**Conceptos básicos de los botones en JavaScript**

En este artículo, discutimos algunos conceptos básicos para escribir código JS (JavaScript) y CSS en un archivo HTML. Tenga en cuenta que asumimos un cierto nivel de conocimientos básicos de HTML.

## Templates

Formato básico

Usamos la siguiente plantilla para escribir código JavaScript y CSS en un archivo HTML:

<!DOCTYPE html>

<html>

<head>

<title>Page Title</title>

<style>

/\* Write CSS styles here \*/

</style>

</head>

<body>

<script>

/\* Write JS code here \*/

</script>

</body>

</html>

Escribimos nuestro código de estilo CSS entre las etiquetas <style> y </style>, y nuestro código JS entre las etiquetas <script> y </script>.

Nota: Cualquier texto entre <! - y -> se considera un comentario HTML. Estos comentarios no se mostrarán en la página web, pero podemos leerlos si vemos el código fuente de la página. Para el contenido entre etiquetas que contienen código real (es decir, estilo y secuencia de comandos), incluimos comentarios entre / \* y \* /.

Trabajar con documentos separados

En una instancia donde todo nuestro código está ubicado en archivos separados (es decir, tenemos un archivo .html con nuestro HTML, un archivo .css con nuestro CSS y un archivo .js con nuestro código JS), usamos esta plantilla para indicar nuestro archivo HTML donde encontrar los archivos JS y CSS:

<!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>

Veamos lo que hace este código:

* Al poner <link rel = "stylesheet" href = "css-file-path" type = "text / css">, donde css-file-path es la ruta del archivo .css, en la sección de cabecera (es decir, entre las etiquetas <head> y </head>), le estamos diciendo al documento que use la hoja de estilo en la ubicación a la que hace referencia el atributo href.
* Al poner <script src = "js-file-path" type = "text / javascript">, donde js-file-path es la ruta del archivo .js, en la sección del cuerpo (es decir, entre <body> y </body>), estamos diciendo que queremos ejecutar un script usando el código JS en la ubicación a la que hace referencia el atributo src.

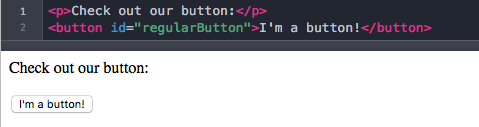
## HTML Buttons

Usamos la etiqueta <button> para crear un botón en el que se puede hacer clic que tiene los siguientes atributos opcionales:

* id: el identificador único del botón dentro de la página
* clase: la (s) clase (s) CSS utilizadas para diseñar el botón

El texto encerrado entre las etiquetas de apertura (<button>) y cierre (</button>) del botón es la etiqueta que se muestra en el botón. También podemos acceder a este texto utilizando la propiedad innerHTML del objeto de botón JS (consulte la sección de JavaScript a continuación). La sintaxis básica de un botón HTML se ve así:

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

Ahora, repasemos lo que hace:

1. document.createElement ('Botón') crea un objeto de botón en el que se puede hacer clic (createElement ('Botón')) al que hace referencia el nombre de la variable, haga clic en Me Botón

2. clickMeButton.id = 'myButton' establece la identificación del botón en myButton.

3. clickMeButton.innerHTML = 'Click Me' establece el HTML interno del botón (es decir, la etiqueta que normalmente vemos entre las etiquetas de los botones HTML) para que diga "Click Me".

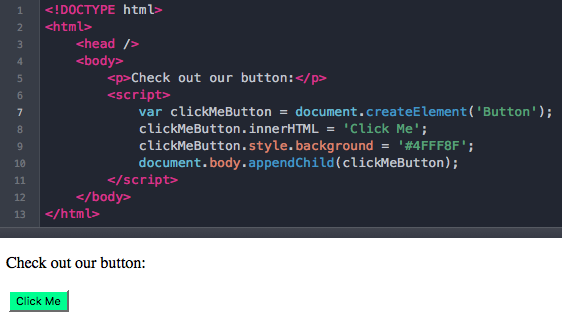
4. clickMeButton.style.background = '# 4FFF8F' establece el color de fondo del botón en verde. Para diseñar varios atributos de nuestro botón usando una clase de estilo, escribiríamos clickMeButton.className = 'myStyleClassName' en su lugar.

5. document.body.appendChild (clickMeButton) se agrega al cuerpo del documento como un niño.

Digamos que queremos modificar la etiqueta en un elemento de botón HTML con el id myButton. Simplemente usamos el método getElementById y pasamos la identificación del elemento deseado como argumento:

var clickMeButton = document.getElementById('myButton');

clickMeButton.innerHTML = 'This is my new label text!';



## 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  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 \*/

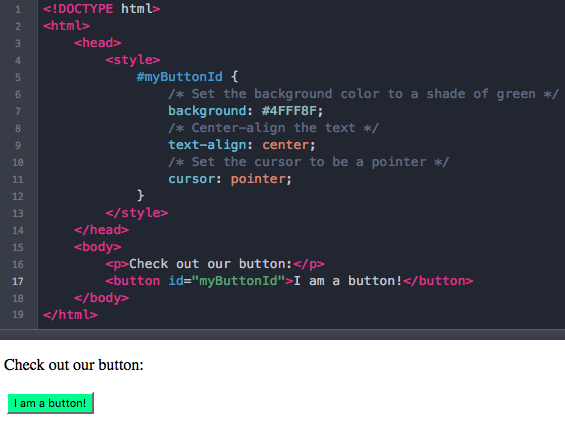
text-align: center;

/\* Set the cursor to be a pointer \*/

cursor: pointer;

}

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



### 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 \*/

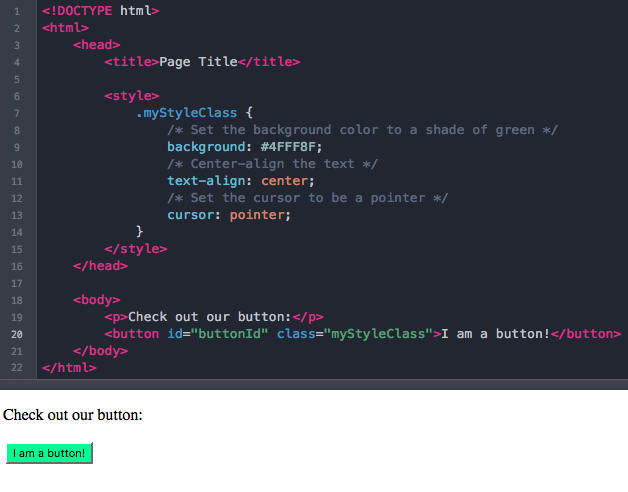
text-align: center;

/\* Set the cursor to be a pointer \*/

cursor: pointer;

}

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



## Combining HTML and JavaScript

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

<!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!</button>

<button id='htmlButton2' class='button'>I'm an HTML button!</button>

<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 \*/

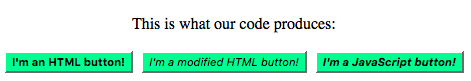
htmlButton.innerHTML = 'I\'m a modified HTML button!';

</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:

