1 Events en event handlers

Javascript is *'een event-driven programmeertaal'*. Met Javascript kunnen we reageren op events. Dit maakt onze webpagina's interactiever.

1.1 Events

Events oftewel gebeurtenissen zijn 'dingen' die gebeuren met de HTML-elementen op een webpagina, zoals bijvoorbeeld:

- het beëindigen van het laden van de webpagina
- het veranderen van de inhoud van een invulveld
- het klikken op een knop
- het hoveren met de muisaanwijzer over een afbeelding
- …

Onderstaande tabel bevat enkele veelvoorkomende events die we kunnen gebruiken om een script uit te voeren. Een volledige lijst van mogelijke events kan je terugvinden op de website van W3Schools (*https://www.w3schools.com/jsref/dom_obj_event.asp*).

Event	**Uitleg**
`change`	Voert een script uit wanneer een element wordt aangepast.
`click`	Voert een script uit wanneer op een element wordt geklikt.
`dblclick`	Voert een script uit wanneer op een element wordt gedubbelklikt.
`mouseover`	Voert een script uit wanneer over een element wordt gegaan met een muisaanwijzer.
`mouseout`	Voert een script uit wanneer van een element wordt weggegaan met een muisaanwijzer.
`keydown`	Voert een script uit wanneer een toetsenbordknop wordt ingedrukt.
`load`	Voert een script uit wanneer de browser klaar is met het laden van de webpagina.
`copy`	Voert een script uit wanneer content van een element wordt gekopieerd.
`submit`	Voert een script uit wanneer een formulier wordt gesubmitteerd.

1.2 Targets

Events hebben altijd een target. Het target is het HTML-element waarop het event van toepassing is. Als we een event willen verwerken, moeten we zowel de eventnaam gebruiken als het target.

1.3 Event handlers, event listeners of callbacks

Met een event handler kunnen we ervoor zorgen dat onze toepassing op een event reageert. De event handler is meestal een functie binnen Javascript waarin andere Javascript-functies of -taken staan die worden uitgevoerd zodra het event plaatsvindt.

Andere woorden voor event handler zijn event listener of callback.

1.3.1 Event handlers als eventattribuut in de HTML-code

We kunnen een event (voorafgegaan door het woord 'on') als eventattribuut toevoegen aan een HTML-element en zo het event toewijzen aan het HTML-element.

Dit is een oude manier waarmee je HTML-code en Javascript-code mixt. Deze manier werkt in de praktijk wel nog, ***maar wordt niet meer gebruikt!***

HTML-code

```
<h1 id=”titel” onclick="veranderTekst()">Dit is een titel!</h1>
```

JS-code

```
function veranderTekst() {
     document.getElementById(“titel”).innerHTML = “Een nieuwe titel!”;
}
```

1.3.2 Event handlers in Javascript

We kunnen via deze tweede manier ook een event (voorafgegaan door het woord 'on') volledig toewijzen aan een HTML-element in ons script zelf.

HTML-code

```
<button id="knop">Toon de datum</button>
<p id="datumparagraaf"></p>
```

JS-code

```
document.getElementById("knop").onclick = toonDatum;
function toonDatum() {
     document.getElementById("datumparagraaf").innerHTML = Date();
}
```

1.3.3 De functie addEventListener()

De functie addEventListener() is een nieuwere en modernere manier om event handlers te schrijven in Javascript. *Naar deze manier gaat onze voorkeur uit!*

Deze functie koppelt een event aan een target zonder invloed te hebben op andere, bestaande event handlers. Het is met deze functie ook mogelijk om meerdere event handlers toe te voegen aan één target. Het target kan daarnaast eender welk DOM-object zijn en beperkt zich niet alleen tot HTML-elementen.

De functie heeft minimaal twee belangrijke onderdelen:

1. Het event (niet meer voorafgegaan door het woord 'on')
2. De functie die wordt uitgevoerd wanneer het event plaatsvindt

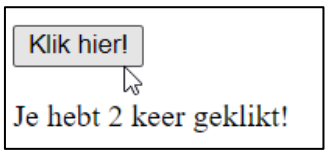
Voorbeeld 1 – X-aantal keer geklikt op een knop

HTML-code

```
<button id="knop">Klik hier!</button>
<p id="clicksparagraaf"></p>
```

JS-code

```
let nummerClicks = 0;
document.getElementById("knop").addEventListener("click", aantalClicks);
function aantalClicks() {
     nummerClicks++;
     document.getElementById("clicksparagraaf").innerHTML = "Je hebt " +
     nummerClicks + " keer geklikt!";
}
```



Voorbeeld 2 - MouseEvents

HTML-code

```
<h1>Test de 'mouseover' en 'mouseout'</h1>
<div id="box"></div>
<p id="uitkomst"></p>
```

CSS-code

```
#box {
     width: 500px;
     height: 250px;
     background-color: #f00;
}
```

JS-code

```
let box = document.getElementById("box");
let uitkomst = document.getElementById("uitkomst");
box.addEventListener("mousover", function inDeBox() {
     uitkomst.innerHTML += "De muis is nu in de box<br>";
});
box.addEventListener("mouseout", function uitDeBox() {
     uitkomst.innerHTML += "De muis is terug uit de box<br>";
});
```

Voorbeeld 3 – Inhoud van tekstvak hergebruiken

HTML-code

```
<input type=”text” name=”Naam” id=”tekstvak” placeholder=”John Doe”>
<button id=”knop”>Toon welkomsboodschap</button>
<p id=”boodschap”></p>
```

JS-code

```
document.getElementById(“knop”).addEventListener(‘click’, welkom);
function welkom() {
    naam = document.getElementById(“tekstvak”).value;
    document.getElementById(“boodschap”).innerHTML = “Welkom “ + naam;
}
```