JSX در ReactJS

ReactJS به جای جاوا اسکریپت معمولی جهت قالب بندی از JSX استفاده می کند. اجباری در استفاده از آن وجود ندارد، اما مزایای زیر را به همراه دارد:

- JSX سریع تر است؛ زیرا طی کامپایل کد به جاوا اسکریپت کار بهینه سازی را انجام می دهد.
- اصطلاحا type-safe است و اغلب خطاها را می توان طی کامپایل شناسایی کرد.
- نوشتن قالب ها به شرط آن که با HTML آشنا باشید را آسان تر و سریع تر می کند.

استفاده از JSX

JSX در بسیاری از موارد شبیه به HTML معمولی است وقبلا در بخش برپا کردن محیط از آن استفاده کرده ایم. به کد App.jsx نگاه کنید، در این کد div برگشت داده می شود.

App.jsx

```
import React from 'react';


class App extends React.Component {
  render() {
    return (
      <div>
        Hello World!!!
      </div>
    );
  }
}
export default App;
```

با وجود این که JSX شبیه به HTML است، اما این دو تفاوت هایی دارند که در زمان کار با JSX باید آن ها را در نظر داشت.

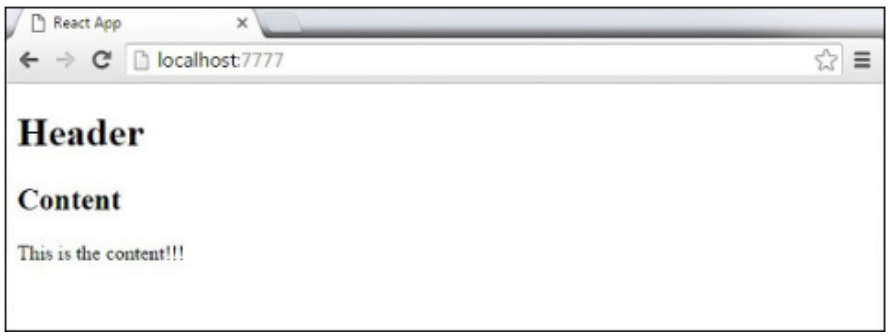
عناصر تو در تو

اگر می خواهید عناصر بیشتری را برگشت دهید، نیاز است که آن را با یک عنصر نگه دارنده بپوشانید. توجه کنید که ما چگونه از div به عنوان یک پوشاننده برای h1، h2 و p استفاده کرده ایم.

App.jsx

```
import React from 'react';


class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        <h2>Content</h2>
        <p>This is the content!!!</p>
      </div>
    );
  }
}
export default App;
```



صفات مربوط ReactJS

علاوه بر ویژگی ها و صفت های HTML معمولی می توانیم از صفات اختصاصی مخصوص به خودمان نیز استفاده کنیم. زمانی که می خواهیم صفات اختصاصی را اضافه کنیم باید از پیشوند -data استفاده کنیم. در مثال زیر ما data-myattribute را به عنوان صفتی برای عنصر p اضافه کرده ایم.

```
import React from 'react';


class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        <h2>Content</h2>
        <p data-myattribute = "somevalue">This is the content!!!</p>
      </div>
    );
  }
}
export default App;
```

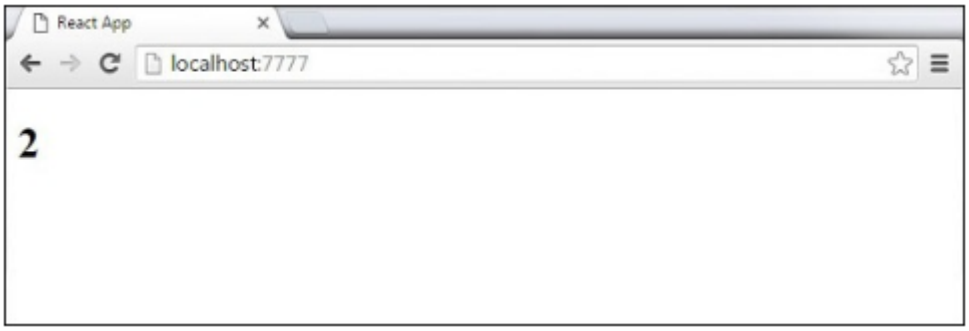
عبارت های جاوا اسکریپت

از این عبارت ها می توان داخل JSX استفاده کرد. تنها کافی است آن را داخل {} قرار دهیم. مثال زیر 2 را نمایش می دهد.

```
```

```
import React from 'react';


class App extends React.Component {
  render() {
    return (
      <div>
        <h1>{1+1}</h1>
      </div>
    );
  }
}
export default App;
```

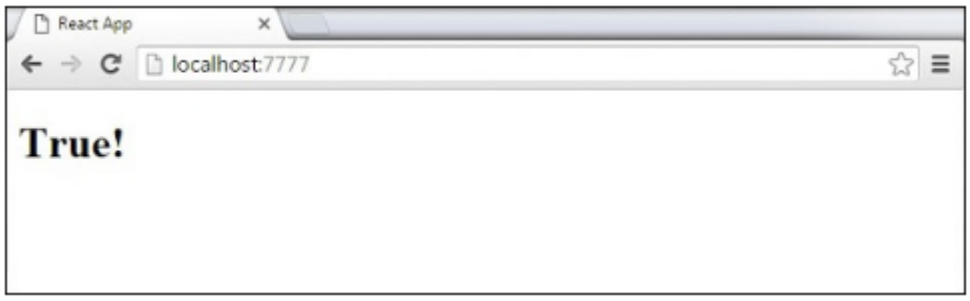


داخل JSX نمی توانیم از دستورات if else استفاده کنیم، در عوض می توانیم از عبارت های شرطی (بر مبنای 3) استفاده کنیم. در مثال زیر متغیر i برابر با 1 است. بنابراین مرورگر true را نمایش می دهد. اگر مقدار این متغیر را تغییر دهیم، مرورگر false را نمایش می دهد.

```
import React from 'react';


class App extends React.Component {
  render() {
    var i = 1;
    return (
```

```
    <div>
      <h1>{i == 1 ? 'True!' : 'False'}</h1>
    </div>
  );
 }
}
export default App;
```



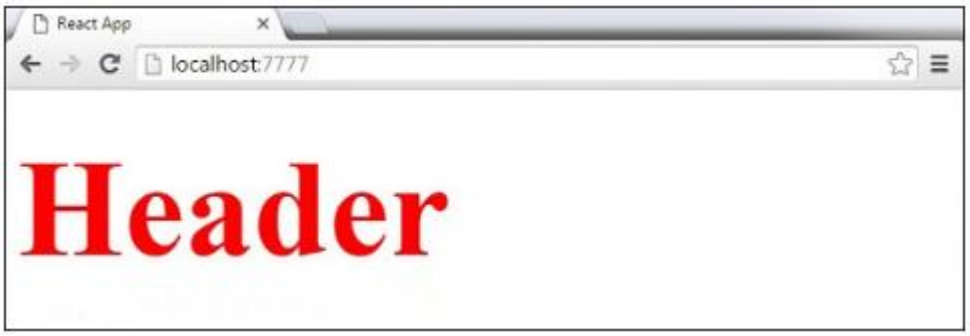
سبک بندی

ReactJS توصیه می کند از سبک های درون خطی استفاده شود. زمانی که می خواهیم این نوع از سبک ها را تنظیم کنیم، نیاز است از سینتکس camelCase استفاده کنیم. ReactJS نیز به صورت خودکار px را پس از مقدار عددی عناصر مشخص می چسباند. در مثال زیر چگونگی اضافه کردن myStyle درون خطی به عنصر h1 نشان داده شده است.

```
import React from 'react';

class App extends React.Component {
 render() {
   var myStyle = {
     fontSize: 100,
     color: '#FF0000'
   }
   return (
```

```
      <div>
        <h1 style = {myStyle}>Header</h1>
      </div>
    );
  }
}
export default App;
```



کامنت

اگر بخواهیم کامنت نویسی کنیم و بخواهیم این کامنت ها را داخل بخش فرزند یک برچسب بنویسیم، باید از {} استفاده کنیم. بهتر است همیشه در زمان نوشتن کامنت ها از {} استفاده شود، چرا که حفظ ثبات طی برنامه نویسی مهم است.

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        {//End of the line Comment...}
        {/*Multi line comment...*/}
```

```
    </div>
  );
 }
}
export default App;
```

قرارداد نام گذاری

همیشه در برچسب های HTML از اسم های برچسب با حروف کوچک استفاده می شود. این در حالی است که اجزای ReactJS با حرف بزرگ آغاز می شوند.

نکته: بهتر است که به جای class و for به عنوان اسامی صفات XML از className و htmlFor استفاده شود.

این مطلب در صفحه ی رسمی ReactJS به صورت زیر توضیح داده شده است:

با توجه به این که JSX جاوا اسکریپت است، بهتر است از شناساگرهایی مانند class و for به عنوان اسامی صفات XML استفاده نشود. در عوض بهتر است در اجزای ReactJS DOM به ترتیب از ویژگی هایی مانند className و htmlFor استفاده شود.