> *Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are – by definition – not smart enough to debug it.*
> —Brian Kernighan

Polling

1 setTimeout
- 1.1 Basiscode
- 1.2 Extra argumenten in setTimeout
- 1.3 Alternatief met anonieme callbackfunctie
- 1.4 Asynchroon programmeren

2 setInterval

Meer info over de `setTimeout` en `setInterval` methodes vind je op https://javascript.info/settimeout-setinterval

1 setTimeout

1.1 Basiscode

De functie `setTimeout` stelt je in staat om een functie uit te voeren na een bepaalde tijd. De functie `setTimeout` heeft normaal gezien (minstens) twee argumenten: de functie die je wil uitvoeren en het aantal milliseconden dat je wil wachten.

Het volgend voorbeeld toont hoe je een waarschuwingsvenster (‘alert’) in de browser toont drie seconden na het openen van de pagina.

```
function verwelkom() {
  alert('Welkom bij de les javaScript!');
}
```

```
// wacht 3000 msec (3 s) en voer dan de functie verwelkom uit
setTimeout(verwelkom, 3000);
```

1.2 Extra argumenten in setTimeout

Na de wachttijd in msec kan je *extra argumenten* meegeven. Zo kan volgende functie gebruikt worden om de boodschap in het waarschuwingsvenster met een parameter aan te passen. De uitvoer in het waarschuwingsvenster is 'Welkom bij de les frontend basis!'.

```
function verwelkom(les) {
  alert(`Welkom bij de les ${les}!`);
}

setTimeout(verwelkom, 3000, 'frontend basis');
```

1.3 Alternatief met anonieme callbackfunctie

Het voorbeeld hierboven kan ook geschreven worden met *een anonieme callbackfunctie* . Vermoedelijk wordt deze methode zelfs meer gebruikt dan met extra parameters in de setTimeout functie zoals hierboven getoond.

```
function verwelkom(les) {
  alert(`Welkom bij de les ${les}!`);
}

setTimeout(() => {
  verwelkom('JavaScript');
}, 3000);
```

1.4 Asynchroon programmeren

De functie setTimeout is asynchroon. Bekijken we even volgend voorbeeld:

```
function hallo() {
  console.log('Tweede');
}

console.log('Eerste');
setTimeout(hallo, 1000);
console.log('Derde');
```

De console geeft volgende uitvoer:

```
Eerste
Derde
Tweede
```

De functie `hallo` wordt pas uitgevoerd na 1000 ms. De uitvoer van de console.log('Derde') gebeurt dus voor de uitvoer van de functie `hallo`.

Dit is een voorbeeld van wat we 'asynchroon programmeren' noemen. De functie `setTimeout` is een asynchrone functie. JS overloopt normaal gezien de regels van een script van boven naar onder. Maar asynchrone functies worden eerst in een aparte wachtrij gezet en pas na de rest uitgevoerd.

Wat als we het uitstel ('delay') gelijk stellen aan 0 ms? Denk even goed na wat de console zal tonen als resultaat van volgende code die enkel als aanpassing de tijd van 1000 ms terugbrengt naar 0 ms:

```
function hallo() {
  console.log('Tweede');
}

console.log('Eerste');
setTimeout(hallo, 0); // uitstel van 0 ms
console.log('Derde');
```

Toon / verberg oplossing

2 setInterval

De methode setInterval heeft dezelfde syntaxis als setTimeout, maar voert de functie die het eerste argument is om de zoveel ms uit als in het tweede argument aangegeven wordt.

We maken volgend voorbeeld. Om de 4 seconden wordt er 1 opgeteld bij een startgetal en wordt het resultaat in de console getoond.

```
let getal = 5; //const zou hier niet werken, waarom niet?
function verhoogEnToon() {
  getal++;
  console.log(getal);
}
```

```
setInterval(verhoogEnToon, 4000);
```

De console toont na 4 s het getal 6, dan 4 s later het getal 7 enz. zonder dat daar een einde aan komt. Stel dat je dit proces na 15 s wil stoppen, dan gebruik je de methode `clearInterval()`

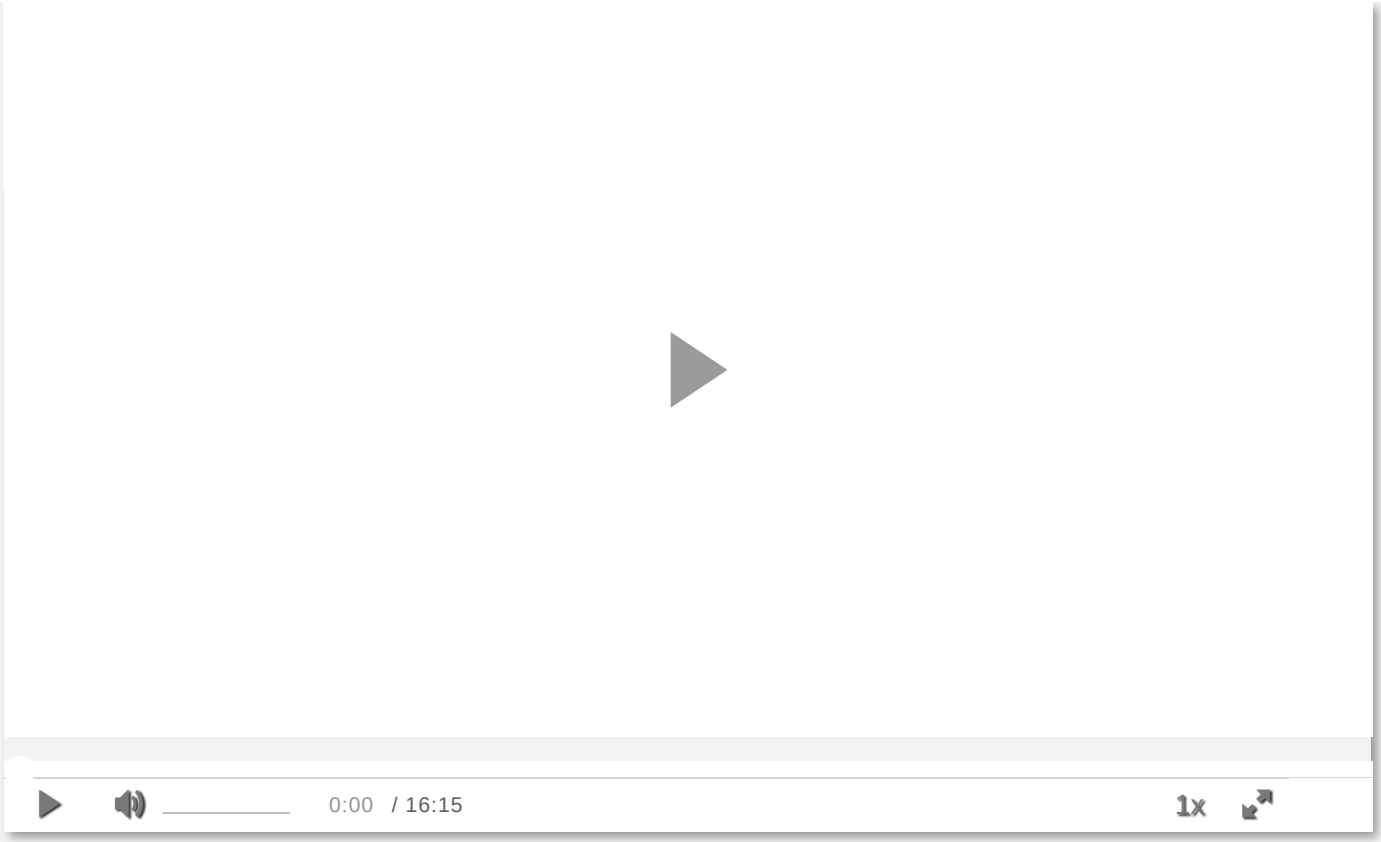
```
let getal = 5;
function verhoogEnToon() {
  getal++;
  console.log(getal);
}

// het resultaat van setInterval is een ID dat je kan gebruiken om het interval te sto
let timerId = setInterval(verhoogEnToon, 4000);
setTimeout(() => {
  clearInterval(timerId);
}, 15000);
```

Welke getallen toont de console als bovenstaande code wordt uitgevoerd?

Toon / verberg oplossing

Onderstaand filmpje toon een eenvoudige (maar naïeve) manier om elke seconde de huidige tijd te tonen in de footer.



↑ Naar top