

JS

H2 Herhaling HTML

Zoek op gebruik form

Formmethode:

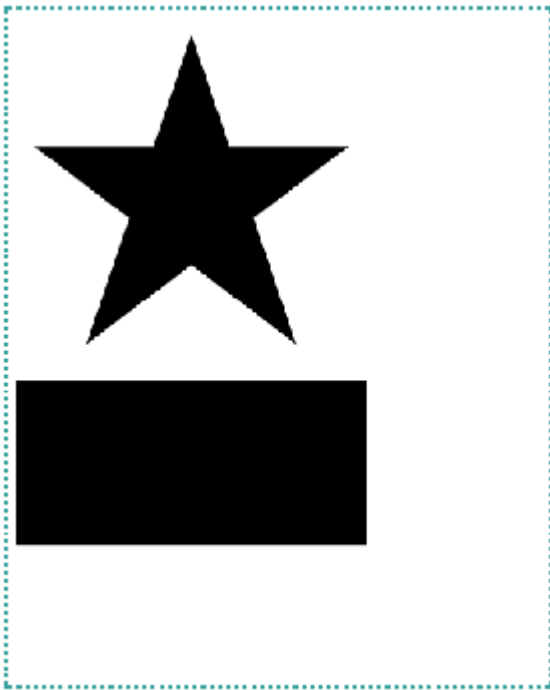
- POST → info op server aanpassen
- GET → info vragen met form als filter (vb: Google Search)

HTML5 = HTML4 + XHTML + DOM API

SVG = Scalable Vector Graphics, figuur in XML-formaat, figuur is resolutie onafhankelijk

```
svg {  
    margin: 5px;  
    padding: 5px;  
    border-color: cadetblue;  
    border-width: 2px;  
    border-style: dotted  
}
```

```
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="400">  
    <polygon id="ster" points="100,10 40,198 190,78 10,78 160,198"/>  
    <rect x="0" y="220" height="100" width="200"/>  
</svg>
```



SVG-bestand als figuur

.html

```

```

UNIVERS
GENT

.svg



```
<svg xmlns="http://www.w3.org/2000/svg" width="300" height="400">
  <circle cx="50" cy="171" r="40"
    style="fill: white;stroke: gray;stroke-width: 1px;"/>
  <circle cx="50" cy="103" r="30"
    style="fill: white;stroke: gray;stroke-width: 1px;"/>
  <circle cx="50" cy="50" r="25"
    style="fill: white;stroke: gray;stroke-width: 1px;"/>
  <line x1="30" y1="25" x2="70" y2="25"
    style="fill: black; stroke: black; stroke-width: 3px;"/>
  <rect x="40" y="10" width="20" height="15"
    style="fill: black; stroke: black; stroke-width: 3px;"/>
  <circle cx="50" cy="82" r="4" style="fill: black;"/>
  <circle cx="50" cy="100" r="4" style="fill: black;"/>
  <circle cx="50" cy="118" r="4" style="fill: black;"/>
  <circle cx="42" cy="42" r="4" style="fill: black;"/>
  <circle cx="58" cy="42" r="4" style="fill: black;"/>
  <polygon class="nose" points="45,60 45,50 60,55" style="fill: orange;"/>
</svg>
```

FACULTEIT INGENIEUR
EN ARCHITECTUUR

H3 Javascript

Variabelen

Geen sterke typering, is gelijk in Python, types worden dynamisch (at runtime) toegekend en eventueel veranderd

Declaratie:

- let: block scope, een variabele die enkel gebruikt kan worden binnen dezelfde scope, meest veilig
- const: constante (gelijk final)
- var: globale scope of functie scope (oude manier)

```

if (true) {
    let y = 5;
}
console.log(y); // ERROR
-----

const MAX = 10;
-----

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

```

Datatypes

- Number
- Boolean
- String
- null
- undefined

```

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

let input; // undefined
if(input === undefined) {
    doThis(); // UITGEVOERD
}
else {
    doThat(); // NIET UITGEVOERD
}

let antwoord = 75; // Number
antwoord = "Dank u wel"; // String
antwoord = "Het resultaat is " + 75; // "Het resultaat is 75"
antwoord = "37" + 7; // "337"

```

==: vergelijk inhoud

===: vergelijk inhoud EN datatype

Array

Eigenlijk een lijst

```
let namen = ["...", "...", "..."];
let kleuren = ["rood", , "groen"]; // Index 1 is undefined

kleuren[1] = "groen"; // Mutable en invulbaar
kleuren[2] = "blauw";

let persoon = {voornaam: "Jarno", achternaam: "Vanruymbeke"}; // OBJECT
let myArray = new Array("Hello", persoon, 3.14); // Heterogeen
```

Objecten

Containers voor eigenschappen

Sleutel/waarde-paren

Waarden: alle datatypes, ook objecten

Operatoren [] en . gebruiken om eigenschappen te kunnen bereiken

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

persoon.naam = "Veerle";
...

persoon["naam"] = "De Smedt";
...

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

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

GEEN foo.2, MAG NIET
```

Destructuring assignment

Items uit array toekennen aan verschillende variabelen

```
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);
```

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}`);
```

Template string

`text \${variabele} text \${variabele} text ...`

Functie

Declaratie

```
function som(a, b) {  
    return a+b;  
}  
  
function multiply(a, b = 1) { // Defaultwaarde  
    return a*b;  
}  
  
function somVeel(...items) { // We kunnen een onbeperkt aantal variabelen  
    // Numbers meegeven met deze function, spread operator  
    let som = 0;  
    for(let item of items) {  
        som += item;  
    }  
    return som;  
}
```

ONDERZOEK WAAR DEZE MOET STAAN INDIEN ER MEERDERE PARAMETERS ZIJN

Uitdrukkingen

```
let som = (a, b) => { return a+b; }; // LAMBDA FUNCTIE  
  
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]);  
OF  
let resultaat = pasFunctieToeOpArray(x => x*x*x, array hier);
```

Array

Lussen

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

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

// Foreach over elementen door middel van OF
for(let kleur of kleuren) {
    console.log(kleur);
}

// Foreach kort
function toon(item) {
    console.log(item);
}
kleuren.forEach(toon); // Mag ook lambda functie zijn

// IN gebruiken om over kenmerken van object te lopen of indexen van arrays
let persoon = {naam: "De Smedt", voornaam: "Thomas", schoenmaat: 45};

for(let kenmerk in persoon) {
    let waarde = persoon[kenmerk];
    console.log("kenmerk: " + kenmerk + ", waarde: " + waarde);
}

for(let index in kleuren) {
    console.log(kleuren[index]);
}
```

Methodes

```
let kleuren = ["rood", "groen", "geel", "blauw"];
kleuren.forEach((itemLijst, index) =>
    console.log("Kleur " + (index + 1) + ": " + itemLijst);
);

let gevonden = kleur.some(kleur => kleur === "rood"); // True als er 1 voldoet

let getallen = [4, 86, 15, 37, 89, 6, 39];
let drievoud = getallen.find(getal => getal%3 === 0); // 1e die voldoet, 15

let getallen_filter = getallen.filter(getal => getal < 10); // [4, 6]
```

```
let rest = getallen.map(getal => getallen%4); // [0, 2, 3, 1, 1, 2, 3]

let getal = getallen.reduce((resultaat, item) => Math.max(resultaat, item))
// Waarde berekenen op basis van waarde uit de lijst en tussenwaarde
```

Methode: functie geassocieerd met een object

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

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

Geneste functies en "closure"

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

Closure:

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

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

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

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

    return myPet;
}

let myPet = pet("Vivie");
console.log(myPet.getName()); // "Vivie"
myPet.speak(); // 1 van de 3
```

Objecten

Klassen

Er zijn geen acces modifiers of voorgemaakte attributen

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }

    // Getter
    get area() {
        return this.calcArea();
    }

    // Methode
    calcArea() {
        return this.height * this.width;
    }
}

const square = new Rectangle(10, 10);
console.log(square.area); // 100
```

Prototype:

- Elk object heeft een prototype
- Object is een einde prototype-ketting

Overerven van objecten

```
class Monster {
    constructor(...) {}

    attack(skill) {
        iets met skill
    }
}

class Dragon extends Monster {
    constructor(...) {
        super(...);
        ...
    }

    breatheFire() {
```



```
        super.attack(...);  
    }  
}
```

Modules

- Op zich zelf staande stukken code
- Afgeschermd van globale scope en andere modules
- Beschikbaar stellen: export
- Gebruiken: import
- Ondersteund door recente browsers
- Soms extensies .mjs

Export - import

```
// In: speelgoed.js  
export {Car, Animal}  
  
class Car {  
    ...  
}  
  
class Animal {  
    ...  
}  
  
// In main.js  
import {Car} from "./speelgoed.js";  
import {Animal as Dier} from "./speelgoed.js"; // ALIAS  
  
let myCar = new Car{...}  
let cat = new Dier({  
    say: "Meow",  
    fur: "black"  
});  
  
import * as speelgoed from "./speelgoed.js"; // ALLES IMPORTEREN  
speelgoed.Car(...);
```

Default export

Module bevat slechts 1 klasse, functie, variabele, ...

```
// Auto.js  
export default class Auto {  
    ...  
}
```

```

}

// Dier.js
export default class {
    ...
}

import Auto from "./Auto.js";

```

Exporteren van functies

```

export default function() {
    ...
}

import mijnFunctie from "./mijnFunctie.js";

```

JS toevoegen aan HTML

```

<script src="script.js"></script>

OF

<script>
    code here
</script>

```

In header: op te roepen functies, initialisatie

In body: onmiddellijk uit te voeren opdracht

Uitvoeren JS in HTML

Laden HTML

- Browser stopt het laden bij script-tag
- Script wordt opgehaald en uitgevoerd
- Browser gaat verder met het laden van de pagina

Script asynchroon laden en daarna onmiddellijk uitvoeren

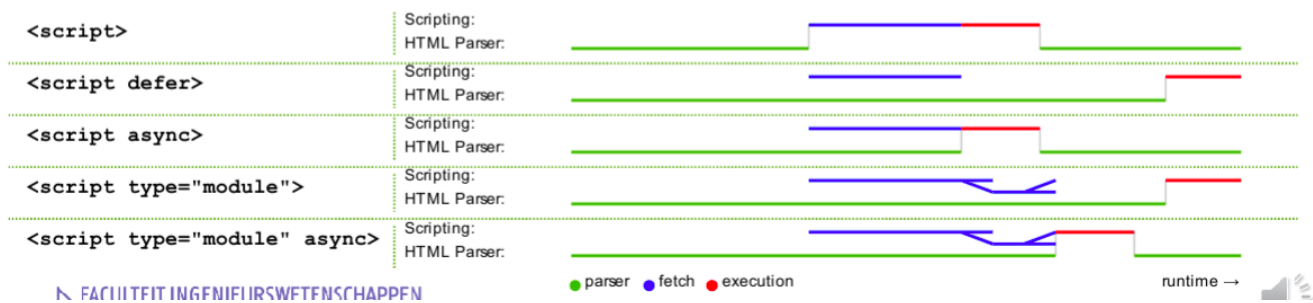
```
<script async src="script.js"></script>
```

Script pas uitvoeren nadat paginastructuur (DOM-boom) opgesteld is

```
<script defer src="script.js"></script>
```

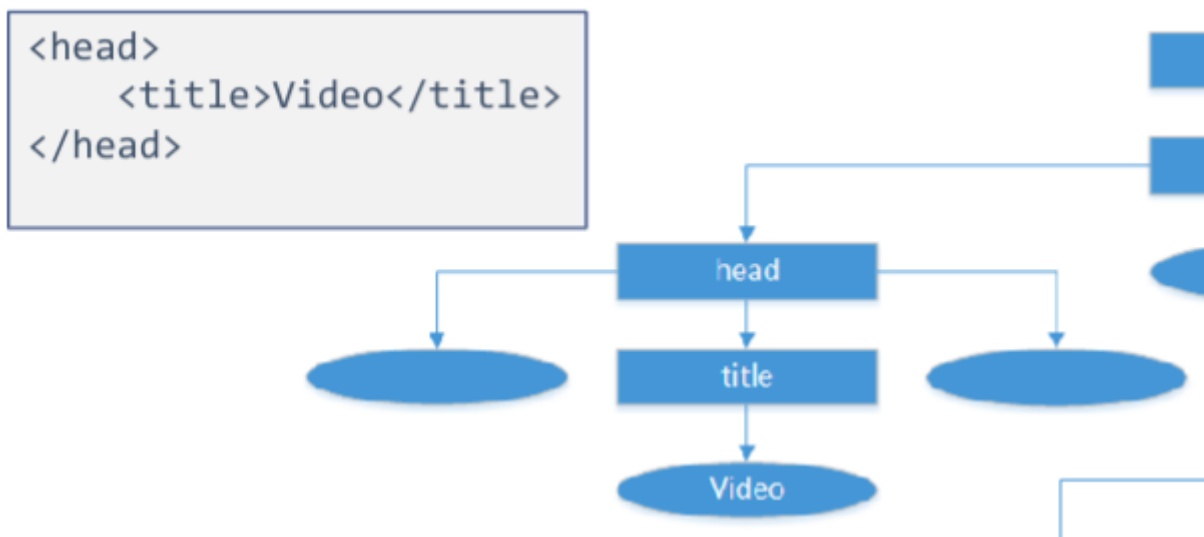
Module -> standaard defer

```
<script src="speelgoed.js" type="module"></script>
```



HTML DOM

- Webpagina
 - Boom van objecten
- Document Object Model (DOM)
 - Model
 - API manipulation webpagina
- Deel van DOM
 - Specificatie W3C
 - DOM lvi 3



Boom bestaat uit knopen (node)

- Document zelf (document)
- Elementen
- Attributen
- Tekst
- ...

Toegang tot HTML

Elementen selecteren

```
document.getElementById(string id); // 1 element
document.getElementsByClassName(string class); // lijst elementen, dus
indien 1 moet je indexeren
document.getElementsByName(string name); // Lijst
document.getElementsByTagName(string tag): // Lijst, vb buttons, ...
```

Attributen

```
<div id="div1">Hi Champ!</div>
```

```
let div1 = document.getElementById("div"); // <div id="div1">Hi Champ!
</div>
let div1_id = div1.getAttribute("id"); // "div1"

OF element.attrNaam

div1.setAttribute("id", "ID1"); // <div id="ID1">Hi Champ!</div>
```

Tekst

```
<div id="div1">This is <span>some</span> text!</div>
```

```
let tekst = document.getElementById("div1").textContent; // This is some
text!

// Past HTML aan
document.getElementById("div1").textContent = "This text is different!";
```

Invoervelden

```
<input type="text" name="naam" size="20">
<textarea id="textOutput" ...> ... </textarea>
```

```
let invoer = document.getElementsByName('naam')[0].value; // Inhoud
tekstvak
document.getElementById("textOutput").value = invoer;
```

Documentstructuur aanpassen

Nieuwe knopen maken

```
document.createAttribute(...);
document.createComment(...);
document.createElement(...);
document.createTextNode(...);
```

Knopen vervangen/verwijderen

```
element.removeChild(...);  
element.replaceChild(newChild, oldChild);
```

Knopen toevoegen aan boomstructuur

```
element.appendChild(...);  
document.appendChild(...);
```

Vb: lijst opvullen met dagen van de week

```
<html>  
  <head>  
    ...  
    <script src="js/dagenWeek.js"></script>  
  </head>  
  <body>  
    Kies dag:  
    <select id="keuzeDag"></select>  
    <script>  
      vulLijst();  
    </script>  
  </body>  
</html>
```

```
function vulLijst() {  
  let lijst = document.getElementById("keuzeDag");  
  let dagen = ["maandag", ...];  
  for(let dag of dagen) {  
    let optie = document.createElement("option");  
    optie.appendChild(document.createTextNode(dag)); //  
<option>dag</option>  
    lijst.appendChild(optie);  
  }  
}
```

Events

- Browser
 - Laden documenten, figuren, ...
- Gebruiker
 - Muis
 - Toetsenbord
- Attributen
 - onload, onunload

- onfocus, onchange, onblur
- onkeydown, onkeyup
- onmouseover, onmousemove, onmouseout
- onsubmit
- onclick
- ...

Afhandelen events

1. Attribuut toevoegen in HTML: deprecated, IDK HOE
2. Eventhandler via property

```
function doeIets() {...};

let knop = document.getElementById("knop");
knop.onclick = doeIets;
```

3. Eventhandler via functie

```
function doeIets() {...};

let knop = document.getElementById("knop");
knop.addEventListener("click", doeIets);
```

Vb: rekenmachine

```
<script src="js/som.js" defer></script>

<form>
  <input id="getal1" type="text"> +
  <input id="getal2" type="text"> =
  <input id="som" type="text" readonly>
</form>
```

```
function init() {
  let getal1 = document.getElementById("getal1");
  let getal2 = document.getElementById("getal2");
  getal1.onchange = berekenSom;
  getal2.onchange = berekenSom;
}

function berekenSom() {
  let getal1 = document.getElementById("getal1");
  let getal2 = document.getElementById("getal2");
  let som = parseInt(getal1.value) + parseInt(getal2.value);
  let somText = document.getElementById("som");
```

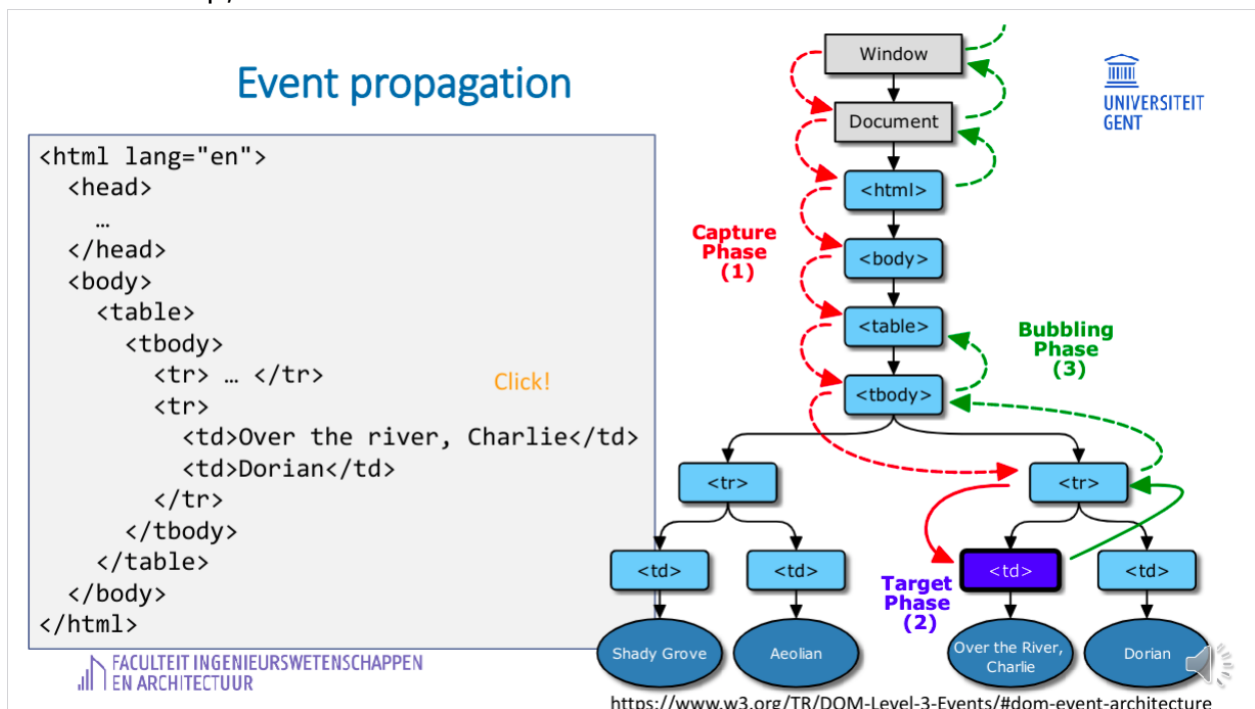
```
somText.value = som.
}

init();
```

Event bubbling en event capture

```
<p>
    <button id="knop">...</button>
</p>
```

- Volgorde waarin eventhandlers uitgevoerd worden
- Event bubbling
 - Standaard
 - Eerst event "meest binnenste element"
 - click-event button, click-event p
- Event capture
 - Eerst event "meest buitenste element"
 - `addEventListener(event, function, useCapture);`
 - click-event p, click-event button



Event-object

- Wordt meegegeven met eventhandler
- Voorzie een parameter voor eventhandler
- Eigenschappen
 - type
 - target

- currentTarget
- bubbles
- ...

Canceling events

Naam methode	Omschrijving
event.preventDefault()	Standaardactie voor event wordt niet uitgevoerd
event.stopImmediatePropagation()	Andere eventhandlers op het element worden niet uitgevoerd en het event bubbelt niet verder naar (voor)ouderelementen
event.stopPropagation()	Zorgt ervoor dat het event niet verder bubbelt naar (voor)ouderelementen

Validatie

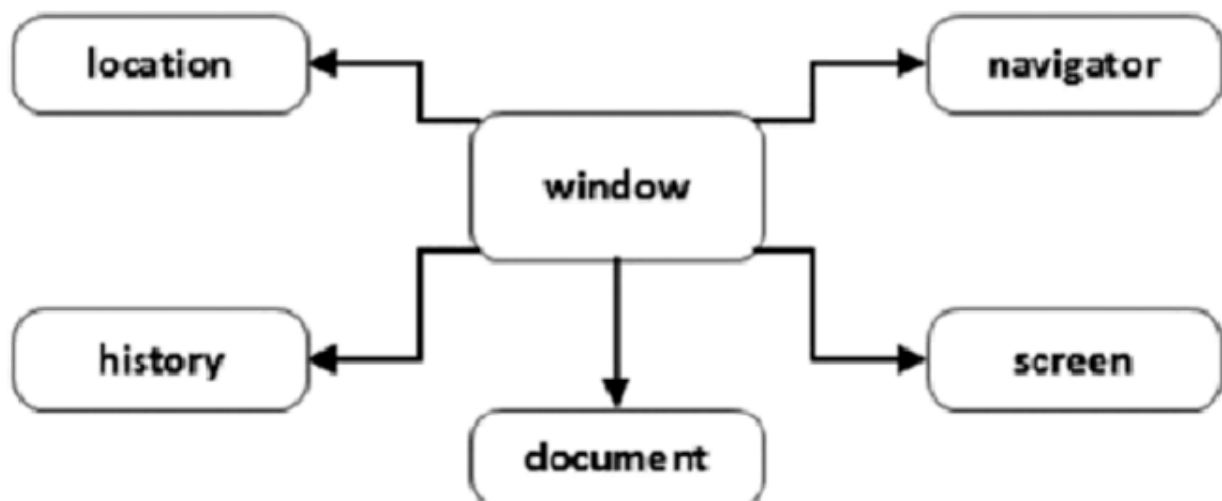
- Controle invoer gebruiker
- Formulier wordt niet ingediend indien controle mislukt
 - Event is "cancelable": standaardactie (indien formulier) kan stopgezet worden
 - onsubmit, onclick
 - Eventhandler: logische functie

In HTML tag: required

Eind na start, indien false: niet reserveren

Vb: controleData.html

Browser Object Model



Timer

```

// 1 keer een functie uitvoeren
// timer instellen
let timer = setTimeout(fct, tijdsinterval);
  
```



```

...
// timer uitschakelen
clearTimeout(timer);

// herhaaldelijk een functie uitvoeren
let timer = setInterval(fct, tijdsinterval);

...
clearInterval(timer);

```

Vb: slides 32-33

Popups

```

window.alert("Naam invoeren!"); of alert("Naam invoeren!");

let antwoord = confirm("..."); // true/false

let invoer = prompt("..."); // ingevoerde tekst

```

Vb: slides 35-37

Canvas API

Gebied om op te tekenen in browser

```

<canvas id="myCanvas" width="200" height="100">
    Your browser does not support the canvas element.
</canvas>

```

```

let c = document.getElementById("myCanvas");
let cxt = c.getContext("2d");
cxt.fillStyle = "#FF0000";
cxt.beginPath();
cxt.arc(70, 18, 15, 0, Math.PI*2, true); // x, y, r, 0 tot 2pi
// tegenwijzerszin
cxt.closePath();
cxt.fill();

```

```

Rechthoeken
fillRect(x, y, width, height) // Volle
strokeRect(x, y, width, height) // Omlijning
clearRect(x, y, width, height) // Clear

```

```

Lijnen en curves
beginPath();
moveTo(x, y);
lineTo(x, y);

```

```
arc(x, y, r, startAngle, endAngle, CCW)
closePath(); // Verbind met beginPath
fill();
stroke();
```

Opmaak

- Afbeeldingen
 - drawImage(image, x, y)
- Kleur
 - Eigenschappen
 - fillStyle
 - strokeStyle
 - Waarden
 - CSS-kleur: "#RRGGBB", rgb(r, g, b), rgba(r, g, b, alpha), ...
 - Gradient
 - Patroon
- Tekst
 - font
 - textAlign
 - textBaseline
 - fillText(text, x, y [, maxWidth])
 - strokeText(text, x, y [, maxWidth])

Canvas

- Toestand
 - save()
 - restore()
- Canvas vervormen
 - translate(x, y)
 - rotate(angle)
 - scale(x, y)
 - transform(m11, m12, m21, m22, dx, dy)
 - setTransform(m11, m12, m21, m22, dx, dy)

Meer tekenen

- Bezierkrommen
- Kwadratische krommen
- Gradiënten
- Patronen

- Overlappende vormen

Vb: cirkels.html

Canvas vs SVG

- SVG
 - Scalable Vector Graphics
 - Figuur in XML-formaat
 - Elke vorm is een object
 - Onafh. v. resolutie
- Canvas
 - Tekening wordt niet onthouden
 - Betere performantie indien veel "objecten"

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
    style="fill:grey;stroke:maroon;stroke-width:5;fill-rule:evenodd;" />
  <rect x="0" y="220" height="100" width="200" />
</svg>
```



INENIEURSWETENSCHAPPEN
Industrial Engineer Informatica UGent

Web Storage API

- Info bewaren op client
- 2 opties
 - localStorage: permanent
 - sessionStorage: zolang tabblad niet gesloten wordt

```
// localStorage
localStorage.setItem("taal", "frans");
let taal = localStorage.getItem("taal");
```

OF

```
localStorage.taal = "frans";
let taal = localStorage.taal;
```

```
// sessionStorage
```

```
sessionStorage.setItem("naam", "Koen");
let naam = sessionStorage.getItem("naam");

OF

sessionStorage.naam = "Koen";
let naam = sessionStorage.naam;
```

Naam/waarde-paren (strings)

Data bewaren

```
let gameData = {
  playerName: "Rex",
  levelCompleted: 5,
  score: 84,
  items: ["hat", "umbrella", "katana"]
};

let gameDataJSON = JSON.stringify(gameData);
localStorage.setItem("gameData", gameDataJSON);

loadedData = localStorage.getItem("gameData");
let data = JSON.parse(loadedData);
```

JSON

Textuele voorstelling van een JS object (serialisatie)

Bijna alles is een string

```
{
  "closet": {
    "id": 0,
    "description": "A dark coat closet.",
    "light": { "on": false },
    "contents": [],
    "exits": ["east"]
  },
  "livingRoom": {
    "id": 1,
    "description": "A living room in an old, rambling house.",
    "light": {
      "on": true
    },
    "contents": ["fireplace", "sofa", "dagger"],
    "exits": ["west", "north", "south"]
  }
}
```

Inlezen

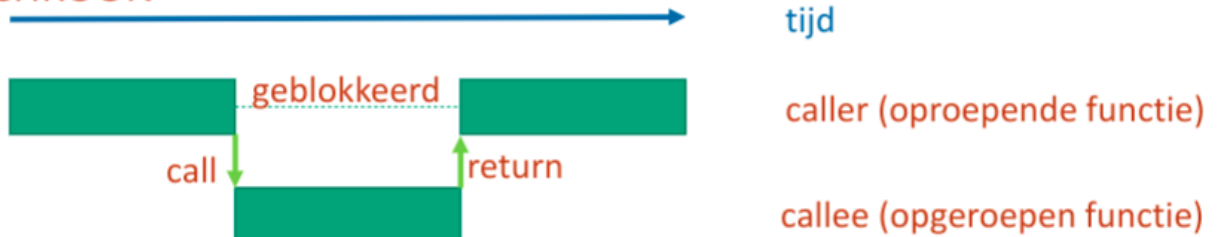
```
fetch("json/rooms.json") // Stuur HTTP-bericht (GET)
  .then(response => response.json()) // Antwoord verwerken
  .then(json => { // Antwoord (tekst) -> js-object
    rooms = json; // Object bewaren
  });
```

Frameworks, tools, ...

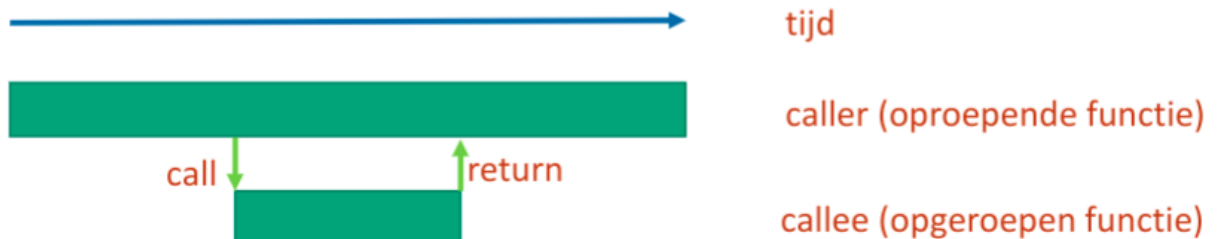
- Meeste browsers hebben ingebouwde debug-tool (F12)
- JSLint: controle JS-code
- JS-bibliotheken
 - JQuery
 - Angular
 - React
 - D3.js C3.js
 - ...
- TypeScript: Programmeertaal uitbreiding op JS
- Built tools: Grunt Gulp
- Testing tools: Mocha&Chai; Karma
- Console/serverside: node.js

Synchrone vs asynchrone functie

SYNCHROON



ASYNCHROON



```
// Synchron
result = query('SELECT * FROM posts WHERE id = 1');
do_smt_with(result);
```

```
// Asynchroon
query('SELECT * FROM posts WHERE id = 1', do_smt_with);
// do_smt_with is de 'callback-functie'
```

```
// Synchroon
let fs = require("fs");
let data = fs.readFileSync('input.txt');
console.log(data.toString());
console.log("Program Ended");
```

- 1) Inhoud bestand
- 2) Program Ended

```
// Asynchroon
let fs = require("fs");
fs.readFile('input.txt', function(err, data) {
    if(err) return console.error(err);
    console.log(data.toString());
})
console.log("Program Ended");
```

- 1) Program Ended
- 2) Inhoud bestand

Callback hell

Veel functies met callbacks die genest zijn

```
request('https://www.somepage.com',
    function (firstError, firstResponse, firstBody) {
        if(firstError) {
            // Handle error
        }
        else {

request('https://www.somepage.com/${firstBody.someValue}',
    function (secondError, secondResponse, secondBody) {
        if(secondError) {
            // Handle error
        }
        else {
            // Use secondBody for smt
        }

    });

}

});
```

Promises

- Oplossing voor callback hell
- Promise = object toekomstig resultaat asynchrone functie
 - 3 toestanden: nog niet afgerond, vervuld, fout opgetreden
- Asynchrone functie
 - Functie keert onmiddellijk terug
 - Caller wacht niet op volledige uitvoeren functionaliteit
- Promise laat toe om callback of foutafhandeling toe te voegen

Syntax promise

```
new Promise(executor);
```

Executor:

- Functie met 2 parameters
 - resolve
 - reject
- De executor-functie wordt onmiddellijk uitgevoerd en start asynchrone opdrachten. Als die opdrachten klaar zijn roept hij de resolve-functie op. Als er een fout optrad, wordt de reject-functie opgeroepen.

```
let p = new Promise((resolve, reject) => {
  console.log("In functie");
  let kans = Math.floor(Math.random() * 2);
  console.log("kans: " + kans);
  if(kans === 0) {
    resolve("Gelukt")
  }
  else {
    reject("Fout")
  }
});
```

```
p.then((tekst) => console.log(text)).catch((fout) => console.log(fout));
console.log("Na functie")
```

- 1) In functie
- 2) kans: ...
- 3) Na functie
- 4) Gelukt of Fout

Promise chaining

```
function log(bericht) {
    console.log(bericht)
}

function logEnGok(bericht) {
    return new Promise((resolve, reject) => {
        log(bericht);
        if(Math.floor(Math.random() * 3) !== 0) {
            resolve("Gelukt");
        }
        else {
            reject("Fout");
        }
    });
}

logEnGok("Start").then(logEnGok).then(logEnGok).then(logEnGok).catch(log);
log("Na logEnGok");
```

- 1) Start
- 2) Na logEnGok
- 3) Gelukt of Fout
- 4) Gelukt of Fout
- 5) Gelukt of Fout

Asynchrone functies

- Gebruik van Promises te vereenvoudigen

```
async function name(paras) {
    statements
}
```

- Resultaat van een asynchrone functie is een Promise
- In een asynchrone functie kan een await-opdracht gebruikt worden. In dat geval pauzeert de uitvoering van de functie, start een asynchrone opdracht, keert terug naar de caller en wacht totdat de opgeroepen promise klaar is.

Vb

```
function doubleAfter2Seconds(x) {
    return new Promise(resolve => {
        setTimeout(() => {
            resolve(x * 2)
        }, 2000);
    });
}
```



```
doubleAfter2Seconds(10).then((r) => console.log(r));

function addPromise(x) {
  return new Promise(resolve => {
    doubleAfter2Seconds(10).then((a) => {
      doubleAfter2Seconds(20).then((b) => {
        doubleAfter2Seconds(30).then((c) => {
          resolve(x + a + b + c);
        });
      });
    });
  });
}

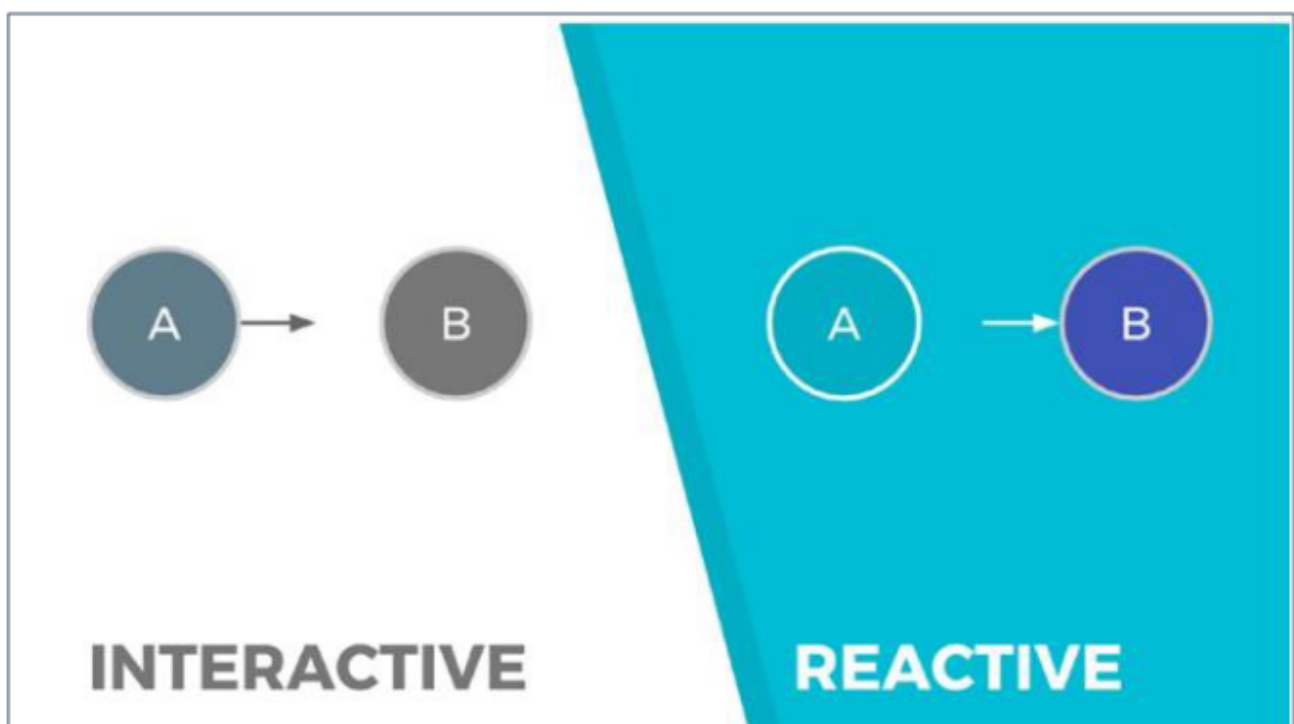
addPromise(10).then((sum) => console.log("Met promises", sum));
console.log("na oproep addPromise");
```

Met asynchrone functie

```
async function addAsync(x) {
  const a = await doubleAfter2Seconds(10);
  const b = await doubleAfter2Seconds(20);
  const c = await doubleAfter2Seconds(30);
  return x + a + b + c;
}

addAsync(10).then((sum) => console.log("Met async functie", sum));
console.log("na oproep addAsync");
```

Reactive programming



Interactive programming

- Object A kent de interface van object B

```
class Calculator {
  sum(a, b) {
    return a + b;
  }
}

class Receipt {
  constructor(calculator) {
    this.calc = calculator;
  }

  print(itemA, itemB) {
    const total = this.calc.sum(itemA, itemB);
    console.log("total receipt £" + total);
  }
}

const pizza = 6.00;
const beer = 5.00;
const calc = new Calculator();
const receipt = new Receipt(calc);

receipt.print(pizza, beer)
```

Reactive programming

Object A reageert op veranderingen in object B

- Zonder interface van B te kennen
- Kent enkel de methode subscribe
Een object, functie, stukje code, ... luistert en reageert op een veranderlijke dataflow

RxJS: veelgebruikte bibliotheek voor reactive programming

```
import * as rxjs from 'rxjs';
import * as ops from 'rxjs/operators/index.js';

class Calculator {
  constructor(itemA, itemB) {
    const obs = rxjs.of(itemA, itemB);
    const sum = obs.pipe(ops.reduce((acc, item) => (acc +
item)));
    return { observable: sum }
  }
}
```

```

    }
}

class Receipt {
    constructor(observable) {
        observable.subscribe(value => console.log('total receipt: €${value}'));
    }
}

const pizza = 6.00;
const beer = 5.00;
const calc = new Calculator(pizza, beer);
const receipt = new Receipt(calc.observable);

```

Programmeerparadigma's

- Imperative programming: stap voor stap beschrijven wat het programma moet doen
- Functional programming
 - Het programma beschrijft de dataflow
 - Functies gebruiken om op basis van bestaande waarden nieuwe waarden te maken
 - Waarden zijn immutable
 - Geen toestandsinformatie - instantievariabelen

RxJS operatoren

- map(functie): zal m.b.v. een functie de ene 'lijst' op de andere mappen

```

function addVAT(item) {
    return (1+22/100)*item;
}

function sum(...items) {
    return items.map(addVAT);
}

```

- reduce(functie): zal m.b.v. een functie de 'lijst' reduceren tot 1 waarde

```

function sum(...items) {
    return items.map(addVAT).reduce((acc, value) => acc+value);
}

```

Samen:

- map: [80.00, 35.00, 90.00, 140.00, 29.00] → [97.60, 42.70, 109.80, 170.80, 35.38]
- reduce: [97.60, 42.70, 109.80, 170.80, 35.38] → 456.28

Observable

- Een collectie van toekomstige waarden of events
 - Er kunnen waarden later toegevoegd worden
- Observer/Consumer (consumeert het resultaat v.e. pijplijn)
 - Reageert telkens er een waarde toegevoegd wordt (emit)
- Subscription
 - Voegt een observer (=functie) toe die uitgevoerd wordt bij een nieuwe waarde
 - Functie 'subscribe'

```
import {Observable} from "rxjs";

let observable = new Observable((observer) => {
  observer.next(1);
  observer.next(2);
  observer.next(3);
  setTimeout(() => {
    observer.next(4);
    observer.complete();
  }, 1000);
});

console.log('just b4 subscribe');
observable.subscribe( {
  next: x => console.log('got value ' + x);
  error: err => console.error('smt wrong occurred: ' + err);
  complete: () => console.log('done');
});

console.log('just after subscribe');
```

1) just b4 subscribe
 2) got value 1
 3) got value 2
 4) got value 3
 5) just after subscribe
 6) got value 4
 7) done

DIT IS ASYNCHROON

- Subscribing to an Observable is analogous to calling a Function
- Pas bij het oproepen van de methode subscribe worden de waarden in de Observable verwerkt

OBSERVABLE OP BASIS VAN EEN ARRAY

```
import * as rxjs from 'rxjs';

function onData(value) {
    console.log(value);
}

function onError(err) {
    console.error(err);
}

function onComplete() {
    console.log("stream complete!");
}

const lijst = Array.from(new Array(10), (x, i) => i + 1);
const observable = rxjs.from(lijst);
const observer = observable.subscribe(onData, onError, onComplete);

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, stream complete!
```

OBSERVABLE OP BASIS VAN INTERVAL

```
import * as rxjs from 'rxjs'

const source = rxjs.interval(1000); // Zal een sequentie van nummers om de 1000ms geven
sub1 = source.subscribe(value => console.log("eerste sub", value));
sub2 = source.subscribe(value => console.log("eerste sub", value));
```

Hot vs Cold Observable

- Cold Observable
 - Data geproduceerd in "Observable"

```
import {Observable} from 'rxjs';

const observable = new Observable(observer =>
    observer.next(Math.random()));

observable.subscribe(data => console.log(data)); // Random nummer

observable.subscribe(data => console.log(data)); // Ander random nummer
```

- Hot Observable
 - Data geproduceerd buiten "Observable"

```
import {Observable} from 'rxjs';

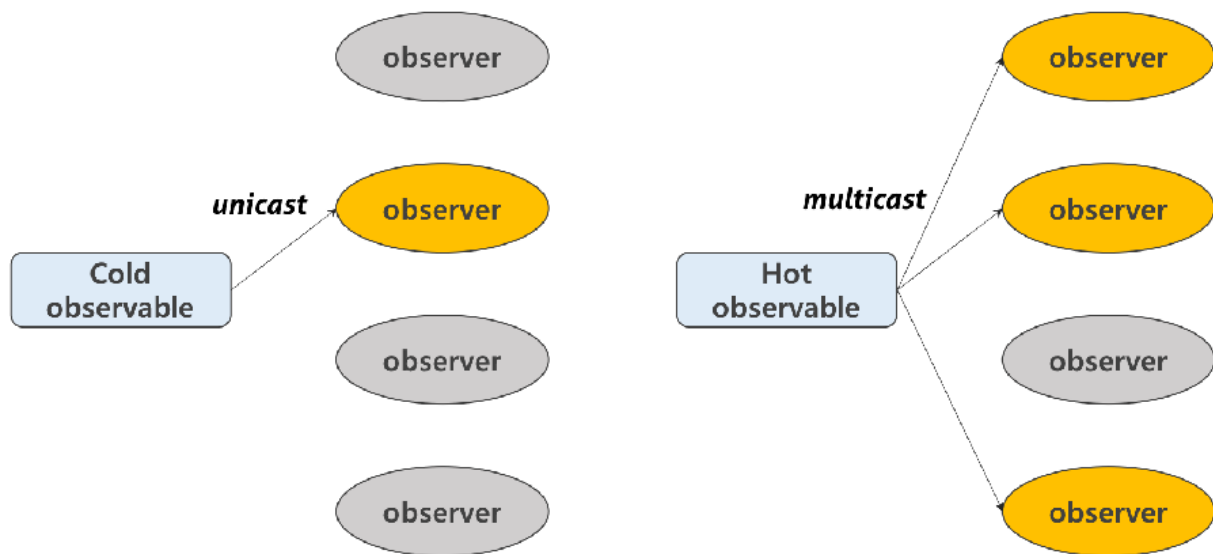
const random = Math.random();

const observable = new Observable(observer => observer.next(random));

observable.subscribe(data => console.log(data)); // Random nummer

observable.subscribe(data => console.log(data)); // Zelfde random nummer
```

Hot vs Cold Observable



H4 HTTP(S)

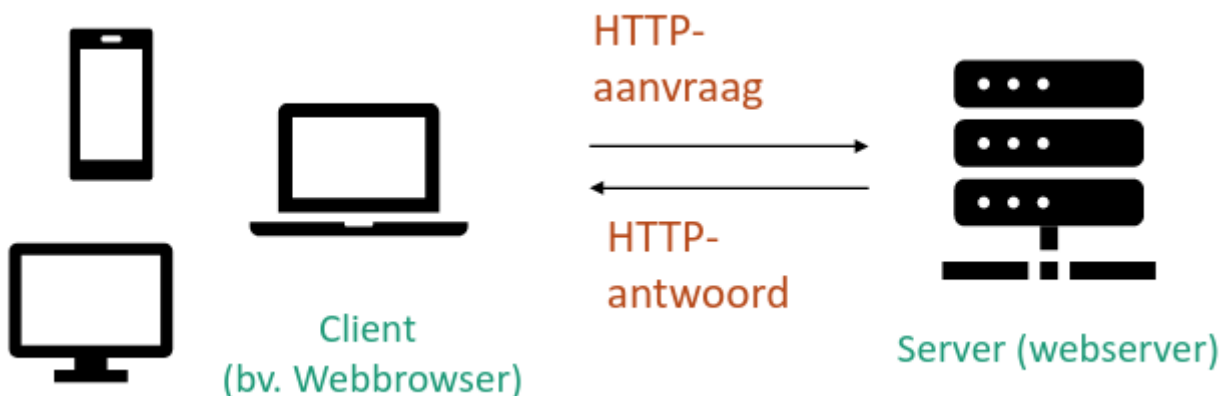
HTTP-protocol

Hypertext Transfer Protocol

Client-Server protocol

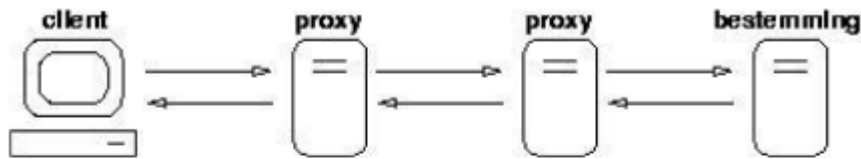
Transportprotocol: TCP op poort 80

Aanvraag-antwoord protocol (request-response)

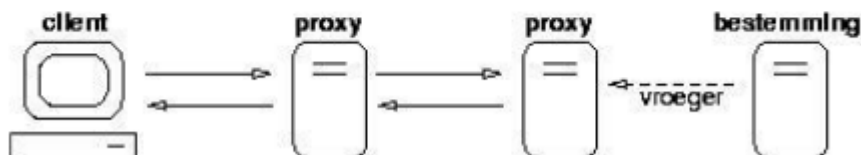


Aanvraag:

- Rechtstreeks naar server met info
- Via tussenliggende HTTP-servers (proxyservers)



➤ Eventueel gecached



Aanvraag

Eenvoudige webpagina: aanvraag

<http://localhost:8080/simpel.html>




Request	
Headers (14) Params (0) Cookies (1) Raw Body	
Key	Value
Host	localhost:8080
Connection	keep-alive
sec-ch-ua	"Chromium";v="88", "Google Chrome";v="88", "Not A Brand";v="99"
sec-ch-ua-mobile	?0
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Sec-Fetch-Site	none
Sec-Fetch-Mode	navigate
Sec-Fetch-User	?1
Sec-Fetch-Dest	document
Accept-Encoding	gzip, deflate, br
Accept-Language	nl-NL,nl;q=0.9,en-US;q=0.8,en;q=0.7
Cookie	Webstorm-33c53794-0bf8e3dd-4361-470f-a82e-9e56013d86a9

```
1 GET http://localhost:8080/simpel.html HTTP/1.1
2 Host: localhost:8080
3 Connection: keep-alive
4 sec-ch-ua: "Chromium";v="88", "Google Chrome";v="88", "Not A Brand";v="99"
5 sec-ch-ua-mobile: ?0
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
8 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: nl-NL,nl;q=0.9,en-US;q=0.8,en;q=0.7
15 Cookie: Webstorm-33c53794-0bf8e3dd-4361-470f-a82e-9e56013d86a9
16
```

Antwoord

Eenvoudige webpagina: antwoord



```
1 HTTP/1.1 200
2 Vary: Origin
3 Vary: Access-Control-Request-Method
4 Vary: Access-Control-Request-Headers
5 Last-Modified: Wed, 10 Feb 2021 13:45:02 GMT
6 Cache-Control: no-store
7 Accept-Ranges: bytes
8 Content-Type: text/html
9 Content-Length: 472
10 Date: Wed, 10 Feb 2021 14:31:39 GMT
11 Keep-Alive: timeout=60
12 Connection: keep-alive
13
14 <!--
15 To change this template, choose Tools | Templates
16 and open the template in the editor.
17 -->
18 <!DOCTYPE html>
19 <html>
20 <head>
21 <title>HTTP Protocol</title>
22 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
23 </head>
24 <body>
25 <h1>HTTP Protocol</h1>
26 Een webpagina zonder figuren.
27 <hr>
28 <address><a href="mailto:Veerle.Ongena@UGent.be">Veerle Ongena</a></address>
29 </body>
30 </html>
31
```



9

Structuur bericht


```
Response
Headers (11)  Cookies (0)  Raw  Preview  Body
1 HTTP/1.1 200
2 Vary: Origin
3 Vary: Access-Control-Request-Method
4 Vary: Access-Control-Request-Headers
5 Last-Modified: Wed, 10 Feb 2021 13:45:02 GMT
6 Cache-Control: no-store
7 Accept-Ranges: bytes
8 Content-Type: text/html
9 Content-Length: 562
10 Date: Wed, 10 Feb 2021 14:55:29 GMT
11 Keep-Alive: timeout=60
12 Connection: keep-alive
13
14 <!--
15 To change this template, choose Tools | Templates
16 and open the template in the editor.
17 -->
18 <!DOCTYPE html>
19 <html>
20   <head>
21     <title>HTML Formulier</title>
22     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
23   </head>
24   <body>
25     <h1>Een formulier</h1>
26     <form action="toonWaarden" method="post">
27       Geef naam: <input type="text" name="naam"><br>
28       Geef adres: <input type="text" name="adres"><br>
29       <input type="submit" value="Verstuur">
30     </form>
31   </body>
32 </html>
33
```

ht

1. Eerste headerlijn: verschillende voor
 - HTTP-aanvraag
 - HTTP-antwoord
2. Volgende headerlijnen
3. Lege lijn
4. Corpus (optioneel)

HTTP-aanvraag

- Eerste headerlijn bestaat uit 3 delen gescheiden door een spatie
 - Aanvraagmethode
 - Aanvraag-URI

- HTTP-versie

Vb: GET <http://gonzo.hogent.be/intranet> HTTP/1.1

Aanvraagmethodes

- GET
 - Enkel headerlijnen, geen corpus
 - Vaak bestand op server
 - Geen veranderingen op server
 - Mag gecached worden
 - URL intikken, link klikker
- HEAD
 - Vraag naar informatie
 - Antwoord bestaat enkel uit headerlijnen
- PUT
 - Bestand op server plaatsen
 - Vb: uploaden eigen webpagina's (i.p.v. FTP)
 - Weinig ondersteund
- POST
 - Laat toe om gegevens aan te passen op server
 - Meestal niet gecached
 - Vaak gebruikt in combinatie met HTML-formulieren
 - Corpus = inhoud formulier

Aanvraag-URL

Uniform Request Identifier

Types

- Absolute URI: vooral aanvragen aan proxy servers
- Absoluut pad: geen protocol en geen servernaam, naam van de server in een aparte Host-headers (host: gonzo.hogent.be)

HTTP-antwoord

- Eerste headerlijn is statuslijn
- Statuslijn
 - Protocolversie
 - Decimale statuscode
 - Kort tekstbericht
- HTTP/1.1 200 OK
- HTTP/1.1 404 Not Found

Statuscodes

- Eerste cijfer geeft type status aan
- 1??: Informatief
- 2??: Succes
- 3??: Omleiding
- 4??: Fout bij client
- 5??: Fout bij server

Vb:

- 200 OK: aanvraag gelukt
- 301 Moved Permanently
- 302 Found / Moved Temporarily
- 303 See Other
 - Aangraag gelukt, maar moet opnieuw (redirect)
 - Location-header bevat nieuwe locatie
 - Gebruiker merkt normaal niets
- 401 Unauthorized: geen toegang
- 403 Forbidden: niet toegankelijk
- 404 Not Found: bestaat niet (meer)
- 501 Not Implemented: aanvraag wordt niet ondersteund door serversoftware

Volgende headerlijnen

- Algemene vorm
 - naam: waarde CRLF (=\r\n)
- Waarde over verschillende lijnen
 - Nieuwe lijn start met spatie of TAB
- Naam: geen onderscheid tussen hoofdletters en kleine letters
- Vb:
 - Accept: text/html, text/plain, image/jpg, image/gif
 - Host: gonzo.hogent.be

Corpus

- Geen interne structuur
- Inhoud vaak gecodeerd m.b.v. MIME (Multipurpose Internet Mail Extensions)
 - Content-type: text/html
- Begin: blanco-lijn
- Einde
 - Content-Length-header: lengte in bytes

- Content-Length: 410
- Chunked transfer coding

Chunked transfer coding

- Headerlijn
 - Transfer-Encoding: chunked
- Corpus verdeeld in verschillende stukken (chunks)
- Chunk
 - Lengte chunk in bytes (hex)
 - CRLF
 - Inhoud
- Laatste chunk heeft lengte 0, daarna volgt een lege lijn

```

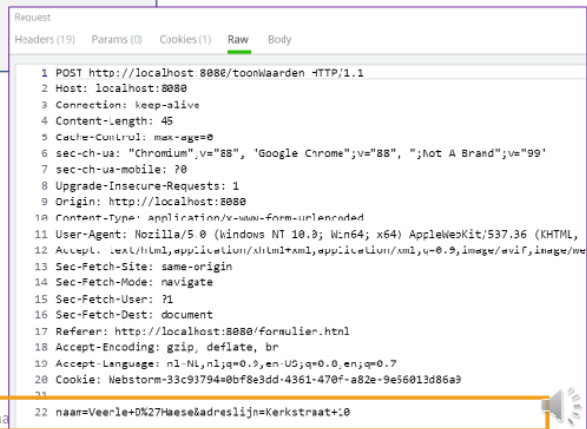
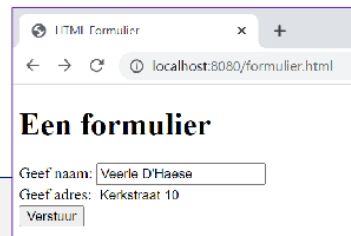
1 HTTP/1.1 200
2 Set-Cookie: nieuw=waarde
3 Content-Type: text/html; charset=UTF-8
4 Content-Language: nl-NL
5 Transfer-Encoding: chunked
6 Date: Wed, 10 Feb 2021 16:12:33 GMT
7 Keep-Alive: timeout=60
8 Connection: keep-alive
9
10 97
11 <!DOCTYPE html>
12 <html lang="en">
13 <head>
14 |   <meta charset="UTF-8">
15 |   <title>Cookie</title>
16 </head>
17 <body>
18 | Pagina met cookie
19 </body>
20 </html>
21 0
22
23

```

Corpus bij formulier

Corpus bij formulier

```
<form action="toonWaarden" method="post">  
  Geef naam: <input type="text" name="naam"><br>  
  Geef adres: <input type="text" name="adreslijn"><br>  
  <input type="submit" value="Verstuur">  
</form>
```



HTML-formulieren

- Tussen <form> en </form>
 - Attributen van de <form>-tag:
 - ACTION: URL "script"
 - METHOD: GET of POST
 - ENCTYPE
 - Bepaalt structuur van corpus (HTTP-bericht)
 - application/x-www-form-urlencoded (standaard)
 - multipart/form-data: maakt bestanden opladen mogelijk

Naam/Waarde-paren

- Informatie van een HTML-formulier (form-tag) doorgeven aan een webapplicatie
 - Parameters
- 2 types (methode-attribuut)
 - GET: querystring in de URL
 - POST: corpus
- Structuur (standaard, NIET bij multipart/form-data):
 - naam1=waarde1&naam2=waarde2&...
- Coding
 - Speciale tekens: % + 2 hexadecimale cijfers (ASCII-code)
 - Spatie: bovenstaande methode of +
 - Vb: value=a b c → value=a+b+c of value=a%20b%20c

Cookies

- HTTP: statusloos
- Oplossing: cookies = klein stukje text
- Bewaard door client
- Teruggestuurd bij elke aanvraag
- In HTTP-headers

Vb

- Bij eerste aanvraag: server stuurt cookie naar client

```
SetCookie: NAAM=WAARDE; Domain=DOMEINNAAM; Path=PAD; Max-Age=SECONDEN
SetCookie: uid=vo; Domain=".rug.ac.be"; Path="/"; Max-Age=86400
```

Cookie: uid=vo

```
1 HTTP/1.1 200
2 Set-Cookie: nieuwswaarde
3 Content-Type: text/html; charset=UTF-8
4 Content-Language: nl-NL
5 Transfer-Encoding: chunked
6 Date: Wed, 10 Feb 2021 16:12:33 GMT
7 Keep-Alive: timeout=60
8 Connection: keep-alive
9
10 97
11 <!DOCTYPE html>
12 <html lang="en">
13 <head>
14   <meta charset="UTF-8">
15   <title>Cookies</title>
```

```
1 GET http://localhost:3000/ HTTP/1.1
2 Host: localhost:3000
3 Connection: keep-alive
4 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7; rv:68.0) Gecko/20100101 Firefox/68.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/svg+xml,*/*;q=0.8
6 Accept-Language: nl-NL;q=0.9,en-US;q=0.8,en;q=0.7
7 Accept-Encoding: gzip, deflate, br
8 Cookie: uid=vo
9
10
11
12
13
14
15
16
```

Virtuele servers

- 1 webserver = HTTP-server voor verschillende instellingen
- Webserver gekend onder verschillende domeinnamen
 - GET / HTTP/1.1
Host: firma1.provider.be
 - GET / HTTP/1.1
Host: firma2.provider.be

Wachtwoorden

- 2 wachtwoordenschema's
 - Basic: niet veilig, wachtwoorden niet gecodeerd, door bijna alle servers en browsers ondersteund
 - Digest: cursus beveiliging

Principie

- Client: aanvraag URI (niet publieke toegang)
- Server: antwoord 401 Unauthorized

```
WWW-Authenticate: Basic realm=TripleEyeIntranet
```

(Wachtwoordenschema en rijik)

- Client vraagt gebruikersnaam en password aan gebruiker (indien eerste keer!)
- Client: nieuwe aanvraag zelfde URI met wachtwoordgegevens

```
Authorization: Basic QKmsdDKJKjkjkhSJDHkjQ==
```

(Gebruikersnaam en wachtwoord: samengevoegd en omgezet in base64 formaat (ook gebruikt door MIME))

- Client kan bij volgende aanvraag onmiddellijk Authorization-header toevoegen bv: bestanden uit zelfde map of subdirectory

Persistente verbindingen

- Vroegere versies HTTP: nieuwe connectie voor elke URI
 - Verbinding tussen client en server openen
 - Client stuurt aanvraag met URI erin
 - Server stuurt antwoord
 - Verbinding wordt afgesloten
- Nadeel
 - Achtereenvolgens verschillende verbindingen met zelfde server
 - Overlast netwerk
- HTTP/1.1: meerdere aanvragen over dezelfde connectie
 - Bij elke aanvraag hoort een afzonderlijk antwoord
 - Verbinding wordt niet telkens afgesloten
 - Client kan verschillende aanvragen na elkaar sturen zonder op antwoord te wachten (pipelining)
- Connection-header
 - Connection: close
 - In aanvraag of antwoord
- Time-out SLIDE BELUISTEREN
- HTTP/2.0: multiplexing - HTTP/3 SLIDE BELUISTEREN

HTTPS

Gevaar bij HTTP: man in the middle attack (afluisteren)

HTTPS: Hypertext Transfer Protocol Secure

Maakt gebruik van SSL (Secure Socket Layer)

SSL

1. Verzoek om een beveiligde verbinding via SSL te initialiseren
2. Weergave en verificatie van het certificaat

- Geldigheid
 - Controle via betrouwbare externe partij
3. Overdracht via unieke encryptie key (versleuteld via public key van de server)
 4. Het decoderen van en encryptie key door de server, met het gebruik van een private key
 5. Een beveiligde verbinding openzetten

H5 Ajax en Fetch API

Ajax

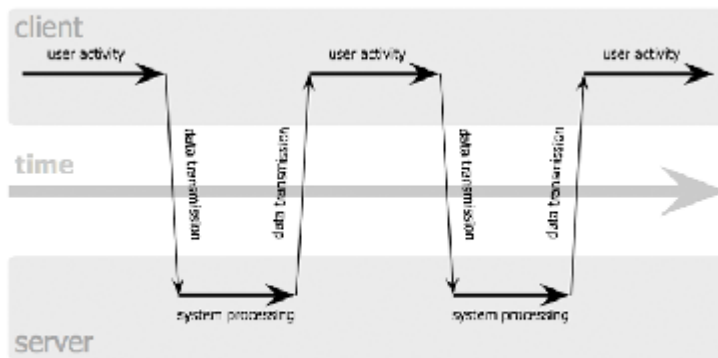
- Site bevriest niet wanneer extra info opgevraagd
- Asynchronous JavaScript and XML
- Bestaant de standaarden anders gebruiken
 - XMLHttpRequest-object
 - Vanuit js HTTP-berichten sturen en ontvangen
 - JS + DOM
 - CSS
 - XML
- Snellere webpagina's
 - Indruk: lokaal werken

Essentie Ajax

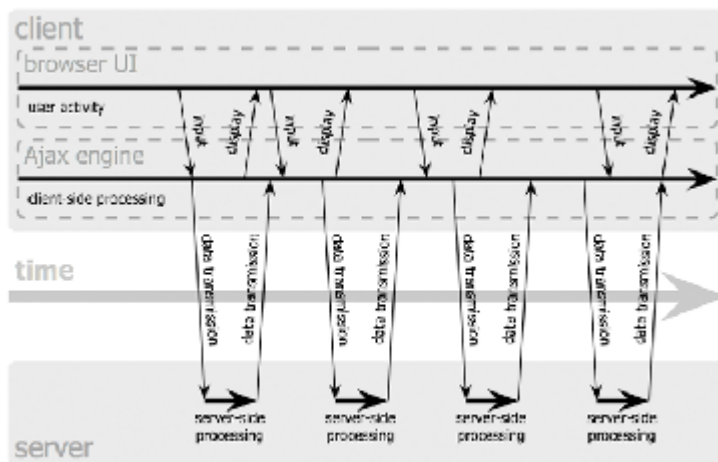
- Klassieke webpagina
 - HTTP-bericht naar server
 - Browser wacht op antwoord
 - Browser toont HTTP-antwoord
 - Nieuw document vervangt volledig vorig document
- Ajax
 - Delen van het HTML-document kunnen vervangen worden

- Gebruiker kan verder werken (asynchrone communicatie met de server)

classic web application model (synchronous)



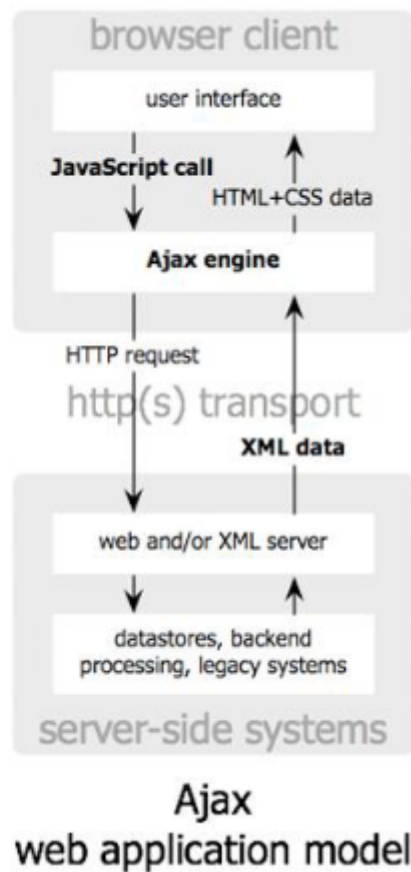
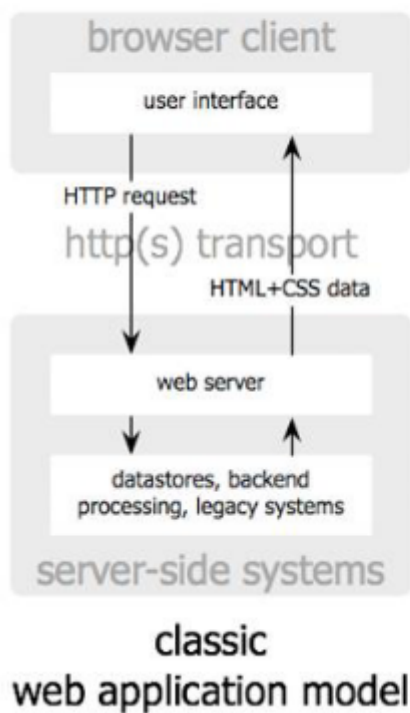
Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

Realisatie Ajax

- Ajax-engine in browser
- HTTP-antwoord
 - XML, JSON, ...



Vb: componist.html en Componist.js

XMLHttpRequest-object

- Communicatie met server
 - Verstuurt HTTP-berichten
 - Ontvang HTTP-berichten
- Oproepen functie indien antwoord ontvangen (call back)
- HTTP-antwoordbericht
 - Inhoud
 - Tekst
 - XML
 - JSON → `JSON.parse(request.responseText)`
 - Omzetting text → JSON-object
 - JavaScript Object Notation

Opstellen HTTP-aanvraag

- Methode open
 - Connectie met server

```
open(aanvraagMethode, URL)
open(aanvraagMethode, URL, asynchroon)
```

```
open(aanvraagMethode, URL, asynchroon, gebruikersnaam)
open(aanvraagMethode, URL, asynchroon, gebruikersnaam, wachtwoord)
```

- aanvraagMethode: GET, POST, ...
- URL: bron met data
- asynchroon
 - Default: true
 - Er wordt niet gewacht op antwoord van de server

Callback-functie instellen

- onreadystatechange
 - Uit te voeren functie bij verandering "readystatechange"
- readyState
 - 0: request not initialized
 - 1: server connection established
 - 2: request received
 - 3: processing request
 - 4: request finished and response is ready

Aanvraag versturen naar server

- send(corpus)
 - Corpus
 - inhoud HTTP-bericht
 - Parameters
 - null
- Na instellen callback-functie

Antwoordbericht - eigenschappen

- status
 - HTTP-antwoord-status
- responseText
 - Corpus antwoord
 - string
- Alternatieven (niet van aantrekken)
 - responseXML → document

Fetch API

- API om bronnen op te halen in een client script
- Vergelijkbaar met XMLHttpRequest

- Gebruikt promises

Principe

```
fetch(
  'http://domain/service', // URL
  {methode: 'GET'}          // Opties, JS-object: headers, method,
  ...
)
.then(response => response.json()) // Antwoord
.then(json => console.log(json))
.catch(error => console.error('error:', error));
```

Parameters doorsturen naar de server

- HTTP-parameters: informatie van een HTML-formulier (form-tag) doorsturen naar de server (webapp, REST-service)
- Antwoord afh.v. doorgestuurde data
- Een aantal opties
 - GET: querystring in de URL
 - POST: in body
 - naam1=waarde1&naam2=waarde2&... = zoals in formulier
 - {"naam1":"waarde1","naam2":"waarde2",...} = JSON-object
- Antwoord
 - Text
 - JSON

Crossdomain calls

- Ajax-aanvragen naar ander domein dan huidige webpagina
 - Webpagina
 - <http://localhost:8080/index.html>
 - Ajax-aanvraag
 - <http://localhost:50633/ComponistService.svc/GetComponisten?letters=ba>

⚠ Cross origin aanvraag geblokkeerd: de Same Origin Policy staat het lezen van de externe bron op <http://localhost:50633/ComponistService.svc/GetComponisten?letters=ba> niet toe. Dit is te verhelpen door de bron naar hetzelfde domein te verplaatsen of door CORS in te schakelen.

- Mogelijke oplossing
 - Server side proxy
 - Webpagina → server
 - Server → ander domein
 - Ander domein → server
 - Server → webpagina
 - CORS
 - Eventueel JSONP

Same origin policy

Zie slide 33-34

Cross-origin resource sharing (CORS)

- Webpagina
 - Aanvraag naar bronnen van een ander domein
 - Niet toegelaten
- CORS
 - Server kan toelating geven om geraadpleegd te worden door een ander domein
 - Nieuwe HTTP-headers

browser

Origin: http://www.example-social-network.com

server

Access-Control-Allow-Origin: http://www.example-social-network.com

JSONP

- JSON with padding
- Browsers laten aanvraag naar ander domein wel toe voor <script>-tag
 - src-attribuut → resultaat URL = functie → functie uitgevoerd
- Opmerking: beveiligingsrisico's: malafide services of "gehackte" services

H6 Angular

Typescript

- Typering aan JS toegevoegd
- Compileerbaar naar JS: tsc bestand.ts → bestand.js

```
function greeter(person: string) {  
    return "Hello, " + persoon;  
}
```

Typescript interface

```
interface Person {  
    firstName: string;  
    lastName: string;  
}  
  
function greeter(person: Person) {
```

```

        return "Hello, " + person.firstName + " " + person.lastName;
    }

    let usr = { firstName: "Jane", lastName: "User" };
    let text = greeter(usr);

```

Benodigheden

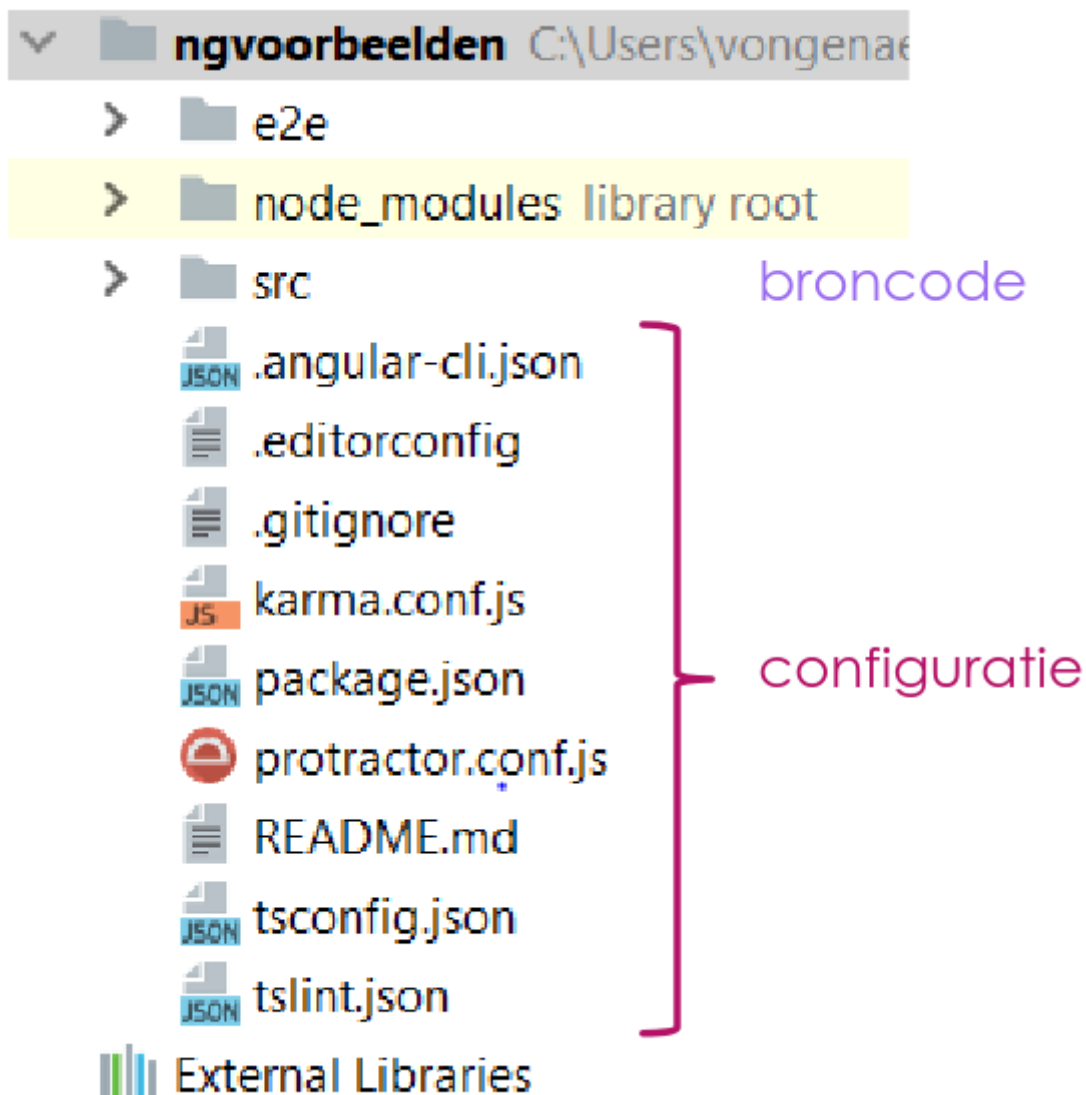
- Node.js
 - Server side JS framework
 - Gebouwd op JS runtime v. Chrome
 - NPM
 - Node Package Manager
 - JS packages installeren
 - Handmatig
 - Op basis van configfile package.json

```

{
    "name" : "NyApp",
    "version" : "1.0.0",
    "dependencies" : {
        "sax" : "0.3.x",
        "nano" : "x",    // Gelijk welke versie
        "request" : ">0.2.0"
    }
}

```

- Angular CLI
 - Command line interface for Angular
 - Functionaliteit: <https://github.com/angular/angular-cli>
 - Start project genereren
 - Webserver starten om app te testen: ng server
 - Angular componenten, ... aanmaken: ng generate component my-component
 - ...
 - Installeren: npm install -g @angular/cli



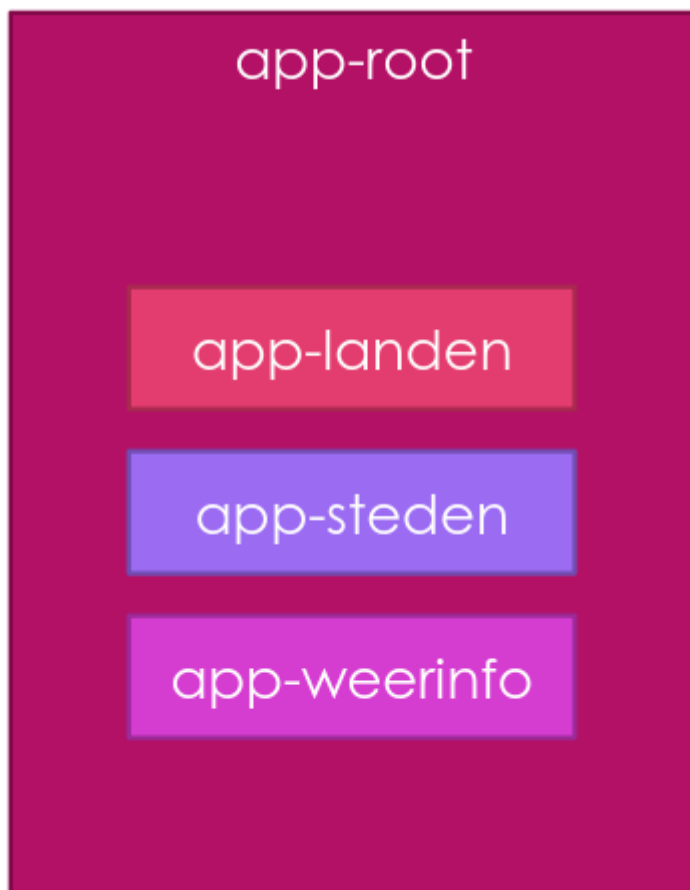
Componenten

Nieuwe tags die angular gebruikt om de website op te bouwen
index.html

```
<app-root></app-root>
```

app.component.html

```
...  
<app-landen ...></app-landen>  
<app-steden ...></app-steden>  
<app-weerinfo ...></app-weerinfo>  
...
```



- Bouwsteen voor webpagina
- Bestaat uit
 - Klasse: functionaliteit
 - HTML-template (sjabloon): presentatie
 - Stylesheet: opmaak

Aanmaak component

```
ng generate component my-component  
ng g component my-component
```

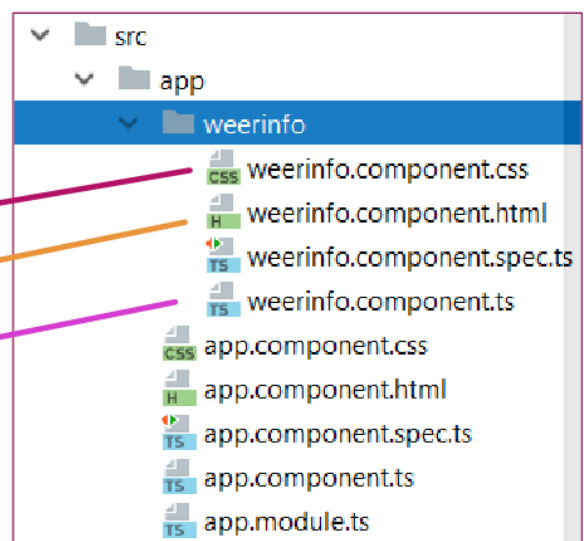
```
ng generate component my-component
```

```
ng g component my-component
```

stylesheet

template

klasse



Interpolation binding

HTML-template

```
<div class="weer">
  <div> {{datum}} </div>
  ...
</div>
```

Klasse: attributen - methodes

```
@Component({
  selector: 'app-weerinfo',
  templateUrl: '....html',
  styleUrls: ['....css']
})
export class WeerinfoComponent {
  datum = '26/10/17';
  ...
}
```

Klasse component

```
import {Component} from '@angular/core';
@Component({
  selector: 'app-weerinfo',
  templateUrl: './weerinfo.component.html',
  styleUrls: ['./weerinfo.component.css']
})
export class WeerinfoComponent {
  datum = '26/10/17';
  temperatuur = 16;
  vochtigheid = 90;
  bewolkt = 80;
  omschrijving = 'bewolkt';
  afbeelding = 'nen url'
  constructor() {}
}
```

Template component

```
<div class="weer">
  <div> {{datum|date:'MM/dd/yyyy'}} </div>
  <div> {{temperatuur|number:'1.0-2'}}°C </div>
  <div> vochtigheid {{vochtigheid}}% </div>
  <div> wolken {{bewolkt}}% </div>
```

```
<div>  </div>
</div>
```

Interpolation binding → data tonen uit component

Stylesheet component

```
.weer {
  text-align: center;
  border: solid;
  border-color: grainsboro;
  border-width: 1px;
  border-radius: 5%;
}
```

Componenten gebruiken

app.component.html

```
<div class="container">
...
  <div class="row">
    <div class="col-sm">
      ...
    </div>
    <app-weerinfo class="col-sm-4">
    </app-weerinfo>
  </div>
</div>
```

Component registreren

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core'

import { AppComponent } from './app.component';
import { WeerinfoComponent } from './weerinfo/weerinfo.component';

@NgModule({
  declarations: [AppComponent, WeerinfoComponent], // componenten
  imports: [BrowserModule], // nodige functionaliteit
  providers: [],
  bootstrap: [AppComponent] // root component
})
export class AppModule { }
```

index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Ngvoorbeelden</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="https://.../bootstrap.min.css">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Structural directives *ngFor en *ngIf : klasse landencomponent

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-landen',
  templateUrl: './landen.component.html',
  styleUrls: ['./landen.component.css']
})

export class LandenComponent implements OnInit {
  landen: string[];
  constructor() { }

  noOnInit() {
    this.landen = ['Nederland', 'België'];
  }
}
```

<app-landen ...></app-landen>

Template landencomponent

landen.component.html

```
<select
  *ngIf="landen !== undefined && landen.length !== 0; else
geenLanden">
  <option *ngFor="let land of landen">{{land}}</option>
</select>
```

```
<ng-template #geenLanden>
  <div>Geen landen gevonden</div>
</ng-template>
```

OF

```
<div *ngIf="landen !== undefined && landen.lenth != 0;
  then wellanden else geenLanden"></div>
<ng-template #wellanden>
  <select>
    <option *ngFor="let land of landen">{{land}}</option>
  </select>
</ng-template>
<ng-template #geenLanden>
  <div>Geen landen gevonden</div>
</ng-template>
```

One way property binding

One way binding

HTML-template

```
<img [src]="afbeelding">
```

Klasse: attributen - methodes

```
@Component({
  selector: 'app-weerinfo',
  templateUrl: '....html',
  styleUrls: ['....css']
})
export class WeerinfoComponent {
  afbeelding = '...';
  ...
}
```

Attributen tags ← Attributen klasse

Property binding

```
export class WeerinfoComponent {
  afbeelding = '...';
}
```

3 mogelijkheden

1. ``
2. ``
3. ``

Vb:

app.component.html

```
<button class="btn" [disabled]="nietgevonden">
  Toon weer
</button>
```

app.component.ts

```
...
export class AppComponent {
  nietgevonden = false;
  ...
}
```

Two-way binding

- [(ngModel)]
 - Two-way binding
 - Property/Field ↔ waarde HTML-component
- Behoort tot module FormsModule
 - Importeren

```
import { FormsModule } from '@angular/forms';
...
@NgModule({
  declarations: [AppComponent, WeerinfoComponent, LandenComponent,
    StedenComponent],
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
```

Vb:

landen.component.ts

```
export class LandenComponent implements OnInit {
  selectLand: land | undefined;
  private _landen: Land[];
  get landen(): Land[] {return this._landen;}
}
```

landen.component.html

```
<select class="custom-select" [(ngModel)]="selectedLand"
(change)="verandenLand()">
  <option *ngFor="let land of landen" [ngValue]="land">{{land.naam}}
</option>
</select>
```

[ngValue]: waarde voor optie → inhoud selectedLand

Two-way binding

HTML-template

```
<select [(ngModel)]="selectedLand" ...>
  <option
    *ngFor="let land of landen"
    [ngValue]="land">
    {{land.naam}}
  </option>
</select>
```

Klasse: attributen - methodes

```
export class LandenComponent {
  selectedLand: Land;

  private _landen: Land[];

  get landen(): Land[] {
    return this._landen;
  }
}
```

Attributen tags ↔ Attributen klasse

Events

HTML-template

```
<select (change)="veranderdLand()" ...>
...
</select>
```

Klasse

```

expose class LandenComponent {
    veranderdLand() {
        ...
    }
}

```

app.component.html

```

<button class="btn" (click)="toonClick($event)">
  Toon weer
</button>

```

(click) = event, toonClick = handler
app.component.ts

```

...
toonClick(event) {
    this.tekst = event.toString();
}

```

landen.component.html

```

<select class="custom-select" ... (change)="veranderdLand()">
  ...
</select>

OF

<li *ngFor="let hero of heroes" (click)="onSelect(hero)">
  ...
</li>

```

Data direction	Syntax	Type
One-way from data source to view target	<pre> {{expression}} [target]="expression" bind-target="expression" </pre>	Interpolation Property Attribute Class Style
One-way from view target to data source	<pre> (target)="statement" on-target="statement" </pre>	Event
Two-way	<pre> [(target)]="expression" bindon-target="expression" </pre>	Two-way

<https://angular.io/guide/template-syntax#binding-syntax-an-overview>

Attributen toevoegen: @Input()

```
<app-landen landen="..."></app-landen>
```

```

export class LandenComponent implements OnChanges {
  @Input()
  landen: Land[] = [];

  ngOnChanges(changes: SimpleChanges): void {
    if(this.landens !== [] && this.selectedLand == undefined) {
      this.selectedLand = this.landens[0];
    }
  }
}

```

app.component.html

```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

app.component.ts


```
export class AppComponent {
    landen: Land[];
}
```

landen.component.ts

```
export class LandenComponent implements OnInit {
    private _landen: Land[];
    @Input()
    landen: Land[] = [];
}
```

Events toevoegen: @Output()

Tag - component

```
<app-landen (landChanged)="..."></app-landen>
```

Klasse - component

```
export class LandenComponent {
    selectedLand: Land;

    @Output() landChanged = new EventEmitter<Land>();
    veranderdLand() {
        this.landChanged.emit(this.selectedLand);
    }
}
```

Zelf events definiëren

- Event Emitter in componentklasse
 - Declareren
 - Event uitsenden
- Registreren voor event
- Component WAAR EVENT OPTREEDT: Event voorzien
- Component DIE NAAR EVENT LUISTERT: Handler voorzien

Vb:

Landen: event toevoegen

landen.component.html

```
<select [(ngModel)]="selectedLand" (change)="veranderdLand()">
    <option *ngFor="let land of landen" [ngValue]="land">
        {{land.naam}}
```

```
        </option>
    </select>
```

landen.component.ts

```
import { ..., Output, EventEmitter } from '@angular/core';
export class LandenComponent implements OnInit {
    selectedLand: Land;
    @Output() landChanged = new EventEmitter<Land>();
    veranderdLand() {
        this.landChanged.emit(this.selectedLand);
    }
}
```

Landen: handler toevoegen

app.component.html

```
<app-landen [landen]="landen" (landChanged)="veranderLand($event)">
</app-landen>
```

Landen: handler

app.component.ts

```
export class AppComponent {
    steden: string[];
    land: Land;
    constructor(private landenService: LandenService) {
        ...
    }

    veranderLand(land: Land) {
        this.land = land;
        this.landenservice.haalSteden(land).then(steden =>
this.steden = steden);
    }
}
```

Oudercomponenten

???

Services

- Biedt diensten aan
 - Data ophalen
 - Logging
 - Berekeningen

- ...
- Wordt automatisch geïnjecteerd in componenten
- Kan ook asynchroon

Overzicht

Klass component

```
export class AppComponent
{
    constructor(private landenService: LandenService) {}
    ...
}
```

Klass service: kan geïnjecteerd worden in andere klassen

```
@Injectable()
export class LandenService {
    haalLanden(): Land[] { ... }
}
```

Service aanmaken en registreren

app.module.ts

```
import { LandenService } from './landen.service';
@NgModule({
    declarations: [AppComponent, WeerinfoComponent, LandenComponent,
StedenComponent],
    imports: [BrowserModule, FormsModule],
    providers: [LandenService],
    bootstrap: [AppComponent]
})
```

landen.service.ts: kan geïnjecteerd worden in andere klassen

```
import { Injectable } from '@angular/core';
@Injectable()
export class LandenService {
}
```

Service implementeren

landen.service.ts

```

import { Injectable } from '@angular/core';
import { Land } from './land';

@Injectable()
export class LandenService {
  haalLanden(): Land[] {
    let landen =
      [{code: 'BE', naam: 'België', hoofdstad: 'Brussel'},
       {code: 'NL', naam: 'Nederland', hoofdstad: 'Den Haag'}];
    return landen;
  }
}

```

Service injecteren en gebruiken

app.component.ts

```

import { LandenService } from '/landen.service';
import { Land } from './land';

export class AppComponent implements OnInit {
  landen: Land[];

  constructor(private landenService: LandenService) { }

  ngOnInit(): void {
    this.landens = this.landensService.haalLanden();
  }
}

```

Lifecycle Hooks

- Angular beheert levensloop component
- Hooks
 - Acties na uitvoeren van een bepaalde fase

constructor

ngOnChanges

ngOnInit

ngDoCheck

ngAfterContentInit

ngAfterContentChecked

ngAfterViewInit

ngAfterViewChecked

ngOnDestroy

<https://angular.io/guide/lifecycle-hooks>

Asynchrone service

landen.service.ts

```
import { Injectable } from '@angular/core';
import { Land } from './land';

@Injectable()
export class LandenService {
  haalLanden(): Promise<Land[]> {
    let landen =
      [{code: 'BE', naam: 'België', hoofdstad: 'Brussel'},
       {code: 'NL', naam: 'Nederland', hoofdstad: 'Den Haag'}];
    return Promise.resolve(landen);
  }
}
```

Asynchrone service gebruiken

app.component.ts

```
import { LandenService } from './landen.service';
import { Land } from './land';

export class AppComponent implements OnInit {
  landen: Land[];

  constructor(private landenService: LandenService) { }

  ngOnInit(): void {
    this landenService.haalLanden().then(landen => this landen
= landen);
  }
}
```

HTTP-services

- Data van een server halen
 - AJAX-call
- Gebruikt HttpClientModule

Klass service

```
@Injectable()
export class LandenService {
  constructor(private http: HttpClient) { }
}
```

Gebruik HTTP-module

app.module.ts

```
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [AppComponent, WeerinfoComponent, LandenComponent,
StedenComponent],
  imports: [BrowserModule, FormsModule, HttpClientModule],
  providers: [LandenService],
  bootstrap: [AppComponent]
})
```

HTTP-service

landen.service.ts

```
import { HttpClient } from '@angular/common/http';
import { RestCountry } from './rest-country';
```

```
@Injectable()
export class LandenService {
  constructor(private http: HttpClient) {
  }
}
```

Interface REST-resultaat

rest-country.ts

```
export interface RestCountry {
  alpha2Code: string;
  name: string;
  capital: string;
}
```

HTTP-service: landen ophalen

landen.service.ts

```
get landen(): Observable<Land[]> {
  return this.http.get<RestCountry[]>(this.landenURL)
    .pipe(
      tap(_ => console.log('fetched landen')),
      map(items => items.map (
        item => new Land(item.alpha2Code, item.name,
item.capital)))
    );
}
```

GET

- http.get → Observable (meerdere asynchrone resultaten)
- RestCountry[] → Land[]

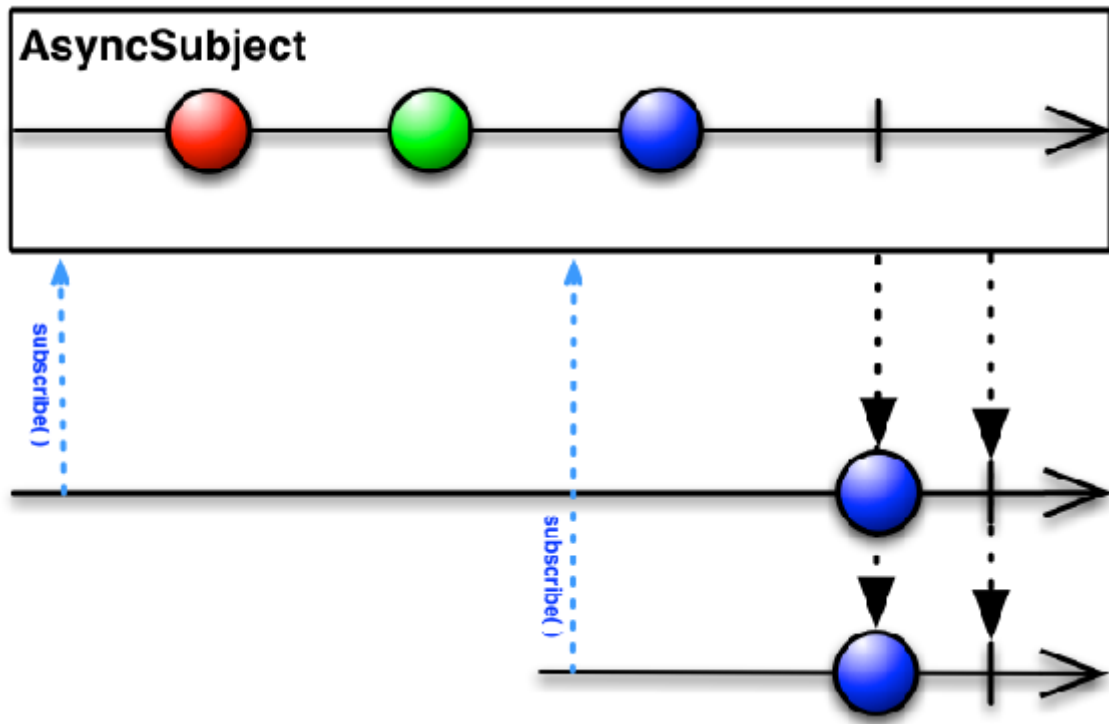
Resultaat methode-oproep

- 1 resultaat: T (synchroon) of Promise<T> (asynchroon)
- veel resultaten: T[] (synchroon) of Observable<T> (asynchroon)

Observable, ZOEKEN OP YT

- Events

- Asynchrone methodes met meerdere resultaten



HTTP-service: landen ophalen

landen.service.ts

```
get landen(): Observable<Land[]> {
  return this.http.get<RestCountry[]>(this.landenURL)
    .pipe(
      tap(_ => console.log('fetched landen')),
      map(items => items.map(
        item => new Land(item.alpha2Code, item.name,
          item.capital)))
    );
}
```

gebruik maken van route parameter

```
constructor(
  private route: ActivatedRoute,
  private location: Location) { }

ngOnInit(): void {
  this.onderwerpId = this.route.snapshot.paramMap.get('onderwerp');
}
```