ual: creative computing institute

Peckham DAZ

Week 3 - Accessible Web Development

These slides were created by Dan Hearn - AL at CCI

ual: creative computing institute

Recap

HTML, CSS, JavaScript

Front-end web languages

Front-end Web Languages

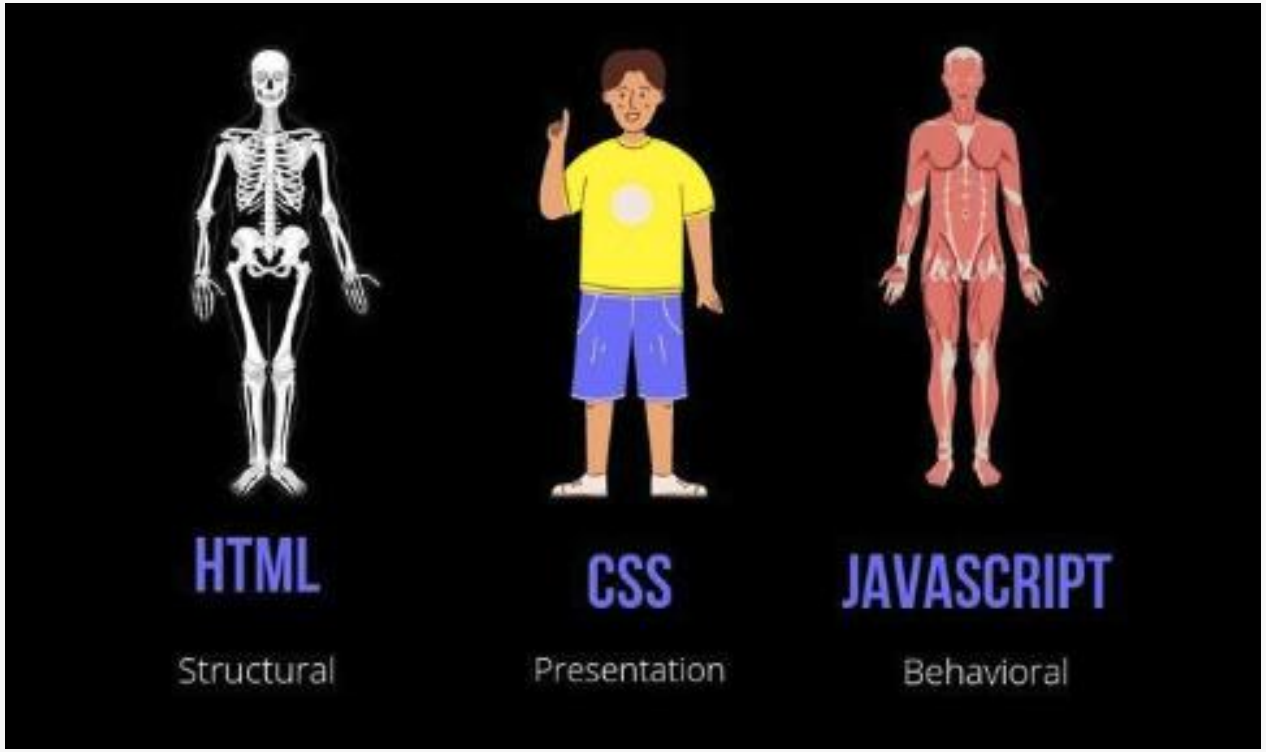
Front-end refers to the visible user interface that a user interacts with.

It’s made up of 3 languages:

HTML: The structure and layout of a page

CSS: The styling and look of a page

JAVASCRIPT: The functionality of a page

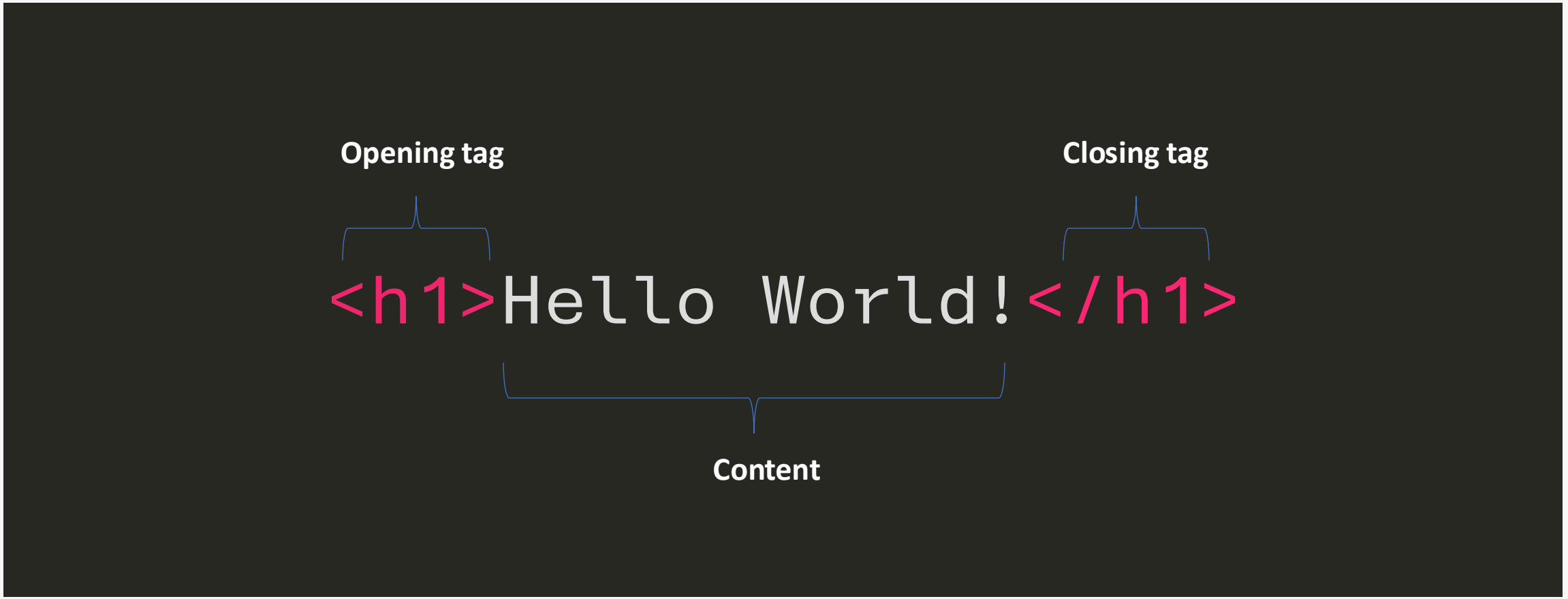


Hyper Text Markup Language (HTML)

- Describes the structure and meaning of a webpage
- Made up of tags (also called elements)
- Denotes text and images/video
- Can be manipulated by CSS and JavaScript
- It forms the foundation of the DOM (Document Object Model)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title> This is the tab title in your browser </title>
</head>
<body>
  Everything inside here gets rendered in your browser!
</body>
</html>
```

Tags/Elements



Tags...

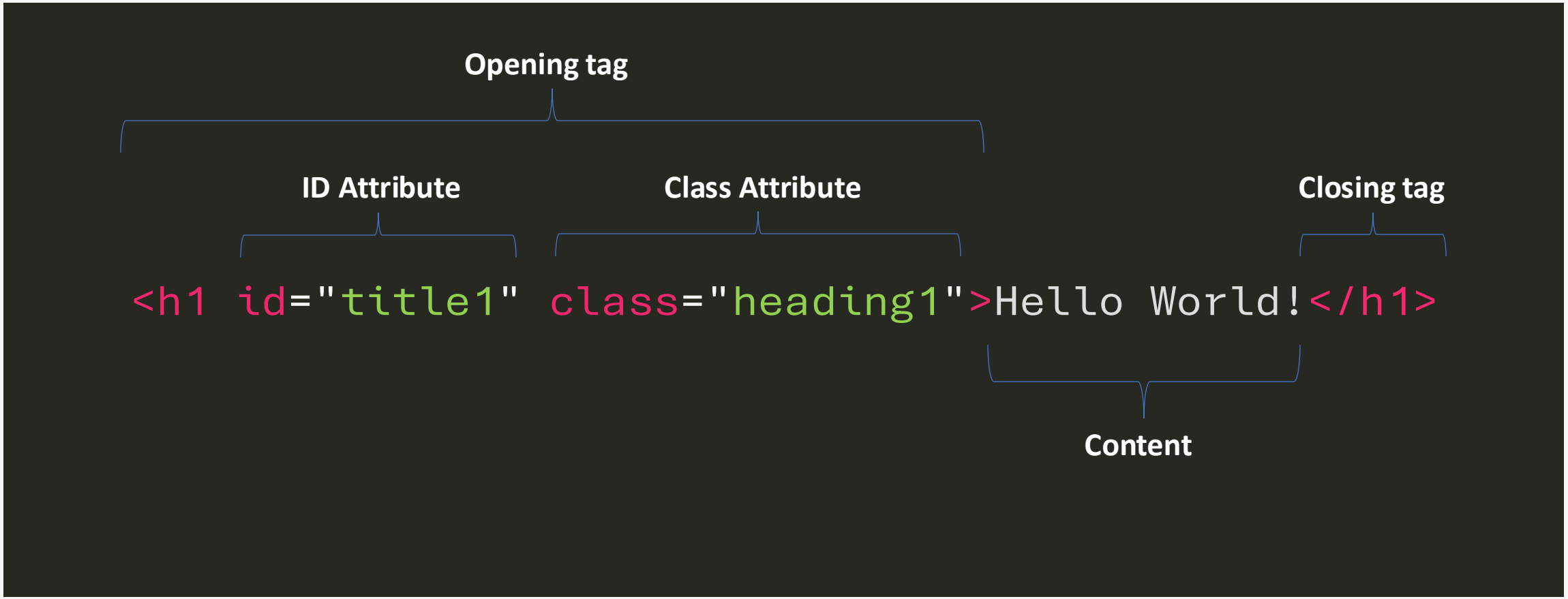
There are loads! We'll mainly be using these...

- Heading `<h1></h1>…<h6></h6>`
- Paragraph `<p></p>`
- Div `<div></div>`
- Button `<button></button>`
- Form `<form></form>`
- Inputs `<input></input>`
- Input label `<label></label>`
- Anchors (page links) `<a></a>`
- Image `<img>`
- Script `<script></script>`
- Option `<option></option>`
- Select `<select></select>`
- Textarea `<textarea></textarea>`
- Link (Stylesheet) `<link>`
- Unordered List `<ul></ul>`
- List Items `<li></li>`

HTML Tag Cheatsheet!

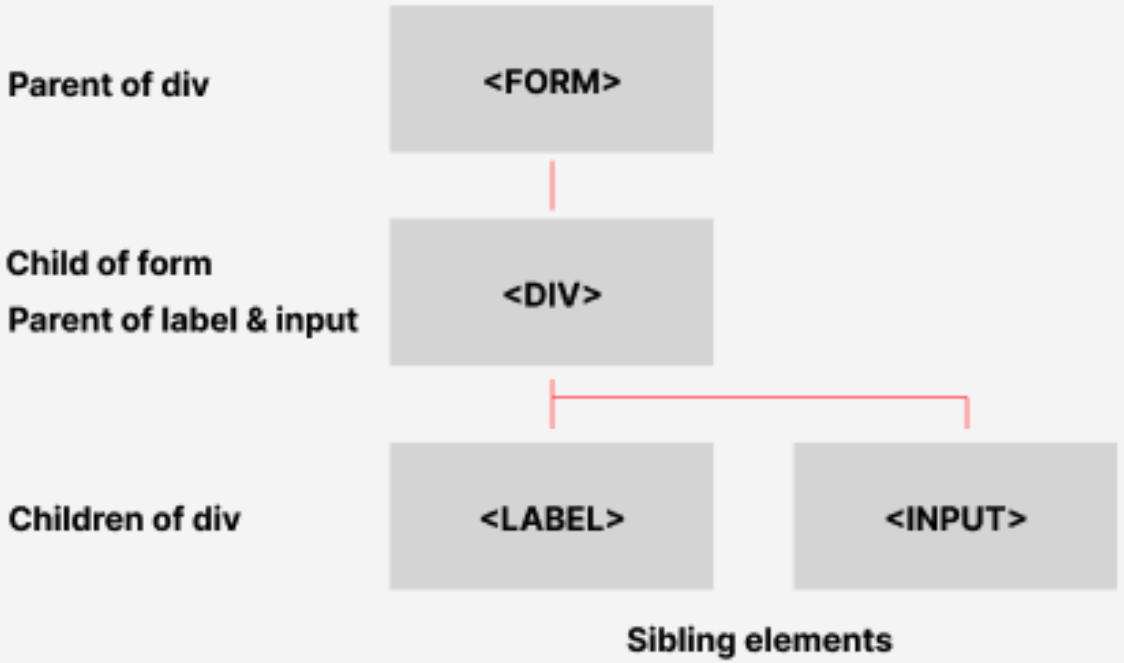
HTML Attributes

List of Attributes



Parent/child Relationship...

…Forms the DOM Family Tree



Cascading Stylesheets (CSS)

- A stylesheet language used to style HTML elements
- Can add some basic functionality to a webpage
- Used to create the 'branding style' of your webpage
- Can be used to improve accessibility and user experience.

```
button {
  background-color: #4CAF50;
  border: none;
  color: white;
}
```

CSS Selectors

Selectors allow you to target specific HTML tags to style them.

Core selectors:

- All selector: * {}
- Tag selectors: button {}
- ID selectors: #id-name {}
- Class selectors: .class-name {}

List of other selectors

```css
/* All Selector */
* {
  font-family: 'Courier New', Courier, monospace;
}
/* HTML Tag selector */
button {
  background-color: #4CAF50;
  color: white;
}
/* ID selector */
#submit-btn {
  border: 1px solid #000;
  padding: 10px;
}
/* Class Selector */
.button {
  font-size: 16px;
  cursor: pointer;
}
```

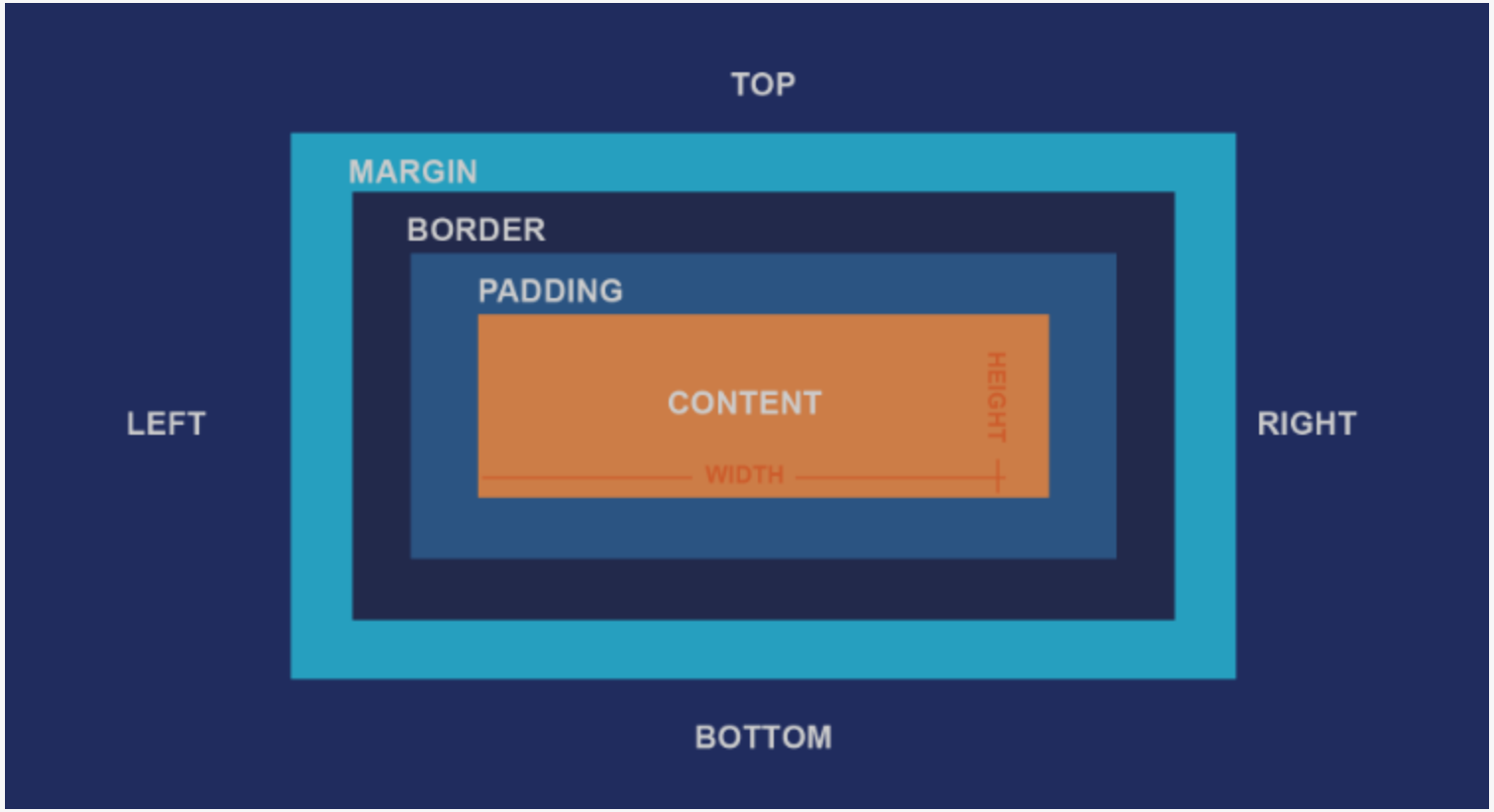
CSS Properties

- CSS properties specify how HTML elements should be displayed
- manage the positioning and spacing of elements on a webpage
- They allow for customisation of colors, fonts, and backgrounds
- Properties can change styles dynamically, improving user experience and accessibility

There are hundreds of CSS properties

```
button {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
  border-radius: 8px;
  transition-duration: 0.4s;
}
```

Properties - Box Model



Use inspector tools to help understand this

Cascade, Specificity, and Inheritance

- **Cascade** refers to how a CSS file is read from top to bottom. If there are two identical selectors with different properties, the second selector will override the properties of the first.
- Some selectors are more **specific** than others - meaning that some can override others.
- Some properties are **inherited** from parent elements

JavaScript (JS)

- A scripting language that adds functionality to a webpage
- Can be used to manipulate the DOM, HTML, and CSS
- Allows us to create a dynamic webpage which improves accessibility and user experience

```
const button = document.getElementById('button1');

function handleClick() {
  console.log('Hello, world!');
}

button.addEventListener('click', handleClick);
```

Variables

- Variables are containers for pieces of data - numbers, strings etc
- Must start with a letter, underscore (_) , or dollar sign ($)
- Declared using **var**, **let**, and **const.**
- **Const** cannot be edited once declared, but **var** and **let** can change throughout your program.

```
let name = "John"; // let keyword
const age = 30; // const keyword (constant value)
var isMarried = false; // var keyword
(global/function scope)
```

Scope

- **Global scope:** vairables that can be accessed from anywhere in the program
- **Function scope**: variables that can be used within a function
- **Block variables:** variables that can be used within a code block

```
// Global Scope
var globalVar = "I am global";

function testScope() {
  // Function Scope
  var functionVar = "I am inside a function";

  if (true) {
    // Block Scope
    let blockVar = "I am inside a block";
    console.log(blockVar); // Accessible here
  }
// console.log(blockVar); // Error: blockVar is not defined
}

testScope();

// console.log(functionVar); // Error: functionVar is not defined
console.log(globalVar); // Accessible here
```

Core Datatypes

- **Boolean**
- **String**
- **Number (Integers & floating points)**
- **Array**
- **Object**
- Undefined
- Null

```
let stringVar = "This is a string"; // String
let numberVar = 42; // Number
let booleanVar = true; // Boolean
let nullVar = null; // Null
let undefinedVar; // Undefined
let objectVar = { key: "value" }; // Object
let arrayVar = [1, 2, 3, 4, 5]; // Array
```

Operators

Arithmetic

Addition(+), Subraction(-), Multiplication(*), Division(/)

Comparison

Equal to (==), Not equal to(!=), Strict equal to(===), Greater than(>), Less than (<)

Logical

AND(&&), OR(||), NOT (!)

```
let a = 10;
let b = 5;
let add = a + b; // Addition
let subtract = a - b; // Subtraction
let multiply = a * b; // Multiplication
let divide = a / b; // Division
let remainder = a % b; // Remainder
let exponent = a ** b; // Exponentiation
let isEqual = a == b; // Equality
let isStrictEqual = a === b; // Strict Equality
let isNotEqual = a != b; // Inequality
let isStrictNotEqual = a !== b; // Strict Inequality
let greaterThan = a > b; // Greater than
let lessThan = a < b; // Less than
```

Expressions

Expressions are code snippets that **evaluate to a value**, such as a combination of variables, operators, and function calls, which can be used to **perform calculations, assign values, or produce outputs**.

```
let sum = 5 + 3; // Addition
let product = 4 * 2; // Multiplication
let greeting = "Hello, " + name + "!"; // String concatenation
```

Conditional Statements

Allows your program to make decisions and perform actions based on wether a given condition is **true** or **false.**

```
if (a > b) {
  console.log("a is greater than b");
} else if (a < b) {
  console.log("a is less than b");
} else {
  console.log("a is equal to b");
}
```

Functions

Functions are **resuable blocks of code** that perform a specific task. Like a machine that takes an input, does something, and produces an output.

```
// Function to add two numbers
function addNumbers(a, b) {
  return a + b;
}
// Calling the addNumbers function
addNumbers(5, 10);
```

Arrays & Loops

- Arrays are data structures that allow you to store **multiple pieces of data** within a single variable.
- Array items are **indexed starting from 0** and are separated by a comma.
- Loops are constructs that allow you to **execute a block of code repeatedly** until a specific condition is met.
- For loops have a **counter variable** with an intial value, **a condition**, and **a counter** incrementer/decrementer.

```
// Array
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // Accessing 0 array element

// For loop
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

DOM Manipulation

We can use JavaScript's built-in document methods to:

- **Select HTML elements** based on element name, id, class, or using CSS selectors
- **Append (add) or remove HTML elements**
- **Listen for document events** such as click, hover, scroll, keyboard events

Let's use JavaScript DOM manipulation on my protfolio

```
const button = document.getElementById('button1');

function handleClick() {
  console.log('Hello, world!');
}

button.addEventListener('click', handleClick);
```

ual: creative computing institute

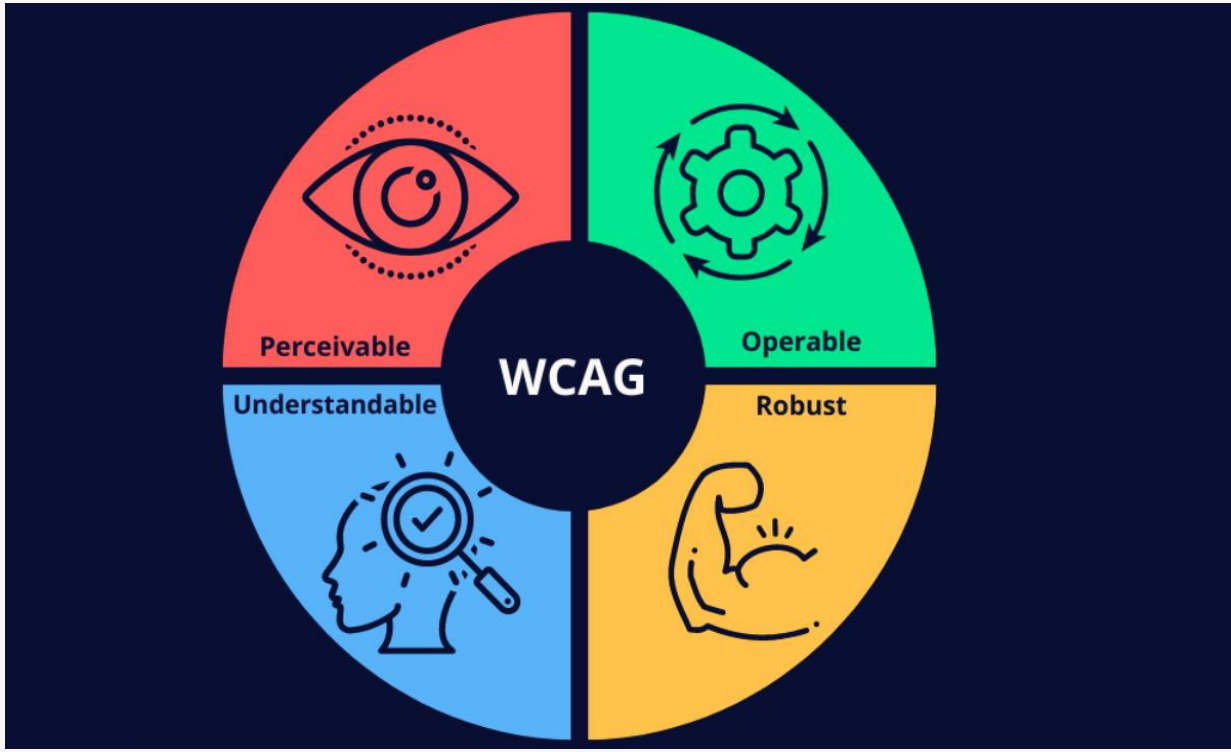
Inclusive Web Design

- Participatory Design Methods
- Accessibility Tools
- Responsive Web Design

WCAG Guidelines

Web content accessibility guidelines

- An accessibiliy guideline maintained by W3C
- Public websites must meet this standard
- There are 3 ratings **A (fail), AA (required), AAA (best)**
- Tools such as **Axe, Wave, Tenon, SiteImprove** can help you test
- **User testing** with **real people** is extrememly important!



Percievable

- Provide alternative text for non-text content
- Provide captions for video
- Create content that can be presented in different ways *e.g text, video, audio*

Operable

- Allow keyboard-only navigation
- Create space around text
- Use labels and headings
- Make navigation as simple as possible
- Test with screen readers, screen magnifiers, voice commands

Understandable

- Make text as clear as possible
- Be clear about how things work
- Make content predictable
- Find ways to help users who are lost

Robust

- Works on different devices and browsers
- Don’t use exclusive technology *e.g plugins*
- Do not use exploitative technology

Participatory Design

Directly involve users into the design process to ensure their needs inform the design and development of software.

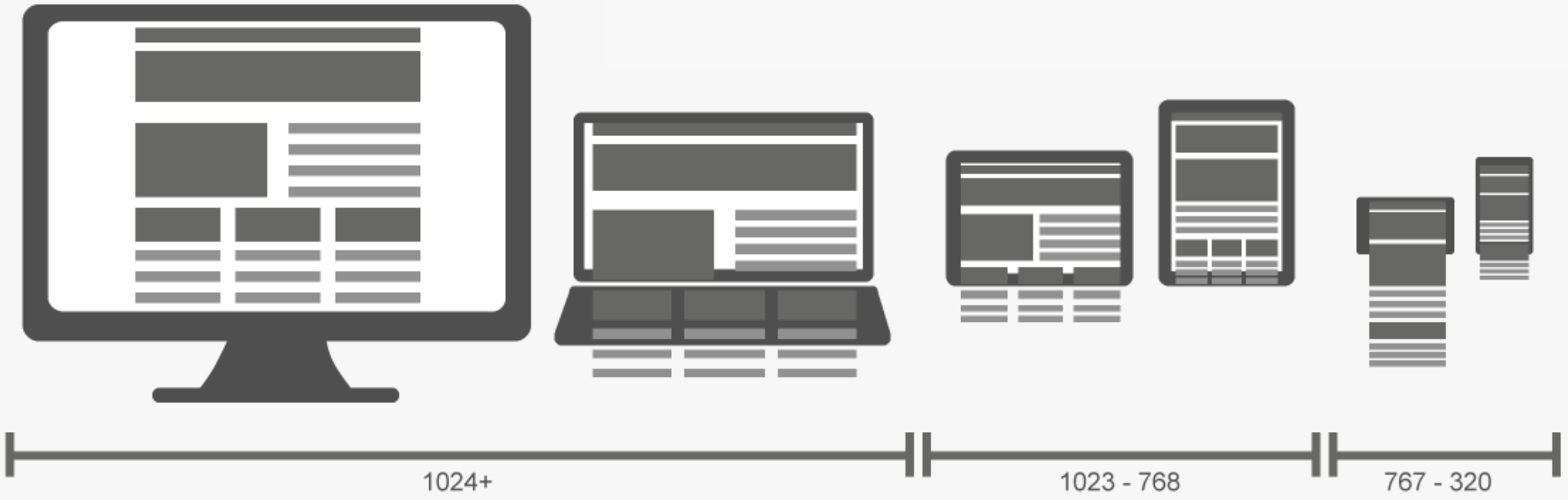
Applying PD to Accessibility

Inclusive Approach: Treat users with disability as experts of their own experience

Early Involvement: Ensure accessibility needs are addressed from the onset of design

Methodologies: Conduct continuous user research and accessibility walkthroughs to refine designs iteratively

Responsive Web Design



Coding Accessible Websites

Resources for accessible web coding best practices:

Accessibility guide from MDN web docs

FreeCodeCamp article

Additional Learning resources:

Extracirricular codealong course from freecodecamp - It's great!

Learn the Web accessibility series - also great!

Accessible HTML

- Use semantic HTML tags
- Provide alternative text for images using 'alt' attribute
- Use labels with form elements
- Create accessible links
- Use heading tags appropriately
- Include captions for multimedia

[Accessible HTML cheatsheet]

```
<body>
  <header>
      <img src="logo.png" alt="Company Logo">
      <nav>
          <ul>
              <li><a href="#main-content">Skip to main content</a></li>
              <li><a href="about.html">About Us</a></li>
              <li><a href="services.html">Services</a></li>
              <li><a href="contact.html">Contact</a></li>
          </ul>
      </nav>
  </header>

  <main id="main-content">
      <h1>Welcome to Our Company</h1>
      <section>
          <h2>Our Mission</h2>
          <p>Our mission is to provide high-quality products and services to our customers.</p>
      </section>
      <section>
          <h2>Contact Us</h2>
          <form action="/submit" method="post">
              <div>
                  <label for="username">Username:</label>
                  <input type="text" id="username" name="username">
              </div>
              <div>
                  <label for="email">Email:</label>
                  <input type="email" id="email" name="email">
              </div>
              <div>
                  <button type="submit">Submit</button>
              </div>
          </form>
      </section>
  </main>

  <footer>
      <p>&copy; 2024 Our Company</p>
  </footer>

  <button aria-expanded="false" aria-controls="menu">Menu</button>
  <video controls>
      <source src="video.mp4" type="video/mp4">
      <track kind="captions" src="captions.vtt" srclang="en" label="English">
  </video>
</body>
```

ual: creative computing institute

Accessible CSS

- Use focus and hover states
- Ensure good colour contrast
- Use responsive layouts with media queries
- Create flexible and adaptive layouts
- Use readable font sizes
- Ensure a consistent look and feel

```css
.btn-submit {
  padding: 10px 40px;
  color: white;
  font-weight: bold;
  background-color: green;
  border-radius: 5px;
  border: 2px solid green;
  cursor: pointer;
}
.btn-submit:hover {
  color: green;
  background-color: white;
}

@media screen and (max-width: 768px) {
  /* Styles for smaller screens */
}
```

Accessible JS

- Create dynamic styling to aid accessibility and user experience
- Handle keyboard events
- Create accessible form validation
- Provide controls for media playback
- Create a dynamic user interface

```css
.btn-submit {
  padding: 10px 40px;
  color: white;
  font-weight: bold;
  background-color: green;
  border-radius: 5px;
  border: 2px solid green;
  cursor: pointer;
}
.btn-submit:hover {
  color: green;
  background-color: white;
}

@media screen and (max-width: 768px) {
  /* Styles for smaller screens */
}
```

Overview

1. Follow the WACG standards
2. Use accessibility tools & test with real people
3. Design with an accessibility mindset
4. Adapt an iterative design & development process
5. Accessible code is just as important as accessible design