Document Object Model

DOM

User Interfaces

420-WC4-AB

Anatomy of a Website

- Your Website = Content + HTML + CSS

```
<p>Hello World</p>
```

Hello World

- "Hello World" is your content
- Having your content in a `<p>` tag makes it look like a paragraph
  - This is the HTML
- Making the font green and bold
  - This would be the CSS

HTML review

- HTML is a markup language
  - `<tag></tag>`
- Elements must be properly nested
- All element must close (closing tag or self closing)

IDs

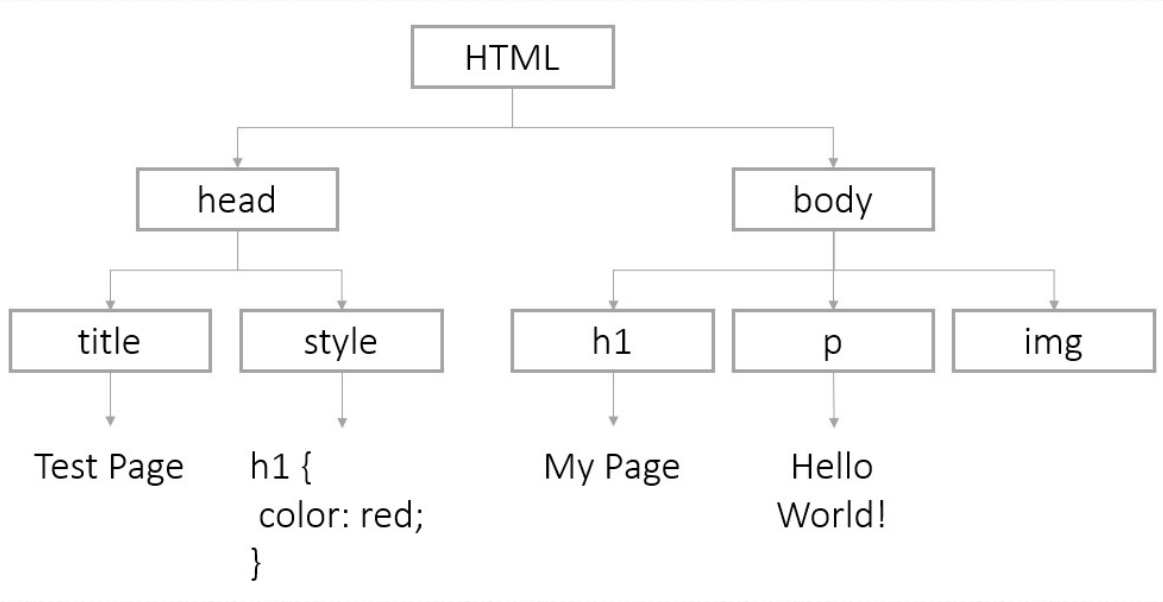
- The importance of IDs was shown slightly in CSS and that will show even more so now in JavaScript.
- They should only ever appear **once** on a webpage
- You can only have 1 ID per element
- The attribute can be added to any HTML element
  - `<p id="intro">Hello World</p>`
  - The # is how you target ids in CSS.

Document Object Model - DOM

- When a web page is loaded, the browser create the Document Object Model
- An HTML page is constructed as a tree of Objects. This tree structure is define by the DOM (Document Object Model)
- Objects in the DOM tree may be accesses and manipulated by using built-in methods on the objects.
  - Create any new element or attribute
  - Modify or remove existing element and attributes
  - Create and interact with events

Sample DOM Tree

```
<!DOCTYPE html>
<html>
<head>
  <title>Test Page</title>
  <style> h1{ color:red; } </style>
</head>
<body>
  <h1>My Page</h1>
  <p>Hello World!</p>
  <img src="img/hi.png" alt="hi" />
</body>
</html>
```



Accessing the DOM

- Browsers automatically loads the content of a webpage into a **Document** object which serves as the entry point to your content.
- All DOM methods use the Document object and use dot notation

```
document._______
```

- Change the content tree any way you want.
- Build an HTML document from scratch.
- Access or replace any existing DOM nodes (HTML elements in the DOM).

Accessing the DOM

- You can use the access DOM nodes using their IDs

```
document.getElementById(elemId)
```

- You can use the access DOM nodes using tag names

```
document.getElementsByTagName(elemTagType)
```

- You can use the access DOM nodes using class names

```
document.getElementsByClassName(className)
```

- You can use the access DOM nodes CSS Selectors)

```
document.querySelector(cssQuery)
document.querySelectorAll(cssQuery)
```

Accessing the DOM with IDs

- You can use the access DOM nodes using their IDs

```
document.getElementById(elemId)
```

- The getElementById(*elemId*) method returns the element that has the id attribute equal to *elemId* (must be exACtlY the same)
- HTML

```
<img src="img/hi.png" alt="hi" id="waveImg" />
```

- JavaScript

```
let imgHi= document.getElementById("waveImg");
```

Accessing the DOM with Tag Name (or class)

- You can use the access multiple DOM nodes using their HTML element tag name or class

```
document.getElementsByTagName(tagName);
document.querySelector("ul.txt_bold li");
document.querySelectorAll(" ul.txt_bold li");
```

- HTML

```
<ul class="txt_bold">
    <li>Daisy</li>
    <li class="txt_bold">Tulip</li>
</ul>
<p class="txt_bold" id="intro">Hello World</p>
```

Single Elements vs Array of Elements

- Methods that return 1 element.

```
document.getElementById("intro");
document.querySelector("ul.txt_bold li");
```

- Methods that return a collection of elements, similar to an array.

```
document.getElementsByTagName("li");
document.getElementsByClassName("txt_bold");
document.querySelectorAll("ul.txt_bold li ");
```

Accessing Array of Elements

- JavaScript – Tag Name

```
let listItems = document.getElementsByTagName("li");
for (let i = 0; i < listItems.length; i++){
     let listItem = listItems[i];
}
```

- JavaScript – Class Name

```
let classItems = document.getElementsByClassName("txt_bold");
for (let i = 0; i < classItems.length; i++){
    let classItems = classItems[i];
}
```

Attributes

- You can add, edit and remove attributes of DOM nodes using dot notation.

```
<img src="img/hi.png" alt="hi" id="waveImg" />
```

- Change the src attribute we could…

```
let imgHi = document.getElementById("waveImg");
let currentSource = imgHi.src;  // http://site.com/img/hi.png
imgHi.src = "img/waveHello.png";
```

- You can also chain your actions as one

```
document.getElementById("waveImg").src = "img/waveHello.png";
```

Attributes – Getters and Setters

- Still using `<img src="img/hi.png" alt="hi" id="waveImg" />`
- Use the global getAttribute, setAttribute, removeAttribute methods

```
let imgHi = document.getElementById("waveImg");
// get the current value of src attribute
imgHi.getAttribute("src");
// set a new value for the src attribute
imgHi.setAttribute("src", "img/waveHello.gif");
// remove the src attribute
imgHi.removeAttribute("src");
```

Attribute - Style

- To change the CSS of an element - modify the style attributee
- You can alter a single style element we use the style object

```
CSS ➔ body { color: red; }
```

- JavaScript

```
let pageBody = document.getElementsByTagName("body")[0];
pageBody.style.color = "red";
```

- JavaScript – Multiple styles

```
pageBody.style.cssText = "color:red; font-weight:bold;";
```

Attribute – Style *cont.*

- When working with the style attribute, you are able to change the CSS style properties that have a dash (-) into a camelCase

```
body {
    background-color: grey;
    padding-left: 20px;
}
```

- JavaScript

```
let pageBody = document.getElementsByTagName("body")[0];
pageBody.style.backgroundColor = "grey";
pageBody.style.paddingLeft = "20px";
```

innerHTML

- Each DOM node has an *innerHTML* property with the HTML and content of its children
- You can use the property to retrieve the content of an HTML element
- You can alter the entire content of an HTML
- Elements that are added to your document using innerHTML are not accessible via the DOM

innerHTML

```
let pageBody = document.getElementsByTagName('body')[0];

// overwrite the entire content
pageBody.innerHTML = "<h1>Uh Oh!</h1>";

// or append at the end
pageBody.innerHTML += "<p>We replaced eveyrthing</p>";

// retrieve the content of specific element
let elemContent = document.getElementById('intro').innerHTML;
```

innerText

- Each DOM node has an *innerText* property with the content of its children
- Use the property to retrieve the content of an HTML element as text
- You can alter the entire content of an HTML as plain text

```
pageBody.innerHTML = "<h1>Uh Oh!</h1>";
// ACTUAL TEXT NO MARKUP
// <h1>Uh Oh!</h1>
```

Create a DOM node

- In order to create nodes from scratch (and accessible via the DOM)

```
document.createElement(tagName);
document.createTextNode(text);
parent.appendChild(newChild);
```

Add Image to Body

```
let pageBody = document.getElementsByTagName('body')[0];

// create our image tag with attributes
let newImg = document.createElement('img');
newImg.src = 'http://placekitten.com/g/500/200';
newImg.style.border = '1px solid black';

// add our image to the body
pageBody.appendChild(newImg);
```

Add Paragraph to Div

```
let introDiv = document.getElementById('intro');

// create a paragraph tag with content
let newParagraph = document.createElement('p');
let paragraphText = document.createTextNode('Squee!');
newParagraph.appendChild(paragraphText);

// add our new paragraph to the div
introDiv.appendChild(newParagraph);
```

Questions ?