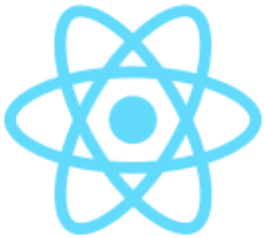
React (web framework)

React (also known as **React.js** or **ReactJS**) is a JavaScript library[3] for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.[4][5][6]

React can be used as a base in the development of single-page or mobile applications, as it is optimal for fetching rapidly changing data that needs to be recorded. However, fetching data is only the beginning of what happens on a web page, which is why complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API:[7][8] Next.js[9] and Gatsby.js[10] are examples of such libraries.

React



Original author(s)	Jordan Walke
Developer(s)	Facebook and community
Initial release	May 29, 2013[1]
Stable release	16.9.0 / August 8, 2019[2]
Repository	github.com/facebook/react (https://github.com/facebook/react)
Written in	JavaScript
Platform	Web platform
Type	JavaScript library
License	MIT License
Website	reactjs.org (http://reactjs.org)

Contents

Basic usage

The following is a rudimentary example of React usage in HTML with JSX and JavaScript.

```
<div id="myReactApp"></div>

<script type="text/babel">
  class Greeter extends React.Component {
    render() {
      return <h1>{this.props.greeting}</h1>
    }
  }
```

```
  ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));
</script>
```

The `Greeter` class is a React component that accepts a property `greeting`. The `ReactDOM.render` method creates an instance of the `Greeter` component, sets the `greeting` property to `'Hello World'` and inserts the rendered component as a child element to the DOM element with id `myReactApp`.

When displayed in a web browser the result will be

```
<div id="myReactApp">
  <h1>Hello World!</h1>
</div>
```

Notable features

Components

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props"[11]:

```
ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));
```

The two primary ways of declaring components in React is via functional components and class-based components.

Functional components

Functional components are declared with a function that then returns some JSX.

```
function Greeting(props) {
  return <div>Hello, {props.name}!</div>;
}
```

Class-based components

Class-based components are declared using ES6 classes. They are also known as "stateful" components, because their state can hold values throughout the component and can be passed to child components through props:

```
class ParentComponent extends React.Component {
  state = { color: 'green' };
  render() {
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Virtual DOM

Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently.[12] This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change.

Lifecycle methods

Lifecycle methods are hooks that allow execution of code at set points during a component's lifetime.

- `shouldComponentUpdate` allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
- `componentDidMount` is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
- `componentWillUnmount` is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any `setInterval()` instances that are related to the component, or an "eventListener" set on the "document" because of the presence of the component)
- `render` is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

JSX

JSX, or JavaScript XML , is an extension to the JavaScript language syntax.[13] Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP.

An example of JSX code:

```
 1 class App extends React.Component {
 2   render() {
 3     return (
 4       <div>
 5         <p>Header</p>
 6         <p>Content</p>
 7         <p>Footer</p>
 8       </div>
 9     );
10   }
11 }
```

Nested elements

Multiple elements on the same level need to be wrapped in a single container element such as the `<div>` element shown above, or returned as an array.[14]

Attributes

JSX provides a range of element attributes designed to mirror those provided by HTML. Custom attributes can also be passed to the component.[15] All attributes will be received by the component as props.

JavaScript expressions

JavaScript expressions (but not statements) can be used inside JSX with curly brackets `{}`:

```
  <h1>{10+1}</h1>
```

The example above will render

```
  <h1>11</h1>
```

Conditional statements

If–else statements cannot be used inside JSX but conditional expressions can be used instead. The example below will render `{ i === 1 ? 'true' : 'false' }` as the string `'true'` because `i` is equal to 1.

```
 1 class App extends React.Component {
 2   render() {
 3     const i = 1;
 4     return (
 5       <div>
 6         <h1>{ i === 1 ? 'true' : 'false' }</h1>
 7       </div>
 8     );
 9   }
10 }
```

The above will render:

```
<div>
  <h1>true</h1>
</div>
```

Functions and JSX can be used in conditionals:

```
 1 class App extends React.Component {
 2   render() {
 3     const sections = [1, 2, 3];
 4     return (
 5       <div>
 6         {sections.length > 0 && sections.map(n => (
 7             /* 'key' is used by react to keep track of list items and their changes */
 8             /* Each 'key' must be unique */
 9             <div key={"section-" + n}>Section {n}</div>
10         ))}
11       </div>
12     );
13   }
14 }
```

The above will render:

```
<div>
  <div>Section 1</div>
  <div>Section 2</div>
  <div>Section 3</div>
</div>
```

Code written in JSX requires conversion with a tool such as Babel before it can be understood by web browsers.[16] This processing is generally performed during a software build process before the application is deployed.

Architecture beyond HTML

The basic architecture of React applies beyond rendering HTML in the browser. For example, Facebook has dynamic charts that render to `<canvas>` tags,[17] and Netflix and PayPal use universal loading to render identical HTML on both the server and client.[18][19]

React Hooks

Hooks are functions that let developers "hook into" React state and lifecycle features from function components. Hooks don't work inside classes — they let you use React without classes.[20]

React provides a few built-in Hooks like `useState`. You can also create your own Hooks to reuse stateful behavior between different components.[21]

Common idioms

React does not attempt to provide a complete "application framework". It is designed specifically for building user interfaces[3] and therefore does not include many of the tools some developers might consider necessary to build an application. This allows the choice of whichever libraries the developer prefers to accomplish tasks such as performing network access or local data storage. Common patterns of usage have emerged as the library matures.

Use of the Flux architecture

To support React's concept of unidirectional data flow (which might be contrasted with AngularJS's bidirectional flow), the Flux architecture represents an alternative to the popular model-view-controller architecture. Flux features *actions* which are sent through a central *dispatcher* to a *store*, and changes to the store are propagated back to the view.[22] When used with React, this propagation is accomplished through component properties.

Flux can be considered a variant of the observer pattern.[23]

A React component under the Flux architecture should not directly modify any props passed to it, but should be passed callback functions that create *actions* which are sent by the dispatcher to modify the store. The action is an object whose responsibility is to describe what has taken place: for example, an action describing one user "following" another might contain a user id, a target user id, and the type `USER_FOLLOWED_ANOTHER_USER`.[24] The stores, which can be thought of as models, can alter themselves in response to actions received from the dispatcher.

This pattern is sometimes expressed as "properties flow down, actions flow up". Many implementations of Flux have been created since its inception, perhaps the most well-known being Redux, which features a single store, often called a single source of truth.[25]

Future development

Project status can be tracked via the core team discussion forum.[26] However, major changes to React go through the Future of React repository issues and pull requests.[27][28] This enables the React community to provide feedback on new potential features, experimental APIs and JavaScript syntax improvements.

History

React was created by Jordan Walke, a software engineer at Facebook, who released an early prototype of React called "FaxJS".[29][30] He was influenced by XHP, an HTML component framework for PHP. It was first deployed on Facebook's News Feed in 2011 and later on Instagram in 2012.[31] It was open-sourced at JSConf US in May 2013.[30]

React Native, which enables native Android, iOS, and UWP development with React, was announced at Facebook's React Conf in February 2015 and open-sourced in March 2015.

On April 18, 2017, Facebook announced React Fiber, a new core algorithm of React framework library for building user interfaces.[32] React Fiber was to become the foundation of any future improvements and feature development of the React framework.[33]

On September 26, 2017, React 16.0 was released to the public.[34]

On February 16, 2019, React 16.8 was released to the public.[35] The release introduced React Hooks.[36]

Licensing

The initial public release of React in May 2013 used the Apache License 2.0. In October 2014, React 0.12.0 replaced this with the 3-clause BSD license and added a separate PATENTS text file that permits usage of any Facebook patents related to the software:[37]

> The license granted hereunder will terminate, automatically and without notice, for anyone that makes any claim (including by filing any lawsuit, assertion or other action) alleging (a) direct, indirect, or contributory infringement or inducement to infringe any patent: (i) by Facebook or any of its subsidiaries or affiliates, whether or not such claim is related to the Software, (ii) by any party if such claim arises in whole or in part from any software, product or service of Facebook or any of its subsidiaries or affiliates, whether or not such claim is related to the Software, or (iii) by any party relating to the Software; or (b) that any right in any patent claim of Facebook is invalid or unenforceable.

This unconventional clause caused some controversy and debate in the React user community, because it could be interpreted to empower Facebook to revoke the license in many scenarios, for example, if Facebook sues the licensee prompting them to take "other action" by publishing the action on a blog or elsewhere. Many expressed concerns that Facebook could unfairly exploit the termination clause or that integrating React into a product might complicate a startup company's future acquisition.[38]

Based on community feedback, Facebook updated the patent grant in April 2015 to be less ambiguous and more permissive:[39]

> The license granted hereunder will terminate, automatically and without notice, if you (or any of your subsidiaries, corporate affiliates or agents) initiate directly or indirectly, or take a direct financial interest in, any Patent Assertion: (i) against Facebook or any of its subsidiaries or corporate affiliates, (ii) against any party if such Patent Assertion arises in whole or in part from any software, technology, product or service of Facebook or any of its subsidiaries or corporate affiliates, or (iii) against any party relating to the Software. [...] A "Patent Assertion" is any lawsuit or other action alleging direct, indirect, or contributory infringement or inducement to infringe any patent, including a cross-claim or counterclaim.[40]

The Apache Software Foundation considered this licensing arrangement to be incompatible with its licensing policies, as it "passes along risk to downstream consumers of our software imbalanced in favor of the licensor, not the licensee, thereby violating our Apache legal policy of being a universal donor", and "are not a subset of those found in the [Apache License 2.0], and they cannot be sublicensed as [Apache License 2.0]".[41] In August 2017, Facebook dismissed the Apache Foundation's downstream concerns and refused to reconsider their license.[42][43] The following month, WordPress decided to switch its Gutenberg and Calypso projects away from React.[44]

On September 23, 2017, Facebook announced that the following week, it would re-license Flow, Jest, React, and Immutable.js under a standard MIT License; the company stated that React was "the foundation of a broad ecosystem of open source software for the web", and that they did not want to "hold back forward progress for nontechnical reasons".[45]

On September 26, 2017, React 16.0.0 was released with the MIT license.[46] The MIT license change has also been backported to the 15.x release line with React 15.6.2.[47]

See also

- AngularJS
- Angular
- Vue.js
- Comparison of JavaScript frameworks
- React Native
- Backbone.js
- Ember.js

References

1. Occhino, Tom; Walke, Jordan. "JS Apps at Facebook" (https://www.youtube.com/watch?v=GW0rj4sNH2w). *YouTube*. Retrieved 22 Oct 2018.
2. "Releases – Facebook/React" (https://github.com/facebook/react/releases). *GitHub*.
3. "React - A JavaScript library for building user interfaces" (https://reactjs.org). *React*. Retrieved 7 April 2018.
4. Krill, Paul (May 15, 2014). "React: Making faster, smoother UIs for data-driven Web apps" (https://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html). *InfoWorld*.
5. Hemel, Zef (June 3, 2013). "Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews" (https://www.infoq.com/news/2013/06/facebook-react). *InfoQ*.
6. Dawson, Chris (July 25, 2014). "JavaScript's History and How it Led To ReactJS" (https://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/). *The New Stack*.
7. Dere, Mohan (2018-02-19). "How to integrate create-react-app with all the libraries you need to make a great app" (https://medium.freecodecamp.org/integrating-create-react-app-redux-react-router-redux-observable-bootstrap-altogether-216db97e89a3). *freeCodeCamp*. Retrieved 2018-06-14.
8. Samp, Jon (2018-01-13). "React Router to Redux First Router" (https://medium.com/about-codecademy/react-router-to-redux-first-router-2fea05c4c2b7). *About Codecademy*. Retrieved 2018-06-14.
9. "The React Framework" (https://nextjs.org/). *Nextjs.org*. 2019-08-14.
10. "Blazing fast modern site generator for React" (https://gatsbyjs.org/). *Gatsbyjs.org*. 2019-08-14.
11. "Components and Props" (https://reactjs.org/docs/components-and-props.html#props-are-read-only). *React*. Facebook. Retrieved 7 April 2018.
12. "Refs and the DOM" (https://reactjs.org/docs/refs-and-the-dom.html). *React Blog*.
13. "Draft: JSX Specification" (https://facebook.github.io/jsx/). *JSX*. Facebook. Retrieved 7 April 2018.
14. Clark, Andrew (September 26, 2017). "React v16.0§New render return types: fragments and strings" (https://reactjs.org/blog/2017/09/26/react-v16.0.html#new-render-return-types-fragments-and-strings). *React Blog*.
15. Clark, Andrew (September 26, 2017). "React v16.0§Support for custom DOM attributes" (https://reactjs.org/blog/2017/09/26/react-v16.0.html#support-for-custom-dom-attributes). *React Blog*.
16. Fischer, Ludovico (2017-09-06). *React for Real: Front-End Code, Untangled* (https://books.google.com/books?id=Tg9QDwAAQBAJ). Pragmatic Bookshelf. ISBN 9781680504484.
17. "Why did we build React? – React Blog" (https://facebook.github.io/react/blog/2013/06/05/why-react.html).
18. "PayPal Isomorphic React" (https://www.paypal-engineering.com/2015/04/27/isomorphic-react-apps-with-react-engine/).

19. "Netflix Isomorphic React" (http://techblog.netflix.com/2015/01/netflix-likes-react.html).
20. "Hooks at a Glance – React" (https://reactjs.org/docs/hooks-overview.html). *reactjs.org*. Retrieved 2019-08-08.
21. "Hooks at a Glance – React" (https://reactjs.org/docs/hooks-overview.html). *reactjs.org*. Retrieved 2019-08-08.
22. "In Depth OverView" (https://facebook.github.io/flux/docs/in-depth-overview.html). *Flux*. Facebook. Retrieved 7 April 2018.
23. Johnson, Nicholas. "Introduction to Flux - React Exercise" (http://nicholasjohnson.com/react/course/exercises/flux/). *Nicholas Johnson*. Retrieved 7 April 2018.
24. Abramov, Dan. "The History of React and Flux with Dan Abramov" (http://threedevsandamaybe.com/the-history-of-react-and-flux-with-dan-abramov/). *Three Devs and a Maybe*. Retrieved 7 April 2018.
25. "State Management Tools - Results" (https://stateofjs.com/2017/state-management/results). *The State of JavaScript*. Retrieved 7 April 2018.
26. "Meeting Notes" (https://discuss.reactjs.org/c/meeting-notes). *React Discuss*. Retrieved 2015-12-13.
27. "reactjs/react-future - The Future of React" (https://github.com/reactjs/react-future). *GitHub*. Retrieved 2015-12-13.
28. "facebook/react - Feature request issues" (https://github.com/facebook/react/labels/Type:%20Feature%20Request). *GitHub*. Retrieved 2015-12-13.
29. Walke, Jordan. "FaxJS" (https://github.com/jordwalke/FaxJs). Retrieved 11 July 2019.
30. Papp, Andrea (4 April 2018). "The History of React.js on a Timeline" (https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/). *RisingStack*. Retrieved 11 July 2019.
31. "Pete Hunt at TXJS" (https://www.youtube.com/watch?v=A0Kj49z6WdM).
32. Frederic Lardinois (18 April 2017). "Facebook announces React Fiber, a rewrite of its React framework" (https://techcrunch.com/2017/04/18/facebook-announces-react-fiber-a-rewrite-of-its-react-framework/). TechCrunch. Retrieved 19 April 2017.
33. "React Fiber Architecture" (https://github.com/acdlite/react-fiber-architecture). *Github*. Retrieved 19 April 2017.
34. "'React v16.0" (https://reactjs.org/blog/2017/09/26/react-v16.0.html). react.js. 2017-09-26. Retrieved 2019-05-20.
35. "'React v16.8" (https://reactjs.org/blog/2017/09/26/react-v16.0.html). react.js. 2019-02-16. Retrieved 2019-05-20.
36. "Introducing Hooks" (https://reactjs.org/docs/hooks-intro.html). react.js. Retrieved 2019-05-20.
37. "React CHANGELOG.md" (https://github.com/facebook/react/blob/master/CHANGELOG.md#0120-october-28-2014). *GitHub*.
38. Liu, Austin. "A compelling reason not to use ReactJS" (https://medium.com/bits-and-pixels/a-compelling-reason-not-to-use-reactjs-beac24402f7b). *Medium*.
39. "Updating Our Open Source Patent Grant" (https://code.facebook.com/posts/1639473982937255/updating-our-open-source-patent-grant/).
40. "Additional Grant of Patent Rights Version 2" (https://github.com/facebook/react/blob/b8ba8c83f318b84e42933f6928f231dc0918f864/PATENTS). *GitHub*.
41. "ASF Legal Previously Asked Questions" (https://www.apache.org/legal/resolved.html). Apache Software Foundation. Retrieved 2017-07-16.
42. "Explaining React's License" (https://code.facebook.com/posts/112130496157735/explaining-react-s-license/). *Facebook*. Retrieved 2017-08-18.
43. "Consider re-licensing to AL v2.0, as RocksDB has just done" (https://github.com/facebook/react/issues/10191#issuecomment-323486580). *Github*. Retrieved 2017-08-18.
44. "WordPress to ditch React library over Facebook patent clause risk" (https://techcrunch.com/2017/09/15/wordpress-to-ditch-react-library-over-facebook-patent-clause-risk/). *TechCrunch*. Retrieved 2017-09-16.
45. "Relicensing React, Jest, Flow, and Immutable.js" (https://code.facebook.com/posts/300798627056246/relicensing-react-jest-flow-and-immutable-js/). *Facebook Code*. 2017-09-23.
46. Clark, Andrew (September 26, 2017). "React v16.0§MIT licensed" (https://reactjs.org/blog/2017/09/26/react-v16.0.html#mit-licensed). *React Blog*.
47. Hunzaker, Nathan (September 25, 2017). "React v15.6.2" (https://reactjs.org/blog/2017/09/25/react-v15.6.2.html). *React Blog*.

External links

- Official website (https://reactjs.org/)
- React Example (https://reactjsexample.com)