SCHEDULED CALLBACKS

Knowledge is Different in Different States of Consciousness

Main Point Preview: Timout callbacks

The asynchronous global methods setTimeout and setInterval take a function reference as an argument and then callback the function at a specified time. Science of Consciousness: Accepting an assignment and carrying it out at a designated time is a fundamental capability required for intelligent behavior. A clear awareness and mind promotes good memory and the ability to successfully execute tasks.

Timers

- setTimeout allows to run a function once after the interval of time.
- setInterval allows to run a function regularly with the interval between the runs.

setTimeout

*

let timerId = setTimeout(func, [delay], [arg1], [arg2], ...)

- Func: Function or a string of code to execute.
- Delay: delay before run, in milliseconds (1000 ms = 1 second), by default 0.
- arg1, arg2… : Arguments for the function

```
function sayHi() {
  alert('Hello');
}
setTimeout(sayHi, 1000);
```

- With arguments:

```
function sayHi(phrase, who) {
  alert( phrase + ', ' + who );
}
setTimeout(sayHi, 1000, "Hello", "John"); // Hello, John
```

Pass a function, but don't run it

- Novice developers sometimes make a mistake by adding brackets ()

```
// wrong!
setTimeout(sayHi(), 1000);
```

- doesn't work,
  - setTimeout expects a reference to a function.
  - here sayHi() runs the function,
  - result of its execution is passed to setTimeout.
  - result of sayHi() is undefined (the function returns nothing), so nothing is scheduled

- function call versus function binding
  - sayHi() versus sayHi
  - execute the function versus reference to the function
  - fundamental concept!!

Canceling with clearTimeout

*

A call to setTimeout returns a “timer identifier” timerId that we can use to cancel the execution.

```
let timerId = setTimeout(...);
clearTimeout(timerId);
```

schedule the function and then cancel it

```
let timerId = setTimeout(() => alert("never happens"), 1000);
alert(timerId); // timer identifier
clearTimeout(timerId);
alert(timerId); // same identifier (doesn't become null after canceling)
```

setInterval

*

The setInterval method has the same syntax as setTimeout:

```
let timerId = setInterval(func, [delay], [arg1], [arg2], ...)
```

Repeatedly calls the function after the given interval of time.

To stop further calls, we should call clearInterval(timerId).

```
// repeat with the interval of 2 seconds
let timerId = setInterval(() => alert('tick'), 2000);
// after 5 seconds stop
setTimeout(() => { clearInterval(timerId); alert('stop'); }, 5000);
```

Zero delay setTimeout

*

- There’s a special use case: setTimeout(func, 0), or just setTimeout(func).

- schedules the execution of func as soon as possible.
  - after the current code is complete.

```
setTimeout(() => alert(“Hello"), 0);
alert(“World");
```

- The first line “puts the call into event queue after 0ms”
  - scheduler will only “check the queue” after the current code is complete
  - “World" is first, and " Hello " – after it.

Exercises

- Output every second
- What will setTimeout show?

Main Point: Timeout callbacks

The asynchronous global methods setTimeout and setInterval take a function reference as an argument and then callback the function at a specified time. Science of Consciousness: Accepting an assignment and carrying it out at a designated time is a fundamental capability required for intelligent behavior. A clear awareness and mind promotes good memory and the ability to successfully execute tasks.