capture, but its children (if any) will not be traversed. int INVISIBLE This view is invisible, but it still takes up space for layout purposes. int KEEP_SCREEN_ON View flag indicating that the screen should remain on while the window containing this view is visible to the user. int LAYER_TYPE_HARDWARE Indicates that the view has a hardware layer. int LAYER_TYPE_NONE Indicates that the view does not have a layer. int LAYER_TYPE_SOFTWARE Indicates that the view has a software layer. int LAYOUT_DIRECTION_INHERIT Horizontal layout direction of this view is inherited from its parent. int LAYOUT_DIRECTION_LOCALE Horizontal layout direction of this view is from deduced from the default language script for the locale. int LAYOUT_DIRECTION_LTR Horizontal layout direction of this view is from Left to Right. int LAYOUT_DIRECTION_RTL Horizontal layout direction of this view is from Right to Left. int MEASURED_HEIGHT_STATE_SHIFT Bit shift of MEASURED_STATE_MASK to get to the height bits for functions that combine both width and height into a single int, such as getMeasuredState() and the childState argument of resolveSizeAndState(int, int, int). int MEASURED_SIZE_MASK Bits of getMeasuredWidthAndState() and getMeasuredWidthAndState() that provide the actual measured size. int MEASURED_STATE_MASK Bits of getMeasuredWidthAndState() and getMeasuredWidthAndState() that provide the additional state bits. int MEASURED_STATE_TOO_SMALL Bit of getMeasuredWidthAndState() and getMeasuredWidthAndState() that indicates the measured size is smaller that the space the view would like to have. int NOT_FOCUSABLE This view does not want keystrokes. int NO_ID Used to mark a View that has no ID. int OVER_SCROLL_ALWAYS Always allow a user to over-scroll this view, provided it is a view that can scroll. int OVER_SCROLL_IF_CONTENT_SCROLLS Allow a user to over-scroll this view only if the content is large enough to meaningfully scroll, provided it is a view that can scroll. int OVER_SCROLL_NEVER Never allow a user to over-scroll this view. int SCREEN_STATE_OFF Indicates that the screen has changed state and is now off. int SCREEN_STATE_ON Indicates that the screen has changed state and is now on. int SCROLLBARS_INSIDE_INSET The scrollbar style to display the scrollbars inside the padded area, increasing the padding of the view. int SCROLLBARS_INSIDE_OVERLAY The scrollbar style to display the scrollbars inside the content area, without increasing the padding. int SCROLLBARS_OUTSIDE_INSET The scrollbar style to display the scrollbars at the edge of the view, increasing the padding of the view. int SCROLLBARS_OUTSIDE_OVERLAY The scrollbar style to display the scrollbars at the edge of the view, without increasing the padding. int SCROLLBAR_POSITION_DEFAULT Position the scroll bar at the default position as determined by the system. int SCROLLBAR_POSITION_LEFT Position the scroll bar along the left edge. int SCROLLBAR_POSITION_RIGHT Position the scroll bar along the right edge. int SCROLL_AXIS_HORIZONTAL Indicates scrolling along the horizontal axis. int SCROLL_AXIS_NONE Indicates no axis of view scrolling. int SCROLL_AXIS_VERTICAL Indicates scrolling along the vertical axis. int SCROLL_CAPTURE_HINT_AUTO The content of this view will be considered for scroll capture if scrolling is possible. int SCROLL_CAPTURE_HINT_EXCLUDE Explicitly exclude this view as a potential scroll capture target. int SCROLL_CAPTURE_HINT_EXCLUDE_DESCENDANTS Explicitly exclude all children of this view as potential scroll capture targets. int SCROLL_CAPTURE_HINT_INCLUDE Explicitly include this view as a potential scroll capture target. int SCROLL_INDICATOR_BOTTOM Scroll indicator direction for the bottom edge of the view. int SCROLL_INDICATOR_END Scroll indicator direction for the ending edge of the view. int SCROLL_INDICATOR_LEFT Scroll indicator direction for the left edge of the view. int SCROLL_INDICATOR_RIGHT Scroll indicator direction for the right edge of the view. int SCROLL_INDICATOR_START Scroll indicator direction for the starting edge of the view. int SCROLL_INDICATOR_TOP Scroll indicator direction for the top edge of the view. int SOUND_EFFECTS_ENABLED View flag indicating whether this view should have sound effects enabled for events such as clicking and touching. int STATUS_BAR_HIDDEN This constant was deprecated in API level 15. Use SYSTEM_UI_FLAG_LOW_PROFILE instead. int STATUS_BAR_VISIBLE This constant was deprecated in API level 15. Use SYSTEM_UI_FLAG_VISIBLE instead. int SYSTEM_UI_FLAG_FULLSCREEN This constant was deprecated in API level 30. Use WindowInsetsController#hide(int) with Type#statusBars() instead. int SYSTEM_UI_FLAG_HIDE_NAVIGATION This constant was deprecated in API level 30. Use WindowInsetsController#hide(int) with Type#navigationBars() instead. int SYSTEM_UI_FLAG_IMMERSIVE This constant was deprecated in API level 30. Use WindowInsetsController#BEHAVIOR_DEFAULT instead. int SYSTEM_UI_FLAG_IMMERSIVE_STICKY This constant was deprecated in API level 30. Use WindowInsetsController#BEHAVIOR_SHOW_TRANSIENT_BARS_BY_SWIPE instead. int SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN This constant was deprecated in API level 30. For floating windows, use LayoutParams#setFitInsetsTypes(int) with Type#statusBars() ()}. For non-floating windows

that fill the screen, call Window#setDecorFitsSystemWindows(boolean) with false. int SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION This constant was deprecated in API level 30. For floating windows, use LayoutParams#setFitInsetsTypes(int) with Type#navigationBars(). For non-floating windows that fill the screen, call Window#setDecorFitsSystemWindows(boolean) with false. int SYSTEM_UI_FLAG_LAYOUT_STABLE This constant was deprecated in API level 30. Use WindowInsets#getInsetsIgnoringVisibility(int) instead to retrieve insets that don't change when system bars change visibility state. int SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR This constant was deprecated in API level 30. Use WindowInsetsController#APPEARANCE_LIGHT_NAVIGATION_BARS instead. int SYSTEM_UI_FLAG_LIGHT_STATUS_BAR This constant was deprecated in API level 30. Use WindowInsetsController#APPEARANCE_LIGHT_STATUS_BARS instead. int SYSTEM_UI_FLAG_LOW_PROFILE This constant was deprecated in API level 30. Low profile mode is deprecated. Hide the system bars instead if the application needs to be in a unobtrusive mode. Use WindowInsetsController#hide(int) with Type#systemBars(). int SYSTEM_UI_FLAG_VISIBLE This constant was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. int SYSTEM_UI_LAYOUT_FLAGS This constant was deprecated in API level 30. System UI layout flags are deprecated. int TEXT_ALIGNMENT_CENTER Center the paragraph, e.g. ALIGN_CENTER. int TEXT_ALIGNMENT_GRAVITY Default for the root view. int TEXT_ALIGNMENT_INHERIT Default text alignment. int TEXT_ALIGNMENT_TEXT_END Align to the end of the paragraph, e.g. ALIGN_OPPOSITE. int TEXT_ALIGNMENT_TEXT_START Align to the start of the paragraph, e.g. ALIGN_NORMAL. int TEXT_ALIGNMENT_VIEW_END Align to the end of the view, which is ALIGN_RIGHT if the view's resolved layoutDirection is LTR, and ALIGN_LEFT otherwise. int TEXT_ALIGNMENT_VIEW_START Align to the start of the view, which is ALIGN_LEFT if the view's resolved layoutDirection is LTR, and ALIGN_RIGHT otherwise. int TEXT_DIRECTION_ANY_RTL Text direction is using "any-RTL" algorithm. int TEXT_DIRECTION_FIRST_STRONG Text direction is using "first strong algorithm". int TEXT_DIRECTION_FIRST_STRONG_LTR Text direction is using "first strong algorithm". int TEXT_DIRECTION_FIRST_STRONG_RTL Text direction is using "first strong algorithm". int TEXT_DIRECTION_INHERIT Text direction is inherited through ViewGroup int TEXT_DIRECTION_LOCALE Text direction is coming from the system Locale. int TEXT_DIRECTION_LTR Text direction is forced to LTR. int TEXT_DIRECTION_RTL Text direction is forced to RTL. String VIEW_LOG_TAG The logging tag used by this class with android.util.Log. int VISIBLE This view is visible. From class android.view.View public static final Property ALPHA A Property wrapper around the alpha functionality handled by the View#setAlpha(float) and View#getAlpha() methods. protected static final int[] EMPTY_STATE_SET Indicates the view has no states set. protected static final int[] ENABLED_FOCUSED_SELECTED_STATE_SET Indicates the view is enabled, focused and selected. protected static final int[] ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is enabled, focused, selected and its window has the focus. protected static final int[] ENABLED_FOCUSED_STATE_SET Indicates the view is enabled and has the focus. protected static final int[] ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates the view is enabled, focused and its window has the focus. protected static final int[] ENABLED_SELECTED_STATE_SET Indicates the view is enabled and selected. protected static final int[] ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is enabled, selected and its window has the focus. protected static final int[] ENABLED_STATE_SET Indicates the view is enabled. protected static final int[] ENABLED_WINDOW_FOCUSED_STATE_SET Indicates the view is enabled and that its window has focus. protected static final int[] FOCUSED_SELECTED_STATE_SET Indicates the view is focused and selected. protected static final int[] FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is focused, selected and its window has the focus. protected static final int[] FOCUSED_STATE_SET Indicates the view is focused. protected static final int[] FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates the view has the focus and that its window has the focus. protected static final int[] PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET Indicates the view is pressed, enabled, focused and selected. protected static final int[] PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, enabled, focused, selected and its window has the focus. protected static final int[] PRESSED_ENABLED_FOCUSED_STATE_SET Indicates the view is pressed, enabled and focused. protected static final int[] PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, enabled, focused and its window has the focus. protected static final int[] PRESSED_ENABLED_SELECTED_STATE_SET Indicates the view is pressed, enabled and selected. protected static final int[] PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, enabled, selected and its window has the focus. protected static final int[] PRESSED_ENABLED_STATE_SET Indicates the view is pressed and enabled. protected static final int[] PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, enabled and its window has the focus. protected static final int[] PRESSED_FOCUSED_SELECTED_STATE_SET Indicates the view is pressed, focused and selected. protected static final int[] PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, focused, selected and its window has the focus. protected static final int[] PRESSED_FOCUSED_STATE_SET Indicates the view is pressed and focused. protected static final int[] PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, focused and its window has the focus. protected static final int[] PRESSED_SELECTED_STATE_SET Indicates the view is pressed and selected. protected static final int[] PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed, selected and its window has the focus. protected static final int[] PRESSED_STATE_SET Indicates the view is pressed. protected static final int[] PRESSED_WINDOW_FOCUSED_STATE_SET Indicates the view is pressed and its window has the focus. public static final Property ROTATION A Property wrapper around the rotation functionality handled by the View#setRotation(float) and View#getRotation() methods. public static final Property ROTATION_X A Property wrapper around the rotationX functionality handled by the View#setRotationX(float) and View#getRotationX() methods. public static final Property ROTATION_Y A Property wrapper around the rotationY functionality handled by the View#setRotationY(float) and View#getRotationY() methods. public static final Property SCALE_X A Property wrapper around the scaleX functionality handled by the View#setScaleX(float) and View#getScaleX() methods. public static final Property SCALE_Y A Property wrapper around the scaleY functionality handled by the View#setScaleY(float) and View#getScaleY() methods. protected static final int[] SELECTED_STATE_SET Indicates the view is selected. protected static final int[] SELECTED_WINDOW_FOCUSED_STATE_SET Indicates the view is selected and that its window has the focus. public static final Property TRANSLATION_X A Property wrapper around the translationX functionality handled by the View#setTranslationX(float) and View#getTranslationX() methods. public static final Property TRANSLATION_Y A Property wrapper around the translationY functionality handled by the View#setTranslationY(float) and View#getTranslationY() methods. public static final Property TRANSLATION_Z A Property wrapper around the translationZ functionality handled by the View#setTranslationZ(float) and View#getTranslationZ() methods. protected static final int[] WINDOW_FOCUSED_STATE_SET Indicates the view's window has focus. public static final Property X A Property wrapper around the x functionality handled by the View#setX(float) and View#getX() methods. public static final Property Y A Property wrapper around the y functionality handled by the View#setY(float) and View#getY() methods. public static final Property Z A Property wrapper around the z functionality handled by the View#setZ(float) and View#getZ() methods. WebView(Context context) Constructs a new WebView with an Activity Context object. WebView(Context context, AttributeSet attrs) Constructs a new WebView with layout parameters. WebView(Context context, AttributeSet attrs, int defStyleAttr) Constructs a new WebView with layout parameters and a default style. WebView(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) Constructs a new WebView with layout parameters and a default style. WebView(Context context, AttributeSet attrs, int defStyleAttr, boolean privateBrowsing) This constructor is deprecated. Private browsing is no longer supported directly via WebView and will be removed in a future release. Prefer using WebSettings, WebViewDatabase, CookieManager and WebStorage for fine-grained control of privacy data. void addJavascriptInterface(Object object, String name) Injects the supplied Java object into this WebView. void autofill(SparseArray values) Automatically fills the content of the virtual children within this view. boolean canGoBack() Gets whether this WebView has a back history item. boolean canGoBackOrForward(int steps) Gets whether the page can go back or forward the given number of steps. boolean canGoForward() Gets whether this WebView has a forward history item. boolean canZoomIn() This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. boolean canZoomOut() This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. Picture capturePicture() This method was deprecated in API level 19. Use onDraw(Canvas) to obtain a bitmap snapshot of the WebView, or saveWebArchive(String) to save the content to a file. void clearCache(boolean includeDiskFiles) Clears the resource cache. static void clearClientCertPreferences(Runnable onCleared) Clears the client certificate preferences stored in response to proceeding/cancelling client cert requests. void clearFormData() Removes the autocomplete popup from the currently focused form field, if present. void clearHistory() Tells this WebView to clear its internal back/forward list. void clearMatches() Clears the highlighting surrounding text matches created by findAllAsync(String). void clearSslPreferences() Clears the SSL preferences table stored in response to proceeding with SSL certificate errors. void clearView() This method was deprecated in API level 18. Use WebView.loadUrl("about:blank") to reliably reset the view state and release page resources (including any running JavaScript). void computeScroll() Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. WebBackForwardList copyBackForwardList() Gets the WebBackForwardList for this WebView. PrintDocumentAdapter createPrintDocumentAdapter(String documentName) Creates a PrintDocumentAdapter that provides the content of this WebView for printing. PrintDocumentAdapter createPrintDocumentAdapter() This method was deprecated in API level 21. Use createPrintDocumentAdapter(java.lang.String) which requires user to provide a print document name. WebMessagePort[] createWebMessageChannel() Creates a message channel to communicate with JS and returns the message ports that represent the endpoints of this message channel. void destroy() Destroys the internal state of this WebView. static void disableWebView() Indicate that the current process does not intend to use WebView, and that an exception should be thrown if a WebView is created or any other methods in the android.webkit package are used. void dispatchCreateViewTranslationRequest(Map viewIds, int[] supportedFormats, TranslationCapability capability, List requests) Dispatch to collect the ViewTranslationRequests for translation purpose by traversing the hierarchy when the app requests ui translation. The implementation calls dispatchCreateViewTranslationRequest(Map, int[], TranslationCapability, List) for all the child views. boolean dispatchKeyEvent(KeyEvent event) Dispatch a key event to the next view on the focus path. void documentHasImages(Message response) Queries the document to see if it contains any image references. static void enableSlowWholeDocumentDraw() For apps targeting the L release, WebView has a new default behavior that reduces memory footprint and increases performance by intelligently choosing the portion of the HTML document that needs to be drawn. void evaluateJavascript(String script, ValueCallback resultCallback) Asynchronously evaluates JavaScript in the context of the currently displayed page. static String findAddress(String addr) This method was deprecated in API level 28. This method is superseded by TextClassifier#generateLinks(android.view.textclassifier.TextLinks.Request). Avoid using this method even when targeting API levels where no alternative is available. int findAll(String find) This method was deprecated in API level 16. findAllAsync(String) is preferred. void findAllAsync(String find) Finds all instances of find on the page and highlights them, asynchronously. View findFocus() Find the view in the hierarchy rooted at this view that currently has focus. void findNext(boolean forward) Highlights and scrolls to the next match found by findAllAsync(String), wrapping around page boundaries as necessary. void flingScroll(int vx, int vy) void freeMemory() This method was deprecated in API level 19. Memory caches are automatically dropped when no longer needed, and in response to system memory pressure. CharSequence getAccessibilityClassName() Return the class name of this object to be used for accessibility purposes. AccessibilityNodeProvider getAccessibilityNodeProvider() Gets the provider for managing a virtual view hierarchy rooted at this View and reported to AccessibilityServices that explore the window content. SslCertificate getCertificate() Gets the SSL certificate for the main top-level page or null if there is no certificate (the site is not secure). int getContentHeight() Gets the height of the HTML content. static PackageInfo getCurrentWebViewPackage() If WebView has already been loaded into the current process this method will return the package that was used to load it. Bitmap getFavicon() Gets the favicon for the current page. Handler getHandler() WebView.HitTestResult getHitTestResult() Gets a HitTestResult based on the current cursor node. String[] getHttpAuthUsernamePassword(String host, String realm) This method was deprecated in API level 26. Use WebViewDatabase#getHttpAuthUsernamePassword instead String getOriginalUrl() Gets the original URL for the current page. int getProgress() Gets the progress for the current page. boolean getRendererPriorityWaivedWhenNotVisible() Return whether this WebView requests a priority of RENDERER_PRIORITY_WAIVED when not visible. int getRendererRequestedPriority() Get the requested renderer priority for this WebView. static Uri getSafeBrowsingPrivacyPolicyUrl() Returns a URL pointing to the privacy policy for Safe Browsing reporting. float getScale() This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. WebSettings getSettings() Gets the WebSettings object used to control the settings for this WebView. TextClassifier getTextClassifier() Returns the TextClassifier used by this WebView. String getTitle() Gets the title for the current page. String getUrl() Gets the URL for the current page. WebChromeClient getWebChromeClient() Gets the chrome handler. static ClassLoader getWebViewClassLoader() Returns the ClassLoader used to load internal WebView classes. WebViewClient getWebViewClient() Gets the WebViewClient. Looper getWebViewLooper() Returns the Looper corresponding to the thread on which WebView calls must be made. WebViewRenderProcess getWebViewRenderProcess() Gets a handle to the WebView renderer process associated with this WebView. WebViewRenderProcessClient getWebViewRenderProcessClient() Gets the renderer client object associated with this WebView. void goBack() Goes back in the history of this WebView. void goBackOrForward(int steps) Goes to the history item that is the number of steps away from the current item. void goForward() Goes forward in the history of this WebView. void invokeZoomPicker() Invokes the graphical zoom picker widget for this WebView. boolean isPrivateBrowsingEnabled() Gets whether private browsing is enabled in this WebView. boolean isVisibleToUserForAutofill(int virtualId) Computes whether this virtual autofill view is visible to the user. void loadData(String data, String mimeType, String encoding) Loads the given data into this WebView using a 'data' scheme URL. void loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl) Loads the given data into this WebView, using baseUrl as the base URL for the content. void loadUrl(String url) Loads the given URL. void loadUrl(String url, Map additionalHttpHeaders) Loads the given URL with additional HTTP headers, specified as a map from name to value. WindowInsets onApplyWindowInsets(WindowInsets insets) Called when the view should apply WindowInsets according to its internal policy. boolean onCheckIsTextEditor() Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. void onChildViewAdded(View parent, View child) This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. void onChildViewRemoved(View p, View child) This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. InputConnection onCreateInputConnection(EditorInfo outAttrs) Creates a new InputConnection for an InputMethod to interact with the WebView. void onCreateVirtualViewTranslationRequests(long[] virtualIds, int[] supportedFormats, Consumer requestsCollector) Collects ViewTranslationRequests which represents the content to be translated for the virtual views in the host view. boolean onDragEvent(DragEvent event) Handles drag events sent by the system following a call to startDragAndDrop(). void onFinishTemporaryDetach() Called after onStartTemporaryDetach() when the container is done changing the view. boolean onGenericMotionEvent(MotionEvent event) Implement this method to handle generic motion events. void onGlobalFocusChanged(View oldFocus, View newFocus) This method is deprecated. WebView should not have implemented ViewTreeObserver.OnGlobalFocusChangeListener. This method does nothing now. boolean onHoverEvent(MotionEvent event) Implement this method to handle hover events. boolean onKeyDown(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyDown(): perform press of the view when KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER is released, if the view is enabled and clickable. boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyMultiple(): always returns false (doesn't handle the event). boolean onKeyUp(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyUp(): perform clicking of the view when KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER or KeyEvent#KEYCODE_SPACE is released. void onPause() Does a best-effort attempt to pause any processing that can be paused safely, such as animations and geolocation. void onProvideAutofillVirtualStructure(ViewStructure structure, int flags) Populates a ViewStructure containing virtual children to fullfil an autofill request. The ViewStructure traditionally represents a View, while for web pages it represent HTML nodes. void onProvideContentCaptureStructure(ViewStructure structure, int flags) Populates a ViewStructure for content capture. void onProvideVirtualStructure(ViewStructure structure) Called when assist structure is being retrieved from a view as part of Activity.onProvideAssistData to generate additional virtual structure under this view. void onResume() Resumes a WebView after a previous call to onPause(). void onStartTemporaryDetach() This is called when a container is going to temporarily detach a child, with ViewGroup.detachViewFromParent. boolean onTouchEvent(MotionEvent event) Implement this method to handle touch screen motion events. boolean onTrackballEvent(MotionEvent event) Implement this method to handle trackball motion events. void onVirtualViewTranslationResponses(LongSparseArray response) Called when the content from View#onCreateVirtualViewTranslationRequests had been translated by the TranslationService. void onWindowFocusChanged(boolean hasWindowFocus) Called when the window containing this view gains or loses focus. boolean overlayHorizontalScrollbar() This method was deprecated in API level 23. This method is now obsolete. boolean overlayVerticalScrollbar() This method was deprecated in API level 23. This method is now obsolete. boolean pageDown(boolean bottom) Scrolls the contents of this WebView down by half the page size. boolean pageUp(boolean top) Scrolls the contents of this WebView up by half the view size. void pauseTimers() Pauses all layout, parsing, and JavaScript timers for all WebViews. boolean performLongClick() Calls this view's OnLongClickListener, if it is defined. void postUrl(String url, byte[] postData) Loads the URL with postData using "POST" method into this WebView. void postVisualStateCallback(long requestId, WebView.VisualStateCallback callback) Posts a VisualStateCallback, which will be called when the current state of the WebView is ready to be drawn. void postWebMessage(WebMessage message, Uri targetOrigin) Post a message to main frame. void reload() Reloads the current URL. void removeJavascriptInterface(String name) Removes a previously injected Java object from this WebView. boolean requestChildRectangleOnScreen(View child, Rect rect, boolean immediate) Called when a child of this group wants a particular rectangle to be positioned onto the screen. boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. Looks for a view to give focus to respecting the setting specified by getDescendantFocusability(). void requestFocusNodeHref(Message hrefMsg) Requests the anchor or image element URL at the last tapped point. void requestImageRef(Message msg) Requests the URL of the image last touched by the user. WebBackForwardList restoreState(Bundle inState) Restores the state of this WebView from the given Bundle. void resumeTimers() Resumes all layout, parsing, and JavaScript timers for all WebViews. void savePassword(String host, String username, String password) This method was deprecated in API level 18. Saving passwords in WebView will not be supported in future versions. WebBackForwardList saveState(Bundle outState) Saves the state of this WebView used in Activity.onSaveInstanceState(Bundle). void saveWebArchive(String filename) Saves the current view as a web archive. void saveWebArchive(String basename, boolean autoname, ValueCallback callback) Saves the current view as a web archive. void setBackgroundColor(int color) Sets the background color for this view. void setCertificate(SslCertificate certificate) This method was deprecated in API level 17. Calling this function has no useful effect, and will be ignored in future releases. static void setDataDirectorySuffix(String suffix) Define the directory used to store WebView data for the current process. void setDownloadListener(DownloadListener listener) Registers the interface to be used when content can not be handled by the rendering engine, and should be downloaded instead. void setFindListener(WebView.FindListener listener) Registers the listener to be notified as find-on-page operations progress. void setHorizontalScrollbarOverlay(boolean overlay) This method was deprecated in API level 23. This method has no effect. void setHttpAuthUsernamePassword(String host, String realm, String username, String password) This method was deprecated in API level 26. Use WebViewDatabase#setHttpAuthUsernamePassword instead void setInitialScale(int scaleInPercent) Sets the initial scale for this WebView. void setLayerType(int layerType, Paint paint) Specifies the type of layer backing this view. void setLayoutParams(ViewGroup.LayoutParams params) Set the layout parameters associated with this view. void setMapTrackballToArrowKeys(boolean setMap) This method was deprecated in API level 17. Only the default case, true, will be supported in a future version. void setNetworkAvailable(boolean networkUp) Informs WebView of the network state. void setOverScrollMode(int mode) Set the over-scroll mode for this view. void setPictureListener(WebView.PictureListener listener) This method was deprecated in API level 15. This method is now obsolete. void setRendererPriorityPolicy(int rendererRequestedPriority, boolean waivedWhenNotVisible) Set the renderer priority policy for this WebView. static void setSafeBrowsingWhitelist(List hosts, ValueCallback callback) Sets the list of hosts (domain names/IP addresses) that are exempt from SafeBrowsing checks. void setScrollBarStyle(int style) Specify the style of the scrollbars. void setTextClassifier(TextClassifier textClassifier) Sets the TextClassifier for this WebView. void setVerticalScrollbarOverlay(boolean overlay) This method was deprecated in API level 23. This method has no effect. void setWebChromeClient(WebChromeClient client) Sets the chrome handler. static void setWebContentsDebuggingEnabled(boolean enabled) Enables debugging of web contents (HTML / CSS / JavaScript) loaded into any WebViews of this application. void setWebViewClient(WebViewClient client) Sets the WebViewClient that will receive various notifications and requests. void setWebViewRenderProcessClient(Executor executor, WebViewRenderProcessClient webViewRenderProcessClient) Sets the renderer client object associated with this WebView. void setWebViewRenderProcessClient(WebViewRenderProcessClient webViewRenderProcessClient) Sets the renderer client object associated with this WebView. boolean shouldDelayChildPressedState() Return true if the pressed state should be delayed for children or descendants of this ViewGroup. boolean showFindDialog(String text, boolean showIme) This method was deprecated in API level 18. This method does not work reliably on all Android versions; implementing a custom find dialog using WebView.findAllAsync() provides a more robust solution. static void startSafeBrowsing(Context context, ValueCallback callback) Starts Safe Browsing initialization. void stopLoading() Stops the current load. void zoomBy(float zoomFactor) Performs a zoom operation in this WebView. boolean zoomIn() Performs zoom in in this WebView. boolean zoomOut() Performs zoom out in this WebView. int computeHorizontalScrollOffset() Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. int computeHorizontalScrollRange() Compute the horizontal range that the horizontal scrollbar represents. int computeVerticalScrollExtent() Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. int computeVerticalScrollOffset() Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range. int computeVerticalScrollRange() Compute the vertical range that the vertical scrollbar represents. void dispatchDraw(Canvas canvas) Called by draw to draw the child views. void onAttachedToWindow() This is called when the view is attached to a window. void onConfigurationChanged(Configuration newConfig) Called when the current configuration of the resources being used by the application have changed. void onDraw(Canvas canvas) Implement this to do your drawing. void onFocusChanged(boolean focused, int direction, Rect previouslyFocusedRect) Called by the view system when the focus state of this view changes. void onMeasure(int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height. void onOverScrolled(int scrollX, int scrollY, boolean clampedX, boolean clampedY) Called by overScrollBy(int, int, int, int, int, int, int, int, boolean) to respond to the results of an over-scroll operation. void onScrollChanged(int l, int t, int oldl, int oldt) This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). void onSizeChanged(int w, int h, int ow, int oh) This is called during layout when the size of this view has changed. void onVisibilityChanged(View changedView, int visibility) Called when the visibility of the view or an ancestor of the view has changed. void onWindowVisibilityChanged(int visibility) Called when the window containing has change its visibility (between GONE, INVISIBLE, and VISIBLE). From class android.view.ViewGroup void addChildrenForAccessibility(ArrayList outChildren) Adds the children of this View relevant for accessibility to the given list as output. void addExtraDataToAccessibilityNodeInfo(AccessibilityNodeInfo info, String extraDataKey, Bundle arguments) Adds extra data to an AccessibilityNodeInfo based on an explicit request for the additional data. void addFocusables(ArrayList views, int direction, int focusableMode) Adds any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. void addKeyboardNavigationClusters(Collection views, int direction) Adds any keyboard navigation cluster roots that are descendants of this view (possibly including this view if it is a cluster root itself) to views. boolean addStatesFromChildren() Returns whether this ViewGroup's drawable states also include its children's drawable states. void addTouchables(ArrayList views) Add any touchable views that are descendants of this view (possibly including this view if it is touchable itself) to views. void addView(View child, ViewGroup.LayoutParams params) Adds a child view with the specified layout parameters. void addView(View child, int index) Adds a child view. void addView(View child, int index, ViewGroup.LayoutParams params) Adds a child view with the specified layout parameters. void addView(View child) Adds a child view. void addView(View child, int width, int height) Adds a child view with this ViewGroup's default layout parameters and the specified width and height. boolean addViewInLayout(View child, int index, ViewGroup.LayoutParams params, boolean preventRequestLayout) Adds a view during layout. boolean addViewInLayout(View child, int index, ViewGroup.LayoutParams params) Adds a view during layout. void attachLayoutAnimationParameters(View child, ViewGroup.LayoutParams params, int index, int count) Subclasses should override this method to set layout animation parameters on the supplied child. void attachViewToParent(View child, int index, ViewGroup.LayoutParams params) Attaches a view to this view group. void bringChildToFront(View child) Change the z order of the child so it's on top of all other children. boolean canAnimate() Indicates whether the view group has the ability to animate its children after the first layout. boolean checkLayoutParams(ViewGroup.LayoutParams p) void childDrawableStateChanged(View child) If addStatesFromChildren() is true, refreshes this group's drawable state (to include the states from its children). void childHasTransientStateChanged(View child, boolean childHasTransientState) Called when a child view has changed whether or not it is tracking transient state. void cleanupLayoutState(View child) Prevents the specified child to be laid out during the next layout pass. void clearChildFocus(View child) Called when a child of this parent is giving up focus void clearDisappearingChildren() Removes any pending animations for views that have been removed. void clearFocus() Called when this view wants to give up focus. void debug(int depth) void detachAllViewsFromParent() Detaches all views from the parent. void detachViewFromParent(int index) Detaches a view from its parent. void detachViewFromParent(View child) Detaches a view from its parent. void detachViewsFromParent(int start, int count) Detaches a range of views from their parents. WindowInsets dispatchApplyWindowInsets(WindowInsets insets) Request to apply the given window insets to this view or another view in its subtree. boolean dispatchCapturedPointerEvent(MotionEvent event) Pass a captured pointer event down to the focused view. void dispatchConfigurationChanged(Configuration newConfig) Dispatch a notification about a resource configuration change down the view hierarchy. void dispatchCreateViewTranslationRequest(Map viewIds, int[] supportedFormats, TranslationCapability capability, List requests) Dispatch to collect the ViewTranslationRequests for translation purpose by traversing the hierarchy when the app requests ui translation. The implementation calls dispatchCreateViewTranslationRequest(Map, int[], TranslationCapability, List) for all the child views. void dispatchDisplayHint(int hint) Dispatch a hint about whether this view is displayed. boolean dispatchDragEvent(DragEvent event) Detects if this View is enabled and has a drag event listener. void dispatchDraw(Canvas canvas) Called by draw to draw the child views. void dispatchDrawableHotspotChanged(float x, float y) Dispatches drawable hotspot changes to child views that meet at least one of the following criteria: void dispatchFreezeSelfOnly(SparseArray container) Perform dispatching of a View.saveHierarchyState(android.util.SparseArray) freeze()} to only this view, not to its children. boolean dispatchGenericFocusedEvent(MotionEvent event) Dispatch a generic motion event to the currently focused view. boolean dispatchGenericPointerEvent(MotionEvent event) Dispatch a generic motion event to the view under the first pointer. boolean dispatchHoverEvent(MotionEvent event) Dispatch a hover event. boolean dispatchKeyEvent(KeyEvent event) Dispatch a key event to the next view on the focus path. boolean dispatchKeyEventPreIme(KeyEvent event) Dispatch a key event before it is processed by any input method associated with the view hierarchy. boolean dispatchKeyShortcutEvent(KeyEvent event) Dispatches a key shortcut event. void dispatchPointerCaptureChanged(boolean hasCapture) void dispatchProvideAutofillStructure(ViewStructure structure, int flags) Dispatches creation of a ViewStructures for autofill purposes down the hierarchy, when an Assist structure is being created as part of an autofill request. This implementation adds in all child views of the view group, in addition to calling the default View implementation. void dispatchProvideStructure(ViewStructure structure) Dispatch creation of ViewStructure down the hierarchy. void dispatchRestoreInstanceState(SparseArray container) Called by restoreHierarchyState(android.util.SparseArray) to retrieve the state for this view and its children. void dispatchSaveInstanceState(SparseArray container) Called by saveHierarchyState(android.util.SparseArray) to store the state for this view and its children. void dispatchScrollCaptureSearch(Rect localVisibleRect, Point windowOffset, Consumer targets) Handle the scroll capture search request by checking this view if applicable, then to each child view. void dispatchSetActivated(boolean activated) Dispatch setActivated to all of this View's children. void dispatchSetPressed(boolean pressed) Dispatch setPressed to all of this View's children. void dispatchSetSelected(boolean selected) Dispatch setSelected to all of this View's children. void dispatchSystemUiVisibilityChanged(int visible) This method is deprecated. Use WindowInsets#isVisible(int) to find out about system bar visibilities by setting a OnApplyWindowInsetsListener on this view. void dispatchThawSelfOnly(SparseArray container) Perform dispatching of a View.restoreHierarchyState(android.util.SparseArray) to only this view, not to its children. boolean dispatchTouchEvent(MotionEvent ev) Pass the touch screen motion event down to the target view, or this view if it is the target. boolean dispatchTrackballEvent(MotionEvent event) Pass a trackball motion event down to the focused view. boolean dispatchUnhandledMove(View focused, int direction) This method is the last chance for the focused view and its ancestors to respond to an arrow key. void dispatchVisibilityChanged(View changedView, int visibility) Dispatch a view visibility change down the view hierarchy. void dispatchWindowFocusChanged(boolean hasFocus) Called when the window containing this view gains or loses window focus. void dispatchWindowInsetsAnimationEnd(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onEnd(WindowInsetsAnimation) when Window Insets animation ends. void dispatchWindowInsetsAnimationPrepare(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onPrepare(WindowInsetsAnimation) when Window Insets animation is being prepared. WindowInsets dispatchWindowInsetsAnimationProgress(WindowInsets insets, List runningAnimations) Dispatches WindowInsetsAnimation.Callback#onProgress(WindowInsets, List) when Window Insets animation makes progress. WindowInsetsAnimation.Bounds dispatchWindowInsetsAnimationStart(WindowInsetsAnimation animation, WindowInsetsAnimation.Bounds bounds) Dispatches WindowInsetsAnimation.Callback#onStart(WindowInsetsAnimation, Bounds) when Window Insets animation is started. void dispatchWindowSystemUiVisiblityChanged(int visible) This method is deprecated. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. void dispatchWindowVisibilityChanged(int visibility) Dispatch a window visibility change down the view hierarchy. boolean drawChild(Canvas canvas, View child, long drawingTime) Draw one child of this View Group. void drawableStateChanged() This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown. void endViewTransition(View view) This method should always be called following an earlier call to startViewTransition(android.view.View). View findFocus() Find the view in the hierarchy rooted at this view that currently has focus. OnBackInvokedDispatcher findOnBackInvokedDispatcherForChild(View child, View requester) Walk up the View hierarchy to find the nearest OnBackInvokedDispatcher. void findViewsWithText(ArrayList outViews, CharSequence text, int flags) Finds the Views that contain given text. View focusSearch(View focused, int direction) Find the nearest view in the specified direction that wants to take focus. void focusableViewAvailable(View v) Tells the parent that a new focusable view has become available. boolean gatherTransparentRegion(Region region) This is used by the ViewRoot to perform an optimization when the view hierarchy contains one or several SurfaceView. ViewGroup.LayoutParams generateDefaultLayoutParams() Returns a set of default layout parameters. ViewGroup.LayoutParams generateLayoutParams(AttributeSet attrs) Returns a new set of layout parameters based on the supplied attributes set. ViewGroup.LayoutParams generateLayoutParams(ViewGroup.LayoutParams p) Returns a safe set of layout parameters based on the supplied layout params. CharSequence getAccessibilityClassName() Return the class name of this object to be used for accessibility purposes. View getChildAt(int index) Returns the view at the specified position in the group. int getChildCount() Returns the number of children in the group. int getChildDrawingOrder(int childCount, int drawingPosition) Converts drawing order position to container position. final int getChildDrawingOrder(int drawingPosition) Converts drawing order position to container position. static int getChildMeasureSpec(int spec, int padding, int childDimension) Does the hard part of measureChildren: figuring out the MeasureSpec to pass to a particular child. boolean getChildStaticTransformation(View child, Transformation t) Sets t to be the static transformation of the child, if set, returning a boolean to indicate whether a static transform was set. boolean getChildVisibleRect(View child, Rect r, Point offset) Compute the visible part of a rectangular region defined in terms of a child view's coordinates. boolean getClipChildren() Returns whether this group's children are clipped to their bounds before drawing. boolean getClipToPadding() Returns whether this ViewGroup will clip its children to its padding, and resize (but not clip) any EdgeEffect to the padded region, if padding is present. int getDescendantFocusability() Gets the descendant focusability of this view group. View getFocusedChild() Returns the focused child of this view, if any. LayoutAnimationController getLayoutAnimation() Returns the layout animation controller used to animate the group's children. Animation.AnimationListener getLayoutAnimationListener() Returns the animation listener to which layout animation events are sent. int getLayoutMode() Returns the basis of alignment during layout operations on this ViewGroup: either LAYOUT_MODE_CLIP_BOUNDS or LAYOUT_MODE_OPTICAL_BOUNDS. LayoutTransition getLayoutTransition() Gets the LayoutTransition object for this ViewGroup. int getNestedScrollAxes() Return the current axes of nested scrolling for this ViewGroup. ViewGroupOverlay getOverlay() Returns the ViewGroupOverlay for this view group, creating it if it does not yet exist. int getPersistentDrawingCache() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. boolean getTouchscreenBlocksFocus() Check whether this ViewGroup should ignore focus requests for itself and its children. boolean hasFocus() Returns true if this view has or contains focus boolean hasTransientState() Indicates whether the view is currently tracking transient state that the app should not need to concern itself with saving and restoring, but that the framework should take special note to preserve when possible. int indexOfChild(View child) Returns the position in the group of the specified child view. final void invalidateChild(View child, Rect dirty) This method is deprecated. Use onDescendantInvalidated(android.view.View, android.view.View) instead to observe updates to draw state in descendants. ViewParent invalidateChildInParent(int[] location, Rect dirty) This method is deprecated. Use onDescendantInvalidated(android.view.View, android.view.View) instead to observe updates to draw state in descendants. boolean isAlwaysDrawnWithCacheEnabled() This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Child views may no longer have their caching behavior disabled by parents. boolean isAnimationCacheEnabled() This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Caching behavior of children may be controlled through View#setLayerType(int, Paint). boolean isChildrenDrawingOrderEnabled() Indicates whether the ViewGroup is drawing its children in the order defined by getChildDrawingOrder(int, int). boolean isChildrenDrawnWithCacheEnabled() This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Child views may no longer be forced to cache their rendering state by their parents. Use View#setLayerType(int, Paint) on individual Views instead. boolean isLayoutSuppressed() Returns whether layout calls on this container are currently being suppressed, due to an earlier call to suppressLayout(boolean). boolean isMotionEventSplittingEnabled() Returns true if MotionEvents dispatched to this ViewGroup can be split to multiple children. boolean isTransitionGroup() Returns true if this ViewGroup should be considered as a single entity for removal when executing an Activity transition. void jumpDrawablesToCurrentState() Call Drawable.jumpToCurrentState() on all Drawable objects associated with this view. final void layout(int l, int t, int r, int b) Assign a size and position to a view and all of its descendants This is the second phase of the layout mechanism. void measureChild(View child, int parentWidthMeasureSpec, int parentHeightMeasureSpec) Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding. void measureChildWithMargins(View child, int parentWidthMeasureSpec, int widthUsed, int parentHeightMeasureSpec, int heightUsed) Ask one of the children of this view to measure itself, taking into account both the MeasureSpec requirements for this view and its padding and margins. void measureChildren(int widthMeasureSpec, int heightMeasureSpec) Ask all of the children of this view to measure themselves, taking into account both the MeasureSpec requirements for this view and its padding. void notifySubtreeAccessibilityStateChanged(View child, View source, int changeType) Notifies a view parent that the accessibility state of one of its descendants has changed and that the structure of the subtree is different. final void offsetDescendantRectToMyCoords(View descendant, Rect rect) Offset a rectangle that is in a descendant's coordinate space into our coordinate space. final void offsetRectIntoDescendantCoords(View descendant, Rect rect) Offset a rectangle that is in our coordinate space into an ancestor's coordinate space. void onAttachedToWindow() This is called when the view is attached to a window. int[] onCreateDrawableState(int extraSpace) Generate the new Drawable state for this view. void onDescendantInvalidated(View child, View target) The target View has been invalidated, or has had a drawing property changed that requires the hierarchy to re-render. If you override this method you must call through to the superclass implementation. void onDetachedFromWindow() This is called when the view is detached from a window. boolean onInterceptHoverEvent(MotionEvent event) Implement this method to intercept hover events before they are handled by child views. boolean onInterceptTouchEvent(MotionEvent ev) Implement this method to intercept all touch screen motion events. abstract void onLayout(boolean changed, int l, int t, int r, int b) Called from layout when this view should assign a size and position to each of its children. boolean onNestedFling(View target, float velocityX, float velocityY, boolean consumed) Request a fling from a nested scroll. boolean onNestedPreFling(View target, float velocityX, float velocityY) React to a nested fling before the target view consumes it. boolean onNestedPrePerformAccessibilityAction(View target, int action, Bundle args) React to an accessibility action delegated by a target descendant view before the target processes it. Subclasses should always call super.onNestedPrePerformAccessibilityAction void onNestedPreScroll(View target, int dx, int dy, int[] consumed) React to a nested scroll in progress before the target view consumes a portion of the scroll. void onNestedScroll(View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) React to a nested scroll in progress. void onNestedScrollAccepted(View child, View target, int axes) React to the successful claiming of a nested scroll operation. boolean onRequestFocusInDescendants(int direction, Rect previouslyFocusedRect) Look for a descendant to call View#requestFocus on. boolean onRequestSendAccessibilityEvent(View child, AccessibilityEvent event) Called when a child has requested sending an AccessibilityEvent and gives an opportunity to its parent to augment the event. PointerIcon onResolvePointerIcon(MotionEvent event, int pointerIndex) Returns the pointer icon for the motion event, or null if it doesn't specify the icon. boolean onStartNestedScroll(View child, View target, int nestedScrollAxes) React to a descendant view initiating a nestable scroll operation, claiming the nested scroll operation if appropriate. void onStopNestedScroll(View child) React to a nested scroll operation ending. void onViewAdded(View child) Called when a new child is added to this ViewGroup. void onViewRemoved(View child) Called when a child view is removed from this ViewGroup. void recomputeViewAttributes(View child) Tell view hierarchy that the global view attributes need to be re-evaluated. void removeAllViews() Call this method to remove all child views from the ViewGroup. void removeAllViewsInLayout() Called by a ViewGroup subclass to remove child views from itself, when it must first know its size on screen before it can calculate how many child views it will render. void removeDetachedView(View child, boolean animate) Finishes the removal of a detached view. void removeView(View view) Note: do not invoke this method from View.draw(android.graphics.Canvas), View.onDraw(android.graphics.Canvas), dispatchDraw(android.graphics.Canvas) or any related method. void removeViewAt(int index) Removes the view at the specified position in the group. void removeViewInLayout(View view) Removes a view during layout. void removeViews(int start, int count) Removes the specified range of views from the group. void removeViewsInLayout(int start, int count) Removes a range of views during layout. void requestChildFocus(View child, View focused) Called when a child of this parent wants focus boolean requestChildRectangleOnScreen(View child, Rect rectangle, boolean immediate) Called when a child of this group wants a particular rectangle to be positioned onto the screen. void requestDisallowInterceptTouchEvent(boolean disallowIntercept) Called when a child does not want this parent and its ancestors to intercept touch events with ViewGroup#onInterceptTouchEvent(MotionEvent). boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. Looks for a view to give focus to respecting the setting specified by getDescendantFocusability(). boolean requestSendAccessibilityEvent(View child, AccessibilityEvent event) Called by a child to request from its parent to send an AccessibilityEvent. void requestTransparentRegion(View child) Called when a child wants the view hierarchy to gather and report transparent regions to the window compositor. boolean restoreDefaultFocus() Gives focus to the default-focus view in the view hierarchy that has this view as a root. void scheduleLayoutAnimation() Schedules the layout animation to be played after the next layout pass of this view group. void setAddStatesFromChildren(boolean addsStates) Sets whether this ViewGroup's drawable states also include its children's drawable states. void setAlwaysDrawnWithCacheEnabled(boolean always) This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Child views may no longer have their caching behavior disabled by parents. void setAnimationCacheEnabled(boolean enabled) This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Caching behavior of children may be controlled through View#setLayerType(int, Paint). void setChildrenDrawingCacheEnabled(boolean enabled) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setChildrenDrawingOrderEnabled(boolean enabled) Tells the ViewGroup whether to draw its children in the order defined by the method getChildDrawingOrder(int, int). void setChildrenDrawnWithCacheEnabled(boolean enabled) This method was deprecated in API level 23. As of Build.VERSION_CODES.M, this property is ignored. Child views may no longer be forced to cache their rendering state by their parents. Use View#setLayerType(int, Paint) on individual Views instead. void setClipChildren(boolean clipChildren) By default, children are clipped to their bounds before drawing. void setClipToPadding(boolean clipToPadding) Sets whether this ViewGroup will clip its children to its padding and resize (but not clip) any EdgeEffect to the padded region, if padding is present. void setDescendantFocusability(int focusability) Set the descendant focusability of this view group. void setLayoutAnimation(LayoutAnimationController controller) Sets the layout animation controller used to animate the group's children after the first layout. void setLayoutAnimationListener(Animation.AnimationListener animationListener) Specifies the animation listener to which layout animation events must be sent. void setLayoutMode(int layoutMode) Sets the basis of alignment during the layout of this ViewGroup. void setLayoutTransition(LayoutTransition transition) Sets the LayoutTransition object for this ViewGroup. void setMotionEventSplittingEnabled(boolean split) Enable or disable the splitting of MotionEvents to multiple children during touch event dispatch. void setOnHierarchyChangeListener(ViewGroup.OnHierarchyChangeListener listener) Register a callback to be invoked when a child is added to or removed from this view. void setPersistentDrawingCache(int drawingCacheToKeep) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, View.setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call View.draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setStaticTransformationsEnabled(boolean enabled) When this property is set to true, this ViewGroup supports static transformations on children; this causes getChildStaticTransformation(android.view.View, android.view.animation.Transformation) to be invoked when a child is drawn. void setTouchscreenBlocksFocus(boolean touchscreenBlocksFocus) Set whether this ViewGroup should ignore focus requests for itself and its children. void setTransitionGroup(boolean isTransitionGroup) Changes whether or not this ViewGroup should be treated as a single entity during Activity Transitions. void setWindowInsetsAnimationCallback(WindowInsetsAnimation.Callback callback) Sets a WindowInsetsAnimation.Callback to be notified about animations of windows that cause insets. boolean shouldDelayChildPressedState() Return true if the pressed state should be delayed for children or descendants of this ViewGroup. boolean showContextMenuForChild(View originalView, float x, float y) Shows the context menu for the specified view or its ancestors anchored to the specified view-relative coordinate. boolean showContextMenuForChild(View originalView) Shows the context menu for the specified view or its ancestors. ActionMode startActionModeForChild(View originalView, ActionMode.Callback callback, int type) Start an action mode of a specific type for the specified view. ActionMode startActionModeForChild(View originalView, ActionMode.Callback callback) Start an action mode for the specified view with the default type ActionMode#TYPE_PRIMARY. void startLayoutAnimation() Runs the layout animation. void startViewTransition(View view) This method tells the ViewGroup that the given View object, which should have this ViewGroup as its parent, should be kept around (re-displayed when the ViewGroup draws its children) even if it is removed from its parent. void suppressLayout(boolean suppress) Tells this ViewGroup to suppress all layout() calls until layout suppression is disabled with a later call to suppressLayout(false). void updateViewLayout(View view, ViewGroup.LayoutParams params) From class android.view.View void addChildrenForAccessibility(ArrayList outChildren) Adds the children of this View relevant for accessibility to the given list as output. void addExtraDataToAccessibilityNodeInfo(AccessibilityNodeInfo info, String extraDataKey, Bundle arguments) Adds extra data to an AccessibilityNodeInfo based on an explicit request for the additional data. void addFocusables(ArrayList views, int direction) Add any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. void addFocusables(ArrayList views, int direction, int focusableMode) Adds any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. void addKeyboardNavigationClusters(Collection views, int direction) Adds any keyboard navigation cluster roots that are descendants of this view (possibly including this view if it is a cluster root itself) to views. void addOnAttachStateChangeListener(View.OnAttachStateChangeListener listener) Add a listener for attach state changes. void addOnLayoutChangeListener(View.OnLayoutChangeListener listener) Add a listener that will be called when the bounds of the view change due to layout processing. void addOnUnhandledKeyEventListener(View.OnUnhandledKeyEventListener listener) Adds a listener which will receive unhandled KeyEvents. void addTouchables(ArrayList views) Add any touchable views that are descendants of this view (possibly including this view if it is touchable itself) to views. ViewPropertyAnimator animate() This method returns a ViewPropertyAnimator object, which can be used to animate specific properties on this View. void announceForAccessibility(CharSequence text) Convenience method for sending a AccessibilityEvent#TYPE_ANNOUNCEMENT AccessibilityEvent to suggest that an accessibility service announce the specified text to its users. void autofill(AutofillValue value) Automatically fills the content of this view with the value. void autofill(SparseArray values) Automatically fills the content of the virtual children within this view. boolean awakenScrollBars(int startDelay, boolean invalidate) Trigger the scrollbars to draw. boolean awakenScrollBars(int startDelay) Trigger the scrollbars to draw. boolean awakenScrollBars() Trigger the scrollbars to draw. void bringToFront() Change the view's z order in the tree, so it's on top of other sibling views. void buildDrawingCache(boolean autoScale) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void buildDrawingCache() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void buildLayer() Forces this view's layer to be created and this view to be rendered into its layer. boolean callOnClick() Directly call any attached OnClickListener. boolean canResolveLayoutDirection() Check if layout direction resolution can be done. boolean canResolveTextAlignment() Check if text alignment resolution can be done. boolean canResolveTextDirection() Check if text direction resolution can be done. boolean canScrollHorizontally(int direction) Check if this view can be scrolled horizontally in a certain direction. boolean canScrollVertically(int direction) Check if this view can be scrolled vertically in a certain direction. final void cancelDragAndDrop() Cancels an ongoing drag and drop operation. void cancelLongPress() Cancels a pending long press. final void cancelPendingInputEvents() Cancel any deferred high-level input events that were previously posted to the event queue. boolean checkInputConnectionProxy(View view) Called by the InputMethodManager when a view who is not the current input connection target is trying to make a call on the manager. void clearAnimation() Cancels any animations for this view. void clearFocus() Called when this view wants to give up focus. void clearViewTranslationCallback() Clear the ViewTranslationCallback from this view. static int combineMeasuredStates(int curState, int newState) Merge two states as returned by getMeasuredState(). int computeHorizontalScrollExtent() Compute the horizontal extent of the horizontal scrollbar's thumb within the horizontal range. int computeHorizontalScrollOffset() Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. int computeHorizontalScrollRange() Compute the horizontal range that the horizontal scrollbar represents. void computeScroll() Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. WindowInsets computeSystemWindowInsets(WindowInsets in, Rect outLocalInsets) Compute insets that should be consumed by this view and the ones that should propagate to those under it. int computeVerticalScrollExtent() Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. int computeVerticalScrollOffset() Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range. int computeVerticalScrollRange() Compute the vertical range that the vertical scrollbar represents. AccessibilityNodeInfo createAccessibilityNodeInfo() Returns an AccessibilityNodeInfo representing this view from the point of view of an AccessibilityService. void createContextMenu(ContextMenu menu) Show the context menu for this view. void destroyDrawingCache() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. WindowInsets dispatchApplyWindowInsets(WindowInsets insets) Request to apply the given window insets to this view or another view in its subtree. boolean dispatchCapturedPointerEvent(MotionEvent event) Pass a captured pointer event down to the focused view. void dispatchConfigurationChanged(Configuration newConfig) Dispatch a notification about a resource configuration change down the view hierarchy. void dispatchCreateViewTranslationRequest(Map viewIds, int[] supportedFormats, TranslationCapability capability, List requests) Dispatch to collect the ViewTranslationRequests for translation purpose by traversing the hierarchy when the app requests ui translation. void dispatchDisplayHint(int hint) Dispatch a hint about whether this view is displayed. boolean dispatchDragEvent(DragEvent event) Detects if this View is enabled and has a drag event listener. void dispatchDraw(Canvas canvas) Called by draw to draw the child views. void dispatchDrawableHotspotChanged(float x, float y) Dispatches drawableHotspotChanged to all of this View's children. void dispatchFinishTemporaryDetach() Dispatch onFinishTemporaryDetach() to this View and its direct children if this is a container View. boolean dispatchGenericFocusedEvent(MotionEvent event) Dispatch a generic motion event to the currently focused view. boolean dispatchGenericMotionEvent(MotionEvent event) Dispatch a generic motion event. boolean dispatchGenericPointerEvent(MotionEvent event) Dispatch a generic motion event to the view under the first pointer. boolean dispatchHoverEvent(MotionEvent event) Dispatch a hover event. boolean dispatchKeyEvent(KeyEvent event) Dispatch a key event to the next view on the focus path. boolean dispatchKeyEventPreIme(KeyEvent event) Dispatch a key event before it is processed by any input method associated with the view hierarchy. boolean dispatchKeyShortcutEvent(KeyEvent event) Dispatches a key shortcut event. boolean dispatchNestedFling(float velocityX, float velocityY, boolean consumed) Dispatch a fling to a nested scrolling parent. boolean dispatchNestedPreFling(float velocityX, float velocityY) Dispatch a fling to a nested scrolling parent before it is processed by this view. boolean dispatchNestedPrePerformAccessibilityAction(int action, Bundle arguments) Report an accessibility action to this view's parents for delegated processing. boolean dispatchNestedPreScroll(int dx, int dy, int[] consumed, int[] offsetInWindow) Dispatch one step of a nested scroll in progress before this view consumes any portion of it. boolean dispatchNestedScroll(int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed, int[] offsetInWindow) Dispatch one step of a nested scroll in progress. void dispatchPointerCaptureChanged(boolean hasCapture) boolean dispatchPopulateAccessibilityEvent(AccessibilityEvent event) Dispatches an AccessibilityEvent to the View first and then to its children for adding their text content to the event. void dispatchProvideAutofillStructure(ViewStructure structure, int flags) Dispatches creation of a ViewStructures for autofill purposes down the hierarchy, when an Assist structure is being created as part of an autofill request. void dispatchProvideStructure(ViewStructure structure) Dispatch creation of ViewStructure down the hierarchy. void dispatchRestoreInstanceState(SparseArray container) Called by restoreHierarchyState(android.util.SparseArray) to retrieve the state for this view and its children. void dispatchSaveInstanceState(SparseArray container) Called by saveHierarchyState(android.util.SparseArray) to store the state for this view and its children. void dispatchScrollCaptureSearch(Rect localVisibleRect, Point windowOffset, Consumer targets) Dispatch a scroll capture search request down the view hierarchy. void dispatchSetActivated(boolean activated) Dispatch setActivated to all of this View's children. void dispatchSetPressed(boolean pressed) Dispatch setPressed to all of this View's children. void dispatchSetSelected(boolean selected) Dispatch setSelected to all of this View's children. void dispatchStartTemporaryDetach() Dispatch onStartTemporaryDetach() to this View and its direct children if this is a container View. void dispatchSystemUiVisibilityChanged(int visible) This method was deprecated in API level 30. Use WindowInsets#isVisible(int) to find out about system bar visibilities by setting a OnApplyWindowInsetsListener on this view. boolean dispatchTouchEvent(MotionEvent event) Pass the touch screen motion event down to the target view, or this view if it is the target. boolean dispatchTrackballEvent(MotionEvent event) Pass a trackball motion event down to the focused view. boolean dispatchUnhandledMove(View focused, int direction) This method is the last chance for the focused view and its ancestors to respond to an arrow key. void dispatchVisibilityChanged(View changedView, int visibility) Dispatch a view visibility change down the view hierarchy. void

dispatchWindowFocusChanged(boolean hasFocus) Called when the window containing this view gains or loses window focus. void dispatchWindowInsetsAnimationEnd(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onEnd(WindowInsetsAnimation) when Window Insets animation ends. void dispatchWindowInsetsAnimationPrepare(WindowInsetsAnimation animation) Dispatches WindowInsetsAnimation.Callback#onPrepare(WindowInsetsAnimation) when Window Insets animation is being prepared. WindowInsets dispatchWindowInsetsAnimationProgress(WindowInsets insets, List runningAnimations) Dispatches WindowInsetsAnimation.Callback#onProgress(WindowInsets, List) when Window Insets animation makes progress. WindowInsetsAnimation.Bounds dispatchWindowInsetsAnimationStart(WindowInsetsAnimation animation, WindowInsetsAnimation.Bounds bounds) Dispatches WindowInsetsAnimation.Callback#onStart(WindowInsetsAnimation, Bounds) when Window Insets animation is started. void dispatchWindowSystemUiVisiblityChanged(int visible) This method was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. void dispatchWindowVisibilityChanged(int visibility) Dispatch a window visibility change down the view hierarchy. void draw(Canvas canvas) Manually render this view (and all of its children) to the given Canvas. void drawableHotspotChanged(float x, float y) This function is called whenever the view hotspot changes and needs to be propagated to drawables or child views managed by the view. void drawableStateChanged() This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown. View findFocus() Find the view in the hierarchy rooted at this view that currently has focus. final OnBackInvokedDispatcher findOnBackInvokedDispatcher() Walk up the View hierarchy to find the nearest OnBackInvokedDispatcher. final T findViewById(int id) Finds the first descendant view with the given ID, the view itself if the ID matches getId(), or null if the ID is invalid (< 0) or there is no matching view in the hierarchy. final T findViewWithTag(Object tag) Look for a child view with the given tag. void findViewsWithText(ArrayList outViews, CharSequence searched, int flags) Finds the Views that contain given text. boolean fitSystemWindows(Rect insets) This method was deprecated in API level 20. As of API 20 use dispatchApplyWindowInsets(android.view.WindowInsets) to apply insets to views. Views should override onApplyWindowInsets(android.view.WindowInsets) or use setOnApplyWindowInsetsListener(android.view.View.OnApplyWindowInsetsListener) to implement handling their own insets. View focusSearch(int direction) Find the nearest view in the specified direction that can take focus. void forceHasOverlappingRendering(boolean hasOverlappingRendering) Sets the behavior for overlapping rendering for this view (see hasOverlappingRendering() for more details on this behavior). void forceLayout() Forces this view to be laid out during the next layout pass. boolean gatherTransparentRegion(Region region) This is used by the ViewRoot to perform an optimization when the view hierarchy contains one or several SurfaceView. void generateDisplayHash(String hashAlgorithm, Rect bounds, Executor executor, DisplayHashResultCallback callback) Called to generate a DisplayHash for this view. static int generateViewId() Generate a value suitable for use in setId(int). CharSequence getAccessibilityClassName() Return the class name of this object to be used for accessibility purposes. View.AccessibilityDelegate getAccessibilityDelegate() Returns the delegate for implementing accessibility support via composition. int getAccessibilityLiveRegion() Gets the live region mode for this View. AccessibilityNodeProvider getAccessibilityNodeProvider() Gets the provider for managing a virtual view hierarchy rooted at this View and reported to AccessibilityServices that explore the window content. CharSequence getAccessibilityPaneTitle() Get the title of the pane for purposes of accessibility. int getAccessibilityTraversalAfter() Gets the id of a view after which this one is visited in accessibility traversal. int getAccessibilityTraversalBefore() Gets the id of a view before which this one is visited in accessibility traversal. float getAlpha() The opacity of the view. Animation getAnimation() Get the animation currently associated with this view. Matrix getAnimationMatrix() Return the current transformation matrix of the view. IBinder getApplicationWindowToken() Retrieve a unique token identifying the top-level "real" window of the window that this view is attached to. int[] getAttributeResolutionStack(int attribute) Returns the ordered list of resource ID that are considered when resolving attribute values for this View. Map getAttributeSourceResourceMap() Returns the mapping of attribute resource ID to source resource ID where the attribute value was set. String[] getAutofillHints() Gets the hints that help an AutofillService determine how to autofill the view with the user's data. final AutofillId getAutofillId() Gets the unique, logical identifier of this view in the activity, for autofill purposes. int getAutofillType() Describes the autofill type of this view, so an AutofillService can create the proper AutofillValue when autofilling the view. AutofillValue getAutofillValue() Gets the View's current autofill value. Drawable getBackground() Gets the background drawable BlendMode getBackgroundTintBlendMode() Return the blending mode used to apply the tint to the background drawable, if specified. ColorStateList getBackgroundTintList() Return the tint applied to the background drawable, if specified. PorterDuff.Mode getBackgroundTintMode() Return the blending mode used to apply the tint to the background drawable, if specified. int getBaseline() Return the offset of the widget's text baseline from the widget's top boundary. final int getBottom() Bottom position of this view relative to its parent. float getBottomFadingEdgeStrength() Returns the strength, or intensity, of the bottom faded edge. int getBottomPaddingOffset() Amount by which to extend the bottom fading region. float getCameraDistance() Gets the distance along the Z axis from the camera to this view. boolean getClipBounds(Rect outRect) Populates an output rectangle with the clip bounds of the view, returning true if successful or false if the view's clip bounds are null. Rect getClipBounds() Returns a copy of the current clipBounds. final boolean getClipToOutline() Returns whether the Outline should be used to clip the contents of the View. final ContentCaptureSession getContentCaptureSession() Gets the session used to notify content capture events. CharSequence getContentDescription() Returns the View's content description. final Context getContext() Returns the context the view is running in, through which it can access the current theme, resources, etc. ContextMenu.ContextMenuInfo getContextMenuInfo() Views should implement this if they have extra information to associate with the context menu. final boolean getDefaultFocusHighlightEnabled() Returns whether this View should use a default focus highlight when it gets focused but doesn't have R.attr.state_focused defined in its background. static int getDefaultSize(int size, int measureSpec) Utility to return a default size. Display getDisplay() Gets the logical display to which the view's window has been attached. final int[] getDrawableState() Return an array of resource IDs of the drawable states representing the current state of the view. Bitmap getDrawingCache() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. Bitmap getDrawingCache(boolean autoScale) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int getDrawingCacheBackgroundColor() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. int getDrawingCacheQuality() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void getDrawingRect(Rect outRect) Return the visible drawing bounds of your view. long getDrawingTime() Return the time at which the drawing of the view hierarchy started. float getElevation() The base elevation of this view relative to its parent, in pixels. int getExplicitStyle() Returns the resource ID for the style specified using style="..." in the AttributeSet's backing XML element or Resources#ID_NULL otherwise if not specified or otherwise not applicable. boolean getFilterTouchesWhenObscured() Gets whether the framework should discard touches when the view's window is obscured by another visible window at the touched location. boolean getFitsSystemWindows() Check for state of setFitsSystemWindows(boolean). int getFocusable() Returns the focusable setting for this view. ArrayList getFocusables(int direction) Find and return all focusable views that are descendants of this view, possibly including this view if it is focusable itself. void getFocusedRect(Rect r) When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method. Drawable getForeground() Returns the drawable used as the foreground of this View. int getForegroundGravity() Describes how the foreground is positioned. BlendMode getForegroundTintBlendMode() Return the blending mode used to apply the tint to the foreground drawable, if specified. ColorStateList getForegroundTintList() Return the tint applied to the foreground drawable, if specified. PorterDuff.Mode getForegroundTintMode() Return the blending mode used to apply the tint to the foreground drawable, if specified. final boolean getGlobalVisibleRect(Rect r) Sets r to the coordinates of the non-clipped area of this view in the coordinate space of the view's root view. boolean getGlobalVisibleRect(Rect r, Point globalOffset) Sets r to the coordinates of the non-clipped area of this view in the coordinate space of the view's root view. Handler getHandler() final boolean getHasOverlappingRendering() Returns the value for overlapping rendering that is used internally. final int getHeight() Return the height of your view. void getHitRect(Rect outRect) Hit rectangle in parent's coordinates int getHorizontalFadingEdgeLength() Returns the size of the horizontal faded edges used to indicate that more content in this view is visible. int getHorizontalScrollbarHeight() Returns the height of the horizontal scrollbar. Drawable getHorizontalScrollbarThumbDrawable() Returns the currently configured Drawable for the thumb of the horizontal scroll bar if it exists, null otherwise. Drawable getHorizontalScrollbarTrackDrawable() Returns the currently configured Drawable for the track of the horizontal scroll bar if it exists, null otherwise. int getId() Returns this view's identifier. int getImportantForAccessibility() Gets the mode for determining whether this View is important for accessibility. int getImportantForAutofill() Gets the mode for determining whether this view is important for autofill. int getImportantForContentCapture() Gets the mode for determining whether this view is important for content capture. boolean getKeepScreenOn() Returns whether the screen should remain on, corresponding to the current value of KEEP_SCREEN_ON. KeyEvent.DispatcherState getKeyDispatcherState() Return the global KeyEvent.DispatcherState for this view's window. int getLabelFor() Gets the id of a view for which this view serves as a label for accessibility purposes. int getLayerType() Indicates what type of layer is currently associated with this view. int getLayoutDirection() Returns the resolved layout direction for this view. ViewGroup.LayoutParams getLayoutParams() Get the LayoutParams associated with this view. final int getLeft() Left position of this view relative to its parent. float getLeftFadingEdgeStrength() Returns the strength, or intensity, of the left faded edge. int getLeftPaddingOffset() Amount by which to extend the left fading region. final boolean getLocalVisibleRect(Rect r) Sets r to the coordinates of the non-clipped area of this view relative to the top left corner of the view. void getLocationInSurface(int[] location) Gets the coordinates of this view in the coordinate space of the Surface that contains the view. void getLocationInWindow(int[] outLocation) Gets the coordinates of this view in the coordinate space of the window that contains the view, irrespective of system decorations. void getLocationOnScreen(int[] outLocation) Gets the coordinates of this view in the coordinate space of the device screen, irrespective of system decorations and whether the system is in multi-window mode. Matrix getMatrix() The transform matrix of this view, which is calculated based on the current rotation, scale, and pivot properties. final int getMeasuredHeight() Like getMeasuredHeightAndState(), but only returns the raw height component (that is the result is masked by MEASURED_SIZE_MASK). final int getMeasuredHeightAndState() Return the full height measurement information for this view as computed by the most recent call to measure(int, int). final int getMeasuredState() Return only the state bits of getMeasuredWidthAndState() and getMeasuredHeightAndState(), combined into one integer. final int getMeasuredWidth() Like getMeasuredWidthAndState(), but only returns the raw width component (that is the result is masked by MEASURED_SIZE_MASK). final int getMeasuredWidthAndState() Return the full width measurement information for this view as computed by the most recent call to measure(int, int). int getMinimumHeight() Returns the minimum height of the view. int getMinimumWidth() Returns the minimum width of the view. int getNextClusterForwardId() Gets the id of the root of the next keyboard navigation cluster. int getNextFocusDownId() Gets the id of the view to use when the next focus is FOCUS_DOWN. int getNextFocusForwardId() Gets the id of the view to use when the next focus is FOCUS_FORWARD. int getNextFocusLeftId() Gets the id of the view to use when the next focus is FOCUS_LEFT. int getNextFocusRightId() Gets the id of the view to use when the next focus is FOCUS_RIGHT. int getNextFocusUpId() Gets the id of the view to use when the next focus is FOCUS_UP. View.OnFocusChangeListener getOnFocusChangeListener() Returns the focus-change callback registered for this view. int getOutlineAmbientShadowColor() ViewOutlineProvider getOutlineProvider() Returns the current ViewOutlineProvider of the view, which generates the Outline that defines the shape of the shadow it casts, and enables outline clipping. int getOutlineSpotShadowColor() int getOverScrollMode() Returns the over-scroll mode for this view. ViewOverlay getOverlay() Returns the overlay for this view, creating it if it does not yet exist. int getPaddingBottom() Returns the bottom padding of this view. int getPaddingEnd() Returns the end padding of this view depending on its resolved layout direction. int getPaddingLeft() Returns the left padding of this view. int getPaddingRight() Returns the right padding of this view. int getPaddingStart() Returns the start padding of this view depending on its resolved layout direction. int getPaddingTop() Returns the top padding of this view. final ViewParent getParent() Gets the parent of this view. ViewParent getParentForAccessibility() Gets the parent for accessibility purposes. float getPivotX() The x location of the point around which the view is rotated and scaled. float getPivotY() The y location of the point around which the view is rotated and scaled. PointerIcon getPointerIcon() Gets the pointer icon for the current view. final List getPreferKeepClearRects() String[] getReceiveContentMimeTypes() Returns the MIME types accepted by performReceiveContent(ContentInfo) for this view, as configured via setOnReceiveContentListener(String[], OnReceiveContentListener). Resources getResources() Returns the resources associated with this view. final boolean getRevealOnFocusHint() Returns this view's preference for reveal behavior when it gains focus. final int getRight() Right position of this view relative to its parent. float getRightFadingEdgeStrength() Returns the strength, or intensity, of the right faded edge. int getRightPaddingOffset() Amount by which to extend the right fading region. AttachedSurfaceControl getRootSurfaceControl() The AttachedSurfaceControl itself is not a View, it is just the interface to the windowing-system object that contains the entire view hierarchy. View getRootView() Finds the topmost view in the current view hierarchy. WindowInsets getRootWindowInsets() Provide original WindowInsets that are dispatched to the view hierarchy. float getRotation() The degrees that the view is rotated around the pivot point. float getRotationX() The degrees that the view is rotated around the horizontal axis through the pivot point. float getRotationY() The degrees that the view is rotated around the vertical axis through the pivot point. float getScaleX() The amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width. float getScaleY() The amount that the view is scaled in y around the pivot point, as a proportion of the view's unscaled height. int getScrollBarDefaultDelayBeforeFade() Returns the delay before scrollbars fade. int getScrollBarFadeDuration() Returns the scrollbar fade duration. int getScrollBarSize() Returns the scrollbar size. int getScrollBarStyle() Returns the current scrollbar style. int getScrollCaptureHint() Returns the current scroll capture hint for this view. int getScrollIndicators() Returns a bitmask representing the enabled scroll indicators. final int getScrollX() Return the scrolled left position of this view. final int getScrollY() Return the scrolled top position of this view. int getSolidColor() Override this if your view is known to always be drawn on top of a solid color background, and needs to draw fading edges. int getSourceLayoutResId() A View can be inflated from an XML layout. final CharSequence getStateDescription() Returns the View's state description. StateListAnimator getStateListAnimator() Returns the current StateListAnimator if exists. int getSuggestedMinimumHeight() Returns the suggested minimum height that the view should use. int getSuggestedMinimumWidth() Returns the suggested minimum width that the view should use. List getSystemGestureExclusionRects() Retrieve the list of areas within this view's post-layout coordinate space where the system should not intercept touch or other pointing device gestures. int getSystemUiVisibility() This method was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. Object getTag() Returns this view's tag. Object getTag(int key) Returns the tag associated with this view and the specified key. int getTextAlignment() Return the resolved text alignment. int getTextDirection() Return the resolved text direction. CharSequence getTooltipText() Returns the view's tooltip text. final int getTop() Top position of this view relative to its parent. float getTopFadingEdgeStrength() Returns the strength, or intensity, of the top faded edge. int getTopPaddingOffset() Amount by which to extend the top fading region. TouchDelegate getTouchDelegate() Gets the TouchDelegate for this View. ArrayList getTouchables() Find and return all touchable views that are descendants of this view, possibly including this view if it is touchable itself. float getTransitionAlpha() This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not side-effect (or get affected by) the real alpha property. String getTransitionName() Returns the name of the View to be used to identify Views in Transitions. float getTranslationX() The horizontal location of this view relative to its left position. float getTranslationY() The vertical location of this view relative to its top position. float getTranslationZ() The depth location of this view relative to its elevation. long getUniqueDrawingId() Get the identifier used for this view by the drawing system. int getVerticalFadingEdgeLength() Returns the size of the vertical faded edges used to indicate that more content in this view is visible. int getVerticalScrollbarPosition() Drawable getVerticalScrollbarThumbDrawable() Returns the currently configured Drawable for the thumb of the vertical scroll bar if it exists, null otherwise. Drawable getVerticalScrollbarTrackDrawable() Returns the currently configured Drawable for the track of the vertical scroll bar if it exists, null otherwise. int getVerticalScrollbarWidth() Returns the width of the vertical scrollbar. ViewTranslationResponse getViewTranslationResponse() Returns the ViewTranslationResponse associated with this view. ViewTreeObserver getViewTreeObserver() Returns the ViewTreeObserver for this view's hierarchy. int getVisibility() Returns the visibility status for this view. final int getWidth() Return the width of your view. int getWindowAttachCount() WindowId getWindowId() Retrieve the WindowId for the window this view is currently attached to. WindowInsetsController getWindowInsetsController() Retrieves the single WindowInsetsController of the window this view is attached to. int getWindowSystemUiVisibility() This method was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. IBinder getWindowToken() Retrieve a unique token identifying the window this view is attached to. int getWindowVisibility() Returns the current visibility of the window this view is attached to (either GONE, INVISIBLE, or VISIBLE). void getWindowVisibleDisplayFrame(Rect outRect) Retrieve the overall visible display size in which the window this view is attached to has been positioned in. float getX() The visual x position of this view, in pixels. float getY() The visual y position of this view, in pixels. float getZ() The visual z position of this view, in pixels. boolean hasExplicitFocusable() Returns true if this view is focusable or if it contains a reachable View for which hasExplicitFocusable() returns true. boolean hasFocus() Returns true if this view has focus itself, or is the ancestor of the view that has focus. boolean hasFocusable() Returns true if this view is focusable or if it contains a reachable View for which hasFocusable() returns true. boolean hasNestedScrollingParent() Returns true if this view has a nested scrolling parent. boolean hasOnClickListeners() Return whether this view has an attached OnClickListener. boolean hasOnLongClickListeners() Return whether this view has an attached OnLongClickListener. boolean hasOverlappingRendering() Returns whether this View has content which overlaps. boolean hasPointerCapture() Checks pointer capture status. boolean hasTransientState() Indicates whether the view is currently tracking transient state that the app should not need to concern itself with saving and restoring, but that the framework should take special note to preserve when possible. boolean hasWindowFocus() Returns true if this view is in a window that currently has window focus. static View inflate(Context context, int resource, ViewGroup root) Inflate a view from an XML resource. void invalidate() Invalidate the whole view. void invalidate(Rect dirty) This method was deprecated in API level 28. The switch to hardware accelerated rendering in API 14 reduced the importance of the dirty rectangle. In API 21 the given rectangle is ignored entirely in favor of an internally-calculated area instead. Because of this, clients are encouraged to just call invalidate(). void invalidate(int l, int t, int r, int b) This method was deprecated in API level 28. The switch to hardware accelerated rendering in API 14 reduced the importance of the dirty rectangle. In API 21 the given rectangle is ignored entirely in favor of an internally-calculated area instead. Because of this, clients are encouraged to just call invalidate(). void invalidateDrawable(Drawable drawable) Invalidates the specified Drawable. void invalidateOutline() Called to rebuild this View's Outline from its outline provider boolean isAccessibilityFocused() Returns whether this View is accessibility focused. boolean isAccessibilityHeading() Gets whether this view is a heading for accessibility purposes. boolean isActivated() Indicates the activation state of this view. boolean isAttachedToWindow() Returns true if this view is currently attached to a window. boolean isAutoHandwritingEnabled() Return whether the View allows automatic handwriting initiation. boolean isClickable() Indicates whether this view reacts to click events or not. boolean isContextClickable() Indicates whether this view reacts to context clicks or not. boolean isDirty() True if this view has changed since the last time being drawn. boolean isDrawingCacheEnabled() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. boolean isDuplicateParentStateEnabled() Indicates whether this duplicates its drawable state from its parent. boolean isEnabled() Returns the enabled status for this view. final boolean isFocusable() Returns whether this View is currently able to take focus. final boolean isFocusableInTouchMode() When a view is focusable, it may not want to take focus when in touch mode. boolean isFocused() Returns true if this view has focus final boolean isFocusedByDefault() Returns whether this View should receive focus when the focus is restored for the view hierarchy containing this view. boolean isForceDarkAllowed() See setForceDarkAllowed(boolean) boolean isHapticFeedbackEnabled() boolean isHardwareAccelerated() Indicates whether this view is attached to a hardware accelerated window or not. boolean isHorizontalFadingEdgeEnabled() Indicate whether the horizontal edges are faded when the view is scrolled horizontally. boolean isHorizontalScrollBarEnabled() Indicate whether the horizontal scrollbar should be drawn or not. boolean isHovered() Returns true if the view is currently hovered. boolean isImportantForAccessibility() Computes whether this view should be exposed for accessibility. final boolean isImportantForAutofill() Hints the Android System whether the AssistStructure.ViewNode associated with this view is considered important for autofill purposes. final boolean isImportantForContentCapture() Hints the Android System whether this view is considered important for content capture, based on the value explicitly set by setImportantForContentCapture(int) and heuristics when it's IMPORTANT_FOR_CONTENT_CAPTURE_AUTO. boolean isInEditMode() Indicates whether this View is currently in edit mode. boolean isInLayout() Returns whether the view hierarchy is currently undergoing a layout pass. boolean isInTouchMode() Returns whether the device is currently in touch mode. final boolean isKeyboardNavigationCluster() Returns whether this View is a root of a keyboard navigation cluster. boolean isLaidOut() Returns true if this view has been through at least one layout since it was last attached to or detached from a window. boolean isLayoutDirectionResolved() boolean isLayoutRequested() Indicates whether or not this view's layout will be requested during the next hierarchy layout pass. boolean isLongClickable() Indicates whether this view reacts to long click events or not. boolean isNestedScrollingEnabled() Returns true if nested scrolling is enabled for this view. boolean isOpaque() Indicates whether this View is opaque. boolean isPaddingOffsetRequired() If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. boolean isPaddingRelative() Return if the padding has been set through relative values setPaddingRelative(int, int, int, int) or through boolean isPivotSet() Returns whether or not a pivot has been set by a call to setPivotX(float) or setPivotY(float). final boolean isPreferKeepClear() Retrieve the preference for this view to be kept clear. boolean isPressed() Indicates whether the view is currently in pressed state. boolean isSaveEnabled() Indicates whether this view will save its state (that is, whether its onSaveInstanceState() method will be called). boolean isSaveFromParentEnabled() Indicates whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. boolean isScreenReaderFocusable() Returns whether the view should be treated as a focusable unit by screen reader accessibility tools. boolean isScrollContainer() Indicates whether this view is one of the set of scrollable containers in its window. boolean isScrollbarFadingEnabled() Returns true if scrollbars will fade when this view is not scrolling boolean isSelected() Indicates the selection state of this view. final boolean isShowingLayoutBounds() Returns true when the View is attached and the system developer setting to show the layout bounds is enabled or false otherwise. boolean isShown() Returns the visibility of this view and all of its ancestors boolean isSoundEffectsEnabled() final boolean isTemporarilyDetached() Tells whether the View is in the state between onStartTemporaryDetach() and onFinishTemporaryDetach(). boolean isTextAlignmentResolved() boolean isTextDirectionResolved() boolean isVerticalFadingEdgeEnabled() Indicate whether the vertical edges are faded when the view is scrolled horizontally. boolean isVerticalScrollBarEnabled() Indicate whether the vertical scrollbar should be drawn or not. boolean isVisibleToUserForAutofill(int virtualId) Computes whether this virtual autofill view is visible to the user. void jumpDrawablesToCurrentState() Call Drawable.jumpToCurrentState() on all Drawable objects associated with this view. View keyboardNavigationClusterSearch(View currentCluster, int direction) Find the nearest keyboard navigation cluster in the specified direction. void layout(int l, int t, int r, int b) Assign a size and position to a view and all of its descendants This is the second phase of the layout mechanism. final void measure(int widthMeasureSpec, int heightMeasureSpec) This is called to find out how big a view should be. static int[] mergeDrawableStates(int[] baseState, int[] additionalState) Merge your own state values in additionalState into the base state values baseState that were returned by onCreateDrawableState(int). void offsetLeftAndRight(int offset) Offset this view's horizontal location by the specified amount of pixels. void offsetTopAndBottom(int offset) Offset this view's vertical location by the specified number of pixels. void onAnimationEnd() Invoked by a parent ViewGroup to notify the end of the animation currently associated with this view. void onAnimationStart() Invoked by a parent ViewGroup to notify the start of the animation currently associated with this view. WindowInsets onApplyWindowInsets(WindowInsets insets) Called when the view should apply WindowInsets according to its internal policy. void onAttachedToWindow() This is called when the view is attached to a window. void onCancelPendingInputEvents() Called as the result of a call to cancelPendingInputEvents() on this view or a parent view. boolean onCapturedPointerEvent(MotionEvent event) Implement this method to handle captured pointer events boolean onCheckIsTextEditor() Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. void onConfigurationChanged(Configuration newConfig) Called when the current configuration of the resources being used by the application have changed. void onCreateContextMenu(ContextMenu menu) Views should implement this if the view itself is going to add items to the context menu. int[] onCreateDrawableState(int extraSpace) Generate the new Drawable state for this view. InputConnection onCreateInputConnection(EditorInfo outAttrs) Create a new InputConnection for an InputMethod to interact with the view. void onCreateViewTranslationRequest(int[] supportedFormats, Consumer requestsCollector) Collects a ViewTranslationRequest which represents the content to be translated in the view. void onCreateVirtualViewTranslationRequests(long[] virtualIds, int[] supportedFormats, Consumer requestsCollector) Collects ViewTranslationRequests which represents the content to be translated for the virtual views in the host view. void onDetachedFromWindow() This is called when the view is detached from a window. void onDisplayHint(int hint) Gives this view a hint about whether is displayed or not. boolean onDragEvent(DragEvent event) Handles drag events sent by the system following a call to startDragAndDrop(). void onDraw(Canvas canvas) Implement this to do your drawing. void onDrawForeground(Canvas canvas) Draw any foreground content for this view. final void onDrawScrollBars(Canvas canvas) Request the drawing of the horizontal and the vertical scrollbar. boolean onFilterTouchEventForSecurity(MotionEvent event) Filter the touch event to apply security policies. void onFinishInflate() Finalize inflating a view from XML. void onFinishTemporaryDetach() Called after onStartTemporaryDetach() when the container is done changing the view. void onFocusChanged(boolean gainFocus, int direction, Rect previouslyFocusedRect) Called by the view system when the focus state of this view changes. boolean onGenericMotionEvent(MotionEvent event) Implement this method to handle generic motion events. void onHoverChanged(boolean hovered) Implement this method to handle hover state changes. boolean onHoverEvent(MotionEvent event) Implement this method to handle hover events. void onInitializeAccessibilityEvent(AccessibilityEvent event) Initializes an AccessibilityEvent with information about this View which is the event source. void onInitializeAccessibilityNodeInfo(AccessibilityNodeInfo info) Initializes an AccessibilityNodeInfo with information about this view. boolean onKeyDown(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyDown(): perform press of the view when KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER is released, if the view is enabled and clickable. boolean onKeyLongPress(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyLongPress(): always returns false (doesn't handle the event). boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyMultiple(): always returns false (doesn't handle the event). boolean onKeyPreIme(int keyCode, KeyEvent event) Handle a key event before it is processed by any input method associated with the view hierarchy. boolean onKeyShortcut(int keyCode, KeyEvent event) Called on the focused view when a key shortcut event is not handled. boolean onKeyUp(int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyUp(): perform clicking of the view when KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER or KeyEvent#KEYCODE_SPACE is released. void onLayout(boolean changed, int left, int top, int right, int bottom) Called from layout when this view should assign a size and position to each of its children. void onMeasure(int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height. void onOverScrolled(int scrollX, int scrollY, boolean clampedX, boolean clampedY) Called by overScrollBy(int, int, int, int, int, int, int, int, boolean) to respond to the results of an over-scroll operation. void onPointerCaptureChange(boolean hasCapture) Called when the window has just acquired or lost pointer capture. void onPopulateAccessibilityEvent(AccessibilityEvent event) Called from dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent) giving a chance to this View to populate the accessibility event with its text content. void onProvideAutofillStructure(ViewStructure structure, int flags) Populates a ViewStructure to fullfil an autofill request. void onProvideAutofillVirtualStructure(ViewStructure structure, int flags) Populates a ViewStructure containing virtual children to fullfil an autofill request. void onProvideContentCaptureStructure(ViewStructure structure, int flags) Populates a ViewStructure for content capture. void onProvideStructure(ViewStructure structure) Called when assist structure is being retrieved from a view as part of Activity.onProvideAssistData. void onProvideVirtualStructure(ViewStructure structure) Called when assist structure is being retrieved from a view as part of Activity.onProvideAssistData to generate additional virtual structure under this view. ContentInfo onReceiveContent(ContentInfo payload) Implements the default behavior for receiving content for this type of view. PointerIcon onResolvePointerIcon(MotionEvent event, int pointerIndex) Returns the pointer icon for the motion event, or null if it doesn't specify the icon. void onRestoreInstanceState(Parcelable state) Hook allowing a view to re-apply a representation of its internal state that had previously been generated by onSaveInstanceState(). void onRtlPropertiesChanged(int layoutDirection) Called when any RTL property (layout direction or text direction or text alignment) has been changed. Parcelable onSaveInstanceState() Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. void onScreenStateChanged(int screenState) This method is called whenever the state of the screen this view is attached to changes. void onScrollCaptureSearch(Rect localVisibleRect, Point windowOffset, Consumer targets) Called when scroll capture is requested, to search for appropriate content to scroll. void onScrollChanged(int l, int t, int oldl, int oldt) This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). boolean onSetAlpha(int alpha) Invoked if there is a Transform that involves alpha. void onSizeChanged(int w, int h, int oldw, int oldh) This is called during layout when the size of this view has changed. void onStartTemporaryDetach() This is called when a container is going to temporarily detach a child, with ViewGroup.detachViewFromParent. boolean onTouchEvent(MotionEvent event) Implement this method to handle touch screen motion events. boolean onTrackballEvent(MotionEvent event) Implement this method to handle trackball motion events. void onViewTranslationResponse(ViewTranslationResponse response) Called when the content from View#onCreateViewTranslationRequest had been translated by the TranslationService. void onVirtualViewTranslationResponses(LongSparseArray response) Called when the content from View#onCreateVirtualViewTranslationRequests had been translated by the TranslationService. void onVisibilityAggregated(boolean isVisible) Called when the user-visibility of this View is potentially affected by a change to this view itself, an ancestor view or the window this view is attached to. void onVisibilityChanged(View changedView, int visibility) Called when the visibility of the view or an ancestor of the view has changed. void onWindowFocusChanged(boolean hasWindowFocus) Called when the window containing this view gains or loses focus. void onWindowSystemUiVisibilityChanged(int visible) This method was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. void onWindowVisibilityChanged(int visibility) Called when the window containing has change its visibility (between GONE, INVISIBLE, and VISIBLE). boolean overScrollBy(int deltaX, int deltaY, int scrollX, int scrollY, int scrollRangeX, int scrollRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent) Scroll the view with standard behavior for scrolling beyond the normal content boundaries. boolean performAccessibilityAction(int action, Bundle arguments) Performs the specified accessibility action on the view. boolean performClick() Call this view's OnClickListener, if it is defined. boolean performContextClick(float x, float y) Call this view's OnContextClickListener, if it is defined. boolean performContextClick() Call this view's OnContextClickListener, if it is defined. boolean performHapticFeedback(int feedbackConstant) BZZZTT!!1! Provide haptic feedback to the user for this view. boolean performHapticFeedback(int feedbackConstant, int flags) BZZZTT!!1! Like performHapticFeedback(int), with additional options. boolean performLongClick(float x, float y) Calls this view's OnLongClickListener, if it is defined. boolean performLongClick() Calls this view's OnLongClickListener, if it is defined. ContentInfo performReceiveContent(ContentInfo payload) Receives the given content. void playSoundEffect(int soundConstant) Play a sound effect for this view. boolean post(Runnable action) Causes the Runnable to be added to the message queue. boolean postDelayed(Runnable action, long delayMillis) Causes the Runnable to be added to the message queue, to be run after the specified amount of time elapses. void postInvalidate() Cause an invalidate to happen on a subsequent cycle through the event loop. void postInvalidate(int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. void postInvalidateDelayed(long delayMilliseconds, int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. void postInvalidateDelayed(long delayMilliseconds) Cause an invalidate to happen on a subsequent cycle through the event loop. void postInvalidateOnAnimation(int left, int top, int right, int bottom) Cause an invalidate of the specified area to happen on the next animation time step, typically the next display frame. void postInvalidateOnAnimation() Cause an invalidate to happen on the next animation time step, typically the next display frame. void postOnAnimation(Runnable action) Causes the Runnable to execute on the next animation time step. void postOnAnimationDelayed(Runnable action, long delayMillis) Causes the Runnable to execute on the next animation time step, after the specified amount of time elapses. void refreshDrawableState() Call this to force a view to update its drawable state. void releasePointerCapture() Releases the pointer capture. boolean removeCallbacks(Runnable action) Removes the specified Runnable from the message queue. void removeOnAttachStateChangeListener(View.OnAttachStateChangeListener listener) Remove a listener for attach state changes. void removeOnLayoutChangeListener(View.OnLayoutChangeListener listener) Remove a listener for layout changes. void removeOnUnhandledKeyEventListener(View.OnUnhandledKeyEventListener listener) Removes a listener which will receive unhandled KeyEvents. void requestApplyInsets() Ask that a new dispatch of onApplyWindowInsets(android.view.WindowInsets) be performed. void requestFitSystemWindows() This method was deprecated in API level 20. Use requestApplyInsets() for newer platform versions. final boolean requestFocus(int direction) Call this to try to give focus to a specific view or to one of its descendants and give it a hint about what direction focus is heading. final boolean requestFocus() Call this to try to give focus to a specific view or to one of its descendants. boolean requestFocus(int direction, Rect previouslyFocusedRect) Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. final boolean requestFocusFromTouch() Call this to try to give focus to a specific view or to one of its descendants. void requestLayout() Call this when something has changed which has invalidated the layout of this view. void requestPointerCapture() Requests pointer capture mode. boolean requestRectangleOnScreen(Rect rectangle) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. boolean requestRectangleOnScreen(Rect rectangle, boolean immediate) Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. final void requestUnbufferedDispatch(int source) Request unbuffered dispatch of the given event source class to this view. final void requestUnbufferedDispatch(MotionEvent event) Request unbuffered dispatch of the given stream of MotionEvents to this View. final T requireViewById(int id) Finds the first descendant view with the given ID, the view itself if the ID matches getId(), or throws an IllegalArgumentException if the ID is invalid or there is no matching view in the hierarchy. void resetPivot() Clears any pivot previously set by a call to setPivotX(float) or setPivotY(float). static int resolveSize(int size, int measureSpec) Version of resolveSizeAndState(int, int, int) returning only the MEASURED_SIZE_MASK bits of the result. static int resolveSizeAndState(int size, int measureSpec, int childMeasuredState) Utility to reconcile a desired size and state, with constraints imposed by a MeasureSpec. boolean restoreDefaultFocus() Gives focus to the default-focus view in the view hierarchy that has this view as a root. void restoreHierarchyState(SparseArray container) Restore this view hierarchy's frozen state from the given container. final void saveAttributeDataForStyleable(Context context, int[] styleable, AttributeSet attrs, TypedArray t, int defStyleAttr, int defStyleRes) Stores debugging information about attributes. void saveHierarchyState(SparseArray container) Store this view hierarchy's frozen state into the given container. void scheduleDrawable(Drawable who, Runnable what, long when) Schedules an action on a drawable to occur at a specified time. void scrollBy(int x, int y) Move the scrolled position of your view. void scrollTo(int x, int y) Set the scrolled position of your view. void sendAccessibilityEvent(int eventType) Sends an accessibility event of the given type. void sendAccessibilityEventUnchecked(AccessibilityEvent event) This method behaves exactly as sendAccessibilityEvent(int) but takes as an argument an empty AccessibilityEvent and does not perform a check whether accessibility is enabled. void setAccessibilityDelegate(View.AccessibilityDelegate delegate) Sets a delegate for implementing accessibility support via composition (as opposed to inheritance). void setAccessibilityHeading(boolean isHeading) Set if view is a heading for a section of content for accessibility purposes. void setAccessibilityLiveRegion(int mode) Sets the live region mode for this view. void setAccessibilityPaneTitle(CharSequence accessibilityPaneTitle) Visually distinct portion of a window with window-like semantics are considered panes for accessibility purposes. void setAccessibilityTraversalAfter(int afterId) Sets the id of a view after which this one is visited in accessibility traversal. void setAccessibilityTraversalBefore(int beforeId) Sets the id of a view before which this one is visited in accessibility traversal. void setActivated(boolean activated) Changes the activated state of this view. void setAllowClickWhenDisabled(boolean clickableWhenDisabled) Enables or disables click events for this view when disabled. void setAlpha(float alpha) Sets the opacity of the view to a value from 0 to 1, where 0 means the view is completely transparent and 1 means the view is completely opaque. void setAnimation(Animation animation) Sets the next animation to play for this view. void setAnimationMatrix(Matrix matrix) Changes the transformation matrix on the view. void setAutoHandwritingEnabled(boolean enabled) Set whether this view enables automatic handwriting initiation. void setAutofillHints(String... autofillHints) Sets the hints that help an AutofillService determine how to autofill the view with the user's data. void setAutofillId(AutofillId id) Sets the unique, logical identifier of this view in the activity, for autofill purposes. void setBackground(Drawable background) Set the background to a given Drawable, or remove the background. void setBackgroundColor(int color) Sets the background color for this view. void setBackgroundDrawable(Drawable background) This method was deprecated in API level 16. use setBackground(android.graphics.drawable.Drawable) instead void setBackgroundResource(int resid) Set the background to a given resource. void setBackgroundTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the tint specified by setBackgroundTintList(android.content.res.ColorStateList)} to the background drawable. void setBackgroundTintList(ColorStateList tint) Applies a tint to the background drawable. void setBackgroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the tint specified by setBackgroundTintList(android.content.res.ColorStateList)} to the background drawable. final void setBottom(int bottom) Sets the bottom position of this view relative to its parent. void setCameraDistance(float distance) Sets the distance along the Z axis (orthogonal to the X/Y plane on which views are drawn) from the camera to this view. void setClickable(boolean clickable) Enables or disables click events for this view. void setClipBounds(Rect clipBounds) Sets a rectangular area on this view to which the view will be clipped when it is drawn. void setClipToOutline(boolean clipToOutline) Sets whether the View's Outline should be used to clip the contents of the View. void setContentCaptureSession(ContentCaptureSession contentCaptureSession) Sets the (optional) ContentCaptureSession associated with this view. void setContentDescription(CharSequence contentDescription) Sets the View's content description. void setContextClickable(boolean contextClickable) Enables or disables context clicking for this view. void setDefaultFocusHighlightEnabled(boolean defaultFocusHighlightEnabled) Sets whether this View should use a default focus highlight when it gets focused but doesn't have R.attr.state_focused defined in its background. void setDrawingCacheBackgroundColor(int color) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setDrawingCacheEnabled(boolean enabled) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setDrawingCacheQuality(int quality) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setDuplicateParentStateEnabled(boolean enabled) Enables or disables the duplication of the parent's state into this view. void setElevation(float elevation) Sets the base elevation of this view, in pixels. void setEnabled(boolean enabled) Set the enabled state of this view. void setFadingEdgeLength(int length) Set the size of the faded edge used to indicate that more content in this view is available. void setFilterTouchesWhenObscured(boolean enabled) Sets whether the framework should discard touches when the view's window is obscured by another visible window at the touched location. void setFitsSystemWindows(boolean fitSystemWindows) Sets whether or not this view should account for system screen decorations such as the status bar and inset its content; that is, controlling whether the default implementation of fitSystemWindows(android.graphics.Rect) will be executed. void setFocusable(boolean focusable) Set whether this view can receive the focus. void setFocusable(int focusable) Sets whether this view can receive focus. void setFocusableInTouchMode(boolean focusableInTouchMode) Set whether this view can receive focus while in touch mode. void setFocusedByDefault(boolean isFocusedByDefault) Sets whether this View should receive focus when the focus is restored for the view hierarchy containing this view. void setForceDarkAllowed(boolean allow) Sets whether or not to allow force dark to apply to this view. void setForeground(Drawable foreground) Supply a Drawable that is to be rendered on top of all of the content in the view. void setForegroundGravity(int gravity) Describes how the foreground is positioned. void setForegroundTintBlendMode(BlendMode blendMode) Specifies the blending mode used to apply the tint specified by setForegroundTintList(android.content.res.ColorStateList)} to the background drawable. void setForegroundTintList(ColorStateList tint) Applies a tint to the foreground drawable. void setForegroundTintMode(PorterDuff.Mode tintMode) Specifies the blending mode used to apply the tint specified by setForegroundTintList(android.content.res.ColorStateList)} to the background drawable. void setHapticFeedbackEnabled(boolean hapticFeedbackEnabled) Set whether this view should have haptic feedback for events such as long presses. void setHasTransientState(boolean hasTransientState) Set whether this view is currently tracking transient state that the framework should attempt to preserve when possible. void setHorizontalFadingEdgeEnabled(boolean horizontalFadingEdgeEnabled) Define whether the horizontal edges should be faded when this view is scrolled horizontally. void setHorizontalScrollBarEnabled(boolean horizontalScrollBarEnabled) Define whether the horizontal scrollbar should be drawn or not. void setHorizontalScrollbarThumbDrawable(Drawable drawable) Defines the horizontal thumb drawable void setHorizontalScrollbarTrackDrawable(Drawable drawable) Defines the horizontal track drawable void setHovered(boolean hovered) Sets whether the view is currently hovered. void setId(int id) Sets the identifier for this view. void setImportantForAccessibility(int mode) Sets how to determine whether this view is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen. void setImportantForAutofill(int mode) Sets the mode for determining whether this view is considered important for autofill. void setImportantForContentCapture(int mode) Sets the mode for determining whether this view is considered important for content capture. void setKeepScreenOn(boolean keepScreenOn) Controls whether the screen should remain on, modifying the value of KEEP_SCREEN_ON. void setKeyboardNavigationCluster(boolean isCluster) Set whether this view is a root of a keyboard navigation cluster. void setLabelFor(int id) Sets the id of a view for which this view serves as a label for accessibility purposes. void setLayerPaint(Paint paint) Updates the Paint object used with the current layer (used only if the current layer type is not set to LAYER_TYPE_NONE). void setLayerType(int layerType, Paint paint) Specifies the type of layer backing this view. void setLayoutDirection(int layoutDirection) Set the layout direction for this view. void setLayoutParams(ViewGroup.LayoutParams params) Set the layout parameters associated with this view. final void setLeft(int left) Sets the left position of this view relative to its parent. final void setLeftTopRightBottom(int left, int top, int right, int bottom) Assign a size and position to this view. void setLongClickable(boolean longClickable) Enables or disables long click events for this view. final void setMeasuredDimension(int measuredWidth, int measuredHeight) This method must be called by onMeasure(int, int) to store the measured width and measured height. void setMinimumHeight(int minHeight) Sets the minimum height of the view. void setMinimumWidth(int minWidth) Sets the minimum width of the view. void setNestedScrollingEnabled(boolean enabled) Enable or disable nested scrolling for this view. void setNextClusterForwardId(int nextClusterForwardId) Sets the id of the view to use as the root of the next keyboard navigation cluster. void setNextFocusDownId(int nextFocusDownId) Sets the id of the view to use when the next focus is FOCUS_DOWN. void setNextFocusForwardId(int nextFocusForwardId) Sets the id of the view to use when the next focus is FOCUS_FORWARD. void setNextFocusLeftId(int nextFocusLeftId) Sets the id of the view to use when the next focus is FOCUS_LEFT. void setNextFocusRightId(int nextFocusRightId) Sets the id of the view to use when the next focus is FOCUS_RIGHT. void setNextFocusUpId(int nextFocusUpId) Sets the id of the view to use when the next focus is FOCUS_UP. void setOnApplyWindowInsetsListener(View.OnApplyWindowInsetsListener listener) Set an OnApplyWindowInsetsListener to take over the policy for applying window insets to this view. void setOnCapturedPointerListener(View.OnCapturedPointerListener l) Set a listener to receive callbacks when the pointer capture state of a view changes. void setOnClickListener(View.OnClickListener l) Register a callback to be invoked when this view is clicked. void setOnContextClickListener(View.OnContextClickListener l) Register a callback to be invoked when this view is context clicked. void setOnCreateContextMenuListener(View.OnCreateContextMenuListener l) Register a callback to be invoked when the context menu for this view is being built. void setOnDragListener(View.OnDragListener l) Register a drag event listener callback object for this View. void setOnFocusChangeListener(View.OnFocusChangeListener l) Register a callback to be invoked when focus of this view changed. void setOnGenericMotionListener(View.OnGenericMotionListener l) Register a callback to be invoked when a generic motion event is sent to this view. void setOnHoverListener(View.OnHoverListener l) Register a callback to be invoked when a hover event is sent to this view. void setOnKeyListener(View.OnKeyListener l) Register a callback to be invoked when a hardware key is pressed in this view. void setOnLongClickListener(View.OnLongClickListener l) Register a callback to be invoked when this view is clicked and held. void setOnReceiveContentListener(String[] mimeTypes, OnReceiveContentListener listener) Sets the listener to be used to handle insertion of content into this view. void setOnScrollChangeListener(View.OnScrollChangeListener l) Register a callback to be invoked when the scroll X or Y positions of this view change. void setOnSystemUiVisibilityChangeListener(View.OnSystemUiVisibilityChangeListener l) This method was deprecated in API level 30. Use WindowInsets#isVisible(int) to find out about system bar visibilities by setting a OnApplyWindowInsetsListener on this view. void setOnTouchListener(View.OnTouchListener l) Register a callback to be invoked when a touch event is sent to this view. void setOutlineAmbientShadowColor(int color) Sets the color of the ambient shadow that is drawn when the view has a positive Z or elevation value. void setOutlineProvider(ViewOutlineProvider provider) Sets the ViewOutlineProvider of the view, which generates the Outline that defines the shape of the shadow it casts, and enables outline clipping. void setOutlineSpotShadowColor(int color) Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value. void setOverScrollMode(int overScrollMode) Set the over-scroll mode for this view. void setPadding(int left, int top, int right, int bottom) Sets the padding. void setPaddingRelative(int start, int top, int end, int bottom) Sets the relative padding. void setPivotX(float pivotX) Sets the x location of the point around which the view is rotated and scaled. void setPivotY(float pivotY) Sets the y location of the point around which the view is rotated and scaled. void setPointerIcon(PointerIcon pointerIcon) Set the pointer icon for the current view. final void setPreferKeepClear(boolean preferKeepClear) Set a preference to keep the bounds of this view clear from floating windows above this view's window. final void setPreferKeepClearRects(List rects) Set a preference to keep the provided rects clear from floating windows above this view's window. void setPressed(boolean pressed) Sets the pressed state for this view. void setRenderEffect(RenderEffect renderEffect) Configure the RenderEffect to apply to this View. final void setRevealOnFocusHint(boolean revealOnFocus) Sets this view's preference for reveal behavior when it gains focus. final void setRight(int right) Sets the right position of this view relative to its parent. void setRotation(float rotation) Sets the degrees that the view is rotated around the pivot point. void setRotationX(float rotationX) Sets the degrees that the view is rotated around the horizontal axis through the pivot point. void setRotationY(float rotationY) Sets the degrees that the view is rotated around the vertical axis through the pivot point. void setSaveEnabled(boolean enabled) Controls whether the saving of this view's state is enabled (that is, whether its onSaveInstanceState() method will be called). void setSaveFromParentEnabled(boolean enabled) Controls whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. void setScaleX(float scaleX) Sets the amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width. void setScaleY(float scaleY) Sets the amount that the view is scaled in Y around the pivot point, as a proportion of the view's unscaled width. void setScreenReaderFocusable(boolean screenReaderFocusable) Sets whether this View should be a focusable element for screen readers and include non-focusable Views from its subtree when providing feedback. void setScrollBarDefaultDelayBeforeFade(int scrollBarDefaultDelayBeforeFade) Define the delay before scrollbars fade. void setScrollBarFadeDuration(int scrollBarFadeDuration) Define the scrollbar fade duration. void setScrollBarSize(int scrollBarSize) Define the scrollbar size. void setScrollBarStyle(int style) Specify the style of the scrollbars. final void setScrollCaptureCallback(ScrollCaptureCallback callback) Sets the callback to receive scroll capture requests. void setScrollCaptureHint(int hint) Sets the scroll capture hint for this View. void setScrollContainer(boolean isScrollContainer) Change whether this view is one of the set of scrollable containers in its window. void setScrollIndicators(int indicators, int mask) Sets the state of the scroll indicators specified by the mask. void setScrollIndicators(int indicators) Sets the state of all scroll indicators. void setScrollX(int value) Set the horizontal scrolled position of your view. void setScrollY(int value) Set the vertical scrolled position of your view. void setScrollbarFadingEnabled(boolean fadeScrollbars) Define whether scrollbars will fade when the view is not scrolling. void setSelected(boolean selected) Changes the selection state of this view. void setSoundEffectsEnabled(boolean soundEffectsEnabled) Set whether this view should have sound effects enabled for events such as clicking and touching. void setStateDescription(CharSequence stateDescription) Sets the View's state description. void setStateListAnimator(StateListAnimator stateListAnimator) Attaches the provided StateListAnimator to this View. void setSystemGestureExclusionRects(List rects) Sets a list of areas within this view's post-layout coordinate space where the system should not intercept touch or other pointing device gestures. void setSystemUiVisibility(int visibility) This method was deprecated in API level 30. SystemUiVisibility flags are deprecated. Use WindowInsetsController instead. void setTag(int key, Object tag) Sets a tag associated with this view and a key. void setTag(Object tag) Sets the tag associated with this view. void setTextAlignment(int textAlignment) Set the text alignment. void setTextDirection(int textDirection) Set the text direction. void setTooltipText(CharSequence tooltipText) Sets the tooltip text which will be displayed in a small popup next to the view. final void setTop(int top) Sets the top position of this view relative to its parent. void setTouchDelegate(TouchDelegate delegate) Sets the TouchDelegate for this View. void setTransitionAlpha(float alpha) This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not side-effect (or get affected by) the real alpha property. final void setTransitionName(String transitionName) Sets the name of the View to be used to identify Views in Transitions. void setTransitionVisibility(int visibility) Changes the visibility of this View without triggering any other changes. void setTranslationX(float translationX) Sets the horizontal location of this view relative to its left position. void setTranslationY(float translationY) Sets the vertical location of this view relative to its top position. void setTranslationZ(float translationZ) Sets the depth location of this view relative to its elevation. void setVerticalFadingEdgeEnabled(boolean verticalFadingEdgeEnabled) Define whether the vertical edges should be faded when this view is scrolled vertically. void setVerticalScrollBarEnabled(boolean verticalScrollBarEnabled) Define whether the vertical scrollbar should be drawn or not. void setVerticalScrollbarPosition(int position) Set the position of the vertical scroll bar. void setVerticalScrollbarThumbDrawable(Drawable drawable) Defines the vertical scrollbar thumb drawable void setVerticalScrollbarTrackDrawable(Drawable drawable) Defines the vertical scrollbar track drawable void setViewTranslationCallback(ViewTranslationCallback callback) Sets a ViewTranslationCallback that is used to display/hide the translated information. void setVisibility(int visibility) Set the visibility state of this view. void setWillNotCacheDrawing(boolean willNotCacheDrawing) This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered

snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. void setWillNotDraw(boolean willNotDraw) If this view doesn't do any drawing on its own, set this flag to allow further optimizations. void setWindowInsetsAnimationCallback(WindowInsetsAnimation.Callback callback) Sets a WindowInsetsAnimation.Callback to be notified about animations of windows that cause insets. void setX(float x) Sets the visual x position of this view, in pixels. void setY(float y) Sets the visual y position of this view, in pixels. void setZ(float z) Sets the visual z position of this view, in pixels. boolean showContextMenu() Shows the context menu for this view. boolean showContextMenu(float x, float y) Shows the context menu for this view anchored to the specified view-relative coordinate. ActionMode startActionMode(ActionMode.Callback callback, int type) Start an action mode with the given type. ActionMode startActionMode(ActionMode.Callback callback) Start an action mode with the default type ActionMode#TYPE_PRIMARY. void startAnimation(Animation animation) Start the specified animation now. final boolean startDrag(ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) This method was deprecated in API level 24. Use startDragAndDrop() for newer platform versions. final boolean startDragAndDrop(ClipData data, View.DragShadowBuilder shadowBuilder, Object myLocalState, int flags) Starts a drag and drop operation. boolean startNestedScroll(int axes) Begin a nestable scroll operation along the given axes. void stopNestedScroll() Stop a nested scroll in progress. String toString() Returns a string representation of the object. void transformMatrixToGlobal(Matrix matrix) Modifies the input matrix such that it maps view-local coordinates to on-screen coordinates. void transformMatrixToLocal(Matrix matrix) Modifies the input matrix such that it maps on-screen coordinates to view-local coordinates. void unscheduleDrawable(Drawable who, Runnable what) Cancels a scheduled action on a drawable. void unscheduleDrawable(Drawable who) Unschedule any events associated with the given Drawable. final void updateDragShadow(View.DragShadowBuilder shadowBuilder) Updates the drag shadow for the ongoing drag and drop operation. boolean verifyDrawable(Drawable who) If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying. boolean willNotCacheDrawing() This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations, setLayerType(int, android.graphics.Paint) handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a Canvas from either a Bitmap or Picture and call draw(android.graphics.Canvas) on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as Config.HARDWARE bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the PixelCopy API is recommended. boolean willNotDraw() Returns whether or not this View draws on its own. From class java.lang.Object Object clone() Creates and returns a copy of this object. boolean equals(Object obj) Indicates whether some other object is "equal to" this one. void finalize() Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. final Class getClass() Returns the runtime class of this Object. int hashCode() Returns a hash code value for the object. final void notify() Wakes up a single thread that is waiting on this object's monitor. final void notifyAll() Wakes up all threads that are waiting on this object's monitor. String toString() Returns a string representation of the object. final void wait(long timeout, int nanos) Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. final void wait(long timeout) Causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed. final void wait() Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object. From interface android.view.ViewParent abstract void bringChildToFront(View child) Change the z order of the child so it's on top of all other children. abstract boolean canResolveLayoutDirection() Tells if this view parent can resolve the layout direction. abstract boolean canResolveTextAlignment() Tells if this view parent can resolve the text alignment. abstract boolean canResolveTextDirection() Tells if this view parent can resolve the text direction. abstract void childDrawableStateChanged(View child) This method is called on the parent when a child's drawable state has changed. abstract void childHasTransientStateChanged(View child, boolean hasTransientState) Called when a child view now has or no longer is tracking transient state. abstract void clearChildFocus(View child) Called when a child of this parent is giving up focus abstract void createContextMenu(ContextMenu menu) Have the parent populate the specified context menu if it has anything to add (and then recurse on its parent). default OnBackInvokedDispatcher findOnBackInvokedDispatcherForChild(View child, View requester) Walk up the View hierarchy to find the nearest OnBackInvokedDispatcher. abstract View focusSearch(View v, int direction) Find the nearest view in the specified direction that wants to take focus abstract void focusableViewAvailable(View v) Tells the parent that a new focusable view has become available. abstract boolean getChildVisibleRect(View child, Rect r, Point offset) Compute the visible part of a rectangular region defined in terms of a child view's coordinates. abstract int getLayoutDirection() Return this view parent layout direction. abstract ViewParent getParent() Returns the parent if it exists, or null. abstract ViewParent getParentForAccessibility() Gets the parent of a given View for accessibility. abstract int getTextAlignment() Return this view parent text alignment. abstract int getTextDirection() Return this view parent text direction. abstract void invalidateChild(View child, Rect r) This method was deprecated in API level 26. Use onDescendantInvalidated(android.view.View, android.view.View) instead. abstract ViewParent invalidateChildInParent(int[] location, Rect r) This method was deprecated in API level 26. Use onDescendantInvalidated(android.view.View, android.view.View) instead. abstract boolean isLayoutDirectionResolved() Tells if this view parent layout direction is resolved. abstract boolean isLayoutRequested() Indicates whether layout was requested on this view parent. abstract boolean isTextAlignmentResolved() Tells if this view parent text alignment is resolved. abstract boolean isTextDirectionResolved() Tells if this view parent text direction is resolved. abstract View keyboardNavigationClusterSearch(View currentCluster, int direction) Find the nearest keyboard navigation cluster in the specified direction. abstract void notifySubtreeAccessibilityStateChanged(View child, View source, int changeType) Notifies a view parent that the accessibility state of one of its descendants has changed and that the structure of the subtree is different. default void onDescendantInvalidated(View child, View target) The target View has been invalidated, or has had a drawing property changed that requires the hierarchy to re-render. abstract boolean onNestedFling(View target, float velocityX, float velocityY, boolean consumed) Request a fling from a nested scroll. abstract boolean onNestedPreFling(View target, float velocityX, float velocityY) React to a nested fling before the target view consumes it. abstract boolean onNestedPrePerformAccessibilityAction(View target, int action, Bundle arguments) React to an accessibility action delegated by a target descendant view before the target processes it. abstract void onNestedPreScroll(View target, int dx, int dy, int[] consumed) React to a nested scroll in progress before the target view consumes a portion of the scroll. abstract void onNestedScroll(View target, int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed) React to a nested scroll in progress. abstract void onNestedScrollAccepted(View child, View target, int nestedScrollAxes) React to the successful claiming of a nested scroll operation. abstract boolean onStartNestedScroll(View child, View target, int nestedScrollAxes) React to a descendant view initiating a nestable scroll operation, claiming the nested scroll operation if appropriate. abstract void onStopNestedScroll(View target) React to a nested scroll operation ending. abstract void recomputeViewAttributes(View child) Tell view hierarchy that the global view attributes need to be re-evaluated. abstract void requestChildFocus(View child, View focused) Called when a child of this parent wants focus abstract boolean requestChildRectangleOnScreen(View child, Rect rectangle, boolean immediate) Called when a child of this group wants a particular rectangle to be positioned onto the screen. abstract void requestDisallowInterceptTouchEvent(boolean disallowIntercept) Called when a child does not want this parent and its ancestors to intercept touch events with ViewGroup#onInterceptTouchEvent(MotionEvent). abstract void requestFitSystemWindows() Ask that a new dispatch of View.fitSystemWindows(Rect) be performed. abstract void requestLayout() Called when something has changed which has invalidated the layout of a child of this view parent. abstract boolean requestSendAccessibilityEvent(View child, AccessibilityEvent event) Called by a child to request from its parent to send an AccessibilityEvent. abstract void requestTransparentRegion(View child) Called when a child wants the view hierarchy to gather and report transparent regions to the window compositor. abstract boolean showContextMenuForChild(View originalView) Shows the context menu for the specified view or its ancestors. abstract boolean showContextMenuForChild(View originalView, float x, float y) Shows the context menu for the specified view or its ancestors anchored to the specified view-relative coordinate. abstract ActionMode startActionModeForChild(View originalView, ActionMode.Callback callback, int type) Start an action mode of a specific type for the specified view. abstract ActionMode startActionModeForChild(View originalView, ActionMode.Callback callback) Start an action mode for the specified view with the default type ActionMode#TYPE_PRIMARY. From interface android.view.KeyEvent.Callback abstract boolean onKeyDown(int keyCode, KeyEvent event) Called when a key down event has occurred. abstract boolean onKeyLongPress(int keyCode, KeyEvent event) Called when a long press has occurred. abstract boolean onKeyMultiple(int keyCode, int count, KeyEvent event) Called when a user's interaction with an analog control, such as flinging a trackball, generates simulated down/up events for the same key multiple times in quick succession. abstract boolean onKeyUp(int keyCode, KeyEvent event) Called when a key up event has occurred. public static final String SCHEME_GEO URI scheme for map address. Constant Value: "geo:0,0?q=" public static final String SCHEME_MAILTO URI scheme for email address. Constant Value: "mailto:" public static final String SCHEME_TEL URI scheme for telephone number. Constant Value: "tel:" public WebView (Context context) Constructs a new WebView with an Activity Context object. Note: WebView should always be instantiated with an Activity Context. If instantiated with an Application Context, WebView will be unable to provide several features, such as JavaScript dialogs and autofill. Parameters context Context: an Activity Context to access application assets This value cannot be null. public WebView (Context context, AttributeSet attrs) Constructs a new WebView with layout parameters. Parameters context Context: an Activity Context to access application assets This value cannot be null. attrs AttributeSet: an AttributeSet passed to our parent This value may be null. public WebView (Context context, AttributeSet attrs, int defStyleAttr) Constructs a new WebView with layout parameters and a default style. Parameters context Context: an Activity Context to access application assets This value cannot be null. attrs AttributeSet: an AttributeSet passed to our parent This value may be null. defStyleAttr int: an attribute in the current theme that contains a reference to a style resource that supplies default values for the view. Can be 0 to not look for defaults. public WebView (Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes) Constructs a new WebView with layout parameters and a default style. Parameters context Context: an Activity Context to access application assets This value cannot be null. attrs AttributeSet: an AttributeSet passed to our parent This value may be null. defStyleAttr int: an attribute in the current theme that contains a reference to a style resource that supplies default values for the view. Can be 0 to not look for defaults. defStyleRes int: a resource identifier of a style resource that supplies default values for the view, used only if defStyleAttr is 0 or can not be found in the theme. Can be 0 to not look for defaults. public WebView (Context context, AttributeSet attrs, int defStyleAttr, boolean privateBrowsing) This constructor is deprecated. Private browsing is no longer supported directly via WebView and will be removed in a future release. Prefer using WebSettings, WebViewDatabase, CookieManager and WebStorage for fine-grained control of privacy data. Constructs a new WebView with layout parameters and a default style. Parameters context Context: an Activity Context to access application assets This value cannot be null. attrs AttributeSet: an AttributeSet passed to our parent This value may be null. defStyleAttr int: an attribute in the current theme that contains a reference to a style resource that supplies default values for the view. Can be 0 to not look for defaults. privateBrowsing boolean: whether this WebView will be initialized in private mode public void addJavascriptInterface (Object object, String name) Injects the supplied Java object into this WebView. The object is injected into all frames of the web page, including all the iframes, using the supplied name. This allows the Java object's methods to be accessed from JavaScript. For applications targeted to API level Build.VERSION_CODES.JELLY_BEAN_MR1 and above, only public methods that are annotated with JavascriptInterface can be accessed from JavaScript. For applications targeted to API level Build.VERSION_CODES.JELLY_BEAN or below, all public methods (including the inherited ones) can be accessed, see the important security note below for implications. Note that injected objects will not appear in JavaScript until the page is next (re)loaded. JavaScript should be enabled before injecting the object. For example: class JsObject { @JavascriptInterface public String toString() { return "injectedObject"; } } webview.getSettings().setJavaScriptEnabled(true); webView.addJavascriptInterface(new JsObject(), "injectedObject"); webView.loadData("", "text/html", null); webView.loadUrl("javascript:alert(injectedObject.toString())"); IMPORTANT: This method can be used to allow JavaScript to control the host application. This is a powerful feature, but also presents a security risk for apps targeting Build.VERSION_CODES.JELLY_BEAN or earlier. Apps that target a version later than Build.VERSION_CODES.JELLY_BEAN are still vulnerable if the app runs on a device running Android earlier than 4.2. The most secure way to use this method is to target Build.VERSION_CODES.JELLY_BEAN_MR1 and to ensure the method is called only when running on Android 4.2 or later. With these older versions, JavaScript could use reflection to access an injected object's public fields. Use of this method in a WebView containing untrusted content could allow an attacker to manipulate the host application in unintended ways, executing Java code with the permissions of the host application. Use extreme care when using this method in a WebView which could contain untrusted content. JavaScript interacts with Java object on a private, background thread of this WebView. Care is therefore required to maintain thread safety. Because the object is exposed to all the frames, any frame could obtain the object name and call methods on it. There is no way to tell the calling frame's origin from the app side, so the app must not assume that the caller is trustworthy unless the app can guarantee that no third party content is ever loaded into the WebView even inside an iframe. The Java object's fields are not accessible. For applications targeted to API level Build.VERSION_CODES.LOLLIPOP and above, methods of injected Java objects are enumerable from JavaScript. Parameters object Object: the Java object to inject into this WebView's JavaScript context. null values are ignored. name String: the name used to expose the object in JavaScript This value cannot be null. public void autofill (SparseArray values) Automatically fills the content of the virtual children within this view. Views with virtual children support the Autofill Framework mainly by: Providing the metadata defining what the virtual children mean and how they can be autofilled. Implementing the methods that autofill the virtual children. onProvideAutofillVirtualStructure(android.view.ViewStructure, int) is responsible for the former, this method is responsible for the latter - see autofill(android.view.autofill.AutofillValue) and onProvideAutofillVirtualStructure(android.view.ViewStructure, int) for more info about autofill. If a child value is updated asynchronously, the next call to AutofillManager#notifyValueChanged(View, int, AutofillValue) must happen after the value was changed to the autofilled value. If not, the child will not be considered autofilled. Note: To indicate that a virtual view was autofilled, ?android:attr/autofilledHighlight should be drawn over it until the data changes. Parameters values SparseArray: map of values to be autofilled, keyed by virtual child id. This value cannot be null. public boolean canGoBack () Gets whether this WebView has a back history item. Returns boolean true if this WebView has a back history item public boolean canGoBackOrForward (int steps) Gets whether the page can go back or forward the given number of steps. Parameters steps int: the negative or positive number of steps to move the history public boolean canGoForward () Gets whether this WebView has a forward history item. Returns boolean true if this WebView has a forward history item Added in API level 11 Deprecated in API level 17 public boolean canZoomIn () This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. Gets whether this WebView can be zoomed in. Returns boolean true if this WebView can be zoomed in Added in API level 11 Deprecated in API level 17 public boolean canZoomOut () This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. Gets whether this WebView can be zoomed out. Returns boolean true if this WebView can be zoomed out Added in API level 1 Deprecated in API level 19 public Picture capturePicture () This method was deprecated in API level 19. Use onDraw(Canvas) to obtain a bitmap snapshot of the WebView, or saveWebArchive(String) to save the content to a file. Gets a new picture that captures the current contents of this WebView. The picture is of the entire document being displayed, and is not limited to the area currently displayed by this WebView. Also, the picture is a static copy and is unaffected by later changes to the content being displayed. Note that due to internal changes, for API levels between Build.VERSION_CODES.HONEYCOMB and Build.VERSION_CODES.ICE_CREAM_SANDWICH inclusive, the picture does not include fixed position elements or scrollable divs. Note that from Build.VERSION_CODES.JELLY_BEAN_MR1 the returned picture should only be drawn into bitmap-backed Canvas - using any other type of Canvas will involve additional conversion at a cost in memory and performance. Returns Picture a picture that captures the current contents of this WebView public void clearCache (boolean includeDiskFiles) Clears the resource cache. Note that the cache is per-application, so this will clear the cache for all WebViews used. Parameters includeDiskFiles boolean: if false, only the RAM cache is cleared public static void clearClientCertPreferences (Runnable onCleared) Clears the client certificate preferences stored in response to proceeding/cancelling client cert requests. Note that WebView automatically clears these preferences when the system keychain is updated. The preferences are shared by all the WebViews that are created by the embedder application. Parameters onCleared Runnable: A runnable to be invoked when client certs are cleared. The runnable will be called in UI thread. This value may be null. public void clearFormData () Removes the autocomplete popup from the currently focused form field, if present. Note this only affects the display of the autocomplete popup, it does not remove any saved form data from this WebView's store. To do that, use WebViewDatabase#clearFormData. public void clearHistory () Tells this WebView to clear its internal back/forward list. public void clearSslPreferences () Clears the SSL preferences table stored in response to proceeding with SSL certificate errors. Added in API level 1 Deprecated in API level 18 public void clearView () This method was deprecated in API level 18. Use WebView.loadUrl("about:blank") to reliably reset the view state and release page resources (including any running JavaScript). Clears this WebView so that onDraw() will draw nothing but white background, and onMeasure() will return 0 if MeasureSpec is not MeasureSpec.EXACTLY. public void computeScroll () Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. This will typically be done if the child is animating a scroll using a Scroller object. public WebBackForwardList copyBackForwardList () Gets the WebBackForwardList for this WebView. This contains the back/forward list for use in querying each item in the history stack. This is a copy of the private WebBackForwardList so it contains only a snapshot of the current state. Multiple calls to this method may return different objects. The object returned from this method will not be updated to reflect any new state. Returns WebBackForwardList This value cannot be null. public PrintDocumentAdapter createPrintDocumentAdapter (String documentName) Creates a PrintDocumentAdapter that provides the content of this WebView for printing. The adapter works by converting the WebView contents to a PDF stream. The WebView cannot be drawn during the conversion process - any such draws are undefined. It is recommended to use a dedicated off screen WebView for the printing. If necessary, an application may temporarily hide a visible WebView by using a custom PrintDocumentAdapter instance wrapped around the object returned and observing the onStart and onFinish methods. See PrintDocumentAdapter for more information. Parameters documentName String: The user-facing name of the printed document. See PrintDocumentInfo This value cannot be null. Returns PrintDocumentAdapter This value cannot be null. public WebMessagePort[] createWebMessageChannel () Creates a message channel to communicate with JS and returns the message ports that represent the endpoints of this message channel. The HTML5 message channel functionality is described here The returned message channels are entangled and already in started state. Returns WebMessagePort[] the two message ports that form the message channel. This value cannot be null. public void destroy () Destroys the internal state of this WebView. This method should be called after this WebView has been removed from the view system. No other methods may be called on this WebView after destroy. public static void disableWebView () Indicate that the current process does not intend to use WebView, and that an exception should be thrown if a WebView is created or any other methods in the android.webkit package are used. Applications with multiple processes may wish to call this in processes that are not intended to use WebView to avoid accidentally incurring the memory usage of initializing WebView in long-lived processes that have no need for it, and to prevent potential data directory conflicts (see setDataDirectorySuffix(String)). For example, an audio player application with one process for its activities and another process for its playback service may wish to call this method in the playback service's Service.onCreate(). public void dispatchCreateViewTranslationRequest (Map viewIds, int[] supportedFormats, TranslationCapability capability, List requests) Dispatch to collect the ViewTranslationRequests for translation purpose by traversing the hierarchy when the app requests ui translation. Typically, this method should only be overridden by subclasses that provide a view hierarchy (such as ViewGroup). Other classes should override View#onCreateViewTranslationRequest for normal view or override View#onVirtualViewTranslationResponses for view contains virtual children. When requested to start the ui translation, the system will call this method to traverse the view hierarchy to collect ViewTranslationRequests and create a Translator to translate the requests. All the ViewTranslationRequests must be added when the traversal is done. The default implementation calls View#onCreateViewTranslationRequest for normal view or calls View#onVirtualViewTranslationResponses for view contains virtual children to build ViewTranslationRequest if the view should be translated. The view is marked as having transient state so that recycling of views doesn't prevent the system from attaching the response to it. Therefore, if overriding this method, you should set or reset the transient state. The implementation calls dispatchCreateViewTranslationRequest(Map, int[], TranslationCapability, List) for all the child views. Parameters viewIds Map: This value cannot be null. supportedFormats int: This value cannot be null. Value is TranslationSpec.DATA_FORMAT_TEXT capability TranslationCapability: This value may be null. requests List: This value cannot be null. public boolean dispatchKeyEvent (KeyEvent event) Dispatch a key event to the next view on the focus path. This path runs from the top of the view tree down to the currently focused view. If this view has focus, it will dispatch to itself. Otherwise it will dispatch the next node down the focus path. This method also fires any key listeners. Parameters event KeyEvent: The key event to be dispatched. Returns boolean True if the event was handled, false otherwise. public void documentHasImages (Message response) Queries the document to see if it contains any image references. The message object will be dispatched with arg1 being set to 1 if images were found and 0 if the document does not reference any images. Parameters response Message: the message that will be dispatched with the result This value cannot be null. public static void enableSlowWholeDocumentDraw () For apps targeting the L release, WebView has a new default behavior that reduces memory footprint and increases performance by intelligently choosing the portion of the HTML document that needs to be drawn. These optimizations are transparent to the developers. However, under certain circumstances, an App developer may want to disable them: When an app uses onDraw(Canvas) to do own drawing and accesses portions of the page that is way outside the visible portion of the page. When an app uses capturePicture() to capture a very large HTML document. Note that capturePicture is a deprecated API. Enabling drawing the entire HTML document has a significant performance cost. This method should be called before any WebViews are created. public void evaluateJavascript (String script, ValueCallback resultCallback) Asynchronously evaluates JavaScript in the context of the currently displayed page. If non-null, resultCallback will be invoked with any result returned from that execution. This method must be called on the UI thread and the callback will be made on the UI thread. Compatibility note. Applications targeting Build.VERSION_CODES.N or later, JavaScript state from an empty WebView is no longer persisted across navigations like loadUrl(java.lang.String). For example, global variables and functions defined before calling loadUrl(java.lang.String) will not exist in the loaded page. Applications should use addJavascriptInterface(Object, String) instead to persist JavaScript objects across navigations. Parameters script String: the JavaScript to execute. This value cannot be null. resultCallback ValueCallback: A callback to be invoked when the script execution completes with the result of the execution (if any). May be null if no notification of the result is required. Added in API level 1 Deprecated in API level 28 public static String findAddress (String addr) This method was deprecated in API level 28. This method is superseded by TextClassifier#generateLinks(android.view.textclassifier.TextLinks.Request). Avoid using this method even when targeting API levels where no alternative is available. Gets the first substring which appears to be the address of a physical location. Only addresses in the United States can be detected, which must consist of: a house number a street name a street type (Road, Circle, etc), either spelled out or abbreviated a city name a state or territory, either spelled out or two-letter abbr an optional 5 digit or 9 digit zip code All names must be correctly capitalized, and the zip code, if present, must be valid for the state. The street type must be a standard USPS spelling or abbreviation. The state or territory must also be spelled or abbreviated using USPS standards. The house number may not exceed five digits. Note: This function is deprecated and should be avoided on all API levels, as it cannot detect addresses outside of the United States and has a high rate of false positives. On API level Build.VERSION_CODES.O_MR1 and earlier, it also causes the entire WebView implementation to be loaded and initialized, which can throw AndroidRuntimeException or other exceptions if the WebView implementation is currently being updated. Parameters addr String: the string to search for addresses Returns String the address, or if no address is found, null public void findAllAsync (String find) Finds all instances of find on the page and highlights them, asynchronously. Notifies any registered FindListener. Successive calls to this will cancel any pending searches. Parameters find String: the string to find. This value cannot be null. See also: setFindListener(WebView.FindListener) public View findFocus () Find the view in the hierarchy rooted at this view that currently has focus. Returns View The view that currently has focus, or null if no focused view can be found. public void flingScroll (int vx, int vy) Added in API level 7 Deprecated in API level 19 public void freeMemory () This method was deprecated in API level 19. Memory caches are automatically dropped when no longer needed, and in response to system memory pressure. Informs this WebView that memory is low so that it can free any available memory. public CharSequence getAccessibilityClassName () Return the class name of this object to be used for accessibility purposes. Subclasses should only override this if they are implementing something that should be seen as a completely new class of view when used by accessibility, unrelated to the class it is deriving from. This is used to fill in AccessibilityNodeInfo.setClassName. public SslCertificate getCertificate () Gets the SSL certificate for the main top-level page or null if there is no certificate (the site is not secure). Returns SslCertificate the SSL certificate for the main top-level page public int getContentHeight () Gets the height of the HTML content. Returns int the height of the HTML content public static PackageInfo getCurrentWebViewPackage () If WebView has already been loaded into the current process this method will return the package that was used to load it. Otherwise, the package that would be used if the WebView was loaded right now will be returned; this does not cause WebView to be loaded, so this information may become outdated at any time. The WebView package changes either when the current WebView package is updated, disabled, or uninstalled. It can also be changed through a Developer Setting. If the WebView package changes, any app process that has loaded WebView will be killed. The next time the app starts and loads WebView it will use the new WebView package instead. Returns PackageInfo the current WebView package, or null if there is none. public Bitmap getFavicon () Gets the favicon for the current page. This is the favicon of the current page until WebViewClient.onReceivedIcon is called. Returns Bitmap the favicon for the current page or null if the page doesn't have one or if no page has been loaded public Handler getHandler () Returns Handler A handler associated with the thread running the View. This handler can be used to pump events in the UI events queue. public WebView.HitTestResult getHitTestResult () Gets a HitTestResult based on the current cursor node. If a HTML::a tag is found and the anchor has a non-JavaScript URL, the HitTestResult type is set to SRC_ANCHOR_TYPE and the URL is set in the "extra" field. If the anchor does not have a URL or if it is a JavaScript URL, the type will be UNKNOWN_TYPE and the URL has to be retrieved through requestFocusNodeHref(Message) asynchronously. If a HTML::img tag is found, the HitTestResult type is set to IMAGE_TYPE and the URL is set in the "extra" field. A type of SRC_IMAGE_ANCHOR_TYPE indicates an anchor with a URL that has an image as a child node. If a phone number is found, the HitTestResult type is set to PHONE_TYPE and the phone number is set in the "extra" field of HitTestResult. If a map address is found, the HitTestResult type is set to GEO_TYPE and the address is set in the "extra" field of HitTestResult. If an email address is found, the HitTestResult type is set to EMAIL_TYPE and the email is set in the "extra" field of HitTestResult. Otherwise, HitTestResult type is set to UNKNOWN_TYPE. Returns WebView.HitTestResult This value cannot be null. Added in API level 1 Deprecated in API level 26 public String[] getHttpAuthUsernamePassword (String host, String realm) This method was deprecated in API level 26. Use WebViewDatabase#getHttpAuthUsernamePassword instead Retrieves HTTP authentication credentials for a given host and realm from the WebViewDatabase instance. Parameters host String: the host to which the credentials apply realm String: the realm to which the credentials apply Returns String[] the credentials as a String array, if found. The first element is the username and the second element is the password. null if no credentials are found. public String getOriginalUrl () Gets the original URL for the current page. This is not always the same as the URL passed to WebViewClient.onPageStarted because although the load for that URL has begun, the current page may not have changed. Also, there may have been redirects resulting in a different URL to that originally requested. Returns String the URL that was originally requested for the current page or null if no page has been loaded public int getProgress () Gets the progress for the current page. Returns int the progress for the current page between 0 and 100 public static Uri getSafeBrowsingPrivacyPolicyUrl () Returns a URL pointing to the privacy policy for Safe Browsing reporting. Returns Uri the url pointing to a privacy policy document which can be displayed to users. This value cannot be null. Added in API level 1 Deprecated in API level 17 public float getScale () This method was deprecated in API level 17. This method is prone to inaccuracy due to race conditions between the web rendering and UI threads; prefer WebViewClient#onScaleChanged. Gets the current scale of this WebView. Returns float the current scale public WebSettings getSettings () Gets the WebSettings object used to control the settings for this WebView. Returns WebSettings a WebSettings object that can be used to control this WebView's settings This value cannot be null. public String getTitle () Gets the title for the current page. This is the title of the current page until WebViewClient.onReceivedTitle is called. Returns String the title for the current page or null if no page has been loaded public String getUrl () Gets the URL for the current page. This is not always the same as the URL passed to WebViewClient.onPageStarted because although the load for that URL has begun, the current page may not have changed. Returns String the URL for the current page or null if no page has been loaded public static ClassLoader getWebViewClassLoader () Returns the ClassLoader used to load internal WebView classes. This method is meant for use by the WebView Support Library, there is no reason to use this method otherwise. Returns ClassLoader This value cannot be null. public Looper getWebViewLooper () Returns the Looper corresponding to the thread on which WebView calls must be made. Returns Looper This value cannot be null. public WebViewRenderProcess getWebViewRenderProcess () Gets a handle to the WebView renderer process associated with this WebView. In Build.VERSION_CODES.O and above, WebView may run in "multiprocess" mode. In multiprocess mode, rendering of web content is performed by a sandboxed renderer process separate to the application process. This renderer process may be shared with other WebViews in the application, but is not shared with other application processes. If WebView is running in multiprocess mode, this method returns a handle to the renderer process associated with the WebView, which can be used to control the renderer process. public void goBack () Goes back in the history of this WebView. public void goBackOrForward (int steps) Goes to the history item that is the number of steps away from the current item. Steps is negative if backward and positive if forward. Parameters steps int: the number of steps to take back or forward in the back forward list public void goForward () Goes forward in the history of this WebView. public void invokeZoomPicker () Invokes the graphical zoom picker widget for this WebView. This will result in the zoom widget appearing on the screen to control the zoom level of this WebView. public boolean isPrivateBrowsingEnabled () Gets whether private browsing is enabled in this WebView. public boolean isVisibleToUserForAutofill (int virtualId) Computes whether this virtual autofill view is visible to the user. Note: By default it returns true, but views providing a virtual hierarchy view must override it. Returns boolean Whether the view is visible on the screen. public void loadData (String data, String mimeType, String encoding) Loads the given data into this WebView using a 'data' scheme URL. Note that JavaScript's same origin policy means that script running in a page loaded using this method will be unable to access content loaded using any scheme other than 'data', including 'http(s)'. To avoid this restriction, use loadDataWithBaseURL() with an appropriate base URL. The encoding parameter specifies whether the data is base64 or URL encoded. If the data is base64 encoded, the value of the encoding parameter must be "base64". HTML can be encoded with Base64.encodeToString(byte[], int) like so: String unencodedHtml = "'%28' is the code for '('"; String encodedHtml = Base64.encodeToString(unencodedHtml.getBytes(), Base64.NO_PADDING); webView.loadData(encodedHtml, "text/html", "base64"); For all other values of encoding (including null) it is assumed that the data uses ASCII encoding for octets inside the range of safe URL characters and use the standard %xx hex encoding of URLs for octets outside that range. See RFC 3986 for more information. Applications targeting Build.VERSION_CODES.Q or later must either use base64 or encode any # characters in the content as %23, otherwise they will be treated as the end of the content and the remaining text used as a document fragment identifier. The mimeType parameter specifies the format of the data. If WebView can't handle the specified MIME type, it will download the data. If null, defaults to 'text/html'. The 'data' scheme URL formed by this method uses the default US-ASCII charset. If you need to set a different charset, you should form a 'data' scheme URL which explicitly specifies a charset parameter in the mediatype portion of the URL and call loadUrl(java.lang.String) instead. Note that the charset obtained from the mediatype portion of a data URL always overrides that specified in the HTML or XML document itself. Content loaded using this method will have a window.origin value of "null". This must not be considered to be a trusted origin by the application or by any JavaScript code running inside the WebView (for example, event sources in DOM event handlers or web messages), because malicious content can also create frames with a null origin. If you need to identify the main frame's origin in a trustworthy way, you should use loadDataWithBaseURL() with a valid HTTP or HTTPS base URL to set the origin. Parameters data String: a String of data in the given encoding This value cannot be null. mimeType String: the MIME type of the data, e.g. 'text/html'. This value may be null. encoding String: the encoding of the data This value may be null. public void loadDataWithBaseURL (String baseUrl, String data, String mimeType, String encoding, String historyUrl) Loads the given data into this WebView, using baseUrl as the base URL for the content. The base URL is used both to resolve relative URLs and when applying JavaScript's same origin policy. The historyUrl is used for the history entry. The mimeType parameter specifies the format of the data. If WebView can't handle the specified MIME type, it will download the data. If null, defaults to 'text/html'. Note that content specified in this way can access local device files (via 'file' scheme URLs) only if baseUrl specifies a scheme other than 'http', 'https', 'ftp', 'ftps', 'about' or 'javascript'. If the base URL uses the data scheme, this method is equivalent to calling loadData() and the historyUrl is ignored, and the data will be treated as part of a data: URL, including the requirement that the content be URL-encoded or base64 encoded. If the base URL uses any other scheme, then the data will be loaded into the WebView as a plain string (i.e. not part of a data URL) and any URL-encoded entities in the string will not be decoded. Note that the baseUrl is sent in the 'Referer' HTTP header when requesting subresources (images, etc.) of the page loaded using this method. If a valid HTTP or HTTPS base URL is not specified in baseUrl, then content loaded using this method will have a window.origin value of "null". This must not be considered to be a trusted origin by the application or by any JavaScript code running inside the WebView (for example, event sources in DOM event handlers or web messages), because malicious content can also create frames with a null origin. If you need to identify the main frame's origin in a trustworthy way, you should use a valid HTTP or HTTPS base URL to set the origin. Parameters baseUrl String: the URL to use as the page's base URL. If null defaults to 'about:blank'. data String: a String of data in the given encoding This value cannot be null. mimeType String: the MIME type of the data, e.g. 'text/html'. This value may be null. encoding String: the encoding of the data This value may be null. historyUrl String: the URL to use as the history entry. If null defaults to 'about:blank'. If non-null, this must be a valid URL. public void loadUrl (String url, Map additionalHttpHeaders) Loads the given URL with additional HTTP headers, specified as a map from name to value. Note that if this map contains any of the headers that are set by default by this WebView, such as those controlling caching, accept types or the User-Agent, their values may be overridden by this WebView's defaults. Some older WebView implementations require additionalHttpHeaders to be mutable. Also see compatibility note on evaluateJavascript(String, ValueCallback). Parameters url String: the URL of the resource to load This value cannot be null. additionalHttpHeaders Map: map with additional headers This value cannot be null. public WindowInsets onApplyWindowInsets (WindowInsets insets) Called when the view should apply WindowInsets according to its internal policy. This method should be overridden by views that wish to apply a policy different from or in addition to the default behavior. Clients that wish to force a view subtree to apply insets should call dispatchApplyWindowInsets(android.view.WindowInsets). Clients may supply an OnApplyWindowInsetsListener to a view. If one is set it will be called during dispatch instead of this method. The listener may optionally call this method from its own implementation if it wishes to apply the view's default insets policy in addition to its own. Implementations of this method should either return the insets parameter unchanged or a new WindowInsets cloned from the supplied insets with any insets consumed that this view applied itself. This allows new inset types added in future platform versions to pass through existing implementations unchanged without being erroneously consumed. By default if a view's fitsSystemWindows property is set then the view will consume the system window insets and apply them as padding for the view. Parameters insets WindowInsets: Insets to apply Returns WindowInsets The supplied insets with any applied insets consumed public boolean onCheckIsTextEditor () Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. Subclasses should override this if they implement onCreateInputConnection(android.view.inputmethod.EditorInfo) to return true if a call on that method would return a non-null InputConnection, and they are really a first-class editor that the user would normally start typing on when the go into a window containing your view. The default implementation always returns false. This does not mean that its onCreateInputConnection(android.view.inputmethod.EditorInfo) will not be called or the user can not otherwise perform edits on your view; it is just a hint to the system that this is not the primary purpose of this view. Returns boolean Returns true if this view is a text editor, else false. public void onChildViewAdded (View parent, View child) This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. Called when a new child is added to a parent view. Parameters parent View: the view in which a child was added child View: the new child view added in the hierarchy public void onChildViewRemoved (View p, View child) This method is deprecated. WebView no longer needs to implement ViewGroup.OnHierarchyChangeListener. This method does nothing now. Called when a child is removed from a parent view. Parameters p View: the view from which the child was removed child View: the child removed from the hierarchy public InputConnection onCreateInputConnection (EditorInfo outAttrs) Creates a new InputConnection for an InputMethod to interact with the WebView. This is similar to View#onCreateInputConnection but note that WebView calls InputConnection methods on a thread other than the UI thread. If these methods are overridden, then the overriding methods should respect thread restrictions when calling View methods or accessing data. Parameters outAttrs EditorInfo: Fill in with attribute information about the connection. public void onCreateVirtualViewTranslationRequests (long[] virtualIds, int[] supportedFormats, Consumer requestsCollector) Collects ViewTranslationRequests which represents the content to be translated for the virtual views in the host view. This is called if this view returned a virtual view structure from onProvideContentCaptureStructure(ViewStructure, int) and the system determined that those virtual views were relevant for translation. The default implementation does nothing. Parameters virtualIds long: This value cannot be null. supportedFormats int: This value cannot be null. Value is TranslationSpec.DATA_FORMAT_TEXT requestsCollector Consumer: This value cannot be null. Returns void This value may be null. public boolean onDragEvent (DragEvent event) Handles drag events sent by the system following a call to startDragAndDrop(). The system calls this method and passes a DragEvent object in response to drag and drop events. This method can then call DragEvent#getAction() to determine the state of the drag and drop operation. The default implementation returns false unless an OnReceiveContentListener has been set for this view (see setOnReceiveContentListener(String[], OnReceiveContentListener)), in which case the default implementation does the following: Parameters event DragEvent: The DragEvent object sent by the system. The DragEvent#getAction() method returns an action type constant that indicates the type of drag event represented by this object. Returns boolean true if the method successfully handled the drag event, otherwise false. The method must return true in response to an ACTION_DRAG_STARTED action type to continue to receive drag events for the current drag and drop operation. The method should return true in response to an ACTION_DROP action type if the dropped data was consumed (at least partially); false, if none of the data was consumed. For all other events, the return value is false. public boolean onGenericMotionEvent (MotionEvent event) Implement this method to handle generic motion events. Generic motion events describe joystick movements, mouse hovers, track pad touches, scroll wheel movements and other input events. The source of the motion event specifies the class of input that was received. Implementations of this method must examine the bits in the source before processing the event. The following code example shows how this is done. Generic motion events with source class InputDevice#SOURCE_CLASS_POINTER are delivered to the view under the pointer. All other generic motion events are delivered to the focused view. public boolean onGenericMotionEvent(MotionEvent event) { if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) { if (event.getAction() == MotionEvent.ACTION_MOVE) { // process the joystick movement... return true; } } if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) { switch (event.getAction()) { case MotionEvent.ACTION_HOVER_MOVE: // process the mouse hover movement... return true; case MotionEvent.ACTION_SCROLL: // process the scroll wheel movement... return true; } } return super.onGenericMotionEvent(event); } Parameters event MotionEvent: The generic motion event being processed. Returns boolean True if the event was handled, false otherwise. public void onGlobalFocusChanged (View oldFocus, View newFocus) This method is deprecated. WebView should not have implemented ViewTreeObserver.OnGlobalFocusChangeListener. This method does nothing now. Callback method to be invoked when the focus changes in the view tree. When the view tree transitions from touch mode to non-touch mode, oldFocus is null. When the view tree transitions from non-touch mode to touch mode, newFocus is null. When focus changes in non-touch mode (without transition from or to touch mode) either oldFocus or newFocus can be null. Parameters oldFocus View: The previously focused view, if any. newFocus View: The newly focused View, if any. public boolean onHoverEvent (MotionEvent event) Implement this method to handle hover events. This method is called whenever a pointer is hovering into, over, or out of the bounds of a view and the view is not currently being touched. Hover events are represented as pointer events with action MotionEvent#ACTION_HOVER_ENTER, MotionEvent#ACTION_HOVER_MOVE, or MotionEvent#ACTION_HOVER_EXIT. The view receives a hover event with action MotionEvent#ACTION_HOVER_ENTER when the pointer enters the bounds of the view. The view receives a hover event with action MotionEvent#ACTION_HOVER_MOVE when the pointer has already entered the bounds of the view and has moved. The view receives a hover event with action MotionEvent#ACTION_HOVER_EXIT when the pointer has exited the bounds of the view or when the pointer is about to go down due to a button click, tap, or similar user action that causes the view to be touched. The view should implement this method to return true to indicate that it is handling the hover event, such as by changing its drawable state. The default implementation calls setHovered(boolean) to update the hovered state of the view when a hover enter or hover exit event is received, if the view is enabled and is clickable. The default implementation also sends hover accessibility events. Parameters event MotionEvent: The motion event that describes the hover. Returns boolean True if the view handled the hover event. public boolean onKeyDown (int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyDown(): perform press of the view when KeyEvent#KEYCODE_DPAD_CENTER or KeyEvent#KEYCODE_ENTER is released, if the view is enabled and clickable. Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses. Parameters keyCode int: a key code that represents the button pressed, from KeyEvent event KeyEvent: the KeyEvent object that defines the button action Returns boolean If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. public boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyMultiple(): always returns false (doesn't handle the event). Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses. Parameters keyCode int: A key code that represents the button pressed, from KeyEvent. repeatCount int: The number of times the action was made. event KeyEvent: The KeyEvent object that defines the button action. Returns boolean If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. public boolean onKeyUp (int keyCode, KeyEvent event) Default implementation of KeyEvent.Callback.onKeyUp(): perform clicking of the view when KeyEvent#KEYCODE_DPAD_CENTER, KeyEvent#KEYCODE_ENTER or KeyEvent#KEYCODE_SPACE is released. Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses. Parameters keyCode int: A key code that represents the button pressed, from KeyEvent. event KeyEvent: The KeyEvent object that defines the button action. Returns boolean If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. public void onPause () Does a best-effort attempt to pause any processing that can be paused safely, such as animations and geolocation. Note that this call does not pause JavaScript. To pause JavaScript globally, use pauseTimers(). To resume WebView, call onResume(). public void onProvideAutofillVirtualStructure (ViewStructure structure, int flags) Populates a ViewStructure containing virtual children to fullfil an autofill request. This method should be used when the view manages a virtual structure under this view. For example, a view that draws input fields using draw(android.graphics.Canvas). When implementing this method, subclasses must follow the rules below: Add virtual children by calling the ViewStructure#newChild(int) or ViewStructure#asyncNewChild(int) methods, where the id is an unique id identifying the children in the virtual structure. The children hierarchy can have multiple levels if necessary, but ideally it should exclude intermediate levels that are irrelevant for autofill; that would improve the autofill performance. Also implement autofill(android.util.SparseArray) to autofill the virtual children. Set the autofill properties of the child structure as defined by onProvideAutofillStructure(android.view.ViewStructure, int), using ViewStructure#setAutofillId(AutofillId, int) to set its autofill id. Call AutofillManager.notifyViewEntered(View, int, Rect) and/or AutofillManager.notifyViewExited(View, int) when the focused virtual child changed. Override isVisibleToUserForAutofill(int) to allow the platform to query whether a given virtual view is visible to the user in order to support triggering save when all views of interest go away. Call AutofillManager.notifyValueChanged(View, int, AutofillValue) when the value of a virtual child changed. Call AutofillManager.notifyViewVisibilityChanged(View, int, boolean) when the visibility of a virtual child changed. Call AutofillManager.notifyViewClicked(View, int) when a virtual child is clicked. Call AutofillManager#commit() when the autofill context of the view structure changed and the current context should be committed (for example, when the user tapped a SUBMIT button in an HTML page). Call AutofillManager#cancel() when the autofill context of the view structure changed and the current context should be canceled (for example, when the user tapped a CANCEL button in an HTML page). Provide ways for users to manually request autofill by calling AutofillManager#requestAutofill(View, int, Rect). The left and top values set in ViewStructure#setDimens(int, int, int, int, int, int) must be relative to the next ViewGroup#isImportantForAutofill() predecessor view included in the structure. Views with virtual children support the Autofill Framework mainly by: Providing the metadata defining what the virtual children mean and how they can be autofilled. Implementing the methods that autofill the virtual children. This method is responsible for the former; autofill(android.util.SparseArray) is responsible for the latter. The ViewStructure traditionally represents a View, while for web pages it represent HTML nodes. Hence, it's necessary to "map" the HTML properties in a way that is understood by the AutofillService implementations: If the WebView implementation can determine that the value of a field was set statically (for example, not through Javascript), it should also call structure.setDataIsSensitive(false). For example, an HTML form with 2 fields for username and password: Username: Password: Would map to: int index = structure.addChildCount(2); ViewStructure username = structure.newChild(index); username.setAutofillId(structure.getAutofillId(), 1); // id 1 - first child username.setAutofillHints("username"); username.setHtmlInfo(username.newHtmlInfoBuilder("input") .addAttribute("type", "text") .addAttribute("name", "username") .addAttribute("label", "Username:") .build()); username.setHint("Email or username"); username.setAutofillType(View.AUTOFILL_TYPE_TEXT); username.setAutofillValue(AutofillValue.forText("Type your username")); // Value of the field is not sensitive because it was created statically and not changed. username.setDataIsSensitive(false); ViewStructure password = structure.newChild(index + 1); username.setAutofillId(structure, 2); // id 2 - second child password.setAutofillHints("current-password"); password.setHtmlInfo(password.newHtmlInfoBuilder("input") .addAttribute("type", "password") .addAttribute("name", "password") .addAttribute("label", "Password:") .build()); password.setHint("Password"); password.setAutofillType(View.AUTOFILL_TYPE_TEXT); Parameters structure ViewStructure: fill in with virtual children data for autofill purposes. flags int: optional flags. public void onProvideContentCaptureStructure (ViewStructure structure, int flags) Populates a ViewStructure for content capture. This method is called after a view that is eligible for content capture (for example, if it isImportantForContentCapture(), an intelligence service is enabled for the user, and the activity rendering the view is enabled for content capture) is laid out and is visible. The populated structure is then passed to the service through ContentCaptureSession#notifyViewAppeared(ViewStructure). The default implementation of this method sets the most relevant properties based on related View methods, and views in the standard Android widgets library also override it to set their relevant properties. Therefore, if overriding this method, it is recommended to call super.onProvideContentCaptureStructure(). Note: views that manage a virtual structure under this view must populate just the node representing this view and return right away, then asynchronously report (not necessarily in the UI thread) when the children nodes appear, disappear or have their text changed by calling ContentCaptureSession#notifyViewAppeared(ViewStructure), ContentCaptureSession#notifyViewDisappeared(AutofillId), and ContentCaptureSession#notifyViewTextChanged(AutofillId, CharSequence) respectively. The structure for a child must be created using ContentCaptureSession#newVirtualViewStructure(AutofillId, long), and the autofillId for a child can be obtained either through childStructure.getAutofillId() or ContentCaptureSession#newAutofillId(AutofillId, long). When the virtual view hierarchy represents a web page, you should also: Note: the following methods of the structure will be ignored: Parameters structure ViewStructure: This value cannot be null. flags int public void onProvideVirtualStructure (ViewStructure structure) Called when assist structure is being retrieved from a view as part of Activity.onProvideAssistData to generate additional virtual structure under this view. The default implementation uses getAccessibilityNodeProvider() to try to generate this from the view's virtual accessibility nodes, if any. You can override this for a more optimal implementation providing this data. Parameters structure ViewStructure public boolean onTouchEvent (MotionEvent event) Implement this method to handle touch screen motion events. If this method is used to detect click actions, it is recommended that the actions be performed by implementing and calling performClick(). This will ensure consistent system behavior, including: obeying click sound preferences dispatching OnClickListener calls handling ACTION_CLICK when accessibility features are enabled Parameters event MotionEvent: The motion event. Returns boolean True if the event was handled, false otherwise. public boolean onTrackballEvent (MotionEvent event) Implement this method to handle trackball motion events. The relative movement of the trackball since the last event can be retrieve with MotionEvent.getX() and MotionEvent.getY(). These are normalized so that a movement of 1 corresponds to the user pressing one DPAD key (so they will often be fractional values, representing the more fine-grained movement information available from a trackball). Parameters event MotionEvent: The motion event. Returns boolean True if the event was handled, false otherwise. public void onWindowFocusChanged (boolean hasWindowFocus) Called when the window containing this view gains or loses focus. Note that this is separate from view focus: to receive key events, both your view and its window must have focus. If a window is displayed on top of yours that takes input focus, then your own window will lose focus but the view focus will remain unchanged. Parameters hasWindowFocus boolean: True if the window containing this view now has focus, false otherwise. Added in API level 1 Deprecated in API level 23 public boolean overlayHorizontalScrollbar () This method was deprecated in API level 23. This method is now obsolete. Gets whether horizontal scrollbar has overlay style. Added in API level 1 Deprecated in API level 23 public boolean overlayVerticalScrollbar () This method was deprecated in API level 23. This method is now obsolete. Gets whether vertical scrollbar has overlay style. public boolean pageDown (boolean bottom) Scrolls the contents of this WebView down by half the page size. Parameters bottom boolean: true to jump to bottom of page Returns boolean true if the page was scrolled public boolean pageUp (boolean top) Scrolls the contents of this WebView up by half the view size. Parameters top boolean: true to jump to the top of the page Returns boolean true if the page was scrolled public void pauseTimers () Pauses all layout, parsing, and JavaScript timers for all WebViews. This is a global requests, not restricted to just this WebView. This can be useful if the application has been paused. public boolean performLongClick () Calls this view's OnLongClickListener, if it is defined. Invokes the context menu if the OnLongClickListener did not consume the event. Returns boolean true if one of the above receivers consumed the event, false otherwise public void postUrl (String url, byte[] postData) Loads the URL with postData using "POST" method into this WebView. If url is not a network URL, it will be loaded with loadUrl(java.lang.String) instead, ignoring the postData param. Parameters url String: the URL of the resource to load This value cannot be null. postData byte: the data will be passed to "POST" request, which must be be "application/x-www-form-urlencoded" encoded. This value cannot be null. public void postVisualStateCallback (long requestId, WebView.VisualStateCallback callback) Posts a VisualStateCallback, which will be called when the current state of the WebView is ready to be drawn. Because updates to the DOM are processed asynchronously, updates to the DOM may not immediately be reflected visually by subsequent WebView#onDraw invocations. The VisualStateCallback provides a mechanism to notify the caller when the contents of the DOM at the current time are ready to be drawn the next time the WebView draws. The next draw after the callback completes is guaranteed to reflect all the updates to the DOM up to the point at which the VisualStateCallback was posted, but it may also contain updates applied after the callback was posted. The state of the DOM covered by this API includes the following: primitive HTML elements (div, img, span, etc..) images CSS animations WebGL canvas It does not include the state of: To guarantee that the WebView will successfully render the first frame after the VisualStateCallback#onComplete method has been called a set of conditions must be met: When using this API it is also recommended to enable pre-rasterization if the WebView is off screen to avoid flickering. See WebSettings#setOffscreenPreRaster for more details and do consider its caveats. Parameters requestId long: An id that will be returned in the callback to allow callers to match requests with callbacks. callback WebView.VisualStateCallback: The callback to be invoked. This value cannot be null. public void postWebMessage (WebMessage message, Uri targetOrigin) Post a message to main frame. The embedded application can restrict the messages to a certain target origin. See HTML5 spec for how target origin can be used. A target origin can be set as a wildcard ("*"). However this is not recommended. See the page above for security issues. Content loaded via loadData(java.lang.String, java.lang.String, java.lang.String) will not have a valid origin, and thus cannot be sent messages securely. If you need to send messages using this function, you should use loadDataWithBaseURL(java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.lang.String) with a valid HTTP or HTTPS baseUrl to define a valid origin that can be used for messaging. Parameters message WebMessage: the WebMessage This value cannot be null. targetOrigin Uri: the target origin. This value cannot be null. public void reload () Reloads the current URL. public void removeJavascriptInterface (String name) Removes a previously injected Java object from this WebView. Note that the removal will not be reflected in JavaScript until the page is next (re)loaded. See addJavascriptInterface(Object, String). Parameters name String: the name used to expose the object in JavaScript This value cannot be null. public boolean requestChildRectangleOnScreen (View child, Rect rect, boolean immediate) Called when a child of this group wants a particular rectangle to be positioned onto the screen. ViewGroups overriding this can trust that: child will be a direct child of this group rectangle will be in the child's content coordinates ViewGroups overriding this should uphold the contract: nothing will change if the rectangle is already visible the view port will be scrolled only just enough to make the rectangle visible Parameters child View: The direct child making the request. This value cannot be null. rect Rect: The rectangle in the child's coordinates the child wishes to be on the screen. immediate boolean: True to forbid animated or delayed scrolling, false otherwise Returns boolean Whether the group scrolled to handle the operation public boolean requestFocus (int direction, Rect previouslyFocusedRect) Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. The rectangle can help give larger views a finer grained hint about where focus is coming from, and therefore, where to show selection, or forward focus change internally. A view will not actually take focus if it is not focusable (isFocusable() returns false), or if it is focusable and it is not focusable in touch mode (isFocusableInTouchMode()) while the device is in touch mode. A View will not take focus if it is not visible. A View will not take focus if one of its parents has ViewGroup.getDescendantFocusability() equal to ViewGroup#FOCUS_BLOCK_DESCENDANTS. See also focusSearch(int), which is what you call to say that you have focus, and you want your parent to look for the next one. You may wish to override this method if your custom View has an internal View that it wishes to forward the request to. Looks for a view to give focus to respecting the setting specified by getDescendantFocusability(). Uses onRequestFocusInDescendants(int, android.graphics.Rect) to find focus within the children of this group when appropriate. Parameters direction int: One of FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, and FOCUS_RIGHT previouslyFocusedRect Rect: The rectangle (in this View's coordinate system) to give a finer grained hint about where focus is coming from. May be null if there is no hint. Returns boolean Whether this view or one of its descendants actually took focus. public void requestFocusNodeHref (Message hrefMsg) Requests the anchor or image element URL at the last tapped point. If hrefMsg is null, this method returns immediately and does not dispatch hrefMsg to its target. If the tapped point hits an image, an anchor, or an image in an anchor, the message associates strings in named keys in its data. The value paired with the key may be an empty string. Parameters hrefMsg Message: the message to be dispatched with the result of the request. The message data contains three keys. "url" returns the anchor's href attribute. "title" returns the anchor's text. "src" returns the image's src attribute. This value may be null. public void requestImageRef (Message msg) Requests the URL of the image last touched by the user. msg will be sent to its target with a String representing the URL as its object. Parameters msg Message: the message to be dispatched with the result of the request as the data member with

"url" as key. The result can be null. public WebBackForwardList restoreState (Bundle inState) Restores the state of this WebView from the given Bundle. This method is intended for use in Activity.onRestoreInstanceState(Bundle) and should be called to restore the state of this WebView. If it is called after this WebView has had a chance to build state (load pages, create a back/forward list, etc.) there may be undesirable side-effects. Please note that this method no longer restores the display data for this WebView. Parameters inState Bundle: the incoming Bundle of state This value cannot be null. public void resumeTimers () Resumes all layout, parsing, and JavaScript timers for all WebViews. This will resume dispatching all timers. Added in API level 1 Deprecated in API level 18 public void savePassword (String host, String username, String password) This method was deprecated in API level 18. Saving passwords in WebView will not be supported in future versions. Sets a username and password pair for the specified host. This data is used by the WebView to autocomplete username and password fields in web forms. Note that this is unrelated to the credentials used for HTTP authentication. Parameters host String: the host that required the credentials username String: the username for the given host password String: the password for the given host public void saveWebArchive (String filename) Saves the current view as a web archive. Parameters filename String: the filename where the archive should be placed This value cannot be null. public void saveWebArchive (String basename, boolean autoname, ValueCallback callback) Saves the current view as a web archive. Parameters basename String: the filename where the archive should be placed This value cannot be null. autoname boolean: if false, takes basename to be a file. If true, basename is assumed to be a directory in which a filename will be chosen according to the URL of the current page. callback ValueCallback: called after the web archive has been saved. The parameter for onReceiveValue will either be the filename under which the file was saved, or null if saving the file failed. public void setBackgroundColor (int color) Sets the background color for this view. Parameters color int: the color of the background Added in API level 1 Deprecated in API level 17 public void setCertificate (SslCertificate certificate) This method was deprecated in API level 17. Calling this function has no useful effect, and will be ignored in future releases. Sets the SSL certificate for the main top-level page. Parameters certificate SslCertificate public static void setDataDirectorySuffix (String suffix) Define the directory used to store WebView data for the current process. The provided suffix will be used when constructing data and cache directory paths. If this API is not called, no suffix will be used. Each directory can be used by only one process in the application. If more than one process in an app wishes to use WebView, only one process can use the default directory, and other processes must call this API to define a unique suffix. This means that different processes in the same application cannot directly share WebView-related data, since the data directories must be distinct. Applications that use this API may have to explicitly pass data between processes. For example, login cookies may have to be copied from one process's cookie jar to the other using CookieManager if both processes' WebViews are intended to be logged in. Most applications should simply ensure that all components of the app that rely on WebView are in the same process, to avoid needing multiple data directories. The disableWebView() method can be used to ensure that the other processes do not use WebView by accident in this case. This API must be called before any instances of WebView are created in this process and before any other methods in the android.webkit package are called by this process. Parameters suffix String: The directory name suffix to be used for the current process. Must not contain a path separator. This value cannot be null. public void setDownloadListener (DownloadListener listener) Registers the interface to be used when content can not be handled by the rendering engine, and should be downloaded instead. This will replace the current handler. Parameters listener DownloadListener: an implementation of DownloadListener This value may be null. public void setFindListener (WebView.FindListener listener) Registers the listener to be notified as find-on-page operations progress. This will replace the current listener. Parameters listener WebView.FindListener: an implementation of FindListener This value may be null. Added in API level 1 Deprecated in API level 23 public void setHorizontalScrollbarOverlay (boolean overlay) This method was deprecated in API level 23. This method has no effect. Specifies whether the horizontal scrollbar has overlay style. Parameters overlay boolean: true if horizontal scrollbar should have overlay style Added in API level 1 Deprecated in API level 26 public void setHttpAuthUsernamePassword (String host, String realm, String username, String password) This method was deprecated in API level 26. Use WebViewDatabase#setHttpAuthUsernamePassword instead Stores HTTP authentication credentials for a given host and realm to the WebViewDatabase instance. Parameters host String: the host to which the credentials apply realm String: the realm to which the credentials apply username String: the username password String: the password public void setInitialScale (int scaleInPercent) Sets the initial scale for this WebView. 0 means default. The behavior for the default scale depends on the state of WebSettings#getUseWideViewPort() and WebSettings#getLoadWithOverviewMode(). If the content fits into the WebView control by width, then the zoom is set to 100%. For wide content, the behavior depends on the state of WebSettings#getLoadWithOverviewMode(). If its value is true, the content will be zoomed out to be fit by width into the WebView control, otherwise not. If initial scale is greater than 0, WebView starts with this value as initial scale. Please note that unlike the scale properties in the viewport meta tag, this method doesn't take the screen density into account. Parameters scaleInPercent int: the initial scale in percent public void setLayoutParams (ViewGroup.LayoutParams params) Set the layout parameters associated with this view. These supply parameters to the parent of this view specifying how it should be arranged. There are many subclasses of ViewGroup.LayoutParams, and these correspond to the different subclasses of ViewGroup that are responsible for arranging their children. Parameters params ViewGroup.LayoutParams: The layout parameters for this view, cannot be null Added in API level 1 Deprecated in API level 17 public void setMapTrackballToArrowKeys (boolean setMap) This method was deprecated in API level 17. Only the default case, true, will be supported in a future version. Parameters setMap boolean public void setNetworkAvailable (boolean networkUp) Informs WebView of the network state. This is used to set the JavaScript property window.navigator.isOnline and generates the online/offline event as specified in HTML5, sec. 5.7.7 Parameters networkUp boolean: a boolean indicating if network is available public void setOverScrollMode (int mode) Set the over-scroll mode for this view. Valid over-scroll modes are OVER_SCROLL_ALWAYS, OVER_SCROLL_IF_CONTENT_SCROLLS (allow over-scrolling only if the view content is larger than the container), or OVER_SCROLL_NEVER. Setting the over-scroll mode of a view will have an effect only if the view is capable of scrolling. Parameters mode int: The new over-scroll mode for this view. Added in API level 1 Deprecated in API level 15 public void setPictureListener (WebView.PictureListener listener) This method was deprecated in API level 15. This method is now obsolete. Sets the Picture listener. This is an interface used to receive notifications of a new Picture. Parameters listener WebView.PictureListener: an implementation of WebView.PictureListener public void setRendererPriorityPolicy (int rendererRequestedPriority, boolean waivedWhenNotVisible) Set the renderer priority policy for this WebView. The priority policy will be used to determine whether an out of process renderer should be considered to be a target for OOM killing. Because a renderer can be associated with more than one WebView, the final priority it is computed as the maximum of any attached WebViews. When a WebView is destroyed it will cease to be considered when calculating the renderer priority. Once no WebViews remain associated with the renderer, the priority of the renderer will be reduced to RENDERER_PRIORITY_WAIVED. The default policy is to set the priority to RENDERER_PRIORITY_IMPORTANT regardless of visibility, and this should not be changed unless the caller also handles renderer crashes with WebViewClient#onRenderProcessGone. Any other setting will result in WebView renderers being killed by the system more aggressively than the application. public static void setSafeBrowsingWhitelist (List hosts, ValueCallback callback) Sets the list of hosts (domain names/IP addresses) that are exempt from SafeBrowsing checks. The list is global for all the WebViews. Each rule should take one of these: Rule Example Matches Subdomain HOSTNAME example.com Yes .HOSTNAME .example.com No IPV4_LITERAL 192.168.1.1 No IPV6_LITERAL_WITH_BRACKETS [10:20:30:40:50:60:70:80]No All other rules, including wildcards, are invalid. The correct syntax for hosts is defined by RFC 3986. Parameters hosts List: the list of hosts This value cannot be null. callback ValueCallback: will be called with true if hosts are successfully added to the allowlist. It will be called with false if any hosts are malformed. The callback will be run on the UI thread This value may be null. public void setScrollBarStyle (int style) Specify the style of the scrollbars. The scrollbars can be overlaid or inset. When inset, they add to the padding of the view. And the scrollbars can be drawn inside the padding area or on the edge of the view. For example, if a view has a background drawable and you want to draw the scrollbars inside the padding specified by the drawable, you can use SCROLLBARS_INSIDE_OVERLAY or SCROLLBARS_INSIDE_INSET. If you want them to appear at the edge of the view, ignoring the padding, then you can use SCROLLBARS_OUTSIDE_OVERLAY or SCROLLBARS_OUTSIDE_INSET. Added in API level 1 Deprecated in API level 23 public void setVerticalScrollbarOverlay (boolean overlay) This method was deprecated in API level 23. This method has no effect. Specifies whether the vertical scrollbar has overlay style. Parameters overlay boolean: true if vertical scrollbar should have overlay style public void setWebChromeClient (WebChromeClient client) Sets the chrome handler. This is an implementation of WebChromeClient for use in handling JavaScript dialogs, favicons, titles, and the progress. This will replace the current handler. Parameters client WebChromeClient: an implementation of WebChromeClient This value may be null. See also: public static void setWebContentsDebuggingEnabled (boolean enabled) Enables debugging of web contents (HTML / CSS / JavaScript) loaded into any WebViews of this application. This flag can be enabled in order to facilitate debugging of web layouts and JavaScript code running inside WebViews. Please refer to WebView documentation for the debugging guide. The default is false. Parameters enabled boolean: whether to enable web contents debugging public void setWebViewClient (WebViewClient client) Sets the WebViewClient that will receive various notifications and requests. This will replace the current handler. Parameters client WebViewClient: an implementation of WebViewClient This value cannot be null. See also: public void setWebViewRendererProcessClient (Executor executor, WebViewRendererProcessClient webViewRendererProcessClient) Sets the renderer client object associated with this WebView. The renderer client encapsulates callbacks relevant to WebView renderer state. See WebViewRendererProcessClient for details. Although many WebView instances may share a single underlying renderer, and renderers may live either in the application process, or in a sandboxed process that is isolated from the application process, instances of WebViewRendererProcessClient are set per-WebView. Callbacks represent renderer events from the perspective of this WebView, and may or may not be correlated with renderer events affecting other WebViews. Parameters executor Executor: the Executor on which WebViewRendererProcessClient callbacks will execute. This value cannot be null. Callback and listener events are dispatched through this Executor, providing an easy way to control which thread is used. To dispatch events through the main thread of your application, you can use Context.getMainExecutor(). Otherwise, provide an Executor that dispatches to an appropriate thread. webViewRendererProcessClient WebViewRendererProcessClient: the WebViewRendererProcessClient object. This value cannot be null. public boolean shouldDelayChildPressedState () Return true if the pressed state should be delayed for children or descendants of this ViewGroup. Generally, this should be done for containers that can scroll, such as a List. This prevents the pressed state from appearing when the user is actually trying to scroll the content. The default implementation returns true for compatibility reasons. Subclasses that do not scroll should generally override this method and return false. Added in API level 11 Deprecated in API level 18 public boolean showFindDialog (String text, boolean showIme) This method was deprecated in API level 18. This method does not work reliably on all Android versions; implementing a custom find dialog using WebView.findAllAsync() provides a more robust solution. Starts an ActionMode for finding text in this WebView. Only works if this WebView is attached to the view system. Parameters text String: if non-null, will be the initial text to search for. Otherwise, the last String searched for in this WebView will be used to start. showIme boolean: if true, show the IME, assuming the user will begin typing. If false and text is non-null, perform a find all. Returns boolean true if the find dialog is shown, false otherwise public static void startSafeBrowsing (Context context, ValueCallback callback) Starts Safe Browsing initialization. URL loads are not guaranteed to be protected by Safe Browsing until after callback is invoked with true. Safe Browsing is not fully supported on all devices. For those devices callback will receive false. This should not be called if Safe Browsing has been disabled by manifest tag or WebSettings.setSafeBrowsingEnabled(boolean). This prepares resources used for Safe Browsing. This should be called with the Application Context (and will always use the Application context to do its work regardless). Parameters context Context: Application Context. This value cannot be null. callback ValueCallback: will be called on the UI thread with true if initialization is successful, false otherwise. This value may be null. public void stopLoading () Stops the current load. public void zoomBy (float zoomFactor) Performs a zoom operation in this WebView. Parameters zoomFactor float: the zoom factor to apply. The zoom factor will be clamped to the WebView's zoom limits. This value must be in the range 0.01 to 100.0 inclusive. public boolean zoomIn () Performs zoom in in this WebView. Returns boolean true if zoom in succeeds, false if no zoom changes public boolean zoomOut () Performs zoom out in this WebView. Returns boolean true if zoom out succeeds, false if no zoom changes protected int computeHorizontalScrollOffset () Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. This value is used to compute the position of the thumb within the scrollbar's track. The range is expressed in arbitrary units that must be the same as the units used by computeHorizontalScrollRange() and computeHorizontalScrollExtent(). The default offset is the scroll offset of this view. Returns int the horizontal offset of the scrollbar's thumb protected int computeHorizontalScrollRange () Compute the horizontal range that the horizontal scrollbar represents. The range is expressed in arbitrary units that must be the same as the units used by computeHorizontalScrollExtent() and computeHorizontalScrollOffset(). The default range is the drawing width of this view. Returns int the total horizontal range represented by the horizontal scrollbar protected int computeVerticalScrollExtent () Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. This value is used to compute the length of the thumb within the scrollbar's track. The range is expressed in arbitrary units that must be the same as the units used by computeVerticalScrollRange() and computeVerticalScrollOffset(). The default extent is the drawing height of this view. Returns int the vertical extent of the scrollbar's thumb protected int computeVerticalScrollOffset () Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range. This value is used to compute the position of the thumb within the scrollbar's track. The range is expressed in arbitrary units that must be the same as the units used by computeVerticalScrollRange() and computeVerticalScrollExtent(). The default offset is the scroll offset of this view. Returns int the vertical offset of the scrollbar's thumb protected int computeVerticalScrollRange () Compute the vertical range that the vertical scrollbar represents. The range is expressed in arbitrary units that must be the same as the units used by computeVerticalScrollExtent() and computeVerticalScrollOffset(). Returns int the total vertical range represented by the vertical scrollbar The default range is the drawing height of this view. protected void dispatchDraw (Canvas canvas) Called by draw to draw the child views. This may be overridden by derived classes to gain control just before its children are drawn (but after its own view has been drawn). Parameters canvas Canvas: the canvas on which to draw the view protected void onAttachedToWindow () This is called when the view is attached to a window. At this point it has a Surface and will start drawing. Note that this function is guaranteed to be called before onDraw(android.graphics.Canvas), however it may be called any time before the first onDraw -- including before or after onMeasure(int, int). If you override this method you must call through to the superclass implementation. protected void onConfigurationChanged (Configuration newConfig) Called when the current configuration of the resources being used by the application have changed. You can use this to decide when to reload resources that can changed based on orientation and other configuration characteristics. You only need to use this if you are not relying on the normal Activity mechanism of recreating the activity instance upon a configuration change. Parameters newConfig Configuration: The new resource configuration. protected void onDraw (Canvas canvas) Implement this to do your drawing. Parameters canvas Canvas: the canvas on which the background will be drawn protected void onFocusChanged (boolean focused, int direction, Rect previouslyFocusedRect) Called by the view system when the focus state of this view changes. When the focus change event is caused by directional navigation, direction and previouslyFocusedRect provide insight into where the focus is coming from. When overriding, be sure to call up through to the super class so that the standard focus handling will occur. If you override this method you must call through to the superclass implementation. Parameters focused boolean: True if the View has focus; false otherwise. direction int: The direction focus has moved when requestFocus() is called to give this view focus. Values are View.FOCUS_UP, View.FOCUS_DOWN, View.FOCUS_LEFT, View.FOCUS_RIGHT, View.FOCUS_FORWARD, or View.FOCUS_BACKWARD. It may not always apply, in which case use the default. Value is View.FOCUS_BACKWARD, View.FOCUS_FORWARD, View.FOCUS_LEFT, View.FOCUS_UP, View.FOCUS_RIGHT, or View.FOCUS_DOWN previouslyFocusedRect Rect: The rectangle, in this view's coordinate system, of the previously focused view. If applicable, this will be passed in as finer grained information about where the focus is coming from (in addition to direction). Will be null otherwise. protected void onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height. This method is invoked by measure(int, int) and should be overridden by subclasses to provide accurate and efficient measurement of their contents. CONTRACT: When overriding this method, you must call setMeasuredDimension(int, int) to store the measured width and height of this view. Failure to do so will trigger an IllegalStateException, thrown by measure(int, int). Calling the superclass' onMeasure(int, int) is a valid use. The base class implementation of measure defaults to the background size, unless a larger size is allowed by the MeasureSpec. Subclasses should override onMeasure(int, int) to provide better measurements of their content. If this method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (getSuggestedMinimumHeight() and getSuggestedMinimumWidth()). Parameters widthMeasureSpec int: horizontal space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec. heightMeasureSpec int: vertical space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec. protected void onOverScrolled (int scrollX, int scrollY, boolean clampedX, boolean clampedY) Called by overScrollBy(int, int, int, int, int, int, int, int, boolean) to respond to the results of an over-scroll operation. Parameters scrollX int: New X scroll value in pixels scrollY int: New Y scroll value in pixels clampedX boolean: True if scrollX was clamped to an over-scroll boundary clampedY boolean: True if scrollY was clamped to an over-scroll boundary protected void onScrollChanged (int l, int t, int oldl, int oldt) This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). This is typically as a result of scrollBy(int, int) or scrollTo(int, int) having been called. Parameters l int: Current horizontal scroll origin. t int: Current vertical scroll origin. oldl int: Previous horizontal scroll origin. oldt int: Previous vertical scroll origin. protected void onSizeChanged (int w, int h, int ow, int oh) This is called during layout when the size of this view has changed. If you were just added to the view hierarchy, you're called with the old values of 0. Parameters w int: Current width of this view. h int: Current height of this view. ow int: Old width of this view. oh int: Old height of this view. protected void onVisibilityChanged (View changedView, int visibility) Called when the visibility of the view or an ancestor of the view has changed. protected void onWindowVisibilityChanged (int visibility) Called when the window containing has change its visibility (between GONE, INVISIBLE, and VISIBLE). Note that this tells you whether or not your window is being made visible to the window manager; this does not tell you whether or not your window is obscured by other windows on the screen, even if it is itself visible.