React for a brighter web

From startups to enterprise
Alex Peretti

Founder of FolioShack
Acquired by Bloomberg
Working in R&D
Building a publishing platform
London Needs One Million Homes in Coming Decade

BY NEIL CALLANAN

Soaring demand means London and its surrounding counties will need at least one million new homes in the next 10 years to meet demand and prevent values and rentals from spiraling higher, the Adam Smith Institute research group said in a report today.

Building on less than 4 percent of the city’s Green Belt, the spaces around the city where development is limited, would provide the homes needed, the report said. Almost all of the homes could be provided within 800 metres of a commuter train station, it said.

London is at risk of a housing bubble as real estate begins to look overvalued after house prices surged 40 percent from the beginning of 2013, UBS Group said in October. A rising population is also pushing up rents, which averaged £1,544 a month in November, more than double the amount charged in the rest of the U.K., according to Homelet, the U.K.'s largest reference-checking and rentals insurance company.

“The increasing demand for housing is putting pressure on our cities, the growth and prospects which are strained by urban containment policies that were introduced nearly 70 years ago,” Tom Papworth wrote in the report. “Green belts also have significant negative effects in human welfare, pushing up accommodation costs, reducing private space, increasing house price volatility and increasing the cost of business.”

The Greater London Authority wants to double house building in the U.K. capital to 42,000 homes a year, the most since the 1930s, because the city’s population is expected to rise to 9 million by 2020 from 8.4 million in 2013.

Five Stand Trial in Climax to Insider-Trading Case

BY SLIZI RING

Five former finance professionals, including a one-time Deutsche Bank manager, will go on trial next week for insider trading, almost six years after the U.K. markets regulator sent shock waves through London with a series of dawn raids and arrests.

The five defendants, who include former bankers, accountants and day traders, are

BORIS WATCH

Boris has been out and about promoting a
NOTE demo
Be Fast
Be Reliable
Why React?

- Battle tested in production
- Great community support
- Easy to reason in terms of components
- React Native
React for your frontend
Tools

A terminal like iTerm

Sublime Text

Google Chrome (good developer tools)
Get Started

Node & NPM

Use Facebook React app builder
https://github.com/facebookincubator/create-react-app

sudo npm install -g create-react-app
npm install --save lodash

create-react-app mit_web_project
cd mit_web_project/
npm start

opens http://localhost:3000/
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <div className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h2>Welcome to React</h2>
        </div>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and save to reload.
        </p>
      </div>
    );
  }
}

export default App;
JSX

Write HTML directly in your JavaScript files

Execute JavaScript code inside the HTML using the {...} syntax
Start wireframing
Latest Comment

1/17/2016

Just added this.

Comments

1/15/2016

first!

1/16/2016

This project is super duper interesting.

1/17/2016

Just added this.

Add Comments

Submit
React is all components
Flesh things out in HTML/CSS

class App extends Component {
    render() {
        return (
            <div className="App">
                <div className="columns">
                    <div className="column">
                        <h1>Latest Comment</h1>
                        <div className="Comments">
                            <div className="Comment">
                                <div className="DateLabel">
                                    today
                                </div>
                                <div className="Text">
                                    some comment
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        )
    }
}
Break into smaller components

class Text extends Component {

  constructor(props) {
    super(props);
  }

  render() {
    return <div className="Text">
      {this.props.children}
    </div>
  }
}

render all the things
Executes every time data changes
<div className="App">
    <div className="columns">
        <div className="column">
            <h1>Latest Comment</h1>
            <Comments />
        </div>
        <div className="column">
            <h1>Comments</h1>
            <Comments />
        </div>
        <div className="column">
            <h1>Add Comments</h1>
            <AddComment />
        </div>
    </div>
</div>
Props

Accessed via `this.props`

Props are passed from component to component

Components don’t change their own props

Props are `immutable`
unidirectional data flow
Props look like HTML attributes

class Comment extends Component {

    constructor(props) {
        super(props);
    }

    render() {
        return <div className="Comment">
            <DateLabel date={this.props.date} />
            <Text>{this.props.children}</Text>
        </div>;
    }
}
class DateLabel extends Component {

  constructor(props) {
    super(props);
  }

  render() {
    var date = this.props.date;
    var dateFormatted = `\${date.getMonth()+1}/\${date.getDate()}/\${date.getFullYear()}`;  
    return <div className="DateLabel">  
      {dateFormatted}  
    </div>
  }
}
Create a Data model at the root of your app

class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      data: [
        {
          date: new Date(2016, 0, 15),
          text: "first comment",
        },
        {
          date: new Date(2016, 0, 16),
          text: "second comment",
        },
        {
          date: new Date(2016, 0, 17),
          text: "third comment",
        }
      ]
    }
  }
}
Pass the data through as props

```jsx
<div className="column">
  <h1>Comments</h1>
  <Comments data={this.state.data} />
</div>
```
Keep data at the top level
map through the data to render items

class Comments extends Component {
    constructor(props) {
        super(props);
    }

    render() {
        return <div className="Comments">
            {_.map(this.props.data, (comment) => {
                return <Comment date={comment.date}>
                    {comment.text}
                </Comment>
            })}
        </div>;
    }
}
State

Accessed via `this.state`

Changing state re-renders the component

Manipulate state in `container` components

State is `mutable`
Mutate state to re-render

```javascript
<AddComment onAddComment={this.onAddComment.bind(this)} />

</div>
</div>

);}

onAddComment(text) {
  this.setState({
    data: this.state.data.concat([{
      date: new Date(),
      text: text,
    }])
  });
}
```
Re-render on every change!
Trigger **callbacks**, using props, when the component changes
class AddComment extends Component {

    constructor(props) {
      super(props);
      this.state = {
        text: ""
      }
    }

    render() {
      return <div>
        <div className="Textarea">
          <textarea onChange={this.onUpdateText.bind(this)} value={this.state.text}>
        </textarea>
        </div>
        <Button onClick={this.onSubmitClick.bind(this)}>Submit</Button>
      </div>;
    }

    onUpdateText(e) {
      this.setState({text: e.target.value});
    }

    onSubmitClick() {
      this.props.onAddComment(this.state.text);
      // and reset the textarea
      this.setState({text: ""});
    }
  }
Structure

Separation of concern makes for better apps

Data handled from parent to child

Make self-contained components

Data flows one way
Ajax

Use fetch API

Trigger initial load from `componentDidMount`

Keep ajax calls at the root of the application
componentDidMount() {
  fetch('http://maps.googleapis.com/maps/api/geocode/json?address="boston"')
    .then((response) => {
      response.json().then((output) => {
        this.setState({ location: output.results[0] });
      });
    });
}

render() {

  let location_address = (this.state.location)
    ? this.state.location.formatted_address
    : "loading address...";

  return (
    <div className="App">
      <h1>{location_address}</h1>
    </div>
  );
}
Production

Run npm build to package up a bundle

Use Amazon S3 to host a serverless frontend
Thanks!

Alex Peretti
aperetti3@bloomberg.net