Slides courtesy

ECE 3822: Software Tools for Engineers

Prof. Joseph Picone

Department of Electrical and Computer Engineering

Temple University

https://www.isip.piconepress.com/courses/temple/ece_3822/

LECTURE 32: INTRO TO WEB DEVELOPMENT

- **Objectives:**

  Basic Web Application Model
  Web Development Frameworks/Languages

- **Resources:**

  Web Frameworks
  Popular Frameworks
  10 Things to Know
  Angular
  React
  Knockout

- **Videos:**

  Rest
  Postman
  Chrome Developer Tools

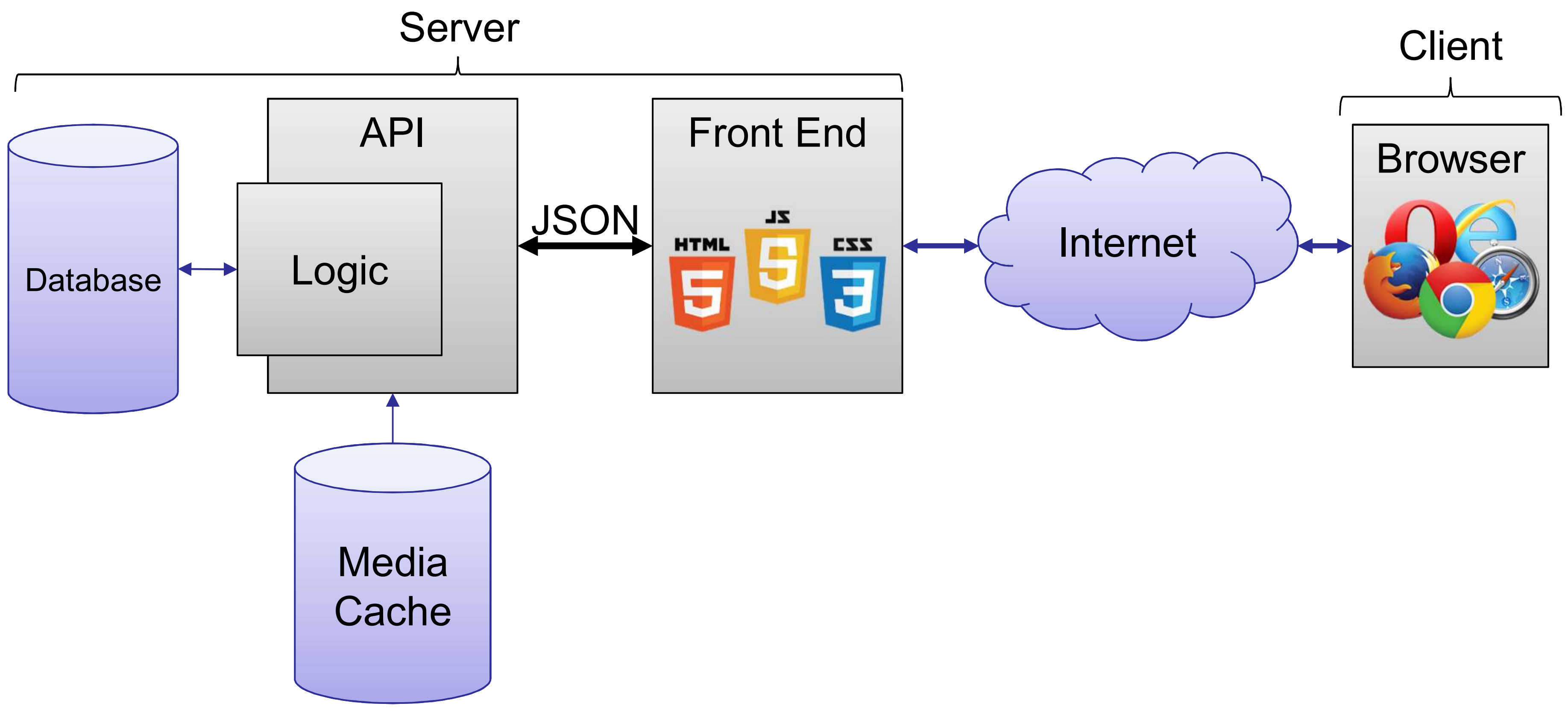
Principles of Web Design

- Availability
- Performance
- Reliability
- Scalability
- Manageability
- Cost

Scalability
We currently receive c10,000 trace requests each month.
We currently have capacity to take on an additional 10,000 trace requests each month.
With un-capped earning potential and heavy bonus penalties for inaccurate results, our teams only real restriction is the number of hours in the day.

Core Components of Web Applications

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



FRONTEND DEVELOPMENT

Front End Languages

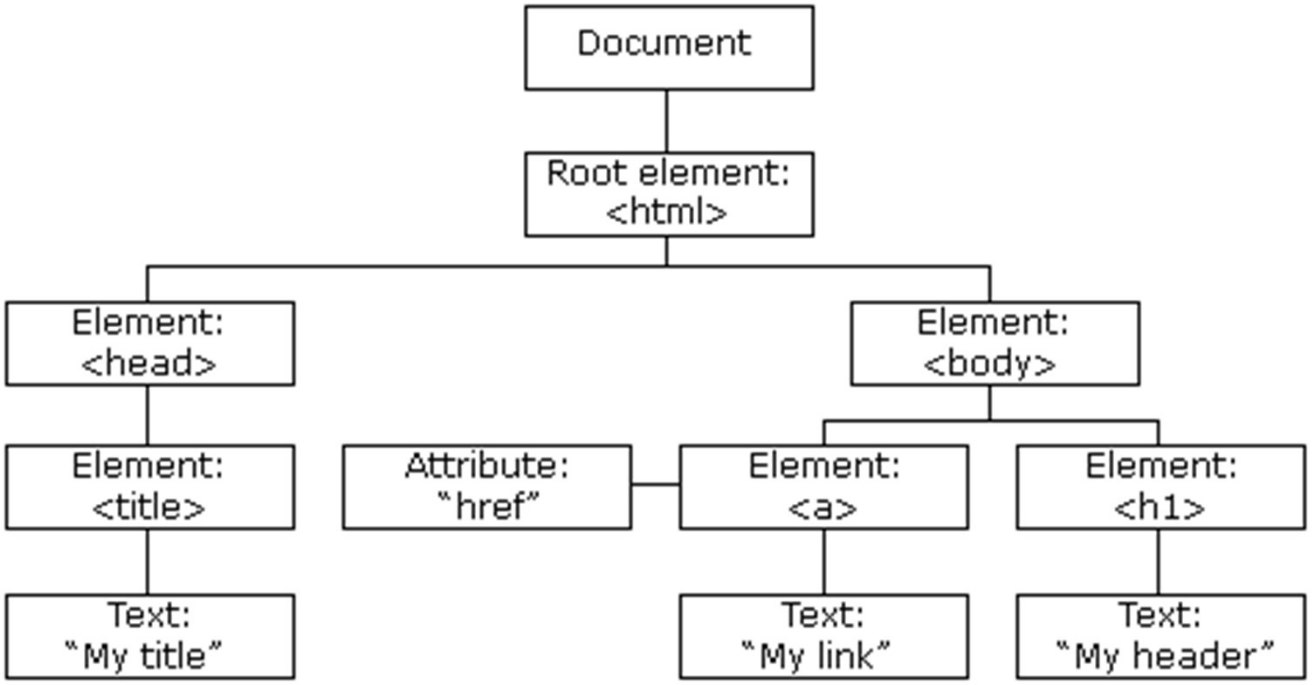
- HTML/CSS
- Javascript
- Java (applets)

What is the most popular?

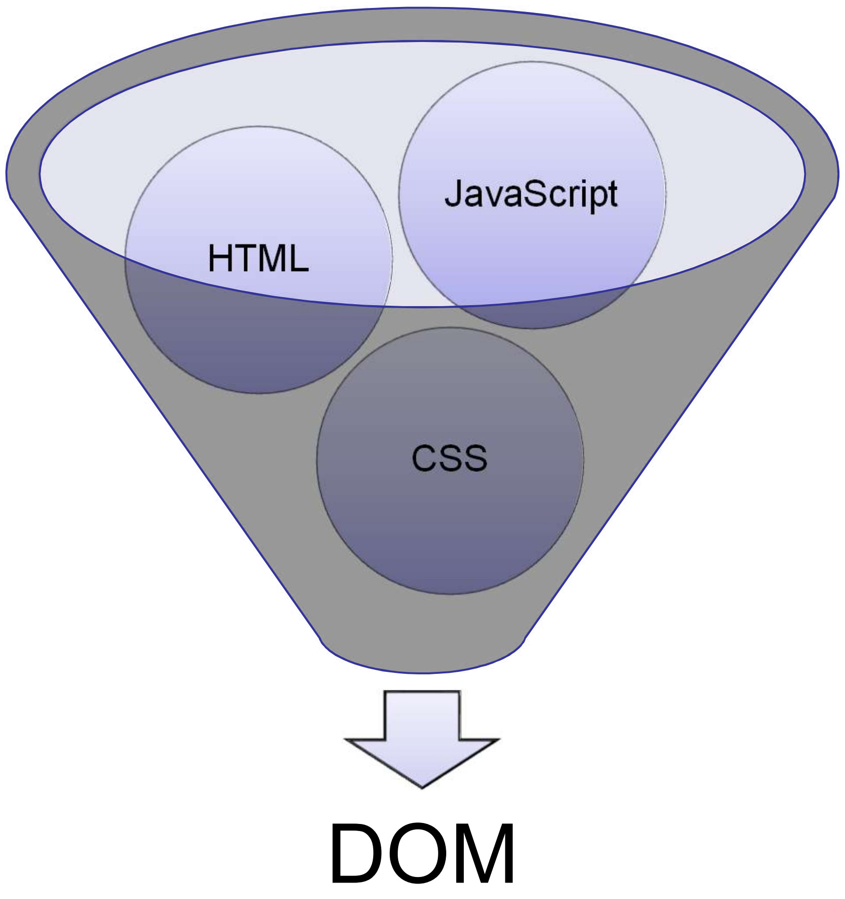
Answer: Javascript/HTML/CSS is the only real option for front-end native languages and is basically the standard. But there are many variations on JavaScript that are used.

DOM (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



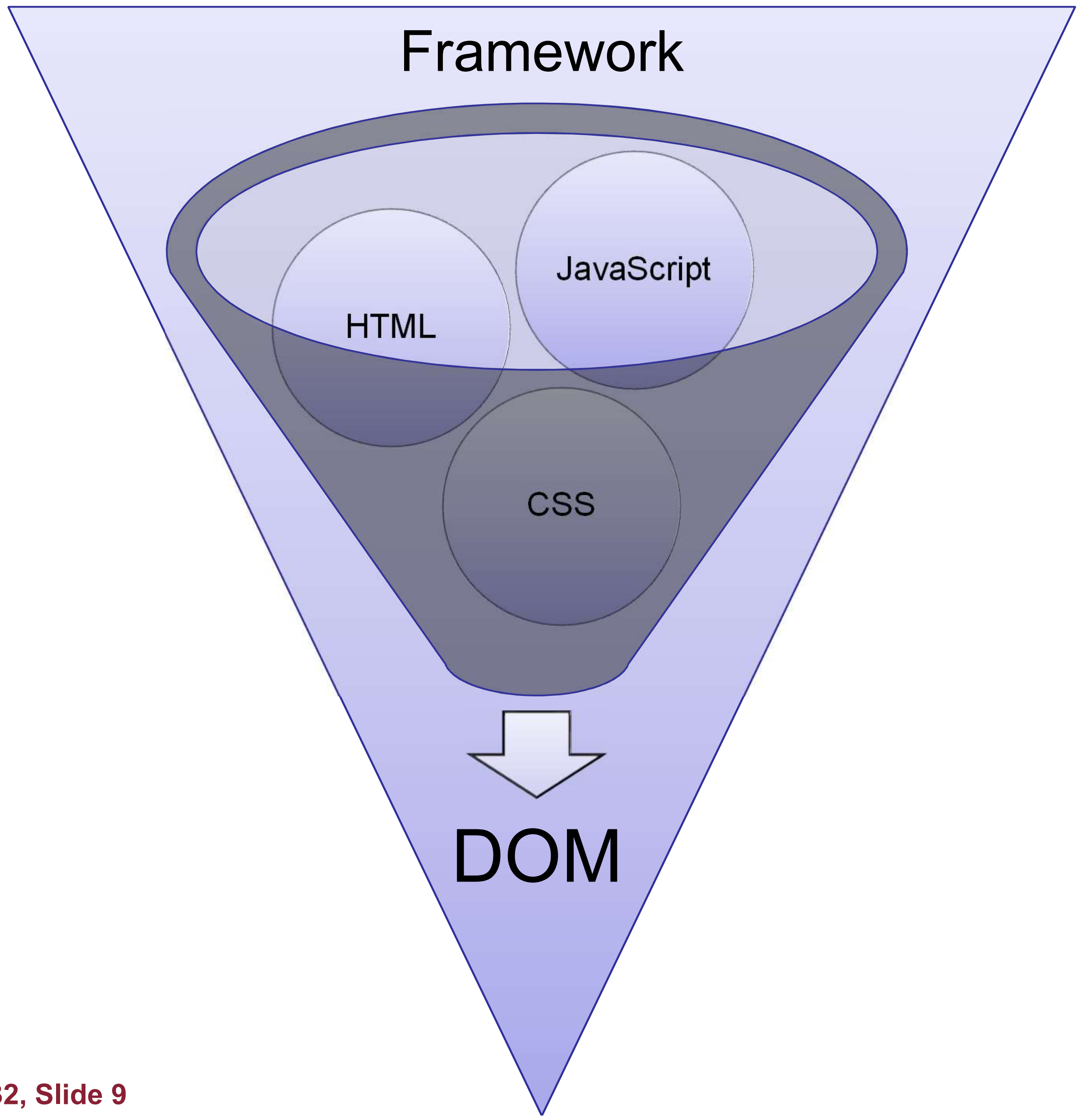
DOM (Document Object Model)



What is a Framework?

- 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)
    - These frameworks use actions that do the required processing, and then "push" the data to the view layer to render the results.
    - Pull-based or “component based” (Lift, Angular2, React)
    - These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed.
  - 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

Framework



Javascript Frameworks

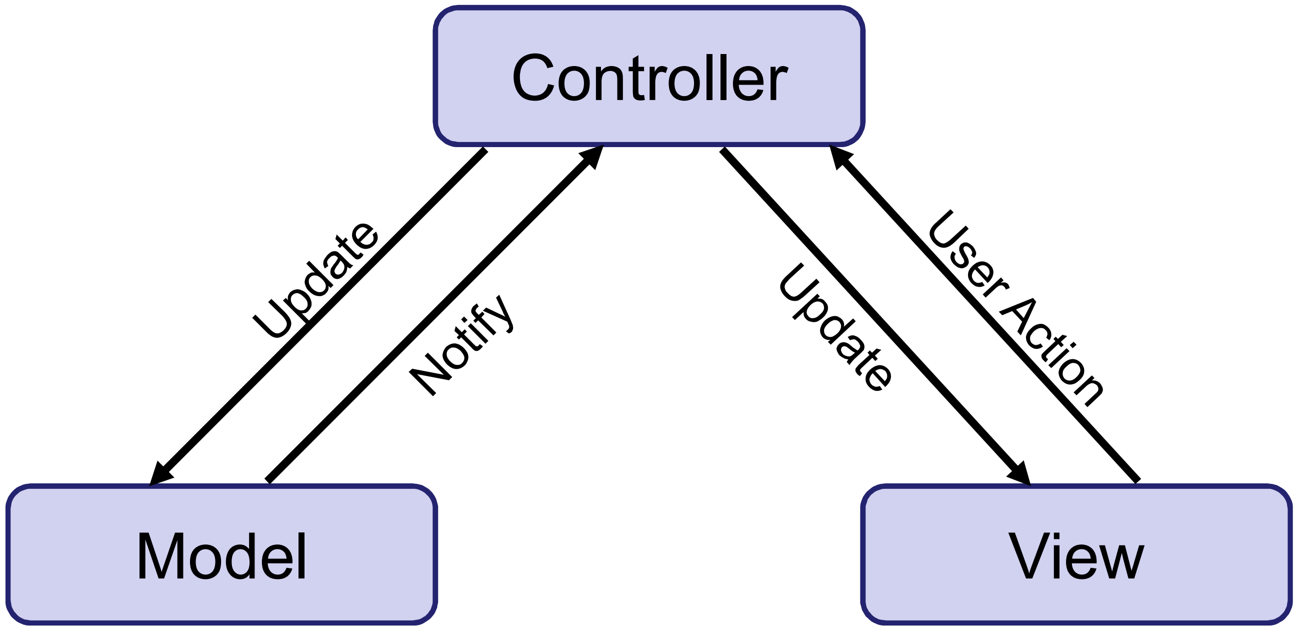
http://hotframeworks.com

- AngularJS/Angular 2
- ASP.net
- React
- Polymer 1.0
- Ember.js
- Vue.js

MVC (Model View Controller)

- A Web Application Development Framework
- 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

MVC Model



BACKEND DEVELOPMENT

What is a Backend?

- All of the awesome 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)

What is a WebAPI?

- 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

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

Popular Tools

Development Tools:

- Chrome/Firefox Developer Tools
- Postman (API)
- Dreamweaver
- Git / SourceTree

Analytics Tools:

- Google/Adobe Analytics

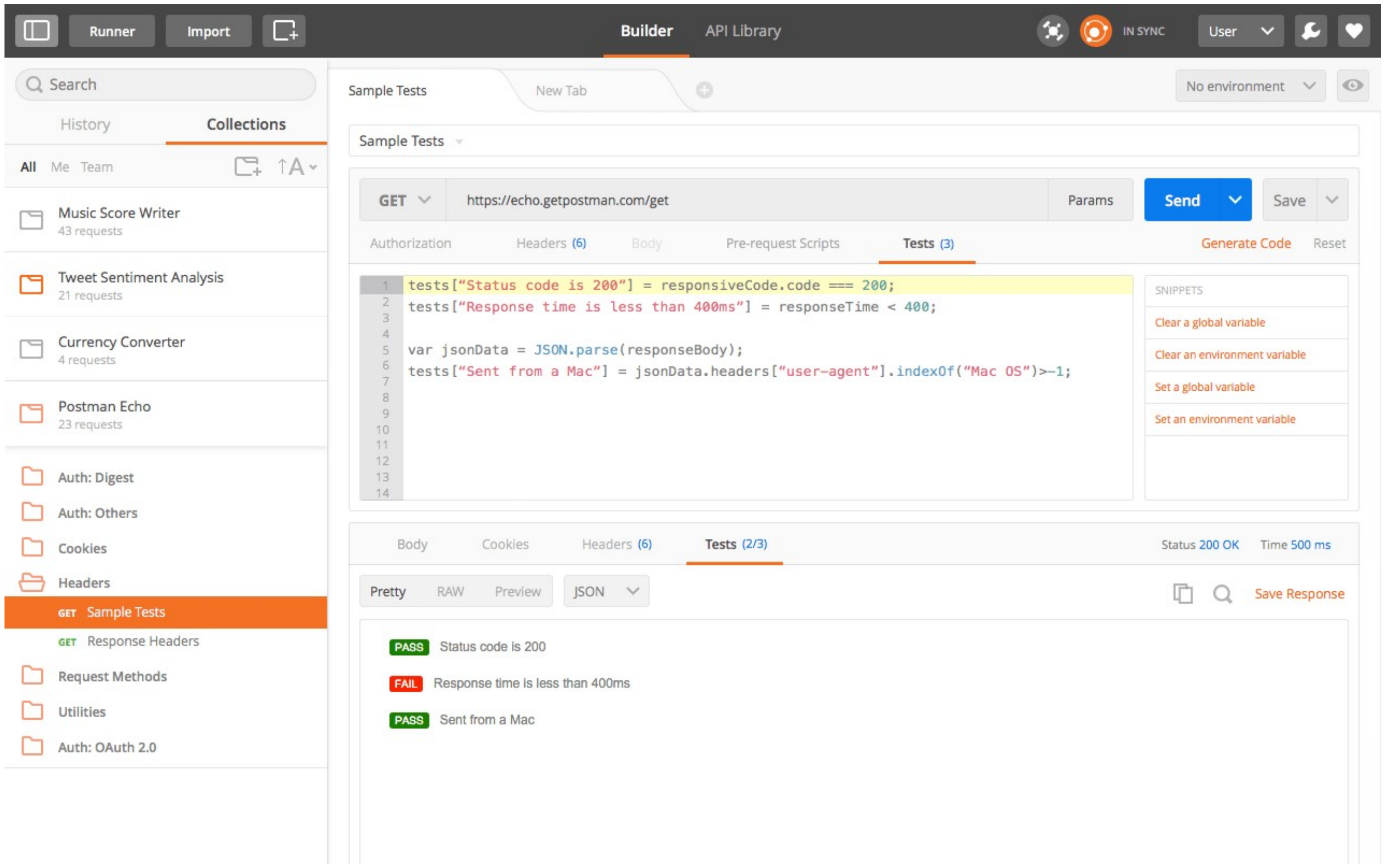
Tools for Testing WebAPI

- Postman Chrome extension

http://bit.ly/postmanext

- Fiddler by Telerik

http://www.Telerik.com/fiddler



```
tests["Status code is 200"] = responsiveCode.code === 200;
tests["Response time is less than 400ms"] = responseTime < 400;

var jsonData = JSON.parse(responseBody);
tests["Sent from a Mac"] = jsonData.headers["user-agent"].indexOf("Mac OS")>-1;
```

```
PASS Status code is 200
FAIL Response time is less than 400ms
PASS Sent from a Mac
```

APPENDIX

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

What is Angular

- MVC Structure
- Framework
- Single Page Application (SPA)
- Client Side Template
- Testing

Why Angular?

New Developers

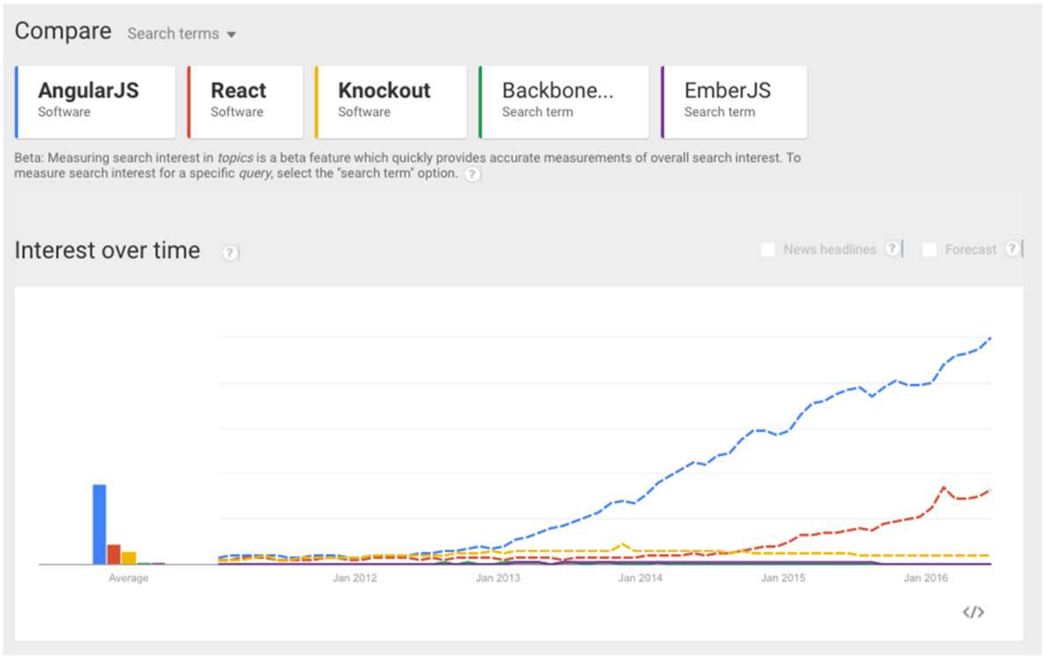
- Popularity
- Demand
- Support and Resources
- Front End

Seasoned Developers

- Structured and Opinionated Framework
- Productivity
- Consistency

Team Leads

- Efficiency
- Longevity



Beta: Measuring search interest in *topics* is a beta feature which quickly provides accurate measurements of overall search interest. To measure search interest for a specific *query*, select the "search term" option.

Angular vs. Angular 2

- Angular 1
  - Structured MVC Framework
  - Separation of HTML and Logic
  - Client Side Templating

- Angular 2
  - Component Based UI
  - More Modular Design
  - TypeScript
  - Backwards Compatible
  - Faster

Angular vs. Angular2

```
angular.module('myModule')
  .controller('myController',function(){
  })

<body>
                <div ng-controller="myController">
                </div>
</body>
```

```
import { Component } from '@angular/core'

@Component({
  selector: 'my-app',
  template: ``
})
export class MyAppComponent {
}
<my-app></my-app>
```

Typescript

JavaScript

```
var num = 5;
var name = "Speros";
var something = 123;
var list = [1,2,3];

function square(num) {
        return num * num;
}
```

TypeScript

```
var num: number = 5;
var name: string = "Speros"
var something: any = 123;
var list: Array<number> = [1,2,3];

function square(num: number):
number {
        return num * num;
}
```

Typescript

JavaScript

```
var Person = (function () {
    function Person(name) {
        this.name = name;
    }
    return Person;
}());

var aPerson = new Person("Ada");
```

TypeScript

```
class Person {
  constructor(public name: string){

  }
}

var aPerson = new Person("Ada
Lovelace");
```

Building Blocks

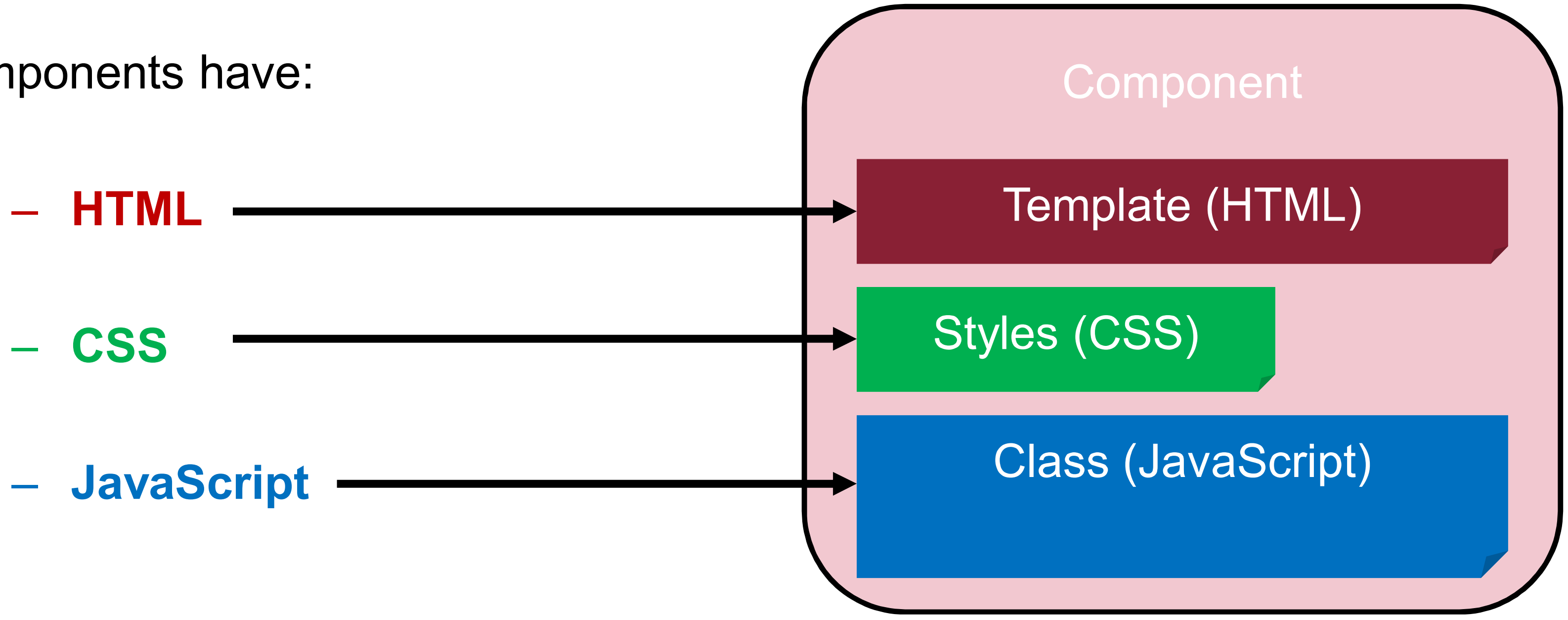
- Directives
  - **Component** – *Templates (HTML), Styles (CSS), & Logic (JavaScript)*
  - **Attribute** – *Styling HTML*
  - **Structural** – *Manipulating HTML*
- Data Flow
  - **Interpolation** – *Variable Printing in Templates*
  - **Event Binding** – *Trigger Events*
  - **2-Way Binding** – *Variables updated in real time*
- Providers
  - **Services**
    - **Reusable Logic**
    - **Data Storing and Manipulation**
  - **Libraries**

Component Directives

"…reusable building blocks for an application"

Components have:

- **HTML**
- **CSS**
- **JavaScript**

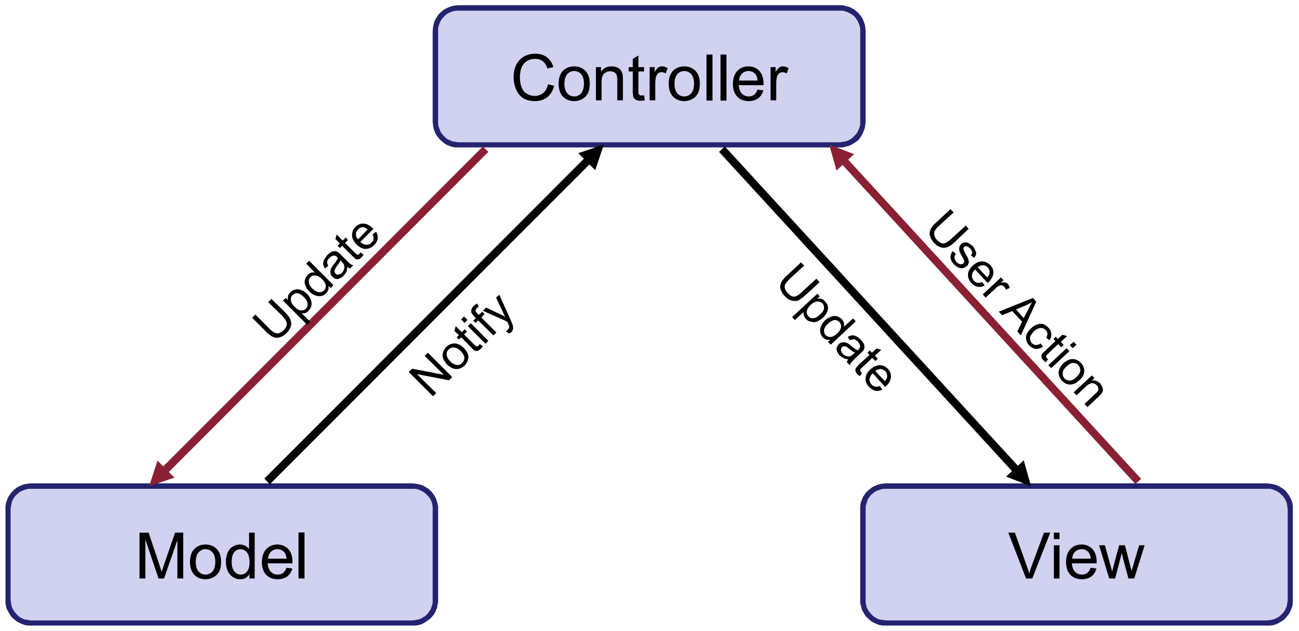


Learn Angular/Angular2

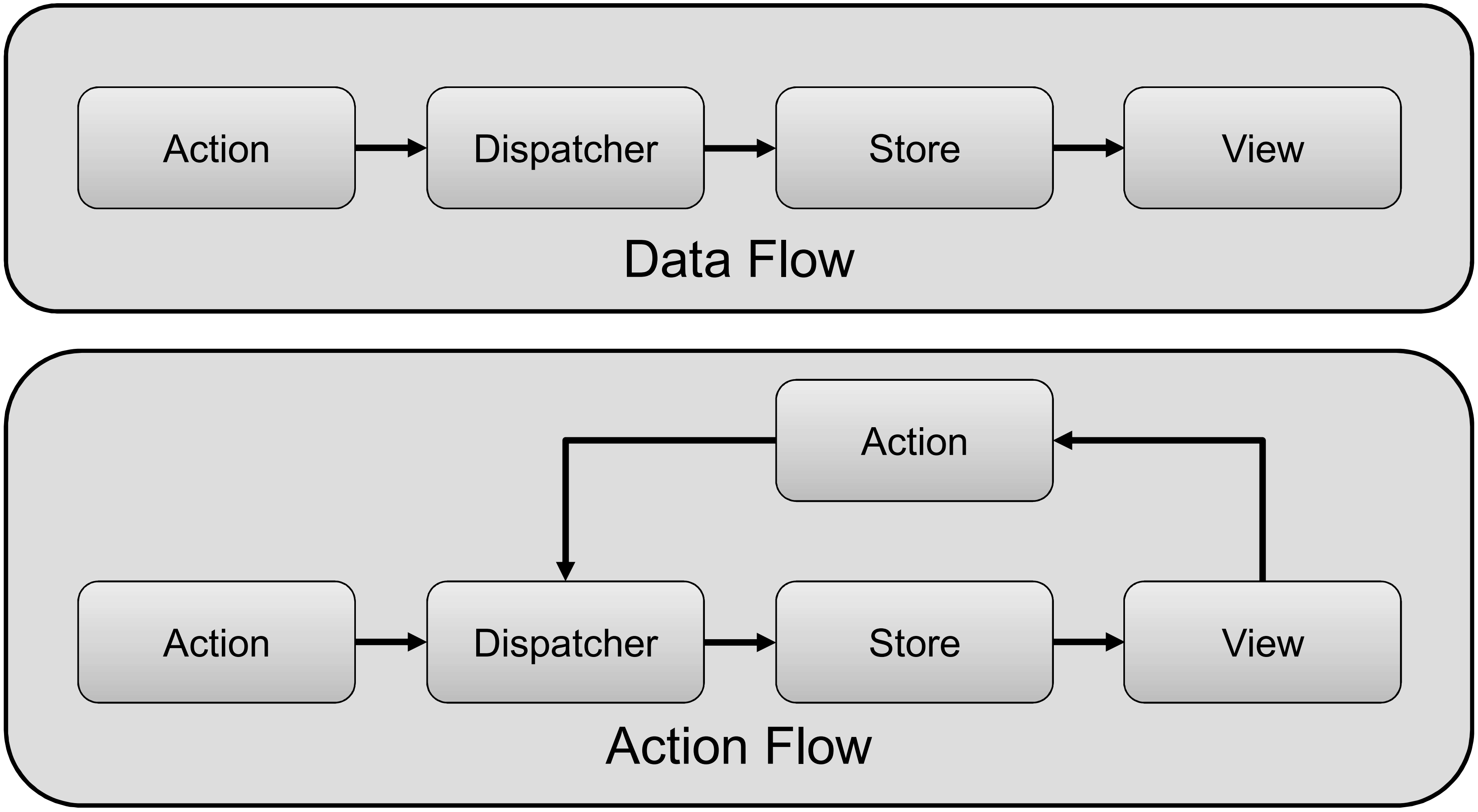
http://www.learn-angular.org/

http://learnangular2.com/

How does React fit MVC?



Flux Model



$$\begin{tikzcd}\text{Action} \arrow[r] & \text{Dispatcher} \arrow[r] & \text{Store} \arrow[r] & \text{View}\end{tikzcd}$$

$$\begin{tikzcd} & & \text{Action} \arrow[dl] & \\ \text{Action} \arrow[r] & \text{Dispatcher} \arrow[r] & \text{Store} \arrow[r] & \text{View} \arrow[ul]\end{tikzcd}$$

React Components

```
// Create a component name MessageComponent
var MessageComponent = React.createClass({
  render: function() {
    return  (
      <div>{this.props.message}</div>
    );
  }
});

// Render an instance of MessageCoponent into document body
ReactDOM.render(
  <MessageComponent message="Hello!" />
  document.body
);
```

React Components

```
// Create a component name MessageComponent
var MessageComponent = React.createClass({
  render: function() {
    return  (
      <div>{this.props.message}</div>
    );
  }
});

// Render an instance of MessageCoponent into document body
ReactDOM.render(
  <MessageComponent message="Hello!" />
  document.body
);
```

What is JSX?

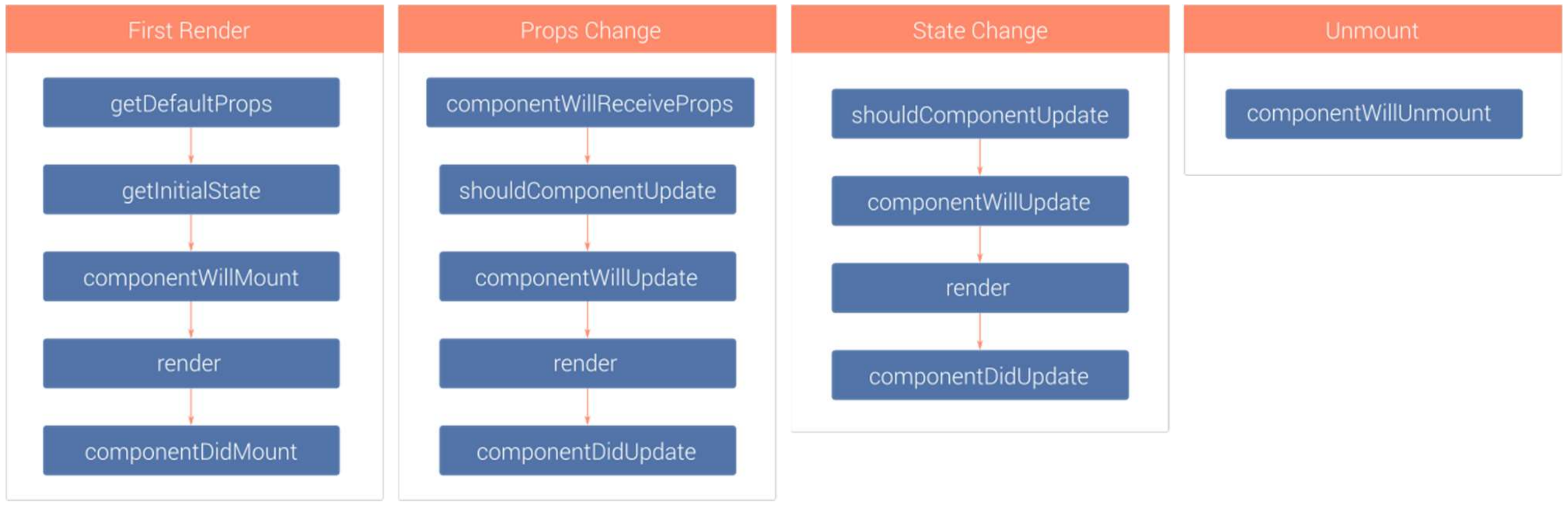
React Components

```
// Create a component name MessageComponent
var MessageComponent = React.createClass({
  render: function() {
    return  (
      <div>{this.props.message}</div>
    );
  }
});
```

What is JSX?

```
// Render an instance of MessageCoponent into document body
ReactDOM.render(
  <MessageComponent message="Hello!" />
  document.body
);
```

React


First Render
getDefaultProps
getInitialState
componentWillMount
render
componentDidMount
Props Change
componentWillReceiveProps
shouldComponentUpdate
componentWillUpdate
render
componentDidUpdate
State Change
shouldComponentUpdate
componentWillUpdate
render
componentDidUpdate
Unmount
componentWillUnmount

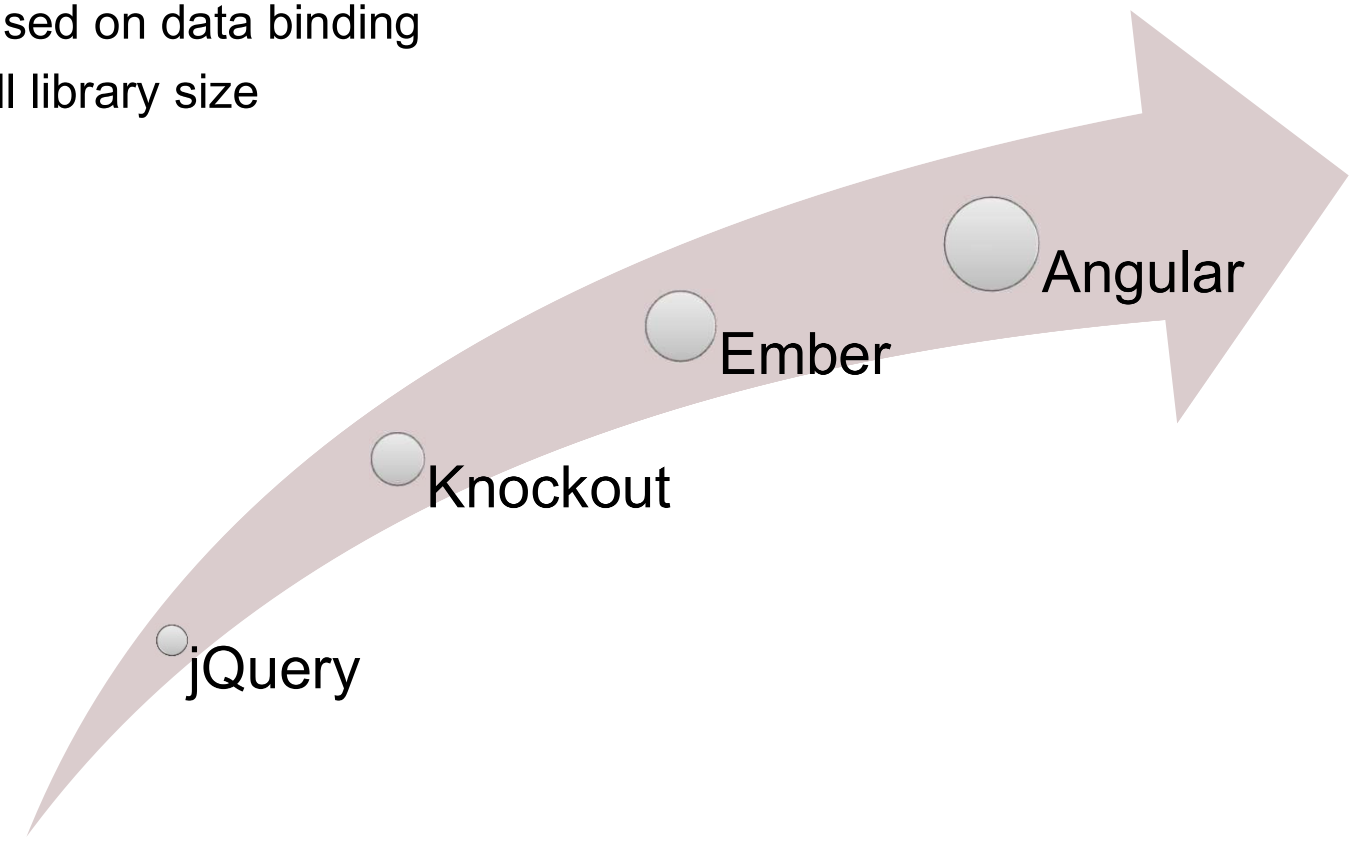
Learn React

https://www.codecademy.com/lrn/react-101

https://css-tricks.com/learning-react-redux/

Intro to Knockout

- An MVVM library
- Automatic UI refresh and updates
- Reusable templates
- Can be used with nearly any framework
- Focused on data binding
- Small library size

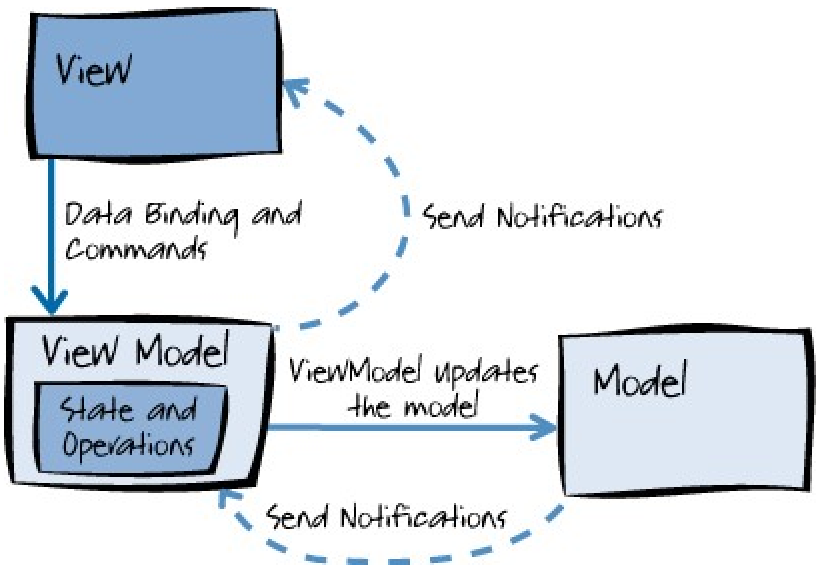


MVVM (Model, View, View-Model)

View
- Defines structure and layout of UI

Model
- Domain Model
- Data Model
- Business logic

View Model
- Intermediary between M/V
- Handles View Logic
- Deals with State Change



Learn Knockout

http://learn.knockoutjs.com/#/?tutorial=intro