




FULL SAIL
UNIVERSITY

scripting for web applications



2

jQuery AJAX and templating

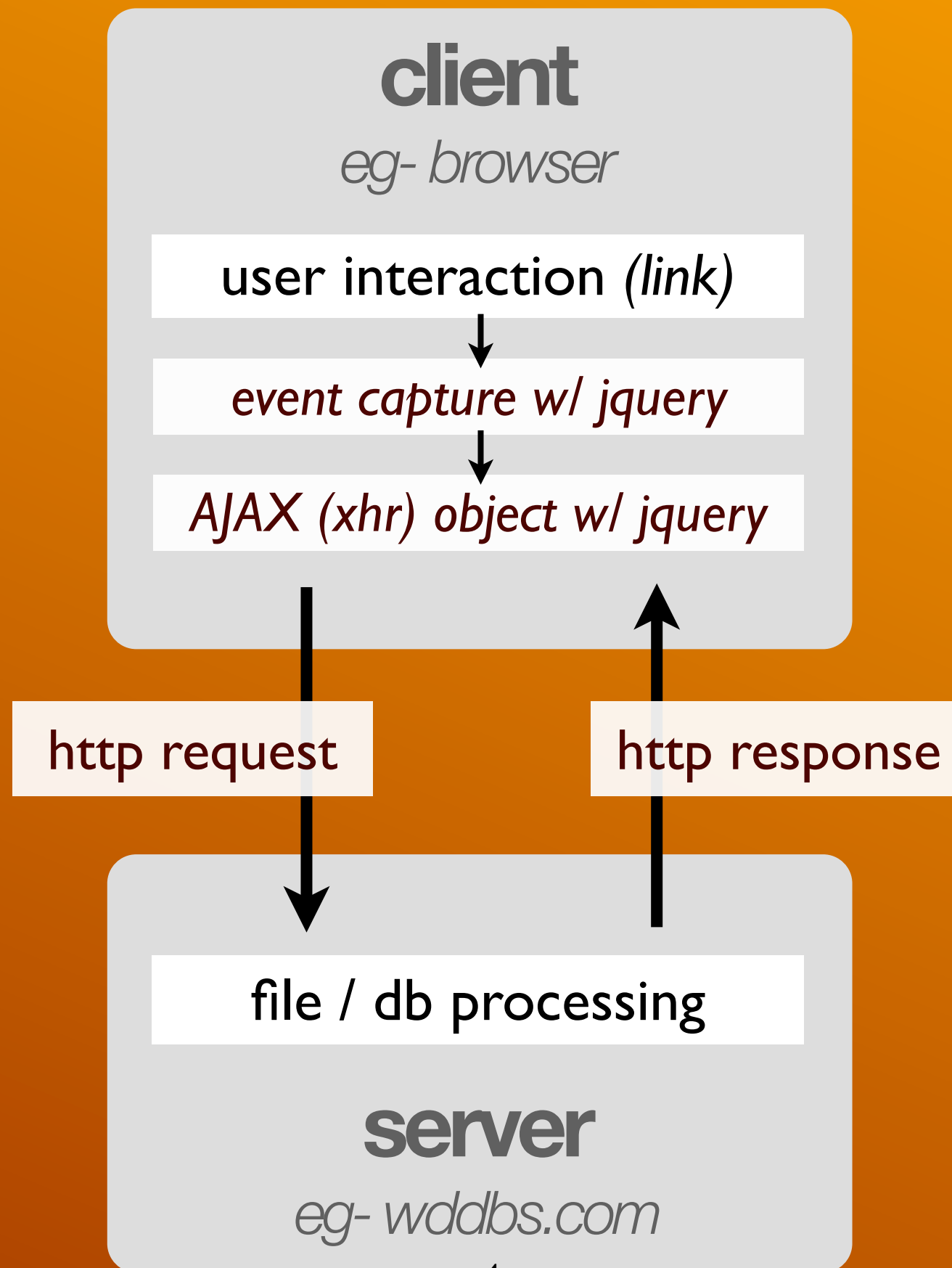


SFW-2 DUE Dates

Scripting for Web Applications

Item	Due Dates
Branding / Logo	07/09/13 After Lab on the First Day
Creative Brief - Finished Document	07/16/13 Before Lecture 4
Site Prototype (<i>html/css</i>)	07/20/13 - After Last Lab of the 2nd Week
Development Milestone (<i>javascript</i>)	07/27/13 - Due End of Lab 7
Inclusion of 5 media center items	08/03/13 - Last Day of Class After Lab
Aesthetics & Usability (<i>finished site</i>)	08/03/13 - Last Day of Class After Lab
Functionality (<i>finished site</i>)	08/03/13 - Last Day of Class After Lab
Professionalism	The duration of the course
Class Participation	The duration of the course

ajax http request



http request types

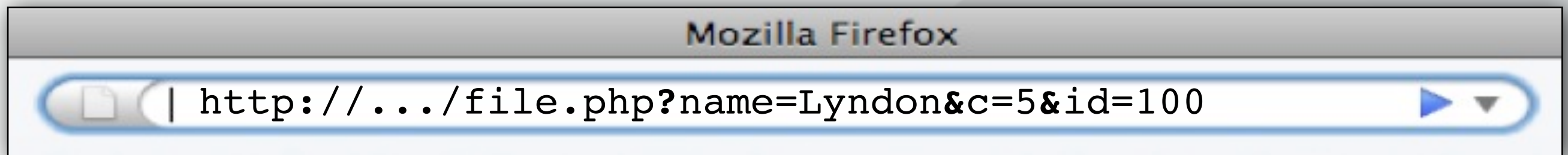
- ▶ Most HTTP Requests use these 2 types:
 - ▶ **GET** : meant for **retrieving resources** from the server (**GETTER**)
 - ▶ *can still be used to send data to server*
 - ▶ *faster request type*
 - ▶ *can be cached by browser, and be bookmarked (dependent on the site being used, i.e Facebook does not cache often)*
 - ▶ *best for repeatable requests and pulling data*
 - ▶ **POST** : meant for **updating data** on a server resource (**changing server state**)
 - ▶ *can be used to send data to server*
 - ▶ *browsers will NOT allow cache this data (security reasons)*
 - ▶ *needed if you're sending large data to the server*
 - ▶ *best for unique data calls (like logging in, or posting a comment)*

sending data

- ▶ Let's compare sending the following data to the server:

```
{name: "Lyndon", c: 5, id:100}
```

- ▶ **GET** requests send data using the URL *(limited by URL length restrictions)*



- ▶ **POST** requests send data through the header *(no length restrictions)*

```
POST http://.....  
User-Agent: Mozilla/3.5, Macintosh  
name: Lyndon, c: 5, id: 100
```

jQuery setting up AJAX requests

AJAX's requirements

- ▶ The core info needed for AJAX:
 1. An HTTP method for the request (*GET or POST*)
 2. URL to a server-side resource (*relative path usually*)
 3. Attach any data to the request (*for the server to process*)
 4. Specify a ***callback function*** (jQuery's callback is called “success”)

AJAX methods

shortcut methods

<code>\$.get(url, data, fn)</code>	Auto-uses GET type, and determines data type
<code>\$.post(url, data, fn)</code>	Auto-uses POST type, and determines data type
<code>\$.getJSON(url, data, fn)</code>	Auto-uses GET type, and expects JSON response
<code>\$.getScript(url, data, fn)</code>	Auto-uses GET type, and injects a <code><script></code> from <i>url</i>

core method

<code>\$.ajax(options)</code>	<i>options</i> : object of AJAX options
---------------------------------	---

.ajax options object

option	type	description
url	string	Request url address (<i>local or remote</i>)
type	string	HTTP method: “ post ” or “ get ”
data	object	Object with data to send to the server resource
dataType	string	Expected return data: <i>xml, html, text, json, script, or jsonp</i>
timeout	number	Milliseconds to wait before cancel (<i>evokes error callback</i>) Default: Infinity
cache	boolean	Default true... use <i>false</i> to not cache the request
global	boolean	Default true... use <i>false</i> to ignore global AJAX callbacks
error	function	function evokes on error or timeout
success	function	function evokes on successful ajax (<i>with response data argument</i>)
complete	function	function always evokes after return

bare minimum AJAX example

```
$.ajax({  
  url: "xhr/myfile.php",  
  type: "get",  
  dataType: "json",  
  success: function(result){  
    console.log(result);  
  }  
});
```

- ▶ *Best practice: use a **xhr** folder for all AJAX-related server files*
- ▶ *POST or GET?*

- jQuery .ajaxSetup()

- ▶ Allows you to create global defaults for ALL future ajax calls.

\$.ajaxSetup(*options*)

options: object: Same as .ajax()

```
$.ajaxSetup({  
  timeout: 6000,  
  ifModified: true,  
  error: function(){}  
});
```

jQuery

Project: Login, placeholder, errors

Live Demo

Event Delegation

```
$(window).on( target, type, function )
```

Binds the event listener to the global *window* object, and delegates to the *target*

```
$(window).on( '#nav a', 'click', function(e){}) ;
```


- ▶ Additionally, the delegated **on** events cannot be removed normally, will need use **.off**

`$(window).off(target, type)`

Unbinds all instances of the specified delegated “.on” event type for the *target selector*.

```
$(window).off( '#nav a', 'click' );
```

lecture resumes in 15 min

jQuery AJAX activity

studentvfiler / lecture activities / day 5 / ajaxsimple

before lecture resumes...

download these files from “Recommended Plugins” on studentvfiler

jsrender.min.js

Lecture Activity

1. Turn on MAMP, and copy the activity folder into your “htdocs”
3. Do testing from <http://localhost:8888/ajax-simple/>
4. Make a **\$.ajax** call to “xhr/list.php”, *there is an example below...*
5. Console log the response data to see how the json object is structured
6. Make a for-loop for the “languages” array in the data
7. In the loop, create a html string as shown in the html file, using the json data.

minimum required options
as example

```
$.ajax({  
    url: "xhr/myfile.php",  
    type: "get",  
    dataType: "json",  
    success: function(result){  
        console.log(result);  
    }  
});
```

jquery templating

lecture resumes in 15 min

*html template
stored in <script>*

```
<script type="text/tmpl" id="thing-template">
  <div class="thing">
    <h3>{{=title}}</h3>
    <p>{{=content}}</p>
  </div>
</script>
```

example json data

```
[
  {
    "title": "First Thing",
    "content": "I get to be first!"
  },
  {
    "title": "Second Thing",
    "content": "Drat bastard, first."
  }
]
```

template should be an HTML string or jQ object

save a template

`$.template("name", template)`

make a template from a **string**

use a template

`$.tmpl(data, "name")`

use a **saved** template by name



FULL SAIL
UNIVERSITY.

Lab 5 *(due end of lab TODAY!!!)*

lab begins in 1 hr

❖ **Next Milestone: Project Prototype**

- ❖ **ALL** html/css markup completed, *no javascript* in deliverable
- ❖ filler content (**NO** *lorem ipsum*) must be used inside html to test your design
- ❖ **ALL** components of your app as HTML (i.e. landing.html, addproject.html)
- ❖ only 1 stylesheet file for the entire project
- ❖ *each html page should **look** like it would when live.*