

# Advanced Design Techniques: JavaScript to the Rescue

Peter-Paul Koch (ppk)

<http://www.quirksmode.org>

VTM Conference, Oct. 24th, 2007

1998



```
if (browser == 'Netscape')  
    document.write  
    ('<body background="pix/bg_nn.gif">');
```

```
else if (browser == 'IE')  
    document.write  
    ('<body background="pix/bg_ie.gif">');
```

```
<noscript>  
<body background="pix/bg_nn.gif">  
</noscript>
```

```
if (browser == 'Netscape')
  document.write
  ('<body background="pix/bg_nn.gif">');
```

Maybe this isn't

```
else if (browser == 'IE')
```

```
document.write
  ('<body background="pix/bg_ie.gif">');
```

such a good idea  
after all.

```
<noscript> (Still, the principle
```

```
<body background="pix/bg_nn.gif">
```

```
</noscript>
```



Fast forward to 2007



Using JavaScript  
responsibly  
to plug a few holes  
in browsers' CSS  
support

The background of the slide is a dark gray zebra pattern with lighter gray wavy stripes. The text "Zebra Lists" is centered in a white serif font.

# Zebra Lists

```
li:nth-child(even) {  
  background-color: #00cc00;  
}
```

Opera: Yes

IE: No

Firefox: No

Safari: No



```
<ul id="navigation">  
  <li><a href="#">My books</a></li>  
  <li><a href="#">Other books</a></li>  
</ul>
```

```
<p>Titles</p>
```

```
<ul>  
  <li>PPK on JavaScript</li>  
  <li>Designing with Web Standards</li>  
  <li>Bulletproof Web Design</li>  
</ul>
```

```
<ul id="navigation">  
  <li><a href="#">My books</a></li>  
  <li><a href="#">Other books</a></li>  
</ul>
```

```
<p>Titles</p>  
<ul class="zebra">  
  <li>PPK on JavaScript</li>  
  <li>Designing with Web Standards</li>  
  <li>Bulletproof Web Design</li>  
</ul>
```

# Rule #1

Use hooks to indicate which HTML elements should be affected.

Typical hooks:

- id
- class

```
var lists = getElementsByTagName('ol,ul');
for (var i=0,list;list=lists[i];i++) {
    if (list.className != 'zebra') continue;
    var lis = list.getElementsByTagName('li');
    for (var j=0;j<lis.length;j+=2) {
        lis[j].style.backgroundColor = '#00cc00';
    }
}
```

# What if JavaScript is not supported?

Users won't see the zebra list  
but they don't know it's supposed to be  
there,  
and the effect is not vital, anyway.

## Rule #2

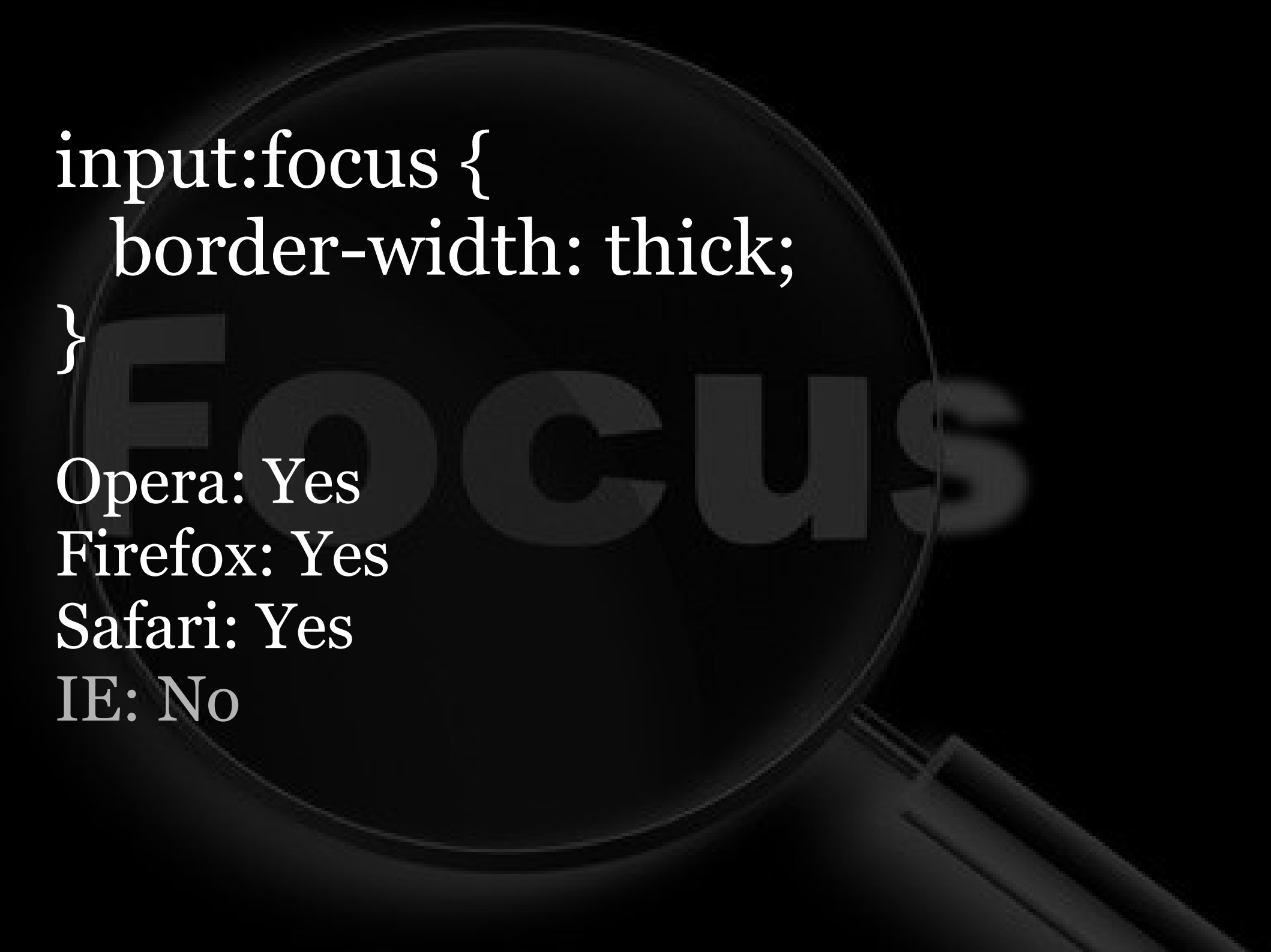
If JavaScript is not supported, usability suffers.

(Can't be helped.)

(Sorry.)

A magnifying glass with a dark handle and frame is positioned over the word 'focus'. The word is written in a large, bold, sans-serif font. The magnifying glass's lens is centered over the word, making it appear larger and more prominent. The background is a solid dark color.

**FO** :focus **US**



```
input:focus {  
  border-width: thick;  
}
```

Opera: Yes

Firefox: Yes

Safari: Yes

IE: No



```
input.onfocus = function () {  
    this.style.borderWidth = 'thick';  
}
```

```
input.onblur = function () {  
    this.style.borderWidth = '';  
}
```

A magnifying glass is centered on the page, with the word "FOCUS" written across its lens in a large, bold, sans-serif font. The magnifying glass handle extends towards the bottom right corner. The background is dark, and the text is white.

Rule #3

JavaScript needs more  
precise instructions than  
CSS.

(It's stupider but more versatile.)

```
input.onfocus = function () {  
    this.style.borderWidth = 'thick';  
}
```

```
input.onblur = function () {  
    this.style.borderWidth = '';  
}
```

```
input.onfocus = function () {  
  this.style.borderWidth = 'thick';  
  this.className += ' focused';  
}
```

```
input.onblur = function () {  
  this.style.borderWidth = '';  
  this.className =  
    this.className.replace(/focused/g, "");  
}
```

```
.focused {  
  border-width: thick;  
}
```

# Rule #4

Add class names; do not change styles.

(Ease of maintenance)

Adding class names keeps all CSS classes intact.

```
var lists = getElementsByTagName('ol,ul');
for (var i=0,list;list=lists[i];i++) {
    if (list.className != 'zebra') continue;
    var lis = list.getElementsByTagName('li');
    for (var j=0;j<lis.length;j+=2) {
        lis[j].style.backgroundColor = '#00cc00';
    }
}
```

```
var lists = getElementsByTagName('ol,ul');
for (var i=0,list;list=lists[i];i++) {
  if (list.className != 'zebra') continue;
  var lis = list.getElementsByTagName('li');
  for (var j=0;j<lis.length;j+=2) {
    lis[j].style.backgroundColor = '#00cccc';
    lis[j].className = 'even';
  }
}
```

:hover





```
<ul id="navigation">
```

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

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

```
<ul>
```

```
<li><a href="#">Web 1.0</a></li>
```

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

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

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

```
</ul>
```

```
li ul {  
  display: none;  
}
```



```
li:hover ul {  
  display: block;  
}
```

IE 7: Yes

Firefox: Yes

Safari: Yes

Opera: Yes

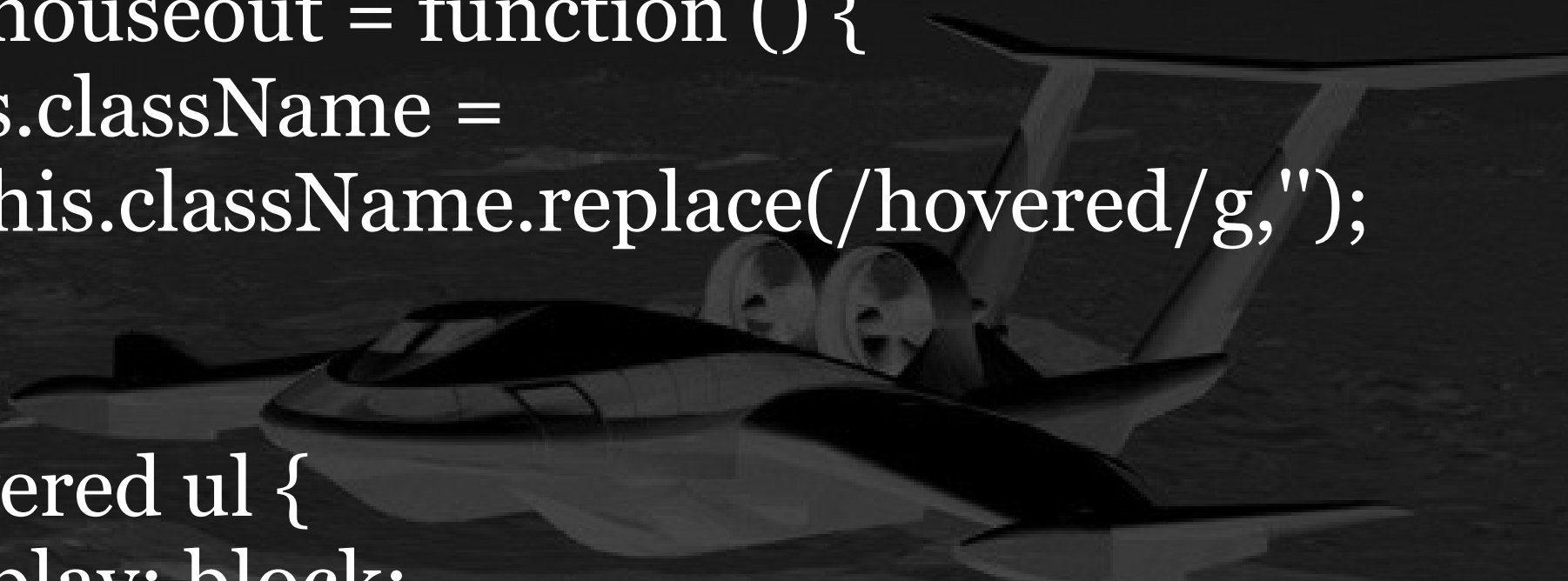
IE 6: No



```
li.onmouseover = function () {  
    this.className += ' hovered';  
}
```

```
li.onmouseout = function () {  
    this.className =  
        this.className.replace(/hovered/g, "");  
}
```

```
li.hovered ul {  
    display: block;  
}
```



```
li.onmouseover = function () {  
  this.className += ' hovered';  
}
```

```
li.onmouseout = function () {  
  this.className =  
    this.className.replace(/hovered/g, '');  
}
```

**It's MUCH more complicated than that.**

```
li.hovered ul {  
  display: block;  
}
```

`<ul>`

`<li>`

`<a>`

*Multimedialize*

`<li>`

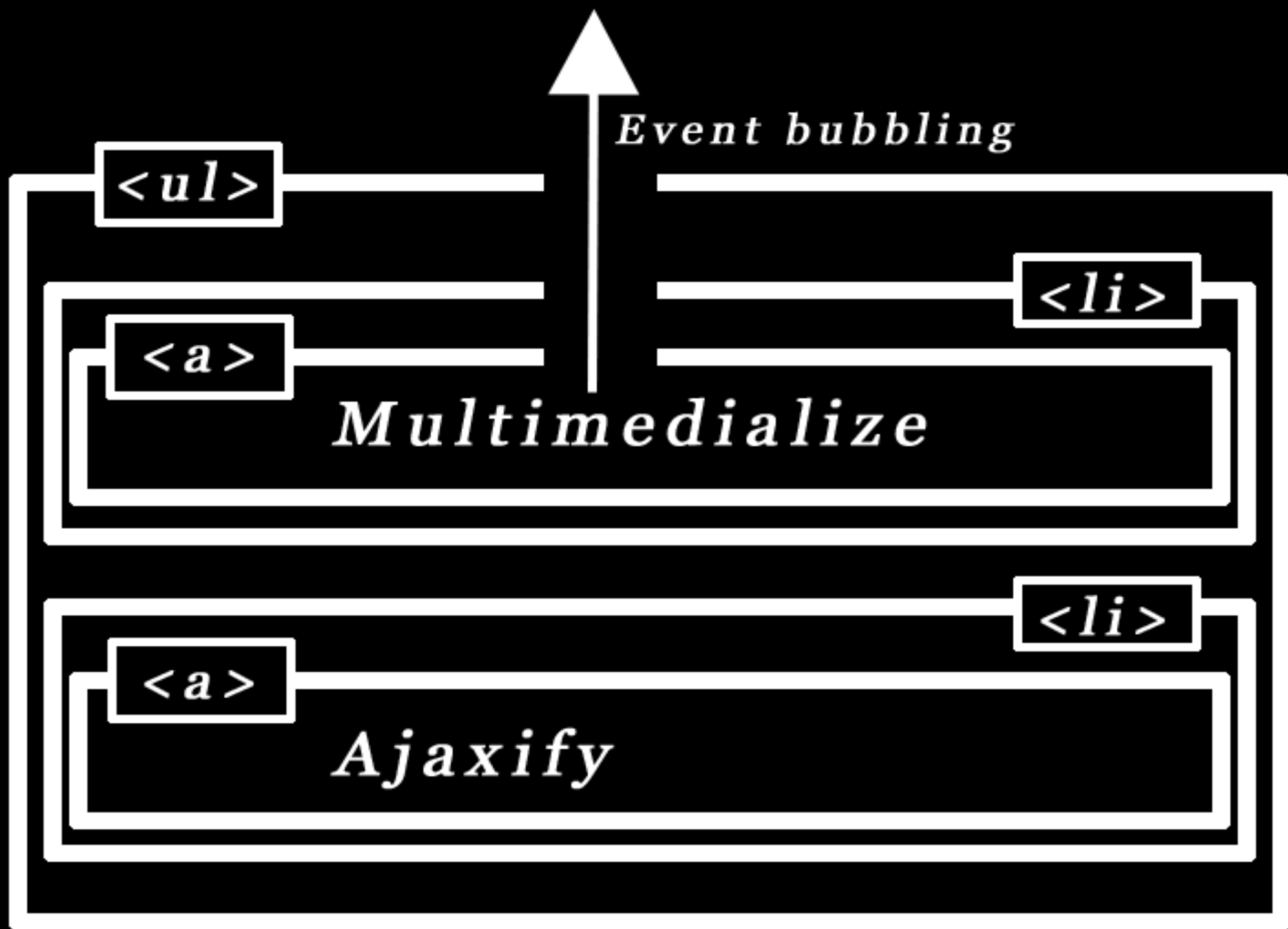
`<a>`

*Ajaxify*

A dark, grayscale image of a boat on water. The boat is a motorboat with a canopy and is positioned in the center-right of the frame. The water is dark with some ripples. The text "Event bubbling" is overlaid in a large, white, serif font. Below it, the text "Mouse events only!" is overlaid in a smaller, white, serif font.

Event bubbling

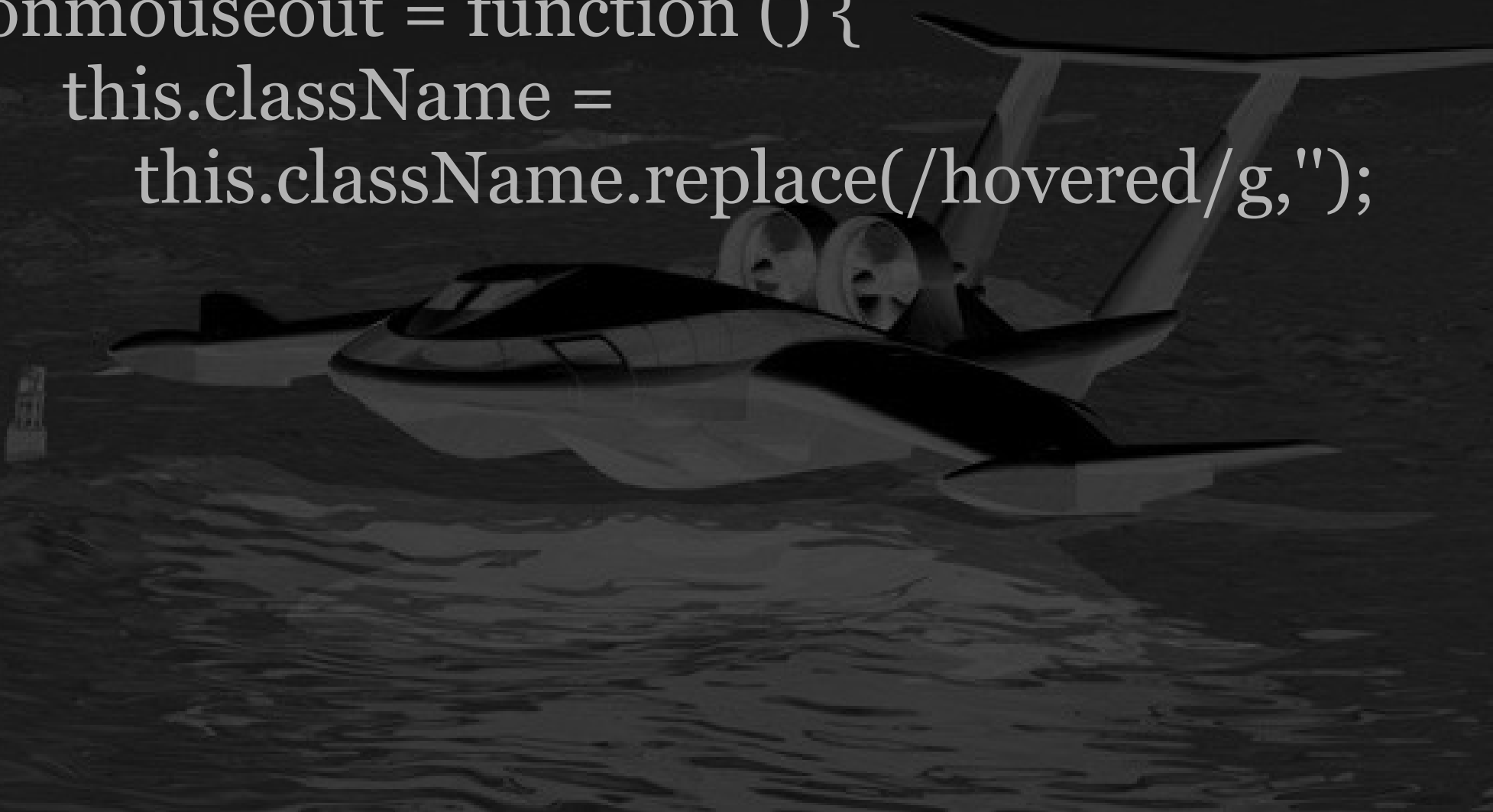
Mouse events only!





```
li.onmouseover = function () {  
    this.className += ' hovered';  
}
```

```
li.onmouseout = function () {  
    this.className =  
        this.className.replace(/hovered/g, "");  
}
```



```
ul.onmouseover = function () {  
  var li = [find correct li];  
  li.className += ' hovered';  
}
```

```
ul.onmouseout = function () {  
  var li = [find correct li];  
  li.className =  
    li.className.replace(/hovered/g, "");  
}
```

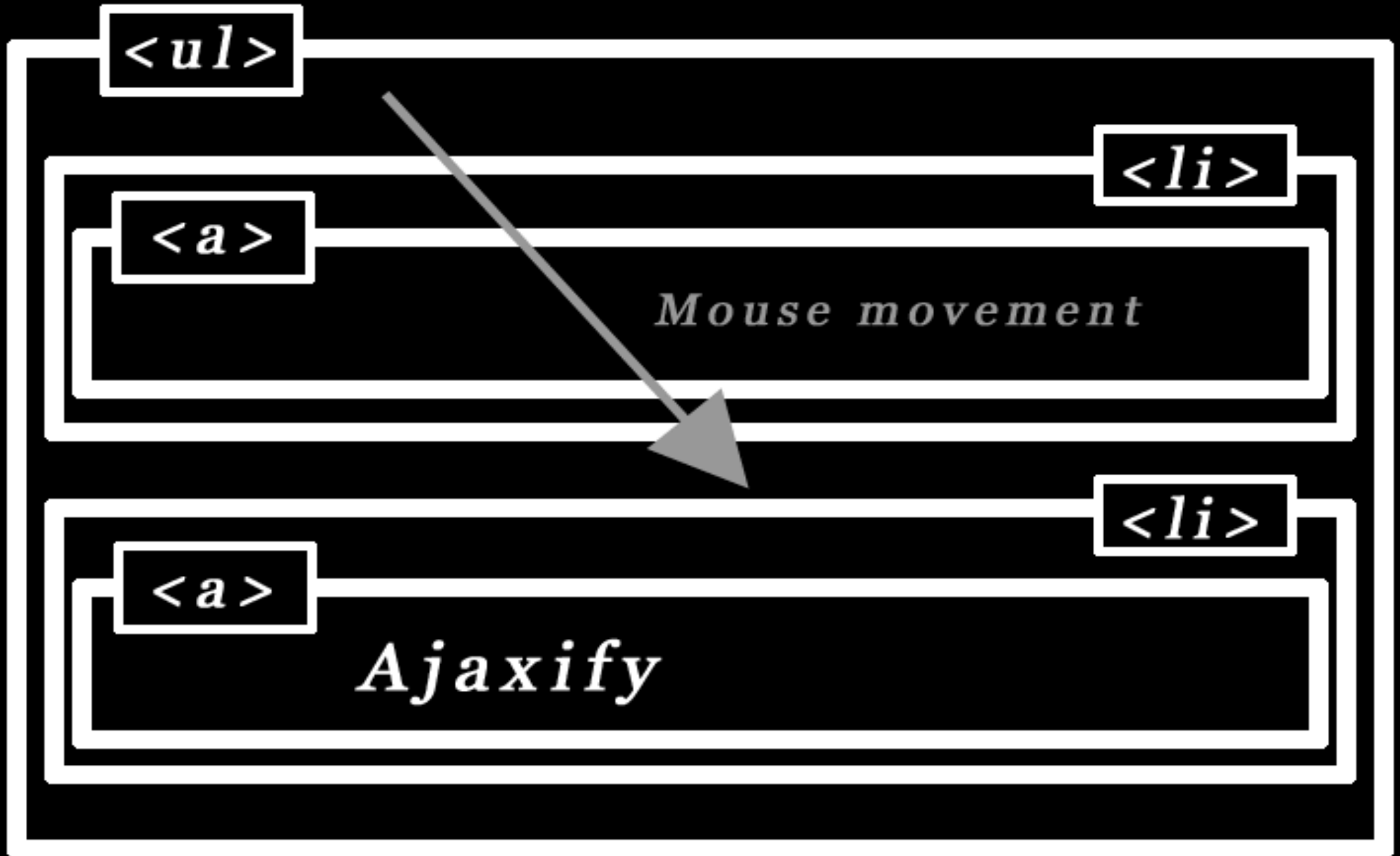
## Event delegation

A dark, monochromatic image of a boat on water. The boat is a small, open-deck vessel with a canopy structure. The water is dark with some ripples. The text is overlaid in the center in a white, serif font.

# Event bubbling

## The down side

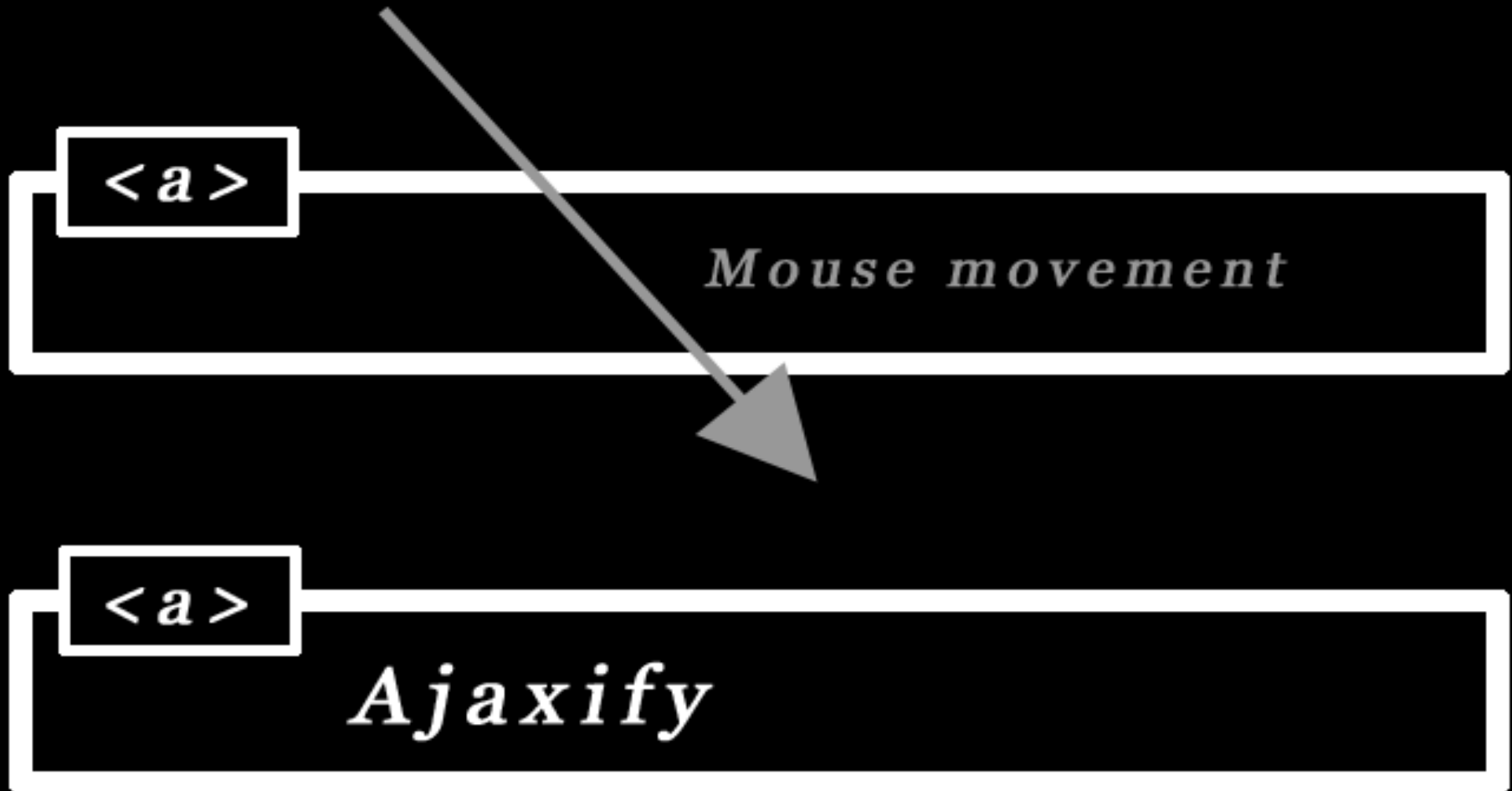
li.mouseover, li.mouseout, a.mouseover,  
a.mouseout, li.mouseover, li.mouseout



```
ul.onmouseover = function () {  
    var li = [find correct li];  
    li.className += ' hovered';  
}
```

```
ul.onmouseout = function () {  
    var li = [find correct li];  
    if (this event is relevant)  
        li.className =  
            li.className.replace(/hovered/g, "");  
}
```

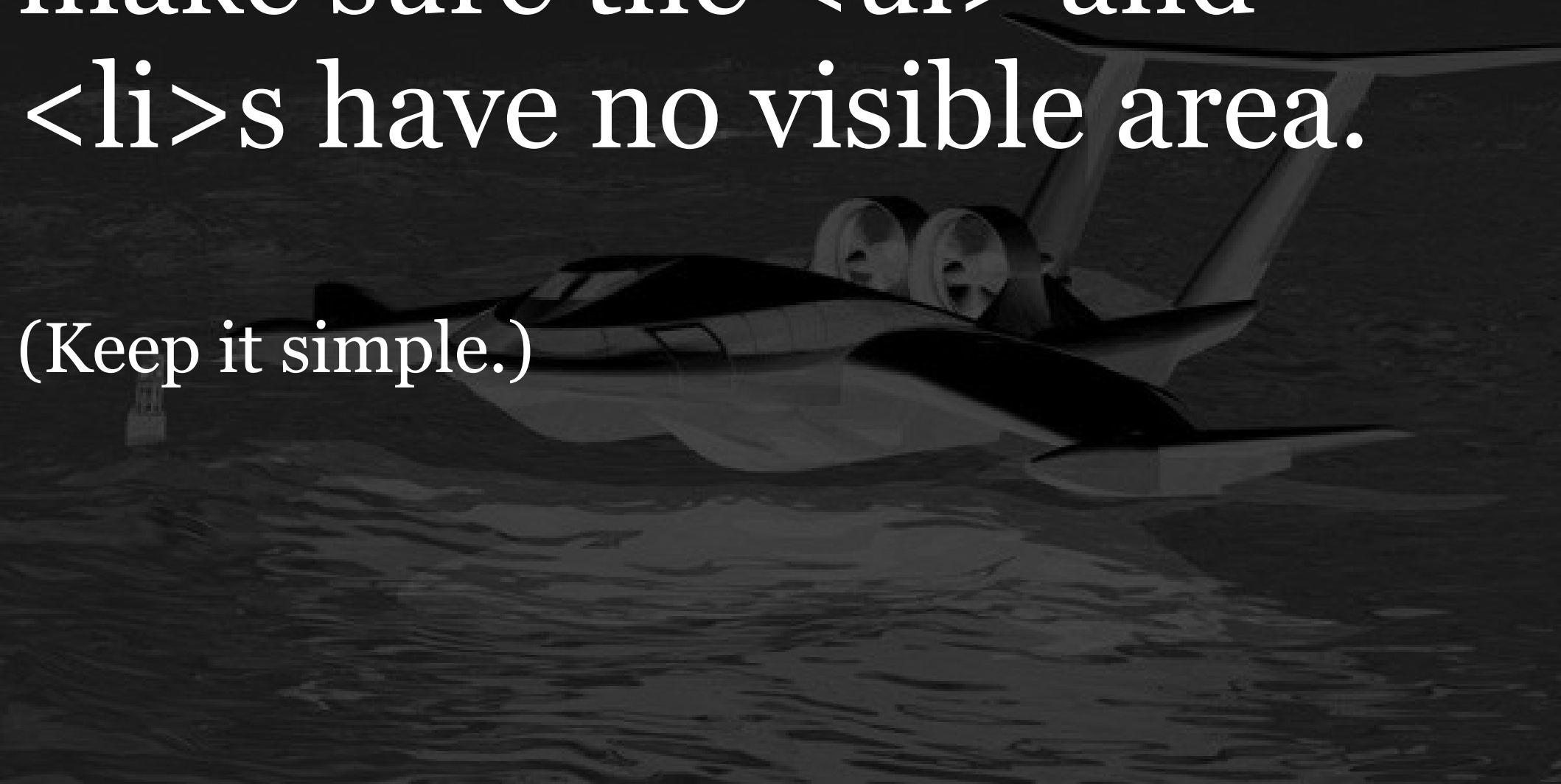
a.onmouseover, a.onmouseout

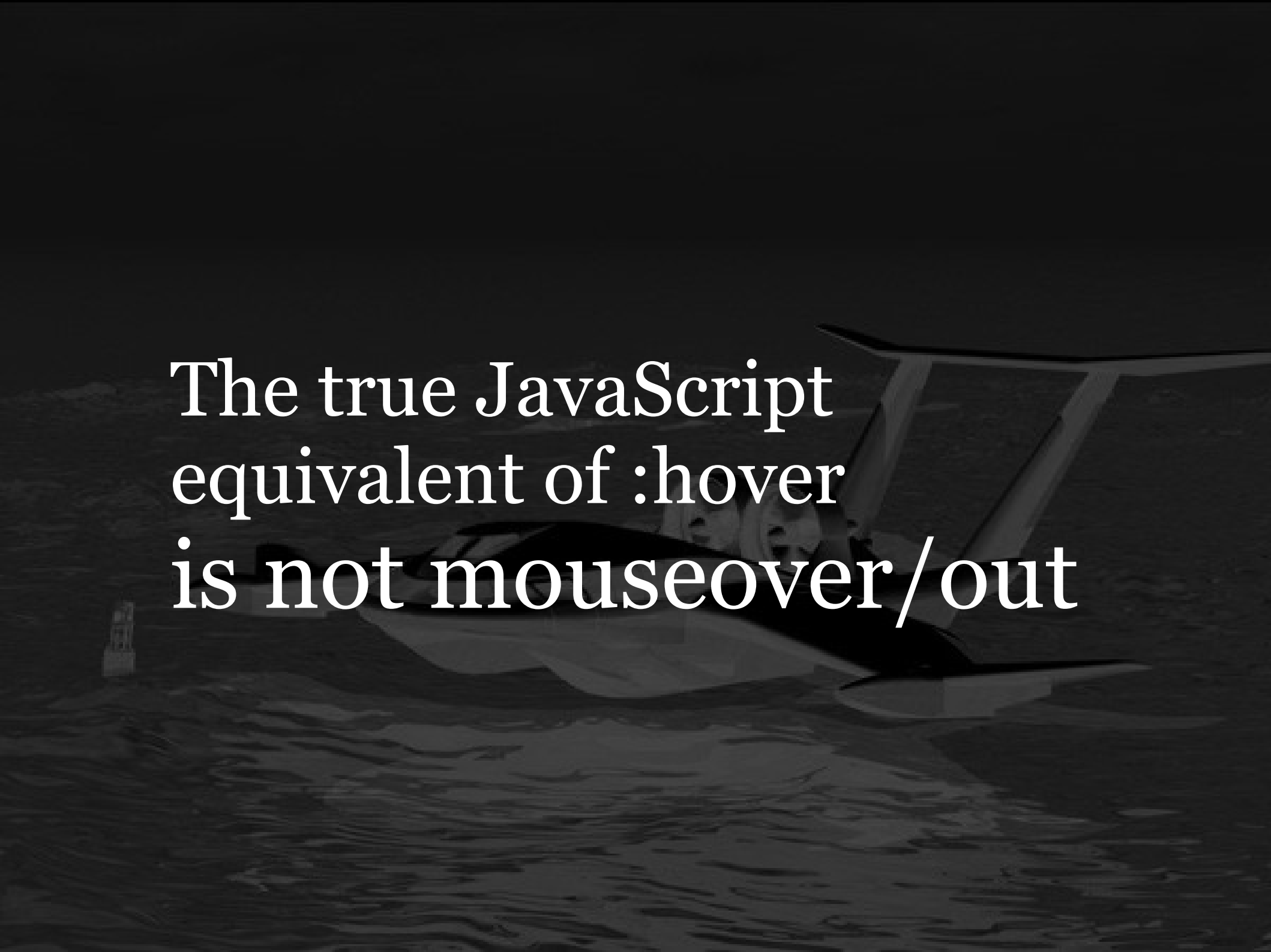


# Rule #5

When creating a dropdown, make sure the `<ul>` and `<li>`s have no visible area.

(Keep it simple.)





The true JavaScript  
equivalent of `:hover`  
is not `mouseover/out`

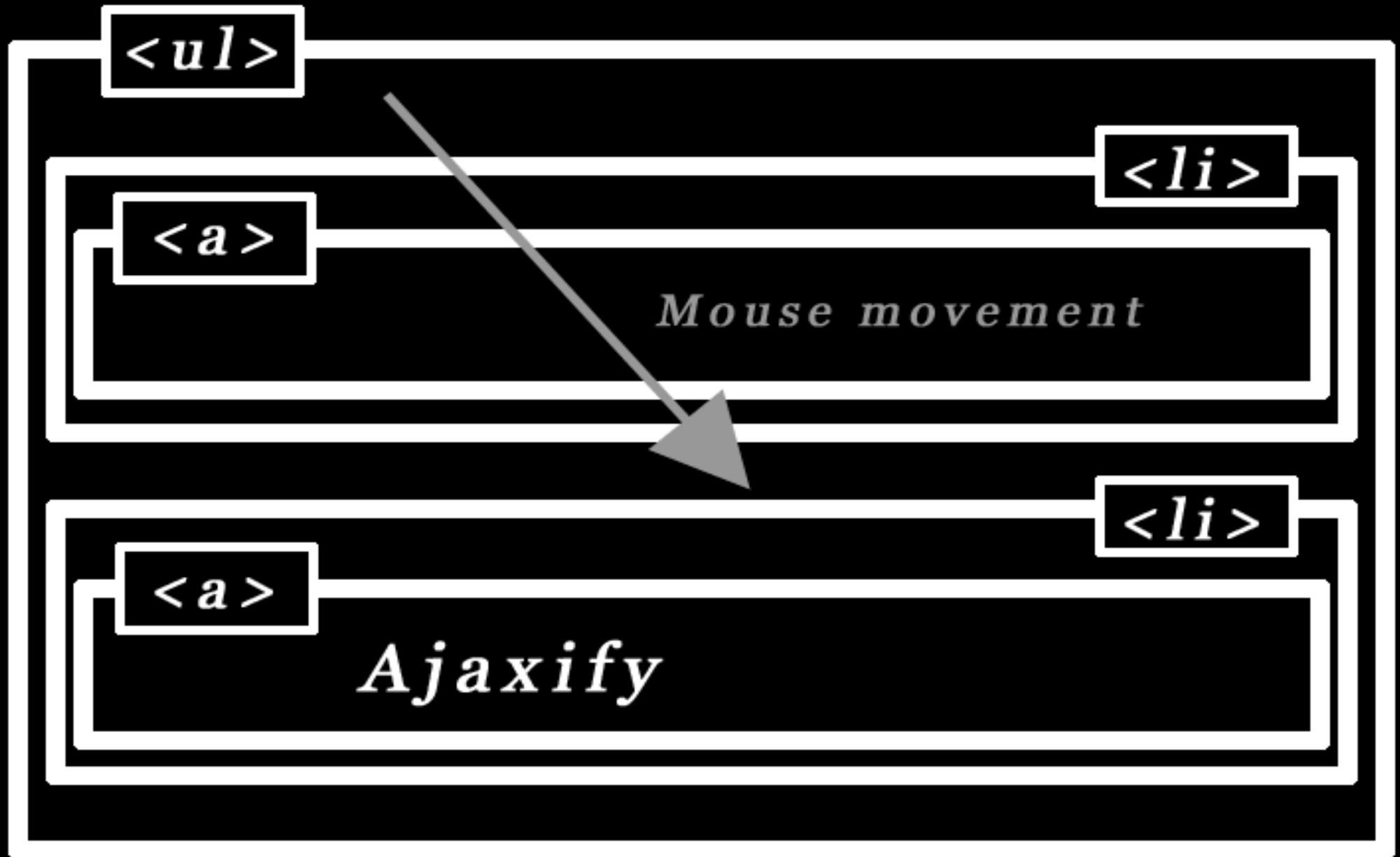




The true JavaScript  
equivalent of :hover  
is mouseenter/leave

IE only!

li.mouseenter, li.mouseleave



```
li.onmouseenter = function () {  
  var li = [find correct li];  
  this.className += ' hovered';  
}
```

```
li.onmouseleave = function () {  
  var li = [find correct li];  
  if (this event is relevant)  
    this.className =  
      this.className.replace(/hovered/g,"");  
}
```

# Rule #6

mouseenter and mouseleave  
are the true JavaScript  
equivalents of :hover.

(IE only!)

(As yet...)



```
ul.onmouseover = function () {  
    var li = [find correct li];  
    li.className += ' hovered';  
}
```


```
ul.onmouseout = function () {  
    var li = [find correct li];  
    if (this event is relevant)  
        li.className =  
            li.className.replace(/hovered/g, "");  
}
```

```
ul.onmouseover = function () {  
  var li = [find correct li];  
  li.className += 'hovered';  
}
```

**That's it, right?**

```
ul.onmouseout = function () {  
  var li = [find correct li];  
  if (this event is relevant) {  
    li.className =  
      li.className.replace(/hovered/g, "");  
  }  
}
```

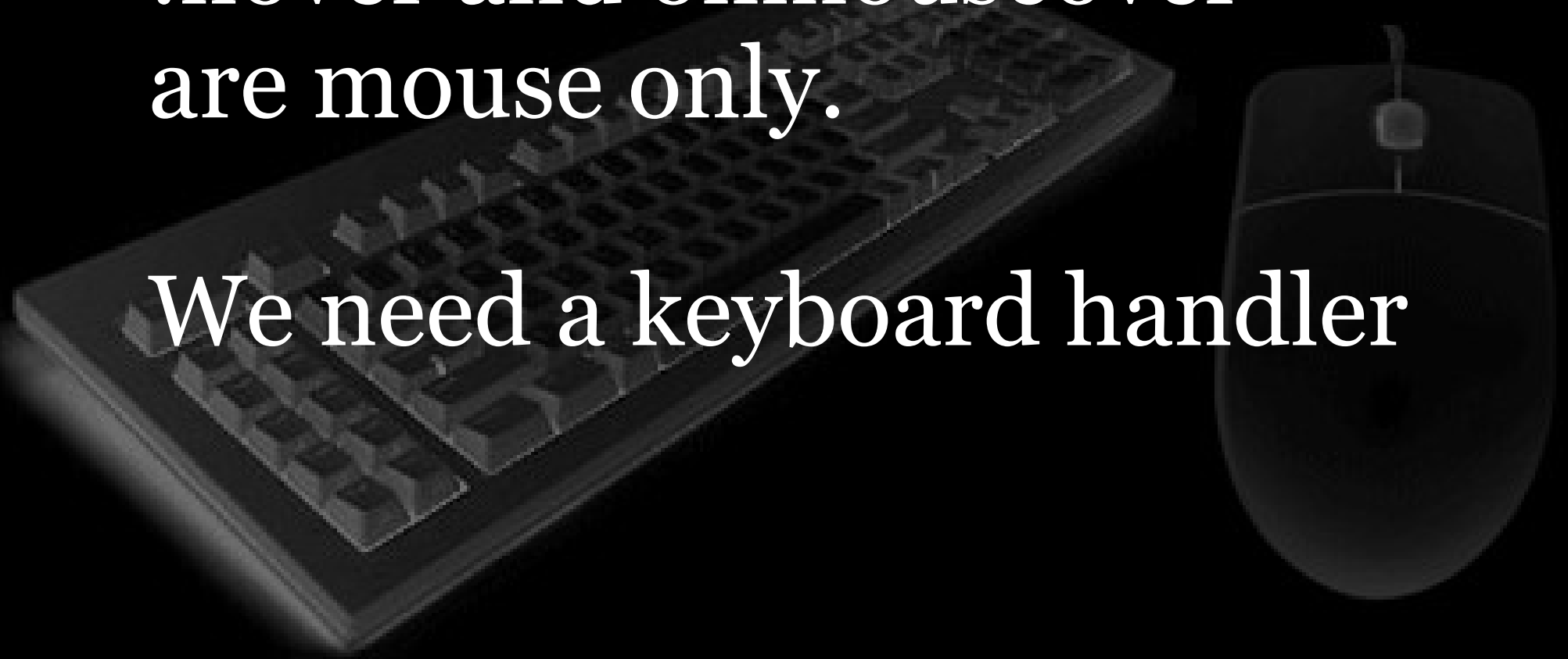
**Wrong!**

A dark, semi-transparent keyboard and mouse are positioned on a black background. The keyboard is on the left, and the mouse is on the right. Overlaid on the keyboard is the text "Device Independence!" in a white, serif font. The word "Device" is on the top line, and "Independence!" is on the bottom line, both centered horizontally.

Device  
Independence!

:hover and onmouseover  
are mouse only.

We need a keyboard handler

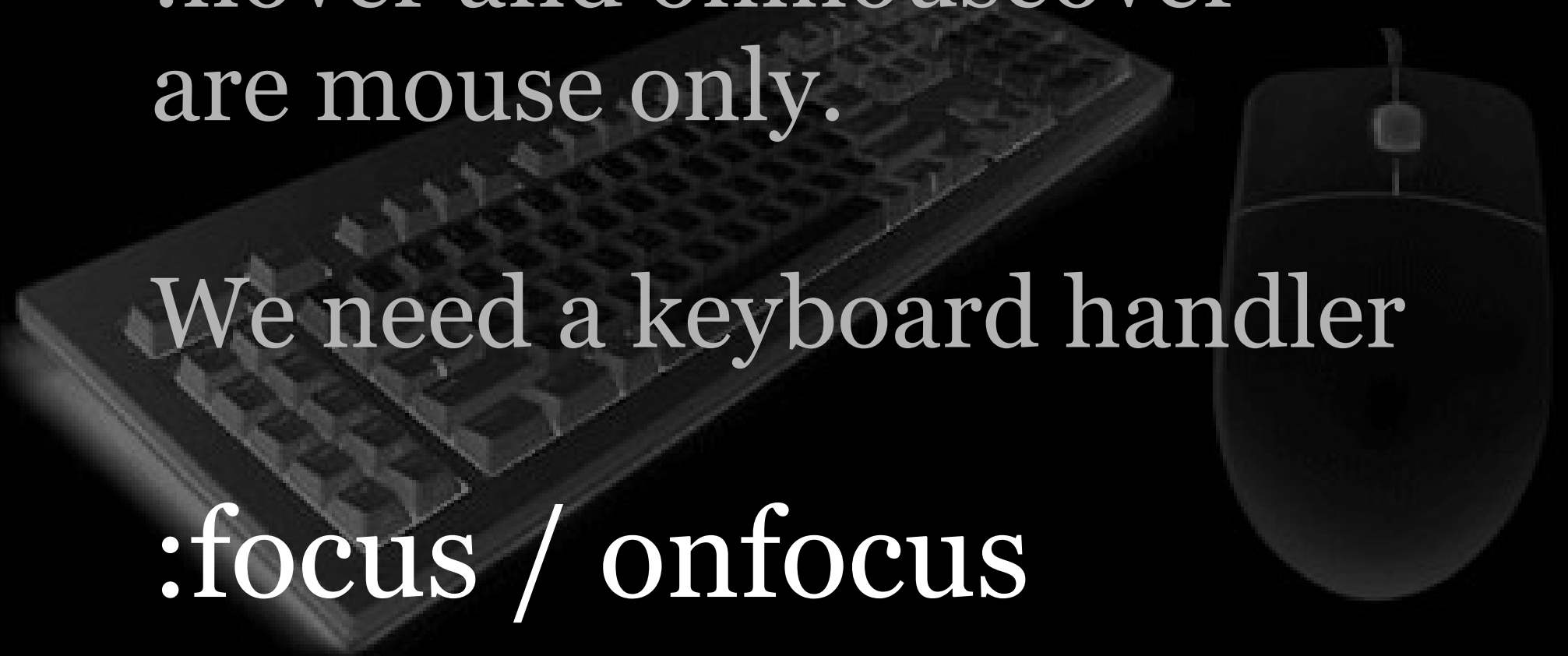




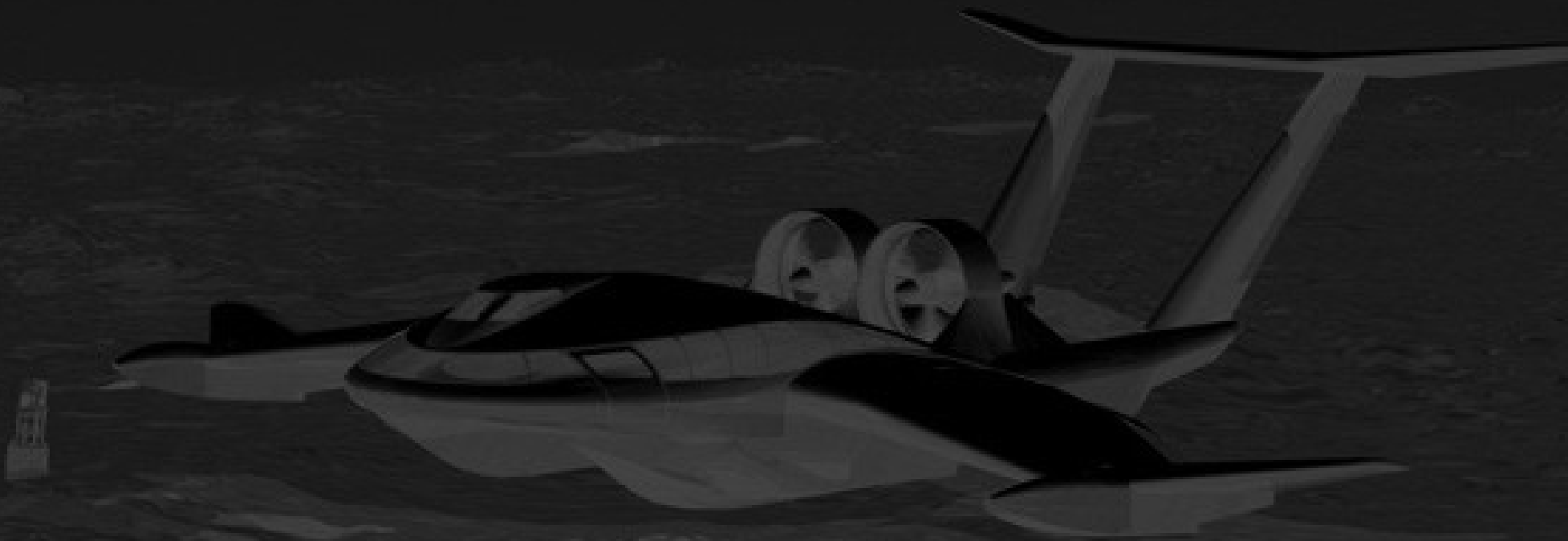
:hover and onmouseover  
are mouse only.

We need a keyboard handler

:focus / onfocus



```
li:hover ul, li:focus ul {  
  display: block;  
}
```



```
li:hover ul, li:focus ul {  
  display: block;  
}
```

Doesn't work.

You can't put the keyboard focus on an `<li>`.



# Rule #7

Only links, form fields, and the location bar can receive the keyboard focus in all browsers.



```
li:hover ul, (li < a:focus) ul {  
  display: block;  
}
```

(Don't try this at home.)



```
li:hover ul, (li ← a:focus) ul {  
  display: block;  
}
```

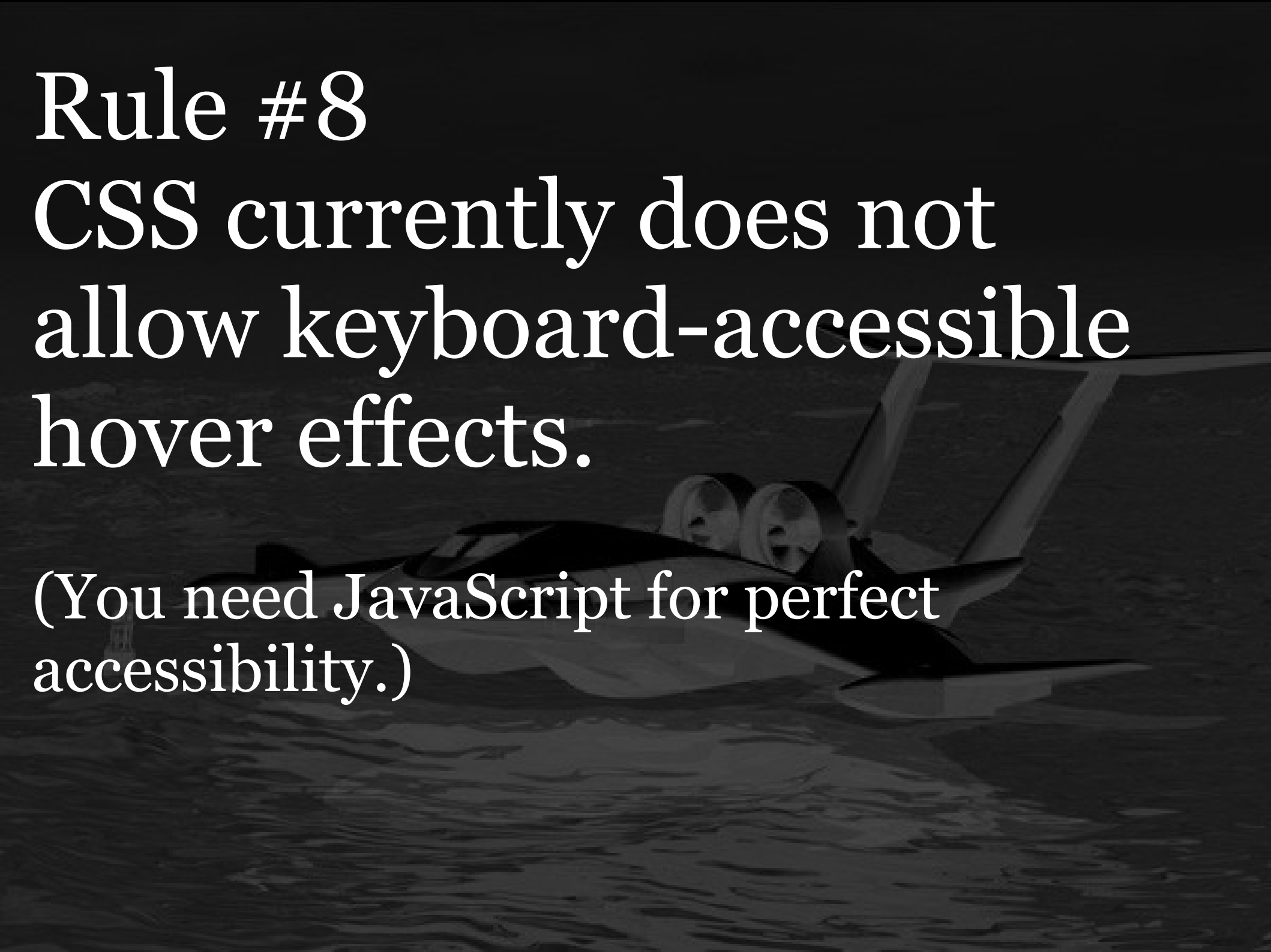
(Don't try this at home.)  
Not supported.  
(Pity, isn't it?)



# Rule #8

CSS currently does not allow keyboard-accessible hover effects.

(You need JavaScript for perfect accessibility.)



```
li.onmouseover = li.onfocus = function () {  
    this.className += ' hovered';  
}
```

```
li.onmouseout = li.onblur = function () {  
    this.className =  
        this.className.replace(/hovered/g, "");  
}
```





```
li.onmouseover = li.onfocus = function () {  
  this.className += ' hovered';  
}
```

```
li.onmouseout = li.onblur = function () {  
  this.className =  
    this.className.replace(/hovered/g, "");  
}
```

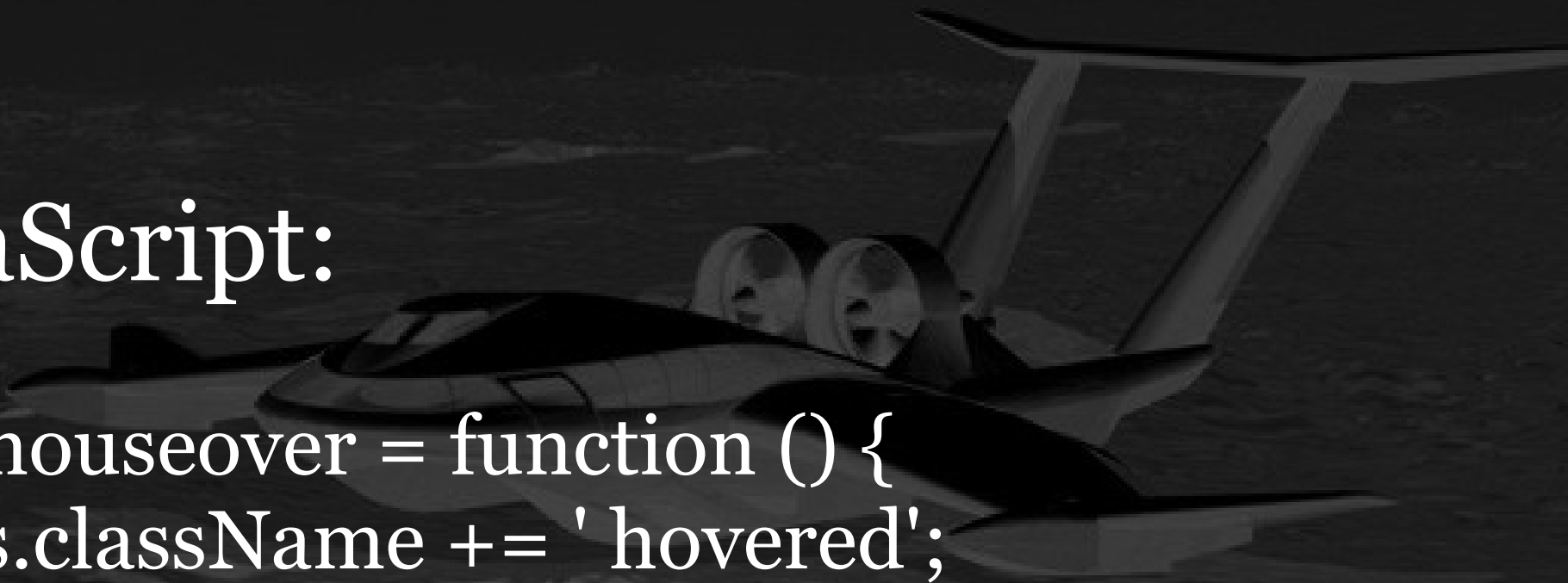
Same problem: you can't focus on an <li>.

# CSS:

```
li:hover ul {  
  display: block;  
}
```

# JavaScript:

```
li.onmouseover = function () {  
  this.className += ' hovered';  
}
```



# CSS:

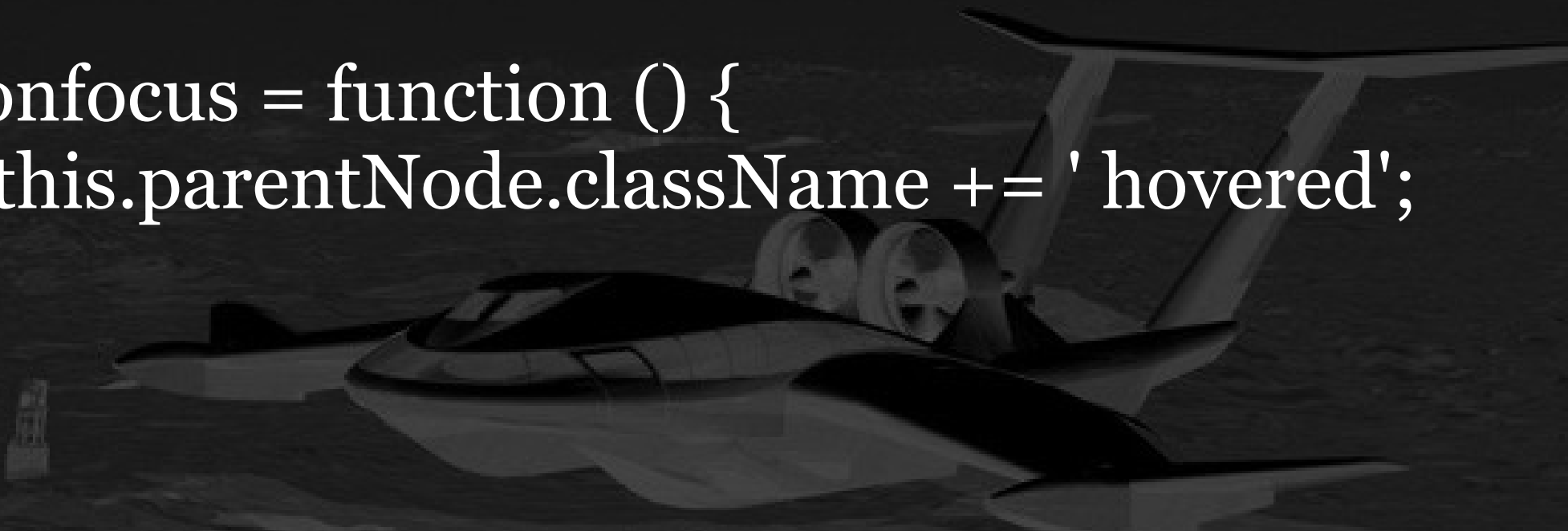
```
(li < a:focus) ul {  
  display: block;  
}
```

# JavaScript:

```
a.onfocus = function () {  
  this.parentNode.className += ' hovered';  
}
```

```
ul.onmouseover = function () {  
  var li = [find correct li];  
  li.className += ' hovered';  
}
```

```
a.onfocus = function () {  
  this.parentNode.className += ' hovered';  
}
```



```
ul.onmouseout = function () {  
    var li = [find correct li];  
    if (this event is relevant)  
        li.className =  
            li.className.replace(/hovered/g,"");  
}
```

```
a.onblur = function () {  
    if (this event is relevant)  
        this.parentNode.className =  
            this.parentNode.className.  
                replace(/hovered/g,"");  
}
```

:before and :after



```
a.external:after {  
    content: “(external link)”;  
}
```

Firefox: Yes

Safari: Yes

Opera: Yes

IE: No

```
var afterNode = document
    .createTextNode(' (external link)');
var links = document
    .getElementsByTagName('a');
for (var i=0;i<links.length;i++) {
    if (links[i].className != 'external') continue;
    links[i].appendChild(afterNode
        .cloneNode(true));
}
```



What happens if both the CSS and the JavaScript are executed?

“

Their [findings \(external link\)](#) [\(external link\)](#)  
suggest ...

“

(Oops)

Situation:

- JavaScript for IE
- CSS for all other browsers

# Conditional comments

```
<!--[if IE]>  
  <script type="text/javascript"  
  src="after.js"></script>  
<![endif]-->
```

# Rule #9

If a script is for IE only, use an IE-only way of including it.

(Conditional comments)

Option #2:

Don't use CSS at all.

Is the “(external link)” text absolutely required?

If so, it should go in the HTML (accessibility).

If not, is it presentation or behavior?

(Personal opinion warning)

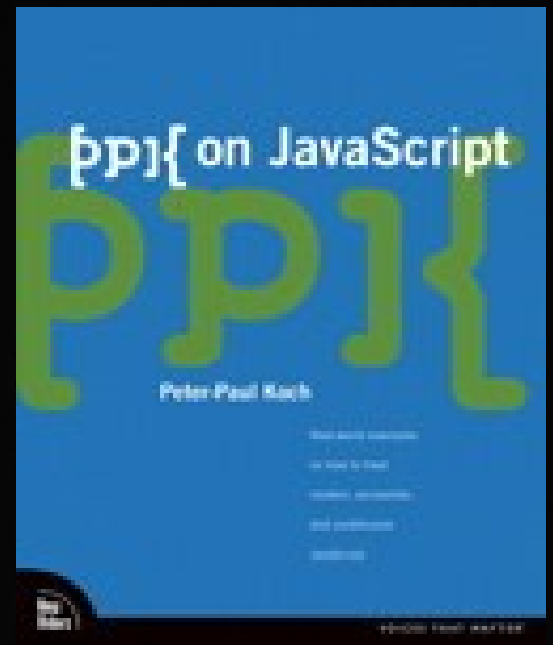
The CSS content property should not be used.

Using it is akin to using text in images.

Text belongs either in the HTML structural layer, or in the JavaScript behavior layer; but never in the CSS presentation layer.



Using JavaScript  
responsibly  
to plug a few holes  
in browsers' CSS  
support



Need any help?





Have fun!