Announcements

- Project 1 Planning Due today
- Quiz 2 on 9/11
  - Honorlock based quiz. Closed book and notes.
  - Practice Honorlock Quiz Released to check your settings
- REST assignment out today
  - This is an individual assignment
- Project 1 Progress Report Due on 9/12

CS3300 Introduction to Software Engineering

Lecture 06: 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
  - DOM, jQuery Objects, wrapper (demo)
- API
  - API calls
  - Web Protocol, HTTP
  - API key (demo)
- AJAX
  - Types of AJAX calls using JQuery (demo)
- REST (demo)

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

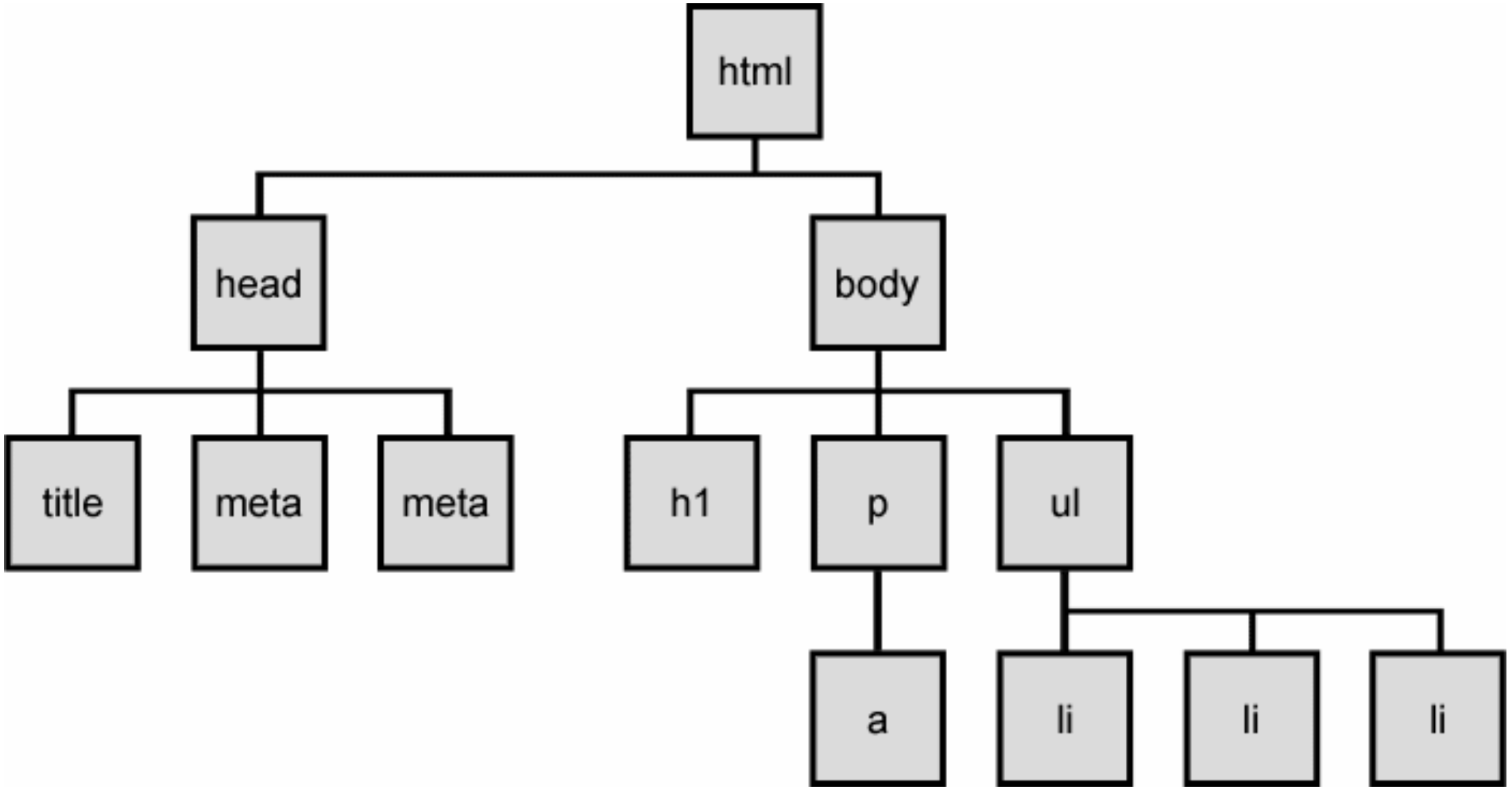
DOM

```
<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

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") == $("#id");
document.querySelectorAll("p") == $("p");
// true
document.getElementById("id") == $("#id")[0];
document.getElementById("id") == $("#id").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 })

Demo Time！

Use of hide/click/hover functions using jQuery

API

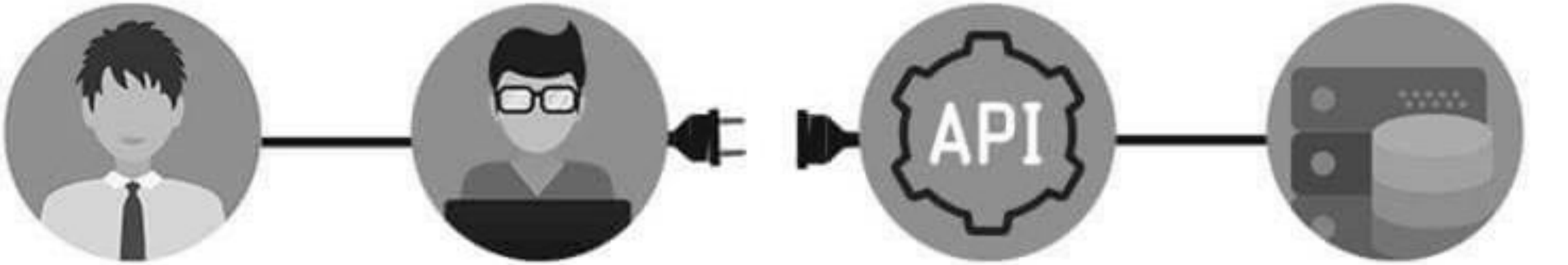
Definition
Application Programming Interface is a computing interface to a software component or a system, that defines how other components or systems can use it.

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.

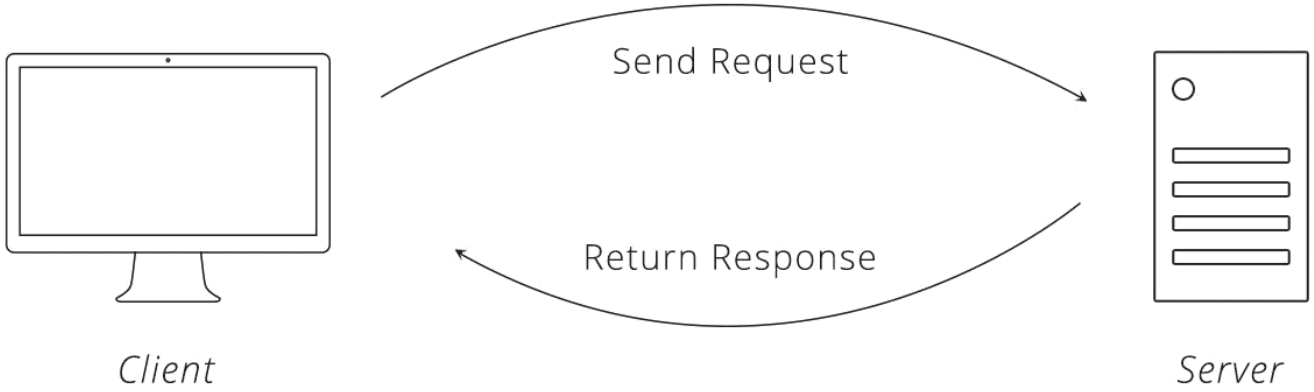
Developers
Developers will design the app such that it could access the data stored in databases/assets via APIs.

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.

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

How It Works:

- Basic Authentication requires a username and password to authenticate the client with the server.
- The client sends these credentials in the Authorization header of the HTTP request.
- Example: *Authorization: Basic base64encoded(username:password)* The credentials are encoded in Base64 format, which is a simple form of encoding but not secure because it can be easily decoded.

Security Considerations:

- The server checks the credentials sent in the header against its stored credentials.
- If the credentials do not match, the server responds with a 401 Unauthorized status code, indicating that the request requires valid authentication.
- If the connection is not secured using **Transport Layer Security (TLS)** (HTTPS), the credentials can be intercepted and stolen, leading to a potential security breach.
- Without **Multi-Factor Authentication (MFA)**, once someone has obtained your credentials, they can access all resources associated with your account.

Drawbacks:

- Basic Authentication is simple but less secure:
    - Passwords are sent with every request, which makes them more vulnerable to interception.
    - If not using HTTPS, passwords can be exposed to attackers.
    - It does not provide any additional layers of security, such as MFA.

Authentication: OAuth

How It Works:

- **OAuth (Open Authorization)** is a more secure and widely-used authentication scheme on the web.
- Instead of sharing the user's password, OAuth uses tokens to authenticate the user.
- The process involves the client and server communicating back and forth to obtain an **access token** that proves the user's identity. More details here.

Benefits:

- **No Password Sharing:** OAuth does not share the user's password directly between the client and the server. This significantly reduces the risk of passwords being compromised.
- **Authorization Tokens:** Instead of passwords, OAuth uses authorization tokens that are typically short-lived or have an expiration date, adding an extra layer of security.
- **Scopes and Permissions:** OAuth allows the client to request specific permissions or scopes of access, limiting the amount of data the client can access. For example, an app might request access only to a user's profile information, not their email or other sensitive data.

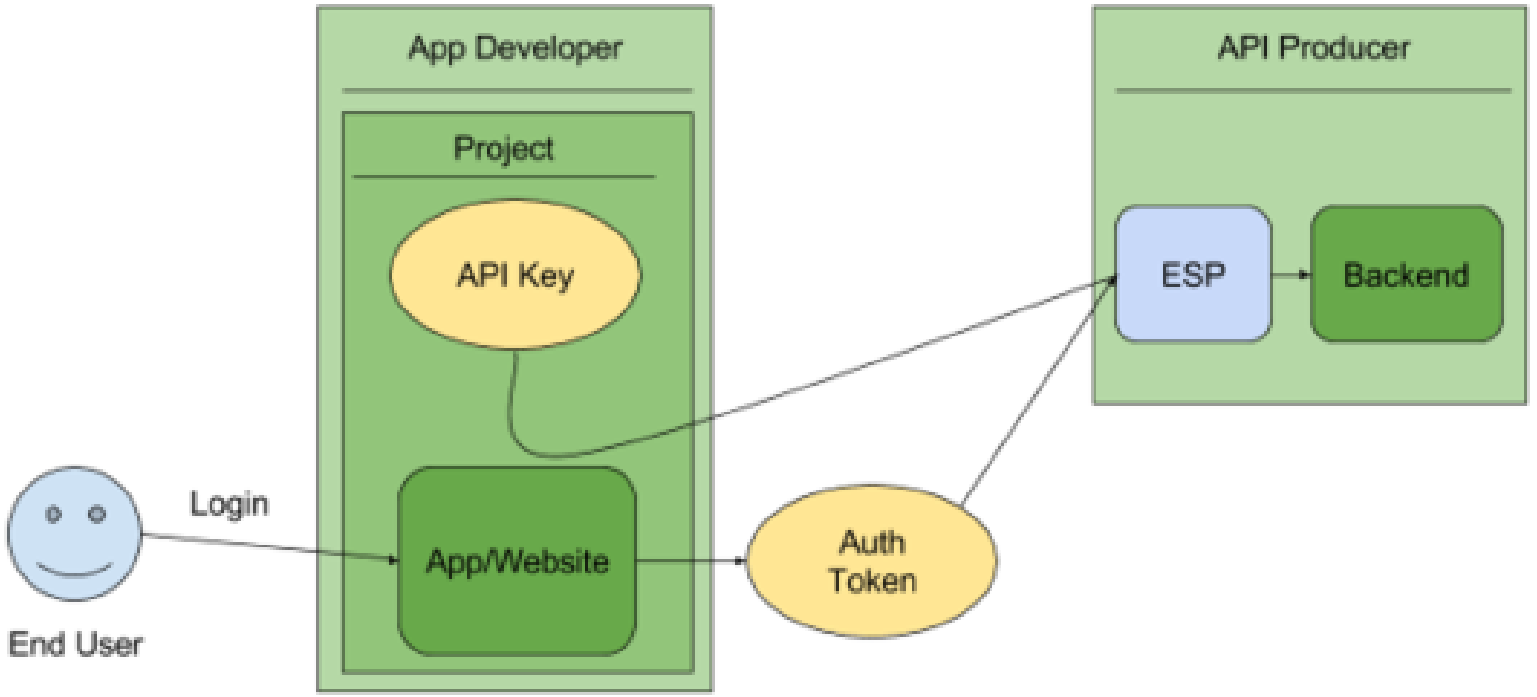
Use Case Example:

- When using a third-party application to log in with Google, Facebook, or Twitter, you will likely use OAuth.
- The application redirects you to the service provider (like Google), you authenticate there, and the provider gives the application an access token to use on your behalf.

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

AJAX

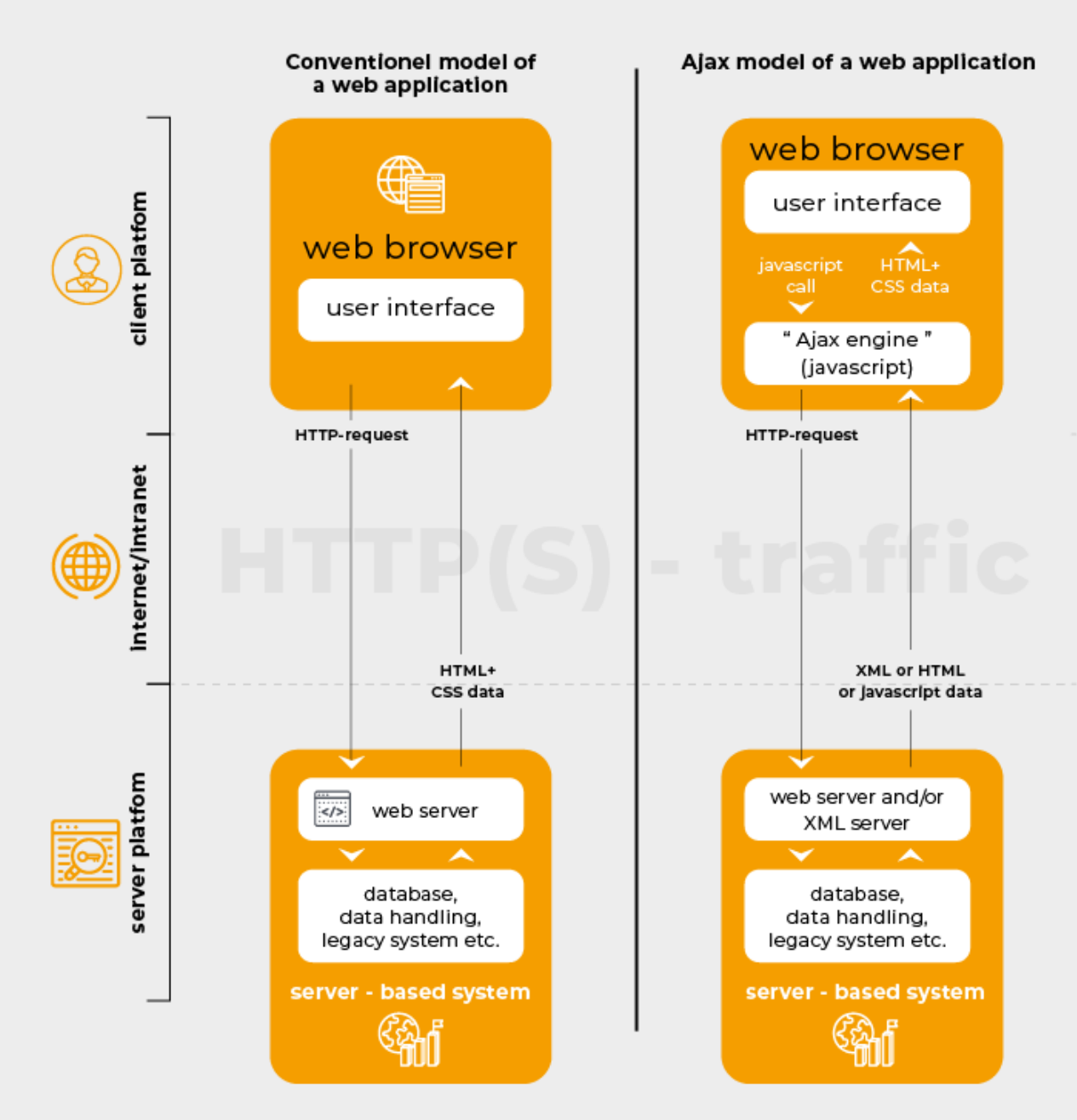
The Popular Approach

- 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:

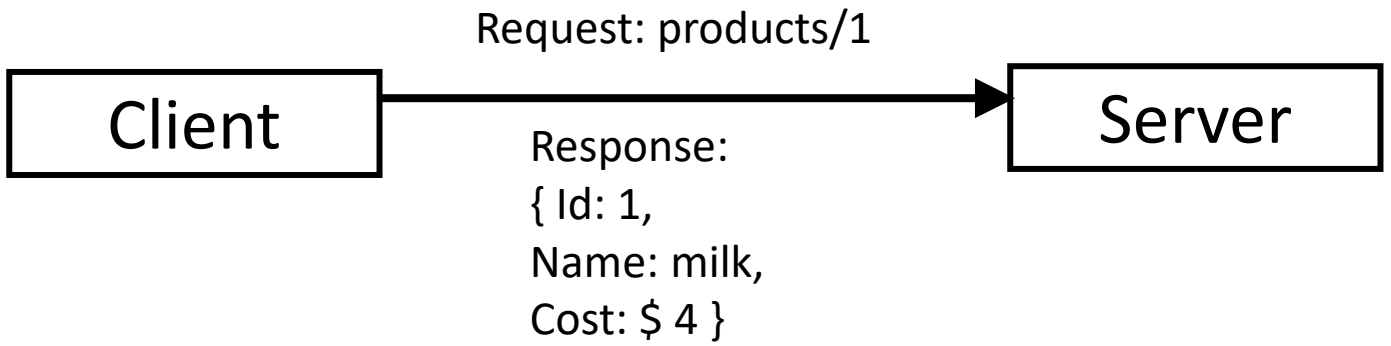
- *url*- the request url. Only mandatory option.
- type of request, “POST” or “GET”. Default is “GET”
- *dtype-atatype- “text”, “html”, “json”*
- *async*- Accepts true/false. False makes request synchronous and will block execution of other codes until response is received
- *cache*- Use to cache response 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

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,

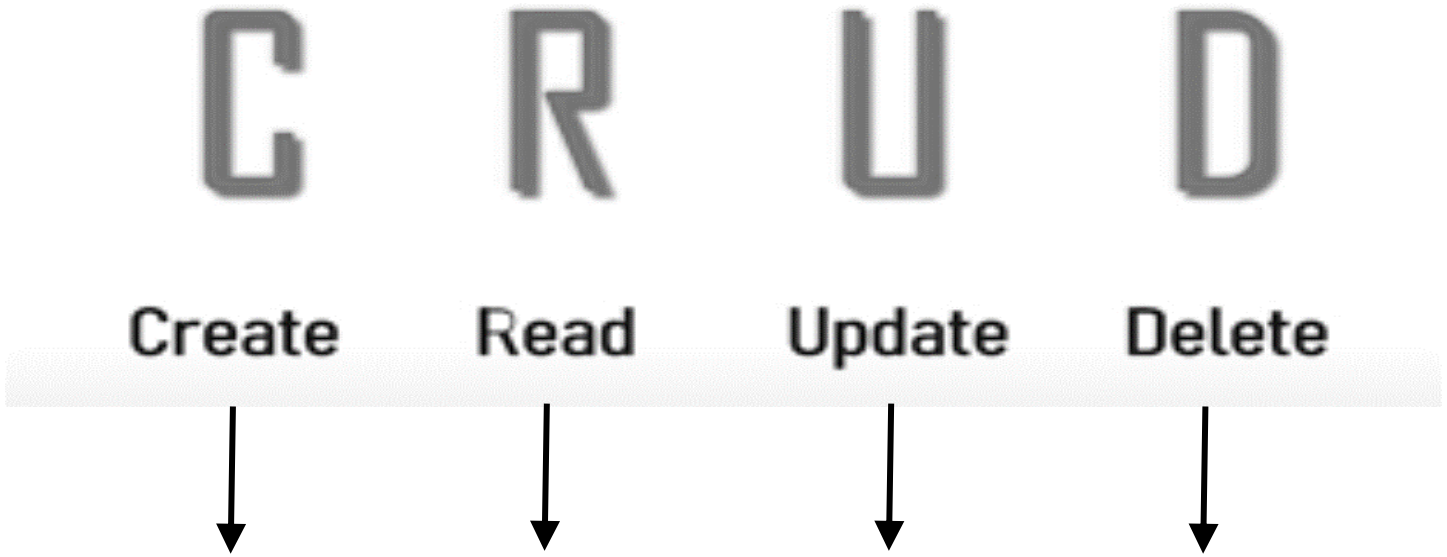
RESTful APIs - Need and Principles

- 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

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**)