

User Interfaces 1

JavaScript

Events

Events

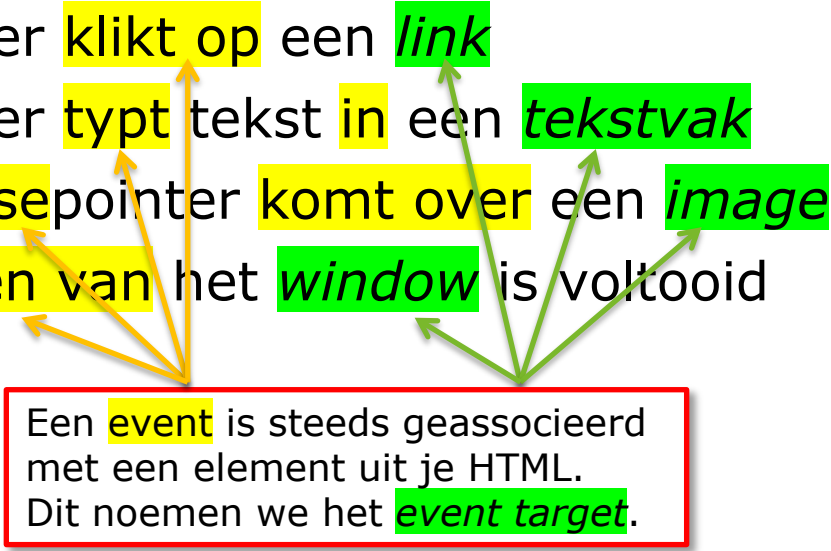
- Wat zijn events?
- Reageren op events
- Event types
- Event handler
- `Event` object
- Even propagation

Wat zijn events?

Event = een gebeurtenis in je browser!

- Voorbeelden:

- Gebruiker **klikt op** een **link**
- Gebruiker **typt** tekst **in** een **tekstvak**
- De **mousepointer** **komt over** een **image**
- Het **laden van** het **window** is voltooid
- ...



Een **event** is steeds geassocieerd met een element uit je HTML. Dit noemen we het **event target**.

→ *Het zou leuk zijn moesten we scripts kunnen starten wanneer een event afgaat...!*

Reageren op events

Buiten een **event** dat op een **target** plaatsvindt, hebben we een **event handler** nodig om hierop te reageren

We hebben dus 3 zaken nodig:

1. Event target
 - Een html-element waarop een event plaatsvind
2. Event type
 - Het soort event dat plaatsvindt
3. Event handler
 - Een functie die aangeroepen/uitgevoerd zal worden als het event zich voor doet

Event types

Naam	Gaat af...
"click"	...bij het klikken op een element
"mousedown"	...wanneer de muisknop ingedruwd wordt
"mouseup"	...wanneer de muisknop losgelaten wordt
"mouseover"	...wanneer de mousepointer over het element gaat
"mouseout"	...wanneer de mousepointer het element verlaat
"mousemove"	...wanneer de mousepointer over element beweegt
"keypress"	...wanneer een toets wordt getypt
"keyup"	...wanneer een toets wordt losgelaten
"keydown"	...wanneer een toets wordt ingedrukt
"focus"	...wanneer het element focus krijgt
"load"	...wanneer het element/document geladen is
...	



Laden DOM script: 'laadprobleem'

=> oplossing: gebruik het `load` event van het `window` object!

Event handler

```
function mijnEventHandler(event) {  
    console.log("Het event " + event + " is net gebeurd!");  
    console.log(event);  
}
```

het Event Object

Functie die uitgevoerd wordt als het event afgaat.
Krijgt een **event object** mee als parameter
(met extra info over het event)

```
Het event [object MouseEvent] is net gebeurd!  
...  
target: <button id="knop">  
type: "click"  
...
```

Event handler registreren: `addEventListener()`

`addEventListener()` is een methode van het **event target** (meestal een element uit de DOM tree, maar kan ook iets anders zijn...)

```
window.addEventListener("load", init, false);  
knop.addEventListener("click", mijnEventHandler, false);
```

- 3 parameters:
 - Het **event type** (een string)
 - De *event handler* (een functie)
 - Een **oncapturing** (een boolean)
(mag weggelaten worden want default = false, zie later)

Event handler registreren: voorbeeld

event target

event type

event handler

```
window.addEventListener("load", init);
```

Registreren van de handler:
als het **load** event afgaat op het **window**, wordt de handler **init** gestart

```
function init(event) {  
  let knop = document.getElementById("knop");  
  knop.addEventListener("click", mijnEventHandler);  
}
```

Registreren van de handler:
als het **click** event afgaat op **knop**, wordt de handler **mijnEventHandler** gestart

```
function mijnEventHandler(event) {  
  console.log("Het event " + event + " is net gebeurd!");  
  console.log(event);  
}
```

Het event [object MouseEvent] is net gebeurd!

```
...  
target: <button id="knop">  
type: "click"  
...
```


Event handler registreren: **bad practices!!**

DOM Level 0 Events...

- Traditioneel model

```
window.onload = init_oud;  
  
function init_oud() {  
    /* uit te voeren code */  
}
```

BAD

- Inline model

```
<body onload="init_oud()">  
    <!-- rest van HTML -->  
</body>
```

VERY BAD!



Event object

```
function mijnEventHandler(event) {  
    console.log("Het event " + event + " is net gebeurd!");  
    console.log(event);  
}
```

- Bevat extra info over het event:
 - `event.target`: het event target = button, window...
 - `event.type`: het event type = click, load...
- Bevat ook een aantal interessante methodes:
 - `event.preventDefault()`
 - `event.stopPropagation()`

Event object: `preventDefault()`

```
function mijnEventHandler(event) {  
    console.log("Het event " + event + " is net gebeurd!");  
    event.preventDefault();  
}
```

De actie die bij een event hoort, wordt nu niet meer uitgevoerd.
M.a.w. als het event te stoppen valt, wordt het hierdoor gestopt.

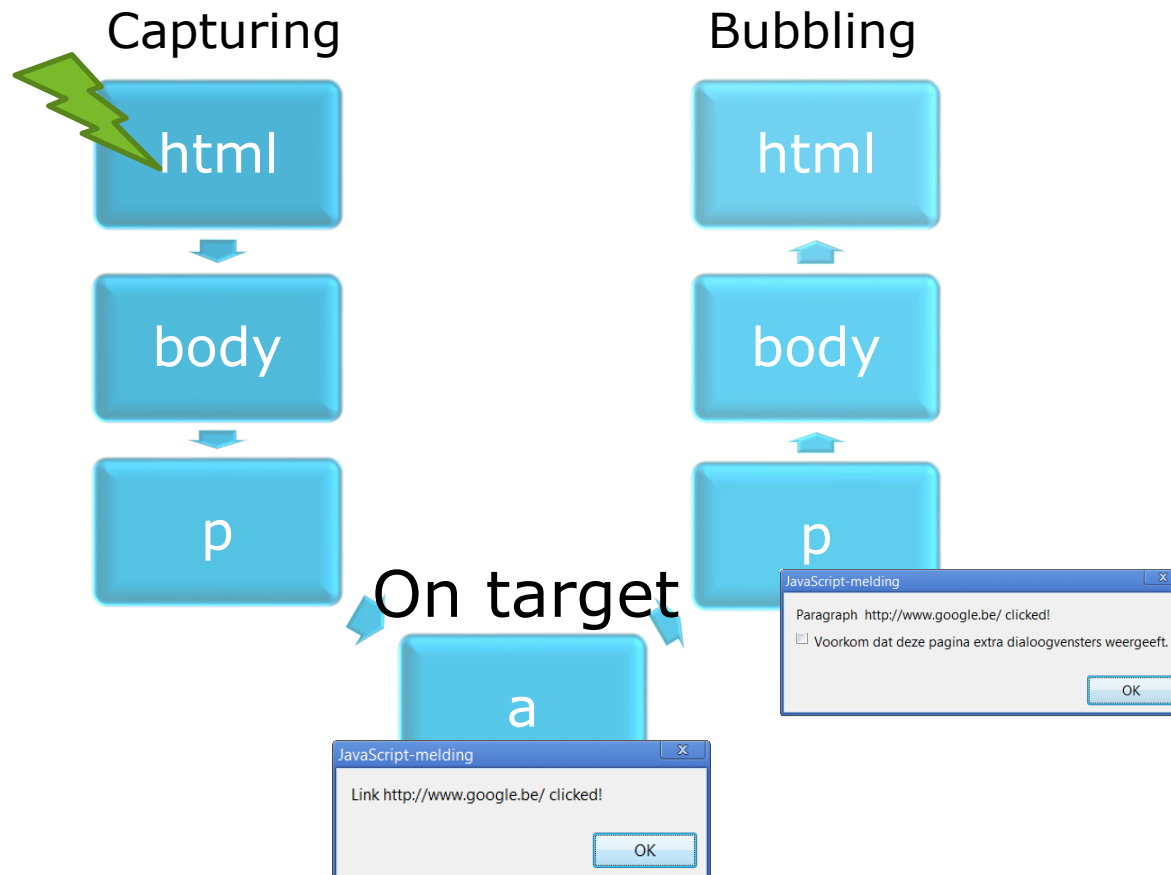
Voorbeeld:

- klikken op een link = de link 'volgen',
door `event.preventDefault()` zal de link niet geopend worden...
- klikken op een checkbox = verander de toestand,
door `event.preventDefault()` zal er echter niets gebeuren...
- ...

Event propagation

Volgorde van afhandelen van event handlers:

1. alle event handlers die `onCapturing` als `true` gezet hebben,
2. diegenen die `onCapturing` op `false` hebben staan (~default)



Event propagation: opmerking

Meestal wordt *event bubbling* gebruikt, en kan dus de `onCapturing` parameter achterwege gelaten worden want deze is default **false**!

Om de propagation flow te onderbreken gebruik je:
`event.stopPropagation() ;`

Denkcoefening: event propagation 1



```
<body>
<p>
  blablabla
  <a href="https://www.google.be">google</a>
</p>
</body>
```

```
addEventListener("load", windowLoaded, false);

function windowLoaded() {
  let theLink = document.querySelector("a");
  let theParagraph = document.querySelector("p");
  theLink.addEventListener("click", linkClicked, false);
  theParagraph.addEventListener("click", pClicked, false);
}

function linkClicked(event) {
  alert("Link " + event.target + " clicked!");
}

function pClicked(event) {
  alert("Paragraph " + event.target + " clicked!");
}
```

Denk oefening: event propagation 2



```
<body>
<p>
  blablabla
  <a href="https://www.google.be">google</a>
</p>
</body>
```

```
addEventListener("load", windowLoaded, false);

function windowLoaded() {
  let theLink = document.querySelector("a");
  let theParagraph = document.querySelector("p");
  theLink.addEventListener("click", linkClicked, false);
  theParagraph.addEventListener("click", pClicked, true);
}

function linkClicked(event) {
  alert("Link " + event.target + " clicked!");
}

function pClicked(event) {
  alert("Paragraph " + event.target + " clicked!");
}
```

Denkcoefening: event propagation 3



```
<body>
<p>
  blablabla
  <a href="https://www.google.be">google</a>
</p>
</body>
```

```
addEventListener("load", windowLoaded, false);

function windowLoaded() {
  let theLink = document.querySelector("a");
  let theParagraph = document.querySelector("p");
  theLink.addEventListener("click", linkClicked, false);
  theParagraph.addEventListener("click", pClicked, true);
}

function linkClicked(event) {
  alert("Link " + event.target + " clicked!");
}

function pClicked(event) {
  alert("Paragraph " + event.target + " clicked!");
  event.stopPropagation();
}
```