De tijd controleren met JavaScript

**Het window object beschikt over verschillende methoden om code op voorafbepaalde tijdstippen te laten uitvoeren. Deze tijdsintervallen worden timing events genoemd. De setTimeout en setInterval zijn beiden methoden van het HTML DOM Window object.**

Probleem

In het normale programmaverloop voert de browser de JavaScript code uit zo rap als het kan. Dat wil zeggen dat 5000 lijnen in een knip worden uitgevoerd. Meestal wensen we dat code bliksemsnel wordt uitgevoerd. Maar als we een filmpje willen bekijken is het niet aangenaam dat beelden voorbijflitsen.

Timing events

1. **BOM - setTimeout**
2. **BOM – setInterval**

## BOM - setTimeout

**We moeten tussen elk beeld een pauze kunnen inlassen. Met de functie setTimeout kunnen we de tijd in JavaScript controleren. I.p.v. van het ene statement zo snel mogelijk na het andere te laten uitvoeren zegt setTimeout tegen het programma een beetje te wachten, even op adem te komen vooraleer het volgende statement uit te voeren.**

## Een callback functie

De functie heeft twee parameters. De eerste parameter bevat de instructie. Dit kan één statement zijn of een een functie met verschillende statements erin die je wilt laten uitvoeren nadat de wachttijd, die je in de tweede parameter hebt opgegeven, verlopen is. De functie die je in de eerste parameter hebt opgegeven wordt ook wel een callback genoemd. De instructie kan de naam van een functie zijn of een anonieme functie. We geven eerst een voorbeeld van hoe je de naam van een functie doorgeeft. Hieronder geven we een voorbeeld van hoe jeen anonieme functie - een functie zonder naam - als parameter doorgeeft.

De functie met de naam wachten, of de anonieme functie, worden een callback functie genoemd.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript - De tijd controleren met JS - setTimeout</title>

<meta charset="utf-8" />

<meta name="application-name" content="Multimedia JavaScript">

<meta name="description" content="De tijd controleren met JS - setTimeout">

<meta name="keywords" content="HTML, JavaScript, setTimeout">

<meta name="author" content="JDK">

<script type="text/javascript">

setTimeout(wachten, 6000);

function wachten() {

alert('Heb je lang moeten wachten?');

};

</script>

</head>

<body>

<p>Binnen 6 seconden zal er een alert messagebox getoond worden...</p>

</body>

</html>

Sla bovenstaande code op in een bestand met de naam setTimeout.html. Als de pagina uitvoert zie je onmiddellijk de tekst verschijnen "Binnen 6 seconden zal er een alert messagebox getoond worden...". Na 6 seconden, je kan het aftellen, verschijn inderdaad een alertbox.

Wat gebeurt er als je setTimeout(wachten, 6000); midden in een blok code plaatst?

Dat statement zal worden uitgesteld tot de wachttijd is afgelopen, maar alle andere statements die erop volgen worden onmiddellijk uitgevoerd. Als setTimeout voor de eerste keer wordt uitgevoerd wordt een timer gemaakt die aftelt tot het moment dat het statement moet worden uitgevoerd.

Als we een timer willen stoppen vooraleer het aftellen is beëindigd moeten we de timer in een variabele opslaan en die doorgeven aan een de functie clearTimeout. Die functie zal het aftellen onmiddellijk stilzetten en de geplande taak zal nooit worden uitgevoerd:

var timer = setTimeout(wachten, 4000);

clearTimeout(timer);

Je kan ook een anonieme functie als parameter doorgeven aan de SetTimeout functie. Het voordeel van een anonieme functie is dat je geen aparte functie met een eigen naam moet maken. Als je de code van de anonieme functie meer dan één keer wilt gebruiken, verdwijnt dat voordeel natuurlijk.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript - De tijd controleren met JS - setTimeout</title>

<meta charset="utf-8" />

<meta name="application-name" content="Multimedia JavaScript">

<meta name="description" content="De tijd controleren met JS - setTimeout">

<meta name="keywords" content="HTML, JavaScript, setTimeout">

<meta name="author" content="JDK">

<script type="text/javascript">

setTimeout(wachten, 6000);

function wachten() {

alert('Heb je lang moeten wachten?');

};

// met anonieme functie (functie zonder naam)

var timer2 = setTimeout(function() {alert('Ik ben sneller dan de eerste timer!');}, 2000);

</script>

</head>

<body>

<p>Binnen 6 seconden zal er een alert messagebox getoond worden...</p>

<p>Maar na 2 seconden al wordt een eerste alert messagebox getoond :)

Alhoewel we die later in gang hebben gezet.</p>

</body>

</html>

## BOM - setInterval

**Met setInterval kan je een zich herhalende blok code na een bepaald interval opnieuw laten uitvoeren.**

## Beschrijving

De setInterval methode heeft twee parameters. De eerste parameter dient voor het opgeven van een referentie naar een functie die om de zoveel tijd moet worden uitgevoerd. Deze functie wordt ook wel eens een calback functie genoemd. De tweede parameter geeft aan om de hoeveel tijd de functie moet worden uitgevoerd. Een interval van 1000 komt overeen met 1 seconde.

Met clearInterval kan je proces stopzetten.

var timer = setInterval(wachten, 6000);

clearInterval(timer);

In een eerste eenvoudig voorbeeld maken we een klok op een webpagina. Meer uitleg vind je in de bijgevoegde commentaar.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript - De tijd controleren met JS - setInterval</title>

<meta charset="utf-8" />

<meta name="application-name" content="Multimedia JavaScript">

<meta name="description" content="De tijd controleren met JS - setInterval">

<meta name="keywords" content="HTML, JavaScript, setInterval">

<meta name="author" content="JDK">

</head>

<body>

<p id="clock1">Klok 1 begint zo te tikken... en stopt na 10 seconden...</p>

<!-- De JavaScript wordt hier geplaatst omdat het p html element

eerst geladen moet zijn vooraleer het in JS gebruikt kan worden -->

<script type="text/javascript">

// Binnen de seconde gaat om de seconde de methode tick1() worden uitgevoerd.

var clock1 = setInterval(tick1, 1000);

// Na 10 seconden leggen we de klok stil.

// Hier zie je het nut van anonieme functies. Als je de functie

// clearInterval(clock1) direct als parameter zou meegeven

// zou die ook onmiddellijk uitgevoerd worden en niet binnen 10 seconden.

// Je kan dat zelf eens uitproberen om het verschil te zien.

// Een andere mogelijkheid bestaat erin de clearInterval(clock1) als

// een string door te geven. Maar dat kan een veiligheidslek creëren.

setTimeout(function() {clearInterval(clock1)}, 10000);

// En 5 seconden na het stoppen begint de klok weer tikken

setTimeout(function() {setInterval(tick1, 1000);}, 15000);

function tick1() {

var d = new Date();

// To access an HTML element, use the document.getElementById(id) method.

// The id attribute defines the HTML element.

// The innerHTML property defines the HTML content.

document.getElementById('clock1').innerHTML = d.toLocaleTimeString();

}

</script>

</body>

</html>

In een tweede voorbeeld geven we een complexer voorbeeld. We werken hier met twee klokken.

<!DOCTYPE html>

<html>

<head>

<title>JavaScript - De tijd controleren met JS - setInterval</title>

<meta charset="utf-8" />

<meta name="application-name" content="Multimedia JavaScript">

<meta name="description" content="De tijd controleren met JS - setInterval">

<meta name="keywords" content="HTML, JavaScript, setInterval">

<meta name="author" content="Jef Inghelbrecht">

</head>

<body>

<p id="clock1">Klok 1 begint zo te tikken... en stopt alweer 10 seconden nadat de tweede klok is gestart...</p>

<p id="clock2">Klok 2 begint te tikken binnen 5 seconden... "</p>

<script type="text/javascript">

var clock1 = setInterval(tick1, 1000);

setTimeout(function() {clearInterval(clock1)}, 15000);

function tick1() {

var d = new Date();

// To access an HTML element, use the document.getElementById(id) method.

// The id attribute defines the HTML element.

// The innerHTML property defines the HTML content.

document.getElementById('clock1').innerHTML = d.toLocaleTimeString();

}

var clock2 = setTimeout(function() {

setInterval(tick2, 2000);

setTimeout(function() {clearInterval(clock1)}, 15000);}, 5000);

function tick2() {

var d = new Date();

// To access an HTML element, use the document.getElementById(id) method.

// The id attribute defines the HTML element.

// The innerHTML property defines the HTML content.

document.getElementById('clock2').innerHTML = d.toLocaleTimeString();

}

</script>

</body>

</html>

Met setInterval heb je zelf geen controle over de herhaling. Het is daarom beter om een aaneengeschakelde ketting van calls naar setTimeout te gebruiken.