

La bibliothèque React

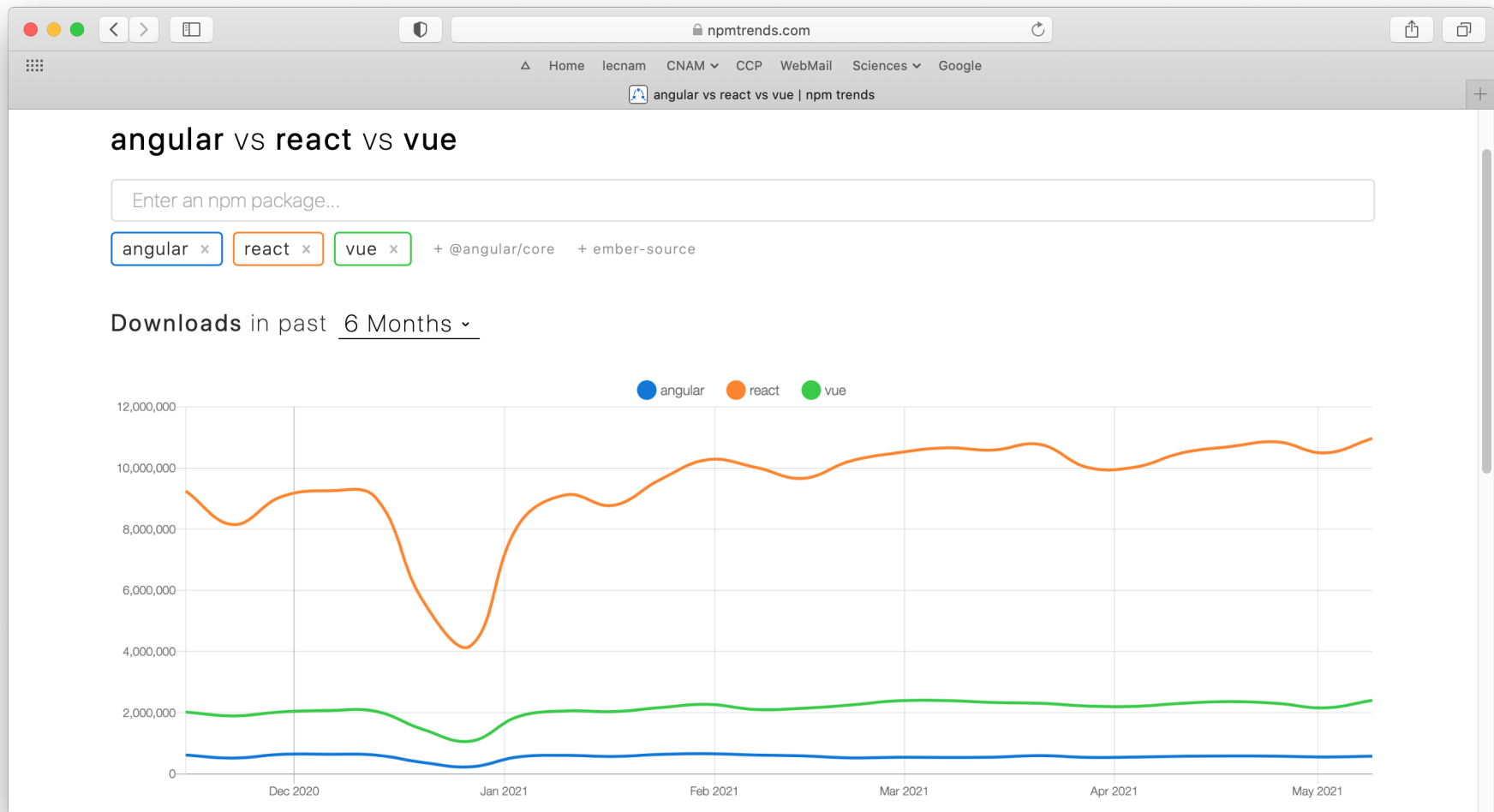
Tristan Crolard

Laboratoire CEDRIC
Equipe « Systèmes Sûrs »

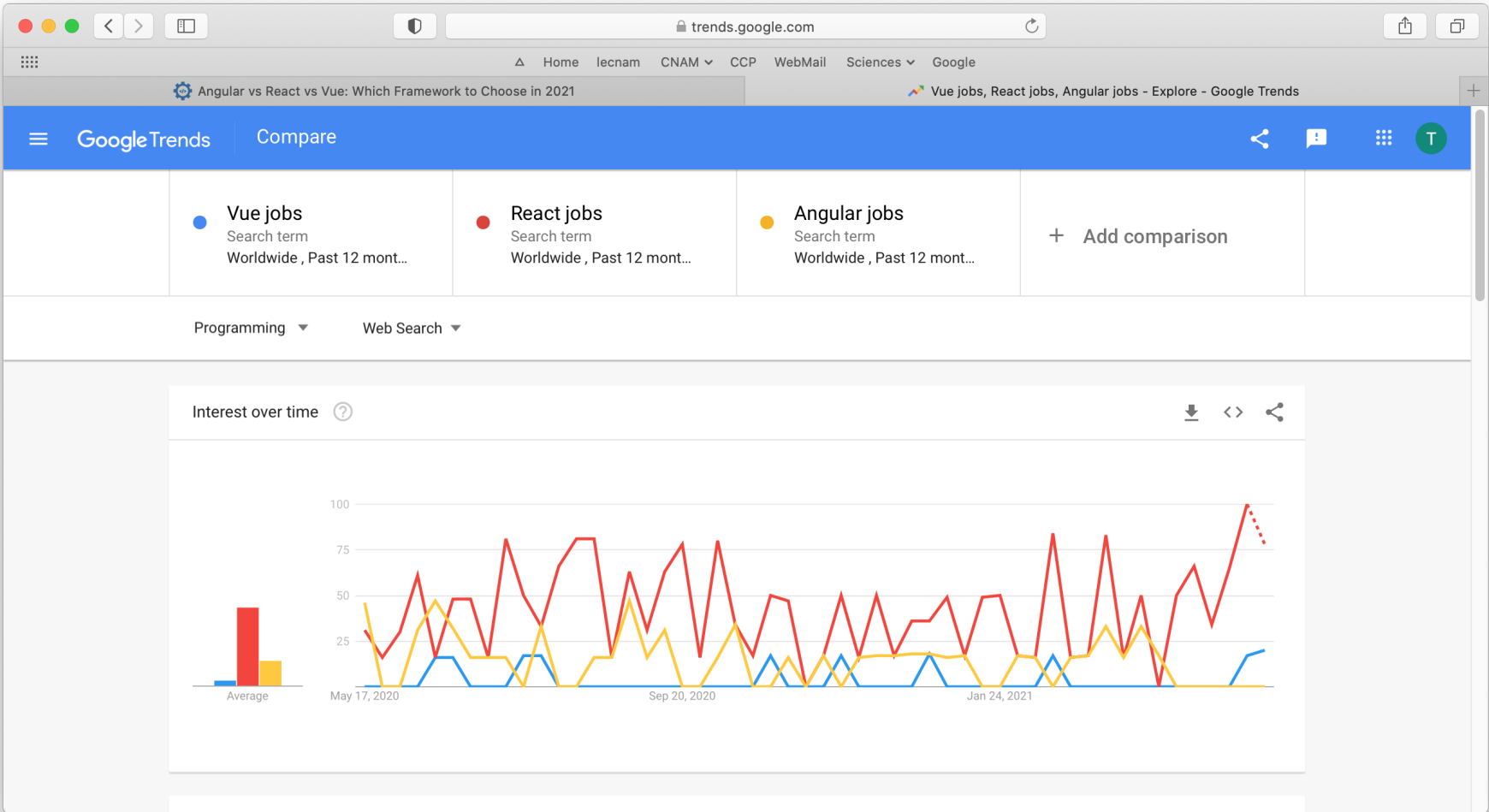
`tristan.crolard@cnam.fr`

`cedric.cnam.fr/sys/crolard`

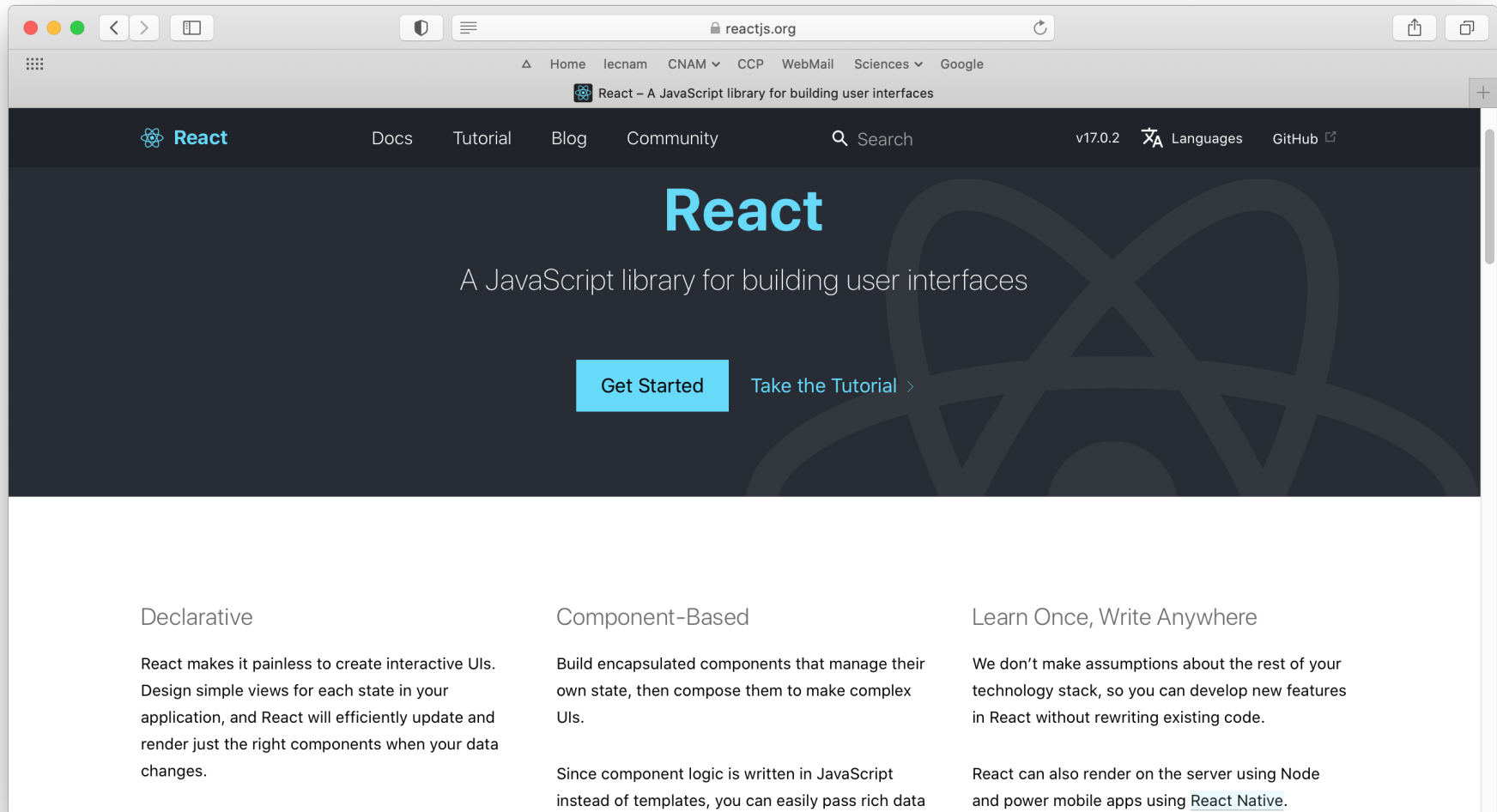
Les principales bibliothèques (*downloads*)



Les principales bibliothèques (*jobs*)



La bibliothèque React



Principe de base (React-Redux)

The screenshot shows the Redux.js website with the following content:

- Navigation:** Getting Started, Tutorial, API, FAQ, Best Practices, GitHub, Need help?
- Search:** Search [K]
- Left Sidebar:**
 - Introduction
 - Getting Started with Redux
 - Installation
 - Core Concepts
 - Learning Resources
 - Ecosystem
 - Examples
 - Tutorials
 - Tutorials Index
 - Redux Essentials
 - Redux Overview and Concepts
 - Redux App Structure
 - Basic Redux Data Flow
 - Using Redux Data
 - Async Logic and Data Fetching
 - Performance and Normalizing Data
 - Redux Fundamentals >
 - Recipes >
- Main Content:**

This is a small example of "one-way data flow":

 - State describes the condition of the app at a specific point in time
 - The UI is rendered based on that state
 - When something happens (such as a user clicking a button), the state is updated based on what occurred
 - The UI re-renders based on the new state
- Diagram:** A circular diagram showing the flow of data. A red circle labeled 'Actions' has a dashed arrow pointing to a purple circle labeled 'State'. A dashed arrow points from 'State' to a green circle labeled 'View'. A dashed arrow points from 'View' back to 'Actions', completing the cycle.
- Right Sidebar:**
 - Introduction
 - How to Read This Tutorial
 - What is Redux?
 - Why Should I Use Redux?
 - When Should I Use Redux?
 - Redux Libraries and Tools
 - Redux Terms and Concepts
 - State Management
 - Immutability
 - Terminology
 - Redux Application Data Flow
 - What You've Learned
 - What's Next?

React Hooks

The screenshot shows a web browser window displaying the React Hooks API Reference page. The browser's address bar shows 'reactjs.org'. The page has a dark navigation bar with the React logo, 'Docs' (highlighted), 'Tutorial', 'Blog', and 'Community' links, a search icon, and version information 'v17.0.2'. The main content area features the title 'Hooks API Reference' and an introductory paragraph: 'Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.' Below this, it states 'This page describes the APIs for the built-in Hooks in React.' and suggests checking the 'overview' and 'frequently asked questions' sections. A bulleted list of 'Basic Hooks' includes 'useState', 'useEffect', and 'useContext', and 'Additional Hooks' includes 'useReducer'. A right-hand sidebar contains a table of contents with sections like 'INSTALLATION', 'MAIN CONCEPTS', 'ADVANCED GUIDES', 'API REFERENCE', 'HOOKS' (expanded to show a list of 8 items, with '7. Hooks API Reference' highlighted), and 'TESTING'.

reactjs.org

Home lecnam CNAM CCP WebMail Sciences Google

Hooks API Reference – React

React Docs Tutorial Blog Community Search v17.0.2 Languages GitHub

Hooks API Reference

Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class.

This page describes the APIs for the built-in Hooks in React.

If you're new to Hooks, you might want to check out [the overview](#) first. You may also find useful information in the [frequently asked questions](#) section.

- [Basic Hooks](#)
 - [useState](#)
 - [useEffect](#)
 - [useContext](#)
- [Additional Hooks](#)
 - [useReducer](#)

INSTALLATION ▾

MAIN CONCEPTS ▾

ADVANCED GUIDES ▾

API REFERENCE ▾

HOOKS ^

1. Introducing Hooks
2. Hooks at a Glance
3. Using the State Hook
4. Using the Effect Hook
5. Rules of Hooks
6. Building Your Own Hooks
- 7. Hooks API Reference**
8. Hooks FAQ

TESTING ▾

React Hooks : *useState*

The image shows a browser window displaying the React Hooks API Reference page for `useState`. The page is titled "Basic Hooks" and "useState". It includes a code snippet for the `useState` hook, a description of its return value, and a description of the `setState` function. A sidebar on the right contains a table of contents with "7. Hooks API Reference" highlighted.

reactjs.org

Home lecnam CNAM CCP WebMail Google Scholar

Hooks API Reference – React

React Docs Tutorial Blog Community Search v17.0.2 Languages GitHub

Basic Hooks

useState

```
const [state, setState] = useState(initialState);
```

Returns a stateful value, and a function to update it.

During the initial render, the returned state (`state`) is the same as the value passed as the first argument (`initialState`).

The `setState` function is used to update the state. It accepts a new state value and enqueues a re-render of the component.

```
setState(newState);
```

During subsequent re-renders, the first value returned by `useState` will always be the most recent state after applying updates.

INSTALLATION ▾

MAIN CONCEPTS ▾

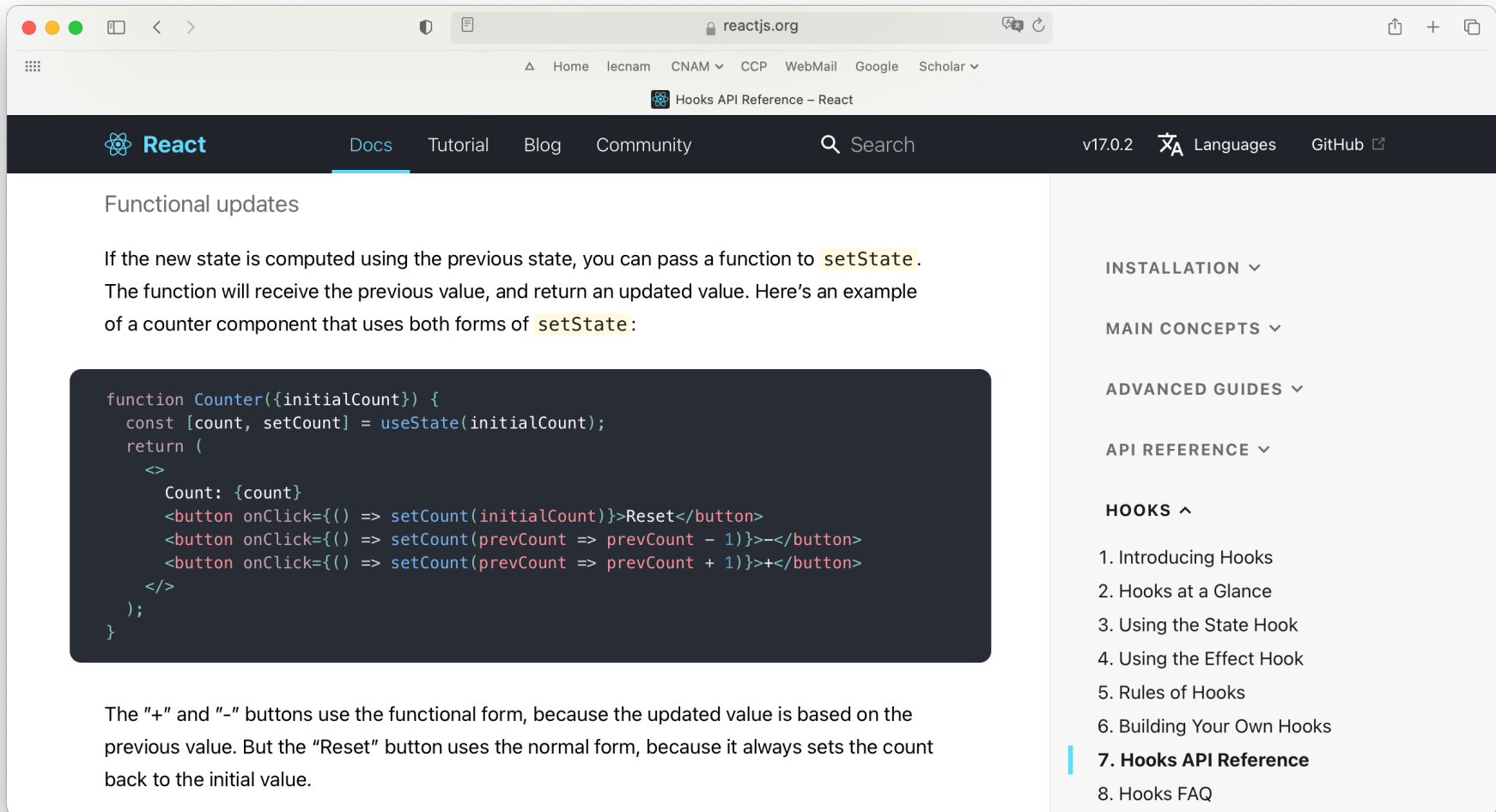
ADVANCED GUIDES ▾

API REFERENCE ▾

HOOKS ▲

1. Introducing Hooks
2. Hooks at a Glance
3. Using the State Hook
4. Using the Effect Hook
5. Rules of Hooks
6. Building Your Own Hooks
- 7. Hooks API Reference**
8. Hooks FAQ

React *Hooks* : *functional updates*



The screenshot shows a browser window at reactjs.org. The page title is "Hooks API Reference - React". The navigation bar includes "React", "Docs", "Tutorial", "Blog", "Community", "Search", "v17.0.2", "Languages", and "GitHub". The main content area is titled "Functional updates" and contains the following text:

If the new state is computed using the previous state, you can pass a function to `setState`. The function will receive the previous value, and return an updated value. Here's an example of a counter component that uses both forms of `setState`:

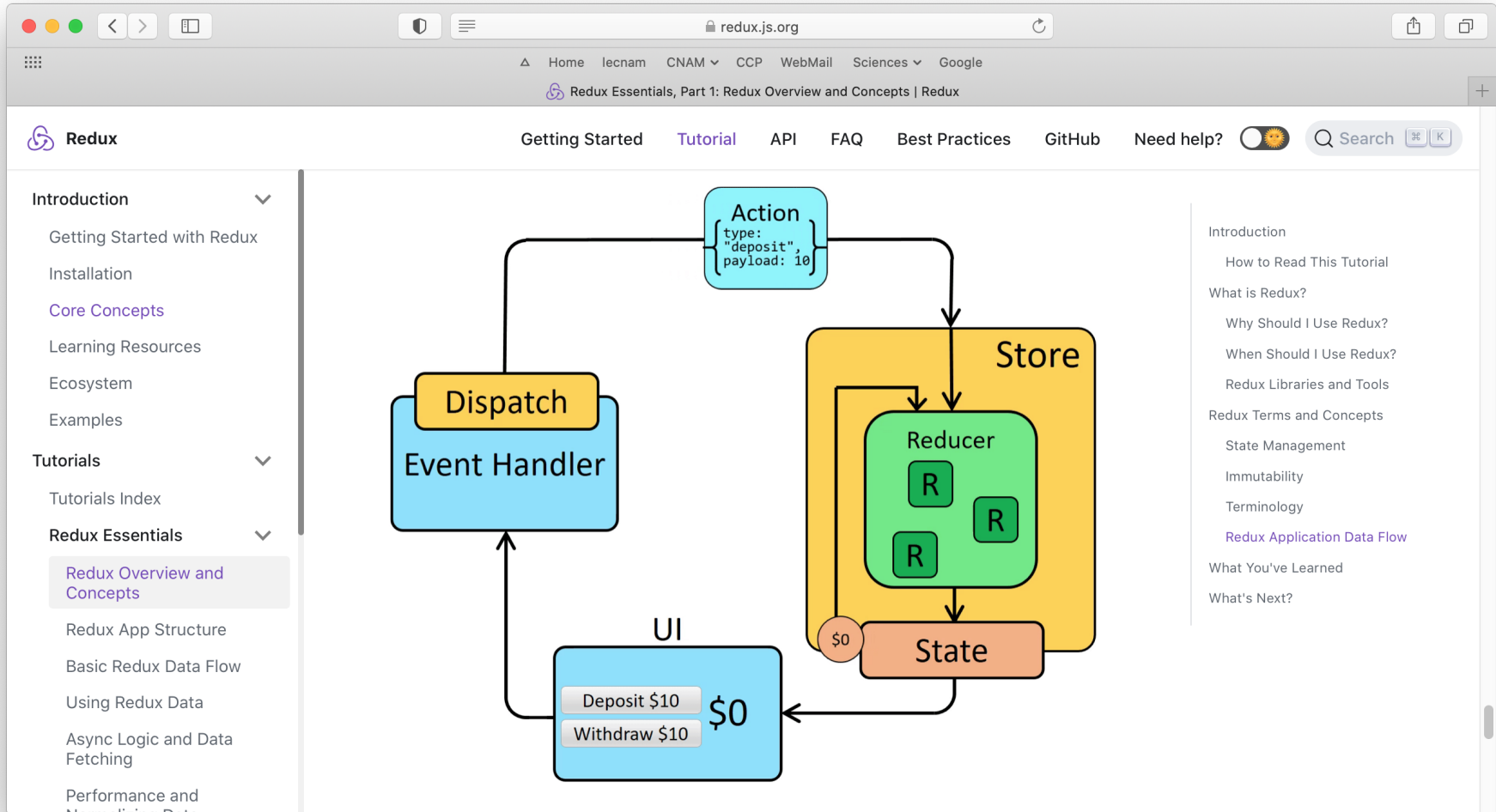
```
function Counter({initialCount}) {
  const [count, setCount] = useState(initialCount);
  return (
    <>
      Count: {count}
      <button onClick={() => setCount(initialCount)}>Reset</button>
      <button onClick={() => setCount(prevCount => prevCount - 1)}>-</button>
      <button onClick={() => setCount(prevCount => prevCount + 1)}>+</button>
    </>
  );
}
```

The "+" and "-" buttons use the functional form, because the updated value is based on the previous value. But the "Reset" button uses the normal form, because it always sets the count back to the initial value.

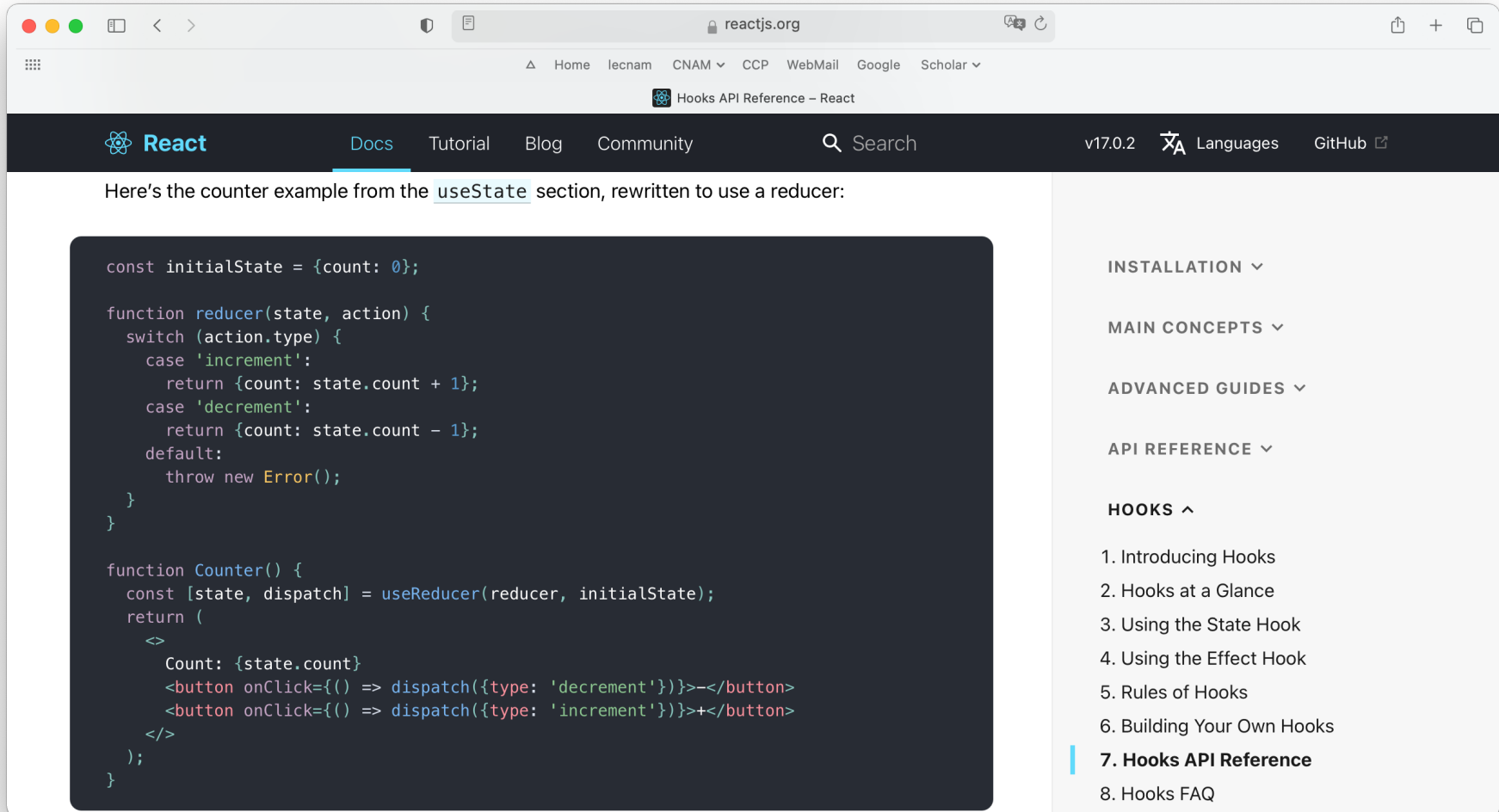
The right sidebar contains a table of contents with the following items:

- INSTALLATION ▾
- MAIN CONCEPTS ▾
- ADVANCED GUIDES ▾
- API REFERENCE ▾
- HOOKS ^
 1. Introducing Hooks
 2. Hooks at a Glance
 3. Using the State Hook
 4. Using the Effect Hook
 5. Rules of Hooks
 6. Building Your Own Hooks
 - 7. Hooks API Reference**
 8. Hooks FAQ

Architecture typique (React-Redux)



React *Hooks* : useReducer (Dispatch)



The screenshot shows a web browser window displaying the React.js website. The address bar shows 'reactjs.org'. The page title is 'Hooks API Reference - React'. The navigation bar includes 'React', 'Docs', 'Tutorial', 'Blog', 'Community', 'Search', 'v17.0.2', 'Languages', and 'GitHub'. The main content area features a text block: 'Here's the counter example from the [useState](#) section, rewritten to use a reducer:'. Below this is a dark-themed code block containing the following code:

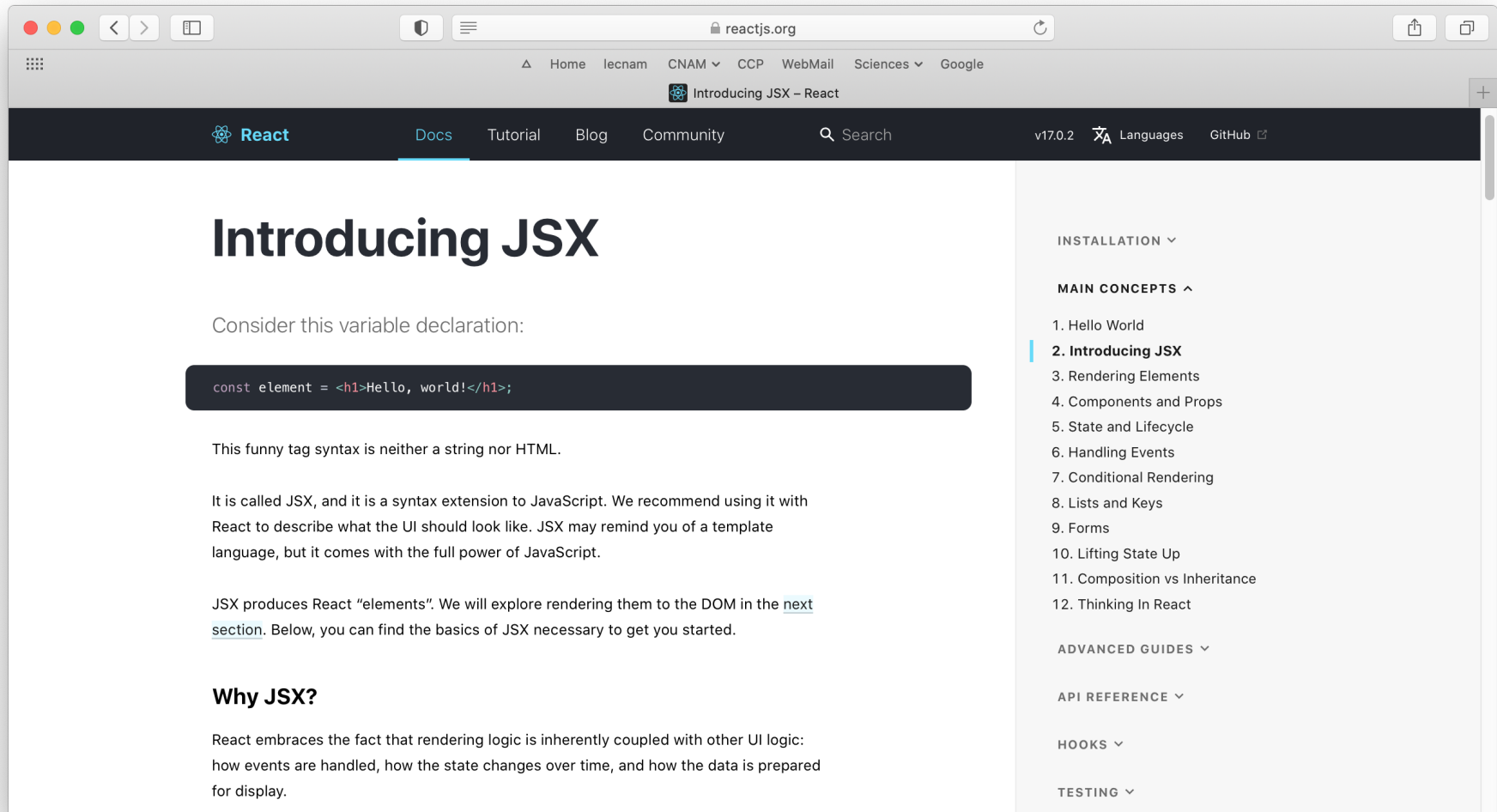
```
const initialState = {count: 0};

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return {count: state.count + 1};
    case 'decrement':
      return {count: state.count - 1};
    default:
      throw new Error();
  }
}

function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);
  return (
    <>
      Count: {state.count}
      <button onClick={() => dispatch({type: 'decrement'})}>-</button>
      <button onClick={() => dispatch({type: 'increment'})}>+</button>
    </>
  );
}
```

On the right side of the page, there is a sidebar with a table of contents. The items are: 'INSTALLATION', 'MAIN CONCEPTS', 'ADVANCED GUIDES', 'API REFERENCE', 'HOOKS', and a list of 8 articles. The 7th article, '7. Hooks API Reference', is highlighted with a blue bar on the left.

React JSX (View)



The screenshot shows a web browser window with the URL `reactjs.org`. The page title is "Introducing JSX - React". The navigation bar includes "React", "Docs", "Tutorial", "Blog", "Community", "Search", "v17.0.2", "Languages", and "GitHub". The main content area features the heading "Introducing JSX" and a code block containing `const element = <h1>Hello, world!</h1>;`. The text explains that JSX is a syntax extension to JavaScript used with React to describe UI. A sidebar on the right lists navigation options: "INSTALLATION", "MAIN CONCEPTS" (with "2. Introducing JSX" selected), "ADVANCED GUIDES", "API REFERENCE", "HOOKS", and "TESTING".

Introducing JSX

Consider this variable declaration:

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

This funny tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React "elements". We will explore rendering them to the DOM in the [next section](#). Below, you can find the basics of JSX necessary to get you started.

Why JSX?

React embraces the fact that rendering logic is inherently coupled with other UI logic: how events are handled, how the state changes over time, and how the data is prepared for display.

- INSTALLATION ▾
- MAIN CONCEPTS ▲
 1. Hello World
 2. Introducing JSX
 3. Rendering Elements
 4. Components and Props
 5. State and Lifecycle
 6. Handling Events
 7. Conditional Rendering
 8. Lists and Keys
 9. Forms
 10. Lifting State Up
 11. Composition vs Inheritance
 12. Thinking In React
- ADVANCED GUIDES ▾
- API REFERENCE ▾
- HOOKS ▾
- TESTING ▾