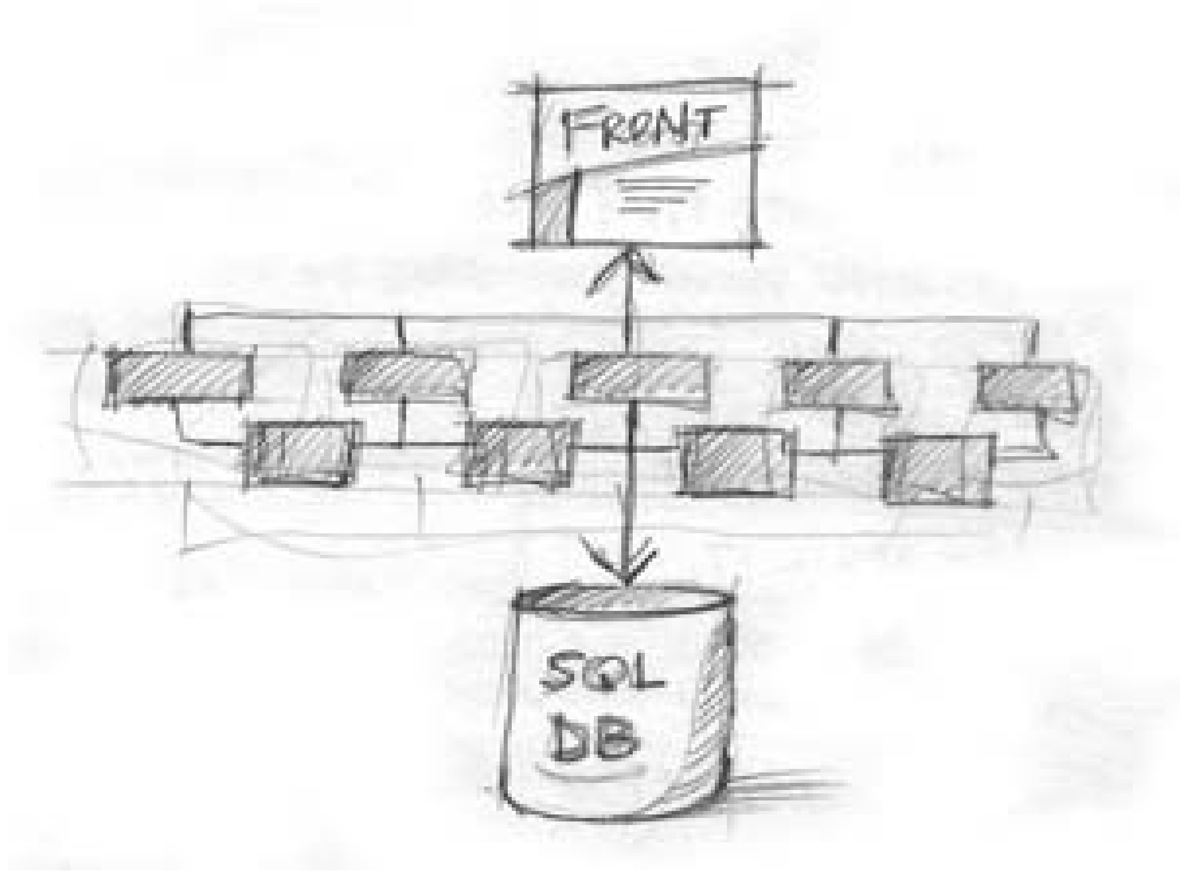
Vitaliy Basyuk aka Wolfgang Bas
becevka@kucoe.net

Express Nodes on the right Angle

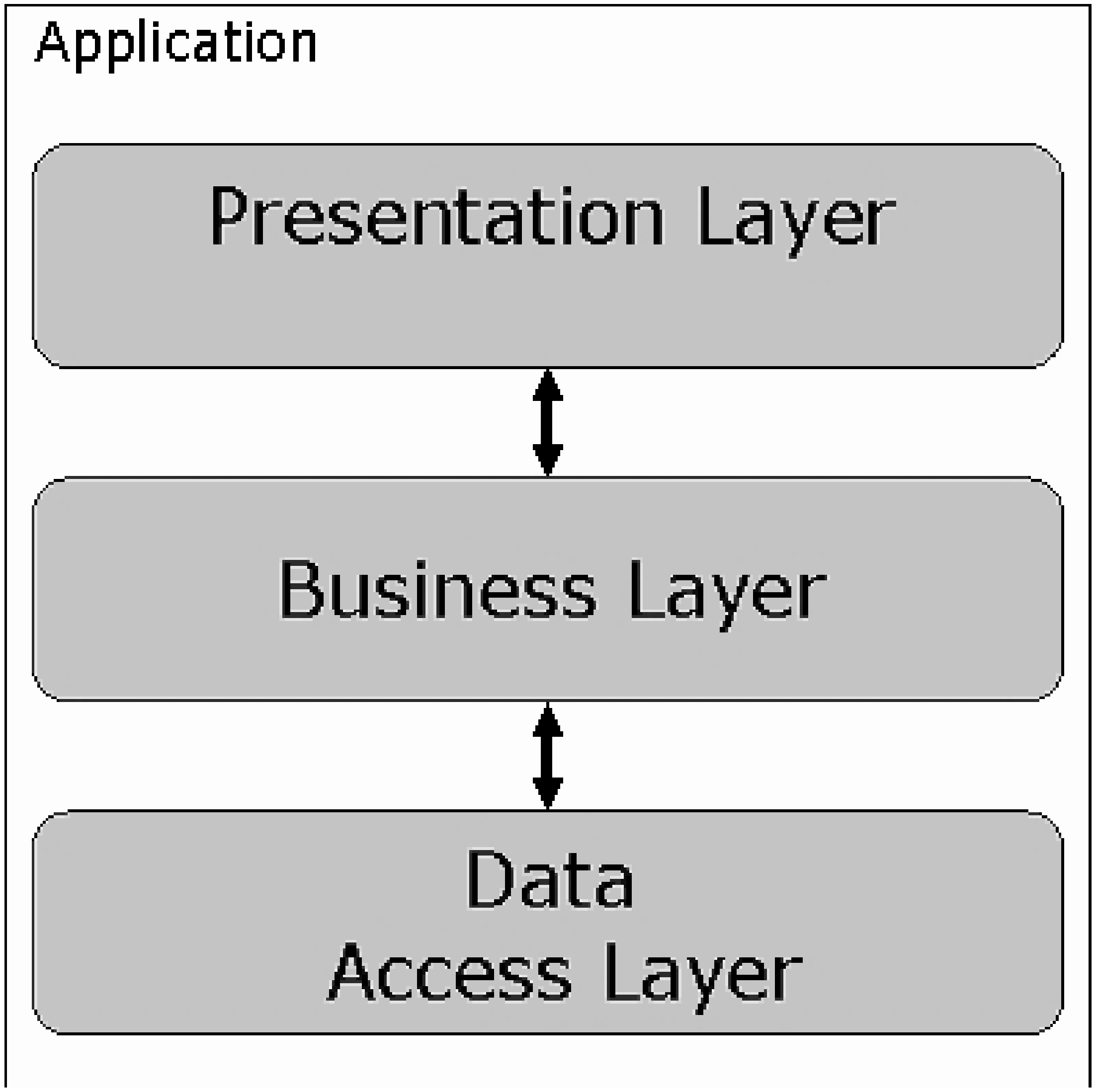
Did you ever have a product idea?

A dream will always triumph over reality, once it is given the chance. **Stanislaw Lem**

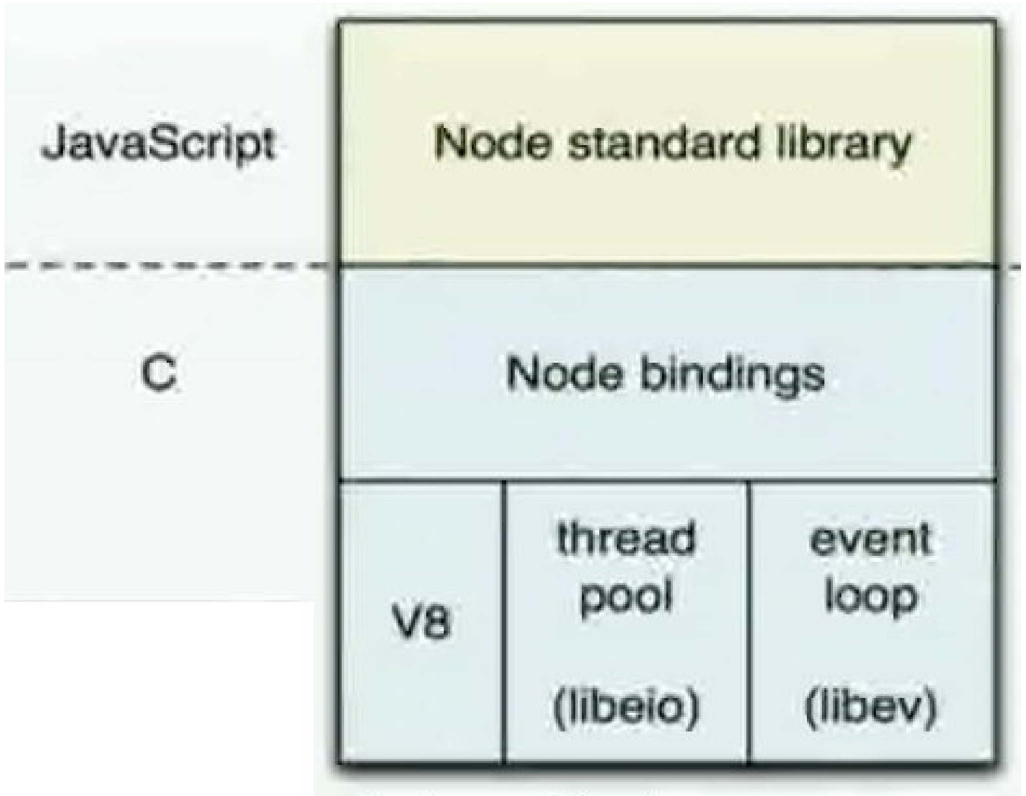


birkey.com

My way on finding technologies which help not to lose excitement of my idea and not let me sink in the sea of boilerplate code.


Application
Presentation Layer
Business Layer
Data
Access Layer

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.



node.js architecture

The basic philosophy of node.js is

- Non-blocking I/O

  Every I/O call must take a callback, whether it is to retrieve information from disk, network or another process

- Built-in support for the most important protocols

  HTTP, DNS, TLS

- Low-level

  Does not remove functionality present at the POSIX layer. For example, support half-closed TCP connections.

- Stream everything

  Never force the buffering of data

Code

```
fs.readFile('/etc/passwd', function(err, data){
   console.log(data);
});


var http = require('http');
http.createServer(function (req, res) {
   res.writeHead(200, {'Content-Type': 'text/plain'});
   res.end('Hello World\n');
}).listen(8081);
console.log('Server running at port 8081');
```

Server routing

```
var http = require('http');
http.createServer(function (req, res) {
var path = url.parse(req.url).pathname;
var hello = {message: "Hello, world"};
    switch (path) {
    case '/json':
        res.writeHead(200, {'Content-Type':'application/json; charset=UTF-
   8'});
        res.end(JSON.stringify(hello));
        break;

    default:
        res.writeHead(404, {'Content-Type': 'text/html; charset=UTF-8'});
        res.end('Sorry, we cannot find that!');
    }
}).listen(8081);
```

Express

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
    res.send('Hello World');
});
app.listen(8081);

var express = require('express');
var app = express();
var hello = {message: "Hello, world"};
app.get('/json', function(req, res){
    res.json(hello);
});
app.all('*', function(req, res){
    res.send(404, 'Sorry, we cannot find that!');
});
app.listen(8081);
```

What Express does?

- Processing request parameters and headers
- Routing
- Rendering response and views support
- Application configuration and middleware support

  session, CSRF, basicAuth, vhost, static content, custom

Express example

```
var express = require('express'),
    routes = require('./routes'),
    api = require('./routes/api');

var app = express();

app.configure(function(){
    app.set('views', __dirname + '/views');
    app.set('view engine', 'jade');
    app.use(express.bodyParser());
    app.use(express.methodOverride());
    app.use(express.static(__dirname + '/public'));
    app.use(app.router);
});
app.configure('development', function(){
    app.use(express.errorHandler({ dumpExceptions: true,
showStack: true }));
});
app.configure('production', function(){
    app.use(express.errorHandler());
});
```

Express example (continue)

```
 app.get('/', routes.index);
 //  res.render('index');

 app.get('/posts', api.posts);
 app.get('/posts/:id', api.post);
 //var id = req.params.id;
 app.post('/posts', api.addPost);
 //res.json(req.body);
 app.put('/posts/:id', api.editPost);
 app.delete('/posts/:id', api.deletePost);

 app.get('*', routes.index);
    app.listen(3000, function(){
    console.log("Express server listening on port %d in %s
mode", app.address().port, app.settings.env);
 });
```

Node.js pros

- Known language (JavaScript)
- Additional API are documented well and understandable
- Effortless building of REST services with Express
- No server configuration, easy to install and run
- Real-time application support
- Quite big community and a lot of tools implemented

Node.js cons

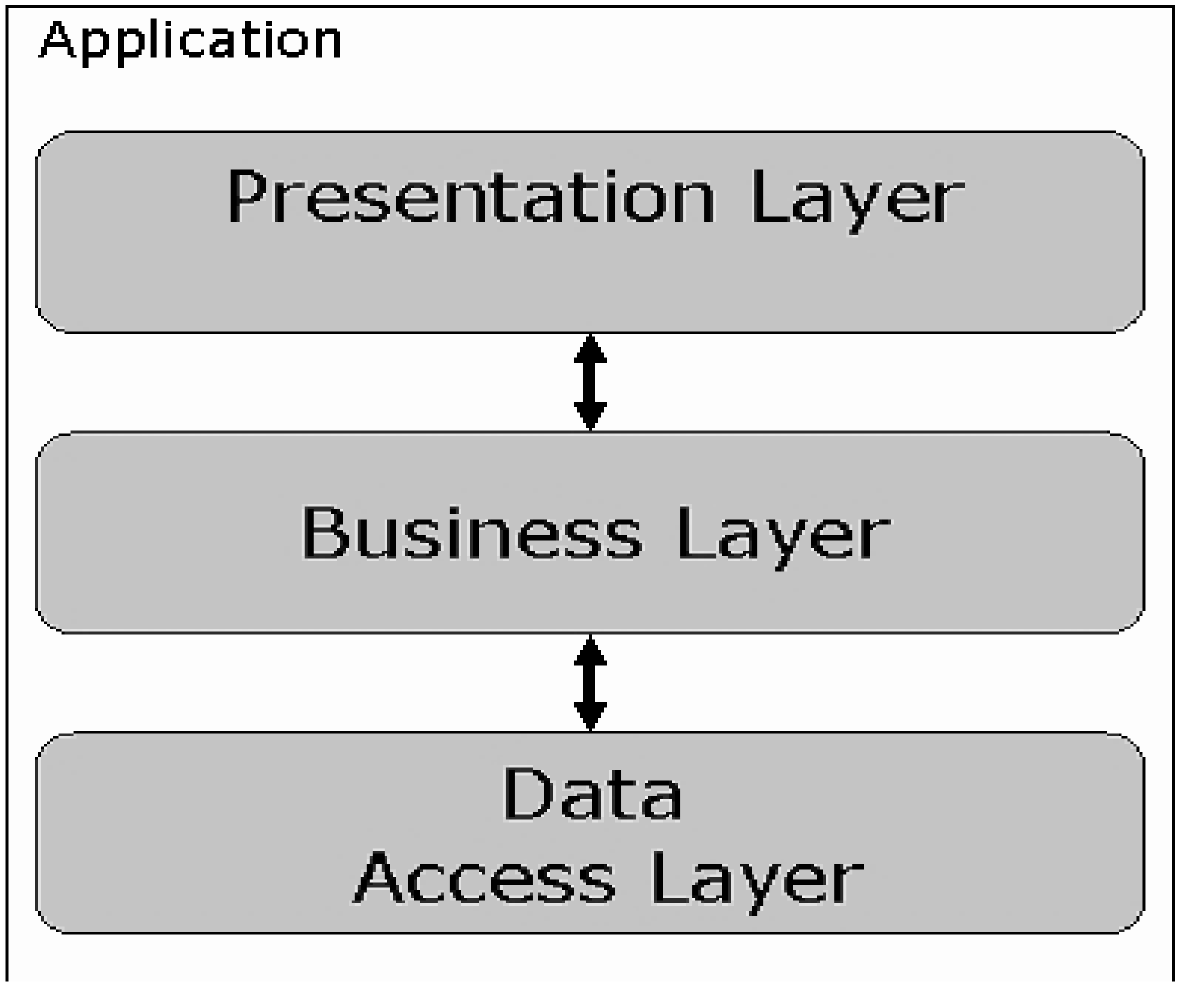
- Not mature enough and some API are not stable
- Running on production server might differ
- Not enough tools and libraries
- Not for a CPU consuming routines
- Async code looks not so pretty

Further reading

http://nodejs.org/api/ - Node.js official API
http://book.mixu.net/ - Mixu Node book
http://ofps.oreilly.com/titles/9781449398583/index.html - Node Up and Running (has chapter on Express)
http://expressjs.com/api.html - Express API

Useful modules

http://socket.io/ - for realtime apps
https://github.com/visionmedia/jade - Express template engine
http://visionmedia.github.io/mocha/ - test framework
http://chaijs.com/ - BDD/TDD assertion library


Application
Presentation Layer
Business Layer
Data
Access Layer

MongoDB

Document-Oriented Storage for JSON-style documents with dynamic schemas, full index support and rich document-based queries.

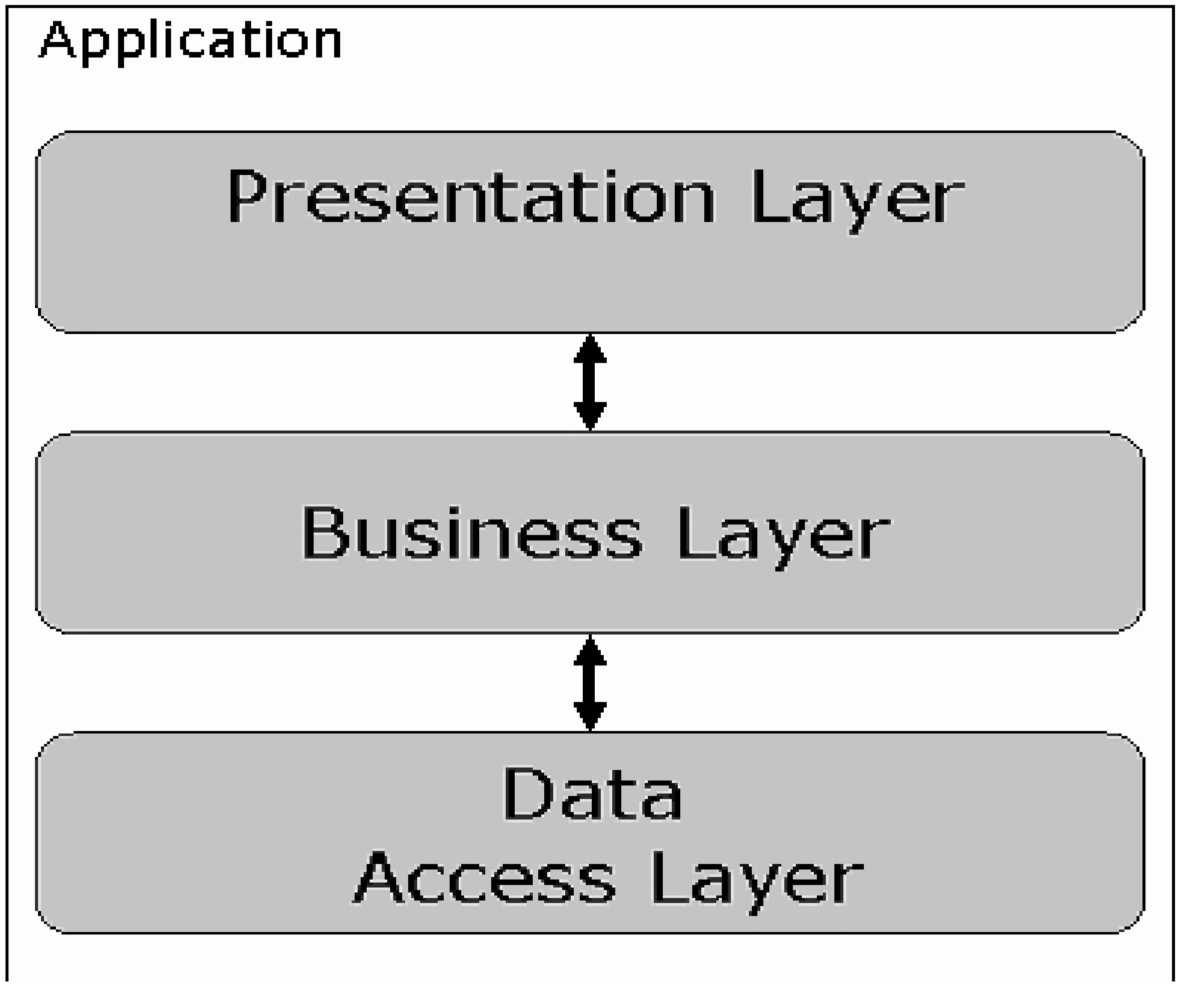
```
db.articles.find({'comments.0.by':'becevka'})
```

Mongoskin

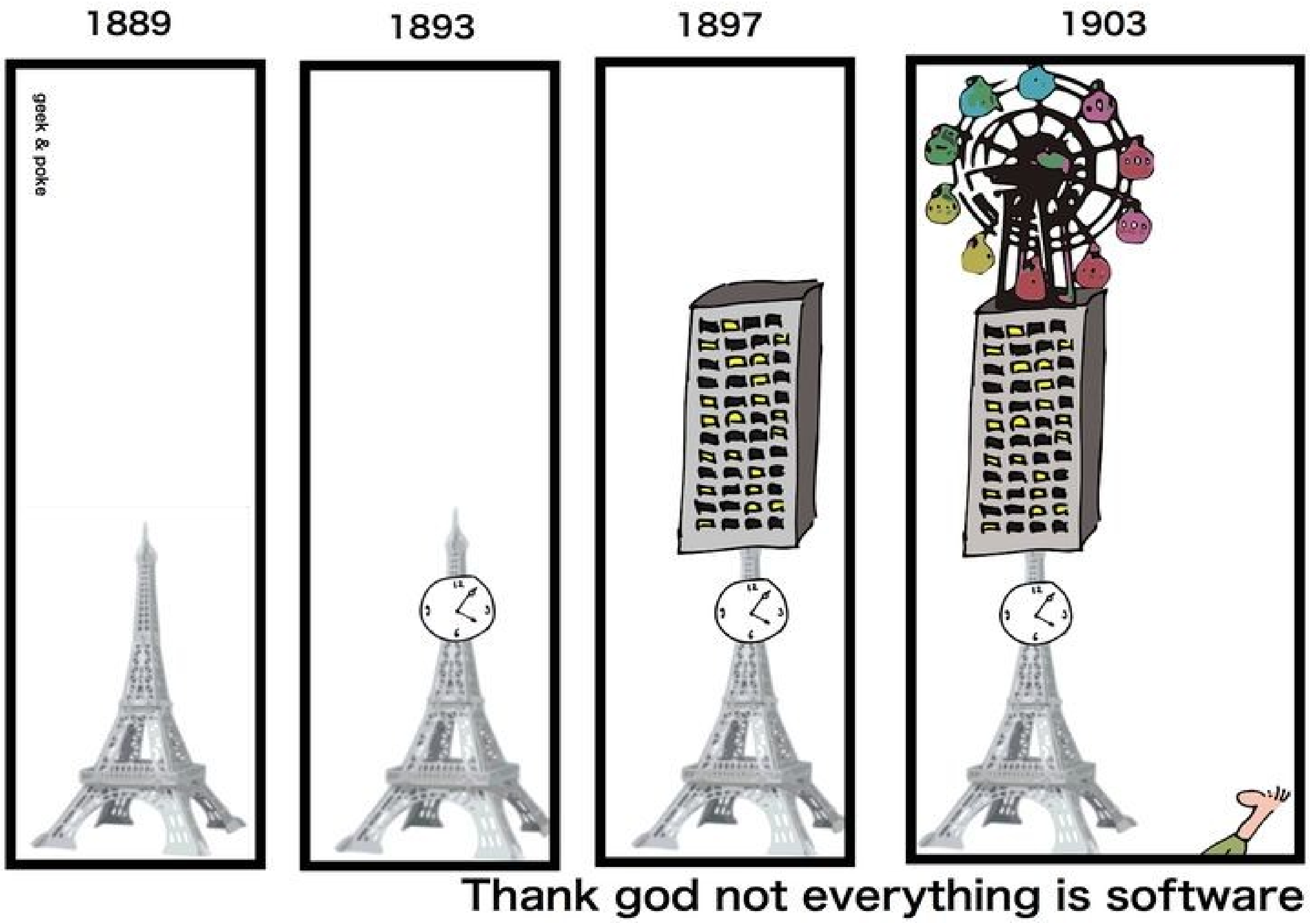
```
var mongo = require('mongoskin');
var db = mongo.db('localhost:27017/test?auto_reconnect');

db.bind('posts', {
  removeTagWith : function (tag, fn) {
        this.remove({tags:tag},fn);
  }
});

db.posts.removeTagWith('delete', function (err, replies){
  //do something
});
```

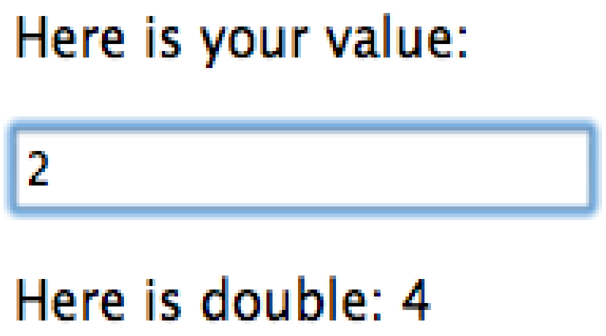

Application
Presentation Layer
Business Layer
Data
Access Layer

Tim (Berners-Lee) bawled me out in the summer of '93 for adding images to the thing **Marc Andreesen, Mosaic**



Requirements for UI framework

- Declarative, simple way to describe our presentation, how it looks and how it lays out
- Dynamic application out of the box
- No need to extend framework classes in order to make it work
- Easy implemented REST support
- Less code


Here is your value:

2

Here is double: 4

```
<label>Here is your value:</label>
<input type="text" name="value"/>
<span>Here is double: {{value * 2}}</span>
```

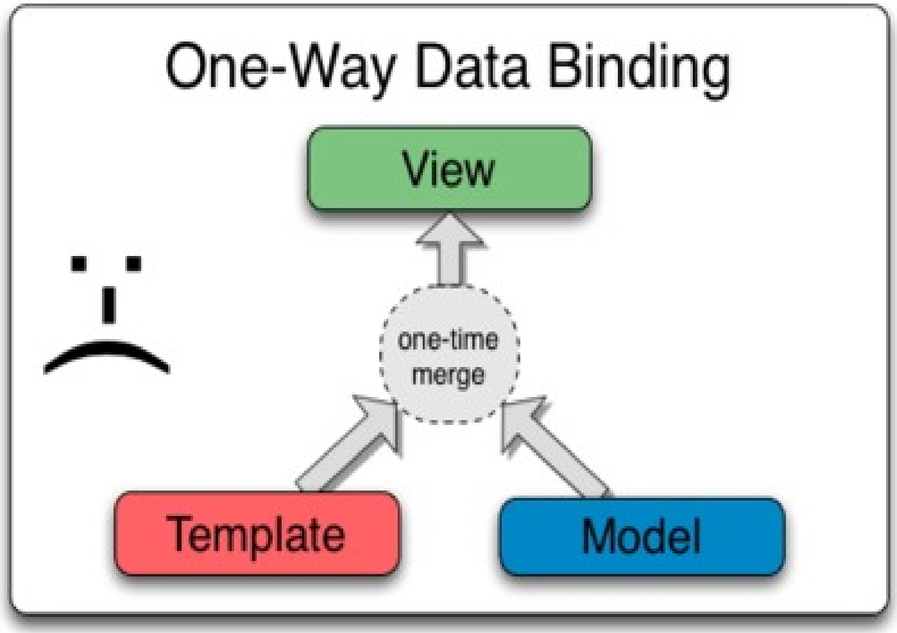
AngularJS

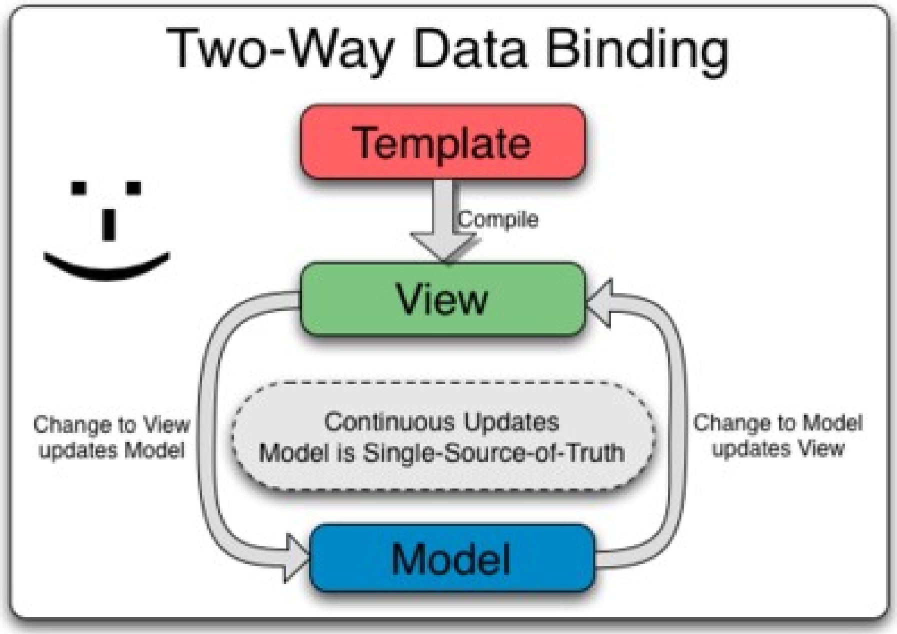
by Google

Core Features of AngularJS

- Two Way Data-binding
- Model View Whatever
- HTML Templates
- Deep Linking
- Dependency Injection
- Directives

Two Way Data-Binding



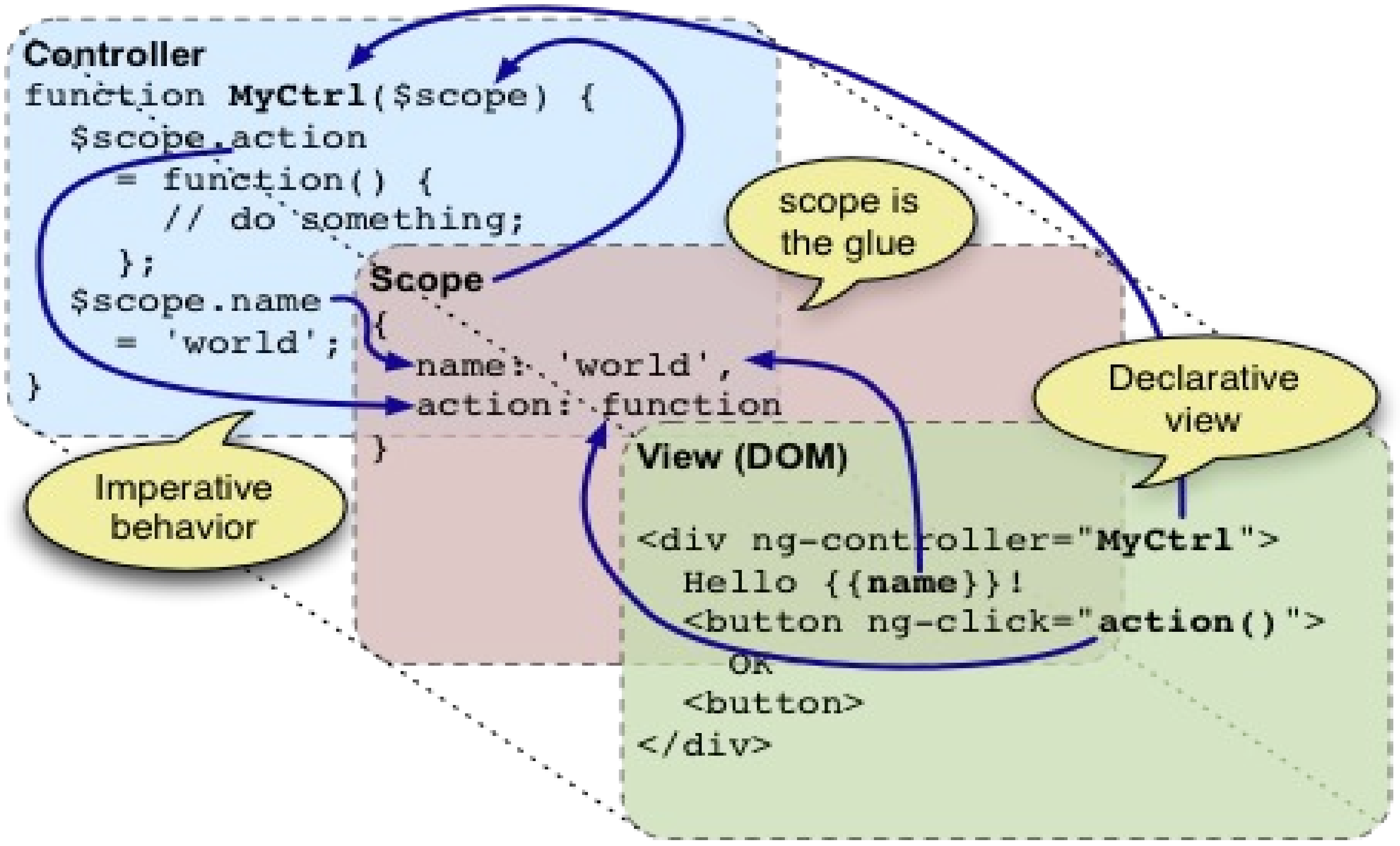

Two-Way Data Binding
Template
Compile
View
Change to View updates Model
Continuous Updates
Model is Single-Source-of-Truth
Change to Model updates View
Model

Two Way Data-Binding

```
<span id="yourName"></span>
document.getElementById('yourName')[0].text = 'bob';
```

```
<span>{{yourName}}</span>
 var yourName = 'bob';
```

Model View Whatever



```
function MyCtrl($scope) {
  $scope.action
    = function() {
      // do something;
    };
  $scope.name
    = 'world';
}
```

```
{
name: 'world',
action: function
}
```

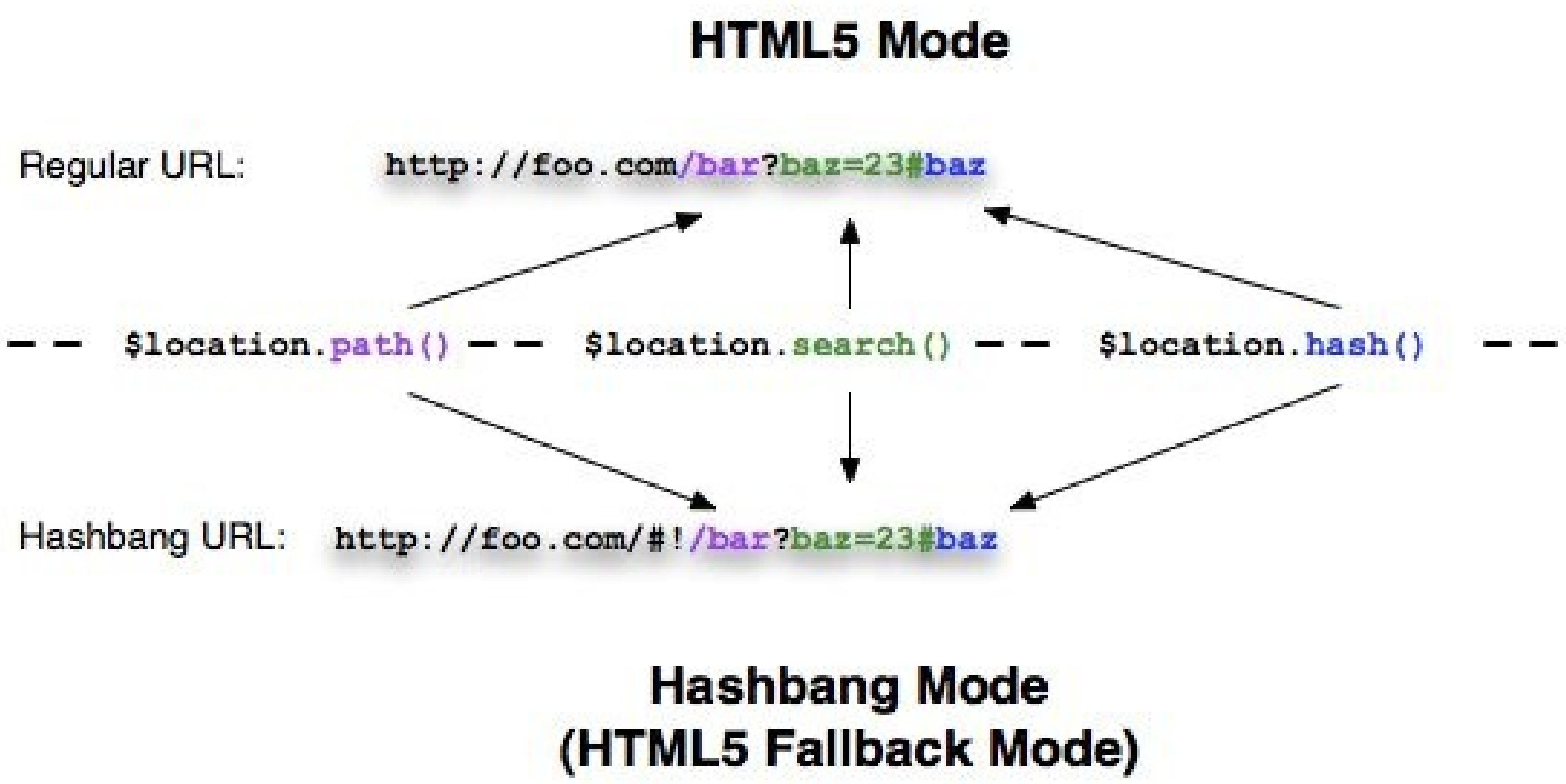
```
<div ng-controller="MyCtrl">
  Hello {{name}}!
  <button ng-click="action()">
    OK
  <button>
</div>
```

HTML Templates

```
<div ng-controller="AlbumCtrl">
    <ul>
    <li ng-repeat="image in images">
            <img ng-src="{{image.thumbnail}}" alt="{{image.description}}">
        </li>
    </ul>
</div>


function AlbumCtrl($scope) {
    scope.images = [
              {"thumbnail":"img/image_01.png", "description":"Image 01
description"},
              {"thumbnail":"img/image_02.png", "description":"Image 02
description"},
              {"thumbnail":"img/image_03.png", "description":"Image 03
description"},
              {"thumbnail":"img/image_04.png", "description":"Image 04
description"},
              {"thumbnail":"img/image_05.png", "description":"Image 05
description"}
     ];
}
```

Deep Linking



Dependency Injection

```
function EditCtrl($scope, $location, $routeParams, User) {
         // Something clever here...
}
```

Directives

```
<div class="container">
  <div class="inner">
    <ul>
      <li>Item
        <div class="subsection">item 2</div>
      </li>
    </ul>
  </div>
</div>
```

```
<dropdown>
   <item>Item 1
      <subitem>Item 2</subitem>
   </item>
</dropdown>
```

Other features

Filters

```
   <td>{{name|uppercase}}</td>
```

Services

```
   app.factory('User', function ($resource) {
      var User = $resource('/users/:_id', {}, {
         update: { method: 'PUT', params: {_id: "@_id" }}
      });
      return User;
   });
```

Routes

```
   app.config(['$routeProvider', function($routeProvider) {
      $routeProvider.
      when('/login', {
         templateUrl: 'partials/login',
         controller: LoginCtrl
      });
   }]);
```

Complete example

```
<html ng-app='myApp'>
<head>
  <title>Your Shopping Cart</title>
</head>
<body ng-controller='CartController'>
    <h1>Your Order</h1>
    <div ng-repeat='item in items'>
        <span>{{item.title}}</span>
        <input ng-model='item.quantity'>
        <span>{{item.price | currency}}</span>
        <span>{{item.price * item.quantity | currency}}</span>
        <button ng-click="remove($index)">Remove</button>
    </div>
    <script src="angular.js"> </script>
    <script>
        angular.module("myApp", []);
        function CartController($scope) {
            $scope.items = [
                {title: 'Paint pots', quantity: 8, price: 3.95},
                {title: 'Polka dots', quantity: 17, price: 12.95},
                {title: 'Pebbles', quantity: 5, price: 6.95}
            ];
            $scope.remove = function(index) {
                $scope.items.splice(index, 1);
            }
        }
    </script>
</body>
</html>
```

Angular Resources

- http://angular-ui.github.io/ - a lot of useful directives and filters
- http://angular-ui.github.io/bootstrap/ - twitter bootstrap directives
- http://angular-ui.github.io/ng-grid/ - grid
- https://github.com/angular/angularjs-batarang - web inspector extension

- http://docs.angularjs.org/ - official documentation
- http://www.egghead.io/ - video tutorials
- http://deansofer.com/posts/view/14/AngularJs-Tips-and-Tricks-UPDATED - Tips and tricks
- http://www.cheatography.com/proloser/cheat-sheets/angularjs/ - Cheat sheets

All together now

Amigo - AngularJS MongoDB Express Node.js project generator. Allows easily scaffold project layout with CRUD operations and UI skeleton.

https://github.com/becevka/amigo

```
$ npm install -g amigo
amigo todo
```

Amigo features

- Resources scaffolding
- MongoDB storage
- Angular Routes
- Angular Resources Factories
- CRUD UI out of the box
- Authentification
- Session support and storing

Demo

Thanks

http://becevka.github.io/amigo/