jQuery and Ajax

Lecture 14
Chapter 12 in HTML Text
Chapter 19 in JavaScript Text

Refresh - all or nothing?

- The basic HTTP/HTML dynamic requires the browser to issue a full request and to receive a full (new) HTML page from the server to refresh content
- As “pages” became “applications”, this heavy-weight round trip mechanism became inadequate

Microsoft to the **rescue**?

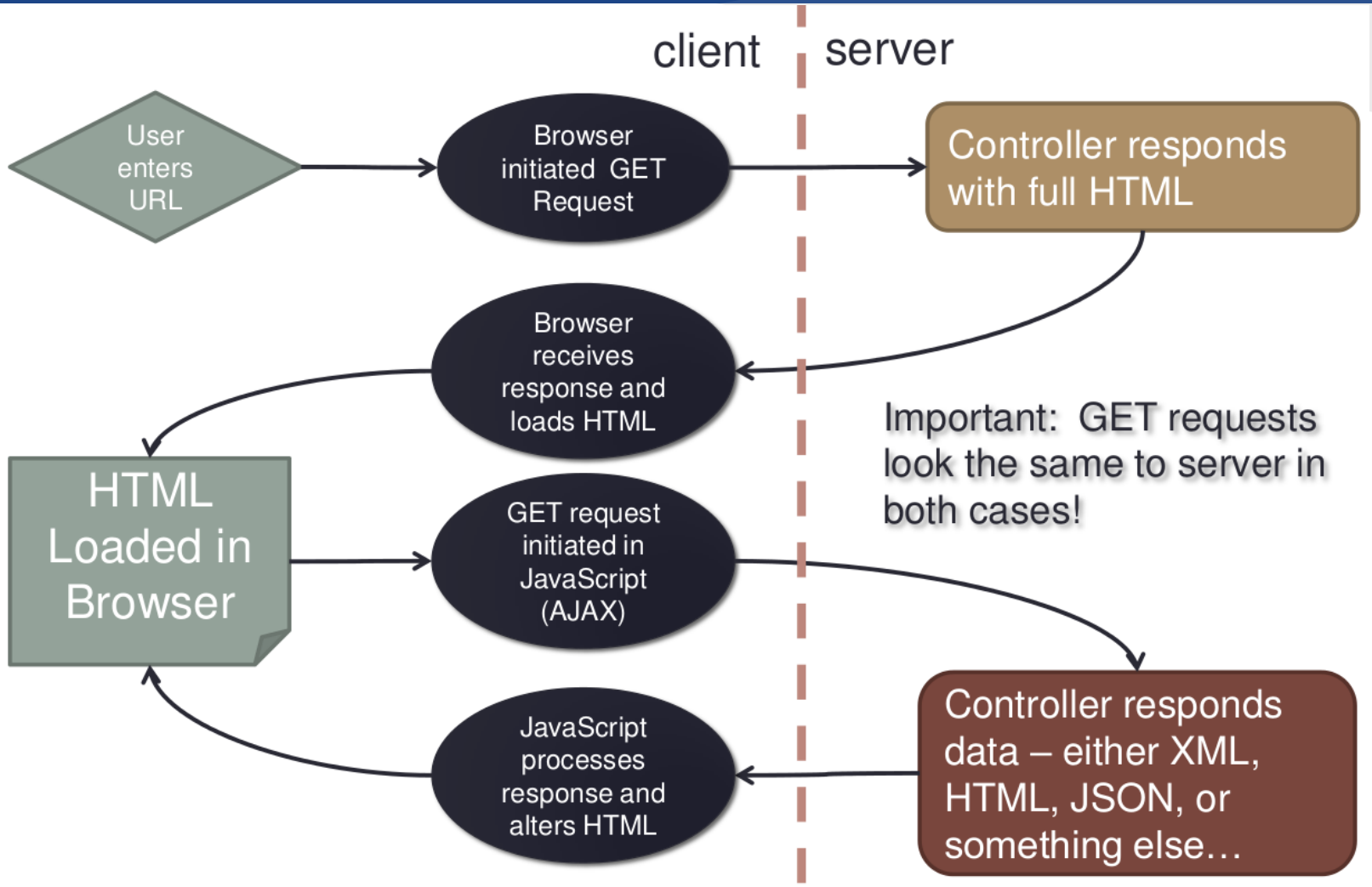
ActiveX, JavaScript, and Xml

- For pages that just wanted to grab a bit of data from the server, Internet Explorer (in late 90's) shipped an ActiveX object that was accessible via JavaScript

```
connection = new ActiveXObject("Microsoft.XMLHTTP");
```

- The object could issue an **HTTP** request to a server and receive **XML** as the response type
- The request / response cycle was done **asynchronously**, meaning the user could continue to interact with the page
  - AJAX: **A**synchronous **J**avaScript **a**nd **X**ML

AJAX Workflow



Good idea?

- The idea was fantastic... but ActiveX wasn't...
  - Security problems
  - Internet Explorer only.

- However, people realized how useful this was. Other browsers follow suit - providing built-in JavaScript object

```
Connection = new XMLHttpRequest();
```

AJAX in practice

- AJAX is a bit tricky to deal with in its native form
  - Browser incompatibilities
  - Lots of tedious coding

- Luckily, jQuery provides a very convenient way of making AJAX calls to the server
  - $.ajax,
  - $.get,
  - $.post,
  - $.getJSON

XML - in brief

- XML stands for eXtensible Markup Language
- It looks like HTML, but the element names are unbounded.
- Its well suited for representing structured data

XML has many uses - ranging from very simple to way too complex... it tends to be very simple with AJAX.

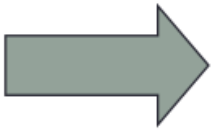
XML example

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<professor id="123">
  <name>Scott Frees</name>
  <contactInfo>
    <office>ASB406</office>
    <email>sfrees@ramapo.edu</email>
    <extension>#7726</extension>
  </contactInfo>
  <courses>
    <course crn="234342">
      <name>CMPS364</name>
      <location>ASB429</location>
      <time>11:30am</time>
    </course>
    <course crn="14342">
      <name>CMPS369</name>
      <location>ASB429</location>
      <time>9:45am</time>
    </course>
  </courses>
</professor>
```

XML or JSON?

- XML used to be the dominant form of communication for AJAX, but no longer…
  - XML is relatively bloated
  - Lots of characters are used for structure
  - Requires more processing client-side as well
- JSON: JavaScript Object Notation is a more compact alternative and is easier to work with!

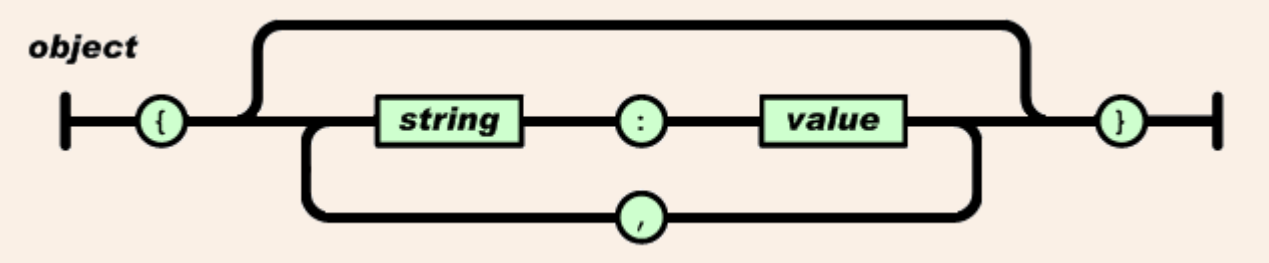
```
<!doctype…>
<message>
        correct
</message>
```



```
{ "message" : "correct" }
```

JSON - details

We've already seen a lot of JSON - lets see the formal language rules...

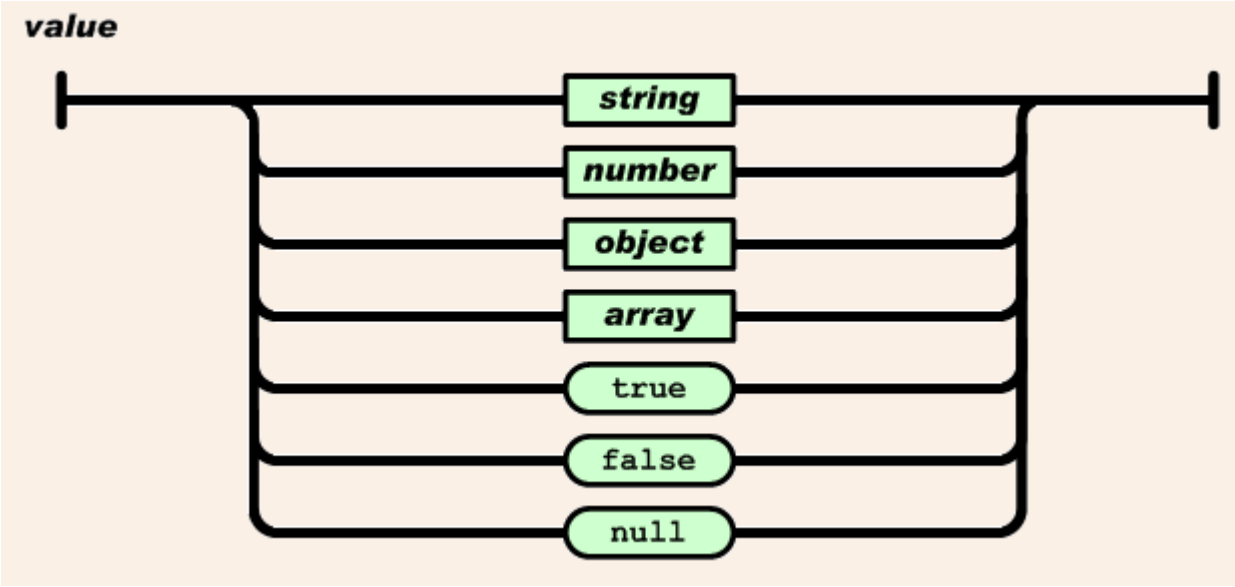


- When written in JavaScript, you'll typically see the "string" part without quotes.
- When written as an data exchange format (text), the string (and value, if a string) must **always** be quoted

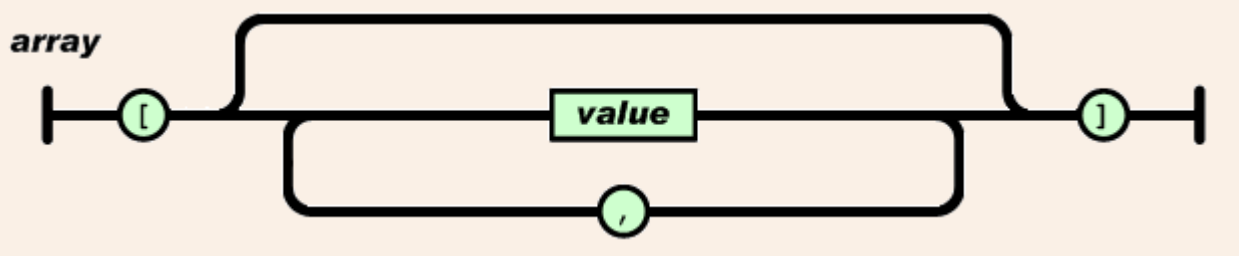
http://www.json.org/

JSON Values

- A **value** can be any number of types. Again, when writing object literals in JavaScript, you use the conventions of the type
  - { id : 2}
- When you are using JSON to exchange data (text), everything is quoted
  - {“id” : “2”}



JSON Arrays



No surprises here - basically the same as an array literal in normal JavaScript

Data Exchange?

- **Make no mistake** - JSON is not tied to JavaScript
- Both XML and JSON enjoy wide adoption among nearly every major programming language on the planet.
- This means you can use XML or JSON to pass data between programs written in any language
- It also means you will rarely ever need to parse or write it "by hand" - there are almost always built-in functionality or external libraries available to handle this for you.

This is an area where Node really shines - your JavaScript objects on the server easily convert to JSON - there is no "serialization" step.

Working with AJAX

Any AJAX Call can contain 4 things:

1. Request Type (Get, Post)
2. URL to send AJAX request to
3. Parameters (Query String)
4. Callback Function (when data returns)

```
jQuery.ajax(
    {
1       type:"get",
2       url: "test.html",
3       data: {foo:"bar1"},
4       success: function(){$("#myelement").addClass("done"); }
    }
);
```

Note the ajax call is just taking object

Other jQuery functions

- The $.ajax method is highly configurable, but for common cases, there are more convenient functions.
- If you want to retrieve partial HTML from the server to update your page - use $.load
  - `$("#stats").load("status_report")`
  - Note - the server shouldn't return full HTML - only something that would go into the #stats element
- To grab JSON quickly use $.getJSON

```
$.getJSON("data.json", {param: value}, function (data) {
        console.log(data.myproperty);
});
```

Newer strategies

```
var jax = $.getJSON( "example.json", {param1:value, param2:value});

jax.done( function() {
            console.log( "success" );
         });

jax.fail(  function() {
            console.log( "error" );
         });

jax.always(function() {
            console.log( "complete" );
         });

jax.complete(function() {
            console.log( "second complete" );
         });
```

The ajax methods return a jQuery XMLHttpRequest (jqXHR).

The jqXHR allows you to register event handlers to specific events along the lifecycle of the ajax request

Most like this style better

Back to Guessing Game!

- We can offload checking the guess against a number held in the session (rather than just by the client)
- Set a session variable on the first page access
- Serve up last week's HTML/CSS/Javascript through the normal HTML page
- Ask the server to report on the result of a guess using AJAX
- This will require some changes to our Node.js server code from our original implementation
- We still have a SPA, but its using the server to do the "heavy lifting"