BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Black Codher Bootcamp

NÍ YÓ

BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

UNIT 2
HTML & CSS

NÍ YÓ

BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

SESSION 1
Preview Time

NÍ YÓ

BLACK CODHER

CODING PROGRAMME

Introduction to HTML

What is HTML?

BLACK CODHER
CODING PROGRAMME

HTML stands for Hypertext Markup Language. The key words are “markup language”.

That means it describes how a page should be set up, as it’s there to mark up the the text.

Technically, HTML isn’t considered a “programming language”. It defines the structure of your web content.

You can see an example to the right.

By the end of this lesson you will understand what all this means!

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.
     Is it me you’re looking for to design your site?
    </p>
    </main>
  </body>
</html>
```

Okay, how does it work?

We'll be taking it line by line, but first of all, some definitions!

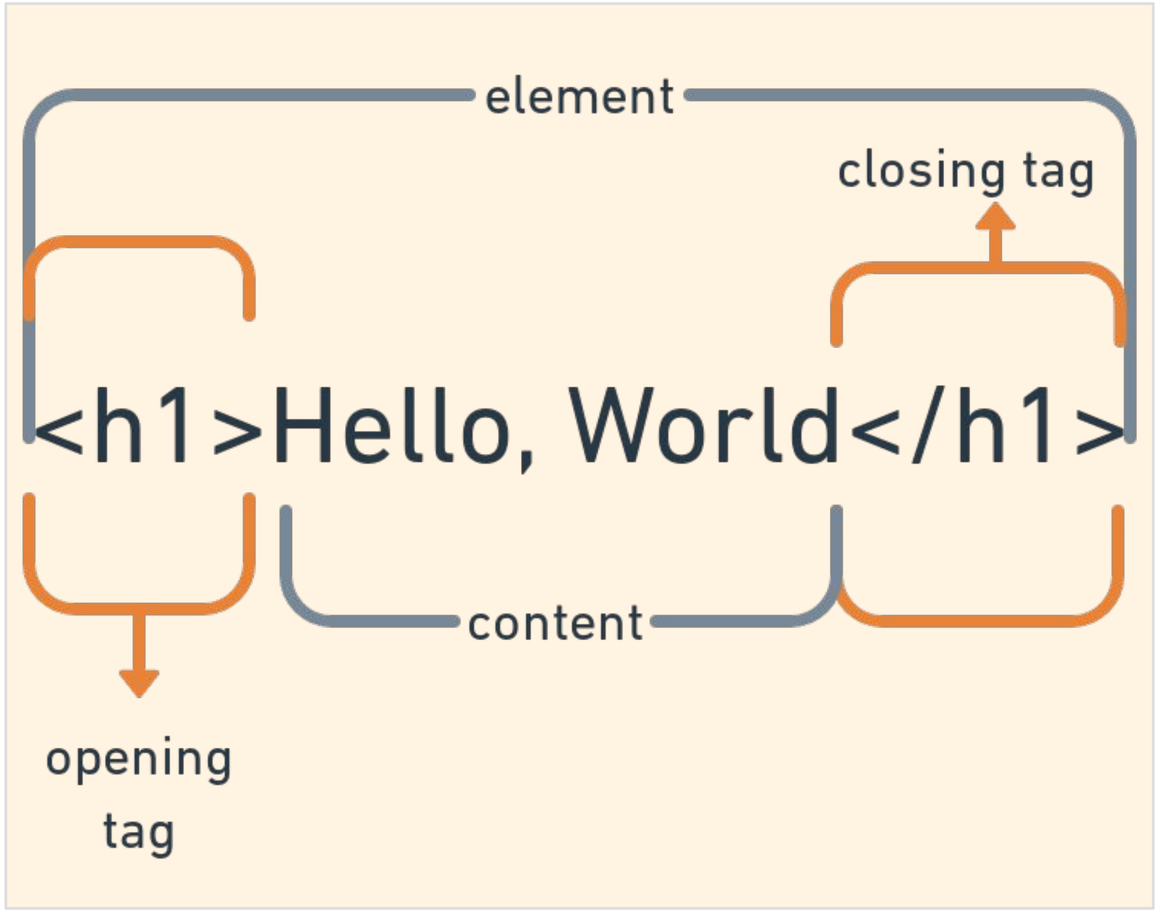
The character at the start of every html tag is an opening set of angle / diamond brackets < >

Tags show where HTML elements start and end. All closing tags include a forward slash - /

An **element** is everything together; all the content and the opening and closing tags.

Content here is of course everything inside of your tags.

(<h1> stands for heading 1. We'll cover that in more depth later.)

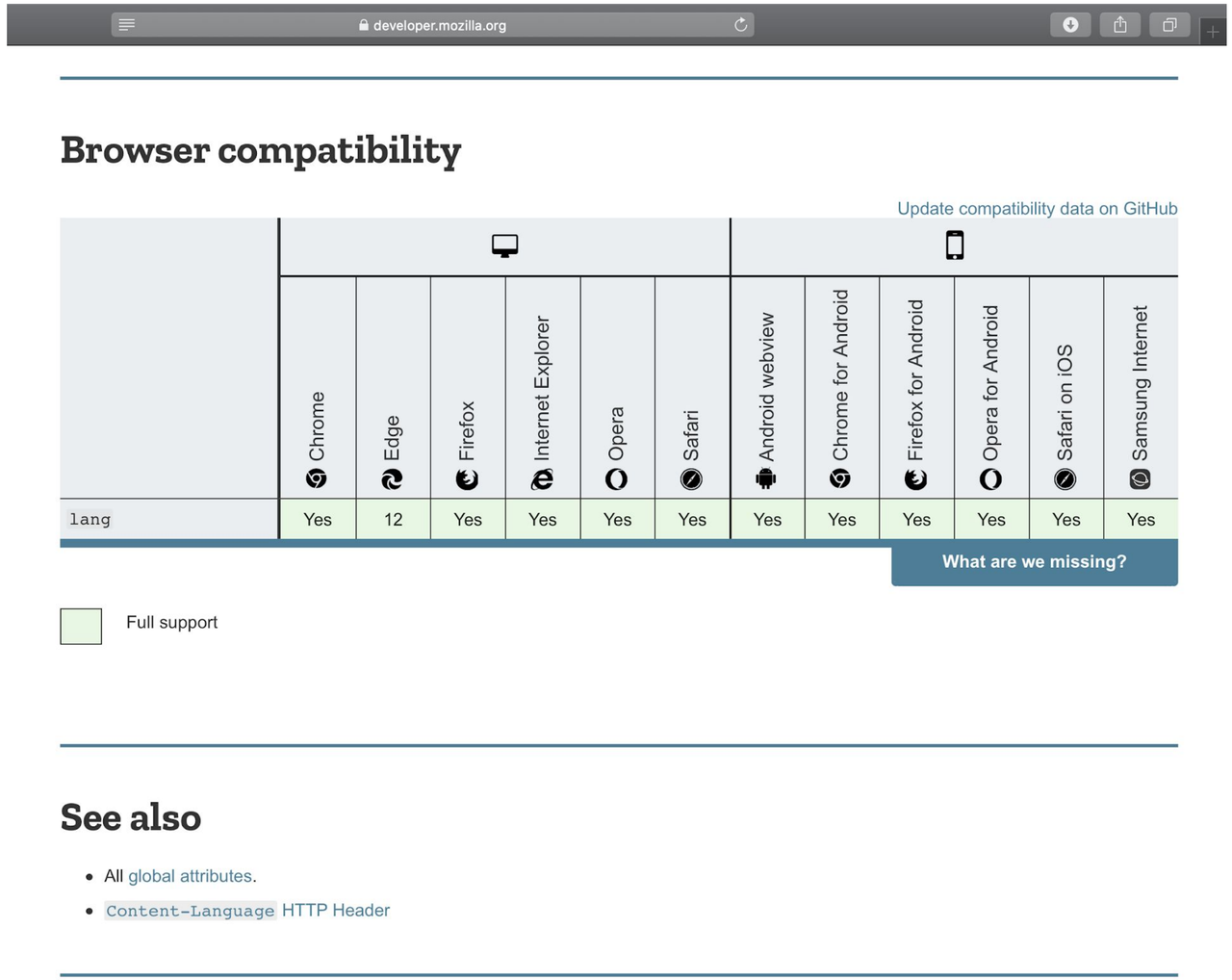


Browser compatibility

Browser compatibility is how flexibly a website can adapt to different web browsers.

It's important because among other things it improves **accessibility**, reach in general, and avoids lower performance - which is factored in for **search engine optimisation (SEO)**.

SEO refers to enhancing your website to get more traffic as in hits from people visiting it based on what they looked for on Google with its 70-85% market share (or Bing, Baidu, Yahoo, Yandex, Ask, DuckDuckGo, Ecosia... *cough cough). So it's about having the site appear ideally in the first 3 organic results.



BLACK CODHER

CODING PROGRAMME

Introduction to CSS

What is CSS?

BLACK CODHER
CODING PROGRAMME

CSS stands for **C**ascading **S**tyle **S**heets.
It controls how your HTML pages look - the design e.g. fonts, colours, and sizes as well as layout i.e. whether things are in centered or several columns, spaced out and so on.

So CSS describes how HTML elements are to be displayed. It can also be used for graphics, animations, etc. What do you think these 3 do?

Modern Web Design
HTML: Structure
CSS: Presentation
JavaScript: Behaviour
React.js, Node.js: JavaScript libraries
MongoDB: Database
Python: Backend

```
p {
  color: purple;
}

#about {
   padding: 10px;
}

.class {
  font-family: Crafty Girls;
}
```

And how does that work?

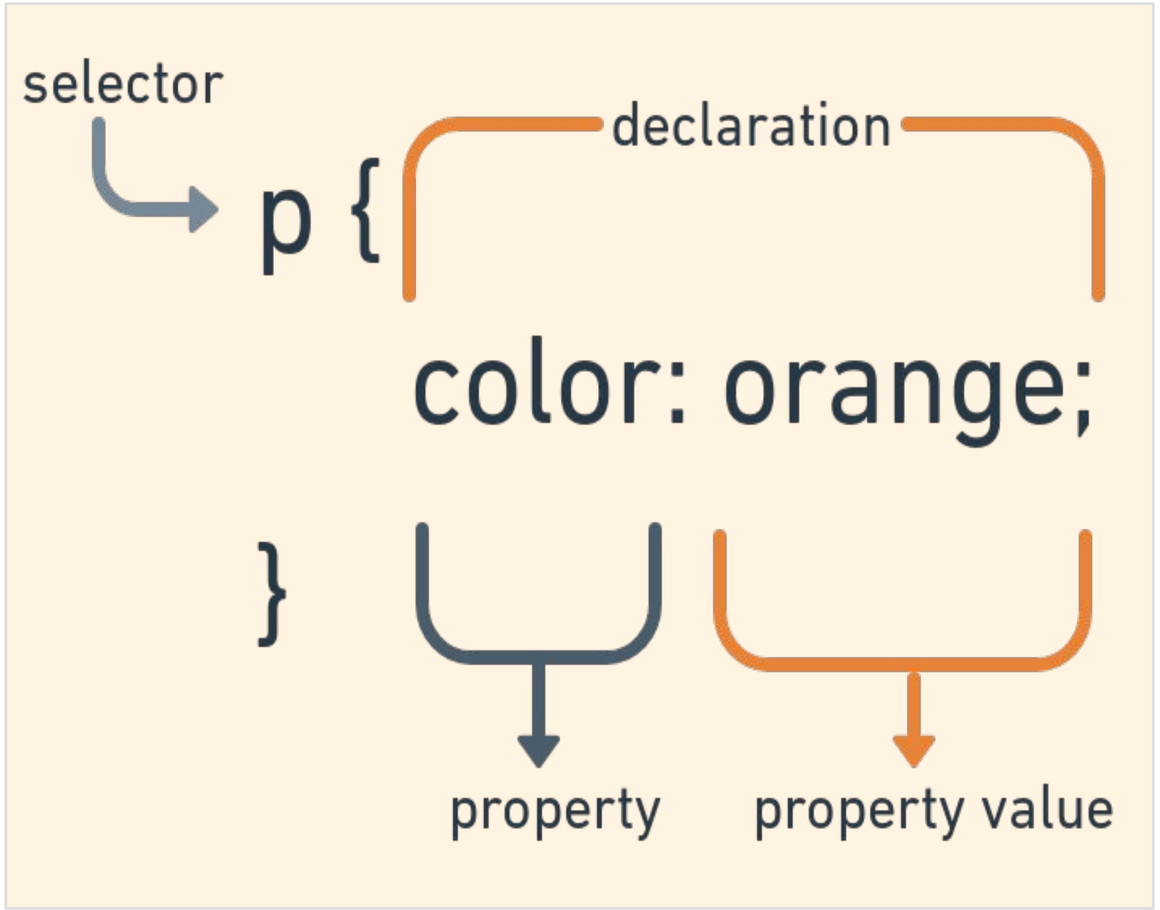
BLACK CODHER
CODING PROGRAMME

At the start of any CSS rule as they're known, is a **selector**, which corresponds to part of the HTML like a tag or an element.

Within the curly braces are a **declaration**, which consists of a **property** - the thing we are changing - and a **property value** - what we want to change it to next.

You can select more than one element at the same time. (p stands for paragraph.)

Notice that here it's curly brackets / braces { } (aka squiggly or flower ones) not angle. Pay attention to colons : and semi-colons ; as they often lead to bugs - code errors.



Commenting our code

BLACK CODHER
CODING PROGRAMME

As we've mentioned before, you can use comments above your code: those extra lines which explain what something is (for) and or does. **You don't have to include them, though I might at times. It's up to you if you want to make notes like that.**

Remember, the computer ignores them because they are not its instructions, they help us humans by serving as a reminder or additional guidance for our colleagues. Another good use case is using them to "turn off" part of your code for testing.

That being said, ideally where possible following clear naming conventions will prevent needing many, especially for HTML.

```
<!-- This is an aide memoire for an index.html file -->
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

```
/* This is an aide memoire for a style.css file */
 body {

 }

 main {

}
```

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

BLACK CODHER

CODING PROGRAMME

Time for a quick break

BLACK CODHER

CODING PROGRAMME

HTML 101

Learning objectives

1. Understand how to start from scratch
2. Understand how to lay out HTML
3. Understand what goes in the head

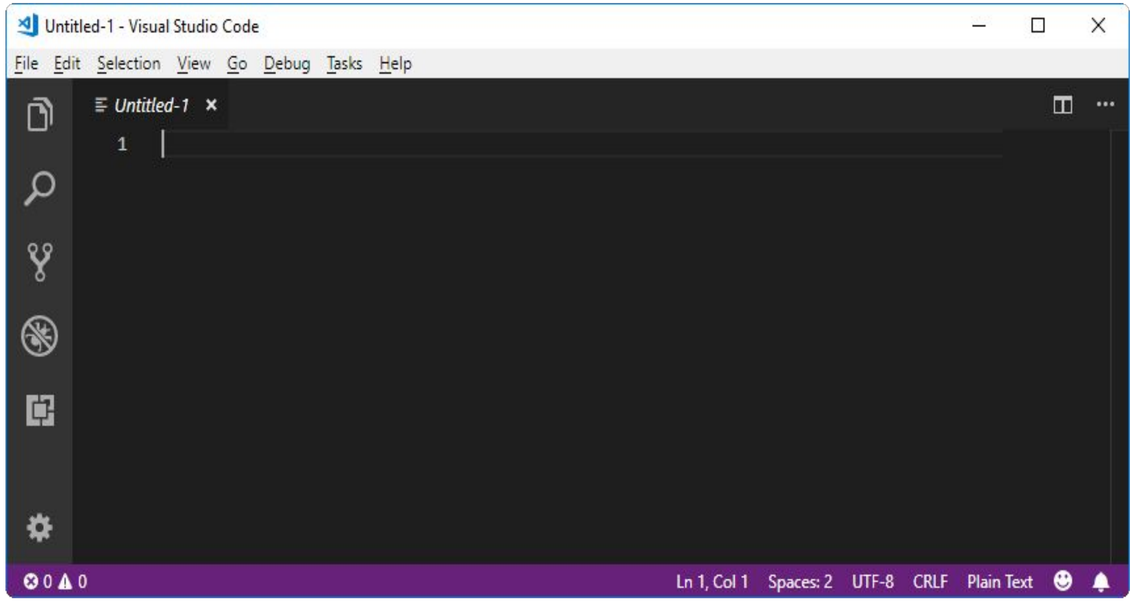
Let’s get it started

Open VSCode, go to File and create a new file. Select the built in text editor, then return to File and save as **index.html**

(*Sidenote:* **index** is used because the world wide web was originally envisioned as just directories to navigate between the links listed. It’s the most common naming convention, but feel free to call yours something different if you prefer.)

Next, create your external style sheet. You can save it as something like **style.css**

Before we go on, please bear in mind that this is about recall vs remembering! I encourage you to follow along each step. We’ll recap at the end.



!DOCTYPE html

BLACK CODHER
CODING PROGRAMME

The first thing we'll put in our index file is **<!DOCTYPE html>** - highlighted in bold ->

It tells the web browser what mode to render your document as and the html type, so that everything works correctly and your HTML is valid. Just remember to include it and you're good to go.

Feel free to add this into your html file now.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.</p>
    </main>
  </body>
</html>
```

HTML root element

Second of all, we've got our first proper HTML element... **<html>**, whose tags wrap all the content on page. It's sometimes referred to as the *root* element. However, it's optional.

It's not considered *best practice* in e.g. the Google Style Guide to include optional tags: https://google.github.io/styleguide/htmlcssguide.html

This is because HTML pages can get bogged down by having too many elements, which makes things trickier for browsers. It's important to keep code *optimised*.

A good way around this is to use it to set the language as an attribute - a special word which adds something to an element.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.</p>
    </main>
  </body>
</html>
```

Head

Now for the opening tag of the **<head>** element.

These tags contains the content that you don't need your user to see: the non-visual or invisible parts of your code which make things work BTS.

For example:

- links to code to style the page
- links to code to make it more interactive
- language settings so that the browsers know what to display for localisation
- keywords and page descriptions for search engines based on what people look for

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.</p>
    </main>
  </body>
</html>
```

Metadata

Metadata is primarily used to give search engines more information about your page. There are a number of different meta tags, and these are some of the most common:

- A basic description of your page
- A selection of applicable keywords
- The viewport to adjust views for devices
- The page author

Viewports apply to responsive web design, which we'll expand upon in later slides.

```
<meta name="description" content="A programming portfolio">

<meta name="keywords" content="business, code, developer>

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

<meta name="author" content="Black Codher">
```

Meta charset

Next is a **<meta charset>** element, indicating the character set of the page.

As you can see, this element does not have a closing tag. Some elements only have an opening tag, and are usually embedding or inserting something into the web page.

This element makes the character set for your document **UTF-8**. That stands for 8-bit Unicode Transformation Format: encoding for the binary language computers speak.

This set has most of the characters from most languages, so it can handle most text. Setting this can help avoid problems later on.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.</p>
    </main>
  </body>
</html>
```

Title

We'll wrap things up at this stage with a **<title>** element, which sets the name of the page in tabs, bookmarks, and so forth.

It should only contain text content, as other elements are ignored anyway. Plus it's always important to think about accessibility e.g. in terms of special characters.

What you put in the title can have a big impact on how search engines rank your page.

This takes us nicely onto our closing head tag.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
    <p>Hello World! I am a developer.</p>
    </main>
  </body>
</html>
```

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

Task 1: Laying out our HTML

BLACK CODHER
CODING PROGRAMME

1. Create your **index.html** file in VSCode if you have yet to do so, with **<!DOCTYPE html>** as line one.
2. Add the **<html>** root element. You can add any language attribute e.g. **<html lang="en">.**
3. Put in your opening and closing **<head>** tags to place all the non-visual invisible elements there.
4. Include the **<meta charset>**, and other metadata tags if you like e.g. a description, keywords...
5. Finish things off with the **<title>** tags: make sure it's meaningful for search engines (read: Google)!

Please feel free to refer to the slides to check what we've covered so far. Happy coding!

BLACK CODHER

CODING PROGRAMME

Time to take a break

Learning objectives

1. Understand what goes in the body
2. Understand what semantic means
3. Understand HTML best practice

Body

Here we'll start with a **<body>** element; this is where the (visible/visual) magic happens!

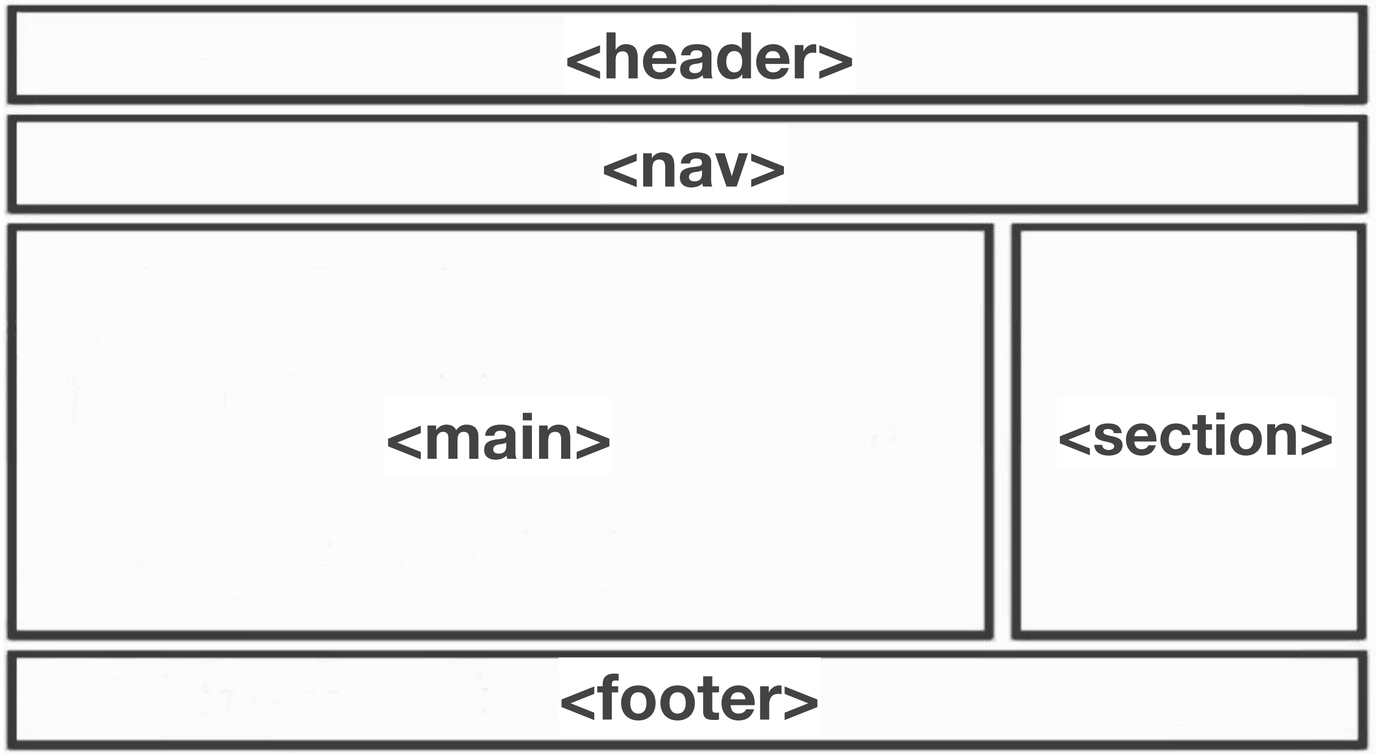
This is the part of the page where all the content your user will see on the page is held.

Images, videos, games, text; anything your user would be interested in, lives here.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Hello World! I am a developer.</h1>
    <main>
      <p>Hello, is it me you're looking for?</p>
    </main>
  </body>
</html>
```

BLACK CODHER

CODING PROGRAMME



Semantic elements have internal meaning which communicates context to search engines.

“I have a dream for the *(Semantic)* Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers.”
- Tim Berners-Lee, inventor of the World Wide Web

Header

The **<header>** element usually houses the links to navigate between the different sections of your page, or pages on your website - as well as content which introduces your site.

You can have more than one header element in your document for instance within the different **sections** of your page.

It often incorporates **headings**, logos / favicons, and or authorship details.

We'll go over the *semantic* elements most relevant to us. You can research the others e.g. <article>, <aside>, <details>, <figcaption>, <figure>, <mark>, <summary>, <time> ... et al

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
  <header>
    <h1>Hello World! I am a developer.</h1>
  </header>
    <main>
      <p>Hello, is it me you're looking for?</p>
    </main>
  </body>
</html>
```

Nav

The **<nav>** element is where we can place navigation links to help people navigate our page and or our website overall.

These often take the form of menu bars, but we can have tables of content, and indexes too.

You can also of course have more than one. Examples could be a **topnav** for the top of your page and **navbar** for the bottom or even to overlay by placing it on top of on image.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My first page</title>
  </head>
  <body>
    <header>
    <nav>
        <a href="#home">Home</a>
        <a href="#about">About</a>
        <a href="#portfolio">Portfolio</a>
        <a href="#contact">Contact</a>
    </nav>
    </header>
  </body>
</html>
```

Main

BLACK CODHER
CODING PROGRAMME

The **<main>** tag is for the primary content on your page - the central topic being covered.

It should be unique to that HTML file. We should not put anything that is present elsewhere including in other documents e.g. sidebars, navigation links, copyright, logos or favicons, and search bars etc.

Understandably, you can't duplicate it so it should only be used once per .html doc.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Hello World! I am a developer.</h1>
    <main>
      <p>Hello, is it me you're looking for?</p>
    </main>
  </body>
</html>
```

Section

The **<section>** element marks an area of your document, either separating things generally, or indicating something particular expanding upon it being semantic.

It is usually used for grouping content with a common theme, in most cases preceded by a heading.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Hello World! I am a developer.</h1>
    <section>
    <h2>Client Testimonials</h2>
    </section>
  </body>
</html>
```

Footer

Fittingly, the final line we'll look at is a **<footer>** element, which is where you normally include things like contact details, copyright, social media links and so on. It's of course semantic.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <main>
      <p>Hello, is it me you're looking for?</p>
    </main>
   <footer>
    <p>Site designed by Black Codher</p>
   </footer>
  </body>
</html>
```

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

Pop Quiz

BLACK CODHER
CODING PROGRAMME

HTML is ___ a programming language and stands for: H___t___ M____ L_____.

Do people directly see what's in the head or only what's in the body?

What are semantic elements? BONUS: Feel free to give 1 or more examples.

BLACK CODHER

CODING PROGRAMME

Time to take a mini break

Heading

Now it’s time for a **<heading>**; these come in a very big size (h1) all the way down to very small (h6). They have meaning.

This meaning is particularly important for those who are visually impaired and other disabled users who may use tools such as screenreaders to navigate the internet.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
      <p>Hello, is it me you’re looking for?</p>
    </main>
  </body>
</html>
```

Paragraph

BLACK CODHER
CODING PROGRAMME

Paragraphs are abbreviated to **<p>** tags. They also have meaning.

Browsers tend to automatically add a single blank line before and after each p element.

You can also use **
** which stands for line **br**eak. It has no closing tags - it is a standalone tag.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
      <p>Hello, is it me you’re looking for?</p>
    </main>
  </body>
</html>
```

Anchor (link)

Let's change things up. This line is now an **<a> anchor** tag - which anchors itself with a link in the form of a *Uniform Resource Locator* (*URL)*. We could also have internal *deeplinking* as we did with the navigation.

In this case, **href** contains the website that we want the user to visit when they click on the link. It stands for hypertext reference.

The *attribute* **target** defines where we want the link to open; **"_blank"** means in a new tab.

The text following it, between the last bracket of the opening tag and the closing tags is what will display on the page for that link.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Noname Developer</h1>
    <main>
      <a href="https://www.youtube.com/watch?v=Wk5DqvY6Eww"
        target="_blank">
        Hello, is it me you're looking for?
      </a>
    </main>
  </body>
</html>
```

BONUS

BLACK CODHER
CODING PROGRAMME

As you might have noticed, links can be styled according to the different states they are in.

The 4 link states are:

- a:link - an unvisited link
- a:visited - a visited link
- a:hover - moving the mouse over it
- a:active - when it's being clicked

Feel free to play around with this.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My first page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
    <div>
      <a
        href="https://www.youtube.com/watch?v=AiC7ZX5K9L4"
        target="_blank">
        Hello, is it me you're looking for?
      </a>
    </div>
  </body>
</html>
```

Div

Next up is a ** ** element. It short for divider.

It is not *semantic*.

This means it has no internal meaning. It can be styled to look like other elements, but it does not come with built in browser styles and accessibility features bundled. Due to this, it should be considered an element of last resort.

Divs often act as a container for other HTML elements. In HTML, most elements are defined as block level or inline elements. Block level elements like div start from a new line.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Noname Developer - Portfolio</title>
  </head>
  <body>
    <h1>Hello World! I am a developer.</h1>
    <div>
      <p>Hello, is it me you're looking for?</p>
    </div>
  </body>
</html>
```

Span

Similarly another example of a non-semantic element is the **** element: an inline element that is often used as a container for some text.

Inline elements are normally displayed without line breaks.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My first page</title>
  </head>
  <body>
    <h1>Hello World!</h1>
      <p><span>Hello, is it me you’re looking
for?</span></p>
    </body>
</html>
```

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

Task 2: Fleshing out our HTML

Let’s have a roleplay - securing the bag!

If there is a mentor or volunteers in your breakout room, they will act as easygoing clients who want to hire you as a talented dream team to build a one pager. If not, you can come up with your own brief or ask me.

Discuss things briefly to agree what is best for this simple starting point, then start adding to the body of your **index.html** file e.g. **a header? Headings? Navs? Main? Sections? Non-semantic elements? Paragraphs? Links? A footer?** Again, as with any dev job, this is an open book setting!

BLACK CODHER

CODING PROGRAMME

Time to take another break

BLACK CODHER

CODING PROGRAMME

CSS 101

Learning objectives

1. Understand the difference between the 3 ways to insert CSS - external style sheets, internal style sections, and inline style attributes
2. Understand what cascading means in terms of their order as you have a go at using each of them
3. Understand the difference between the 3 simple selectors in CSS - names, ids, and classes

The 3 ways of inserting style sheets

We have 3 ways to insert a style sheet:

- External CSS
- Internal CSS
- Inline CSS

An external style sheet changes the look of a whole website just with that 1 file.

An internal style section would be used for any single HTML page with a distinct aesthetic.

An inline style attribute can be used to apply specific designs to a single element.

External style sheets

BLACK CODHER
CODING PROGRAMME

External style sheet files must be linked to with the **<link rel>** element inside **head**. You can place it anywhere, but that's best practice.

These can be written in any text editor, and must be saved with a .css file extension. They should not contain any HTML tags, but they reference elements.

Ideally, the structure of content should be kept separate from the presentation of it - in other words to each their own with HTML and CSS.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
   <title>My first page</title>
    <link rel="stylesheet" href="style.css">
   </head>
   <body>
     <h1>Hello World!</h1>
     <main>
       <p>Hello, is it me you're looking for?</p>
     </main>
   </body>
</html>
```

Internal style sections

Internal styles are defined within the **<style>** *element*, inside the **<head>** of an HTML page. We can think of them being embedded. Take note of the 2 approaches to indentation here.

In the example to the right, we're styling the body section and and heading tags before we see all of that within the body element itself.

What you see here with internal styles is similar to how things look on external ones, except naturally you won't have any actual HTML elements, save for references to them.

```
<head>
<style>
  body {
    background-color: white;
    margin-left: 10px;
       }

  h1 {color: aqua;}
  h2 {font-family: Calibri;}

</style>
</head>
<body>
<h1>This is the biggest heading.</h1>
<h2>This is the next largest heading</h2>
</body>
```

Inline style attributes

To use inline insertion, add the **style** *attribute* to the relevant *element*. It can contain any CSS property applicable to the presentation / design you're aiming to create.

As mentioned it's not best practice to use CSS insertion methods which don't keep things separate i.e. inline and internal. Also in terms of keeping our code clean, how tidy do you think things would look if we had several properties and property values on an element?

That being said, at times you may find it's more suitable to what you want to achieve.

```
<head>
</head>
<body>
  <h1 style="color:black;">This is a heading
  with inline styling to set it as black</h1>
</body>

<!-- DRY      Don't
              Repeat
              Yourself

     WET      We
              Enjoy
              Typing
-->
```

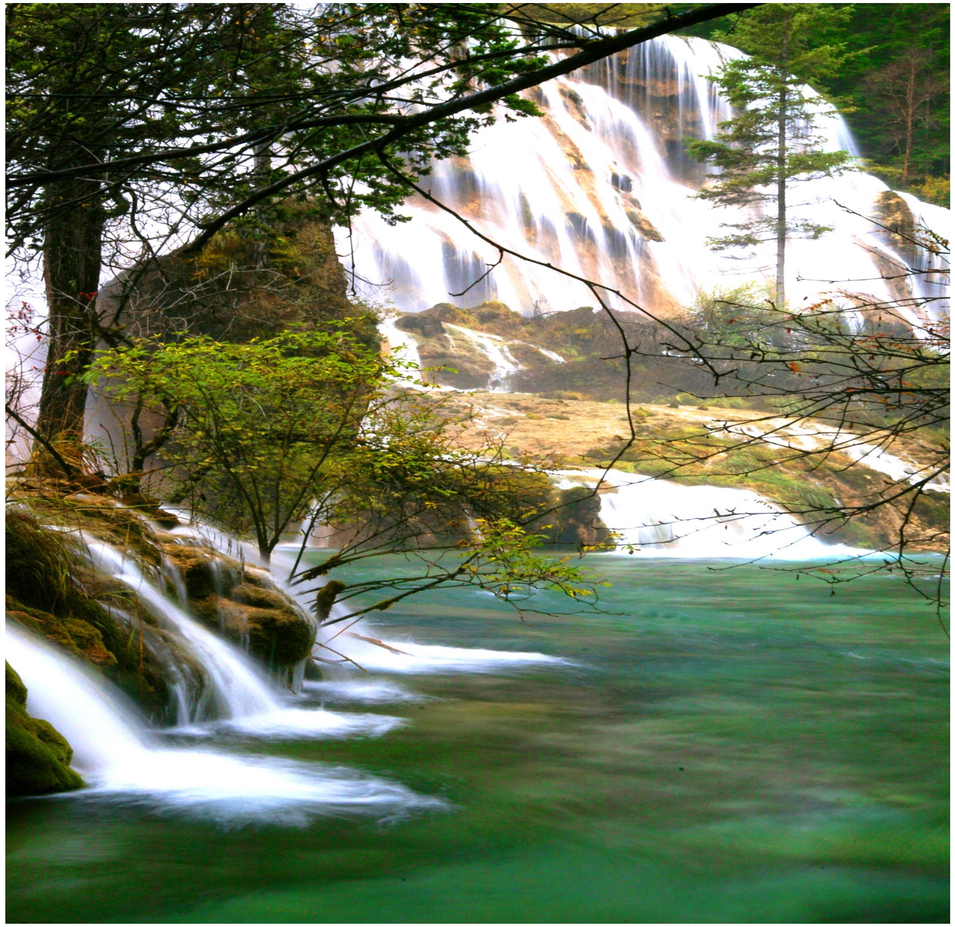
What does Cascading mean in CSS?

BLACK CODHER
CODING PROGRAMME

All the styles in a web page should **cascade** by the following rules, where number one has top priority:

1. Inline style attribute (inside any HTML element)
2. Internal style section (in head element)
3. External style sheet (linked)
4. Browser default

So inline styles take precedence, thus it overrides our external and internal styles as well as browser defaults. That being said, it shouldn't be our first choice if we're designing with best practice in mind.



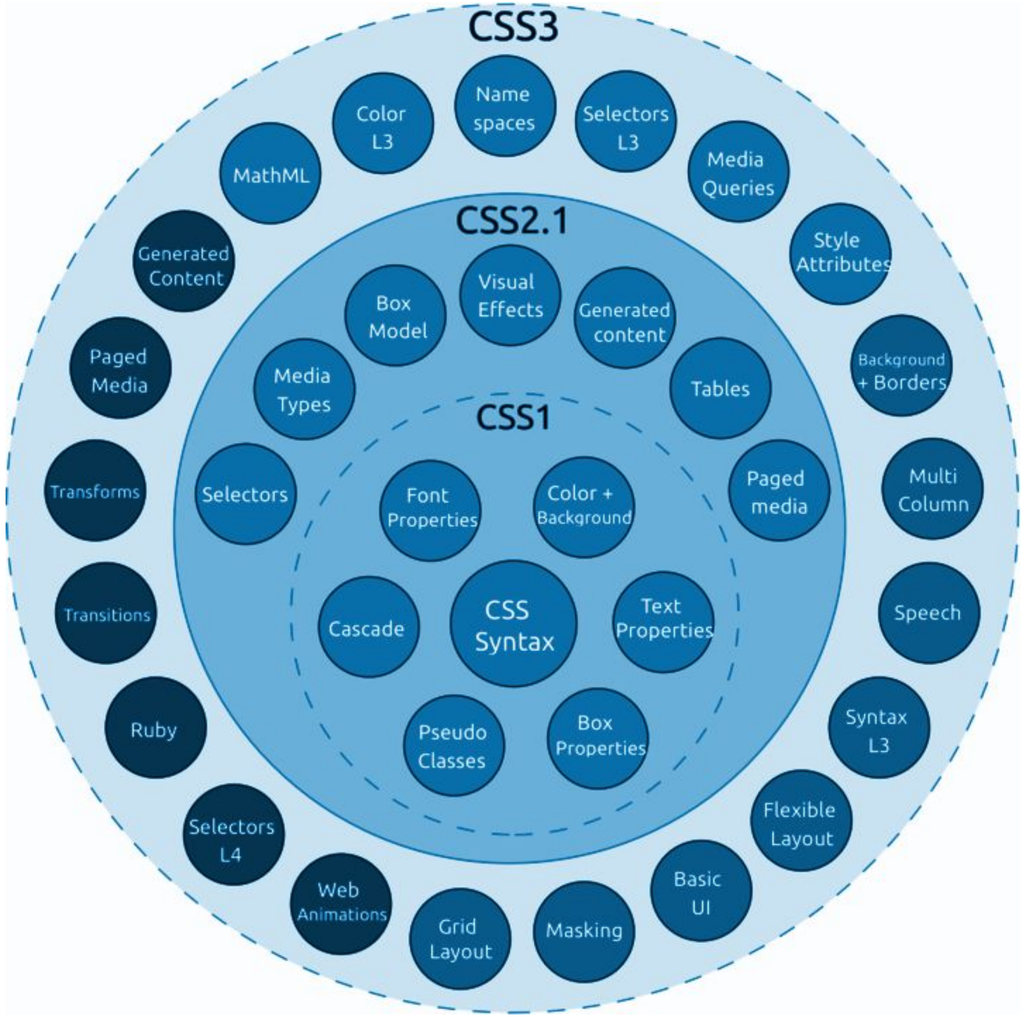
3 basic ways of selecting elements

CSS selectors are used to "find" (or select) the HTML elements you want to style. You can select more than one item at the same time.

We'll focus on simple selectors as they are most commonly used:

- Name
- ID
- Class

The last two are attributes.



Name Selectors

The name element selector selects HTML elements based on the element name.

Here, all <h3> elements on the page will be font size 30px and center-aligned.

```
<!-- index.html -->
<head>
   <link rel="stylesheet" … href="style.css">
</head>
<body>
  <h3>H3H3</h3>
</body>
```

```
/* style.css */
 h3 {
      font-size: 30px;
      text-align: center;
    }
```

ID Selectors

The **id selector** uses the id *attribute* of an HTML element to select a specific .

IDs are attributes which do just that: they uniquely identify HTML elements. They must be unique within a document, so you cannot have more than one element with the same id. As such, this selector only selects a single element.

To select an element using id, we put the hash (#) character, followed by the id.

```
<!-- index.html -->
<head>
   <link rel="stylesheet" … href="style.css">
</head>
<body>
  <section id=“portfolio”>
   …
  </section>
</body>
```

```
/* style.css */
#portfolio {
      font-weight: 900;
      border-radius: 45px;
}
```

Class Selectors

The **class selector** also uses an *attribute*.

Classes can be a useful way of creating a default design for a group of similar elements that you always want to display in the same way. Ties in nicely with the DRY and WET principles that were mentioned just a few slides earlier with regards to opting for clean optimised code.

We use a periodt (.) 😉 followed by the class name to select elements with that specific class.

```
<!-- index.html -->
<!DOCTYPE html>
<head>
   <link rel="stylesheet" … href="style.css">
</head>
<body>
  <section class=“portfolio-pictures”>
  ...
  </section>
</body>
```

```
/* style.css */
.portfolio-pictures {position: absolute;}
```

Checkpoint!

How are you feeling?

RED - I have no idea what you're talking about

YELLOW - I have some questions but feel like I understand some things

GREEN - I feel comfortable with everything you've said

BLACK CODHER

CODING PROGRAMME

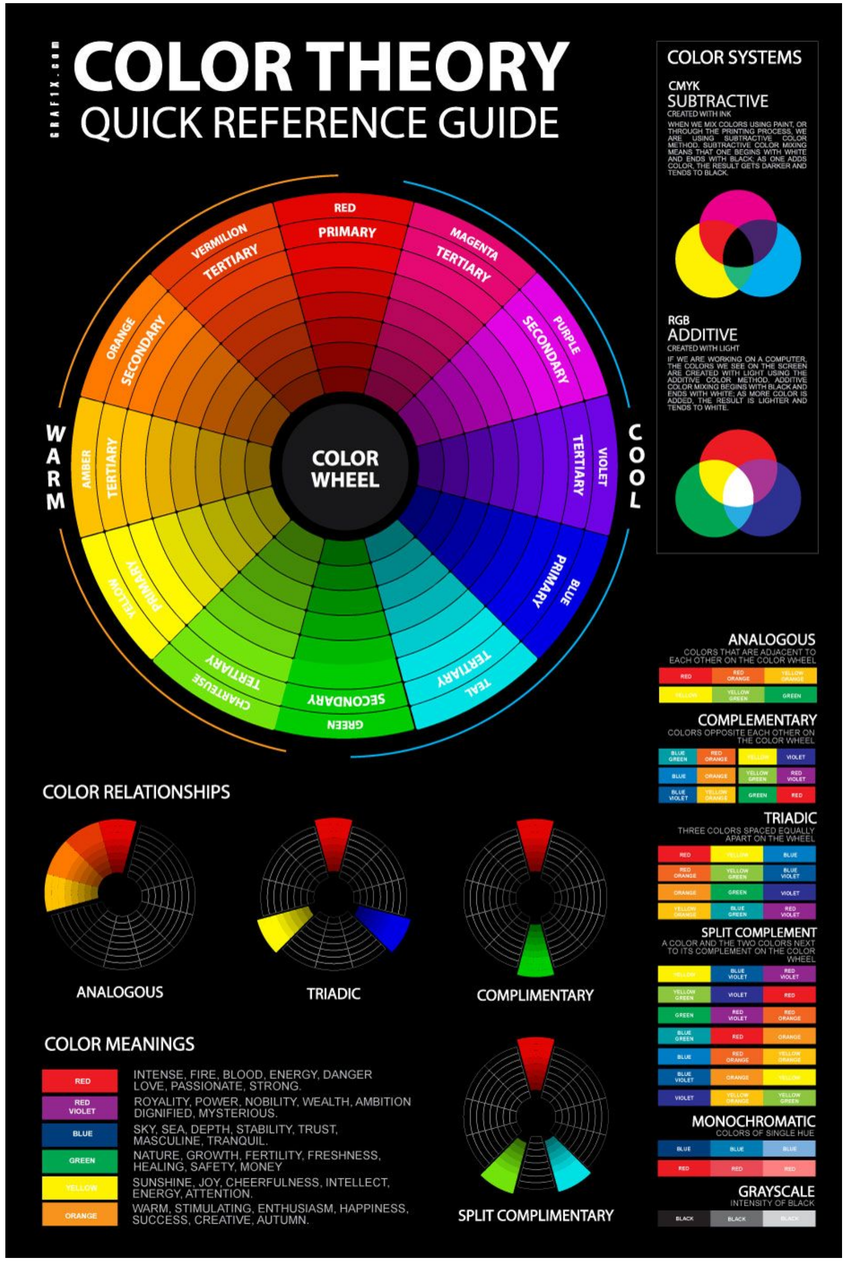
One more brief break

BLACK CODHER

CODING PROGRAMME

Colour

Source: User Testing



Understanding colour theory - one of a few concepts outside of development that we'll touch upon - can be helpful for designing sites. It's also relevant with regards to accessibility.

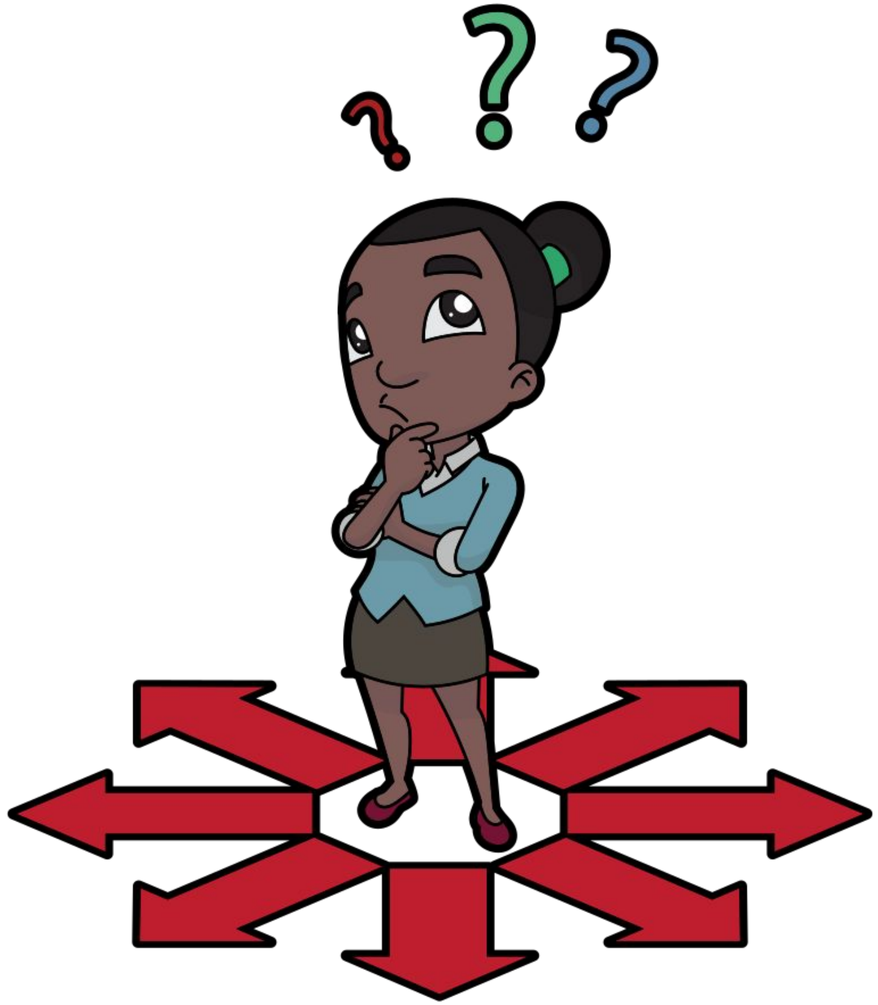
BLACK CODHER
CODING PROGRAMME

Task 3: Styling out our CSS

BLACK CODHER
CODING PROGRAMME

1. Create your external **style.css** file if you have yet to do so, then link it with the **<link rel>** element in your **<head>** tags.
2. Have a go at adding CSS rules, as well as trying things out with **internal** style sections and **inline** style attributes.
3. Test at least **3 CSS selectors** and share what you did with your group e.g. explaining colour choices etc.

Again, please feel free to refer to the slides to check what we've covered so far.



BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Any questions?

NÍ YÓ

BLACK CODHER

CODING PROGRAMME

CODING BLACK FEMALES

Well done! Get some rest and see you soon!

NÍ YÓ