

Introduction to the DOM



[Download Demo Code](#)

Goals

- Understand what the DOM is
- Select HTML elements using document methods
- Compare and contrast elements and nodes

What Is The Dom?

- Document Object Model
- It is the programming interface for HTML
- A representation of our HTML that can be accessed using JavaScript

How does it get created?

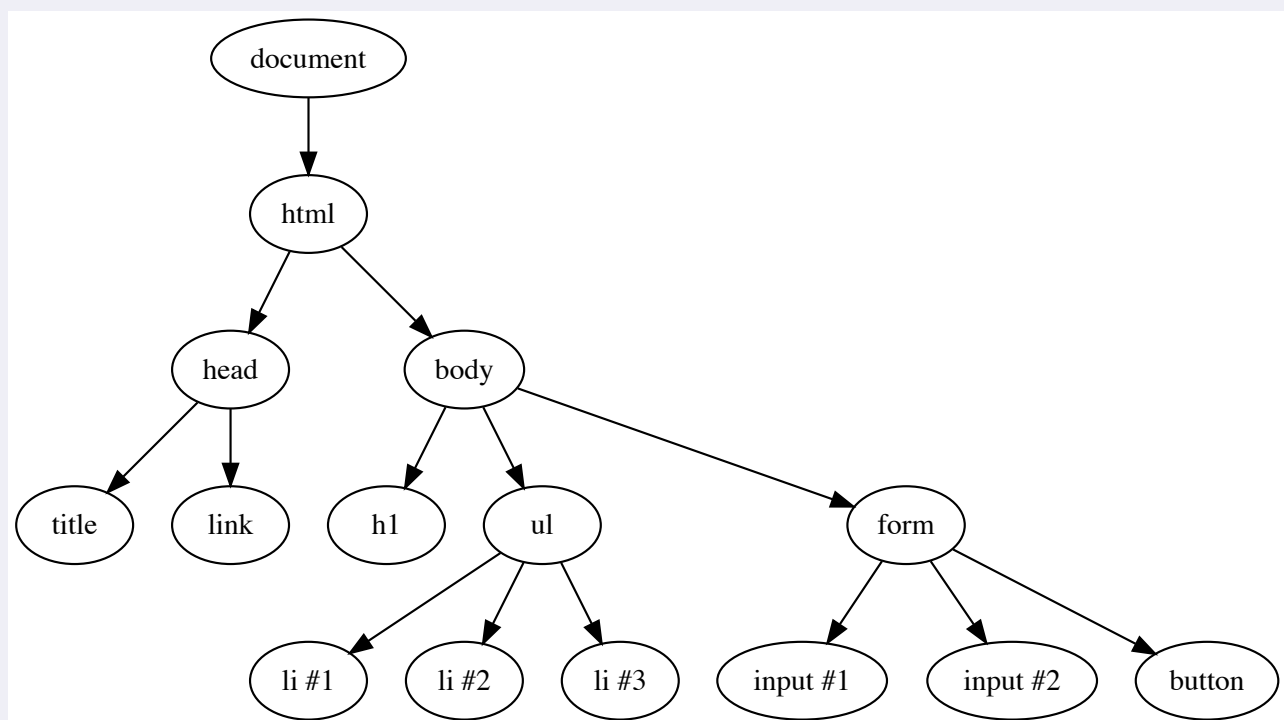
- When a web page is loaded, the browser creates the DOM for that specific page
- This allows for the creation of dynamic web pages where users can interact with the page

What does it look like?

```
...▼<body class="default-theme des-mat" style="background: rgb(255, 255, 255);">
  <div id="prpd"></div>
  ▼<div class id="mngb">
    ▼<div id="gb" class="gb_T">
      ►<div class="gb_nb gb_Pg gb_R gb_Og gb_Sg gb_T" style="min-width: 241px;
        ">...</div>
      </div>
    </div>
    <span id="prt"></span>
    ►<div id="TZA4S">...</div>
    <textarea name="csi" id="csi" style="display:none"></textarea>
    ►<script nonce="MHwVu6oUuFKDgG6HDEGpzQ==">...</script>
    ►<div id="xjsd">...</div>
    ►<div id="xjsi">...</div>
    <script src="/xjs/_/js/k=xjs.ntp.en_US.4JEq6NIP0CY.0/m=spch/am=gAgSMw/rt=j/
      d=1/exm=sx,jsa,ntp,d,csi/ed=1/rs=ACT90oGmcAW5XA4yhGQRyZR_wRqm9_fi3w?xjs=s1"
```

The DOM under the hood

The structure of the DOM uses something called a tree, where the topmost node is the **document** object.



Using the `document` object

- the **document** object represents the web page that has been loaded
- it acts as the "starting point" for access to the DOM.

The things we can do with the `document` object

- Finding elements
- Making new elements
- Updating elements
- Changing properties on elements
- Listening for events like clicks

Selecting Elements in the DOM

How To Select Elements In The Dom

To access the DOM, we make use of the document object.

This object has properties and functions that we use to access our HTML elements which we can manipulate with JavaScript.

Different Methods

JavaScript has quite a few different methods for selecting elements in the DOM

We're going to be starting with one method called ***getElementById***.

`getElementById`

`getElementById` accepts a string which is the name of an ***id*** in the DOM.

It finds the first matching id

```
document.getElementById("main");
```

What do we get back?

We get back a special object called an `HTMLElement`.

The exact kind of object we get back will depend on what we select (`HTMLDivElement` vs `HTMLParagraphElement`)

This special object contains quite a few helpful methods that we will see later!

`getElementsByName`

`getElementsByName` accepts a string which is the name of an element in the DOM.

It returns a list of all of the elements that match the string passed to the function

```
document.getElementsByName("li");
```

What do we get back?

This function returns an `HTMLCollection` to us!

It looks a lot like an array, and you can access it at a specific index or use a for loop

However, you can not use common methods like ***push***, ***pop***, ***indexOf*** or ***includes***

`getElementsByClassName`

`getElementsByClassName` accepts a string which is the name of an element in the DOM.

It returns a list of all of the elements that have a class attribute, which matches the string passed to the function

```
document.getElementsByClassName("heading");
```

What do we get back?

Just like ***getElementsByName***, we get back a special kind of array called an `HTMLCollection`.

Don't get too caught up in the difference between an array and an `HTMLCollection`, just know that you can not use almost all array methods on these special collections.

`querySelector`

`querySelector` accepts a string which is a valid CSS selector

It returns the first element that matches the CSS selector passed to the function.

```
document.querySelector("#main");
document.querySelector("h2.section-heading");
```

Just like ***getElementById***, this function returns a special `HTMLElement` object to us.

`querySelectorAll`

`querySelectorAll` accepts a string which is a valid CSS selector

It returns all the elements that matches the CSS selector passed to the function.

```
document.querySelectorAll("li");
document.querySelectorAll("ul .nav-links");
```

What do we get back

This function returns a `NodeList` to us!

It looks a lot like an array, and you can access it at a specific index or use a for loop

However, you can not use common methods like ***push***, ***pop***, ***indexOf*** or ***includes***

It's almost identical to an `HTMLCollection` except it can include special kinds of nodes. You will not need to worry about this for now!

Dom Manipulation

Now that we know how to select elements - let's change them!

Don't worry it's not permanent

Modifying Properties with `innerText`

```
let h1 = document.querySelector("h1");
h1.innerText = "Something new!"
```

Recap

- The DOM allows us to use JavaScript to manipulate HTML
- We can use methods like `querySelector` to access elements on the page
- Using the DOM we can modify elements and attributes