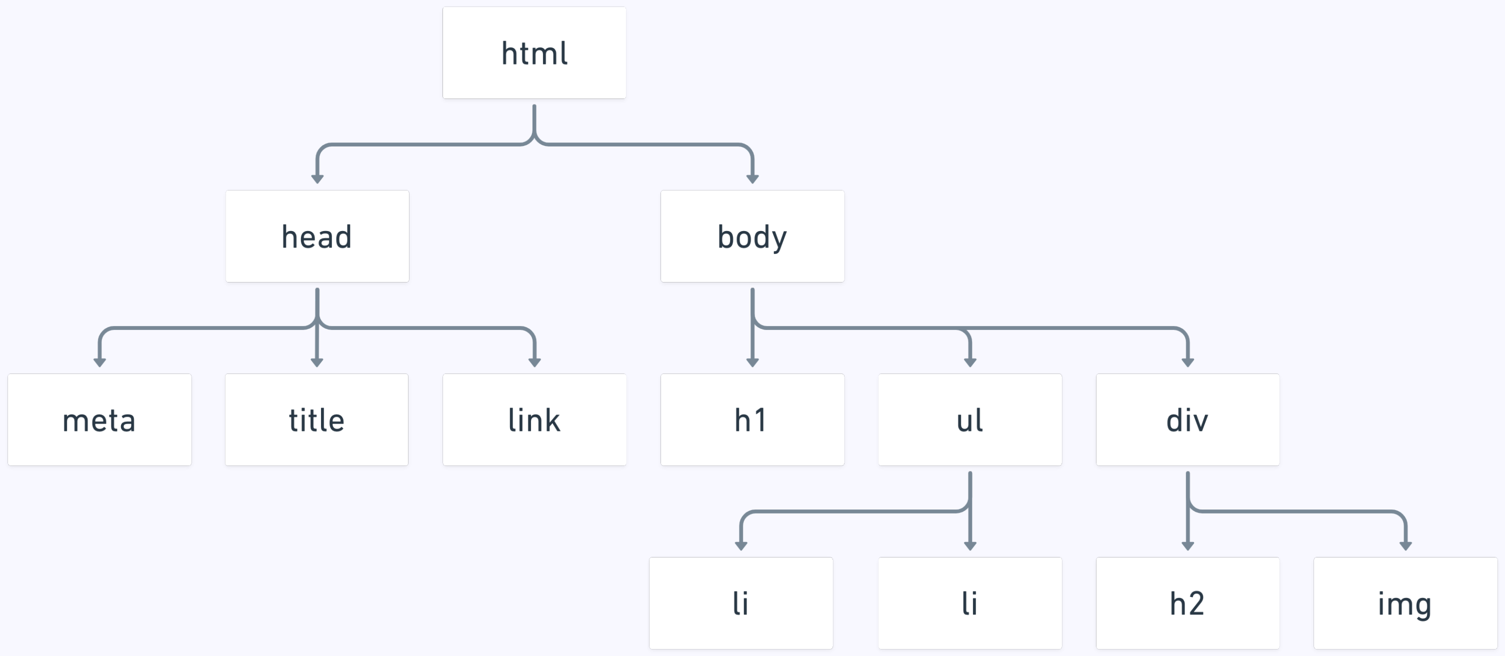
# The Document Object Model

## What is the DOM?

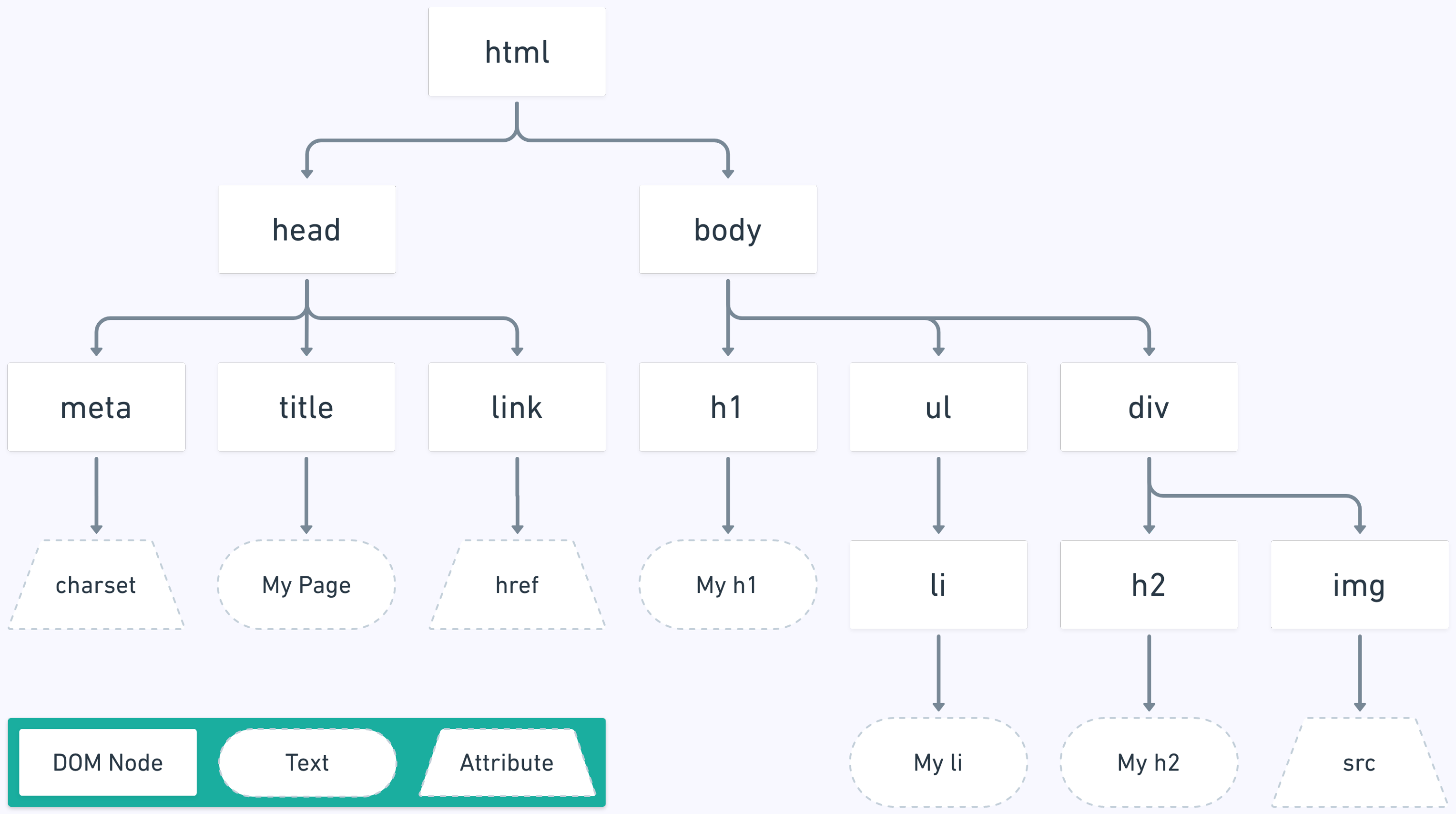
## What is the DOM?

- It stands for the **Document Object Model**
- It is way that JS interacts with HTML & CSS
- It's basically a JavaScript object
    - Though it is referred to as a **tree**
- The browser always has it
- It is your HTML when it is received and parsed by the browser

## What does the DOM look like?



## The DOM Tree



**The DOM Tree (with Attributes and Key)**

## Key Terminology

- Each point of data is calle a **node**
- Each **node** can have **parents**, **siblings** and **children**
- The **DOM** is accessed through a global variable called:
    - `document`
- We can call methods and access properties (just like an object)

## Identify Away!

## Identify Away!

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>The Document Object Model</title>
</head>
<body>

  <h1>Hello World</h1>

  <img src="https://fillmurray.com/400/400">

  <div>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
    <a href="https://ga.co">General Assembly</a>
  </div>

  <script src="main.js"></script>
</body>
</html>
```

## The `document`

The `document` object gives us a way of:

- Accessing the DOM
- Finding Elements
- Changing Styles
- Creating Elements
- etc.

## DOM Manipulation

The general strategy for DOM manipulation:

- Find the DOM node by using an access method
    - Store it in a variable
- Manipulate the DOM node
    - e.g. By changing its attributes, style, innerHTML

## Selecting Elements

## DOM Access Methods

- `document.getElementById("id");`
- `document.getElementsByTagName("tag");`
- `document.getElementsByClassName("class");`
- **`document.querySelector("CSS Selector");`**
- **`document.querySelectorAll("CSS Selector");`**

## document.querySelector("CSS Selector");

```html
<h1>Hello World</h1>
<img class="bill" src="https://fillmurray.com/400/400">
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
```

```javascript
// To select the H1
var heading = document.querySelector("h1");

// To select the img with class of "bill"
var bill = document.querySelector(".bill");

// To select the paragraph
var paragraph = document.querySelector("p");
```

All valid CSS selectors work!

## document.querySelectorAll("CSS Selector");

```html
<ul>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ul>

<img src="https://fillmurray.com/300/300">
<img src="https://placecage.com/300/300">
```

```js
var allListItems = document.querySelectorAll("li");
var allImages = document.querySelectorAll("img");
```

- All valid CSS selectors work!
- Returns a `NodeList`, which is very similar to an `Array`

## document.querySelectorAll("CSS Selector");

```
var allListItems = document.querySelectorAll("li");

for (var i = 0; i < allListItems.length; i += 1) {
  var currentListItem = allListItems[i];
  console.log(currentListItem);
}
```

## `querySelector` vs. `querySelectorAll`

- Both receive a valid CSS selector (as a string)
- `querySelector` will `return`:
    - The **first** `Element` (like an Object) it finds that matches
    - `null` if it doesn't find anything
- `querySelectorAll` will always `return`:
    - A `NodeList` (like an Array)

## In-class Lab / Exercise

The DOM Detective!

## Manipulating Elements

## Manipulating Elements

- We can:
    - Get and set attributes
    - Change the HTML within elements
    - Get and set values from inputs and textareas
    - Change styles
    - etc.

## el.getAttribute("attr");

```html
<img src="http://fillmurray.com/200/200" alt="Bill!">

<a href="https://ga.co" id="generalAssembly">
  A link to GA
</a>
```

```javascript
var image = document.querySelector("img");
var srcText = image.getAttribute("src");
var altText = image.getAttribute("alt");

var aTag = document.querySelector("a");
var href = aTag.getAttribute("href");
var id = aTag.getAttribute("id");
```

## el.setAttribute("attr name", "new attr value");

```html
<img src="http://fillmurray.com/200/200" alt="Bill!">

<a href="https://ga.co" id="generalAssembly">
  A link to GA
</a>
```

```javascript
var image = document.querySelector("img");
image.setAttribute("src", "http://placecage.com/200/200");
image.setAttribute("alt", "Another image");

var aTag = document.querySelector("a");
aTag.setAttribute("href", "/home");
aTag.setAttribute("id", "home-link");
```

## el.innerText

```html
<h1>Hello World</h1>
```

```javascript
var heading = document.querySelector("h1");

// Accesses the current text
var currentText = heading.innerText;

// Changes the current text
heading.innerText = "This is the text";

// Appends "!!!" to the end of heading
heading.innerText += "!!!"
```

## el.innerHTML

```html
<h1>Hello World</h1>
```

```javascript
var heading = document.querySelector("h1");

// Accesses the current HTML within heading
var currentHTML = heading.innerHTML;

// Sets the HTML to something else
heading.innerHTML = "<span>Hi there</span>";

// Appends "!!!" to the end of heading
heading.innerHTML += "!!!";
```

## el.value

```html
<input type="text" value="User types here">
```

```javascript
var input = document.querySelector("input");

// Get the current value
var currentValue = input.value;

// Change the value
input.value = "New Value";
```

# Getting CSS Styles

```html
<h1>Hello World</h1>
```

```javascript
var heading = document.querySelector("h1");

// Getting Styles
var currentStyles = getComputedStyle(heading);

// Get text color of heading
var color = currentStyles.color;

// Get the font size of heading (notice the camelCase!)
var fontSize = currentStyles.fontSize;
```

## Setting CSS Styles

```html
<h1>Hello World</h1>
```

```javascript
var heading = document.querySelector("h1");

// Change the width (you need the units!)
heading.style.width = "400px";

// Change the font-size (notice the camelCase!)
heading.style.fontSize = "24px";
```

## In-class Lab / Exercise

Change the page!

## Creating DOM Nodes

## Creating DOM Nodes

```javascript
// Create Element in Memory
var newPara = document.createElement( "p" );

// Set the text
newPara.innerText = "Created with JS";

// Set the styles
newPara.style.fontSize = "24px";
newPara.style.color = "hotpink";
```

## Putting Elements on the page

## el.appendChild( NODE );

```
var newPara = document.createElement( "p" );
newPara.innerText = "Created with JS";

// Put it at the end of the body
document.body.appendChild( newPara );
```

## el.insertBefore(NEW NODE, REFERENCE NODE)

```html
<div>
  <h1>Add the paragraph before here!</h1>
</div>
```

```javascript
var newPara = document.createElement( "p" );
newPara.innerText = "Created with JS";

var div = document.querySelector("div");
var h1 = document.querySelector("h1");

// Put the newPara right before the h1 that is in the div
div.insertBefore(newPara, h1);
```

## In-class Lab / Exercise

More DOM Manipulation!

## Events

## Some Terminology

- **Event:**
  - Something that happens
- **Callback:**
  - A function that gets called as a response
- **Node || Target:**
  - The Element that will be interacted with
- **Event listener:**
  - Event + Callback + Target

## Events with JavaScript

Three important things:

- The element that is going to be interacted with (body, h1, p etc.)
- The event type ("click", "hover", "scroll" etc.)
- The response (often called the callback - a function!)

## Events Pseudocode

```
WHEN the element with ID of "toggle" is CLICKED

    SELECT the body tag and save as body
    STORE the currentBackground of body

    IF currentBackground === "hotpink"
        CHANGE the body CSS to have a ghostwhite background

    ELSE
        CHANGE the body CSS to have a hotpink background
```

## Events Pseudocode

```
WHEN the button with class of "login" is CLICKED

    STORE the email that was typed in as userEmail
    STORE the password that was typed in as userPassword

    IF it is the right combination of email and password

      Tell the user that they are logged in

    ELSE

      Tell the user that something went wrong
```

## `el.addEventListener(TYPE, CALLBACK);`

```
var myButton = document.querySelector("button");

function myCallback() {
  console.log("The button was clicked");
}

myButton.addEventListener("click", myCallback);
```

- Find the element
- Add the event listener, and pass in:
    - An event type
    - A callback function

## Anonymous Functions

```javascript
var myButton = document.querySelector("button");

myButton.addEventListener("click", function() {
  console.log("button clicked!");
});
```

## Referenced Events

```
var myButton = document.querySelector("button");

function myCallback() {
  console.log("The button was clicked!");
}

myButton.addEventListener("click", myCallback);

// Some time later...
myButton.removeEventListener("click", myCallback);
```

## What events are there?

Here are some of the available event types:

- Mouse Events
- Keyboard Events
- Browser Events
- Form Events

## Mouse Events

- click
- dblclick
- mousemove
- mousedown
- mouseup
- contextmenu
- ...

## Key Events

- keydown
- keyup
- keypress
- ...

## Window/Browser Events

- resize
- scroll
- ...

## Form Events

- submit
- ...

## They always look the same!

```
TARGET.addEventListener(
  EVENT_TYPE,
  CALLBACK_FUNCTION
);
```

## In-class Lab / Exercise

Picture Generator

## The event parameter

## The `event` parameter

When JS runs an event handler, it gives us a JS object as a parameter

That `event` object has lots of information about things like:

- How long we have been on the page
- Where the mouse was
- What key was pressed
- The target of the event

We can call it whatever we like (though `e` and `event` are very common)

## The event parameter

```
var button = document.querySelector("button");

function onButtonClick(event) {
  console.log(event);
}

button.addEventListener("click", onButtonClick);
```

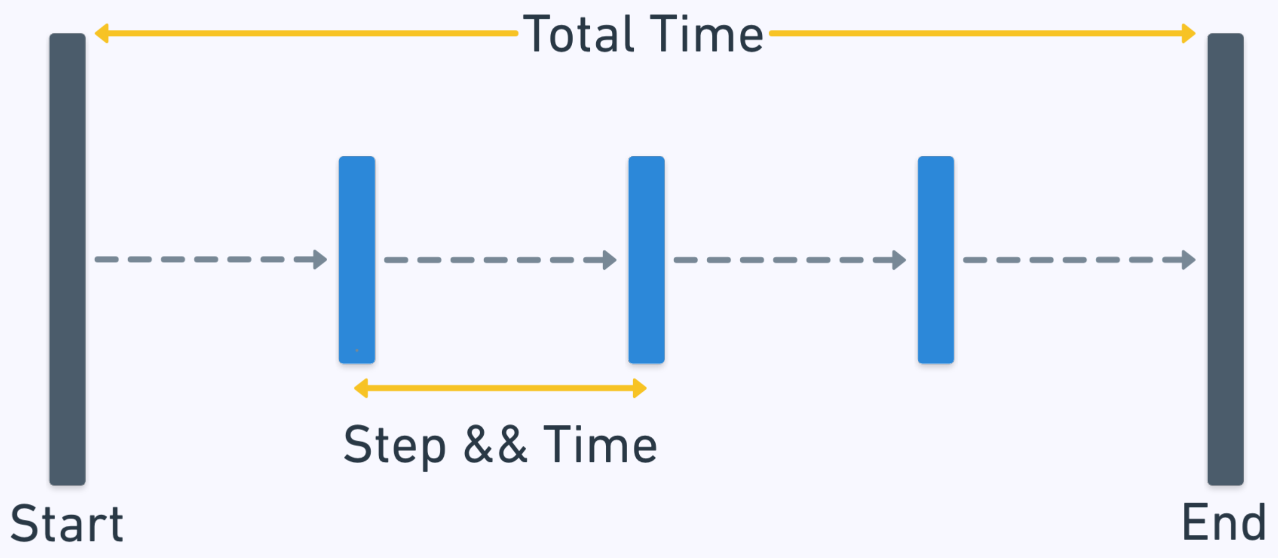
## The event parameter

```
window.addEventListener("mousemove", function (event) {
  console.log(event);
});
```

## In-class Lab / Exercise

MadLibs and Analytics!

## Animations


Total Time
Step && Time
Start
End

## Animations

Things you need to define:

- **Starting Point**
- **Step**
- **Time between steps**
- **Total time**
- **Ending Point**

## Working with time

There are two main ways to work with time in JavaScript

- You can set a **delay** with `setTimeout`
- You can set an **interval** with `setInterval`

## Timers in JavaScript

```
function delayedFunction() {
  console.log("Called once");
}
setTimeout( delayedFunction, 1000 );

function regularlyScheduledProgram() {
  console.log("Called regularly");
}
setInterval(regularlyScheduledProgram, 1000);
```

## Fade Away: Pseudocode

```
SELECT and STORE the image as bill

CREATE a function called fadeBillAway
  GET the current opacity and store as currentOpacityAsString
  GET the current opacity as a number and store as currentOpacity
  CREATE newOpacity by subtracting 0.01 from currentOpacity
  UPDATE bill opacity to be newOpacity

  IF the currentOpacity is >= 0
    CALL fadeBillAway in 10ms

CALL fadeBillAway to start the animation
```

## Fade Away

```
var bill = document.querySelector("img");

function fadeBillAway() {
  var currentOpacityAsString = getComputedStyle(bill).opacity;
  var currentOpacity = parseFloat(currentOpacityAsString, 10);
  var newOpacity = currentOpacity - 0.01;
  bill.style.opacity = newOpacity;
  if (currentOpacity >= 0) {
    window.setTimeout(fadeBillAway, 10);
  }
}

fadeBillAway();
```

## In-class Lab / Exercise

- Finish off the in-class exercises
- Do the calculator exercise
- Dancing Cats!