УДК 004.4'236

JSX – УДОБНАЯ ТЕХНОЛОГИЯ ПРИ РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ

Р.П. ГУСЕВСКИЙ
(*Представлено: канд. тех. наук, доц. И.Б. БУРАЧЕНОК*)

В статье рассмотрены особенности технологии JSX. Описан процесс транспилирования файлов формата JSX в JavaScript-код.

На первоначальном этапе своего развития «веб» состоял только из статических HTML-страниц и при открытии адреса, сервер возвращал контент страницы. Верстка, стили, скрипты – все было в общей массе. С развитием интернет-технологий стало появляться все больше одностраничных веб-приложений, где основная часть логики сосредоточена на клиентской стороне. На любой Uniform Resource Locator — унифицированный указатель ресурса (URL), сервер отдает один и тот же статичный HTML, а JavaScript-код в браузере пользователя определяет текущий адрес и делает запрос на сервер за нужными данными. При клике на ссылку, страница не перезагружается целиком, а лишь обновляется URL, а JavaScript-код делает новый запрос для следующей страницы. Появление веб-приложений в корне поменяло подход к разработке. Появились новые специальные фреймворки, предназначенные для создания веб-приложений с большим количеством кода и сложной логикой пользовательского интерфейса. Рендеринг (процесс получения изображения по модели с помощью компьютерной программы) данных теперь происходит на клиенте, в ресурсах страницы хранятся статические html-шаблоны, которые заполняются данными, полученные с сервера. Наиболее подходящим способом организации такой массы кода стал *компонентный подход*. Каждый более или менее независимый блок оформляется в отдельный компонент, со своими стилями, шаблоном и JavaScript (JS), а потом из этих блоков собирается страница целиком. Этот подход реализуется большинством современных JS-фреймворков, но с одним отличием: Angular, Ember поощряют создание отдельного файла с шаблоном, а React предлагает писать HTML внутри JS [1]. Технология, которая позволяет нам писать HTML внутри JS – это JavaScript XML (JSX) технология.

JSX – это технология, которая была представлена React. Данная технология расширяет синтаксис JavaScript, которое выглядит подобно XML. Многие думают, что использование JSX это как смесь HTML и JavaScript, но это не совсем так, мы также можем производить манипуляции и с CSS. JSX в основном используется в качестве строительных блоков в React. По своей сути JSX ни что иное как компонента, в которой мы можем контролировать и верстку, и логику, и стили. Все находится в одном месте, что уже дает огромное преимущество в разработке,

Рассмотрим пример написания HTML-элемента в JSX. Для примера рассмотрим обозначения h1-тега, который содержит в себе содержимое в виде строки, как показано на рисунке 1.

```
const element = <h1>Hello, world!</h1>
```

Рисунок 1. – H1-тег, объявленный в JSX

Данный синтаксис очень напоминает странную смесь JavaScript с HTML, но в реальности это все только JavaScript. А то, что похоже на HTML, на самом деле является «синтаксическим сахаром» для определения компонентов и их позиционирования внутри разметки. Внутри JSX выражения, атрибуты могут быть довольно легко вставлены (рисунок 2).

```
const myId = 'test'

const element = <h1 id={myId}>Hello, world!</h1>
```

Рисунок 2. – Вставка атрибутов в JSX

Самое важное знать, что когда у атрибута присутствует знак тире, то JSX конвертирует его в camelCase нотацию. Также следует помнить о двух специальных случаях: вместо class используется className, а вместо атрибута for используется htmlFor. Так как это зарезервированные слова в JavaScript.

Далее рассмотрим пример JSX, который заключает два компонента в div-тег. Часть кода JSX с двумя пользовательскими компонентами показана на рисунке 3.

```
<div>
<BlogPostsList />
<Sidebar />
</div>
```

Рисунок 3. – Часть кода JSX с двумя пользовательскими компонентами

Существует ряд правил при объявлении компонент. Тег всегда должен быть закрыт, так как это скорее XML, чем HTML. Поэтому в этом случае используется самозакрывающийся тег. Как описывалось ранее, в React в качестве компонент принято использовать файлы с расширением jsx. Как и во всех популярных фреймворках, в React можно генерировать определенную компоненту и возвращать её представление в виде html. За отображение представления в компоненте JSX отвечает функция render. У функции render, имеется одно ограничение, она может вернуть только один html-элемент. Так что в случае, когда вам надо вернуть две компоненты, нужно обязательно добавить родителя. В роли может быть вообще абсолютно любой тег. Данный случай очень часто встречается на практике и из-за этого React-разработчики придумали компоненту-обертку для таких случаев и назвали React.Fragment или пустой тег при помощи которого можно «оборачивать» один или более возвращаемых узлов (рисунок 4).

```
<React.Fragment></React.Fragment>
или
<></>
```

Рисунок 4. – Синтаксис написания обертки React.Fragment

Не маловажным аспектом является транспиляция JSX. Под транспиляцией понимается процесс преобразования кода, написанного с помощью следующих версий языка или на диалектах JavaScript в некий стандартный вариант, понимаемый всеми браузерами [2]. Браузер не может понимать JS-файлы, которые содержат JSX-код. Сначала они должны быть трансформированы в обычный JavaScript. Ранее было упомянуто что JSX опционален, потому что для каждой JSX строки, есть соответствующая JavaScript альтернатива и именно в нее трансформируется JSX. Для примера, следующие две конструкции полностью равнозначны (рисунок 5, 6).

```
ReactDOM.render(
React.DOM.div(
{ id: 'test' },
React.DOM.h1(null, 'A title'),
React.DOM.p(null, 'A paragraph')
),
document.getElementById('myapp')
)
```

Рисунок 5. – Код в формате JS

А это уже JSX:

```
ReactDOM.render(

<div id="test">

<h1>A title</h1>

<p>A paragraph</p>

</div>,

document.getElementById('myapp')

)
```

Рисунок 6. – Код в формате JSX

Первоначально рассмотрим наиболее простой случай. Сложный синтаксис на чистом JS сравним с использованием JSX. Во время написания, самым популярным способом является транспиляция с использованием Babel. Babel – это транспайлер, переписывающий код новых стандартов JavaScript в код на предыдущем стандарте ES5, который поддерживается на всех браузерах [3]. Вывод переменной JS в тэге p представлен на рисунке 7.

```
const paragraph = 'A paragraph'

ReactDOM.render(

<div id="test">

<h1>A title</h1>

<p>{paragraph}</p>

</div>,

document.getElementById('myapp')

)
```

Рисунок 7. – Вывод переменной JS в тэге p

JSX также в себя включает работу с JavaScript. Всякий раз, когда нужно в представление компоненты добавить какой-либо JS-код (функции, перебирающие методы и многое другое), его необходимо объявлять в фигурных скобках { }.

Для более простого понимания можно считать JSX как Javascript-объект. React создает элемент (JavaScript-объект) при помощи функции createElement (рисунок 8). Данные объекты носят название «React-элементы».

```
const element = React.createElement(
      'h1',
      {className: 'greeting'},
      'Hello, world!'
    );
```

Рисунок 8. – Пример создания «React-элемента»

React читает эти объекты, использует их для построения дерева DOM и хранит их в актуальном состоянии. На таких объектах и строится примитивное React-приложение.

Рассмотрим работу CSS в JSX. Без различных сторонних библиотек разработчики React навязывают написание стилей в атрибуте style. С одной стороны, это кажется ужасным решением, но если посмотреть то, как обстоит работа с CSS подробнее, то оказывается данных подход имеет свои плюсы. Во-первых, вместо принятия строки, содержащей свойства CSS, JSX атрибут style принимает только объект. Пример определения стилей в JSX показан на рисунке 9. Это означает то, что определение свойства происходит в объекте:

```
var divStyle = {

color: 'white'

}

ReactDOM.render(<div style={divStyle}>Hello World!</div>, mountNode)
```

Рисунок 9 – Пример определения стилей в JSX

CSS значения, которые объявляются в JSX, немного другие, чем чистый CSS: ключевые имена пишутся, как и атрибуты в camelCase нотации, значения – это просто строки, а также каждое «определение» отделяется при помощи запятой. Также JSX позволяет компонентам полностью инкапсулировать их стили. Данный подход покрывает примитивные случаи в объявлениях стилей, но, если описывать анимации либо медиа запросы, код становится нагроможденным при объявлении в JSX сложных CSS-объектов.

В JSX довольно частой операцией является назначение значений атрибутам. Вместо того, чтобы делать это вручную, как показано на рисунке 10.

```
<div>

<BlogPost title={data.title} date={data.date} />

</div>
```

Рисунок 10 – Назначение значений атрибутов вручную

Для того, чтобы избежать таких длительных объявлений, на помощь приходит «оператор расширения» или «spread operator». Оператор расширения – позволяет инициализировать части литерала массива из итерируемого выражения (например, другого литерала массива) или развернуть выражение в несколько аргументов (в вызовах функций) [4]. Используем оператор расширения для назначения значений атрибутов (рисунок 11). Свойства из объекта data используются как атрибуты в автоматическом порядке.

```
<div>

<BlogPost {…data} />

</div>
```

Рисунок 11 – Назначение значений атрибутов при помощи оператора расширения

Таким образом, можно сделать вывод, что JSX является очень доступной технологией в освоении и очень практичной для разработки и проектирования веб-приложений, так как приложение будет состоять из маленьких строительных блоков, так называемыми React-компонентами. И при помощи JSX формата код станет более читабельным, структурированным, а возможность хранить все в одном месте дает возможность верно и правильно закладывать архитектуру приложения под определенные потребности веб-приложения.

Заключение. При разработке программного обеспечения большинство людей находятся в поисках тех или иных технологий, которые будут подходить по их критериям. Если рассматривать данную технологию в качестве использования её вместе с ReactJS, то она имеет ряд преимуществ, которые можно дополнить различными библиотеками, что поможет быстро, а самое главное качественно разработать программное обеспечение.

ЛИТЕРАТУРА

1 Habr. Сайт интересных статей по программированию. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/311226. – Дата доступа: 24.08.2019.
2 Happy JavaScripting. Сайт для изучения JavaScript. [Электронный ресурс]. – Режим доступа: https://kodaktor.ru. – Дата доступа: 25.08.2019.
3 Тостер – вопросы и ответы для IT-специалистов [Электронный ресурс]. – Режим доступа: https://toster.ru/q/393850. – Дата доступа: 10.09.2019.
4 TechNet Microsoft [Электронный ресурс]. – Режим доступа: https://technet.microsoft.com/ru-ru/dn919259(v=vs.90). – Дата доступа: 12.09.2019.