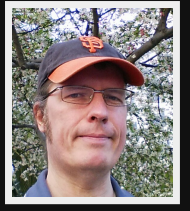
JavaScript Web Frameworks

Oliver Zeigermann / @DJCordhose

Online Version: http://bit.ly/1EicQwV

Oliver Zeigermann

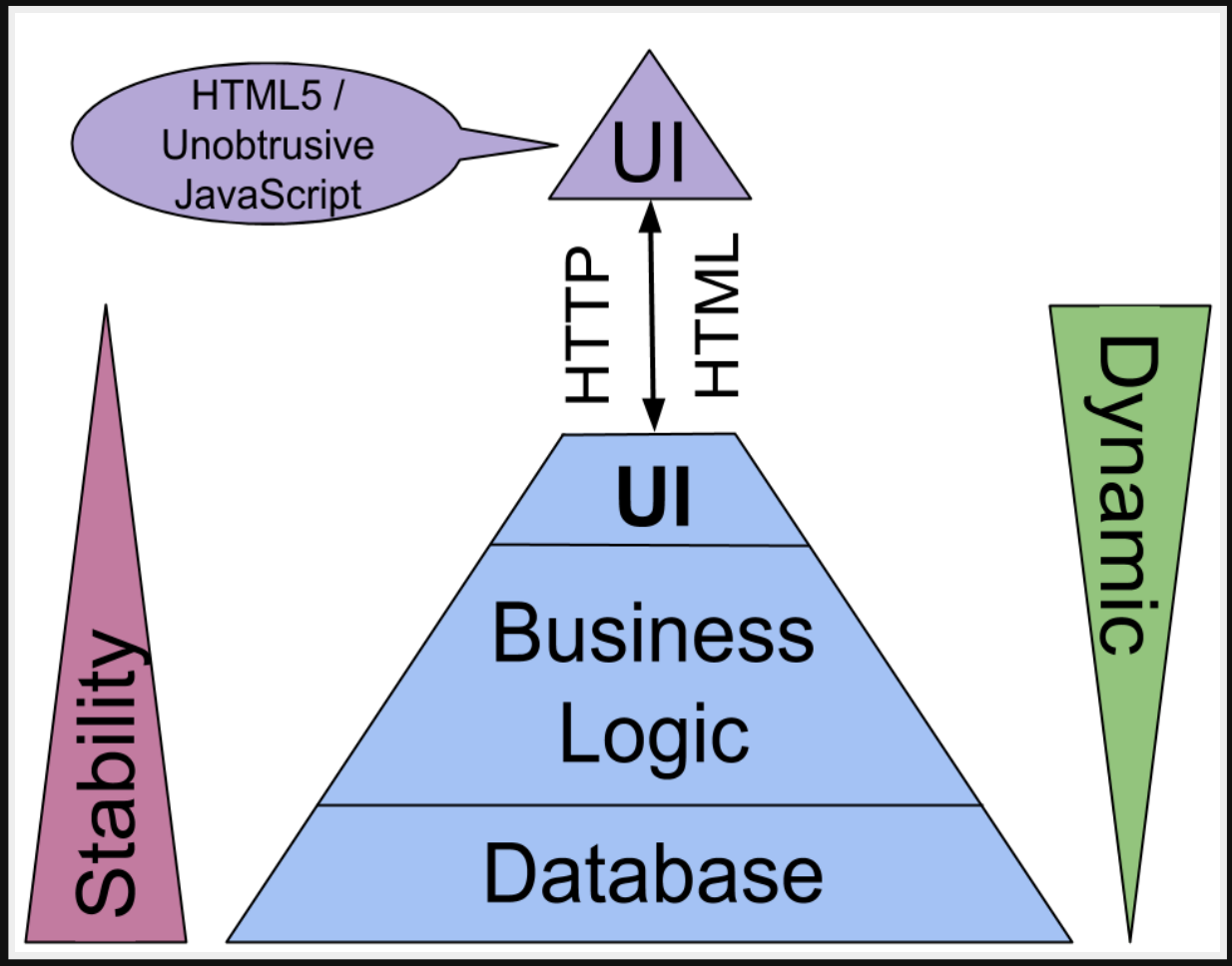
- Developer, Architect, Consultant, and Coach
- Hamburg, Germany
- http://zeigermann.eu
- Working for *embarc Software Consulting GmbH*
- @DJCordhose
- Book: JavaScript für Java-Entwickler



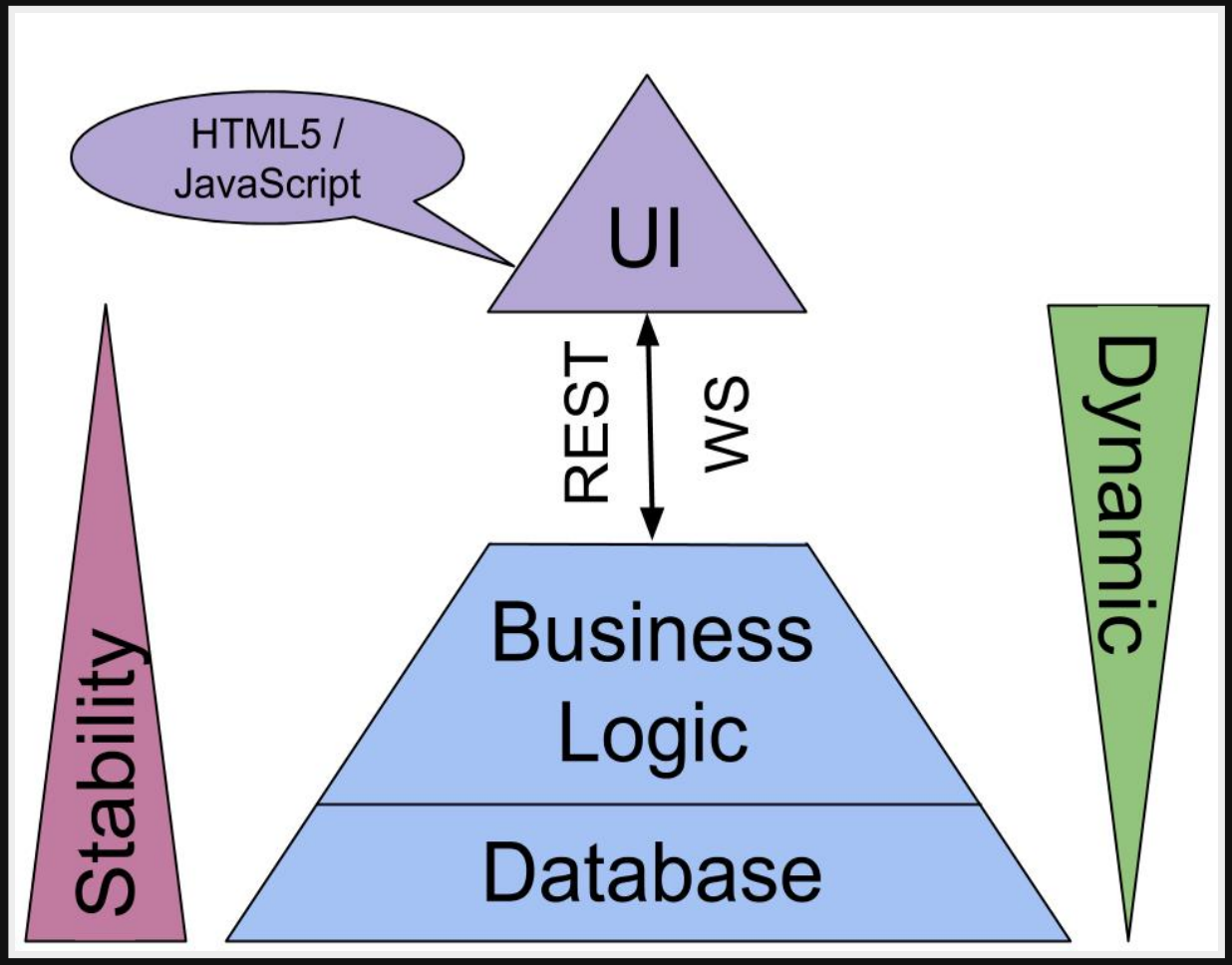
Why client side (browser) rendering and frameworks in the first place?

Single Page Applications (SPAs) move the application to the browser

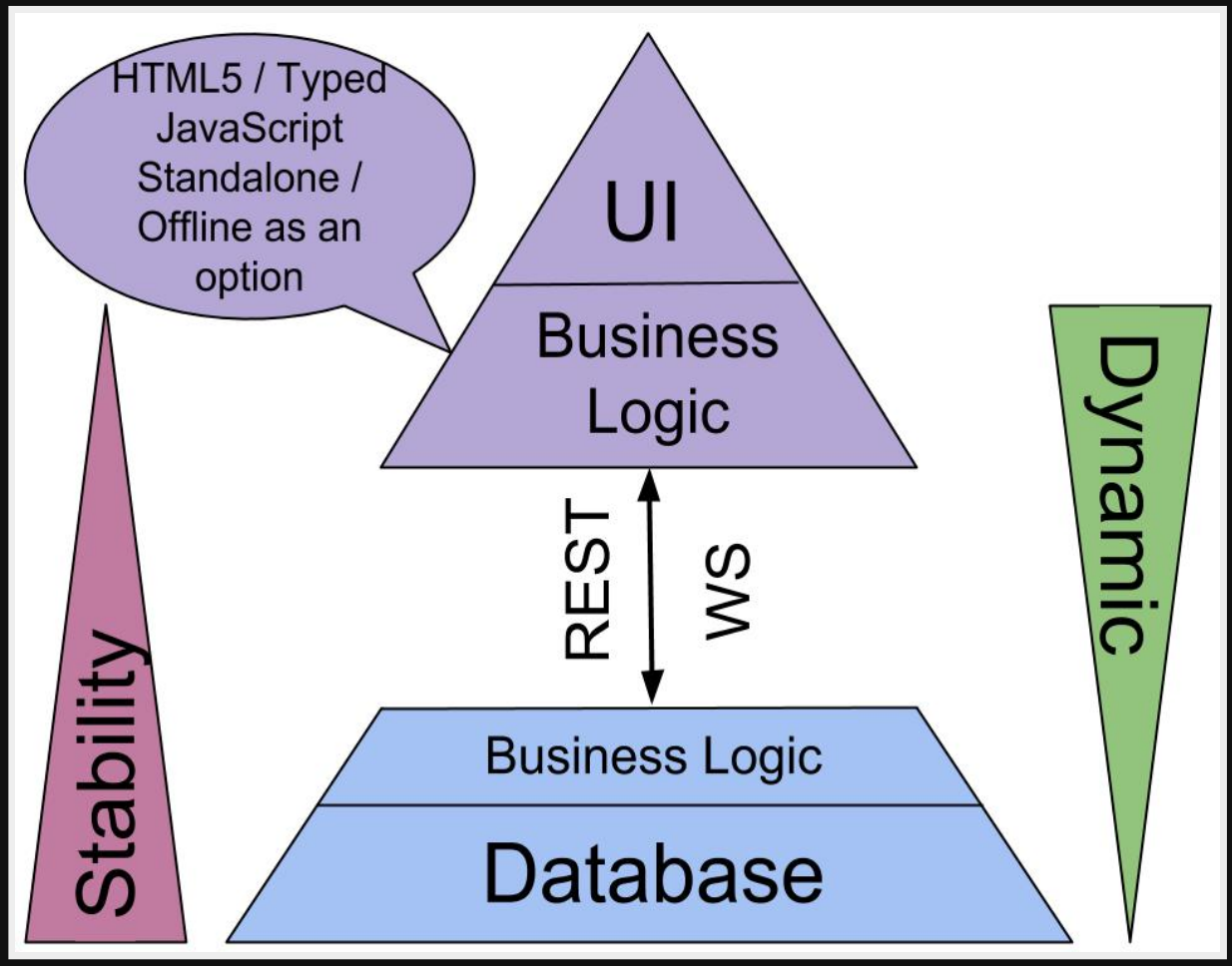
Classic Server Side Rendering (Blue: Server, Violett: Browser)



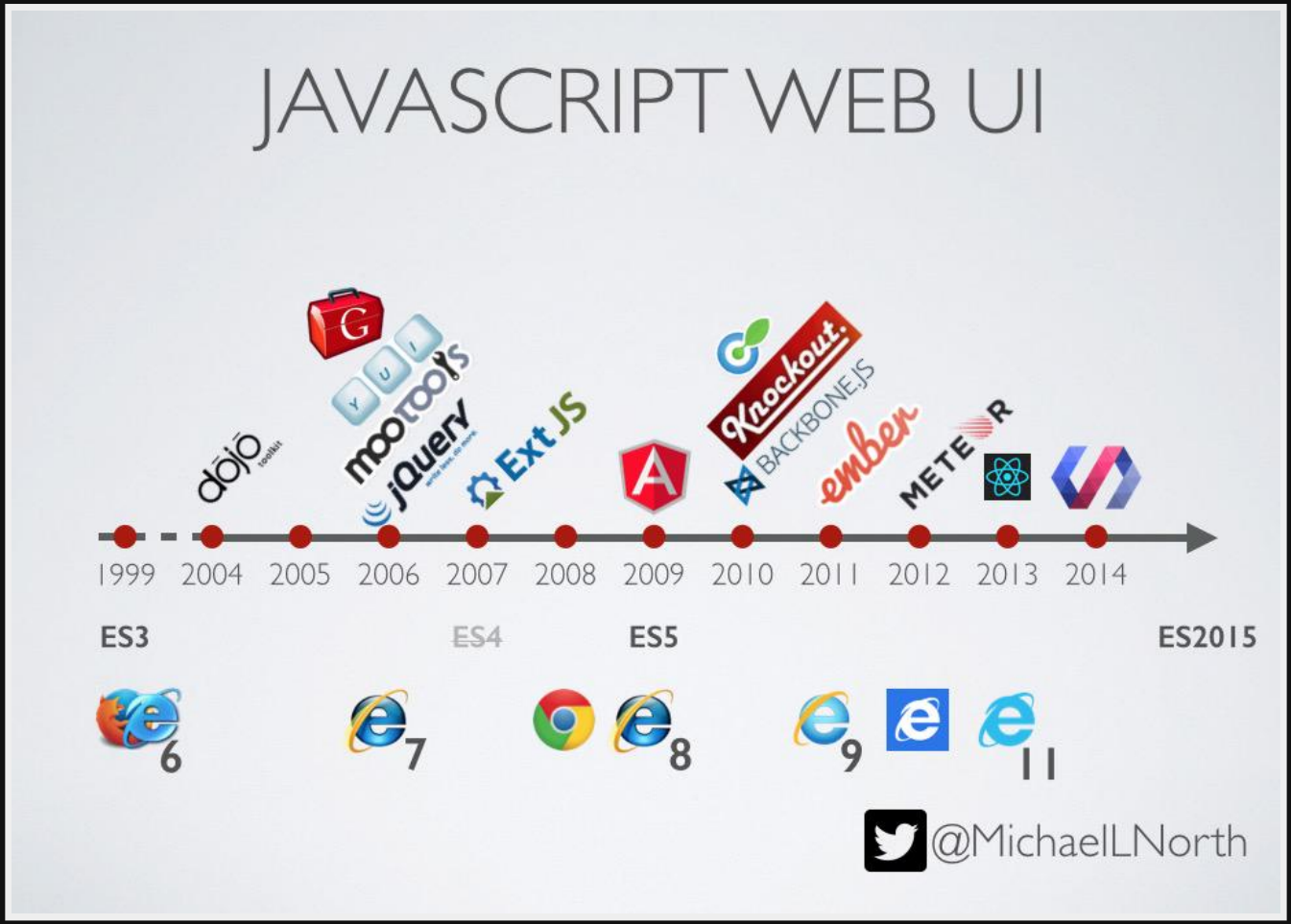
SPA Lean: Rendering and UI in Browser



SPA Fat: Additional business logic in Browser



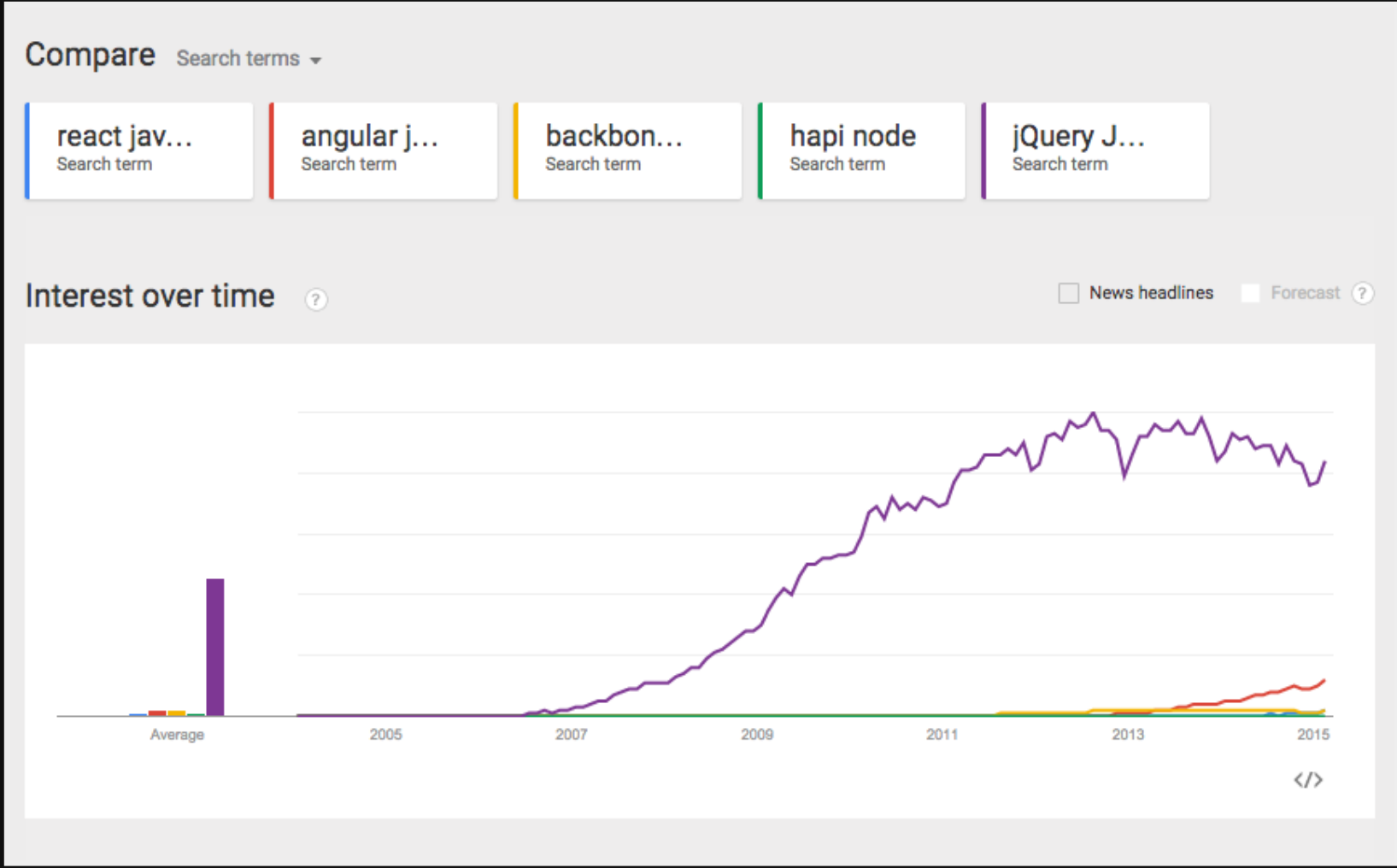
Overview



Our candidates

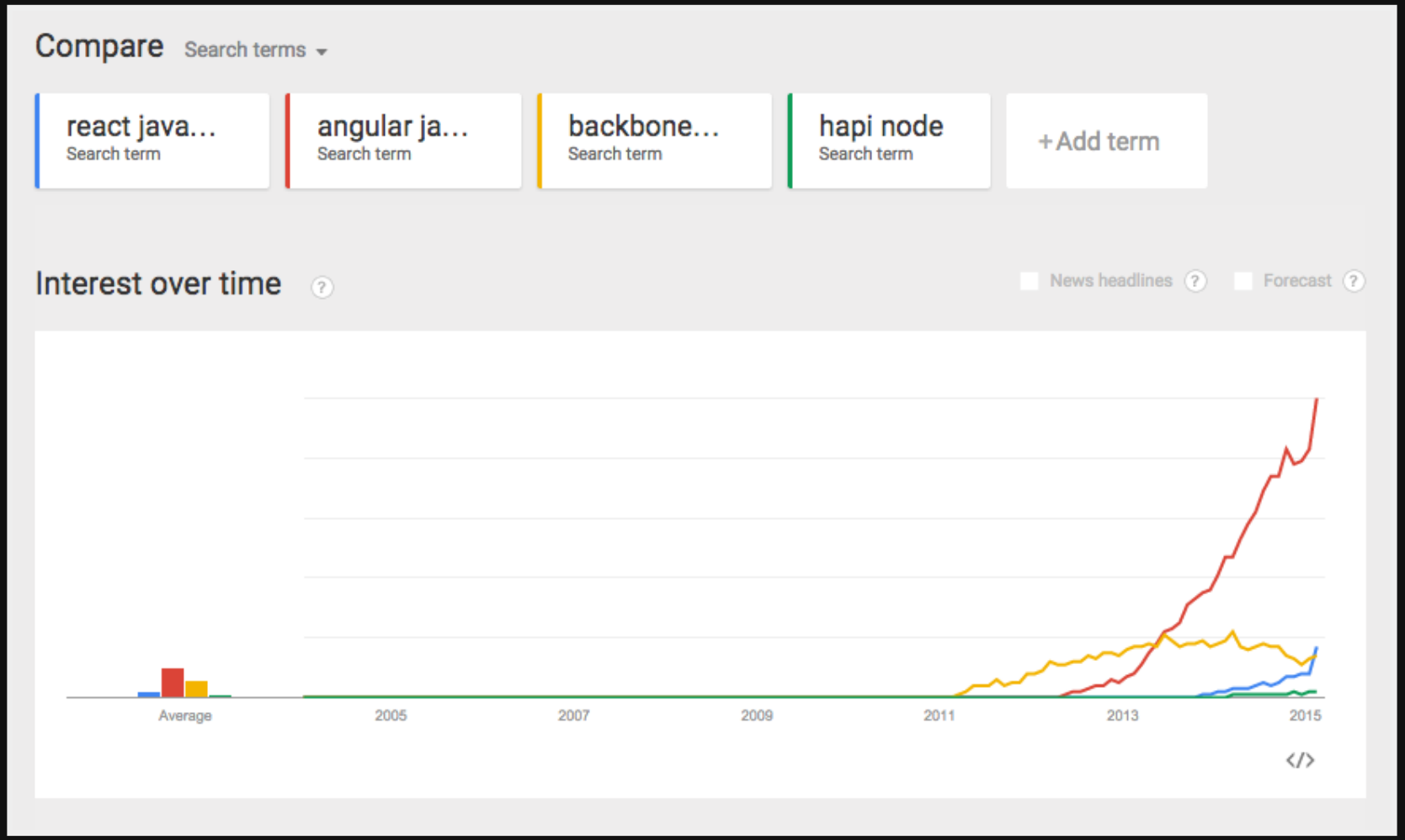
1. Classic Backend: hapi
2. Frontend Classic: jQuery
3. Adding structure: Backbone
4. Introducing Magic: Angular
5. Back to simplicity: React

Trends



https://www.google.com/trends/

Trends (without jQuery)



https://www.google.com/trends/

Classic Backend: hapi

To start with something more familiar

- Server Side JavaScript framework
- Driven by team at Walmart
- runs on node.js / io.js
- Request/Response based
- Server side rendering using template language of your choice

Hello World Hapi / Handlebars Template

```
<form>
    <input name="greeting" value="{{greeting}}">
    <p>{{greeting}}, World</p>
    <a href="/?greeting=">Clear</a>
    <input type="submit" value="Send">
</form>
```

```
var server = new Hapi.Server();
server.route({
    method: 'GET',
    path: '/',
    handler: function (request, reply) {
        reply.view('index', { greeting: request.query.greeting || 'Hello' });
    }
});
```

Run: node index.js

Frontend Classic: jQuery

- Standard JavaScript library
- Found almost everywhere
- "Fixes" DOM programming
- Does not abstract from DOM, though
- best when you have an existing DOM, e.g. a classic web application
  - modify something (make something visible depending on a click)
  - extract data (get data from input fields and do validation, fill other areas etc.)
  - It is possible to create new nodes or clone sections but then you are probably already lost.
  - locate something (find a button or an input field and submit a click or data - e.g. for E2E Tests)

Hello World jQuery

```
<input id="in">
<p><span id="log"></span>, World</p>
<input id="btn" type="button" value="Clear">
<script>
    var model;
    function setModel(value) {
        model = value;
        $("#log").html(model);
        $("#in").val(model);
    }
    $(document).ready(function () {
        $("#in").on("keyup", function (event) {
            setModel($("#in").val());
        });
        $("#btn").on("click", function () {
            setModel("");
        });
        setModel("Hello");
    });
</script>
```

Run

Adding structure: Backbone

- Builds on jQuery and underscore
- Key component is a model
- Model interacts with RESTful servers using CRUD operations
- Views structure application
- 2-way binding using Stickit

Hello World Backbone/Marionette: Putting it together

```
(new HelloView({
    model: new Model(),
    el: '#example'
})).render();
```

Run

Hello World Backbone/Marionette: Model

```
var Model = Backbone.Model.extend({
    defaults: {
        greeting: 'Hello'
    }
});
```

Hello World Backbone/Marionette: View

```
var HelloView = Marionette.ItemView.extend({
    events: {
        'click #btn': 'onBtnClick'
    },
    bindings: {
        '#in': 'greeting',
        '#out': 'greeting'
    },
    onBtnClick: function () {
        this.model.set('greeting', '');
    },
    onRender: function () {
        this.stickit(); // sets 2-way bindings
    },
    template: _.template('<input id="in"><p><span id="out"></span>, World</p>' +
                         '<input id="btn" type="button" value="Clear">')
});
```

Introducing Magic: Angular

- Current Hype
- By Google
- HTML enhanced for web apps!
- HTML extended by *directives* (=== components)
- Three Ds
  - Data Binding (2-way)
  - Dependency Injection
  - Directives (you can write your own directives)
- designed for good testability

Hello World AngularJS 1.x

```
<body np-app ng-controller="HelloController as helloController">
<input ng-model="helloController.greeting.text">
<p>{{helloController.greeting.text}}, World!</p>
<input type="button" value="Clear" ng-click="helloController.clear()">
</body>
```

```
function HelloController() {
    this.greeting = {
        text: 'Hello'
    };
}
HelloController.prototype.clear = function() {
    this.greeting.text = '';
};
```

Run

Angular 2.x

- 2.0 version in the make
- say goodbye to two-way data binding
- say goodbye to controllers
- say goodbye to scopes
- Will eventually feature form support as a replacement for two-way data binding
- works very well with TypeScript

Hello World AngularJS 2.x

```
@Component({
  selector: 'hello-app'
})
@Template({
inline: `<input [value]="greeting" #in (keyup)="updateModel(in.value)">
  <p>{{greeting}}, World</p>
  <button (click)="reset(in)">Clear</button>`
})
```

```
class HelloCmp {
  greeting: string;
  constructor() {
    this.greeting = 'Hello';
  }
  reset(input) {
    this.greeting = '';
    input.focus();
  }
  updateModel(value) {
    this.greeting = value;
  }
}
```

Back to simplicity: React

- By Facebook
- Framework for Facebook und Instagram
- Based on components, holding both view and logic
- Templates can be written in pure JavaScript or using the JSX template language
- Reactive one-way data-binding: Changes to state will be rendered automatically
- Smart diffing algorithm computes the changes between old and new virtual DOM
- Browser DOM experiences minimal amount of updates
- Can render both on server and on client side

Hello World React

```
class HelloMessage extends React.Component {
    constructor(props) {
        super(props);
        this.state = {greeting: this.props.greeting};
    }
    updateModel(event) {
        this.setState({greeting: event.target.value});
    }
    reset() {
        this.setState({greeting: ""});
    }
    render() {
        return (<div>
            <input value="{this.state.greeting}" onchange="{this.updateModel.bind(this)}">
            <p>{this.state.greeting}, World</p>
            <input type="button" value="Clear" onclick="{this.reset.bind(this)}">
        </div>);
    }
}
var mountNode = document.getElementById('example');
React.render(<HelloMessage greeting="Hello" />, mountNode);
```

Run

Summary

- Hypes come and go quickly in JavaScript land
- There is no perfect "silver bullet" framework
- What is the framework for you depends on your context und requirements

Thank you!

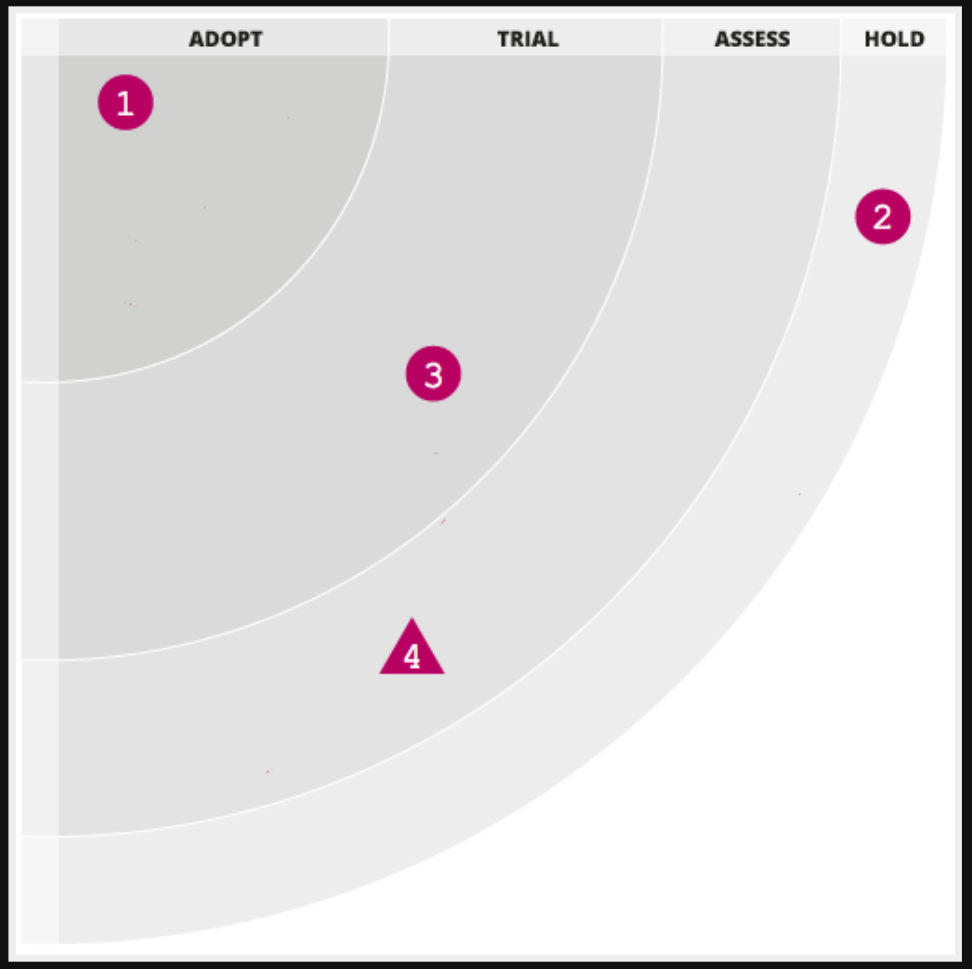
Questions / Discussion

Oliver Zeigermann / @DJCordhose

Bonus Material

ThoughtWorks JavaScript framework technology radar

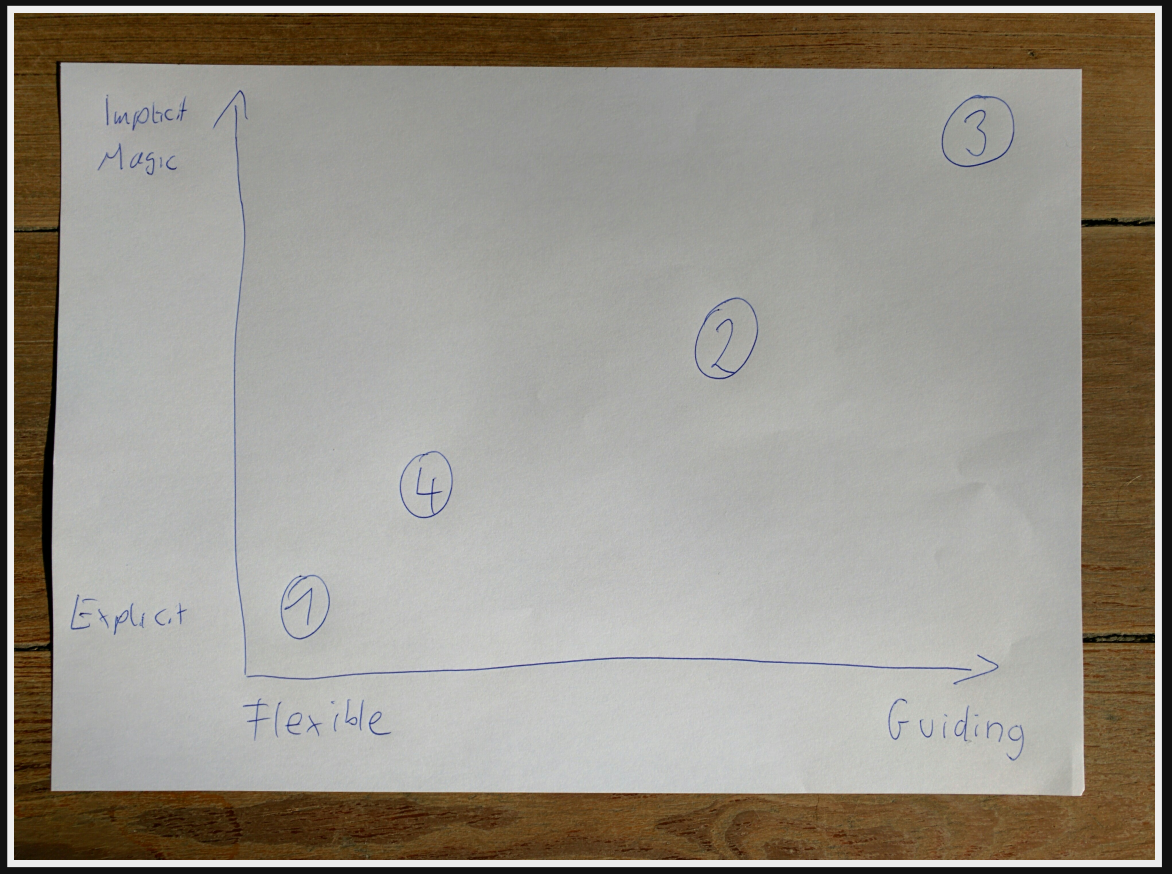
1. jQuery (implied)
2. Backbone
3. Angular.js
4. React



http://www.thoughtworks.com/radar/a-z

Magic and Flexibility

1. jQuery
2. Backbone/Marionette
3. Angular
4. React



Consider: What is your history?

- CSS/HTML without jQuery
  - => jQuery
- CSS/HTML with jQuery
  - => Backbone/Marionette
- Java, C# or C++
  - => Angular
- JavaScript (maybe with other frameworks)
  - => React