Cascading Style Sheets

Key Notes

- content HTML = House structure
- CSS content look better more presentable = House design
- color, position, effect, font size

CSS Rules/Rule section Syntax

- selectors are used to target element
- declaration of key value pair

```
div {   /* selector*/
  font-size:20px; /* declaration*/
  margin:20px; /* declaration*/
}
p{ /* selector*/
  padding:30px /* declaration*/
}
```

adding external CSS to html

- in head or anywhere

```
<style>
  h1 {
    color:red;
  }
  p {
    color:blue;
  }
</style>
```

- not the best approach how cn we fix this?
- link it to html page `<link rel="stylesheet href="style.css">`

some styles

```
html {
  padding:50px;
  margin:"auto";
  scrollbar-width: "none";
  background-color: rgb(252, 241, 224);
  font-family: 'Roboto', sans-serif;
  text-align: left;
  letter-spacing: 0.25px;
}

h1 {
  color:saddlebrown;
  text-decoration: underline;
  font-family: 'Radio Canada', sans-serif;
  text-align: center;
}
```

some styles to add - 2

```
ul, ol {
  color:darkgoldenrod;
  /* border-width:2px;
  border-style: dashed;
  border-color:cadetblue */
  border: 4px solid darkblue;
  /* border-bottom:8px dotted red;
  border-left:3px dashed darksalmon */
}
li{
/* the disk in li tag */
list-style-type: none;
text-shadow: 2px 2px white;
/* 3 values right bottom colors */
}
p {
  column-count: 2;
  column-gap: 20px;
```

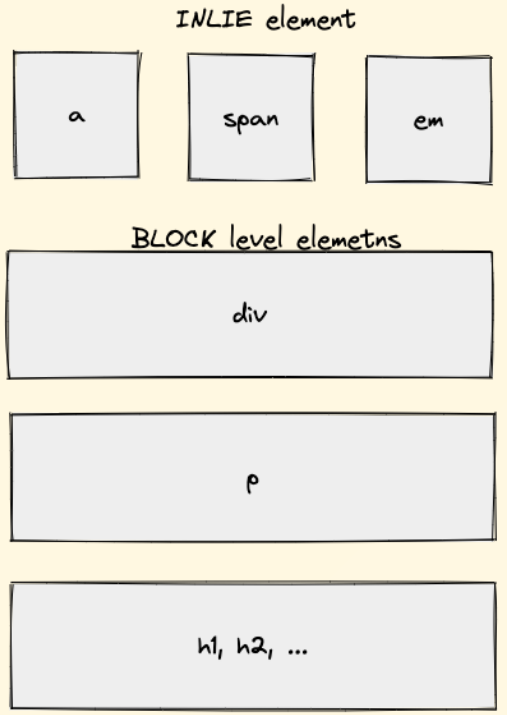
custom colors

RGB-hex code

- red blue green
- 0 darkest f whites
- #440000 would be maroon i.e dark red
- #EE0000 would be red i.e light red
- YOU WILL NAVER MAKE YOUR OWN HEX COLOR
- use vscode :P

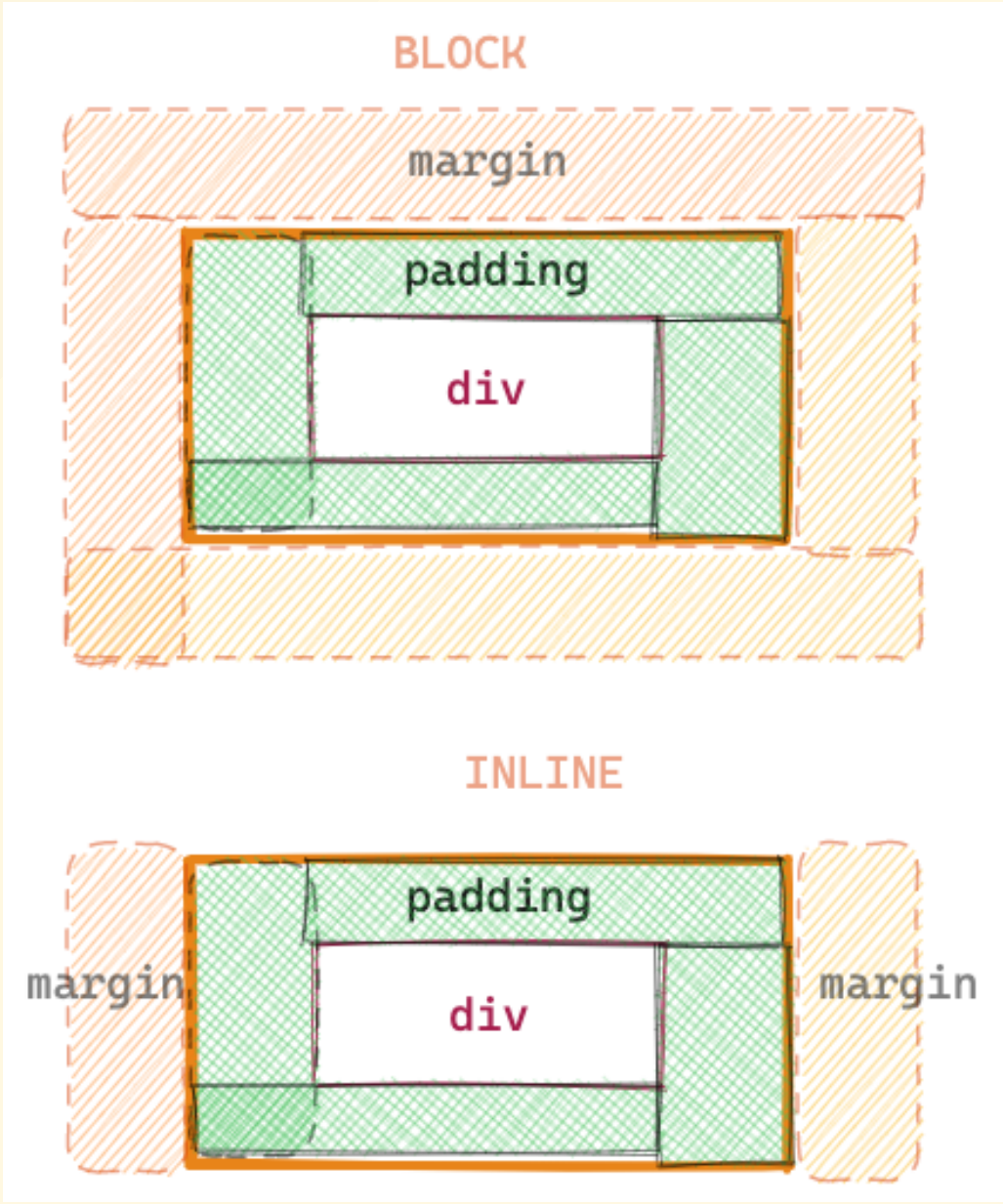
inline/block html element

- line up next to each other
- take up whole width
- block-> we want a link without a new line sdad <a>link
- change inline to block, block to inline using css :)



Box-Model

- How element reprrenst themslevs in tervs of space
- we can control this spacing
  - margin
  - padding
  - border



css basic summary

- what is it how we added it
- basic selectors
- hex code and vscde colors
- inline and block element
- margin and padding
- default styles by browses

Classes and Sector

class selection

- select a class with `.` to style

```
<div class="coolClassName">some text</div>
<style>
  .coolClassName {
    color:red;
  }
</style>
```

- only get specific element with the class
- we do using tag.coolClassName `p.coolClassName`

multiple classes

- style element with multiple class name using `div.coolClassName.coolClassName2`

```
<div class="coolClassName coolClassName2">
```

id selection

- we can used classes or ids
- more commonly used in JS
- target the content using `#`
- can be only used `ONLY ONCE, HAS TO BE UNIQUE`

```
<div id="coolId"> some text </div>
<style>
  #coolId {
    color:red;
  }
</style>
```

- `parent child {declaration}`
- this is specific targeting
- if we want to target a specific class in an element b

```
<div>
  <div class="child">
    <p>some text</p>
  </div>
</div>
  <p class="child">
    <span>some text</span>
  </p>
  <style>
    div .child {
      color:red;

    }
  </style>
```

- selecting with and without space

attribute selectors

We want to select links ith href Attribute

```
a [href] {color:darksalmon}
```

or with some values

```
a[href="https://www.google.com"] {color:darksalmon}
```

> ***Tips:***
>
> - You can edit default link underline property with `text-decoration: none`
> - insted of looking up whole link you can serch akeyword `a[href*="google"] {color:darksalmon}`
> - or ends in with `a[href$=".pdf"] {color:darksalmon}`

The CASCADE

- go down like waterfall
- html children can inherit parent styles
- p inherits div style but not all, we can force inherit them
- last one has precedence over the previous one unless its not specifically Set

```
<div>
  <p>
    <span>some text</span>
  </p>
</div>
<style>
  div {
    color:red;
    border:2px solid red;
    font-style: italic;
    margin:20px;
    font-size:20px;
  }
 p {
   border:inherit;
   margin:inherit;
 }
</style>
```

Layout and position

- Static : default position
- Relative: can position relative to other elements or its original position
  - we do this position child absolute to the parent
- Absolute: positioned within relative loses original space
- Fixed: position relative to the viewport always fixed
- Sticky: position relative to the viewport always fixed and can be fixed to top or bottom based on scroll

```
<section class="banner">
  <img src="img/banner.jpg" alt="banner image">
<div class="welcome">
  <h2>Static page to Learn/Teach <br><span>HTML & CSS</span></h2>
</div>
</section>
<style>
.banner {
position: relative;
}
.banner .welcome {
  position: absolute;
}
</style>
```

The nav bar

- `position:fixed` with `top:0px` and `left:0px`
  - it losses its width, set `width:100%`
- the `z-index` everything has a z-index of 0

```
header {
  background-color:darksalmon;
  padding: 2px 15px;
  width: 100%;
  top:0;
  left:0;
  position:fixed;
  z-index: 1;
}
```

position sticky

```
  nav{
  background-color: rgba(63, 61, 122, 0.891);
  position: sticky;
  padding: 10px;
  width: 100%;
  top:80px
}
```

other css declaration

1. `box-sizing: border-box` : incorporate to total width not add to width
2. `line-height:2em` : takes the parent hight p=16px give it 2em therefore 32px twice height

pseudo classes

1. Target elements when they are in a particular state
    1. `hover` when mouse is on top
    2. `focus` when an item is selected
    3. `valid` built in validation styling (when its valid)
    4. `active` when the element is active
    5. `first-child` first child of the element
2. many sudo classes w3schools

pseudo elements

1. Inject dynamic content in the element
2. smiler uses `::` instead of `:`
    1. first-line
    2. first-letter

media queries

- make design responsive (looks good on all devices)
- tell the browser how to style based on viewport dimensions
  - `media queries` style things differently based on screen size
  - responsive design load small size images for small devices
- MOBILE FIRST approach
  - FITS LESS; fit imp Content
  - start less than add more for large webpage

viewport meta tag

- goes in head tell the browser something about the website size
- What are we saying here

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

  - meta tag: info about website
  - width=devive-width: the width of the device
  - initial-scale=1.0: the initial scale of the page
- `viewport` meta tag is used to control the size of the browser window

media queries

- different sizs on different devices
- at a point i want to make my text small, image row and so on

```
@media screen and (max-width:700px){

  .welcome h2 {
    font-size: 3rem;
    color:wheat
  }
  .welcome h2 span{
    font-size: 1.5em;
  }
}
```

CSS flex-box

- we use different position props
- many fixes and hacky ways to position Layout
- container -> display:flex->i.e become flexible

flexible container basics

- create a flex container using just `display:flex`
- we can now control how they grow, shrink, add sapce between them and so on
- default left to right

flexible growth

- grow into any available room
- like columns in excel we give it a value to take up this many columns
- `flex-grow:2` grow at this rate.

flexible shrink

- `flex-shrink:1` will shirnk content based on browser sizing
  - the bigger the number i.e shrink rate the more it shrinks
  - we shrink elements in relation to one another

flex-wrap

- when we shrink the page by defaults it shrink each elements upon reaching a point it can no longer shrink the elements it goes off horizontally of the page
- when we reach min-width we want the elements to wrap around to the next line
- we do that by adding `flex-wrap: wrap` to the container
- if we have flex grow enabled the next line will grow accordingly
  - we can also do `flex: wrap-reverse` to reverse the wrap on top

flex-basis

- similar to minimum width; it sets the width of the element
- set the starting size of the elements
  - `flex-basis`
  - minimum width will cause the scrolling behavior to be unpredictable where as flex-basis will just shrink the elements on reaching the minimum width

flex short hand and justify and space props

- we did `flex-grow`, `flex-shirk`, `flex-basis`
  - we can just use `flex:1 0 200px`
- justify-content: space-between; center; space-around; space-evenly; start; end;