**De tweede manier**
De tweede manier om een event-handler te implementeren, is met behulp van de methode `addEventListener()`.
Dit is iets meer werk dan de eerste manier, maar de flexibiliteit is groter: HTML en JavaScript zijn gescheiden, u kunt meer handlers (eventueel van hetzelfde type) aan een element toevoegen, u kunt een handler verwijderen en u kunt aangeven hoe events op bubbling moeten reageren. Op bubbling komen we nog terug.

De methode `addEventListener()` roept u meestal aan met twee parameters: de naam van de event en de functie die moet worden uitgevoerd (de derde parameter heeft betrekking op bubbling/capture en kunt u vaak weglaten):

```
element.addEventListener(event, function);
```

In de naam van de event laat u 'on' weg, dus "`onclick`" wordt "`click`" en "`onmouseover`" wordt "`mouseover`".
Op de plaats van `element` staat het element waar de event betrekking op heeft, bijvoorbeeld een knop die u opvraagt met:

```
document.getElementById("knop1")
```

Voor het gebruik van `addEventListener()`: zie w3schools.com – Dom_EventListener.

Een volledige aanroep van `addEventListener()` kan nogal lang worden, bijvoorbeeld:

```
document.getElementById("knop1").addEventListener("click", myFunction);
```

Het kan dan overzichtelijker zijn het element eerst in een variabele op te slaan:

```
var knop1 = document.getElementById("knop1");
knop1.addEventListener("click", myFunction);
```

**De derde manier**
Om de verwarring groter te maken, is er nog een derde manier. Deze manier is wat ouder dan `addEventListener()`, maar is wel redelijk kort en flexibel en wordt door alle browsers wordt ondersteund. Daarom wordt deze manier veel toegepast.

Voorbeeld:

```
<button id="knop1" type="button">Klik</button>

<script>
  var knop1 = document.getElementById("knop1");
  knop1.onclick = myFunction;
  ...
  <script>
```

Ook bij deze manier, net als bij de tweede manier, zijn HTML en JavaScript gescheiden.

De algemene vorm van deze manier is:

```
element.eventnaam = functienaam;
```

Merk op dat er staat `knop1.onclick=myFunction` en **niet** `knop1.onclick=myFunction()`.
In het eerste geval staat alleen de naam van de functie (een referentie), in het tweede geval staat een functie-aanroep.

**Een anonieme functie**
Soms is het nogal veel werk een aparte functie te schrijven voor een handler. Het kan korter met behulp van een anonieme functie.
Eerst een voorbeeld met een knop die een event-handler krijgt volgens de derde manier, maar zonder anonieme functie.

```
<button id="knop1" type="button">Klik</button>
<p id="demo"></p>

<script>
  var knop1 = document.getElementById("knop1");
  knop1.onclick = myFunction;

  function myFunction() {
      document.getElementById("demo").innerHTML = "Dank u!";
  }
</script>
```

Nu hetzelfde voorbeeld met behulp van een anonieme functie voor de event-handler.
De algemene vorm van deze manier is:

```
element.eventnaam = definitie anonieme functie;
```

Een anonieme functie begint met `function()`, gevolgd door een body tussen accolades.
Bovenstaand voorbeeld wordt dan:

```
<button id="knop1" type="button">Klik</button>
<p id="demo"></p>

<script>
  var knop1 = document.getElementById("knop1");
  knop1.onclick =
    function() {document.getElementById("demo").innerHTML = "Dank u!"};
</script>
```

Vooral als in de body van de functie maar één statement staat, kan dit erg handig zijn.
Nadeel is dat de functie geen naam heeft en dus niet anders kan worden aangeroepen dan vanuit de situatie waarin hij is gedefinieerd.

**Een argument doorgeven aan een event-handler**

Er is nog een situatie waarin een anonieme functie handig blijkt, en dat is wanneer u een argument aan een event-handler wilt doorgeven. Stel u hebt een pagina met twee tekstvakken, en u wilt dat zodra de gebruiker in een vak klikt en dat vak dus de focus krijgt, de achtergrondkleur van het vak verandert.

De pagina met twee tekstvakken:

```
<body>
  <input id="vak1" type="text">
  <input id="vak2" type="text">
</body>
```

Het script definieert een functie die de achtergrondkleur van een element verandert:

```
  function myFunction(element, kleur) {
     element.style.background = kleur;
  }
```

Als de gebruiker in `vak1` klikt, moet dat vak geel worden. Dat gebeurt door in de body van de anonieme functie de functie `myFunction` aan te roepen met de juiste argumenten.

```
  vak1.onfocus = function() {myFunction(vak1, "yellow")};
```

Evenzo voor `vak2`:

```
<script>
  var vak1 = document.getElementById( "vak1" );
  var vak2 = document.getElementById( "vak2" );

  vak1.onfocus = function() {myFunction(vak1, "yellow")};
  vak2.onfocus = function() {myFunction(vak2, "red")};

  function myFunction(element, kleur) {
     element.style.background = kleur;
  }
</script>
```