1. VueJs 相关文献

Vue.js (pronounced /vjuː/, like view) is a library for building interactive web interfaces. The goal of Vue.js is to provide the benefits of reactive data binding and composable view components with an API that is as simple as possible.

Vue.js是一个构建数据驱动的 web 界面的库。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

Vue.js itself is not a full-blown framework - it is focused on the view layer only. It is therefore very easy to pick up and to integrate with other libraries or existing projects. On the other hand, when used in combination with proper tooling and supporting libraries, Vue.js is also perfectly capable of powering sophisticated Single-Page Applications.

Vue.js 自身不是一个全能框架——它只聚焦于视图层。因此它非常容易学习，非常容易与其它库或已有项目整合。另一方面，在与相关工具和支持库一起使用时，Vue.js 也能完美地驱动复杂的单页应用。

Vue.js uses an HTML-based template syntax that allows you to declaratively bind the rendered DOM to the underlying Vue instance’s data. All Vue.js templates are valid HTML that can be parsed by spec-compliant browsers and HTML parsers.

Vue.js 使用了基于 HTML 的模版语法，允许开发者声明式地将 DOM 绑定至底层 Vue 实例的数据。所有 Vue.js 的模板都是合法的 HTML ，所以能被遵循规范的浏览器和 HTML 解析器解析。

Under the hood, Vue compiles the templates into Virtual DOM render functions. Combined with the reactivity system, Vue is able to intelligently figure out the minimal amount of components to re-render and apply the minimal amount of DOM manipulations when the app state changes.

在底层的实现上， Vue 将模板编译成虚拟 DOM 渲染函数。结合响应系统，在应用状态改变时， Vue 能够智能地计算出重新渲染组件的最小代价并应用到 DOM 操作上。

At the core of Vue.js is a reactive data-binding system that makes it extremely simple to keep your data and the DOM in sync. When using jQuery to manually manipulate the DOM, the code we write is often imperative, repetitive and error-prone. Vue.js embraces the concept of data-driven view. In plain words, it means we use special syntax in our normal HTML templates to “bind” the DOM to the underlying data. Once the bindings are created, the DOM will then be kept in sync with the data. Whenever you modify the data, the DOM updates accordingly. As a result, most of our application logic is now directly manipulating data, rather than messing around with DOM updates. This makes our code easier to write, easier to reason about and easier to maintain.

Vue.js 的核心是一个响应的数据绑定系统，它让数据与 DOM 保持同步非常简单。在使用 jQuery 手工操作 DOM 时，我们的代码常常是命令式的、重复的与易错的。Vue.js 拥抱数据驱动的视图概念。通俗地讲，它意味着我们在普通 HTML 模板中使用特殊的语法将 DOM “绑定”到底层数据。一旦创建了绑定，DOM 将与数据保持同步。每当修改了数据，DOM 便相应地更新。这样我们应用中的逻辑就几乎都是直接修改数据了，不必与 DOM 更新搅在一起。这让我们的代码更容易撰写、理解与维护。

Pass an ordinary JavaScript Object to the Vue instance data options, Vue will iterates through all attributes, the Object and use the Object. The defineProperty all these attributes to getter/setter.Object. The defineProperty is only ES5 support, and unable to shim features, that is why the Vue browser does not support IE8 and lower version.Users can't see the getter/setter, but inside they make Vue track dependencies, when property is accessed and modified change notice.Here need to be aware of the problem is the browser console in print format of data object getter/setter and different, so you may need to install the vue - devtools to get more friendly interface.Each component instance has the corresponding watcher object instance, it will be in the process of the component rendering attributes record for rely on, when the dependency of the setter is called, will inform watcher recount, which led to the update of its associated components.

把一个普通 JavaScript 对象传给 Vue 实例的 data 选项，Vue 将遍历此对象所有的属性，并使用 Object.defineProperty 把这些属性全部转为 getter/setter。Object.defineProperty 是仅 ES5 支持，且无法 shim 的特性，这也就是为什么 Vue 不支持 IE8 以及更低版本浏览器的原因。用户看不到 getter/setter，但是在内部它们让 Vue 追踪依赖，在属性被访问和修改时通知变化。这里需要注意的问题是浏览器控制台在打印数据对象时 getter/setter 的格式化并不同，所以你可能需要安装 vue-devtools 来获取更加友好的检查接口。每个组件实例都有相应的 watcher 实例对象，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的 setter 被调用时，会通知 watcher 重新计算，从而致使它关联的组件得以更新。

Vue asynchronous execution DOM updates.As long as the observed data changes, the Vue will usher in a queue, and the buffer change all the data in the same event loop.If the same watcher be triggered many times, it will only push to queue at a time.The removal of duplicate data in buffer to avoid unnecessary calculation and DOM manipulation is very important.Then, the next event loop "tick" Vue refresh queue and perform the actual (have to) work.Vue internally to try for asynchronous queue use native Promise. Then and MutationObserver, if you don't support the execution environment, will use setTimeout (fn, 0) instead.. For example, when you set the vm someData = 'new value, the component will not render it back into place immediately.When the refresh queue, the component will be when the event loop queue to empty the next update "tick".Most of the time we don't need to care about the process, but if you want to do something after DOM status updates, it may be tricky.Although the Vue. Js usually encourage developers along the "data driven" way of thinking, to avoid direct contact with the DOM, but sometimes we do have to do so.In order to complete update the DOM waiting Vue after data changes, can be used immediately after the data changes Vue. NextTick (the callback).This callback function after completion of the DOM to update is called.

Vue 异步执行 DOM 更新。只要观察到数据变化，Vue 将开启一个队列，并缓冲在同一事件循环中发生的所有数据改变。如果同一个 watcher 被多次触发，只会一次推入到队列中。这种在缓冲时去除重复数据对于避免不必要的计算和 DOM 操作上非常重要。然后，在下一个的事件循环“tick”中，Vue 刷新队列并执行实际（已去重的）工作。Vue 在内部尝试对异步队列使用原生的 Promise.then 和 MutationObserver，如果执行环境不支持，会采用 setTimeout(fn, 0) 代替。例如，当你设置 vm.someData = 'new value' ，该组件不会立即重新渲染。当刷新队列时，组件会在事件循环队列清空时的下一个“tick”更新。多数情况我们不需要关心这个过程，但是如果你想在 DOM 状态更新后做点什么，这就可能会有些棘手。虽然 Vue.js 通常鼓励开发人员沿着“数据驱动”的方式思考，避免直接接触 DOM，但是有时我们确实要这么做。为了在数据变化之后等待 Vue 完成更新 DOM ，可以在数据变化之后立即使用 Vue.nextTick(callback) 。这样回调函数在 DOM 更新完成后就会调用。

1. Vuex 相关文献

Vuex is a state management pattern + library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion. It also integrates with Vue's official devtools extension to provide advanced features such as zero-config time-travel debugging and state snapshot export / import.

Vuex 是一个专为 Vue.js 应用程序开发的状态管理模式。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。Vuex 也集成到 Vue 的官方调试工具 devtools extension，提供了诸如零配置的 time-travel 调试、状态快照导入导出等高级调试功能。

It is a self-contained app with the following parts:

1. The state, which is the source of truth that drives our app;
2. The view, which is just a declarative mapping of the state;
3. The actions, which are the possible ways the state could change in reaction to user inputs from the view.

这个状态自管理应用包含以下几个部分：

1. state，驱动应用的数据源；

2. view，以声明方式将state映射到视图；

3. actions，响应在view上的用户输入导致的状态变化。

the simplicity quickly breaks down when we have multiple components that share common state:

1. Multiple views may depend on the same piece of state.
2. Actions from different views may need to mutate the same piece of state.

当我们的应用遇到多个组件共享状态时，单向数据流的简洁性很容易被破坏：

1. 多个视图依赖于同一状态。

2. 来自不同视图的行为需要变更同一状态。

For problem one, passing props can be tedious for deeply nested components, and simply doesn't work for sibling components. For problem two, we often find ourselves resorting to solutions such as reaching for direct parent/child instance references or trying to mutate and synchronize multiple copies of the state via events. Both of these patterns are brittle and quickly lead to unmaintainable code.

对于问题一，传参的方法对于多层嵌套的组件将会非常繁琐，并且对于兄弟组件间的状态传递无能为力。对于问题二，我们经常会采用父子组件直接引用或者通过事件来变更和同步状态的多份拷贝。以上的这些模式非常脆弱，通常会导致无法维护的代码。

So why don't we extract the shared state out of the components, and manage it in a global singleton? With this, our component tree becomes a big "view", and any component can access the state or trigger actions, no matter where they are in the tree!

因此，我们为什么不把组件的共享状态抽取出来，以一个全局单例模式管理呢？在这种模式下，我们的组件树构成了一个巨大的“视图”，不管在树的哪个位置，任何组件都能获取状态或者触发行为！

This is the basic idea behind Vuex, inspired by Flux, Redux and The Elm Architecture. Unlike the other patterns, Vuex is also a library implementation tailored specifically for Vue.js to take advantage of its granular reactivity system for efficient updates.

这就是 Vuex 背后的基本思想，借鉴了 Flux、Redux、和 The Elm Architecture。与其他模式不同的是，Vuex 是专门为 Vue.js 设计的状态管理库，以利用 Vue.js 的细粒度数据响应机制来进行高效的状态更新。

1. Vue-Router相关文档

Creating a Single-page Application with Vue.js + vue-router is dead simple. With Vue.js, we are already composing our application with components. When adding vue-router to the mix, all we need to do is map our components to the routes and let vue-router know where to render them.

用 Vue.js + vue-router 创建单页应用，是非常简单的。使用 Vue.js ，我们已经可以通过组合组件来组成应用程序，当你要把 vue-router 添加进来，我们需要做的是，将组件(components)映射到路由(routes)，然后告诉 vue-router 在哪里渲染它们。

1. Webpack相关文献

webpack is a module bundler for modern JavaScript applications. It is incredibly configurable, but to get started you only need to understand Four Core Concepts: entry, output, loaders, and plugins.

webpack 是一个现代的 JavaScript 应用程序的模块打包器(module bundler)。它是高度可配置的，但是，在开始前你需要先理解四个核心概念：入口(entry)、输出(output)、loader、插件(plugins)。

Webpack is a popular module bundler, a tool for bundling application source code in convenient chunks and for loading that code from a server into a browser.

Webpack是一个流行的模块绑定器，用于将应用程序源代码捆绑在方便的块中 并将该代码从服务器加载到浏览器中的工具。

Webpack is a powerful module bundler. A bundle is a JavaScript file that incorporates assets that belong together and should be served to the client in a response to a single file request. A bundle can include JavaScript, CSS styles, HTML, and almost any other kind of file.

Webpack打包机是一种强大的模块。一捆是一个JavaScript文件,包含资产属于彼此,应该在响应客户端请求一个文件。包可以包含JavaScript、CSS样式的HTML,几乎任何其他类型的文件。

Webpack roams over your application source code, looking for import statements, building a dependency graph, and emitting one or more bundles. With plugins and rules, Webpack can preprocess and minify different non-JavaScript files such as TypeScript, SASS, and LESS files. You determine what Webpack does and how it does it with a JavaScript configuration file, webpack.config.js.

Webpack出没在你的应用程序源代码,寻找导入语句,构建一个依赖图,和释放一个或多个包。通过插件和规则,Webpack可以进行预处理和贬低打印稿等不同的javascript文件,SASS,和更少的文件。您可以确定Webpack的功能以及JavaScript配置文件的运行方webpack.config.js。

1. ES6相关文献

ECMAScript (hereinafter referred to as ES6 6.0) is the next generation of the JavaScript language standard, has been officially released in June 2015.Its goal is to make the JavaScript language can be used to write complex large application, become enterprise development language.

ECMAScript 6.0（简称 ES6）是 JavaScript 语言的下一代标准，已经在2015年6月正式发布了。它的目标，是使得 JavaScript 语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

In November 1996, the creator of the JavaScript Netscape, decided to ECMA JavaScript is submitted to the international organization for standardization, hope the language will become an international standard.The following year, ECMA standard no. 262 document issued (the first edition of ECMA - 262), rules on the browser scripting language standard, and will be the language called ECMAScript, this version is version 1.0.

1996年11月，JavaScript 的创造者 Netscape 公司，决定将 JavaScript 提交给国际标准化组织ECMA，希望这种语言能够成为国际标准。次年，ECMA 发布262号标准文件（ECMA-262）的第一版，规定了浏览器脚本语言的标准，并将这种语言称为 ECMAScript，这个版本就是1.0版。

This standard is formulated for the JavaScript language from the beginning, but don't call JavaScript, for two reasons.A trademark, Java is Sun's trademark, according to the authorization agreement, only the name Netscape can legally use JavaScript, and JavaScript itself has also been Netscape is registered as a trademark.The second is to reflect the makers of this language is ECMA, not Netscape, so that to ensure the openness and neutrality of the language.

该标准从一开始就是针对 JavaScript 语言制定的，但是之所以不叫 JavaScript，有两个原因。一是商标，Java 是 Sun 公司的商标，根据授权协议，只有 Netscape 公司可以合法地使用 JavaScript 这个名字，且 JavaScript 本身也已经被 Netscape 公司注册为商标。二是想体现这门语言的制定者是 ECMA，不是 Netscape，这样有利于保证这门语言的开放性和中立性。

ECMAScript relationship with JavaScript is, therefore, the former is the latter's specifications, the latter is an implementation of the former (another ECMAScript dialect and Jscript and ActionScript).Everyday situations, these two words are interchangeable.

因此，ECMAScript 和 JavaScript 的关系是，前者是后者的规格，后者是前者的一种实现（另外的 ECMAScript 方言还有 Jscript 和 ActionScript）。日常场合，这两个词是可以互换的。