

# Announcements

- Quiz 1 statistics
  - Mean: 4.02/5
- REST assignment out today
  - Rubric uploaded on the class website

CS3300 Introduction to Software Engineering

# Lecture 08: Tools of the Trade #4

JQuery, API, AJAX, RESTful API

Nimisha Roy ▶ [nroy9@gatech.edu](mailto:nroy9@gatech.edu)

# 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 API (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

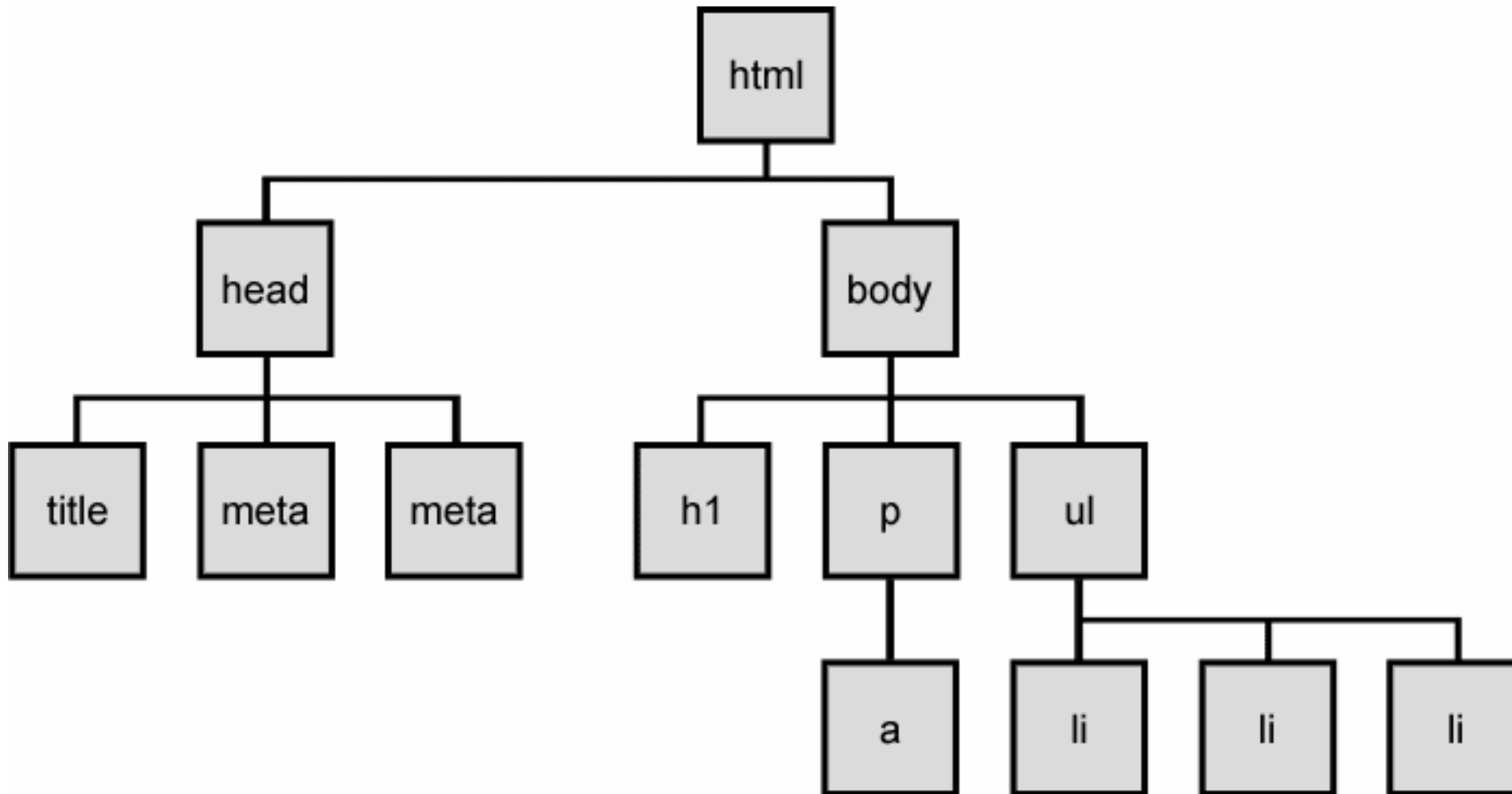
```
<html>
  <head>
    <script>
      // run this function when the document is loaded
      window.onload = function() {

        // 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) is a programming interface for web documents (XML & HTML).
- Represents the document as nodes and objects
- Can be used to modify the document with a scripting language like JavaScript

# The DOM tree



# jQuery / DOM comparison

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

# jQuery Terminology

- the jQuery function  
refers to the global jQuery object
- a jQuery object  
the object returned by the jQuery function that often represents a group of elements
- selected elements  
the DOM elements that you have selected for, most likely by some CSS selector passed to the jQuery function and possibly later filtered further



# 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 })`

# 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

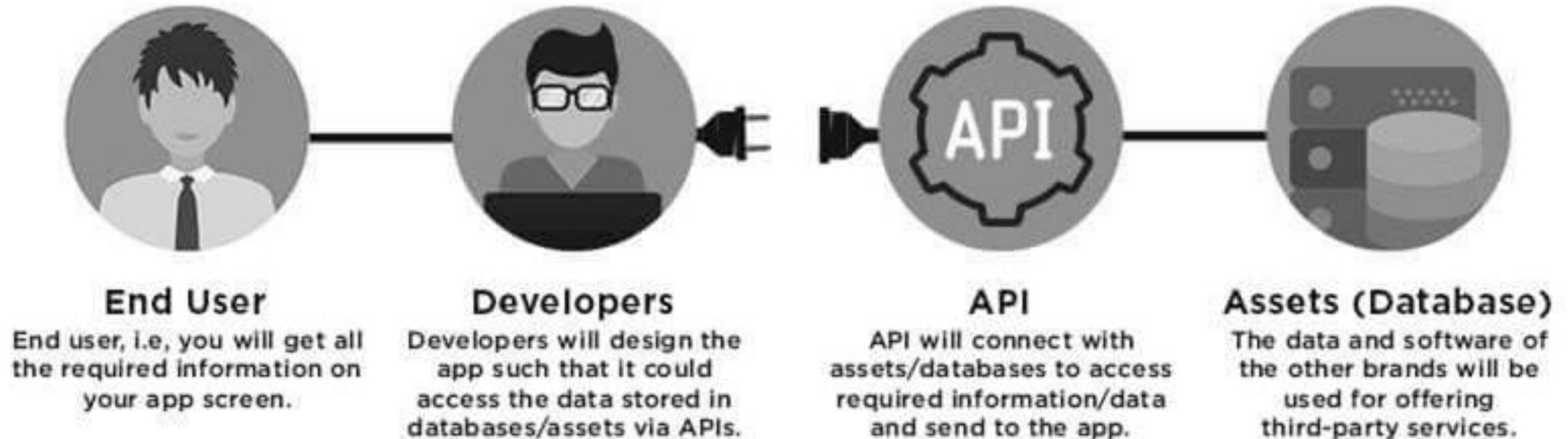
...has the DOM API that lets your code interact with the document

# How is an API used?

You want to book a flight

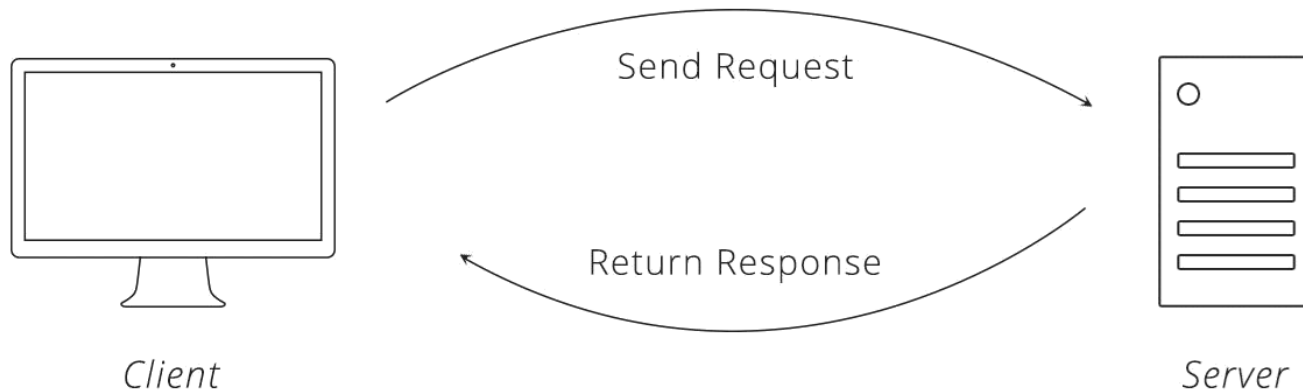
Endpoints: Flight booking platform and airlines website

Most common formats of representing data are JSON and XML



# 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 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.

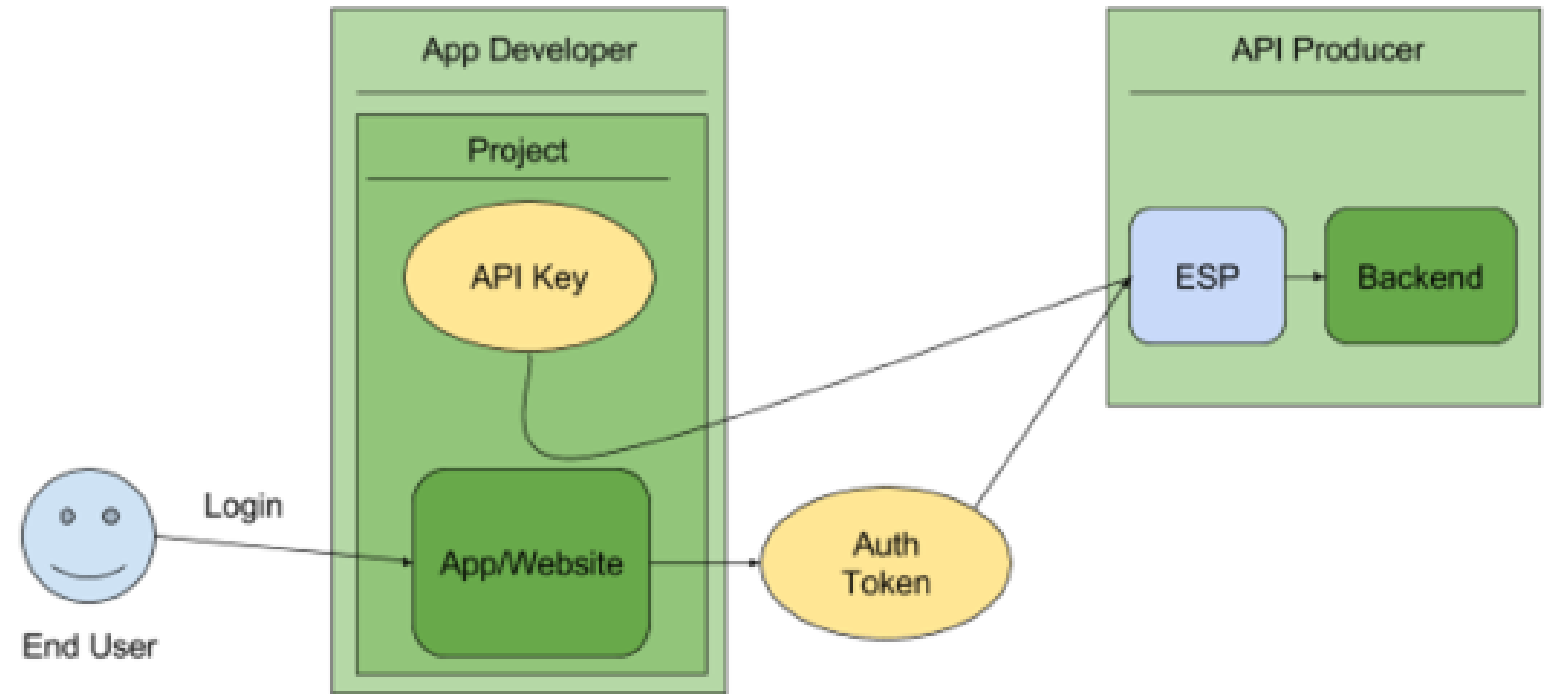
## OAuth Authentication:

- 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. More details [here](#).

# 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

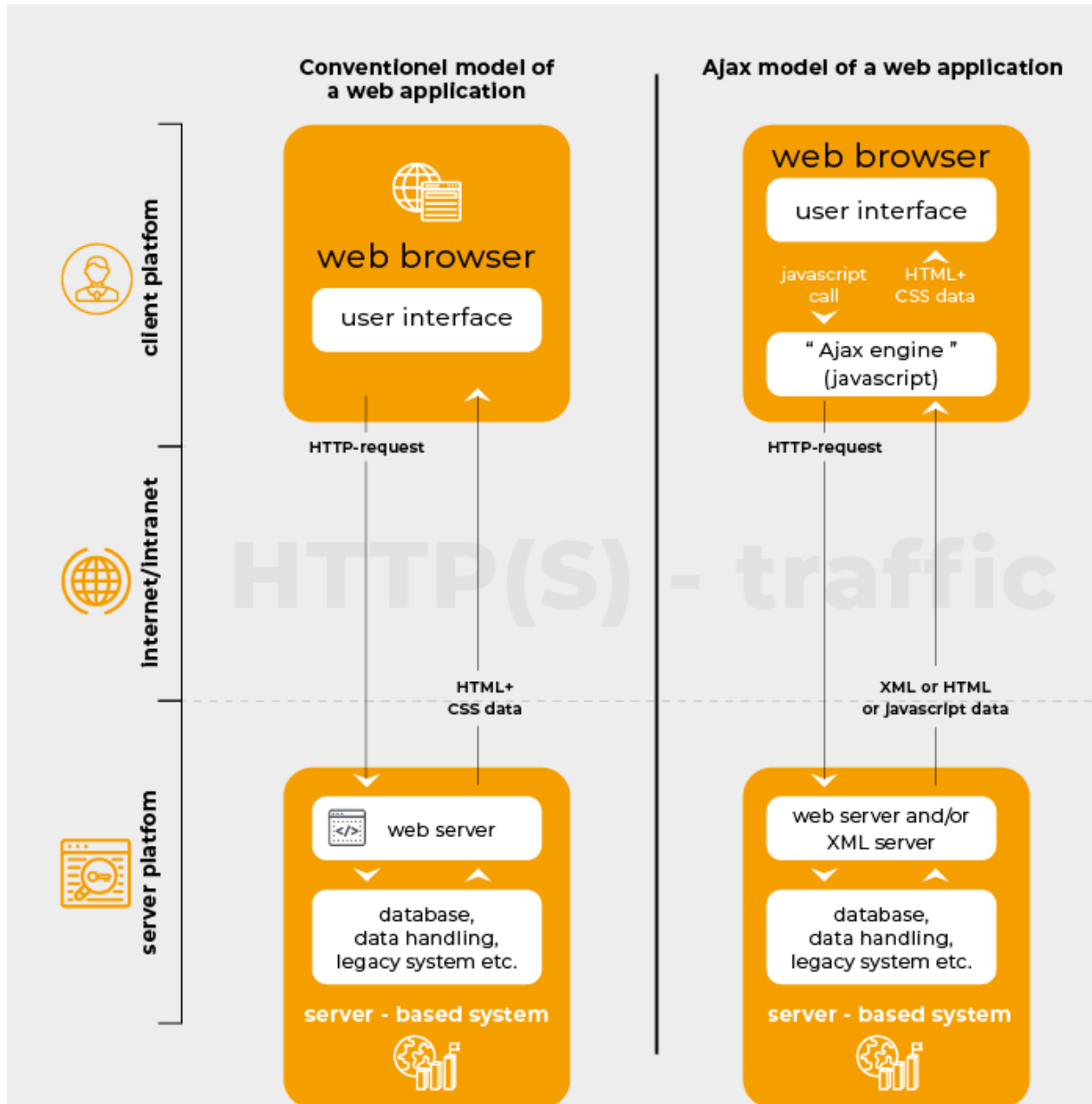
## The Popular Approach

- 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 XML**

Data serialization format

Now replaced by JSON

But AJAX 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:

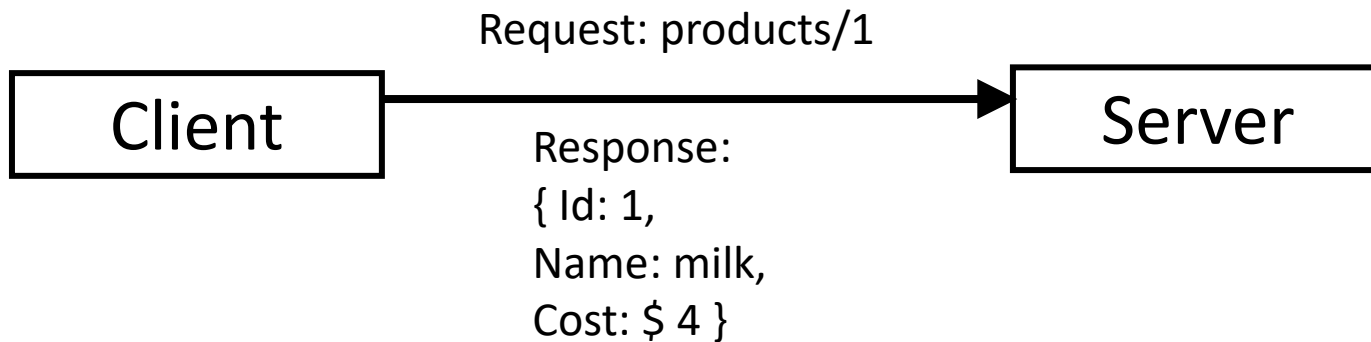
- *url*- the request url. Only mandatory option.
- *type* of request, "POST" or "GET". Default is "GET"
- *dtype-atastype*- "text", "html", "json"
- *async*- Accepts true/false. False makes request synchronous and will block execution of other codes until response is received
- *cache*- Use to cache 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 **State Transfer**
- Architectural style of web services development. Often called “Language of the Internet”
- Literally means transferring the state of representation of a resource.



- 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
  - 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

**C R U D**

**Create**

**Read**

**Update**

**Delete**



**POST**

**GET**

**PUT**

**DELETE**



# 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**)

# Next Class

- TOTT #5
- Spring & SpringBoot
- Get your Laptops!!