COESIONE ITALIA 21-27 PIEMONTE
Cofinanziato dall'Unione europea
REGIONE PIEMONTE

Web Developer

Programmazione - Javascript e Typescript

Docente: Shadi Lahham

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Timing

Timeout, interval & date

Shadi Lahham - Web development

Timeout and interval

Timing

We may decide not to execute a function immediately, but at a certain time. That's called "scheduling a call". There are two methods for it:

- setTimeout allows to run a function once after an interval of time.
- setInterval allows to run a function regularly with a specified interval.

setTimeout

Syntax

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

function
Required. The function that will be executed

milliseconds
Optional. The number of milliseconds to wait before executing the code. If omitted, the value 0 is used

param1, param2, ...
Optional. Additional parameters to pass to the function. Not supported in very old browsers

setTimeout

```
// Named function
function sayHi() {
  console.log('Hello');
}

setTimeout(sayHi, 1000);

// Anonymous function
setTimeout(function () {
  console.log('Hello');
}, 3000);

// Anonymous arrow function
setTimeout(() => {
  console.log('Hello');
}, 3000);
```

```
// With arguments
function sayHi(phrase, who) {
  console.log(phrase + ', ' + who);
}

setTimeout(sayHi, 1000, 'Hello', 'John');

// Anonymous with arguments
setTimeout(
  function (value) {
    console.log(value);
  }, 1000, 'hello'
);

// Works in IE9
setTimeout(function () {
  sayHi('Hello', 'John');
}, 1000);
```

clearTimeout

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

Example

```
let timerId = setTimeout(function () {
  console.log('never happens');
}, 1000);
clearTimeout(timerId);
```

setInterval

Syntax
Method setInterval has the same syntax as setTimeout:

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

function
Required. The function that will be executed

milliseconds
Optional. The intervals, in milliseconds, on how often to execute the code. If the value is not specified, the default value can be different in different browsers

param1, param2, ...
Optional. Additional parameters to pass to the function. Not supported in very old browsers

setInterval

```
// Named function
function sayHi() {
  console.log('Hello');
}

setInterval(sayHi, 1000);

// Anonymous function
setInterval(function () {
  console.log('Hello');
}, 3000);

// Anonymous arrow function
setInterval(() => {
  console.log('Hello');
}, 3000);
```

```
// With arguments
function sayHi(phrase, who) {
  console.log(phrase + ', ' + who);
}

setInterval(sayHi, 1000, 'Hello', 'John');

// Anonymous with arguments
setInterval(
  function (value) {
    console.log(value);
  }, 1000, 'hello'
);

// Works in IE9
setInterval(function () {
  sayHi('Hello', 'John');
}, 1000);
```

clearInterval

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

Example

```
// repeat with the interval of 2 seconds
let timerId = setInterval(function () {
  console.log('tick');
}, 2000);

// after 5 seconds stop
setTimeout(function () {
  clearInterval(timerId);
  console.log('stop');
}, 5000);
```

JavaScript is single threaded

```
function logDelayedMessages() {
  setTimeout(() => console.log('1 second'), 1000);
  setTimeout(() => console.log('2 seconds'), 2000);

  while (true) {} // infinite loop
}

logDelayedMessages();
```

Date and time

The Date object

A date consists of a year, a month, a day, an hour, a minute, a second, and milliseconds

Syntax

```
new Date()
new Date(milliseconds)
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

The Date object

```
// creating a new date

let d = new Date();
let d = new Date("October 13, 2024 11:13:00"); // implementation-dependent; browser/version/os
let d = new Date(86400000); // Fri Jan 02 1970
let d = new Date(2024, 3, 18, 11, 33, 30, 0);
```

The Date object

```
// which dates are these?
let d = new Date(0);
let d = new Date(-86400000);

// first parameter 'year' - values from 0 to 99 map to the years 1900 to 1999
let d = new Date(99, 5, 24, 11, 33, 30, 0); // Jun 24 1999
let d = new Date(24, 5, 24, 11, 33, 30, 0); // Jun 24 1924
let d = new Date(2024, 5, 24, 11, 33, 30, 0); // Jun 24 2024
```

Displaying Dates

```
// Javascript tries to automatically convert dates to strings if they are assigned as Strings
// the first two outputs are the same

let d = new Date();
console.log(d);
console.log(d.toString()); // implementation-dependent; browser/version/os
console.log(d.toUTCString());
console.log(d.toDateString());
```

Date formats

There are generally 4 types of JavaScript date input formats

```
ISO Date    "2024-03-25" (The International Standard)
Short Date  "03/25/2024"
Long Date   "Mar 25 2024" or "25 Mar 2024"
Full Date   "Wednesday March 25 2024"
```

Date get methods

```
// get date
let d = new Date();
console.log(d);
console.log(d.getMonth());
console.log(d.getMinutes());
console.log(d.getFullYear());

new Date(); // creates a Date object representing current date/time
new Date().valueOf(); // returns number of milliseconds since midnight 01 January, 1970 UTC
new Date().getTime(); // same as above
Date.now(); // same as above
```

Complete list

https://www.w3schools.com/js/js_date_methods.asp

Date methods | mdn

Date set methods

```
// set date
d.setFullYear(2024, 0, 14); // Jan 14 2024
d.setDate(d.getDate() + 50); // Mar 05 2024 setDate() changes the day of the month

// parse valid dates
let msec = Date.parse('March 21, 2024');
let date = new Date(msec);
```

Complete list

https://www.w3schools.com/js/js_date_methods.asp

Date methods | mdn

Comparing dates

Comparison operators work also on dates

```
let date1 = new Date();
let date2 = new Date("Feb 24, 2024 15:50:00");
if (date1 > date2) {
    console.log('break time');
} else {
    console.log('stay in class');
}
```

Your turn

1.Slow list

- Create an array that holds a list of 30 items (food, books, etc.)
- Print one item of the list every second until the list is completely printed
  - Use setInterval to achieve this goal
  - Do the same thing using setTimeout

2.Oh no you don’t

- Write a function “useful” that does something useful in Javascript
- Schedule it to run after 10 seconds
- Write another function that cancels the scheduling of the first function
- Use the second function to cancel the first one after 5 seconds and output ‘function cancelled’ to the console

3.Weekday

- Write a function getWeekDay(date) to show the weekday in short format: ‘MO’, ‘TU’, ‘WE’, ‘TH’, ‘FR’, ‘SA’, ‘SU’
- Write another function that does the same in Italian
- Add a language parameter to the function that accepts ‘en’ or ‘it’ and outputs the day in the correct language

4.Timed calculator

- We will modify ‘Calculator’ exercise from the lesson about functions
- Rewrite the last function that performs all 4 operations so that there is a delay of 3 seconds between one operation and the next

5.My setInterval

- Pretend that setInterval() doesn't exist
- Re-create it using setTimeout naming your function mySetInterval
- Test your new function
- Modify your function so that it automatically stops after 15 intervals

Bonus

6.Date ago

- Create a function getDateAgo(date, days) that returns the day of the month n days ago from the given date
- For instance, if today is the 20th, then getDateAgo(new Date(), 1) should be 19th and getDateAgo(new Date(), 2) should be 18th
- Test the function to make sure it works reliably with any valid Date object
- Decide what to do with a negative 'days' parameter
    - e.g. getDateAgo(new Date(), -2)

7.Seconds

Write two functions that based on the current date and time output the number of seconds:

- getSecondsToday() returns the number of seconds from the beginning of today
- getSecondsToTomorrow() returns the number of seconds till tomorrow

8.Timed conversion

- We will modify ‘Temperature conversion’ exercise from the lesson about functions
- Call celsiusToFahrenheit on temperatures from 0 to 100 so that one temperature is printed to the console every second
    - Use setInterval to achieve this goal.
    - Do the same thing using setTimeout.

9. Format date

Write a function formatDate(date) that accepts a date and outputs it as follows:

- If less than a second has passed since the date, output "right now"
- If less than a minute has passed since the date, output "n sec. ago"
- If less than an hour has passed since the date, output "m min. ago"
- Otherwise, output the date in this format "DD.MM.YY HH:mm"
    - e.g. 17.04.16 10:00

10.Clock

- Implement a javascript clock that prints the current time to the console every second
- The output should be in the format HH:mm:ss e.g. 17:03:06

References

JavaScript Timing Events

setTimeout() | MDN

setInterval() | MDN

References

JavaScript Date Objects
Date() constructor | MDN
JavaScript Date Reference
JavaScript Date Formats
JavaScript Date Methods
Coordinated Universal Time
Greenwich Mean Time