DOM Manipulation

Why Should You Care?

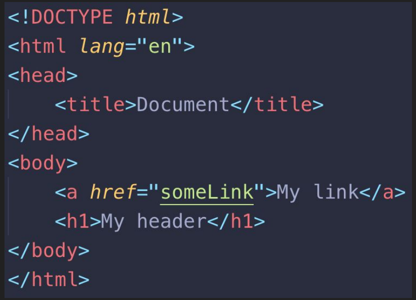
- A few examples:
  - Games
  - Scrolling Effects
  - Dropdown Menus
  - Form Validations
  - Interactivity
  - Animations
  - Every awesome site ever

Document Object Model a.k.a. The DOM

- The Document Object Model is the interface between your javascript and HTML + CSS

DOM cont.

- The browser turns every HTML tag into a JavaScript object that we can manipulate.

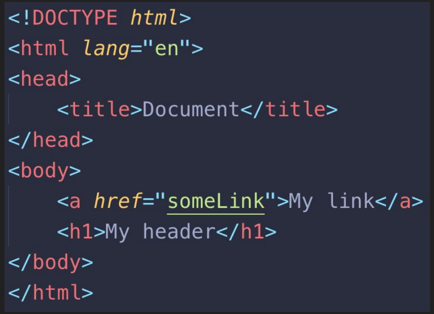


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <a href="someLink">My link</a>
    <h1>My header</h1>
</body>
</html>
```

- Everything is stored inside of the document object.

The Process

- SELECT an element and then MANIPULATE



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <a href="someLink">My link</a>
    <h1>My header</h1>
</body>
</html>
```

- For our example, we’ll change the <h1> color using JS

The Process cont.

- SELECT the <h1> and save to a variable.

```
var h1 = document.querySelector('h1')
```

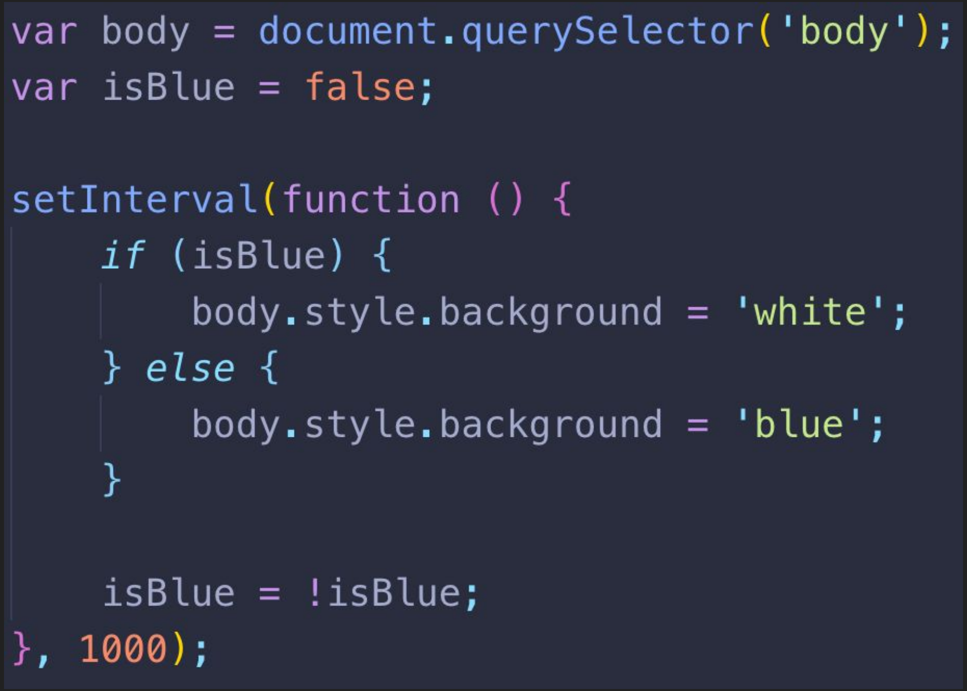
- MANIPULATE using the <h1> we selected.

```
var h1 = document.querySelector('h1');

h1.style.color = 'pink';
```

One more example

- SELECT the <body> and change its color every second



```
var body = document.querySelector('body');
var isBlue = false;

setInterval(function () {
    if (isBlue) {
        body.style.background = 'white';
    } else {
        body.style.background = 'blue';
    }

    isBlue = !isBlue;
}, 1000);
```

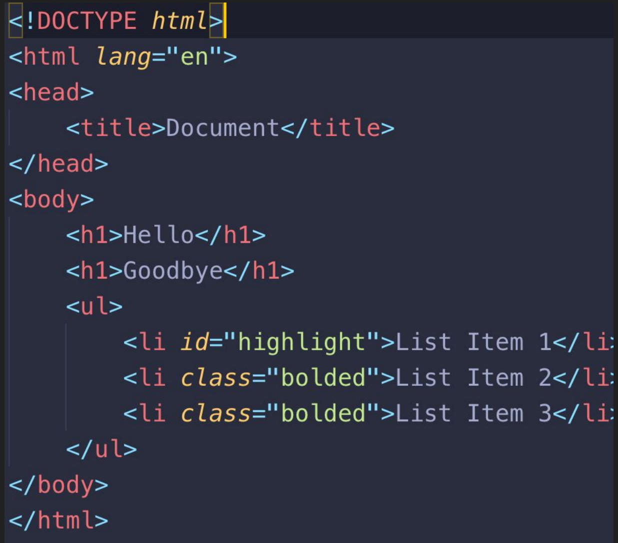
Important Selectors

- The document comes with a bunch of methods for selecting elements. We're going to learn about the following 5:
  - document.getElementById()
  - document.getElementsByClassName()
  - document.getElementsByTagName()
  - document.querySelector()
  - document.querySelectorAll()

getElementById

- Takes a string argument and returns the one element with a matching ID.

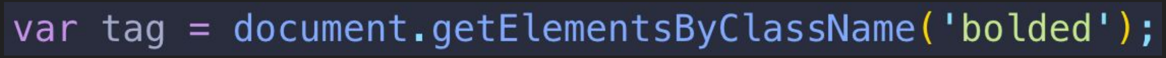
```
var tag = document.getElementById("highlight");
```



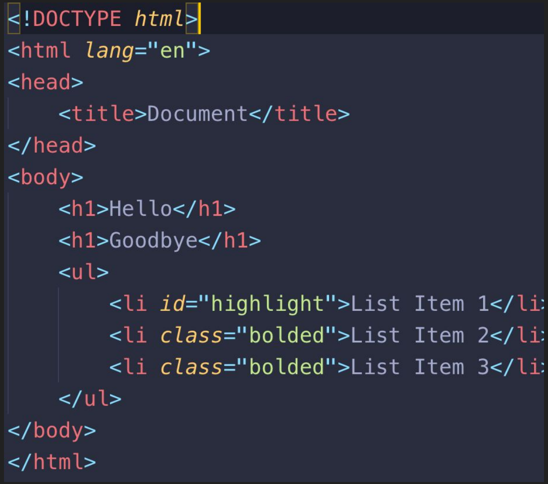
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1>Hello</h1>
    <h1>Goodbye</h1>
    <ul>
        <li id="highlight">List Item 1</li
        <li class="bolded">List Item 2</li
        <li class="bolded">List Item 3</li
    </ul>
</body>
</html>
```

getElementsByClassName

- Takes a string argument and returns a list of elements that have a matching class.



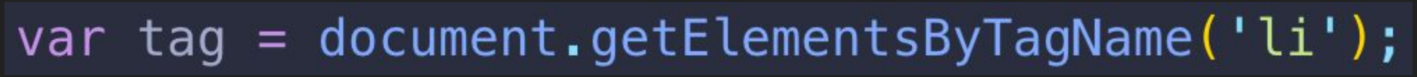
```
var tag = document.getElementsByClassName('bolded');
```



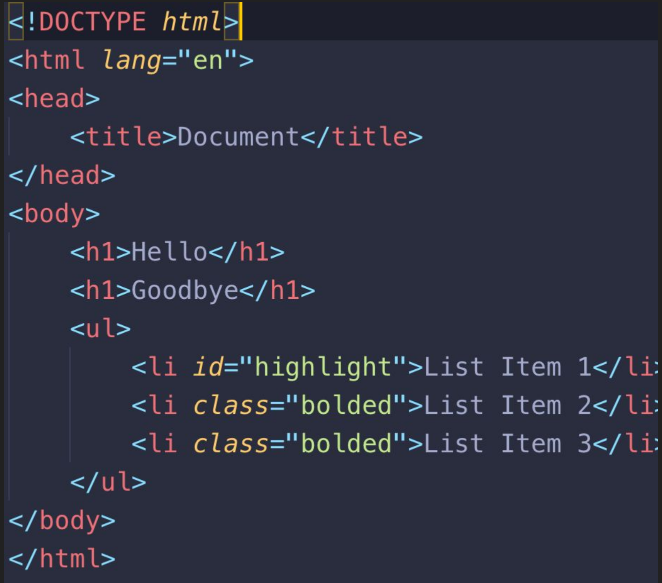
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1>Hello</h1>
    <h1>Goodbye</h1>
    <ul>
        <li id="highlight">List Item 1</li
        <li class="bolded">List Item 2</li
        <li class="bolded">List Item 3</li
    </ul>
</body>
</html>
```

getElementsByTagName

- Returns a list of all elements of a given tag name, like <li> or <h1>



```
var tag = document.getElementsByTagName('li');
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>
    <h1>Hello</h1>
    <h1>Goodbye</h1>
    <ul>
        <li id="highlight">List Item 1</li
        <li class="bolded">List Item 2</li
        <li class="bolded">List Item 3</li
    </ul>
</body>
</html>
```

querySelector

- Returns the first element that matches a given CSS-style selector.

```
var tag = document.querySelector("#highlight");
```

```
var tag = document.querySelector(".bolded");
```

```
var tag = document.querySelector("h1");
```

querySelectorAll

- Returns **a list of elements** that matches a given CSS-style selector.

```
var tag = document.querySelectorAll("#highlight");
```

```
var tag = document.querySelectorAll(".bolded");
```

```
var tag = document.querySelectorAll("h1");
```

Important Note!

- If you select multiple elements with one selector, those elements will be put into an array.
- For example,

```html
<ul>
    <li class="bolded">First Item</li>
    <li>Second Item</li>
    <li class="bolded">Third Item</li>
</ul>
```

```javascript
var tags = document.querySelectorAll('.bolded');
```

- Our variable “tags” is now an array and looks like this.

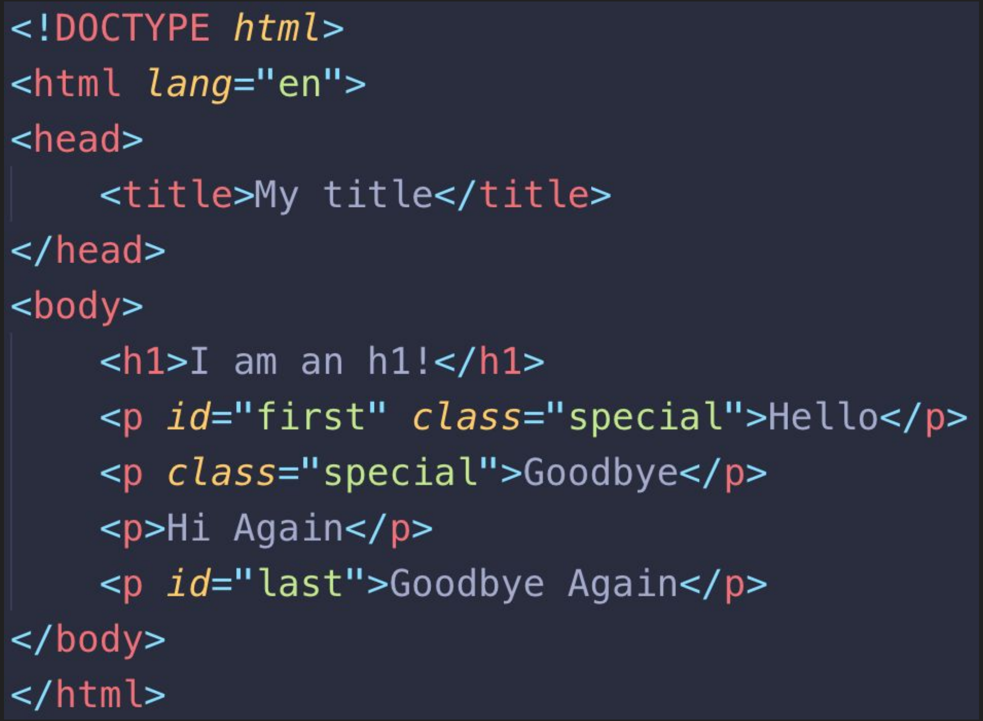
```javascript
var tags = ['<li>First Item</li>', '<li>Third Item</li>'];
```

Important Note cont.

- Now you can use array methods and iterations such as *.length, slice(), push(), forEach(), etc.*
- If you know that your selector is going to return multiple elements, make sure your variable is plural.

Assignment 11.1: Selector Exercise

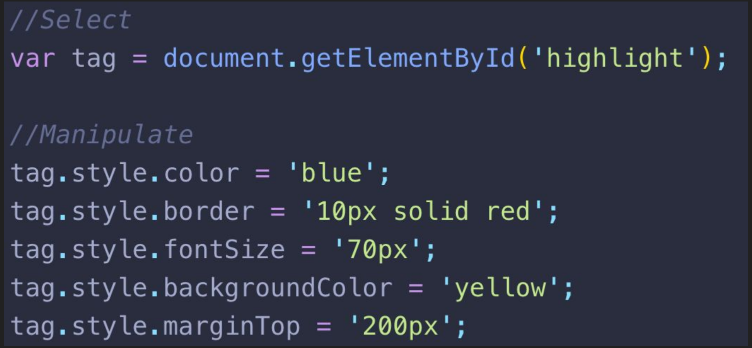
- Come up with 4 different ways to select the first <p> tag.



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My title</title>
</head>
<body>
    <h1>I am an h1!</h1>
    <p id="first" class="special">Hello</p>
    <p class="special">Goodbye</p>
    <p>Hi Again</p>
    <p id="last">Goodbye Again</p>
</body>
</html>
```

Manipulating: Style

- The style property is one way to manipulate an element’s style.



```
//Select
var tag = document.getElementById('highlight');

//Manipulate
tag.style.color = 'blue';
tag.style.border = '10px solid red';
tag.style.fontSize = '70px';
tag.style.backgroundColor = 'yellow';
tag.style.marginTop = '200px';
```

- It is recommended for style to be defined in a separate file or files. The style property allows for quick styling i.e. testing purposes.

An Alternative to Style

- Rather than directly manipulating style with JS, we can define a CSS class and then toggle it on or off with JS.

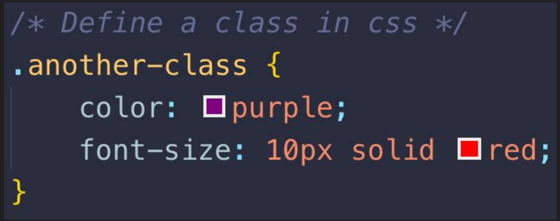
```
//Instead of this:
var tag = document.getElementById('highlight');
tag.style.color = 'blue';
tag.style.border = '10px solid red';
```

```
/* Define a class in css */
.some-class {
    color: blue;
    border: 10px solid red;
}
```

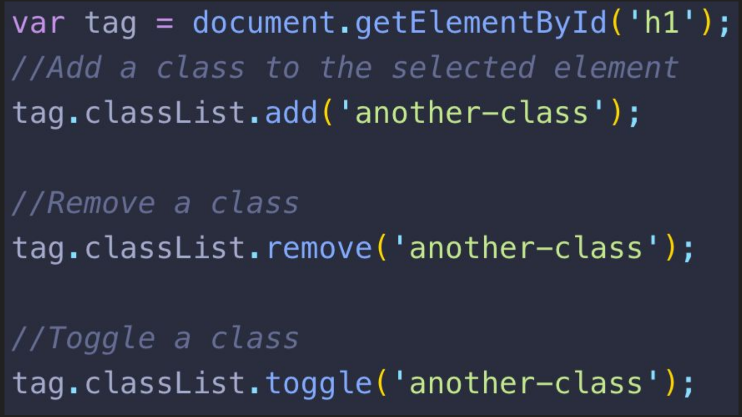
```
var tag = document.getElementById('highlight');
//Add the new class to the selected element
tag.classList.add('some-class');
```

classList

- A read-only list that contains the classes for a given element. It is **not an array**.
- Can use the *add(), remove(), or toggle()* methods along with classList.



```css
/* Define a class in css */
.another-class {
    color: purple;
    font-size: 10px solid red;
}
```



```javascript
var tag = document.getElementById('h1');
//Add a class to the selected element
tag.classList.add('another-class');

//Remove a class
tag.classList.remove('another-class');

//Toggle a class
tag.classList.toggle('another-class');
```

Manipulating: textContent

- Returns a string of all the text contained in a given element.

```html
<p>This is an <strong>awesome</strong> paragraph</p>
```

```javascript
//Select the <p> tag:
var tag = document.querySelector('p');

//Retrieve the textContent:
tag.textContent; //"This is an awesome paragraph"

//Alter the textContent:
tag.textContent = 'blah blah blah';
```

Manipulating: innerHTML

- Similar to textContent, except it returns a string of all the HTML contained in a given element.

```
<p>This is an <strong>awesome</strong> paragraph</p>
```

```
//Retrieve the textContent:
tag.textContent; //"This is an awesome paragraph"

tag.innerHTML;
//"This is an <strong>awesome</strong> paragraph"
```

```
//Alter the html content
tag.innerHTML = '<h1>Hello World</h1>';
```

Manipulating: Attributes

- Use getAttribute() and setAttribute() to read and write attributes like src or href.

```html
<a href="www.google.com">I am a link</a>
<img src="logo.png">
```

```javascript
var link = document.querySelector('a');
link.getAttribute('href'); //"www.google.com"
//Change href attribute
link.setAttribute('href', 'www.dogs.com');
//<a href="www.dogs.com">I am a link</a>

//To change the image src
var img = document.querySelector('img');
img.setAttribute('src', 'corgi.png');
//<img src="corgi.png">
```