

Module 3-7

DOM

Objectives

- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure
- innerText on HTML elements
- Create new DOM elements
- Traverse the DOM
- Investigate the living DOM in the browser

Document Object Model

- The Document Object Model (DOM for short) is a tree representation of all the HTML elements on a given web page.
- Most browsers have a “Developer Tools” interface that allows for quick inspection of a DOM element and how it relates to other elements on the page.
- The focus of today’s lecture is how to use JavaScript to interact with the DOM.

DOM Manipulation

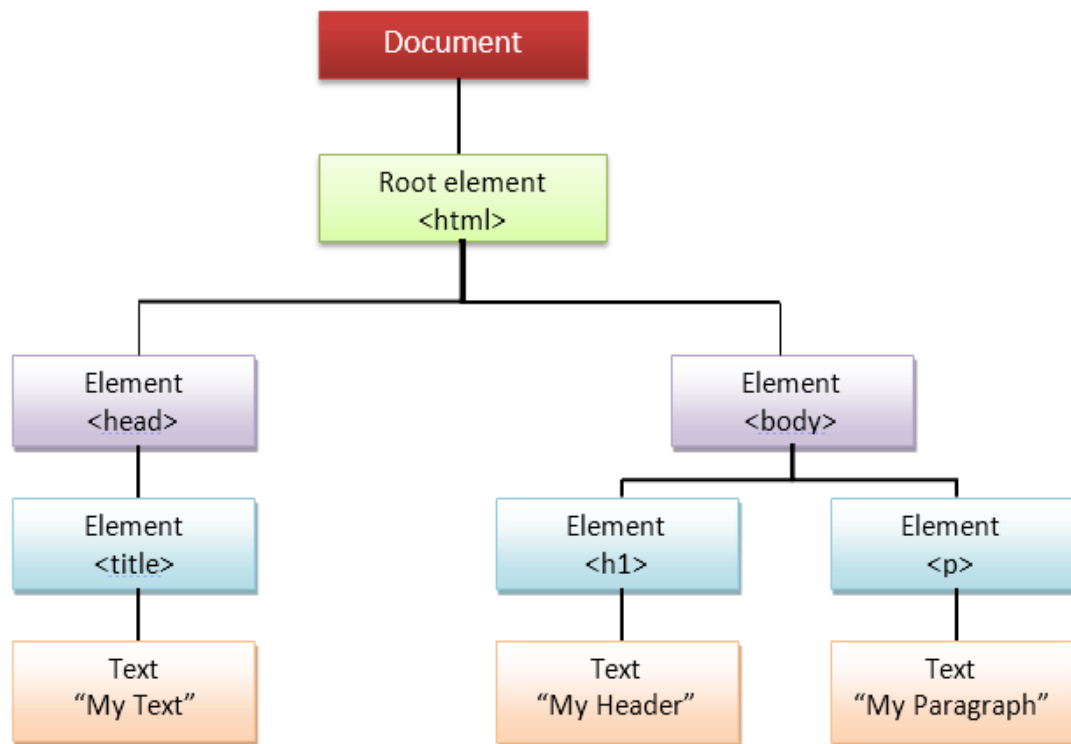
- Writing code to change and select information on the DOM using JavaScript while the page is loaded in the browser. We will be using Vanilla JavaScript for DOM manipulation
- Vanilla JavaScript
 - JavaScript that does not rely on any outside utility libraries to do things that can be done with functions and objects defined in the ECMAScript specification.



DOM vs. HTML

- The DOM is a model of a document with an associated API for manipulating it.
- HTML is markup language that lets you represent a certain kind of DOM in text.
- DOM is tree model to represent HTML.
- DOM doesn't always match the HTML source code

DOM



Chrome Developer Tools Demo



DOM Elements: ID's and Classes

Let's review id and classes for HTML elements. Consider the following HTML code:

```
<p id='intro'>I dedicate this page to my dog Horace</p>

<p class = 'content'>Some Widgets are Doodads</p>
<p class = 'content'>Some Doodads are Thingamagjigs</p>
<p class = 'content'>All Thingamajigs are Whatchamacallits</p>
```

- The first paragraph is marked with an id - ideally we use an id to uniquely identify one element.
- All other paragraphs are marked with a class - ideally we can apply a class to several elements that we feel share some commonality.

DOM Elements: Properties


The id and class names are properties of a DOM Object. We have already dealt with a lot of these properties while learning CSS: height, width, color, etc.



getElementById

We can use `getElementById` to identify and assign a DOM element to a JavaScript variable. We can then interrogate or change its properties. Consider this example:

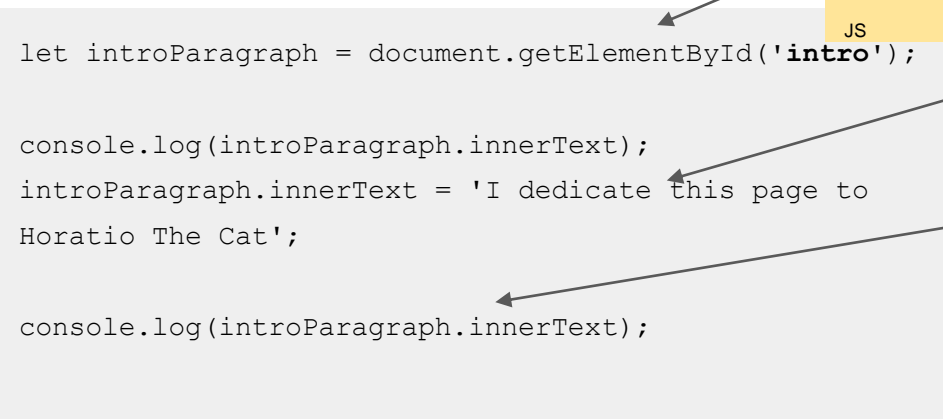
```
<body>
<p id='intro'>I dedicate this page to my dog.</p>
<script src="thisScript.js"></script>
</body>
```



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

console.log(introParagraph.innerText);
introParagraph.innerText = 'I dedicate this page to
Horatio The Cat';

console.log(introParagraph.innerText);
```



- Note that we start off by targeting the intro paragraph, since we know it has an id of intro we can use the `getElementById` method.

- We assigned this DOM object to a variable called `introParagraph`.

- We changed the `innerText` property to contain a different sentence.

getElementById

- The end result of this example is that the HTML page will have “I dedicate this page to Horatio The Cat”, thus changing the original text.
- There is a similar property called innerHTML, that should be avoided as it allows for injection of unwanted JavaScript content beyond the text.
 - innerHTML that takes input from a user sets your page up for XSS
 - Rule of thumb - if you want to change text, use innerText like we have done here.

Cross-Site Scripting attack (XSS)

- Hackers execute malicious JavaScript within a victim's browser
 - Code is run within user's browser
 - Code sits on top of legitimate website, tricking browsers into executing malware
- Persistent XSS
- Reflected XSS
- Self XSS
- Blind XSS
- DOM-based XSS

<https://sucuri.net/guides/what-is-cross-site-scripting/>

querySelectorAll

- getElementById is useful for identifying one DOM element but sometimes we need to identify several elements in one blow.
- In order to do this, we can leverage querySelectorAll which will return all matching elements and place them in an array.

querySelectorAll

Let's look at this example again:

```
<p id='intro'>I dedicate this page to my dog.</p>  
<p class = 'content'>Some Widgets are Doodads</p>  
<p class = 'content'>Some Doodads are Thingamagjigs</p>  
<p class = 'content'>All Thingamajigs are Whatchamacallits</p>
```

HTML

```
let paragraphs = document.querySelectorAll('.content');  
console.log(paragraphs.length);  
  
for (i = 0; i < paragraphs.length; i++) {  
  let paragraph = paragraphs[i];  
  paragraph.style.color = 'blue';  
}
```

JS

browser:

I dedicate this page to my dog.

Some Widgets are Doodads

Some Doodads are Thingamagjigs

All Thingamajigs are Whatchamacallits

querySelectorAll

Here's another example note what we've passed to the querySelectorAll method:

```
<p id='intro'>I dedicate this page to my dog.</p>  
<p class = 'content'>Some Widgets are Doodads</p>  
<p class = 'content'>Some Doodads are Thingamagjigs</p>  
<p class = 'content'>All Thingamajigs are Whatchamacallits</p>
```

HTML

```
let paragraphs = document.querySelectorAll('p');  
console.log(paragraphs.length);  
  
for (i = 0; i < paragraphs.length; i++) {  
  let paragraph = paragraphs[i];  
  paragraph.style.color = 'blue';  
}
```

JS

browser:

I dedicate this page to my dog.
Some Widgets are Doodads
Some Doodads are Thingamagjigs
All Thingamajigs are Whatchamacallits

querySelector

Finally, we have `querySelector()` which returns the first element found that matches a given criteria.

```
<p id='intro'>I dedicate this page to my dog Horace</p>
<p class = 'content'>Some Widgets are Doodads</p>
<p class = 'content'>Some Doodads are Thingamagjigs</p>
<p class = 'content'>All Thingamajigs are Whatchamacallits</p>
```

```
let paragraph = document.querySelector('p');
console.log(paragraphs.innerText);
```

“I dedicate this page to my
dog Horace”

value and checked properties

value gets the value from a text field. checked returns status of radio or checkbox elements:

```
Name: <input type="text" id="myText" value="Mickey"><br><br>
<form>
  What color do you prefer?<br>
  <input type="radio" name="colors" id="red">
  <label for="red">Red</label><br>
  <input type="radio" name="colors" id="blue">
  <label for="blue">Blue</label>
</form>
```

HTML

Using value, set the text field to "Johnny Bravo"

```
document.getElementById("myText").value = "Johnny Bravo";
document.getElementById("red").checked = true;
```

JS

Using checked, set the red box to true (or checked).

Creating DOM Elements

We can create brand new DOM elements from scratch. Consider the following code:

```
<ul id='theList'>  
  <li>Some Widgets are Doodads</li>  
  <li>Some Doodads are Thingamajigs</li>  
  <li>All Thingamajigs are Whatchamacallits</li>  
</ul>  
<script src="thisScript.js"></script>
```

HTML

A brand new element (a list item) is being created.

```
let extraListItem = document.createElement('li');  
extraListItem.innerText = 'All Foos are Bars';
```

JS

We identify the parent.

```
let parentList = document.getElementById('theList');  
parentList.appendChild(extraListItem);
```

Append the brand new element to the parent.

Assigning a class to an element

We can create brand new DOM elements from scratch. Consider the following code:

```
<ul id='theList'>
  <li>Some Widgets are Doodads</li>
  <li>Some Doodads are Thingamajigs</li>
  <li>All Thingamajigs are Whatchamacallits</li>
</ul>
<script src="thisScript.js"></script>
```

HTML

```
let extraListItem = document.createElement('li');
extraListItem.innerText = 'All Foos are Bars';
extraListItem.setAttribute('class', 'importantStuff');

let parentList = document.getElementById('theList');
parentList.appendChild(extraListItem);
```

JS

```
.importantStuff {
  color:red;
}
```

CSS

browser:

- Some Widgets are Doodads
- Some Doodads are Thingamajigs
- All Thingamajigs are Whatchamacallits
- All Foos are Bars

Inserting elements into the DOM

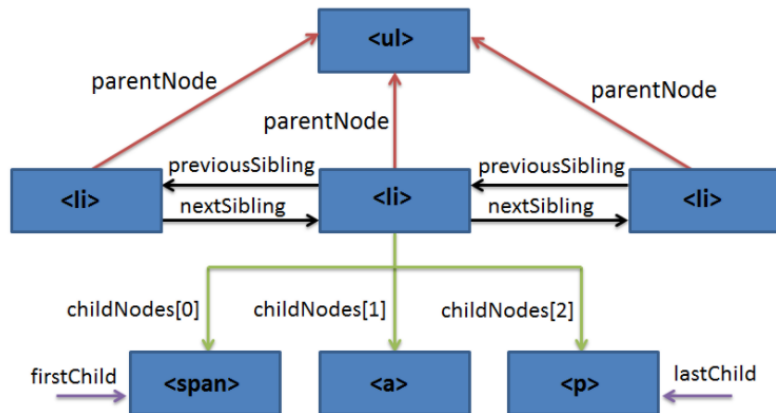
- insertAdjacentElement
 - beforeBegin
 - afterBegin
 - beforeEnd
 - afterEnd

```
<!-- beforebegin -->
<p>
  <!-- afterbegin -->
  foo
  <!-- beforeend -->
</p>
<!-- afterend -->
```

Traversing the DOM

```
<ul>  
<li>node</li>  
<li><span>node</span><a href="#">node</a><p>node</p></li>  
<li>node</li>  
</ul>
```

- previousSibling
- nextSibling
- childNodes
- firstChild
- lastChild
- parentNode



Selecting children with children and childNodes

- children
 - Returns an HTML collection, which you can turn into array
 - Returns elements that are children
 - Only contains HTML elements
 - Not text that might be in element
- childNodes
 - Returns a NodeList object that contains all nodes inside element (can also turn into array)
 - Returns nodes that are children of element
 - Includes text and comments that are in DOM

parentNode and adjacent elements

- parentNode
 - Returns parent of element
- Adjacent elements
 - nextElementSibling
 - previousElementSibling

Exercise time...

(but first)...

Product Reviews for Cigar Parties for Dummies

Host and plan the perfect cigar party for all of your squirrely friends.

Malcolm Gladwell

★★★★★ **What a book!**

It certainly is a book. I mean, I can see that. Pages kept together with glue (I hope that's glue) and there's writing on it, in some language.

Tim Ferriss

★★★★★ **Had a cigar party started in less than 4 hours.**

It should have been called the four hour cigar party. That's amazing. I have a new idea for muse because of this.

Ramit Sethi

★ **What every new entrepreneurs needs. A door stop.**

When I sell my courses, I'm always telling people that if a book costs less than \$20, they should just buy it. If they only learn one thing from it, it was worth it. Wish I learned something from this book.

Gary Vaynerchuk

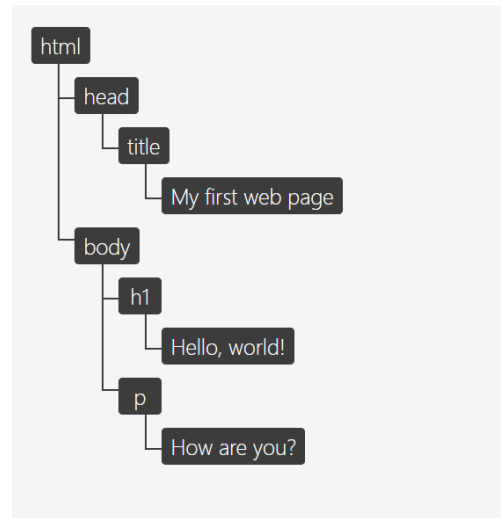
★★★★★ **And I thought I could write**

There are a lot of good, solid tips in this book. I don't want to ruin it, but prelighting all the cigars is worth the price of admission alone.

Objectives

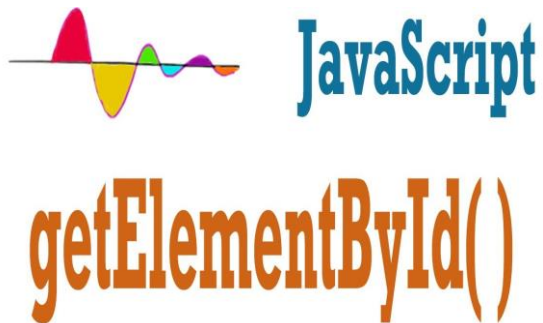
- Difference between the DOM and HTML

```
<!doctype html>
<html lang="en">
  <head>
    <title>My first web page</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <p>How are you?</p>
  </body>
</html>
```



Objectives

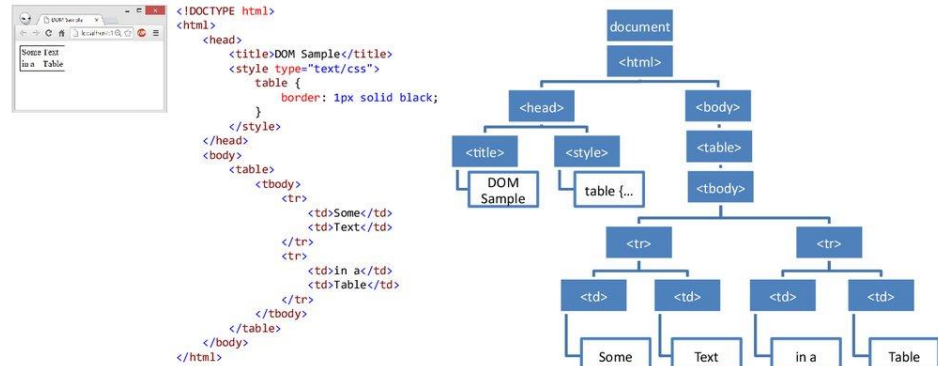
- Difference between the DOM and HTML
- Select elements from the DOM



Objectives

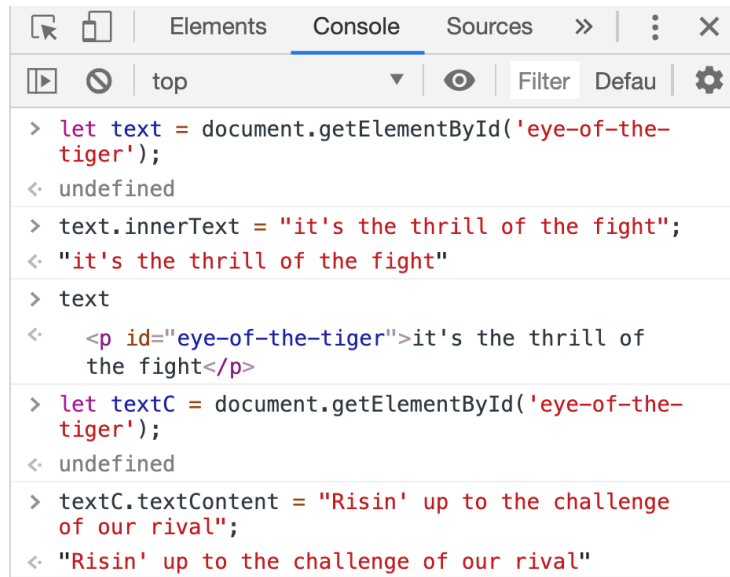
- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure

DOM Tree



Objectives

- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure
- innerText on HTML elements



```
> let text = document.getElementById('eye-of-the-tiger');
< undefined

> text.innerText = "it's the thrill of the fight";
< "it's the thrill of the fight"

> text
< <p id="eye-of-the-tiger">it's the thrill of the fight</p>

> let textC = document.getElementById('eye-of-the-tiger');
< undefined

> textC.textContent = "Risin' up to the challenge of our rival";
< "Risin' up to the challenge of our rival"
```

Objectives

- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure
- innerText on HTML elements
- Create new DOM elements

```
1
2
3  const mainDiv = document.querySelector(".main-div");
4  mainDiv.setAttribute("draggable", "true");
5
6  const deleteButton = document.createElement("button");
7  deleteButton.classList.add("deleteButton");
8  mainDiv.appendChild(deleteButton);
9
10
11
12
13
14
```

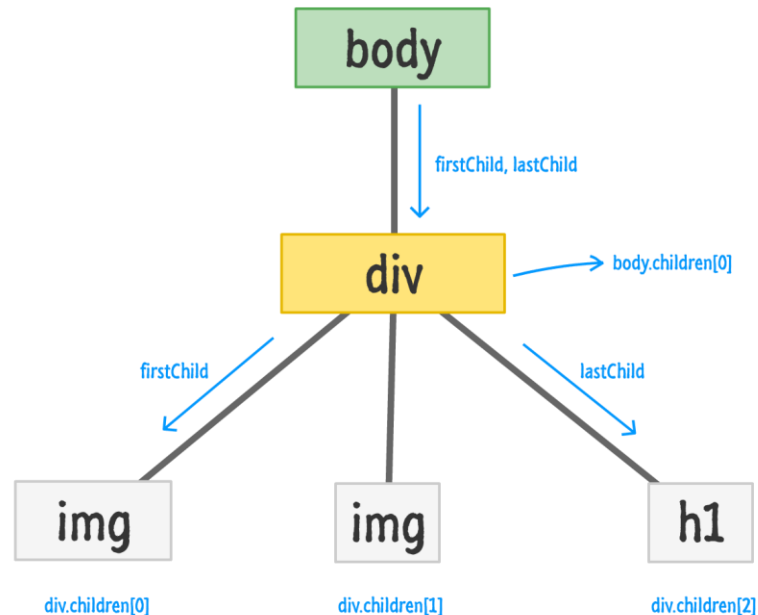
Create New Nodes

```
let paragraph = document.createElement("p");
paragraph.appendChild(p);
```

ADVANCED JAVASCRIPT PART - 61

Objectives

- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure
- innerText on HTML elements
- Create new DOM elements
- Traverse the DOM



Objectives

- Difference between the DOM and HTML
- Select elements from the DOM
- Describe the DOM structure
- innerText on HTML elements
- Create new DOM elements
- Traverse the DOM
- Investigate the living DOM in the browser

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  ...<body> == $0
    <div id="main" class="mx-auto">
      <h2 id="page-title">...</h2>
      <p class="description">Host and plan the perfect cigar party for all of your
        squirrely friends.</p>
      <!-- reviews go here -->
    </div>
    <script src="js/app.js"></script>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```