PXL DIGITAL

Web Expert

Objects & functions

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Has
www.pxl.be - www.pxl.be/facebook

Destructuring

Object literal uitpakken in 1 of meerdere variabelen

```
let person = { id:1, name:'tim' };
let id;
{ id } = person;

let person = { id:1, name:'tim' };
let id, name;
{ id, name } = person;


let person = { id:1, name:'tim' };
let { id, name } = person;
```

Shorthand notation

Waarde van een bestaande variabele gebruiken als veld in nieuw object

```
let id = 1;
let person = { id, name:'tim' };
// ipv let person = { id:id, name: 'tim' };


let id = 1;
let name = 'tim';
let person = {id, name};
// ipv let person = {id: id, name: name};
```

Functions (map, filter, reduce)

```
let persons = [
  { id: 1, name: 'pauline', age: 15 },
  { id: 2, name: 'tim', age: 9 },
  { id: 3, name: 'sofie', age: 7 }
];

let ages = persons.map( function(person) {return person.age;} );
console.log( ages );      // [15, 9, 7]
```

map: pas function toe op elke waarde in een array om een nieuwe array te maken

Arrow notation function
(map, filter, reduce)

```
let persons = [
  { id: 1, name: 'pauline', age: 15 },
  { id: 2, name: 'tim', age: 9 },
  { id: 3, name: 'sofie', age: 7 }
];

let sumAges = persons.reduce(
           (sum, person) => { return sum + person.age;} ,
            0);
console.log( sumAges );
```

reduce: array wordt herleid tot één waarde adhv function

Arrow notation function
(map filter reduce)

```
let persons = [
  { id: 1, name: 'sofie', age: 15 },
  { id: 2, name: 'tim', age: 9 },
  { id: 3, name: 'sofie', age:7 }
];

let youngPersons = persons.filter( (person) => person.age < 12);
console.log( youngPersons );
```

filter: nieuwe array met enkel die waarden die voldoen aan de voorwaarde

Arrow notation function
(map filter reduce)

```
let persons = [
  { id: 1, name: 'pauline', age: 15 },
  { id: 2, name: 'tim', age: 9 },
  { id: 3, name: 'sofie', age:7 }
];

let sumAgesOfYoungPersons = persons
    .filter( (person) => person.age < 12 )
    .reduce( (sum, person) => { return sum + person.age; } , 0);

console.log( sumAgesOfYoungPersons );
```

Arrow notation function
(forEach)

```
let person={
    name:'tim',
    hobbies:['reading', 'running'],
    print(){
        console.log(this.name);
        this.hobbies.forEach( (hobby) => {
            console.log(`${this.name} has hobby ${hobby}`);
        });
    }
}


person.print()
```

```
tim
tim has hobby reading
tim has hobby running
```

In arrow functions this retains the value of the enclosing lexical context's this
dus hier this komt overeen met person-object

Arrow notation function
(forEach)

```
let person={
    name:'tim',
    hobbies:['reading', 'running'],
    print(){
        console.log(this.name);
        this.hobbies.forEach( function( hobby ) {
            console.log(`${this.name} has hobby ${hobby}`);
        });
    }
}


person.print()
```

```
tim
undefined has hobby reading
undefined has hobby running
```

In een function:
this = object als de function deel uitmaakt van object
global anders
this maakt hier deel uit van de anonieme functie dus **global**

Arrow notation function
(find)

```
let persons = [ {id:1, name:"tim"},
                {id:2, name:"toon"},
                {id:3, name:"sofie"}  ];
let person = persons.find((x)=>x.name.charAt(0)=='t');
console.log(person);
let person2 = persons.find((x)=>x.name.charAt(0)=='q');
console.log(person2);
```

```
{ id: 1, name: 'tim' }
undefined
```