HACK YALE

< ADVANCED JAVASCRIPT />

WWW.HACKYALE.COM

HACK YALE

< ADVANCED JAVASCRIPT />

DAY 2

DIVING DEEPER INTO JAVASCRIPT

WELCOME BACK!

The Agenda

- Feedback
- Homework review
- Objects
- Javascript Events
- Plugins

FEEDBACK

http://goo.gl/LMY91x

HACK YALE

HOMEWORK REVIEW

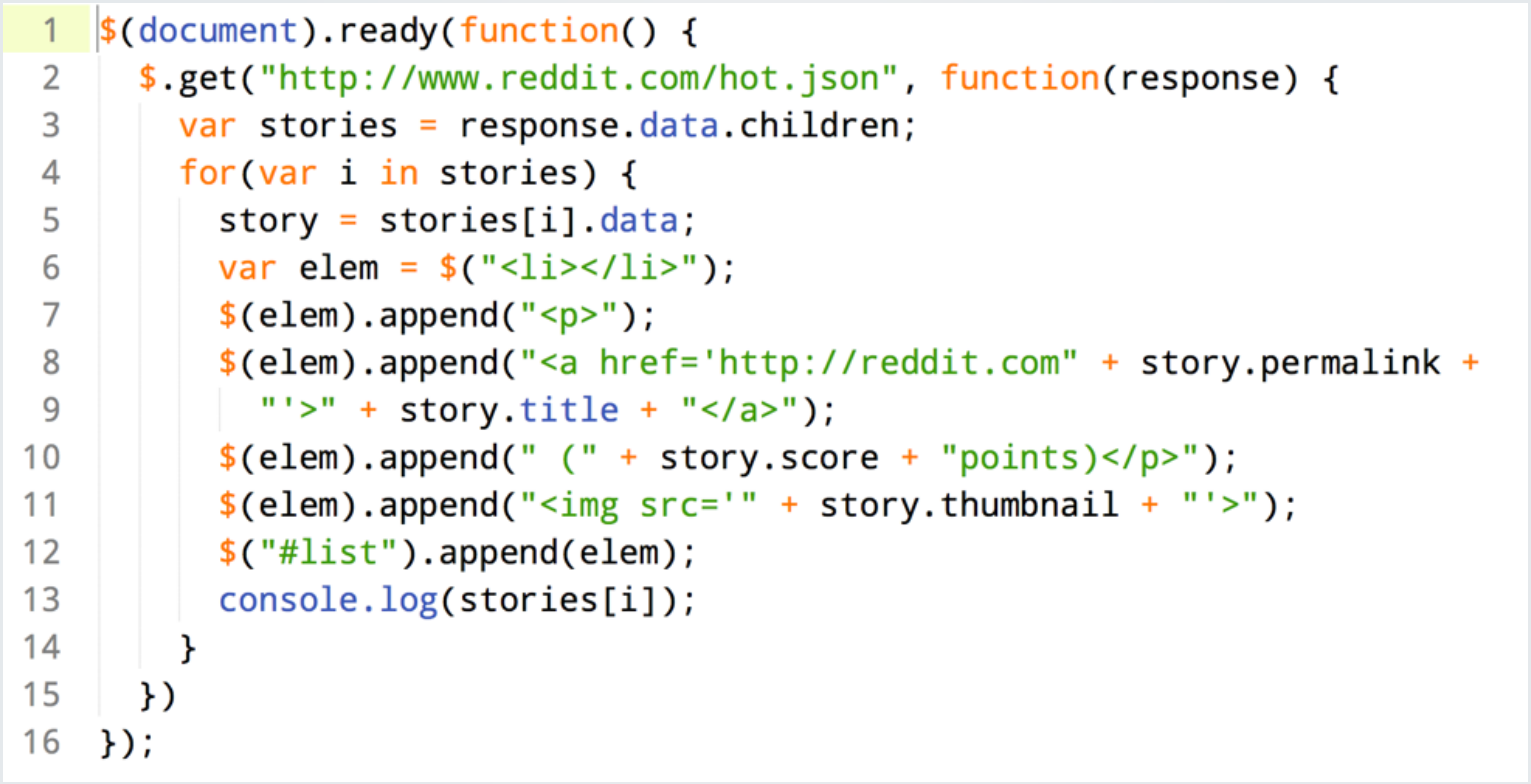
HACK YALE

COMPARE NOTES

CHECK IN WITH THE PEOPLE NEAR YOU—

WHAT DID YOU DO SIMILARLY?
WHAT WAS DIFFERENT?

MY SOLUTION



```javascript
$(document).ready(function() {
  $.get("http://www.reddit.com/hot.json", function(response) {
    var stories = response.data.children;
    for(var i in stories) {
      story = stories[i].data;
      var elem = $("<li></li>");
      $(elem).append("<p>");
      $(elem).append("<a href='http://reddit.com" + story.permalink +
        "'>" + story.title + "</a>");
      $(elem).append(" (" + story.score + "points)</p>");
      $(elem).append("<img src='" + story.thumbnail + "'>");
      $("#list").append(elem);
      console.log(stories[i]);
    }
  })
});
```

JAVASCRIPT OBJECTS

HACK YALE

WHAT ARE OBJECTS, ANYHOW?

One of the six (6) Javascript data types

A collection of properties and methods

- Properties: variables attached to an object
- Methods: functions attached to an object

OBJECTS ARE WRITTEN IN JSON

Javascript Object Notation

- Key-value pairs
  - Keys are usually strings, but can be "anything"
    - Arrays are just objects where keys are numbers
    - Can't be null, undefined, and some other values
  - Values can be anything, including other objects

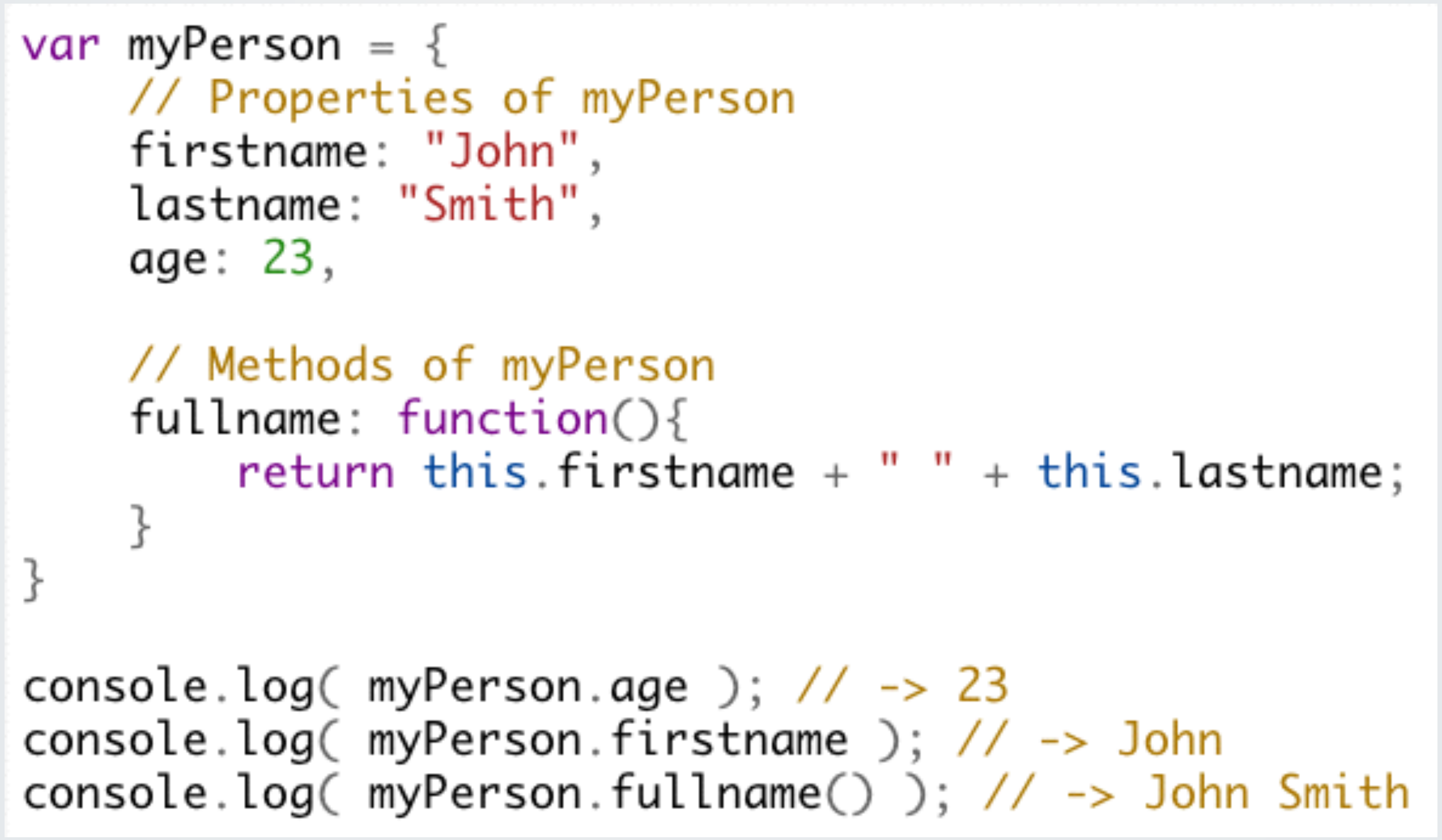
DOT NOTATION

In Javascript, we can access *properties* and *methods* of objects using dot notation

- SYNTAX: `myObject.my_prop` or `myObject.my_method()`
- We have already seen this! Remember `console.log`?
  - **console** is an *object*
  - **log** is a *method* of the console object

SYNTAX IN DEPTH

To create a singular instance of an object:



```javascript
var myPerson = {
    // Properties of myPerson
    firstname: "John",
    lastname: "Smith",
    age: 23,

    // Methods of myPerson
    fullname: function(){
        return this.firstname + " " + this.lastname;
    }
}

console.log( myPerson.age ); // -> 23
console.log( myPerson.firstname ); // -> John
console.log( myPerson.fullname() ); // -> John Smith
```

BRACKET NOTATION

We can also access *properties* and *methods* of objects using bracket notation

- SYNTAX: `myObject["my_prop"]` or `myObject["my_method"]()`
- We prefer dot syntax because it is easier to type and read
- Sometimes, however, bracket notation will be more convenient, so keep it in mind

PRACTICE

- Write an object to represent yourself :)
- Write an object with the same keys, but fill in the values with your neighbor's info
  - Any conflicts?
- Write a function `setName(name)` that sets the 'name' property of your object to the given name
  - `var setName = function(name) { … };`

JAVASCRIPT EVENTS

HACK
UNIVERSITY

INTRO TO EVENTS

- JavaScript is an event-driven language
- Events are fired in response to user actions and normal browser functions
- Events are asynchronous (KEY BENEFIT)
- Events trigger functions to process the event

HACK UNIVERSITY

PLAY-BY-PLAY

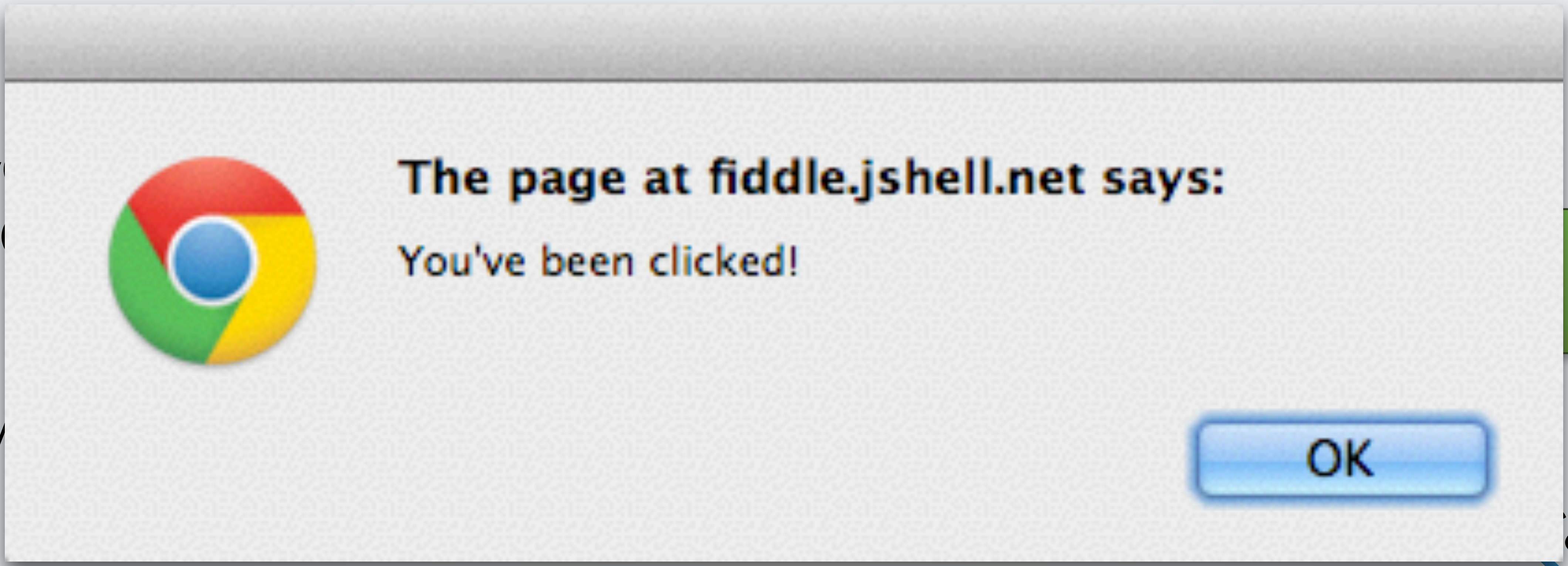
- STEP 1: Register ***handlers*** (functions) for specific event types occurring with specific DOM nodes
- STEP 2: User interacts with your page (e.g. clicking a button), and an event corresponding to the type of action is fired
- STEP 3: The handler for that event/node pairing is called and the code executes

HACK UNIVERSITY

EVENT EXAMPLE

```
var my
   alert
}

$("#my
```


The page at fiddle.jshell.net says:

You've been clicked!

on myButton

Browser fires click event
and sees attached handler

EVENT TYPES: MOUSE

- click *fires when user clicks on an element with any mouse button*
- dblclick *fires when user double clicks on an element*
- mousedown *fires when user's mouse button is depressed*
- mouseup *fires when user's mouse button is released after it was depressed on the element*
- mouseenter *fires when user's cursor enters an element*
- mouseleave *fires when user's cursor leaves an element*
- mousemove *fires when user's cursor moves in an element*

HACK
UNIVERSITY

EVENT TYPES: KEYBOARD & FORMS

- keydown *fires when user presses a key. continues to fire while key is held down*
- keypress *fires when a character is going to be inserted into a text input*
- keyup *fires when user releases a key (after default action has been performed)*
- focus *fires when a form element obtains focus (i.e. selected)*
- blur *fires when a form element loses focus*
- change *fires when a form element's value is changed by the user*
- submit *fires when a form is going to be submitted*

EVENT TYPES: BROWSER

- load *fires when the page begins to load (before all HTML has loaded)*
- unload *fires when the page is about to be changed or closed*
- ready *fires when all HTML has been loaded*
- resize *fires when the user resizes the browser window*
- scroll *fires when an element (or the whole page) is scrolled*

HACK UNIVERSITY

EVENT HANDLERS

RESPONDING TO EVENTS

EVENT HANDLERS

Event handlers are functions that are attached to: (1) a node and (2) an event type

- Only called when the unique pairing of node and event type match the registered handler
- Handler is called each and every time the matching node/event type is triggered (until the handler is removed)

UNIQUE PROPERTIES OF (JQUERY) HANDLERS

- Receive an event object as their first parameter
- Are scoped to the context of their attached node
  - What does this mean? The keyword `this` refers to the node that the handler is attached to.

HACK UNIVERSITY

EVENT OBJECT

Used to provide more information about the event to the handler function

- Event type (event.type)
- Target element (event.target and event.currentTarget)
- Mouse position (event.pageX and event.pageY)
- Which key or mouse button was pressed (event.which)
- Timestamp (event.timeStamp)

EVENT OBJECT EXAMPLE

jQuery always passes the event object as the first argument to the handler function

- Common to call this **e** or **evt**

```
var myClickHandler = function(e){
    console.log( e.type ); // -> click
    console.log( e.which ); // -> 1 (if user used left mouse button)
}

$("#myButton").click( myClickHandler );
```

HACK UNIVERSITY

AN EXAMPLE IN ACTION

```
<html>
<body>
    <div id="myDiv">
        <p id="myPara">
            My text here that contains a
            <a href="#">link</a> and
            more text.
        </p>
    </div>
</body>
</html>
```

```
var myParaHandler = function(e){
    if( e.which === 3 ){
        // if user used right-click
        $(this).css('color', 'red');
    }
};

$("#myPara").click( myParaHandler );
```

In this case, our keyword this *doesn't help us too much because our handler only is attached to one node. We could have written:*

```
$("#myPara").css(...);
```

HACK UNIVERSITY

ANOTHER EXAMPLE

```
<html>
<body>
    <ul id="todoList">
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
    </ul>
</body>
</html>
```

```
var myItemHandler = function(e){
    if( e.which === 3 ){
        // if user used right-click
        $(this).css('color', 'red');
    }
};

// Attach handler to click event for all
// li's of #todoList
$("#todoList li").click( myItemHandler );
```

Here we see the utility of `this` *since we'd otherwise have to use* `$(e.currentTarget)`*; still simple just longer and less concise.*

HACK UNIVERSITY

USING EVENTS

FOR REAL NOW!

ATTACHING A HANDLER

```
$( selector ).on( event, handler );
```

Node(s) to which the event handler will be attached

jQuery method to attach an event handler

A string representing the event type

The function handling the triggered event

ATTACHING A HANDLER

```
$( "#myButton" ).on( 'click', buttonClicked );
```

Handler will be attached to the element with the ID *myButton*

The function *buttonClicked* will be called when the event is triggered

Handler will be listening for *click* events on *myButton*

HACK UNIVERSITY

USING AN ANONYMOUS FUNCTION

When we only need our handler function to be attached to one element (i.e. not reused), we can write the function anonymously

- An anonymous function is declared without a name and often passed directly to a function or method as it is here

```
$( "#myButton" ).on( 'click', function(e){
  alert( "I'm an anonymous function" );
});
```

PRACTICE

- Make an input in your HTML. Then make a button in your HTML that, when clicked, console.log's the text in the input field
  - Hint: give your elements IDs!
- Instead of console.log'ing the value, instead set the "name" property of the object you created earlier
  - console.log the whole object to make sure it worked

HACK UNIVERSITY

AJAX

HACK
UNIVERSITY

A QUICK HISTORY LESSON

WOOHOO!

THE INTERNET

UNLIKE RAP MUSIC AND ATHLETIC FASHION, IT USED TO BE NOT AS COOL IN THE 90'S.

HACK
UNIVERSITY

THE JAVA APPLET

- Debuted in 1995 after the release of the Java programming language.
- Allows for client-side code to asynchronously load data from the server after the page has loaded.
- E.g. Slime Volleyball

THE IFRAME

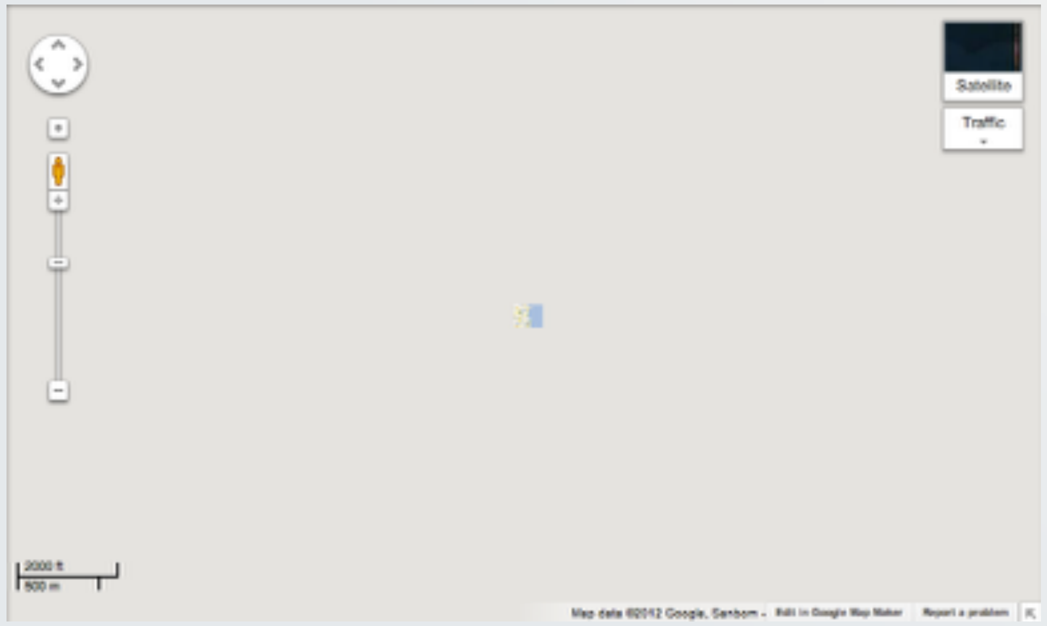
- Introduced by Internet Explorer in 1996.
- Allow you to nest another HTML document within your page.
- Allowed browsers to asynchronously load content into an iframe without reloading the entire page.

AJAX

- Introduced in Internet Explorer 5, which came out in 1999.
- Invented by Microsoft (not a typo).
- Stands for Asynchronous JavaScript and XML
  - but the XML part can easily be HTML. I like to think of it as "Asynchronous JavaScript and Data."

AJAX

- Allows pages to make asynchronous requests to urls and handle responses dynamically.
- A great use example: Google Maps



+ AJAX =

Chicago
Cicero
Lake Michigan
Satellite
Traffic

HACK UNIVERSITY

NOWADAYS

- AJAX is ubiquitous.
  - Many sites and apps only load the home page and then do everything else through AJAX, like twitter
- Any bonafide developer knows and uses AJAX

NOWADAYS

- AJAX is ubiquitous.
  - Many sites and apps only load the home page and then do everything else through AJAX, like twitter
- Any bonafide developer knows and uses AJAX

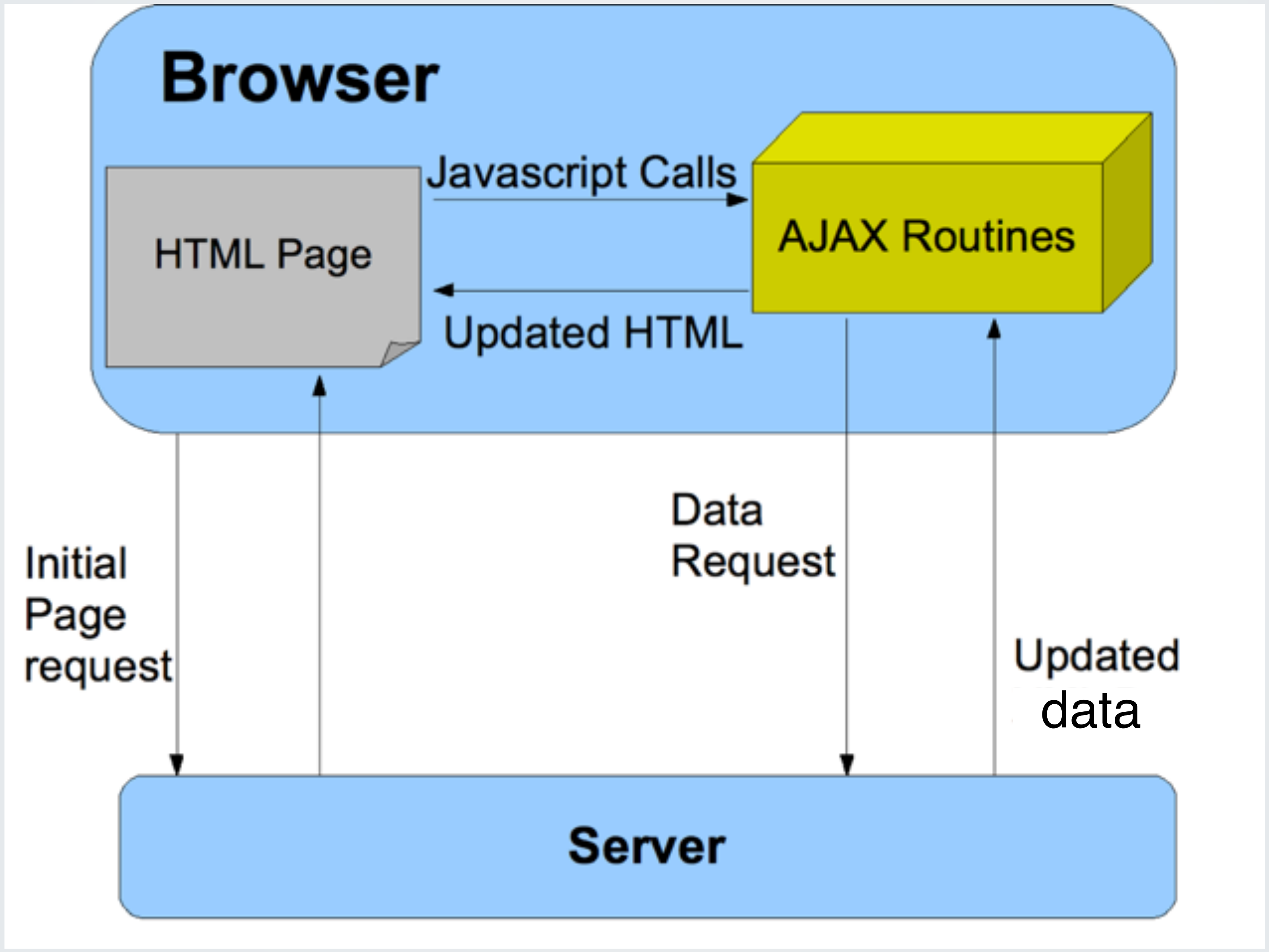
SO LEARN IT WE SHALL.

HACK UNIVERSITY

MAKING AN AJAX REQUEST

HOLLA HOLLA BACK

AJAX


Browser
HTML Page
Javascript Calls
AJAX Routines
Updated HTML
Initial Page request
Data Request
Updated data
Server

IN NATIVE JAVASCRIPT

- Incredibly difficult
- For anyone who would like to exert ten times as much effort as is necessary:
  - http://www.ibm.com/developerworks/web/library/wa-ajaxintro2/
- Moving on..

AJAX IN JQUERY

- Achievable in one easy function call
- Cross-browser compatible
- Easy to organize and read
- Includes the capacity for global ajax event handlers.

$.AJAX

$.ajax is a function used to make ajax requests with jQuery.

- Two part process
    - 1. makes an asynchronous request to a specified url
    - 2. gets a response and acts accordingly

USING $.AJAX()

$.ajax(settings);

- settings is an object telling jQuery how to execute and respond to the request.
- In the settings hash, one 'key: value' pair is
  url: request_path

THE SETTINGS HASH

JUST A BUNCH OF 'KEY: VALUE' PAIRS THAT INSTRUCT JQUERY HOW TO CARRY OUT THE AJAX REQUEST.

HACK UNIVERSITY

KEY SETTINGS

- url *equal to the url we want to query. Example: "/destination"*
- type *the HTTP method to use in the request. Example: "post"*
- data *either a string or an object (we'll use an object mostly) that tells the server information it will later use. This is similar to what happens with inputs when forms are submitted. Example: { email: "a@a.com" , password: "M9hp41cXZ7", favorite_veggie: "tomatoes" }*
- success *a callback function to be invoked if the request succeeds. It can take parameters of data, textStatus, and jqXHR. Example: function(data, textStatus, jqXHR){ for(key in data) { console.log(key + " " + data[key]); } }*

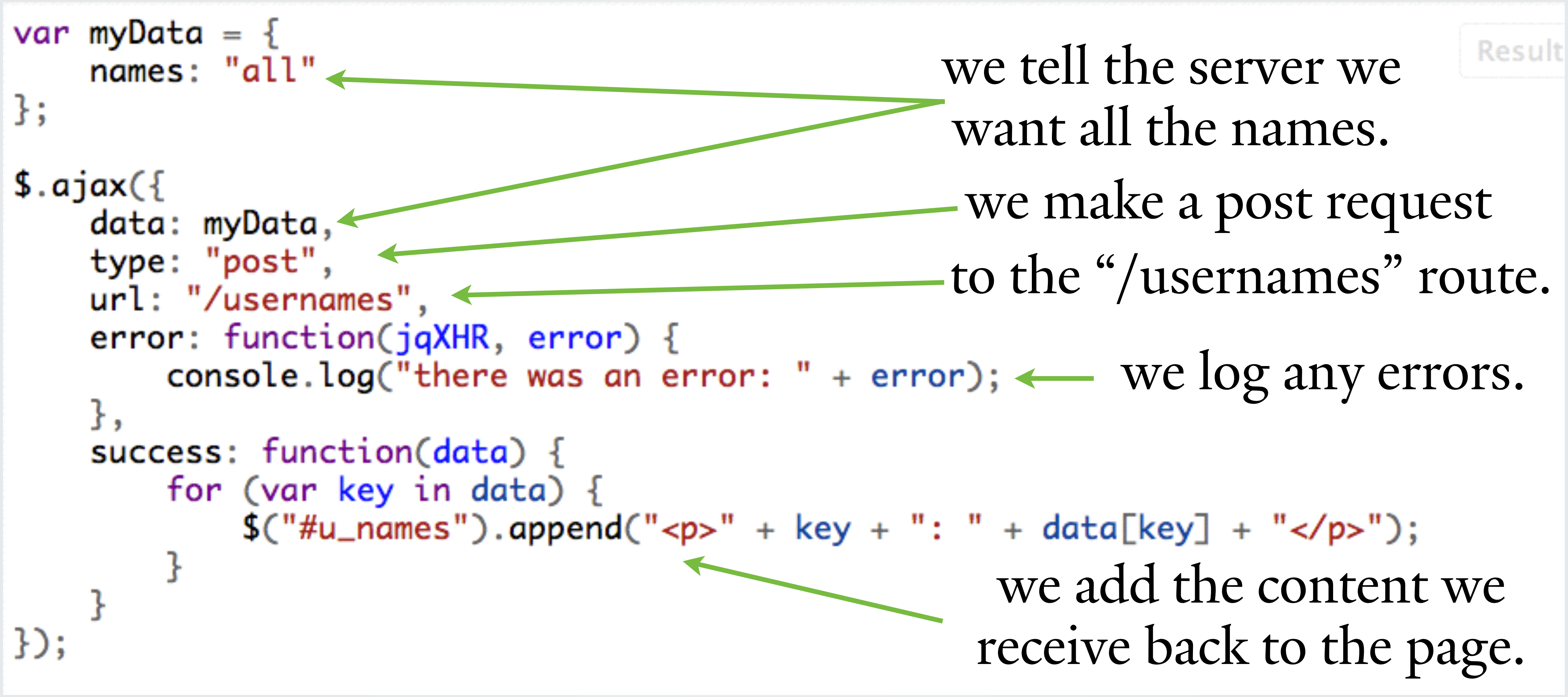
HACK UNIVERSITY

CALLBACK FUNCTIONS

- beforeSend *called before the request is sent. Takes the jqXHR object and the settings hash as parameters.*
- error *called if there is an error when the server responds. Takes the jqXHR object, a string for errorType, and an exception object.*
- dataFilter *called when the server successfully responds, before success callbacks. Takes data and dataType, and should return (potentially altered) data to be used in success.*
- success *a callback function to be invoked if the request succeeds. It can take parameters of data, textStatus, and jqXHR. Example: function(data, textStatus, jqXHR){ for(key in data) { console.log(key + " " + data[key]); } }*
- complete *a callback function to be invoked when the request is done, whether or not it succeeded. Takes a jqXHR object, and a string with either the success or error code depending on whether all went smoothly or not.*

HACK UNIVERSITY

EXAMPLE



```
var myData = {
    names: "all"
};

$.ajax({
    data: myData,
    type: "post",
    url: "/usernames",
    error: function(jqXHR, error) {
        console.log("there was an error: " + error);
    },
    success: function(data) {
        for (var key in data) {
            $("#u_names").append("<p>" + key + ": " + data[key] + "</p>");
        }
    }
});
```

we tell the server we want all the names.

we make a post request to the “/usernames” route.

we log any errors.

we add the content we receive back to the page.

HACK UNIVERSITY

JAVASCRIPT LIBRARIES

HACK UNIVERSITY

LOTS AND LOTS

- Most functionality that you want on the web, there's probably a library to do it.
- My favorites:
  - Underscore.js: convenient Javascript functions
  - NVD3 / High charts: data-driven visualization and graphs
  - Moment.js: easily handle time
  - Bootstrap: lots of easy UI features

HACK UNIVERSITY

MANY MORE

PARALLAX, MOBILE TOUCH, FILE UPLOAD, TYPEAHEAD, MAPS…

http://www.9kdesigns.com/resources/40-javascript-libraries-and-jquery-plugins

USING PLUGINS

- One of the most important skills you can have with Javascript
  - Knowing how to use *other* people's code is an irreplaceable skill
- Learning to make sense of documentation
  - => remember to document your own code!

HACK UNIVERSITY

PRACTICE

Let’s use one now!