# Web Performance

- Secure Web Communications
    - Private Key Encryption: Sender/Receiver share private key
    - Public Key Encryption: For authentication
    - The receiver has public and private keys: Privacy
    - RSA Algorithm/Encryption
        - Prime Numbers
    - Cryptographic Functions
        - MD5 and SHA
    - Secure communication on web uses combination of public key encryption and one way cyphers
        - Bulk Cypher: Same key used to encrypt and decrypt (fast)
    - Certificate authority to make sure person is who they say they are
    - SSL, is a protocol for establishing an encrypted link between server and client, that uses authentication and encryption of transactional data

```
Secure Sockets Layer (SSL) provides end-to-end
security between client and server
authentication of both parties is done using
digital certificates
privacy is maintained using encryption
message integrity is accomplished using message
digests
SSL for HTTP is referred to as HTTPS and
operates on port 443
```

- Web Server Performance
    - DNS Redirection is an approach to Load Balancing (Has problems)
- Web Server as Proxy Cache
    - An intermediary server that accepts requests from clients and either forwards them or services the request from its own cache.

# APIS

REST

- Representational State Transfer
- Client, Server, Resources
- Postman for API testing

# AJAX

- Asynchronous Javascript and XML

- XMLHttpRequest
- Ajax Engine
- Fetch API

## Responsive

- Mobile Web Design
  - .mobi site
  - Subdomain
  - Responsive Web Design

To transform pixel-based column widths into percentage-based, *flexible* measurements use the formula: `target ÷ context = result`

## JS Frameworks

1. Node.js is a JavaScript runtime built on **Chrome's V8 JavaScript engine**.
   https://v8.dev/
1. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
2. Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionality.
3. Modules handle file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams and other core functions.

AngularJS is a complete JavaScript-based open-source front-end web application framework.

It is mainly maintained by Google and some community of individuals.

It provides a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures.

AngularJS is the frontend part of the **MEAN stack**, consisting of **M**ongoDB database, **E**xpress.js web application server framework, **A**ngular.js itself, and **N**ode.js runtime environment.

## jQuery

A framework for client-side JavaScript.

Frameworks provide useful alternatives for common programming tasks.

An open-source project at jquery.com

It simplifies

- HTML document traversing
- Event Handling
- Animating
- AJAX interactions

Example: Instead of

```
var myButton = document.getElementById("myButton");
```

In jQuery, it's just

```
$("#myButton");
```

It's a useful library **when used wisely.**

It will allow you to write JavaScript differently

- **Write less, do more.**

Remember: jQuery is just JavaScript

- What you can do with jQuery, **you can always do without jQuery** but with ***more code.***

## High-Performance Sites

- 80% of time spent in 20% of the code
- 80-90% of the end-user response time is spent on the front- end

## The "initial" 14 Rules

Make fewer HTTP requests
Use a CDN (content distribution network)
Add an Expires header
Gzip components
Put stylesheets at the top
Move scripts to the bottom
Avoid CSS expressions
Make JS and CSS external
Reduce DNS lookups
Minify JS
Avoid redirects
Remove duplicate scripts
Configure Etags
Make AJAX cacheable

Result: Primed cache is fastest, Fast Network is second, no JavaScript is third.

- Google's Page Speed Insights

# React

## How does React tackle challenges ?

- Uses 1-way data binding (**not 2-way** like Angular)
- Virtual DOM ( Efficient for frequent updates )
- Easy to understand what a component will render
- JSX - Easy to mix HTML and JS
- React dev tools and excellent community
- Server-side rendering (useful for SEO)

**Class components** are ECMAScript 6 (ES6) **classes** that can maintain a state, independent existence and a lifecycle of its own.

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

| STATE | PROPS |
|---|---|
| Internal data | External data |
| Can be changed inside component | Cannot be changed |
| Cannot be changed by parent component | Can be changed by parent component |

**React** (a.k.a. ReactJS or React.js) is is a JavaScript **library** you use for building dynamic, high performing, responsive UI for your web interfaces.

**React Native** is an entire **platform** allowing you to build native, cross-platform mobile apps.

React.js is the heart of React Native, and it embodies all React's principles and syntax, so the learning curve is easy.

Let (global) var (local)
Const (cant reassign variable)

## Spread operator (…)

- Used to split up array elements OR object properties .
- The spread operator is extremely useful for **cloning arrays and objects**. Since both are reference types (and not primitives), copying them safely (i.e., preventing future mutation of the copied original) can be tricky. With the spread operator you have an easy way of creating a (shallow!) clone of the object or array.

- Used to **merge** a list of function arguments into an array.

```
function sortArgs = ( ...args) => {
      return args.sort();
}
```

## Serverless Lambda

### Features of Serverless Architectures

No compute resource to manage

Provisioning and scaling handled by the service itself

You write code and the execution environment is provided by the service

Core functionality (e.g., database, authentication and authorization) is provided by at-scale Web Services

## Serverless Functions

- Scalable pay as you go Functions-as-a-Service (FaaS) to run your code with zero server management.
  - No servers to provision, manage, or upgrade
  - Automatically scale based on the load
  - Integrated monitoring, logging, and debugging capability
  - Built-in security at role and per function level based on the principle of least privilege
  - Key networking capabilities for hybrid and multi-cloud scenarios

Front end as a service and Backend as a service (Faas and Baas)
A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

# Using AWS Lambda

- No Servers to Manage
- Continuous Scaling
- Subsecond metering
- Bring your own code
- Simple resource model
- Flexible Authorization and Use
- Stateless but you can connect to others to store state
- Authoring functions
- Makes it easy to
  - Perform real time data processing
  - Build scalable backend services
  - Glue and choreograph systems

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

Creates a unified API front end for multiple microservices

DDoS (Distributed Denial of Service) Protection and throttling for back-end systems

Authenticate and authorize requests

## Cookies

# What is a Cookie?

Short pieces of text generated during web activity and stored in the user's machine by the user's web browser for future reference

Cookies are created by website authors who write software for reading and writing cookies

Cookies were initially used so websites would remember that a user had visited before, allowing customization of sites without need for repeating preferences

A cookie is associated with a website's domain and contains **name**, **value**, **path**, and **expiration date**

**Store** and manipulate any **information** you explicitly provide to a site

**Track** your interaction with the **site** such as pages visited, time of visits, number of visits

Use any information available to the web server including: your IP address, Operating System, Browser Type

## Six Ways to Opt Out of cookies

1. **Select "do not track" in your browser Settings.** This setting is available Firefox 9+, Chrome, Safari 5.1+, Internet Explorer 9/10. See "Web Tracking Protection" submission: http://www.w3.org/Submission/2011/SUBM-web-tracking-protection-20110224/#dnt-uas
2. **Download opt-out cookies**. This is a process that usually involves clicking on a button to download the opt-out cookie.
   - you go to the marketer's web site, find the privacy policy, then find the "opt out" information. The cookie your computer will get tells the company not to track you anymore.
3. **Use the cookie management tools in your web browser**. In most web browsers, you can set your browser to accept only session cookies, or to turn all cookies into session cookies. Session cookies are generally harmless.
   - For Macintosh Safari users, you can tell the browser to only accept cookies from "the site you are navigating to." This means that you will not accept third party cookies.
4. **View current cookies and delete what you don't need.** Most web browsers allow you to see what cookies you already have stored.
   - Some cookies, such as registration cookies for web sites you visit frequently, are useful to keep around. But other cookies, like tracking cookies from atdmt.com, doubleclick.net, 2o7.net, atwola.com, and other advertisers aren't necessarily helpful to you.
5. **Check your account preferences on registration sites**. Some sites, such as eBay, require registration and the use of cookies. On eBay, for example, if you do not opt-out of advertising tracking, information about your eBay activities can be used by other sites and advertisers outside of eBay.
6. **Use browser Add-ons**. Free browser extensions are available for most browsers to control tracking, such as Ghostery (www.ghostery.com)

# HTML5

Canvas, svg, section, nav, article, video and audio codecs, localStorage, geoLocation, APIs

# Web Intrusion

**Cross-site scripting** attacks (XSS)

- Most common form of attack since 2008. More details on this later

**SQL injection** attacks

- These attacks maliciously alter the backend databases of websites thus making them redirect users to malware sites.

Search Engine result **Redirection**

- Example :Easter related search results poisoned redirecting users to malicious software

Attacks on **backend virtual hosting** companies

Vulnerabilities in web-server or forum-hosting software

- Example: PhPBB (PHP Bulletin Boards) vulnerabilities

Using **social networks** to infect users (these are a combination of social engineering and above attacks)

- Example: MySpace SAMY worm (see the bookmarks)

"Poor Alice" can get the malware planted by "Joe" in many ways

- By installing "**fake codecs**" embedded with Trojans.
  - Example: zlob Trojan.
- By viewing "malicious **advertisements**"
  - Example: Flash Banner ads as seen in 2008.
- By installing "**fake scanners**" or "misleading applications" (also called scareware/ rogueware).
  - Example: Some malware trick users into believing that their computer is infected and urges them to install software like "Antivirus 2009" which itself is a malware.
- By visiting malicious **P2P sites** and downloading malicious content
- By visiting websites sent as email links by the hacker
  - This is also a form of "social engineering attack"
- By visiting links posted on "**Blog Sites**" under "Blog Comments"
  - Blog Spam is very common and many unsuspecting fall prey to links posted by malicious individuals posing as honest opinionates.
- By installing **pirated software** from *warez* sites which are maliciously modified by hackers.

Weak Passwords, Weak Password Validation,
Passphrases
Session ID Attack
Client Side Attacks

- Cross site scripting (XSS): Due to breaches of browser security, XSS enables attackers to inject client-side script into Web pages viewed by other users.

  The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type

  - when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to parse and display a page of results for and to that user, without properly sanitizing the request

  The *persistent* (or *stored*) XSS occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing.

-

ClickJacking
Plugins Vulnerability
JSON Array to Injection, SQL Injection

Search Worms: Search for vulnerabilities
Tor