Getting Started With The DOM

Justice Reskill

What Is the DOM?

DOM stands for Document Object Model, which is an programming interface for HTML and XML in Javascript. In layman's terms, that means you can modify the html on the page using Javascript!

DOM Uses Script Tags

To use javascript in your HTML, it must reside inside <script></script> tags that can be place in either or your head or after your body tag.

```
<!DOCTYPE html>
<html>
  <head>
    <script type=”text/javascript”>
        <!-- javascript in here -->
    </script>
  </head>
  <body>
   </body>
    <script type=”text/javascript”>
        <!-- javascript in here -->
    </script>
</html>
```

Document Object

To access the DOM, all browsers have a built in object called ‘document’.

Remember, document is like any other object and has function and variables.

What happens when you console.log document?

```
document
```

```
console.log(document);
```

Getting The Element By ID

Remember the ID attribute that an element can have and how the ID must be unique?

We can use that ID to reference that html element in javascript and assign it to a variable.

The function used is ‘**getElementById**’ on the document object. What happens when you console.log the element?

```
<script type=”text/javascript”>

var element =  document.getElementById(‘p1’);

console.log(element);

</script>
.....
<body>
        <p id=”p1”>Some Text</p>
 </body>
```

Getting The Content

The retrieved element is only an object, and all the html objects will have the same functions and properties.

To get the content that is inside an element, use the objects property innerHtml.

What happens when you console.log element.innerHtml?

```
<script type=”text/javascript”>

var element =  document.getElementByID(‘p1’);

console.log(element.innerHtml);

</script>
.....
<body>
        <p id=”p1”>Some Text</p>
 </body>
```

Update The Content

The object is a representation of the html object in code. Any modifications to the object in javascript will directly modify the element in html.

When you set the innerHtml to new content in javascript, it will replace all the content inside the text.

```
<script type=”text/javascript”>

var element =  document.getElementByID(‘p1’);

element.innerHtml = “New Text”;

</script>
.....
<body>
        <p id=”p1”>Some Text</p>
 </body>
```

Update The Style

Another example of how the object is directly linked to an html element is updating its css.

When you update any of the style attribute on the object, it will also change the css style of the responding element.

```
<script type=”text/javascript”>

var element =  document.getElementById(‘p1’);

element.style.color = “red”;

element.style.fontSize = “15px”;

</script>
.....
<body>
        <p id=”p1”>Some Text</p>
 </body>
```

Any Element Called By The ID

Any element can be called by its id. This includes divs, form elements, even the body!

Remember, we always only want to assign a unique id to each html element that we want to reference.

```
<script type=”text/javascript”>

var element1 =  document.getElementById(‘p1’);

var element2 =  document.getElementById(‘b3’);

var element3 =  document.getElementById(‘d5’);

</script>
.....
<body id=”b3” >
        <p id=”p1”>Some Text</p>
        <div id=”d5”>Some Content</div>
 </body>
```

Getting The Content Of Inputs

Input elements are slightly different than html elements. Instead of using the innerHTML to the content, you will use the value attribute.

This will return what the user has inputted.

```
<script type=”text/javascript”>

var input1 =  document.getElementById(‘i1’);

console.log(input1.value);

</script>
.....
<body>
        <input id=”i1” type=”text” />
 </body>
```

TextArea Input

TextArea is also used to get the value for getting the users input.

```
<script type=”text/javascript”>

var input1 =  document.getElementByID(‘i1’);

var text1 =  document.getElementByID(‘t1’);

console.log(input1.value);

console.log(text1.value);

</script>
.....
<body>
        <input id=”i1” type=”text” />
        <textarea id=”t1” ></textarea>
 </body>
```

Update Input Content

To update the content of an input element, we can text the value to a new string.

How do you think we could clear user input with this method?

```
<script type=”text/javascript”>

var input1 =  document.getElementByID(‘i1’);

input1.value = “New Text”;

</script>
.....
<body>
        <input id=”i1” type=”text” />
 </body>
```

Getting Multiple Elements

While getElementByID only gets one element, we can get multiple elements with **getElementsByClassName**.

This will return an array of objects that represents the html with that class name. We can iterate over that array with a for loop.

```
<script type=”text/javascript”>

var elements =
document.getElementsByClassName(‘items’);

for (var i =0; i < elements.length; i++) {
        var tmp = elements[i];
        console.log(tmp.innerHtml);
}
</script>
.....
<body>
        <p class=”items”>Some Text 1</p>
        <p class=”items”>Some Text 2</p>
        <p class=”items”>Some Text 3</p>
        <p class=”items”>Some Text 4</p>
 </body>
```

Same Update Rules Apply

Even when the elements are retrieved by their class name inside of array, they are still linked to the html element on the page.

Meaning any modifications we make the object in the array will be reflected in the html.

```
<script type="text/javascript">

var elements =
document.getElementsByClassName('items');

for (var i =0; i < elements.length; i++) {
        var tmp = elements[i];
        tmp.innerHtml = i;
        tmp.style.fontSize = "30px';
}
</script>
.....
<body>
        <p class="items">Some Text 1</p>
        <p class="items">Some Text 2</p>
        <p class="items">Some Text 3</p>
        <p class="items">Some Text 4</p>
 </body>
```

Getting Multiple Elements Part 2

Similar to getElementsByClassName, with form elements we can use the name attribute in the html with **getElementsByName**.

This works perfect for unique form elements like radio buttons.

Remember, because its a form element, we use the value and not innerHtml

```
<script type=”text/javascript”>

var elements =  document.getElementsByName(‘r1’);

for (var i =0; i < elements.length; i++) {
        var tmp = elements[i];
        if(tmp.checked) {
                alert(“Select item:” + tmp.value);
        }
}
</script>
.....
<body>
        <input name=”r1” type=”radio” value=”1” />
        <input name=”r1” type=”radio” value=”2” />
        <input name=”r1” type=”radio” value=”3” />
        <input name=”r1” type=”radio” value=”4” />
 </body>
```

HTML Calling Javascript Functions

HTML has attributes that can call javascript button. This is called event handling.

One such event is the onclick attribute, when a user clicks on the element, it will call the associated function defined in javascript.

```
<script type=”text/javascript”>

function clickedButton() {

        alert(“Hello Word!”);

}

</script>
.....
<body>
  <button type=”button” onclick=”clickedButton()” >
        Click Me
  </button>
 </body>
```

Form Input On HTML Events

We can use these action events for getting and processing data.

In this example, when the onclick attribute calls the clickedButton function, it will get the input by its id, and then alert the value the user has inputted to the screen.

```
<script type=”text/javascript”>

function clickedButton() {

        var input1 =  document.getElementByID(‘i1’);

        alert(input1.value);

}

</script>
.....
<body>

  <input id=”i1” type=”text” />

  <button type=”button” onclick=”clickedButton()” >
        Click Me
  </button>
 </body>
```

Many Html Event Handlers

There are many types of event handlers we can use. Some of the examples include but are not limited to:

- **onmouseover** - When a users mouse cursor goes over the element
- **ondrag** - When a user clicks down and attempts to drag an element
- **onchange** - When form input has been changed

For a larger list of events, see here:
https://www.w3schools.com/tags/ref_eventattributes.asp

Creating Elements

HTML elements can also be created in the DOM using the createElement function.

We simply have to pass in the element tag we want to create. The same properties as a normal html element still apply.

```
<script type=”text/javascript”>

var element =  document.createElement(‘p’);

console.log(element);

element.innerHtml = “New Content”;

element.style.fontSize = “30px”;

</script>
.....
<body id=”b1”>
 </body>
```

Add Elements To HTML

Once an element is created, is can be added to the page by appending it to another element.

We can do this with the appendChild on a current element.

```
<script type=”text/javascript”>

var body =  document.getElementByID(‘b1’);

var element =  document.createElement(‘p’);

element.innerHtml = “New Content”;
element.style.fontSize = “30px”;
body.appendChild(element);
</script>
.....
<body id=”b1”>
 </body>
```

Adding Multiple Elements

You can create as many elements as you want and append them to an element.

Notice in this example, we create a paragraph, a div and h1. We append the paragraph to the div, and then we append the h1 and div to the body.

```
<script type=”text/javascript”>

var body =  document.getElementByID(‘b1’);

var paragraph =  document.createElement(‘p’);

paragraph.innerHtml = “New Content”;


var title =  document.createElement(‘h1’);

title.innerHtml = “My Header”;


var container =  document.createElement(‘div’);

container.appendChild(paragraph);



body .appendChild(title);

body .appendChild(container);

</script>
.....
<body id=”b1”>
 </body>
```

innerHtml vs appendChild

innerHTML will replace all the contents inside an element. This means the old content is deleted and replaced with new content.

appendChild will add content to the current content. The current content will NOT be deleted.

```
<script type=”text/javascript”>

var body =  document.getElementByID(‘b1’);

var paragraph =  document.createElement(‘p’);

paragraph.innerHtml = “New Content”;


var title =  document.createElement(‘h1’);

title.innerHtml = “My Header”;


body.innerHtml = “”;

body .appendChild(title);

body .appendChild(paragraph);

</script>
.....
<body id=”b1”>
        <h1 >Hello World</p>
 </body>
```