Hell is other browsers - *Sartre*

The principles of unobtrusive JavaScript

Peter-Paul Koch (ppk)
http://quirksmode.org
An Event Apart Boston, June 24th, 2008

Unobtrusive JavaScript

Wikipedia:
“an emerging paradigm in the JavaScript programming language.”

Me:
it's just a good idea.

Unobtrusive JavaScript

It's not a technique

It's more like a philosophy
for using JavaScript in its context:

usable, accessible, standards-
compliant web pages

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything

Unobtrusive JavaScript

Two fundamental principles:

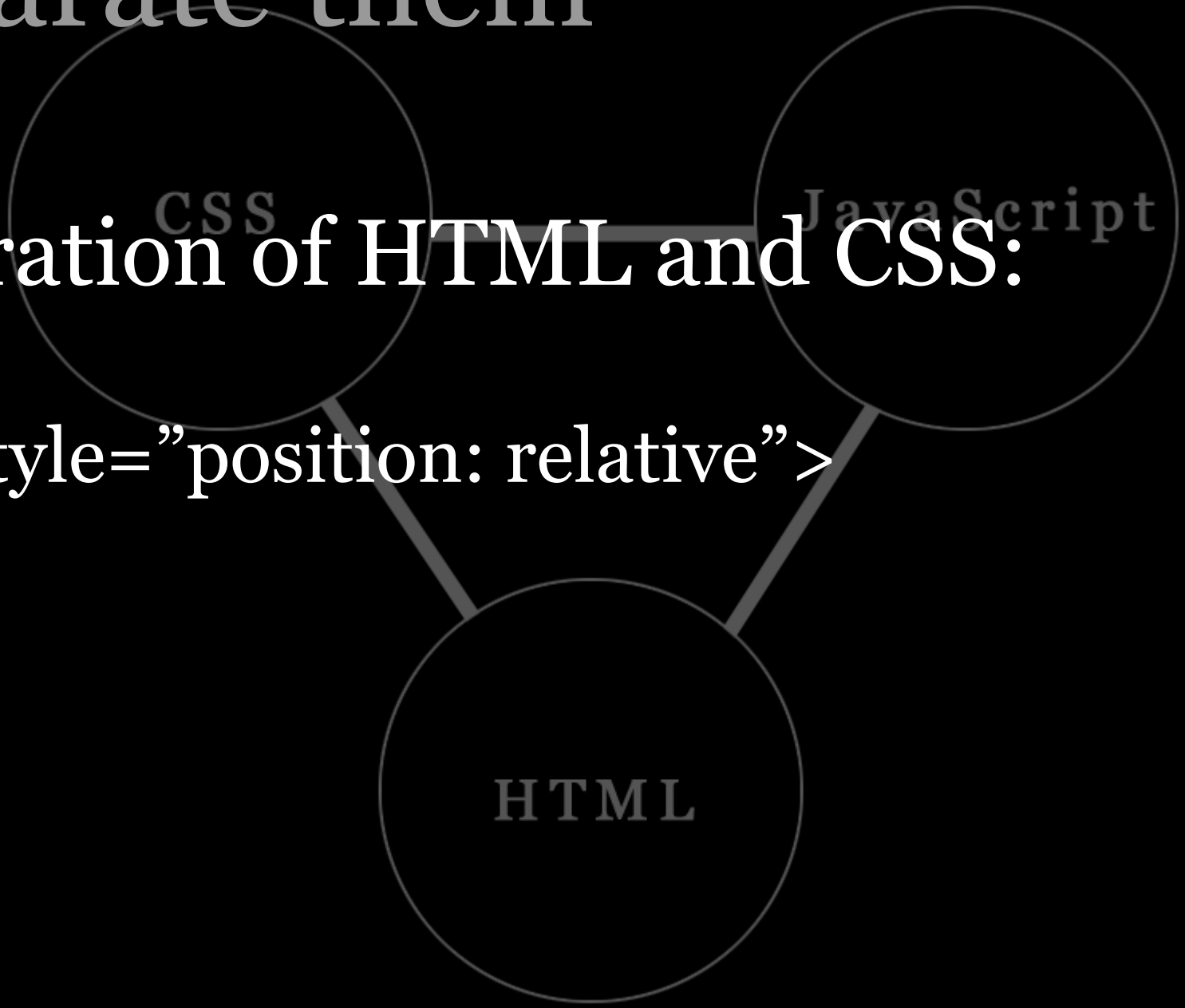
1) Separation of structure, presentation, and behavior
   - Separate them
   - Connect them

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
   - Separate them
   - Connect them

Separate them



Separation of HTML and CSS:

<div style=”position: relative”>

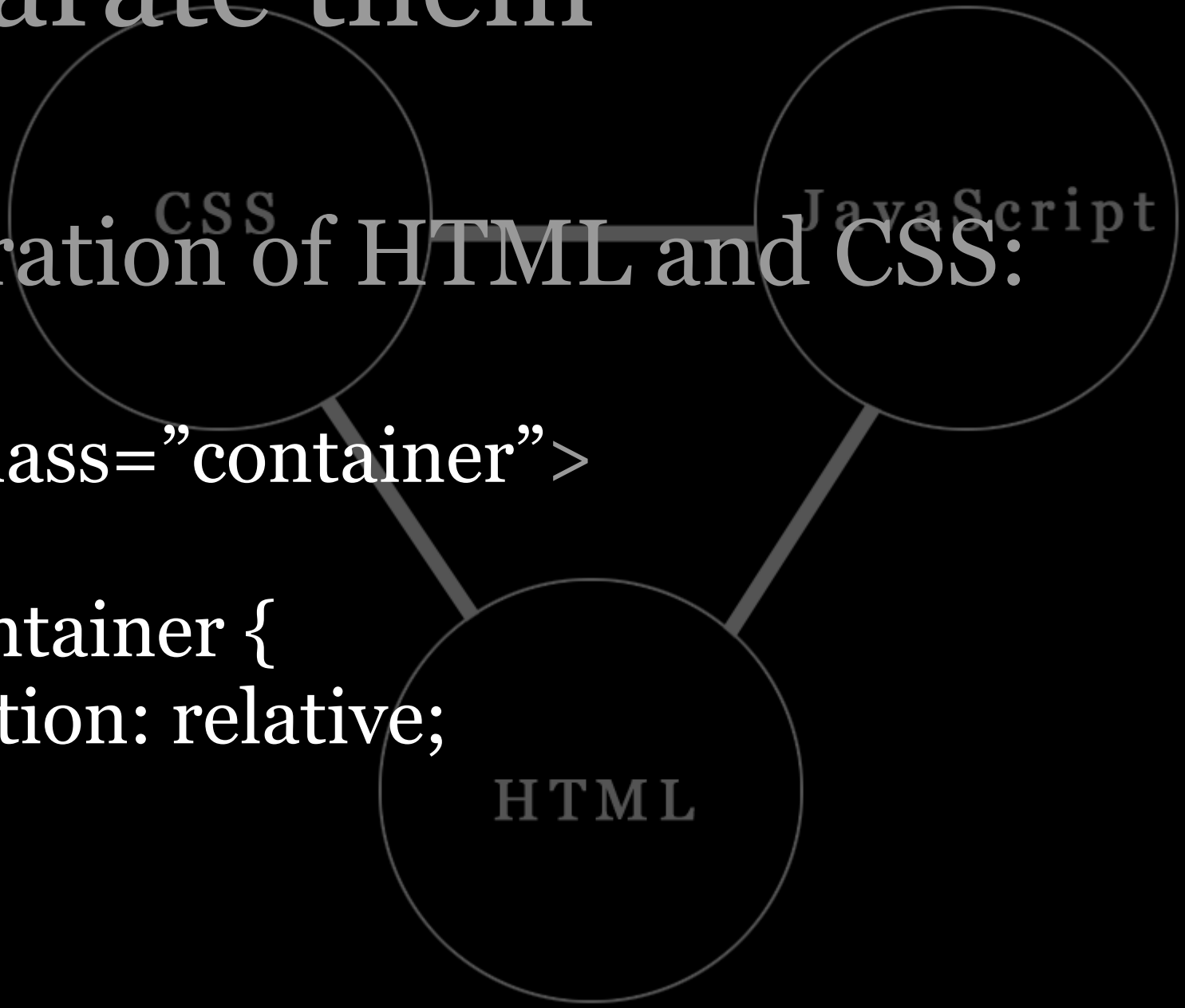
Separate them

Separation of HTML and CSS:

<div ~~style="position: relative"~~>

No inline styles!

Separate them



Separation of HTML and CSS:

```
<div class=”container”>

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

Separate them

Separation of HTML and JavaScript:

<input onmouseover=”doSomething()” />

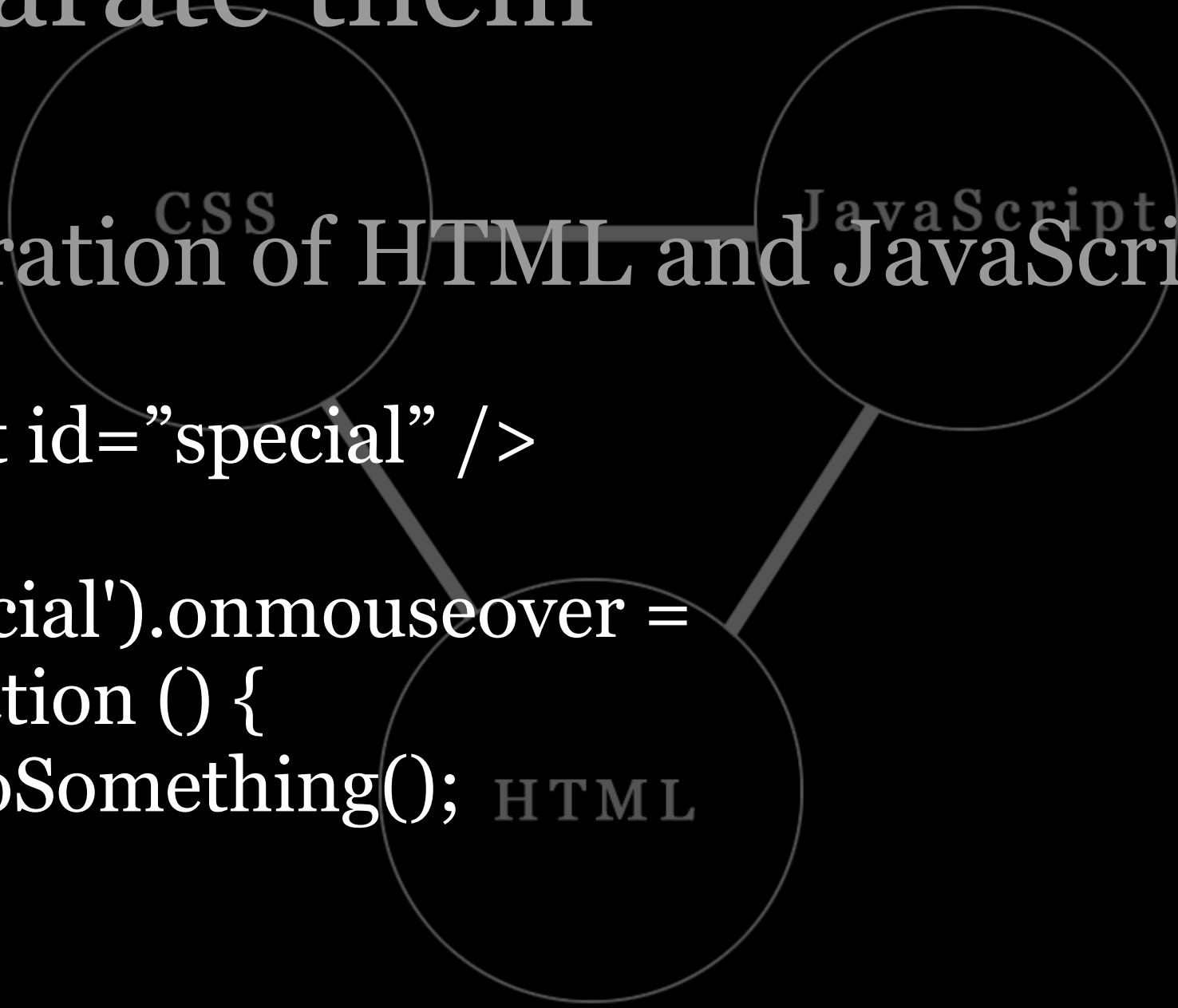
Separate them

Separation of HTML and JavaScript:

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

No inline event handlers!

Separate them



Separation of HTML and JavaScript:

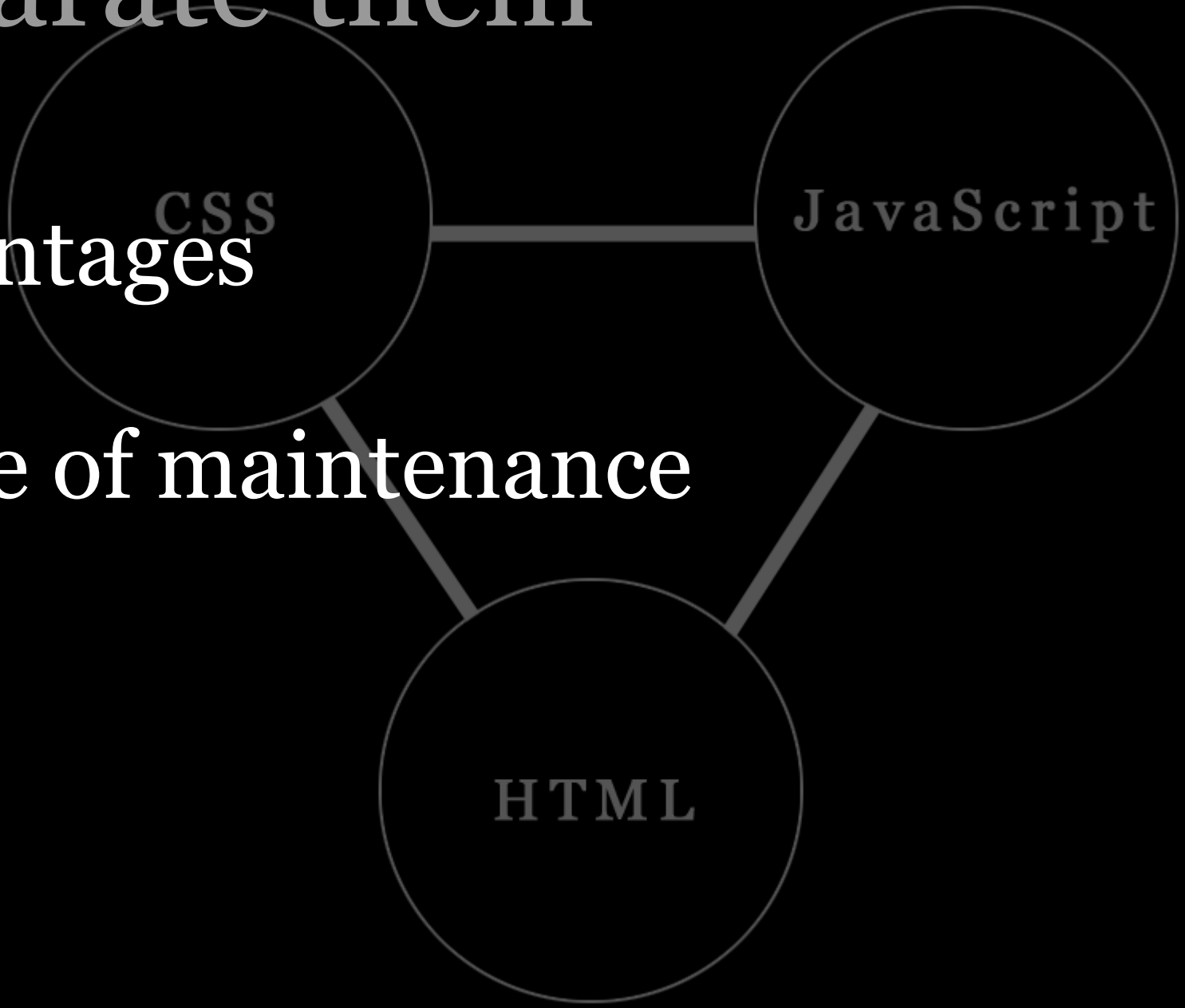
```
<input id=”special” />

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

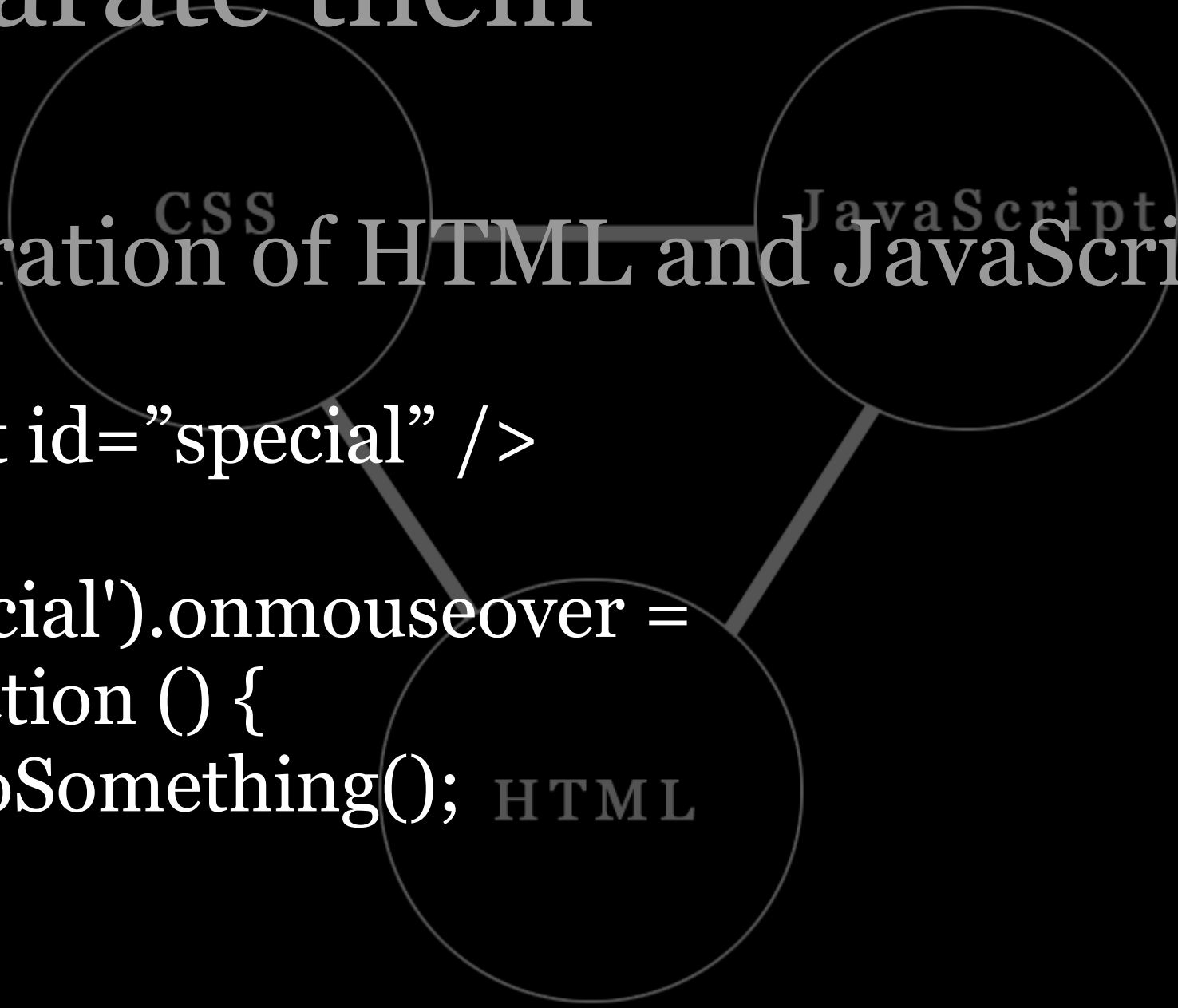
Separate them

Advantages

- Ease of maintenance



Separate them

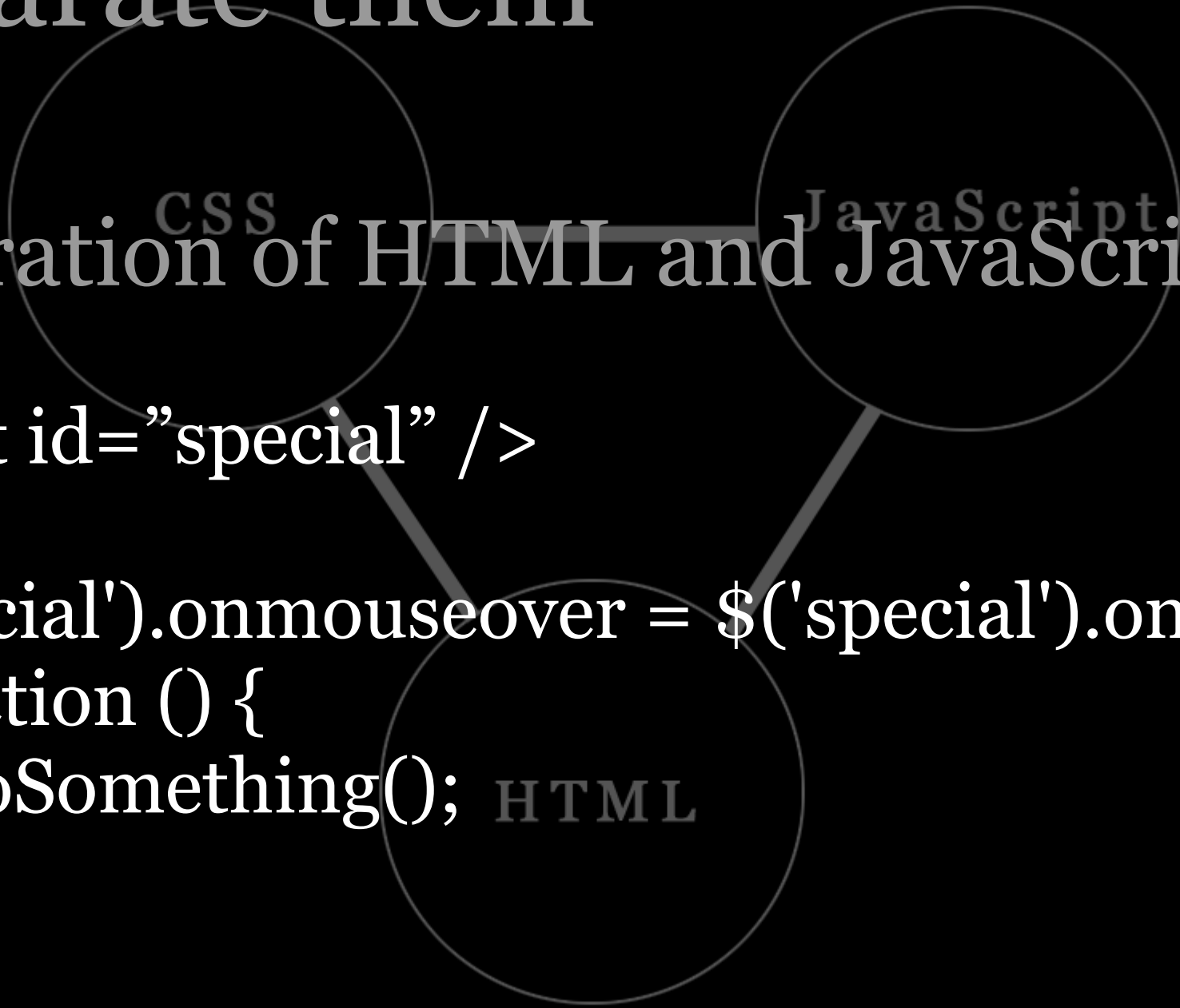


Separation of HTML and JavaScript:

```
<input id=”special” />

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

Separate them

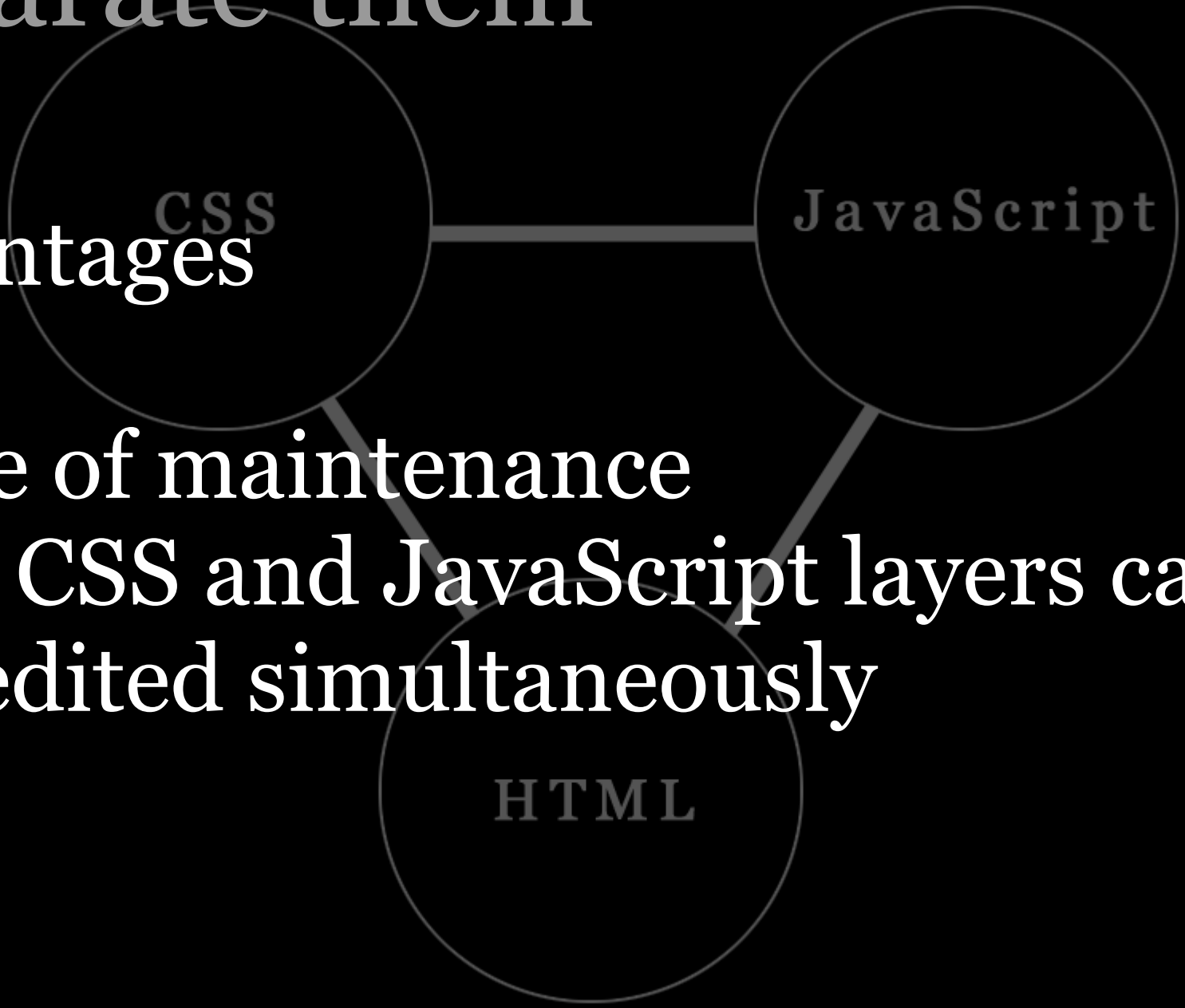


Separation of HTML and JavaScript:

```
<input id=”special” />

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

Separate them



Advantages

- Ease of maintenance
- The CSS and JavaScript layers can be edited simultaneously

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
   - Separate them
   - Connect them

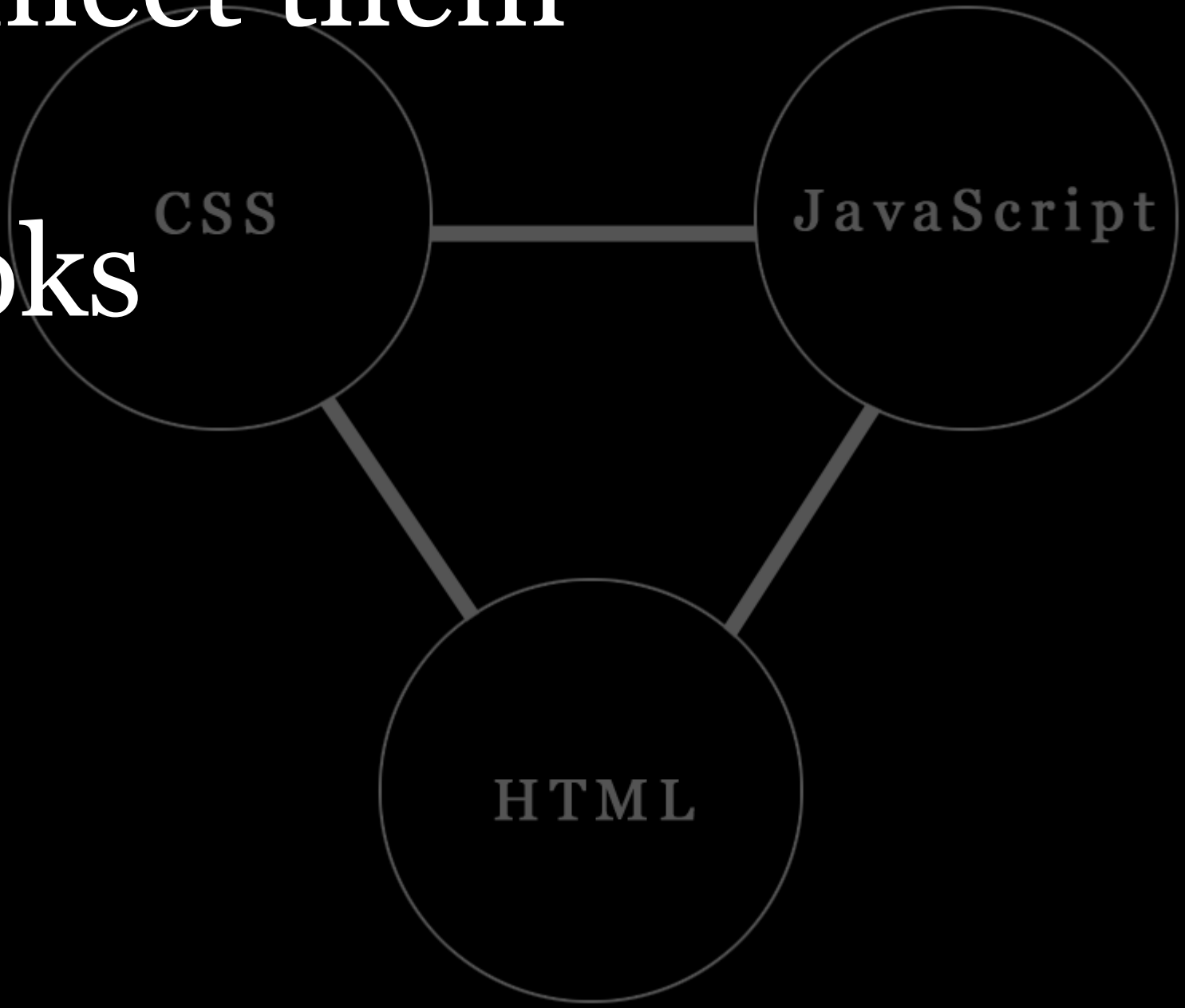
Unobtrusive JavaScript

Two fundamental principles:

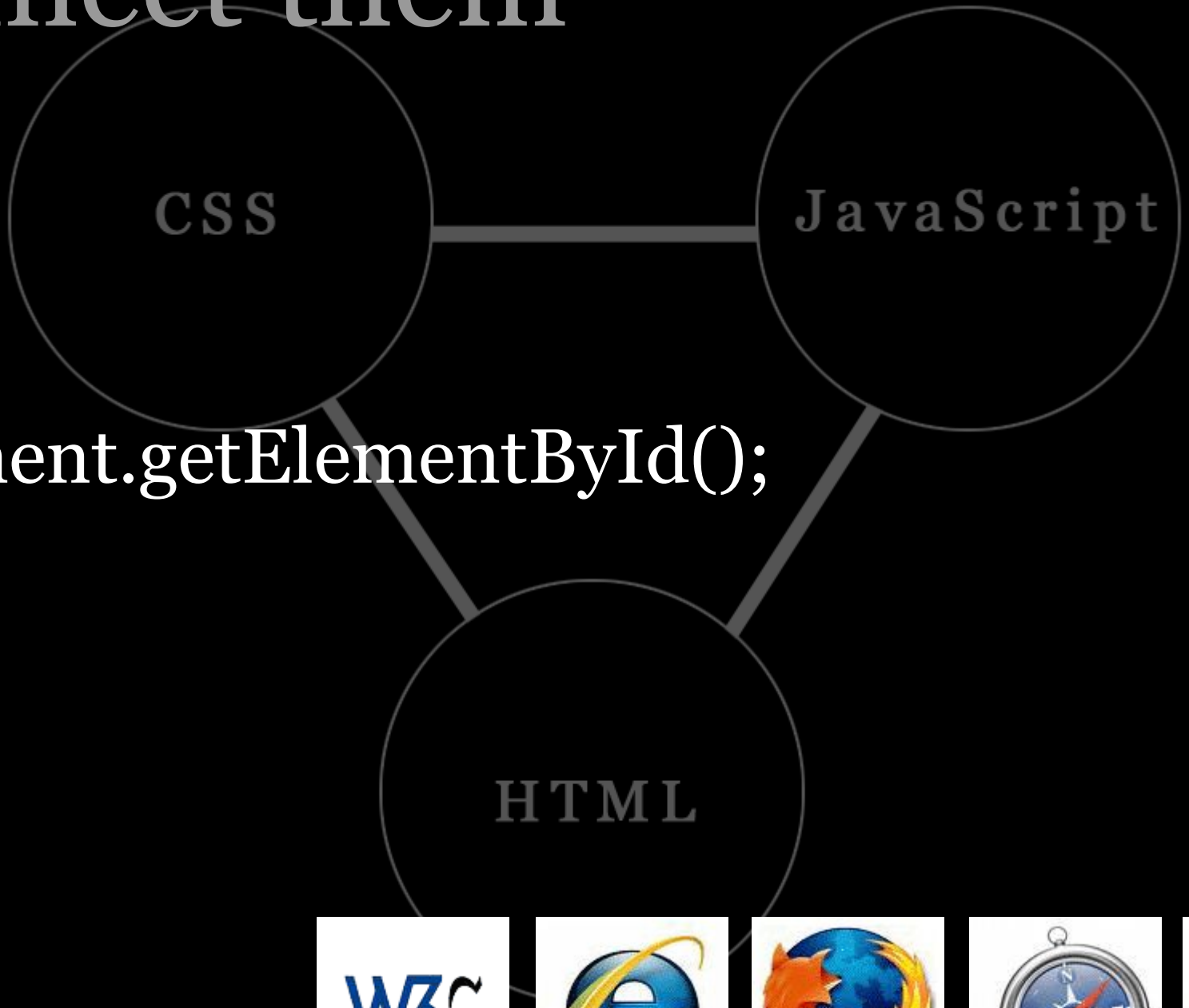
1) Separation of structure, presentation, and behavior
   - Separate them
   - Connect them

Connect them

Hooks



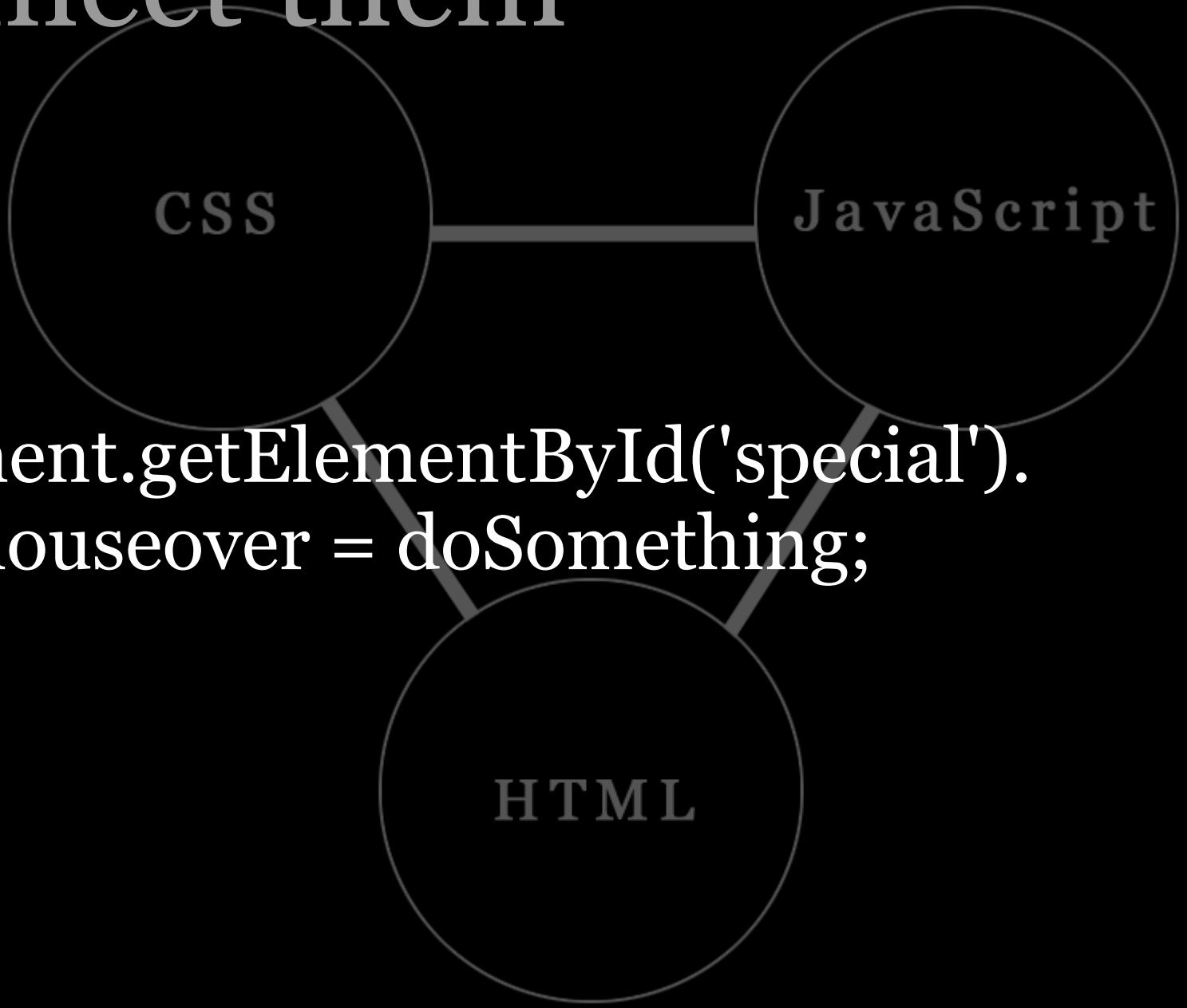
Connect them



- id

document.getElementById();

Connect them



- id

```
document.getElementById('special').
    onmouseover = doSomething;
```

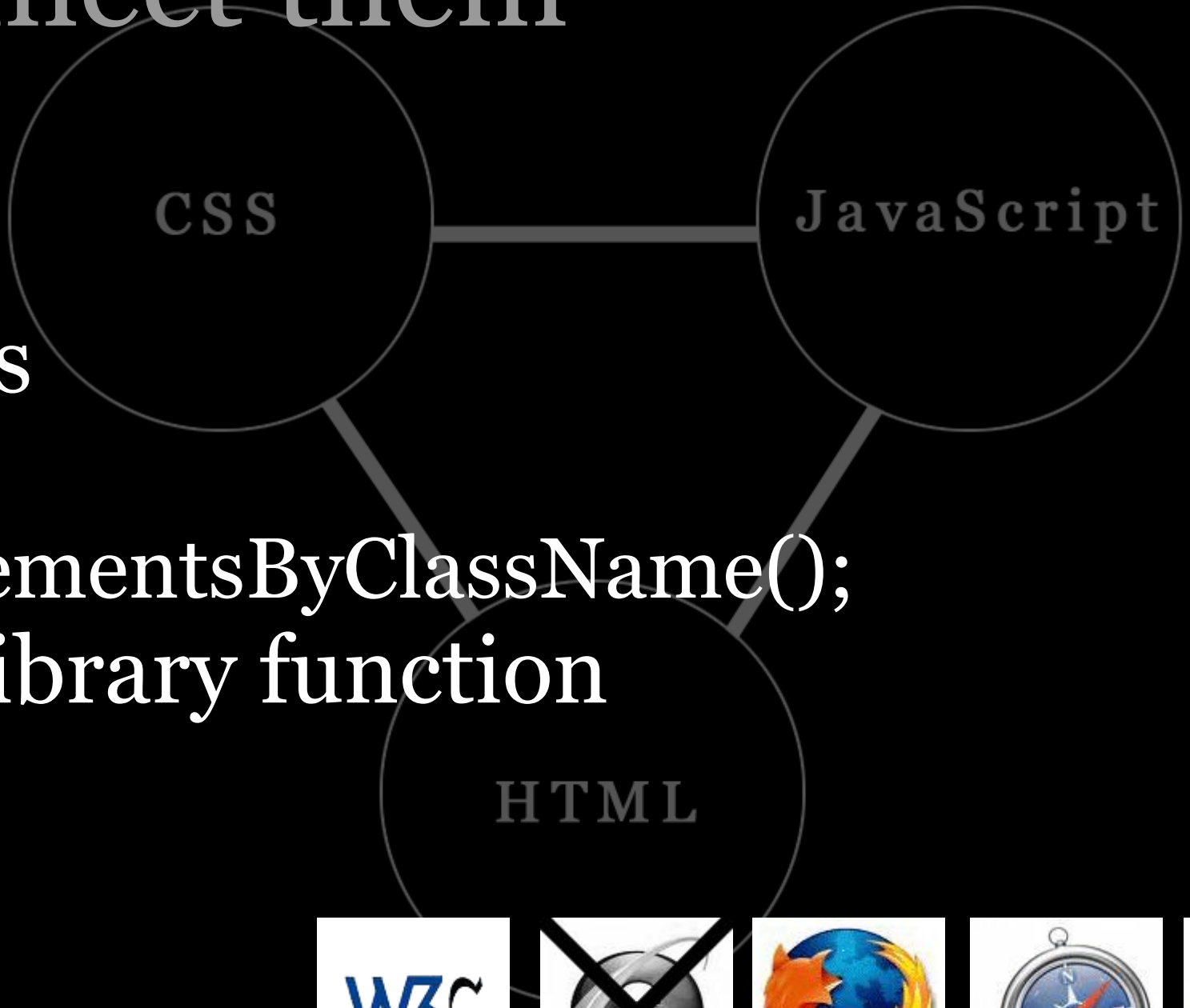
Connect them

- id

```
var el = document.getElementById('special');
if (el) {
    el.onmouseover = doSomething;
}
```

“Is this element available?”

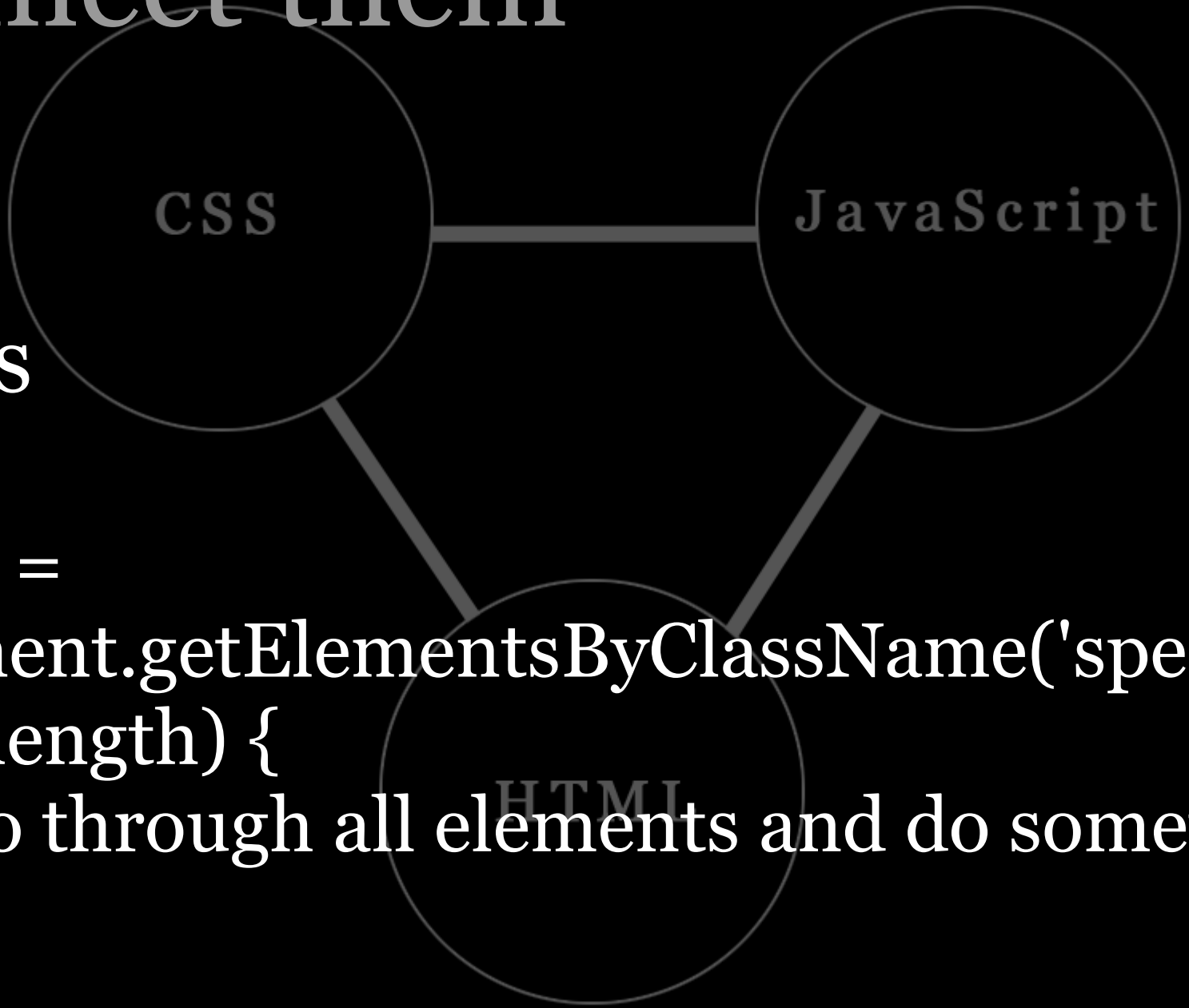
Connect them



- id
- class

getElementsByClassName();
or a library function

Connect them



- id
- class

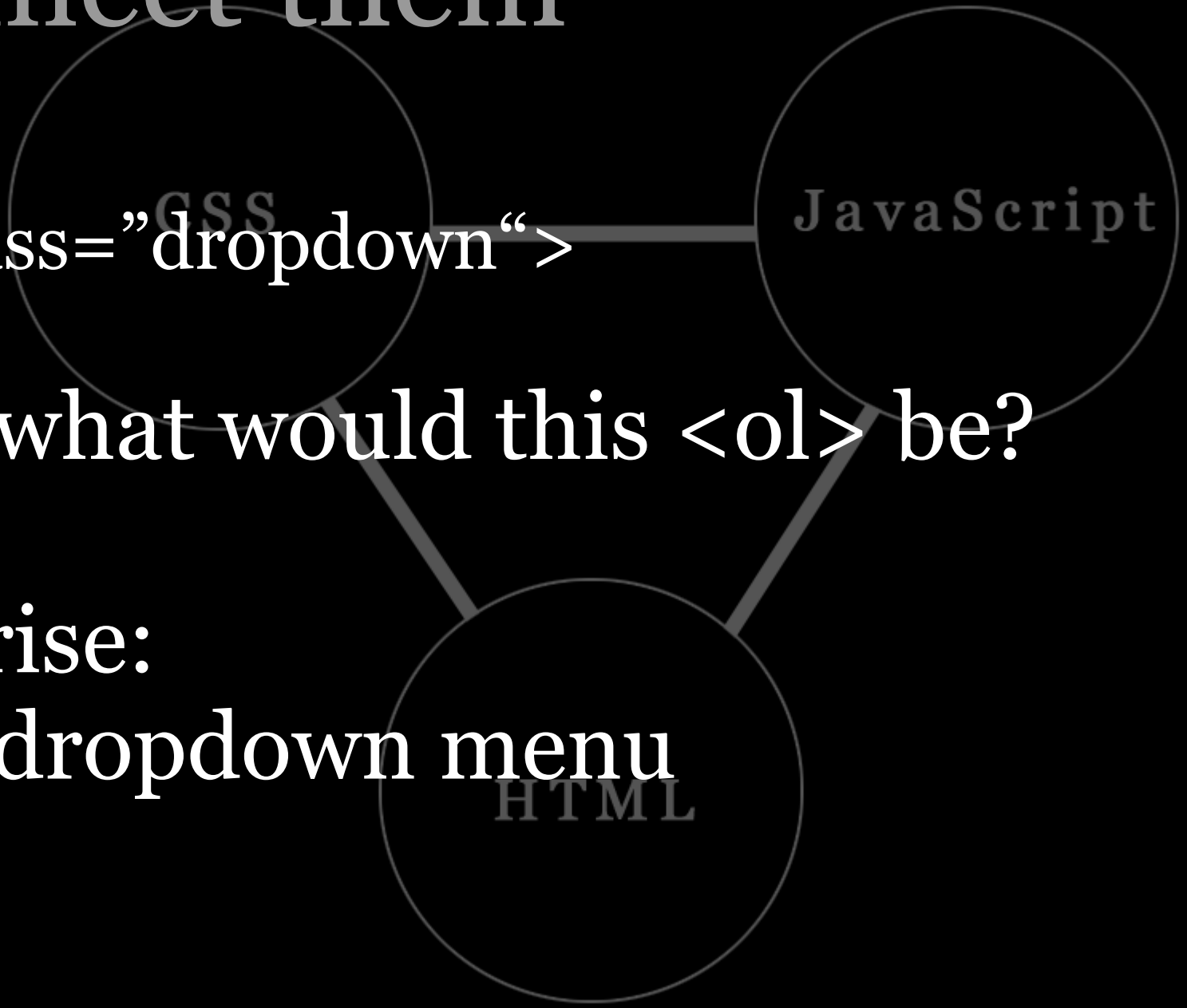
```
var els = document.getElementsByClassName('special')
if (els.length) {
    // go through all elements and do something
}
```

Connect them

- id
- class

Use the same hook for presentation and behavior; for CSS and JavaScript.

Connect them

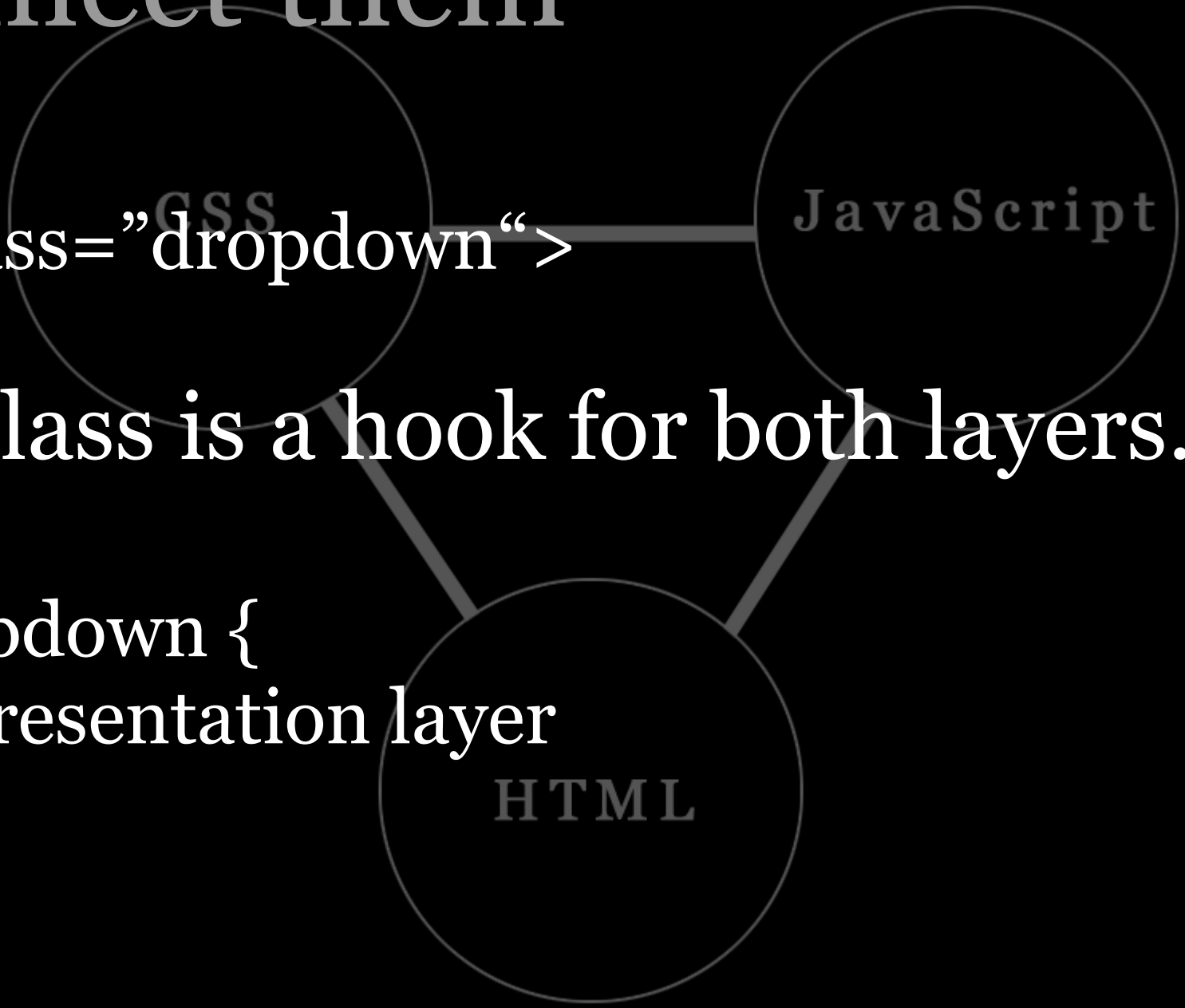


<ol class=”dropdown“>

Now what would this <ol> be?

Surprise:
it's a dropdown menu

Connect them

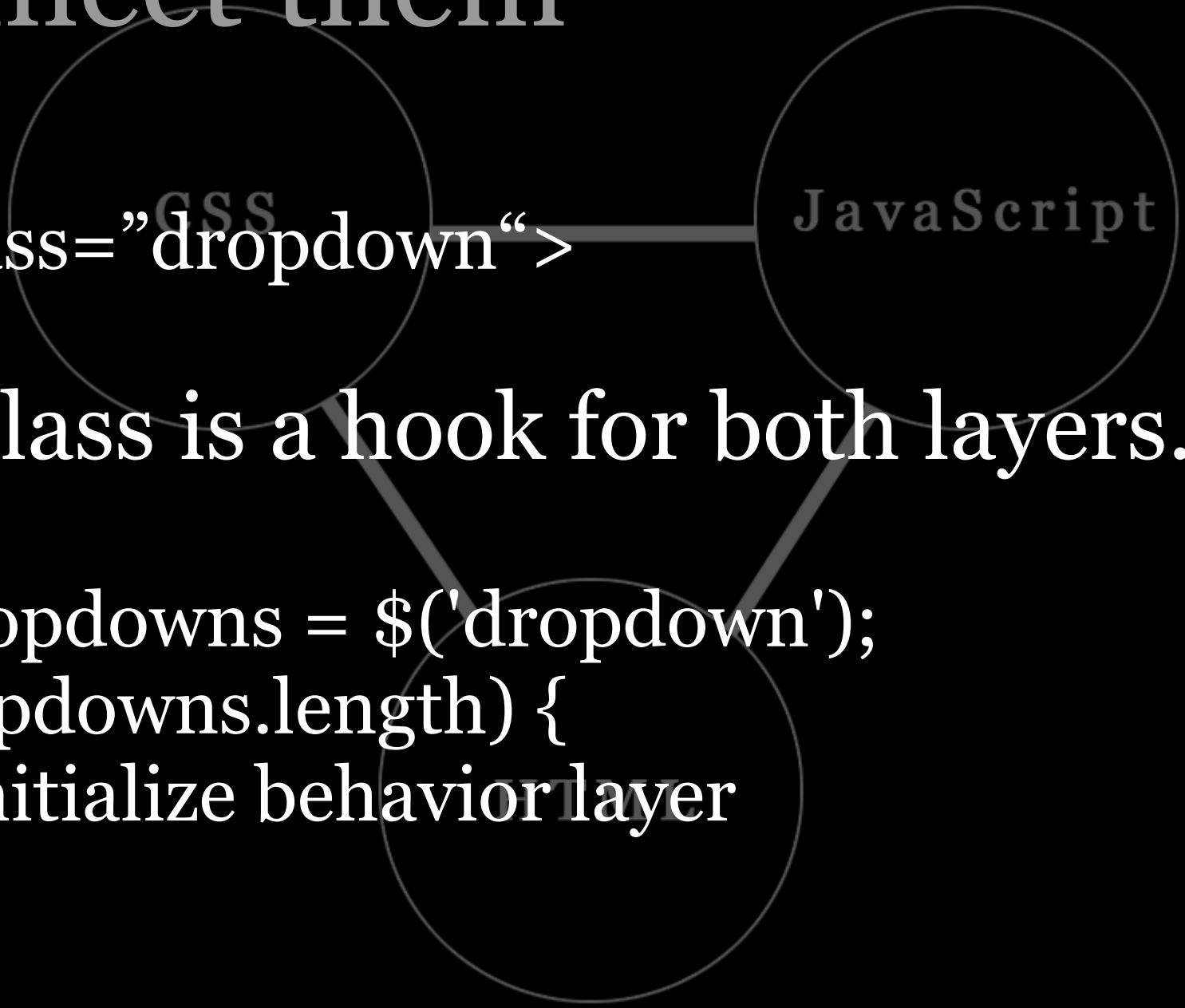



```
<ol class=”dropdown“>
```

The class is a hook for both layers.

```
ol.dropdown {
    // presentation layer
}
```

Connect them



<ol class=”dropdown“>

The class is a hook for both layers.

```
var dropdowns = $('dropdown');
if (dropdowns.length) {
    // initialize behavior layer
}
```

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
   - Separate them
   - Connect them

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything


BE
CAREFUL
THIS MACHINE
HAS NO BRAIN
USE YOUR OWN

Unobtrusive JavaScript

Two fundamental principles:

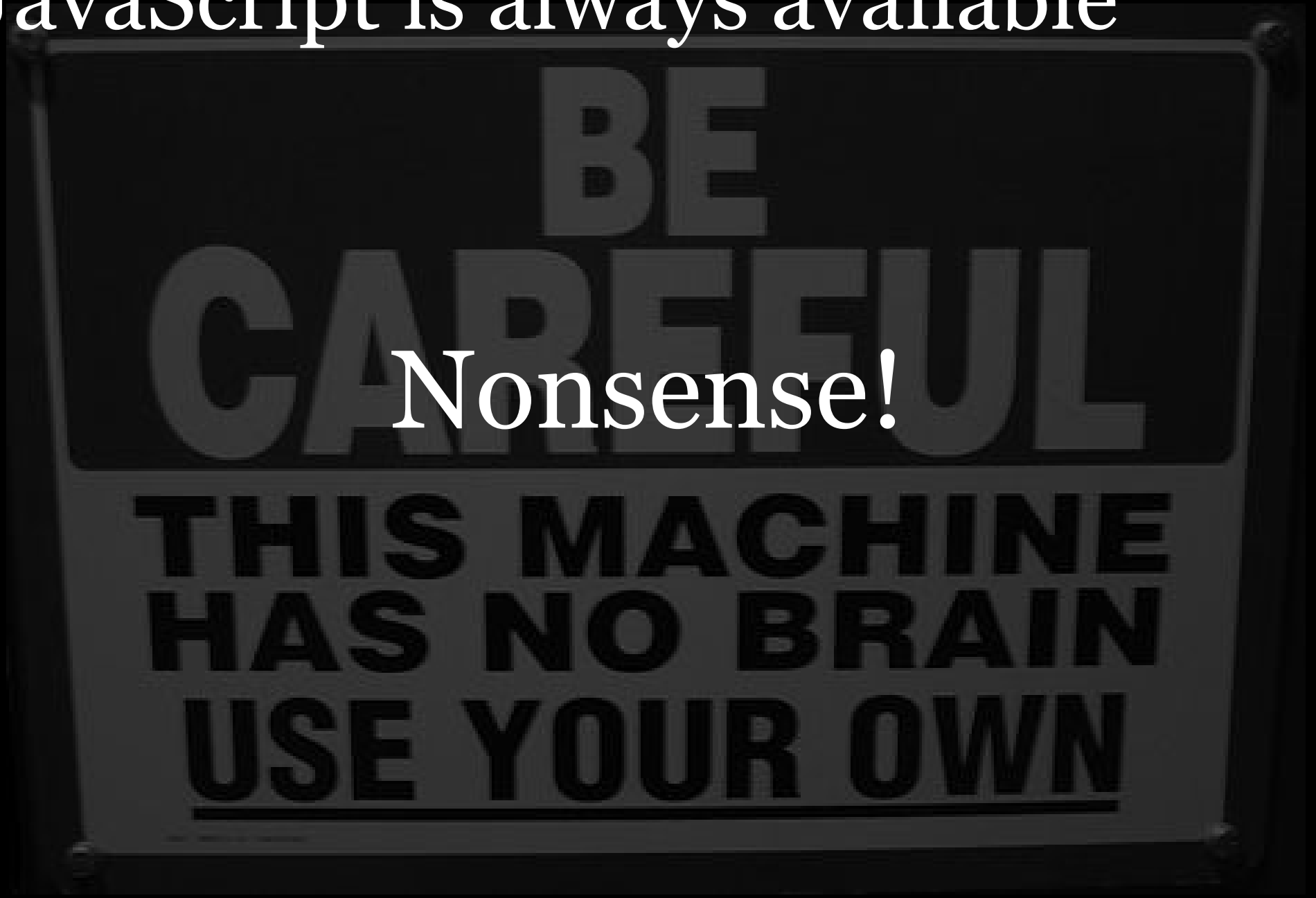
1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything
   - “JavaScript is always available”
   - “Everybody uses a mouse”

Unobtrusive JavaScript

Two fundamental principles:

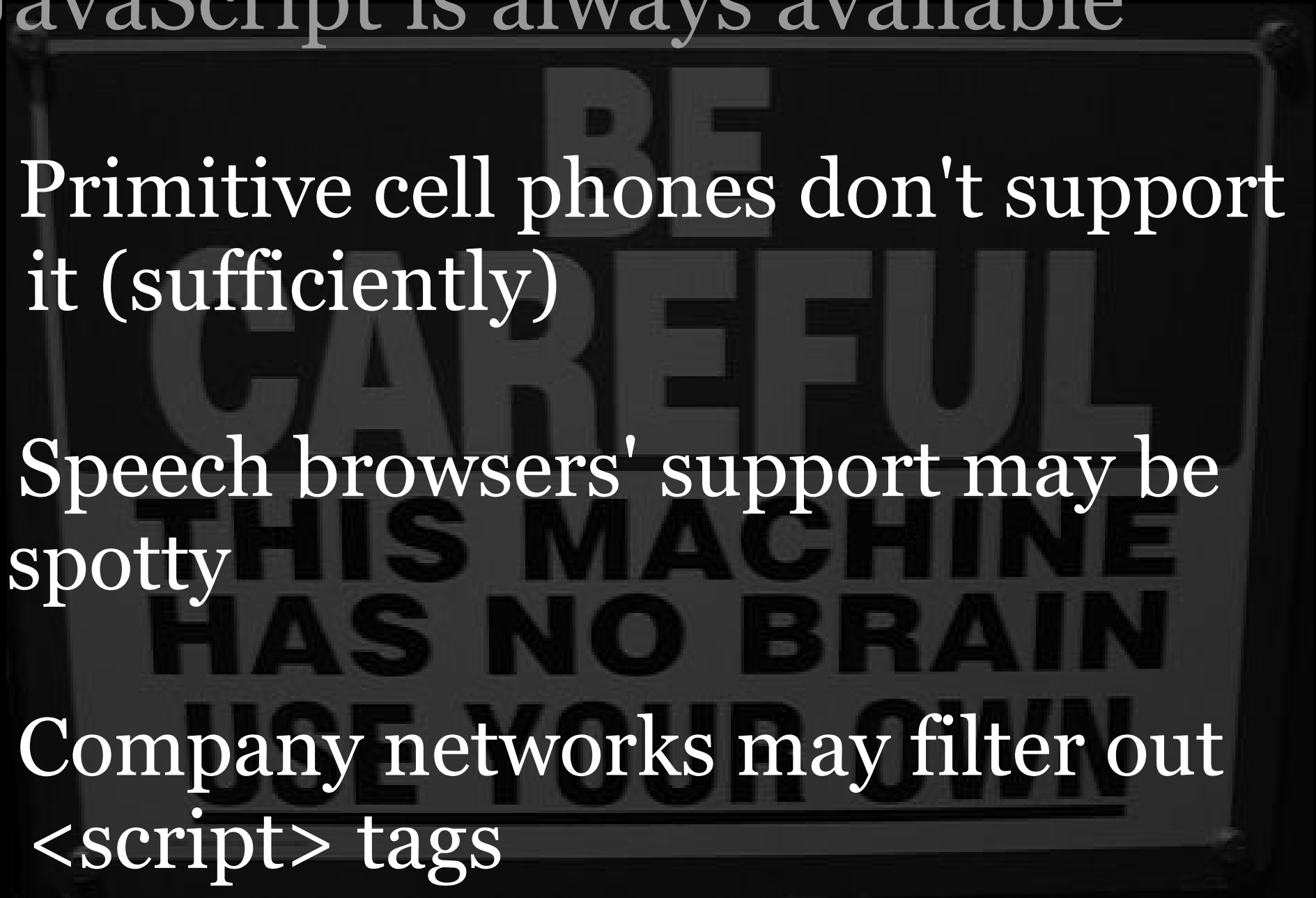
1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything
   - “JavaScript is always available”
   - “Everybody uses a mouse”

JavaScript is always available



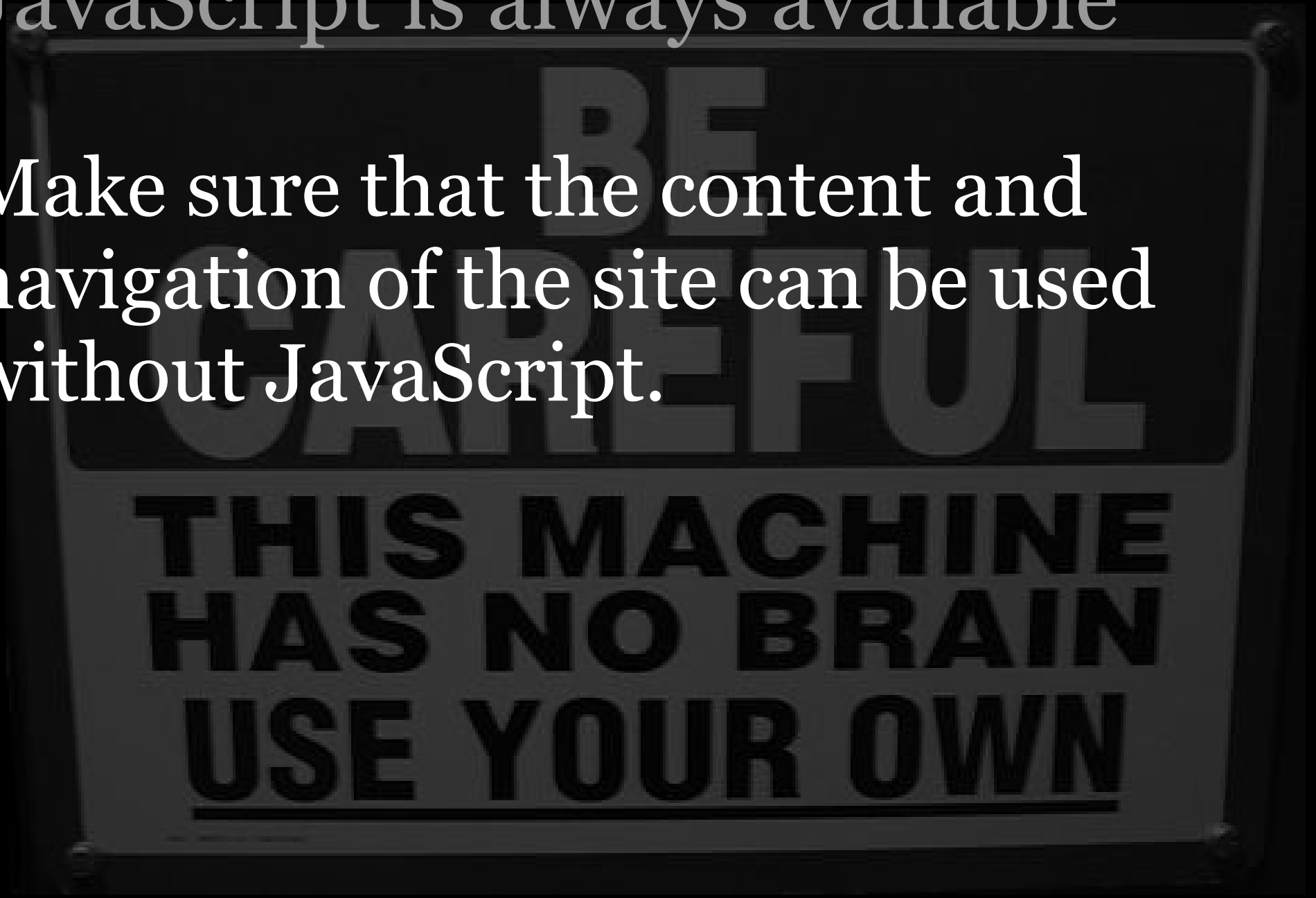
Nonsense!

JavaScript is always available



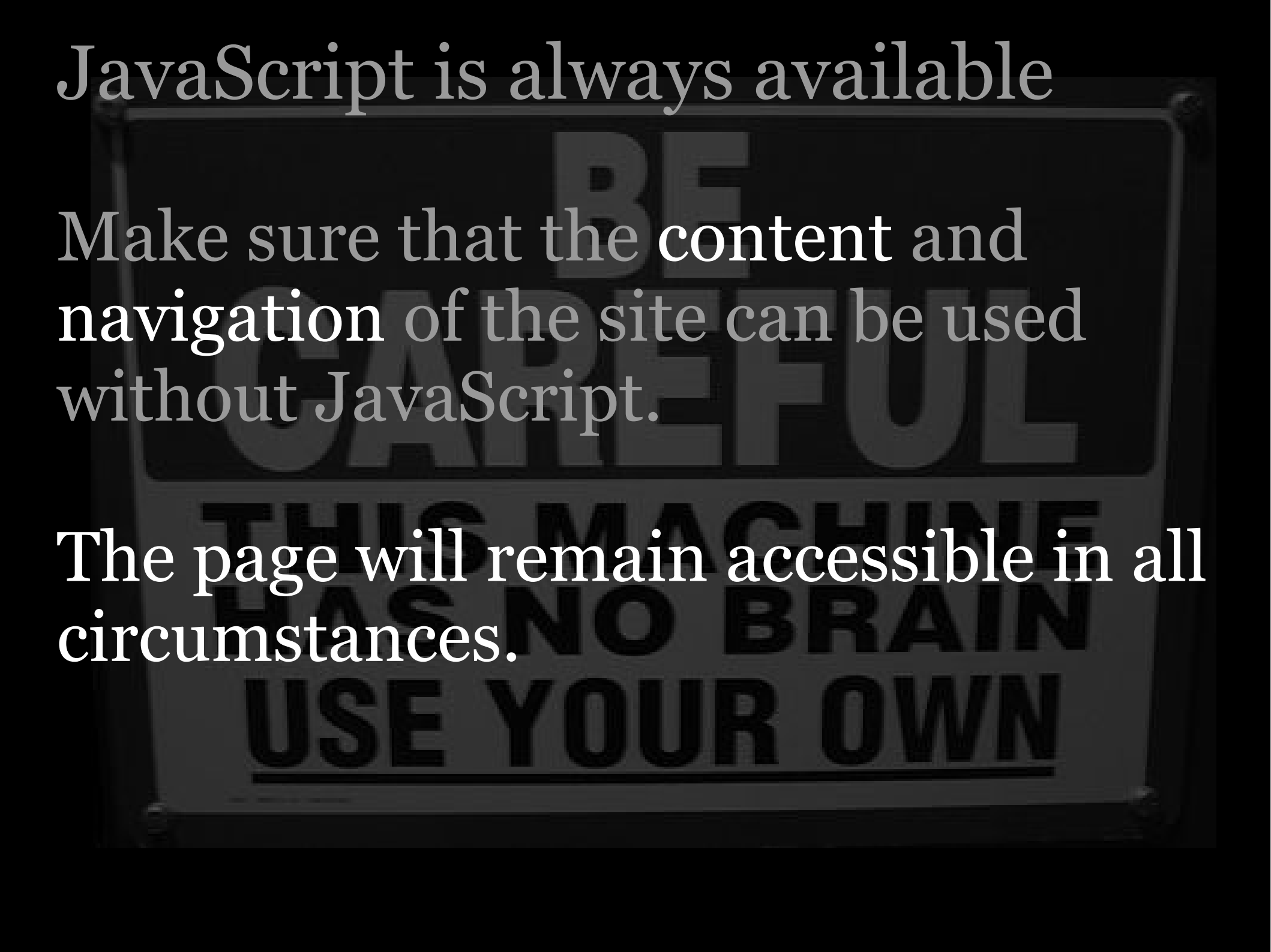
- Primitive cell phones don't support it (sufficiently)
- Speech browsers' support may be spotty
- Company networks may filter out <script> tags

JavaScript is always available



Make sure that the content and navigation of the site can be used without JavaScript.

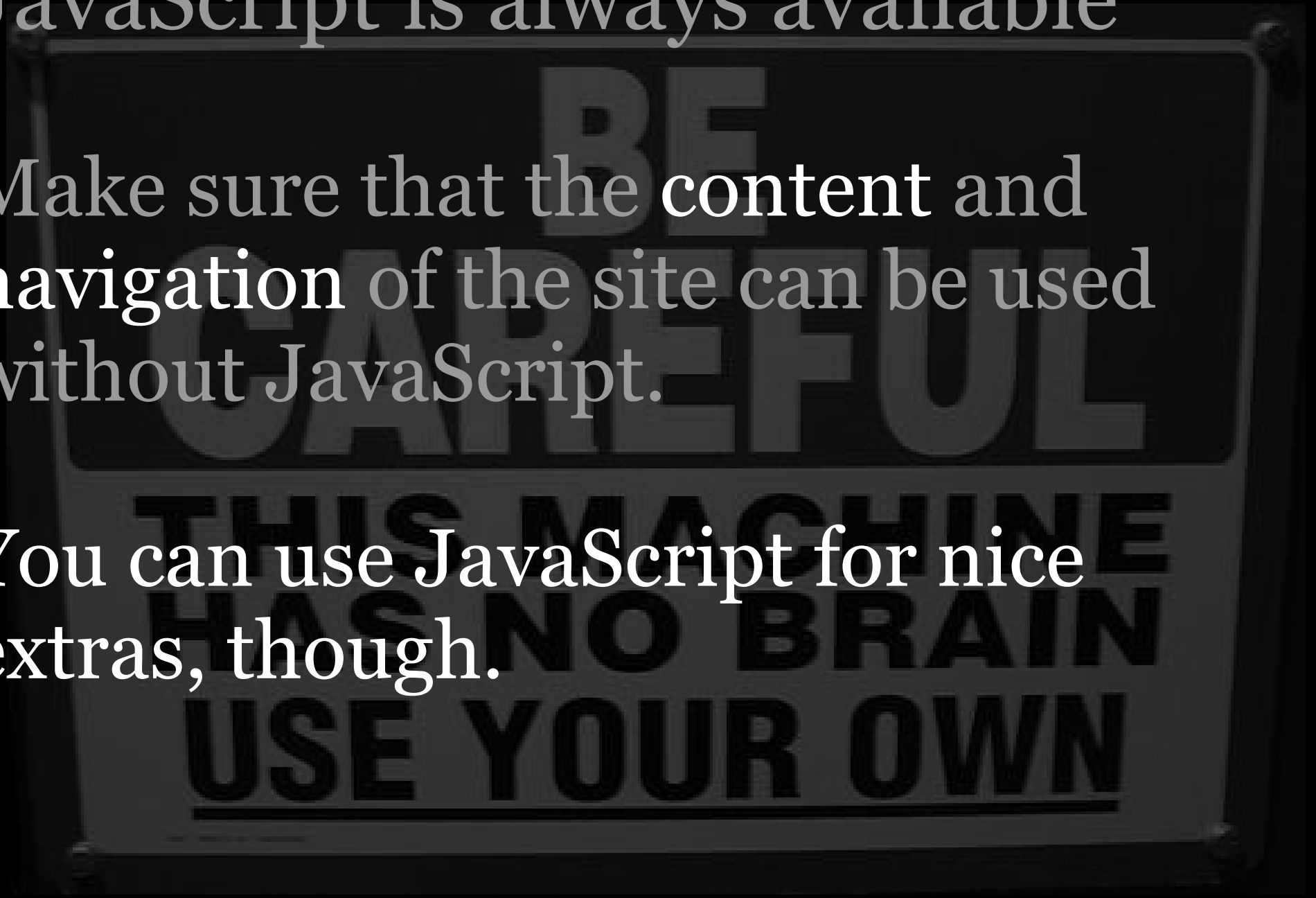
JavaScript is always available



Make sure that the content and navigation of the site can be used without JavaScript.

The page will remain accessible in all circumstances.

JavaScript is always available



Make sure that the content and navigation of the site can be used without JavaScript.

You can use JavaScript for nice extras, though.

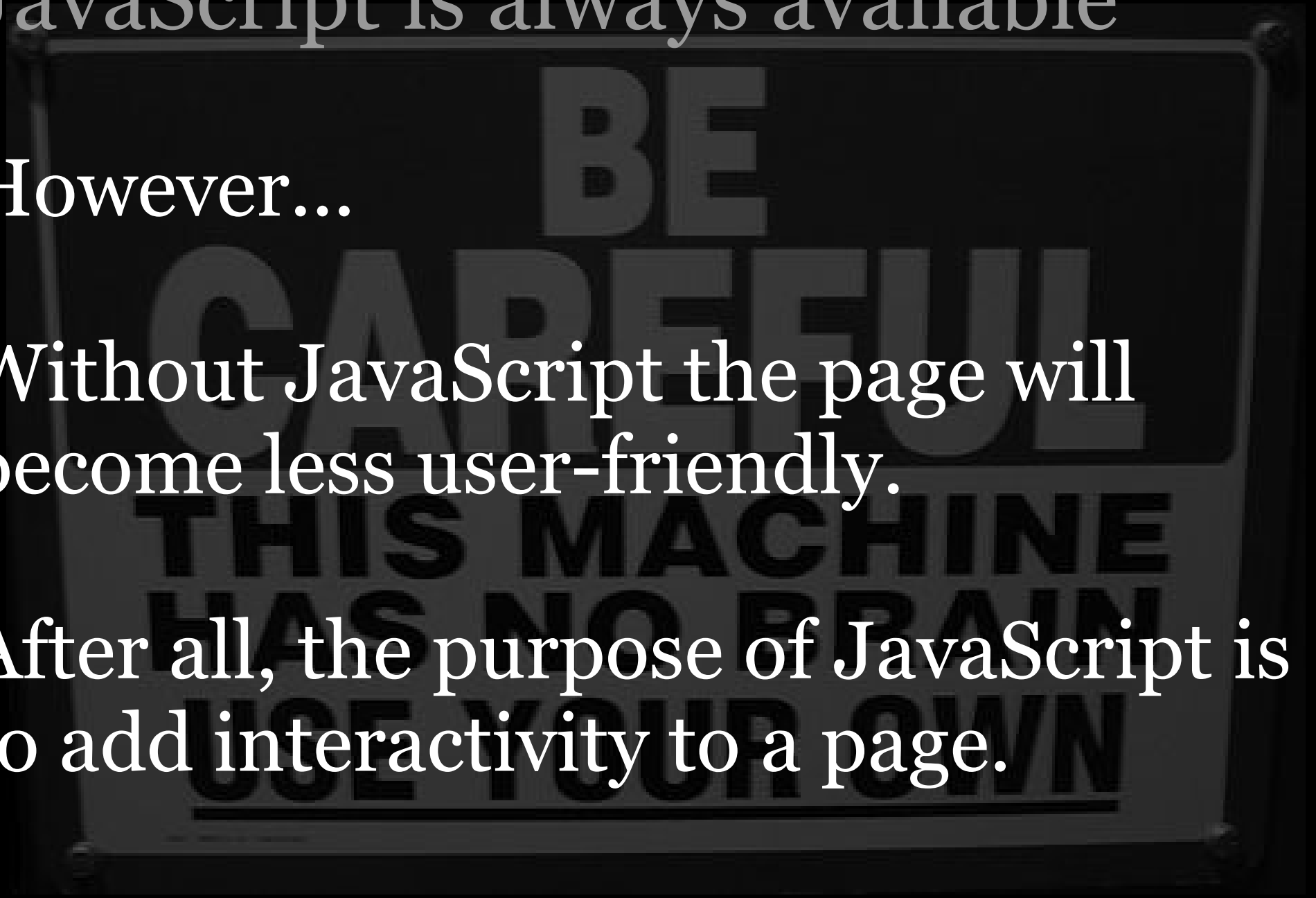
JavaScript is always available

However...

Without JavaScript the page will become less user-friendly.

Can't be helped.

JavaScript is always available



However...

Without JavaScript the page will become less user-friendly.

After all, the purpose of JavaScript is to add interactivity to a page.

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything
   - “JavaScript is always available”
   - “Everybody uses a mouse”

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything
   - “JavaScript is always available”
   - “Everybody uses a mouse”

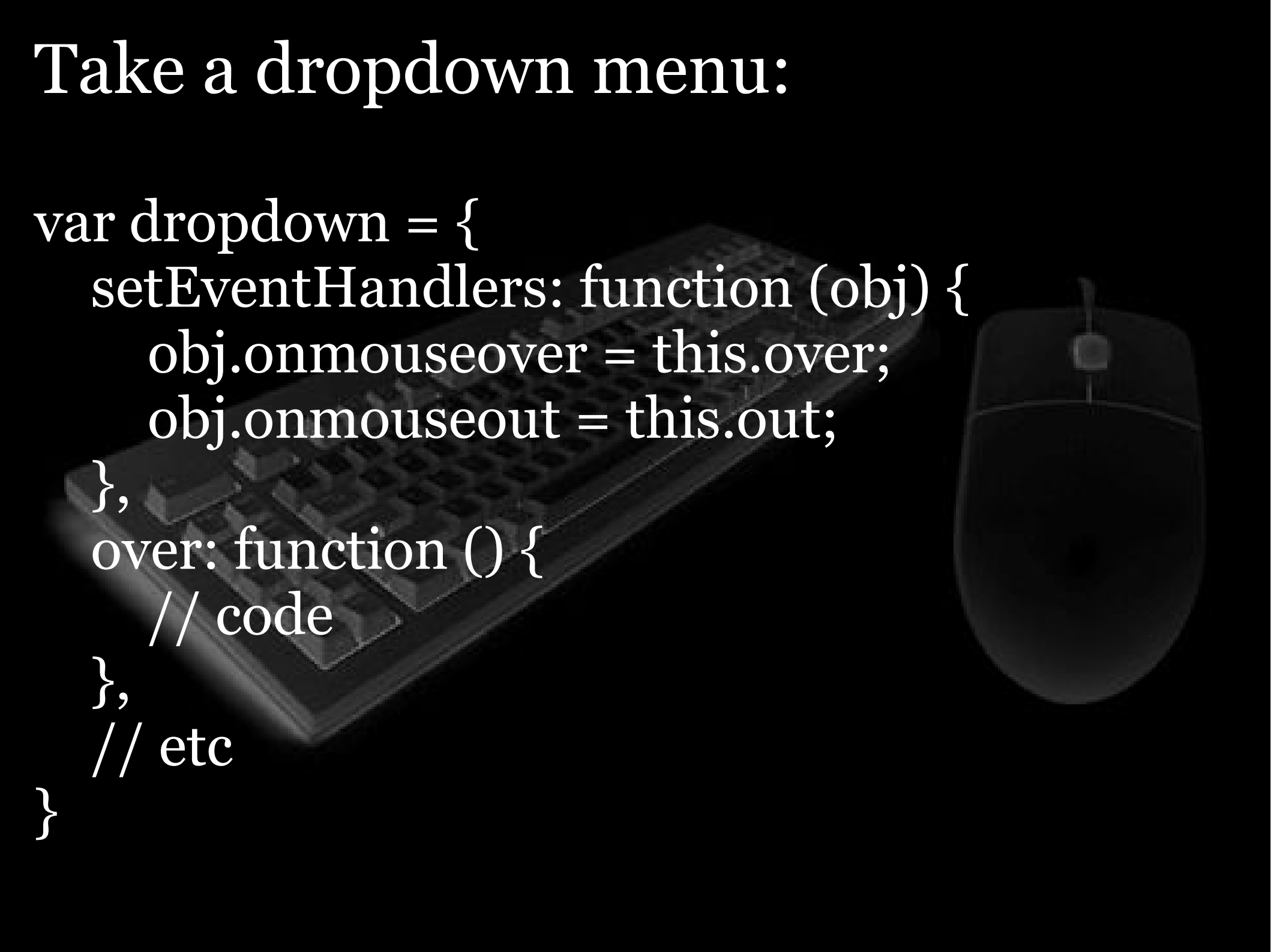
Everybody uses a mouse

Nonsense!

Device independence

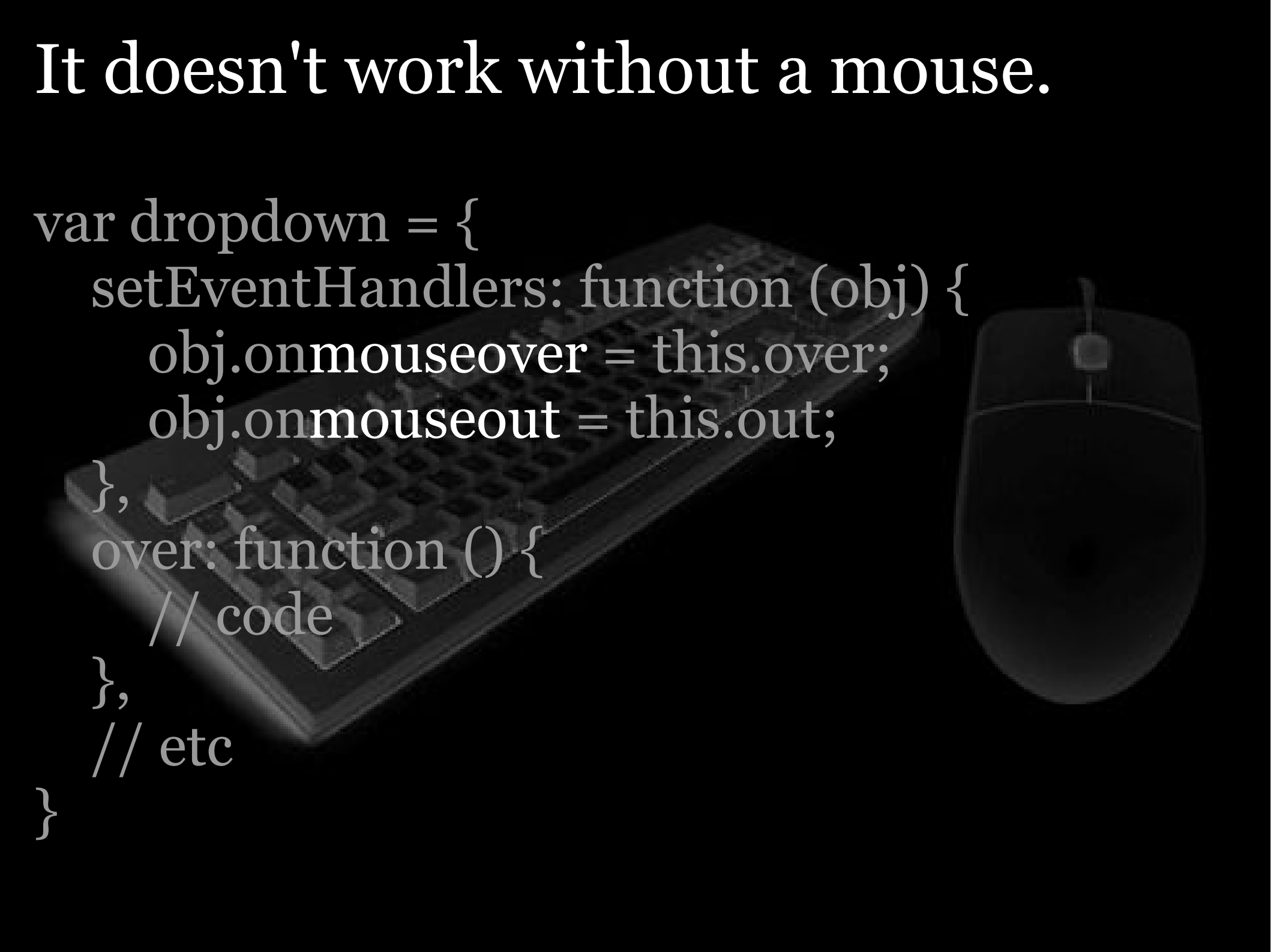
Take a dropdown menu:

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



It doesn't work without a mouse.

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



We need evens that are fired when the user “enters” or “leaves” a link by using the keyboard.

focus and blur

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





Restriction:
the object must be able to gain the keyboard focus.

- links
- form fields

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



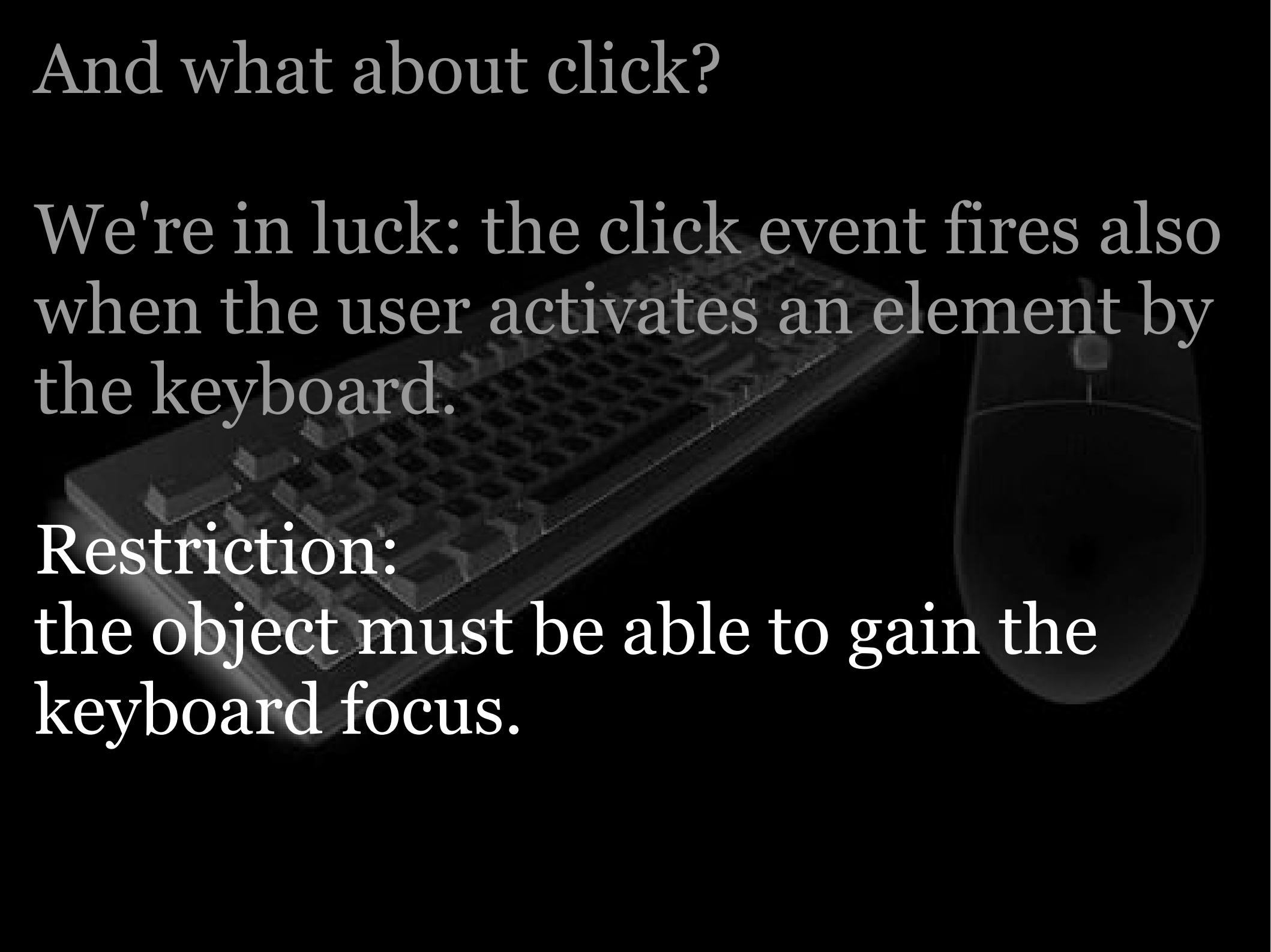

Restriction:
the object must be able to gain the keyboard focus.

- links
- form fields
- elements with tabindex

And what about click?

We're in luck: the click event fires also when the user activates an element by the keyboard.

click should be called activate.



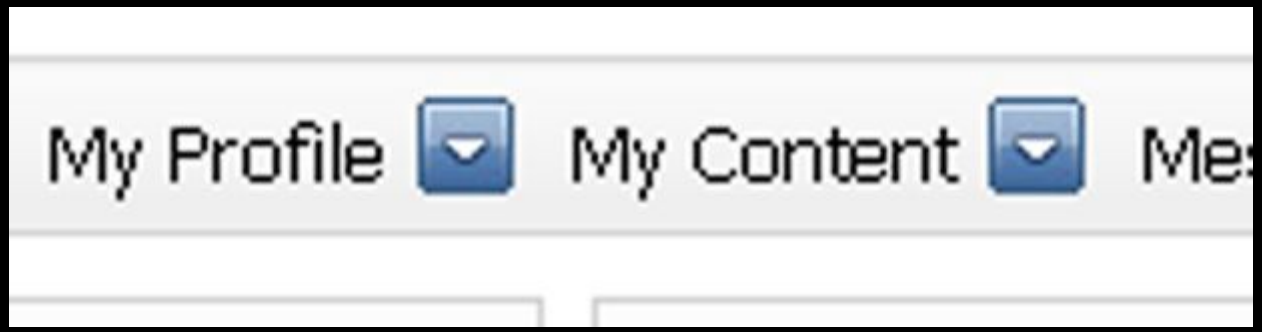
And what about click?

We're in luck: the click event fires also when the user activates an element by the keyboard.

Restriction:
the object must be able to gain the keyboard focus.

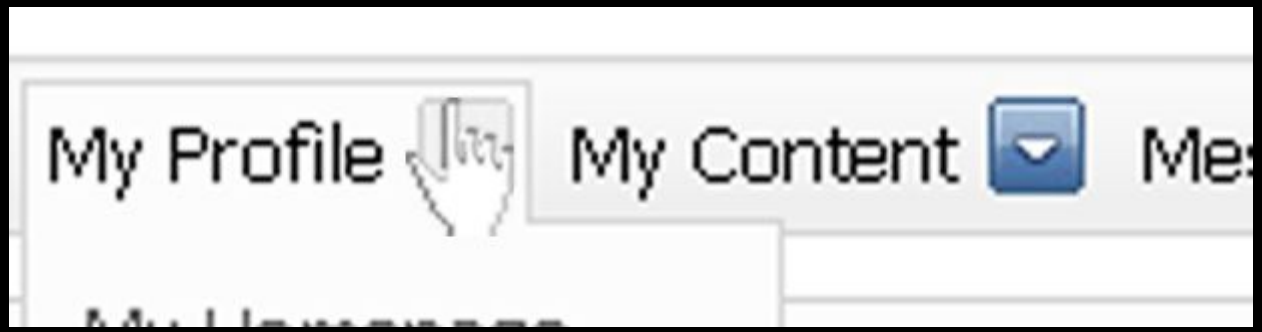
Click as activate

arrow.onclick = showMenu;



Click as activate

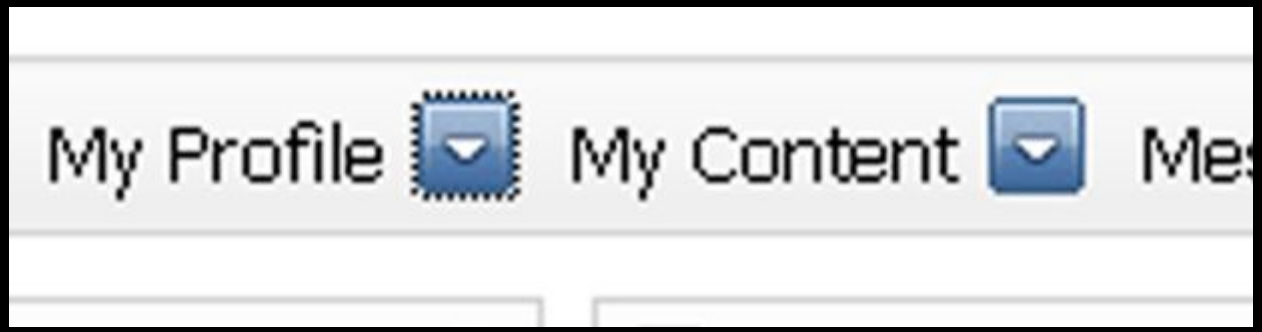
arrow.onclick = showMenu;



1) Mouse click on the arrow

Click as activate

arrow.onclick = showMenu;



1) Mouse click on the arrow
2) a) Keyboard focus on the arrow

Click as activate

arrow.onclick = showMenu;

1) Mouse click on the arrow
2) a) Keyboard focus on the arrow
   b) Space bar on the arrow

That's two actions.

Click as activate

arrow.onclick = arrow.onfocus = showMenu;

1) Mouse click on the arrow
2) Keyboard focus on the arrow
   ~~b) Space bar on the arrow~~

Click as activate

arrow.onclick = arrow.onfocus = showMenu;

1) Mouse click on the arrow
2) Keyboard focus on the arrow

The next tab will focus on the sub-menu. The user won't be able to skip it.

Click as activate

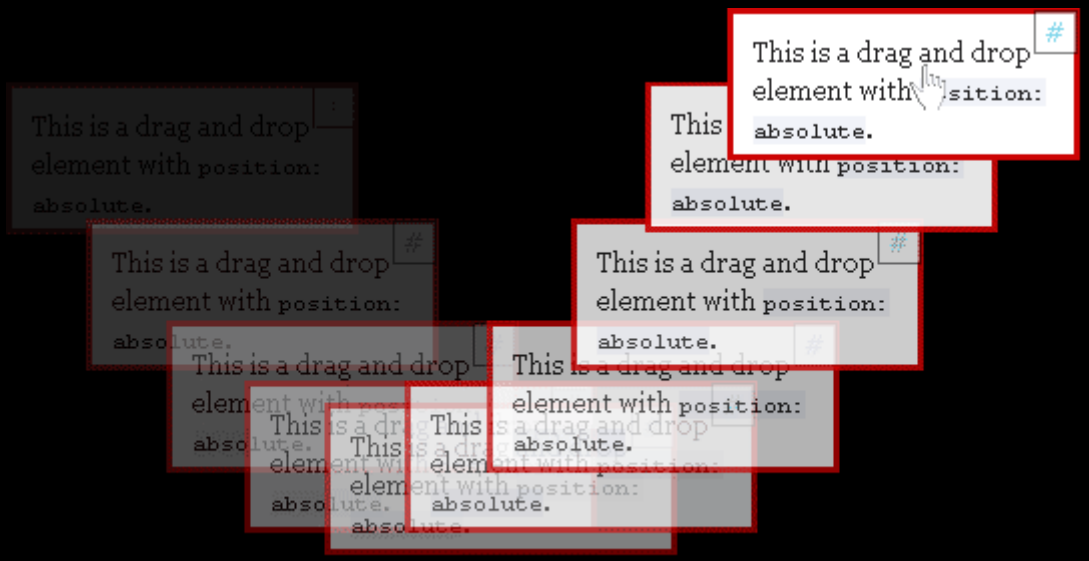
arrow.onclick = ~~arrow.onfocus =~~ showMenu;

Generally, keyboard users need more actions to achieve the same goals as mouse users.

Don't interfere too much. There are reasons for this behavior, and keyboard users are used to it.

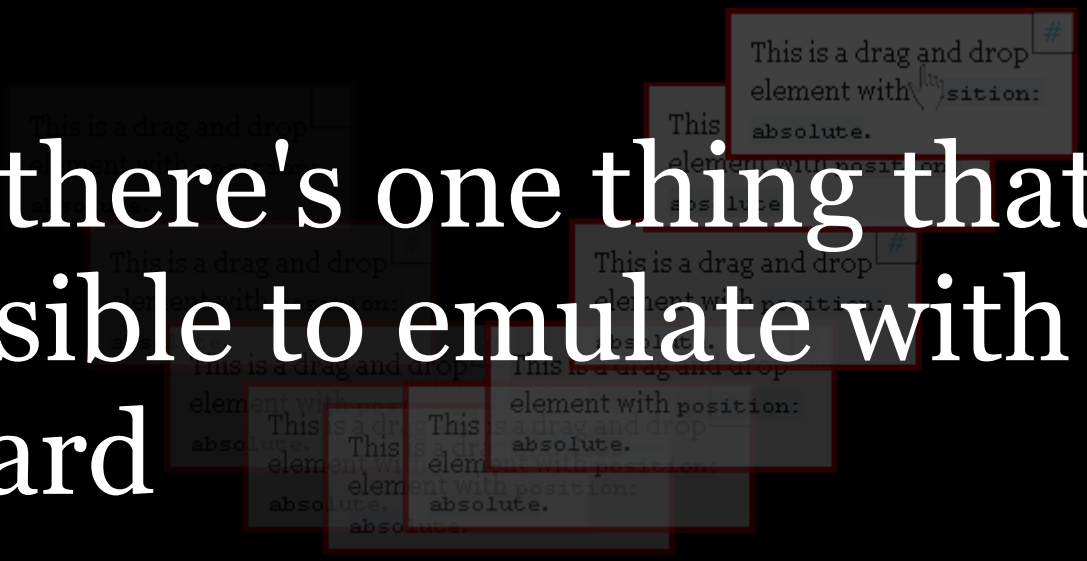
Separate concepts

Drag-and-drop
uses the mousemove event



Separate concepts

Drag-and-drop
uses the mousemove event



and if there's one thing that's impossible to emulate with the keyboard

it's moving the mouse

Separate concepts

Drag-and-drop
uses the mousemove event

How do we make this accessible?

By allowing the user to use the arrow keys.
Key events.

Separate concepts

Drag-and-drop

For detecting arrow keys (or other special keys) we need the keydown event.

Not keypress. (Doesn't work in IE and Safari)

Separate concepts

Drag-and-drop

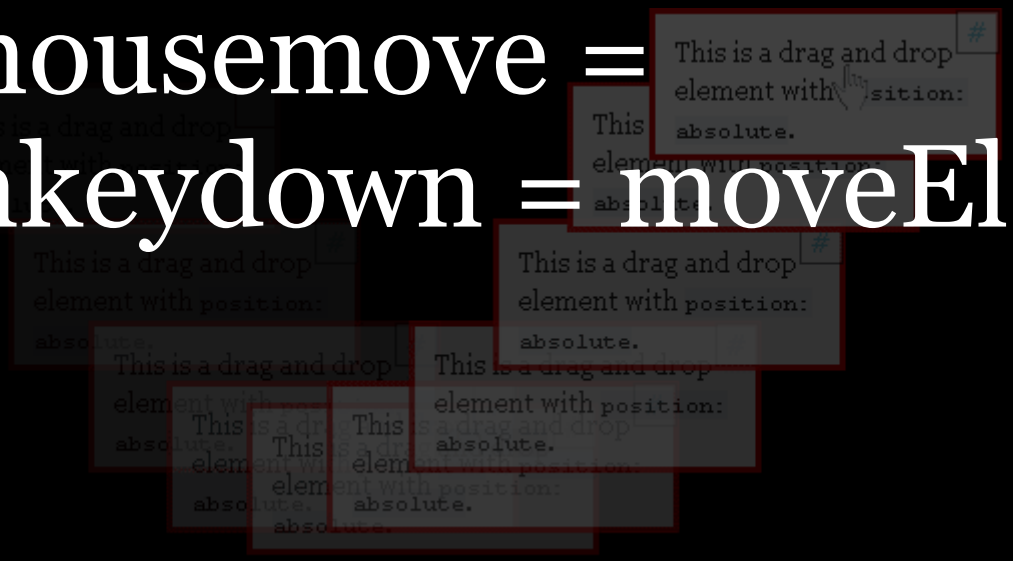
For detecting arrow keys (or other special keys) we need the keydown event.

Not ~~keypress~~. (Doesn't work in IE and Safari)

Separate concepts

Drag-and-drop

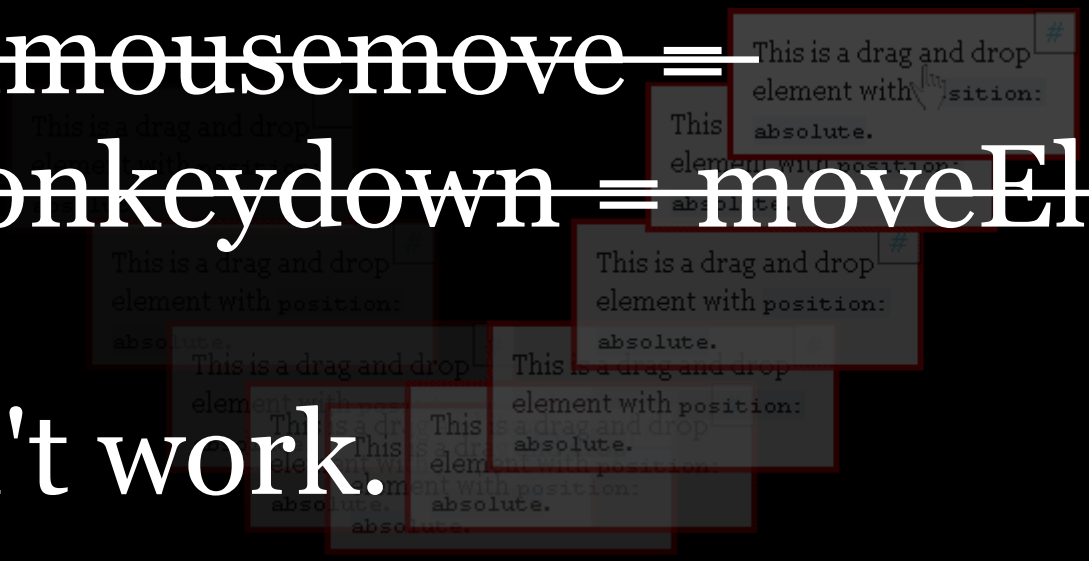
```
obj.onmousemove =
   obj.onkeydown = moveElement;
```



Separate concepts

Drag-and-drop

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

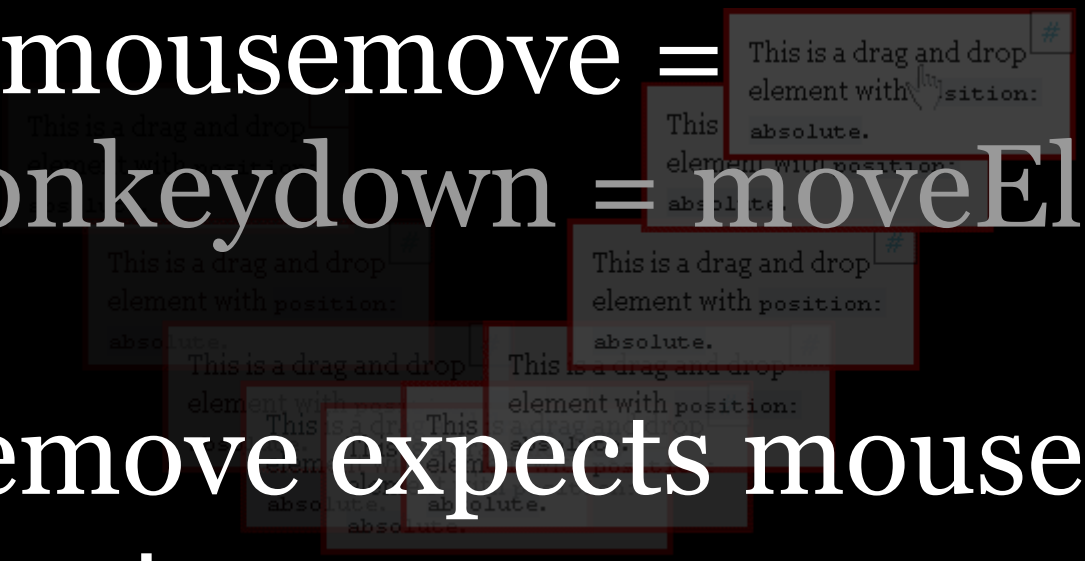


Doesn't work.

Separate concepts

Drag-and-drop

obj.onmousemove =
  obj.onkeydown = moveElement;

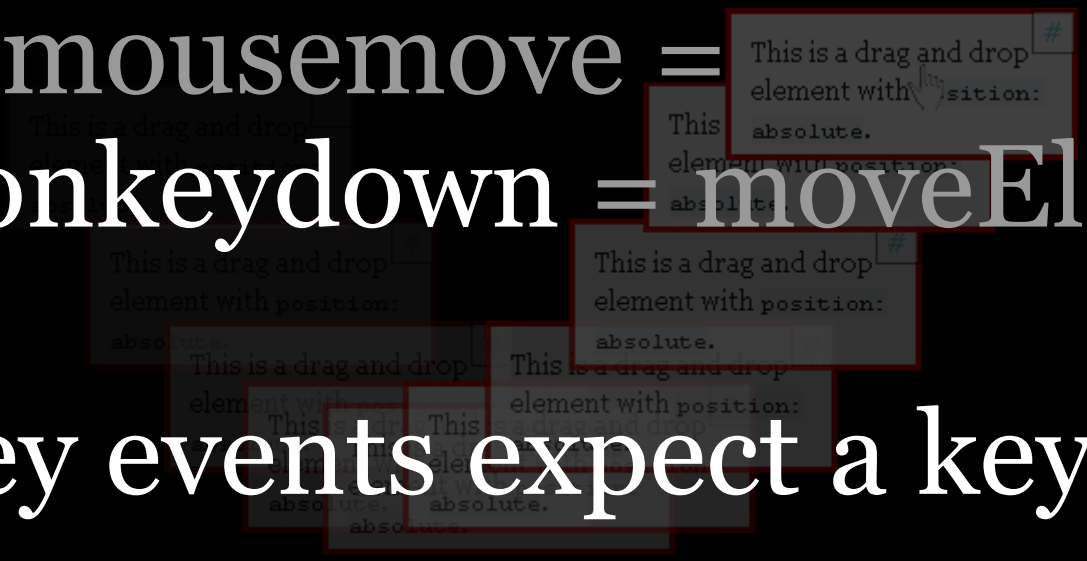


Mousemove expects mouse coordinates.
The layer moves to these coordinates.

Separate concepts

Drag-and-drop

obj.onmousemove =
  obj.onkeydown = moveElement;



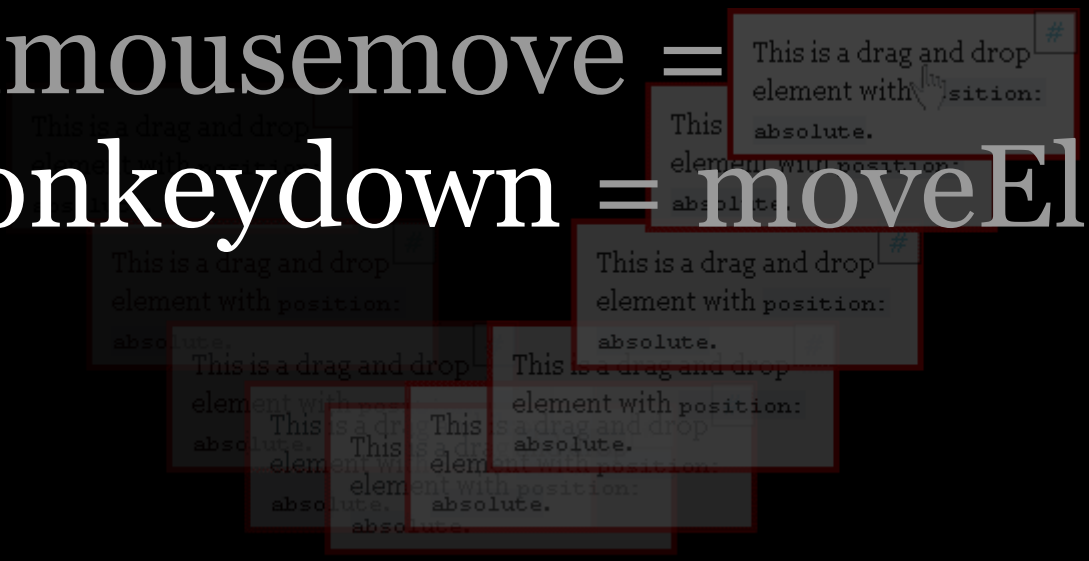
The key events expect a keystroke.

But what does “user hits right arrow once” mean?

Separate concepts

Drag-and-drop

obj.onmousemove =
  obj.onkeydown = moveElement;

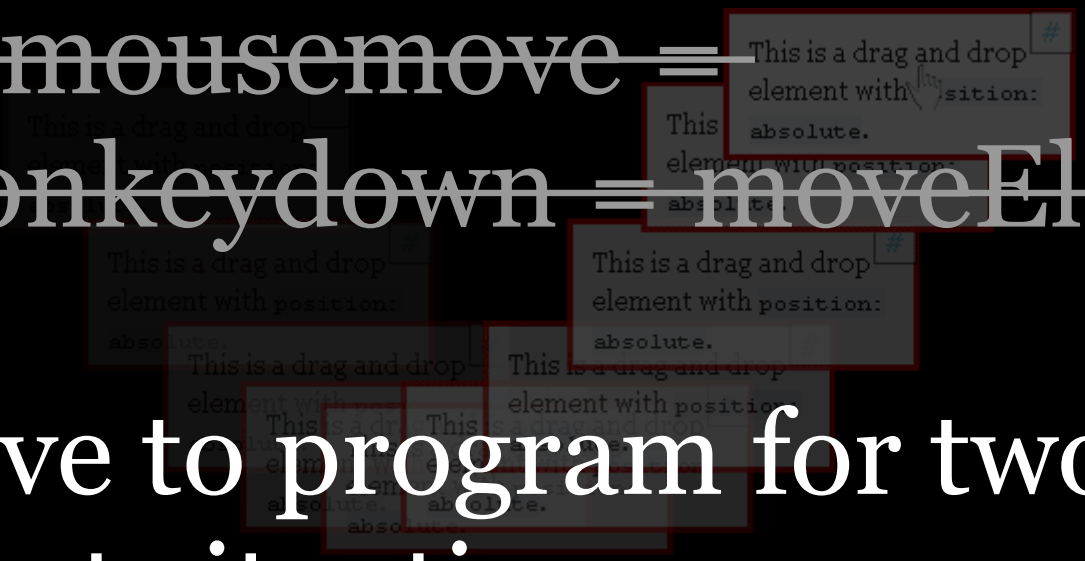


10px?
50px?
“Move to next receptor element?”
Something else that fits your interface?

Separate concepts

Drag-and-drop

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

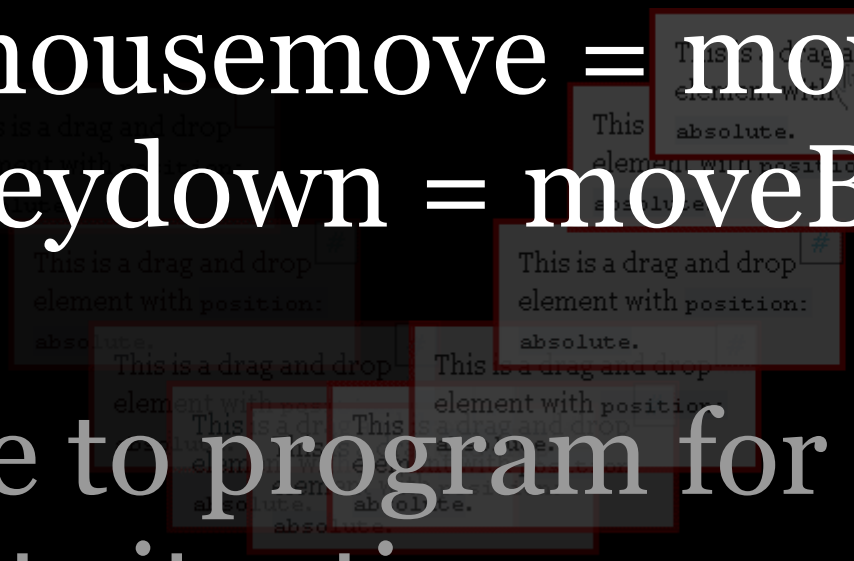


We have to program for two totally different situations.
We need separate scripts.

Separate concepts

Drag-and-drop

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

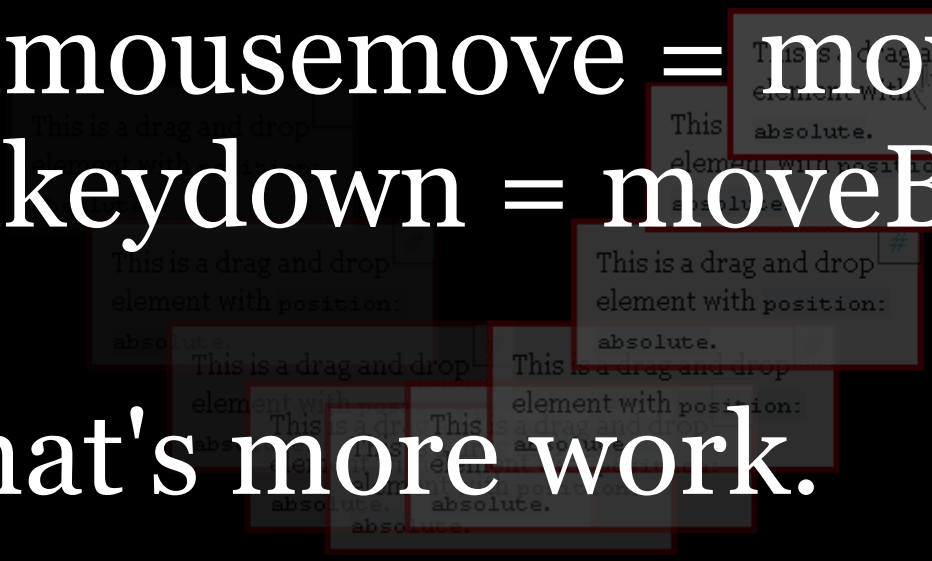


We have to program for two totally different situations.
We need separate scripts.

Separate concepts

Drag-and-drop

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

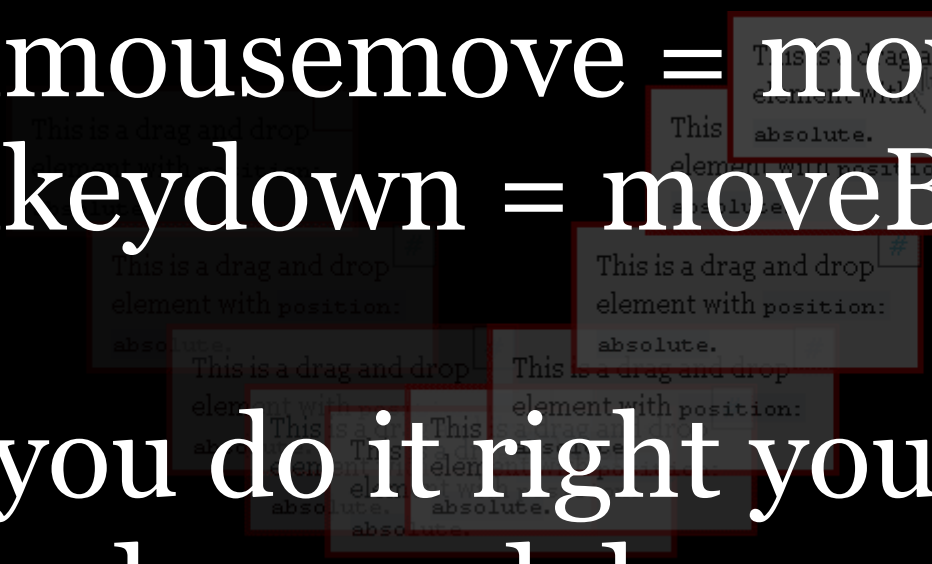


Yes, that's more work.

Separate concepts

Drag-and-drop

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

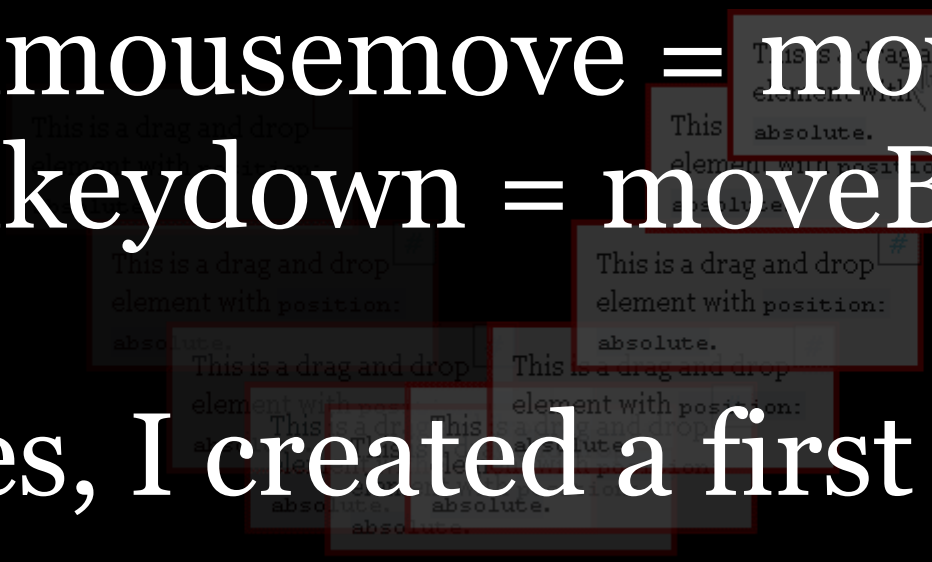


But if you do it right you've got a generic drag and drop module you can use anywhere.

Separate concepts

Drag-and-drop

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



Besides, I created a first draft for you.

Separate concepts

Drag-and-drop

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

Besides, I created a first draft for you.

Unobtrusive JavaScript

Two fundamental principles:

1) Separation of structure, presentation, and behavior
2) The script doesn't assume anything.

Unobtrusive JavaScript

It's not that hard

Need help?

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

and of course quirksmode.org


Questions?