Announcements

- Project 1 Planning Due today
- Quiz 2 on 9/15
  - Sample Honorlock Quiz Released
- REST assignment out today
  - This is an individual assignment
- Project 1 Progress Report Due on 9/14

CS3300 Introduction to Software Engineering

Lecture 05: Tools of the Trade #3

JQuery, API, AJAX, RESTful API

Dr. Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Marty Stepp, Jessica Miller, Victoria Kirst and David Karger

Contents

jQuery

- jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development

- Why jQuery?
  - Write less, do more
    - *$("p.neat").addClass("ohmy").show("slow");*
  - Performance
  - Plugins
  - Standard practice

That little snippet loops through all <p> elements with the class "neat" and then adds the class "ohmy" to it, whilst slowly showing the paragraph in an animated effect. No browser checks, no loop code, no complex animation functions, just one line of code!

DOM

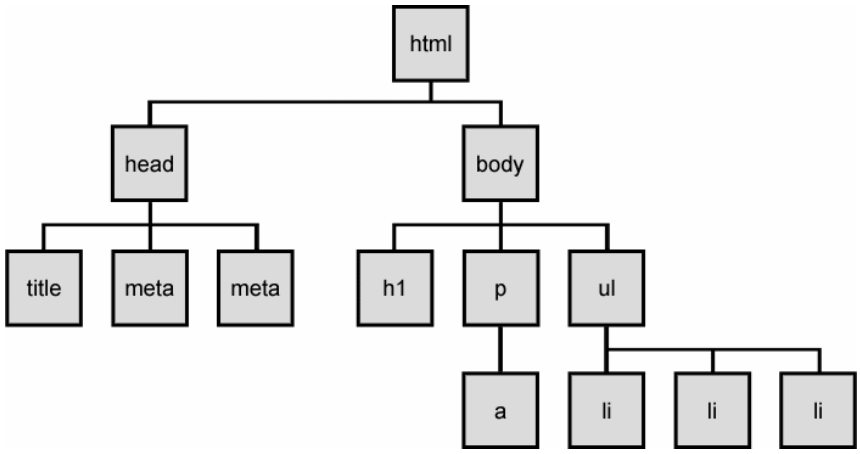
```
<html>
  <head>
    <script>
       // run this function when the document is loaded
       window.onload = function() {

         // create a couple of elements in an otherwise empty HTML page
         const heading = document.createElement("h1");
         const heading_text = document.createTextNode("Big Head!");
         heading.appendChild(heading_text);
         document.body.appendChild(heading);
       }
    </script>
  </head>
  <body>
  </body>
</html>
```

- Document Object Model (DOM) represents the structure of an HTML document in a web browser, allowing for manipulation and interaction via JavaScript.
- Represents the document as nodes and objects
- Can be used to modify the document with a scripting language like JavaScript

Webpage is a document. can be either displayed in the browser window or as the HTML source. But it is the same document in both cases. The Document Object Model (DOM) represents that same document in an object oriented way so it can be manipulated.
XML stands for **extensible markup language**. A markup language is a set of codes, or tags, that describes the text in a digital document.
Say an empty document is present. On loading, creates a new H1 element, adds text to that element, and then adds the H1 to the tree for this document:

The DOM tree



Plain JavaScript DOM manipulation vs. jQuery

Aspect	DOM method	jQuery equivalent
Availability	Native in all modern web browsers	Requires inclusion of library
Verbosity	More verbose	More concise
Cross Browser Support	May have browser specific quirks	Unified API across browsers
Functionality	Fundamental methods for page content	Additional utilities, animations, plugins
Performance	Direct manipulation can be faster	Some overload
Selection and Chaining	Requires more code for complex tasks	Powerful
Community and Plugins	No specific community, user standard JS	Rich ecosystem of plugins and vast community

Plain JavaScript DOM manipulation vs. jQuery

Description	**DOM method**	**jQuery equivalent**
returns array of descendants with the given ID, such as “header"	getElementById("id")	$("#id")
returns array of descendants with the given tag, such as "div"	getElementsByTagName("tag")	$("tag")
returns array of descendants with the given name attribute	getElementsByName("somename")	$("[name='somename']")
returns the first element that would be matched by the given CSS selector string	querySelector("selector")	$("selector")
returns an array of all elements that would be matched by the given CSS selector string	querySelectorAll("selector")	$("selector")

jQuery object

- The $ function always (even for ID selectors) returns an array-like object called a jQuery object.
- jQuery objects are wrapper objects around single or multiple DOM elements.
- You can access the actual DOM object by accessing the elements of the jQuery object

```
// false
document.getElementById("id") == $("#myid");
document.querySelectorAll("p") == $("p");
// true
document.getElementById("id") == $("#myid")[0];
document.getElementById("id") == $("#myid").get(0);
document.querySelectorAll("p")[0] == $("p")[0];
```

Using $ as a wrapper

- $ adds extra functionality to DOM elements
- passing an existing DOM object to $ will give it the jQuery upgrade

```
// convert regular DOM objects to a jQuery object
var elem = document.getElementById("myelem");
elem = $(elem);
var elems = document.querySelectorAll(".special");
elems = $(elems);
```

$(document).ready()

Before you can safely use jQuery to do anything to your page, you need to ensure that the page is in a state where it's ready to be manipulated.

Using DOM: window.onload = function() { // do stuff with the DOM }

Using jQuery: $(document).ready(function() { // do stuff with the DOM })

Using jQuery shorthand: $(function() { // do stuff with the DOM })

- **Library**: jQuery
- **Trigger Time**: Executes when the entire HTML document has been completely loaded, but before images, iframes, and other resources finish loading.
- **Use Case**: Most commonly used in jQuery scripts to ensure that the DOM is fully accessible. It's especially useful if your JavaScript is in the <head> section of your HTML document, ensuring that your script doesn't run before the HTML content is rendered.

window.onload

- **Library**: Plain JavaScript
- **Trigger Time**: Executes when the entire page is fully loaded, including all frames, objects, images, and other resources. This means it may take a bit longer before this event fires, especially if there are large resources or external content linked.
- **Use Case**: Suitable for actions that rely on a fully rendered page such as image dimensions calculations, canvas rendering based on images, or other operations that require all external content to be loaded.

Demo Time！

Use of hide/click/hover functions using jQuery

API

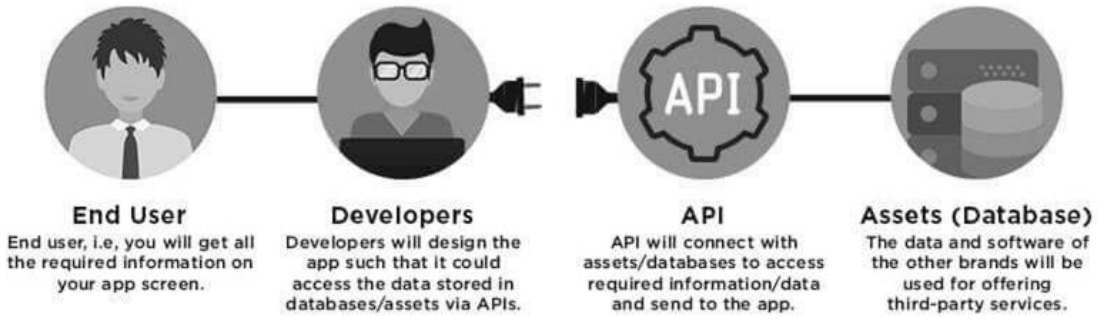
Definition
Application Programming Interface is a computing interface to a software component or a system, that defines how other components or systems can use it.

API vs. Application
Application/Library/Service: A blob of code that does things
API: The way your code can ask the application/library/service to do things. The boundary of the application/library through which requests go in and responses come out

The Browser..
…has a UI that lets you interact with the document
…has the DOM API that lets your code interact with the document

How is an API used?

You want to book a flight
Endpoints: Flight booking platform and airlines website
Most common formats of representing data are JSON and XML



End User
End user, i.e, you will get all the required information on your app screen.

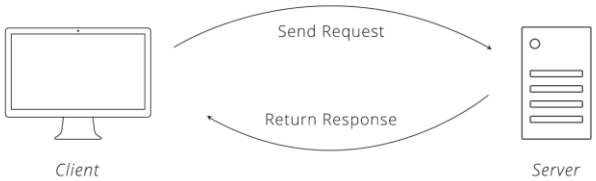
Developers
Developers will design the app such that it could access the data stored in databases/assets via APIs.

API
API will connect with assets/databases to access required information/data and send to the app.

Assets (Database)
The data and software of the other brands will be used for offering third-party services.

Protocol of the Web

- A protocol is a system of rules that define how data is exchanged
- Main protocol on web is HTTP- Hyper-Text Transfer Protocol
- Follows Client-server Protocol.



- http://example.com, the "http" tells the browser to use the rules of HTTP when talking with the server. With the ubiquity of HTTP on the web, many companies choose to adopt it as the protocol underlying their APIs.
- Communication in HTTP centers around a concept called the Request-Response Cycle.

client-server protocol, which means requests are initiated by the client, usually the Web browser.
The messages sent by the client, are called *requests* and the messages sent by the server as an answer are called *responses*.

Request-Response Parameters

Request

- **URL**: Uniform Resource Locator – Unique address
- **Method**: Type of Action –
  - GET: Retrieve a resource
  - POST: Create a resource
  - PUT: Edit/update existing resource
  - DELETE: delete a resource
- **Header**: Meta-information about a request.
  - Referer: page making the request
  - Accept: acceptable response data types (text, pdf, etc.)
  - Cookie: contains specific user data
- **Body**: data to PUT/POST/GET

Response

- **Status Code**: 200 OK, 301 redirect, 401 Unauthorized, 500 server error…
- **Header**: Meta-information about response.
  - Content-Type: of returned object (text, pdf, etc.)
  - Set-Cookie: user data to store in browser
  - Location: Instruction to look elsewhere
- **Body**: content such as web page

Authentication

Basic Authentication:

- Requires username and password; client passes in the request in an HTTP header called Authorization.; Server compares header to stored credentials, if doesn't match returns a 401 status code.
- If your connection isn't secured through transport layer security (TLS), your password may be compromised. If you don't set up multi-factor authentication (MFA), there are no additional layers of security to prevent people who now have your credentials from accessing all resources in your account whenever they want.

OAuth Authentication:

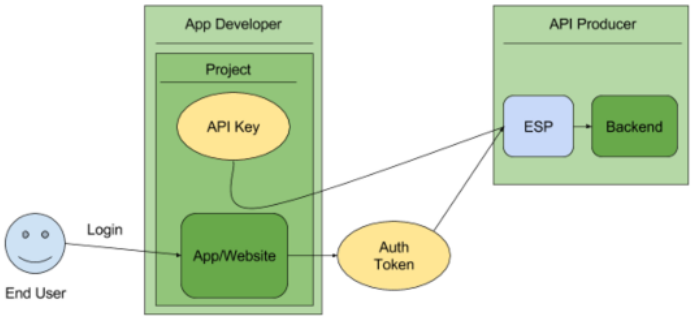
- Most widely used authentication scheme on the web. The client and server communicate back and forth to get the user a valid key.
- OAuth doesn't share password data but instead uses authorization tokens to prove an identity between consumers and service providers.
- Access token don't expire or are short lived. More details here.

Though Basic Auth is a valid authentication scheme, the fact that it uses the same username and password to access the API and manage the account is not ideal. That is like a hotel handing a guest the keys to the whole building rather than to a room.Oauth: authorizing espn to access your FB content without sharing FB password. If something happens to ESPN, your FB password is safe. Session based and token based auth

API KEY

API Key Authentication:

- Requires API to be accessed with a unique key
- Server now has the option to limit administrative functions, like changing passwords or deleting accounts.



DEMO TIME!

Use API-key to access weather data

API keys allow the Extensible Service Proxy (ESP) to reject calls from projects that haven't been granted access or enabled in the API.

AJAX

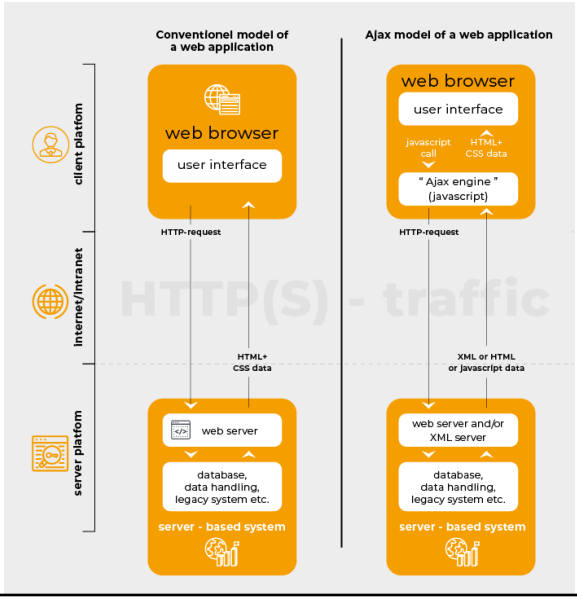
The Popular Approach

- 90s web applications ran on the server
- To take an action, user submitted a form
- To await changes, keep reloading the page
- Browser was just to display results and accept user input

The Insight

- Many actions change little of the UI
- Client can do significant computation
- Why bother involving the server?
  - Demands more server horsepower
  - Network makes slow UI
  - Reload loses your place in the page—disruptive

AJAX



Asynchronous
promises etc.
JavaScript
Computation on client
And **X**ML
Data serialization format
Now replaced by JSON
But AJAJ doesn't sound as cool

Allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. MUCH FASTER!!

All major browsers now support XMLHttpRequest (XHR) object

jQuery AJAX

- jQuery offers many Ajax-related simple & convenient methods to create GET & POST requests
- Considered a good practice to use $.ajax() method over other methods

Options:

- *url*- the request url. Only mandatory option.
- type of request, “POST” or “GET”. Default is “GET”
- *dtype-atatype- “text”, “html”, “json”*
- *async*- Accepts true/false. False makes request synchronous and will block execution of other codes until response is received
- *cache*- Use to cache response if available
- *complete*- triggers a callback function to run when request is complete, regardless of success or failure
- *data*- data (string or object type) to be sent to the server
- *error*- callback function to run when request gets an error
- *success*- callback function to be run if request is success
- *timeout*- specifies time in milliseconds to wait , otherwise consider request to be a failure

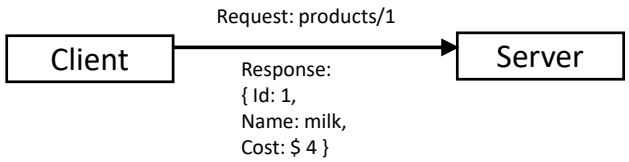
DEMO TIME !!
Access data in JSON format from web via AJAX API call

there are a few scenarios (though rare) where one might consider using synchronous requests:
1.Initialization: If an application absolutely requires certain data from the server before anything else can proceed, a synchronous request might be used. For example, if some configuration data is needed immediately upon page load.
2.Sequential Dependencies: If there's a sequence of requests where one request is wholly dependent on the result of the previous one, and you don't want to manage complex promise chains or callbacks.

RESTful APIs

- **RE**presentational **S**tate **T**ransfer
- Architectural style of web services development. Often called “Language of the Internet”
- Literally means transferring the state of representation of a resource.

- It is a set of constraints that, when applied as a whole, emphasizes performance, scalability, simplicity, modifiability, visibility, portability, and reliability.



- Follows HTTP protocol. GET, POST, DELETE,

Products is resource present in server, that client is requesting. Server transfers the representation of current state of the resource to client.
Before REST, SOAP architecture was used. Simple Object Access Protocol

RESTful APIs - Need and Principles

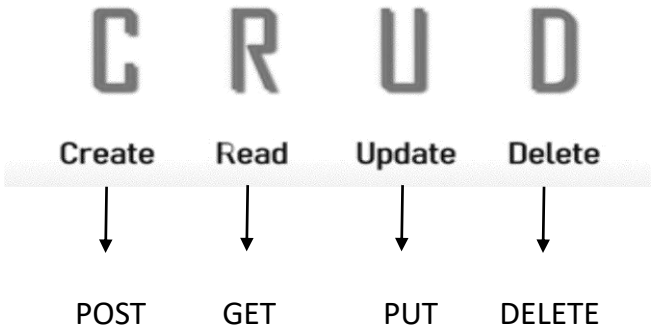
- Need:
  - Allows loosely coupled client-server applications. E.g.: server can be angular/reach, server: node js/php/.net
  - Independent of platform and languages
  - Scalability – Server does not store any client data
  - Can return data in any format as requested

- Principles:
  - Stateless: Every method call must include all the state the server needs to provide the method. Increases scalability
  - Client-Server: Separate; increases portability of interface
  - Cacheable: label the response as cacheable; can be reused by client
  - Layered System: load balancing, shared caches, enhance scalability and stability

Before REST, SOAP architecture was used. Simple Object Access Protocol

RESTful APIs- Methods

Based on Create, Read, Update and delete resources built on HTTP methods



DEMO TIME!!

Create a basic web app using REST API, VS Code IDE, Node.JS, Express

Tools used:

HTML/CSS/JS: Front-end Programming (**Client Side**)

VS Code: Programming IDE (**Client Side**)

Node.js: Runtime environment for applications in JavaScript (To implement/create **web server** in JavaScript language)

Express: Backend web application framework for Node. Js (**Web Server side**)

Front-end back-end same language