FULL SAIL
UNIVERSITY

programming for web applications

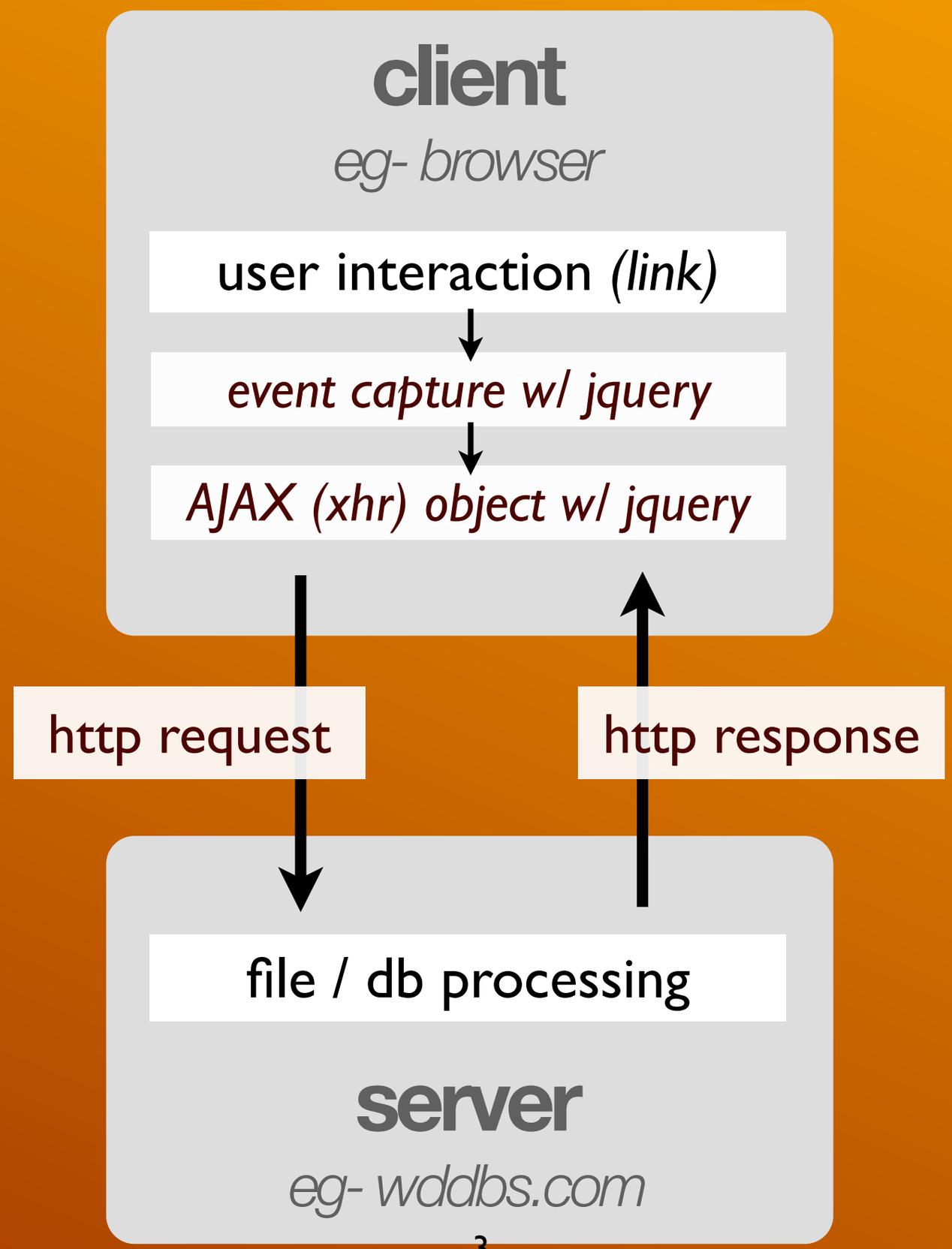
2

jQuery AJAX and templating

due.**Dates**

Item	Due Dates
Branding / Logo	11/25/13 - After Lab on the First Day
Project Pitch	12/02/13 - Before Lecture 2
Creative Brief - Finished Document	12/4/13 - Before Lecture 3
Site Prototype (html/css)	12/09/13 - Before Lecture 5
Development Milestone (javascript)	12/13/13 - Due End of Lab 7
Inclusion of 5 media center items	12/20/13 - Last Day of Class After Lab
Aesthetics & Usability (finished site)	12/20/13 - Last Day of Class After Lab
Functionality (finished site)	12/20/13 - 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***

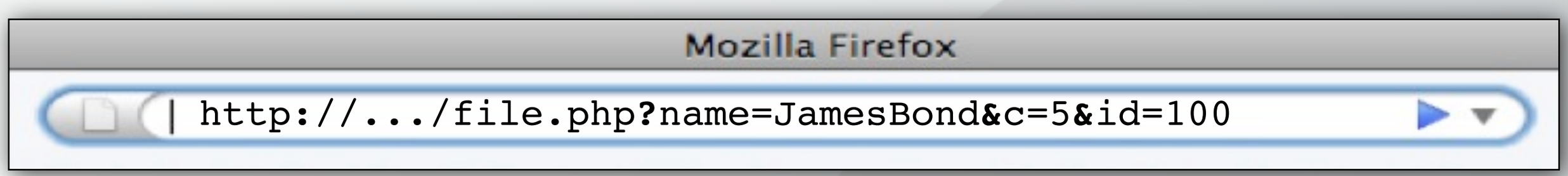
- **POST :** meant for **updating data** on a server resource **(changing server state)**
  - *can be used to* ***send data to server***
  - *browsers will* ***NOT allow cache*** *this data (security reasons)*
  - *needed if you're* ***sending large data*** *to the server*
  - ***best for unique data*** *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: JamesBond, c: 5, id: 100
```

setting up AJAX requests

ajax.Requirements

Most HTTP Requests use these 2 types:

- 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

$.get(*url, data, fn*)	Auto-uses GET type, and determines data type
$.post(*url, data, fn*)	Auto-uses POST type, and determines data type

$.getJSON(*url, data, fn*)	Auto-uses GET type, and expects JSON response
$.getScript(*url, data, fn*)	Auto-uses GET type, and injects a <script> from *url*

core method

$.ajax(*options*)	*options:* object of AJAX options

ajax.**Options** object

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

ajax.Example (bare minimum)

```
$.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?*

ajax.**Setup**

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(){}
});
```

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){});
```

event.**Delegation**

- Additionally, the delegated **on** events cannot be removed normally, will need to 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.**Activity** (ajax)

in FSO's Reference Tab: Course Material - Lecture Activities 'Activity for Goal5-1'

1. Turn on MAMP, and copy the activity folder into your "htdocs"
2. Do testing from http://localhost:8888/courseActivity5_1/
3. Make a **$.ajax** call to "xhr/list.php", *there is an example below...*
4. Console log the response data to see how the json object is structured
5. Make a for-loop for the "languages" array in the data
6. 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

jquery.Templating

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."
 }]
```

jquery.**Templating**

template should be an HTML string or jQuery 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

assignment.**Goal5** (see schedule for due dates)

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.*
- Turn into GitHub Repo: **lastname_firstname_prototype.zip**