



Web advanced

Promises (async, await)
fetch, axios

**DE HOGESCHOOL
MET HET NETWERK**

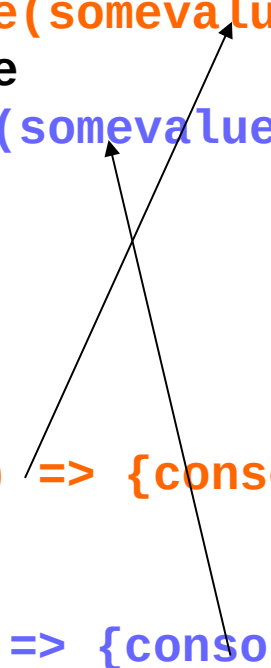
Hogeschool PXL - Dep. PXL-IT - Elfde-Liniestraat 26 - B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Promises

```
let promise = new Promise( (resolve, reject) => {  
  // do something asynchronous  
  // if success  
  //   resolve(somevalue);  
  // otherwise  
  //   reject(somevalue);  
} );
```

```
promise  
  .then(  
    (result) => {console.log(result); }  
  )  
  .catch(  
    (error) => {console.log(error); }  
  )
```



```

function factorial(number) {
  let promise = new Promise((resolve, reject)=>{
    setTimeout( () => {
      if(!Number.isInteger(number)){
        reject(`${number} is not a number`);
        return;
      }
      let result=1;
      for (let i=1; i<=number;i++){
        result=result*i;
        if (result==Infinity){
          break;
        }
      }
      if (result==Infinity){
        reject( `${number}! is Infinity` );
        return;
      }
      resolve( result );
    }, Math.floor(Math.random() * 1000) );
  });
  return promise;
}

factorial(20)
  .then( (result) => {console.log('resolved: ', result);} )
  .catch( (error) => {console.log('rejected: ', error);} );

```

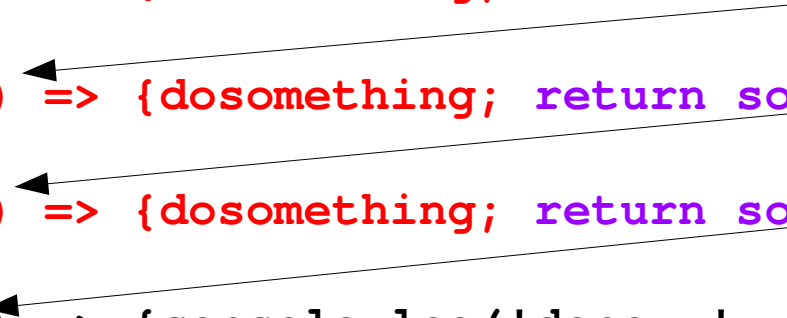


Chaining

Meerdere .then's naast elkaar.

`promise`

```
.then( (result) => {dosomething; return somevalue; } )  
.then( (result) => {dosomething; return somevalue; } )  
.then( (result) => {dosomething; return somevalue; } )  
.then( (result) => {console.log('done: ', result);} )
```



Chaining

Meerdere `.then`'s naast elkaar. **Optie 1**: het resultaat van een promise wordt gechained aan een niet-asynchrone actie.

In het voorbeeld is `factorial` de promise. Na de promise wordt verdubbeld en afgedrukt (dit zijn geen promises).

```
factorial(100)
  .then( (result) => result*2 )
  .then( (result) => {console.log('resolved: ', result);} )
  .catch( (error) => {console.log('rejected: ', error);} );
```

Chaining

Meerdere .then's naast elkaar. **Optie 2**: het resultaat van een promise wordt gechained aan een promise

```
function double(number){
  let promise = new Promise((resolve,reject)=>{
    setTimeout(()=>{
      if(!Number.isInteger(number)){
        reject(`${number} is not a number`);
        return;
      }
      let result = 2 * number;
      if (result == Infinity){
        reject(`${number} is not a number`);
        return;
      }
      resolve(result);
    }, Math.floor(Math.random() * 1000) );
  });
  return promise;
}

factorial(20)
  .then( (result) => double(result) )
  .then( (result) => {console.log('resolved: ', result);} )
  .catch( (error) => {console.log('rejected: ', error);} );
```

async / await

async:

- enkel binnen een async function mag await gebruikt worden
- de function geeft een **altijd** Promise terug
als de function geen Promise teruggeeft dan wordt de teruggegeven waarde in een Promise gestoken die de teruggegeven waarde resolve't

```
async function f(){  
  return 1;  
} → async function f(){  
  return new Promise((resolve)=>{resolve(1);});  
}
```

```
f().then((result)=>{console.log(result);});
```

await:

Wacht op het resultaat van een promise
Kan enkel in een async function gebruikt worden (!!)
Veel beter leesbare code!

Zonder async / await

```
const fetch = require('node-fetch');
```

```
function fetchPokemonName(id) {  
  let promise = new Promise((resolve, reject) => {  
    fetch(`https://pokeapi.co/api/v2/pokemon/${id}`)  
      .then((response) => {  
        if (response.ok) {  
          return response.json();  
        } else {  
          reject(`error status code ${response.status}`);  
          return;  
        }  
      })  
      .then((pokemon) => { resolve(pokemon.name); })  
      .catch((error) => { reject(error); })  
  });  
  return promise;  
}
```

```
for (let i = 0; i < 10; i++) {  
  fetchPokemonName(i)  
    .then((name) => { console.log(name); })  
    .catch((error) => { console.log(error); })  
}
```



Zonder async / await

```
$ ls
app.js
$ npm i node-fetch@2.6

added 1 package, and audited 2 packages in 552ms

found 0 vulnerabilities
$ ls
app.js  node_modules  package.json  package-lock.json
$ node app.js
error status code 404
ivysaur
bulbasaur
venusaur
charmander
charizard
blastoise
squirtle
charmeleon
wartortle
```

Met async / await

```
const fetch = require('node-fetch');
```

```
async function fetchPokemonName(id){  
  let response = await fetch(`https://pokeapi.co/api/v2/pokemon/${id}`);  
  let ok = response.ok;  
  if(!ok){  
    let status = response.status  
    throw new Error(`error status code ${status}`);  
  }  
  let pokemon = await response.json();  
  return pokemon.name;  
}
```

```
for (let i = 0; i < 10; i++){  
  fetchPokemonName(i)  
    .then((name)=>{console.log(name)})  
    .catch((error)=>{console.log(error.message)})  
}
```

*Niet in async function dus
hier geen await wel then*

Axios ipv fetch

```
const axios = require('axios').default;
```

```
async function fetchPokemonName(id) {  
  const response = await axios.get(`https://pokeapi.co/api/v2/pokemon/${id}`);  
  const pokemon = response.data;  
  return pokemon.name;  
}
```

```
for (let i = 0; i < 10; i++) {  
  fetchPokemonName(i)  
    .then((name) => { console.log(name) })  
    .catch((error) => { console.log(error.message) })  
}
```



Axios ipv fetch

```
$ npm i axios
added 2 packages, and audited 3 packages in 765ms

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
$ node app.js
charmander
bulbasaur
ivysaur
venusaur
squirtle
charizard
blastoise
wartortle
charmeleon
Request failed with status code 404
```

Besluit

Nieuw:

async / await
axios