

# Approaches to DOM Traversal

---

THINKING BEYOND JQUERY

Brian Rinaldi  
[@remotesynth](https://twitter.com/remotesynth)  
<http://flippinawesome.org>

# Who Am I?

---

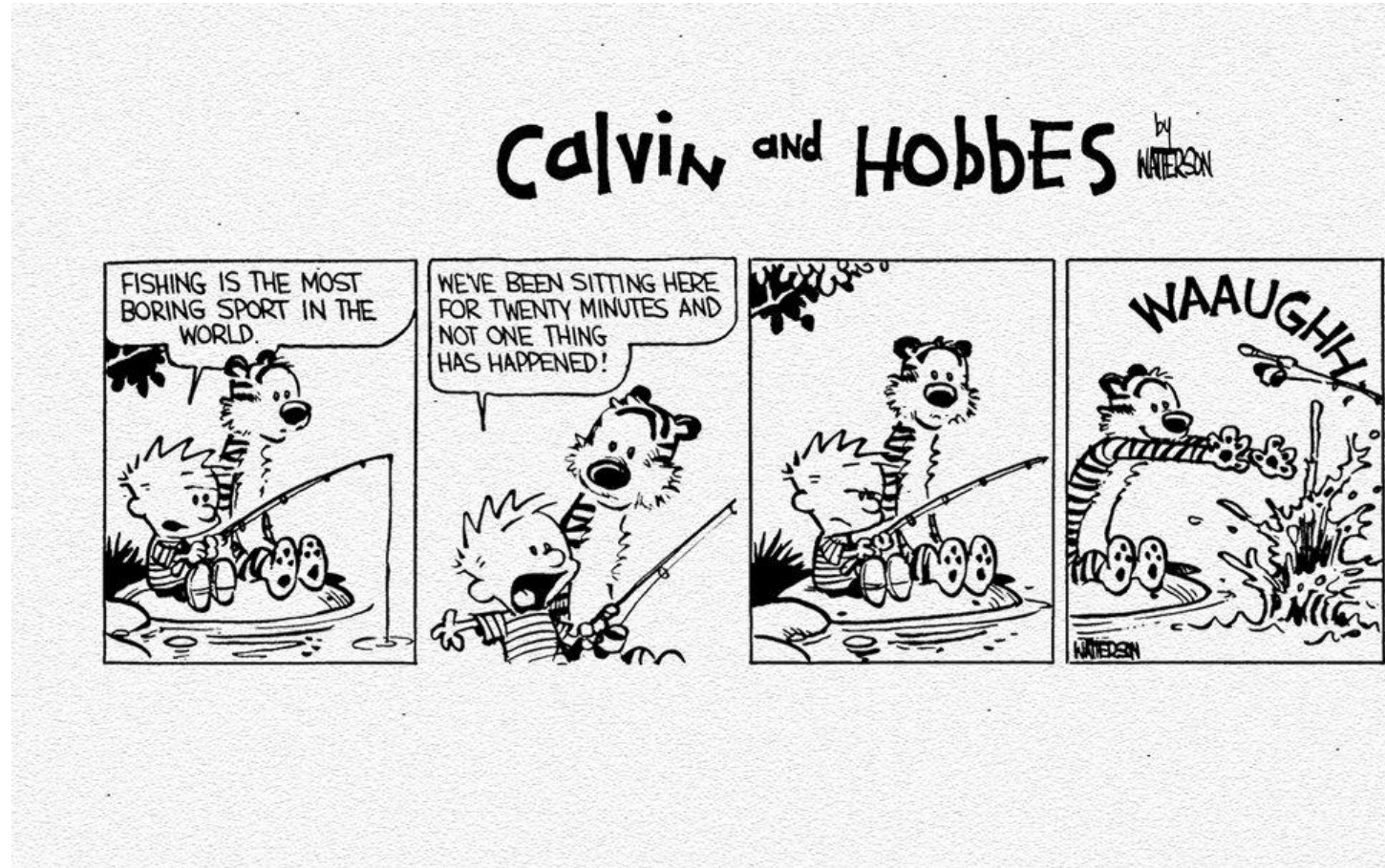
Developer Content Manager at Telerik

Founder of Flippin' Awesome (<http://flippinawesome.org>)

@remotesynth on Twitter

# Breaking the Routine

---



# How We Usually do DOM Traversal

---



# How We Usually do DOM Traversal

---

Start with selectors:

```
$("#item")
```

```
$(".fancybutton")
```

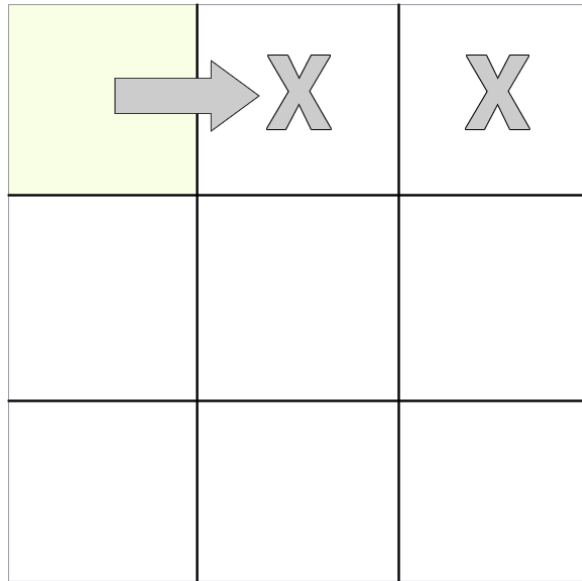
```
$("li:even")
```

```
$("ul:nth-child(3) ")
```

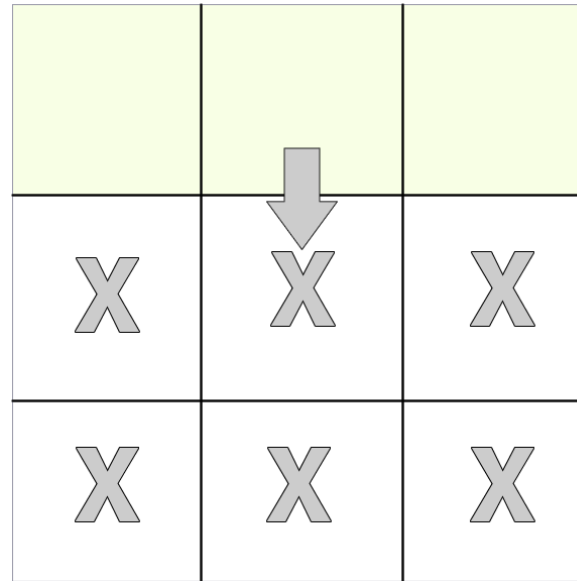
# How We Usually do DOM Traversal

---

`$("#cell111").siblings()`



`$("#row1").siblings()`



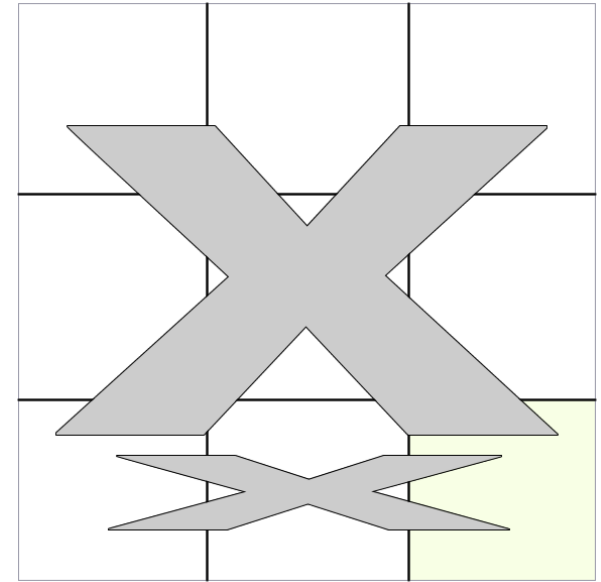
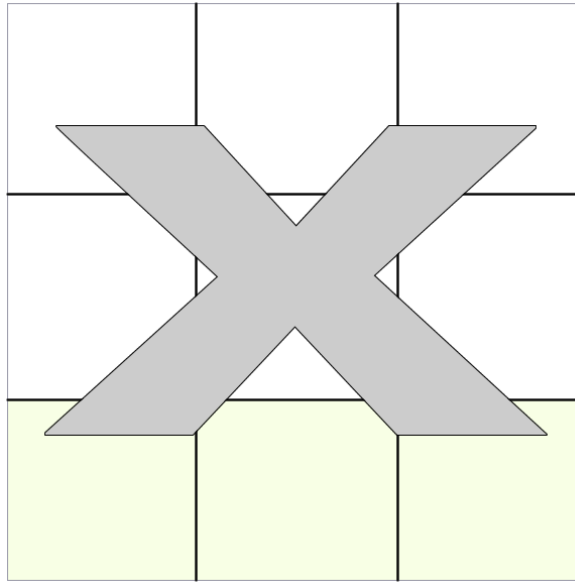
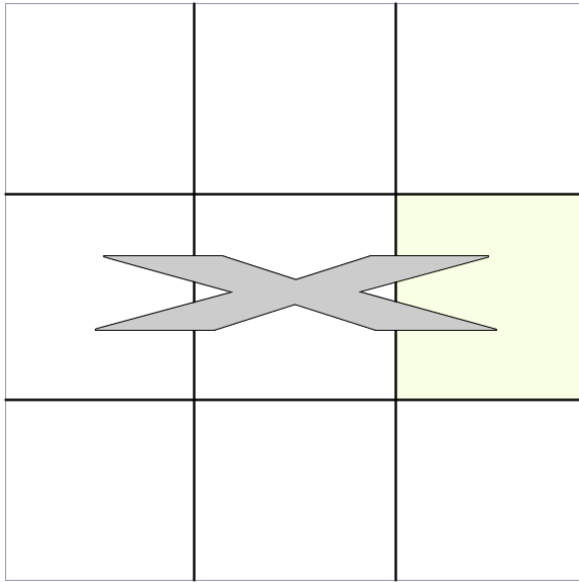
# How We Usually do DOM Traversal

---

`$("#cell123").parent()`

`$("#row3").parent()`

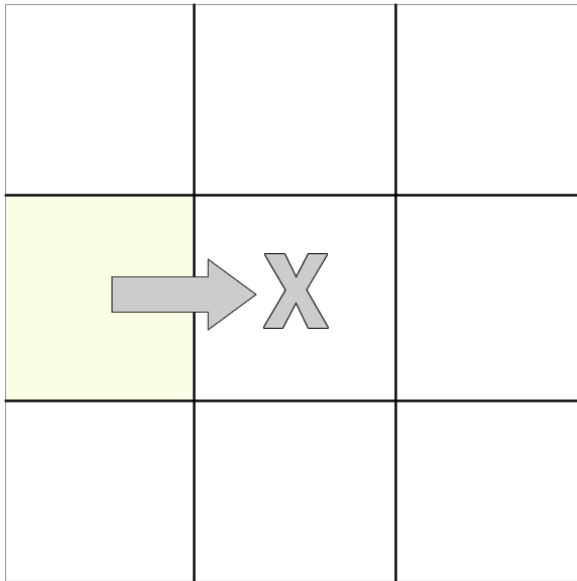
`$("#cell133").parents()`



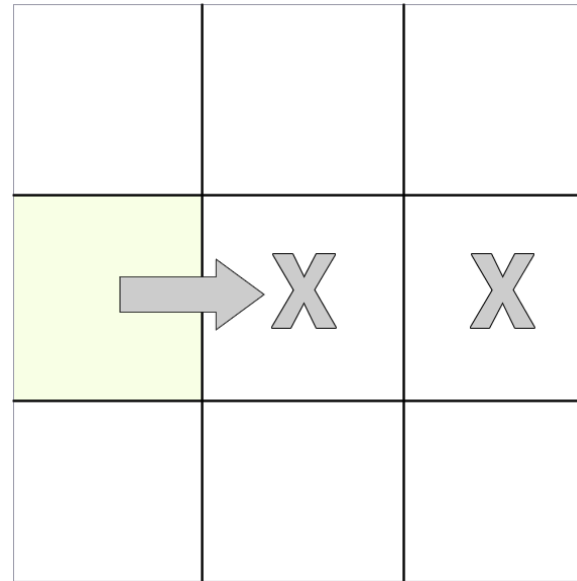
# How We Usually do DOM Traversal

---

`$("#cell21").next()`



`$("#cell21").nextAll()`

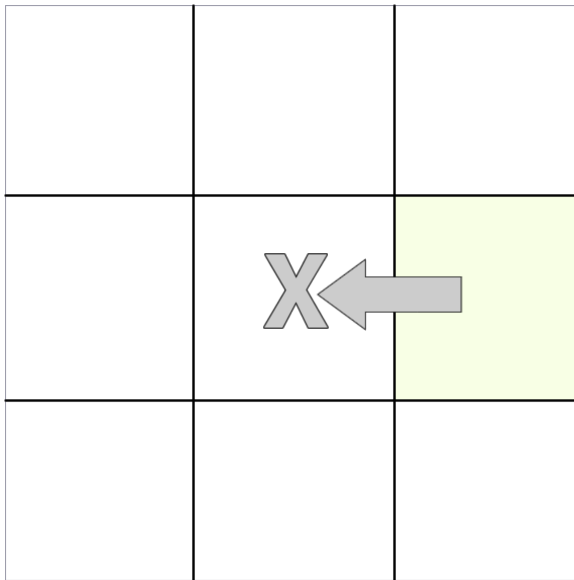




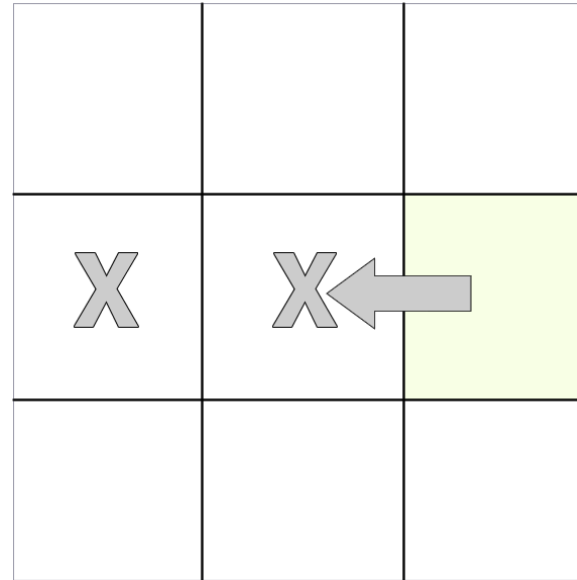
# How We Usually do DOM Traversal

---

`$("#cell23").prev()`



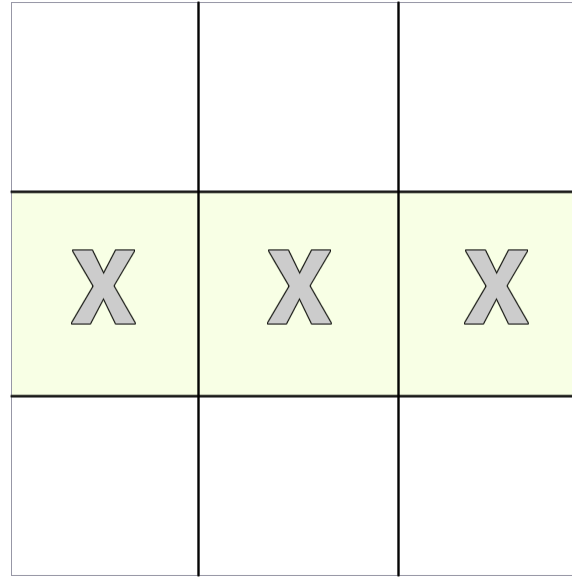
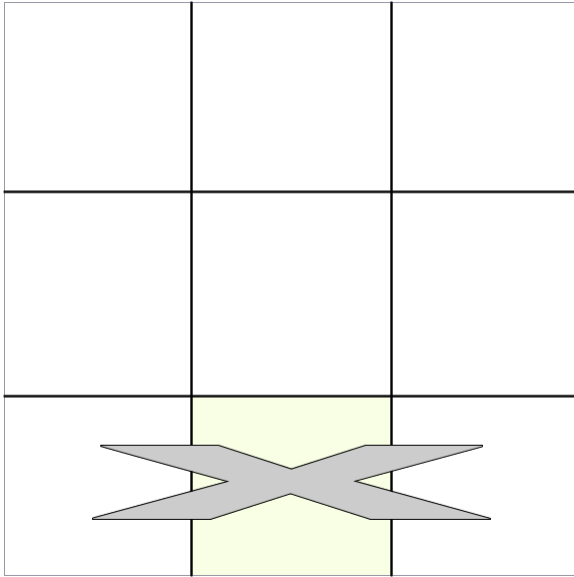
`$("#cell23").prevAll()`



# How We Usually do DOM Traversal

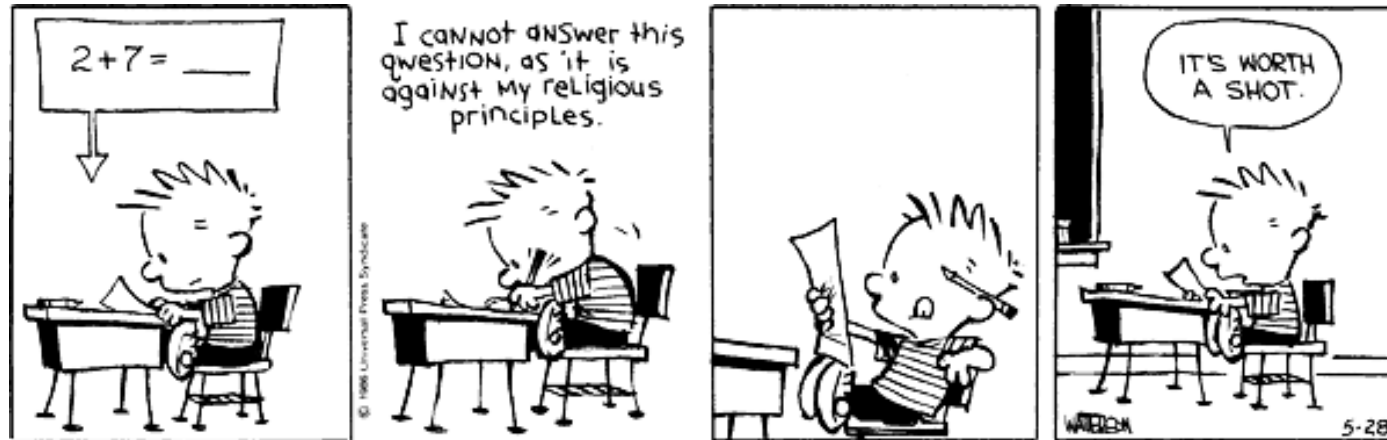
---

```
$("#cell132").closest(".row")     $("#row2").find(".cell")
```



# Do You Need jQuery?

---



# You Might Not Need jQuery

---

## YOU MIGHT NOT NEED JQUERY

jQuery and its cousins are great, and by all means use them if it makes it easier to develop your application.

If you're developing a library on the other hand, please take a moment to consider if you actually need jQuery as a dependency. Maybe you can include a few lines of utility code, and forgo the requirement. If you're only targeting more modern browsers, you might not need anything more than what the browser ships with.

At the very least, make sure you know what [jQuery is doing for you](#), and what it's not. Some developers believe that jQuery is protecting us from a great demon of browser incompatibility when, in truth, post-IE8, browsers are pretty easy to deal with on their own.

Tweet "You might not need jQuery"

5,832

Star YouMightNotNeedjQuery on Github

3,076

Search...

What's the oldest version of IE you need to support?

☐ 8 ☒ 9 ☐ 10

<http://youmightnotneedjquery.com/>

# I'm Not Hating on jQuery!

---



# Plain Old JavaScript

---

Start with selectors:

```
document.querySelector("#item")
```

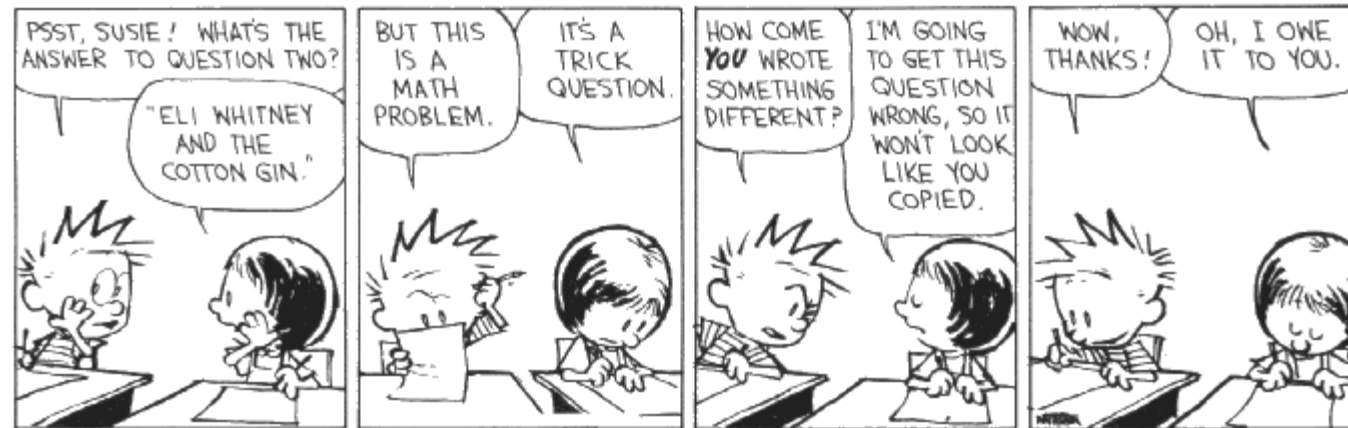
```
document.querySelectorAll(".fancybutton")
```

...?

```
document.querySelectorAll("ul")[0].children[2]
```

# That's Cheating!!!

---



# Plain Old JavaScript

---

Some pseudo-selectors in jQuery are convenience methods:

```
function selectEven(elems) {  
    var i = 0, length = elems.length, matches = [];  
    for ( ; i < length; i += 2 ) {  
        matches.push(elems[i]);  
    }  
    return matches;  
}  
  
selectEven(document.querySelectorAll("li"));
```



# Plain Old JavaScript

---

```
function getSiblings(elem) {  
    var i = 0, n = elem.parentNode.firstChild, matches = [];  
    for ( ; n; n = n.nextSibling )  
        if ( n.nodeType == 1 && n != elem )  
            matches.push( n );  
    return matches;  
}
```

`getSiblings(document.querySelector("#cell111"))`

	X	X

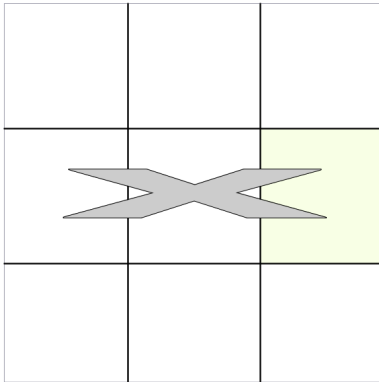
`getSiblings(document.querySelector("#row1"))`

X	X	X
X	X	X

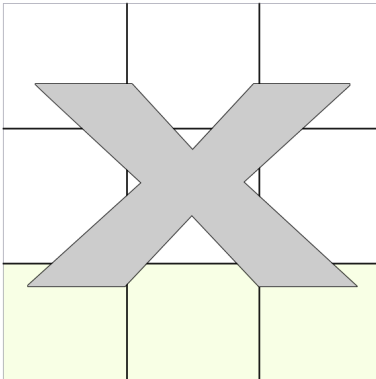
# Plain Old JavaScript

---

```
document.querySelector("#cell23").parentNode
```



```
document.querySelector("#row3").parentNode
```

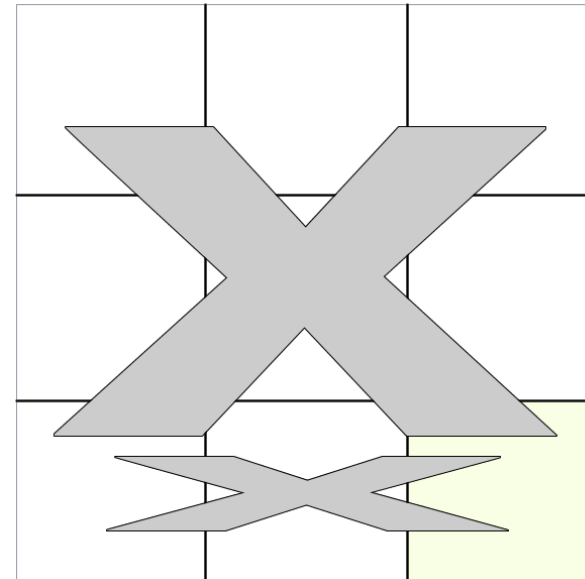


# Plain Old JavaScript

---

```
function getParents(elem) {  
    var n = elem.parentNode; matches = [];  
    for ( ; n; n = n.parentNode ) {  
        if ( n.nodeType == 9 ) {  
            return matches;  
        }  
        else if ( n.nodeType == 1 )  
            matches.push( n );  
    }  
}
```

```
getParents(document.querySelector("#cell33"))
```

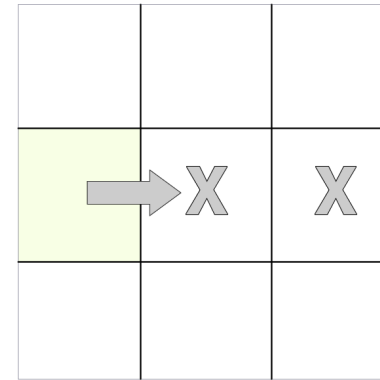


# Plain Old JavaScript

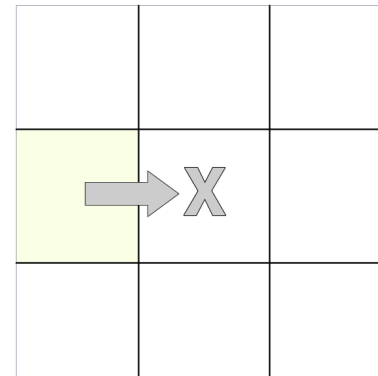
---

```
function getNext(elem,all) {  
    var n = elem.nextSibling, matches = [];  
    for ( ; n; n = n.nextSibling ) {  
        if ( n.nodeType == 1) {  
            matches.push( n );  
            if (!all) return matches;  
        }  
    }  
    return matches;  
}
```

`getNext(document.querySelector("#cell121"))`



`getNext(document.querySelector("#cell121"),true)`

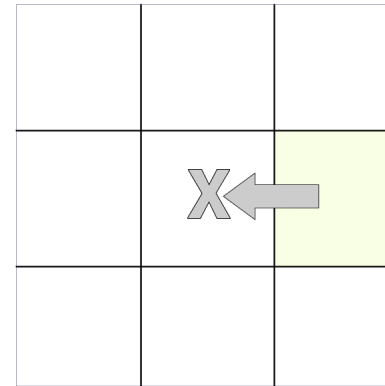


# Plain Old JavaScript

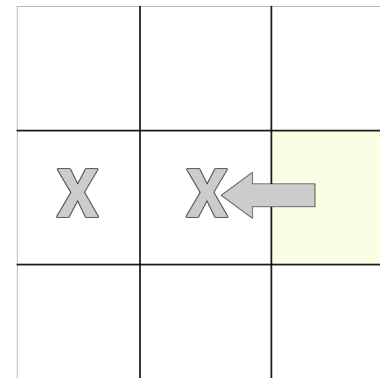
---

```
function getPrev(elem,all) {  
    var n = elem.previousSibling, matches = [];  
    for ( ; n; n = n.previousSibling ) {  
        if ( n.nodeType == 1 ) {  
            matches.push( n );  
            if (!all) return matches;  
        }  
    }  
    return matches;  
}
```

`getPrev(document.querySelector("#cell123"))`



`getPrev(document.querySelector("#cell123"),true)`

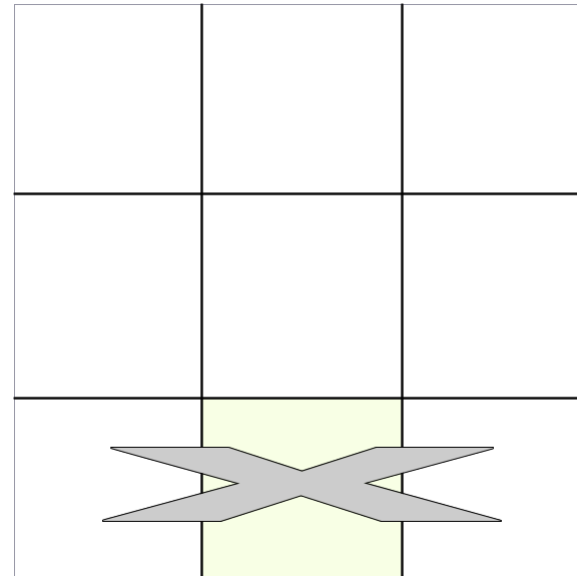


# Plain Old JavaScript

---

```
function getClosest(elem, selector) {  
    var n = elem, el;  
    for ( ; n; n = n.parentNode ) {  
        el = n;  
        // browser compatibility for matchesSelector:  
        // https://developer.mozilla.org/en-US/docs/Web/API/Element.matches#Browser_compatibility  
        if ( n.webkitMatchesSelector(selector))  
            return el;  
    }  
}
```

```
getClosest(document.querySelector("#cell32"), ".row");
```



# Plain Old JavaScript

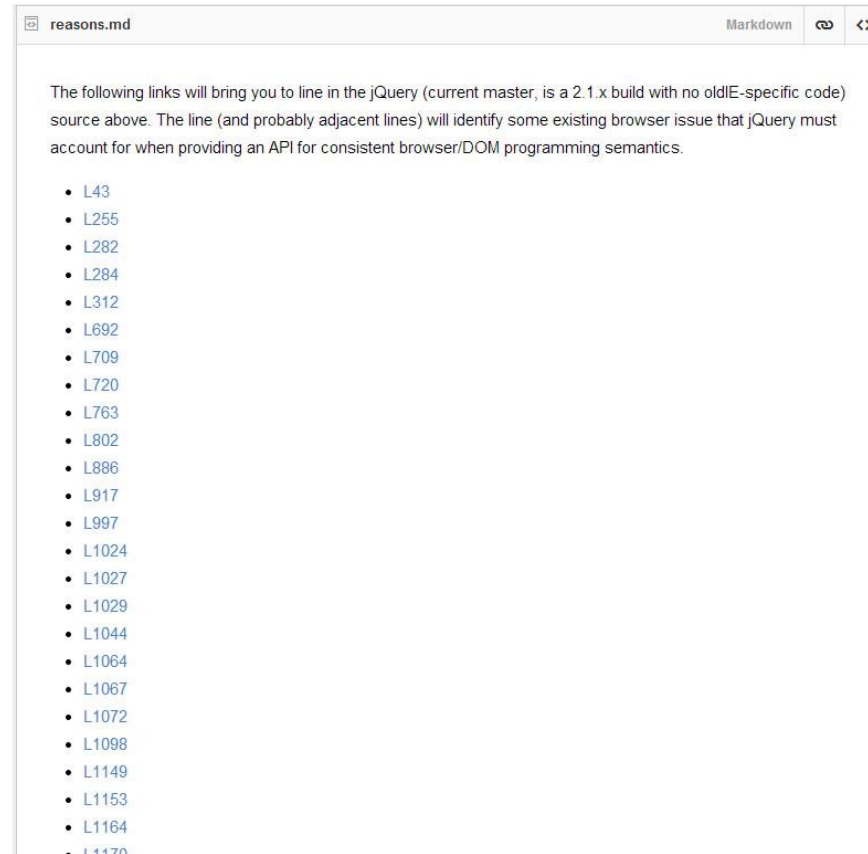
---

```
document.querySelector("#row2").querySelectorAll(".cell")
```

X	X	X

# You Might Need jQuery

---



The screenshot shows a GitHub Gist interface for a file named 'reasons.md'. The top bar includes the filename, a 'Markdown' tab, and icons for a link and code. The main content area contains a paragraph of text followed by a bulleted list of 25 jQuery issue links. The text explains that these links point to specific lines in the jQuery source code where browser compatibility issues are addressed. The list of links starts with 'L43' and ends with 'L1170'.

reasons.md    Markdown    🔗    <>

The following links will bring you to line in the jQuery (current master, is a 2.1.x build with no oldIE-specific code) source above. The line (and probably adjacent lines) will identify some existing browser issue that jQuery must account for when providing an API for consistent browser/DOM programming semantics.

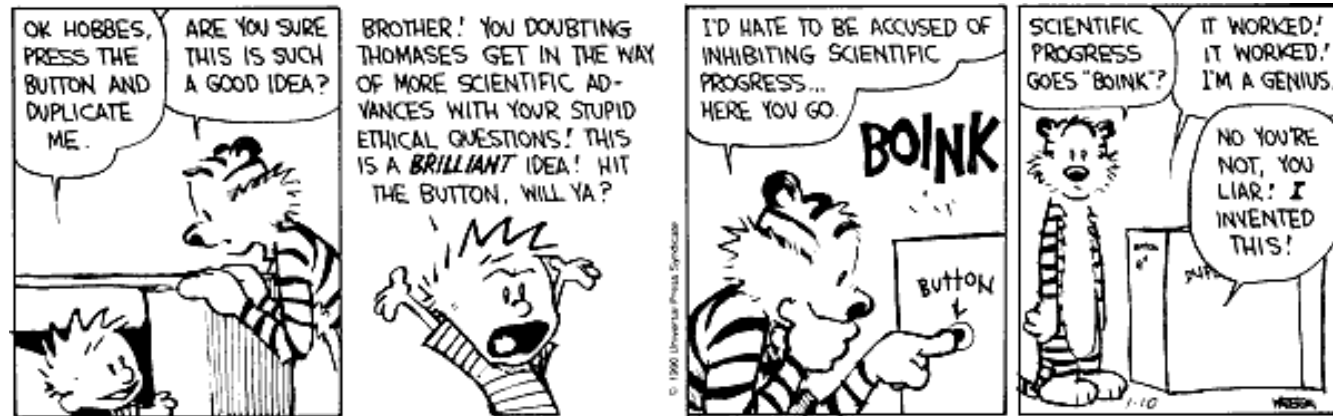
- [L43](#)
- [L255](#)
- [L282](#)
- [L284](#)
- [L312](#)
- [L692](#)
- [L709](#)
- [L720](#)
- [L763](#)
- [L802](#)
- [L886](#)
- [L917](#)
- [L997](#)
- [L1024](#)
- [L1027](#)
- [L1029](#)
- [L1044](#)
- [L1064](#)
- [L1067](#)
- [L1072](#)
- [L1098](#)
- [L1149](#)
- [L1153](#)
- [L1164](#)
- [L1170](#)

<https://gist.github.com/rwaldron/8720084#file-reasons-md>



# You Might Need jQuery

---



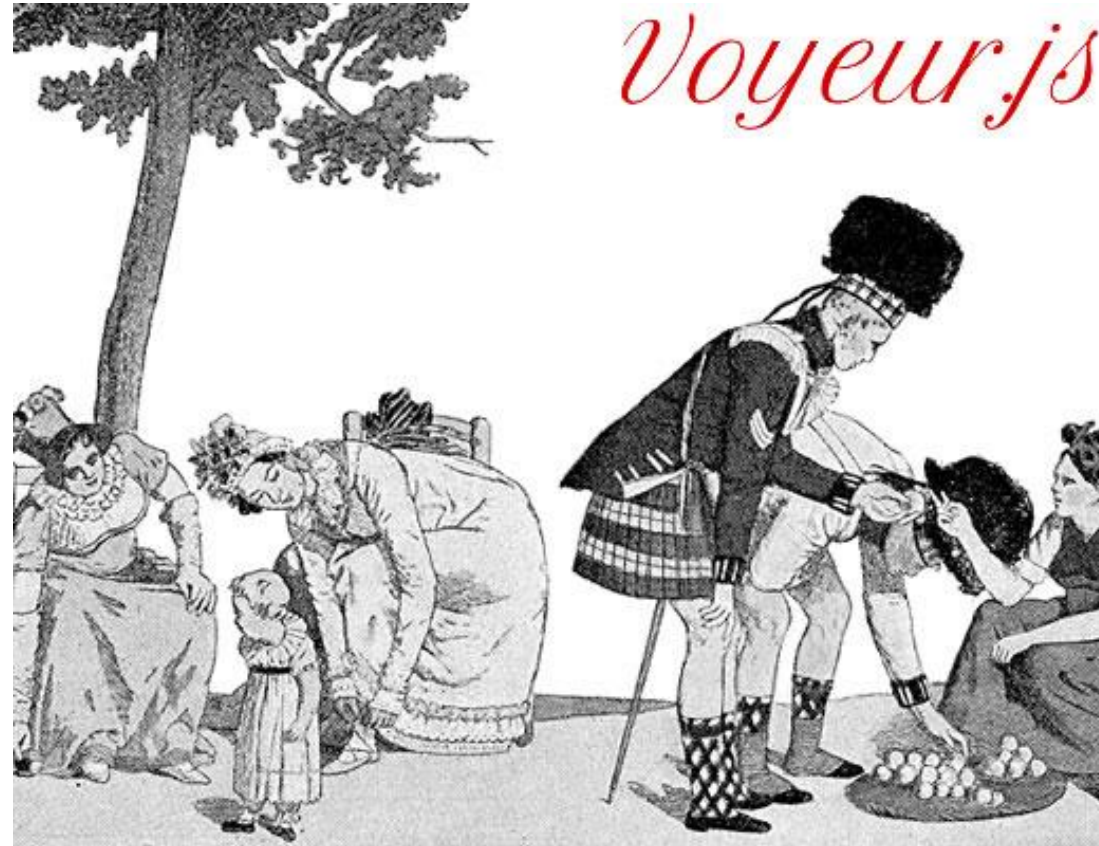
# And Now For Something Completely Different?

---



# And Now For Something Completely Different?

---



<http://flippinawesome.org/2013/07/22/dom-traversal-and-manipulation-with-voyeur/>

# And Now For Something Completely Different?

---

“ The way I see it, I have two options.

1. Change the syntax. I feel the whole appeal of Voyeur is it's syntax. From the feedback I received, the ability to traverse the DOM like a Javascript object really seemed to resonate with developers so taking this option would defeat the purpose of Voyeur.
2. Freeze the project until the performance is acceptable.

This may seem like a cop out but I can see no other option that will allow Voyeur to maintain it's uniqueness. I'm not the only one who has noticed the Object.defineProperty performance so I think putting the project on a halt until the various engine's catch up and the performance of Voyeur is satisfactory is the right way to go.

I'm going to go with option 2. I have left a visible warning on the README. Voyeur can be used for trivial projects and in the console but **it should never used in production, under any circumstances.** ”

# And Now For Something Completely Different?

## HTML(js)

Befriend the DOM!

HTML is a small, powerful way for you to enjoy working directly with the DOM. [HTML.min.js \(~2.7kb, gzip\)](#)

[Tweet](#) 478 [Fork me on GitHub](#)

**INPUT**

```
//Now you try it out for yourself! Edit me.
```

**OUTPUT**

```
<div>
  <h1><em>Hello world!</em></h1>
  <section id="empty"></section>
  <section id="full">
    <ul>
      <li>foo</li>
      <li>bar</li>
      <li>baz</li>
      <li>woogie</li>
    </ul>
  </section>
</div>
```

**Intuitive**

HTML lets you navigate, manipulate and use the DOM with intuitive, readable, consistent code.

**Powerful**

HTML's functions enable you to flexibly use all native DOM features with brevity, clarity, and more power than ever.

**Extensible**

HTML has a modular design that is easy to extend, adapt or customize to meet your needs.

[See our F.A.Q.!](#)

### API

**HTML** `document.documentElement`

The global HTML is the actual document root element and all element tags queried via the dot operator are descendants.

**HTML.**`<tag...>` `Element|Array`

HTML allows you to access document elements via their tag name. If multiple elements of the same tag exist, a proper *array* of elements is returned; otherwise, an `Element` is returned. Only the first node returned is "HTML-ified" (prepared for further chaining).

<http://nbubna.github.io/HTML/>

# HTML.js

---

Start with selectors:

```
HTML.query("#item")
```

```
HTML.query(".fancybutton")
```

```
HTML.query("li").only(function(o,i){ return !(i%2); })
```

```
HTML.query("ul").only(2)
```

# HTML.js

---

```
HTML.query("#row1").div.only(function(o,i){ if (!o.webkitMatchesSelector("#cell11")) return o; })
```

X	X

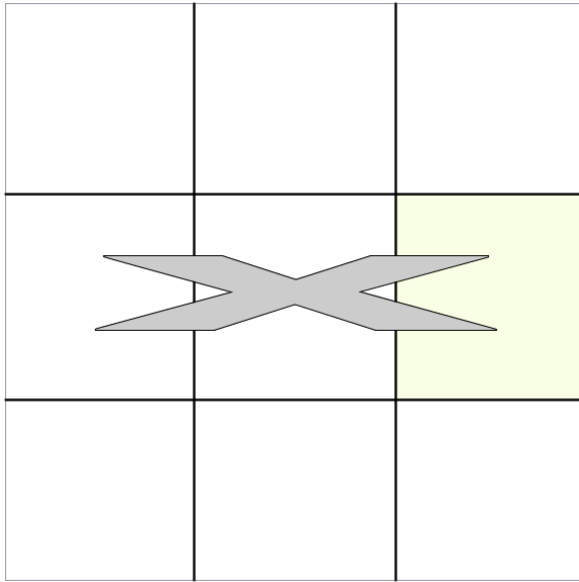
```
HTML.query(".board").div.only(function(o,i){ if (!o.webkitMatchesSelector("#row1")) return o; })
```

X	X	X
X	X	X

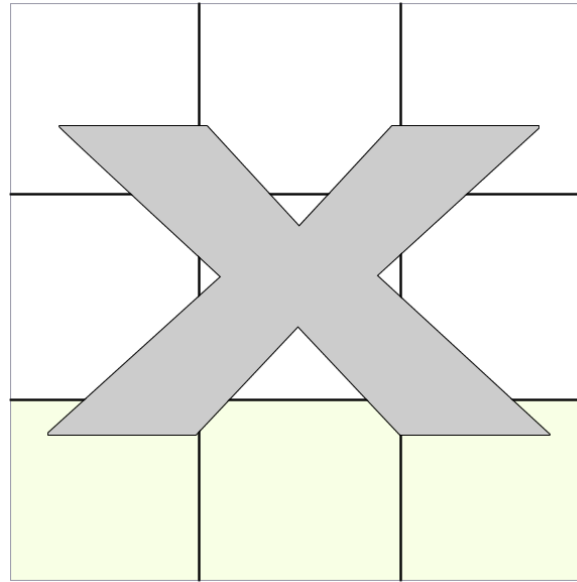
# HTML.js

---

`HTML.query("#cell23").parentNode`



`HTML.query("#row3").parentNode`



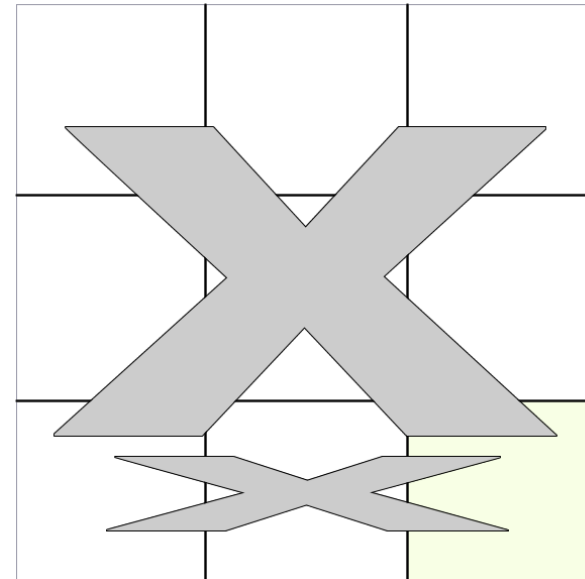


# HTML.js

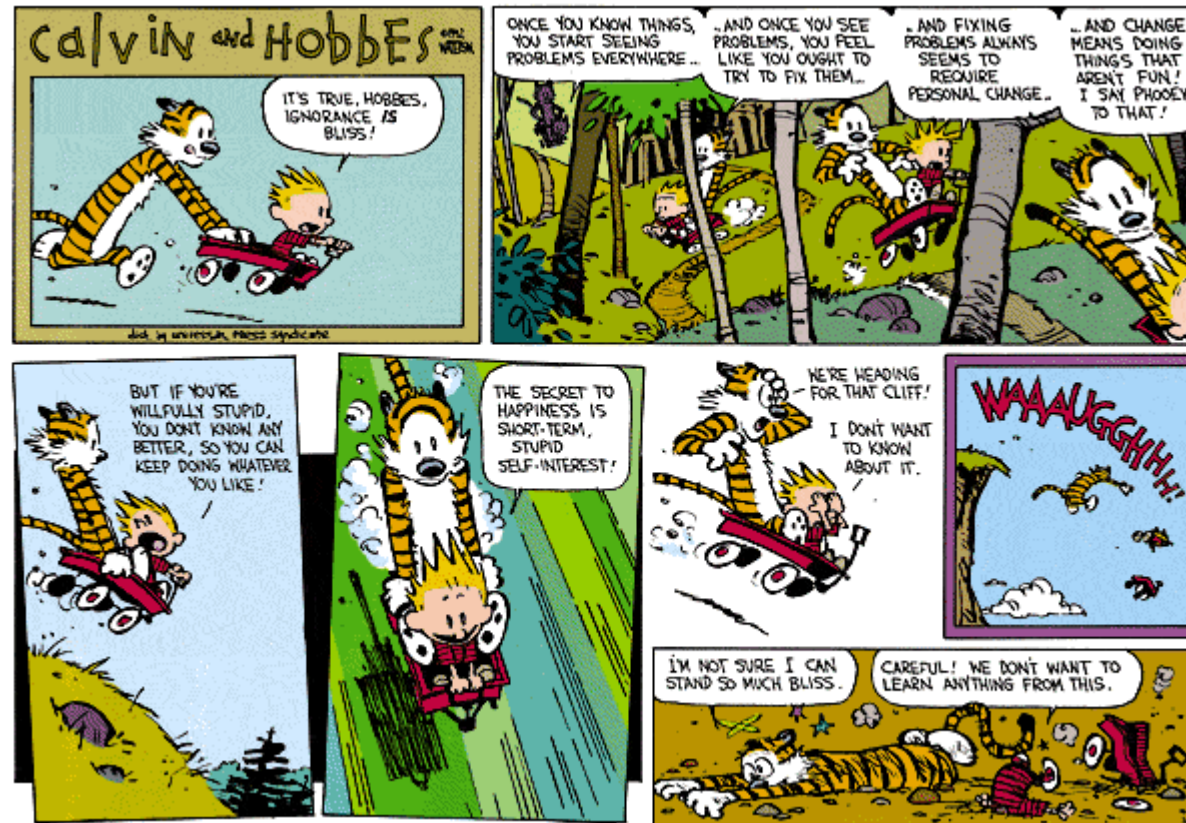
---

```
function getParents(elem) {  
    var n = elem; matches = [];  
    for ( ; n; n = n.parentNode ) {  
        if ( n.nodeType == 9 ) {  
            return matches;  
        }  
        else if ( n.nodeType == 1 )  
            matches.push( n );  
    }  
}
```

```
getParents(HTML.query("#cell133"))
```



# This Doesn't Seem Any Better!?!






# HTML.js

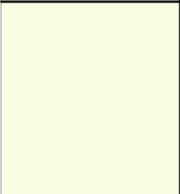

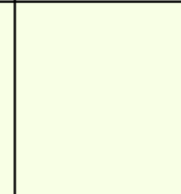
---

## Chaining:

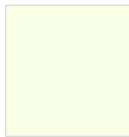
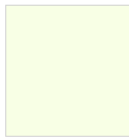

`HTML.body.div.div.div`

`HTML.body.div.div.only(1)`

`HTML.body.div.div.only(1).query(".cell")`

# HTML.js

---

Each:

```
HTML.body.div.div.query(".cell").each(function(el, i, all) {  
    el.textContent = "X";  
});
```

X	X	X
X	X	X
X	X	X

# Wrap-up

---

jQuery still offers the easiest and cleanest API for DOM traversal...

...however, jQuery might be more than you need.

In many cases, you can replace 1 line of jQuery with 1 line of plain JavaScript...

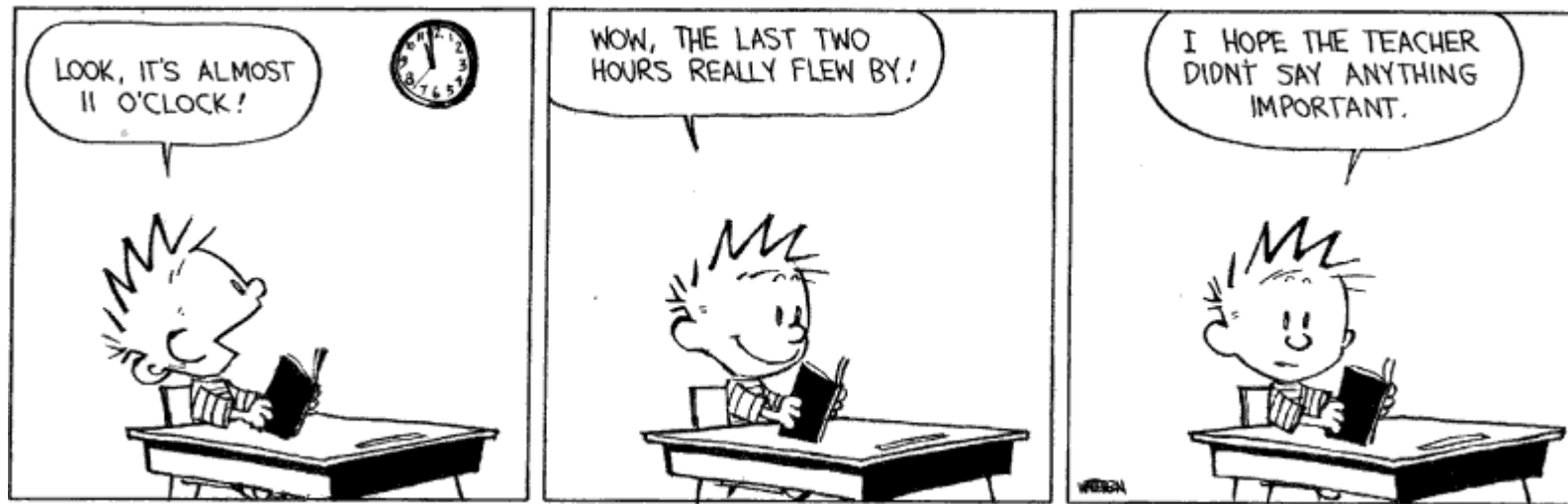
...however, it's not always so easy and you may end up rewriting a lot of methods.

HTML.js offers a unique way to traverse the DOM that can feel intuitive in many cases...

...however, it's missing a lot of methods that you might need.

# Questions?

---



Contact: Brian Rinaldi  
[@remotesynth](https://twitter.com/remotesynth)  
[brian.rinaldi@telerik.com](mailto:brian.rinaldi@telerik.com)