WEEK 6: Javascript cont.

Javascript is a general purpose language but is designed to run in a browser and so input and output needs special care.

Output of a JavaScript can be

- a visible part of an HTML element, using innerHTML
- directly written to HTML document (document.write())
- a popup window (window.alert())
- browser console for debugging purposes, console.log()
- print HTML document (to a printer) (window.print())

Most common way is using innerHTML (as seen in previous examples). Note that innerHTML is a property and not a function.

```
<!DOCTYPE html>
<html>
<body>
<span id="demo"></span>

<button type="button"
onclick="document.getElementById('demo').innerHTML = 'He'+'llo World' ">
Say Hello World</button>

</body>
</html>
```

Using document.write() needs special care, as after an HTML document is loaded, document.write() will delete all existing document:

```
<!DOCTYPE html>
<html>
<body>

<h1>Original page</h1>

<script>
document.write('<p>Calling document.write() after the document has finished
loading will overwrite the whole document.</p>');
</script>
```

```
<button onClick="document.write('something new');">Overwrite
document</button>

</body>
</html>
```

Function window.alert() opens a small alert window. The window object is the default global scope object in JavaScript that means “window.” can be missed.

```
<!DOCTYPE html>
<html>
<body>

<script>
window.alert('Hello World');
alert('Hello World again');
</script>

</body>
</html>
```

Many JavaScript expressions are used repeatedly in the same code. JavaScript programs tend to be more and more unreadable, we can make some functions to access objects in a shorter way.

```
<script>
function $(id)
{
      return document.getElementById(id); // save 22 chars...
}
</script>
```

Note: JavaScript indentifiers begin with

- a letter (a-z or A-Z)
- a dollar sign ($)
- an underscore sign (_)

Note: the $ function above gave the idea of creating sophisticated JavaScript function librarys such az JQuery.

Data types

JavaScript variables hold quite few data types:

- number

```
<script>
x = 1;
x = 1.1;
x = 6e23;
</script>
```

- string

```
<script>
s = "apple";
s = 'pie';
</script>
```

- boolean

```
<script>
function decide()
{
return Boolean(3 > 1); // returns true
}
</script>
```

Note: everything with a value is true. 0 (zero), empty string, null, NaN, or undefined value is false.

- array

```
<script>
var dow =
["Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"];
document.getElementById("result").innerHTML = "First day of a week is " +
dow[1];
</script>
```

- object

```
<script>
var person = {sex: "male", yearofbirth: 1999, firstName:"David",
lastName:"Colibri", eyeColor:"blue"};
</script>
```

Array functions (and function-like operations)

toString: creates a string with comma separated values of the array.

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

output:
```
Alpha,Beta,Gamma,Delta
```

Note: JavaScript calls toString() automatically where only string value is allowed:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
document.getElementById("demo").innerHTML = alphabet;
</script>
```

output:
```
Alpha,Beta,Gamma,Delta
```

Note: Even this automatic conversion is very convenient, for better understanding we will use toString() method in our examples.

join: creates a string where elements are joined with the given character or string

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
document.getElementById("demo").innerHTML = alphabet.join(' * ');
</script>
```

output:
```
Alpha * Beta * Gamma * Delta
```

`pop: removes last element from the array`

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
alphabet.pop();
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

output:
```
Alpha,Beta,Gamma
```

push: add new element to the array (to the end)

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
alphabet.push('Epsilon');
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

output:
```
Alpha,Beta,Gamma,Delta,Epsilon
```

splice: add new elements to the array (and remove some, if needed)

- first parameter: index of where new elements should be added
- second parameter: number of elements in the array to be deleted before adding the new ones
- the rest of the parameters: the elements to be added

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
alphabet.splice(2,1,'x','y'); // delete 1 element from index 2, then add
the 2 new elements
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

output:
```
Alpha,Beta,x,y,Delta
```

splice can be parametrized to remove elements from the array:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
alphabet.splice(1,2); // remove 2 elements from index 1
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

output:
```
Alpha,Delta
```

shift: removes first element

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
r=alphabet.shift();
document.getElementById("demo").innerHTML = alphabet.toString() + "<br>
removed: " + r;
</script>
```

```
output:
Beta,Gamma,Delta
removed: Alpha
```

Note: “shift” means that all the remaining indices in the array are shifted.

delete: deletes the value of the array. Element remains in place with undefined value.

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
delete alphabet[1];
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

```
output:
Alpha,,Gamma,Delta
```

unshift: adds new element to the array to its beginning (index 0)

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
alphabet.unshift('New');
document.getElementById("demo").innerHTML = alphabet.toString();
</script>
```

```
output:
New,Alpha,Beta,Gamma,Delta
```

length: Number of elements of the array.

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
document.getElementById("demo").innerHTML = alphabet.length;
</script>
```

```
output:
4
```

Note: length is not a function, it is used as a property of the array! There are no brackets().

concat: merge two arrays:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
var alphabet2 = ["Epsilon", "Zeta", "Eta"];
a = alphabet.concat(alphabet2); // concat returns a new array!
document.getElementById("demo").innerHTML = a.toString();
</script>
```

output:
```
Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta
```

concat can merge (concatenate) many arrays:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
var alphabet2 = ["Epsilon", "Zeta", "Eta"];
var alphabet3 = ["Theta", "Iota"];
a = alphabet.concat(alphabet2,alphabet3);
document.getElementById("demo").innerHTML = a.toString();
</script>
```

output:
```
Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota
```

Note instead of an array, concat can merge arrays with values:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta"];
var alphabet2 = ["Epsilon", "Zeta", "Eta"];
var alphabet3 = ["Theta", "Iota"];
a = alphabet.concat(alphabet2,alphabet3,"Kappa");
document.getElementById("demo").innerHTML = a.toString();
</script>
```

output:
```
Alpha,Beta,Gamma,Delta,Epsilon,Zeta,Eta,Theta,Iota,Kappa
```

sort: Array Sorting (alphabetically):

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];
document.getElementById("demo").innerHTML = alphabet.sort().toString();
</script>
```

output:
```
Alpha,Beta,Delta,Epsilon,Eta,Gamma,Iota,Theta,Zeta
```

reverse: reverses the elements in the array (by index)

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];
document.getElementById("demo").innerHTML = alphabet.reverse().toString();
</script>
```

output:
```
Iota,Theta,Eta,Zeta,Epsilon,Delta,Gamma,Beta,Alpha
```

Note: reverse does not sort in alphabetically reverse order!

sorting to alphabetically descending order:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];
document.getElementById("demo").innerHTML =
alphabet.sort().reverse().toString();
</script>
```

output:

```
Zeta,Theta,Iota,Gamma,Eta,Epsilon,Delta,Beta,Alpha
```

Iteration over arrays

for-in statement loops over an array:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];
for (i in alphabet)
 document.write("<li>" + i + ". element: " + alphabet[i]);
</script>
```

output:

```
 ● 0. element: Alpha
 ● 1. element: Beta
 ● 2. element: Gamma
 ● 3. element: Delta
 ● 4. element: Epsilon
 ● 5. element: Zeta
 ● 6. element: Eta
 ● 7. element: Theta
 ● 8. element: Iota
```

foreach uses a function for doing some action on each element:

```
<span id="demo"></span>
<script>

function f(value,index,array)
 {
 document.write("<li>" + index + ". element: " + array[index] + " (" +
value + ")");
 }

var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];

alphabet.forEach(f)
 </script>
```

output:

```
• 0. element: Alpha (Alpha)
• 1. element: Beta (Beta)
• 2. element: Gamma (Gamma)
• 3. element: Delta (Delta)
• 4. element: Epsilon (Epsilon)
• 5. element: Zeta (Zeta)
• 6. element: Eta (Eta)
• 7. element: Theta (Theta)
• 8. element: Iota (Iota)
```

traditional C-like for:

```
<span id="demo"></span>
<script>
var alphabet = ["Alpha", "Beta", "Gamma", "Delta","Epsilon", "Zeta",
"Eta","Theta", "Iota"];
//document.getElementById("demo").innerHTML =
alphabet.sort().reverse().toString();

for (i=0; i<alphabet.length;i++)
 {
 document.write("<li>" + i + ". element: " + alphabet[i]);
 }
```

```
</script>
```

output:

```
● 0. element: Alpha
● 1. element: Beta
● 2. element: Gamma
● 3. element: Delta
● 4. element: Epsilon
● 5. element: Zeta
● 6. element: Eta
● 7. element: Theta
● 8. element: Iota
```