REACT ON RAILS

David Anguita <3 CiudadReal.rb

Hi, I'm @danguita

The UI as a first-class citizen

Why React?

Library for building user interfaces

Who is behind React?

The V in MVC

- It's a library, not a framework
- Excellent for managing UIs with data that changes over time
- Makes no assumptions about the rest of your stack
- Runs entirely on the client

Can be adopted gradually

React basics

- Based in components
- Component lifecycle
- Props and State
- Virtual DOM

```
class Greeting extends React.Component {
  handleClick() {
    alert(`Hey, ${this.props.name}`);
  }

  render() {
    return (
      <button onClick={this.handleClick.bind(this)}>
        Click me
      </button>
    )
  }
}
```

```
<Greeting name='CiudadReal.rb'/>
```

```
class Greeting extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: 'Unknown' };
  }

  handleClick() {
    this.setState({ name: this.props.name });
  }

  render() {
    return (
      <div>
        <div>Name: {this.state.name}</div>
        <button onClick={this.handleClick.bind(this)}>
          Click me
        </button>
      </div>
    )
  }
}
```

Short learning curve

We have come here to talk about Rails!

REACT ON RAILS

David Anguita <3 CiudadReal.rb

Integration approaches

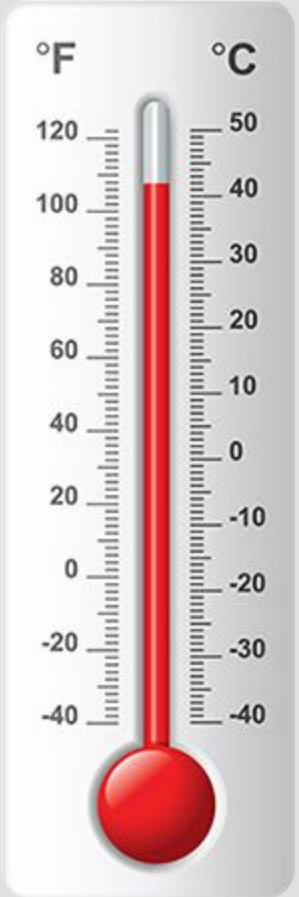
The react-rails gem

- Easy to use for Rails developers, especially with legacy code
- JSX compilation in the asset pipeline
- Render components into Views via helpers
- Component generator

Rails front-end, retrieving data in the server

- Data are passed to React components directly from Rails Controllers
- Rails Controllers are for routing and retrieving data
- Rails Views are for mounting components and passing data to them
- Good starting point since it is just replacing the View
- There's room for improvement in terms of architecture

```ruby
# app/controllers/posts_controller.rb

def show
  @post = Post.find(params[:id])
end

# app/views/posts/show.html.erb

= react_component("Post", post: @post)

# app/assets/javascripts/components/post.es6.jsx

class Post extends React.Component {
  render() {
    return (
      <div>
        <div>Title: {this.props.post.title}</div>
        <div>Body: {this.props.post.body}</div>
      </div>
    )
  }
}
```


°F
°C
120
100
80
60
40
20
0
-20
-40
50
40
30
20
10
0
-10
-20
-30
-40

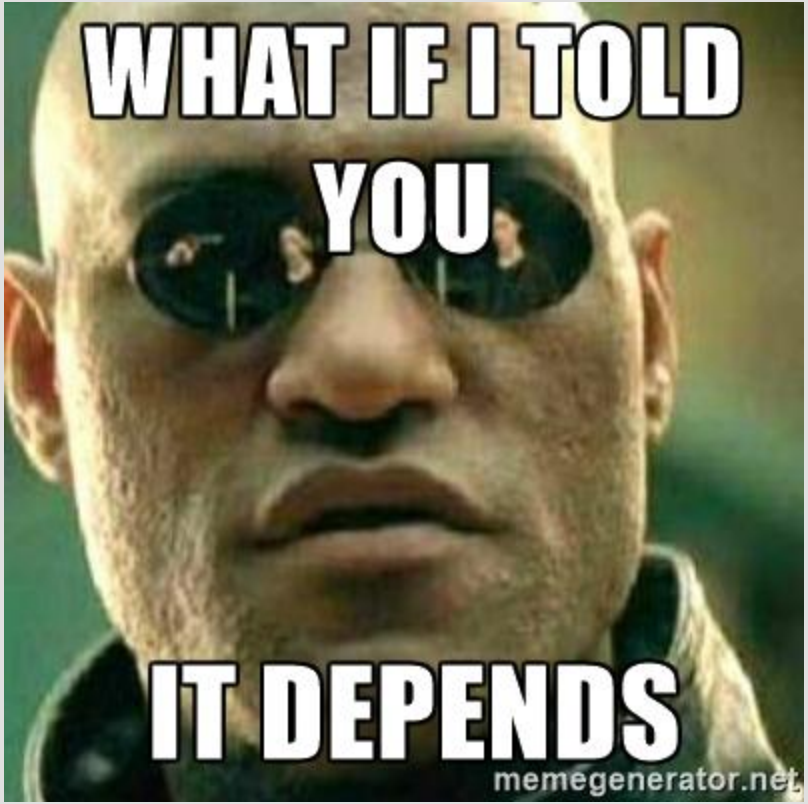
Rails front-end, retrieving data in the client

- The front-end can now be virtually isolated from the back-end
- The back-end is providing a well-defined API
- We still have Rails Controllers for routing and Views for mounting components
- Slightly better approach in terms of architecture
- Not leaving the comfort of Rails

```
# app/views/posts/show.html.erb

= react_component("Post", postId: params[:id])

# app/assets/javascripts/components/post.es6.jsx

class Post extends React.Component {
  componentDidMount() {
    this.setState({ post: this.findPostById(this.props.postId) });
  }

  render() {
    return (
      <div>
        <div>Title: {this.state.post.title}</div>
        <div>Body: {this.state.post.body}</div>
      </div>
    )
  }
}
```


°F
°C
120
100
80
60
40
20
0
-20
-40
50
40
30
20
10
0
-10
-20
-30
-40

Should we stop here?


WHAT IF I TOLD YOU
IT DEPENDS
memegenerator.net

Standalone front-end app + API-only back-end

- Front-end and back-end are separate applications
- The back-end is just a stateless API
- The front-end is a static bundle of code consuming those well-defined APIs
- Separation of concerns
- Easier to test in isolation
- Having two applications may require separate development cycles

How does it look like?

back-end

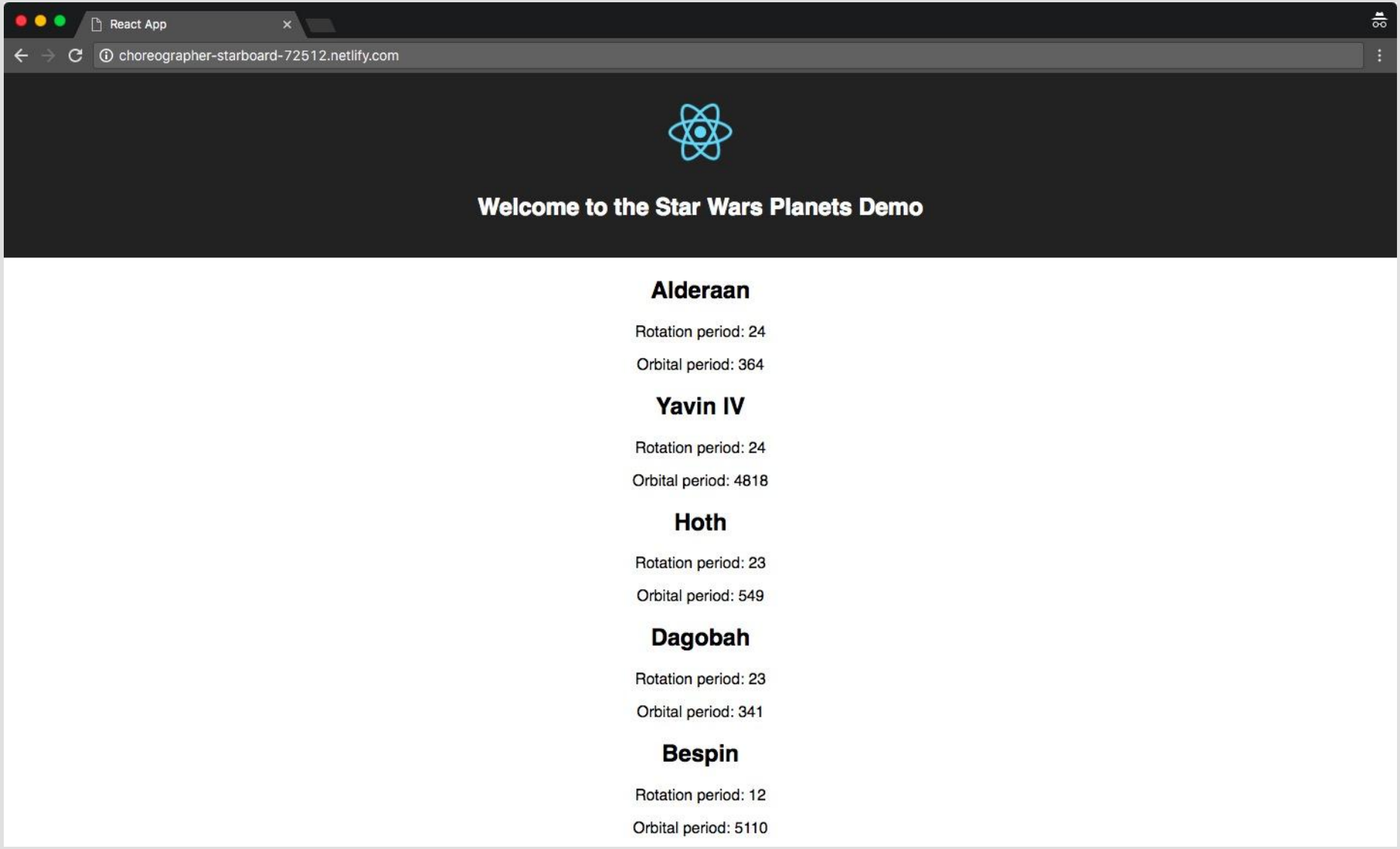
```
$ curl http://swapi.co/api/planets/

[
  {
    "name": "Alderaan",
    "rotation_period": "24",
    "orbital_period": "364"
  },
  {
    "name": "Yavin IV",
    "rotation_period": "24",
    "orbital_period": "4818"
  },
  ...
]
```

front-end

```
├── config
│   ├── ...
├── node_modules
│   ├── ...
├── package.json
├── public
│   ├── favicon.ico
│   └── index.html
└── src
    ├── App.js
    ├── PlanetItem.js
    ├── PlanetList.js
    └── index.js
```

```
class PlanetList extends React.Component {
  componentDidMount() {
    this.fetchPlanets();
  }

  fetchPlanets() {
    fetch("http://swapi.co/api/planets/")
      .then(function(res) {
        return res.json();
      }).then(function(json) {
        this.setState({ planets: json.results });
      }.bind(this));
  }

  render() {
    return (
      <div className="PlanetList">
        {this.state.planets.map(function(planet, index) {
          return (
            <PlanetItem key={index} planet={planet}/>
          )
        })}
      </div>
    );
  }
}
```

```
class PlanetItem extends React.Component {
  render() {
    return (
      <div className="PlanetItem">
        <div className="PlanetItem-header">
          <h2>{this.props.planet.name}</h2>
        </div>
        <div className="PlanetItem-body">
          <p>Rotation period: {this.props.planet.rotation_period}</p>
          <p>Orbital period: {this.props.planet.orbital_period}</p>
        </div>
      </div>
    );
  }
}
```


React App
choreographer-starboard-72512.netlify.com
Welcome to the Star Wars Planets Demo
Alderaan
Rotation period: 24
Orbital period: 364
Yavin IV
Rotation period: 24
Orbital period: 4818
Hoth
Rotation period: 23
Orbital period: 549
Dagobah
Rotation period: 23
Orbital period: 341
Bespin
Rotation period: 12
Orbital period: 5110


°F
°C
120
100
80
60
40
20
0
-20
-40
50
40
30
20
10
0
-10
-20
-30
-40

Benefits of having a API-only back-end

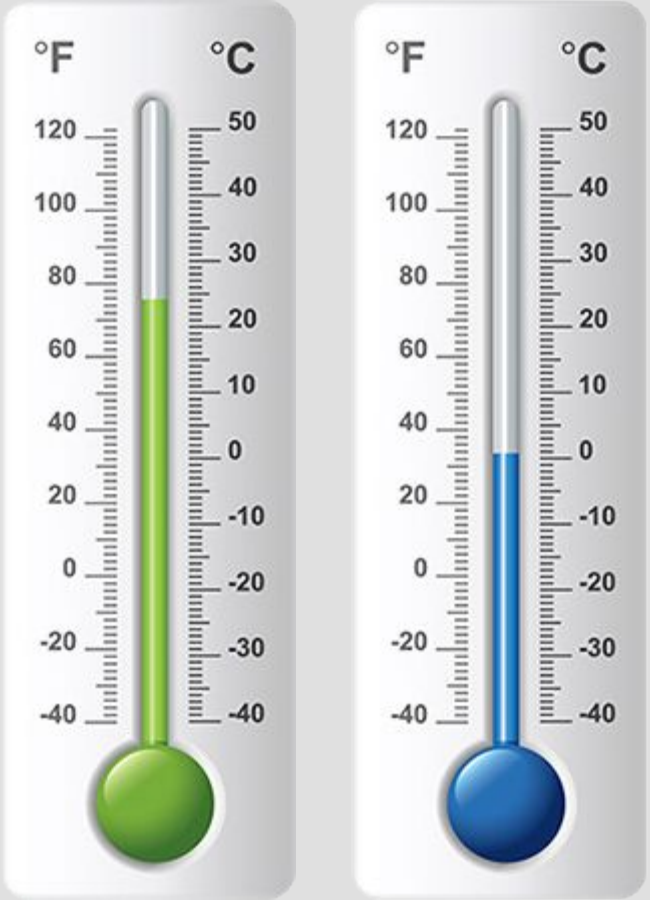
- There could be multiple clients hitting exactly the same back-end
- The technology behind the interface is replaceable
- It is not dealing with Views logic or HTML rendering
- It is focused on data representation

Benefits of having a standalone front-end

- Easy delivery
- Separate development cycle
- The front-end application itself is replaceable
- It is focused on user interactions

Drawbacks of having a standalone front-end

- Limited client-side performance
- Limited browser compatibility
- The UI initialization may be delayed in slow Internet connections
- Higher level of complexity


°F
°C
120
100
80
60
40
20
0
-20
-40
50
40
30
20
10
0
-10
-20
-30
-40

First do it,
then do it right,
then do it better

Thank you

Questions?

CiudadReal.rb, October 2016