> *I am more and more convinced that our happiness or unhappiness depends more on the way we meet the events of life than on the nature of those events themselves.*
> —Alexander Humboldt (1769–1859)

Events in JS

Een gebeurtenis (we zullen vaak de Engelse benaming ‘event’ gebruiken) is bvb. op een knop klikken, een toets indrukken, met de muis naar boven bewegen, de rechtermuisknop indrukken, een CSS-animatie bereikt haar einde enz. Kort gezegd is een *event* een signaal dat er iets gebeurd is. In dit hoofdstukje bekijken we hoe je via JS kan *reageren op een event*.

Om te reageren op een gebeurtenis ken je een *event handler* (letterlijk: ‘gebeurtenisafhandelaar’) toe, die dan een functie uitvoert. Er zijn verschillende manieren om met event handlers te werken. We bekijken eerst enkele wat oudere methoden en tenslotte de betere methode `addEventListener()`.

We volgen in het vervolg van dit hoofdstuk de opbouw in https://javascript.info/introduction-browser-events

1 Oude manier 1: via een HTML-attribuut

Bij wijze van voorbeeld maken we een knop die bij elke klik de uitlijning van deze aline verandert van links naar rechts en omgekeerd. Probeer de knop eerst uit! De code en de uitleg volgen onderaan.

Lijn alinea hierboven rechts / links uit

De HTML-code van die eerste alinea en de knop eronder ziet er zo uit:

```
<p id="alineUitlijning">Bij wijze van voorbeeld maken we een knop die … onderaan.</p>
<button onclick="lijnUit()">Lijn alinea hierboven rechts / links uit</button>
```

In het JS bestand dat we in elke pagina van deze site laden, staat volgende code. Deze code voegt via een `toggle()` de klasse ‘tekst-rechts’ toe als ze er nog niet staat, en haalt ze weg als deze klasse wel al bestaat op het element.

```
function lijnUit() {
  document.querySelector("#alineUitlijning").classList.toggle("tekst-rechts");
}
```

Om die knop te doen werken moet er natuurlijk nog één ding ergens in de code toegevoegd worden. Een idee *wat* je *waar* nog moet aanvullen?

Toon / verberg oplossing

Het CSS-bestand moet uitgebreid worden met de stijlregels die voor deze nieuwe klasse van toepassing zijn, dus voegen we het volgende toe aan het stijlblad. De default uitlijning is `text-align: left;`, dus die hoeven we niet in CSS expliciet te noteren.

```
.tekst-rechts {
  text-align: right;
}
```

Deze event handlers via een HTML-attribuut zoals `onclick` laten niet toe om meerdere events te koppelen aan één element. Bovendien mengen ze het *gedrag van een element (JS) met de structuur en betekenis* (HTML). Om die reden zouden we je voorstellen om het afhandelen van gebeurtenissen **niet** op deze manier te doen. Je ziet deze manier van werken nog vaak in oude code terugkomen.

2 Oude manier 2: via een DOM-eigenschap

10

De tweede (wat oudere) manier om met event handlers te werken is via een DOM-eigenschap. We illustreren dit even met een kleine teller. Elke keer als je met de muis over de teller links gaat, wordt die met 1 verhoogd. Je vindt de benodigde code hieronder.

Dit keer hoeven we in de HTML geen attribuut toe te voegen dat begint met ‘on…’. Alles gebeurt in JS. De teller is een gewone alinea met een uniek `id="teller"` en via CSS is het lettertype groter gezet, met grijze achtergrond, links gefloat enz.

De JS-code zorgt voor de event handler en de verhoog-functie. Het event `onmouseover` wordt afgevuurd (zo heet dat in het Engels: ‘an event fires’) als de muis van buiten het object over het object beweegt.

```
let eventTeller = document.querySelector("#teller");
eventTeller.onmouseover = function() {
  let getal = eventTeller.innerText;
  getal++;
  eventTeller.innerText = getal;
  if (getal === 10) {
    eventTeller.onmouseover = null;
  }
}
```

Kan je aan de hand van de JS-code verklaren wat er gebeurt als je meer dan 10 keer over de teller-alinea gaat?

Toon / verberg oplossing

Op het moment dat de variabele `getal` de waarde 10 bereikt, wordt de eigenschap `onmouseover` van het object `eventTeller` op de waarde `null` gezet. Dit verwijdert de event handler.

3 Moderne manier: eventListeners

Beide vorige manieren (via een HTML-attribuut dat begint met 'on...' of via een DOM-eigenschap met een gelijkaardige naam) laten niet toe dat je aan hetzelfde event meerdere event handlers koppelt.

Het gebruik van `eventListener(event, handler)` lost dit probleem op. Ook hier werken we een klein voorbeeldje uit. Een eenvoudige typografische basisregel luidt: regels mogen niet te lang zijn, want dan wordt lezen heel vermoeiend. Een prima gebied is een regellengte tussen 60 en 90 karakters (spaties inbegrepen). We maken hier een klein formulier met alleen een text input field. Je kan in dit formulier tekst typen of een tekst inplakken en dan wordt het aantal karakters geteld.

Plak hieronder 1 regel tekst of typ zelf in het veld:

0

In de vorige secties zagen we al enkele events: onclick en onmouseover. Er zijn er echter meer dan 100. Scroll eens door de lange lijst op MDN. Dit voorbeeld gebruikt het `input` event. Niet alle events beginnen dus met 'on...'. Dit event vuurt af als de `value` van een `input`, `select` of `textarea` HTML-element verandert, wat dus gebeurt elke keer als we in het `input` element van het `form` een letter toevoegen. Dit is de HTML-code van dit voorbeeld:

```
<form>
  <label for="regel">Plak hieronder 1 regel tekst of typ zelf in het veld:</label>
```

```
  <input id="regel" type="text" size="100">
</form>
```

De bijhorende JS-code koppelt een `eventListener()` aan het `input` element in het formulier, nl een `input` event. Als dit event afvuurt wordt de functie `telLetters` uitgevoerd, die de `length` eigenschap toepast op de `value` eigenschap van het inputveld. Dat getal is dan de inhoud van `p class="groot"`

```
let regel = document.querySelector('#regel');
regel.addEventListener('input',telLetters);

function telLetters() {
  let lengte = regel.value.length;
  document.querySelector('p.groot').innerText = lengte;
}
```

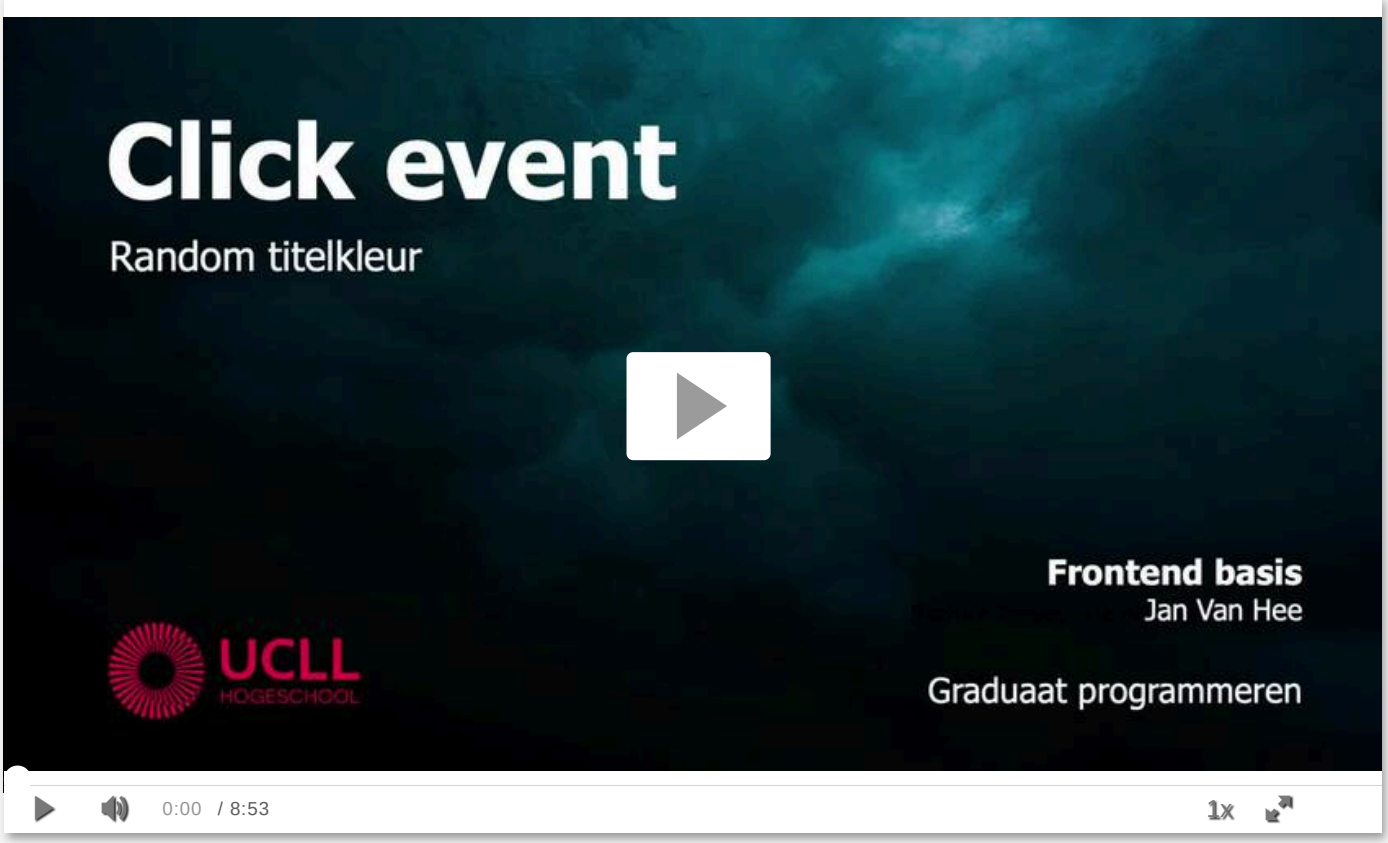
Met de methode `removeEventListener()` verwijder je de event handler.

4 Random titelkleur

Hoog tijd om enkele kleine toepassingen in onze citatenapp te demonstreren. Sommige voorbeelden zullen we verder in onze applicatie terug verwijderen omdat ze verder geen praktisch nut hebben voor de eigenlijke werking van de applicatie.

 In het eerste filmpje reageren we op een ‘click’ event door de h1 titel een random kleur te geven. Volgende topics worden behandeld:

- `addEventListener` met anonieme functie in function-notatie;
- `Math.random()` en `Math.round()`;
- CSS RGB kleurennotatie;
- JS `style` eigenschap aanpassen;
- callbackfunctie herschrijven met vette pijl notatie.



5 JS toggle

Vaak doet een knop dienst als *schakelaar*. Je kiest dan tussen twee standen: licht / donker, aan / uit, groot / klein ... In het volgende voorbeeld maken we een knop die de layout van de citaten doet schakelen tussen één of drie kolommen.

Via de `classList` eigenschap kunnen we van een element alle class attributen opvragen. Je kan hierop dat volgende methodes toepassen: `add()`, `remove()`, `replace()` en `toggle()`. Deze laatste methode gebruiken we hier om onze schakelaar te modelleren.

Onderstaand filmpje toont hoe je een knop maakt die schakelt tussen twee toestanden: citaten allemaal onder elkaar / in drie kolommen. Volgende topics komen aan bod:

- button krijgt uniek id;
- CSS voor button (ook animatie);
- click eventListener toevoegen;
- event handler = classlist.toggle().

6 Het event object

Vaak willen we om een event goed te kunnen afhandelen meer informatie over de gebeurtenis. Niet alleen ‘er is een toetst ingedrukt’, maar ook ‘welke toets?’. De muis staat over het element, maar op welke coördinaten juist?

Als een event gebeurt maakt de browser een *event object*. Dit object bevat details i.v.m. het event en wordt als een argument meegegeven met de event handler.

Zoals altijd leggen we een nieuw concept uit met een klein praktisch voorbeeld. Onderstaande `div` is een vierkant van 255 op 255 pixels. Voor dit specifieke voorbeeld maakten we gebruik van het `style` attribuut, omdat dit voorbeeld uniek is en het eigenlijk geen zin heeft om deze code in het stijlblad van de hele site op te nemen. *Beweeg met de muiscursor over het vierkant*. Code en uitleg komen onder de figuur.

De HTML-code (en specifieke CSS voor deze lege div) is eenvoudig:

```
<div id="kleurVlak" style="width:255px;height:255px;margin:10px auto;background-color:#ccc;"></div
```

De JS-code maakt een `mousemove` eventListener vast aan de `div`. Telkens als de muis bewogen wordt in het vierkant, vuurt het event af en wordt de functie `bepaalKleur()` uitgevoerd. Deze functie krijgt als parameter het event object, dat we de naam `e` gaven, mee. Dit event object bevat details over het event, o.a. de coördinaten van de muispointer t.o.v. het vierkant: `offsetX` en `offsetY`. De linkerbovenhoek van het vierkant heeft coördinaten (0, 0), rechtsonder is (255, 255). Deze twee getallen worden dan als rood- en groenwaarde in een RGB-kleur gestopt. Deze kleur wordt dan toegekend als achtergrondkleur voor het vierkant. Bovendien wordt een string met deze informatie in de `div` toegevoegd.

```
let kleurDiv = document.querySelector("#kleurVlak");
kleurDiv.addEventListener('mousemove',bepaalkleur);

function bepaalkleur(e) { // e bevat nu alle details van het event
  let rood = e.offsetX; // t.o.v. linkerkant van de div
  let groen = e.offsetY; // t.o.v. de bovenkant van de div
  kleurDiv.innerText = ' rood: ' + rood + ' en groen: ' + groen;
  let kleur = 'rgb(' + rood + ',' + groen + ',' + '0)';
  kleurDiv.style.backgroundColor = kleur;
}
```

Het laatste filmpje in dit hoofdstuk toont een klein voorbeeld in onze citatenapp: via de muis kan je de grijstint van een achtergrond, rand enz. aanpassen. Aan bod komen o.a.:

- details over een event opvragen;
- balkje maken met HTML en CSS;
- eigenschappen van een event tonen in de console;
- achtergrondkleur koppelen aan een x-coördinaat;
- anonieme callbackfunctie doet verschillende dingen.

↑ Naar top