De principes van unobtrusive JavaScript

Peter-Paul Koch (ppk)
http://www.quirksmode.org
PFCongrez, 12 april 2008

Unobtrusive JavaScript

Bescheiden?
Onopvallend?

Unobtrusive JavaScript

Volgens Wikipedia is het een “emerging paradigm in the JavaScript programming language”

Volgens mij is het gewoon een goed idee.

Unobtrusive JavaScript

Twee basisprincipes:

1) Scheiding van structuur, presentatie en gedrag
2) Het script maakt geen annames

Unobtrusive JavaScript

Twee basisprincipes:

1) Scheiding van structuur, presentatie en gedrag
2) Het script maakt geen annames

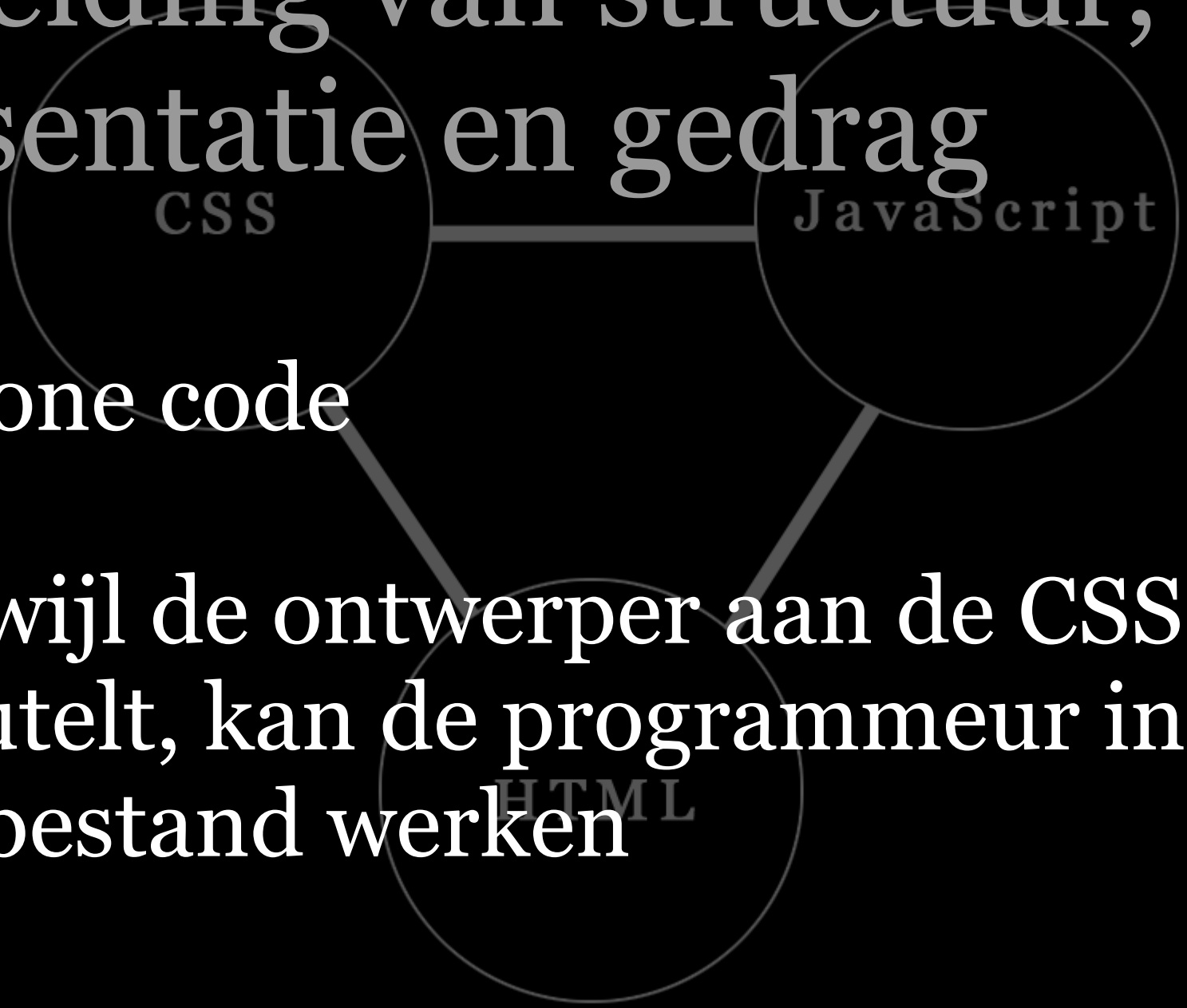
Scheiding van structuur, presentatie en gedrag

Scheiding van structuur, presentatie en gedrag

Eén bestand bevat slecht één soort code.

HTML, CSS, of JavaScript.

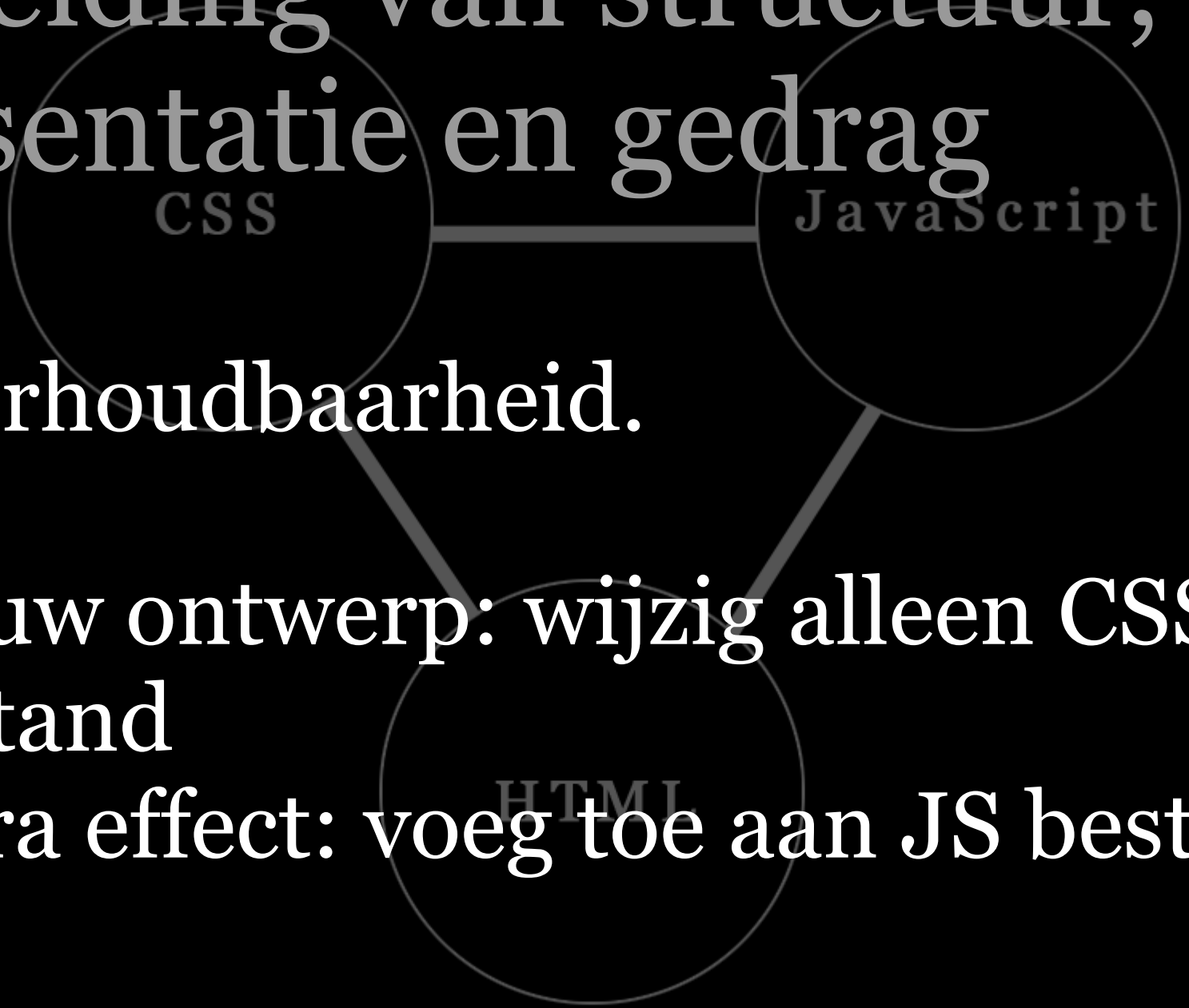
Scheiding van structuur, presentatie en gedrag



- Schone code

- Terwijl de ontwerper aan de CSS sleutelt, kan de programmeur in het JS bestand werken

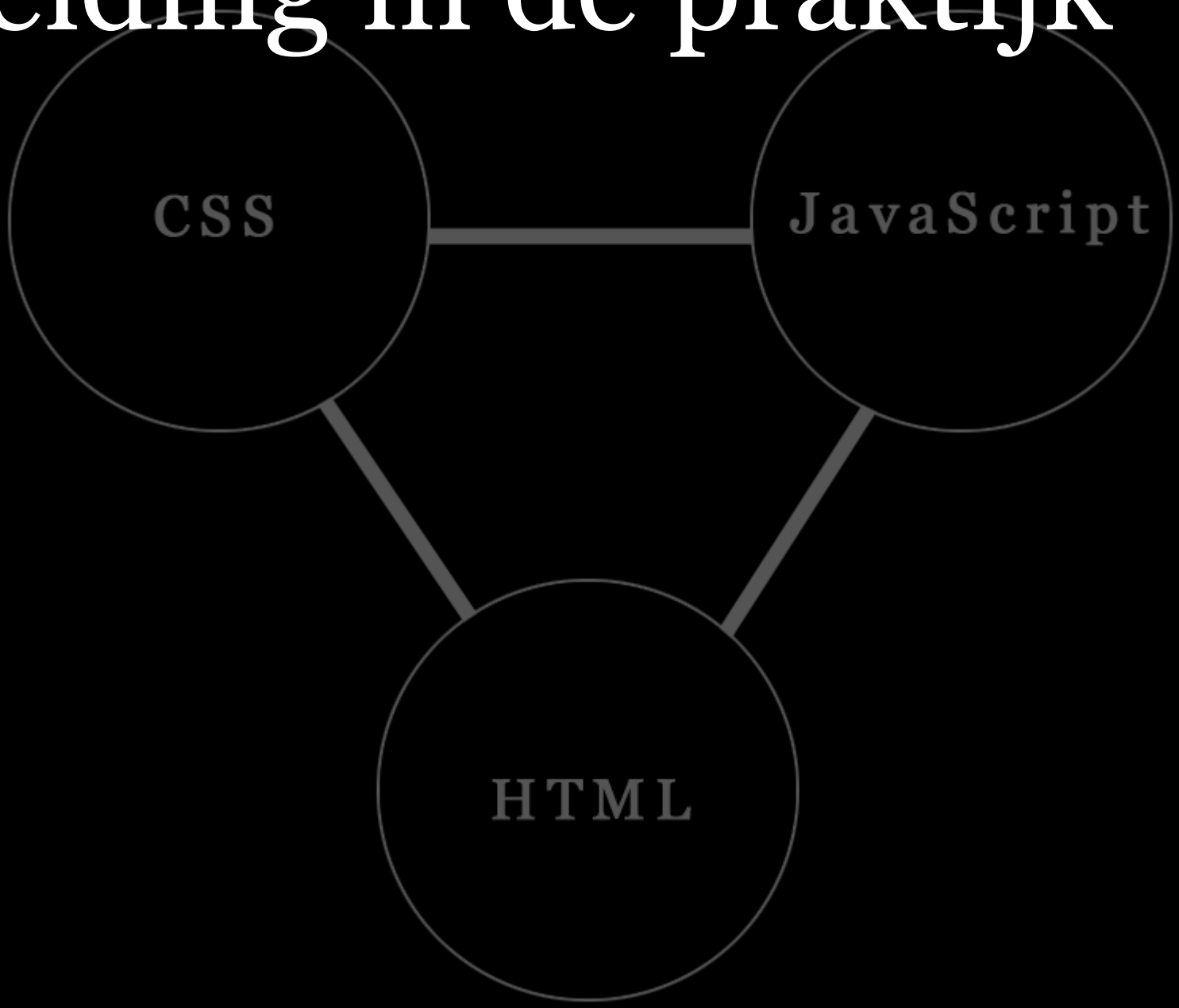
Scheiding van structuur, presentatie en gedrag



Onderhoudbaarheid.

- Nieuw ontwerp: wijzig alleen CSS bestand
- Extra effect: voeg toe aan JS bestand

Scheiding in de praktijk



Scheiding in de praktijk

Scheiding van HTML en CSS:

<div style=”position: relative”>

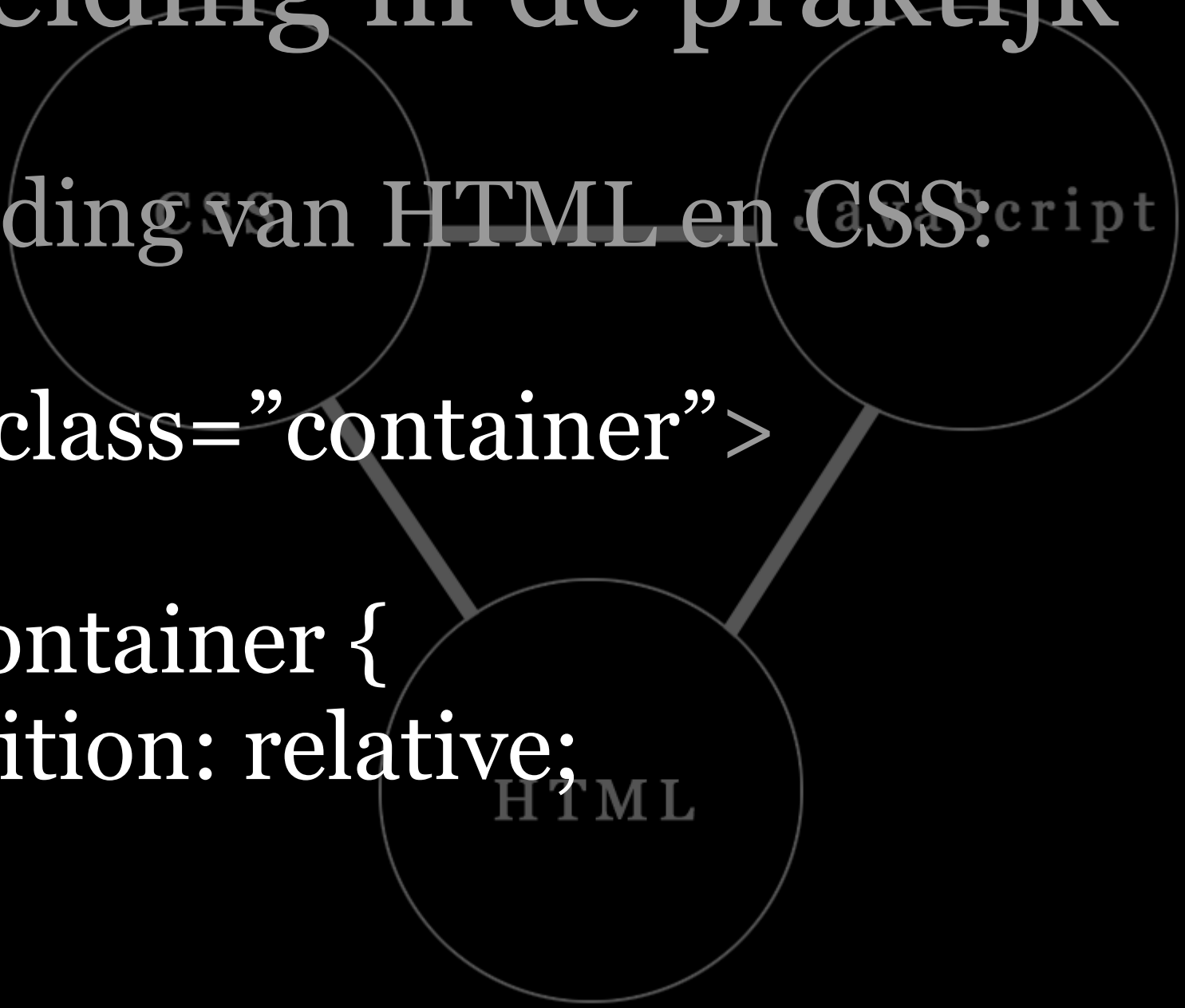
Scheiding in de praktijk

Scheiding van HTML en CSS:

<div ~~style=”position: relative”~~>

Geen inline styles!

Scheiding in de praktijk



Scheiding van HTML en CSS:

```
<div class=”container”>
```

```
div.container {
   position: relative;
}
```

Scheiding in de praktijk

Scheiding van HTML en JavaScript:

<input onmouseover=”doSomething()” />

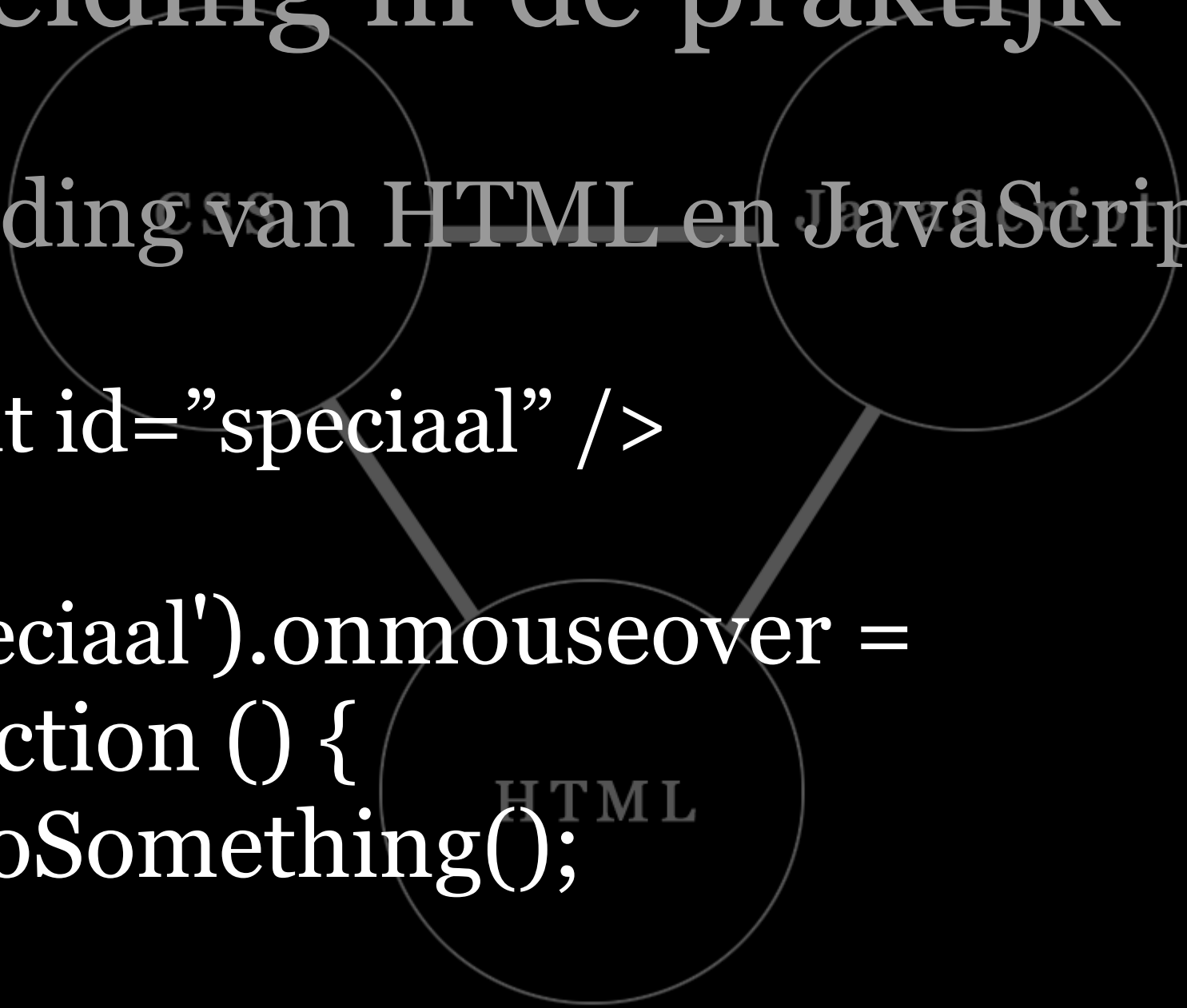
Scheiding in de praktijk

Scheiding van HTML en JavaScript:

<input ~~onmouseover=”doSomething()”~~ />

Geen inline event handlers!

Scheiding in de praktijk

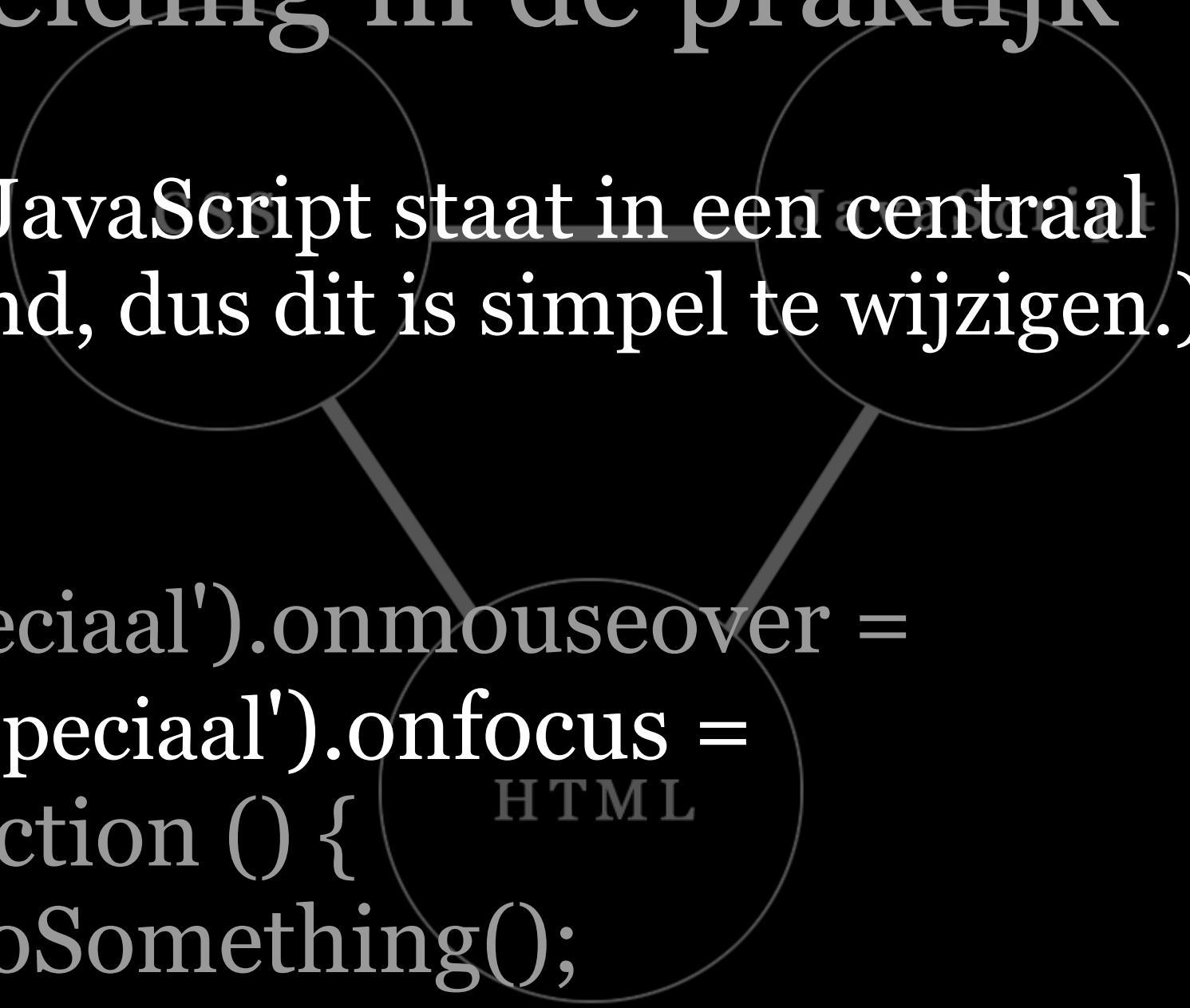


Scheiding van HTML en JavaScript:

```
<input id=”speciaal” />

$('speciaal').onmouseover =
   function () {
      doSomething();
   }
```

Scheiding in de praktijk

(Alle JavaScript staat in een centraal bestand, dus dit is simpel te wijzigen.)




```
$('speciaal').onmouseover =
   $('speciaal').onfocus =
   function () {
      doSomething();
   }
```

Aanhaakpunten

Je CSS en JavaScript moeten weten met welke HTML-elementen ze iets moeten doen.

Je moet aanhaakpunten definiëren.

Aanhaakpunten

- id
- class

Laat CSS en JavaScript zoveel mogelijk gebruik maken van dezelfde aanhaakpunten.

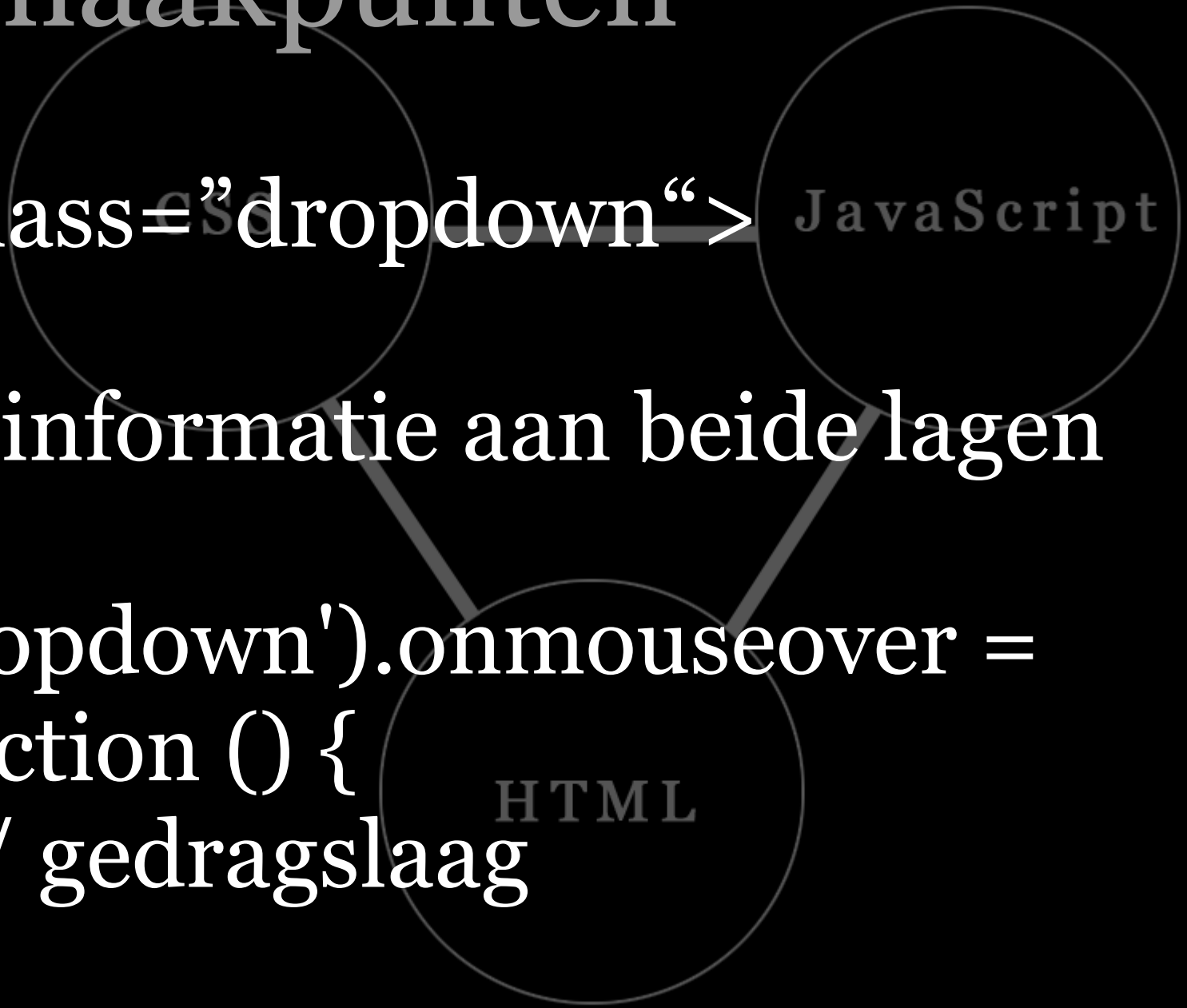
Aanhaakpunten

<ol class=”dropdown“>

geeft informatie aan beide lagen

```
ol.dropdown {
   // presentatielaag
}
```

Aanhaakpunten




```
<ol class=”dropdown“>

geeft informatie aan beide lagen

$('dropdown').onmouseover =
   function () {
      // gedragslaag
   }
}
```

Unobtrusive JavaScript

Twee basisprincipes:

1) Scheiding van structuur, presentatie en gedrag
2) Het script maakt geen annames

Unobtrusive JavaScript

Twee basisprincipes:

1) Scheiding van structuur, presentatie en gedrag
2) Het script maakt geen annames


BE
CAREFUL
THIS MACHINE
HAS NO BRAIN
USE YOUR OWN

Aanname

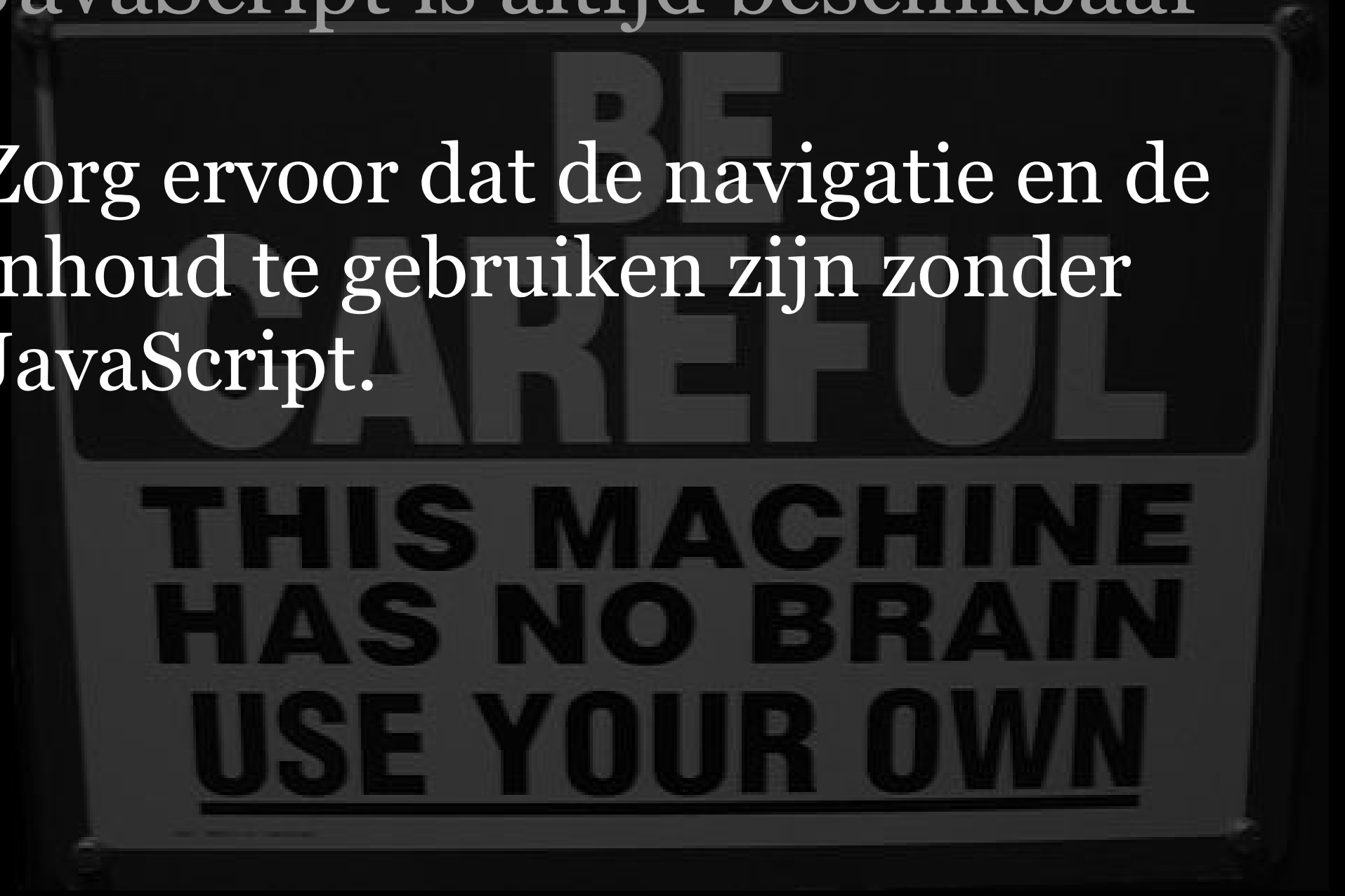
JavaScript is altijd beschikbaar

Onzin!

JavaScript is altijd beschikbaar

- Primitieve mobieltjes
- Voorleesbrowsers (niet omdat ze geen JavaScript ondersteunen, maar omdat het raar werkt)
- Bedrijfsnetwerken die JavaScript wegfilteren

JavaScript is altijd beschikbaar



Zorg ervoor dat de navigatie en de inhoud te gebruiken zijn zonder JavaScript.

JavaScript is altijd beschikbaar

Zorg ervoor dat de navigatie en de inhoud te gebruiken zijn zonder JavaScript.

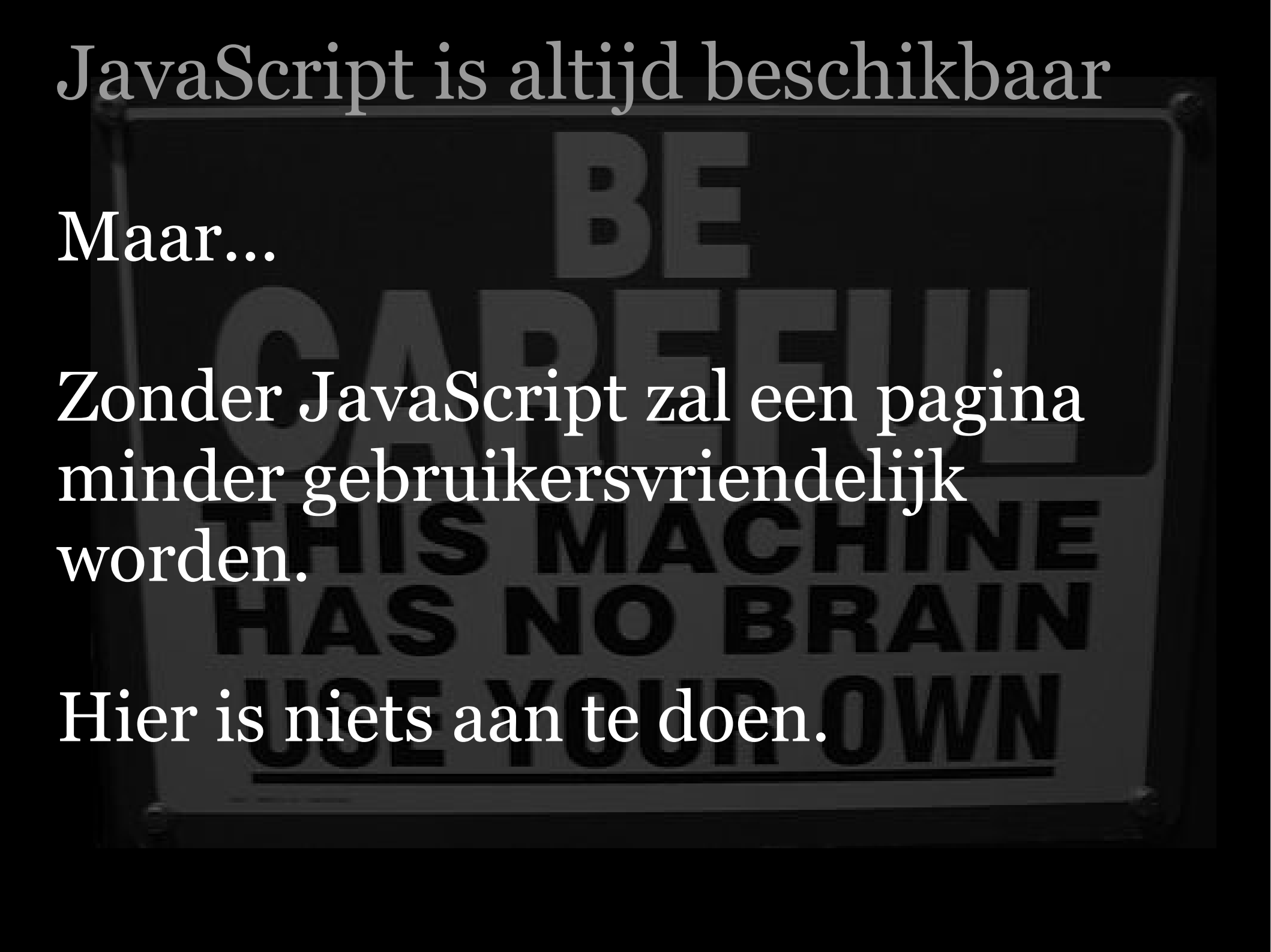
De pagina zal dan altijd een basisniveau van toegankelijkheid houden.

JavaScript is altijd beschikbaar

Zorg ervoor dat de navigatie en de inhoud te gebruiken zijn zonder JavaScript.

Extra functionaliteit kan echter zonder probleem in JavaScript geschreven worden.

JavaScript is altijd beschikbaar



Maar...

Zonder JavaScript zal een pagina minder gebruikersvriendelijk worden.

Hier is niets aan te doen.

JavaScript is altijd beschikbaar

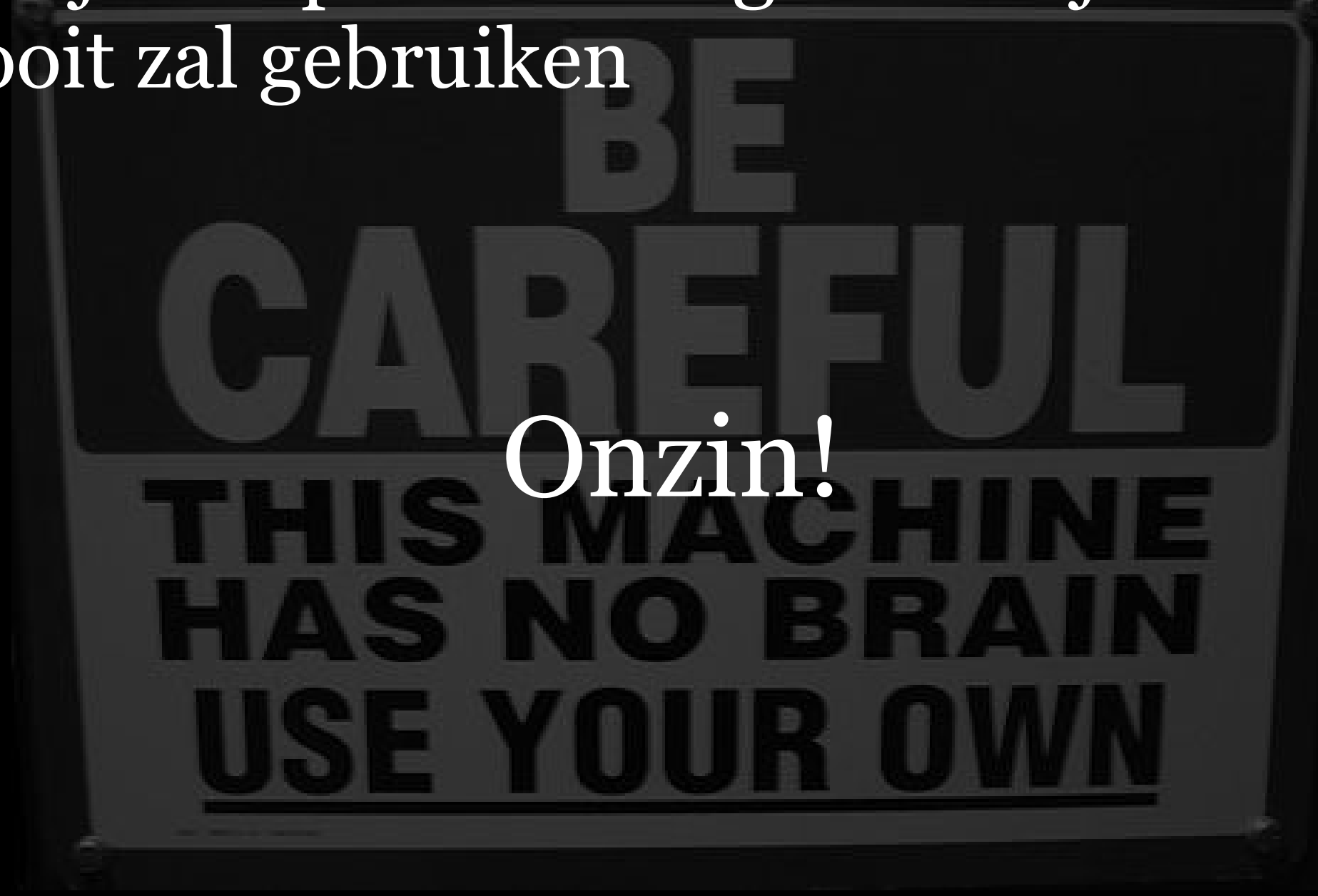
Maar...

Zonder JavaScript zal een pagina minder gebruikersvriendelijk worden.

Het doel van JavaScript is immers gebruikersvriendelijkheid aan de pagina toe te voegen.

Aanname

Mijn script is het enige dat mijn klant ooit zal gebruiken

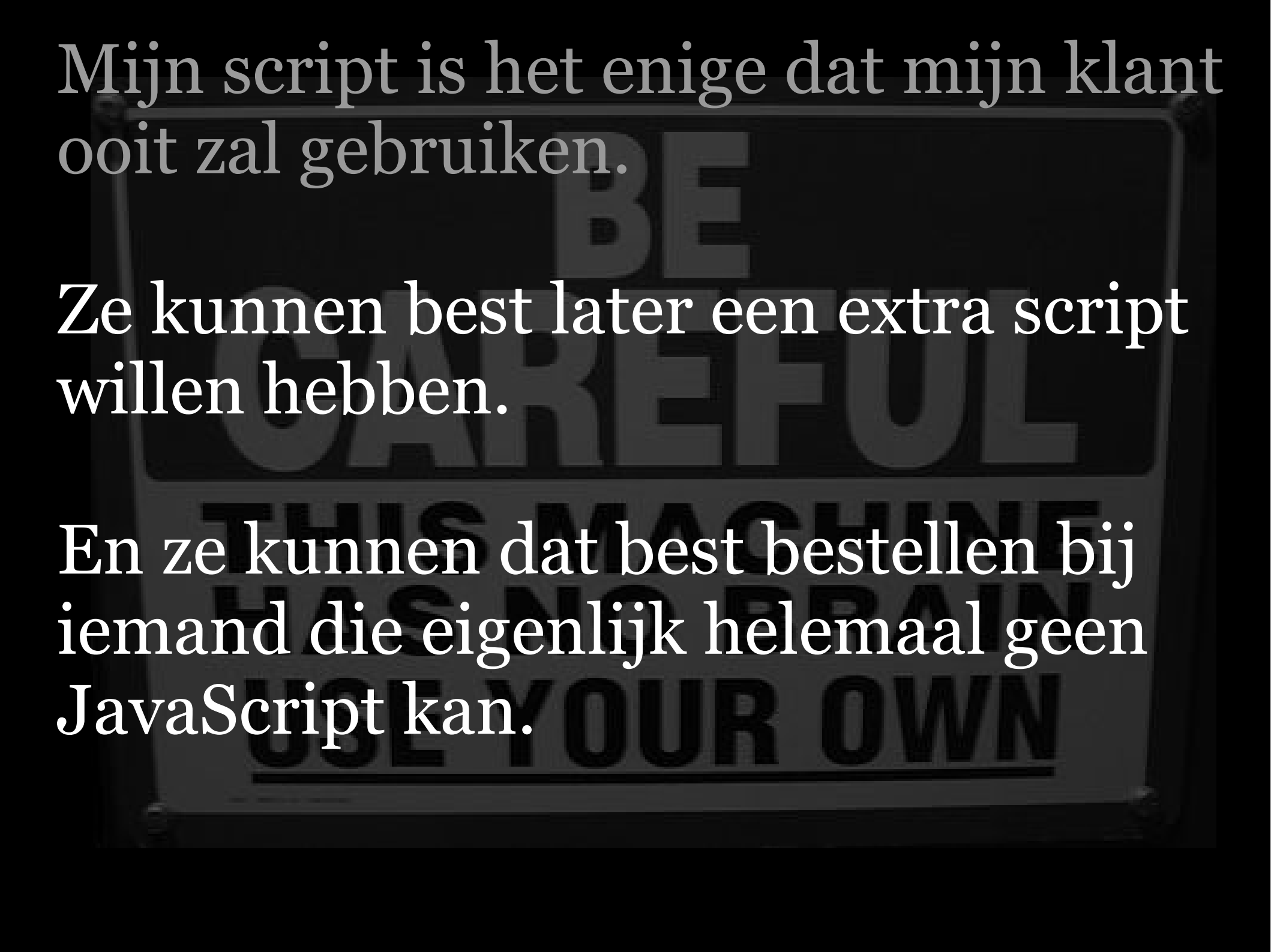


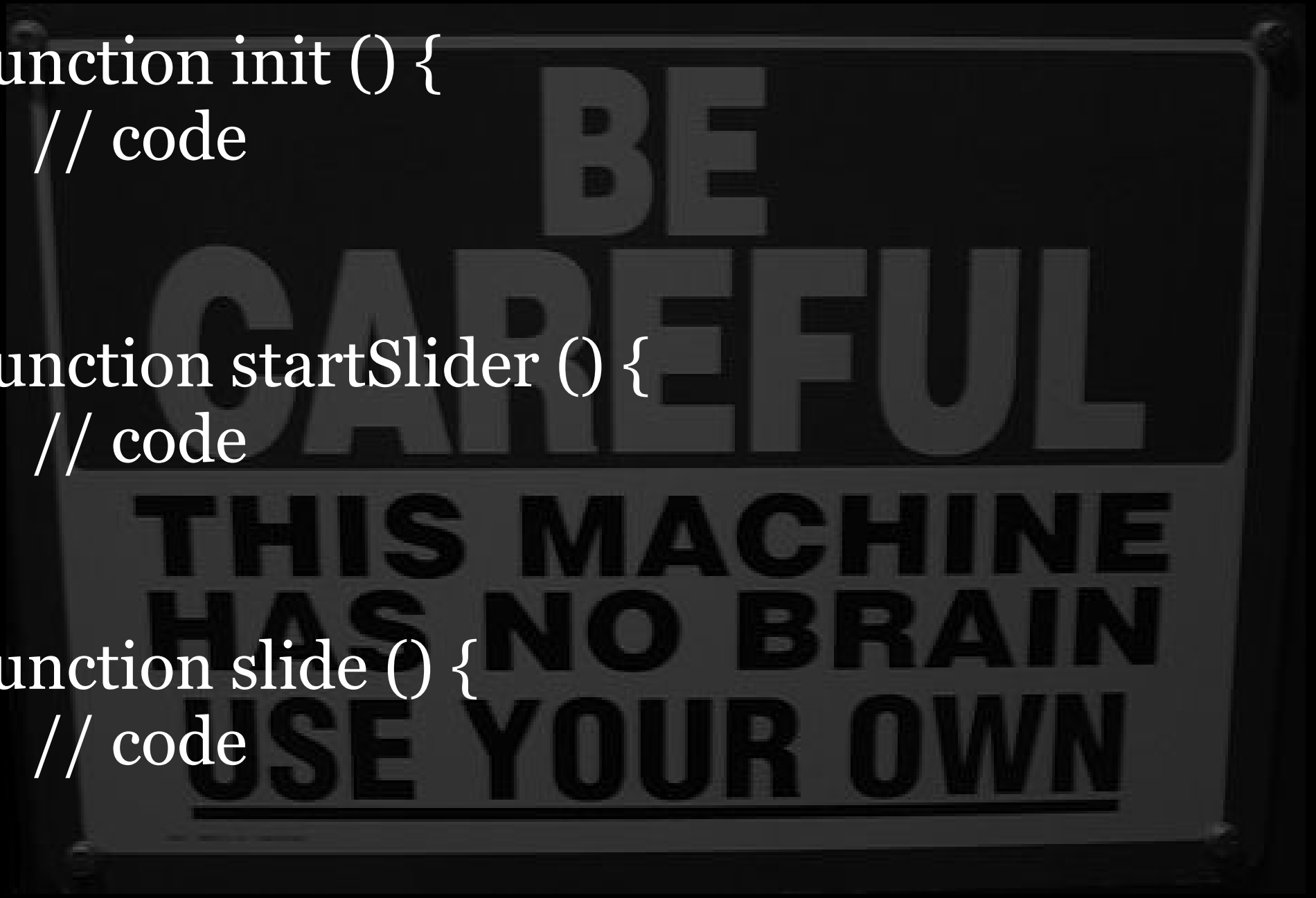
Onzin!

Mijn script is het enige dat mijn klant ooit zal gebruiken.

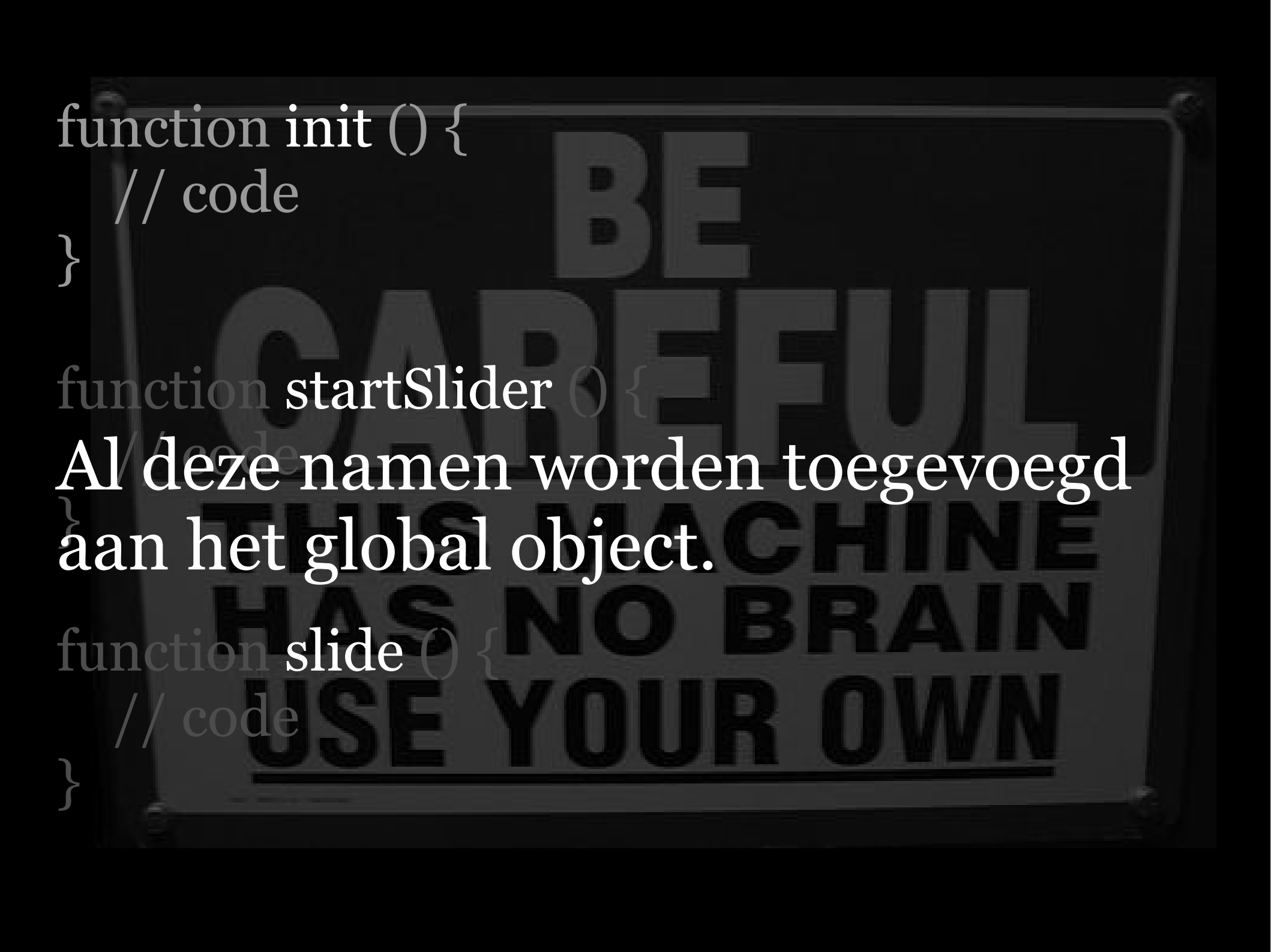
Ze kunnen best later een extra script willen hebben.

En ze kunnen dat best bestellen bij iemand die eigenlijk helemaal geen JavaScript kan.






```
function init () {
    // code
}

function startSlider () {
    // code
}

function slide () {
    // code
}
```



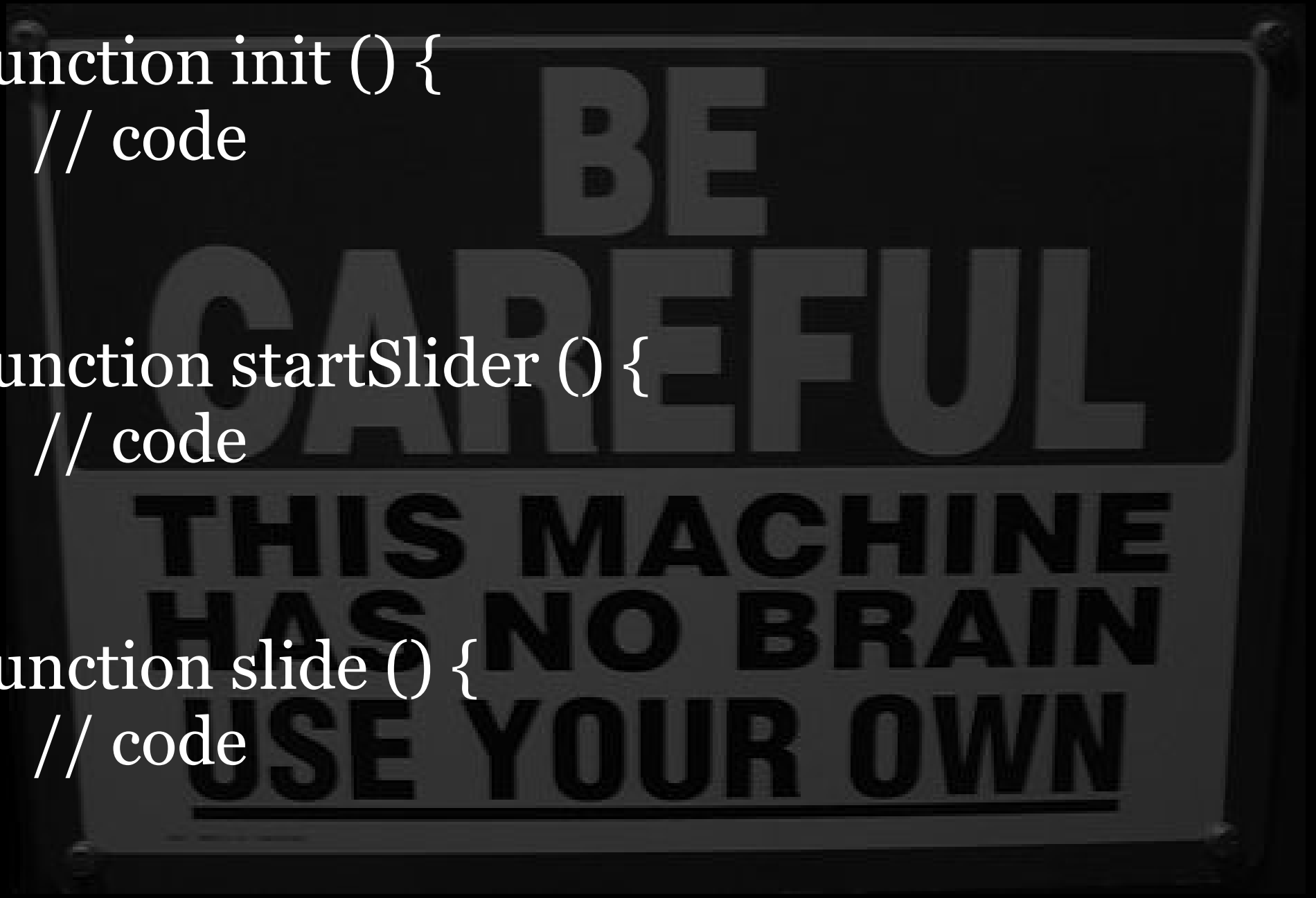

```
function init () {
    // code
}

function startSlider () {
    // code
}

function slide () {
    // code
}
```

Al deze namen worden toegevoegd aan het global object.

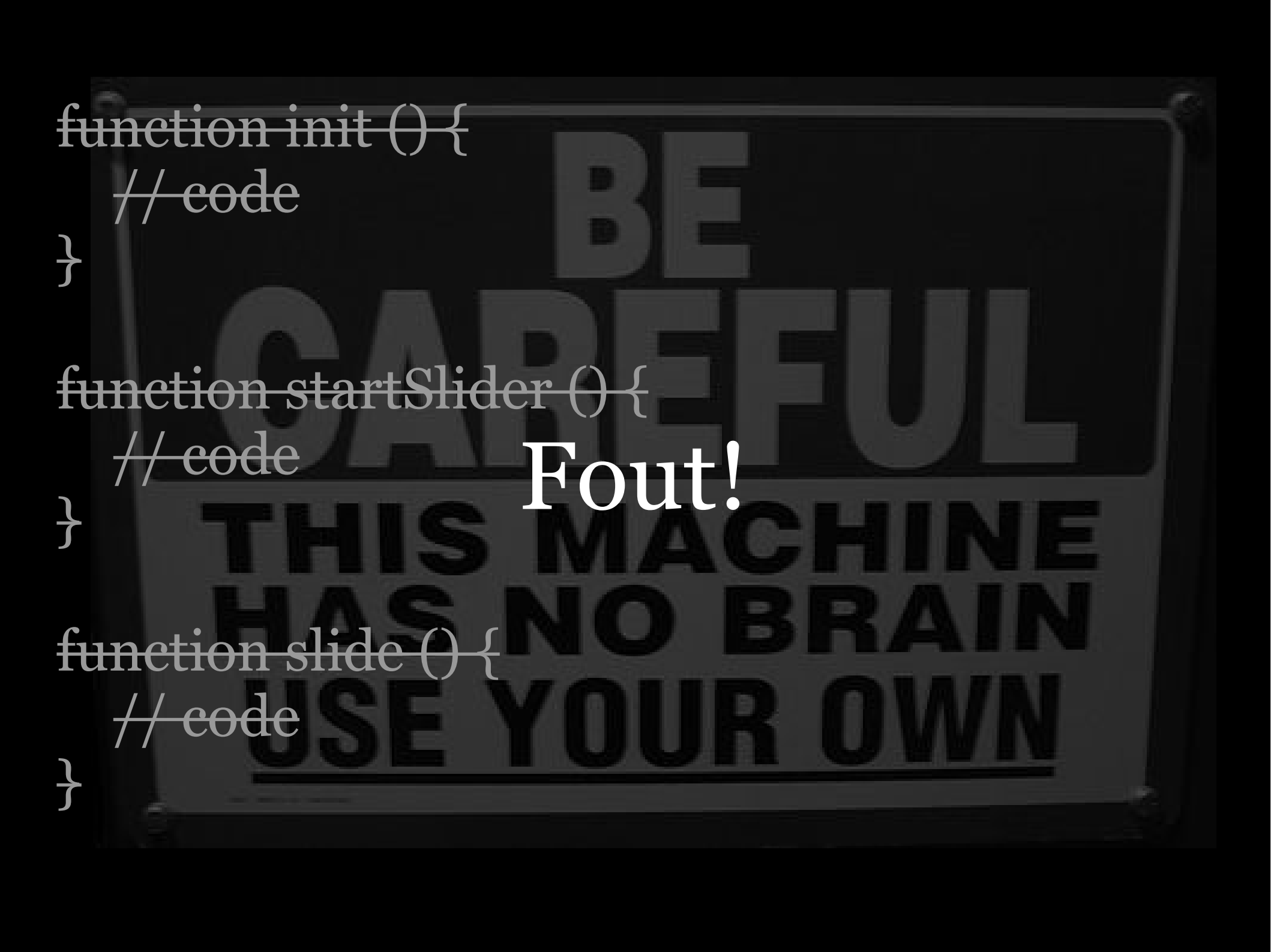
```
function init () {
    // code
}
```

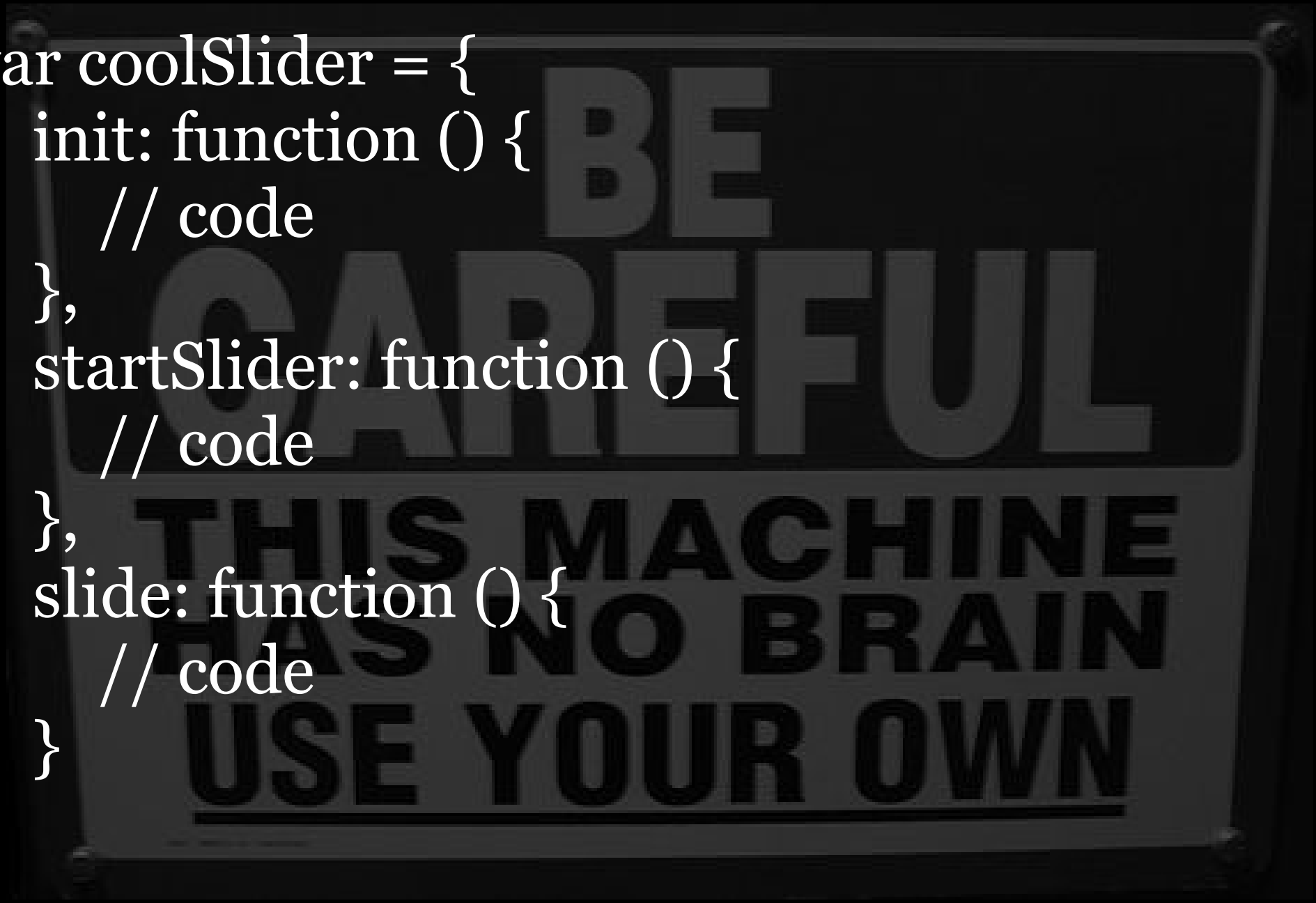
Vooral “init” is een nogal voor de hand liggende naam.

Wat als iemand een andere functie “init” toevoegt?

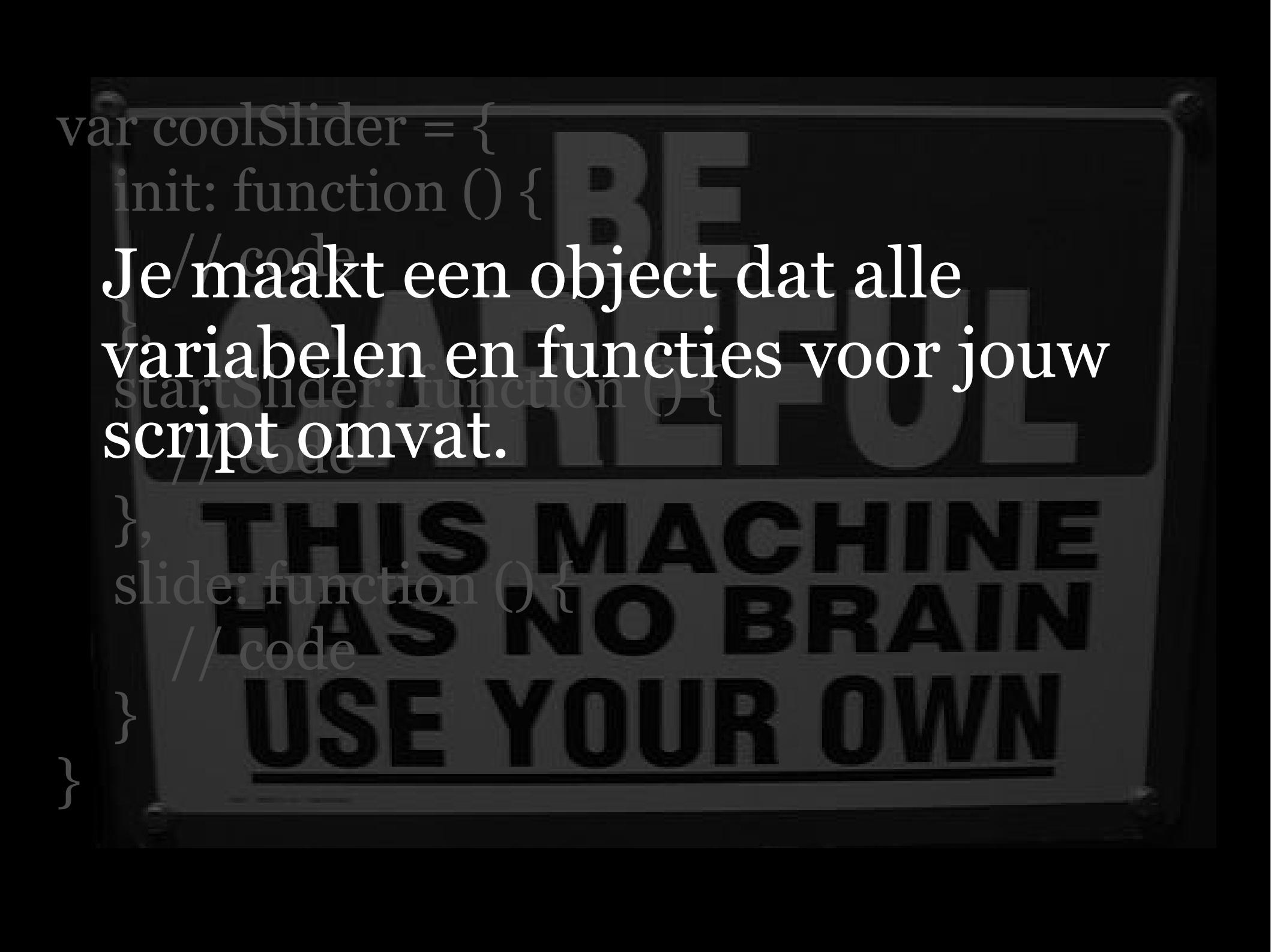



```
function init () {
    // code
}

function startSlider () {
    // code
}

function slide () {
    // code
}
```

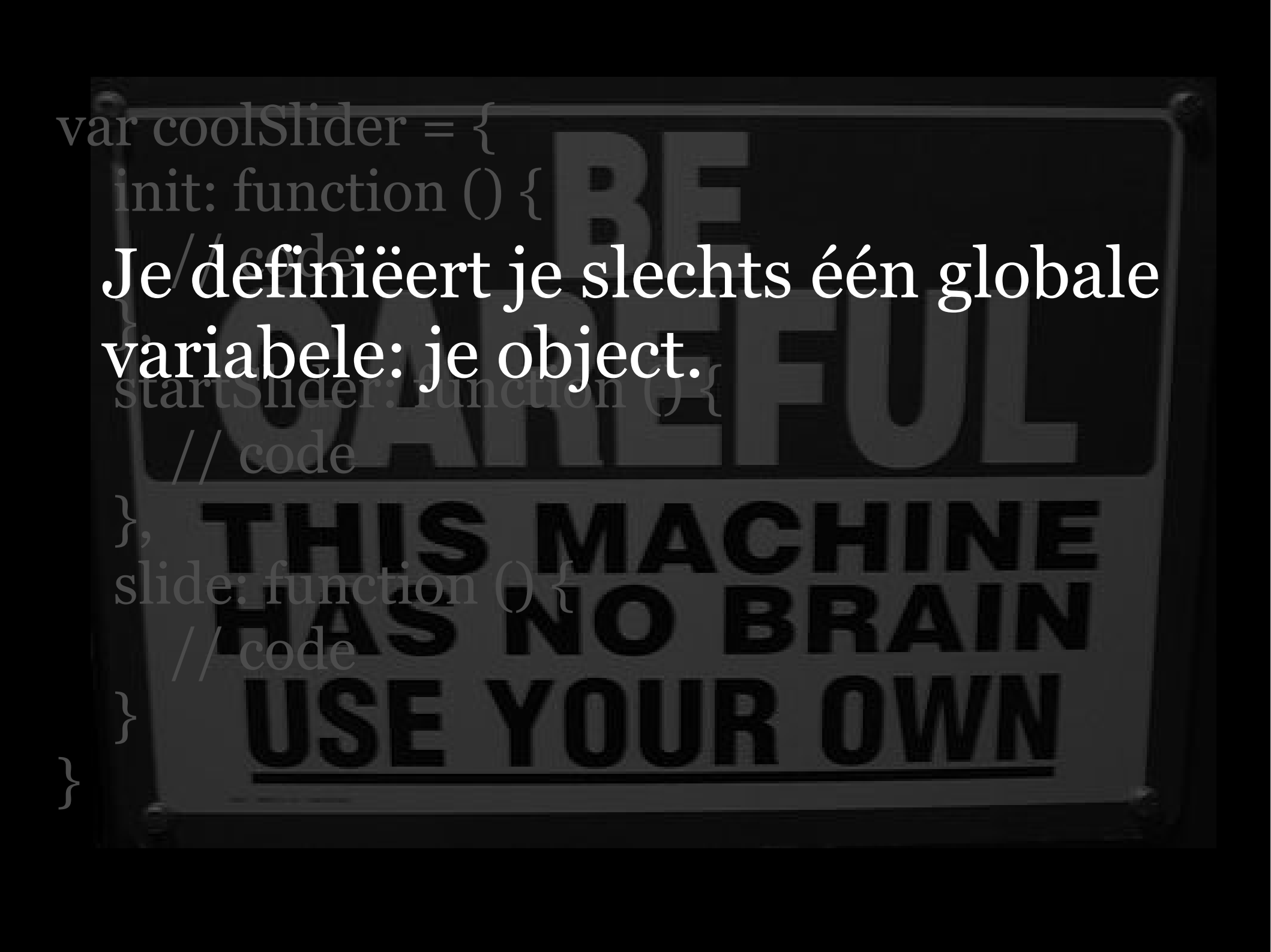

function init () {
// code
}
function startSlider () {
// code
}
function slide () {
// code
}
Fout!
BE
CAREFUL
THIS MACHINE
HAS NO BRAIN
USE YOUR OWN




```
var coolSlider = {
    init: function () {
        // code
    },
    startSlider: function () {
        // code
    },
    slide: function () {
        // code
    }
}
```



Je maakt een object dat alle variabelen en functies voor jouw script omvat.

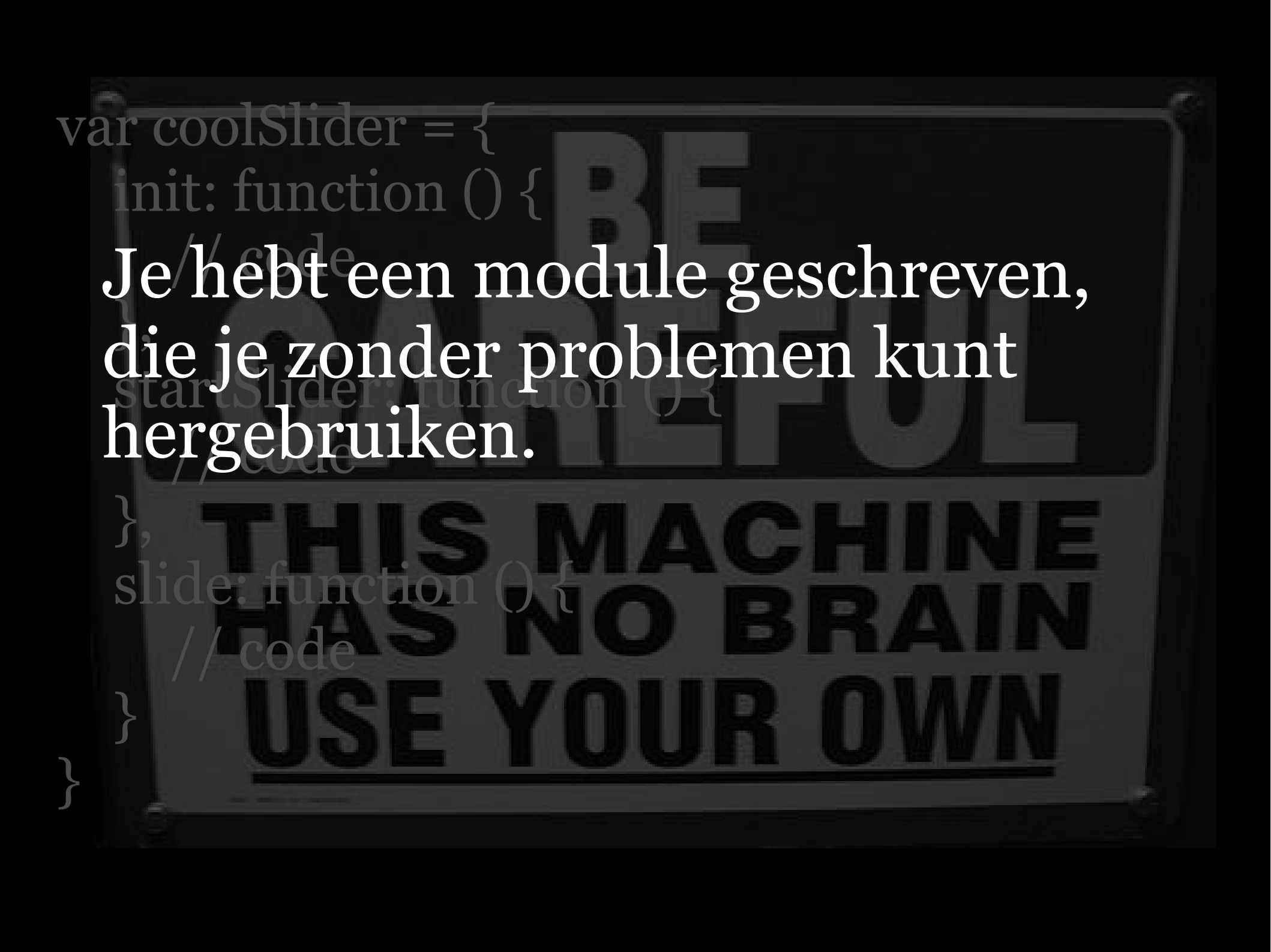


Je definiëert je slechts één globale variabele: je object.

```
var coolSlider = {
  init: function () {
    // code
  },
  startSlider: function () {
    // code
  },
  slideFunction () {
    // code
  }
}
```

Je definiëert je slechts één globale variabele: je object.

Bovendien ligt de naam “coolSlider” minder voor de hand, en zal dus minder snel overschreven worden.



Je hebt een module geschreven, die je zonder problemen kunt hergebruiken.

Dit is geen object-georiënteerd JavaScript.

Het is wel bijzonder handig.

Aanname

Iedereen gebruikt een muis

Onzin!

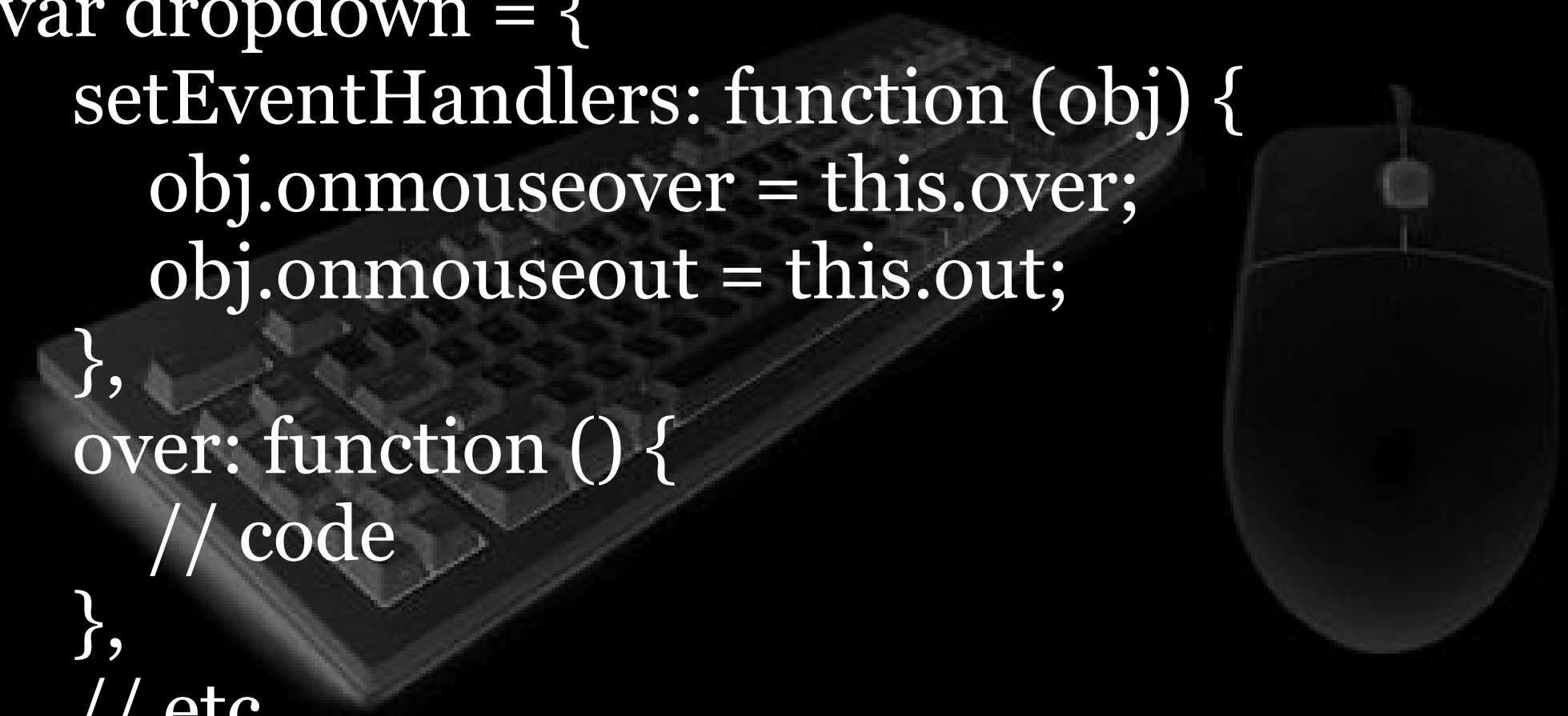
Iedereen gebruikt een muis

Hier moeten we wat dieper op in gaan.


Apparaat
onafhankelijk
heid

Neem een dropdown-menu:

```
var dropdown = {
    setEventHandlers: function (obj) {
        obj.onmouseover = this.over;
        obj.onmouseout = this.out;
    },
    over: function () {
        // code
    },
    // etc
}
```



Dit werkt niet zonder muis.

```
var dropdown = {
    setEventHandlers: function (obj) {
        obj.onmouseover = this.over;
        obj.onmouseout = this.out;
    },
    over: function () {
        // code
    },
    // etc
}
```

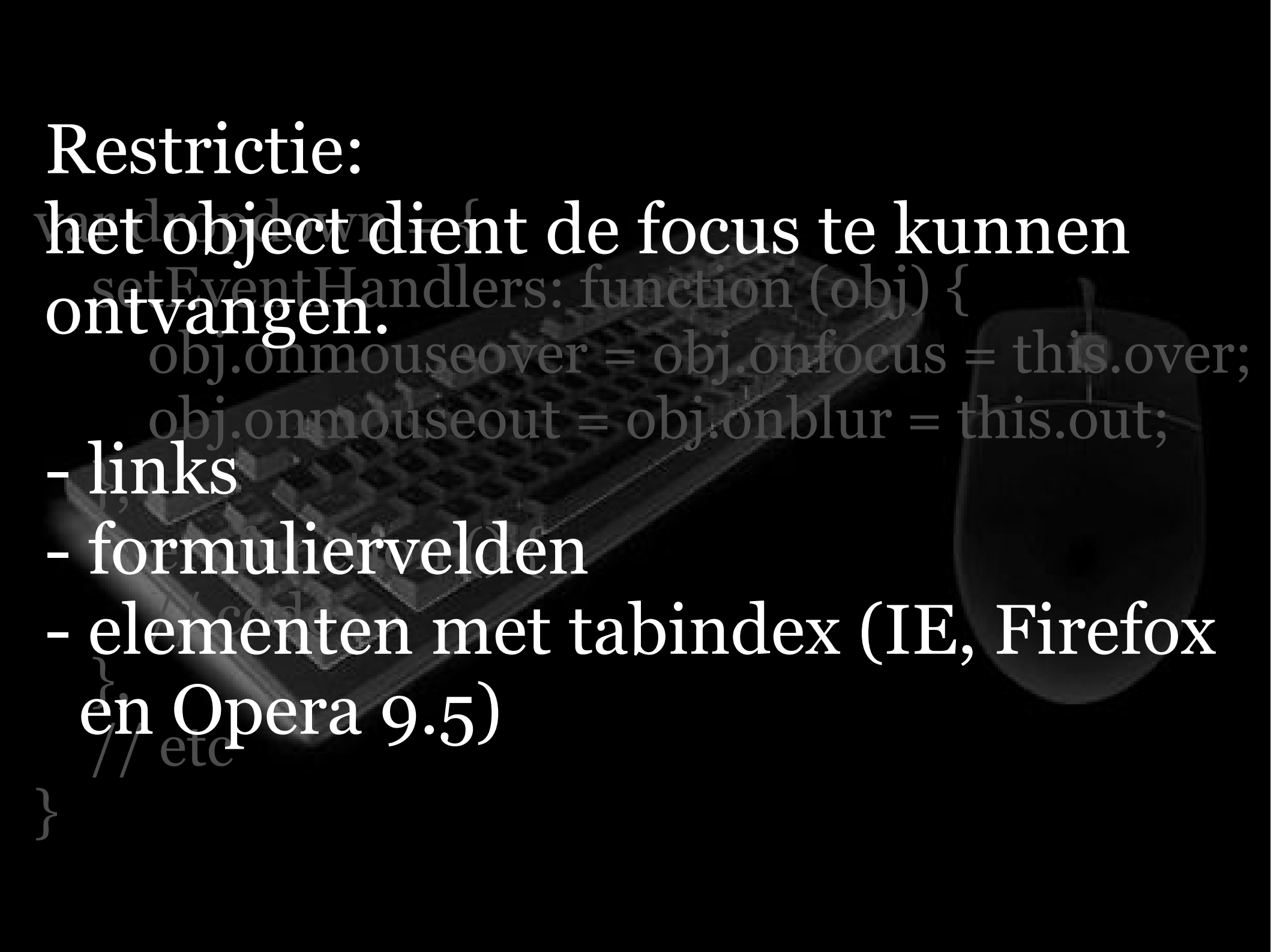
We hebben events nodig die aangeven of de gebruiker met het toetsenbord een link “binnenkomt” of verlaat.

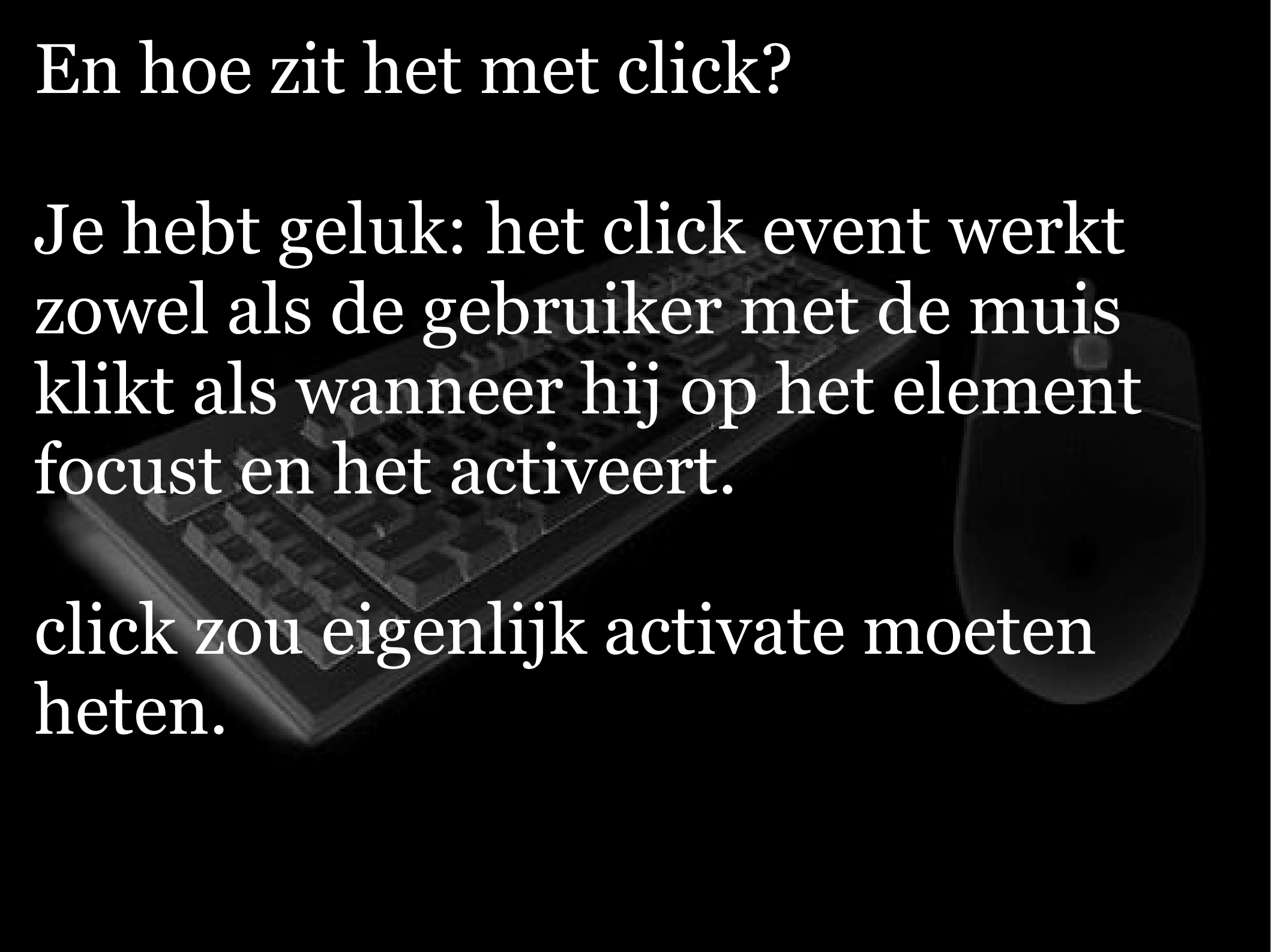
focus en blur

```
var dropdown = {
    setEventHandlers: function (obj) {
        obj.onmouseover = obj.onfocus = this.over;
        obj.onmouseout = obj.onblur = this.out;
    },
    over: function () {
        // code
    },
    // etc
}
```

Restrictie:
het object dient de focus te kunnen ontvangen.

- links
- formuliervelden
- elementen met tabindex (IE, Firefox en Opera 9.5)





En hoe zit het met click?

Je hebt geluk: het click event werkt zowel als de gebruiker met de muis klikt als wanneer hij op het element focust en het activeert.

click zou eigenlijk activate moeten heten.

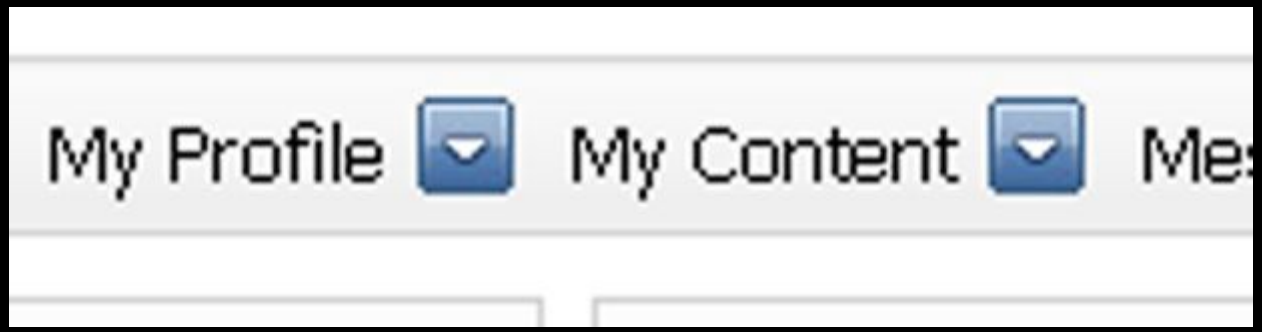
En hoe zit het met click?

Je hebt geluk: het click event werkt zowel als de gebruiker met de muis klikt als wanneer hij op het element focust en het activeert.

Restrictie: het element dient de focus te kunnen ontvangen.

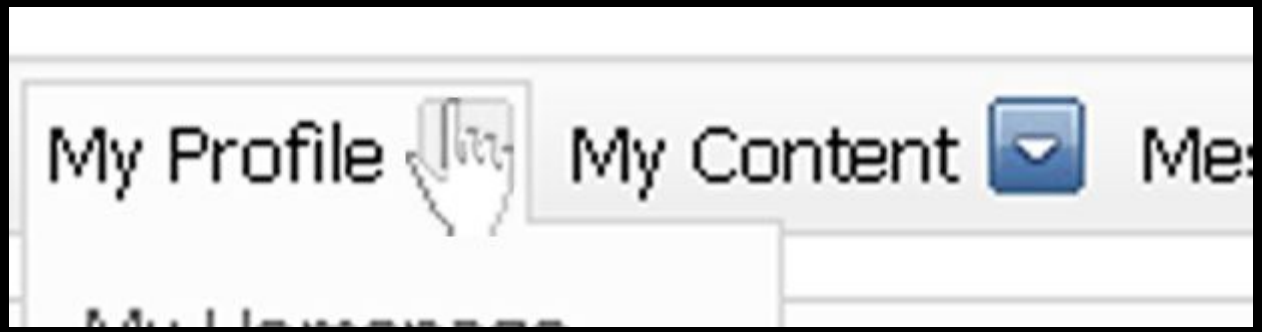
Click als activate

pijltje.onclick = toonMenu;



Click als activate

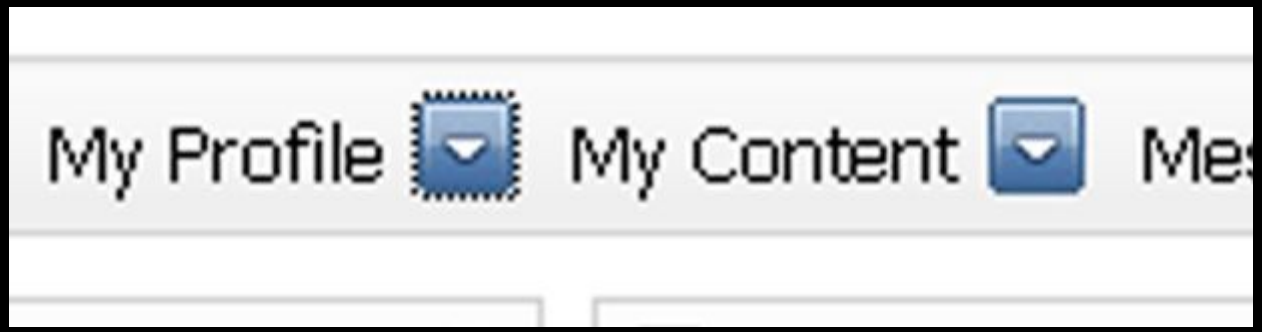
pijltje.onclick = toonMenu;



1) Een muisklik op het pijltje

Click als activate

pijltje.onclick = toonMenu;



1) Een muisklik op het pijltje
2) a) Toetsenbord-focus op het pijltje

Click als activate

pijltje.onclick = toonMenu;

1) Een muisklik op het pijltje
2) a) Toetsenbord-focus op het pijltje
   b) Spatiebalk op het pijltje

Dat zijn twee acties. Is dat erg?

Click als activate

pijltje.onclick = pijltje.onfocus = toonMenu;

1) Een muisklik op het pijltje
2) Toetsenbord-focus op het pijltje
~~b) Spatiebalk op het pijltje~~

Click als activate

pijltje.onclick = pijltje.onfocus = toonMenu;

1) Een muisklik op het pijltje
2) Toetsenbord-focus op het pijltje

Maar: de volgende tab gaat dan altijd het menu in. De gebruiker kan het menu niet overslaan.

Click als activate

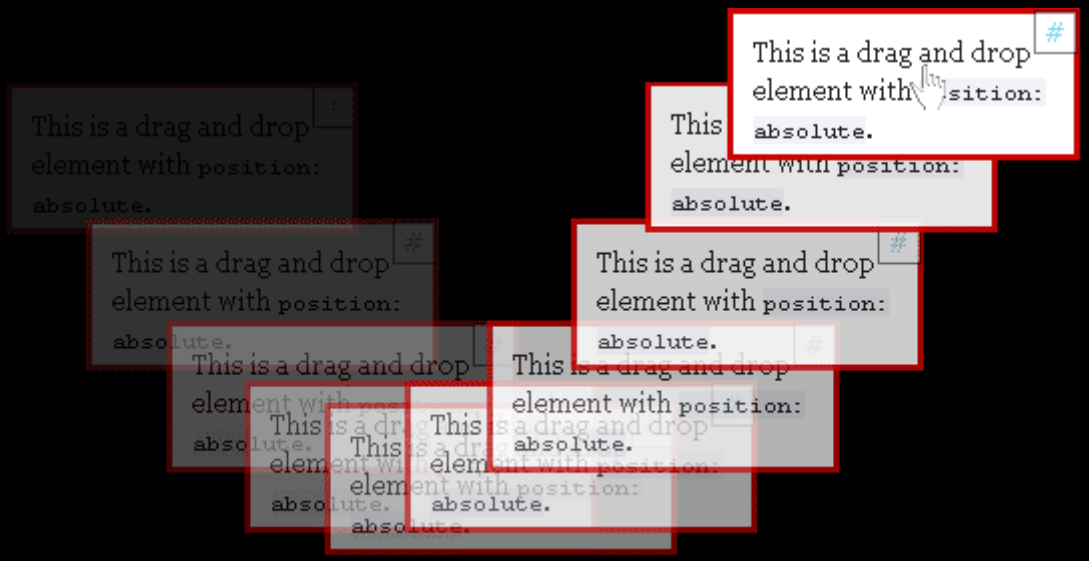
pijltje.onclick = ~~pijltje.onfocus =~~ toonMenu;

Over het algemeen hebben toetsenbord-gebruikers meer acties nodig om een bepaald doel te bereiken dan muisgebruikers.

Grijp niet teveel in. Er zijn redenen voor, en mensen zijn er aan gewend.

Twee aparte scripts

Drag-and-drop
werkt met het mousemove event



Twee aparte scripts

Drag-and-drop
werkt met het mousemove event

en als er iets is wat niet met een toetsenbord valt na te doen

is het wel het bewegen van de muis

Twee aparte scripts

Drag-and-drop
werkt met het mousemove event

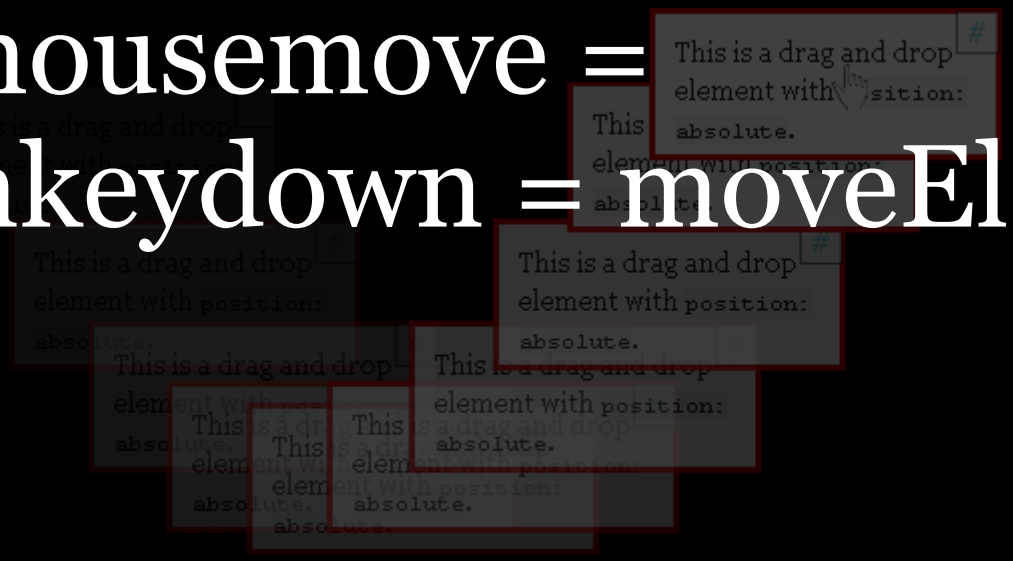
Hoe maken we het toegankelijk?

Door de gebruiker toe te staan de pijltjestoetsen te gebruiken.
Key events.

Twee aparte scripts

Drag-and-drop

```
obj.onmousemove =
   obj.onkeydown = moveElement;
```



Twee aparte scripts

Drag-and-drop

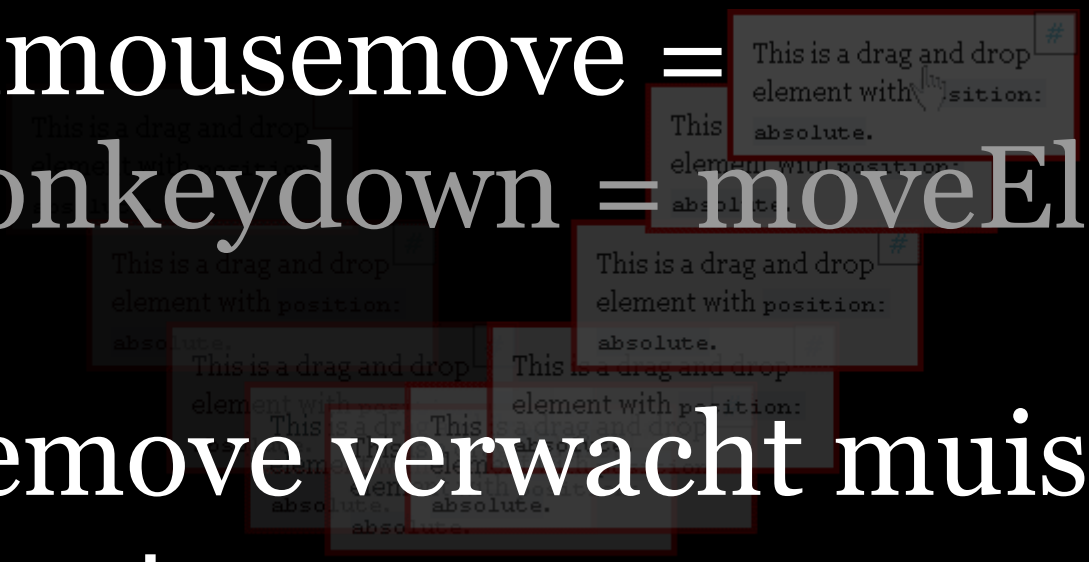
~~obj.onmousemove =~~
~~obj.onkeydown = moveElement;~~

Werkt totaal niet.

Twee aparte scripts

Drag-and-drop

```
obj.onmousemove =
   obj.onkeydown = moveElement;
```

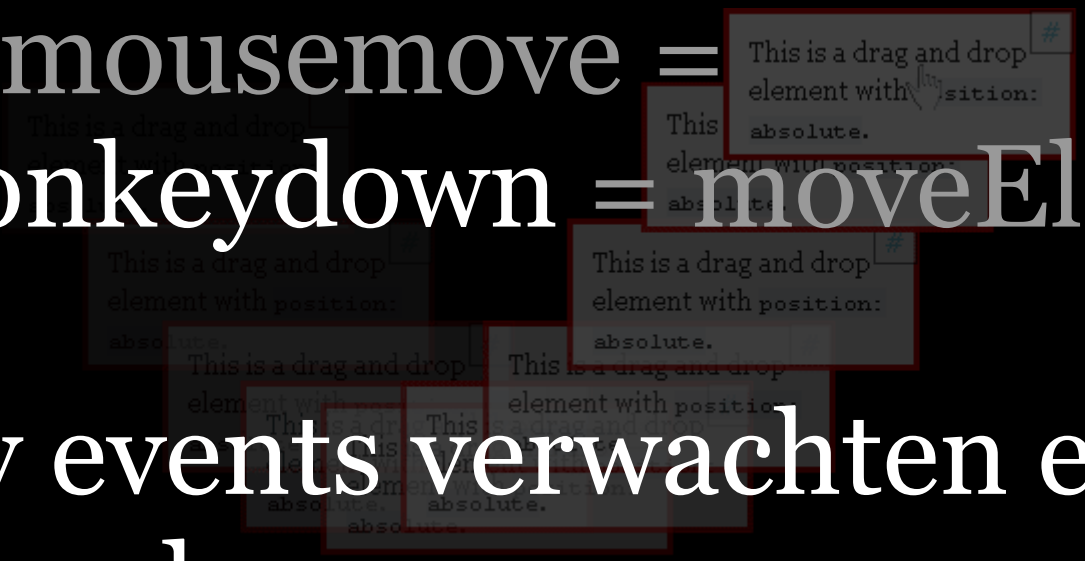


Mousemove verwacht muis-coordinaten.

Twee aparte scripts

Drag-and-drop

obj.onmousemove =
obj.onkeydown = moveElement;

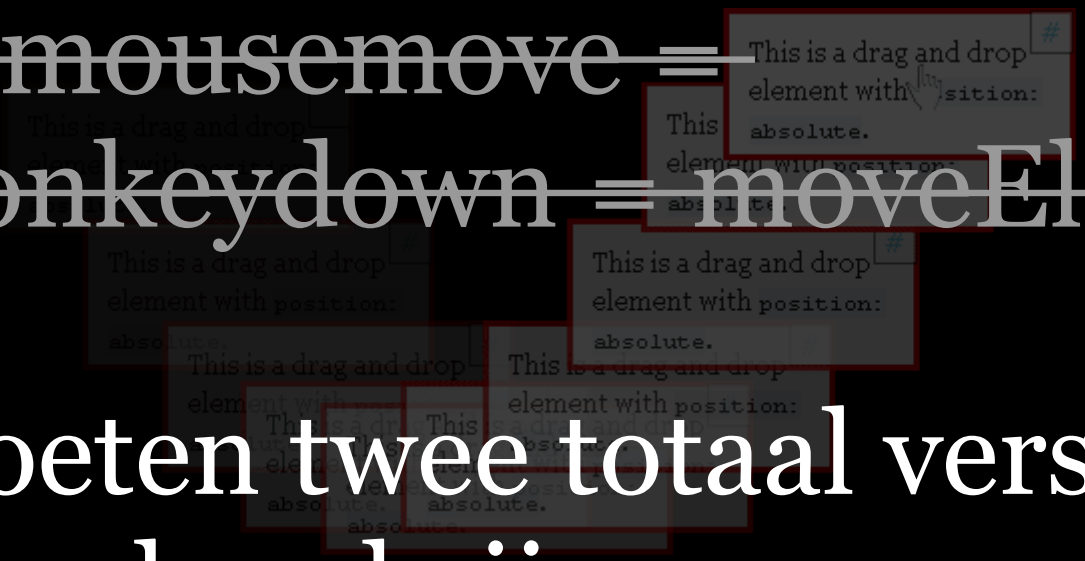


De key events verwachten een toetsaanslag.

Twee aparte scripts

Drag-and-drop

~~obj.onmousemove =~~
~~obj.onkeydown = moveElement;~~



We moeten twee totaal verschillende situaties beschrijven.

Twee aparte scripts

Drag-and-drop

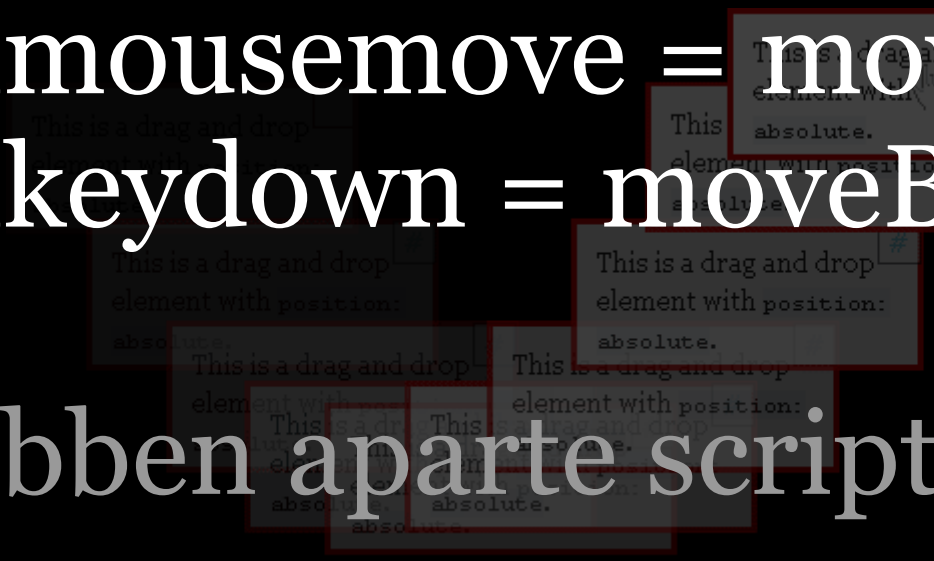
~~obj.onmousemove =~~
~~obj.onkeydown = moveElement;~~

We hebben aparte scripts nodig.

Twee aparte scripts

Drag-and-drop

```
obj.onmousemove = moveByMouse;
obj.onkeydown = moveByKeys;
```

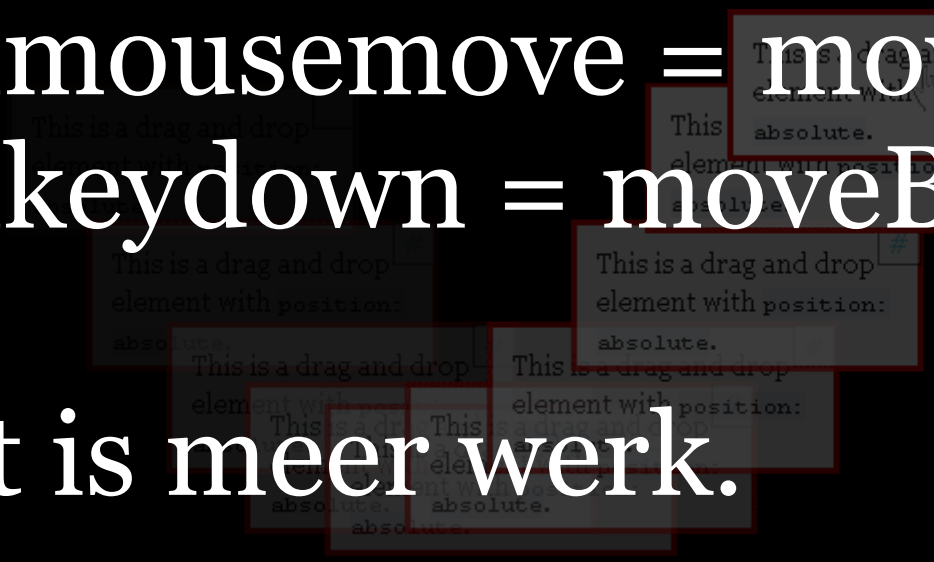


We hebben aparte scripts nodig.

Twee aparte scripts

Drag-and-drop

```
obj.onmousemove = moveByMouse;
obj.onkeydown = moveByKeys;
```



Ja, dat is meer werk.

Twee aparte scripts

Drag-and-drop

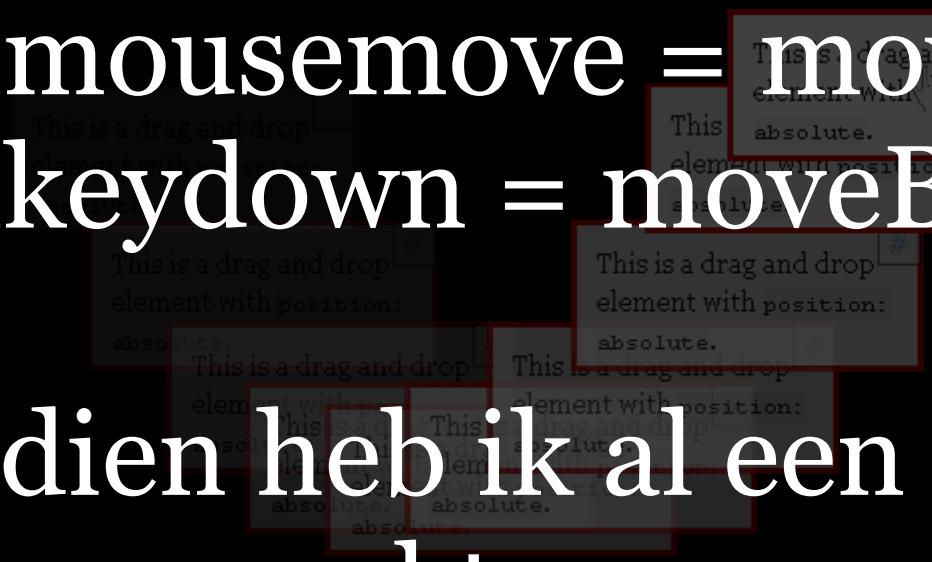
obj.onmousemove = moveByMouse;
obj.onkeydown = moveByKeys;

Maar als je het goed aanpakt, heb je direct een toegankelijke drag and drop module.

Twee aparte scripts

Drag-and-drop

```
obj.onmousemove = moveByMouse;
obj.onkeydown = moveByKeys;
```



Bovendien heb ik al een eerste opzet voor je gemaakt.

Twee aparte scripts

Drag-and-drop

http://quirksmode.org/
js/dragdrop.html

Bovendien heb ik al een eerste opzet voor je gemaakt.

Unobtrusive JavaScript

Twee basisprincipes:

1) Scheiding van structuur, presentatie en gedrag
2) Het script maakt geen annames

Unobtrusive JavaScript

Niet eens zo moeilijk

Hulp nodig?

Chris Heilmann:

http://onlinetools.org/articles/unobtrusivejavascript/
http://icant.co.uk/articles/seven-rules-of-unobtrusive-javascript/

Jeremy Keith:

http://www.alistapart.com/articles/behavioralseparation

Het moet ook van de overheid:

http://www.webrichtlijnen.nl/handleiding/ontwikkeling/productie/client-side-script-dom/optionele-karakter/

en natuurlijk quirksmode.org


Vragen?