Web technology – JavaScript

functions

dotNET academy

Arrays

Array syntax:

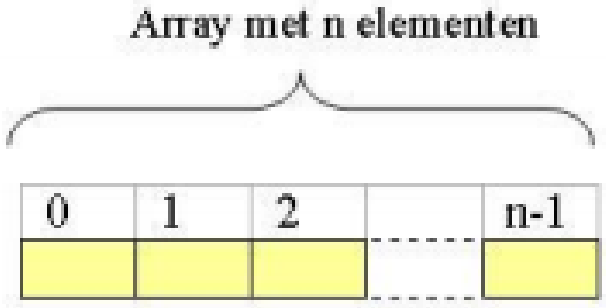
```
let myArray1 = new Array()
let myArray2 = []
```

=> Use the 2nd spelling!

You can always assign only one value at a time to a variable. An array, on the other hand, can contain different values. One speaks of different elements, in which each element behaves as variable. So instead of 10 variables, you can declare an array of 10 elements.

Arrays

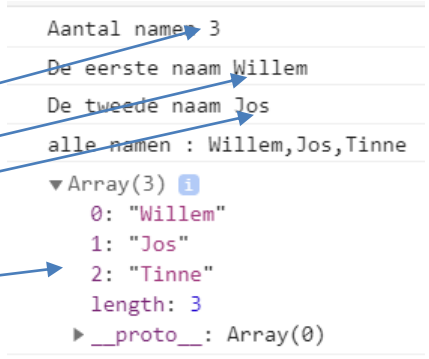
Array have another big advantage: the different elements are numbered continuously. In a loop you can then go through the array via an automatic counter



Arrays : an example

```
const namen = [];
namen[0]    = "Willem";
namen[1]    = "Jos";
namen[2]    = "Tinne";

console.log("Aantal namen " + namen.length);
console.log("De eerste naam " + namen[0]);
console.log("De tweede naam " + namen[1]);
console.log("alle namen : " + namen);
console.log(namen);
```



Array : an example

The array is known by the name "namen"

You can fill the array by assigning a value to each element of the array. The first element is names[0] and is assigned the value 'Willem'.

Elements can therefore be addressed by having the name of the array followed by an index in square brackets [ ].

This index is zero-based: the first digit is a zero

Example: loop through an array

```
const namen = [];
namen[0]    = "Willem";
namen[1]    = "Jos";
namen[2]    = "Tinne";

console.log("Aantal namen " + namen.length);
console.log("De eerste naam " + namen[0]);
console.log("De tweede naam " + namen[1]);
console.log("alle namen : " + namen);
console.log(namen);

for(let i=0; i<namen.length;i++){
    console.log(namen[i]);
}
```

Array loop – new method

```
for(let naam of namen){
    console.log(naam);
}
```

Array methods

The length of an array can be found by: "length"

Arrays have a number of methods by which the individual elements can be manipulated

Reverse allows us to flip an array so that the last element becomes the first and vice versa

```
let x  =["a","b","c"];
console.log(x);
console.log(x.reverse());
```

Array methods

The sort method allows us to sort the elements of the array alphabetically:

```
const dieren = ["mier","aap","leeuw"];
console.log(dieren.sort());
```

some methods

Methode	Beschrijving
concat	Voeg de elementen van twee (of meer) Arrays samen
join	Plaatst alle elementen van een Array in een String. De elementen worden gescheiden door een aangegeven delimeter (standaard een komma).
pop	Verwijdert en retourneert het laatste element van de Array
push	Voegt een of meerdere elementen toe aan een Array en retourneert de nieuwe lengte van de Array
reverse	Keert de volgorde van de Arrayelementen om
shift	Verwijdert en retourneert het eerste element van de Array
slice	Maakt een nieuwe Array van een aangegeven bereik in een Array. Argumenten: beginpositie en optioneel de eindpositie
sort	Sorteert de elementen in de Array
splice	Elementen verwijderen en toevoegen aan een Array. Deze methode retourneert de verwijderde elementen. Argumenten: startpositie voor verwijderen, aantal te verwijderen elementen, nieuwe elementen
unshift	Voegt een of meerdere elementen toe aan het begin van de Array en retourneert de nieuwe lengte

Some methods

Methode	Beschrijving
concat	Voeg de elementen van twee (of meer) Arrays samen
join	Plaatst alle elementen van een Array in een String. De elementen worden gescheiden door een aangegeven delimeter (standaard een komma).
pop	Verwijdert en retourneert het laatste element van de Array
push	Voegt een of meerdere elementen toe aan een Array en retourneert de nieuwe lengte van de Array
reverse	Keert de volgorde van de Arrayelementen om
shift	Verwijdert en retourneert het eerste element van de Array
slice	Maakt een nieuwe Array van een aangegeven bereik in een Array. Argumenten: beginpositie en optioneel de eindpositie
sort	Sorteert de elementen in de Array
splice	Elementen verwijderen en toevoegen aan een Array. Deze methode retourneert de verwijderde elementen. Argumenten: startpositie voor verwijderen, aantal te verwijderen elementen, nieuwe elementen
unshift	Voegt een of meerdere elementen toe aan het begin van de Array en retourneert de nieuwe lengte

```
const a = [4,3,2,1];
const b = [8,5,3,5];

const c = a.concat(b)
console.log(c);
```

Is hetzelfde (spread operator)

```
const d = [...a,...b]//spread operator
console.log(d)
```

Removing Items

Delete a single element by pop() or shi

pop() : remove the last element

```
const numbers = [1,2,3,4,5,6,7,8,9];
numbers.pop();
console.log(numbers);
```

```
▼Array(8) i
  0: 1
  1: 2
  2: 3
  3: 4
  4: 5
  5: 6
  6: 7
  7: 8
  length: 8
```

shift() : removes the first element

```
const numbers = [1,2,3,4,5,6,7,8,9];
numbers.shift();
console.log(numbers);
```

```
▼Array(8) i
  0: 2
  1: 3
  2: 4
  3: 5
  4: 6
  5: 7
  6: 8
  7: 9
  length: 8
```

Removing Items

Delete a single element at random positions by splice()

```
const numbers = [1,2,3,4,5,6,7,8,9];
numbers.splice(2,3);
console.log(numbers);
```

```
▼Array(6) i
    0: 1
    1: 2
    2: 6
    3: 7
    4: 8
    5: 9
    length: 6
```

Shift & Unshift

```
let todoList = [];
function remember(task) {
    todoList.push(task);
}
function getTask() {
    return todoList.shift();
}
function rememberUrgently(task) {
    todoList.unshift(task);
}

remember("winkelen");
remember("studeren");
remember("joggen");
remember("naar bioscoop");
remember("meeting");

console.log(todoList);

getTask();

console.log(todoList);

rememberUrgently("boek kopen");

console.log(todoList);
```

Add and remove elements from an array.

Push: Add a task at the end of an array

Shift: Remove the first element

UnShift: Add at the beginning of the array

```
(5) ["winkelen", "studeren", "joggen", "naar bioscoop", "meeting"]
(4) ["studeren", "joggen", "naar bioscoop", "meeting"]
(5) ["boek kopen", "studeren", "joggen", "naar bioscoop", "meeting"]
```

Searching in an array

indexOf()
returns the index from the first found or -1 at not found *: a.indexOf()*

lastIndexOf()
returns the index from last found or -1 when not found *: a.lastIndexOf()*

find()
returns the first item that returns true or undefined *:*
a.find((element,index,array)=>{//return true or false});

findIndex()
returns the index of the first item that returns true or undefined *:*
a.findIndex((element,index,array)=>{//return true or false});

Includes() => meest recente (2016)
returns a boolean if the element is in the array or false

Iterative methods on arrays

- `every()` — Runs the given function on every item in the array and returns `true` if the function returns `true` for every item.
- `filter()` — Runs the given function on every item in the array and returns an array of all items for which the function returns `true`.
- `forEach()` — Runs the given function on every item in the array. This method has no return value.
- `map()` — Runs the given function on every item in the array and returns the result of each function call in an array.
- `some()` — Runs the given function on every item in the array and returns `true` if the function returns `true` for any one item.

Reduction methods

Reduce en ReduceRight : Both methods repeat all items in the array and build a value that is returned. The reduce () method does this from the first item and towards the last, while reduceRight () starts on the last and goes to the first. Both methods accept two arguments: a function to call each item and an optional initial value on which the reduction is based.

The function passed for reduce() or reduceRight() applies accepts four arguments: the previous value, current value, index, and array object.