Visual Studio Code Tip: If you type tag label ex. 'h' for <h> and press tab, it will autocomplete tag for you.

<blockquote cite="im">...</blockquote>

- Used to quote from another source or website
- Only indents slightly from edge of page

Can place comments for developers as follows:

<!-- ... -->

Can also comment out code (Select code and press ctrl + /)

HTML Web Forms

- Made of various Input fields

```
<form action = "">...</form>
```

action has to do w/ PHP and server-side data

```
<input type = "text"
       id = "user">
```

can use id as a label

```
<label for = "user">...</label>
```

Another type is email as well as password

```
<label for = "email">...</label>
<input type = "email" id = "email">
       required
```

(makes field required)

- Another attribute that can be added to these input fields is the name attribute

Used for server side processing where a PHP file on server could use the name to retrieve data from input field

- Usually id and name can be the same

- Names can also be used to group radio buttons like:

```
<input type="radio" name="gender
    value="Male"
        or
    "Female"
        or
    "Other" >
```

- Male
- Female
- Other

Meant to help server see which one is submitted

Note: Can only choose one button among inputs w/ same name

```
<input type="radio" name="age" value="0-25" id="option-1">
<label for="option-1">0-25</label>
```

This code will show


0-25

Another input field is a Select box is a drop down type of form

- When selecting an option from the drop down, the select box will then have the value corresponding to that selection

- Like radio in functionality, can only select one at a time

Can be made by using label and select tag:
- Nested option tags

```
<label for="question">Security Question:</label>
<select name="question" id="question">
    <option value="q1">Favorite color?</option>
    <option value="q2">Mother's Maiden name?</option>
</select>
```

Last form field is the text area AKA comments/biographies used when normal text input is not enough

- Actually NOT an input
- Uses textarea tag

```
<label for="bio">Your Bio:</label>
<br>
<textarea name="bio" cols="30"
  id="bio" rows="10"
  placeholder="..." </textarea>
```

For new line

Can be used in other input fields

Default text in the field that'll disappear

VS code tip, if you use alt + *click* you can type in multiple places at once

---

Also to note is the submit button

```
<input type="submit"
       value="Submit Form"
```

↳ text in button

When you submit a form, browser takes data and sends it to the reference of the action attribute

```
<form action="~">
```

↳ Backend script (PHP)

CSS Basics

In a css sheet you have selectors and declarations

selector

```
div {
    color: red;
    margin: 20px;
}
```

declarations

In CSS you can work w/ different units of measurement

```
font-size: 20px;
```

most common

Some mentioned decorations are:

color
background-color
font-size
text-decoration: line-through
underline
none ;

font-family: arial;
↳ need to use web safe font

text-align: left
right
center ;

line-height: 20px;
↳ vertical spacing btw lines

letter-spacing: 1px;

Can split text into columns

column-count: 3;

column-gap: 60px;

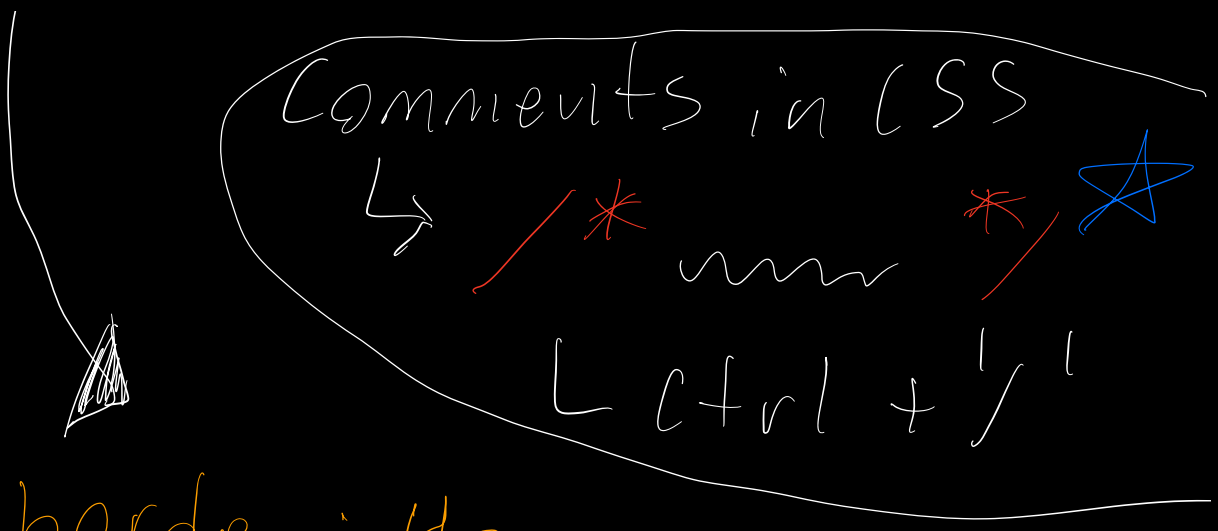
border-width: 4px;

border-style: solid;

↳ Need style for width

border-color: crimson;

↳ Can shorthand this




```
border: 4px solid crimson;
```
OR
```
border-bottom: "_____";
border-left: 8px dashed crimson;
border-top: 6px dotted crimson;
```
for more customization

For <li> tags inside

```
list-style-type: disc / square / none;
text-shadow: 2px 2px lightgray;
```
↳ adds depth to text

In order to get a color that not covered by a keyword ex/ "crimson" we use hex codes

Hex codes are made of 3 channels RGB

ex/ overall color

42 f4 f1

Red Green Blue

channel channel Channel

For each Channel, 0 is the darkest shade and f is the most vivid Shade

Types of HTML Elements

Inline vs. Block Elements

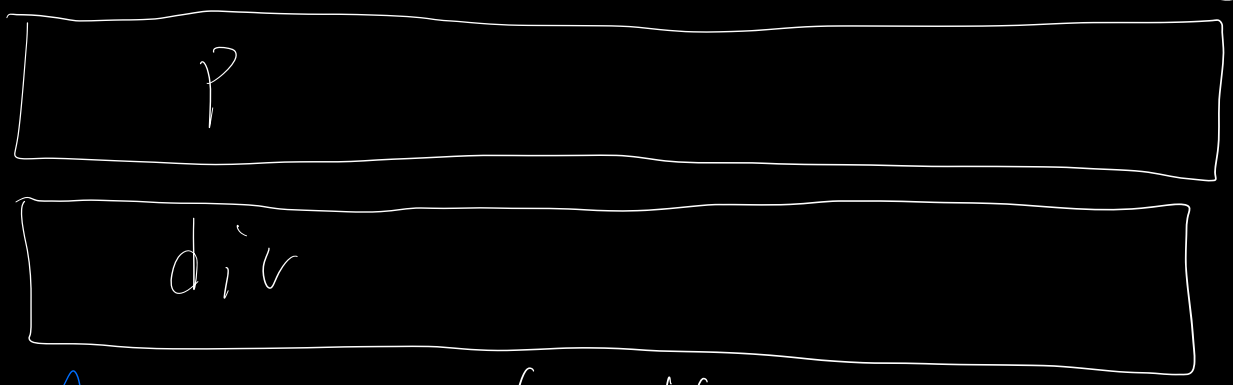
- Difference is how much room they take up and how they're displayed

Inline elems line up next to each other and don't take up more room than their content needs

Ex/ span, img, strong, em, a tags and more

Block elem take up the whole width of a page regardless of content

Ex/ p, div, h1, h2, h3, ul, li and more



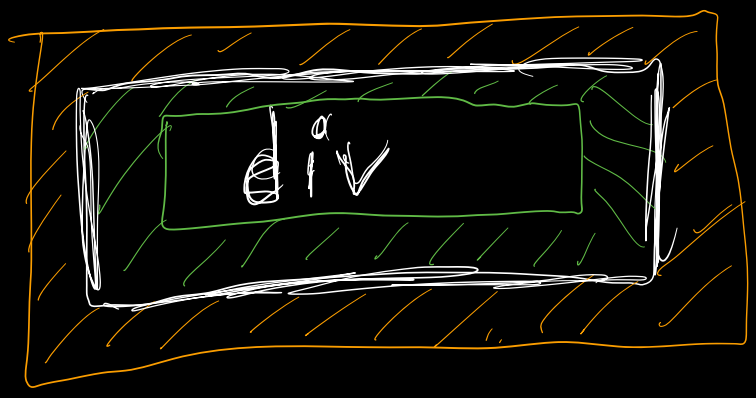
Can actually change inline elem to block using CSS and vice versa

ex/

```
span {
  display: block;
}
```

```
div {
  display: inline;
}
```

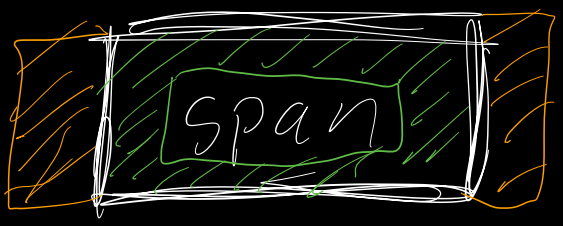
Margin and Padding



/// - Margin

/// - Padding

No problems when margin and padding added to block element, not so much w/ inline



As shown, padding is the same, but margin now only affects the sides (doesn't have margin-top margin-bottom)

☆ Depending on Screen width, Inline padding can collapse on one another if two tags are placed one ontop of the other

☆ Same exact concept applies to Block margins

If one div has a larger margin than the other, the larger pixel count will act as the margin between them if stacked vertically

Best of Both world's is an element called Inline Block

Inline Block elements still sit next to other elements on the page, but it will be treated as a block in terms of margin and padding

---

Default Browser Styles

- Headings are by default big and bold
- Anchor tags underlined in blue/Purple
- Paragraph tags have margin applied to them

etc...

All default browser styles applied to different elements so we can tell what they are

Even without CSS, these styles help us recognize what is on the page

User Agent Stylesheet - Browser's Default styles

↳ can be overwritten

CSS Classes And Selectors

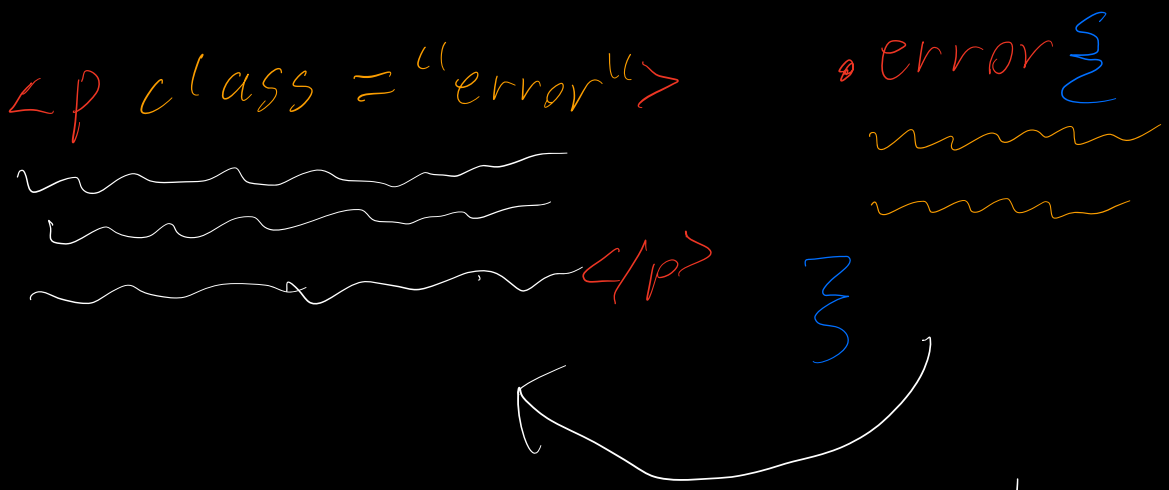
Problem w/ selectors are that it targets every single tag corresponding to it and not a specific one

ex/ p {
3

Classes solve this problem:

HTML

CSS



only changes tags w/ that class attribute

* Doesn't have to be the same kind of tag, ex/ above class name "error" can be given to a div tag

If you did want to target only p tags w/ the class error when other tags have that class, you can do:

```
p.error{
}
```

which will only affect paragraph tags

You can give a tag multiple class attributes

ex/ `<p class="success feedback"></p>`

This has two classes, "success" and "feedback" but in the same quote separated by a space

If you wanted to style all p tags w/ both classes you can do the following:

```
p.success.feedback {
}
```

Essentially chaining them together

⭐ Important to note:
p.success declarations will affect a tag as long as at least one of it's tags is success regardless of the other tags

⭐ The id attribute can also be used in CSS as hooks but commonly more used for javascript

ID's in CSS example//

HTML

```
<div id="content">
</div>
```

CSS

```
#content{
}
```

★ This can also be chained to target specifically a "content" id within a div

ex/

```
div #content{
}
```

★ Partially Redundant however since every ID on the same page has to be unique from one another

Descendent Selectors

- Used to be more specific about the elements that are styled
- Targets children elements inside parent elements

If you wanted to target p tags inside a div, the CSS looks as follows:

```
div p {

}
```

First the parent element, space, child element, curly braces and declarations

This process can be several layers deep (grandchildren etc.)

Also, you can target classes inside a parent!

```
div .error {
}
```

The space is important in this scenario to distinguish parent-child and not a div with a "error" class

---

Attribute selectors

- Used if you want to target attributes of tags regardless of their value

Written in the form:

```
tag name [attribute] {
}
```

ex

```
a[href] {
}
```

- If you did want to target a specific value you can have the attribute equal the value inside the square brackets

ex/

```
a[href="www.google.com"]{
}
```

↳ These declarations only affect a tags that link to google

Shorthand Form:

```
a[href*="google"]{
}
```

↳ Asterick means any href value that includes "google" will be affected

☆ `a[href$=".com"]` targets any link ending in ".com"

The Cascade

- Made up of 2 concepts,
  Inheritance and Specificity

HTML elements can inherit CSS rules applied to parent elements

ex/

HTML

```
<div>
    <p> hello </p>
</div>
```

CSS

```
div {
    color: red;
}
```

☆ p tag would 'inherit' div's property

☆ If p tag had span tag within it, the content of the span tag would also inherit it's parent's parent's property

☆ Mainly font, text styles, colors inherited, not margin, padding, background and borders

You can actually make children elements inherit some properties of their parents manually:

```
p {
  border: inherit;
  margin: inherit;
}
```

Usually border and margin are not inherited but now there's the option to do so

The concept of Specificity on the other hand is capable of overriding inheritance:

HTML

```
<div>
  <p> hello </p>
</div>
```

CSS

```
div {
  color: red;
}

p {
  border: inherit;
  margin: inherit;
  color: lightcoral;
}
```

- The "hello" in the p tag will be lightcoral since the property of the p tag is prioritized over the div
- If there was another p tag with a different color property, whichever is lower on the stylesheet will take precedence
- Only exception to above is a descendent selector `div p { ~~~ }` which would have highest specificity

HTML 5 Semantics

- HTML is always evolving, always new elements added
- HTML5 shipped in more tags for better structure/description

```
<div>
  <p> content </p>
  <p> more content </p>
</div>
```

VS.

```
<article>
  <p> content </p>
  <p> more content </p>
</article>
```

New Tags

<article>
Defines content that makes up an article (ex/ Blog)

<main>
For main content of webpage, unique to that page

<section>
Defines a certain part of a webpage (ex/ navigation, contact info)

<aside>
Defines similar/related content
(ex/ related blogs)

<header>
For header of site - contains nav, title etc.

<footer>
For the footer of a website

<nav>
Contains navigation content of the page

☆ All of the tags mentioned has an opening and closing tag

Chrome Dev Tools

☆ Right-Click + Inspect on a website

- Useful to debug and inspect elements of your webpage

There are 5 tabs available: Elements, Console, Sources, Network and Performance

In the elements tag, you can:

- Right-click and Copy Selector if you don't know how to represent it in CSS
- Right-click and Edit as HTML to test quick changes
- Right-click and Scroll into View to see what something represents on the page

- Right-click and Hide Element or Delete element to remove pieces
- Right-click and Edit attribute / Add attribute to alter tag's elements
- Can Also Inspect CSS and Browser Styles by Inspecting and Scrolling Down
- Can add, edit, remove CSS in the styles tab for testing, when a style is overwritten, old style becomes crossed out
- Can enter hover mode by clicking an icon and shows what HTML corresponds to whats on the page

The Sources tab shows the tree of files and folders that make up a project

- You can live edit the sources inside the editor on the side

- When you refresh, you lose the changes in the editor

- If you want to save the changes, press the "Filesystem" sub-tag and add your corresponding project and then edit and save

- This is a great way to play around with the code and you have access to all the tabs/tools you need

In the console, you can mess around with Javascript

- In the device preview tab, you can see what your site looks like on different screen sizes
- You can also scroll the Chrome Dev tools side to side to preview at different dimensions

---

CSS Position & Layout

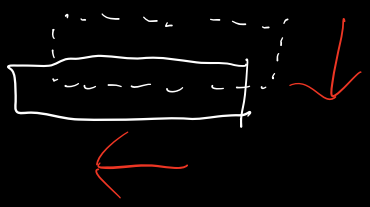
- The Position Property can have one of five values:

- Static - Relative - Fixed
- Absolute - Sticky

HTML elements have a default value of Static (Document Flow)

Relative means you can shift element around relative to it's original position in the page

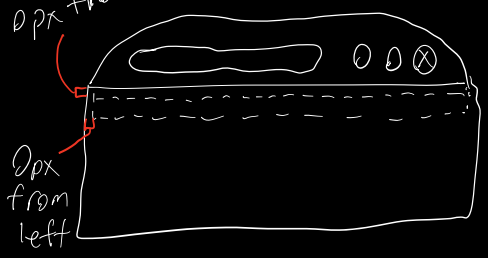
ex



```
{
position: relative;
left: 20px;
bottom: 20px;
}
```

Fixed positions an element relative to viewport (Part of browser that we see webpage on) so it stays as you scroll

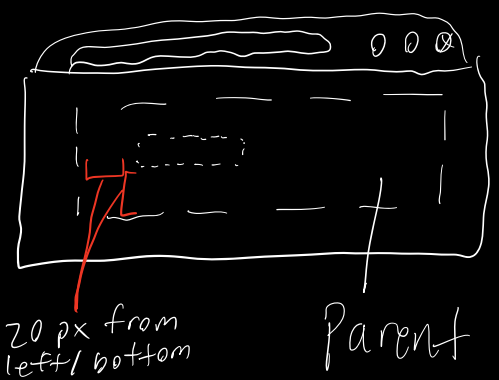
ex




```
{
position: fixed;
left: 0px;
top: 0px;
}
```

Absolute positions absolutely relative to it's closest parent element, which has anything but a static position

ex




```
{
position: absolute;
left: 20px;
bottom: 20px;
}
```

☆ Sticky is a mix of 'static' and 'fixed'

☆ If you want a picture to fit the view port, in CSS you must apply

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

☆ When you position absolately, it's taken out of Normal Document Flow - lose space element was originally in

☆ Unit of Measurement you can use other than px is em's.
ex/ 1.3em means original size multiplied by 1.3

☆ When using fixed position, the element may be overlapped or overlapping another element, we can use z-index values to fix this; Default zindex value is 0, anything more brings element forward, less puts it under

Something to note with CSS, when you use the padding property with only 2 numbers:

ex/

```
{
    padding: 6px 12px;
}
```

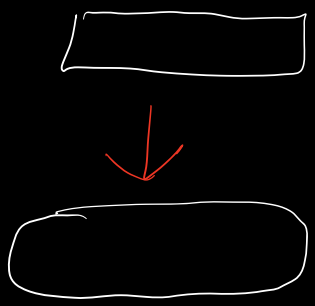
6px → top/bottom

12px → left/right

New property 'border-radius' gives a curved effect on element borders:

ex/

```
{
    border-radius: 10px;
}
```



If you gave an element a position of sticky, it will initially behave like static but when you reach a certain point, it will become fixed and scroll with the page

You can prevent inline-block elements from going into the next line despite being spread evenly via 'width' by giving a parent element

```
{
    white-space: nowrap;
}
```

You can centralize if extra space by adding

```
margin: 0px auto;
```

A CSS reset is meant to strip default browser styles by reverting some properties

ex/

```
{
  margin: 0;      - eliminate margin
  padding: 0;     - eliminate padding
  font-family: arial;
                  L web safe font
  L may actually affect spacing
}
```

☆ You can incorporate padding into total width using

```
{
  max-width: 100%      for smaller screens
  width: 1200px;       for our screen
  padding: 0 40px;     extra space
  box-sizing: border-box;   ensures padding doesn't mess up width
}
```

Pseudo classes & Elements

- Style elements when they are in a particular state
  - hover, focus, first child of a parent

:hover :focus :first-child

  - These are tacked onto end of selectors
  - They are in the form of a colon followed by a keyword

There are more of them than will be covered

When we 'hover' over an anchor tag, we can style that in a specific way

ex/

```
nav li a:hover {
    text-decoration: underline;
}
```

Will cause links to show underline when a cursor hovers over them

`bottom: 4px;` is the same as `top: -4px;`

If a field is in 'focus', for example entering text into a form, we can style that differently

ex/

```
form input:focus {
    border: 4px dashed #4b4b4b;
}
```

Makes it so that when input is selected, a custom border appears

When a form field is 'valid' for example, using @ for an email, we can style it

ex

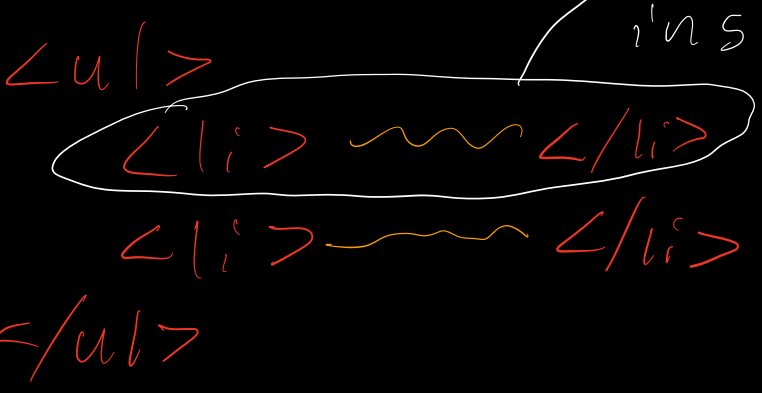
```
form input:valid {
    border: 4px solid #71d300;
}
```

This will cause the border to go from dashed gray to solid green when the text inputted is valid

When an element is the 'first-child' of a parent, it can be targetted and styled

ex

HTML



That first <li> tag can be separately styled:

ex

```
nav li:first-child {
border: 3px solid #F63232;
}
```

Adds red border to first <li> tag only

Pseudo Elements are similar to Pseudo classes and use same idea except 2 colons instead of one before Keyword

- Pseudo Elements allow injection of dynamic content

ex/ p::after p::before
for before/after <p> tag

- Also can style content inside tag

ex/ p::first-letter p::first-line
p::selection
for styling first letter, line and selected portion of an element

Intro to Media Queries

Responsive Design makes a website look good on all devices regardless of width of viewport

Media Queries are a part of this design which tells websites how to style an element at particular viewport dimensions

Viewport meta tag tells the browser what width the viewport should be

Responsive images means to only load smaller images for mobile devices

The Viewport Meta Tag is placed inside <head>:

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

specify width of viewport

No Zoom

Media Queries target elements and style them differently at different viewpoint widths

ex/ CSS

condition

```
@media screen and (---) {
  .banner .welcome h2 {
    font-size: 60px;
  }
}
```

Only put desired changes into nested selectors

Some conditions are

max-width: 1400 px
↳ anything with a width less than or equal to 1400 pixels will have these properties applied

min-width: 800px
↳ anything with a width more than or equal to 800 pixels will have these properties applied

If you style a
 tag in CSS with the property: display: none; the tag will not add a break as it's supposed to

Next Steps

Check out The Net Ninja's other playlists as well as his JavaScript Course on Udemy!