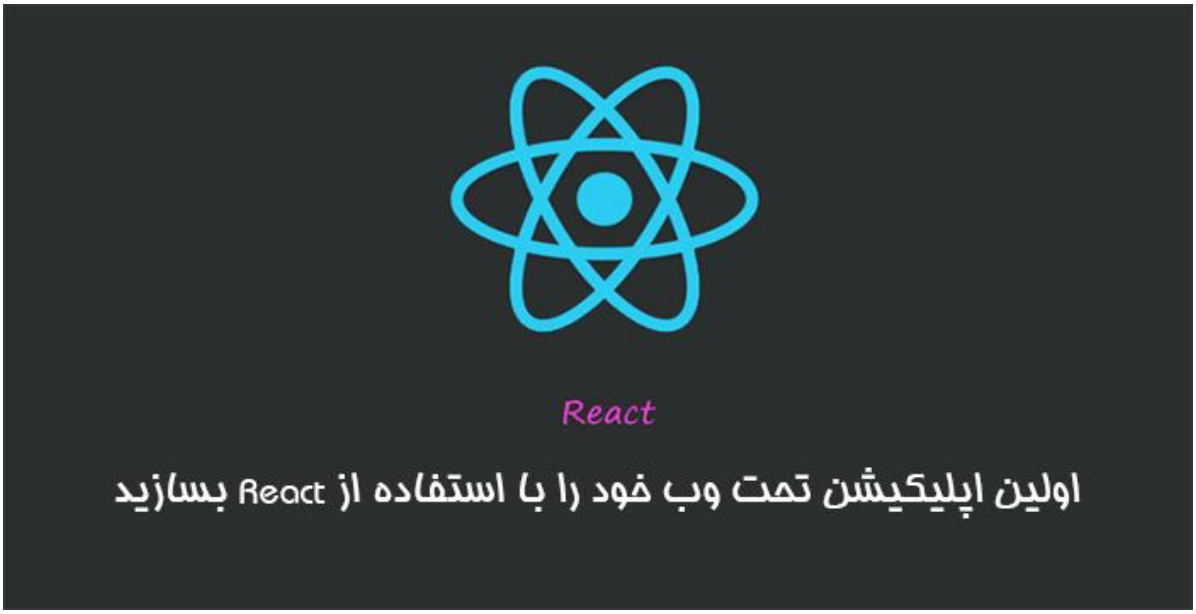
این روزها react محبوبیت زیادی پیدا کرده و اخیرا گروه زیادی از جامعه ی فعالان وب را به خودش جذب کرده است. این موضوع باعث شده که حجم زیادی از کامپوننت تکراری آماده در دسترس شما باشد تا زمان زیادی را هنگام برنامه نویسی ذخیره کنید.کتابخانه ی آن به خودی خود شما را تشویق به نوشتن تنها چند جفت کد ماژولار می کند.در این آموزش به شما نشان خواهیم داد چگونه یک برنامه ی کوچک بسازید و آنرا به اجزای جدا از هم تقسیم کنید که با یکدیگر ارتباط داشته باشند .



در ابتدا ما یک مثال آماده از روش NPM_driven مربوط به قبل را اینبار با استفاده از React انجام می دهیم . مقایسه ی این دو نتایج بسیار جالبی خواهد داشت در ورژن React خطوط برنامه ی ما بسیار کمتر از ورژن JQuery است با این حال باید قبول کنیم که از نظم بهتری نیز برخوردار است .

آنچه که باید راجع به React بدانید

-کتابخانه/فریم ورکی بسیار محبوب سمت کلاینت است و برای طراحی رابط کاربری (UI)است که توسط فیسبوک طراحی و استفاده شده است.

-با استفاده از آن برنامه ی خود را حول کامپوننت های مجزا منظم می کنید، به صورتی که هرکدام مسئول خروجی ها و حالت های خود هستند کامپوننت ها می توانند درون همدیگر باشند.

-Reactسریع است زیرا React تعداد نوشته های) DOMکند ترین قسمت هر اپلیکیشن سمت کاربر)را به حداقل می رساند.

-راه پیشنهادی برای نوشتن کدهای React با استفاده از JSX است – مشتق شده از همان javascript که المان ها را به عنوان HTML تعریف می کند. JSX باید به عنوان JS کامپایل شود (مسئول این کار خود فریم ورک می باشد) تا در مرورگرها بازشود و کار کند .(JSXیک نوع سینتکس و یک مدل نوشتن است و به شما اجازه می دهد کدهای html را در کد های جاوا سکریپت بنویسید)

-هنوز به ورژن 1.0 نرسیده در نتیجه شاید در آینده تغییراتی داشته باشد .

چه چیزی خواهیم ساخت:

ما یک اپلیکیشن تحت وب ساده خواهیم ساخت که افراد را دعوت به جستجوی موقعیت ((Location خواهد کرد و این موقعیت ها را در حافظه ی (localStorage) مرورگر آن ها ذخیره خواهد کرد . با کمک پلاگین GMaps این موقعیت ها در google map نشان داده خواهد شد. برای رابط کابری ((UI ما از Bootstrap و تم Flatly استفاده می کنیم. دراین پروسه ما برنامه را به اجزای منطقی تقسیم خواهیم کرد و ارتباط آن اجزا((component را باهم برقرار خواهیم کرد.

اجرا کردن دمو

اگر نمی خواهید تمام مراحل آموزش را بخوانید می توانید منبع کدها را از پایین دانلود کنید . برای اجرای آن نیاز به Node.js و همینطور npm دارید . با فرض اینکه هردوی آن ها را دارید باید مراحل زیر را انجام دهید:

-فایل زیپ شده زیر را دانلود کنید .

-آن را extract کنید.

-ترمینال جدیدی ((command promptباز کنید و آدرس محل extract شده را به آن بدهید .

-دستور .npm installرا اجرا کنید . این دستور تمام موارد مورد نیاز را دانلود و نصب می کند.

-دستور npm run build را اجرا کنید. اینکار اجزای React را به صورت javascript که تحت نام compiled.js در می آورد .

-Index.htmlرا در مرورگر خود بازکنید . برنامه را خواهید دید.

یک دستور دیگر npm برای شما آماده کرده ام که کار شمارا ساده تر خواهد کرد.

```
npm run watch
```

این دستور کد JSX شما را به صورت Javascript کامپایل خواهد کرد و برای تغییرات مانیتور می کند .در صورتی که فایل را تغییری دهید کدها به صورت اتوماتیک برای شما کامپایل می شوند. این دستور ها را میتوانید در فایل package.jsonببینید.

(مانیتور کردن یعنی اگرشما کدی را تغییر بدهید، بصورت اتوماتیک کد دوباره کامپایل می شود)

سورس کد ما بسیار ساده و قابل فهم است و توضیحاتی بسیاری دارد پس برای گروهی از شما که ترجیح می دهید سورس کد را بخوانید ، خواندن بقیه ی مقاله الزامی نیست.

آماده سازی

همانطور که در بالا گفته شد ، راه پیشنهادی برای نوشتن کدهای React بوسیله یک تکنیکی به نام JSX است، که لازم است در آخر به JavaScript تبدیل شود. ابزارهای مختلفی هستند که این کار را انجام می دهند اما پیشنهاد ما reactify و browserify است . با استفاده از این ها شما علاوه بر کامپایل کردن کدهای JSX تحت javascript به فراخوانی ()node.js requireهم دسترسی خواهید داشت و توانایی نصب و استفاده از کتابخانه های npm را نیز دارید.

برای راه اندازی reactify,browserify و بقیه دستور زیر را اجرا کنید

```
npm install browserify reactify watchify uglify-js react
```

برای ایجاد یک فایل Javascript آماده و که می توانید آن را آنلاین قرار دهید این دستور را در ترمینال خود اجرا کنید.

```
NODE_ENV=production browserify -t [ reactify --es6 ] main.js | uglifyjs > compiled.min.js
```

Reactify از تعداد محدودی از قابلیت های جدید ES6 تحت فلگ((flag --es6 پشتیبانی می کند .که ما در سورس کدمان استفاده کرده ایم .

در حین کدنویسی از فرمان زیر استفاده کنید :

```
watchify -v -d -t [ reactify --es6 ] main.js -o compiled.js
```

watchify فایل های شما را مانیتور می کند تا در صورت تغییر در آن ها ، دوباره کامپایل را انجام دهد .

(مانیتور کردن یعنی اگرشما کدی را تغییر بدهید، بصورت اتوماتیک کد دوباره کامپایل می شود)

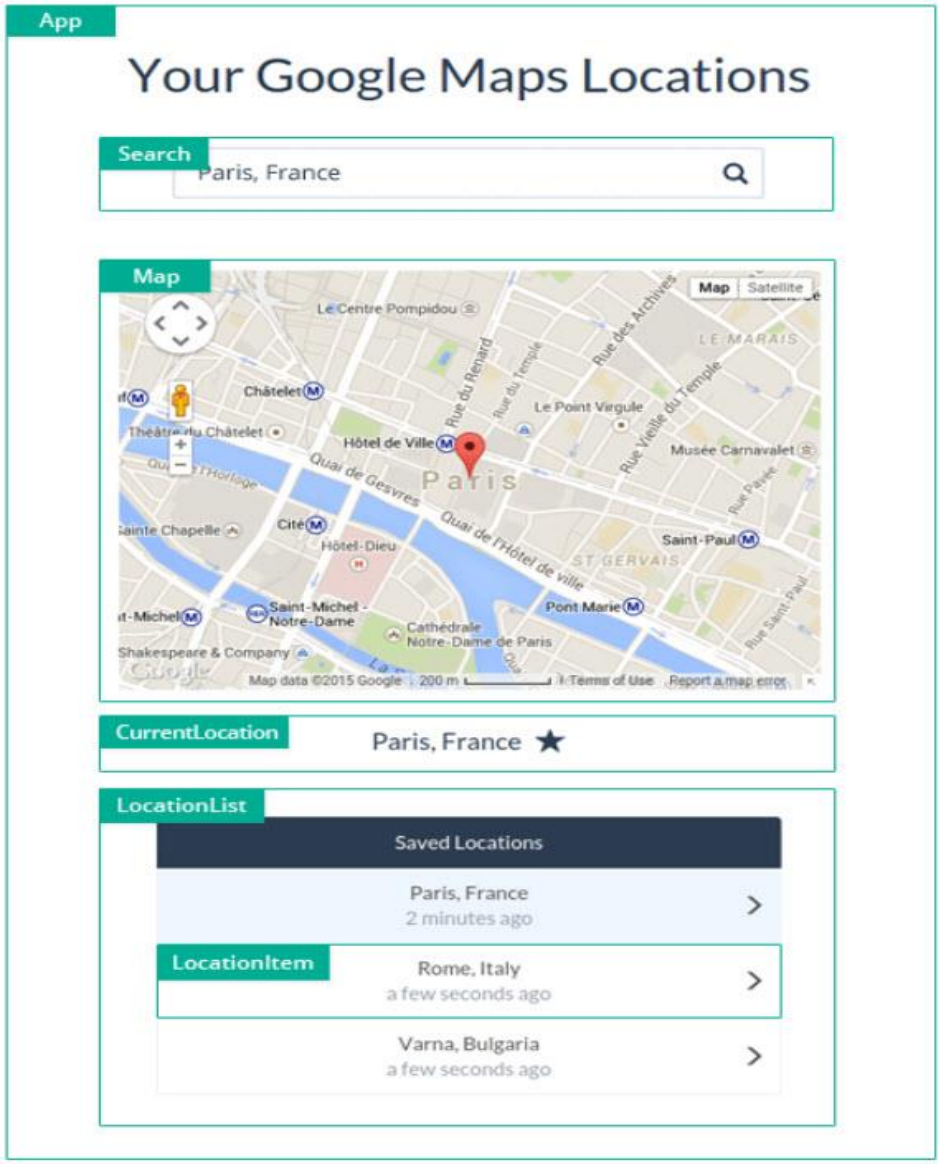
هم چنین شما می توانید با استفاده از دیباگر کروم در کد خود گشت و گذار کنید.

عالی است! حالا شما می توانید ماژول های react بنویسید، کتابخانه های npm require() وحتی برخی از قابلیت های ES6 را استفاده کنید . پس شما آماده ی کد نویسی هستید !

کدها

این ها اجزایی هستند که ما می خواهیم بنویسیم

- App که کامپوننت اصلی است. تشکیل شده از متدهایی برای کارهایی که کاربر انجام می دهد مانند سرچ کردن، اضافه کردن موقعیت های مورد علاقه و غیره . دیگر اجزا درون آن هستند.
- (CurrentLocationموقعیت کنونی که آدرس کنونی رویت شده را نشان میدهد. موقعیت ها را می توان بوسیله ی کلیک کردن روی آیکون ستاره به محبوب ها اضافه یا حذف کرد
- (LocationListلیست موقعیت ها موقعیت های مورد علاقه ی را نمایش می دهد و برای هرکدام یک LocationItem می سازد.
- LocationItemیک لوکیشن خاص است که وقتی روی آن کلیک می شود ادرس آن جست و جو می شود و در نقشه برجسته می شود.
- (Mapنقشه با کتابخانه ی google map ارتباط برقرار می کند و نقشه را نمایش می دهد .
- (Searchجستجو یک کامپوننت است که هنگامی که ادرسی در آن جستجوی شود دنبال موقعیت آن مکان می گردد.



App.js

ابتدا نوبت App است ، علاوه بر(متد های چرخه ی حیات) lifecycle methods که react لازم دارد ، موارد دیگری نیز هست که منعکس کننده دیگر عملیات های کاربر مانند اضافه یا کم کردن آدرس به آدرس های مورد علاقه و جستجو ها است . توجه داشته باشید که ما از سینتکس های کوتاه تر ES6 برای تعریف فانکشن ها در اشیاء استفاده میکنیم .

var React = require('react');

```
var Search = require('./Search');
var Map = require('./Map');
var CurrentLocation = require('./CurrentLocation');
var LocationList = require('./LocationList');

var App = React.createClass({

  getInitialState(){

    // Extract the favorite locations from local storage

    var favorites = [];

    if(localStorage.favorites){
      favorites = JSON.parse(localStorage.favorites);
    }

    // Nobody would get mad if we center it on Paris by default

    return {
      favorites: favorites,
      currentAddress: 'Paris, France',
      mapCoordinates: {
        lat: 48.856614,
        lng: 2.3522219
```

```
        }
    };
},

toggleFavorite(address){

    if(this.isAddressInFavorites(address)){
        this.removeFromFavorites(address);
    }
    else{
        this.addToFavorites(address);
    }

},

addToFavorites(address){

    var favorites = this.state.favorites;

    favorites.push({
        address: address,
        timestamp: Date.now()
    });

    this.setState({
        favorites: favorites
```

```
    });

    localStorage.favorites = JSON.stringify(favorites);
  },

  removeFromFavorites(address){

    var favorites = this.state.favorites;
    var index = -1;

    for(var i = 0; i < favorites.length; i++){

      if(favorites[i].address == address){
        index = i;
        break;
      }

    }

    // If it was found, remove it from the favorites array

    if(index !== -1){

      favorites.splice(index, 1);

      this.setState({
```

```
            favorites: favorites
        });

        localStorage.favorites = JSON.stringify(favorites);
    }

},

isAddressInFavorites(address){

    var favorites = this.state.favorites;

    for(var i = 0; i < favorites.length; i++){

        if(favorites[i].address == address){
            return true;
        }

    }

    return false;
},

searchForAddress(address){

    var self = this;
```

```
    // We will use GMaps' geocode functionality,
    // which is built on top of the Google Maps API

    GMaps.geocode({
      address: address,
      callback: function(results, status) {

        if (status !== 'OK') return;

        var latlng = results[0].geometry.location;

        self.setState({
          currentAddress: results[0].formatted_address,
          mapCoordinates: {
            lat: latlng.lat(),
            lng: latlng.lng()
          }
        });

      }
    });

  },

  render(){
```

```
    return (

      <div>
        <h1>Your Google Maps Locations</h1>

        <Search onSearch={this.searchForAddress} />

        <Map lat={this.state.mapCoordinates.lat} lng={this.state.mapCoordinates.lng} />

        <CurrentLocation address={this.state.currentAddress}
          favorite={this.isAddressInFavorites(this.state.currentAddress)}
          onFavoriteToggle={this.toggleFavorite} />

        <LocationList locations={this.state.favorites}
activeLocationAddress={this.state.currentAddress}
          onClick={this.searchForAddress} />

      </div>

    );
  }

});

module.exports = App;</span>
```

در متد render ، بقیه کامپوننت ها را مقدار دهی اولیه میکنیم (initialize) . هر کامپوننت فقط داده هایی را که برای انجام کارش نیاز دارد را دریافت میکنند .هم چنین ما داده ها را در برخی موارد به متدهایی که کامپوننت های فرزند صدا می کنند پاس می دهیم . که این روش خوبی است برای اینکه کامپوننت ها ضمن ایزوله بودن نسبت به یکدیگر،بازهم بتوانند باهم ارتباط برقرار کنند.

CurrentLocation.js

مورد بعدی موقعیت فعلی (CurrentLocation) است . این کامپوننت آدرس نمایش داده شده ی کنونی را به سایز تگ H4 نشان می دهد ، و یک آیکون ستاره ی قابل کلیک دارد . هنگامی که روی این آیکون کلیک شود متد `toggleFavorite` اپلیکیشن صدا زده می شود .

```
<span dir="rtl">var React = require('react');

var CurrentLocation = React.createClass({

  toggleFavorite(){
    this.props.onFavoriteToggle(this.props.address);
  },

  render(){

    var starClassName = "glyphicon glyphicon-star-empty";

    if(this.props.favorite){
      starClassName = "glyphicon glyphicon-star";
    }

    return (
      <div className="col-xs-12 col-md-6 col-md-offset-3 current-location">
```

```
            <h4 id="save-location">{this.props.address}</h4>
            <span  className={starClassName}  onClick={this.toggleFavorite}  aria-hidden="true"></span>
        </div>
    );
  }

});

module.exports = CurrentLocation;</span>
```

LocationList.js

لیست موقعیت ها (LocationListآرایه ای از موقعیت های مورد علاقه ای که ما به آن پاس داده ایم را میگیرد و برای هرکدام یک LocationItem ایجاد کرده و آن را در یک لیست گروپ بوت استرپی نمایش می دهد .

```
<span dir="rtl">var React = require('react');
var LocationItem = require('./LocationItem');

var LocationList = React.createClass({

  render(){

    var self = this;

    var locations = this.props.locations.map(function(l){

      var active = self.props.activeLocationAddress == l.address;
```

```
        // Notice that we are passing the onClick callback of this
        // LocationList to each LocationItem.

        return <LocationItem address={l.address} timestamp={l.timestamp}
                active={active} onClick={self.props.onClick} />
      });

      if(!locations.length){
        return null;
      }

      return (
        <div className="list-group col-xs-12 col-md-6 col-md-offset-3">
          <span className="list-group-item active">Saved Locations</span>
          {locations}
        </div>
      )

    }

});

module.exports = LocationList;</span>
```

LocationItem.js

LocationItemیک موقعیت مورد علاقه ی خاص را نمایش می دهد . و برای محاسبه طول مدتی که آن لوکیشن به محبوب ها اضافه شده است از کتابخانه ی moment استفاده می کند.

```
<span dir="rtl">var React = require('react');
var moment = require('moment');

var LocationItem = React.createClass({

  handleClick(){
    this.props.onClick(this.props.address);
  },

  render(){

    var cn = "list-group-item";

    if(this.props.active){
      cn += " active-location";
    }

    return (
      <a className={cn} onClick={this.handleClick}>
        {this.props.address}
        <span      className="createdAt">{      moment(this.props.timestamp).fromNow()
}</span>
        <span className="glyphicon glyphicon-menu-right"></span>
      </a>
```

```
    )

  }

});

module.exports = LocationItem;</span>
```

Map.js

MAP یک کامپوننت خاص است . حول پلاگین Gmaps کار میکند و از آنجایی که این یک کامپوننت تحت react نیست با وصل کردن آن به متد `componentDidUpdate` ، می توانیم یک نقشه واقعی داشته باشیم . وهرگاه که موقعیت نمایش داده شده تغییر کند میتوانیم درون div با آیدی #mapآن را ببینیم .

```
<span dir="rtl">var React = require('react');

var Map = React.createClass({

  componentDidMount(){

    // Only componentDidMount is called when the component is first added to
    // the page. This is why we are calling the following method manually.
    // This makes sure that our map initialization code is run the first time.

    this.componentDidUpdate();
  },

  componentDidUpdate(){
```

```
    if(this.lastLat == this.props.lat && this.lastLng == this.props.lng){

      // The map has already been initialized at this address.
      // Return from this method so that we don't reinitialize it
      // (and cause it to flicker).

      return;
    }

    this.lastLat = this.props.lat;
    this.lastLng = this.props.lng

    var map = new GMaps({
      el: '#map',
      lat: this.props.lat,
      lng: this.props.lng
    });

    // Adding a marker to the location we are showing

    map.addMarker({
      lat: this.props.lat,
      lng: this.props.lng
    });
  },
```

```
  render(){

    return (
      <div className="map-holder">
        <p>Loading...</p>
        <div id="map"></div>
      </div>
    );
  }

});

module.exports = Map;</span>
```

Search.js

کامپوننت جستجو(Searchاز یک فرم بوت استرپی ساخته شده است . وقتی که تایید فرم زده شود(وقتی که فرم submit شود) متد `searchForAddress` اپلیکیشن صدا زده می شود.

(هر فرمی یک submit دارد که با زدن آن ، اطلاعات به مقصد ارسال می شود)

```
<span dir="rtl">var React = require('react');

var Search = React.createClass({

  getInitialState() {
    return { value: '' };
  },
```

```
  handleChange(event) {
    this.setState({value: event.target.value});
  },

  handleSubmit(event){

    event.preventDefault();

    // When the form is submitted, call the onSearch callback that is passed to the component

    this.props.onSearch(this.state.value);

    // Unfocus the text input field
    this.getDOMNode().querySelector('input').blur();
  },

  render() {

    return (
      <form id="geocoding_form" className="form-horizontal" onSubmit={this.handleSubmit}>
        <div className="form-group">
          <div className="col-xs-12 col-md-6 col-md-offset-3">
            <div className="input-group">
              <input type="text" className="form-control" id="address" placeholder="Find a location..."
```

```
                value={this.state.value} onChange={this.handleChange} />
                <span className="input-group-btn">
                  <span className="glyphicon glyphicon-search" aria-hidden="true"></span>
                </span>
              </div>
            </div>
          </div>
        </form>
      );

  }
});

module.exports = Search;</span>
```

main.js

تنها کاری که باقیمانده اضافه کردن کامپوننت App به صفحه ی سایت است . آن را به یک div با آیدی main# اضافه می کنیم (میتوانید آن را در فایل index.html ببینید)

```
<span dir="rtl">var React = require('react');
var App = require('./components/App');

React.render(
  <App />,
  document.getElementById('main')
);</span>
```

به علاوه ی اینها کتابخانه یGmaps و JavaScript API google map را در تگ <script> نیز اضافه کرده ایم

کار ما انجام شد !

امیدوارم این آموزش به شما درک بهتری از React داده باشد . کارهای بسیار بیشتری است که شما می توانید با این کتابخانه انجام دهید از جمله رندرهای سمت سرور . که امیدواریم در آینده به آن ها بپردازیم.