Media Queries & File Paths

Media Queries Explained

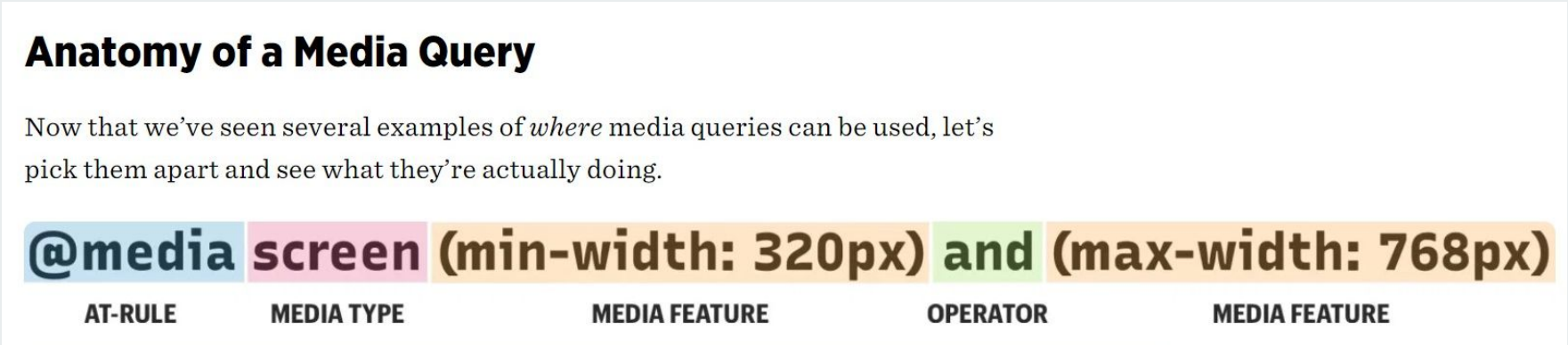
```
/* Example */

/* Mobile and Tablet */
@media (max-width: 768px) {
  html {
    font-size: 16px + 2 * ((100vw - 360px) / 768px);
  }
}

/* Laptop and Desktops screens */
@media (min-width: 769px) {
  html {
    font-size: 14px + 10 * ((100vw - 769px) / 2048px);
  }
}

/* Excessively large screens */
html {
  font-size: 36px;
}
```

Media Queries

```css
/* When the browser is at least 600px and above */
@media screen and (min-width: 600px) {
  .element {
    /* Apply some styles */
  }
}
```

“Media queries are a way to target browser by certain characteristics, features, and user preferences, then apply styles or run other code based on those things. Perhaps the most common media queries in the world are those that target particular viewport ranges and apply custom styles, which birthed the whole idea of responsive design.”

Media Queries - Breakdown



@media

▼ @media

```css
@media [media-type] ([media-feature]) {
  /* Styles! */
}
```

The first ingredient in a media query recipe is the `@media` rule itself, which is one of many CSS at-rules. Why does `@media` get all the attention? Because it's geared to the *type* of media that a site is viewed with, what *features* that media type supports, and *operators* that can be combined to mix and match simple and complex conditions alike.

Media Type

▼ Media types

```
@media screen {
  /* Styles! */
}
```

What type of media are we trying to target? In many (if not most) cases, you'll see a `screen` value used here, which makes sense since many of the media types we're trying to match are devices with screens attached to them.

But screens aren't the only type of media we can target, of course. We have a few, including:

- `all` : Matches all devices
- `print` : Matches documents that are viewed in a print preview or any media that breaks the content up into pages intended to print.
- `screen` : Matches devices with a screen
- `speech` : Matches devices that read the content audibly, such as a screenreader. This replaces the now deprecated `aural` type since Media Queries Level 4.

Media Features

▼ Media features

Once we define the type of media we're trying to match, we can start defining what features we are trying to match it to. We've looked at a lot of examples that match screens to width, where `screen` is the *type* and both `min-width` and `max-width` are *features* with specific values.

But there are many, many (many!) more "features" we can match. Media Queries Level 4 groups 18 media features into 5 categories.

Operators

`and`

But we can use the `and` operator if we want to target screens within a range of widths:

```css
/* Matches screen between 320px AND 768px */
@media screen (min-width: 320px) and (max-width: 768px) {
  .element {
    /* Styles! */
  }
}
```

`or` (or comma-separated)

We can also comma-separate features as a way of using an `or` operator to match different ones:

```css
/*
  Matches screens where either the user prefers dark mode or the screen is at least 1200px wide
@media screen (prefers-color-scheme: dark), (min-width 1200px) {
  .element {
    /* Styles! */
  }
}
```

`not`

Perhaps we want to target devices by what they do **not** support or match. This declaration removes the body's background color when the device is a printer and can only show one color.

```css
@media print and ( not(color) ) {
  body {
    background-color: none;
  }
}
```

Some Media Queries

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

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

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

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

Other Examples

Orientation

One well-supported media feature is `orientation`, which allows us to test for portrait or landscape mode. To change the body text color if the device is in landscape orientation, use the following media query.

```
@media (orientation: landscape) {
    body {
        color: rebeccapurple;
    }
}
```

Open this example in the browser, or view the source.

A standard desktop view has a landscape orientation, and a design that works well in this orientation may not work as well when viewed on a phone or tablet in portrait mode. Testing for orientation can help you to create a layout which is optimised for devices in portrait mode.

Other Examples

"and" logic in media queries

To combine media features you can use `and` in much the same way as we have used `and` above to combine a media type and feature. For example, we might want to test for a `min-width` and `orientation`. The body text will only be blue if the viewport is at least 600 pixels wide and the device is in landscape mode.

```
@media screen and (min-width: 600px) and (orientation: landscape) {
    body {
        color: blue;
    }
}
```

Desktop vs Mobile Approaches

A mobile-first approach to styling means that styles are applied first to mobile devices. Advanced styles and other overrides for larger screens are then added into the stylesheet via media queries.

This approach uses `min-width` media queries.

Here's a quick example:

```
// This applies from 0px to 600px
body {
  background: red;
}

// This applies from 600px onwards
@media (min-width: 600px) {
  body {
    background: green;
  }
}
```

Desktop vs Mobile Con't - Desktop

On the flipside, a desktop-first approach to styling means that styles are applied first to desktop devices. Advanced styles and overrides for smaller screens are then added into the stylesheet via media queries.

This approach uses `max-width` media queries.

Here's a quick example:

```
body {
  background: green;
}

// This applies from 0px to 600px
@media (max-width: 600px) {
  body {
    background: red;
  }
}
```

`<body>` will have a background colour of green for all widths. If the screen goes below 600px, the background colour becomes red instead.

File Paths

- `/` is the root of the current drive;
- `./` is the current directory;
- `../` is the parent of the current directory.

`src="/image.jpg"` == the root folder of the website

`src="image.jpg"` == same folder as current page

`src="../image.jpg"` == one folder up from current page

`src="../../image.jpg"` == two folders up from current page, etc

References:

https://css-tricks.com/a-complete-guide-to-css-media-queries/

https://www.w3schools.com/howto/howto_css_media_query_breakpoints.asp

https://zellwk.com/blog/how-to-write-mobile-first-css/

https://css-tricks.com/quick-reminder-about-file-paths/