

Hell is other browsers - *Sartre*

fundamentos

web 2008

Ajax Workshop

Peter-Paul Koch (ppk)

<http://quirksmode.org>

Fundamentos del Web, 28 October 2008

Hell is other browsers - *Sartre*

fundamentos

web 2008

Ajax Workshop

Part I- Unobtrusive
JavaScript

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

Separate them

Separation of HTML and JavaScript:



```
graph TD; CSS((CSS)) --- JS((JavaScript)); HTML((HTML)) --- CSS; HTML --- JS;
```

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

Separate them

Separation of HTML and JavaScript:



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

No inline event handlers!

HTML

Separate them

Separation of HTML and JavaScript:



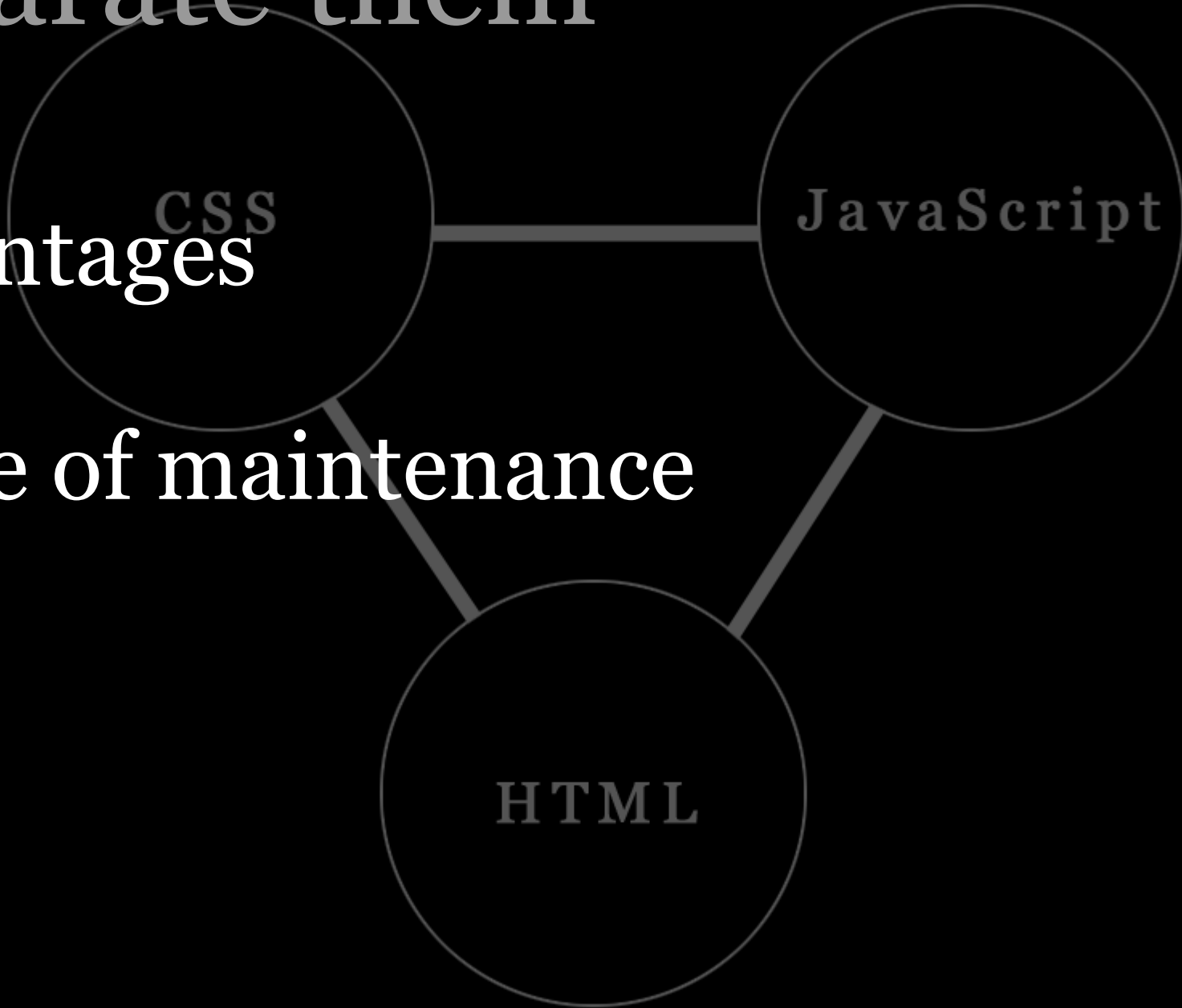
```
<input id="special" />
```

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


Separate them

Advantages

- Ease of maintenance



Separate them

Separation of HTML and JavaScript:



```
<input id="special" />
```

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

Separate them

Separation of HTML and JavaScript:



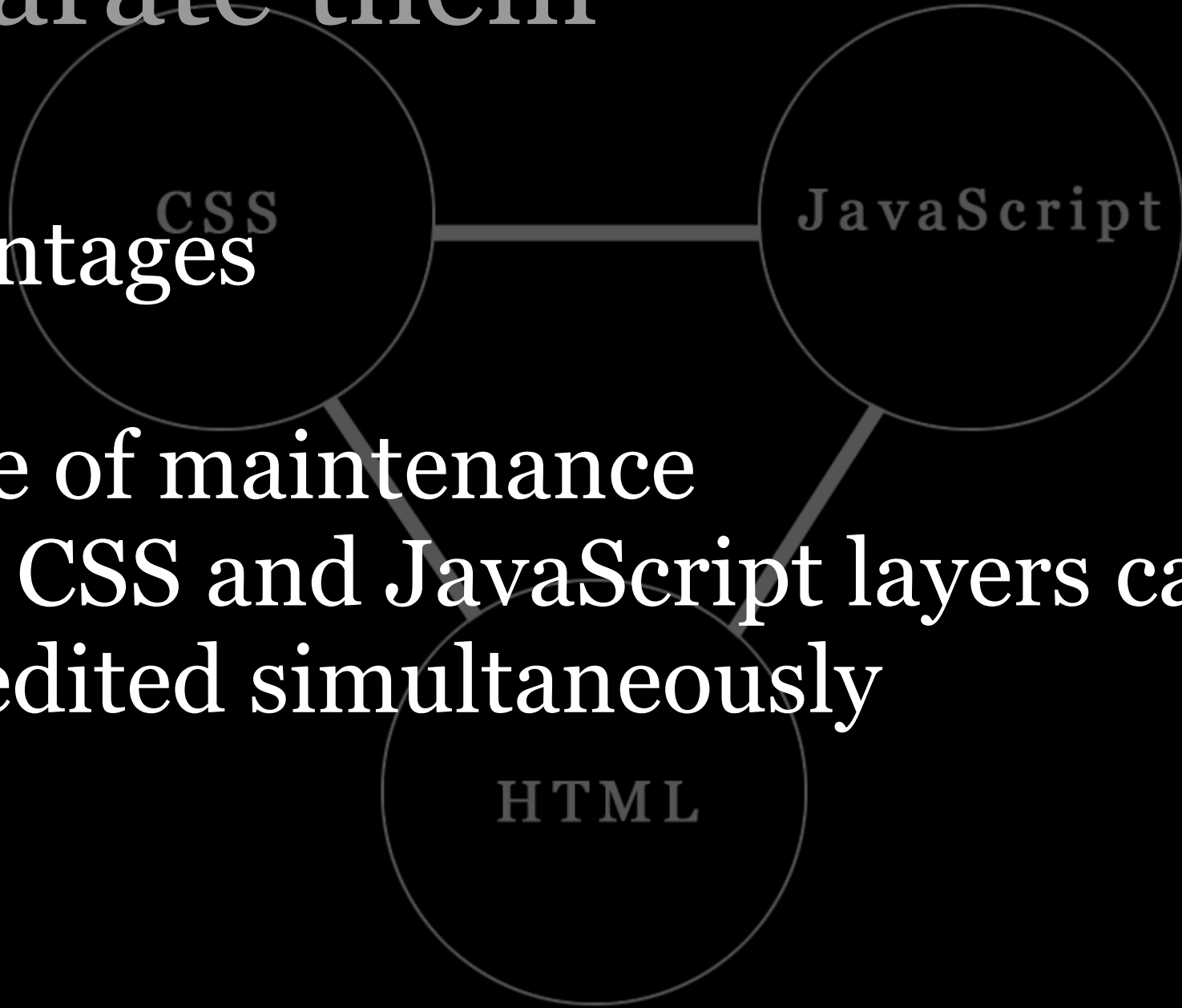
```
<input id="special" />
```

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

Separate them

Advantages

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



Exercise:

fundamentos
web 2008

Do you use inline event handlers?

If so, why?

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
- 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”

**BE
CAREFUL**

**THIS MACHINE
HAS NO BRAIN
USE YOUR OWN**

JavaScript is always available

**BE
CAREFUL**

Nonsense!

**THIS MACHINE
HAS NO BRAIN
USE YOUR OWN**

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

- Primitive cell phones don't support



- Company networks may filter out
<script> tags

Exercise:

fundamentos
web 2008

How does your site perform when
JavaScript is disabled?

JavaScript is always available

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

A large, dark, rectangular sign with a white border. The sign contains the text 'BE CAREFUL' at the top, 'THIS MACHINE HAS NO BRAIN' in the middle, and 'USE YOUR OWN' at the bottom. The text is in a bold, sans-serif font. The sign is slightly tilted and has a shadow effect.

**BE
CAREFUL
THIS MACHINE
HAS NO BRAIN
USE YOUR OWN**

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.

BE
CAREFUL
THIS MACHINE
HAS NO BRAIN
USE YOUR OWN

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.

Exercise:

fundamentos
web 2008

Can you make your navigation accessible without JavaScript?

(We'll talk about the content later.)

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”

Hell is other browsers - *Sartre*

fundamentos

web 2008

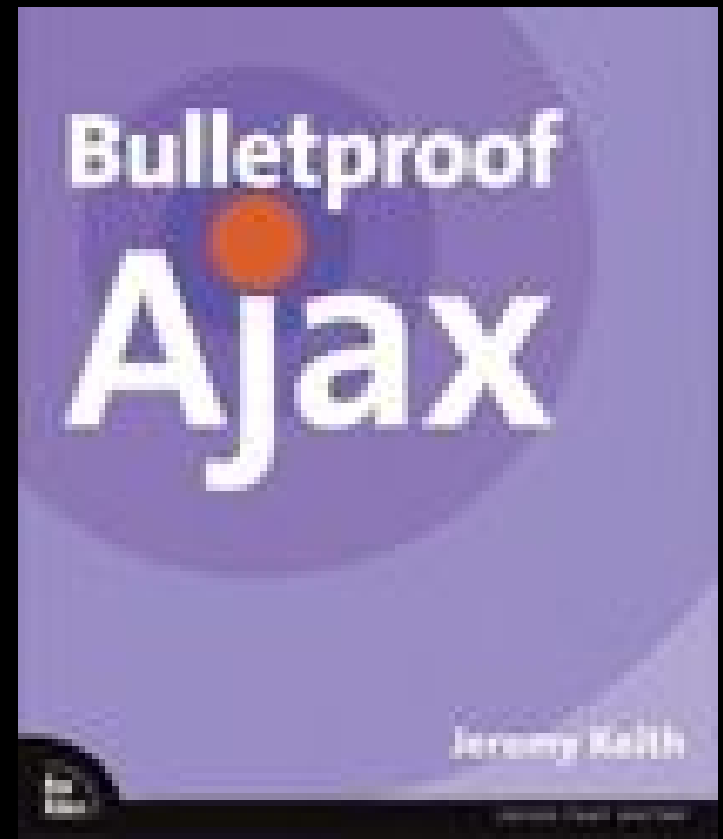
Ajax Workshop

Part 2- Hijax

Jeremy Keith

High Performance
Web Sites

His idea: Hijax



Hijax

An Ajax site works with JavaScript
which means that it won't work
in browsers that don't support
JavaScript

Hijax

But there's a simple tool that will help you make your Ajax site somewhat accessible

the hyperlink

```
<a href="page.html">Text</a>
```

Hijax

Links exist to lead users to a next page

so if we can use them that way in our Ajax sites, too

some accessibility problems disappear

Hijax

Search

Filter

Hijax

The best phone

Search

Filter

Hijax

The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

Hijax

The best phone

Search



Filter

Futurix 3BT0-N3

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

Futurix 3LTo-N4

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

Futurix 3LTo-N4

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

The Futurix 3BTo-N3 is by far the best phone in the world. Not only does it support Flarby 1.0 and Warblegarble, its users are universally of the opinion that its high-fidelity implementation of Huntigarby 3.5b is simply the best available.

This is a link!



The best phone

Search



Filter

Hijax

Search



Filter

Futurix 3BTo-N3

[more info](#)

Futurix 3LTo-N4

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

This is a link!



The best phone

Search



Filter

Futurix 3BTo-N3

[more info](#)

Futurix 3LTo-N4

[more info](#)

Futurix 5GS-N12

[more info](#)

Hijax

Search

Filter

This is a link!

Futurix 3BT0-N3

Futurix 5GS-N12

[more info](#)

[more info](#)



Hijax

The best phone

Search



Filter

The Futurix 3BT0-N3 is by far the best phone in the world. Not only does it support Flarby 1.0 and Warblegarble, its users are universally of the opinion that its high-fidelity implementation of Huntigarby 3.5b is simply the best available.

Hijax

However, every time the user clicks on a link, the browser loads a new page.

The site becomes less usable.
It's still accessible, though.

Hijax

How do you do this?

Start with a link

```
<a href="3BTo-N3.html">more info</a>
```

You're going to need it for the accessible version.

Hijax

```
<a href="3BTo-N3.html">more info</a>
```

```
link.onclick = function () {  
    var dataFile = this.href.replace(/\.html/, '.xml');  
    sendRequest(dataFile);  
    return false;  
}
```

Hijax

```
<a href="3BTo-N3.html">more info</a>
```

```
link.onclick = function () {  
    var dataFile = this.href.replace(/\.html/, '.xml');  
    sendRequest(dataFile);  
    return false;  
}
```

Hijax

```
<body>
```

```
<ul id="navigation">[etc.]</ul>
```

```
<form><input name="search" />
```

```
<input type="submit" value="Search" />
```

```
<div id="mainData">
```

```
<p>The Futurix 3BTo-N3 is by far the best phone  
in the world. Not only does it support Flarby 1.0  
and Warblegarble, its users are universally of the  
opinion that its high-fidelity implementation of  
Huntigarby 3.5b is simply the best available.</p>
```

```
</div>
```

```
<div class="footer">We're great!</div>
```

Hijax

```
function sendRequest(file) {  
    var req = createXMLHttpRequest();  
    req.open("GET",file,true);  
    req.setRequestHeader('User-Agent','XMLHTTP')  
    req.onreadystatechange = function () {  
        [send back to caller];  
    }  
    req.send();  
}
```

Hijax

```
function sendRequest(file) {  
    var req = createXMLHttpRequest();  
    req.open("GET",file,true);  
    req.setRequestHeader('User-Agent','XMLHTTP')  
    req.onreadystatechange = function () {  
        [send back to caller];  
    }  
    req.send();  
}
```

Hijax

```
<body>
```

```
<ul id="navigation">[etc.]</ul>
```

```
<form><input name="search" />
```

```
<input type="submit" value="Search" />
```

```
<div id="mainData">
```

```
<p>The Futurix 3BTo-N3 is by far the best phone  
in the world. Not only does it support Flarby 1.0  
and Warblegarble, its users are universally of the  
opinion that its high-fidelity implementation of  
Huntigarby 3.5b is simply the best available.</p>
```

```
</div>
```

```
<div class="footer">We're great!</div>
```

Exercise:

fundamentos
web 2008

Determine how you can use Hijax
in your Ajax site.

Hell is other browsers - *Sartre*

fundamentos

web 2008

Ajax Workshop

Part 3- Events

| Event | IE 5.5 | IE 6 | IE 7 | IE8b1 | FF 2 | FF 3b5 | Saf 3.0 Win | Saf 3.1 Win | Opera 9.26 | Opera 9.5b | Konqueror 3.5.7 |
|---|--------|------|---------|-------|----------|--------|-------------|-------------|------------|------------|-----------------|
| <p>blur</p> <p>When an element loses the focus.</p> <ul style="list-style-type: none"> Firefox 2 fires too many events in a variety of circumstances. Firefox 3 fires too many events when blurring the window. Safari and Opera don't support these events on links and/or form fields in all circumstances. Konqueror doesn't support these events on the browser window. | yes | yes | yes | yes | too many | almost | incomplete | almost | incomplete | incomplete | incomplete |
| <p>change</p> <p>When a form field value changes.</p> <ul style="list-style-type: none"> IE has a serious bug in its handling of this event on checkboxes and radios. | buggy | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| <p>click</p> <p>When a mousedown and mouseup event occur on the same element OR an element is activated by the keyboard.</p> | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes | yes |
| Event | IE 5.5 | IE 6 | IE 7 | IE8b1 | FF 2 | FF 3b5 | Saf 3.0 Win | Saf 3.1 Win | Opera 9.26 | Opera 9.5b | Konqueror 3.5.7 |
| <p>contextmenu</p> <p>When the user right-clicks to get the context menu.</p> <p>Preventing the default (i.e. preventing the context menu from appearing) is the whole point of this event.</p> | yes | yes | minimal | yes | yes | buggy | yes | yes | no | no | no |

| Event | IE 5.5 | IE 6 | IE 7 | IE8b1 | FF 2 | FF 3b5 | Saf 3.0
Win | Saf 3.1
Win | Opera
9.26 | Opera
9.5b | Konqueror 3.5.7 |
|-------------|--------|------|------|-------|----------|--------|----------------|----------------|---------------|---------------|-----------------|
| <u>blur</u> | | yes | | | too many | almost | incomplete | almost | incomplete | | incomplete |

When an element loses the focus.

- Firefox 2 fires too many events in a variety of circumstances.
- Firefox 3 fires too many events when blurring the window.
- Safari and Opera don't support these events on links and/or form fields in all circumstances.
- Konqueror doesn't support these events on the browser window.

| | | | | | | | | | | | |
|---------------|-------|--|--|--|-----|--|-----|--|-----|--|-----|
| <u>change</u> | buggy | | | | yes | | yes | | yes | | yes |
|---------------|-------|--|--|--|-----|--|-----|--|-----|--|-----|

When a form field value changes.

- IE has a serious bug in its handling of this event on checkboxes and radios.

<http://quirksmode.org/dom/events/>

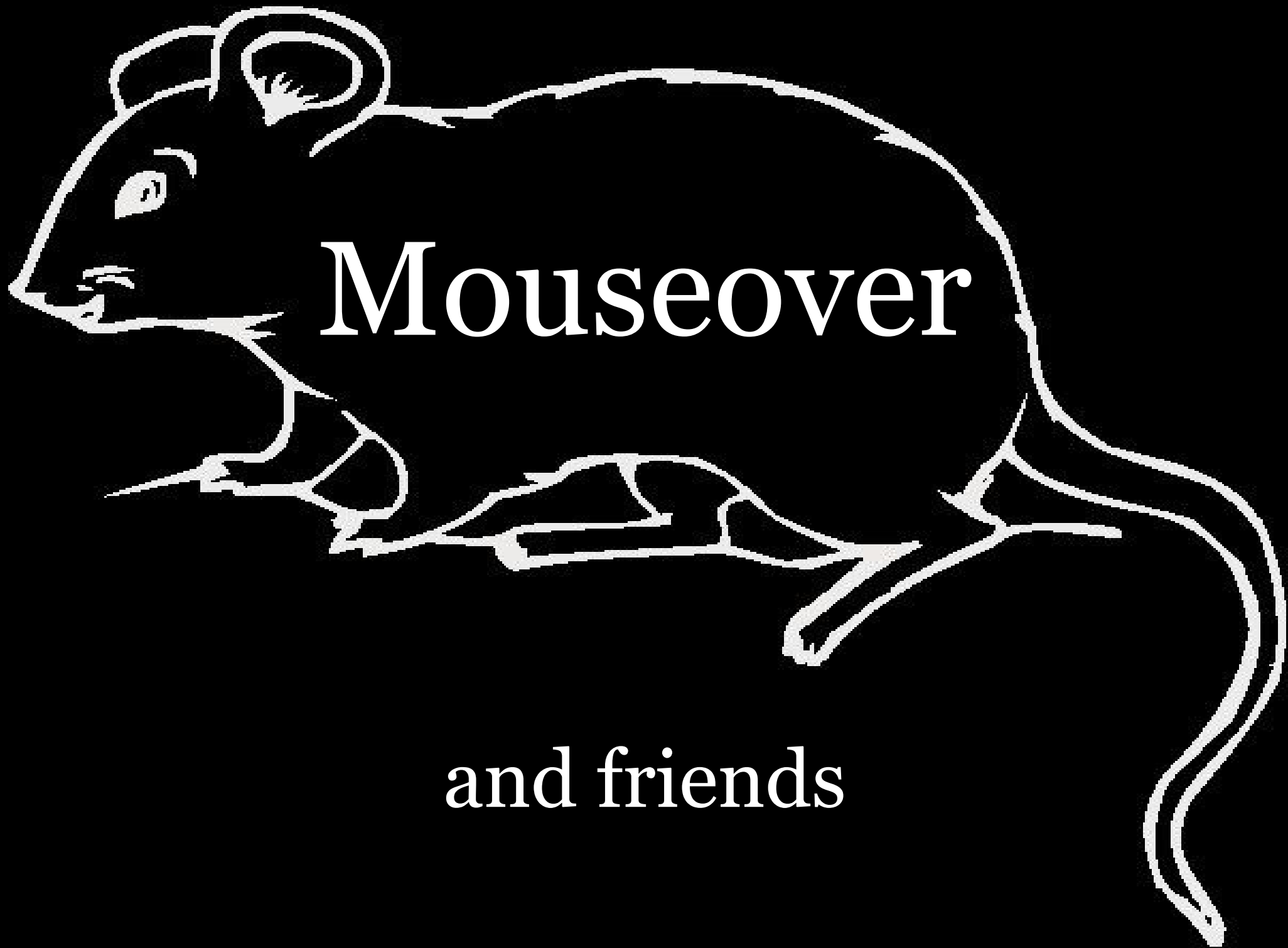
| | | | | | | | | | | | |
|--------------|-----|--|--|--|-----|--|-----|--|-----|--|-----|
| <u>click</u> | yes | | | | yes | | yes | | yes | | yes |
|--------------|-----|--|--|--|-----|--|-----|--|-----|--|-----|

When a mousedown and mouseup event occur on the same element OR an element is activated by the keyboard.

| Event | IE 5.5 | IE 6 | IE 7 | IE8b1 | FF 2 | FF 3b5 | Saf 3.0
Win | Saf 3.1
Win | Opera
9.26 | Opera
9.5b | Konqueror 3.5.7 |
|--------------------|--------|------|------|---------|------|--------|----------------|----------------|---------------|---------------|-----------------|
| <u>contextmenu</u> | | yes | | minimal | yes | buggy | yes | | no | | no |

When the user right-clicks to get the context menu.

Preventing the default (i.e. preventing the context menu from appearing) is the whole point of this event.



Mouseover

and friends

The mouseover event fires when the user's mouse enters an element .

The mouseout event fires when the user's mouse leaves an element.

Perfect support

The logo for the World Wide Web Consortium (W3C), consisting of the letters 'W3C' in a blue, sans-serif font.

Dropdown menu <sigh />

```
<ul>
```

```
<li><a href="#">Multimedialize</a>
```

```
<ul>
```

```
<li><a href="#">Sound</a></li>
```

```
<li><a href="#">Java applets</a></li>
```

```
</ul></li>
```

```
<li><a href="#">Ajaxify</a>
```

```
<ul>
```

```
<li><a href="#">Web 2.0</a></li>
```

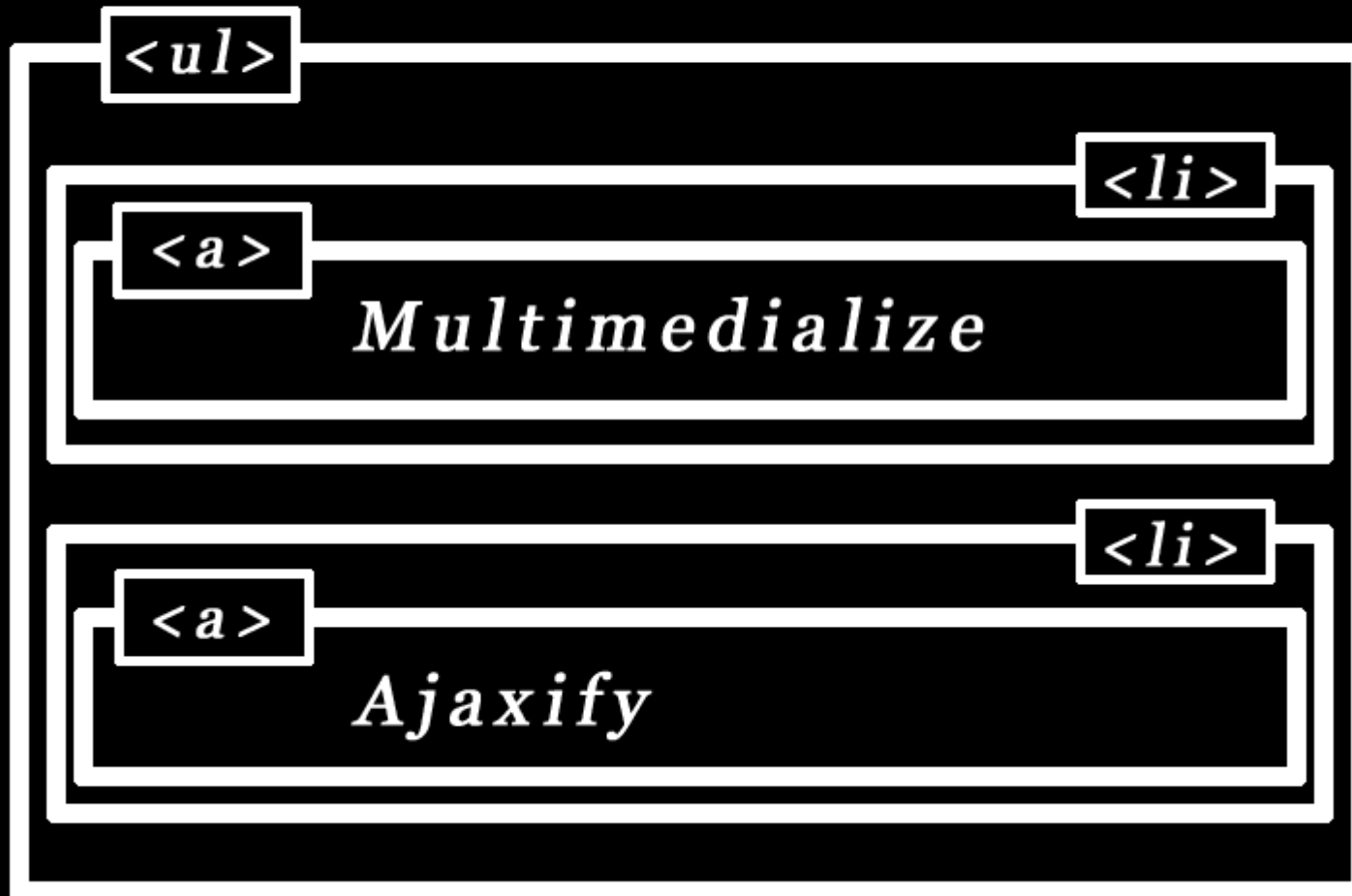
```
<li><a href="#">Web 3.0</a></li>
```

```
<li><a href="#">Web 4.0b</a></li>
```

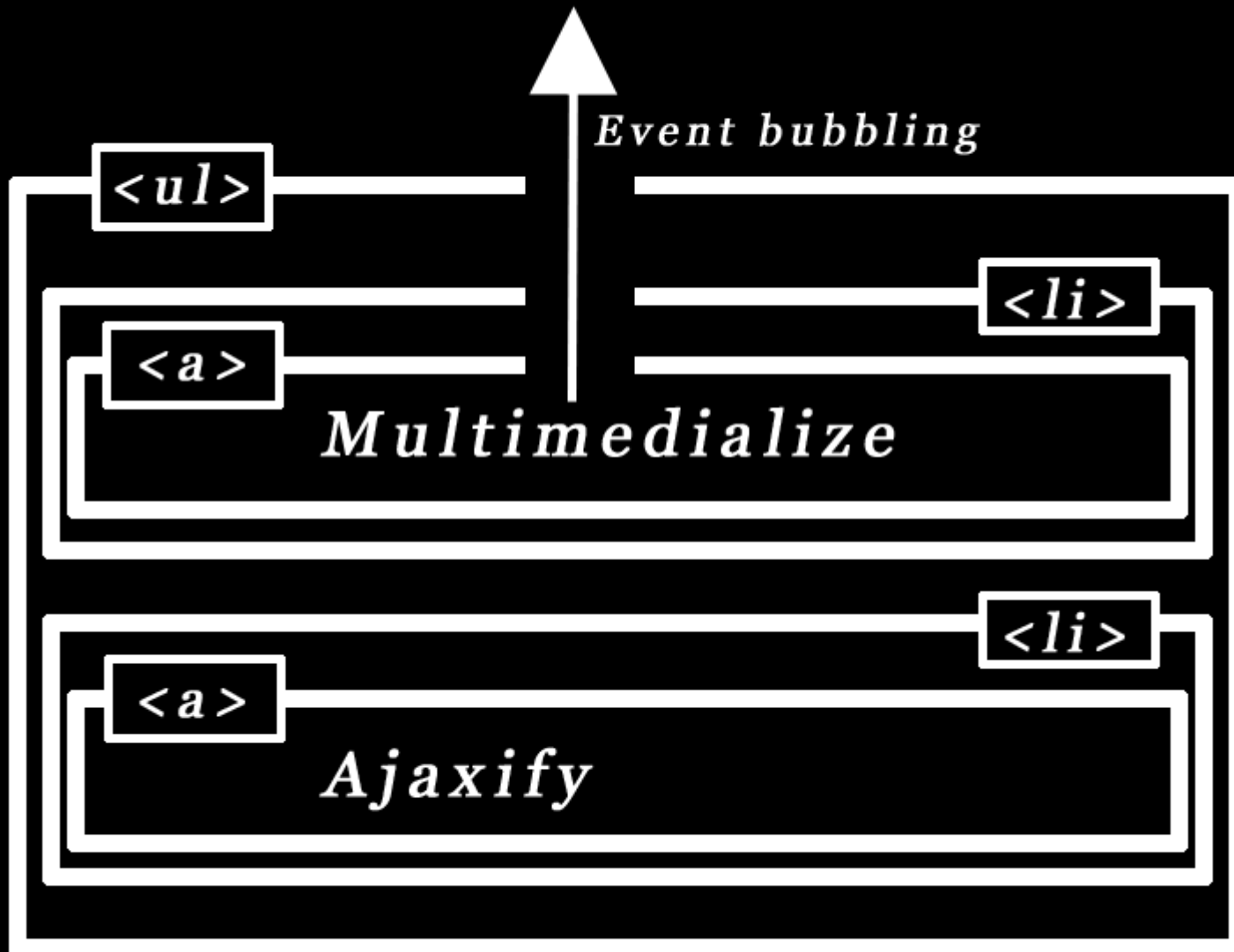
```
</ul></li>
```

```
</ul>
```

Dropdown menu <sigh />



Dropdown menu <sigh />



Dropdown menu <sigh />

Event bubbling has advantages.

```
var dropdown = {  
  init: function (dropdown) {  
    var x = dropdown.getElementsByTagName('a');  
    for (var i=0;i<x.length;i++) {  
      x[i].onmouseover = mouseOver;  
      x[i].onmouseout = mouseOut;  
    }  
  }  
}
```

Dropdown menu <sigh />

Event bubbling has advantages.

```
var dropdown = {  
  init: function (dropdown) {  
    var x = dropdown.getElementsByTagName('a');  
    for (var i=0;i<x.length;i++){  
      x[i].onmouseover = mouseOver;  
      x[i].onmouseout = mouseOut;  
    }  
  }  
}
```



Dropdown menu <sigh />

Event bubbling has advantages.

```
var dropdown = {  
  init: function (dropdown) {
```

We don't do this any more. Instead
we use event delegation.

```
  }  
}
```

Dropdown menu <sigh />

The event bubbles up to the
anyway.

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = mouseOver;  
    dropdown.onmouseout = mouseOut;  
  }  
}
```

So why not handle it at that level?

Saves a lot of event handlers.

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = mouseOver;  
    dropdown.onmouseout = mouseOut;  
  }  
}
```

Works in all browsers.

The logo for the World Wide Web Consortium (W3C), consisting of the letters 'W3C' in a blue, sans-serif font.

Exercise:

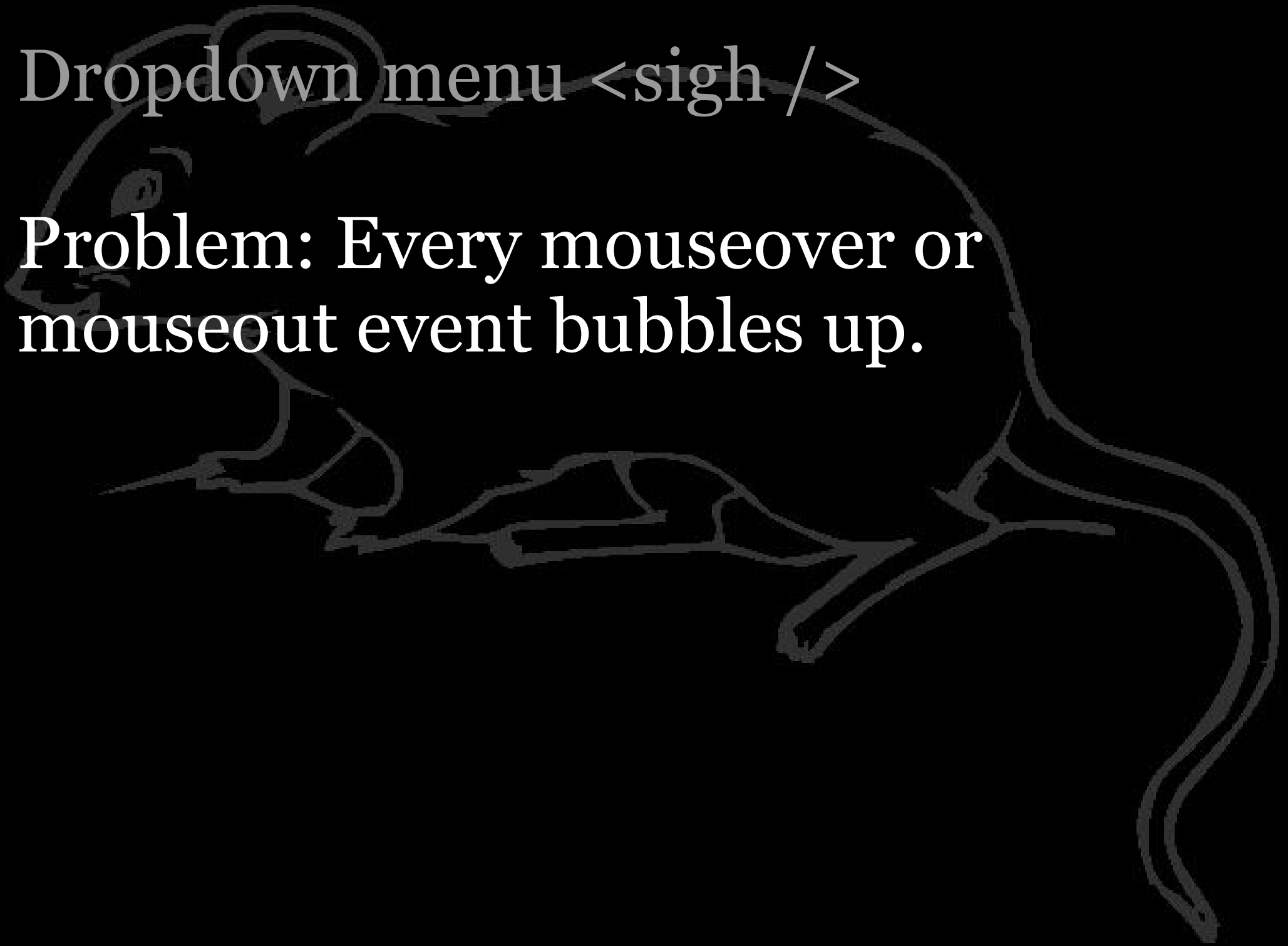
fundamentos
web 2008

Do you use event delegation?

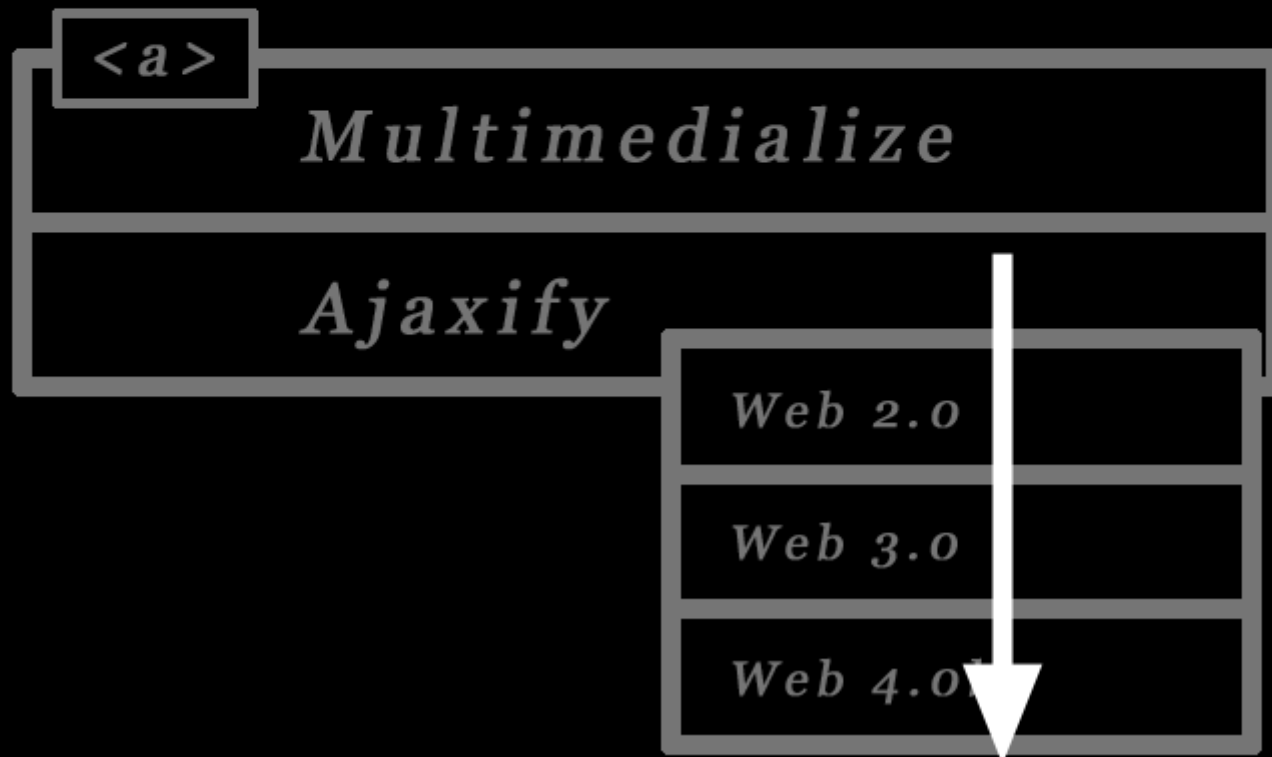
If not, how many event handlers
would you save if you did use it?

Dropdown menu <sigh />

Problem: Every mouseover or mouseout event bubbles up.



Dropdown menu <sigh />



Dropdown menu <sigh />

a.mouseover

a.mouseout and a.mouseover

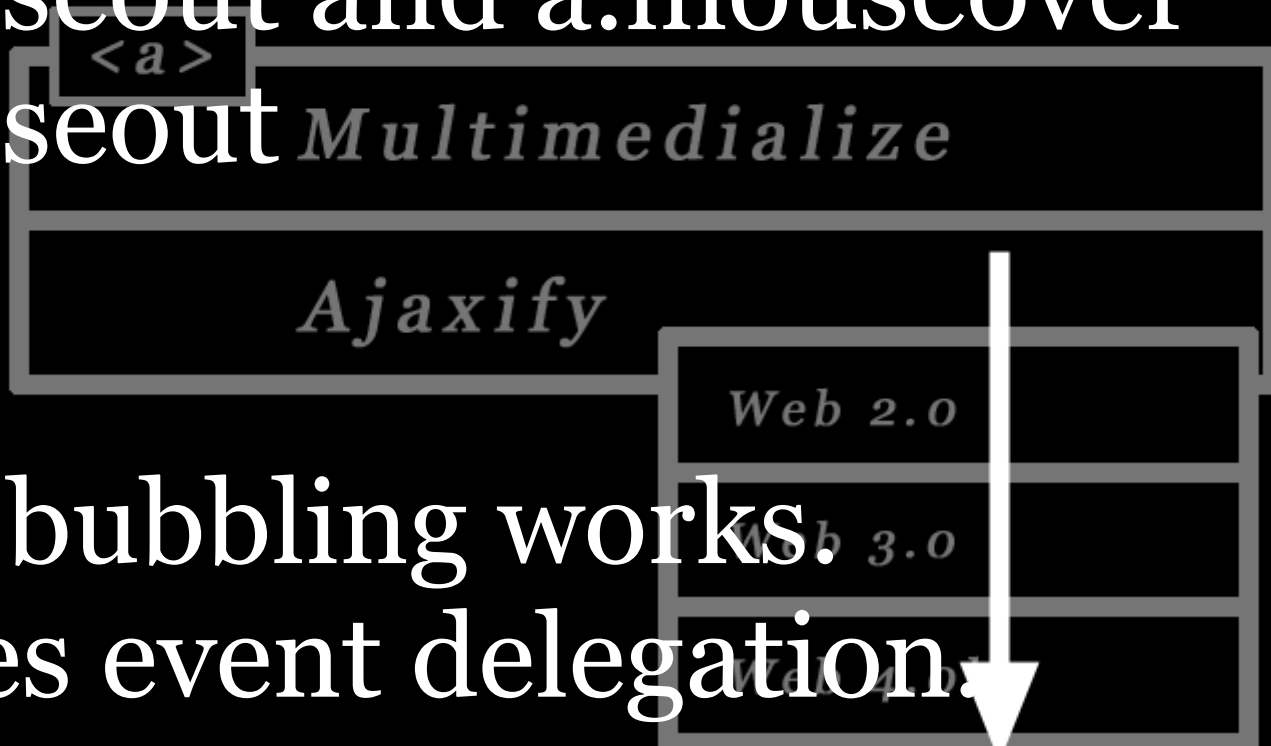
a.mouseout and a.mouseover

a.mouseout *Multimedialize*

Fun!

Event bubbling works.

As does event delegation.





Dropdown menu <sigh />

a.mouseover

a.mouseout and a.mouseover

a.mouseout and a.mouseover

a.mouseout

But has the mouse left the submenu or not?!

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  },  
  mouseOut: function (e) {  
    if (this.mouseout is important) {  
      this.closeSubMenu();  
    }  
  }  
}
```

Dropdown menu <sigh />

```
var dropdown = {
  init: function (dropdown) {
    dropdown.onmouseover = this.mouseOver;
    dropdown.onmouseout = this.mouseOut;
  },
  mouseOut: function (e) {
    if (this.mouseout is important) {
      this.closeSubMenu();
    }
  }
}
```

Development time: about 10 minutes

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  },  
  mouseOut: function (e) {  
    if (this.mouseout is important) {  
      this.closeSubMenu();  
    }  
  }  
}
```

Development time: about 2 days

Dropdown menu <sigh />

How do we do this?

onmouseout, find out which element the mouse goes *to*.

If that element is *not* a part of the submenu, fold the submenu.



Dropdown menu <sigh />

How do we do this?

```
mouseout: function (e) {  
  e = e || window.event;  
  var el = e.relatedTarget || e.toElement;  
  if (!submenu.contains(el)) {  
    this.closeSubMenu();  
  }  
}
```



Dropdown menu <sigh />

Find the element the mouse goes to.

```
mouseout: function (e) {  
  e = e || window.event;  
  var el = e.relatedTarget || e.toElement;  
  if (!submenu.contains(el)) {  
    this.closeSubMenu();  
  }  
}
```


Dropdown menu <sigh />

Find the element the mouse goes to.

```
mouseout: function (e) {  
  e = e || window.event;  
  var el = e.relatedTarget || e.toElement;  
  if (!submenu.contains(el)) {  
    this.closeSubMenu();  
  }  
}
```



Dropdown menu <sigh />

Find the element the mouse goes to.

```
mouseout: function (e) {  
    e = e || window.event;  
    var el = e.relatedTarget || e.toElement;  
    if (!submenu.contains(el)) {  
        this.closeSubMenu();  
    }  
}
```



Dropdown menu <sigh />

See whether that element is contained
by the submenu.

```
mouseout: function (e) {  
  e = e || window.event;  
  var el = e.relatedTarget || e.toElement;  
  if (!submenu.contains(el)) {  
    this.closeSubMenu();  
  }  
}
```

Dropdown menu <sigh />

See whether that element is contained by the submenu.

```
mouseout: function (e) {  
    e = e || window.event;  
    var el = e.relatedTarget || e.toElement;  
    if (!submenu.contains(el)) {  
        this.closeSubMenu();  
    }  
}
```



Dropdown menu <sigh />

```
var dropdown = {
  init: function (dropdown) {
    dropdown.onmouseover = this.mouseOver;
    dropdown.onmouseout = this.mouseOut;
  },
  mouseOut: function (e) {
    e = e || window.event;
    var el = e.relatedTarget || e.toElement;
    if (!submenu.contains(el)) {
      this.closeSubMenu();
    }
  }
}
```

Dropdown menu <sigh />

That's it, right?

```
<grin type="evil" />
```

```
mouseOut: function (e) {  
  e = e || window.event;  
  var el = e.relatedTarget || e.toElement;  
  if (!submenu.contains(el)) {  
    this.closeSubMenu();  
  }  
}
```

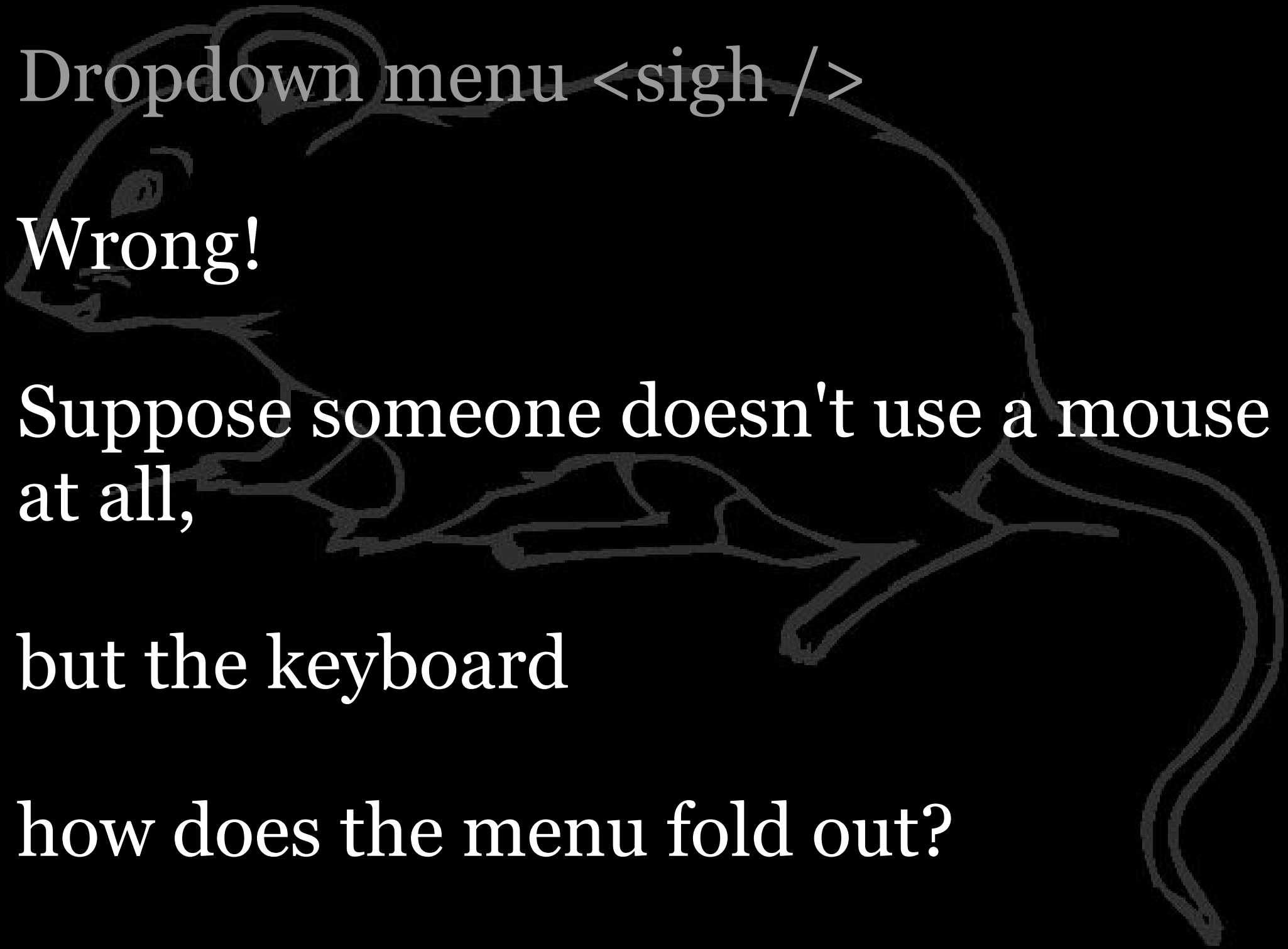
Dropdown menu <sigh />

Wrong!

Suppose someone doesn't use a mouse
at all,

but the keyboard

how does the menu fold out?



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!

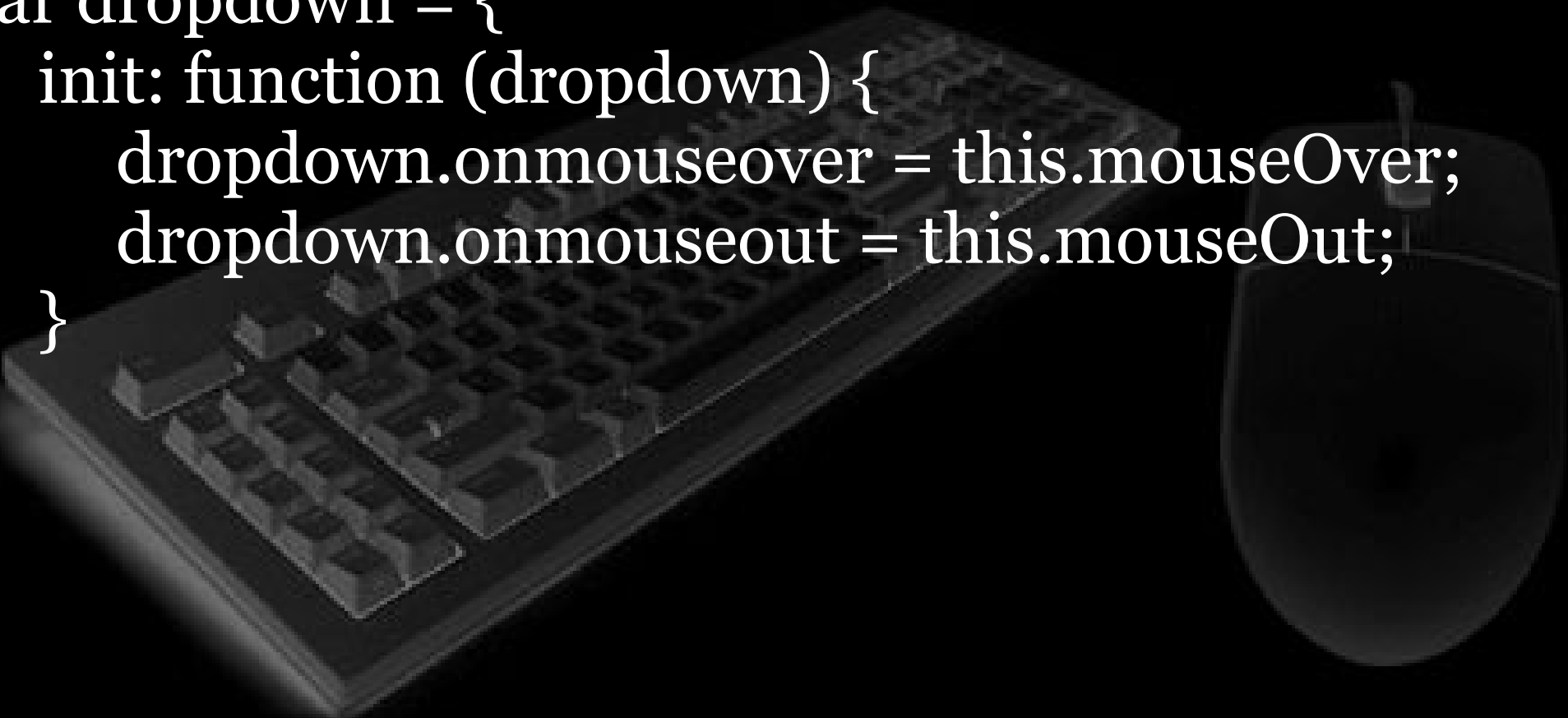


A computer keyboard and mouse are shown on a black background. The keyboard is on the left, and the mouse is on the right. The text "Device independence" is overlaid in the center in a white serif font.

Device
independence

Dropdown menu < sigh />

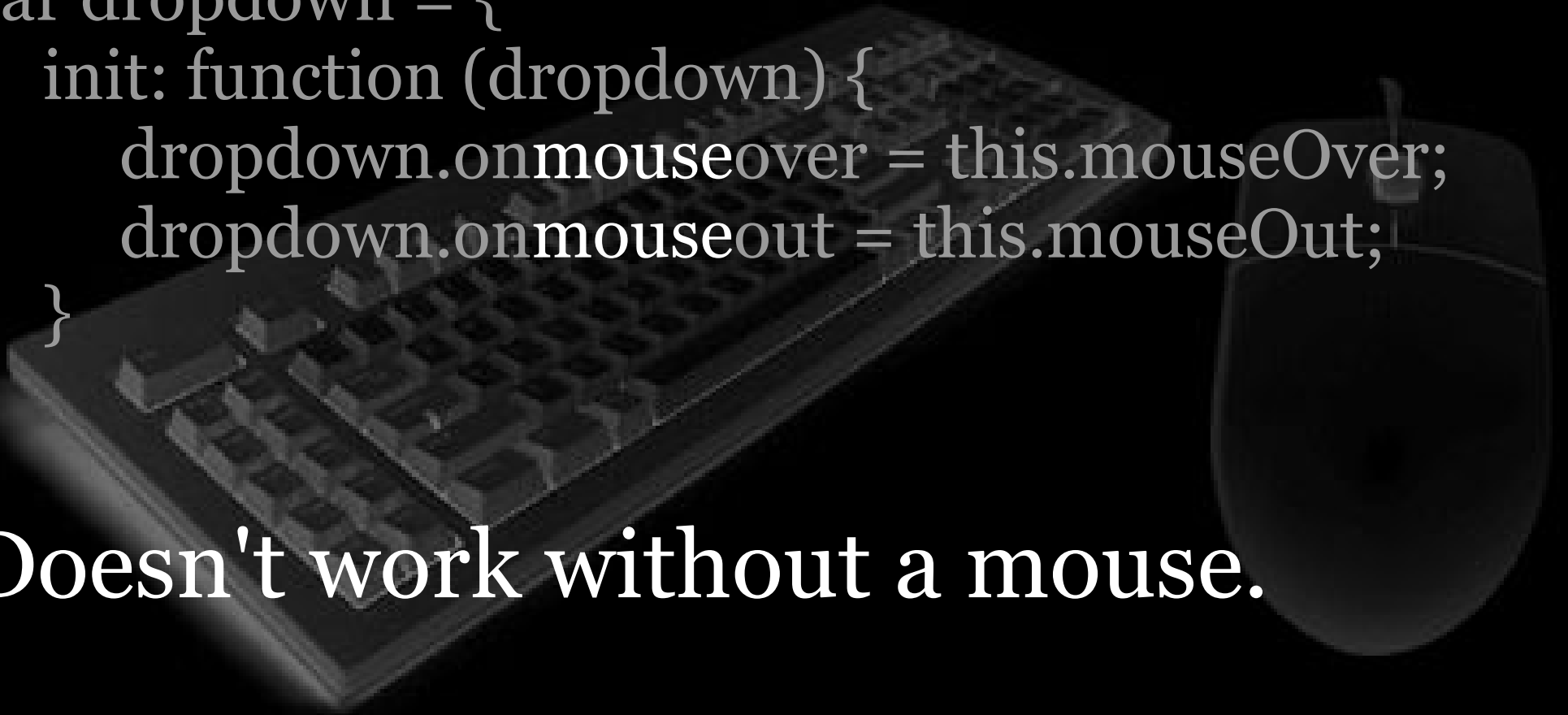
```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  }  
}
```

A computer keyboard and mouse are visible in the background, rendered in a dark, semi-transparent style. The keyboard is on the left and the mouse is on the right.

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  }  
}
```

Doesn't work without a mouse.




Exercise:

fundamentos
web 2008

Do you use mouseover and mouseout without paying attention to keyboard users?

Dropdown menu <sigh />

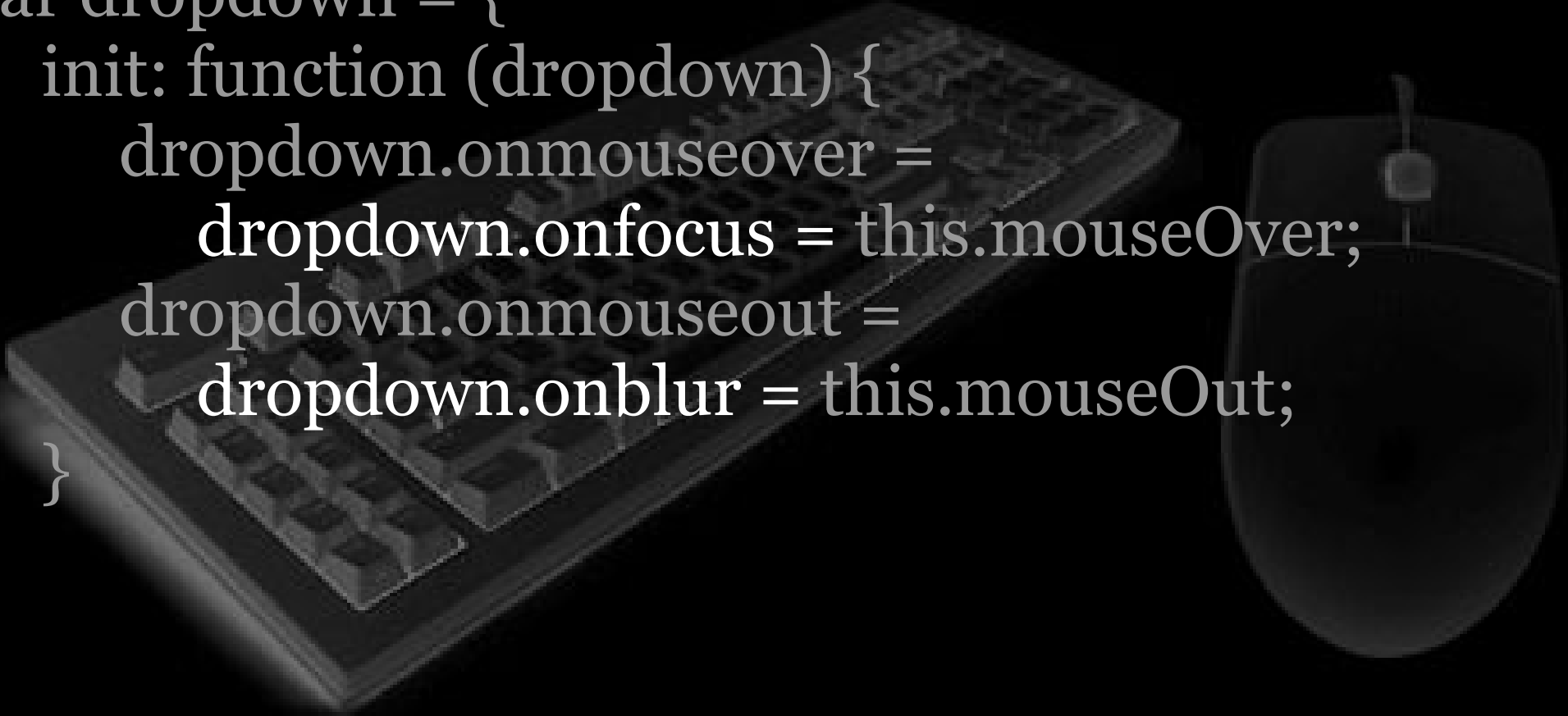
```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  }  
}
```



We need events that tell us whether the user enters or leaves a link.
focus and blur

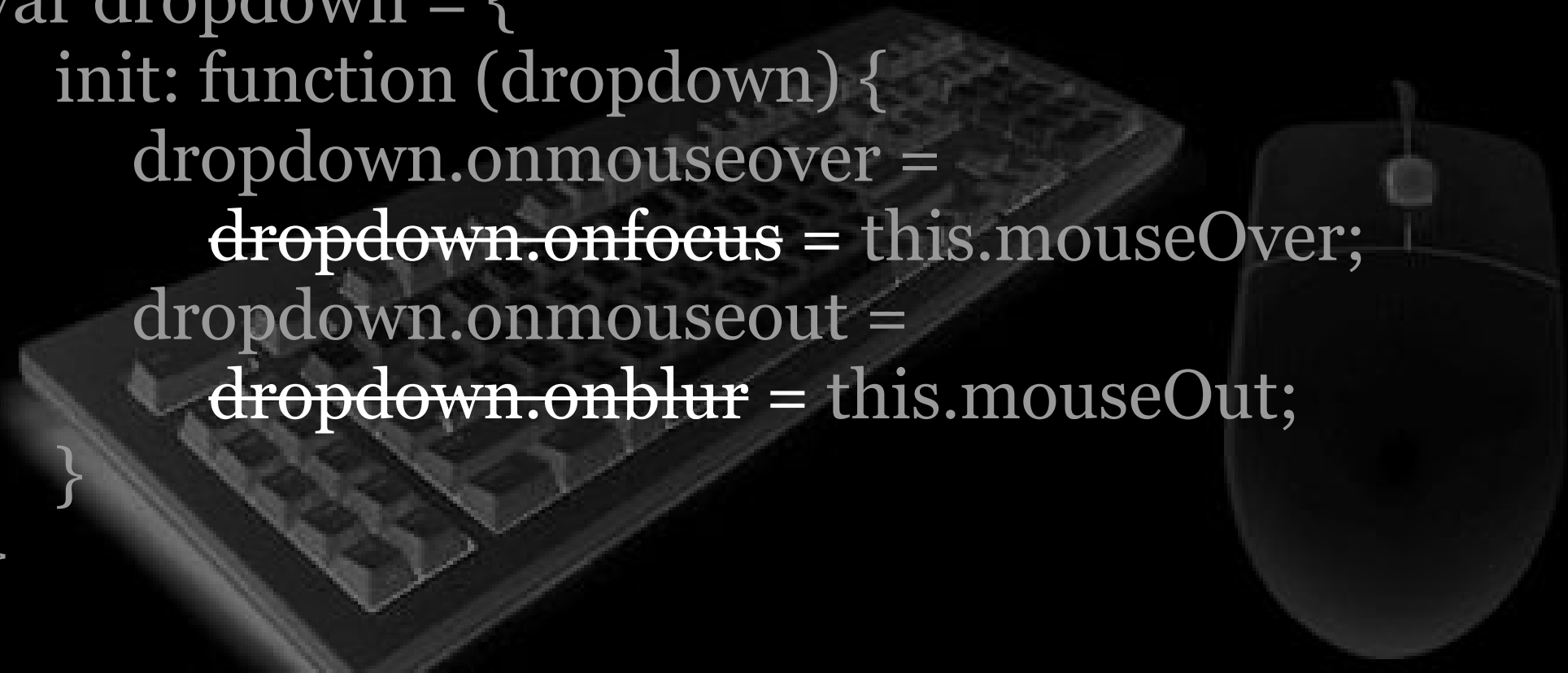
Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover =  
      dropdown.onfocus = this.mouseOver;  
    dropdown.onmouseout =  
      dropdown.onblur = this.mouseOut;  
  }  
}
```



Dropdown menu <sigh />

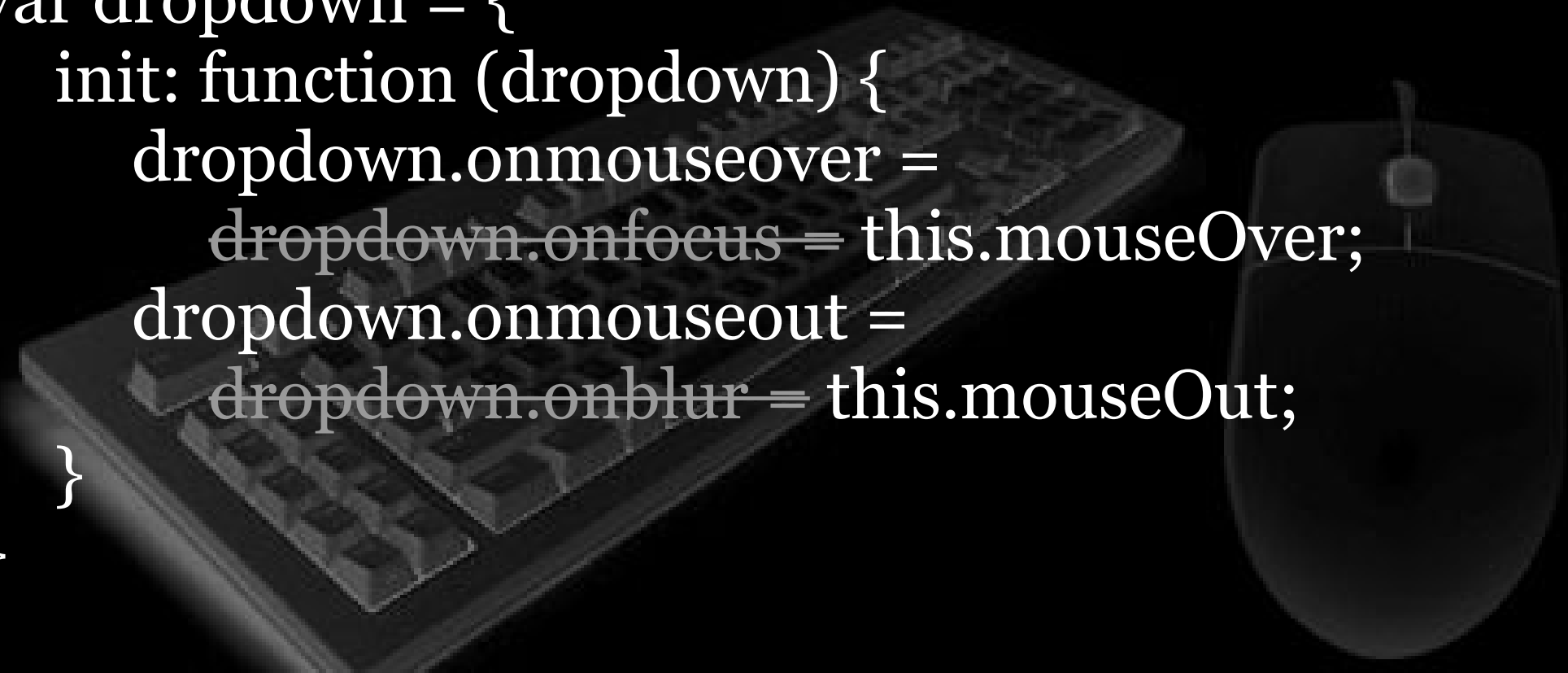
```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover =  
    dropdown.onfocus = this.mouseOver;  
    dropdown.onmouseout =  
    dropdown.onblur = this.mouseOut;  
  }  
}
```



Doesn't work.

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover =  
      dropdown.onfocus = this.mouseOver;  
    dropdown.onmouseout =  
      dropdown.onblur = this.mouseOut;  
  }  
}
```



Focus and blur don't bubble.

To bubble or not to bubble

Two kinds of events:

- 1) Mouse and key events
- 2) Interface events



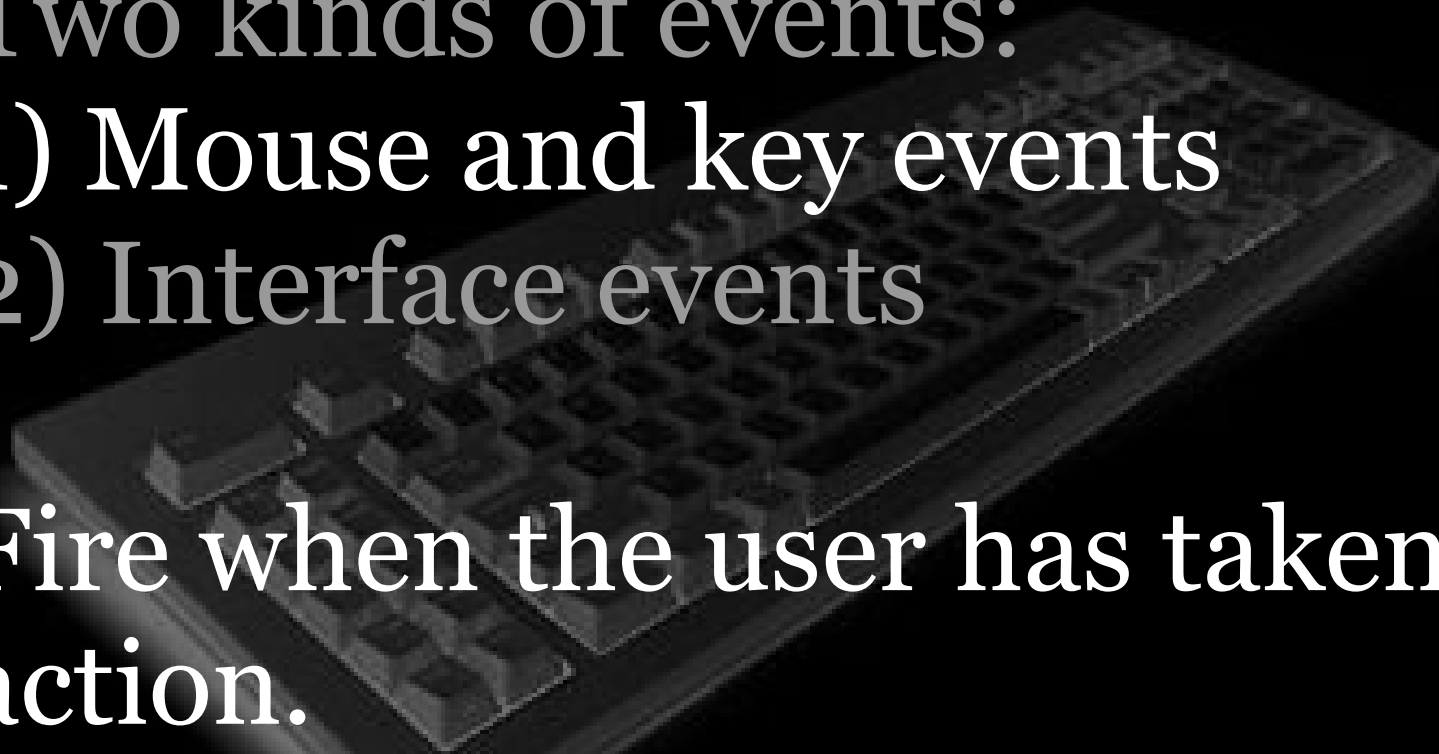
To bubble or not to bubble

Two kinds of events:

- 1) Mouse and key events
- 2) Interface events

Fire when the user has taken a certain action.

mouseover, mouseout, click, keydown, keypress

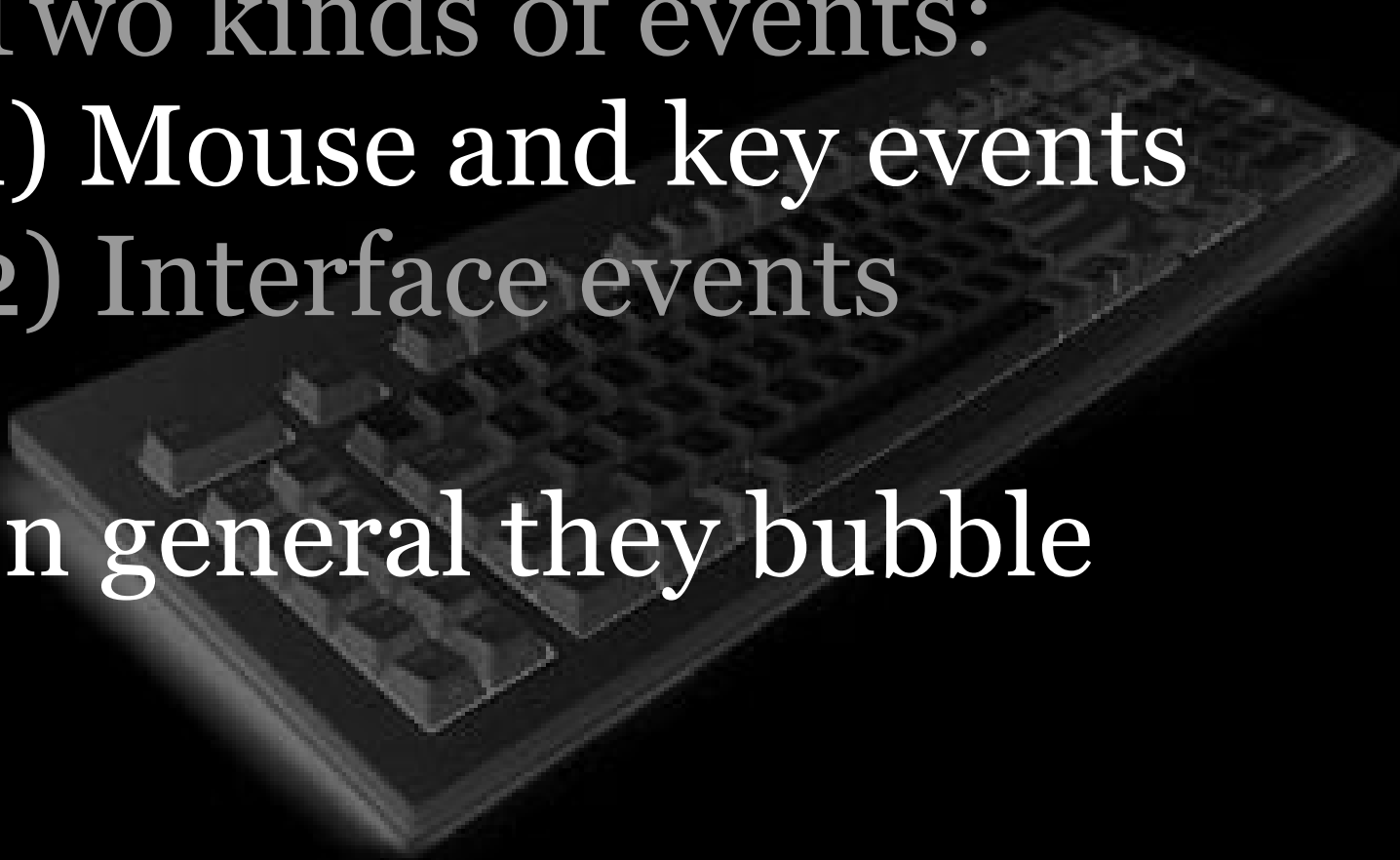


To bubble or not to bubble

Two kinds of events:

- 1) Mouse and key events
- 2) Interface events

In general they bubble



To bubble or not to bubble

Two kinds of events:

- 1) Mouse and key events
- 2) Interface events

Fire when a certain event takes place, regardless of how it was initialised.
load, change, submit, focus, blur



To bubble or not to bubble

Two kinds of events:

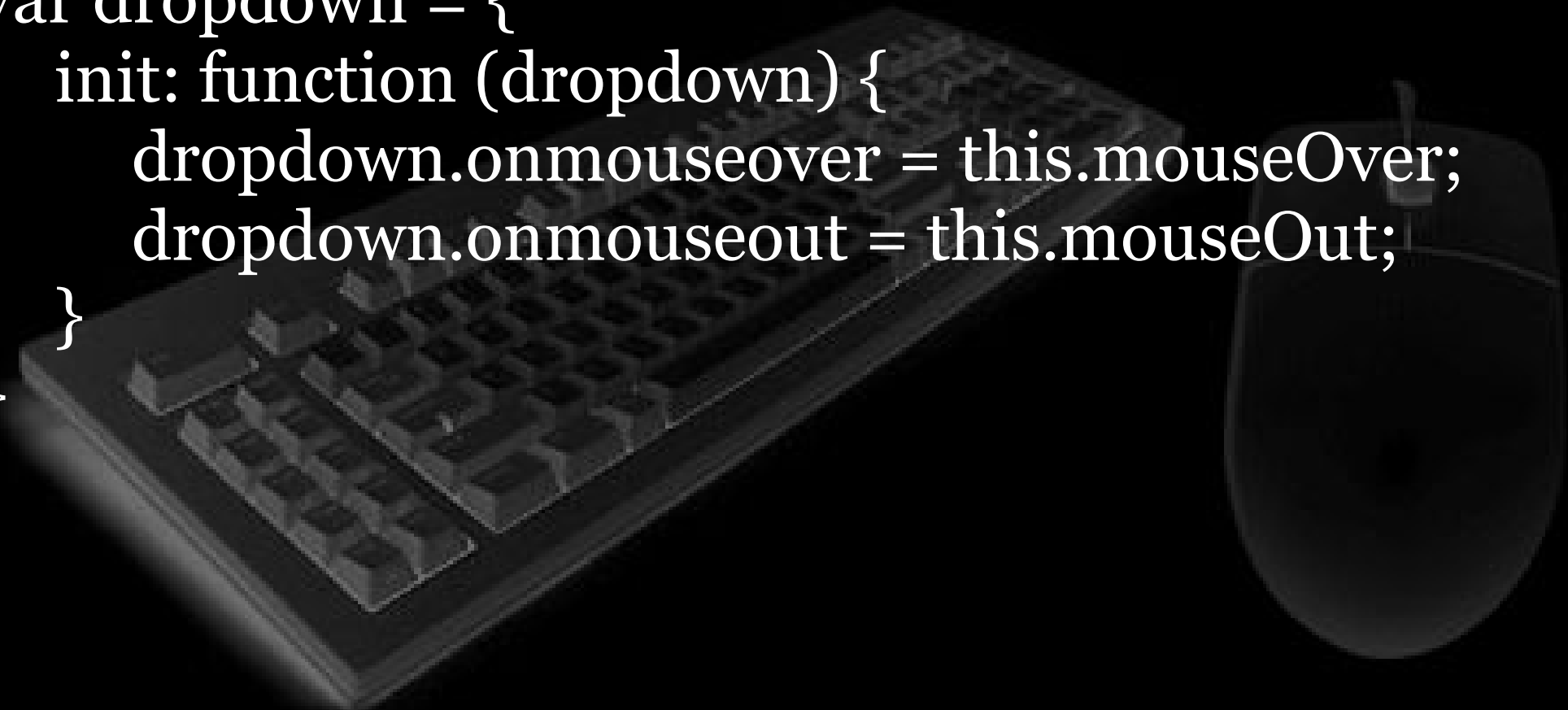
- 1) Mouse and key events
- 2) Interface events

Generally don't bubble



Dropdown menu < sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
  }  
}
```



Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
    var x = dropdown.getElementsByTagName('li');  
    for (var i=0;i<x.length;i++) {  
      x[i].onfocus = this.mouseOver;  
      x[i].onblur = this.mouseOut;  
    }  
  }  
}
```


Dropdown menu <sigh />

```
var dropdown = {
  init: function (dropdown) {
    dropdown.onmouseover = this.mouseOver;
    dropdown.onmouseout = this.mouseOut;
    var x = dropdown.getElementsByTagName('li');
    for (var i=0;i<x.length;i++) {
      x[i].onfocus = this.mouseOver;
      x[i].onblur = this.mouseOut;
    }
  }
}
```

Doesn't work.

Dropdown menu <sigh />

The HTML elements must be able to receive the keyboard focus.

- links
- form fields



Dropdown menu <sigh />

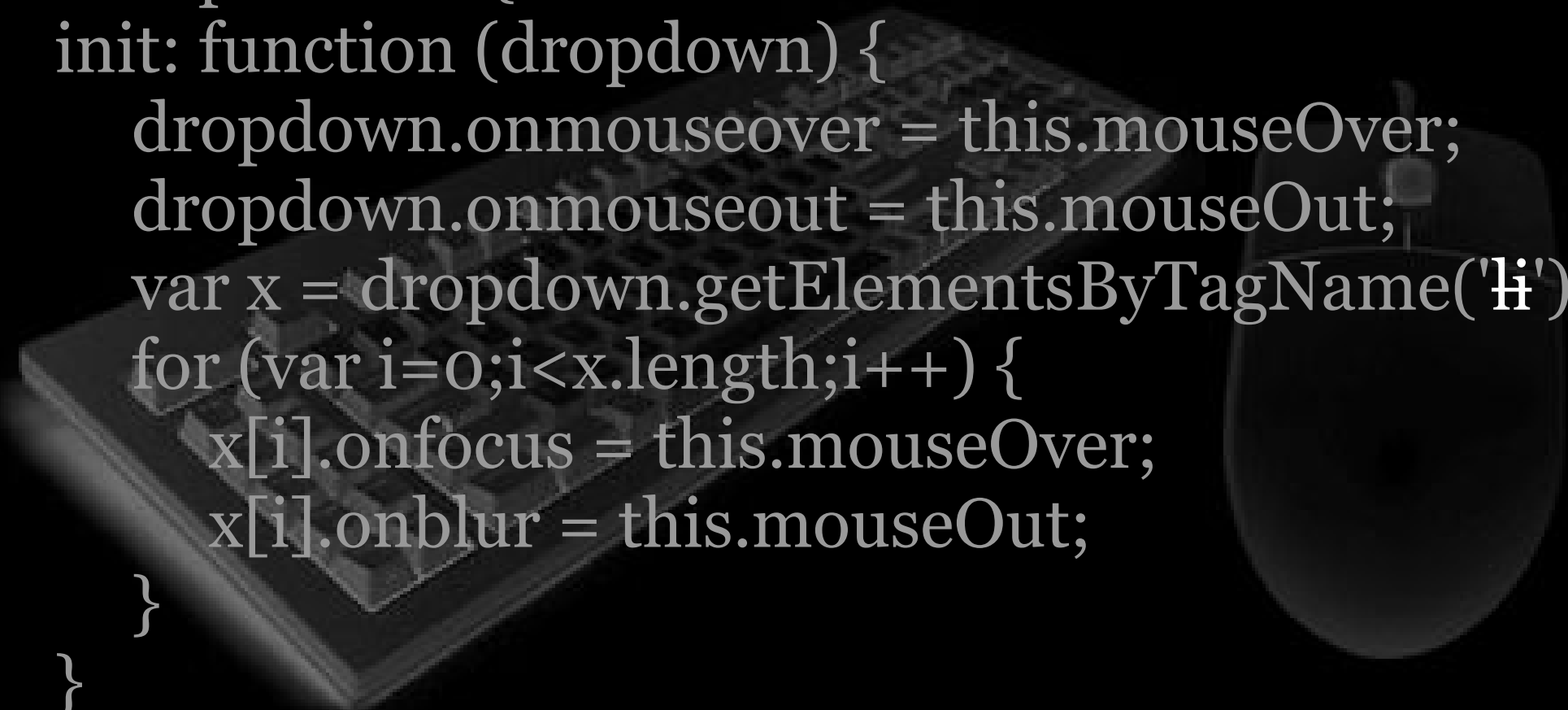
The HTML elements must be able to receive the keyboard focus.

- links
- form fields
- elements with tabindex



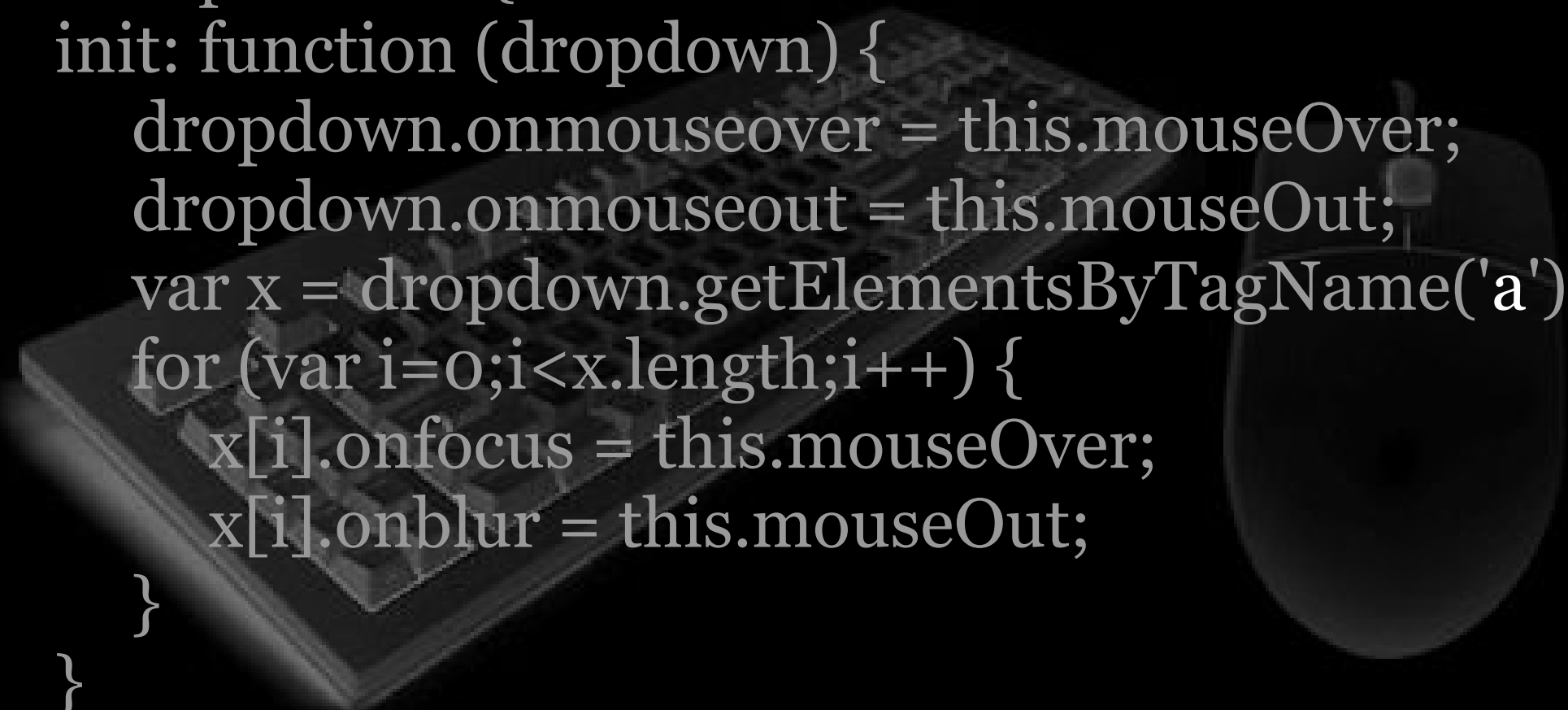
Dropdown menu <sigh />

```
var dropdown = {
  init: function (dropdown) {
    dropdown.onmouseover = this.mouseOver;
    dropdown.onmouseout = this.mouseOut;
    var x = dropdown.getElementsByTagName('li');
    for (var i=0;i<x.length;i++) {
      x[i].onfocus = this.mouseOver;
      x[i].onblur = this.mouseOut;
    }
  }
}
```



Dropdown menu <sigh />

```
var dropdown = {
  init: function (dropdown) {
    dropdown.onmouseover = this.mouseOver;
    dropdown.onmouseout = this.mouseOut;
    var x = dropdown.getElementsByTagName('a');
    for (var i=0;i<x.length;i++) {
      x[i].onfocus = this.mouseOver;
      x[i].onblur = this.mouseOut;
    }
  }
}
```

A faint, grayscale background image of a computer keyboard and mouse is visible behind the text. The keyboard is on the left and the mouse is on the right, both appearing as semi-transparent overlays.

Dropdown menu <sigh />

```
var dropdown = {  
  init: function (dropdown) {  
    dropdown.onmouseover = this.mouseOver;  
    dropdown.onmouseout = this.mouseOut;  
    var x = dropdown.getElementsByTagName('a');  
    for (var i=0;i<x.length;i++) {  
      x[i].onfocus = this.mouseOver;  
      x[i].onblur = this.mouseOut;  
    }  
  }  
}
```

And what about click?

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

Restriction:
the element must be able to receive
the keyboard focus



Exercise:

fundamentos
web 2008

If you added focus and blur events, would they call the same function, or would you have to write new ones?



The key events

A dark, close-up photograph of a computer keyboard, showing several keys in detail. The keys are dark with light-colored characters. The background is a dark, slightly blurred grid of keys.

keydown

When a key is depressed.

Repeats.

keypress

keyup



keydown

When a key is depressed.

Repeats.

keypress

When a *character* key is depressed.

Repeats.

keyup



keydown

When a key is depressed.

Repeats.

keypress

When a *character* key is depressed.

Repeats.

keyup

When a key is released.



keydown and keypress

keydown only



Originally this theory was created by Microsoft.

Recently Safari 3.1 has copied it.

It's the only theory; Firefox and Opera just fire some random events.

keydown

When a key is depressed.

Repeats.

keypress

When a *character* key is depressed.

Repeats.



Exercise:

fundamentos
web 2008

Which key events do you use?
Keydown or keypress? Why?



Which key did my user press?

Two properties:
keyCode and charCode

Two bits of data:
- the key code
- the character code

Which key did my user press?

Obviously, having one property contain one bit of data and the other property the other

would be far too simple.

Which key did my user press?

Two properties:
keyCode and charCode

And what about W3C?

Which key did my user press?

Two properties:
~~keyCode~~ and ~~charCode~~
keyIdentifier

And what about W3C?

Which key did my user press?

keyCode

- onkeydown: key code
- onkeypress: ASCII value



Which key did my user press?

charCode

- onkeydown: 0
- onkeypress: ASCII value



Which key did my user press?

keyIdentifier

- A name, such as “Shift”, or a code such as “U+000041” (hexadecimal 65) for “a”

W3C



Which key did my user press?

If you need the key:

```
el.onkeydown = function (e) {  
  e = e || window.event;  
  var realKey = e.keyCode;  
}
```

Which key did my user press?

If you need the key:

```
el.onkeydown = function (e) {  
  e = e || window.event;  
  var realKey = e.keyCode;  
}
```



Which key did my user press?

If you need the character:

```
el.onkeypress = function (e) {  
  e = e || window.event;  
  var char = e.keyCode || e.charCode;  
}
```

Which key did my user press?

If you need the character:

```
el.onkeypress = function (e) {  
  e = e || window.event;  
  var char = e.keyCode || e.charCode;  
}
```



How can I prevent the default action?

```
el.onkeydown = function (e) {  
  e = e || window.event;  
  var key = e.keyCode;  
  if (key is incorrect) {  
    // cancel default action  
  }  
}
```

How can I prevent the default action?

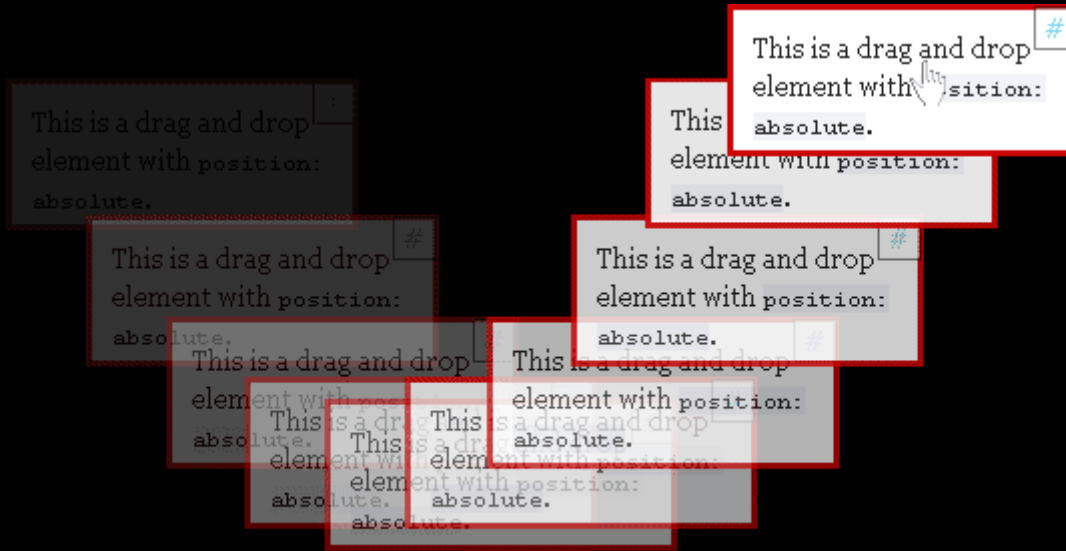
```
el.onkeydown = function (e) {  
  e = e || window.event;  
  var key = e.keyCode;  
  if (key is incorrect) {  
    // cancel default action  
  }  
}
```

W3C



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

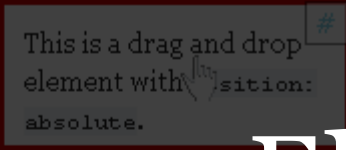

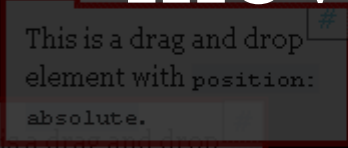
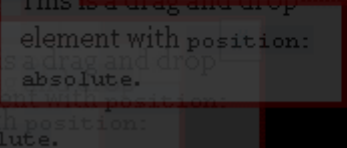
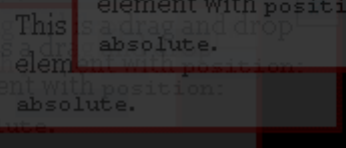
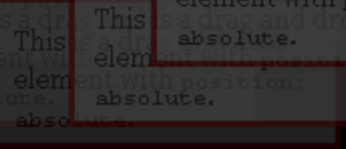
How do we make this accessible?

By allowing the user to use the arrow keys.

Key events.

Separate concepts

Drag-and-drop

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

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 = This is a drag and drop  
element with position:  
absolute.  
obj.onkeydown = This is a drag and drop  
element with position:  
absolute. 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](http://quirksmode.org/js/dragdrop.html)

Besides, I created a first draft for you.

Exercise:

fundamentos
web 2008

Do you have a mouse-driven functionality that you have to write new functions for if you make them keyboard-accessible? How would you write such new functions?

Hell is other browsers - *Sartre*

fundamentos

web 2008

Ajax Workshop

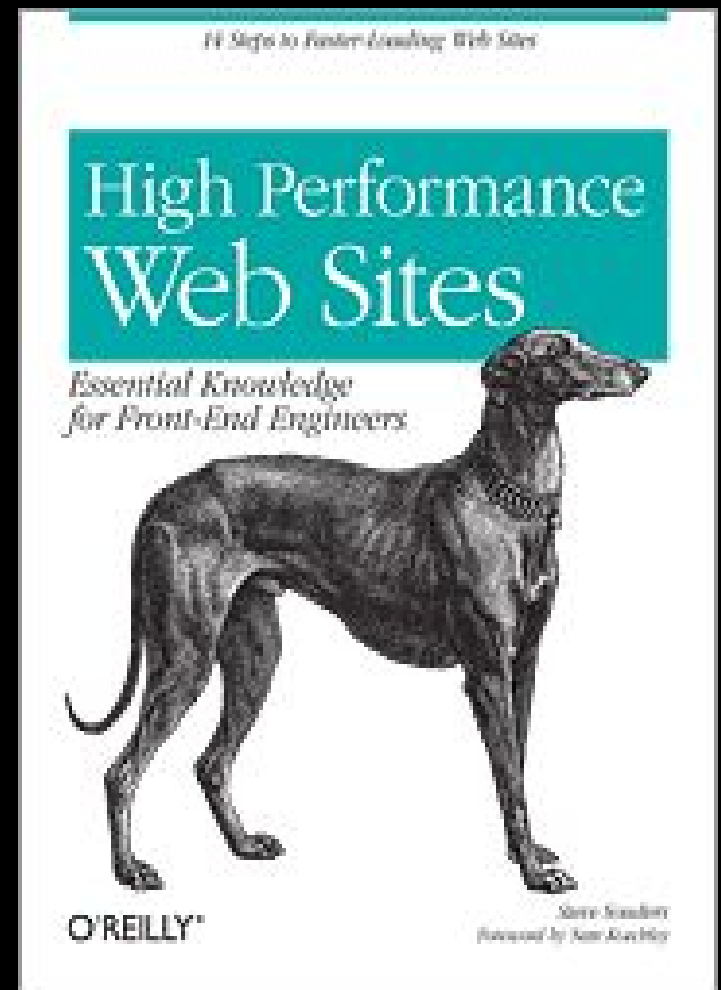
Part 4- Performance

Steve Souders

High Performance
Web Sites

Buy this book.

You'll need it.



Rules:

- 1) Make fewer HTTP requests
- 2) Use a Content Delivery Network
- 3) Add an Expires header
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 7) Avoid CSS expressions
- 8) Make JavaScript and CSS external
- 9) Reduce DNS lookups
- 10) Minify JavaScript
- 11) Avoid redirects
- 12) Remove duplicate scripts

Rules:

- 1) Make fewer HTTP requests
- ~~2) Use a Content Delivery Network~~
- ~~3) Add an Expires header~~
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- ~~7) Avoid CSS expressions~~
- 8) Make JavaScript and CSS external
- ~~9) Reduce DNS lookups~~
- ~~10) Minify JavaScript~~
- ~~11) Avoid redirects~~
- ~~12) Remove duplicate scripts~~

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

It's better to use one CSS and one JavaScript file than several of each.

We'll get back to some other tricks later.

Exercise:

fundamentos
web 2008

Determine how many HTTP requests your site makes.
Exclude dynamically loaded assets.

Exercise:

fundamentos
web 2008

Could you merge several
JavaScript files into one file?

Could you merge several CSS files
into one file?

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

All of them.

HTML, CSS, JavaScript, images.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

The browser waits until the last style sheet is loaded before rendering the page.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

Use `<link>` tags, and not `@import`.
`@imported` style sheet are loaded
LAST, which causes a blank screen
until **ALL** style sheets have been
loaded.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

When the browser loads a script, it blocks all other downloads because the script might contain a `document.write()`

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

Besides, when you put your script at the bottom you don't need an onload event handler.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

External files will be cached, so that the user will have to download them only once.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

Keep an iron grip on the assets
you have to load
and on their order of loading

Example: Project X

Site is meant for viewing images in a nice, user-friendly environment.

To the users, it's all about the images. They don't care about CSS or JavaScript.

Exercise:

fundamentos
web 2008

Determine which assets the
USERS of your site think most
important.

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image 1
- Background image 2
- Background image 3
- Background image 4
- Image 1
- Image 2
- Image 3
- Image 4
- Image 5
- Image 6
- Image 7
- Image 8

16 HTTP requests. Slooooooowwww

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image 1
- Background image 2
- Background image 3
- Background image 4
- Image 1
- Image 2
- Image 3
- Image 4
- Image 5
- Image 6
- Image 7
- Image 8

We'll get back to the images later.

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image 1
- Background image 2
- Background image 3
- Background image 4

Reduce number of HTTP requests

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image 1
- Background image 2
- Background image 3
- Background image 4

Example: Project X

Assets:

- Background image 1
- Background image 2
- Background image 3
- Background image 4

All background images in one file.

CSS Sprites.

Example: Project X

Assets:

- Background image 1
- ~~- Background image 2~~
- ~~- Background image 3~~
- ~~- Background image 4~~

All background images in one file.

CSS Sprites.

Saves 3 HTTP requests.

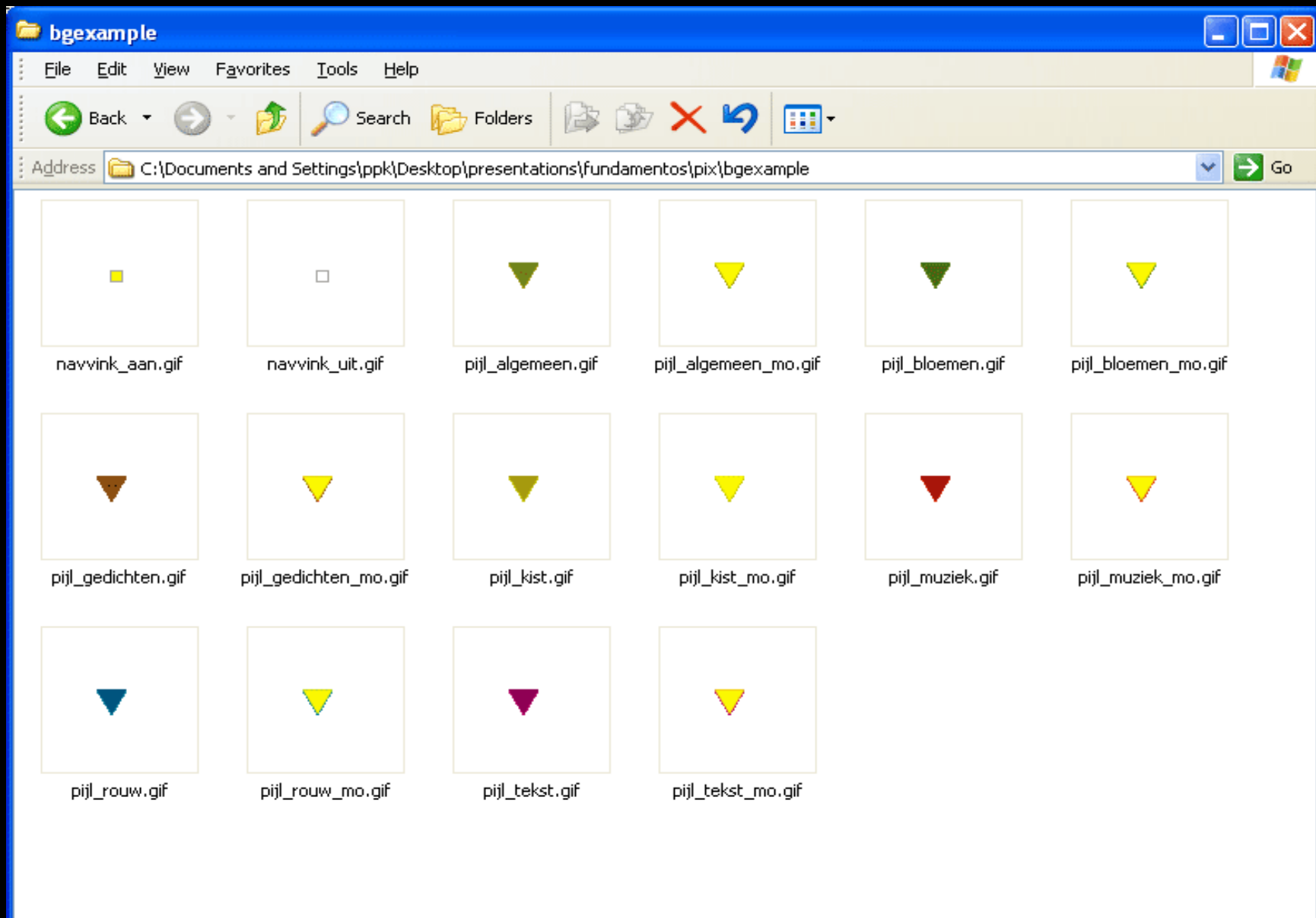
CSS Sprites

```
div.nav a {  
    background: url(pix/bg1.gif) no-repeat;  
}
```

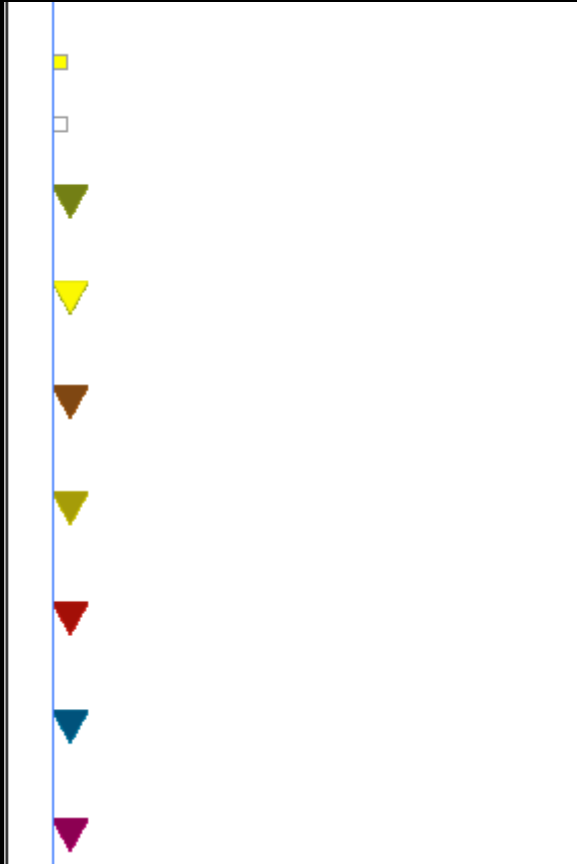
```
div.nav a:hover {  
    background: url(pix/bg2.gif) no-repeat;  
}
```

```
div.nav a.special {  
    background: url(pix/bg3.gif) no-repeat;  
}
```

CSS Sprites



CSS Sprites



sprite.gif

Use background-position
to select the part of the
sprite that's visible

CSS Sprites

```
div.nav a {  
    background: url(pix/sprite.gif) no-repeat;  
    background-position: 10px 10px;  
}
```

```
div.nav a:hover {  
    background-position: -40px 10px;  
}
```

```
div.nav a.special {  
    background-position: -90px 10px;  
}
```


Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image 1
- Background image 2
- Background image 3
- Background image 4

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image sprite

Exercise:

fundamentos
web 2008

How many HTTP requests can you save by using CSS Sprites?

Example: Project X

Assets:

- HTML page
- Style sheet
- JavaScript file
- Data file (JSON)
- Background image sprite

Gzip all these assets. It'll save some download time.

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
  <script src="js.js"></script>
</head>
<body>
  <div id="viewer">
    <!-- Filled by Ajax -->
  </div>
</body>
</html>
```

Example: Project X

```
<html>
```

```
<head>
```

```
  <title>Image viewer</title>
```

```
  <link rel="stylesheet" href="styles.css">
```

```
  <script src="js.js"></script>
```

```
</head> - Request for CSS
```

```
<body> - Request for CSS sprite
```

```
  <div id="viewer">
```

```
    <!-- Filled by Ajax -->
```

```
  </div>
```

```
</body>
```

```
</html>
```

Example: Project X

```
<html>
```

```
<head>
```

```
  <title>Image viewer</title>
```

```
  <link rel="stylesheet" href="styles.css">
```

```
  <script src="js.js"></script>
```

```
</head> - Request for JS (block)
```

```
<body> - Request for JSON
```

```
  <div id="viewer">
```

```
    <!-- Filled by Ajax -->
```

```
  </div>
```

```
</body>
```

```
</html>
```

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
  <script src="js.js"></script>
</head> - Request for image
<body>
  <div id="viewer">
    <!-- Filled by Ajax -->
  </div>
</body>
</html>
```


Example: Project X

```
<html>
```

```
<head>
```

```
<title>Image viewer</title>
```

```
<link rel="stylesheet" href="styles.css">
```

```
<script src="js.js"></script>
```

```
</head>
```

```
<body>
```

```
<div id="viewer">
```

```
<!-- Filled by Ajax -->
```

```
</div>
```

```
</body>
```

```
</html>
```

The image, which is what the user really wants to see, is the fifth asset to be loaded.

Exercise:

fundamentos
web 2008

How many assets do you load
BEFORE the first asset that the
user thinks is important, is
downloaded?

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
  <script src="js.js"></script>
</head>
<body>
  <div id="viewer">
    <!-- Filled by Ajax -->
  </div>
</body>
</html>
```

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
  <script src="js.js"></script>
</head>
<body>
  <div id="viewer">
    
  </div>
</body>
</html>
```

Still the fifth asset.

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="viewer">
    
  </div>
  <script src="js.js"></script>
</body>
</html>
```

Now it's the third asset.

Example: Project X

```
<html>
<head>
  <title>Image viewer</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div id="viewer">
    
  </div>
  <script src="js.js"></script>
</body>
</html>
```

Exercise:

fundamentos
web 2008

Do you put your scripts at the bottom of the page?

Do you have problems with onload event handlers? They'd be solved.

Rules:

- 1) Make fewer HTTP requests
- 4) Gzip components
- 5) Put stylesheets at the top
- 6) Put scripts at the bottom
- 8) Make JavaScript and CSS external

Keep an iron grip on the assets
you have to load
and on their order of loading

Hell is other browsers - *Sartre*

fundamentos

web 2008

Ajax Workshop

The End