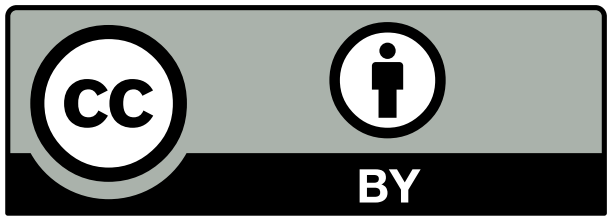
Responsive Web Design

Péter Jeszenszky

Faculty of Informatics, University of Debrecen

December 2, 2024

Terms of Use



Website Visits from Desktop and Mobile Devices

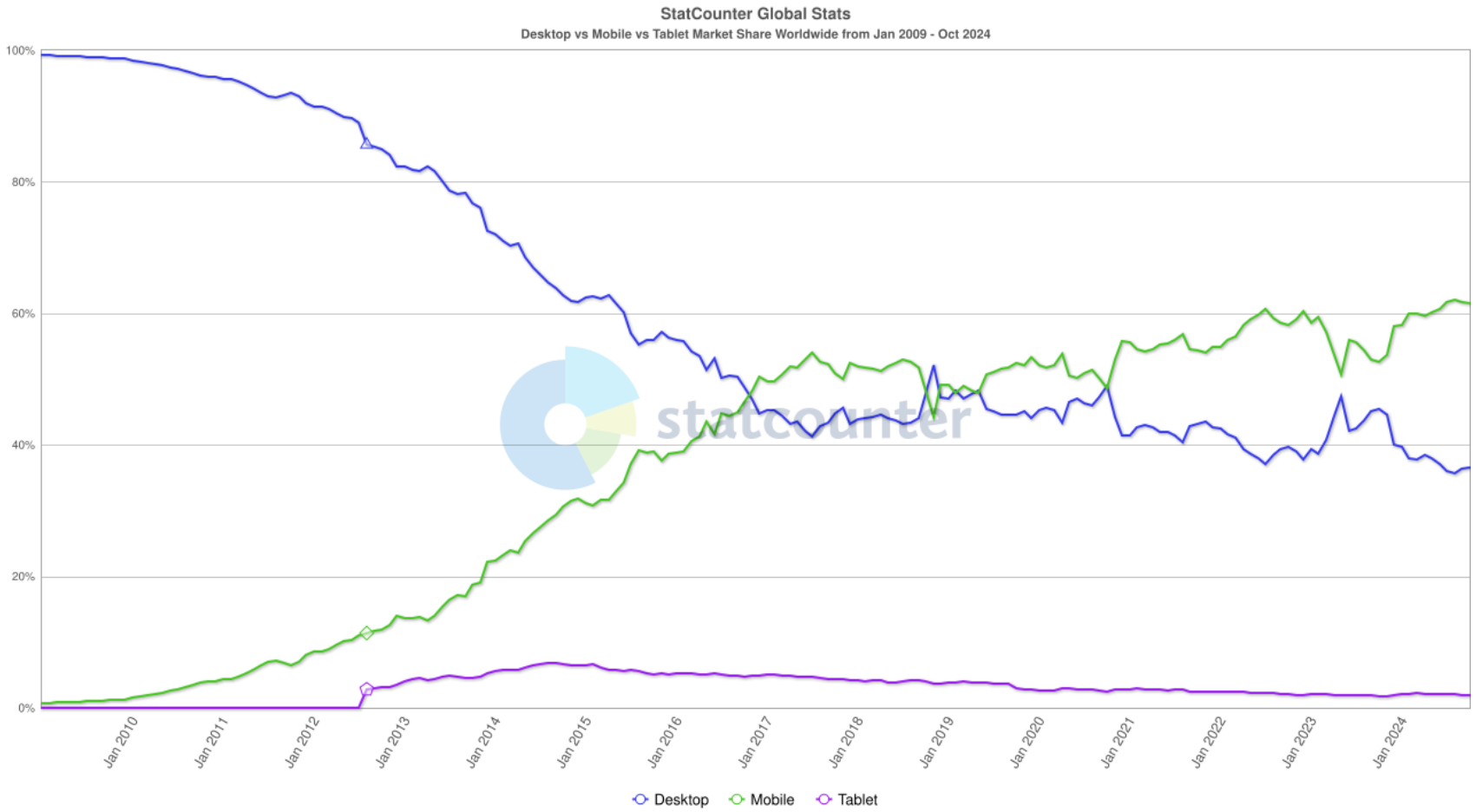


Figure 1: Source: StatCounter Global Stats – Desktop vs Mobile vs Tablet Market Share Worldwide

Mobilegeddon

- The term expresses the panic among web masters and web developers caused by Google's announcement about giving preference to mobile-friendly web pages when ranking search results for mobile devices.
  - See:
    - Chuck Price. "Mobilegeddon" Is Coming on April 21 – Are You Ready? March 9, 2015.
- Announcement:
  - Google Webmaster Central Blog. Rolling out the mobile-friendly update. April 21, 2015.

Mobile-First Indexing (1)

- "Mobile-first indexing means Google will predominantly use the mobile version of the content for indexing and ranking."
  - See: https://developers.google.com/search/mobile-sites/mobile-first-indexing
- Announcement:
  - Google Webmaster Central Blog. Rolling out mobile-first indexing. March 26, 2018.

Mobile-First Indexing (2)

Tools:

- Chrome Lighthouse:
  - License: Apache License 2.0
  - Website: https://developer.chrome.com/docs/lighthouse/
  - Repository: https://github.com/GoogleChrome/lighthouse
  - Lighthouse reports are available in Chrome DevTools on the Lighthouse tab.

Serving Mobile Content (1)

- The content negotiation mechanism of HTTP can be used to serve different representations of the same resource for different devices (e.g., for mobile and desktop devices).
- Adaptive and responsive web design makes it possible to serve content that adapts to the device.

Serving Mobile Content (2)

- It is still a common practice to serve mobile web content from a separate site hosted on a subdomain (often named `m`, such as `m.example.com`).
- Examples:
  - Facebook: https://m.facebook.com/
  - Wikipedia: https://en.m.wikipedia.org/wiki/Main_Page
  - YouTube: https://m.youtube.com/

What is Responsive Web Design (1)

- Responsive web design (RWD) is a web development approach for creating web pages that look and behave optimally on a wide range of devices, from mobile phones to desktop monitors, no matter what the window/screen size or the orientation is.
- Responsive web pages dynamically adjust themselves to the device on which they are displayed.
    - They respond properly to changes in the environment, such as resizing the window, or switching from portrait to landscape orientation.

What is Responsive Web Design (2)

- The term "Responsive Web Design" was coined by Ethan Marcotte in 2010.
  - See: Ethan Marcotte. Responsive Web Design. May 25, 2010.
- Further reading:
  - https://ethanmarcotte.com/
  - Ethan Marcotte. Responsive Web Design. 2 nd ed. A Book Apart, 2014.
  - Ethan Marcotte. Responsive Design: Patterns & Principles. A Book Apart, 2015.

What is Adaptive Web Design? (1)

- Adaptive web design (AWD) means the creation of multiple versions of the same web page tailored to different devices and serving the most suitable one to a specific user agent.
- Once a web page is loaded its layout will not adapt to changes in the environment.

What is Adaptive Web Design? (2)

The term "Adaptive Web Design" was coined by Aaron Gustafson in 2011.

- See: Aaron Gustafson. Adaptive Web Design – Crafting Rich Experiences with Progressive Enhancement. Easy Readers, 2011.

Page Layouts

See: https://web.archive.org/web/20190315030321/http://www.liquidapsive.com/

- **Static/fixed width**: a layout that uses a fixed width expressed in absolute units (typically in pixels).
- **Fluid/liquid**: a layout that uses a width expressed in relative units.
- **Adaptive**: can be considered as a series of static layouts.
  - Uses media queries to specify different static layouts for different viewport widths.
- **Responsive**: can be considered as a series of fluid layouts.
  - Uses media queries to specify different fluid layouts for different viewport widths.

Responsive Websites

Examples:

- Dropbox
- Facebook
- GitHub
- Microsoft
- Mozilla
- Wired
- YouTube
- ...

What is a Pixel? (1)

- Relevant CSS specification:
  - CSS Values and Units Module Level 3 (W3C Candidate Recommendation Draft, 22 March 2024)
- CSS pixels and device pixels are different!

What is a Pixel? (2)

Device pixel:

- The smallest unit of area on the device output capable of displaying its full range of colors.
  - For typical color screens, it's a square or somewhat rectangular region containing a red, green, and blue subpixel.
- CSS does not use device pixels to express lengths. Instead, it uses CSS pixels.

What is a Pixel? (3)

- Note that today's mobile phones have displays with very high resolutions.
  - If CSS were based on device pixels, the very small size of a device pixel would make most of the existing web pages unusable on mobile screens.
    - Text shown in a font size of, e.g., 12 device pixels would be incredibly small.
- To resolve the problem, a CSS pixel may consist of several device pixels.
  - The number of device pixels that make up a CSS pixel can vary.
  - Zooming is implemented by allocating more and more device pixels to a CSS pixel.

What is a Pixel? (4)

Reference pixel:

- The reference pixel is the visual angle of one pixel on a device with a pixel density of 96dpi and a distance from the reader of an arm's length.
  - For a nominal arm's length of 28 inches, the visual angle is therefore about 0.0213 degrees.
  - For reading at arm's length, 1px thus corresponds to about 0.26 mm (1/96 inch).
- See: https://www.w3.org/TR/css3-values/#reference-pixel

What is a Pixel? (5)

Reference pixel (continued):

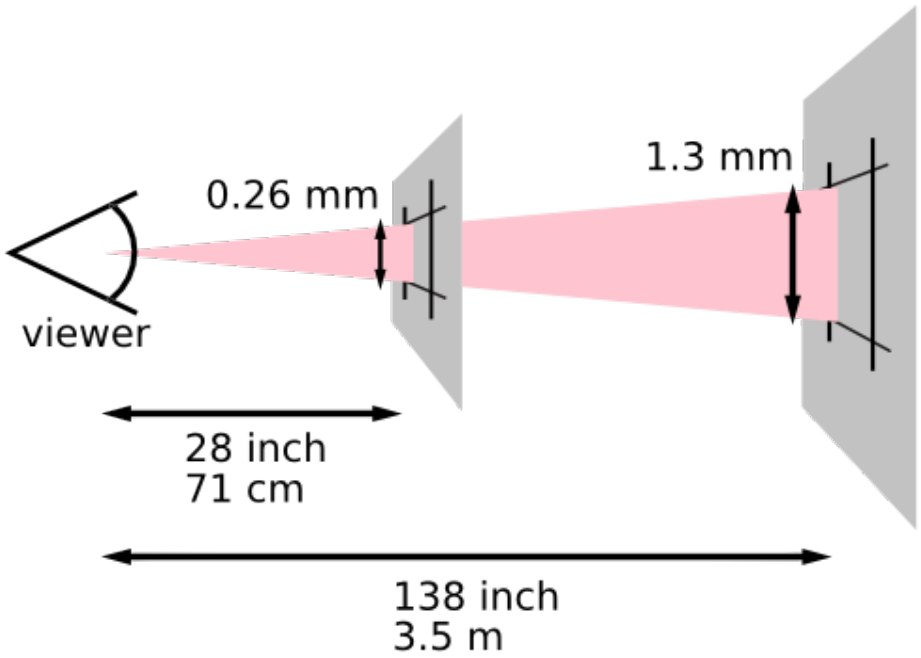


Figure 2: The image illustrates the effect of viewing distance on the size of a reference pixel: a reading distance of 71 cm (28 inches) results in a reference pixel of 0.26 mm, while a reading distance of 3.5 m (12 feet) results in a reference pixel of 1.3 mm.

What is a Pixel? (6)

- DPI (dots per inch), DPCM (dots per centimeter), PPI (pixels per inch), PPCM (pixels per centimeter):
  - They measure the dot/pixel density (resolution) of printing/image devices.
- The terms DPI (DPCM) and PPI (PPCM) are often used interchangeably.
- Online tools for determining the pixel density:
  - DPI love
  - mydevice.io

What is a Pixel? (7)

The absolute length units of CSS:

- They are fixed in relation to each other and anchored to some physical measurement.

Unit	Name	Equivalence
`cm`	centimeters	$1\texttt{cm} = 96\texttt{px}/2.54$
`mm`	millimeters	$1\texttt{mm} = 1\texttt{cm}/10$
`Q`	quarter-millimeters	$1\texttt{Q} = 1\texttt{cm}/40$
`in`	inches	$1\texttt{in} = 2.54\texttt{cm} = 96\texttt{px}$
`pc`	picas	$1\texttt{pc} = 1\texttt{in}/6$
`pt`	points	$1\texttt{pt} = 1\texttt{in}/72$
`px`	pixels	$1\texttt{px} = 1\texttt{in}/96$

What is a Pixel? (8)

- The absolute length units are anchored either:
  - by relating the physical units (i.e., `cm`, `mm`, `Q`, `in`, `pc`, `pt`) to their physical measurements, or
  - by relating the `px` unit to the reference pixel.
- The `px` unit is also called the **visual angle unit**.

What is a Pixel? (9)

- For screen media, low-resolution devices, and devices with unusual viewing distances, the anchor unit should be the `px` unit.
  - The unit `px` represents the whole number of device pixels that best approximates the reference pixel.
  - On such devices the physical units (e.g., `cm`, `in`, `mm`) might not match their physical counterparts!
- For print media at typical viewing distances, the anchor unit should be one of the standard physical units (e.g., `cm`, `in`, `mm`).
  - On such devices the `px` unit might not map to a whole number of device pixels!

What is a Pixel? (10)

The `devicePixelRatio` attribute of the `Window` interface returns the ratio of the CSS pixel size to the device pixel size for the current display device.

- See: MDN Web Docs – Window.devicePixelRatio

What is a Pixel? (11)

Demonstration: https://pixel-example.surge.sh/

What is a Pixel? (12)

Further recommended reading:

- Peter-Paul Koch. A pixel is not a pixel is not a pixel. 20 April 2010.

Relative length units of CSS

- `em`: equals to the computed value of the `font-size` property of the element on which it is used.
- `ex`: equals to the x-height of the element's font (in case of the of the `font-size` property itself it refers to the x-height of the parent element).
  - It is so called because it is often equal to the height of the lowercase *x* character.
- `rem`: equals to the computed value of the `em` unit on the root element.
- `vw`, `vh`, `vmin`, `vmax`: see later.

The Building Blocks of Responsive Web Design

- The main components of responsive design:
  - Media queries
  - Viewport
  - Responsive layouts (fluid/flexible grids)
  - Responsive images
  - Responsive typography
- See: MDN Web Docs – The building blocks of responsive design

Viewport

- User agents for continuous media generally offer users a viewport (a window or other viewing area on the screen) through which users consult a document.
  - User agents may change the document's layout when the viewport is resized.
  - When the viewport is smaller than the area of the canvas on which the document is rendered, the user agent should offer a scrolling mechanism.
  - There is at most one viewport per canvas, but user agents may render to more than one canvas (i.e., provide different views of the same document).
- See: CSS 2.1 Specification – Visual formatting model – The viewport

CSS Viewport Units

- Relevant specification:
  - CSS Values and Units Module Level 3 (W3C Candidate Recommendation Draft, 22 March 2024)
- Viewport-percentage lengths:
  - `vw`: equals to 1% of the width of the viewport.
  - `vh`: equals to 1% of the height of the viewport.
  - `vmin`: equals to the smaller of `vw` or `vh`.
  - `vmax`: equals to the larger of `vw` or `vh`.

What are Media Queries? (1)

- A media query is a method of testing certain aspects of the user agent or device that the document is being displayed in.
- They are used to conditionally apply styles to a document.

What are Media Queries? (2)

- Media queries are (almost) always independent of the document being rendered.
- They're only dependent on "external" information unless another feature explicitly specifies that it affects their resolution.
  - See, for example, the `@viewport` at-rule.

Media Query Specification

- Current standard:
  - Media Queries Level 3 (W3C Recommendation, 21 May 2024)
- The forthcoming next version of the standard:
  - Media Queries Level 4 (W3C Candidate Recommendation Draft, 25 December 2021)

Media Query Support

See: https://caniuse.com/css-mediaqueries

Availability of Media Queries

Media queries can be used in the following languages:

- CSS
- XML
- HTML
- JavaScript

Media Queries in CSS (1)

`@import`:

- Relevant specification:
  - CSS Cascading and Inheritance Level 3 (W3C Recommendation, 11 February 2021)
    - See: Importing Style Sheets: the @import rule
- Example:

```
@import "screen.css" screen;
@import "print.css" print;
```

Media Queries in CSS (2)

`@import`: (continued)

- The optional media query list (the import conditions) states the conditions under which the `@import` applies.
- If the media query does not match, the rules in the imported stylesheet do not apply, exactly as if the imported stylesheet were wrapped in an `@media` block with the given media query.

Media Queries in CSS (3)

`@media`:

- Relevant specification:
  - CSS Conditional Rules Module Level 3 (W3C Candidate Recommendation Draft, 15 August 2024)
    - Lásd: Media-specific style sheets: the @media rule
- Example:

```
@media screen {
  * {
    font-family: sans-serif;
  }
}
```

Media Queries in CSS (4)

`@media`: (continued)

- A `@media` rule is a conditional group rule whose condition is a media query.
- Conditional group rules associate a condition with a group of other CSS rules.
  - Each conditional group rule has a condition, which at any time evaluates to true or false.
  - When the condition is true, CSS processors must apply the rules inside the group rule as though they were at the group rule's location; when the condition is false, CSS processors must not apply any of rules inside the group rule.

Media Queries in XML

Relevant specification:

- Associating Style Sheets with XML documents 1.0 (Second Edition) (W3C Recommendation, 28 October 2010)
  - The `media` pseudo-attribute of the `xml-stylesheet` processing instruction gives the media for which the referenced stylesheet applies.
- Example:

```
<?xml-stylesheet type="text/css" media="screen" href="screen.css"?>
<?xml-stylesheet type="text/css" media="print" href="print.css"?>
```

Media Queries in HTML

- Relevant specification: HTML Living Standard
- See the `media` attribute that can be specified on the following elements:
  - `link`
  - `meta`
  - `source`
  - `style`
- Example:

```
<link rel="stylesheet" media="screen" href="screen.css">
<link rel="stylesheet" media="print" href="print.css">
```

Media Queries in JavaScript (1)

- Relevant specification: CSSOM View Module (W3C Working Draft, 17 March 2016)
- The `MediaQueryList` interface allows the evaluation of a media query programatically.
  - Event listeners can be registered to be notified when the result of the media query changes.
  - See: The MediaQueryList Interface
- Support: https://caniuse.com/mdn-api_mediaquerylist

Media Queries in JavaScript (2)

Example: (source: MDN Web Docs – Testing media queries programmatically)

```
// Create a media query list:
var mql = window.matchMedia("(orientation: portrait)");

// Check the result of the query:
if (mql.matches) {
  // The viewport is currently in portrait orientation
} else {
  // The viewport is currently in landscape orientation
}
```

Media Queries in JavaScript (3)

Example: (continued)

```
// Create a media query list:
var mql = window.matchMedia("(orientation: portrait)");

// Define a callback function for the event listener:
function handleOrientationChange(event) {
  if (event.matches) {
    // The viewport is currently in portrait orientation
  } else {
    // The viewport is currently in landscape orientation
  }
}

// Register the callback function as an event listener:
mql.addListener(handleOrientationChange);
```

Units in Media Queries

- The units used in media queries are the same as in other parts of CSS.
- Relative length units in media queries are based on the initial value, which means that units are never based on results of declarations.
    - For example, in HTML, the `em` unit is relative to the initial value of `font-size`, defined by the user agent or the user's preferences, not any styling on the page.

Media Query Syntax (1)

A media query consists of an optional media query modifier, an optional media type, and a media condition:

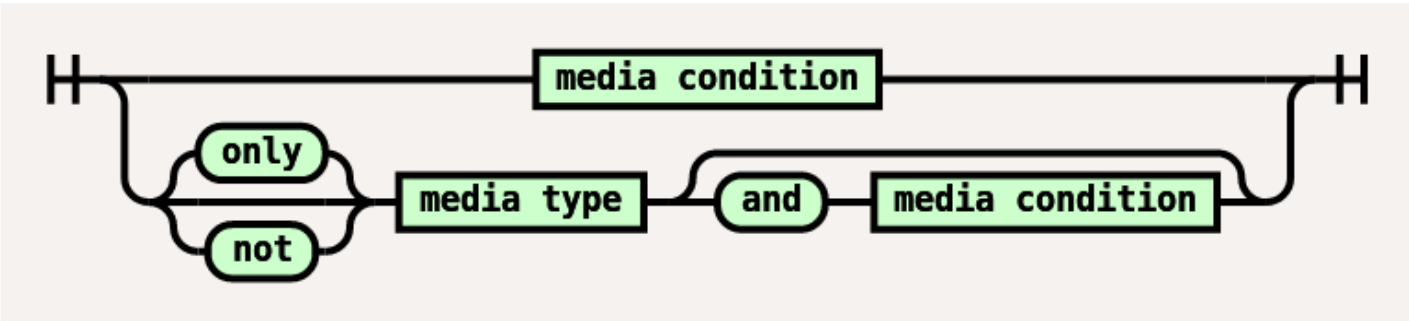


Figure 3: Media query syntax

Media Query Syntax (2)

Several media queries can be combined into a comma-separated media query list:

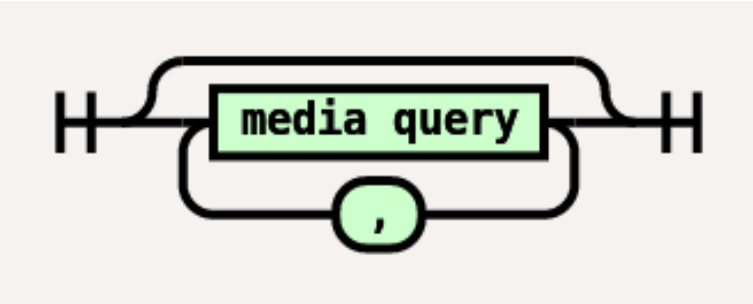


Figure 4: Media query list syntax

Media Query Evaluation (1)

- A media query is a logical expression that is either true or false. A media query is true if both of the following conditions are met:
  - The media type, if specified, matches the media type of the device where the user agent is running.
  - The media condition is true.
- A media query list is true if any of its component media queries are true, and false only if all of its component media queries are false (i.e., logical "or").
  - Example:

```
@media screen, print { /* ... */ }
```

Media Query Evaluation (2)

- User agents must re-evaluate media queries in response to changes in the user environment that they're aware of, and change the behavior of any constructs dependent on those media queries accordingly.
- For example, if the device is tilted from landscape to portrait orientation.

Media Query Modifiers (1)

A media query may optionally be prefixed by a single media query modifier, which is a single keyword which alters the meaning of the following media query.

Media Query Modifiers (2)

`not` modifier:

- An individual media query can have its result negated by prefixing it with the keyword `not`.
- If the media query would normally evaluate to true, prefixing it with `not` makes it evaluate to false, and vice versa.
- Example:

```
<link rel="stylesheet" media="not screen and (color)"
      href="style.css">
```

Media Query Modifiers (3)

`only` modifier:

- To hide media queries from legacy user agents, the media query can be prefixed with the `only`.
- The `only` keyword has no effect on the media query's result, but will cause the media query to be parsed by legacy user agents as specifying the unknown media type `only`, and thus be ignored.
- Example:

```
<link rel="stylesheet" media="only screen and (color)"
      href="style.css">
```

Media Types (1)

- A media type is a broad category of user-agent devices on which a document may be displayed.
- Not to be confused with IANA media types, such as `text/html`!

Media Types (2)

The following media types are defined for use in media queries:

- `all`: matches all devices.
- `print`: matches printers, and devices intended to reproduce a printed display, such as a web browser showing a document in "Print Preview".
- `screen`: matches all devices that aren't matched by `print`.

Media Types (3)

- In addition, the following deprecated media types are defined: `tty`, `tv`, `projection`, `handheld`, `braille`, `embossed`, `aural`, `speech`.
- Authors must not use these media types; instead, it is recommended that they select appropriate media features that better represent the aspect of the device that they are attempting to style against.
- User agents must recognize these media types as valid, but must make them match nothing.

Media Features (1)

A media feature is a more fine-grained test than media types, testing a single, specific feature of the user agent or display device.

- Examples: `width`, `height`, `orientation`, `resolution`

Media Features (2)

- Syntactically, media features resemble CSS properties: they consist of a feature name, a colon, and a value to test for.
- They may also be written in boolean form as just a feature name, or in range form with a comparison operator.

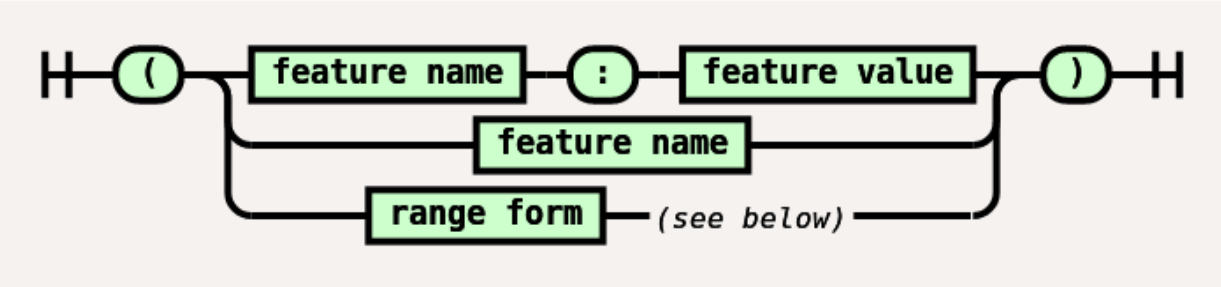


Figure 5: Media feature syntax

Media Features (3)

Differences between properties and media features:

- Properties are used to give information about how to present a document. Media features are used to describe requirements of the output device.
- Media features are always wrapped in parentheses and combined with the `and` or `or` keywords, like `(color) and (min-width: 600px)`, rather than being separated with semicolons.
- A media feature may be given with only its name (omitting the colon and value) to evaluate the feature in a boolean context. This is a convenient shorthand for features that have a reasonable value representing 0 or `none`.
  - For example, `(color)` is true if the `color` media feature is non-zero.

Media Features (4)

Differences between properties and media features: (continued)

- Media features with range type can be written in a range context, which uses standard mathematical comparison operators rather than a colon, or have their feature names prefixed with `min-` or `max-`.
- Properties sometimes accept complex values, e.g., calculations that involve several other values. Media features only accept single values: one keyword, one number, etc.

Media Features (5)

- If a media feature references a concept which does not exist on the device where the UA is running, the media feature must always evaluate to false.
- For example, speech UAs do not have a concept of “width”.

Media Feature Types

- Every media feature defines its type as either range or discrete.
  - Discrete media features take their values from a set.
    - The values may be keywords or boolean numbers (0 and 1), but the common factor is that there's no intrinsic "order" to them.
  - Range media features take their values from a range.
    - Any two values can be compared to see which is lesser and which is greater.
- The only significant difference between the two types is that range media features can be evaluated in a range context and accept `min-` or `max-` prefixes on their name.

Evaluating Media Features in a Boolean Context

- Media features can be written more simply as just the feature name, like `(color)`.
- When written like this, the media feature is evaluated in a boolean context.
  - If the feature would be true for any value other than the number 0, a dimension with the value 0, or the keyword `none`, the media feature evaluates to true.
  - Otherwise, it evaluates to false.

Evaluating Media Features in a Range Context (1)

Media features with a range type can be alternately written in a range context that takes advantage of the fact that their values are ordered, using ordinary mathematical comparison operators:

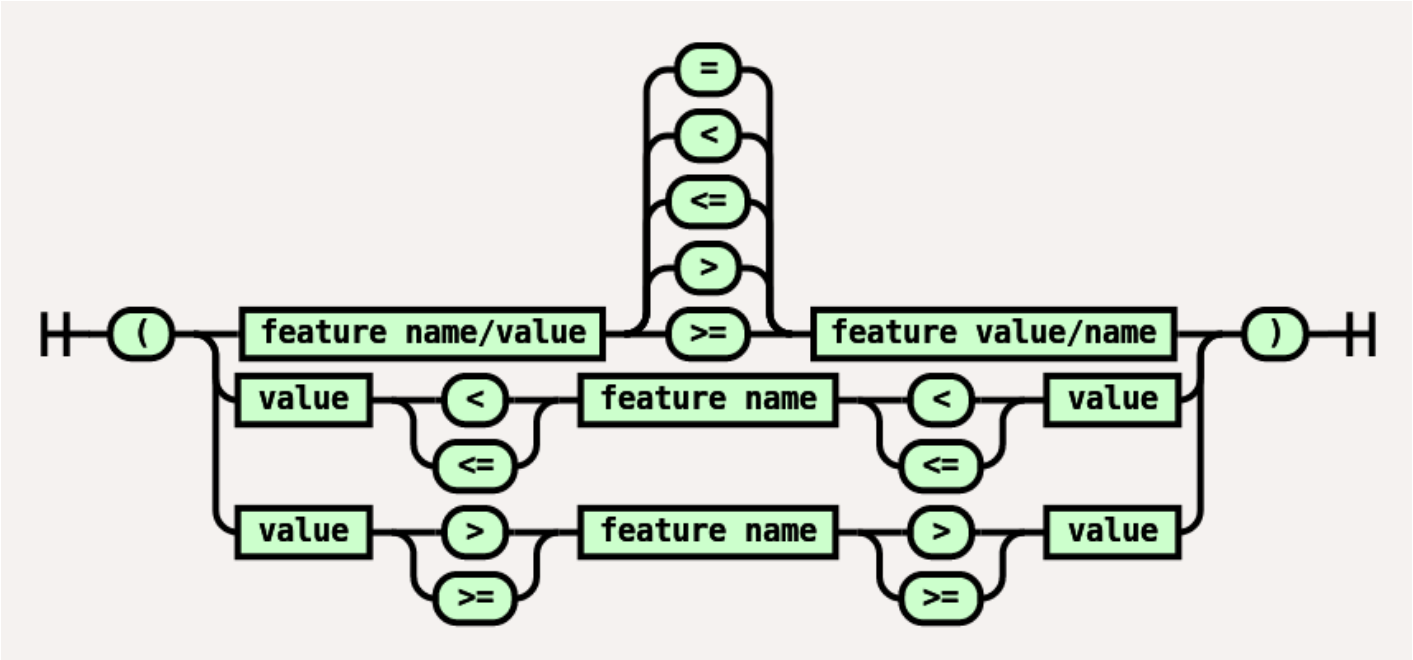


Figure 6: Media feature syntax in range context

Evaluating Media Features in a Range Context (2)

- The basic form, consisting of a feature name, a comparison operator, and a value, returns true if the relationship is true.
  - Example: `(height > 600px)`, `(600px < height)`
- The remaining forms, with the feature name nested between two value comparisons, returns true if both comparisons are true.
  - Example: `(400px < width < 1000px)`

Evaluating Media Features in a Range Context (3)

- Support:
  - Supported by Firefox since 2018.
  - Supported by Chromium-based browsers since 2022.
  - Support was added recently (i.e., in March 2023) in Safari.
- See: https://caniuse.com/css-media-range-syntax

Using `min-` and `max-` Prefixes on Range Features

- Rather than evaluating a range type media feature in a range context, the feature may be written as a normal media feature, but with a `min-` or `max-` prefix on the feature name.
  - Using a `min-` prefix on a feature name is equivalent to using the `>=` operator.
    - For example, `(min-height: 600px)` is equivalent to `(height >= 600px)`.
  - Using a `max-` prefix on a feature name is equivalent to using the `<=` operator.
    - For example, `(max-width: 40em)` is equivalent to `(width <= 40em)`.
- Discrete type media features do not accept `min-` or `max-` prefixes.

Combining Media Features

- Multiple media features can be combined together into a media condition using full boolean algebra (`not`, `and`, `or`).
  - Any media feature can be negated by placing not before it, such as `not (color)`.
  - Two or more media features can be chained together by placing `and`/`or` between them.
    - Examples: `(width < 600px) and (height < 600px)`, `update: slow) or (hover: none)`
  - Media conditions can be grouped by wrapping them in parentheses, such as `not ((color) or (hover))`.
- It is invalid to mix `and`/`or`/`not` at the same "level" of a media query.
  - For example, `(color) and (pointer) or (hover)` is illegal.

Error Handling in Media Queries

A syntactically invalid media query must be replaced by `not all` during parsing.

Available Media Features (1)

Viewport/page dimensions media features:

- `aspect-ratio: <ratio>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/aspect-ratio
- `height: <length>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/height
- `orientation: portrait | landscape`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/orientation
- `width: <length>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/width

Available Media Features (2)

Display quality media features:

- `grid: 0 | 1`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/grid
- `overflow-block: none | scroll | paged`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/overflow-block
- `overflow-inline: none | scroll`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/overflow-inline
- `resolution: <resolution> | infinite`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/resolution
- `scan: interlace | progressive`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/scan
- `update: none | slow | fast`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/update-frequency

Available Media Features (3)

Color media features:

- `color: <integer>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/color
- `color-index: <integer>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/color-index
- `color-gamut: srgb | p3 | rec2020`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/color-gamut
- `monochrome: <integer>`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/monochrome

Available Media Features (4)

Interaction media features:

- `any-hover: none | hover`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/any-hover
- `any-pointer: none | coarse | fine`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/any-hover
- `hover: none | hover`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/hover
- `pointer: none | coarse | fine`
  https://developer.mozilla.org/en-US/docs/Web/CSS/@media/pointer

Media Query Examples

```
@media (orientation: landscape) { /* ... */ }

@media (orientation: portrait) { /* ... */ }

@media print and not (color) { /* ... */ }

@media screen and (prefers-color-scheme: dark) { /* ... */ }

@media screen and (max-width: 576px) { /* ... */ }

@media screen and (min-width: 576px) and (max-width: 768px) {
  /* ... */
}

@media screen and (min-width: 768px) { /* ... */ }
```

Media Query Order (1)

Example: mobile first design

```
/* Small devices (phones, less than 768px): */
main {
  width: 100%;
}

/* Medium devices (tablets, 768px and up): */
@media (min-width: 768px) {
  main {
    width: 720px;
  }
}

/* Large devices (desktops, 992px and up): */
@media (min-width: 992px) {
  main {
    width: 960px;
  }
}
```

Media Query Order (2)

Example: desktop first design

```
/* Large devices (desktops, 992px and up): */
main {
  width: 960px;
}

/* Medium devices (tablets, less than 992px): */
@media (max-width: 992px) {
  main {
    width: 720px
  }
}

/* Small devices (phones, less than 768px): */
@media (max-width: 768px) {
  main {
    width: 100%;
  }
}
```

Typical Media Query Breakpoints (1)

Example: Bootstrap – Responsive breakpoints

```
// Extra small devices (portrait phones, less than 576px):
// No media query for `xs` since this is the default in Bootstrap
/* ... */

// Small devices (landscape phones, 576px and up):
@media (min-width: 576px) { /* ... */ }

// Medium devices (tablets, 768px and up):
@media (min-width: 768px) { /* ... */ }

// Large devices (desktops, 992px and up):
@media (min-width: 992px) { /* ... */ }

// Extra large devices (large desktops, 1200px and up):
@media (min-width: 1200px) { /* ... */ }

// XX-Large devices (larger desktops, 1400px and up):
@media (min-width: 1400px) { /* ... */ }
```

Typical Media Query Breakpoints (2)

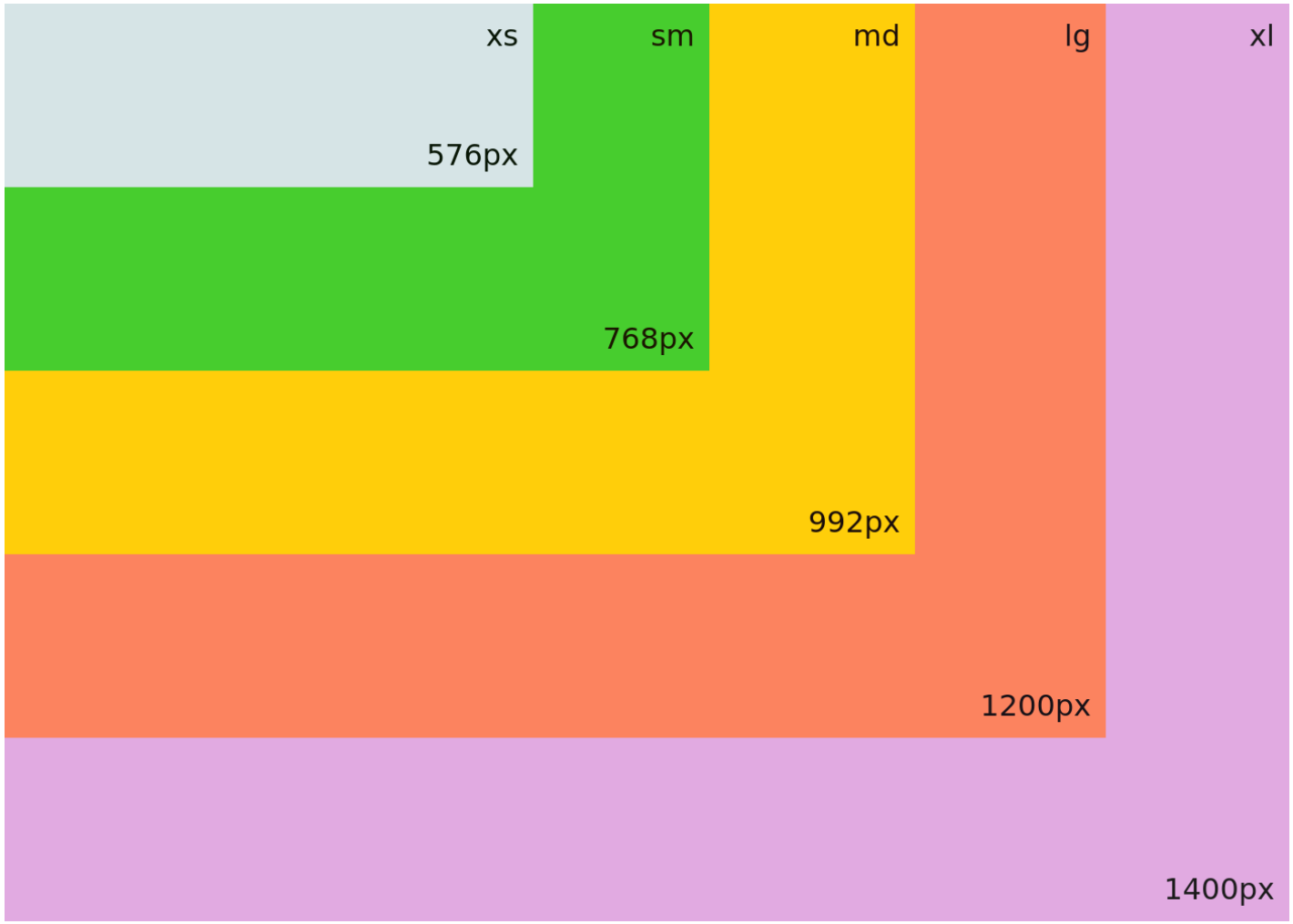


Figure 7: The responsive breakpoints of Bootstrap

Media Queries Level 5

- Specification: Media Queries Level 5 (W3C Working Draft, 18 December 2021)
- Example for a new media feature:
  - `prefers-color-scheme: light | dark`
    https://developer.mozilla.org/en-US/docs/Web/CSS/outline
    - Support: https://caniuse.com/prefers-color-scheme
    - E.g., Firefox: see the `ui.systemUsesDarkTheme` option (`about:config`).
  - See also: Thomas Steiner. prefers-color-scheme: Hello darkness, my old friend

Calculations in the Values of Media Features (1)

- The media queries specification states the following:
  Properties sometimes accept complex values, e.g., calculations that involve several other values. ***Media features only accept single values****: one keyword, one number, etc.*
- However, it seems that all major browsers allow the use of the `calc()` function in the values of media features.
  - See: https://caniuse.com/mdn-css_at-rules_media_calc

Calculations in the Values of Media Features (2)

For example, the following works in all major browsers:

```
@media screen and (max-width: calc(10in + 5em)) {
  .cards {
    flex-direction: column;
  }
}
```

Media Queries: Further Reading

- MDN Web Docs – Media queries
- MDN Web Docs – Using media queries
- MDN Web Docs – Using Media Queries for Accessibility

Viewport (1)

- Mobile/handheld device browsers have a viewport that is generally a lot narrower than a desktop browser window at a zoom level that gives a CSS pixel size recommended by CSS.
- The narrow viewport is a problem for documents designed to look good in desktop browsers.

Viewport (2)

- Thus, mobile browser maintain two viewports:
  - **Visual viewport**: the part of the page that's currently shown on-screen.
  - **Layout viewport**: the viewport into which the browser draws a web page.
- See:
  - MDN Web Docs Glossary – Viewport
  - MDN Web Docs Glossary – Visual Viewport
  - MDN Web Docs Glossary – Layout Viewport

Viewport (3)

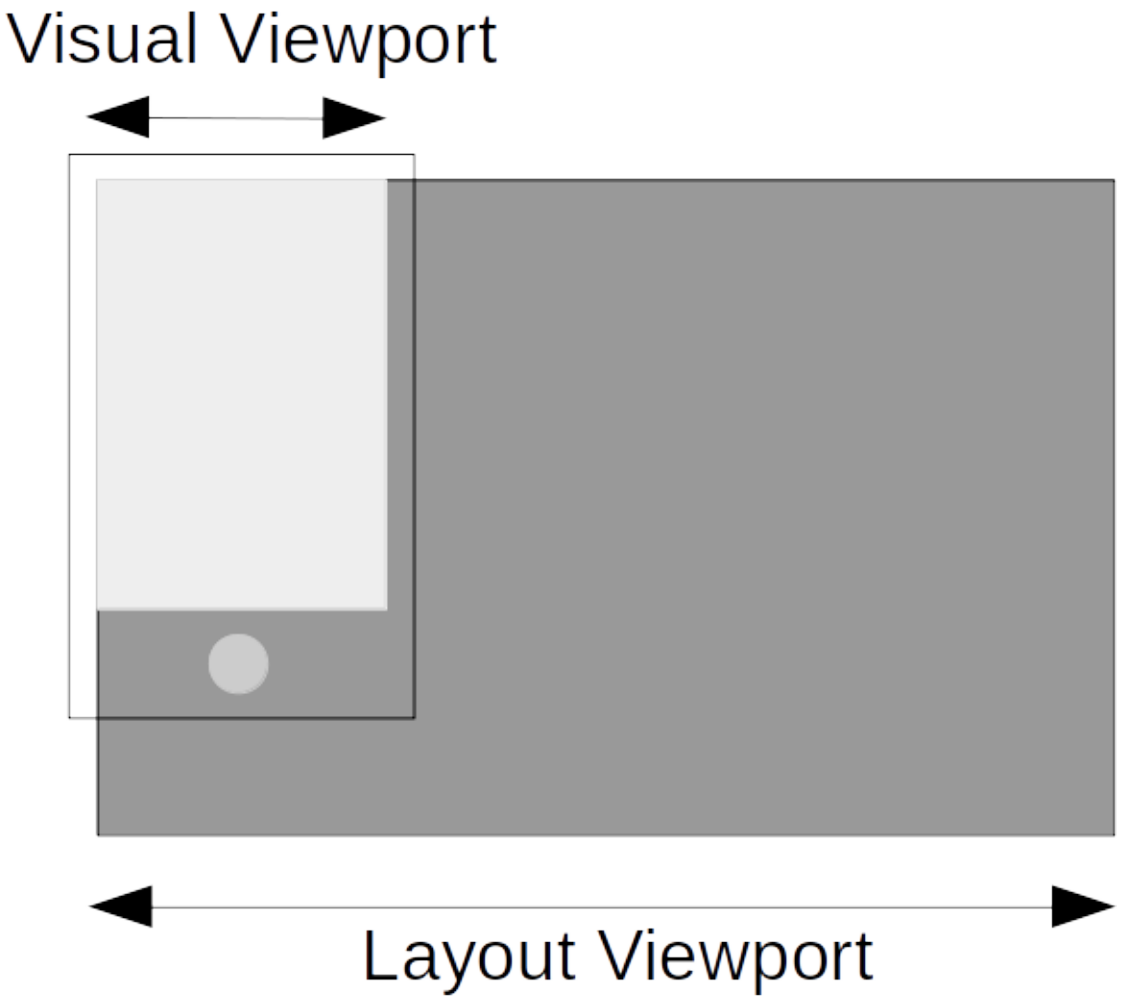


Figure 8: The two viewports

Viewport (4)

Testing viewport sizes in media queries:

- On mobile/handheld device browsers, the `height` and `width` media features test the height and width of the layout viewport, respectively.
  - See: https://www.quirksmode.org/css/tests/mediaqueries/width.html

Viewport (5)

Querying viewport sizes in JavaScript:

- Layout viewport width and height:
  `document.documentElement.clientWidth`,
  `document.documentElement.clientHeight`
  - https://developer.mozilla.org/en-US/docs/Web/API/Element/clientWidth
  - https://developer.mozilla.org/en-US/docs/Web/API/Element/clientHeight
- Visual viewport width and height:
  `window.innerWidth`/`window.visualViewport.width`, ,
  `window.innerHeight`/`window.visualViewport.height`
  - https://developer.mozilla.org/en-US/docs/Web/API/Window
  - https://developer.mozilla.org/en-US/docs/Web/API/Window/visualViewport

Viewport (6)

Common viewport sizes for mobile and tablet devices:

- Screen Sizes – Viewport Sizes and Pixel Densities for Popular Devices
- YesViz.com – Viewport Size for Devices

Viewport (7)

- The `viewport meta` tag has been introduced for allowing an author to specify the size of the (layout) viewport, and the initial zoom factor directly.
- It was first implemented by Apple for the Safari/iPhone browser, but has since been implemented for many other mobile browsers.
- Example:

```
<meta name="viewport"
      content="width=device-width, initial-scale=1">
```

- Standardization effort: CSS Viewport Module Level 1 (W3C First Public Working Draft, 25 January 2024)

Viewport (8)

Examples that demonstrate the presence/absence of the `viewport meta` tag in HTML:

https://jeszy75.github.io/viewport-example/
https://github.com/jeszy75/viewport-example/

Viewport (9)

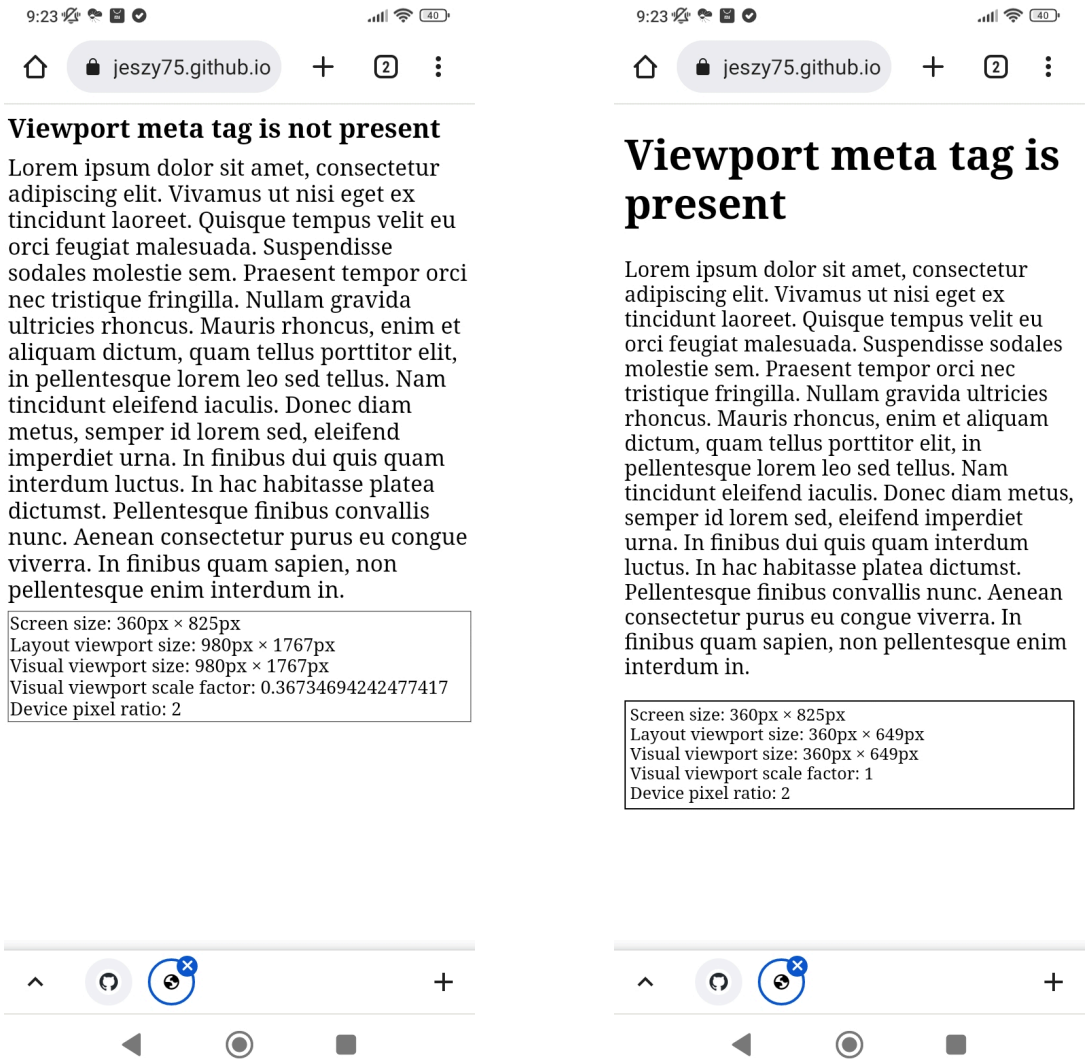


Figure 9: The effect of the absence/presence of the `viewport meta` tag

Viewport (10)

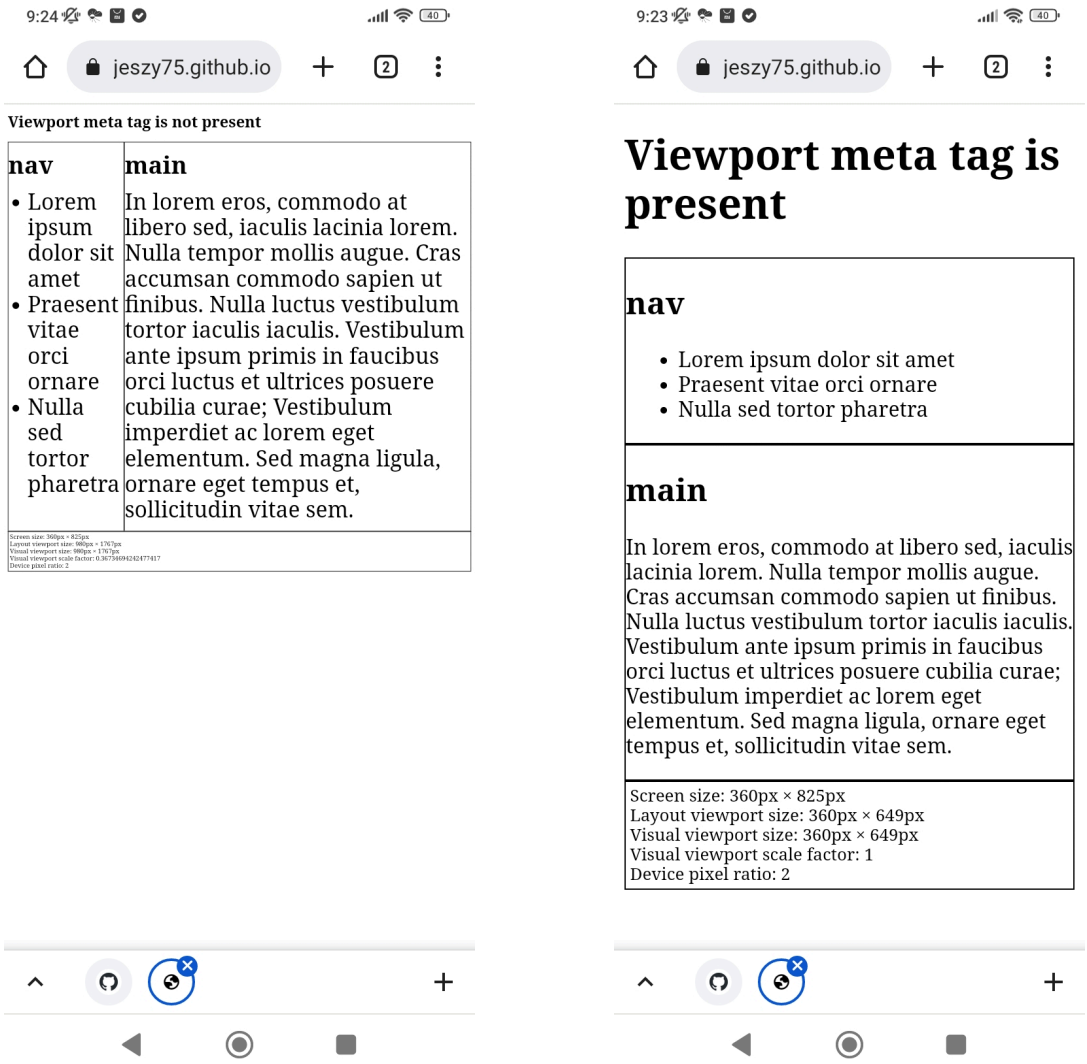


Figure 10: The effect of the absence/presence of the `viewport meta` tag

Viewport (11)

Further reading:

- MDN Web Docs – Viewport concepts
- MDN Web Docs – Viewport meta tag
- Peter-Paul Koch. A tale of two viewports – part one.
- Peter-Paul Koch. A tale of two viewports – part two.
- Peter-Paul Koch. A new Device Adaptation spec.
- Peter-Paul Koch. The Mobile Web Handbook. Smashing Magazine, 2014.
- Safari Web Content Guide – Configuring the Viewport

Fluid Grids

- RWD requires a flexible grid-based layout where the term “grid” is used in a very general sense.
  - Many existing CSS frameworks can be used to implement a grid.
  - Now, there exist also standard techniques (see the flexbox and the grid layouts).
- See:
  - Ethan Marcotte. Fluid Grids. March 03, 2009.

Flexbox Layout (1)

Relevant specification:

- CSS Flexible Box Layout Module Level 1 (W3C Candidate Recommendation, 19 November 2018)

Flexbox Layout (2)

- A CSS box model optimized for user interface design.
- In the flex layout model, the children of a flex container can be laid out in any direction, and can "flex" their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent.
- Both horizontal and vertical alignment of the children can be easily manipulated.
- Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.

Flexbox Layout (3)

- Support: https://caniuse.com/flexbox
- Further reading:
  - MDN Web Docs – CSS Flexible Box Layout
  - MDN Web Docs – Learn web development – Flexbox

Flexbox Layout (4)

- Playgrounds for playing with Flexbox:
  - https://flexbox.tech/
  - https://flexiting.com/playground/
  - https://flexbox-css.com/
- Examples:
  - https://jeszy75.github.io/responsive-layout-examples/
    https://github.com/jeszy75/responsive-layout-examples

Flexbox Layout (5)

- Fun fact: using the flexbox layout it is possible to implement responsiveness without using media queries at all.
- Consider, for example, the following style rules:

```
body {
  margin: 0;
}

.cards {
  display: flex;
  flex-wrap: wrap;
  gap: 1em;
  margin: 1em;
}
```

- The layout created by the style rules above is fully responsive thanks to the `flex-wrap: wrap` declaration.

Grid Layout (1)

Relevant specification:

- CSS Grid Layout Module Level 1 (W3C Candidate Recommendation Draft, 18 December 2020)

Grid Layout (2)

- Defines a two-dimensional grid-based layout system, optimized for user interface design.
- In the grid layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout grid.
- Grid layout allows to control the sizing and positioning of boxes and their contents.
- Fluid grid:
  - The `fr` unit specifies a flexible length, which represents a fraction of the leftover space in the grid container.
  - See: https://www.w3.org/TR/css-grid-1/#fr-unit

Grid Layout (3)

Unlike flexbox, which is single-axis-oriented, grid layout is optimized for 2-dimensional layouts: those in which alignment of content is desired in both dimensions.

Grid Layout (4)

- Support: https://caniuse.com/css-grid
- Further reading:
  - MDN Web Docs – CSS Grid Layout
  - MDN Web Docs – Learn web development – Grids

Grid Layout (5)

- Playgrounds for playing with grid layout:
  - CSS Grid Playground
  - Grid Garden
  - Interactive CSS Grid Generator
- Examples:
  - https://github.com/jeszy75/responsive-layout-examples
    https://jeszy75.github.io/responsive-layout-examples/

Grid Layout (6)

- Fun fact: using the grid layout it is possible to implement responsiveness without using media queries at all.
  - See, for example: Juan Diego Rodríguez. Beyond CSS Media Queries. May 16, 2024.
- Consider, for example, the following style rules:

```
body {
  margin: 0;
}

.cards {
  display: grid;
  gap: 1em;
  margin: 1em;
}
```

Grid Layout (7)

An example of a responsive grid with media queries:

```
@media screen and (min-width: calc(5in + 3em)) {
  .cards {
    grid-template-columns: repeat(2, 2.5in);
  }
}

@media screen and (min-width: calc(7.5in + 4em)) {
  .cards {
    grid-template-columns: repeat(3, 2.5in);
  }
}

@media screen and (min-width: calc(10in + 5em)) {
  .cards {
    grid-template-columns: repeat(4, 2.5in);
  }
}
```

Grid Layout (8)

The same layout without using any media query:

```
.cards {
  display: grid;
  gap: 1em;
  grid-template-columns: repeat(auto-fit, 2.5in);
  margin: 1em;
}
```

- This works in all major browsers, see: https://caniuse.com/mdn-css_properties_grid-template-rows_repeat

Multi-column Layout (1)

Relevant specification:

- CSS Multi-column Layout Module Level 1 (W3C Candidate Recommendation Snapshot, 16 May 2024)

Multi-column Layout (2)

A layout that lets authors declare that the content of an element is to be laid out in multiple columns with a gap and a rule between them.

Multi-column Layout (3)

- Support: https://caniuse.com/multicolumn
- Further reading:
  - MDN Web Docs – CSS multi-column layout

Multi-column Layout (4)

- Examples:
  - https://github.com/jeszy75/responsive-layout-examples
    https://jeszy75.github.io/responsive-layout-examples/

Flexible Replaced Elements (1)

Replaced element: an element whose content is outside the scope of the CSS formatting model, i.e., its contents is not affected by the current document's styles.

- Examples: `audio`, `iframe`, `img`, `video`
- See:
  - CSS 2.1 Specification – Conformance: Requirements and Recommendations
  - MDN Web Docs – Replaced elements

Flexible Replaced Elements (2)

- Stretch a replaced element to fill the containing element by setting the `width` property to 100%.

Flexible Replaced Elements (3)

- Prevent a replaced image from exploding out of its containing element by setting the `max-width` property to 100%.
  - Example:

```
img {
  max-width: 100%;
}
```

- When the `height` and `width` attributes are present on the `<img>` element the `height` property must also be set to `auto` to prevent distortion.
  - Example:

```
img {
  height: auto;
  max-width: 100%;
}
```

Responsive Images (1)

See:

- HTML Living Standard – Images
- MDN Web Docs – Responsive images
- MDN Web Docs – <img>: The Image Embed element

Responsive Images (2)

- There are many situations in which the author might wish to use multiple image resources that the user agent can choose from:
  - Different users might have different environmental characteristics, such as
    - physical screen size,
    - screen pixel density,
    - zoom level,
    - screen orientation,
    - network speed and bandwidth cost.
  - Authors might want to show the same image content but with different rendered size depending on, usually, the width of the viewport (viewport-based selection).
  - Authors might want to show different image content depending on the rendered size of the image (art direction).
- The above situations are not mutually exclusive.

Responsive Images (3)

The `srcset` and the `sizes` attributes:

- They can be specified on the `img` and the `source` elements.
- They are provided for specifying a set of alternative resources from which the user agent can choose the most appropriate one to use.

Responsive Images (4)

The `srcset` attribute:

- The value of the attribute is a comma-separated list of one or more strings (image candidate strings) each of which consists of the following whitespace-separated components:
  - A URI to an image.
  - Optionally, one of the following descriptors:
    - **Width descriptor**: a positive integer directly followed by `w`, it specifies the inherent width of the image in pixels.
    - **Pixel density descriptor**: a positive floating-point number followed by `x`.
  - If no descriptor is specified for an image candidate string, the default is `1x`.

Responsive Images (5)

The `sizes` attribute:

- The value of the attribute is a comma-separated list of one or more strings each of which consists of the following components:
  - A media condition that must be omitted for the last item in the list.
  - A source size value (i.e., a non-negative length) that specifies the intended display width of the image.

Responsive Images (6)

The `picture` element:

- Contains **zero or more** `source` elements, followed by **exactly one** `img` element to provide multiple versions of an image for different display/device scenarios.
  - The browser will consider each child source element and choose the best match among them; if no matches are found, the URI of the `img` element's `src` attribute is selected.
  - The selected image is then presented in the space occupied by the `img` element.
- See:
  - HTML Living Standard – The picture element
  - MDN Web Docs – <picture>: The Picture element

Responsive Images (7)

Examples:

https://jeszy75.github.io/responsive-image-examples/
https://github.com/jeszy75/responsive-image-examples

Responsive Images: device-pixel-ratio-based selection when the rendered size of the image is fixed (1)

The user agent can choose any of the given resources depending on the user's screen's pixel density, zoom level, and possibly other factors such as the user's network conditions.

Responsive Images: device-pixel-ratio-based selection when the rendered size of the image is fixed (2)

Example:

```
<style>
  img#cat {
    width: 320px;
  }
</style>

<img src="cat-320.jpg"
     srcset="cat-480.jpg 1.5x, cat-640.jpg 2x"
     alt="A cat napping"
     id="cat">
```

Responsive Images: viewport-based selection (1)

- Multiple images are provided that only vary in their size.
- The user agent will calculate the effective width of each image from the specified `w` descriptors and the specified rendered size in the `sizes` attribute.
- It can then choose any of the given resources depending on the user's screen's pixel density, zoom level, and possibly other factors such as the user's network conditions.

Responsive Images: viewport-based selection (2)

Example:

```
<img srcset="cat-320.jpg 320w,
             cat-640.jpg 640w,
             cat-800.jpg 800w,
             cat-1024.jpg 1024w"
     sizes="(max-width: 768px) 320px,
            (max-width: 992px) 640px
            (max-width: 1200px) 800px,
            1024px"
     src="cat-640.jpg"
     alt="A cat napping">
```

Responsive Images: art direction-based selection

- The `picture` element and the `source` element, together with the `media` attribute can be used to provide multiple images that vary the image content.
  - For example, a smaller image might be a cropped version of a bigger image.
- Example:

```
<picture>
  <source media="(min-width: 1200px)" srcset="cat-desktop.jpg">
  <source media="(min-width: 768px)" srcset="cat-tablet.jpg">
  <img src="cat-mobile.jpg" alt="A cat napping">
</picture>
```

  - The rendered size of the image varies depending on which resource is chosen.
  - To specify dimensions that the user agent can use before having downloaded the image, CSS can be used.

Responsive Images: image format-based selection

- The `type` attribute on the `source` element can be used to provide multiple images in different formats.
- Example:

```
<picture>
  <source srcset="cat.avif" type="image/avif">
  <source srcset="cat.webp" type="image/webp">
  <img src="cat.jpg" alt="A cat napping"
       width="640" height="480">
</picture>
```

Tools (1)

Web browser tools:

- Firefox: Responsive Design Mode
- Chromium, Google Chrome: Device Mode (Chrome DevTools)
- Chromium-based Edge: Device Emulation (Microsoft Edge DevTools)

Tools (2)

Kiwi: a mobile browser based on Chromium and WebKit that includes web developer tools

- Website: https://kiwibrowser.com/
- Repository: https://github.com/kiwibrowser/src
  https://github.com/kiwibrowser/src.next/
- Platform: Android
- License: New BSD License

Tools (3)

- Screen Size Map

Responsive Frameworks (1)

- Bootstrap (license: MIT License) https://getbootstrap.com/ https://github.com/twbs/bootstrap
- Bulma (license: MIT License) https://bulma.io/ https://github.com/jgthms/bulma
- Foundation (license: MIT License) https://get.foundation/ https://github.com/foundation/foundation-sites
- Pure (license: New BSD License) https://purecss.io/ https://github.com/pure-css/pure
- Tailwind CSS (license: MIT License) https://tailwindcss.com/ https://github.com/tailwindlabs/tailwindcss

Responsive Frameworks (2)

Market share and popularity:

- Usage statistics of CSS frameworks for websites (W3Techs)
- The State of CSS 2023 – CSS Frameworks
- CSS Framework Ranking (OSS Insight)

Bootstrap (1)

Documentation:

- Get started with Bootstrap (Official Docs)

Bootstrap (2)

- It supports the latest, stable releases of all major mobile and desktop browsers and platforms.
  - The list of supported browsers: `.browserslistrc`
- It follows a mobile-first approach, thus, the viewport `<meta>` tag should be added to the `<head>` element.
- It is written in Sass and requires Dart Sass for compiling `.scss` source files into `.css` files.
- It sets the value of the `box-sizing` CSS property to `border-box` to all element.
- It's responsive grid system is implemented with flexbox instead of CSS grid layout.

Bootstrap (3)

Installing locally with `npm` (needed for, e.g., customization):

```
npm install bootstrap@5.3.3
```

Bootstrap (4)

Tool Support:

- Visual Studio Code:
  - Bootstrap 5 Quick Snippets https://marketplace.visualstudio.com/items?itemName=AnbuselvanRocky.bootstrap5-vscode https://github.com/anburocky3/bootstrap5-snippets
  - Bootstrap IntelliSense https://marketplace.visualstudio.com/items?itemName=hossaini.bootstrap-intellisense https://github.com/aviiceena/bootstrap-intellisense

Bootstrap (5)

"Hello, World!" example from the documentation:

```
<!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Hello, Bootstrap!</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
      crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, Bootstrap!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
      crossorigin="anonymous"></script>
  </body>
</html>
```

Bootstrap (6)

Examples:

- https://bootstrap-container-example.surge.sh/
- https://bootstrap-grid-examples.surge.sh/