Animating Elements in Javascript

Overview

JavaScript animations are done by programming gradual changes in an element's style.

The changes are called by a timer. When the timer interval is small, the animation looks continuous.

Think about various types of animations you can create by using this concept.

setInterval and clearInterval Functions

The `setInterval()` method calls a function or evaluates an expression at specified intervals (in milliseconds).

The `setInterval()` method will continue calling the function until `clearInterval()` is called, or the window is closed.

The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.

1000 ms = 1 second.

setInterval Function

```
id = setInterval(function, milliseconds, param1, param2, ...)
```

A Number, representing the ID value of the timer that is set. Use this value with the clearInterval() method to cancel the timer

required. The function that should execute after each interval

optional time interval in milliseconds. If omitted, 0 will be used.

An optional set of parameters that can be passed to the function as input.

clearInterval Function

```
clearInterval(id)
```

The id that was returned by setInterval. This will stop the execution cycle for this id.

You should always use `setInterval` with a corresponding `clearInterval`, otherwise the execution of the function will continue forever or until you close the window.

setTimeout Function

The `setTimeout()` method calls a function after a specified number of milliseconds.

Unlike `setInterval()`, the function is only executed once.

You can use the `clearTimeout()` method to prevent the function from running.

setTimeout Function

```
id = setTimeout(function, milliseconds, param1, param2, ...)
```

A Number, representing the ID value of the timer that is set. Use this value with the clearTimeout() method to cancel the timer

required. The function that should execute after the specified time

optional time interval in milliseconds. If omitted, 0 will be used.

An optional set of parameters that can be passed to the function as input.

clearTimeout Function

The `clearTimeout()` method clears a timer set with the `setTimeout()` method.

The ID value returned by `setTimeout()` is used as the parameter for the `clearTimeout()` method.

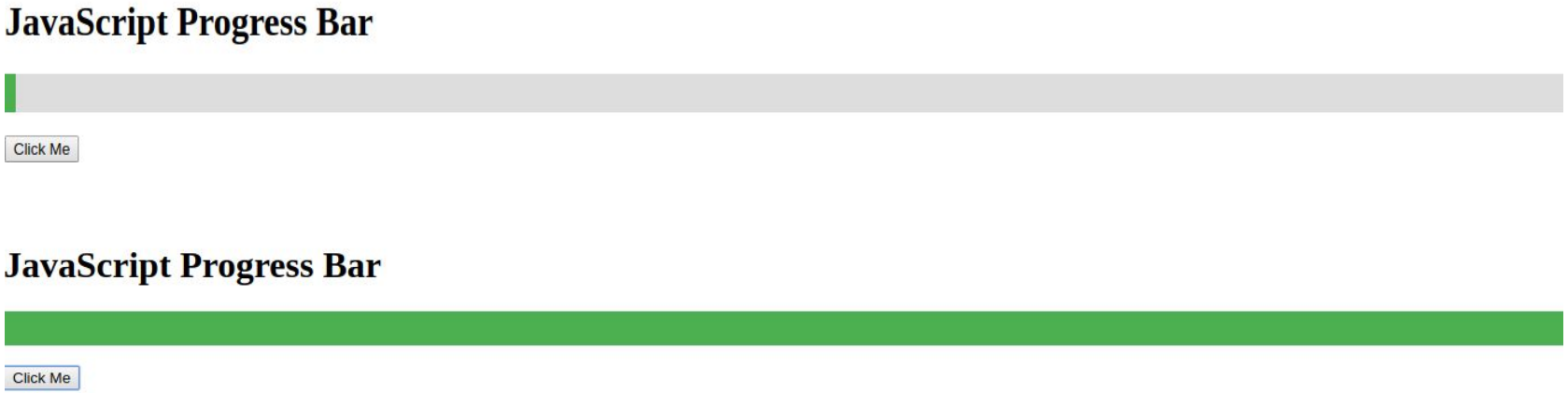
Then, if the function has not already been executed, you will be able to stop the execution by calling the `clearTimeout()` method.

```
clearTimeout(id)
```

Example: Progress Bar

In this example we are going to create a web page with a progress bar and a button.

Whenever the button is clicked, the progress bar should start filling up until it is completely filled



Progress Bar: HTML

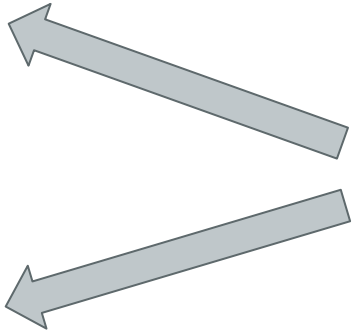
```
<!DOCTYPE html>
<html>
 <head>
  <title>Progress Bar</title>
  <link rel="stylesheet" href="css/progress.css">
  <script src="js/progress.js" defer></script>
 </head>
 <body>
  <h1>JavaScript Progress Bar</h1>
  <div id="myProgress">
   <div id="myBar"></div>
  </div>
  <br>
  <button>Click Me</button>
 </body>
</html>
```

I put two divs inside each other. The parent shows the gray background and the child will show the green progress bar.

Progress Bar: CSS

```
#myProgress {
  width: 100%;
  height: 30px;
  position: relative;
  background-color: #ddd;
}

#myBar {
  background-color: #4CAF50;
  width: 10px;
  height: 30px;
  position: absolute;
}
```



Note that the parent has a **relative** position and the child which is going to be animated has an **absolute** position. This is recommended when you want to animate objects

Progress Bar: JS

First lets access the button and attach an eventListener to its click event

```
const button = document.querySelector('button');
button.addEventListener('click', move);
```

So now we need to define the function 'move' which is called each time the button is clicked.

But, what should this function do?

It should change the width of the child element gradually

Progress Bar: JS

We can achieve this gradual change by using `setInterval()` function.

```
function move() {
   let id = setInterval(frame, 10)
}
```

In the above implementation, each time this function is called, a timer will be created which calls a function named **'frame'** every 10 milliseconds.

What should be done in the **'frame'** function?

Progress Bar: JS

```
function frame() {
  if (width == 100) {
    clearInterval(id);
  } else {
    width++;
    elem.style.width = width + '%';
  }
}
```

If the width of the green progress bar is 100%. stop and remove the timer. notice the id used in clearInterval function

otherwise, add 1% to the width of the progress bar

Progress Bar: JS

```
function frame() {
  if (width == 100) {
    clearInterval(id);
  } else {
    width++;
    elem.style.width = width + '%';
  }
}

function move() {
  id = setInterval(frame, 10);
}

var width = 0;
var elem = document.querySelector("#myBar");
var id;
const button = document.querySelector('button');
button.addEventListener('click', move);
```

This is the complete javascript code for this example. Note how we use global variables. Can you find the reason?

Example: Progress Bar

You can find the full HTML, CSS and Javascript code for this example ath the following address:

http://ww2.cs.fsu.edu/~faizian/cgs3066/sandbox/progress

Read the code and think about other things you could do to create other animations using js.

Example: Move Mario

In this example we will create a page which allows the user to move mario around in the web page and some other functionality.

Check out the example at the following address:

http://ww2.cs.fsu.edu/~faizian/cgs3066/sandbox/animate

Let's see how this works.