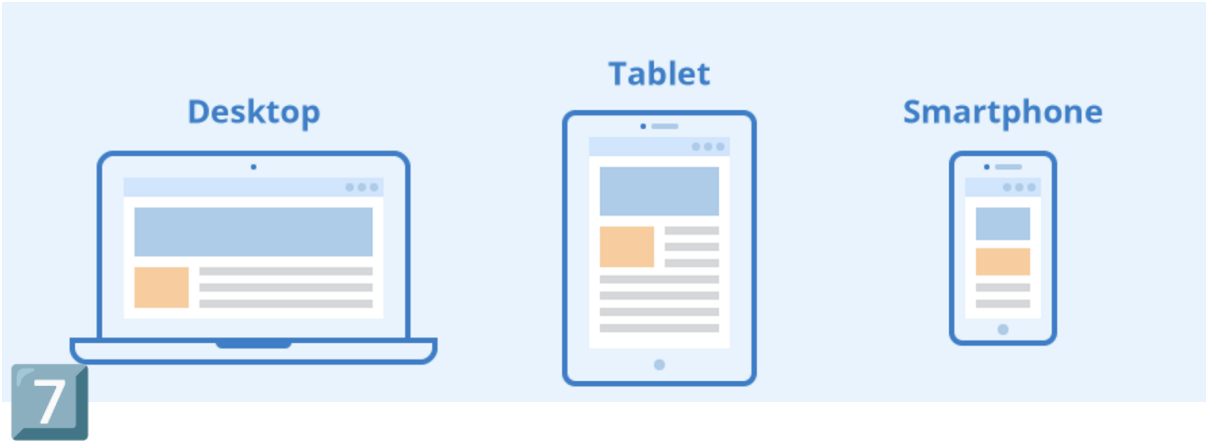


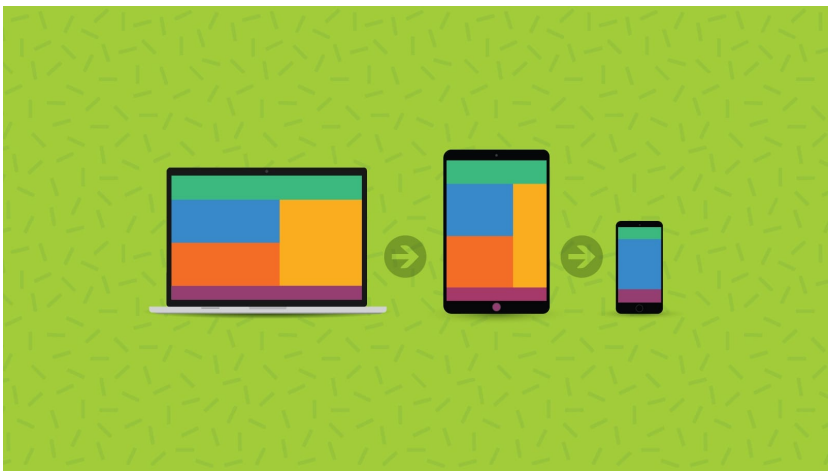
Day-7: Media Queries & Specificity

Introduction:

- Media queries are what make modern responsive design possible. With them you can set different styling based on things like a users screen size, device capabilities or user preferences.

What are media queries?

- What is a media query? A media query is a specific feature of CSS that lets you conditionally apply styling based on a media type, a media feature or both.
- You use them primarily to check the screen dimensions and apply CSS based on that, but media queries can do many other powerful things.



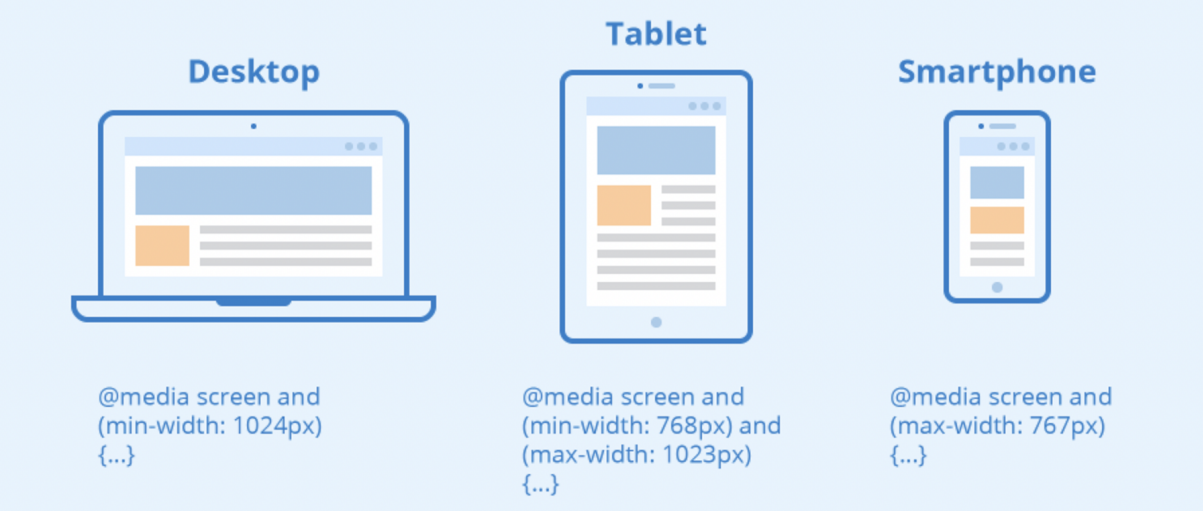
- A media query is an expression that evaluates to either True or False.
- A media query consists of a test, followed by as many CSS rules as we want, with the rule block enclosed in a set of braces. If the test condition is false, the browser simply ignores the rule block and moves on.

Syntax of Media Query

```
@media screen (min-width: 320px) and (max-width: 768px)
```

AT-RULE | MEDIA TYPE | MEDIA FEATURE | OPERATOR | MEDIA FEATURE

- `@media` specifies media queries declaration.
- `Media type` specifies what type of media are we trying to target, In many 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. It also includes
  - `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
- `Media Feature` defines what features we are trying to match it to.
  - Desktop: (`min-width:1024px`)
  - Tablet: `(min-width:768px)` and `(max-width:1023px)`
  - Smartphone : `(min-width:340px)` and `(max-width:767px)`



```css
/* large screens   */
      @media all and (min-width: 1024px) {
        /* write code here */
      }

/* medium screens   */
      @media all and (min-width: 768px) and (max-width: 1024px) {
        /* write code here */
      }

/* small screens */
      @media all and (min-width: 320px) and (max-width: 767px) {
        /* write code here */
      }
```

- Let's see how media query works by taking small example.
- In this example, we are just giving background color property to body, by default whatever we write is for large screens.

```
<html>
  <head>
    <title>Document</title>
    <style>
      body {
        background-color: teal;
      }
    </style>
  </head>
  <body></body>
</html>
```

- `Medium Screens`

```
<html>
  <head>
    <title>Document</title>
    <style>
      body {
        background-color: teal;
      }
@media all and (min-width: 768px) and (max-width: 1024px) {
        body {
          background-color: pink;
        }
      }
    </style>
  </head>
  <body></body>
</html>
```

- `Large Screens:`

```
<html>
  <head>
    <title>Document</title>
    <style>
      body {
        background-color: teal;
      }
      /* meidum screens */
      @media all and (min-width: 768px) and (max-width: 1024px) {
        body {
          background-color: pink;
        }
      }

      /* small screens */
      @media all and (min-width: 320px) and (max-width: 767px) {
        body {
          background-color: brown;
        }
      }

    </style>
  </head>
  <body></body>
</html>
```

Output Video for all screens

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/800546f4-3503-406f-8395-94f2fe8e2db6/mq.mp4

- In the above example we are just changing background color, we can also change whatever property we want for particular screen size.

Specificity:

- In Selectors lecture we have seen scores of various selectors

Scores of various selectors

Selector Name	Score
Inline	1000
Id	100
Class	10
Tag	1
Universal	0

- We have also learned few more selectors like
  - Attribute Selector
  - Pseudo Class
    - :hover
    - :nth-child
    - :focus
  - Pseudo Element
    - ::first-line
    - ::after

- What about scores of above selectors?

Scores of various selectors

Selector Name	Score
Inline	1000
Id	100
Class,attribute,pseudo classes	10
Tag, pseudo element	1
Universal,combinators (+, > , ~,)	0

What is Specificity?

- If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.
- Think of specificity as a score/rank that determines which style declaration are ultimately applied to an element.
- Let's take an example

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
    <style>
      /* Specificity score - 100 points */
      #masai {
        background-color: teal;
      }
      /* specificity score - 1 point */
      h1 {
        background-color: red;
      }
    </style>
  </head>
  <body>
    <div>
      <h1 id="masai">Masai school</h1>
    </div>
  </body>
</html>
```

- In the above example id has more specificity score compared to tag, so background of `teal` will be applied.
- Now, what if I use combinators to style `h1` tag

```
<style>
      /* Specificity score - 100 points */
      #masai {
```

```
      background-color: teal;
    }

    /* specificity score - 1 point */
    h1 {
      background-color: red;
    }

    /* specificity score - 1+1 = 2 points */
    div > h1 {
      background-color: yellow;
    }

    /* specificity score - 1+100 = 101 points */
    div > #masai {
      background-color: pink;
    }
  </style>
```

- Now since `div>#masai` has highest specificiy score of 101, `pink` will be applied

The table below shows some examples on how to calculate specificity values:

Selector	inline Style (1000)	#id (100)	.class (10)	tag (1)	**Calculation**	Specificity Value
ul.class	0	0	10	1	0+0+10+1	11
h1 + p	0	0	0	1+1	0+0+0+2	2
h1 ~ p::first-letter	0	0	0	1+1+1	0+0+0+3	3
.heading p.main	0	0	10+10	1	0+0+20+1	21
#heading p ul li .disabled li	0	100	10	1+1+1+1	100+10+4	114
.heading p.main ul li.disabled li	0	0	10+10+10	1+1+1+1	0+0+30+4	34
#navbar p#demo	0	100+100	0	1	0+200+1	201
<p style="color: pink;">	1000	0	0	0	1000	1000