Code Fellows

jQuery

A JavaScript library for working with the DOM and handling browser events

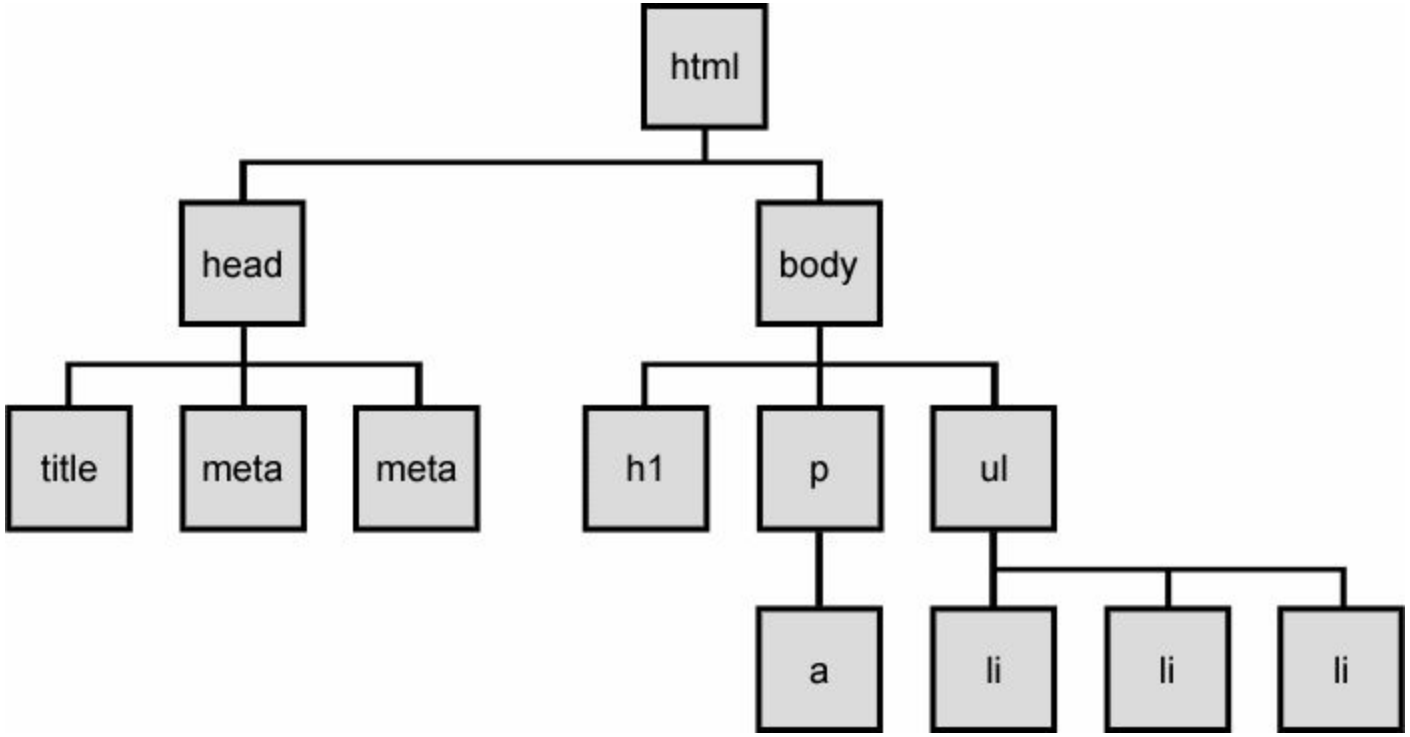
TABLE OF CONTENTS

The DOM

What is it?

NOPE.

NOPE.



YEP.

What is the DOM?

- Document Object Model
- What the browser does with HTML
- Internal representation of a web page

What is the DOM?

“The DOM states what your script can ask the browser about the current page, and how to tell the browser to update what is being shown to the user.”

JON DUCKETT

Author of our textbooks

How have we worked with the DOM so far?

Find elements:

- `var el = document.getElementById(‘articles’);`
- `var els = document.getElementsByTagName(‘li');`
- `var els = document.querySelectorAll(‘#articles > li');`

Change elements:

- `document.getElementById('articles').innerHTML = '<h1>News</h1>';`
- `document.getElementById('articles').textContent = 'News';`
- `document.getElementById('articles').innerText = ‘News1';`

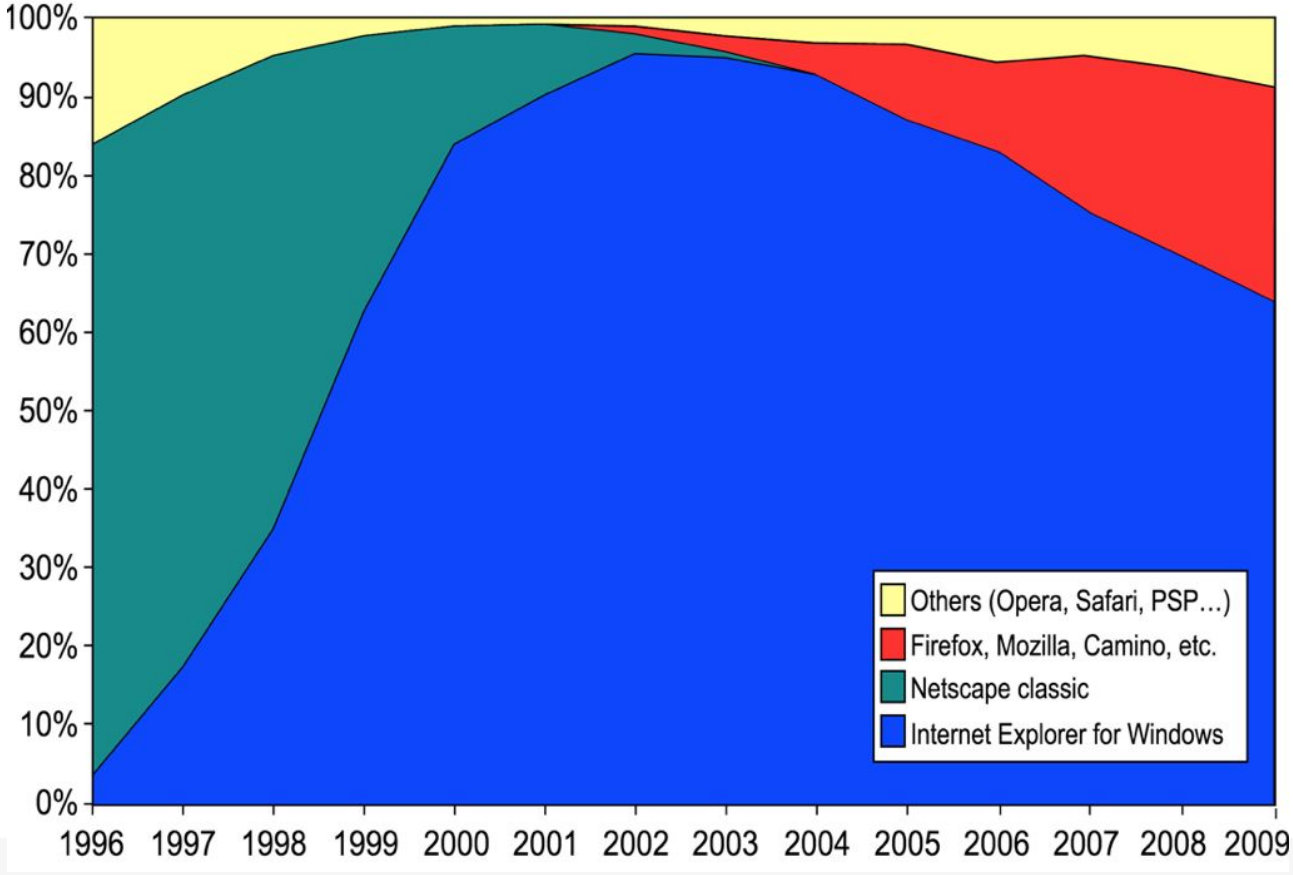
Also:

- Add and remove elements
- Read attributes and values
- Navigate the tree… Manage user interactions, events, etc…

jQuery

“jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.” - **jquery.com**

Browser Wars

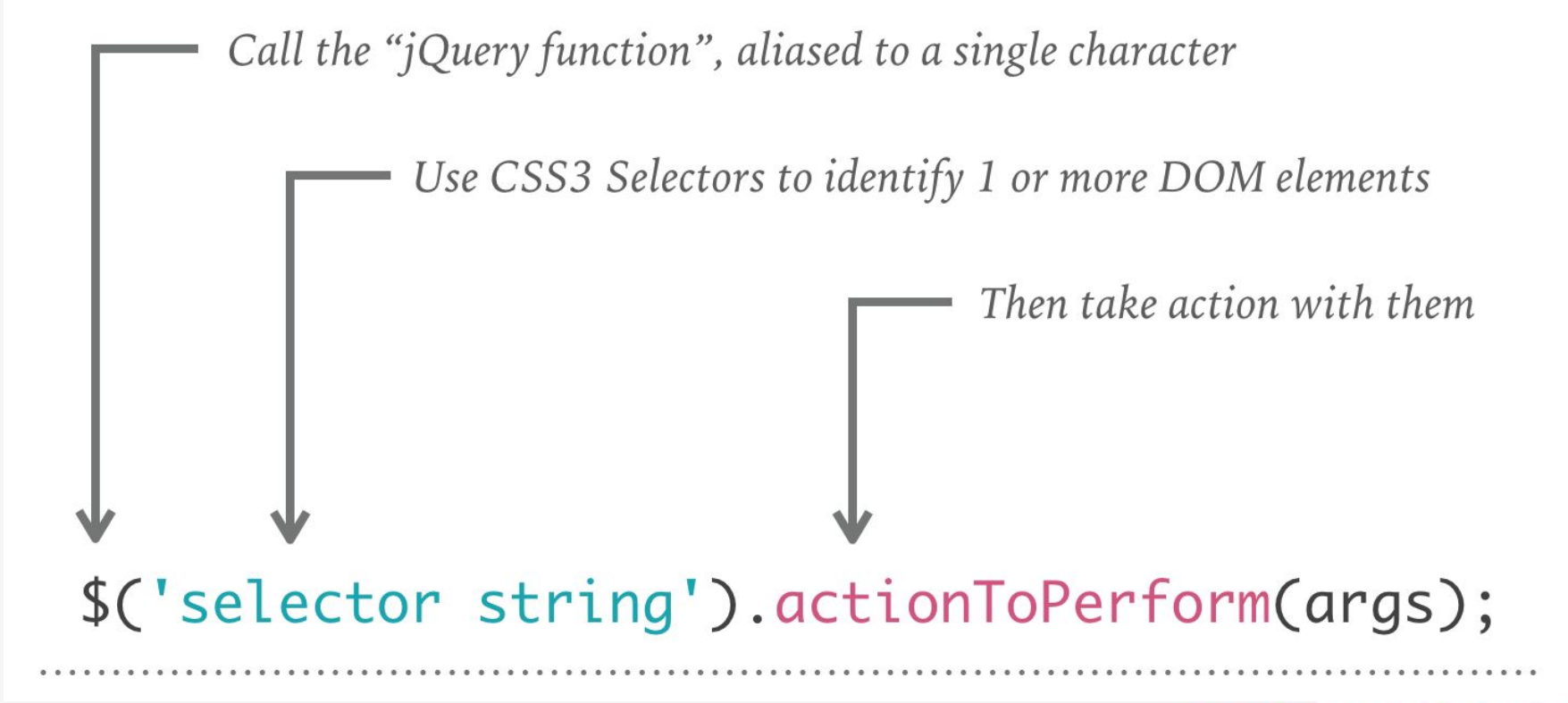



What are we?!
Browsers!
Browsers!
Browsers!
What do we want?!
More speed!
More speed!
More speed!
And when do we want it?!
Right now!
Right now!
Right now!
Browsers!
EATLIVER.COM

What is jQuery?

- A JavaScript library (10,361 LOC in v. 3.3.1)
- It provides an API for:
    - DOM selection
    - DOM manipulation
    - DOM traversal
    - Event management
    - AJAX
    - …and more!

What is jQuery?



jQuery and the DOM

It's time for a leaner and smoother way to do common DOM manipulation tasks!

VANILLA JS: FADE IN

```
function fadeIn(el) {
  el.style.opacity = 0;

  var last = +new Date();
  var tick = function() {
    el.style.opacity = +el.style.opacity + (new Date() - last) / 400;
    last = +new Date();

    if (+el.style.opacity < 1) {
      (window.requestAnimationFrame && requestAnimationFrame(tick)) ||
       setTimeout(tick, 16)
    }
  };

  tick();
}

fadeIn(el);
```

JQUERY: FADE IN

```
$(el).fadeIn();
```

VANILLA JS: TOGGLE A CLASS

```
1 if (el.classList) {
2   el.classList.toggle(className);
3 } else {
4   var classes = el.className.split(' ');
5   var existingIndex = classes.indexOf(className);
6
7   if (existingIndex >= 0)
8     classes.splice(existingIndex, 1);
9   else
10    classes.push(className);
11
12  el.className = classes.join(' ');
13 }
```

JQUERY: TOGGLE A CLASS

```
1 $(el).toggleClass(className);
```

VANILLA JS: GET JSON FROM A SERVER

```
var request = new XMLHttpRequest();
request.open('GET', '/my/url', true);
request.onload = function() {
  if (request.status >= 200 && request.status < 400) {
    // Success!
    var data = JSON.parse(request.responseText);
  } else {
    // Server returned an error
  }
};
request.onerror = function() {
  // There was a connection error
};
request.send();
```

JQUERY: GET JSON FROM A SERVER

```
$.getJSON('/my/url', function(data) {
  // Do something with the data
});
```

Events in the browser

Event handling with jQuery offers many advantages and streamlined interfaces as compared with traditional vanilla techniques!

Browser events

- When _____, do _____.
- When the user does something, run some code.
- When an event is fired, invoke a function.
- We register a callback function as an event handler

- It’s a simple concept…with some radical implications
    - Out of order code execution
    - aka: “Asynchronous Callbacks”

Browser events

Which ones do we care about the most?

A selection of useful events: pp 246-247 (Duckett):

- Browser UI: `load, unload, resize, scroll`
- Mouse: `click, dblclick, mouseover, mouseout`
- Keyboard: `keyup, keydown, keypress`
- Element focus: `focus, blur`
- Form: input, `change, submit`

jQuery and browser events

“Events are the beating heart of any JavaScript application.”

--Peter-Paul Koch, quirksmode.org


THIS IS
JAVASCRIPT

jQuery and browser events

Goodbye!

```
<a href="#" onclick="jumpHigh();">Jump!</a>
```

and...

```
domEle.onclick = function() { height = prompt('How high?'); };
```

and...

```
var domEle = document.getElementById('some-dom-element');
domEle.addEventListener('click', jumpHigh, true);
```

Hello!

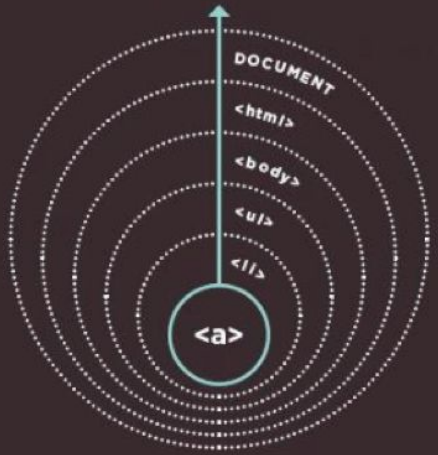
```
$el.on('click', jumpHigh);
```

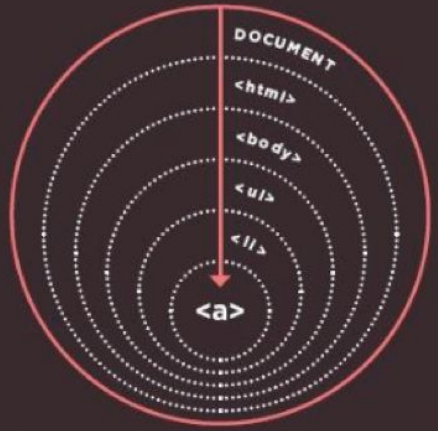
Let's take a look: http://api.jquery.com/on/

HTML elements nest inside other elements. If you hover or click on a link, you will also be hovering or clicking on its parent elements.

Imagine a list item contains a link. When you hover over the link or click on it, JavaScript can trigger events on the <a> element, and also any elements the <a> element sits inside.

Event handlers/listeners can be bound to the containing <li>, <ul>, <body>, and <html> elements, plus the **document** object, and the **window** object. The order in which the events fire is known as **event flow**, and events flow in two directions.





EVENT BUBBLING

The event starts at the *most* specific node and flows outwards to the *least* specific one. This is the default type of event flow with very wide browser support.

EVENT CAPTURING

The event starts at the *least* specific node and flows inwards to the *most* specific one. This is not supported in Internet Explorer 8 and earlier.

Browser events

- JavaScript can respond to hundreds of events
- You can get a “handle” on them with jQuery
- The .on() method is your go-to method for attaching event handlers
- ***Delegation can reduce the number of handlers you need to attach***
- ***Delegation can capture events for elements not in the page when the handler is registered***
- Events bubble from the in-most outward

Code Fellows

jQuery

A JavaScript library for working with the DOM and handling browser events