Web Development

JavaScript – Labo 8



Werken met datums

- *Date* objecten
- Voorstelling van één enkel moment in de tijd in een platformonafhankelijk formaat
- Niet meer en niet minder dan een Number
 - Aantal milliseconden sinds de ECMAScript epoch (1 / 1 / 1970)



- Vijf basisvormen
 - Geen parameters
 - Waarde of Timestamp
 - Timstamp string
 - Date object
 - Individuele componenten

```
let d = new Date();

// Wed Apr 27 2022
    09:42:04 GMT+0200
    (Midden-Europese zomertijd)
```



- Vijf basisvormen
 - Geen parameters
 - Waarde of Timestamp
 - Timstamp string
 - Date object
 - Individuele componenten

```
let d = new Date(609120000000);
```



- Vijf basisvormen
 - Geen parameters
 - Waarde of Timestamp
 - Timstamp string
 - Date object
 - Individuele componenten

```
let d =
  new Date('2022-04-27 23:59:59');
```



- Vijf basisvormen
 - Geen parameters
 - Waarde of Timestamp
 - Timstamp string
 - Date object
 - Individuele componenten

```
let d = new Date();
let d2 = new Date(d);
```



[Jan, Feb,... Dec] = [0, 1, ... 11]

- Vijf basisvormen
 - Geen parameters
 - Waarde of Timestamp
 - Timstamp string
 - Date object

```
new Date(2022, /3)
                   new Date(2022, 3, 27)
                   new Date(2022, 3, 27, 23)
                   new Date(2022, 3, 27, 23, 59)
• Individuele componenten new Date(2022, 3/27, 23, 59, 59)
                   new Date(2022(13) 27, 23, 59, 59, 30)
                         // yyyy,
                                  m, d, u,
                                              m,
```

> 11 → naar volgend jaar new Date(2022, 13) wordt Februari 2023



toString



- toString
- toISOString

```
let d = new Date();
d.toISOString();
// '2022-04-27T08:25:29.080
Z'
```



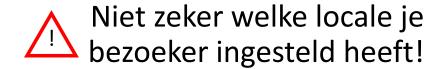
- toString
- tolSOString
- toDateString

```
let d = new Date();
d.toDateString();
// 'Wed Apr 27 2022'
```



- toString
- tolSOString
- toDateString
- toTimeString





- toString
- toISOString
- toDateString
- toTimeString
- toLocaleString
- toLocaleDateString
- toLocaleTimeString

```
let d = new Date();
d.toLocaleString('nl-be')
// '24/4/2022 10:32:52'
d.toLocaleDateString('nl');
// '24-4-2022'
d.toLocaleTimeString()
// '10:32:52'
```



Datum weergeven in eigen formaat?

Volledig zelf op te bouwen

```
let d = new Date();

console.log(
    d.getDate() + "-" + (d.getMonth() + 1)
    + "-" + d.getFullYear() + " " + d.getHours()
    + ":" + d.getMinutes() + ":" + d.getSeconds()
);
```



Objectgeoriënteerd programmeren

- Objecten in JavaScript
- Eenvoudige variabelen
 - Initialiseren met accolades ({})
 - Properties
 - Kunnen eender welk type zijn (ook andere objecten!)
 - Kunnen vooraf gedefinieerd zijn
 - Kunnen ook "at runtime" toegevoegd worden

```
let schilder = {};

schilder.naam = 'Van Gogh';
schilder.aantalOren = 1;
schilder.werken = [ /* ... */ ];
schilder.geboorteplaats = {
  land: 'Nederland',
  gemeente: 'Zundert'
}
```



Properties aanspreken

- Properties aanspreken
 - Via .property
 - Via ['property']

```
console.log(
   `${schilder.naam} werd geboren in
   ${schilder.geboorteplaats.gemeente}`);

console.log(
   `${schilder['naam']} werd geboren in
   ${schilder['geboorteplaats']['gemeente']}`);
```



Properties aanspreken

- .property of ['property']?
 - Aan jullie de keuze
 - .property is tegenwoordig wel het meest verkozen
 - Minder / makkelijker typen
 - Gelijkaardig aan andere (moderne) programmeertalen
 - ['property'] is well handig om een variabele eigenschap op te halen in code

```
let toonEnkelLand = false;
let prop;

if (toonEnkelLand) {
   prop = 'land';
} else {
   prop = 'gemeente';
}

console.log(schilder.geboorteplaats[prop]);
```



Object literals

```
let schilder = {};
schilder.geboorteplaats = {
   land: 'Nederland',
   gemeente: 'Zundert'
}
```



JSON

- JavaScript Object Notation
- Tekstuele voorstelling van objecten
 - Niets meer dan een string van een object literal
- Gebruikt om gegevens uit te wisselen, bvb.
 - Tussen webapplicatie en server
 - Export / import tussen twee systemen
- Gebruikt om gegevens te bewaren
 - In cookies
 - In localStorage



Waarom JSON?

- Meer en meer de vervanger van zijn voorganger XML
- Voordelen t.o.v. XML
 - Leesbaarder
 - Makkelijker op te stellen
 - Gewoon een object literal, dus door iedere JS'er gekend



JSON parsen

• Eenvoudig om JSON/object (literal) om te zetten naar een string

```
let tekst = JSON.stringify(schilder);
```

Eenvoudig een JSON-string omzetten naar JSON/object (literal)

```
let schilder = JSON.parse(tekst);
```

