CSS Responsive

Learning Objectives

- [ ] Understanding the concept of responsive design
- [ ] Understanding why we work mobile first (in design and CSS)
- [ ] Understanding responsive units
- [ ] Understanding media queries

Mobile First Design

When authoring CSS, it's a very helpful convention to first define all your mobile styles, and then add media queries to adjust the styles for larger screens. It's easier to reason about, and it results in a simpler CSS structure. Your code will be more similar to the way other people write CSS.

When designing, it makes sense to begin designing mobile first, because most users are on mobile devices. This also helps you to focus on the most important information (you have less space), and to structure it in a way that makes sense for mobile users.

Responsive Units

Responsive units are units that are relative to the size of the viewport, the current font size, or the size of their parent element.

- `vh` (viewport height): 1vh is 1% of the viewport height
- `vw` (viewport width): 1vw is 1% of the viewport width
- `em`: 1em is the font size of the current element
- `rem`: 1rem is the font size of the root element
- `%`: is relative to the related property of the parent or current element - every property has it's own rules what it is relative to:
  - `width: 1%` is set relative to the parents element width
  - `padding-top: 10%` means 10% of the parents height
  - `font-size: 50%` means half as big as the parent font-size
  - `transform: translateX(50%)` means translate on the x axis by 50% of the current elements width
- `calc()`: allows you to combine multiple units and do math
  - e.g. `calc(100vh - 100px)`
  - or `calc(50% - 10rem)`

Media Queries

Media queries allow you to write CSS for specific media types, screen sizes, orientations and more.

They follow the following syntax:

```
@media (media feature) {
  /* CSS rules */
}
```

Media Types

You can target a specific media type with the `screen` and `print` media types.

```
@media screen {
  /* CSS rules that are only applied on screens */
}

@media print {
  /* CSS rules that are only applied when printing */
}
```

Screen Size

You can target specific screen sizes with the `min-width` and `max-width` media features.

```
@media (min-width: 600px) {
  /* CSS rules that are only applied when the screen is at least 600px
wide */
}

@media (max-width: 600px) {
  /* CSS rules that are only applied when the screen is at most 600px wide
*/
}
```

> 💡 Avoid `max-width` media queries, because they are harder to reason about. It's easier to think about the smallest screen size first, and then add media queries for larger screens.

Orientation

You can target specific orientations with the `orientation` media feature.

```
@media (orientation: portrait) {
  /* CSS rules that are only applied when the screen is in portrait
orientation */
}

@media (orientation: landscape) {
  /* CSS rules that are only applied when the screen is in landscape
orientation */
}
```

> 💡 You can also target a specific aspect ratio with the `aspect-ratio` media feature.

Pointer

You can target specific pointer types with the `any-pointer` media feature.

```
@media (any-pointer: none) {
  /*
        CSS rules that are only applied when the device has no pointer
        (neither touch nor cursor)
    */
}

@media (any-pointer: coarse) {
  /*
        CSS rules that are only applied when the device has a coarse
pointer
        (mostly touch)
    */
}

@media (any-pointer: fine) {
  /*
        CSS rules that are only applied when the device has a fine pointer
        (cursor)
    */
}
```

Device Pixel Ratio (Pixel Density)

You can target specific device pixel ratios with the `device-pixel-ratio` media feature.

```
@media (device-pixel-ratio: 1) {
  /*
        CSS rules that are only applied when the device has a pixel ratio
of 1
        (mostly older screens)
    */
}

@media (device-pixel-ratio: 2) {
  /*
        CSS rules that are only applied when the device has a pixel ratio
of 2
        (newer screens like the retina screen on your MacBook)
    */
}

@media (device-pixel-ratio: 3) {
```

```
  /*
        CSS rules that are only applied when the device has a pixel ratio
of 3
        (some high resolution tablets and phones)
    */
}
```

Color Scheme

You can target specific color schemes with the `prefers-color-scheme` media feature.

```
@media (prefers-color-scheme: dark) {
  /* CSS rules that are only applied when the user prefers a dark color
scheme */
}

@media (prefers-color-scheme: light) {
  /* CSS rules that are only applied when the user prefers a light color
scheme */
}
```

> 💡 You can change your preferred color scheme in your operating system settings. On macOS, you can do this in System Preferences > General > Appearance.

Reduced Motion

You can target users who are sensitive to animations and movement (and set up their system accordingly) with the `prefers-reduced-motion` media feature.

```
@media (prefers-reduced-motion: reduce) {
  /* CSS rules that are only applied when the user prefers reduced motion
*/
}
```

> 💡 You can change your preferred reduced motion setting in your operating system settings. On macOS, you can do this in System Preferences > Accessibility > Display > Reduce motion.

High Contrast

You can target users who prefer a higher contrast with the `prefers-contrast` media feature.

```
@media (prefers-contrast: more) {
  /* CSS rules that are only applied when the user prefers a higher
contrast */
}
```

> 💡 You can change your preferred contrast setting in your operating system settings. On macOS, you can do this in System Preferences > Accessibility > Display > Increase contrast.

Other Media Features

There are also media features for other (accessibility) features, like `inverted-colors`. The list of all media features is always growing. Check out the full list on mdn.

Combining Media Features

You can combine multiple media features with `and`.

```
@media (min-width: 600px) and (orientation: landscape) {
  /* CSS rules that are only applied when the screen is at least 600px
wide and in landscape orientation */
}
```

Testing for multiple Media Features

You can use a comma-separated list to apply styles when the user's device matches any one of various media types using `,`

```
@media (min-width: 600px), (orientation: portrait) {
  /* CSS rules that are only applied when the screen is either at least
680px high or in portrait orientation */
}
```

Simulating Media Features

You can simulate media features in the browser devtools. For example, you can change your screen size in the devtools by clicking the device icon in the top left corner of the devtools.

Common Breakpoints

When using `min-width` media queries it can be helpful to use common breakpoints.

- no media query (default): extra-small, xs, mobile
- `(min-width: 600px)`: small, sm, large mobile
- `(min-width: 900px)`: medium, md, tablet
- `(min-width: 1200px)`: large, lg, desktop
- `(min-width: 1536px)`: extra-large, xl, large desktop

> 💡 These breakpoints are based on the MUI breakpoints. Other frameworks and projects might define a completely different set of breakpoints. Mostly they are defined to be between the most common screen sizes. Another example for common breakpoints are the ones from Tailwind CSS.

🪄 **Pro Tip**: You can use media queries to redefine the values of CSS custom properties. This way you can use the property as a value that dynamically changes based on the media query.

```
:root {
  --font-size: 12px;
}

@media (min-width: 600px) {
  :root {
    --font-size: 16px;
  }
}

@media (min-width: 1200px) {
  :root {
    --font-size: 20px;
  }
}

body {
  font-size: var(--font-size);
}
```

Showing different images based on media queries

You can use media queries to show different images based on the screen size.

The html `picture` element allows you to define multiple `source` elements for an image. The browser will choose the *first* source that matches the given media query. If no `source` element matches, the browser will use the `img` element as a fallback.

The `img` element is required, so that there is always a fallback.

```
<picture>
  <source
    media="(min-width: 1200px)"
    srcset="https://source.unsplash.com/random/1400x1050"
  />
  <source
    media="(min-width: 900px)"
    srcset="https://source.unsplash.com/random/800x600"
  />
  <img src="https://source.unsplash.com/random/400x300" alt="" />
</picture>
```

> 💡 Note that the `source` element doesn't have a `src` attribute but uses the `srcset` attribute instead. This can be used to help with performance but is not relevant for this session.

Resources

- Using Media Queries
- Responsive Design on mdn
- Responsive Web Design Basics on web.dev
- Values and Units on mdn
- Beginner's Guide to Media Queries on mdn
- @media on mdn
- The Picture element on mdn
- Is your web page mobile-friendly? on Google