JavaScript Timing Events and Callback

Topics Covered:

- Timing Events
- setTimeout() Method
- setInterval() Method
- Function Sequence
- JavaScript CallBack

Topics in Detail:

Timing Events

- At specified time intervals, the **window object** allows **code execution**.
- **Timing Events** are nothing but these **time intervals**.
- The two key methods are
    - ***setTimeout(function, milliseconds)***
    - ***setInterval(function, milliseconds)***

setTimeout() method

The **function** is executed after **waiting** for certain **milliseconds**.

Syntax:

```
window.setTimeout(function, milliseconds);
```

- The **window** prefix can be **omitted.**
- The **first parameter** has the **function** to be executed**.**
- The **second parameter** has the **wait time before execution in milliseconds.**

```
<button onclick="setTimeout(myFunction, 3000)">Try it</button>

<script>
function myFunction() {
  alert('Hello');
}
</script>
```

How to stop the execution?

- To stop the function execution, use the **clearTimeout().**

```
window.clearTimeout(timeoutVariable)
```

- **The window** prefix can be **omitted.**
- The **variable returned from setTimeout()** method is used in the **clearTimeout()** method.

```
myVar = setTimeout(function, milliseconds);
clearTimeout(myVar);
```

setInterval() Method

- The **function** is executed **repeatedly** after a given **time interval**.

Syntax

```
window.setInterval(function, milliseconds);
```

- The **window** prefix can be **omitted.**
- The **first parameter** has the **function** to be executed**.**
- The **second parameter** has the **time interval** between each execution**.**

```
<button onclick="setInterval(myFunction, 1000);">Try it</button>

<script>
function myFunction() {
  alert('Hello');
}
</script>
```

How to stop the execution?

- To stop the function execution, use the **clearInterval().**

```
window.clearInterval(timerVariable)
```

- **The window** prefix can be **omitted.**
- The **variable returned from setInterval()** method is used in the **clearInterval()** method**.**

```
let myVar = setInterval(function, milliseconds);
clearInterval(myVar);
```

Function Sequence

- The functions in JavaScript are **executed in the sequence** they are **called**.

```
function myFirst() {
  myDisplayer("Hello");
}

function mySecond() {
  myDisplayer("Goodbye");
}

mySecond();
myFirst();
```

- It is better to have **control** over the **function execution**.
- To **control the sequence of function execution**, we go for JavaScript **callbacks**.

JavaScript Callbacks

- When a **function** is passed as an **argument** to **another function**, it is called a **callback**.
- **Callback functions** are used in the case of **asynchronous functions**, where one function **waits** for another function.

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```