Day 8: Create a Button

by AvmnuSng

Button Basics in JavaScript

In this article, we discuss some basics of writing JS (JavaScript) and CSS code in an HTML file. Note that we assume a certain level of basic HTML knowledge.

Table Of Contents

Templates

Basic Format

We use the following template to write JavaScript and CSS code to an HTML file:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>

        <style>
            /* Write CSS styles here */
        </style>
    </head>
    <body>
        <script>
            /* Write JS code here */
        </script>
    </body>
</html>
```

We write our CSS style code between the `<style>` and `</style>` tags, and our JS code between the `<script>` and `</script>` tags.

Note: Any text between `<!--` and `-->` is considered to be an *HTML comment*. These comments won't render on the webpage, but we can read them if we view the page's source code. For content between tags that contain actual code (i.e., *style* and *script*), we enclose comments between `/*` and `*/`.

Working with Separate Documents

In an instance where all our code is located in separate files (i.e., we have a `.html` file with our HTML, a `.css` file with our CSS, and a `.js` file with our JS code), we use this template to tell our HTML file where to find the JS and CSS files:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>

        <!-- Link to the style sheet in the 'head' section -->
        <link rel="stylesheet" href="css-file-path" type="text/css">
    </head>

    <body>
        <!-- Link to the JS code in the 'body' section -->
        <script src="js-file-path" type="text/javascript">
        </script>
    </body>
</html>
```

Let's look at what this code does:

- By putting `<link rel="stylesheet" href="css-file-path" type="text/css">`, where `css-file-path` is the path of the `.css` file, in the *head* section (i.e., between the `<head>` and `</head>` tags), we're telling the document to use the style sheet at the location referenced by the `href` attribute.
- By putting `<script src="js-file-path" type="text/javascript">`, where `js-file-path` is the path of the `.js` file, in the *body* section (i.e., between the `<body>` and `</body>` tags), we're saying that we want to run a script using the JS code at the location referenced by the `src` attribute.

HTML Buttons

We use the `<button>` tag to create a clickable button that has the following optional attributes:

- `id`: the button's unique identifier within the page
- `class`: the CSS class(es) used to style the button

The text enclosed between the button's opening (`<button>`) and closing (`</button>`) tags is the label that displays on the button. We can also access this text using the *innerHTML* property of the JS button object (see the JavaScript section below). The basic syntax for an HTML button looks like this:

```
<button id="buttonIdentifier" class="buttonStyleClass">Click Me</button>
```

- EXAMPLE

The image below shows two lines of HTML code and the content they produce:

```
1 <p>Check out our button:</p>
2 <button id="regularButton">I'm a button!</button>
```

Check out our button:

I'm a button!

JavaScript Buttons

Consider the following code:

```
var clickMeButton = document.createElement('button');
clickMeButton.id = 'myButton';
clickMeButton.innerHTML = 'Click Me';
clickMeButton.style.background = '#4FFF8F';
document.body.appendChild(clickMeButton);
```

Now, let's walk through what it does:

1. `document.createElement('Button')` creates a clickable button object (*createElement('Button')*) referenced by the variable name ***clickMeButton***.
2. `clickMeButton.id = 'myButton'` sets the button's *id* to be `myButton`.
3. `clickMeButton.innerHTML = 'Click Me'` sets the button's inner HTML (i.e., the label we normally see between the HTML *button* tags) to say *"Click Me"*.
4. `clickMeButton.style.background = '#4FFF8F'` sets the button's background color to green. To style multiple attributes of our button using a style class, we would write `clickMeButton.className = 'myStyleClassName'` instead.
5. `document.body.appendChild(clickMeButton)` appends ***clickMeButton*** to the body of the document as a child.

Let's say we want to modify the label on an HTML button element with the *id* `myButton`. We simply use the *getElementById* method and pass the desired element's *id* as an argument:

```
var clickMeButton = document.getElementById('myButton');
clickMeButton.innerHTML = 'This is my new label text!';
```

- EXAMPLE

The image below shows some HTML code with a JavaScript script and the content they produce:

```
<!DOCTYPE html>
<html>
    <head />
    <body>
        <p>Check out our button:</p>
        <script>
            var clickMeButton = document.createElement('Button');
            clickMeButton.innerHTML = 'Click Me';
            clickMeButton.style.background = '#4FFF8F';
            document.body.appendChild(clickMeButton);
        </script>
    </body>
</html>
```

Check out our button:

Click Me

Styling Buttons with CSS

First, we want to define the style constraints for our button. We can do this in either of the two ways below.

1. Using an ID Selector

For instances where we want to apply a style to a single element within an HTML page, we use a CSS *id selector* using the syntax `#identifier` (where ***identifier*** is the *id* of the element to style within the page). We then follow it with a pair of curly braces that contain the desired style constraints for all elements within the container that has that identifier. For example:

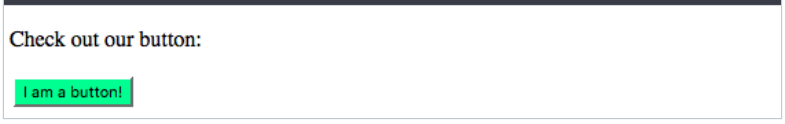
```
#myButtonId {
    /* Set the background color to a shade of green */
    background: #4FFF8F;
    /* Center-align the text */
    text-align: center;
    /* Set the cursor to be a pointer */
    cursor: pointer;
}
```

Note that we must put our CSS inside the *stye* tags in the *head* section.

- EXAMPLE

Let's look at what happens when we style the element with the *id* ***myButtonId***:

```html
<!DOCTYPE html>
<html>
    <head>
        <style>
            #myButtonId {
                /* Set the background color to a shade of green */
                background: #4FFF8F;
                /* Center-align the text */
                text-align: center;
                /* Set the cursor to be a pointer */
                cursor: pointer;
            }
        </style>
    </head>
    <body>
        <p>Check out our button:</p>
        <button id="myButtonId">I am a button!</button>
    </body>
</html>
```



Note that the button's background is *green* because of the *background* attribute we defined for the *id* ***myButtonId***.

2. Using a Class Selector

For instances where we want to apply the same styling to multiple elements, we define a CSS class using the syntax `.className` (where ***className*** is the name of our class). We then follow it with a pair of curly braces that contain the desired style constraints for all elements of that class. For example:

```
.myStyleClass {
    /* Set the background color to a shade of green */
    background: #4FFF8F;
    /* Center-align the text */
    text-align: center;
    /* Set the cursor to be a pointer */
    cursor: pointer;
}
```

Note that we must put our CSS inside the *stye* tags in the *head* section.

\- EXAMPLE

Let's look at what happens when we style any element with the *class* ***myStyleClass***:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>

        <style>
            .myStyleClass {
                /* Set the background color to a shade of green */
                background: #4FFF8F;
                /* Center-align the text */
                text-align: center;
                /* Set the cursor to be a pointer */
                cursor: pointer;
            }
        </style>
    </head>

    <body>
        <p>Check out our button:</p>
        <button id="buttonId" class="myStyleClass">I am a button!</button>
    </body>
</html>
```

Check out our button:

I am a button!

Note that the button's background is *green* because of the *background* attribute we defined for the class ***myStyleClass***.

Combining HTML and JavaScript

Now, let's look at how we can combine what we've learned about HTML and JavaScript buttons.

\- EXAMPLE

Take some time to read through the following code:

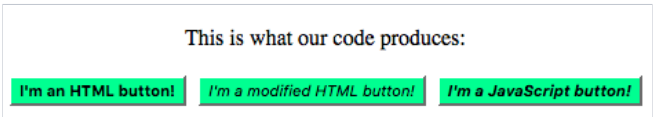
```
<!DOCTYPE html>
<html>
    <head>
        <style>
            body {
                text-align: center;
            }
            /* Styling for the 'button' class */
            .button {
                background-color: #4FFF8F;
            }
            /* Styling for id='htmlButton1' */
            #htmlButton1 {
                font-weight: bold;
            }
            /* Styling for id='htmlButton2' */
            #htmlButton2 {
                font-style: italic;
            }
            /* Styling for id='jsButton' */
            #jsButton {
                font-weight: bold;
                font-style: italic;
            }
        </style>
    </head>
    <body>
        <p>This is what our code produces:</p>
        <button id='htmlButton1' class='button'>I'm an HTML button!</b
utton>
        <button id='htmlButton2' class='button'>I'm an HTML button!</b
utton>

        <script>
            /* Create a button element */
```

```
            var clickMeButton = document.createElement('button');
            /* Set the button's text label */
            clickMeButton.innerHTML = 'I\'m a JavaScript button!';
            /* Set the button's id */
            clickMeButton.id = 'jsButton';
            /* Set the button's style class */
            clickMeButton.className = 'button';
            /* Add the button to the page */
            document.body.appendChild(clickMeButton);

            /* Get the element with id='htmlButton2' */
            var htmlButton = document.getElementById('htmlButton2');
            /* Modify the text label for htmlButton2 */
            htmlButton.innerHTML = 'I\'m a modified HTML button!';
        </script>
    </body>
</html>
```

When we render the above code, it looks like this:



Observe that:

- All three buttons have the background color styling from the *button* class, but each button has additional font styling specific to its distinct *id*.
- The initial text label for `htmlButton2` was `I'm an HTML button!`, but we used JavaScript to modify it to say `I'm a modified HTML button!` instead.

Click Events

When a user clicks a button, we call it a *click event*. Let's look at using *onclick* and *addEventListener* to prompt an action in response to a click event.

Using `onclick`

Consider the following code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>

        <style>
            .buttonClass {
                color: #4CAF50;
            }
        </style>
    </head>

    <body>
        <!-- This puts a button with the id 'button id' on our page. -->
        <button id="buttonId" class="buttonClass">I am a button!</button>

        <script>
            /* This assigns the element with id 'buttonId' to 'btn' */
            var btn = document.getElementById('buttonId');

            /* This sets the action to perform on a click event */
            btn.onclick = function() {
                /* This changes the button's label */
                btn.innerHTML = 'You clicked me!';
            };
        </script>
    </body>
</html>
```

The image below shows what the button looks like before and after it's clicked:

Using `addEventListener`

Consider the following code:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>

        <style>
            .buttonClass {
                color: #4CAF50;
            }
        </style>
    </head>

    <body>
        <!-- This puts a button with the id 'button id' on our page. -->
        <button id="buttonId" class="buttonClass">I am a button!</button>

        <script>
            /* This assigns the element with id 'buttonId' to 'btn' */
            var btn = document.getElementById("buttonId");

            /* This sets the action to perform on a click event */
            btn.addEventListener("click", function() {
                /* This changes the button's label */
                btn.innerHTML = 'You clicked me!';
            });
        </script>
    </body>
</html>
```

The image below shows what the button looks like before and after it's clicked: