Day 21 - Document Object Model (DOM)

Document Object Model

An HTML document is structured as a JavaScript object. Every HTML element has different properties which can help to manipulate it. It’s possible to get, create, append or remove HTML elements using JavaScript. Selecting an HTML element using JavaScript is similar to selecting using CSS. To select an HTML element, we use tag name, id, class name, or other attributes.

Getting Element

We can access an already created element or elements using JavaScript. To access or get elements, we use different methods. The code below has 4 h1 elements. Let’s see the different methods to access the h1 elements.

```
<!DOCTYPE html>
  <html lang="en">
    <head>
      <title>Document Object Model</title>
    </head>
    <body>

     <h1 class='title' id='first-title'>First Title</h1>
     <h1 class='title' id='second-title'>Second Title</h1>
     <h1 class='title' id='third-title'>Third Title</h1>
     <h1></h1>

    </body>
  </html>
```

Getting elements by tag name

getElementsByTagName(): Takes a tag name as a string parameter and returns an HTMLCollection object. An HTMLCollection is an array-like object of HTML elements. The length property provides the size of the collection. Whenever we use this method we access the individual elements using index or after loop through each individual item. An HTMLCollection doesn’t support all array methods therefore we should use a regular for loop instead of forEach.

```
// syntax
document.getElementsByTagName('tagname')
```

```
document.addEventListener("DOMContentLoaded" , () => {
    const allTitles = document.getElementsByTagName('h1')

    console.log(allTitles)
    console.log(allTitles.length)

    for(let i = 0; i < allTitles.length; i++) {
        console.log(allTitles[i])
    }
})
```

Getting elements by class name

getElementsByClassName(): Returns an HTMLCollection object, which, again, is an array-like list of HTML elements.

```
// syntax
document.getElementsByClassName('classname')
```

```
const allTitles = document.getElementsByClassName('title')

console.log(allTitles) //HTMLCollections
console.log(allTitles.length) // 4

for (let i = 0; i < allTitles.length; i++) {
  console.log(allTitles[i]) // prints each elements in the HTMLCollection
}
```

Getting an element by id

getElementsById(): targets a single HTML element. We pass the id without # as an argument

```
// syntax
document.getElementById('id')
```

```
let firstTitle = document.getElementById('first-title')
console.log(firstTitle) // <h1>First Title</h1>
```

Getting elements by using querySelector methods

querySelector(): Can be used to select HTML elements by tag name, id or class. If the tag name is used it selects only the first element.

```
let firstTitle = document.querySelector('h1') // select the first available h1 element
let firstTitle = document.querySelector('#first-title') // select id with first-title
let firstTitle = document.querySelector('.title') // select the first available element with class title
```

querySelectorAll: Can be used to select html elements by tag name or class. Returns a nodeList which is an array-like object that supports array methods. We can use for loop or forEach to loop through nodeList elements.

```
const allTitles = document.querySelectorAll('h1') # selects all the available h1 elements in the page

console.log(allTitles.length) // 4
for (let i = 0; i < allTitles.length; i++) {
  console.log(allTitles[i])
}

allTitles.forEach(title => console.log(title))
const allTitles = document.querySelectorAll('.title') // the same goes for selecting using class
```

Adding attribute

An attribute is added in the opening tag of HTML which gives additional information about the element. Common HTML attributes are id, class, src, style, href, disabled, title, alt. Lets add id and class to the fourth title.

Adding attribute using setAttribute

setAttribute(): Allows you to set any HTML attribute. Takes two parameters; the type of the attribute and the name of the attribute.

```
const titles = document.querySelectorAll('h1')
titles[3].setAttribute('class', 'title')
titles[3].setAttribute('id', 'fourth-title')
```

Adding class using classList

classList: This method is good for appending an additional class. It doesn't override the original class and instead adds it after the existing class for the element.

```
titles[3].classList.add('title', 'header-title')
```

Removing class using remove

Similar to adding a class, we can remove a class from an element.

```
titles[3].classList.remove('title')
```

Adding Text to HTML element

An HTML is a building block of an opening tag, a closing tag, and text content. We can add text content using the property *textContent* or *innerHTML

Adding Text content using textContent

The *textContent* property is used to add text to an HTML element.

```
const titles = document.querySelectorAll('h1')
titles[3].textContent = 'Fourth Title'
```

Adding Text Content using innerHTML

Most people get confused between textContent and innerHTML. textContent is meant to add text to an HTML element, however, innerHTML can add text, an HTML element, or elements as a child

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>JavaScript for Everyone:DOM</title>
  </head>
  <body>
    <div class="wrapper">
        <h1>Asabeneh Yetayeh challenges in 2020</h1>
        <h2>30DaysOfJavaScript Challenge</h2>
        <ul></ul>
    </div>
    <script>
    const lists = `
    <li>30DaysOfPython Challenge Done</li>
            <li>30DaysOfJavaScript Challenge Ongoing</li>
            <li>30DaysOfReact Challenge Coming</li>
            <li>30DaysOfFullStack Challenge Coming</li>
            <li>30DaysOfDataAnalysis Challenge Coming</li>
            <li>30DaysOfReactNative Challenge Coming</li>
            <li>30DaysOfMachineLearning Challenge Coming</li>`
  const ul = document.querySelector('ul')
  ul.innerHTML = lists
    </script>
  </body>
</html>
```

We can also remove all of the children of a parent element using innerHTML:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>JavaScript for Everyone:DOM</title>
  </head>
  <body>
    <div class="wrapper">
        <h1>Asabeneh Yetayeh challenges in 2020</h1>
        <h2>30DaysOfJavaScript Challenge</h2>
        <ul></ul>
    </div>
    <script>
    const lists = `
    <li>30DaysOfPython Challenge Done</li>
            <li>30DaysOfJavaScript Challenge Ongoing</li>
            <li>30DaysOfReact Challenge Coming</li>
            <li>30DaysOfFullStack Challenge Coming</li>
            <li>30DaysOfDataAnalysis Challenge Coming</li>
            <li>30DaysOfReactNative Challenge Coming</li>
            <li>30DaysOfMachineLearning Challenge Coming</li>`
  const ul = document.querySelector('ul')
  ul.innerHTML = lists
    </script>
  </body>
</html>
```

Adding style

Adding Style Color

In the example below, if the element has an even index, we'll give it a green color, else, red.

```
const titles = document.querySelectorAll('h1')
titles.forEach((title, i) => {
  title.style.fontSize = '24px'
  if (i % 2 === 0) {
    title.style.color = 'green'
  } else {
    title.style.color = 'red'
  }
})
```

Adding Style Background Color

```
const titles = document.querySelectorAll('h1')
titles.forEach((title, i) => {
  title.style.fontSize = '24px' // all titles will have 24px font size
  if (i % 2 === 0) {
    title.style.backgroundColor = 'green'
  } else {
    title.style.backgroundColor = 'red'
```

```
  }
})
```

Adding Style Font Size

```
const titles = document.querySelectorAll('h1')
titles.forEach((title, i) => {
  title.style.fontSize = '24px' // all titles will have 24px font size
  if (i % 2 === 0) {
    title.style.fontSize = '20px'
  } else {
    title.style.fontSize = '30px'
  }
})
```