Supporting Different Screens in Web Apps

In this document

See also

Because Android is available on devices with a variety of screen sizes and pixel densities, you should account for these factors in your web design so your web pages always appear at the appropriate size.

When targeting your web pages for Android devices, there are two major factors that you should account for:

The viewport

The viewport is the rectangular area that provides a drawable region for your web page. You can specify several viewport properties, such as its size and initial scale. Most important is the view port width, which defines the total number of horizontal pixels available from the web page's point of view (the number of CSS pixels available).

The screen density

The `WebView` class and most web browsers on Android convert your CSS pixel values to density-independent pixel values, so your web page appears at the same perceivable size as a medium-density screen (about 160dpi). However, if graphics are an important element of your web design, you should pay close attention to the scaling that occurs on different densities, because a 300px-wide image on a 320dpi screen will be scaled up (using more physical pixels per CSS pixel), which can produce artifacts (blurring and pixelation).

Specifying Viewport Properties

The viewport is the area in which your web page is drawn. Although the viewport's total visible area matches the size of the screen when zoomed all the way out, the viewport has its own pixel dimensions that it makes available to a web page. For example, although a device screen might have physical a width of 480 pixels, the viewport can have a width of 800 pixels. This allows a web page designed at 800 pixels wide to be completely visible on the screen when the viewport scale is 1.0. Most web browsers on Android (including Chrome) set the viewport to a large size by default (known as "wide viewport mode" at about 980px wide). Many browsers also zoom out as far as possible, by default, to show the full viewport width (known as "overview mode").

> **Note:** When your page is rendered in a `WebView`, it does not use wide viewport mode (the page appears at full zoom) by default. You can enable wide viewport mode with `setUseWideViewPort()`.

You can define properties of the viewport for your web page, such as the width and initial zoom level, using the `<meta name="viewport" ...>` tag in your document `<head>`.

The following syntax shows all of the supported viewport properties and the types of values accepted by each one:

```
<meta name="viewport"
      content="
          height = [pixel_value | "device-height"] ,
          width = [pixel_value | "device-width"] ,
          initial-scale = float_value ,
          minimum-scale = float_value ,
          maximum-scale = float_value ,
          user-scalable = ["yes" | "no"]
          " />
```

For example, the following `<meta>` tag specifies that the viewport width should exactly match the device screen's width and that the ability to zoom should be disabled:

```
<head>
    <title>Example</title>
    <meta name="viewport" content="width=device-width, user-scalable=no" />
</head>
```

When optimizing your site for mobile devices, you should usually set the width to `"device-width"` so the size fits exactly on all devices, then use CSS media queries to flexibly adapt layouts to suit different screen sizes.

> **Note:** You should disable user scaling only when you're certain that your web page layout is flexible and the content will fit the width of small screens.

Targeting Device Density with CSS

The Android Browser and `WebView` support a CSS media feature that allows you to create styles for specific screen densities—the `-webkit-device-pixel-ratio` CSS media feature. The value you apply to this feature should be either "0.75", "1", or "1.5", to indicate that the styles are for devices with low density, medium density, or high density screens, respectively.

For example, you can create separate stylesheets for each density:

```
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.5)" href="hdpi.css" />
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.0)" href="mdpi.css" />
```

Or, specify the different styles in one stylesheet:

```
#header {
    background:url(medium-density-image.png);
}

@media screen and (-webkit-device-pixel-ratio: 1.5) {
    /* CSS for high-density screens */
    #header {
        background:url(high-density-image.png);
    }
}

@media screen and (-webkit-device-pixel-ratio: 0.75) {
    /* CSS for low-density screens */
    #header {
        background:url(low-density-image.png);
    }
}
```

For more information about handling different screen densities, especially images, see High DPI Images for Variable Pixel Densities.

Targeting Device Density with JavaScript

The Android Browser and `WebView` support a DOM property that allows you to query the density of the current device—the `window.devicePixelRatio` DOM property. The value of this property specifies the scaling factor used for the current device. For example, if the value of `window.devicePixelRatio` is "1.0", then the device is considered a medium density device and no scaling is applied by default; if the value is "1.5", then the device is considered a high density device and the page is scaled 1.5x by default; if the value is "0.75", then the device is considered a low density device and the page is scaled 0.75x by default. Of course, the scaling that the Android Browser and `WebView` apply is based on the web page's target density—as described in the section about Defining the viewport target density, the default target is medium-density, but you can change the target to affect how your web page is scaled for different screen densities.

For example, here's how you can query the device density with JavaScript:

```
if (window.devicePixelRatio == 1.5) {
  alert("This is a high-density screen");
} else if (window.devicePixelRatio == 0.75) {
  alert("This is a low-density screen");
}
```