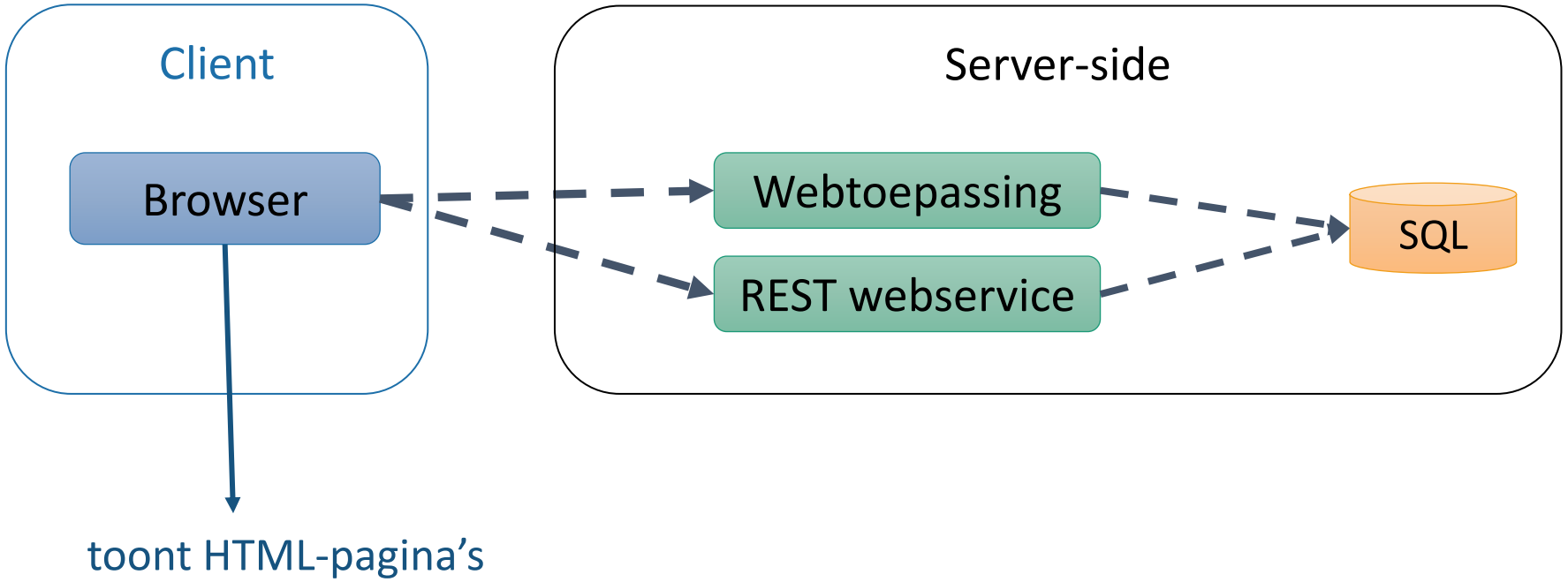
Javascript

Veerle Ongenae

Inhoud


Client
Browser
toont HTML-pagina's
Server-side
Webtoepassing
REST webservice
SQL

Doel

UNIVERSITEIT GENT

- Basisconcepten programmeertaal javascript kennen en kunnen gebruiken

JavaScript

UNIVERSITEIT GENT

- Scripttaal (ECMAScript), huidige versie 12 (2021)
- Ontworpen om HTML-pagina's interactiever en dynamischer te maken
- Vaak ingebed in HTML-pagina's
- Gebruik maken van de API's in de browser
- Lage drempel
  - Oorspronkelijk geen kennis van OOP
- Syntax lijkt op Java, C#, C++, …
- Veel verschillende versies

De programmeertaal JavaScript

UNIVERSITEIT GENT

- Variabelen en types

Variabelen

UNIVERSITEIT GENT

- Geen sterke typering
  - Types worden dynamisch (at runtime) toegekend en eventueel veranderd
- Declaratie
  - let
    - Block scope
  - const
    - Constante waarde (~final)
  - var
    - Globale scope of functie scope

```
if (true) {
  let y = 5;
}
console.log(y);  // error
```

```
const MAX = 10;
```

```
if (true) {
  var x = 5;
}
console.log(x);  // 5
```

Datatypes

UNIVERSITEIT GENT

- Number
- Boolean
- String
- null
- undefined

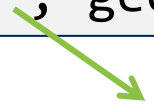
```
let n = null;
console.log(n * 32); // 0
```

```
let input;
if (input === undefined){
  doThis(); //uitgevoerd
} else { doThat(); }
```

```
let antwoord = 75;
antwoord = “Dank u wel”;
antwoord = “Het resultaat is " + 75; // “Het resultaat is 75”
antwoord = “37” + 7; // “377”
```


Array

```
let namen = ["Veerle", "Thomas", "Joris"];
let kleuren = ["rood", ,"geel"];
```



undefined

```
kleuren[1] = "groen";
kleuren[3] = "blauw";
```

```
let persoon = {voornaam: "Veerle", naam: "Ongenae"}
let myArray = new Array("Hello", persoon, 3.14159);
```

Objecten

UNIVERSITEIT GENT

- Containers voor eigenschappen (*properties*)
  - Sleutel/waarde-paren
  - Waarden: primitieve types, andere objecten, …

```
let persoon = {naam: "Janssens", voornaam: "Nele", schoenmaat: 40};
```

- Operatoren . en []
  - Toegang tot eigenschappen

Objecten - eigenschappen

```
let persoon = {naam: "Janssens", voornaam: "Nele", schoenmaat: 40};

persoon.naam = "Ongenae";
persoon.voornaam = "Veerle";
persoon.schoenmaat = 38;

persoon["naam"] = "De Smedt";
persoon["voornaam"] = "Thomas";
persoon["schoenmaat"] = 45;
```

```
let foo = {a: "alpha", 2: "two"};

let test = foo.a;       // "alpha"
test = foo["a"];        // "alpha"
test = foo[2];          // "two"
test = foo["2"];        // "two"
```

Destructuring assignment

UNIVERSITEIT GENT

- Items uit een array aan verschillende variabelen toekennen

```
let [x, y] = [2.3, 8.4];
console.log(x); // 2.3
console.log(y); // 8.4
console.log(`(${x},${y})`); // (2.3,8.4)
```

destructuring assignment

template string

- Eigenschappen van een object aan verschillende variabelen toekennen

```
let persoon = {naam: "Janssens", voornaam: "Nele", schoenmaat: 40};
let {voornaam, schoenmaat} = persoon;
console.log(`${voornaam} heeft schoenmaat ${schoenmaat}`);
```

```
Nele heeft schoenmaat 40
```


De programmeertaal JavaScript

UNIVERSITEIT GENT

- Variabelen en types
- Functies en methodes

Functie - declaratie

```
function som(a,b) {
    return a+b;
};
```

standaardwaarde

```
function multiply(a, b = 1) {
  return a * b;
}
console.log(multiply(5, 2));     // 10
console.log(multiply(5));        // 5
```

spread operator

```
function somVeel(...items) {
    let som = 0;
    for (let item of items) { som += item; }
    return som;
}
console.log(somVeel());                  // 0
console.log(somVeel(17));                // 17
console.log(somVeel(2,9, -7, 2.3));      // 6.3
```

Functie-uitdrukkingen

```
let som = (a,b) => {
    return a+b;
};
```

```
function pasFunctieToeOpArray(functie, lijst) {
    let resultaat = new Array;
    for (let i = 0; i < lijst.length; i++)
        resultaat[i] = functie(lijst[i]);
    return resultaat;
}
```

```
let resultaat =
   pasFunctieToeOpArray((x) => { return x * x * x;}, [0, 1, 2, 5, 10]);
```

ALTERNATIEF

```
let resultaat = pasFunctieToeOpArray(x => x * x * x, [0, 1, 2, 5, 10]);
```

Array - lussen

```
let kleuren = ["rood", "groen", "geel", "blauw"];
```

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

```
for (let kleur of kleuren) {
    console.log(kleur);
}
```

```
function toon(item) {
    console.log(item);
}

kleuren.forEach(toon);
```

uitvoer (3x)

```
rood
groen
geel
blauw
```

callback function

Lussen en objecten


```
let persoon = {naam: "De Smedt", voornaam: "Thomas", schoenmaat: 45};
for (let kenmerk in persoon) {
    let waarde = persoon[kenmerk];
    console.log("kenmerk: " + kenmerk + ", waarde: " + waarde);
}
```

```
kenmerk: naam, waarde: De Smedt
kenmerk: voornaam, waarde: Thomas
kenmerk: schoenmaat, waarde: 45
```

```
let kleuren = ["rood", "groen", "geel", "blauw"];
for (let index in kleuren) {
    console.log(index + ": " + kleuren[index]);
}
```

```
0: rood
1: groen
2: geel
3: blauw
```

Methodes array

```
let kleuren = ["rood", "groen", "geel", "blauw"];
```

```
kleuren.forEach((itemLijst, index) =>
    console.log("Kleur " + (index + 1) + ": " + itemLijst)
);
```

uitvoer

```
Kleur 1: rood
Kleur 2: groen
Kleur 3: geel
Kleur 4: blauw
```

```
let gevonden = kleuren.some(kleur => kleur === "rood")
```

Is er één?

Waarde en type gelijk

```
let getallen = [4, 86, 15, 37, 89, 6, 39];
let drievoud = getallen.find(getal => getal%3 === 0);
```

Eerste die voldoet

```
15
```

```
let getallen = [4, 86, 15, 37, 89, 6, 39];
getallen = getallen.filter(getal => getal < 10);
```

Alle waarden die voldoen

```
[4, 6]
```

Methodes array

```
let getallen = [4, 86, 15, 37, 89, 6, 39];
let rest = getallen.map(getal => getal%4);
```

```
[0, 2, 3, 1, 1, 2, 3]
```

nieuwe array: functie toepassen op alle items in de array

```
let getallen = [4, 86, 15, 37, 89, 6, 39];
let getal = getallen.reduce(
    (resultaat, item) => Math.max(resultaat, item));
```

```
89
```

waarde berekenen op basis van array

accumulator of tussenwaarde (=geaccumuleerde waarde)

huidige waarde

Methode: functie geassocieerd met een object

UNIVERSITEIT GENT

```
function langeNaam() {
  return this.naam + " " + this.voornaam;
}
persoon.langeNaam = langeNaam;
```

this: eigenaar functie

methode

```
let naam = persoon.langeNaam();
```


Geneste functies en “closure”

```
function addSquares(a, b) {
  function square(x) {
    return x * x;
  }
  return square(a) + square(b);
}
```

geneste (interne) functie

Closure

- Interne functie: toegang tot alle variabelen en functies huidige scope
- Toegang blijft ook als omringde functie verdwijnt = Closure

Closure - voorbeeld

UNIVERSITEIT GENT

```
function pet(name) {        // declaratie parameter name
    let myPet, words = ["Food!", "Sleep!", "Video games!"]; // declaratie words

    function random(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    myPet = {}; // aanmaken myPet → resultaat functie
    myPet.getName = () => name;
    myPet.speak = () => console.log(words[random(0,2)]);

    return myPet;
}

let myPet = pet("Vivie");
console.log(myPet.getName()); // "Vivie"
myPet.speak();                // Food!", "Sleep!" of "Video games!"
```

interne functies

De programmeertaal JavaScript

UNIVERSITEIT GENT

- Variabelen en types
- Functies en methodes
- Objecten en klassen

Objecten en klassen

UNIVERSITEIT GENT

- Voor ECMAscript 6 (2015) waren er geen klassen
  - Functies om objecten te maken

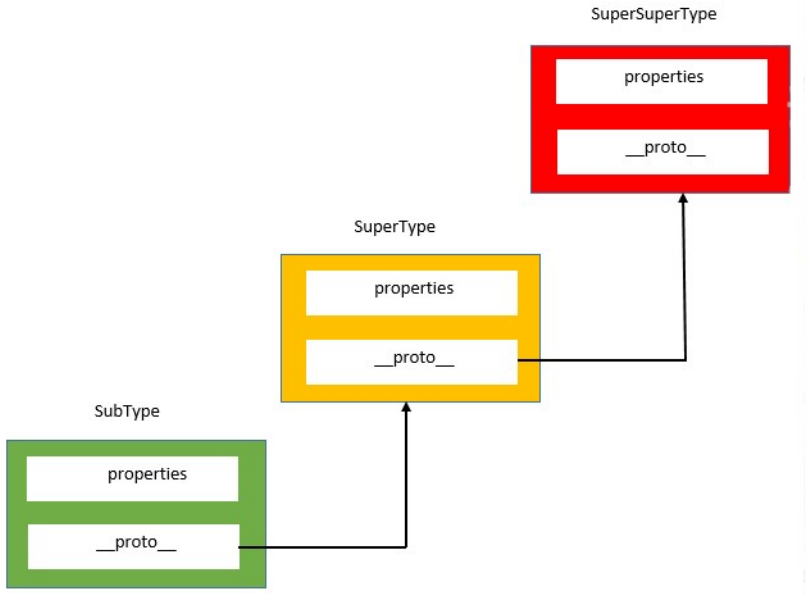
```
let cat = animal({
    say: "Meow",
    fur: "black"
});

cat.speak();

console.log(Object.keys(cat));
```

```
Meow
["legs", "eyes", "say",
   "speak", "fur"]
```

```
function animal(config) {
    //template
    let pet = {
        legs: 4,
        eyes: 2,
        say: "Huh?",
        speak() {
            console.log(this.say);
        }
    };
    //Copy properties
    Object.assign(pet, config);
    return pet;
}
```

Klassen in JavaScript

- Speciale functies
- Prototype
  - Elk object heeft een prototype
  - Object is einde prototype-ketting




```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  // Getter
  get area() {
    return this.calcArea();
  }
  // Method
  calcArea() {
    return this.height * this.width;
  }
}

const square = new Rectangle(10, 10);

console.log(square.area); // 100
```

https://medium.com/@raphaelkhan8/prototypal-inheritance-in-javascript-773e432b83e

Overerven van objecten

UNIVERSITEIT GENT

```
class Monster {
    constructor(hitPoints, scariness) {
        this.name = "Monster";
        this.hitPoints = hitPoints;
        this.scariness = scariness;
    }
    speak() {
        console.log(`I'm a ${this.scariness} scary ${this.name} with ${this.hitPoints} hit points`);
    }
    attack(skill) {
        console.log(`The ${this.name} attacks with ${skill}!`);
    }
}
```

```
class Dragon extends Monster {
    constructor(hitPoints, scariness, weapon) {
        super(hitPoints, scariness);
        this.name = "Dragon";
        this.weapon = weapon;
    }
    breatheFire() {
        super.attack(`flaming ${this.weapon}`);
    }
}
```


De programmeertaal JavaScript

UNIVERSITEIT GENT

- Variabelen en types
- Functies en methodes
- Objecten en klassen
- Modules

Modules

- Op zich zelf staande stukken code
- Afgeschermd van globale scope en andere modules
- Beschikbaar stellen: export
- Gebruiken: import
- Ondersteund door recente browsers
- Soms extensie .mjs
- Sinds ECMAScript 6
- Andere types: CommonJS, AMD, …

Module – export - import

speelgoed.js

```
export {Car, Animal}

class Car {
  …
}

class Animal {
  …
}
```

```
import {Car} from "./speelgoed.js";
import {Animal as Dier} from "./speelgoed.js" ;
                              alias

let myCar = new Car("Eagle", "Talon TSi", 1993);

let cat = new Dier({
    say: "Meow",
    fur: "black"
});
```

Volledige module importeren

speelgoed.js

```
export {Car, Animal}

class Car {
  …
}

class Animal {
  …
}
```

```
import * as speelgoed from "./speelgoed.js";

let myCar = new speelgoed.Car("Eagle", "Talon TSi", 1993);

let cat = new speelgoed.Animal({
    say: "Meow",
    fur: "black"
});
```

Module – default export

UNIVERSITEIT GENT

- Module bevat slechts één klasse, functie, variabele, …

Auto.js

```
export default class Auto {
  …
}
```

Dier.js

```
export default class {
  …
}
```

```
import Auto from "./Auto.js";
import Dier from "./Dier.js";

let mijnAuto = new Auto("Eagle", "Talon TSi", 1993);

let mijnKat = new Dier({
    say: "Meow",
    fur: "black"
});
```


Exporteren van functies

mijnfunctie.js

```
export default function () {
  …
}
```

```
import mijnFunctie from "./mijnfunctie.js";
```

De programmeertaal JavaScript

UNIVERSITEIT GENT

- Variabelen en types
- Functies en methodes
- Objecten en klassen
- Modules