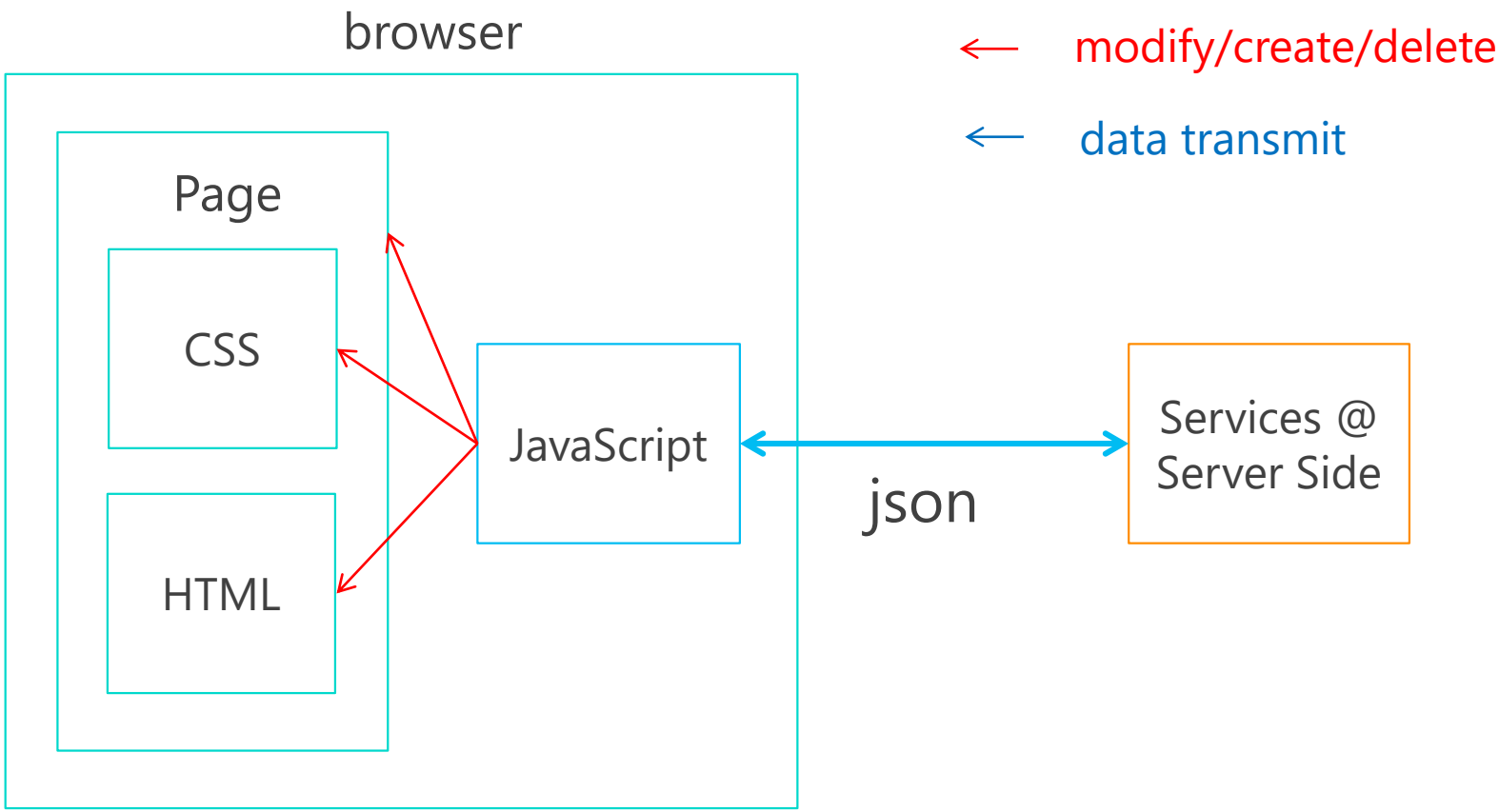
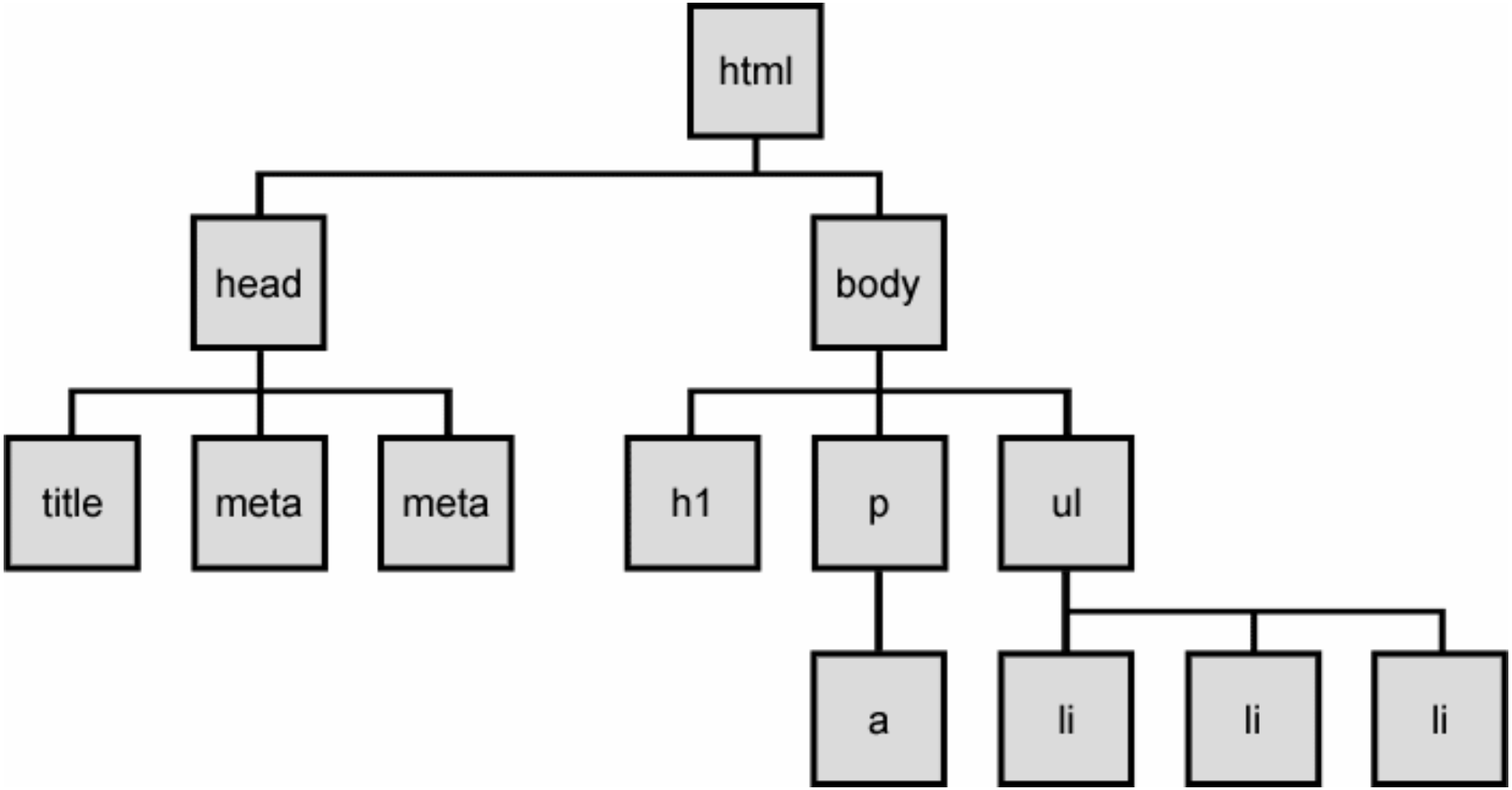
DOM & jQuery

How JavaScript fits in the big Picture



The DOM tree



Document Object Model (DOM)

- Every HTML element is accessible via the JavaScript DOM API
- The **event model** lets a document to react when the user does something on the page
- Advantages
  - Create interactive pages
  - Updates the objects of a page without reloading it

Example

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```

DOM Element Object

Property	Value
tagName	"IMG"
src	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing Elements

Access elements via their ID attribute

```
var element = document.getElementById("some-id")
```

Via the **name** attribute

```
var elArray = document.getElementsByName("some-name")
```

Via tag name

```
var imgTags = document.getElementsByTagName("img")
```

- Returns array of `<img>` elements

DOM Manipulation

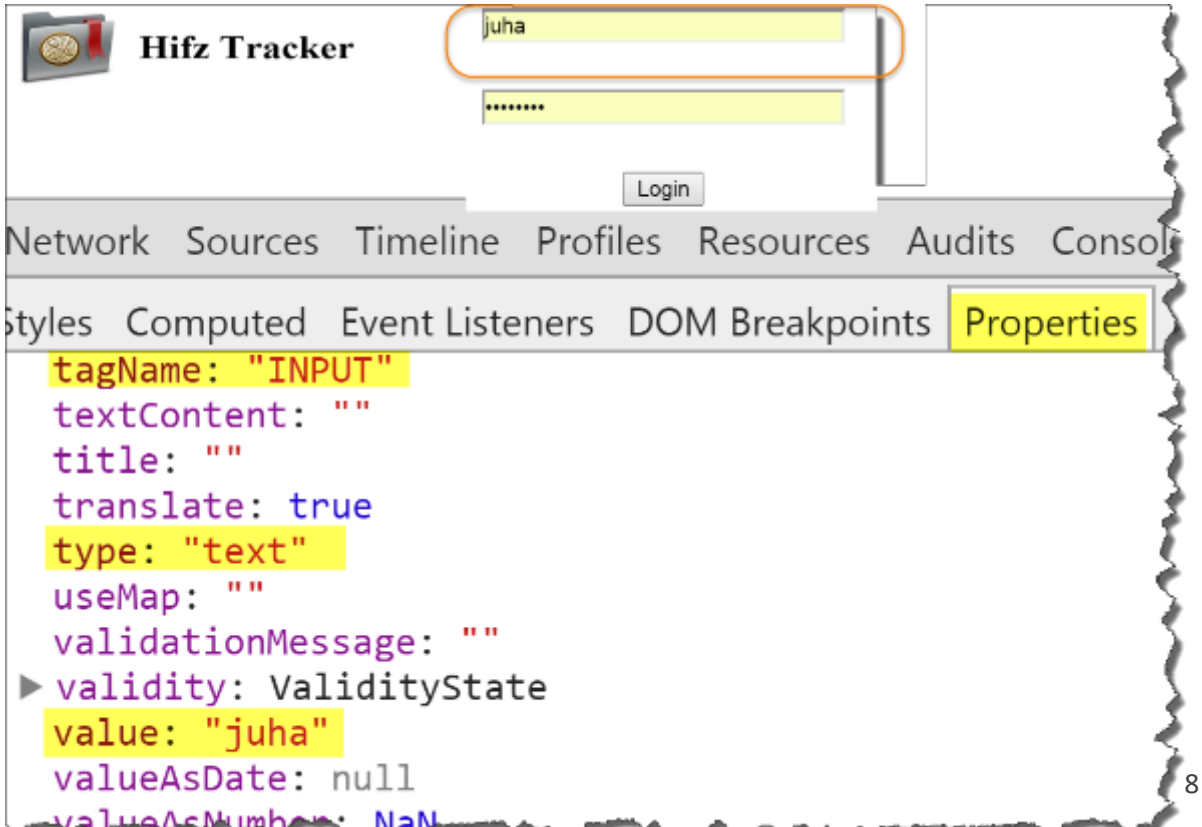
Once we access an element, we can read and write its attributes

```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".png";
  var statusDiv =
    document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state";
}
…
<img src="test_on.gif"  id="lamp"
  onmouseover="change('off')"
  onmouseout="change('on')" />
```

Common Element Properties

- `innerHTML` – holds all the entire HTML code inside the element
- `className` – the `class` attribute of the tag

User Chrome Dev Tool to see the Properties of Page element



The HTML DOM Event Model

JavaScript can register event handlers

- Events are fired by the Browser and are sent to the specified JavaScript event handler function
- Can be set with HTML attributes:

```
<img src="test.gif" onclick="imageClicked()" />
```

- Can be set through the DOM:

```
var img = document.getElementById("myImage");
img.addEventListener('click', imageClicked);
```

Common DOM Events

Mouse events:

- `onclick`, `onmousedown`, `onmouseup`
- `onmouseover`, `onmouseout`, `onmousemove`

Key events:

- `onkeypress`, `onkeydown`, `onkeyup`
- Only for input fields

Interface events:

- `onblur`, `onfocus`
- `onscroll`

Common DOM Events (2)

Form events

- `onchange` – for input fields
- `onsubmit`
  - Allows you to cancel a form submission
  - Useful for form validation

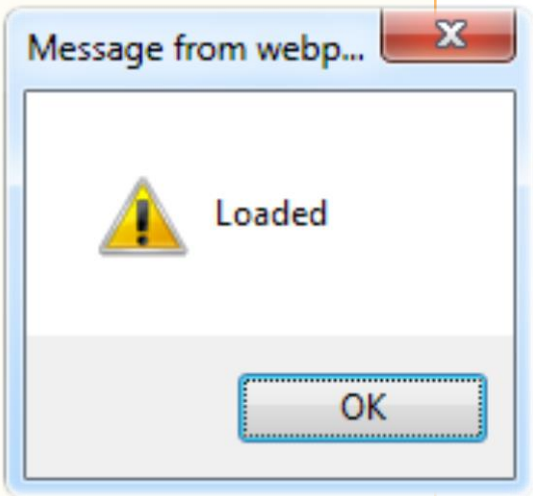
Document events

- `onload`
  - Allowed only for the `<body>` element
  - Fires when all content on the page was loaded

onload Event – Example

onload event

```
<html>
<head>
  <script type="text/javascript">
    function greet() {
      alert('Loaded!');
    }
  </script>
</head>
<body onload="greet()" >
</body>
</html>
```



Event Handler

```
<script>

document.getElementById("myBtn").
addEventListener("click",
displayDate);


function displayDate() {

document.getElementById("demo").innerHTML =
Date();

}

</script>
```

Try it @
http://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_displaydate

Introduction to jQuery

jQuery

- Simplifies HTML document traversing, event handling, animating, and AJAX.

- To include jQuery in your website, all you need is a script tag with its *src* pointed to the hosted location.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
```

jQuery key capabilities

- Accessing HTML elements
- Set or get HTML element properties
- Handle HTML element events
- Traverse HTML elements as nodes
- Interact with service services using Ajax
- Animation

jQuery Syntax

- You can use the **$()** function to **select** HTML elements and perform some **action** on the element(s)
- Basic syntax is: **$(*selector*).*action*()**
  - A **$** sign to define/access jQuery
  - A (***selector***) to "query (or find)" HTML elements
  - A jQuery ***action***() to be performed on the element(s)

jQuery Selectors

jQuery supports CSS :

1. By element: `$(“div”)`
2. By id : `$(“#id”)`
3. By class: `$(“.classname”)`
4. By attribute: `$(“a[href]”)`
5. …

DOM method	jQuery equivalent
getElementById("id")	$("#id")
getElementsByTagName("tag")	$("tag")
getElementsByName("somename")	$("[name='somename']")

jQuery Syntax

Examples:

- $("p").hide() - hides all <p> elements
- $(".test").hide() - hides all elements with class="test"
- $("#test").hide() - hides the element with id="test“
- $('div').css('background', 'blue'); - Make all DIVs blue

jQuery – Selector Elements

HTML element access examples:

```
<h2 id="acer">Acer</h2>
<h2 id="ibm">???</h2>

-----------------------------

$("h2").css("color", "blue");

$("#htc").text("lenovo");

$("h2").click(
  function() {
    $(this).css("color", "red");
  }
);
```

.ready() event

- jQuery provides a *ready* event that is fired when the document is ready to be manipulated

- You'll put most of your code in this method

```
$(document).ready(function(){
  // Your code here e.g.,
  alert("Ok document is ready…");
});
```

jQuery – Basic Scripting Structure

Typical structure:

```
$(document).ready(
  function() {
    alert("Ok document is ready…");
    myTask();
  }
);

function myTask {
  $(selector).action(
      function() {
        other tasks...
      }
  );
}
```

Creating Elements

Creating new elements is also easy

- with the **jQuery** HTML parser

```
var divElement = $('<div>');
var anotherDiv = $('<div />');
var paragraph = $('<p>Some text</p>');
```

- with **document.createElement**
  - A little bit faster

```
var divElement =
$(document.createElement('div'));
```

Adding Elements

Adding elements can be done on the fly

- **jQuery.appendTo()** / **jQuery.prependTo()**
- **jQuery.append()** / **jQuery.prepend()**

```
<h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, student</div>
  <div>Goodbye, student</div>
</div>
```

```
h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, student<p>Test</p></div>
  <div>Goodbye, student<p>Test</p></div>
</div>
```

```
$('#wrapper div').append('<p>Test</p>');
```

```
<div>First</div>
<h2>QU Greetings</h2>
<div id="wrapper">
  <div>Hello, studentdiv>
  <div>Goodbye, student<div>
</div>
```

```
$('<div>First</div>').prependTo('body');
```

Removing Elements

You can also remove elements from the DOM

- Just as easy

```
<div>
  <p>Red</p>
  <p>Green</p>
</div>

<script>
  $('p').remove(); // Remove all paragraphs
</script>
```

```
<!-- Result: -->
<div>
</div>
```

jQuery Events

jQuery has a convenient way for attaching and detaching events Using methods **on()** and **off()**

```
function onButtonClick() {
  $(this).hide();
// "this" is the event source (the button clicked)
}

$('#button').on('click', onButtonClick);
```

Looping over the DOM

Using the DOM

```
var elems = document.querySelectorAll("li");
for (var i = 0; i < elems.length; i++) {
      var e = elems[i];
      // do stuff with e
}
```

Using jQuery

```
$("li").each(function(idx, e) {
      // do stuff with e
});
```

Inside the jQuery each loop

```
$("li").each(function(idx, e) {
      // do stuff with e
});
```

- return false to exit the loop early
- e is a plain old DOM object
  - We can upgrade it again using $ if we want

```
$("li").each(function(idx, e) {
    e = $(e); // do stuff with e
});
```

Getting/setting CSS classes in jQuery

```
function highlightField() {
      // turn text yellow and make it bigger
      if (!$("#myid").hasClass("invalid")) {
            $("#myid").addClass("highlight");
      }
}
```

- addClass, removeClass, hasClass, toggleClass manipulate CSS classes

Accessing styles in jQuery

```
function biggerFont() {
      // turn text yellow and make it bigger
      var size = parseInt($("#clickme").css("font-size"));
      $("#clickme").css("font-size", size + 4 + "pt");
}
```

- css function of the jQuery object allows reading pre-existing styles
- css(*property*) gets the property value, css(*property*, *value*) sets the property value

jQuery css method parameters

getter syntax:

```
$("#myid").css(propertyName);
```

setter syntax:

```
$("#myid").css(propertyName, value);
```

multi-setter syntax:

```
$("#myid").css({
        'propertyName1': value1,
        'propertyName2': value2,
        ...
        });
```

modifier syntax:

```
$("#myid").css(propertyName, function(idx, oldValue) {
        return newValue;
});
```

More node manipulation with jQuery

jQuery method	functionality
.hide()	toggle CSS display: none on
.show()	toggle CSS display: none off
.empty()	remove everything inside the element, innerHTML = ""
.html()	get/set the innerHTML without escaping html tags
.text()	get/set the innerHTML, HTML escapes the text first
.val()	get/set the value of a form input, select, textarea, ...
.height()	get/set the height in pixels, returns a Number
.width()	get/set the width in pixels, return a Number

Creating complex nodes in jQuery

- **The terrible way, this is no better than innerHTML hacking**

```
$("<p id='myid' class='special'>My paragraph is awesome!</p>")
```

- **The bad way, decent jQuery, but we can do better**

```
$("<p>")
        .attr("id", "myid")
        .addClass("special")
        .text("My paragraph is awesome!");
```

- **The good way**

```
$("<p>", {
        "id": "myid",
        "class": "special",
        "text": "My paragraph is awesome!"
});
```

jQuery Visual Effects

jQuery Visual Effects

Getting attention

- Highlight effect
- Scale effect
- Pulsate effect
- Shake effect

Appear

- show
- fadeIn
- slideDown
- slide effect

Disappear

- hide
- fadeOut
- slideUp

Disappear Visual Effects

- Blind effect
- Bounce effect
- Clip effect
- Drop effect
- Explode effect
- Drop effect
- Explode effect
- Fold effect
- Puff effect
- Size effect

Effect options

```
element.effect(effectName, {

    option: value,
    option: value,
    ...
});
```

```
$("#myid").effect("explode", {
    "pieces": 25
});
```

Applying effects to an element

```
$("#sidebar").slideUp();

// No need to loop over selected elements, as usual
$("#results > button").effect("pulsate");
```

- the effect will begin to animate on screen (asynchronously) the moment you call it
- One method is used behind the scenes to do most of the work, **animate()**

Effects chaining

```
$('#demo_chaining')
    .effect('pulsate')
    .effect('highlight')
    .effect('explode');
```

- Effects can be chained like any other jQuery methods
- Effects are queued, meaning that they will wait until the previous effects finish

Effect duration

- You can specify how long an effect takes with the duration option
- Almost all effects support this option
- Can be one of slow, normal, fast or any number in milliseconds

```
$('#myid').effect('puff', {}, duration)
```

Custom effects - animate()

```
$('#myid').animate(properties, [duration]);
```

- You can animate any numeric property you want
- You can also animate these
  - color
  - background-color

```
$('#myid')
      .animate({
            'font-size': '80px',
            'color': 'green'
      }, 1000);
```

Custom effects easing

```
$('#myid')
        .animate(properties, [duration], [easing]);
```

Your animations don't have to progress linearly

There are many other options

- slide
- easeInSin

```
$('#myid')
        .animate({
                'font-size': '80px',
                'color': 'green'
        }, 1000, 'easeOutBounce');
```

Better Custom Effects* - toggleClass()

* if you don't need easing or special options

use the toggleClass method with its optional duration parameter

```
.special {
    font-size: 50px;
    color: red;
}
$('#myid').toggleClass('special', 3000);
```

Adding delay()

```
$('#myid')
    .effect('pulsate')
    .delay(1000)
    .slideUp()
    .delay(3000)
    .show('fast');
```

Effect complete event

```
$("#myid").effect('puff', [options], [duration], [function]);
```

- All effects can take a fourth optional callback parameter that is called when the animation ends
- the callback can use the this keyword as usual to address the element the effect was attached to

```
$('#myid').effect('clip', {}, 'default', function() {
      alert('finished');
});
```

AJAX is acronym of Asynchronous JavaScript and XML

- AJAX == technique for asynchronously loading (in the background) of dynamic Web content and data from the Web server into a HTML page
- Allows dynamically changing the DOM (client-side) in Web applications

Two styles of AJAX

- Partial page rendering
  - Load an HTML fragment and display it in a **`<div>`**
- JSON service with client-side rendering
  - Loading a JSON object and render it at the client-side with JS / jQuery

Example

```
$.get( "www.yoursite.com/api" ).done(function() {
  console.log('this will run if the $.get succeeds');
})
.fail(function() {
  console.log('this will run if the $.get fails');
});
```

See Posted Examples

Summary

jQuery – the most popular client-side JS library

Select DOM elements with jQuery

- `$([selector])`

DOM Traversal:

- `$([selector]).next()` / `parent()`

Altering the DOM:

- `$([selector]).html(…)` / `append(…)`

jQuery Events

- `$([selector]).on([event], [callback]);`

jQuery tutorials

- Code School:

  http://www.codeschool.com/courses/jquery-air-first-flight

- W3C School:

  http://www.w3schools.com/jquery/