Comparison to React

Component Definitions

- Vue components
  - defined by a JavaScript literal object
  - event handling is implemented with methods defined in that object,
    and attached to component instances
  - no way to use plain functions
- React components
  - can be defined by a plain function, as can event handling
  - in the past React components usually were
    not defined by a plain function because
    a class was required to implement state and lifecycle methods
  - with the introduction of "hooks", that is no longer true
  - so most React code will use plain functions for nearly everything
  - removes need to understand value of `this` keyword and how classes work

Error Messages

- React
  - currently provides better error messages than Vue
  - when source files are saved from any editor or IDE
    React detects the changes, compiles them,
    and refreshes the browser tab where the app is running
  - any errors are displayed in the browser with very clear messages
    explaining exactly what needs to be fixed
- Vue
  - error messages appear in devtools console
  - they are often brief and contain a stack trace
    where most of the lines refer to code in the library
    rather than application-specific code
  - sometimes no stack trace lines refer to your code!

Performance

- React
  - uses virtual DOM diffing to determine actual DOM updates that are required
- Vue
  - Vue 2 changes all data objects and Vuex state objects to track property changes by modifying them at startup using the `defineProperty` method which adds getter and setter methods for all properties
  - Vue refers to this as making objects "reactive"
  - has a startup cost, but results in faster determination of required DOM changes than React
  - perhaps this is why live reload is slower in Vue than in React
  - Vue 3 changes from using generated getters and setters to monitor data changes to using Proxies
  - faster (both in initialization of components and runtime usage of them), uses less memory, and supports detecting changes from more JavaScript features
  - likely means the following are no longer needed: `$set`, `$delete`, and `Array $set`

CLI Startup

- React
  - in applications created with create-react-app,
    entering **`npm start`** starts a local HTTP server,
    opens a tab in default web browser,
    and loads the app
- Vue
  - in applications created with Vue CLI,
    entering **`npm run serve`** starts a local HTTP server,
    but it does not open a browser tab for the application
  - must be done manually

Ejecting

- React
  - in projects created using create-react-app,
    the kinds of configuration changes that can be made are limited
  - some kinds of changes required "ejecting"
  - removes dependency on **`react-scripts`** library and
    copies many configuration files and dependencies into project
  - after ejecting, it is no longer easy to
    take advantage of improvements to **`react-scripts`**
- Vue
  - projects created using Vue CLI
    never require ejecting to customize their build process
  - can always take advantage of improvements
    to **`@vue/cli-service`**

Ease of Learning

- Many claim a benefit of Vue is that it is easier to learn than React's JSX
- This comes down to comparing
  the ways in which JSX differs from HTML
  to the things that Vue adds to HTML
- Let's compare them point by point

Inserting Content

- React
  - the result of a JavaScript expression is inserted into the DOM by enclosing it in curly braces.
  - example: `{ new Date().toString() }`
- Vue
  - the same approach is used, but with double curly braces
  - example: `{{ new Date().toString() }}`
- Evaluation: tie

Attribute Names

- React
  - some HTML attributes must be specified with a different name because they are keywords in JavaScript
  - the most common are **`class`** which must be specified as className and **`for`** (often used on **`label`** elements) which must be specified as **`htmlFor`**
  - another difference is that event handling attributes must be camel-cased
    - for example, **`onclick`** must be specified as **`onClick`**
  - while these differences are an annoyance, when developers forget to use the alternate names, React gives very clear error messages
- Vue
  - event handling attributes are specified using the **`v-on`** directive or its shorthand form **`@`**
  - for example, instead of **`onclick`**, **`v-on:click`** or **`@click`** must be used
- Evaluation: small win for Vue

Directives

- Vue
  - defines many directives that can be used in HTML
  - some support conditional logic and iteration
  - examples include `v-if` and `v-for`
  - also supports defining custom directives

use of these should probably be discouraged since developers that are familiar with Vue will not be familiar with custom directives being used in an application that is new to them

- React
  - relies on knowledge of JavaScript and the DOM for implementing the same tasks
  - JavaScript in curly braces is used for conditional logic and iteration
  - iteration is primarily achieved using the `Array` `map` method
- Vue
  - for developers that already have strong knowledge of JavaScript and the DOM, Vue requires additional learning not required in React
- Evaluation: small win for React

Two-way Data Binding

- Vue
  - the **`v-model`** directive can be used to bind
    the value of a form element to a component data value
- React
  - implementing this functionality requires
    custom event handling methods and use of the **`setState`** method
- Evaluation: win for Vue