Getting your website to do things using JavaScript

Objectives

- What is javascript
- Basic functions

What is JavaScript

JavaScript is a computer language that brings interactivity to otherwise static web pages. JavaScript is the third pillar of building web pages. Nearly every website on the internet is using Javascript for some purpose. On a web page, whether it is advertising, user input validation or some other complex action, it was probably programmed in JavaScript.

Up to this point we have only been using HTML tags and CSS styles to change the look of our page. In this module we are going to look at how we can use some simple code in JavaScript to make our web page interactive.

Red Alert!

So let's start with the simplest example of JavaScript on your page. We will add a simple alert that will run as soon as your web page has completed loading. Let's add the following onload attribute to the <body> tag.

```
<body onload="alert('Page is Loaded');">
```

When you save and reload your page, you should get a popup window that let's you know the page is loaded.

But wait...did your browser warn you about running scripts or ActiveX controls. Maybe you were running Internet Explorer and this warning came up:

IE warning

It's OK...but it is good to point out that there can be good JavaScript and bad JavaScript. There are some emails that will use local JavaScript to try to steal personal and private information from you in order to do bad things. Since you are running this JavaScript locally in your browser (rather than pulling it from the internet) IE is just making sure you know there is potentially a risk.

How do we add JavaScript to our Page

It is interesting that we just started writing code in our <body> tag. For a simple interaction, this may be acceptable, but you likely will not want a lot of code in your tags.

We have already seen that we can add code directly to your tag. Let's fix our <body> tag to give us flexibility for more code. Change the onload attribute to the following:

```
<body onload="onLoad()">
```

This will tell the page to call the "onLoad" function when our page is done loading. Let's add that "onLoad" function to our page. We are going to do this at the end of the <head> tag.

```
<head>
    <title>My First Web Page</title>
    ...
    <script>
        function onLoad() {
            alert("Function onLoad() called");
        }
    </script>
</head>
```

Now, we could also be explicit and tell the browser that our script is in fact JavaScript and you may see this in many example scripts, but it isn't required.

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

Keep in mind that JavaScript is case sensitive, so "OnLoad" and "onLoad" are not the same thing. If your code isn't doing what you expected, you may want to check the spelling and casing of your function.

This example put the <script> code in the <head> tag. We could also put the <script> in the <body> tag. When placing scripts in the <body> tag, it is a good practive to put all of the script at the bottom to improve page load performance.

If you have a lot of script on a single page, your page can start to get out of hand. In this case, it might be a good idea to move your <script> code out to another file and simply add a reference in your page.

Let's do that with the script we have right now. In VS Code, add a new file and we'll call it "app.js". Next, copy the function to the JavaScript file.

```
function onLoad() {
    alert("Function onLoad() called");
}
```

Be careful not to include <script> tags in the external file.

Next, we need to modify the script tag in our <head> to load the script from an external file.

```
<head>
    <title>My First Web Page</title>
    ...
    <script src="app.js" type="application/javascript"></script>
</head>
```

Save and reload the **index.html** page, and your page should behave just as it did before you moved the code to the **app.js** file.

One of the advantages to putting your script in an external file is that it separates your code from your text. Additionally, it allows you to reuse the code from one page to another. If you are in the habit of copying your scripts from page to page, if you decide you want to change the functionality, you will have to change your code in multiple places which is bad. If you find yourself copying code, you should start thinking about external files.

Making your content dynamic

Popping up annoying alerts is not the best use of JavaScript, so let's take a look at some better uses of JavaScript.

First we will take a look at dynamic content. We are going to start simple and update content when you click a button. First, let's add a placeholder for some content in the footer.

```
    <footer>
        Copyright 2015
        <p id="timestamp"></p>
    </footer>
```

Notice we gave an "id" to the <p> tag in the footer. This is going to give us the ability to modify that tag specifically. Let's modify the JavaScript code and then look at what is going on.

```
function onLoad() {
    document.getElementById("timestamp").innerHTML = Date();
}
```

Save and reload the page. Next, examine the footer on your page and notice that the content shows the Date and Time of the page load. Once the document was loaded, we asked for the element that had an id of "timestamp" and then modified all of the HTML inside to be the result of the "Date()" function.

It's important when doing code like this that you wait until the document has completed loading. If the code ran before the <footer> had been loaded, the code would have executed and nothing would have happened.

Responding to events

So far, we have looked at running code when a page loads. However, there are a lot of interesting things happening when a user is looking at our page. If they are using a mouse, there are events that can set code in motion when the mouse "hovers" over an element or "clicks" an element.

If you create a form for people to enter data, you might want to make sure they are entering valid data. Let's add to our web page and see what that might look like. We are going to make a textbox that will check that you entered a number between 1 and 10.

First, let's add a textbox to collect data from the user. We'll add it to the top of our main section content. If we use the <input> tag, one way we can control what people enter is to use the "type" attribute. In this case, we will set the type equal to "number". For some browsers, this will restrict the input to numeric or hint that the onscreen keyboard should be numeric.

Next, let's add a place to put a message after we have checked the number.

Finally, we'll add a button to allow the user to submit the value that was entered into the textbox.

```
<section class="col-sm-6 col-sm-push-3">

    <input id="smallnumber" type="number">
    <p id="numberMessage"></p>
    <button onclick="checkNumber()">Submit</button>
...
```

Save your work and refresh the browser and you should see these new elements on your web page. Right now, you might be able to add characters and really big numbers. We will need to add script that tells you wether or not the number is a valid number between 1 and 10.

Next, we need to create our "checkNumber" function. Open up the "app.js" file and create a function called "checkNumber" at the end of the file. Remember casing is important here.

```
function onLoad() {
    ...
}

function checkNumber() {

}
```

Next, let's create some variables to hold our data.

```
function checkNumber() {
    var theNumber, theMessage;

}
```

Now, use the "getElementById" function to get the value of the textbox that the user entered.

```
function checkNumber() {
    var theNumber, theMessage;

    theNumber = document.getElementById("smallnumber").value;
}
```

Next, we will use an "if" statement to check the value of "theNumber" to see if it passes the check for being between 1 and 10. We will make three checks. If any of them is true, then we don't need to test any further...it's a bad number. We do this with a logical OR statement. In JavaScript this is "||".

So, first we want to see if it is "not a number" (that's the isNaN). If that is not true, then we will check to see if theNumber is less than 1. If that check is not true, then we check to see if theNumber is greater than 10. If all of these checks are not true, then our number is good and we set the message to tell the user gave us a good number. If any one of these conditions is true, we don't need to check any further, we just tell the user their number was bad.

Finally, we use the "getElementById" to set the message on our web page to give feedback to our user.

```
function checkNumber() {
    var theNumber, theMessage;

    theNumber = document.getElementById("smallnumber").value;

    // If x is Not a Number or less than one or greater than 10
    if (isNaN(theNumber) || theNumber < 1 || theNumber > 10) {
        theMessage = "Number was expected to be between 1 and 10";
    } else {
        theMessage = "Number is Good";
    }
    document.getElementById("numberMessage").innerHTML = theMessage;

}
```

Save the **app.js** and **index.html** files and reload the **index.html** page in your browser. Try entering numbers in the textbox that are out of the range of what we put in code. Did the message get inserted in the HTML under the textbox?

Trust No One

One thing to keep in mind is that just because you use JavaScript to keep a user from entering an unexpected value, NEVER trust that number to be sent back to your server. If a user turned JavaScript off your code would not run and you could get invalid data. If an evil user used a tool to modify the value after your JavaScript ran, he could potentially send harmful data to your server. Always be careful of what data is sent to your server.

In this module we covered the basics of JavaScript to interact with your user. We displayed a pop-up to the user, we added the current time to the page on the fly, and finally we looked at validating input by the user.

In the next module, we will look at using JavaScript to bring data from other web sites into our web page.