Javascript and Web Pages!

Up until now, we've worked either only with javascript, or only with html/css. What makes sites so powerful is that we can combine the two!

Where to put javascript in your page:

Javascript, like CSS can live either inside our `<body>`, inside our `<head>`, or in a separate file with a `.js` file extension. For our purposes, we're going to use that separate file (so nicely provided for us by codepen).

However, before we move on too much further, let's see what code inside the `<body>` and `<head>` looks like:

Javascript in the `<head>`

Just like how we put css in our header, we can also put javascript there, but instead of placing it inside of `<style>` elements, we place it inside of `<script>` elements! Let's take a quick look at linking up a button and a function that says hi!

```
<!doctype html>
<html>
  <head>
    <script>
      const sayHello = function(){
        console.log('Hello friends!');
      }
    </script>
</head>

<body>
  <button onClick='sayHello()'> Press me to say hi! </button>
</body>

</html>
```

In the above code, we have a basic button. That button is powered by the `sayHello()` function described up in the `<script>` element in the header. When we click the button `Press me to say hi!`, that triggers the function `sayHello()`.

Javascript in the `<body>`

Writing javascript in our body is also doable, but can become a bit confusing, mostly because it's scattered throughout our whole site's page. Nonetheless, sometimes you may just want to have a tiny bit of code on your page:

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <script>
      const sayHello = function(){
        console.log('Hello friends!');
      }
    </script>
    <button onClick='sayHello()'> Press me to say hi! </button>
  </body>

</html>
```

Notice that when the page renders, even though we have a lot more in our body now, we still only see the button! That's because what's inside of a script tag is not actually rendered.

Working with built in Javascript Functions:

Revisiting Alert and Confirm:

Previously, we've seen how alert and confirm work. Now we're going to actually use them and understand what's actually happening!

A lot of times on your site, you'll want to ask a user something. For now, let's just look at a yes or no:

```
<!doctype html>
<html>
  <head>
    <script>
      const sayHello = function(){
```

```
        confirm('Hello friend! How are you?');
      }
    </script>
</head>

<body>
  <button onClick='sayHello()'> Press me to say hi! </button>
</body>

</html>
```

You've seen this kind of popup before. But what we haven't talked about before is what that `confirm` is returning. Remember how we could return values from our javascript functions? Guess what! `confirm` is returning something too!

```
<!doctype html>
<html>
  <head>
    <script>
      const sayHello = function(){
        var confirmResponse = confirm('Hello friend! How are you?');

        console.log('Confirm response is: ', confirmResponse)
      }
    </script>
</head>

<body>
  <button onClick='sayHello()'> Press me to say hi! </button>
</body>

</html>
```

Notice that the confirm returns either true or false! We can take that data, and use it with conditionals!

```
<!doctype html>
<html>
  <head>
    <script>
      const sayHello = function(){
        var confirmResponse = confirm('Hello friend! How are you?');

```

```
        // we could write "confirmResponse === true" but since it's already
        // either true or false, we can just leave it as "confirmResponse"
        if (confirmResponse) {
          alert('You pressed ok!')
        } else {
          alert('You pressed cancel!')
        }

        console.log('Confirm response is: ', confirmResponse)
      }
    </script>
  </head>

  <body>
    <button onClick='sayHello()'> Press me to say hi! </button>
  </body>

</html>
```

Take in strings with Prompt:

There are a number of ways to take in user data. Let's look at how to take in string data. Prompt prompts the user to enter something into a text field:

```
<!doctype html>
<html>
  <head>
    <script>
      const sayHello = function(){
        var promptResponse = prompt("Hi there! What's your name?",
                                    "Write some cool name here!");

        alert("Hi there " + promptResponse + ", pleased to meet you!")
        console.log("prompt response = " + promptResponse)
      }
    </script>
  </head>

  <body>
    <button onClick='sayHello()'> Press me to say hi! </button>
  </body>

</html>
```

Working with Document.write():

Sometimes you'll want to write something to the page. Javascript comes with a lot of built in functions! By using the built in function `document.write()`, we can actually use javascript to write something to our page! `document.write()` works a lot like `console.log()`, where whatever you put in the parentheses is what gets displayed, but this time, it'll get displayed on our page! Let's try it out:

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <p> I'm just some boring text that was written to the page with HTML</p>
    <script>
      document.write('I am a string, and I was written to the page with javascript!')
    </script>
  </body>

</html>
```

This is great because it will allow us to use javascript to write things to our page! We can also use elements within our `document.write`. By writing our elements in the function, we can use the power of HTML along with our javascript:

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <p> I'm just some boring text </p>
    <script>
      document.write('I am a <em> surprisingly </em> <strong> BOLD </strong> string!')
    </script>
  </body>

</html>
```

Since we've already covered functions, you're probably wondering "what if we put that `document.write` inside of a function?" That's a great question, could we use a function?

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <p>
      Oh man, we've got a ton of cool code coming up! There's so much cool stuff out there that we'll get to learn!
    </p>
    <script>
      const sayHelloOnThePage = function(){
        document.write("Hello friends! (Where'd all that other text go?)");
      }
    </script>
    <button onClick='sayHelloOnThePage()'> Press me to say hi! </button>
  </body>

</html>
```

You bet your bottom dollar we can! But what happened to our button and all of our text?! `document.write` is a very useful function for a lot of reasons, but, you should most definitely avoid trying to use it to write things to your page after your page has loaded because it ultimately rewrites the whole page! We'll get to how to do that a little bit later. For now, though, `document.write` is something that could be very useful to display a time, or a number of page visits, or any number of things when the page is first loading!