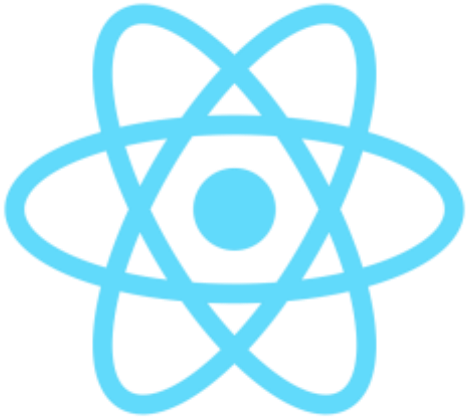


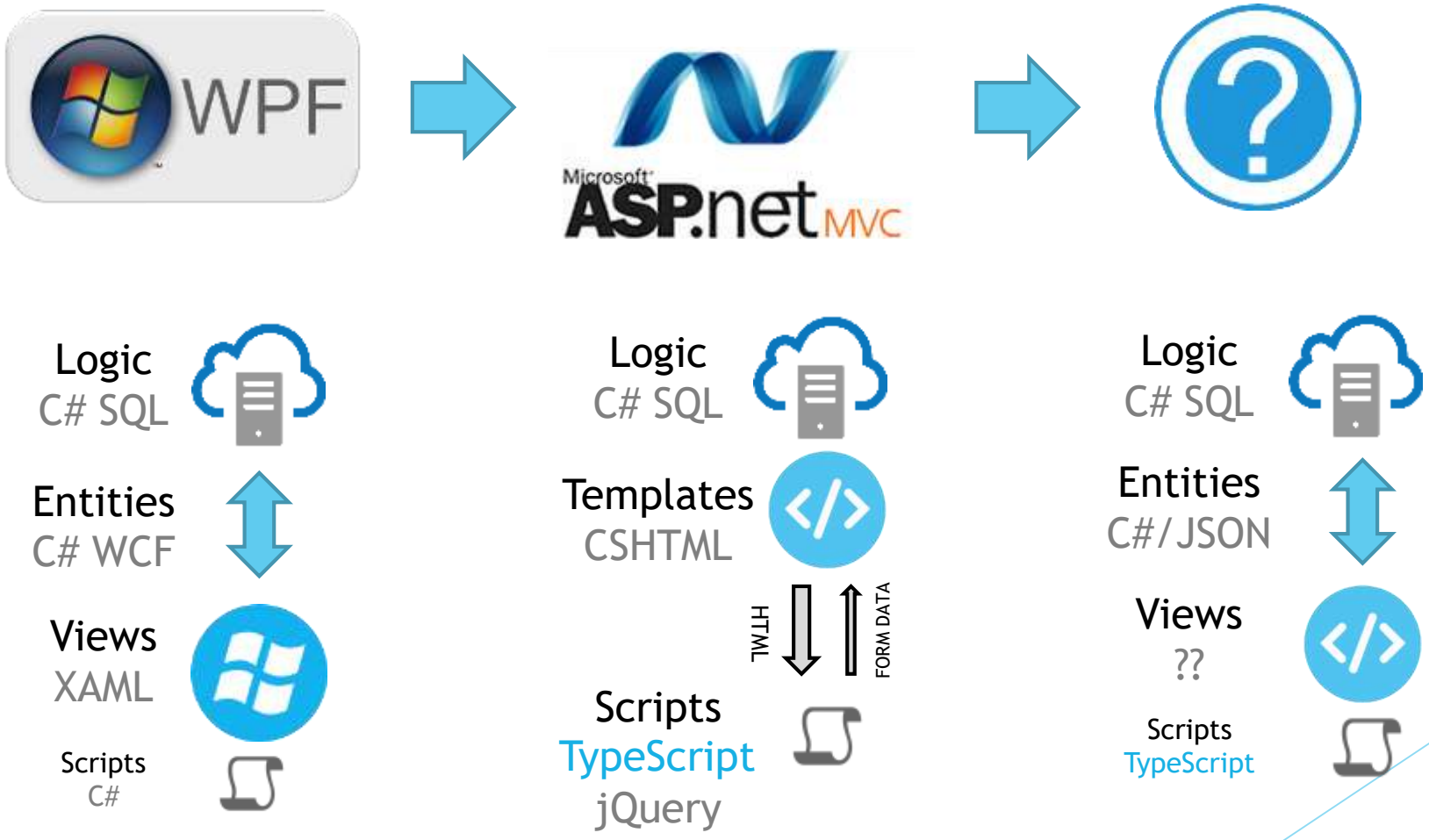
React

TypeScript

Olmo de Corral
Signum Framework Developer
#olmocc olmo@signumsoftware.com

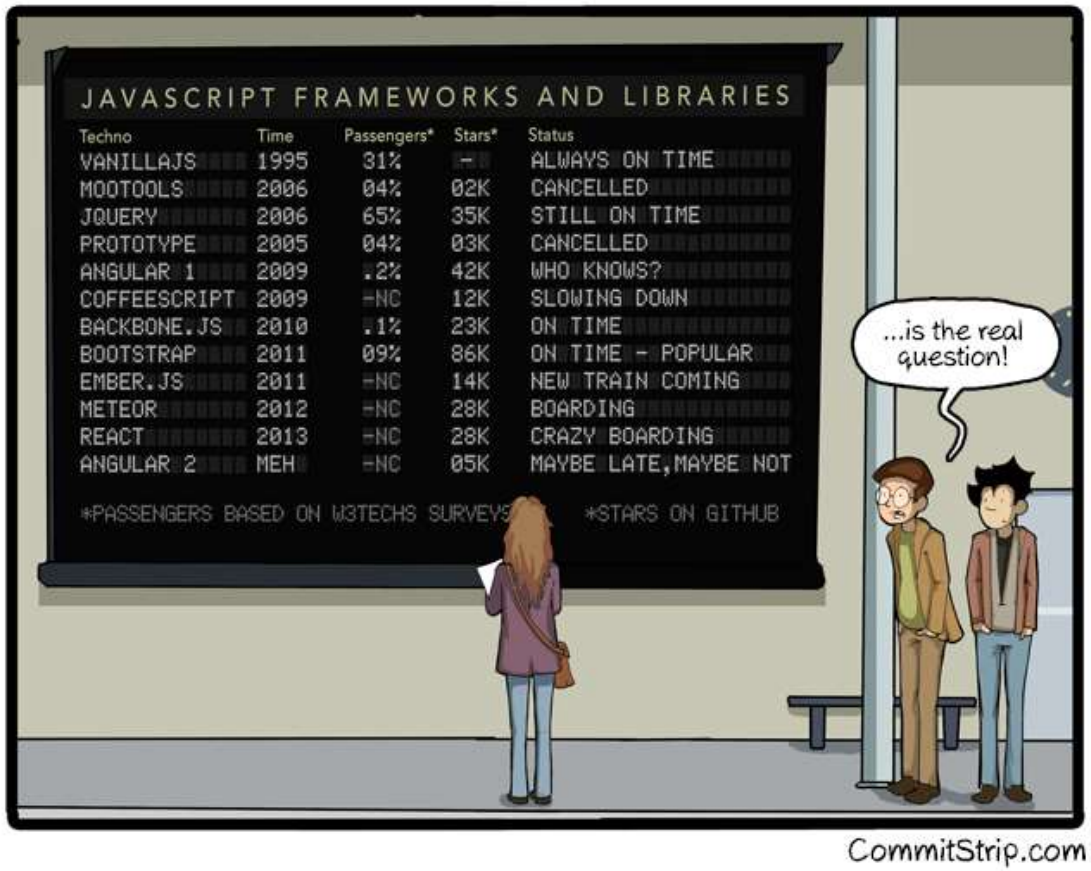
www.signumsoftware.com www.signumframework.com www.controlexpert.com

A little bit of (my) History



Searching for a JS Framework

Searching for a JS Framework

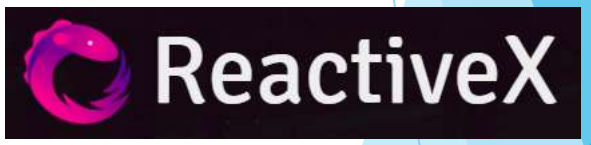


Searching for a JS Framework

Model
View
Co~~ntroller~~ Whatever

Just
Render
Components™

Reactive



The Web

JSX Basics

```
import * as React from 'react'

//1. HTML literal
var mySpan = <span>React is Cool!</span>;

//2. Heads Up! class -> className
var mySpan2 = <span className="active">JSX is awesome!</span>;

//3. Without JSX
var mySpan3 = React.createElement("span", { className: "active" }, "JSX is awesome!");

//4. You can turn back to JS/TS using { }
var myDiv = <input type="text" value={"Back in " + "JS"} disabled={true}/>;

//5. Also for children elements
var myDiv2 = <div>Conclussion: {Math.random() > 0.5 ? mySpan : mySpan2 }</div>;
```

JSX Advanced

```
//6. Styles are objects
var mySpan4 = <span style={{ border: "1px solid red", color: "black" }}/>;

//7. No ng-repat, just use map, filter, etc.. but use 'key' !!
var countries = ["Germany", "France", "Spain", "Italy"];
var countriesDiv2 = <div>{countries.map((c, i) => <span key={i}>{c}</span>) }</div>;

//8. Abuse bool operators!
var showTitle = false;
var myDiv5 = <div>{ showTitle && <h1>The Title</h1> }</div>;

//9. Use spread operator {...}
var props: React.HTMLAttributes = { className: "my-class", title: "My title" };
var div6 = <div {...props}/>;

//10. Clone elements to modify them
function addCustomClass(element: React.ReactElement<any>) {
    return React.cloneElement(element, { className: "my-custom-class" });
}

var clonedDiv = addCustomClass(<div/>);
var clonedDiv2 = <div className="my-custo-class"/>;
```

Custom Components

```
interface PersonProps {
    name: string;
    dateOfBirth: Date;
}

class Person extends React.Component<PersonProps, void> {
    render() {
        return (
            <div>
                <input type="text" value={this.props.name}/>
                <DateTimePicker date={this.props.dateOfBirth} />
            </div>
        );
    }
}
```

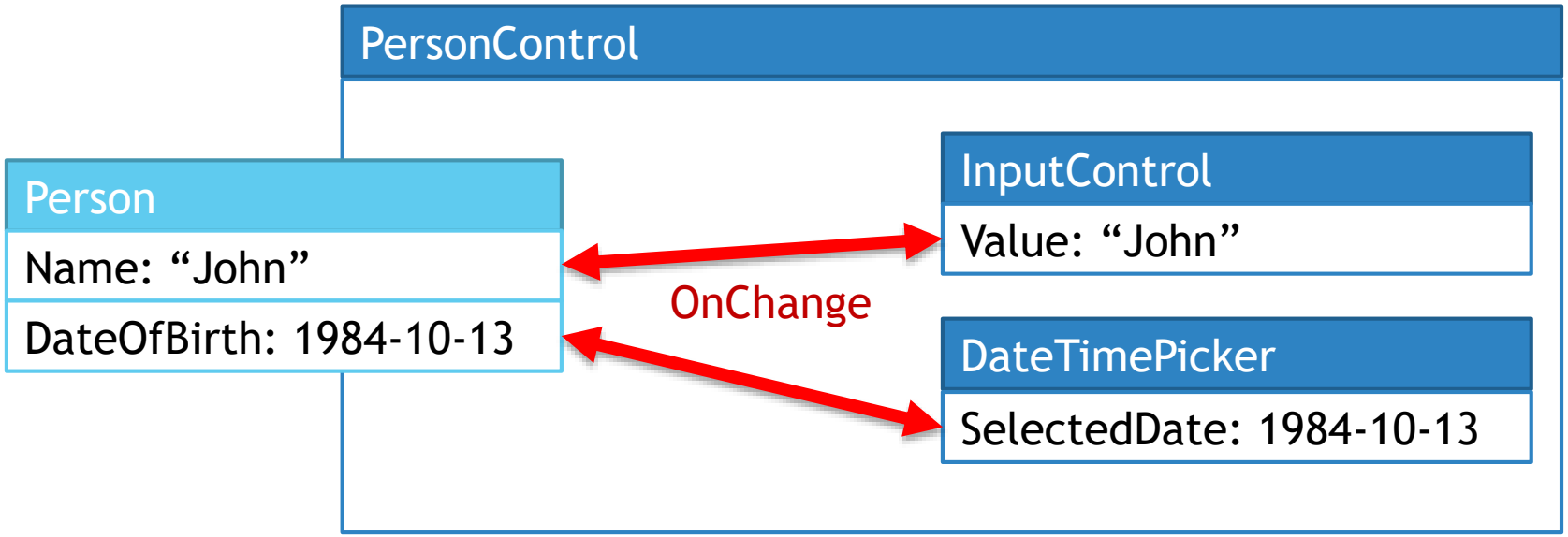
Intrinsic HTML Elements

- In **lowercase**

React Components

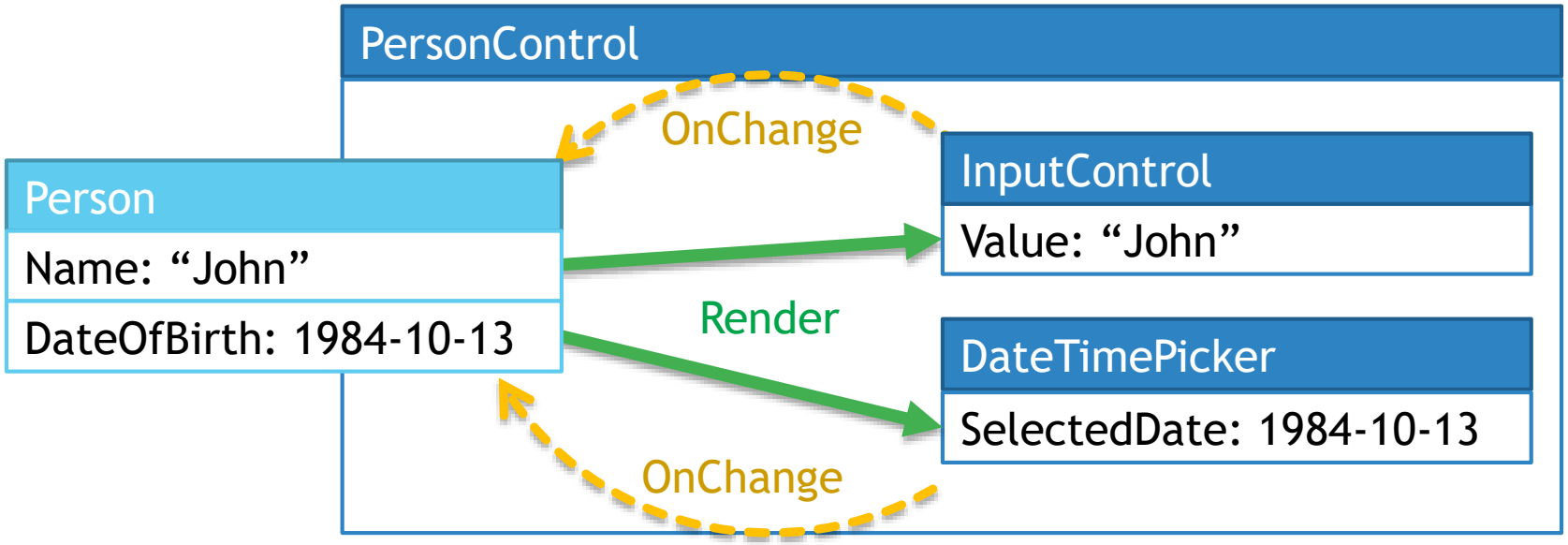
- In **Uppercase**

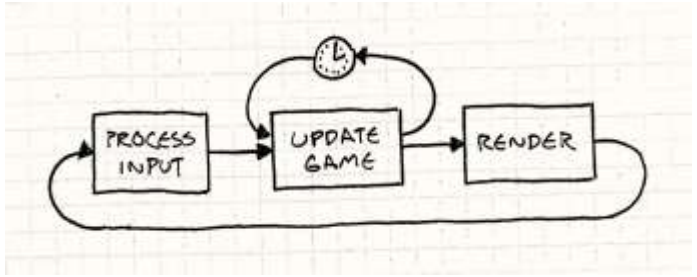
Two-Way Databinding



Unidirectional Data Flow

React





DEMO 1:
Person Component

- Handling Events
  - Be careful with `this`!
- ForceUpdate

Props vs State

Props

- Public
- Read-only

Recommended

(a.k.a. Parameters)

State

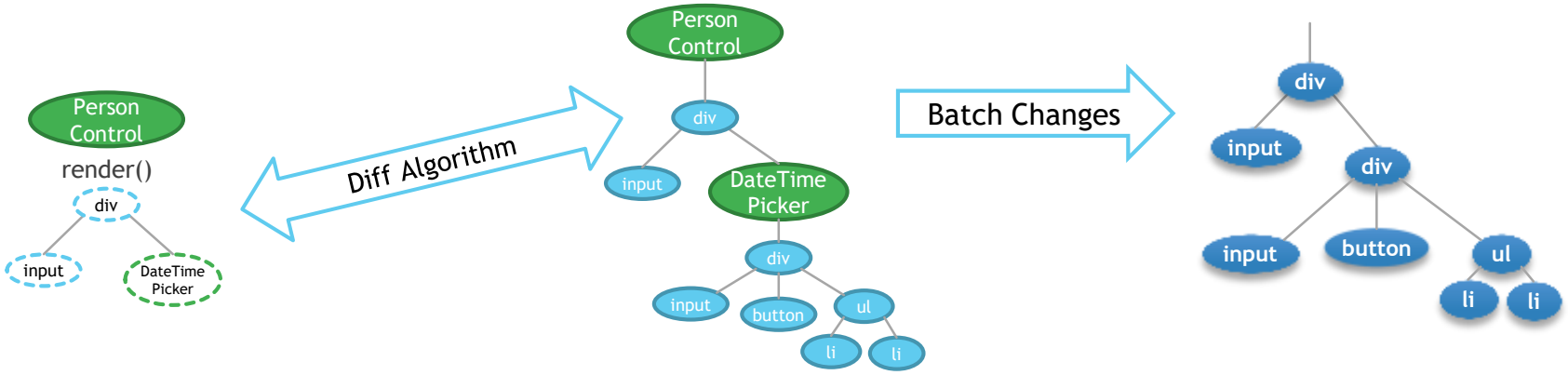
- Private
- Modifiable

Only if needed

DEMO 2: DateTimePicker

- Props vs State
- SetState

Virtual DOM



render()

- FAST (React Elements)
- Shallow Tree
- Contains:
  - ReactElement('div')
  - ReactElement(DateTime Picker)

Virtual DOM

- FAST (javascript objects)
- Recursive Tree
- Contains:
  - **Instances** of React Components and HTML elements

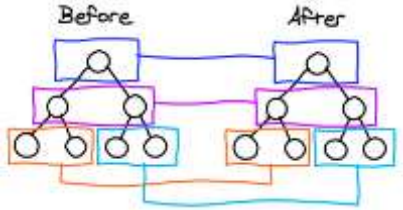
Native DOM

- SLOW (layout & render)
- Complete Native Tree
- Contains:
  - Real DOM nodes of HTML elements

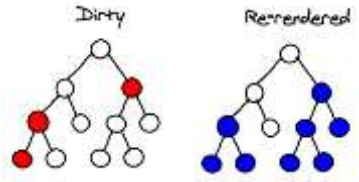
Diff Algorithm

Fast and Straightforward

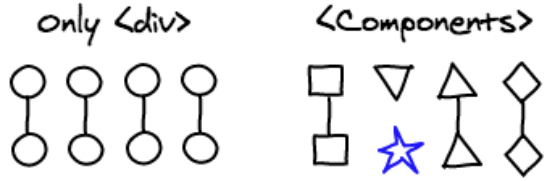
Level by Level



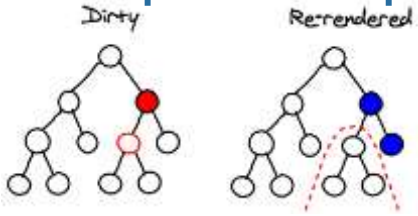
Sub-tree Rendering



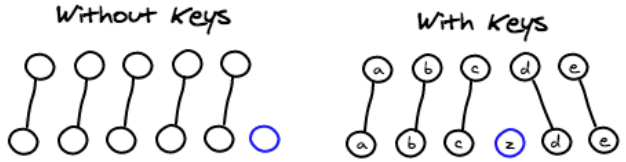
Comparing Nodes



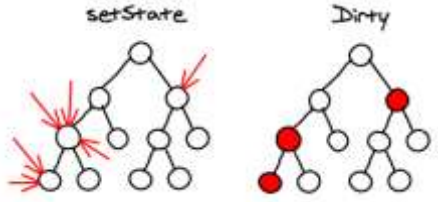
shouldComponentUpdate



Comparing Lists



Batching



http://calendar.perfplanet.com/2013/diff/

DEMO 3:
React Developer Tools

Component API

```
// Base component for plain JS classes
class Component<P, S> implements ComponentLifecycle<P, S> {
    constructor(props?: P, context?: any);
    setState(state: S, callback?: () => any): void;
    setState(f: (prevState: S, props: P) => S, callback?: () => any): void;
    forceUpdate(callBack?: () => any): void;
    render(): JSX.Element;
    props: P;
    state: S;
    context: {};
    refs: {
        [key: string]: ReactInstance
    };
}
```

```
//Props that ever controls has
interface Props<T> {
    children?: ReactNode;
    key?: string | number;
    ref?: string | ((component: T) => any);
}

//In ReactDOM
function findDOMNode(instance: ReactInstance): Element;
```

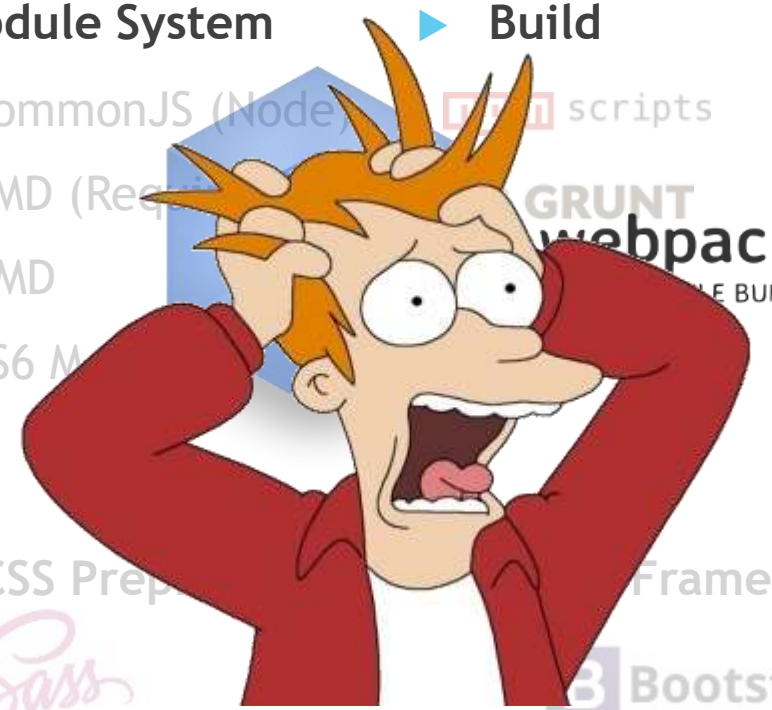
```
// Component Specs and Lifecycle
interface ComponentLifecycle<P, S> {
    componentWillMount?(): void;
    componentDidMount?(): void;
    componentWillReceiveProps?(nextProps: P, nextContext: any): void;
    shouldComponentUpdate?(nextProps: P, nextState: S, nextContext: any): boolean;
    componentWillUpdate?(nextProps: P, nextState: S, nextContext: any): void;
    componentDidUpdate?(prevProps: P, prevState: S, prevContext: any): void;
    componentWillUnmount?(): void;
}
```

GRADUATED IN REACT!!!

...but

Just the V in MVC…

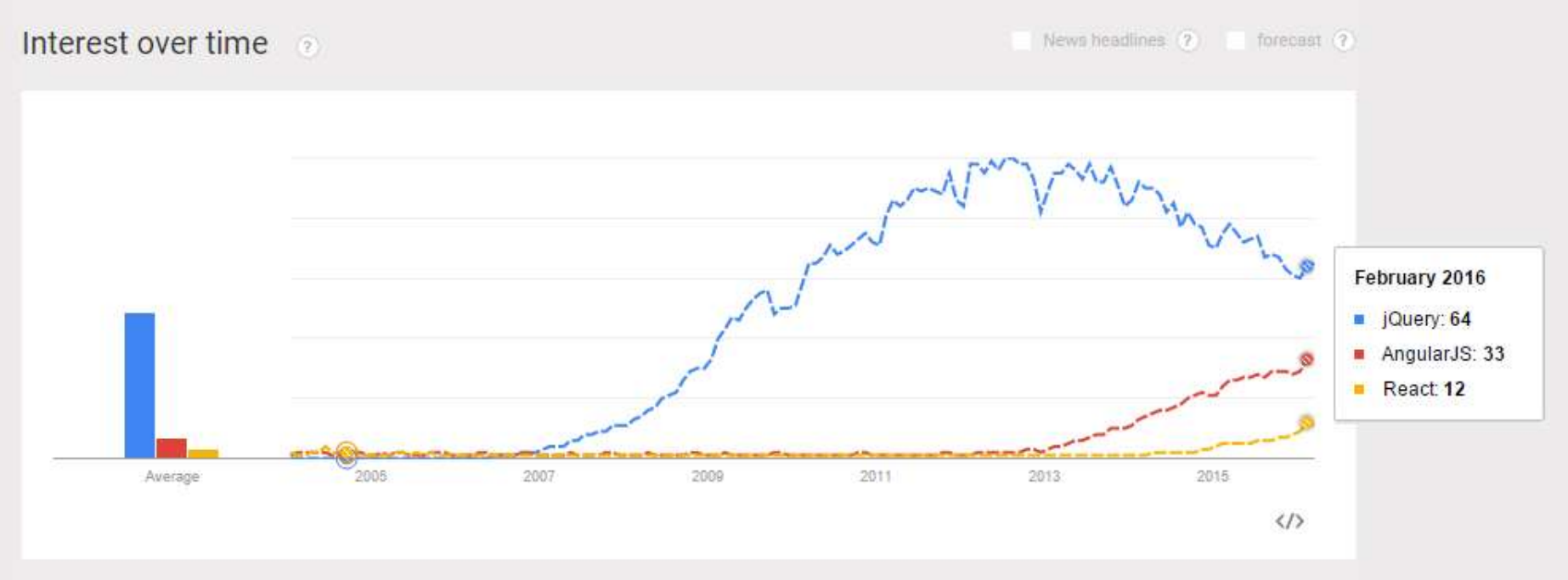
- Language
  - CoffeeScript
  - Babel
  - flow
  - TypeScript
  - elm
  - Dart
  - AtScript
- Module System
  - CommonJS (Node)
  - AMD (Req
  - UMD
  - ES6 M
- Build
  - scripts
  - GRUNT
  - webpack
  - LE BUNDLER
- Bundle
  - browserify
  - webpack
  - MODULE BUNDLER



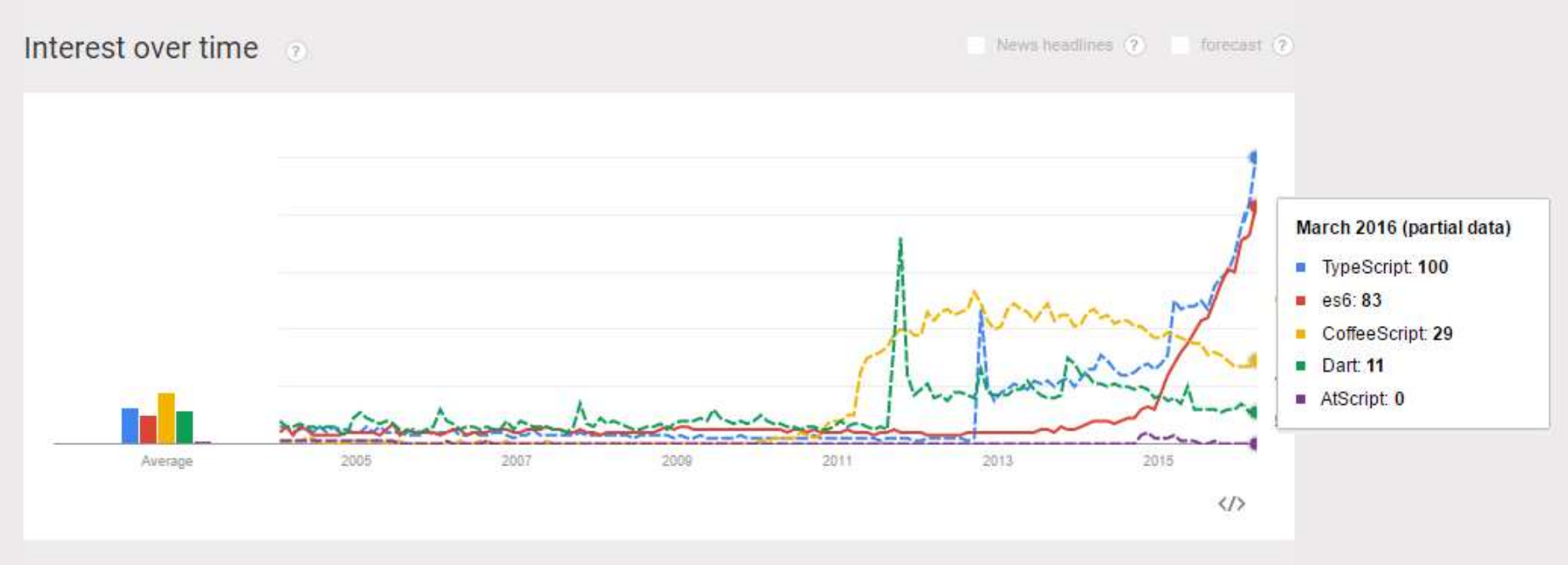
- CSS Prep
  - Sass
  - less
- Framework
  - Bootstrap
  - Foundation

React + Typescript

Popular Javascript Frameworks



Popular To-Javascript Languages



React + Typescript + Angular

React TypeScript ANGULARJS by Google

React + Typescript

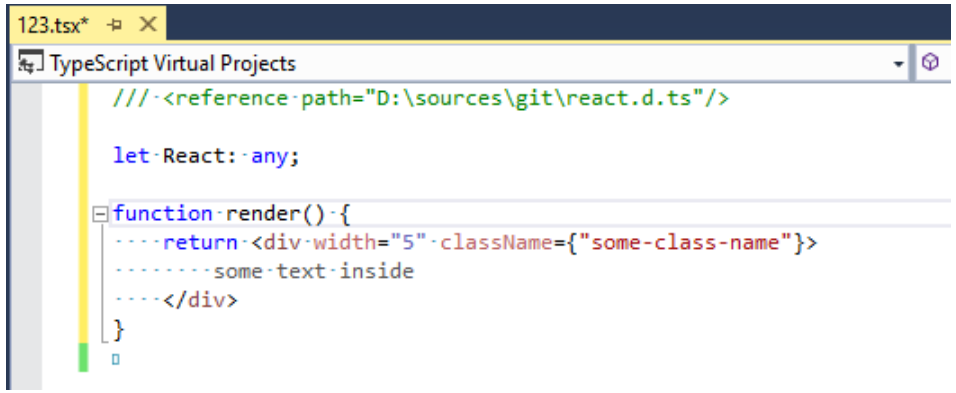
tsxElementResolution1.tsx | tsxElementResolution2.tsx | tsxElementResolution4.tsx | solution18.tsx

```tsx
class MyClass extends React.Component {
    render() { <div>Hello, {this.props.who}</div>; }
    props: {
        who?: string;
    }
}

var x = <MyClass who='hello'  />;
```

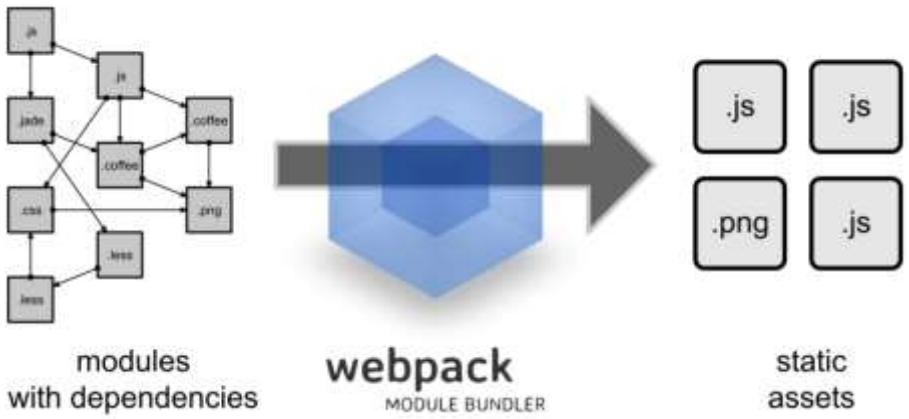
Line 3, Column 13 | Tab Size: 4 | TypeScript

TSX = TS + JSX

- Typescript 1.6
  - Full JSX support
    - Strongly typed
    - Props vs State
    - Spread operator
  - New `as` operator for casting
  - Two outputs: preserve / react
  - BUG: Formatting Problems
- Typescript 1.8
  - Syntax coloring
  - Stateless Function Components
  - Simplified props
  - Custom JSX Factories
  - BUG: Auto-completion problems
  - Flow analysis (return)
  - Lack of import auto-complete



React Ecosystem (Tested)



- Solves the whole bundling problem
  - Follows JS `require` / `import`
  - Sync and Async dependencies
  - Styles, fonts and other assets
  - Hash & Cache Invalidation
  - `--watch` mode
- Tricky to configure
- Black box

- Lots of plugins and loaders
  - Typescript / Babel / CoffeeScript
  - CSS / LESS / SASS
  - Customize bundles
  - Uglyfx
  - Notify

WebPack Task Runner for
Visual Studio 2015 … useful?

REACT/ROUTER

keeps your UI in sync with the URL

- Associates Components with URLs
  - Using JSX!
- Cumbersome API with many breaking changes
- History repository needed?
- Incomplete Typescript definition files (.d.ts)

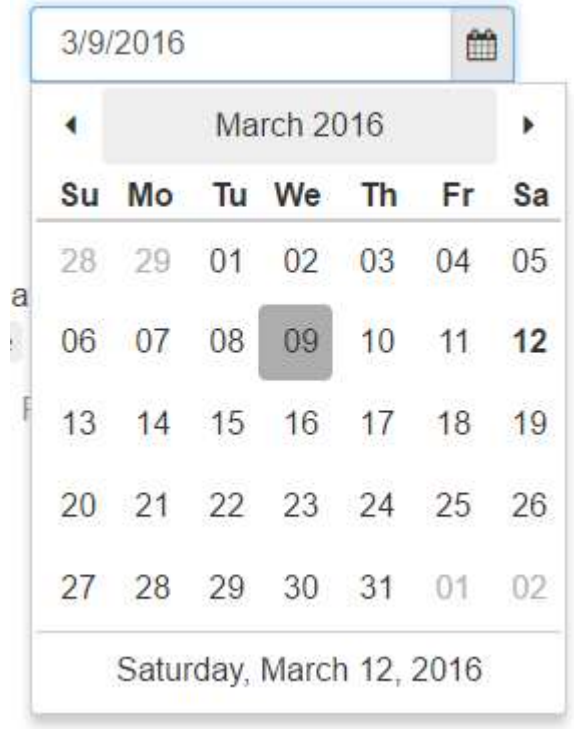
https://github.com/reactjs/react-router

React-Bootstrap

The most popular front-end framework, rebuilt for React.

- Good documentation
- Very Complete (Modals, Tabs, Tooltips)
  - Too much? (Button, MenuItem)
- Overlays repository need?
- Incomplete Typescript definition files (.d.ts)

https://react-bootstrap.github.io/



React Widgets

offers a set of html form inputs, built from scratch with React

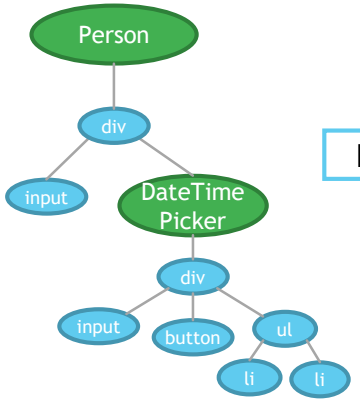
- Good documentation
- High quality components
  - Dropdown / Combobox
  - Date-Time picker
- No definition files (I built my own)

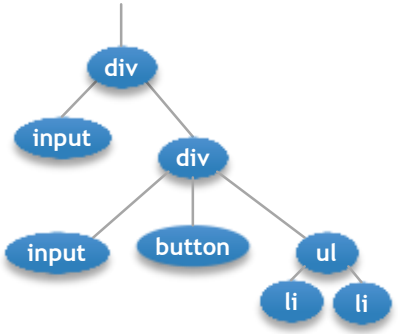
https://jquense.github.io/react-widgets/docs

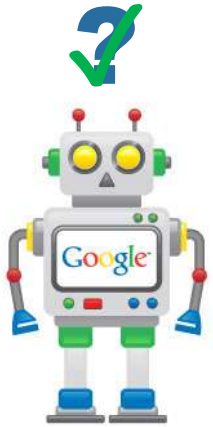
React Ecosystem (Untested)

Server-Side Rendering

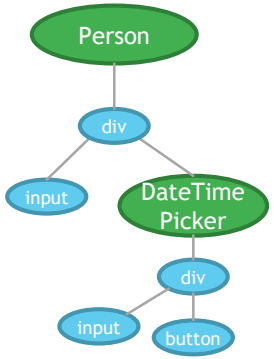
Browser

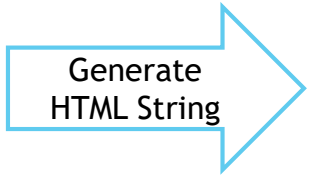






Server






```
<div>
    <input/>
    <div>
        <input/>
        <button/>
    /div>
</div>
```

ReactJS.NET makes it easier to use Facebook's React and JSX from C# and other .NET languages, focusing specifically on ASP.NET MVC (although it also works in other environments).

- On-the-fly JSX to JavaScript compilation

```
<script src="@Url.Content("~/Scripts/HelloWorld.jsx")"></script>
```

- JSX to JavaScript compilation via popular minification/combination libraries

```
bundles.Add(new JsxBundle("~/bundles/main").Include("~/Scripts/HelloWorld.jsx"))
```

- **Server-side component rendering**

```
@Html.React("CommentsBox", new { initialComments = Model.Comments })
```

- **Typescript support?**

React Hot Loader will keep it mounted, preserving the state.

http://gaearon.github.io/react-hot-loader/

Works on IIS?

React Native

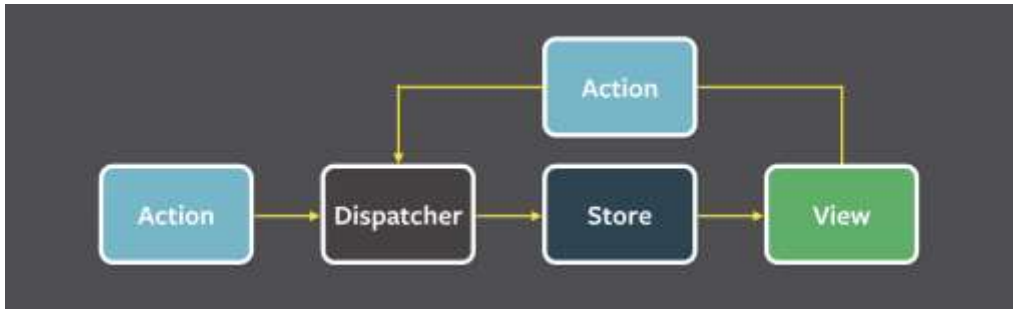
A FRAMEWORK FOR BUILDING NATIVE APPS USING REACT

- Native Components
- Asynchronous Execution
- Touch Handling
- Flexbox and Styling
- Extensibility

```
// iOS

var React = require('react-native');
var { TabBarIOS, NavigatorIOS } = React;

var App = React.createClass({
  render: function() {
    return (
      <TabBarIOS>
        <TabBarIOS.Item title="React Native" selected={true}>
          <NavigatorIOS initialRoute={{ title: 'React Native' }} />
        </TabBarIOS.Item>
      </TabBarIOS>
    );
  },
});
```

```
// Android

var React = require('react-native');
var { DrawerLayoutAndroid, ProgressBarAndroid } = React;

var App = React.createClass({
  render: function() {
    return (
      <DrawerLayoutAndroid
        renderNavigationView={() => <Text>React Native</Text>}>
        <ProgressBarAndroid />
      </DrawerLayoutAndroid>
    );
  },
});
```

Flux Implementations



Flux

Reflux

Redux

FLUXXOR

Alt

- Mat
- Fluxette
- Freezer
- Fluxury
- Fluxy
- ummox
- Marty.js
- McFly
- Delorean
- Lux

Signum.React

Signum Framework UIs

Logic
C# SQL

Entities
C# WCF

Views
XAML

Scripts
C#

Logic
C# SQL

Templates
CSHTML

HTML
FORM DATA

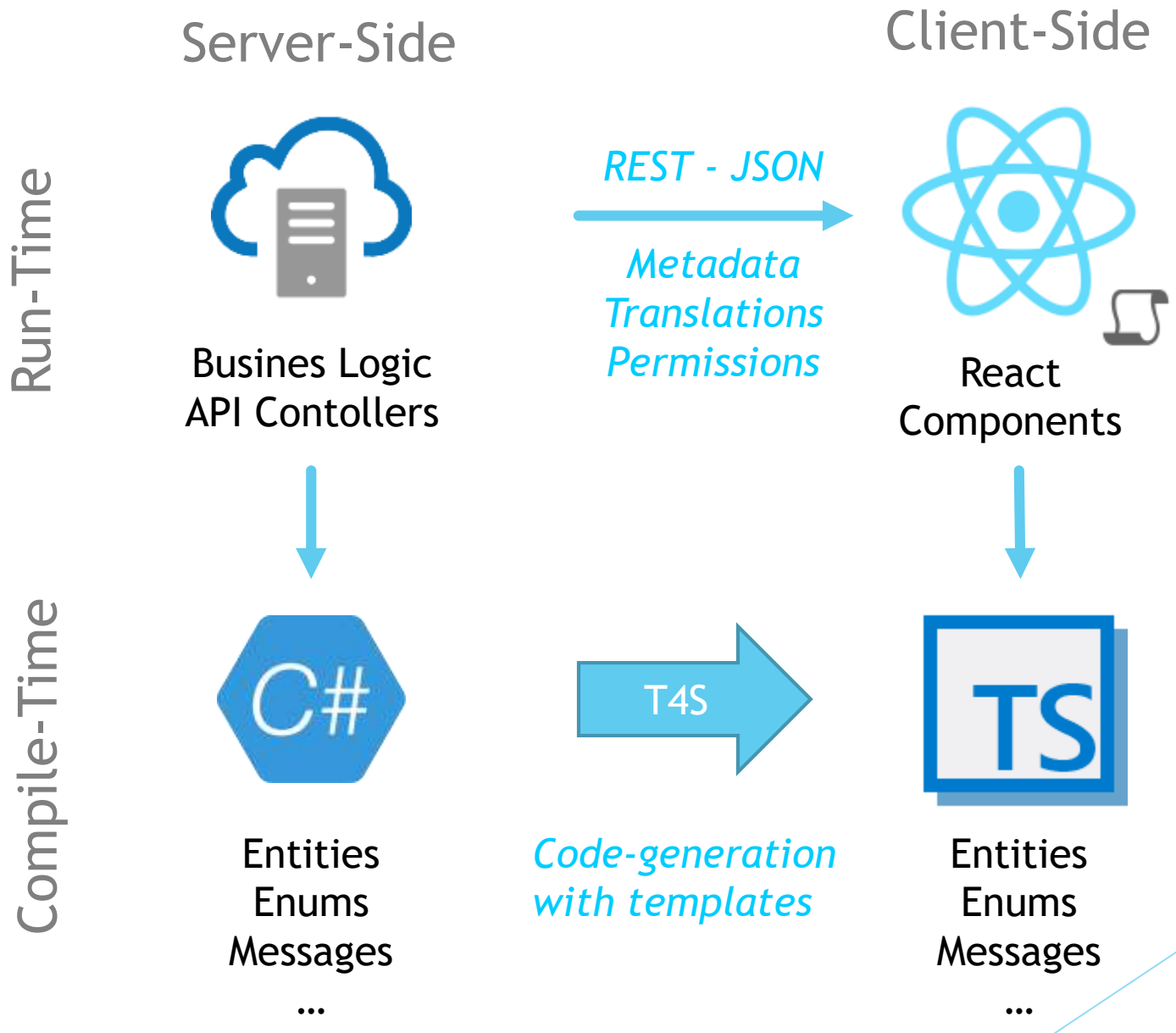
Scripts
TypeScript
jQuery

Logic
C# SQL

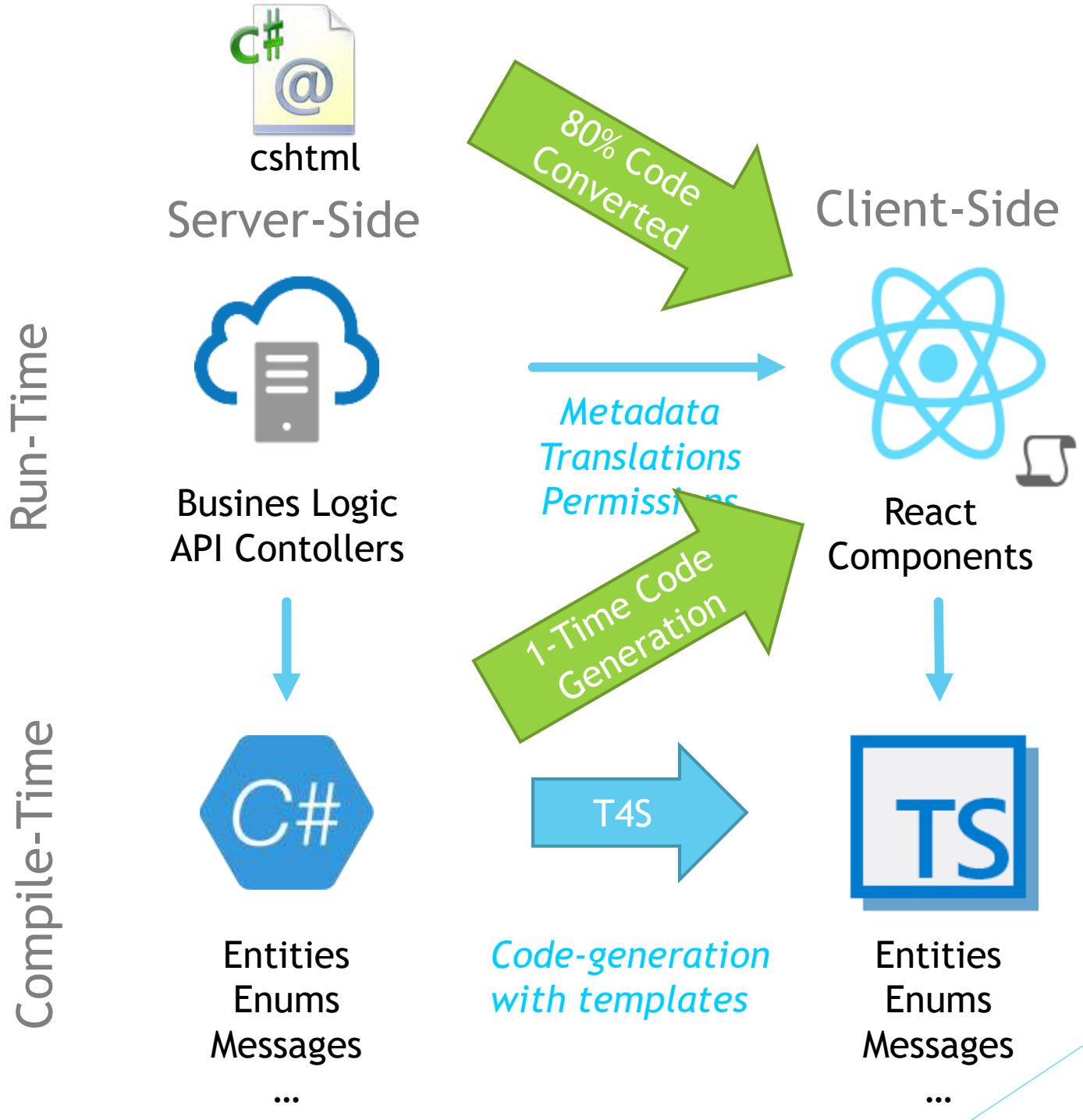
Entities
C#/JSON

Views
TSX

Scripts
TypeScript


Server-Side
Client-Side
Run-Time
REST - JSON
Metadata
Translations
Permissions
Busines Logic
API Contollers
React
Components
Compile-Time
C#
T4S
TS
Entities
Enums
Messages
...
Code-generation
with templates
Entities
Enums
Messages
...

DEMO 4:
Signum.React


cshtml
Server-Side
Client-Side
80% Code Converted
Run-Time
Metadata
Translations
Permissions
Busines Logic
API Contollers
React
Components
1-Time Code Generation
Compile-Time
C#
T4S
TS
Entities
Enums
Messages
...
Code-generation with templates
Entities
Enums
Messages
...

Conclusion

1. **React is Awesome**
   - Simplicity
   - Expressivity
   - Performance
   - … but incomplete

1. **React + Typescript is even better**
   - Compile-time checked
   - Auto completion
   - Safe renames
   - ...missing .d.ts

3. **Big Ecosystem**
   - Webpack
   - React Bootstrap
   - React Widgets
   - React Router
   - React Native
   - React.Net
   - React Hot Loader
   - …

4. **You may need server-side solution**
   - Auto-generate definition files from C#
   - Expose meta-data with reflection
   - Facilitate migration

THE END

Olmo de Corral
Signum Framework Developer
#olmocc olmo@signumsoftware.com

www.signumframework.com

www.signumsoftware.com

www.controlexpert.com