

3.13: JavaScript Examples

DOM Navigation

Now we will look at some other techniques we can use to interact with the DOM using JavaScript. We specified we wanted to use a particular element by referencing its ID (`getElementById()`) but we can also interact with groups of elements that share a special type or class. To reference by type we would call the `getElementsByTagName` method, giving us a list of elements that we can review, or modify:

```
1. <div id="output"></div>
2. <input type="text" size="20"><br>
3. <input type="text" size="20"><br>
4. <input type="text" size="20"><br><br>
5. <input type="button" value="Submit Form">
6. <script>
7. var x=document.getElementsByTagName("input");
8. document.getElementById("output").innerHTML = ("Number of elements: "+x.length);
9. </script>
```

Perhaps we only want elements inside of another specific element. We can grab inputs from a particular form. To do this, we change our reference from the document as a whole to the element we want to inspect:

```
1. <div id="output"></div>
2. <form id="form1">
3. <input type="text" size="20"><br>
4. <input type="text" size="20"><br>
5. <input type="text" size="20"><br><br>
6. <input type="button" value="Submit Form">
7. </form>
8. <form id="form2">
9. <input type="text" size="20"><br>
10. <input type="text" size="20"><br>
11. <input type="text" size="20"><br><br>
12. <input type="button" value="Submit Form">
13. </form>
14. <script>
15. var x=document.getElementsByTagName("input");
16. document.getElementById("output").innerHTML = ("Number of elements: "+x.length+"<br>Number in form2:
    "+form2.getElementsByTagName("input").length);
17. </script>
```

In this example all we changed was doubling our inputs and putting them into forms. We also used form2 as our reference instead of document to create our second output, although this time we just tacked it onto the end without creating a second variable. You can see in this example that not only can we concat function results easily in JavaScript, we were still able to refer to the `length()` method, even though it was added to the end of the `getElementsByTagName` method. This chain is perfectly valid in JavaScript, and will always be evaluated from the inside out just like sets of parenthesis. In this case, counting the inputs in form2 occurred first, creating a temporary object that `.length` was instructed to access.

Events

It is important to keep in mind that JavaScript is not living within the confines of an object or function and will be processed as soon as it is loaded. Depending on its placement in the document this means our script may attempt to utilize something in the DOM that has not been created yet as the page is still loading. If we want our code to wait until the full page is complete, we need to wrap our code into one of these two items and then use a trigger, or event, to signify when the page is ready. One approach to this is adding an `onload` attribute to our body tag:

1. `<body onload="ourFunction()">`

We can even monitor events that do not involve the document itself like the location of the mouse. Monitoring the location of the mouse may have little practical application in the average website, but it can be useful in web based games and analytics applications, where knowing the position or path the mouse is on can influence how the page acts:

1. `<script>`
2. `function getCoords(event){`
3. `var x=event.clientX;`
4. `var y=event.clientY;`
5. `alert("X: " + x + ", Y: " + y);`
6. `}`
7. `</script>`
8. `<p onmousedown="getCoords(event)">Click anywhere on this sentence to see the x and y coordinates of the mouse. Clear the alert and try again to see the numbers change.</p>`

When an event we are waiting for occurs, we can react by executing other steps before it is processed, or even replace it. We can see an example of this by saying goodbye before a visitor leaves our page:

1. `<script>`
2. `function sayGoodbye(){`
3. `alert("Thanks for visiting!");`
4. `}`
5. `</script>`
6. `<body onunload="sayGoodbye()">`

To stop an event, we can use the `preventDefault()` method, which allows us to supersede what would have taken place. For instance, we may want to use a link to trigger a JavaScript function, but do not want the page to reload, or follow the URL that was clicked on. In this example, we could have the following:

1. `<script type="text/javascript">`
2. `addEventListener("load",function(){`
3. `var link= document.getElementById("google");`
4. `links.addEventListener("click",function(e){`
5. `e.preventDefault(); //prevent event action`
6. `}`
7. `});`
8. `</script>`
9. `<body>`
10. `Google`
11. `</body>`

Geolocation

Devices and browsers that have access to GPS, cellular, or wireless router location data may provide that information to web pages (usually also in conjunction with user permission). When this data is available, our webpage can receive the information from the device. We can use it to improve the user experience, or provide features that are location dependent, like finding the nearest restaurant, or offering local deals. This example demonstrates how to access the information if it is available. Once we have the coordinates, we can write additional scripts on our backend (PHP and/or MySQL) that use them in order to tailor the user experience.

Keep in mind most browsers/devices allow the user to turn these features on and off, and not all devices will have access to or share this type of information. Because of this, it should not be a required or relied upon features unless you are sure your target users will be on such a device, and are aware that this type of feature is a requirement.

1. `<p id="text">Where am I?</p>`
2. `<button onclick="getLocation()">Find Me</button>`
3. `<script>`

```
4. var response=document.getElementById("text");
5. function getLocation(){
6. if(navigator.geolocation){
7. navigator.geolocation.getCurrentPosition(position);
8. }
9. else{response.innerHTML="Geolocation is not available from this device or browser.";}
10. }
11. function position(loc){
12. response.innerHTML="Latitude: " + loc.coords.latitude +
13. "<br>Longitude: " + loc.coords.longitude;
14. }
15. </script>
```

3.13: JavaScript Examples is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- **3.13: JavaScript Examples** by [Michael Mendez](#) has no license indicated.