INTRODUCTION TO FULL-STACK DEVELOPMENT

FULL-STACK?

- What is the “full-stack” in tems of full-stack development?
  - The Data Structure?
  - The function-call stack?
  - The program stack?
  - None of the above.
  - The “stack” here is a not-necessarily-LIFO arrangement of the various languages and frameworks used to handle all aspects of a client server application end-to-end.

FULL-STACK THROUGH TIME

- Client-server programming with responsive front-ends evolved with the Internet.
- Before using a “stack” of framweoks become the de-facto standad, we had stacks made of the same native programming language.
    - Java (Swing -> Servelets -> JDBC -> Oracle)
    - C++ (Qt GUI -> C++ back ends -> Some SQL DB)
    - The .NET suite of languages, etc.
    - Ruby on Rails
    - The Python stack: JS -> Python-> Django -> Some DB
- One of the earliest “stacks” was LAMP -> JS on the Front End and Linux -> Apache -> MySQL->PHP on the back end
    - A WAMP stack also existed.

PRINCIPLES OF FULL-STACK DEVELOPMENT

- vailability
- Performance
- Reliability
- Scalability
- Manageability
- Cost

CURRENTLY POPULAR STACKS

- MEAN stack: JavaScript - MongoDB - Express - AngularJS - Node.js
- MERN stack: JavaScript - MongoDB - Express - ReactJS - Node.js
- Django stack: JavaScript - Python - Django - MySQL
- LAMP stack: JavaScript - Linux - Apache - MySQL - PHP
- LEMP stack: JavaScript - Linux - Nginx - MySQL - PHP
- Ruby on Rails: JavaScript - Ruby - SQLite - Rails

ADVANTAGES

The advantage of being a full stack web developer is:

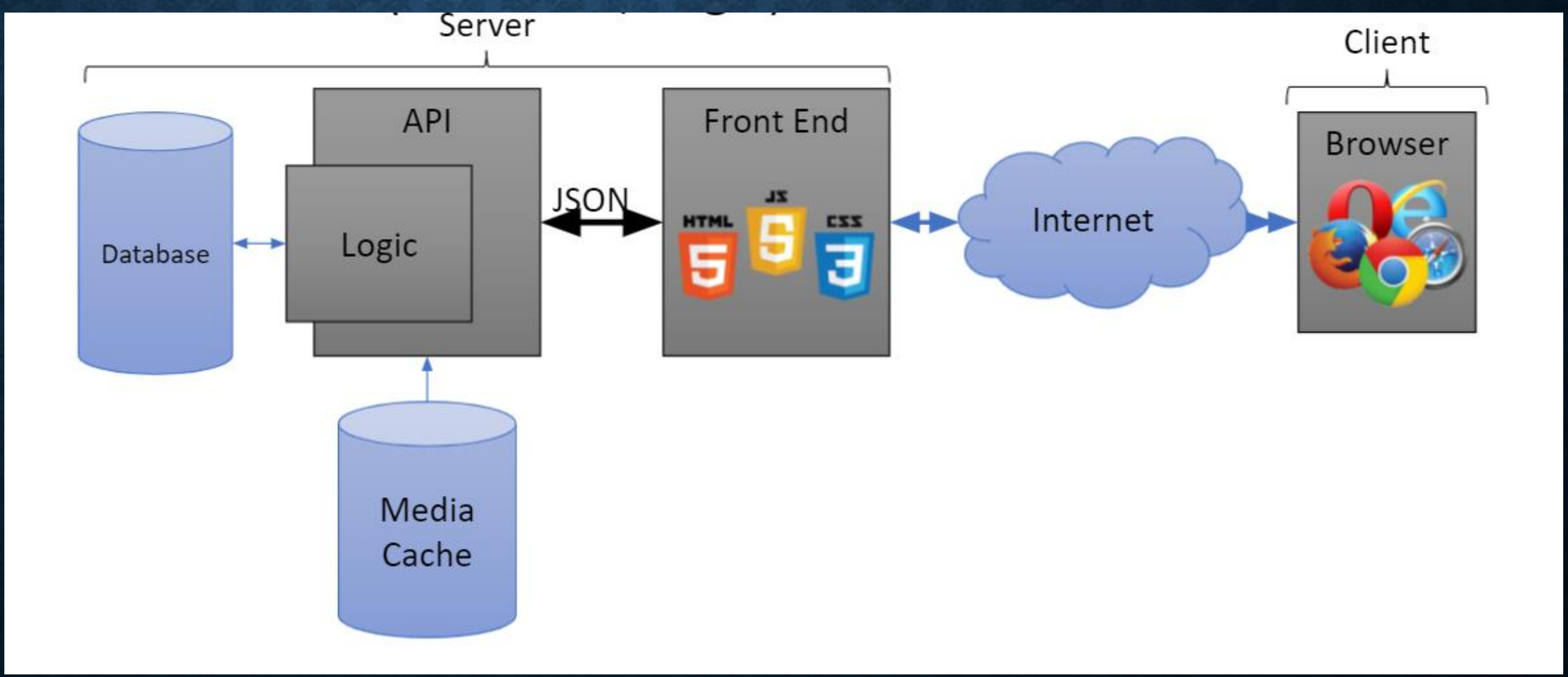
- Master all the techniques involved in a development project
- Make a prototype very rapidly
- Switch between front and back end development based on requirements
- Understand all aspects of new and upcoming technologies

DISADVANTAGES

- The solution chosen can be wrong for the project
- The solution chosen can be dependent on developer skills
- The solution can generate a key person risk
- Being a full stack developer is increasingly complex

CORE COMPONENTS

- UI (Front End (DOM, Framework))
- Request Layer (Web API)
- Back End (Database, Logic)



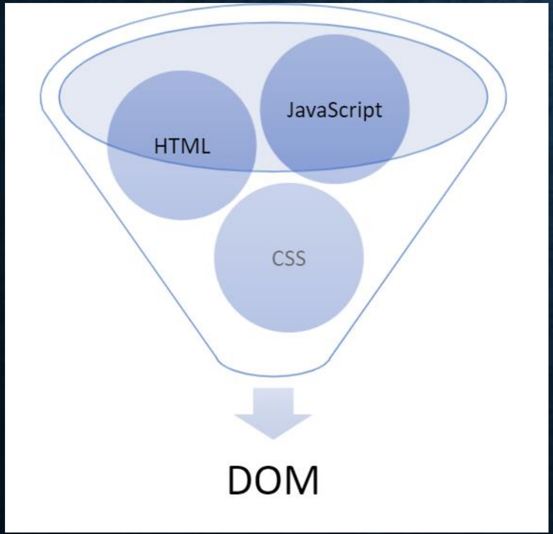
THE FRONT-END

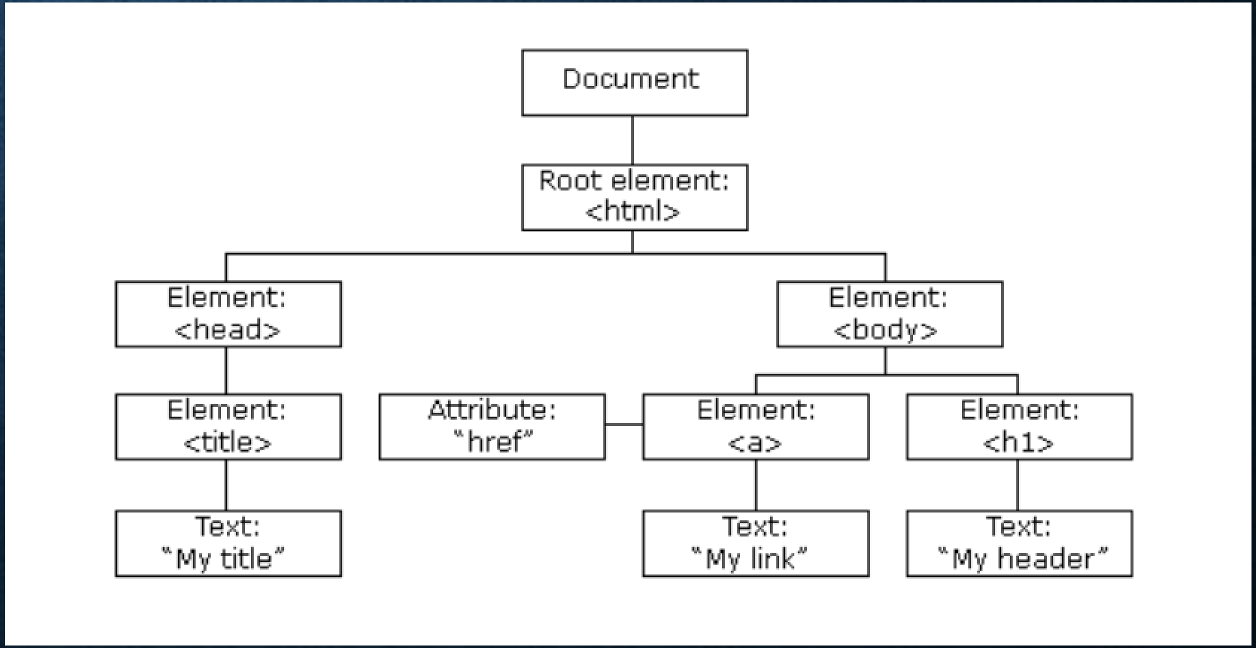
- HTML/CSS
- Javascript
- Java (applets)

- What is the most popular?
- Javascript/HTML/CSS is the only real option for front-end native languages and is basically the standard. But there are many variations on JavaScript that are used.

THE DOCUMENT-OBJECT MODEL

- Document Object Model makes every addressable item in a web application an Object that can be manipulated for color, transparency, position, sound and behaviors.
- Every HTML Tag is a DOM object





FRAMEWORKS

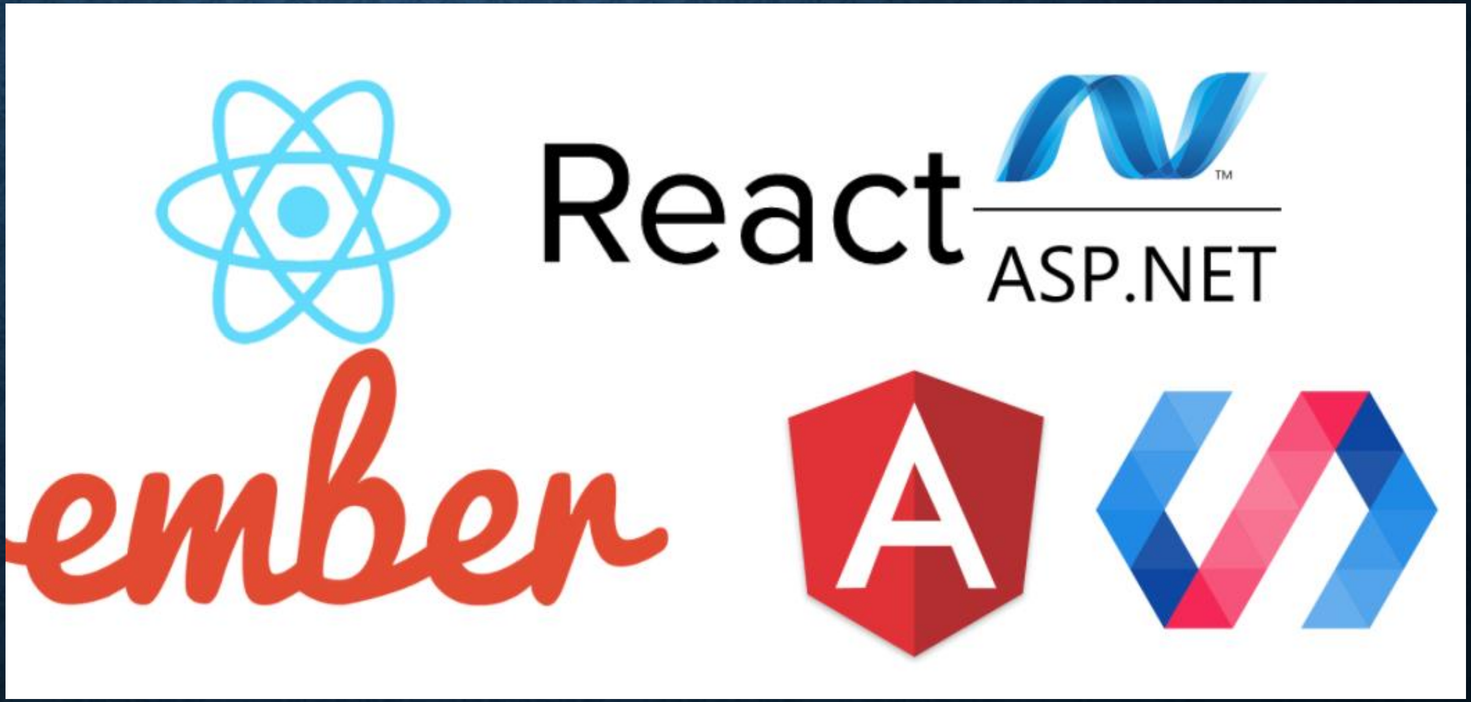
- A Software Framework designed to reduce overhead in web development
- Types of Framework Architectures
  - Model-View-Controller (MVC)
  - Push vs Pull Based
    - Most MVC Frameworks user push-based architecture “action based” (Django, Ruby on Rails, Symfony, Stripes)
    - Pull-based or “component based” (Lift, Angular2, React)
- Three Tier Organization
  - Client (Usually the browser running HTML/Javascipt/CSS)
  - Application (Running the Business Logic)
  - Database (Data Storage)
- Types of Frameworks
  - Server Side: Django, Ruby on Rails
  - Client Side: Angular, React, Vue

MVC ARCHITECTURE

- A Web Application Framework Development Principle
- Model (M):
  - Where the data for the DOM is stored and handled)
  - This is where the backend connects
- View (V):
  - Think of this like a Page which is a single DOM
  - Where changes to the page are rendered and displayed
- Control (C):
  - This handles user input and interactions
    - Buttons
    - Forms
    - General Interface

POPULAR FRONT-END FRAMEWORKS

- React
- Vue.js
- AngularJS/Angular 2
- ASP.net
- Polymer
- Ember.js



THE BACK-END

- All of the awesome code that runs your application.
- Web API
    - Connection layer between the frontend and backend
    - Connected through API calls (POST, GET, PUT, etc.)
    - Transmit Content from the Backend to the Frontend commonly in JSON Blobs
- Service Architecture that drives everything (Where all the logic is)

WEB API’S

- The intermediate layer between front end and back-end systems
- A “must have” if your APIs will be consumed by third-party services
- Attention to details:
  - How consumable is the API (signature, content negotiation)?
  - Does it comply with standards (response codes, etc.)?
  - Is it secure?
  - How do you handle multiple versions?
  - Is it truly RESTful?

REPRESENTATIONAL STATE TRANSFER (REST)

- Client-server
- Stateless
- Resource-based (vs. remote procedure call)
- HTTP methods (GET, POST, PUT, DELETE)
- Side Effects
- It’s a style, not a standard
- Don’t hate on HATEOAS

HYPERMEDIA AS THE ENGINE OF APPLICATION STATE (HATEOAS)

- Hypermedia is the key
- It all starts at a URL
- Resources are returned
- Media types and locations are included
- References based on state

WEBAPI TERMS

- GET – “read”
- POST – “insert” (collection)
- PUT – “replace”
- DELETE – “remove”
- PATCH – “update”
- Custom (proceed with caution)

WEB STATUS CODES

- 200 – OK – things are great (return the item)
- 201 Created – after POST (HATEOAS – return location)
- 204 No Content (i.e. successful DELETE)
- 400 – Bad Request (validation error, missing parms, etc.)
- 401 – Unauthorized – Who are you?
- 403 – Forbidden – No soup for you
- 404 – Not Found

JQUERY

- One of the first “frameworks”
- Allows for DOM Manipulation
- Does not provide structure to your code
- Does not allow for two way binding

ANGULAR JS

- MVC Javascript Framework by Google for Rich Web Application Development
- Structure, Quality and Organization
- Lightweight (< 36KB compressed and minified)
- Free
- Separation of concern
- Modularity
- Extensibility & Maintainability
- Reusable Components

FEATURES OF ANGULARJS

- Two-way Data Binding – Model as single source of truth
- Directives – Extend HTML
- MVC
- Dependency Injection
- Testing
- Deep Linking (Map URL to route Definition)
- Server-Side Communication

ANGULAR-BINDING

- <html ng-app>
- <head>
-   <script src='angular.js'></script>
- </head>
- <body>
-   <input ng-model='user.name'>
-   <div ng-show='user.name'>Hi {{user.name}}</div>
- </body>
- </html>

REACT

- A JavaScript Library For Building User Interfaces
- Renders your UI and responds to events.
- It also uses the concept called Virtual DOM, creates an in-memory data structure cache, enumerates the resulting differences, and then updates the browser's displayed DOM efficiently.
- One of the unique features of React.js is not only it can perform on the client side, but it can also be rendered on the server side, and they can work together interoperably.

HOW IS IT DIFFERENT FROM ANGULAR?

- ~~Angular has~~
- ~~modules~~
- ~~controllers~~
- ~~directives~~
- ~~scopes~~
- ~~templating~~
- ~~linking functions~~
- ~~filters~~
- ~~dependency injection~~

Just Components