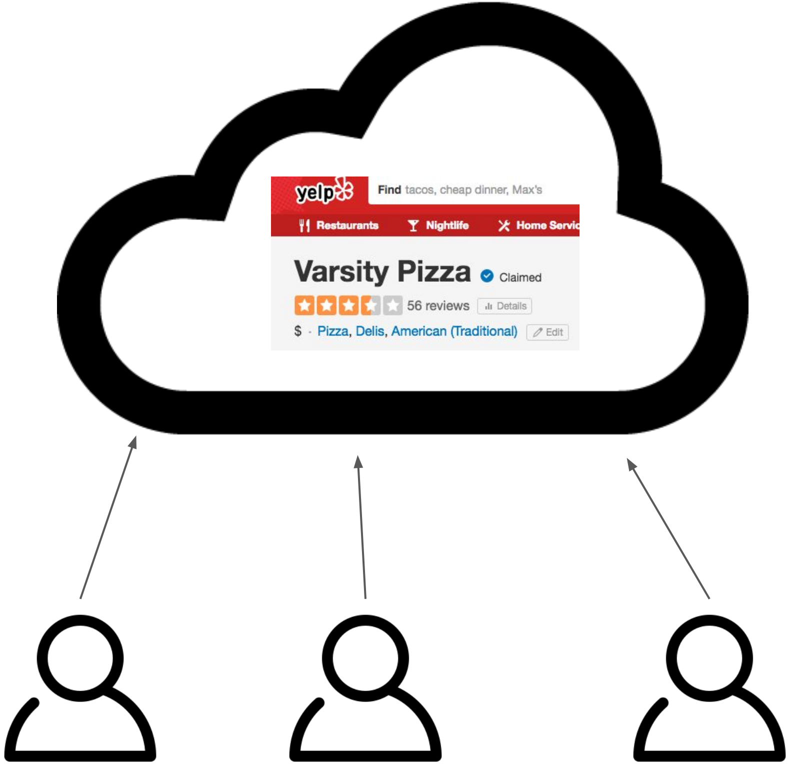
Secret Voting

Using MEAN Stack

Ju Chen, D.O. final project (Spring 2017)

Motivation - Online Voting



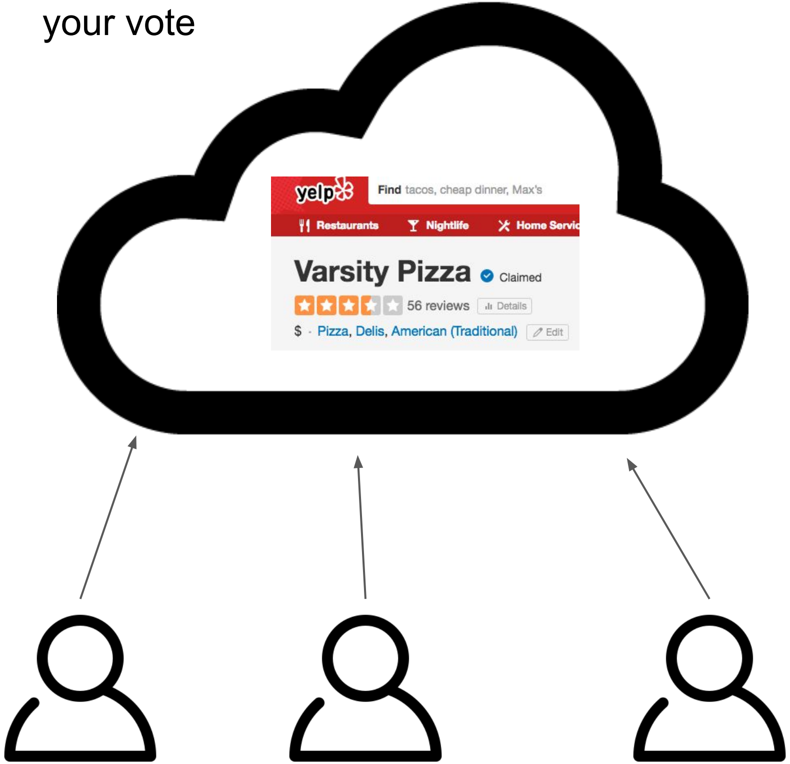
Varsity Pizza Claimed

56 reviews

$ · Pizza, Delis, American (Traditional)

Motivation - Your personal preferences should keep private

Learn you from your vote



yelp Find tacos, cheap dinner, Max's

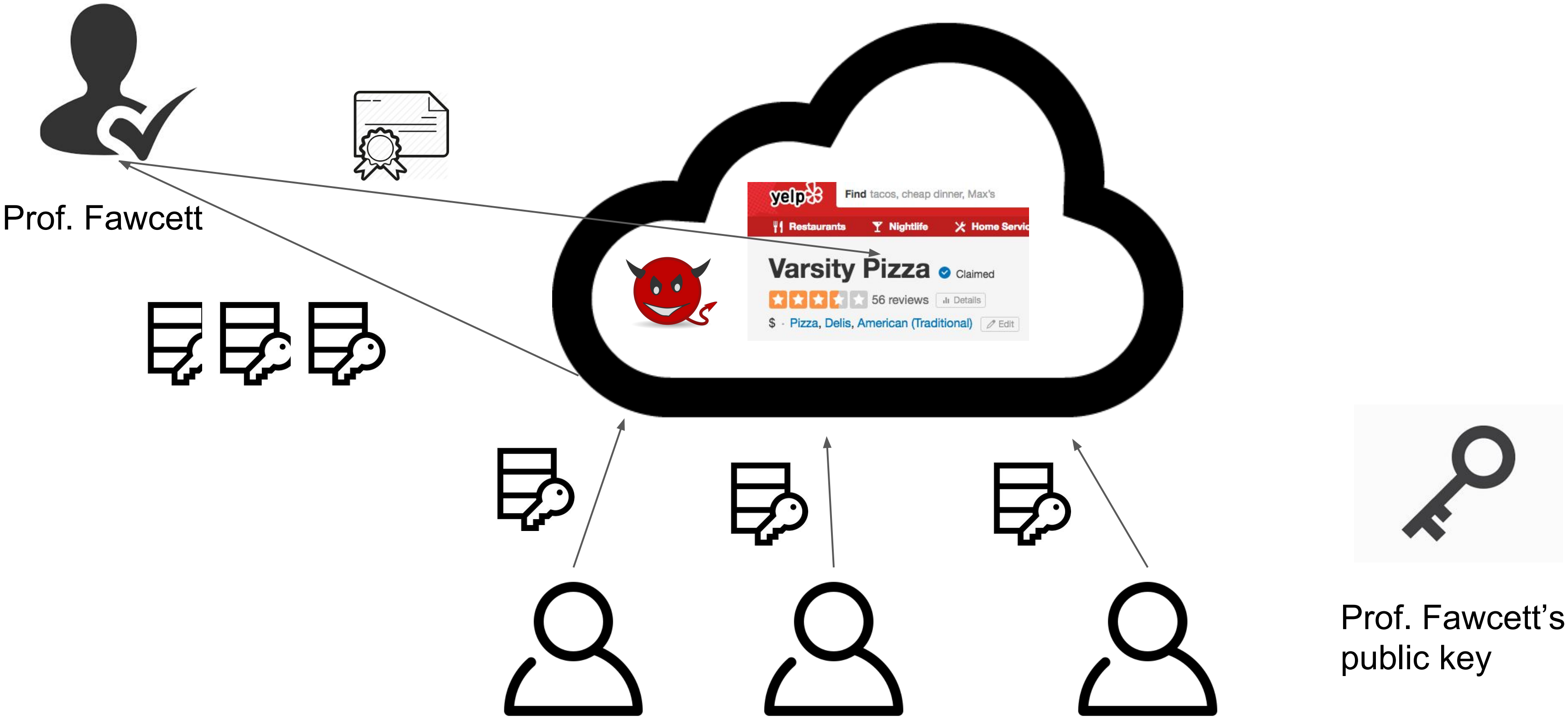
Restaurants Nightlife Home Servic

Varsity Pizza Claimed

★★★★☆ 56 reviews Details

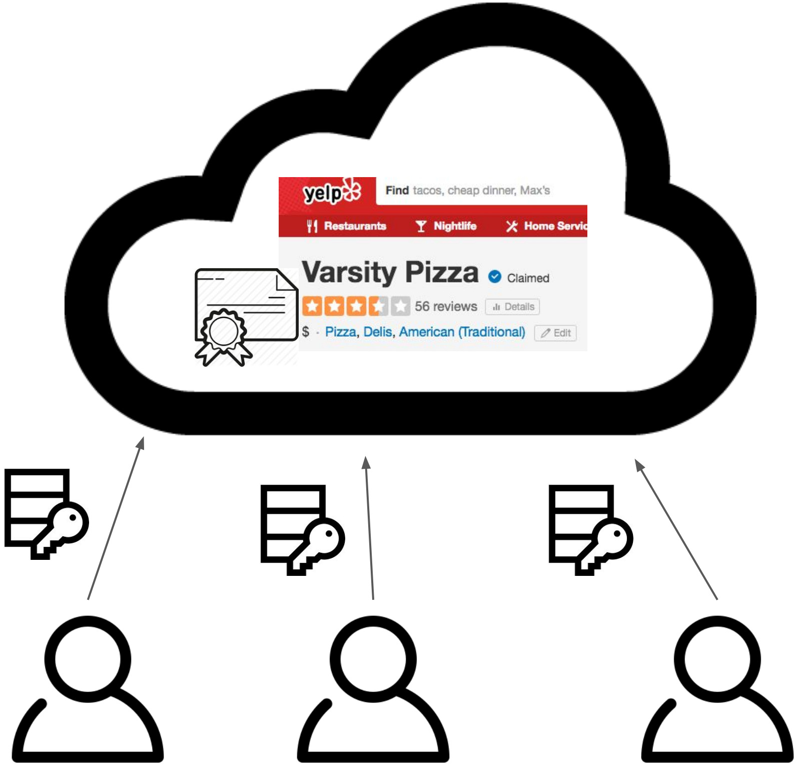
$ · Pizza, Delis, American (Traditional) Edit

Idea - Encrypt, Offload and Sign



Prof. Fawcett
yelp
Find tacos, cheap dinner, Max's
Restaurants
Nightlife
Home Servic
Varsity Pizza
Claimed
56 reviews
Details
$ · Pizza, Delis, American (Traditional)
Edit
Prof. Fawcett's public key

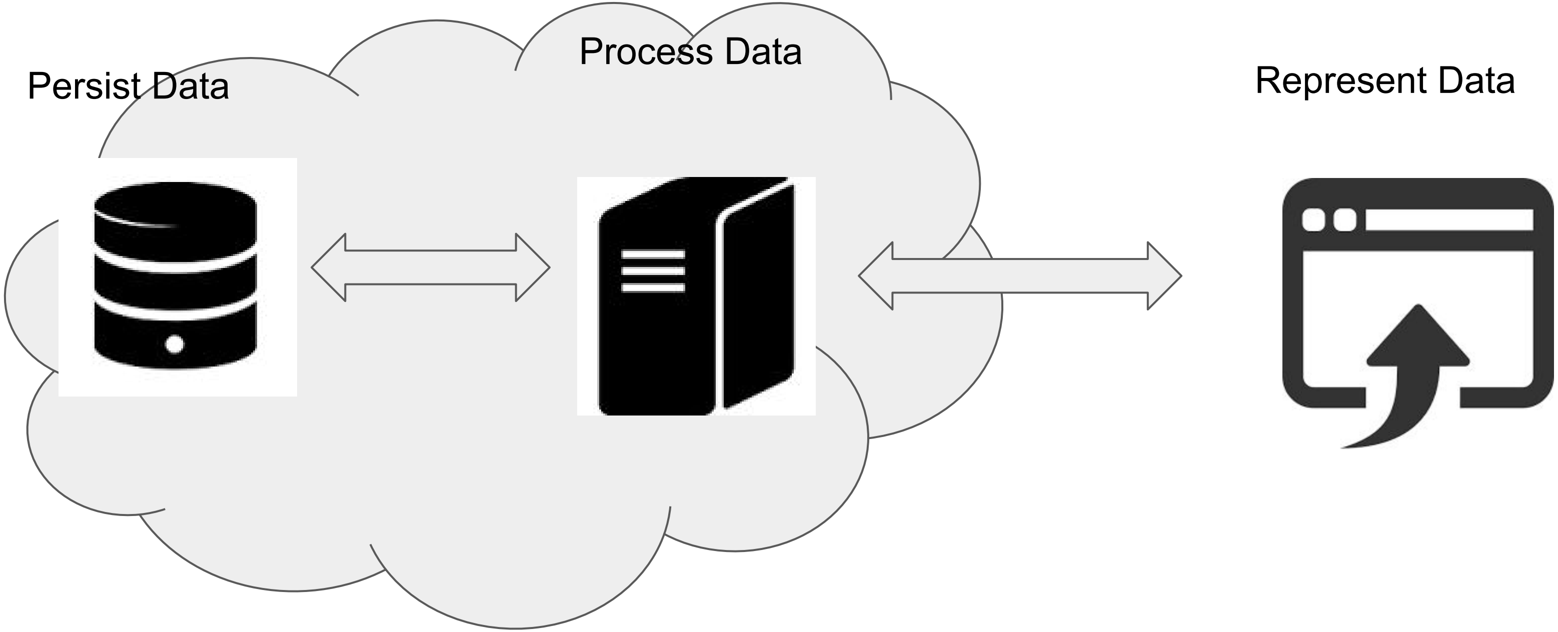
Idea - Encrypt, Offload and Sign



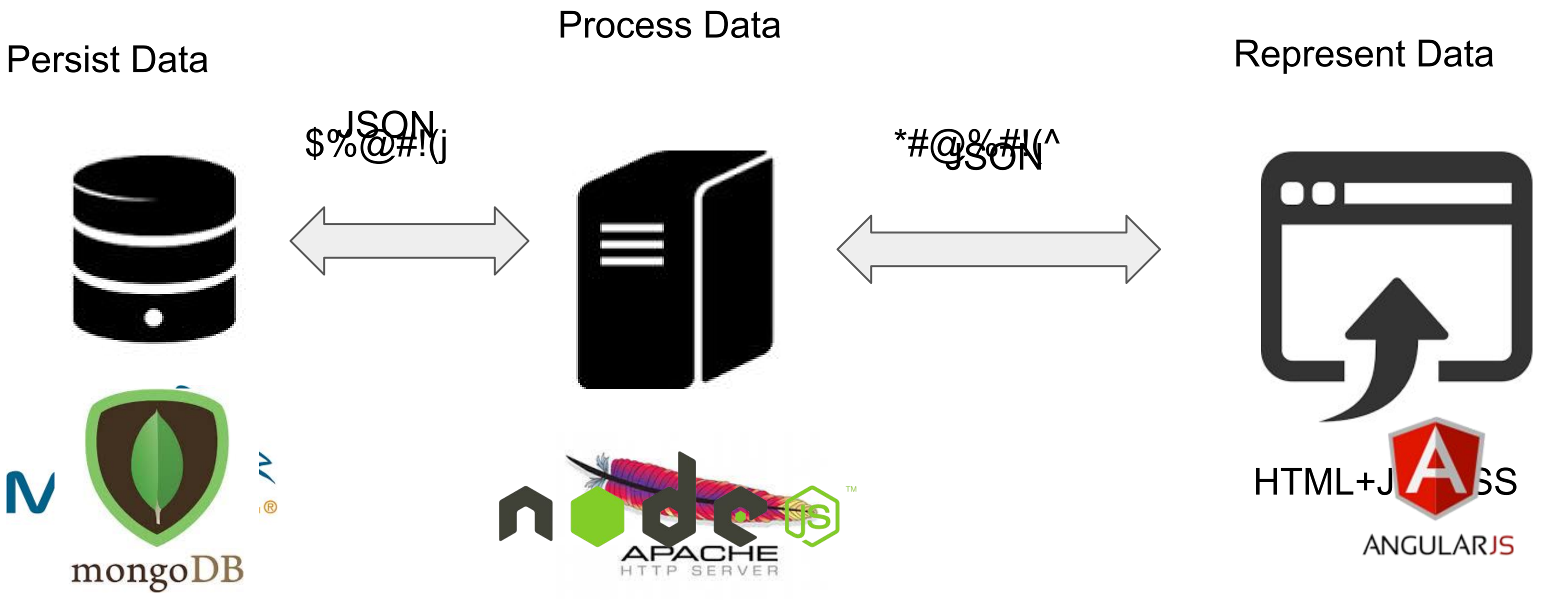
Project Deliverables

- A small voting website doing voting
  - User Login
  - Roles (Admin, Voter, Scrutineer)
  - Trusted-peer publish its Public Key
  - Anyone can initialize a vote event
  - Vote and Submit (Using openssl lib to do the encryption)
  - Trusted-peer receives request from server to count the votes
  - Trusted-peer submits and signs the results. The result is displayed when the event is completed
- And A personal website
  - Story page
  - Code Repository
  - Profile

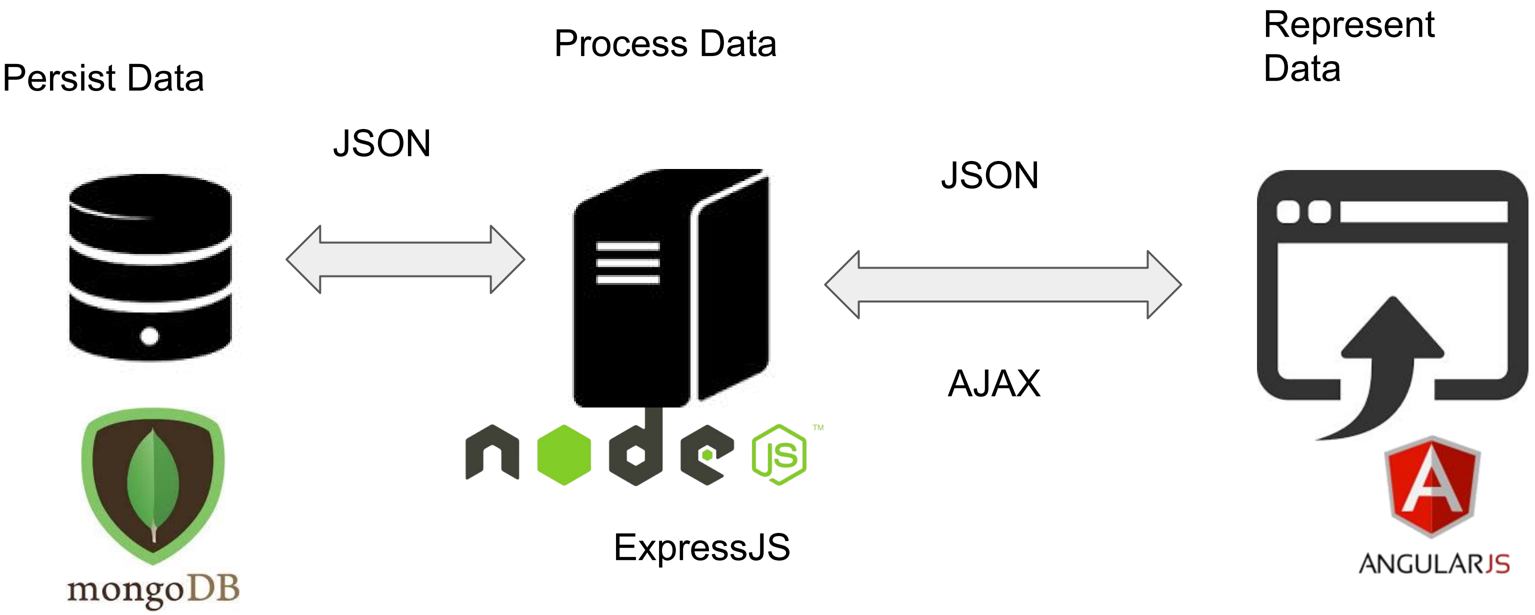
BS software revisit



BS software revisit (LAMP stack vs MEAN stack)



Mean stack (One Language, One Data Type)



Persist Data

JSON

Process Data

JSON

AJAX

Represent Data

ExpressJS

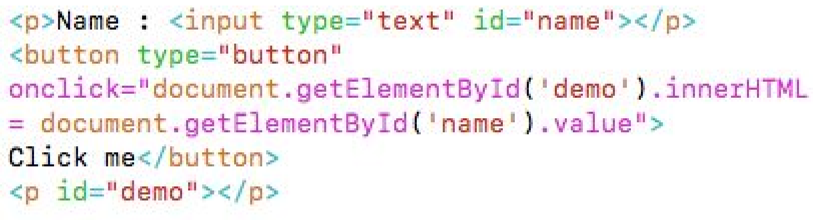
AngularJS - A front end JavaScript Framework

Example:

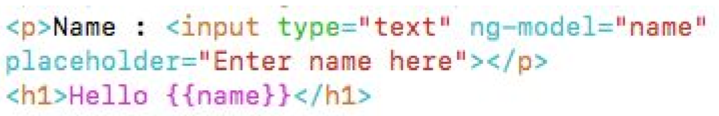
http://localhost:8000/demo1.html

http://localhost:8000/demo2.html

JavaScript vs AngularJS



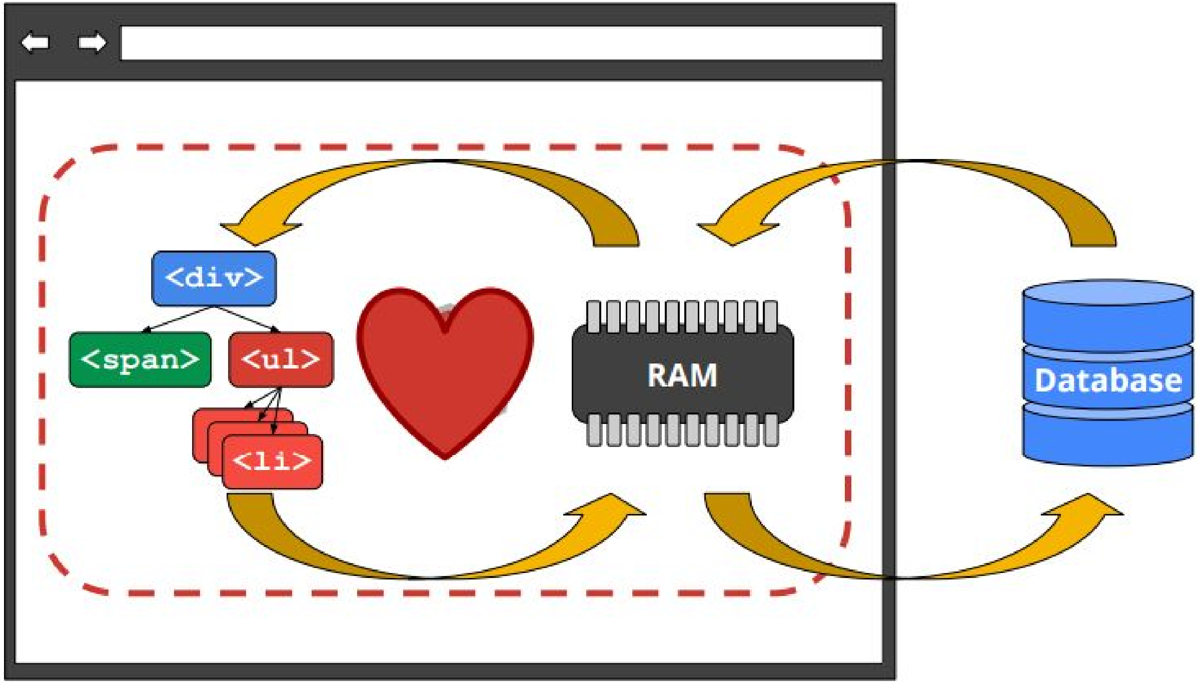
```
<p>Name : <input type="text" id="name"></p>
<button type="button"
onclick="document.getElementById('demo').innerHTML
= document.getElementById('name').value">
Click me</button>
<p id="demo"></p>
```



```
<p>Name : <input type="text" ng-model="name"
placeholder="Enter name here"></p>
<h1>Hello {{name}}</h1>
```

	JavaScript	AngularJS
How does it work	**CODE**	**TAGS**
Program Paradigm	**How** (imperative)	**What** (declarative)
DOM Manipulation	**Yes**	**No**
How is data circulated	**Accessing DOM objects**	**Data Binding**

What is really happening?



Source:http://commondatastorage.googleapis.com/io-2013/presentations/232%20-%20Google%20I_O%20232-%20Design%20Principles%20of%20AngularJS.pdf

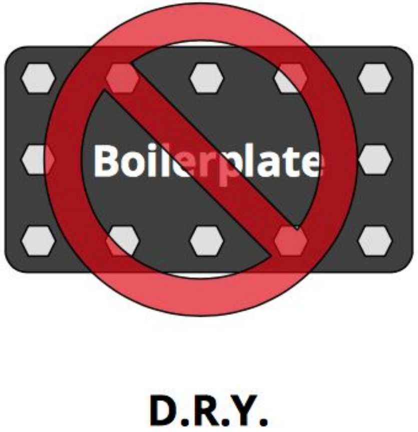
So what is the big deal?

Example:

http://localhost:8000/demo3.html

What are the problems?

- Business logic is mixed with representation (Less maintainability, readability)
- Lots of repetition code (Less productivity, Less maintainability)



Source:http://commondatastorage.googleapis.com/io-2013/presentations/232%20-%20Google%20I_O%20232-%20Design%20Principles%20of%20AngularJS.pdf

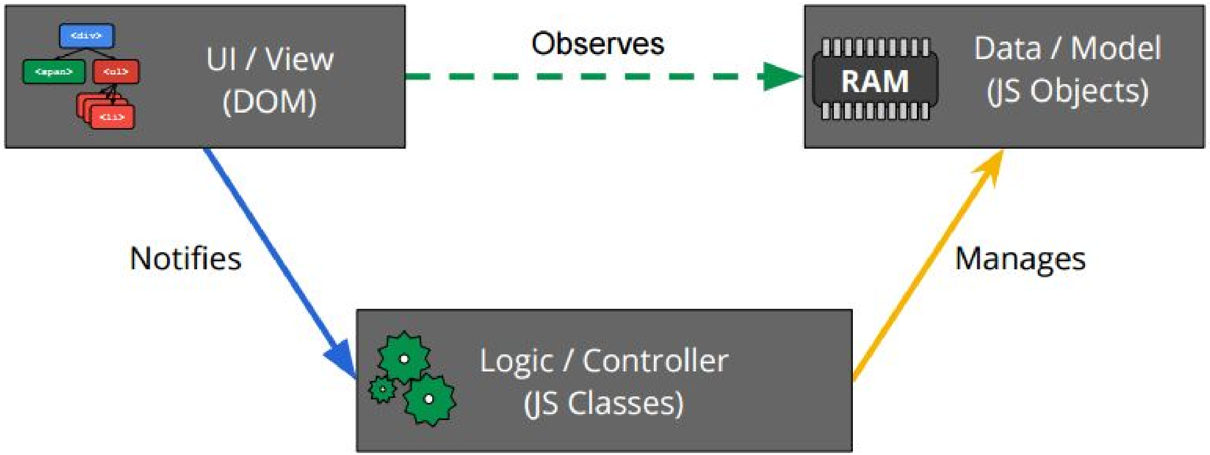
So, how angular solves the problem?

Example:

http://localhost:8000/demo5.html

Decoupling Representation, Data and Logic

Structure



Source:http://commondatastorage.googleapis.com/io-2013/presentations/232%20-%20Google%20I_O%20232-%20Design%20Principles%20of%20AngularJS.pdf

Dependency Injection

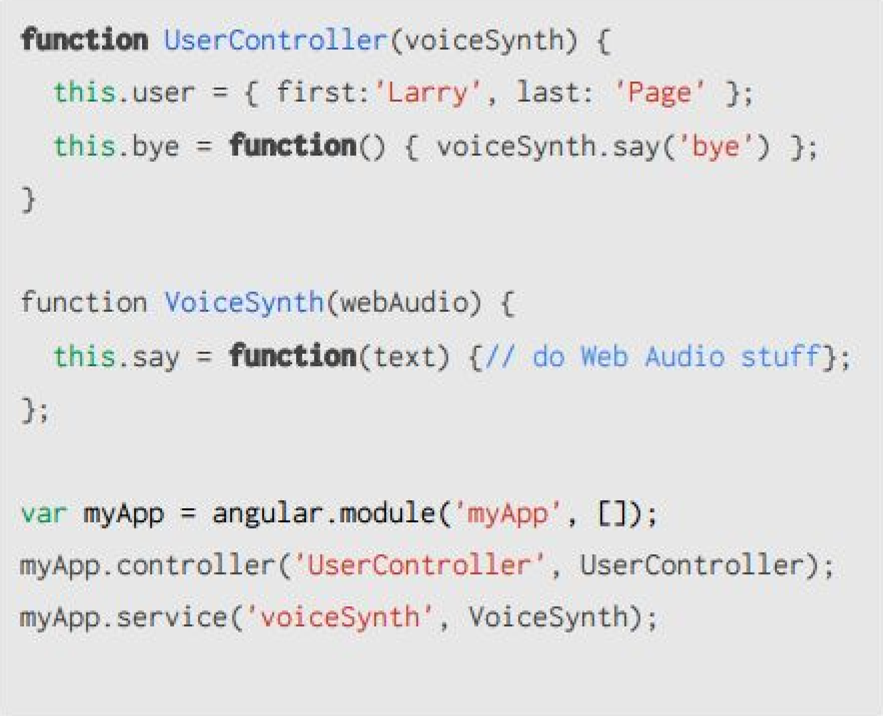
```
public class client {
    private ServiceExample service;
    client () {
        service = new ServiceExample();
    }
    public  String greet() {
        return “Hello”+service.getName();
    }
}
```

☹

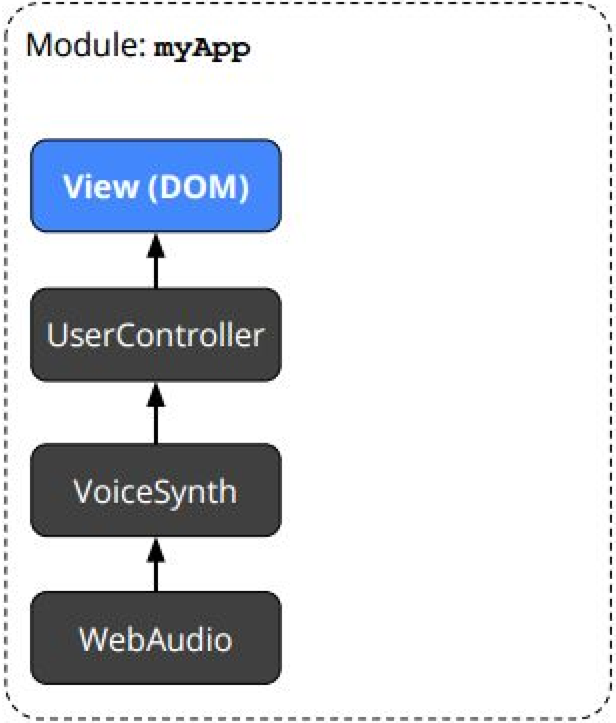
```
Client (Service service) {
    this.service = service;
}
```

☺

Dependency Injection (Example)

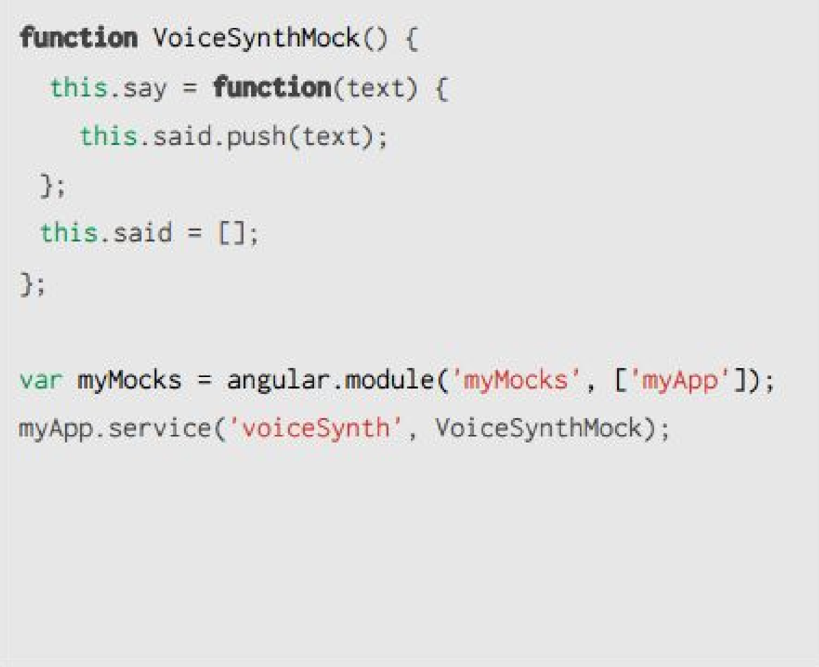


```
function UserController(voiceSynth) {
  this.user = { first:'Larry', last: 'Page' };
  this.bye = function() { voiceSynth.say('bye') };
}

function VoiceSynth(webAudio) {
  this.say = function(text) {// do Web Audio stuff};
};

var myApp = angular.module('myApp', []);
myApp.controller('UserController', UserController);
myApp.service('voiceSynth', VoiceSynth);
```

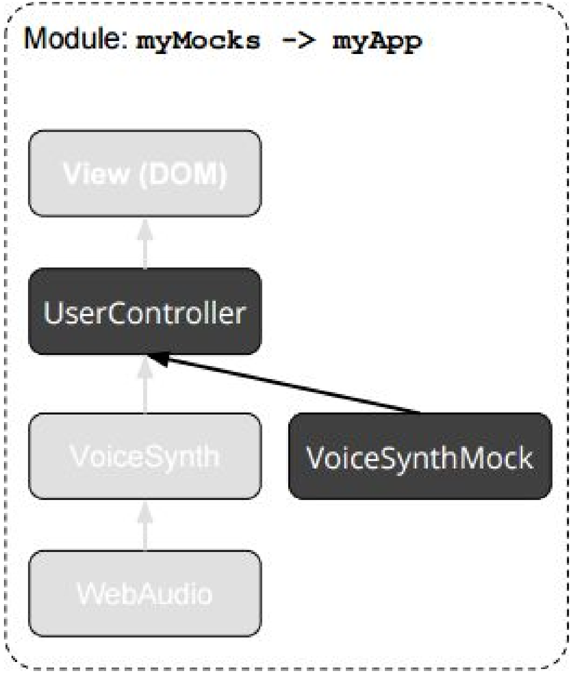


Source:http://commondatastorage.googleapis.com/io-2013/presentations/232%20-%20Google%20I_O%20232-%20Design%20Principles%20of%20AngularJS.pdf

Dependency Injection is good for testing



```
function VoiceSynthMock() {
  this.say = function(text) {
    this.said.push(text);
 };
 this.said = [];
};

var myMocks = angular.module('myMocks', ['myApp']);
myApp.service('voiceSynth', VoiceSynthMock);
```

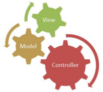


Source:http://commondatastorage.googleapis.com/io-2013/presentations/232%20-%20Google%20I_O%20232-%20Design%20Principles%20of%20AngularJS.pdf

AngularJS Conclusion

- No boilerplate

- Better structure



- Testability

Project Plan

- Set up a nutshell website using full-stack of MEAN (almost done)
- Encryption/Decryption and interaction with MEAN (on-going)
- Design data storage (on-going)
  - People (roles, credentials)
  - Event
  - Vote
  - Picture,Text and Code
- Design WebUI (on-going)
  - Login
  - Initiate Event
  - All the event
  - Vote
  - Vote count request
  - Result submission
- Implement

Thank you!