Webscripting

Hoofdstuk 4

Datastructures: Objects & arrays

DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Arrays

```
let listOfNumbers = [2, 3, 5, 7, 11];
let emptyArray = [];
```

indexed properties: `array[ index ]`

```
console.log(listOfNumbers[0]); // 2
listOfNumbers[1] = 3;
```

named properties: `array.name array["name"]`

```
console.log(listOfNumbers.length);  // 5
console.log(listOfNumbers[ "length" ] );  // 5
```

Array loops

```
let numbers = [1,2,3,4];
for ( let i = 0; i < numbers.length; i++){
   console.log(numbers[i]);
}


for ( let number of numbers ){
   console.log(number);
}
```

Array methods

indexOf, lastIndexOf

```
console.log([1, 2, 3, 2, 1].indexOf(2));  // 1
console.log([1, 2, 3, 2, 1].indexOf(5));  // -1
console.log([1, 2, 3, 2, 1].lastIndexOf(2)); // 3
```

pop: verwijdert het laatste element

```
let numbers = [1,2,3];
let last = numbers.pop();  // numbers = [1,2]
                           // last = 3
```

push: plaatst een element acheraan bij

```
let numbers = [1,2,3];
numbers.push(4);           // numbers = [1,2,3,4]
// ook:
numbers[numbers.length] = 5;
// // numbers = [1,2,3,4,5]
```

Array methods

shift verwijdert een element vooraan de array

```
let numbers = [1,2,3];
let first = numbers.shift();  // numbers = [2,3]
                              // first = 1
```

unshift: plaatst een element vooraan bij

```
let numbers = [1,2,3];
numbers.unshift(0); // numbers=[0,1,2,3]
```

Array methods

slice: copy van de originele array van index1 tot index2

```
let names = ["jan","tim","sofie","geert","nele"];
console.log(names.slice(1,3));// index 1 tot 3
      // ["tim", "sofie"]
console.log(names.slice(1));      // index 1 tot eind
      // ["tim","sofie","geert","nele"]
```

concat: samenvoegen van 2 arrays of array en element

```
let names = ["jan","tim"];
names = names.concat("sofie");
names = names.concat(["geert", "nele"]);
```

Array: spread / rest (...)

```
// rest-operator
// parameters bij aanroepen v. sum worden in de
// array numbers geplaatst
function sum( ...numbers ) {
    let sum=0;
    for (number of numbers){
        sum += number;
    }
    return sum;
}
console.log(sum(1,2,3));


let a = [1,2,3,4];
// spread-operator
// de array a wordt uitgepakt als parameters
console.log(sum(...a));
```

Array: destructuring

Waarde van array uitpakken in variabelen

```
let a, b ;
let waarden = [1,2];
[a,b] = waarden;  // a=1, b=2
[a,b] = [b,a];  // a=2, b=1
console.log(a, b);

function printRij1 ([ rij1, ...rijen ]){
   console.log(rij1);
}

let matrix=[[1,2],[3,4],[5,6]];
printRij1(matrix);  // [1,2]
```

Objects

Verzameling van eigenschappen (properties)
   property-name correspondeert met value
   of met function (later)

```
let person  = { name:"tim", age:22 };
console.log(person.name); // tim
console.log(person["name"]); // tim
person.age=23;
person.address="unknown";
console.log(person);
//{ name: 'tim', age: 23, address: 'unknown' }
```

Objects: destructuring

Object uitpakken in variabelen

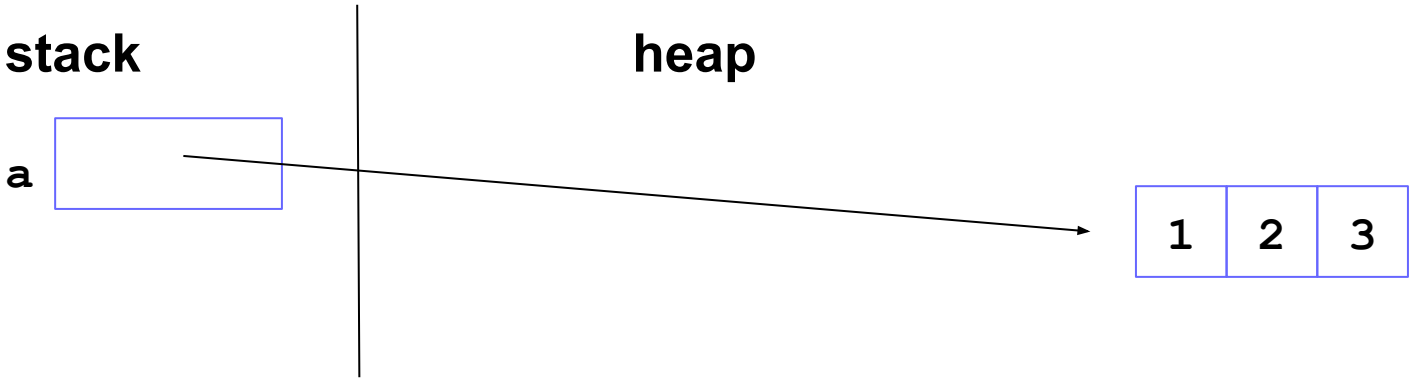
```
let person = {name:"tim",age:12};
let {age} = person;
console.log(age);   // 12
```

```
let person = {name:"tim",age:12};
let {name, age} = person;
console.log(name);  // tim
console.log(age);   // 12
```

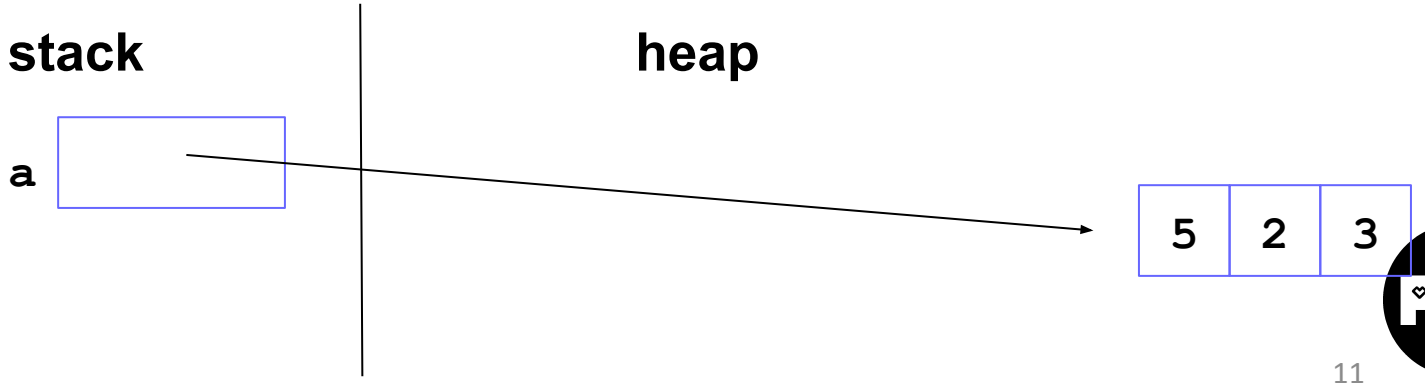
mutable vs immutable

Reference: mutable

```
let a = [1,2,3];
```



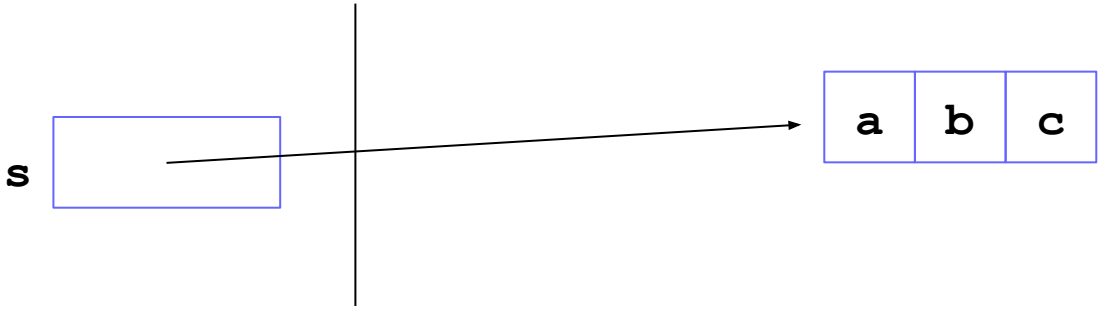

```
a[0] = 5;
```



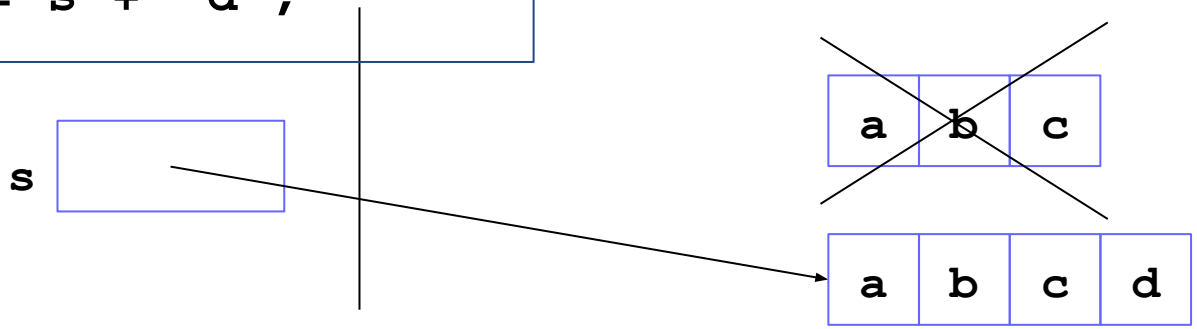
mutable vs immutable

Primitive: immutable

```
let s = "abc";
```




```
s = s + "d";
```



String functions

Primitive datatype string heeft strikt genomen geen functions!

Wanneer we een methode oproepen op een string dan wordt de omzetting naar een String-object gemaakt.

toUpperCase / toLowerCase

```
let s = "abc";
s = s.toUpperCase();
console.log( s );  // ABC
```

indexOf / lastIndexOf

```
let s = "abc";
let index = s.indexOf( "b" );
console.log( index ); // 1
```

String functions

slice: maak een substring

```
let s = "abcdef";
console.log( s.slice( 2,4 ) ); // cd
console.log( s.slice( 2 ) );   // cdef
```

trim: verwijder spaties vooraan & achteraan

```
let s = "   abc   ";
console.log( s.trim( ) ); // abc
```

split: splits een string en een maak een array

```
let s = "dit is een zin";
console.log( s.split( " " ) );
//[ 'dit', 'is', 'een', 'zin' ]
```

join: voeg een array samen in een string

```
let a = [ 1, 2, 3 ];
console.log( a.join( "," ) ); // 1,2,3
```

repeat

```
let s = "ha"
console.log( s.repeat( 5 ) ); // hahahahaha
```

Math

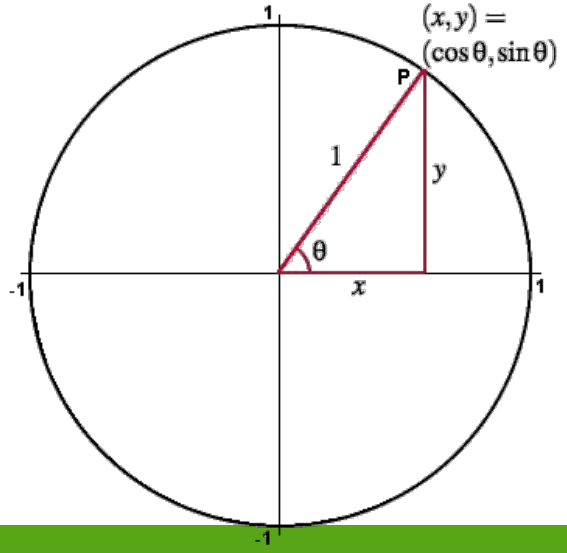
```
console.log( Math.PI ); // 3.141592653589793
console.log( Math.E );  // 2.718281828459045

console.log( Math.max( 1, 2, 3 ) ); // 3
console.log( Math.max( ...[1, 2, 3] ) ); // 3
console.log( Math.min( 1, 2, 3 ) ); // 1
console.log( Math.min( ...[1, 2, 3] ) ); // 1

Math.cos( angle ), Math.sin …
```

Math

```
// Math.random()   [0, 1[
function randomPointOnCircle(radius) {
    let angle = Math.random() * 2 * Math.PI;
    return { x: radius * Math.cos(angle),
             y: radius * Math.sin(angle) };
}
console.log(randomPointOnCircle(2));
// {x: 0.3667, y: 1.966}
```



JSON

JavaScript Object Notation is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse (ontleden) and generate

Lijkt heel sterk op object:

```
{
    "name": "tim",
    "hobbies": ["reading", "running", "tennis"]
}
```

JSON

JSON.stringify: vorm een object om naar een JSON-string
JSON.parse: ontleed een JSON-string, maak er een object van

```
let person = {
    name: "tim",
    hobbies: ["reading", "running", "tennis"]
};
let personJSON = JSON.stringify( person );
console.log( personJSON );
let person2 = JSON.parse( personJSON);
console.log(person2);
```

```
{"name":"tim","hobbies":["reading","running","tennis"]}
{ name: 'tim', hobbies: [ 'reading', 'running', 'tennis' ] }
```

Besluit

Reference types : object, array
Mutable
object: named properties

```
let person={name:"tim", age:12};
console.log(person.age);
console.log(person["age"]);
```

array: indexed properties

```
let a = [1,2,3];
console.log( a[ 0 ] );
```

named properties

```
console.log( a.length );
console.log( a["length"] );
```

JSON