Web Development

JavaScript – Labo 7

Timers

- Twee soorten
  - Herhaaldelijk, om de x aantal (milli)seconden bepaalde logica uitvoeren
    - `setInterval()`
  - X aantal (milli)seconden wachten om iets eenmaligs uit te voeren
    - `setTimeout()`

setInterval

```
const doeIets = () => { /* code weggelaten */ }

let taakId = setInterval(doeIets, 5000);
```

ID van de taak

Uit te voeren functie

(callback)

Interval in ms

clearInterval

- Stopzetten van een intervalfunctie

```
clearInterval(taakId);
```

setTimeout

```
const doeIets = () => { /* code weggelaten */ }

let taakId = setTimeout(doeIets, 5000);
```

ID van de taak

Uit te voeren functie (callback)

Aantal ms dat gewacht wordt

clearTimeout

- Annuleren van een geplande timeout

```
clearTimeout(taakId);
```

setInterval(f, 5000)

- Altijd uitgevoerd iedere 5 seconden?
  - **Nee!**
  - Browser heeft maar één thread om JavaScript uit te voeren
  - Er gebeurt dus niets op hetzelfde moment
  - M.a.w. de timer kan pas uitgevoerd worden als er niets anders bezig is

- Gevolgen
  - **setInterval**: als callbackfunctie langer duurt dan de opgegeven periode -> programma “raakt achterop”
  - **setTimeout**: als er na de oproep nog iets loopt dat langer duurt dan de periode -> logica wordt later uitgevoerd

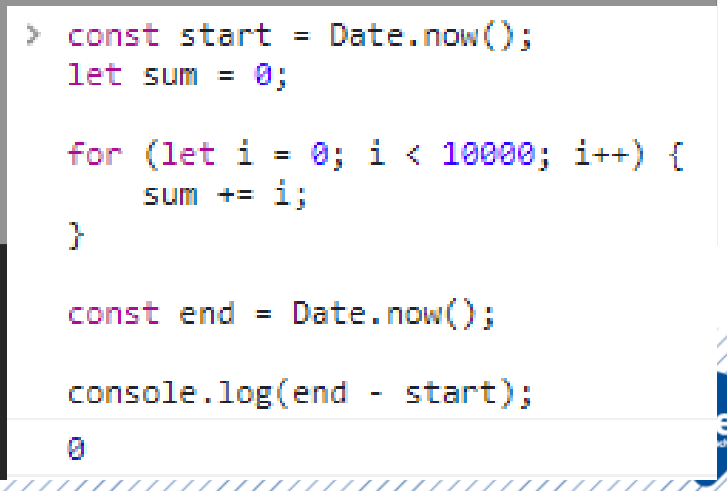
Timen van code – labodocument

```
const start = Date.now();

let sum = 0;

for (let i = 0; i < 10000; i++) {
    sum += i;
}

const end = Date.now();
console.log(end - start);
```



Timen van code – verbeterde versie

```
console.time('naam');

let sum = 0;

for (let i = 0; i < 10000; i++) {
    sum += i;
}

console.timeEnd('naam');
```

```
> console.time('naam');
  let sum = 0;

  for (let i = 0; i < 10000; i++) {
      sum += i;
  }

  console.timeEnd('naam');
  naam: 0.3720703125 ms
```

Willekeurige getallen

```
let rand1 = Math.random();                    // [0, 1[

// willekeurig getal tussen 0 en 15
let rand2 = Math.random() * 15;               // [0, 15[

// willekeurig getal tussen 10 en 25
let rand3 = 10 + (Math.random() * 15); // [10, 25[
```

Afronden

```
let positief = 21.4;
let negatief = -21.4;

Math.floor(positief);
// 21
Math.floor(negatief);
// -22
```

```
Math.ceil(positief);
// 22
Math.ceil(negatief);
// -21

Math.round(positief);
// 21
Math.round(negatief);
// -21
```

Globale variabelen

- Variabelen die overal op je pagina/in je logica beschikbaar zijn
- Zoveel mogelijk beperken – zoals we al (zouden moeten) doen
  - Conflicten als twee scripts dezelfde variabelennaam gebruiken
  - Moeilijker om te debuggen (welk stuk code heeft die variabele die waarde gegeven?)

Globale variabelen – Risico beperken

```
// globale variabele
let global = {
  score : 0,
  players : [],
  PATH_PREFIX : "images\sprite",
  PATH_SUFFIX : ".png"
}
```

```
const getScore = () => {
  let score = global.score;

  console.log(score);
}
```