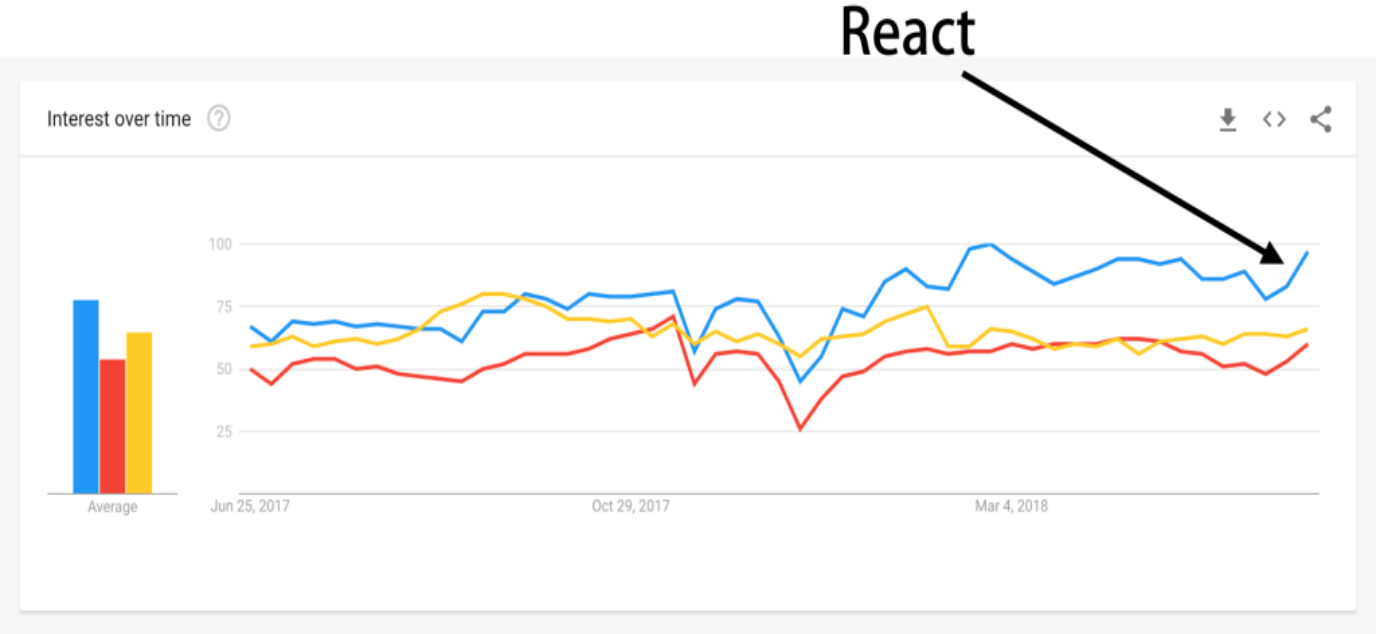
Title: What is react and why we should use it?

Answer:

What is React?

React is a JavaScript library for building fast and interactive user interfaces.

It was developed at Facebook in 2011 and currently, it's the most popular JS library for building user interfaces. Here's a comparison of the top 3 libraries/frameworks for building user-interfaces: React, Angular and Vue:



As you can see, it is dominating the space. So, if you want to expand your job opportunities as a front-end developer, you should have it on your resume.

Architecture

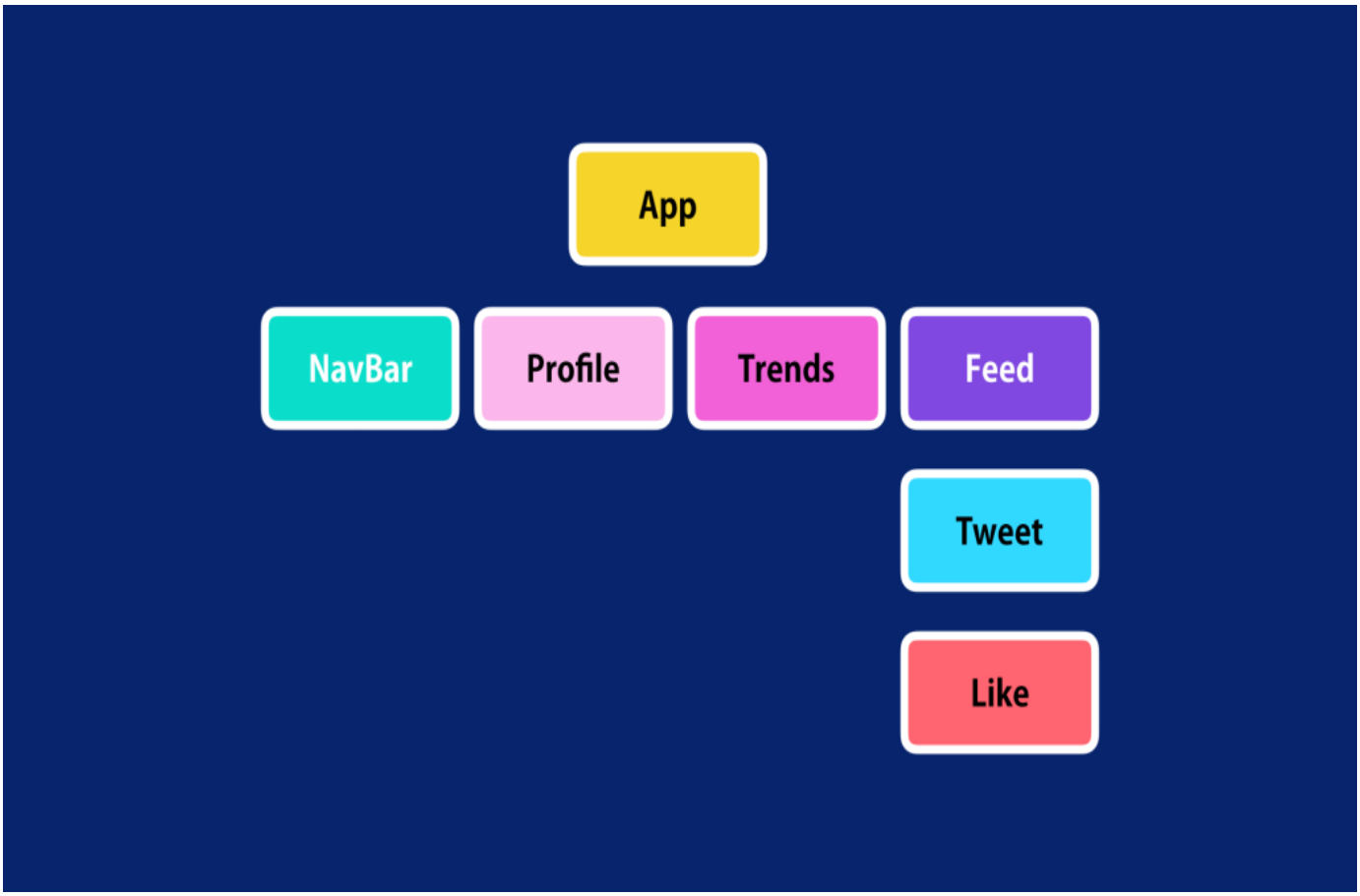
At the heart of all React applications are **components**.

A component is essentially a piece of the user interface. So, when building applications with React, we build a bunch of independent, isolated and reusable components and then compose them to build complex user interfaces.

Every React app has at least one component, which we refer to as the *root component.* This component represents the entire application and contains other child components. So, every React application is essentially a tree of components. If you've worked with Angular 2 or higher, this should sound familiar!

A Real Example

Here's an example. Let's imagine we want to build an application like Twitter (https://twitter.com/). We can split this page into components like NavBar, Profile, Trends, and Feed. Here's a representation of these components in a tree.



So, on the top, we have **App**. This component has 4 children: **NavBar**, **Profile**, **Trends**, and **Feed**, which includes several **Tweet** components. Each **Tweet** component can include a **Like**component, which we can re-use on other pages or even in different applications!

So, as you see, each component is a piece of UI. We can build these components in isolation and then put them together to build complex user interfaces.

Show Me the Code!

OK, I know you've been curious what the code looks like. A component is typically (but not always) implemented as a JavaScript class that has some state, and a render method.

```
class Tweet {
   state = {};
   render() {
      // Here we return a React element
   }
}
```

The **state** here is the data that we want to display when that component is rendered. And the **render** method, as you can tell, is responsible for describing what the UI should look like. If a component doesn't have a state, we can implement it using a pure function instead of a class. More on

this later.

Virtual DOM

The output of this **render** method is a React element which is a simple, plain JavaScript object that maps to a DOM element. It's not a real DOM element, it's just a plain JS object that represents that DOM element in memory.

So, React keeps a light-weight representation of the DOM in memory. We refer to this as the *virtual DOM*. Unlike the browser or the real DOM, this virtual DOM is cheap to create. When we change the state of a component, we get a new React element. React will then compare this element and its children with the previous one, it figures out what is changed, and then, it'll update a part of the real DOM to keep it in sync with the virtual DOM.

React vs Vanilla JavaScript/ jQuery

So, that means when building applications with React, unlike vanilla JavaScript or jQuery, we no longer have to work with the DOM API in browsers. In other words, we no longer have to write code to query and manipulate the DOM or attach event handlers to DOM elements.

```
const element = document.querySelector('#course');
element.classList.add('active');
element.addEventListener('click', ...);
```

We simply change the state of our components, and React will automatically update the DOM to match that state.

React vs Angular 2+

React and Angular are similar in terms of their component-based architecture. But Angular is a *framework* or a complete solution, while React is a *library*. It only takes care of rendering the view and making sure that the view is in sync with the state. That's all React does, nothing less, nothing more! For this very reason, it has a small API to learn.

So, when building applications with React, we need to use other libraries for things like routing or calling HTTP services and so on. But this is not necessarily a bad thing, because you get to choose the libraries that you prefer as opposed to being fixed with what Angular gives you, which often breaks from one version to another!

Tags: react