Week 2

Web Design 2

04 October

Recap

- Variables
  - Data types
    - Primitive types
      - Numbers
      - Strings
      - Boolean
      - `undefined` and `null`
    - Reference Types
      - Arrays
      - Objects
      - Functions
  - Type coercion
    - Variables can be evaluated to `true` or `false` based on their value

- Functions
- Conditionals
  - `if`
  - `if...else`
  - `if...else if...else`
- Loops
  - `for` loop
  - `while` loop
- Scoping
  - global scope
  - local scope
  - block scope

Checking variable types

The `typeof` operator returns the type of any value

```
console.log(typeof(100)) // number
console.log(typeof('london')) // string
console.log(typeof(false)) // boolean
...
```

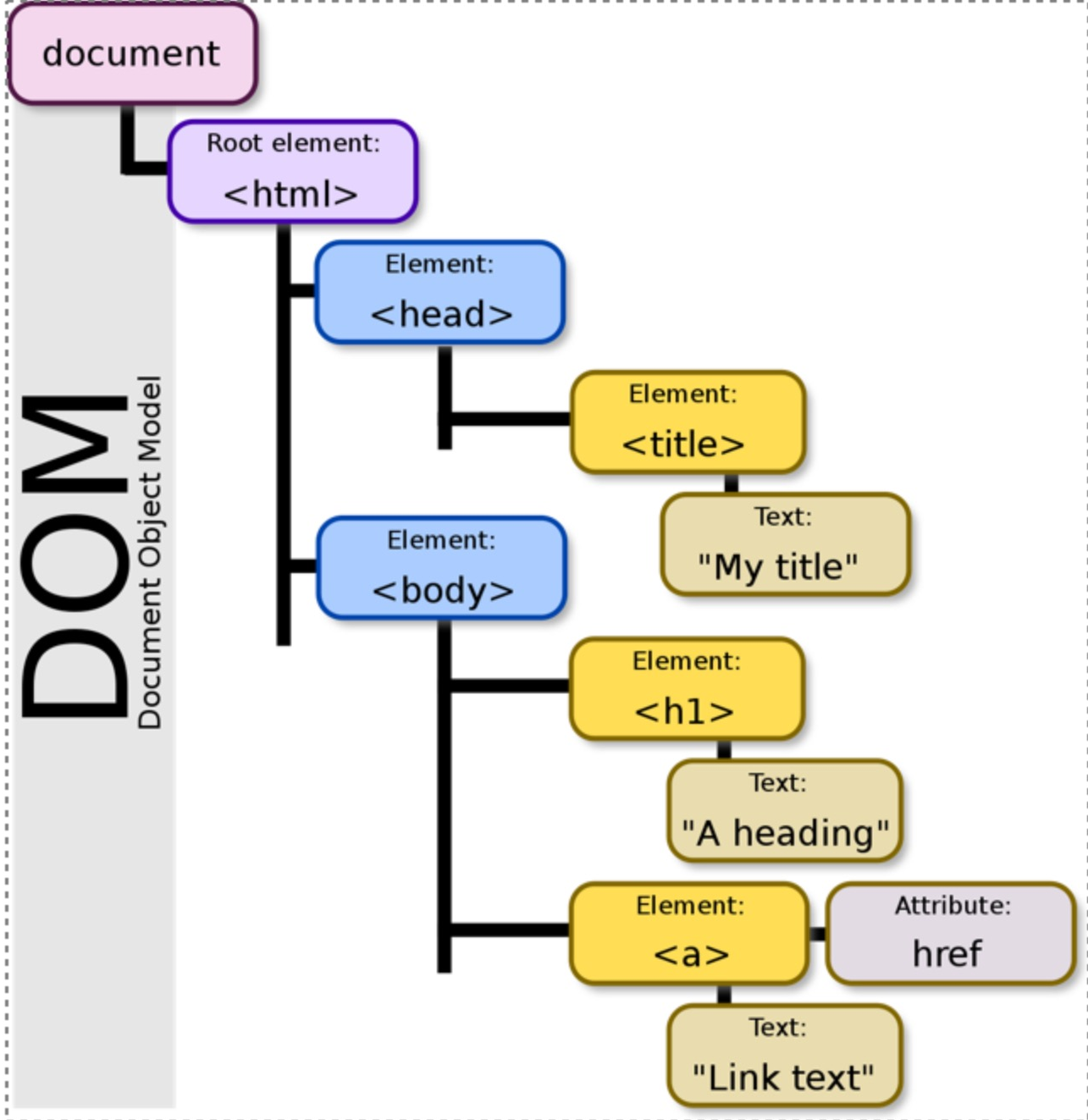
Session 2

Interactive JavaScript

HTML DOM

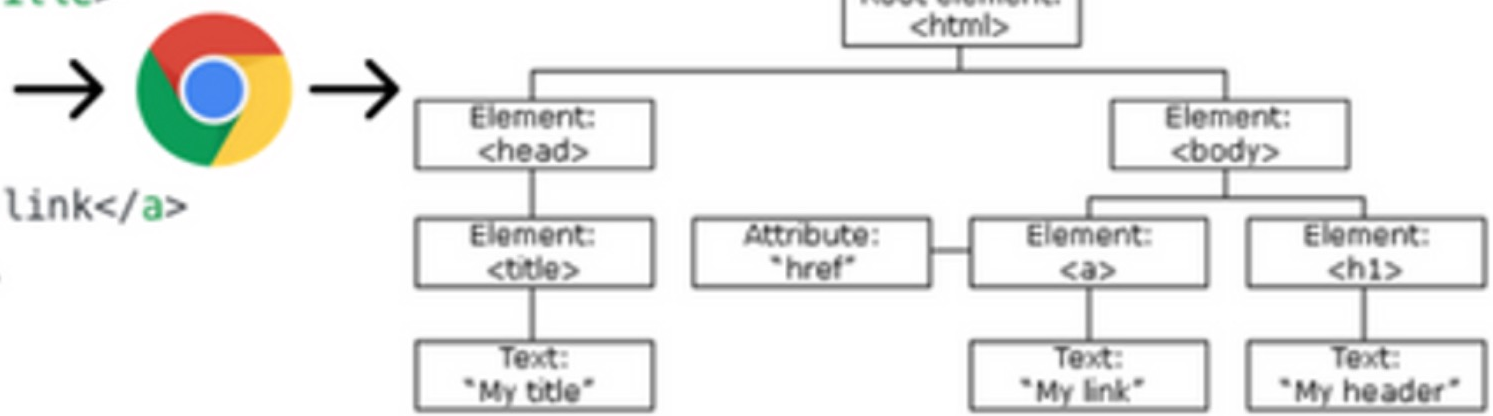
Document Object Model

- DOM is a visual representation of HTML elements on the web page
- DOM follows a tree-like shape
- Elements are nested



https://en.wikipedia.org/wiki/Document_Object_Model

```html
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="href">My link</a>
    <h1>My header</h1>
  </body>
</html>
```



JavaScript Nodes

- Nodes are array-like objects that contain information about the HTML element
- Nodes have attributes
- Nodes allow us dynamically manipulate the contents of the HTML element
- Nodes also allow us to listen and respond to user-events like mouse clicks, scrolling, keyboard inputs

The `document` in DOM

- `document` is the global entry point to DOM
- Provides methods for navigating and accessing the DOM elements

Searching for an HTML Element with JavaScript

1. `document.querySelector`
2. `document.getElementByID`
3. `document.getElementsByClassName`
4. `document.getElementByTagName`

`querySelector`

- Takes a CSS-like specifier as argument
- `".className"` or `"#ID"` or `"tagName"`

1. `querySelector` returns the first match
2. `querySelectorAll` returns an array of all matching elements

A Note about JavaScript Naming Convention

- Names are usually `camelCased` in JavaScript
- Meaning, every new word within the same name starts with a capital letter
- `myVariableName` for example

CSS Property

```
background-color
border-radius
font-family
z-index
...
```

JavaScript Equivalent

```
backgroundColor
borderRadius
fontFamily
zIndex
...
```

Using JS to modify styles

```
...
<button id="submit-btn">Submit</button>
...
```

```
const myButton = document.querySelector("#submit-btn");
```

```
myButton.style.backgroundColor = "yellow";
myButton.style.padding = "10px";
```

Working Directly with CSS Classes

`classList` property returns a list of all the CSS classes an element contains

Adding Classes

```
myButton.classList.add("new-class");
```

Removing Classes

```
myButton.classList.remove("new-class");
```

Toggling Classes

```
myButton.classList.toggle("new-class");
```

Modifying Text

`innerText` property allow us to control the contents of the HTML elements.

`innerHTML` lets us control the HTML content directly.

```
const container = document.querySelector("div");
```

```
container.innerText = "A sample text string";
```

Output:

A sample text string

```
container.innerHTML = "<h1>Adding Heading with JS</h1>";
```

Output:

Adding Heading with JS

Updating attributes

HTML attributes can be accessed with the `.` property

```
<img src="cat.jpg" id="my-image" />
```

The `src` attribute from the image can be accessed and updated as follows

```
const myImage = document.querySelector("#my-image");

myImage.src = "penguin.jpg"; // updates the src value of the image
```

Creating new DOM Elements

```
document.createElement("tag");
```

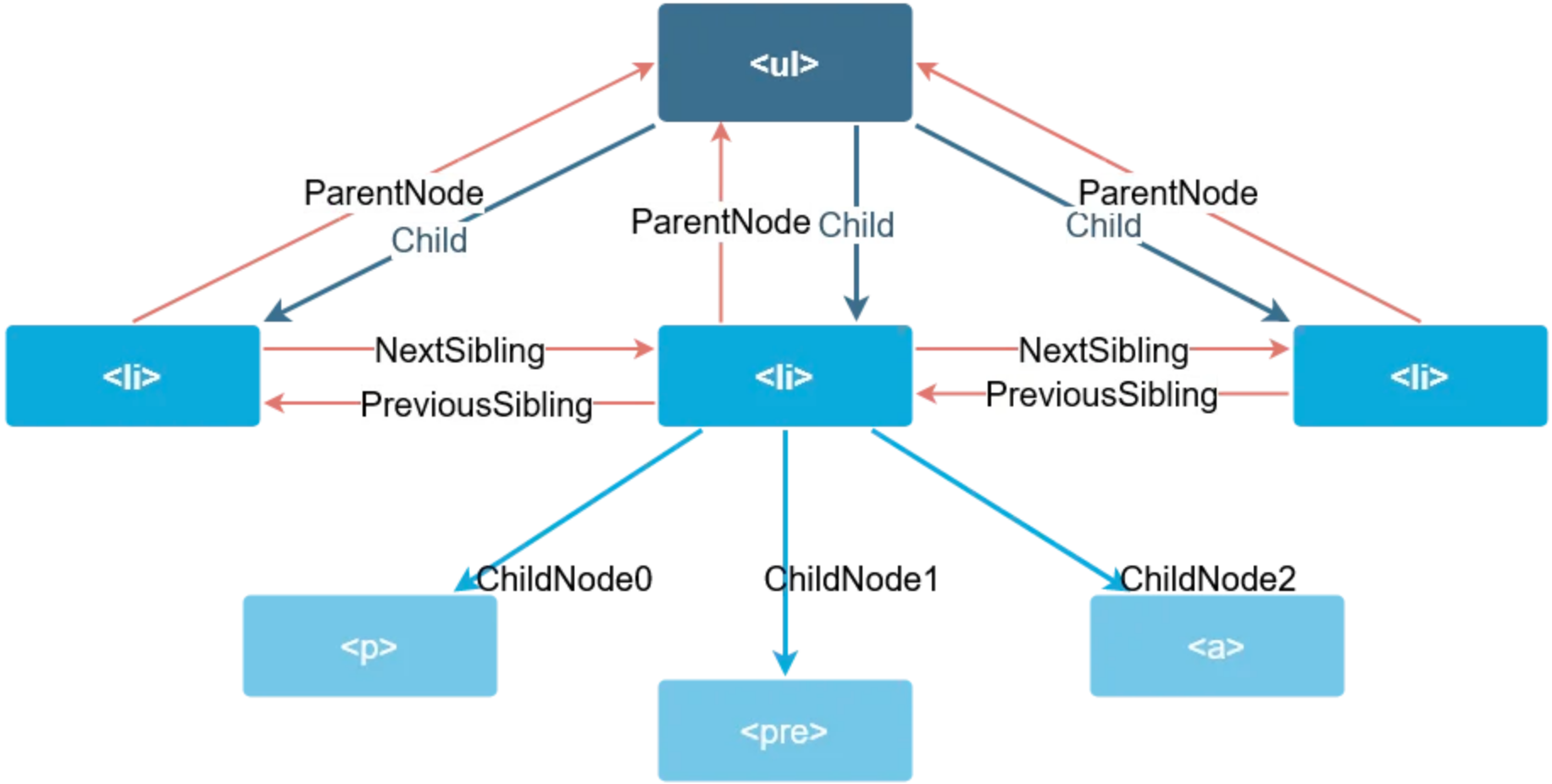
Appending Child Elements

With `.append("tag")` method, we can append a new element at the end of a selected node.

Example:

```
const gallery = document.querySelector("#gallery);
gallery.append("img");
```

Navigating DOM Nodes



The Next Big Step

JavaScript Events

JavaScript Events

- JavaScript events allow us make websites interactive
- We can `listen` to user events like mouse clicks, keyboard inputs, page scroll etc.
- And we can respond to those events- change styling; add a new element, remove an element- all on-the-fly

Callback Functions

The syntax

- A callback function is a function passed into another function as an argument
- They don"t look any different from standard JS functions
- Except, we automatically get access to the data through the input arguments
- In event callbacks, we get access to the `event` data

```
(event) => {
  // event data is passed as an argument
  console.log(event);
};
```

How do we do listen to events?

- We can add event listeners to any `node` we have access to

```
myButton.addEventListener("event-type", callback);
```

Example

```
const clickHandler = (event) => {
  alert("clicked");
};
```

```
myButton.addEventLisener("click", clickHandler);
```

Inline Functions

You might also come across codebases with callback functions written inline

```
myButton.addEventListener("click", (e) => alert("clicked"));
```

Shorthand syntax

- ES6 arrow function syntax!
- If you"re returning only one line from within the function, JS allows you to get rid of the curly braces

Standard Callback Function

```
(event) => {
  mouseX = event.clientX;
};
```

Shorthand Callback

```
(event) => (mouseX = event.clientX);
```

All the different events

- Click
- Double click
- Drag
- Drop
- Mouseover
- Mouseout
- Key press
- Mousewheel
- and soooo many more
  documentation here - https://developer.mozilla.org/en-US/docs/Web/Events

Recap: Setting up projects

- We need an `index.html` file to be able to use javascript in our browser
- Create a javascript file (you can name it anything! But `script.js` is what you will see most often)
- Once you have created a `*.js` file, within your `index.html`, add the following line to link your HTML to JS

```
<script src="script.js" />
```

> Note: It is important to add the above line only at the end of your HTML page. Just before the `</body>` closing tag.

> This ensures the site pages load all their content before the JavaScript is processed.

Lab Activity

Create a webpage with the following elements only using JavaScript

- A heading
- A button that creates an alert
- A paragraph
- An image

Research on how to add styles to an element using JavaScript, and implement it on your elements

- Add colours
- Add margins & paddings
- Change the font-family / font-size etc
- Use flex / grid systems