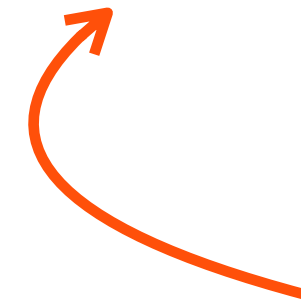


# JAVASCRIPT



*and the Web!*

# JAVASCRIPT

popular scripting language on the Web,  
supported by browsers

separate scripting from structure (HTML) and  
presentation (CSS)

client- and server-side programming

object-oriented, imperative, functional

# HOW TO EMBED JS IN HTML

## Embed external file

```
<script type="text/javascript" src="code.js"></script>
```

## Inline in HTML

```
<script type="text/javascript">
```

```
<![CDATA[
```

Javascript goes here...

```
]]>
```

```
</script>
```

*everything inside ignored by parser*



*Revisiting the Dom*

# DOM DOCUMENT OBJECT

root node of HTML document

selector properties/methods:

`document.body`

`document.getElementById()`

`document.getElementsByClassName()`

`document.getElementsByTagName()`

# DOM ELEMENT OBJECT

Element metadata:

`element.tagName`

`element.className`

`element.id`

`element.attributes`

`element.innerHTML`

Node metadata:

`element.nodeName`

`element.nodeType`

`element.nodeValue`

# DOM ELEMENT OBJECT

properties for traversing the DOM tree:

`element.childNodes/element.children`

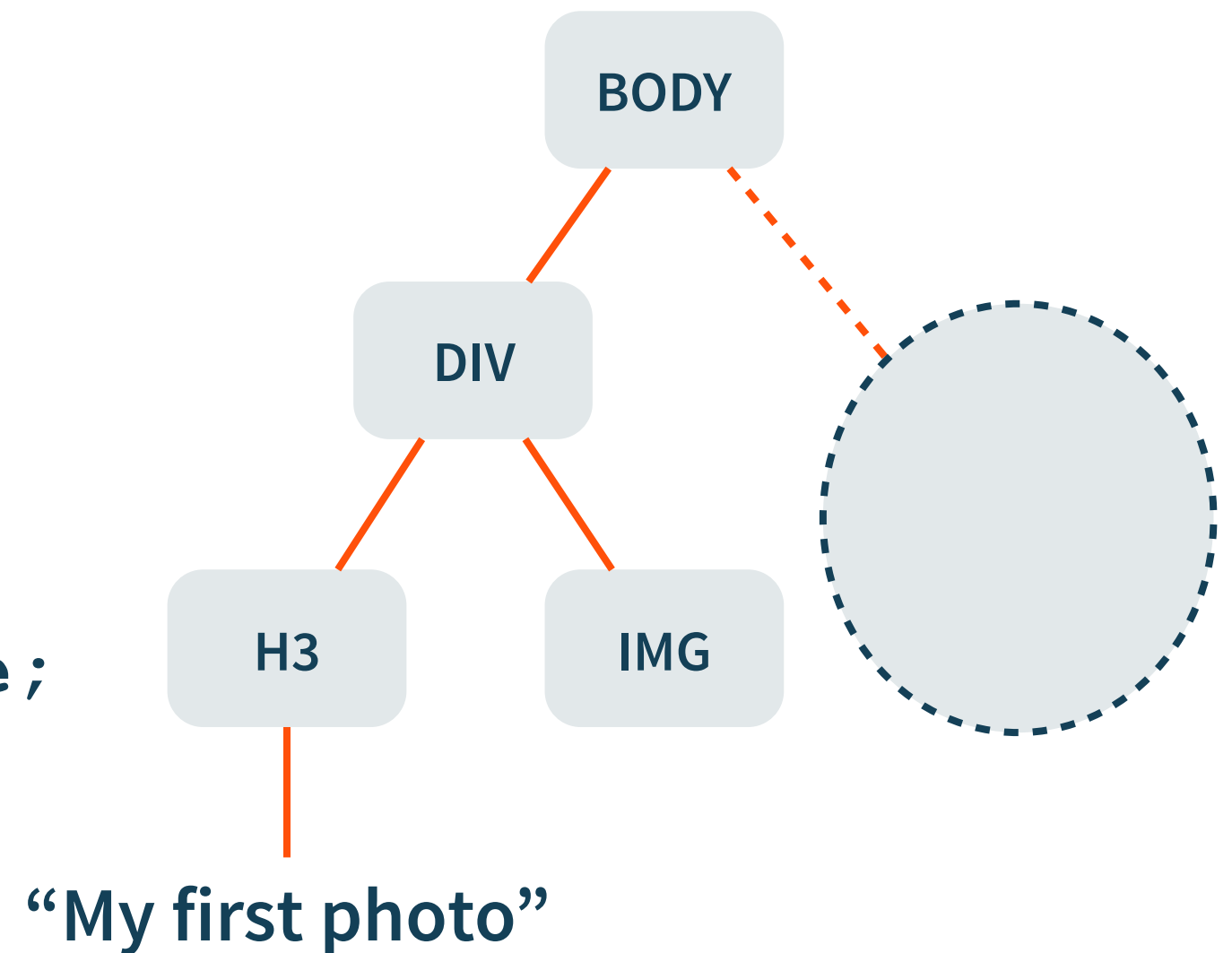
`element.parentNode/element.parentElement`

`element.previousSibling/element.previousElementSibling`

`element.nextSibling/element.nextElementSibling`

# TRAVERSING THE DOM

```
var body = document.body;  
var div = body.children[0];  
var h3 = div.children[0];  
var textNode = h3.childNodes[0];  
var textString = textNode.nodeValue;
```





# DOM ELEMENT OBJECT

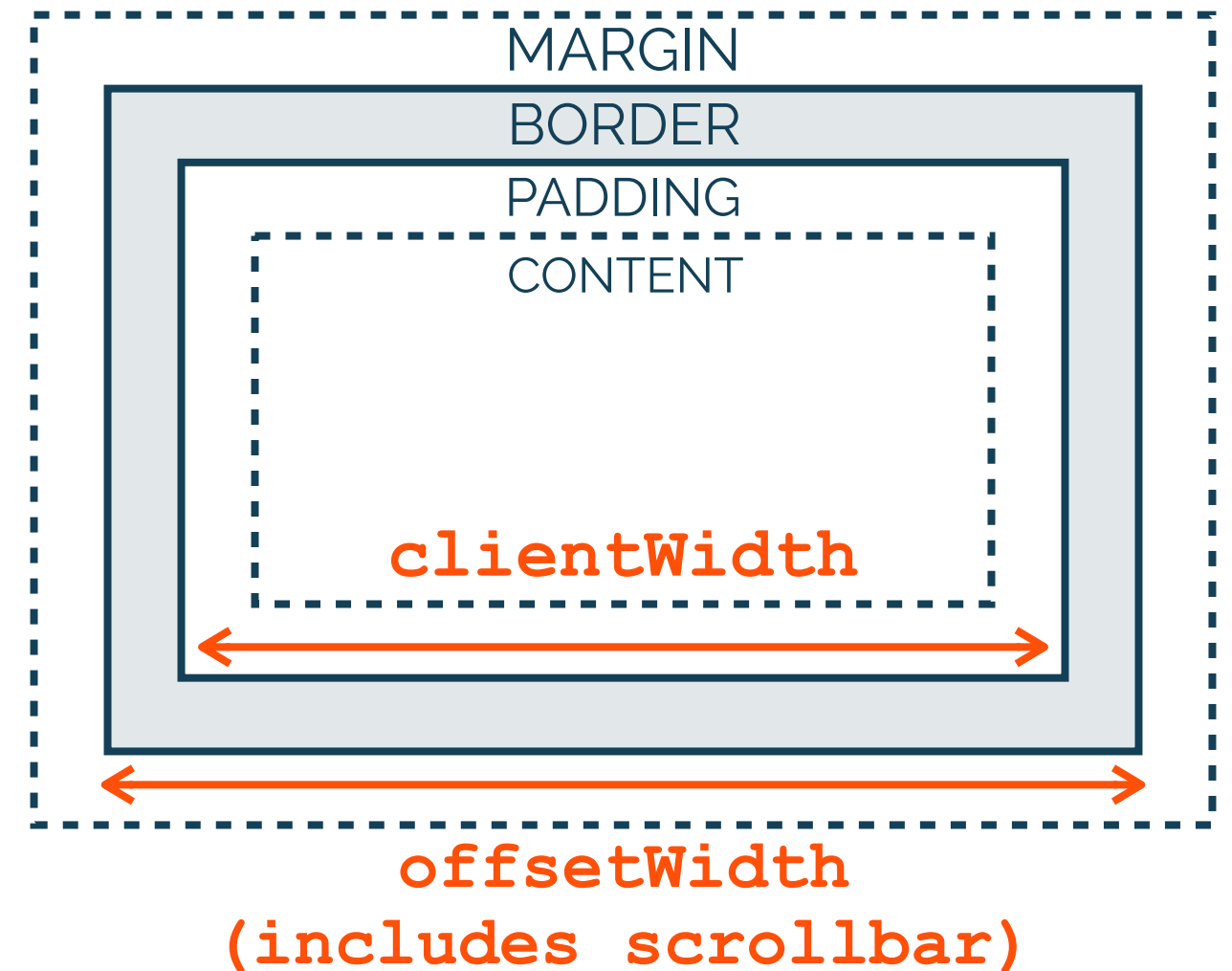
relative to  
offsetParent



position: `element.offsetTop`,  
`element.scrollTop`, ...

dimensions: `element.clientWidth`,  
`element.offsetWidth`, ...

style: `element.style`



# DOM MANIPULATION

programmatically change the structure and modify element properties

```
element.style.backgroundColor = "red";
```

```
element.innerHTML = "<div><h3>Llama!</h3>..</div>"
```

augment DOM structure:

```
element.appendChild(), element.removeChild(), ...
```

*Events*

# TYPES OF EVENTS

User: *mouse clicks, mouse moves, key presses*

Browser: *page load/unload*

Network: *responses to AJAX request*

Timer

# TIMER EVENTS

**`setTimeout(fn, ms);`**

calls function after specified amount of time (ms)

**`setInterval(fn, ms);`**

calls function at specified intervals (ms) until

**`clearInterval()`** or window is closed

# EVENT HANDLERS

 *also known as listeners*

callback functions

specify: what happened, where it happened, and how to handle it

# EVENT HANDLERS

DOM LEVEL 0  


```
<div onclick="alert( 'Llama!' );">...</div>
```

In HTML

DOM LEVEL 1  


```
element.onclick = function() {alert( 'Llama!' );}
```

In Javascript using the DOM

# EVENT HANDLERS

DOM LEVEL 2

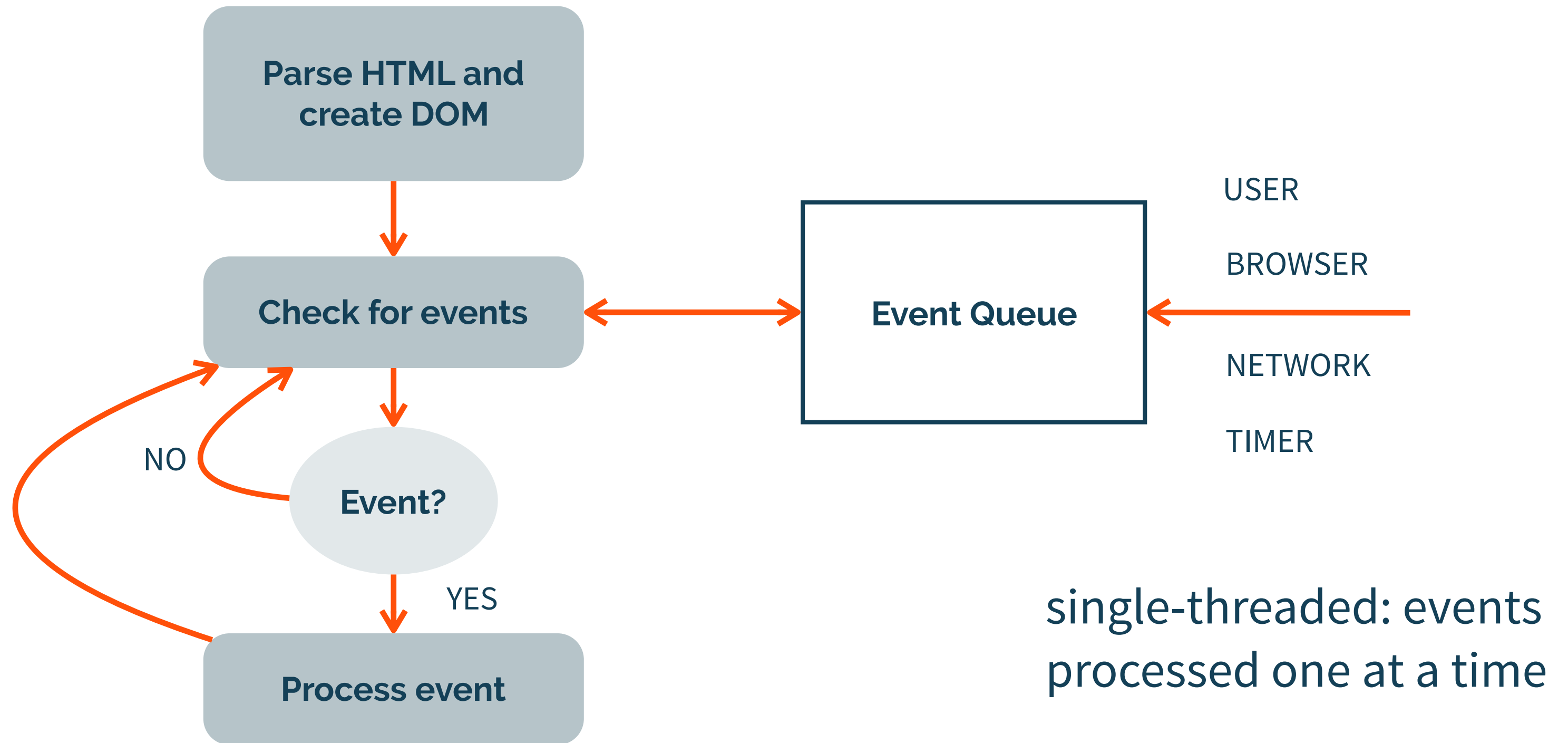


```
var el = document.getElementById( 'myButton' );  
el.addEventListener( 'click', function() {  
    alert( 'Llama!' ); } );
```

supports multiple handlers per event



# THE BROWSER EVENT LOOP



# EVENT OBJECT

contains the information about the event

HTML `<div onclick="mouseClick(event) ;">`

DOM `element.onclick = mouseClick;`

`function mouseClick(event) {...} ;`

DOM (IE) `function mouseClick() {...`  
`x = window.event.clientX;`  
`...} ;`

# EVENT PROCESSING

events propagate in two phases

*capture phase*: root to innermost element

*bubble phase*: innermost element to root

DOM standard: *capture* then *bubble*

# EVENT PROCESSING

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

 *set capture or bubble phase*

```
event.stopPropagation()
```

CODEPEN

# Event Example 1

CODEPEN

# Anonymous Functions

```
function animateIt(elementId, speed) {  
    var elem = document.getElementById(elementId);  
    tick = 0;  
    var timer = setInterval(function() {  
        if (tick < 100) {  
            elem.style.left = tick*speed + "px";  
            tick++;  
        }  
        else {clearInterval(timer);}  
    }, 30);  
}
```

# Closures

```
function animateIt(elementId, speed) {  
    var elem = document.getElementById(elementId);  
    tick = 0;  
    var timer = setInterval(function() {  
        if (tick < 100) {  
            elem.style.left = tick*speed + "px";  
            tick++;  
        }  
        else {clearInterval(timer);}  
    }, 30);  
}
```

# Event Example 2

CODEPEN



# Classes and Mouse Events

```
function Dragger(id) {  
    this.isMouseDown = false;  
    this.element = document.getElementById(id);  
    var obj = this;  
    this.element.onmousedown = function(event) {  
        obj.mouseDown(event);  
    }  
}
```

why obj instead of this?



# Classes and Mouse Events

```
Dragger.prototype.mouseDown = function(event) {  
    var obj = this;  
    this.oldMoveHandler = document.body.onmousemove;  
    document.body.onmousemove = function(event) {  
        obj.mouseMove(event);  
    };  
    this.oldUpHandler = document.body.onmouseup;  
    document.body.onmouseup = function(event) {  
        obj.mouseUp(event);  
    };  
    this.oldX = event.clientX;  
    this.oldY = event.clientY;  
    this.isMouseDown = true;  
}
```

why body?

# *Troubles with Browsers and Other Quirks*

# BROWSERS

stable APIs, but different implementations

JavaScript libraries duplicate existing  
event handling and DOM APIs

# JQUERY

cross-browser

use for all DOM manipulation:

*(e.g., positioning relative to document and not offsetParent)*

CODEPEN

## == (negated: !=)

When using two equals signs for JavaScript equality testing, some funky conversions take place.

	true	false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	" "	null	undefined	Infinity	-Infinity	[]	{}	[[[]]]	[0]	[1]	NaN
true																					
false																					
1																					
0																					
-1																					
"true"																					
"false"																					
"1"																					
"0"																					
"-1"																					
" "																					
null																					
undefined																					
Infinity																					
-Infinity																					
[]																					
{}																					
[[[]]]																					
[0]																					
[1]																					
NaN																					

Moral of the story:

Always use 3 equals unless you have a good reason to use 2.

[dorey.github.io/JavaScript-Equality-Table/](https://dorey.github.io/JavaScript-Equality-Table/)

# TIPS & TRICKS

[developers.google.com/speed/articles/  
optimizing-javascript](https://developers.google.com/speed/articles/optimizing-javascript)

[jonraasch.com/blog/10-javascript-  
performance-boosting-tips-from-nicholas-  
zakas](https://jonraasch.com/blog/10-javascript-performance-boosting-tips-from-nicholas-zakas)

# NEXT CLASS: RESPONSIVE DESIGN

[courses.engr.illinois.edu/cs498rk1/](https://courses.engr.illinois.edu/cs498rk1/)