Table of contents (4 parts)

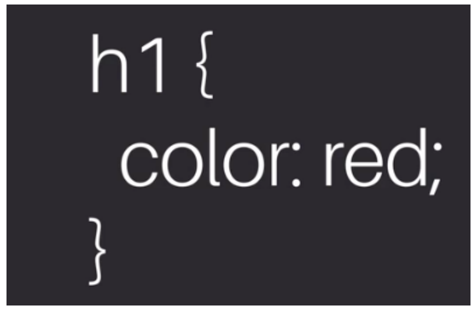
%%

Part #1 - Introduction to HTML and CSS

%%

- **1. Write your first lines of HTML and CSS**
    - -> HTML, CSS for websites
    - -> **HTML** -> structure / **content**
    - -> **CSS** -> **style** sheets (colours)
    - **-> index.html file <- open this in Chrome opens the project (content)**
    - -> write HTML then see it update int he browser
    - **-> the same webpage has an HTML and CSS file** -> which is compiled by the browser
- **2. Create content using HTML**
    - -> HTML -> telling a story on the webpage (hierarchy of information)
    - -> better for search engines/ browsers / accessibility
    - -> **you mark elements in HTML with tags <>element </> for CSS to format / style them - the two files containing the content and the styling for the content are linked**

- **3. Decorate your content with CSS**
  - -> style of content (content is in HTML)
  - -> styles can be different depending on
    - HTML content
    - **-> HTML (content) then CSS (styling)**
  - **-> h1 corresponds to the h1 tag in HTML (that content) and this is CSS**

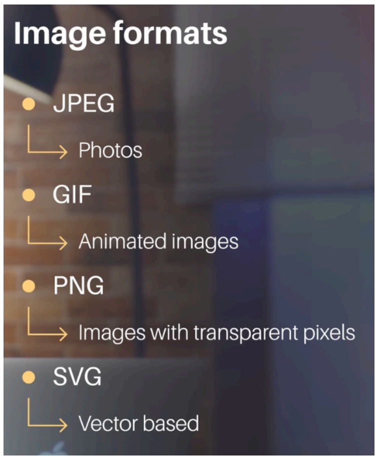


%%
Part #2 - Create HTML text elements
%%

- **1. Create headings for strong page structure**
  - -> **codepen is like overleaf but for HTML and CSS**
  - -> 1-6
  - **-> <h1> heading </h1> <- from h1 (big to small) to h6**
- **2. Add text in paragraphs**
  - -> <p> paragraph text </p> <- you can't just have a block of text in HTML without tags around it - the browser will lump the entire thing togather -> e.g
    - sdfsdfg
    -
    - dfgsdfg
    - would show up as sdfsdfg dfgsdfg without these tags
- **3. Strengthen and emphasize text**
  - -> bold / italics
    - **->   <- bold (strong)**
    - **->   <- italics (emphasis)**
  - -> you are wrapping the content in these tags
- **4. Add links and understand attributes**
  - -> linking sites to other sites
  - -> attributes -> in HTML
    - **<element attribute_name="value"> Contents </element>**
    - -> these can be used to create links
    - **-> attribute_name = href, value = URL name**
    - **-> tags are for the elements (the element, this is a paragraph etc, the attributes are the properties of that element (which aren't the styling because that is CSS)**
- **5. Organize elements in a list**
  - **-> ordered (bullet points) and unordered lists (numbers)**

- **-> <- unordered list (bullet points)**
        - ** list item <- li (list item)**
    - **
**
- **-> to make it numbered, change <ul> to <ol> (ordered list)**

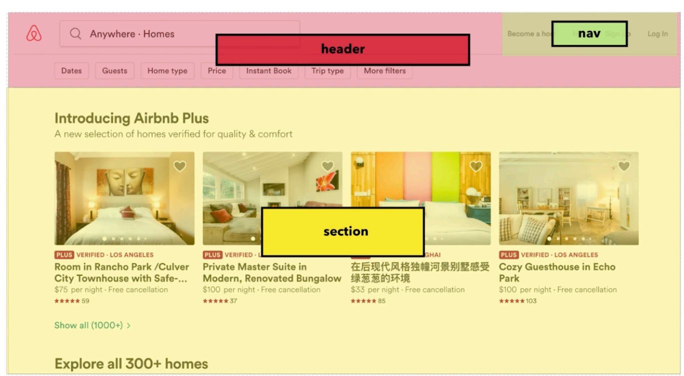
- **6. Add images to your web page**
    - **-> <img src="URL to image" alt="text for people listening to the webpage" title = "name of the image (descriptive)">**
- **7. Use the best images possible**
    - **-> stock images not paid images**
    - -> image formats

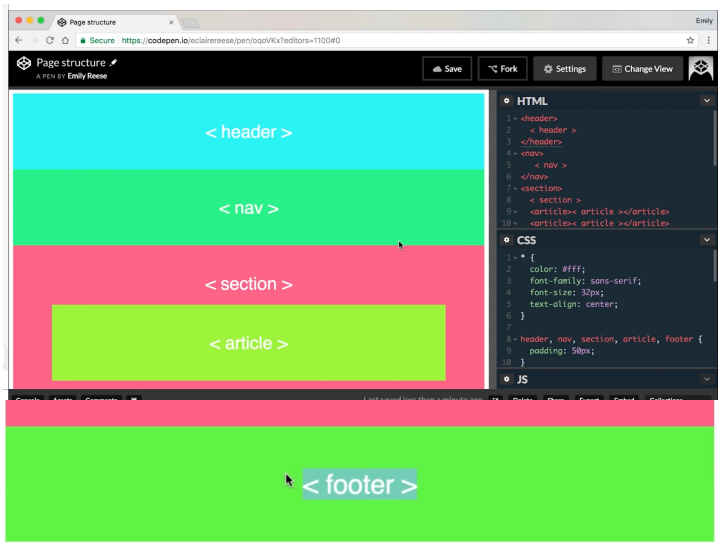


- JPEG for webpages with a lot of images
- two types of PNG -> colour types
- Vector -> this can scale without pixelating

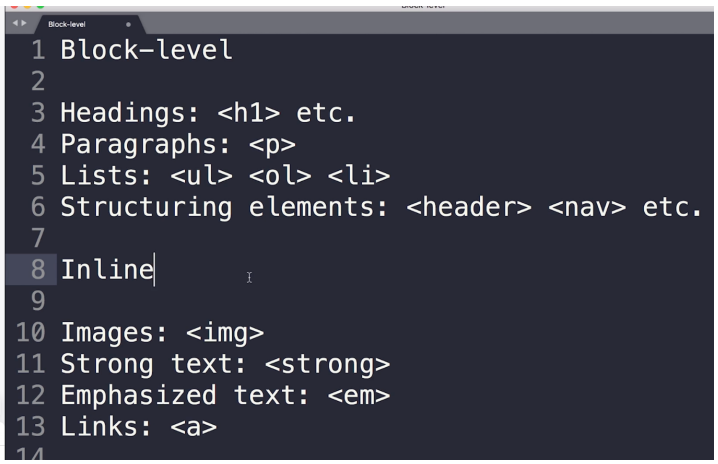
%%%
Part #3 - Structure an entire page
%%%

- **1. Create general page structure**
    - -> the webpage is a story -> it's telling a story
    - -> different parts of the webpage
        - header includes the logo
        - **-> all websites are made of the same page structure elements**
        - **-> you can take an entire website and split it into different parts -> header / nav / etc**
        - **-> each part has a different tag**
            - **<></>**





- **2. Understand block-level and inline elements (aka these elements stack and these don't)**
  - -> each element has a default behaviour
    - **block level elements -> span the entire width of the page (paragraphs)**
    - **inline elements -> stack next to each other (images are inline elements)**



- **3. Group content with divs and spans**
  - -> zones of content
  - -> the entire page is split into different zones of content
  - -> you can also define group of content via divs and spans
    - to make your own groups of content
    - **-> a div -> group e.g two paragraphs which revolve around the same theme**
    - **-> div -> to divide up the content (into your own sections), you can decorate certain ones differently in CSS**
    - **-> spans -> this does the same thing, but for in line content (not entire blocks of things)**
- **4. Add classes and ids to elements**
  - -> classes and IDs
  - -> to make one paragraph green but not another
  - **-> Content for the paragraph **
    - **-> you can give other paragraphs that same class**
    - **-> then format that class of paragraphs (those types of paragraphs) differently in CSS**
    - **-> this is an attribute**
  - **-> an ID -> this is a class but for one element (not a group of them, if you just want to format one element)**
    - ** Content for the paragraph </ p>**
- **5. Add breaks and lines to your content**
  - -> adding lines in HTML
  - **-> if you want to add more space ->
 (rather than putting the entire thing in a new**

paragraph), this is a line break
    - -> to add a horizontal line <hr> (horizontal rule)
- **6. Add a head to your HTML for information about your website**
    - -> metadata in the head of the document -> which the browsers use
    - -> the content goes in the body
    - -> **information about the document goes in the head**
    - -> <meta charset="utf-8"> <- to use other character sets

```html
<!doctype html>
<html lang="en">
    <head>
    </head>
    <body>
    </body>
</html>
```

```html
<meta charset="utf-8">
<meta name="description" content="Dogs are incredible creatures. There may still be some facts you don't know about them!">

<!-- LINKS TO CSS FILE ("STYLESHEET") AND SITE ICON -->
<link href="css/style.css" type="text/css" rel="stylesheet">
<link href="favicon.ico" type="image/x-icon" rel="shortcut icon" >

<!-- FACEBOOK METADATA -->
<meta property="og:image" content="https://dogsdogsdogs.com/images/dog.png">
<meta property="og:description" content="Dogs are incredible creatures. There may still be some facts you don't know about them!">
<meta property="og:title" content="10 Fun Facts About Dogs">

<!-- TWITTER METADATA -->
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="10 Fun Facts About Dogs">
```

    - **-> if you get confused -> go and inspect a webpage on the internet and it will tell you the code for the element you want**

%%%
Part #4 - Spice up your content with CSS
%%

- **1. Apply CSS to elements <- CSS vs HTML syntax**
    - -> CSS has different syntax to HTML
    - **-> HTML <>, CSS {}**
    - **-> name_of_html_tag {**
        - **property: value;**
    - **}**
        - **-> the selector is the element in HTML which you are formatting in CSS**
        - **-> you have the CSS code which is formatting the styling of the content marked up in the HTML document**
    - -> except from classes and IDs
    - .class_name {
        - property: value;
    - }
    - #id_name {
        - property: value;
    - }

CSS

```css
selector {
  font-family: Helvetica;
  color: blue;
  background-color: yellow;
}
```

HOW TO FORMAT/STYLE AN ELEMENT IN CSS (SYNTAX)

< ></ >

- **2. Decide where to write CSS <- different files and CSS (html and css for same project or all in html)**
    - -> **you can write the CSS in the html file, or in a separate one in the same project folder as the HTML html file (the entire project has a css and an html file -> the html file - index.html manages the content and the css manages the formatting / styling for it)**
        - **-> the css file is linked to in the header of the html file (or the css can be written**

directly into the html file)

- **-> then in the CSS file, it's the tag of the HTML index {**
- **}**
  - **-> and the styling for it goes in between**

- -> in the project folder, there are html and css files (for the same project - one for the content and the other for the styling of the content)
  - -> index.html file
- -> **to have a CSS and HTML file and link them**
  - **in the head of the HTML file**
    - **<link href = "relative_path_to_css_file/name_of_css_file.css" type = "text/css" rel="stylesheet">**

How to link the index.html content file to the .css style file for the project

- -> she's linked the CSS file to the html file -> then changed one of the elements in CSS, refreshed the webpage and it's updated
- -> **to have the CSS in the HTML file**
  - **for an entire section**
    - **under the name of the section type (<head> </head> in this case) she's written**
      - **<style type = "text/css">**
        - **h1 {**
          - **property: value;**
        - **} <- the CSS gets indented in the <\style> tag**
      - **</style>**
  - **for an element**

you can have an index.html file for content, and a separate css file for styling / formatting → or you can put the css in the html file

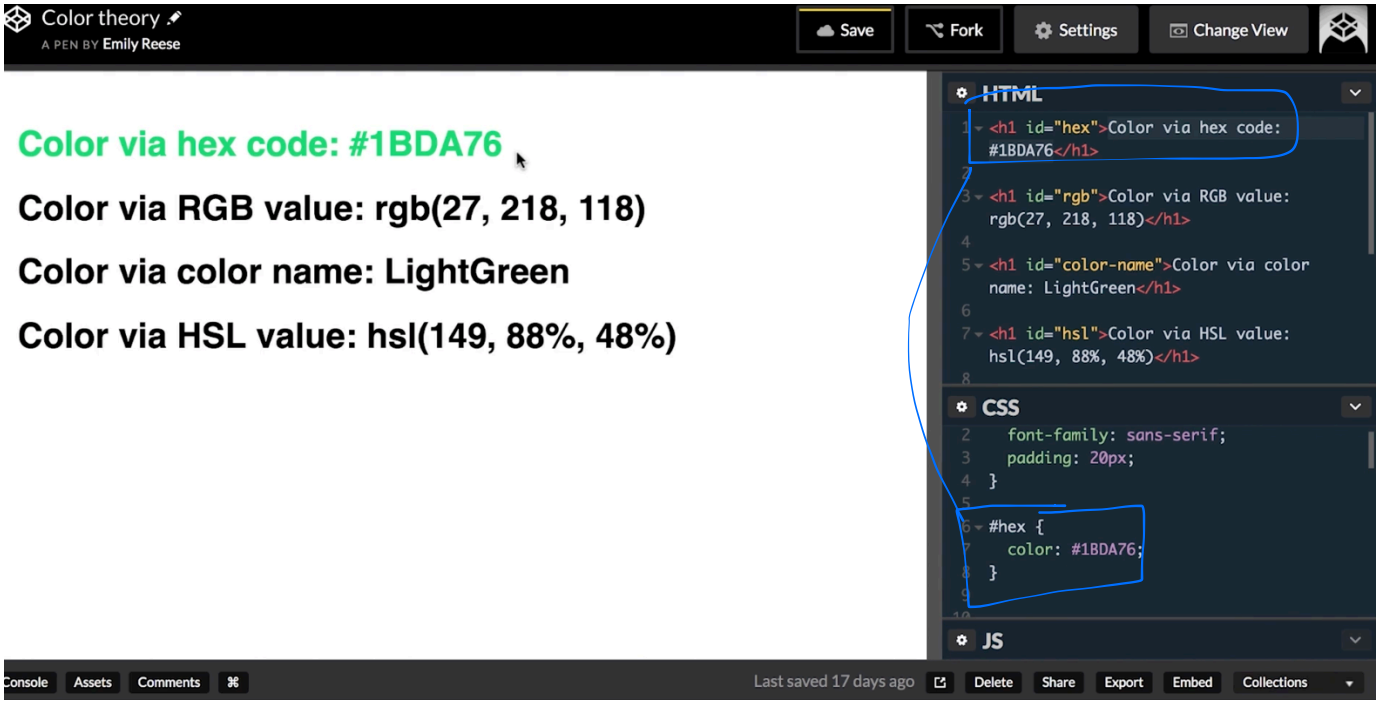
→ this is putting the css in the html file (awfully useless), above is the x2 separate files (css, html file) and linking them method

don't like this method, use the other one → proper css, html in separate documents

creates content + styling of webpage

```
<body>
  <h1 style="color: blue;">Hello</h1>
</body>
```

- **3. Set your first colours <- <u>how to do colours in CSS (syntax)</u>**



- -> colour and background colour
- -> colours in CSS
- h1 {
- }
- -> h1 corresponds to an HTML tag (but this is in CSS)
- h1 {

CSS - style

        - color: green;
    - }
    - -> this is compared to <> </> HTML tags
    - **-> you can target any HTML tag in CSS (e.g a -> link types)**
    - -> properties are:
        - color: ...;
        - background color: ...;
    - **it's**
        - **aka <> ... </> <- this is the tag in html**
        - **in CSS, you target that element to format it**
            - **-> CSS is for the styling of the html, and html is for the content (but it's marked up -> to be formatted / styled in CSS)**
            - **-> so you find the element on the webpage you want to target**
            - **-> figure out what the tag name is in html**
            - **-> then go into CSS to format that element**
            - **html_tag_name {**
            - **}**
            - **-> {} is CSS <- and <> </> is html**
            - **-> in the {}'s -> that's where you write the colour of the element you are dealing with**
                - **-> the syntax aka format of that is**
                    - **color: ...;**
                    - **background color: ...;**
                - **-> in the CSS {}'s**
- **4. Understand colour theory <- <u>how to do colours in CSS (units)</u>**
    - **three ways -> the name of the colour in text, or hexcode, or rgb**
    - **<u>-> this is color: ...; <- what the ... is</u>**
    - hexcode
        - 6 digits
        - for the hexcode, it's
            - color: #....;
        - for the rgb, it's
            - color: rgb(x, y, z);
        - for the name of the colour, it's
            - color: lightgreen;
        - hsl ~~method (hue, saturation, lightness)~~
            - color: hsl(149, 88%, 48%);
- **5. Set fonts <- <u>how to do fonts in CSS</u>**
    - -> the feeling of the website
    - **<u>default fonts</u>**
        - -> **ticks on the end of the t's are serifs**
        - **-> font-family: serif;**
        - **-> font-family: sans-serif;**
        - **-> font-family: monospace; <- each letter takes up the same width**
        - **-> font-family: 'font_name;**
    - **<u>backup fonts</u>**
        - **-> the user has to have the font installed on their computer -> so there are backup fonts in case the primary ones aren't installed**
            - **they're done this by setting two of them in CSS**
            - **-> font-family: serif, sans-serif;**
- **6. Control font sizes, line spacing, and word spacing <- <u>how to control the space between</u>**

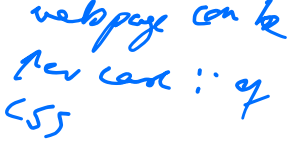
things in CSS
- -> controlling the appearance of the font
- -> e.g adding space between words
- -> changing font sizes
  - pixels, percentages and ems
  - font-size: 1.4em;
  - font-size: 1.4%;
  - font-size: 1.4px;
- -> changing spacing
  - html_element_tag_name {
    - letter-spacing: 0.5 em; <- to add space in between letters
  - }
  - **other ones are word-spacing (to add space in between words), line-height (adds space in between lines)**

NAMED + BACKUP FONT IN CASE THE FIRST ISN'T ON THE COMPUTER

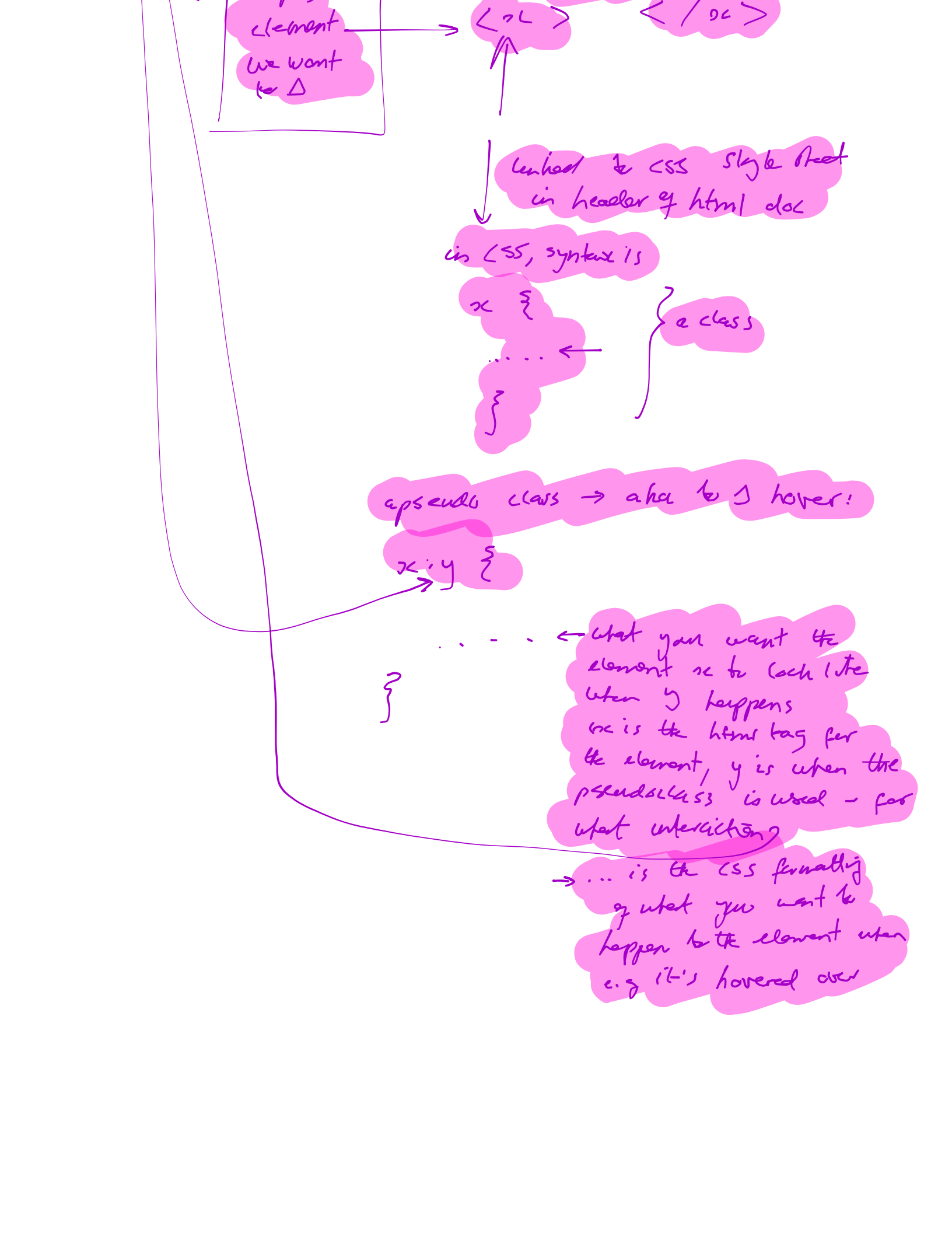
TO CHANGE FONT SIZES IN CSS

- **7. Trick out your text <- defining pseudoclasses, aka how to make the text change when it's being hovered over e.g**
  - -> effects in CSS -> hovering
  - **-> more example properties in CSS <- aka, the things which go inside the**
    - **html_tag_name {**
      - **here**
    - **}**
    - **-> this is CSS**
    - **-> to format the html element**
    - **-> the html_tag_name is what's in the <> </>'s in html, this is the content**
    - **-> then this (above) is the syntax of the CSS for it (to change the sytle of it)**
    - **-> then below are the different things where 'here' can go**
  - **font-weight: normal / 200 / ; <- how bold the font is**
  - **text-decoration: none / underline / wavy underline; <- to change the underlines**
  - **test-transform: uppercase / lowercase;**
  - **color: #....;**
  - **-> pseudoclasses <- an effect when the element has been interacted with by the user**
    - html_tag_name:visited {
      - color: ...; <- changes the colour of the element when the mouse have clicked on it
    - }
    - html_tag_name:hover { <- when it's hovered over
    - }
    - html_tag_name:active { <- when it's being clicked on
    - }
    - **the general format for this is, in CSS**
    - **html_tag_name:name_of_pseudoclass {**
      - **here (effect we want)**
    - **}**
      - **-> the name_of_pseudoclass is**
        - **visited -> changing the element when it's been clicked on**
        - **hover -> changing the element when it's being hovered over**
        - **active -> changing the element when it's being clicked on (and the click is being held down)**

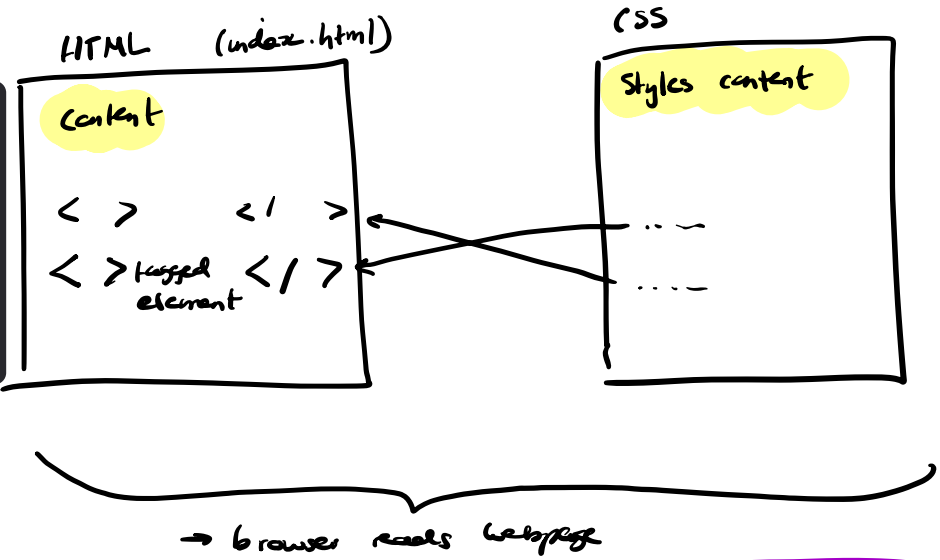
HOW TO USE PSEUDOCLASSES → CLASSES WHICH ARE USED WHILE - TO CHANGE AN ELEMENT E.G WHILE HOVERED OVER



- **8. Get some practice building your first page with HTML and CSS**
  - -> this chapter is a project-based exercise


element we want to Δ
<x>
</x>
linked to CSS style sheet in header of html doc
in CSS, syntax is
x { ..... }
a class
pseudo class → aka to ʃ hover!
x:y { ..... }
what you want the element x to look like when y happens
x is the html tag for the element, y is when the pseudoclass is used - for what interaction?
... is the CSS formatting of what you want to happen to the element when e.g. it's hovered over

HTML TAGS AND CSS

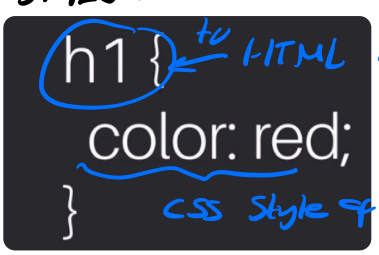


HOW TO FORMAT AN HTML TAG

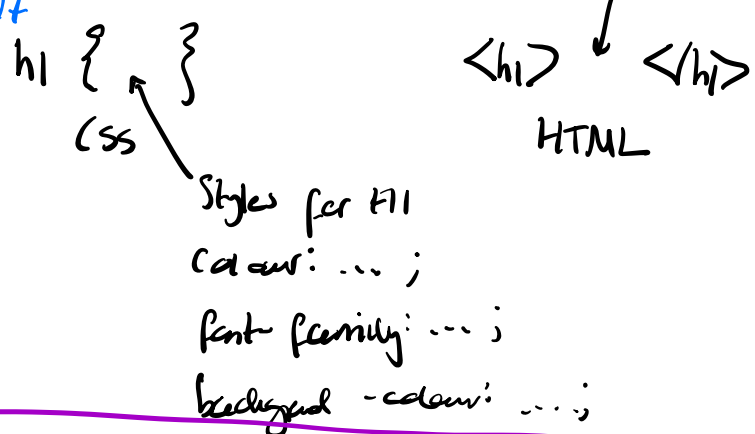
<h1> </h1> ← to tag a header

  ← " " = paragraph

CSS → STYLE HTML CONTENT AFTER HAVE CONTENT:



HOW TO FORMAT CSS



HOW TO ADD DIFFERENT THINGS INTO HTML

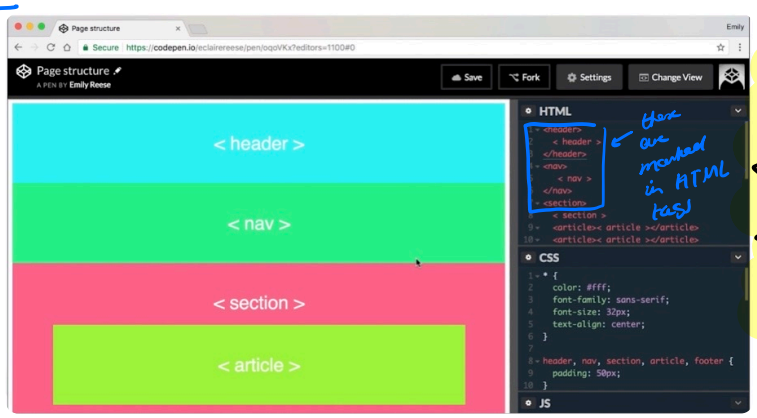
- -> HEADINGS: <h1> heading </h1> <- from h1 (big to small) to h6
- -> PARAGRAPHS: paragraph
- -> BOLD AND ITALICS:
  - ->   <- bold (strong)
  - ->   <- italics (emphasis)
- -> URLS / HYPERLINKS:
  - <element attribute_name="value"> Contents </element>
    - -> attribute_name = href, value = URL name
    - -> (an attribute of that element, like in Python - an attribute of the object)
- -> BULLET POINTS AND NUMBERED LISTS:
  - -> <- unordered list (bullet points)
    - list item <- li (list item)
    -
  - -> to make it numbered, change <ul> to <ol> (ordered list)
- -> TO ADD IMAGES
  - -> <img src="URL to image" alt="text for people listening to the webpage" title = "name of the image (descriptive)">
  - -> stock images not paid ones

TO GROUP SECTIONS OF A WEBPAGE

TO GROUP CONTENT ON AN ENTIRE WEBPAGE -> HTML

GROUPING OBJECTS ON A WEBPAGE TOGETHER

DEFAULT SECTIONS OF WEBPAGES



(lowercase!)

<HEADER>
<NAV>
<SECTION>
<ARTICLE>
<FOOTER>

< > </ >
HTML tags

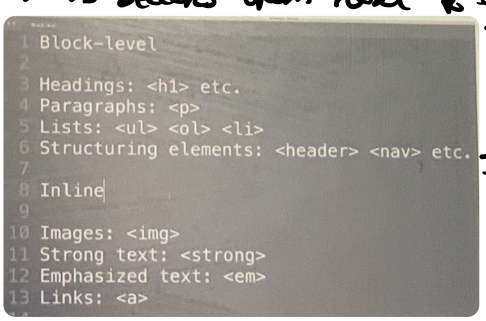
DIY SECTIONS OF WEBPAGES

< div > < / div >
< span > < / span >

e.g grouping x2 pieces togeth to theme style them

-> Span width of page (inline)
-> vs stacks them next to eachother (divs them (block)

divides them vs spans entire page



TO STYLE A SINGLE PART OF A WEBPAGE

PICKING OUT INDIVIDUAL SECTIONS TO STYLE SEPARATELY

→ Multiple – use classes
→ Individual one – use id

- -> Content for the paragraph
  - -> you can give other paragraphs that same class
  - -> then format that class of paragraphs (those types of paragraphs) differently in CSS
  - -> this is an attribute
- -> an ID -> this is a class but for one element (not a group of them, if you just want to format one element)
  - Content for the paragraph

you are adding an attribute to access of para types e.g.

then style that class in CSS

TO ADD LINES / GAPS

CREATING GAPS AND LINES IN BETWEEN INDIVIDUAL SECTIONS OF TEXT → You could also do a div (span + add padding etc.

→
 ← a break (less space than a new para) – line break

→ <hr> ← a HORIZONTAL LINE (rule)

TO ADD A LINE / HORIZONTAL LINE

TO GET HELP

→ inspect a webpage and look at how the code for particular elements of the webpage was made

TO ADD METADATA ABOUT THE WEBPAGE IN THE HEAD OF THE HTML PAGE

→

```
<meta charset="utf-8">
<meta name="description" content="Dogs are incredible creatures.
There may still be some facts you don't know about them!">

<!-- LINKS TO CSS FILE ("STYLESHEET") AND SITE ICON -->
<link href="css/style.css" type="text/css" rel="stylesheet">
<link href="favicon.ico" type="image/x-icon" rel="shortcut icon" >

<!-- FACEBOOK METADATA -->
<meta property="og:image" content="https://dogsdogsdogs.com/images/
dog.png">
<meta property="og:description" content="Dogs are incredible
creatures. There may still be some facts you don't know about them!
">
<meta property="og:title" content="10 Fun Facts About Dogs">

<!-- TWITTER METADATA -->
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="10 Fun Facts About Dogs">
```

← "name" is the thing @ the top of the tab in Chrome

→ "content" is used for SEO

→ then links to CSS file (in HTML file header)

→ SEE COMPUTER/ TYPED NOTES FOR SECTION 4 HOW TO'S

→ IT'S CSS

Quiz points

rgba → vs (r, g, b) → a is transparency