OSC源创会

年终盛典 2016

Vue.js 2.0 前端架构及后端渲染实践

阴明

稀土掘金创始人

www.juejin.im

Vue.js

- Vue 简介
- Vue 在掘金中的实践
- Vue 2.0 主要变化
- Vue 2.0 Render Function & 后端渲染

OSC源创会

年终盛典 2016

阴明

Bc.CUHK

MPhil.Cambridge

xitu.io / juejin.im

前端开发

OSC源创会

年终盛典 2016

Vue.js 简介

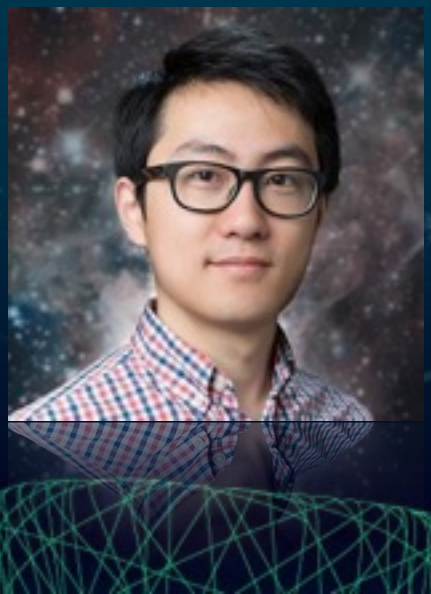
Why

What

How

Vue.js/about

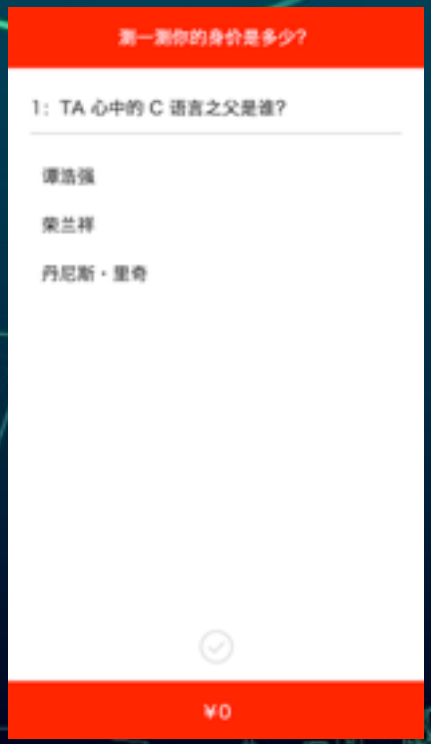
- The Progressive JavaScript Framework
- 由尤雨溪开发并开源
- GitHub 上有超过［3万5千］个 ⭐
- NPM 上每月会有超过［22万］次 ⏬



OSC源创会

年终盛典 2016

我用 Vue.js 做的第一个东西



coder-price

http://xitu.github.io/coder-price/

Vue.js 主要是做啥的?

1.0 它是一个前端 MVVM 框架

Model / View / ModelView

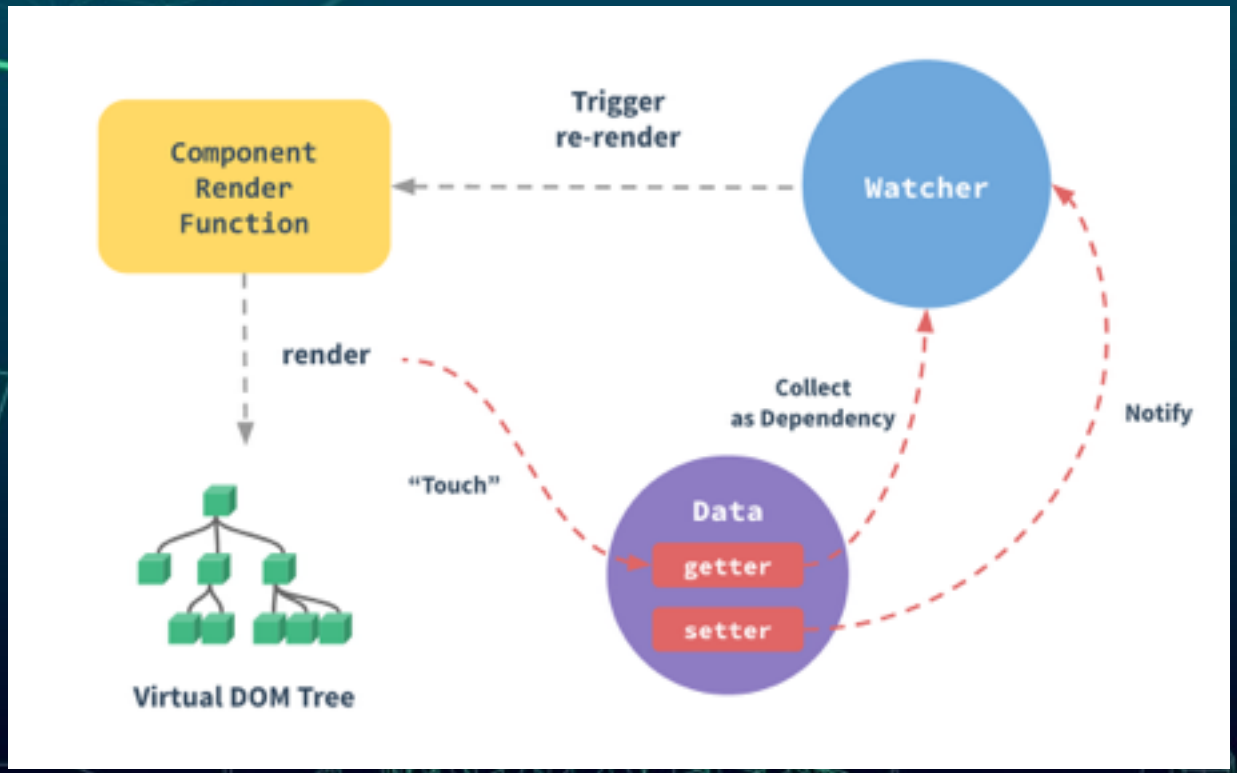
```
UI = VM(State)
```

Vue.js 主要是做啥的?

2.0 一个渐进式前端解决方案

OSC源创会
年终盛典 2016

Vue.js 原理：Object.defineProperty



OSC源创会
年终盛典 2016

Vue.js 的最简单的一个 Demo

```html
<div id="demo">
  <p>{{message ? message : "no content"}}</p>
  <input v-model="message">
</div>
```

```
var demo = new Vue({
  el: '#demo',
  data: {
    message: '喜欢前端技术，竟然不上掘金？'
  }
})
```

Vue.js 在掘金的实践

掘金从 0.12.5 开始

1.x

2.x

...

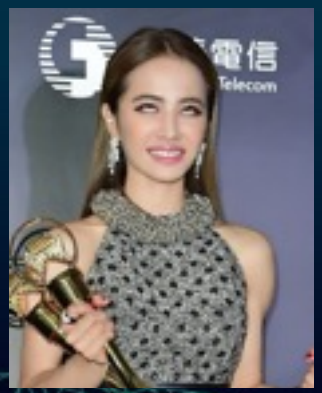
OSC源创会

年终盛典 2016

在现在的前端开发时代

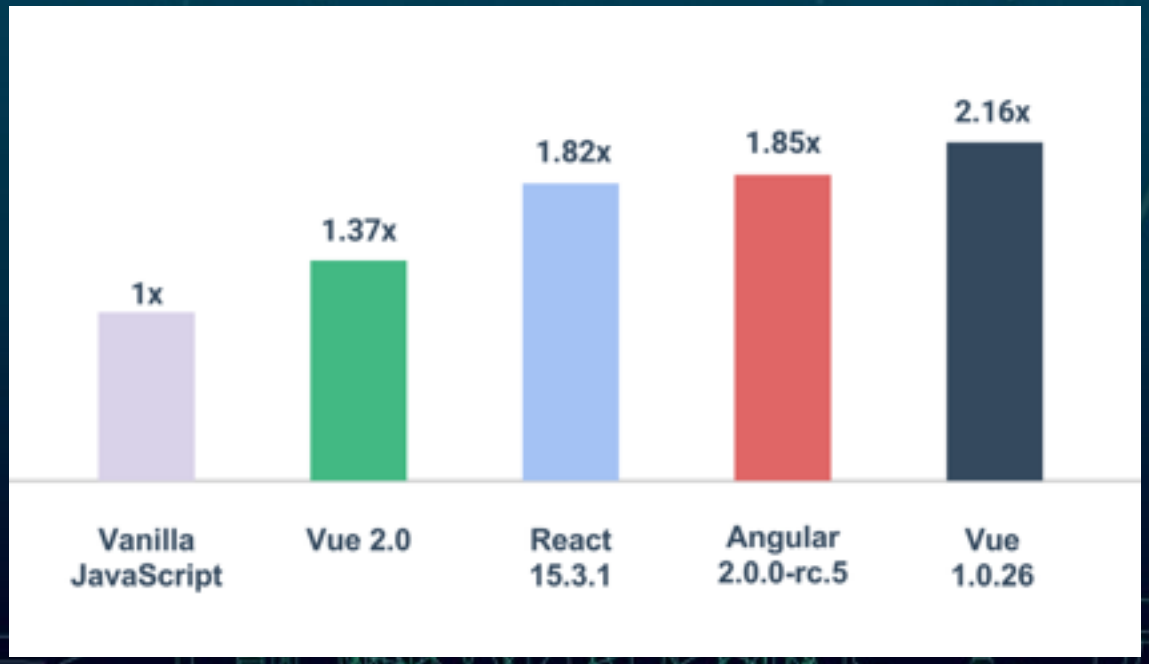
选择一个框架就像是选择一个老婆

错了是一辈子的事情!



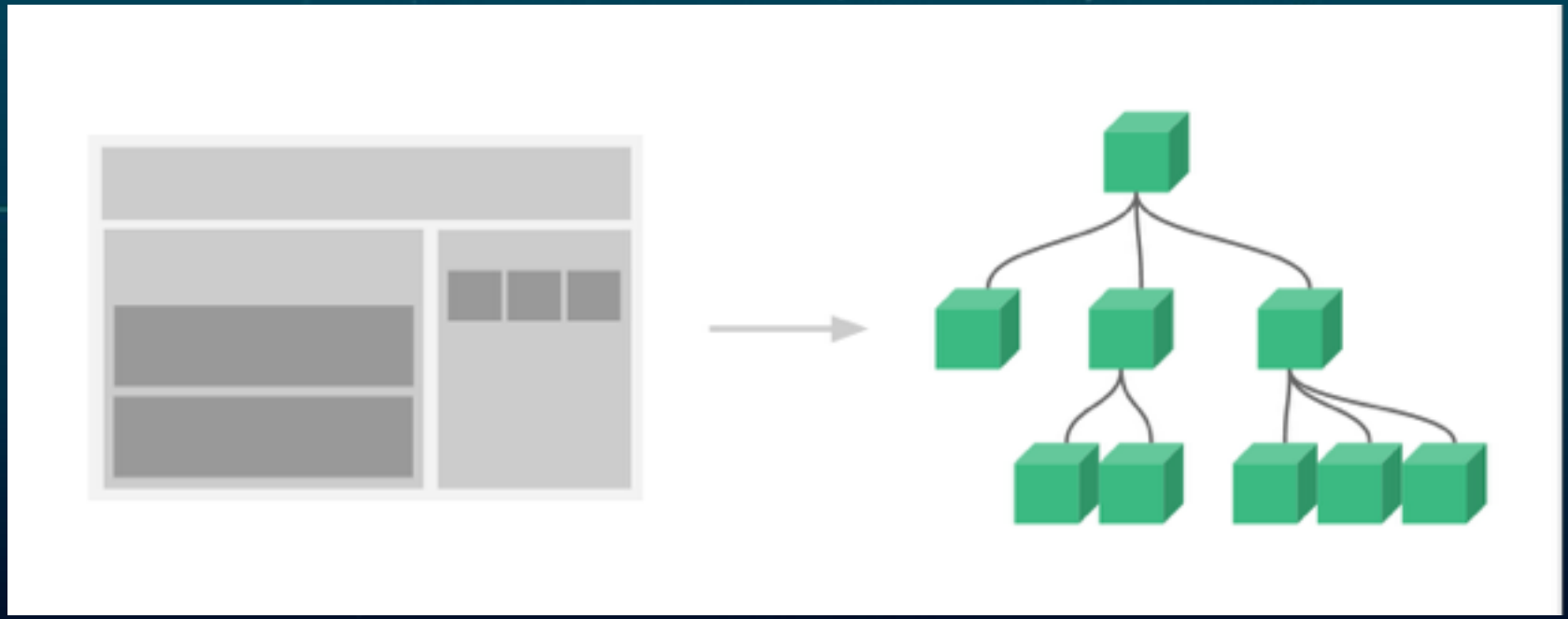
选择一个框架

- 开发效率：现在 1 个人顶过去 3 个人
- 代码维护：业务进展 > 扩展成本
- 速度性能：要能满足需求



Vue.js 的组件化编程

- `Vue.component`: vue-loader / vueify
- Single File Component



Single File Component

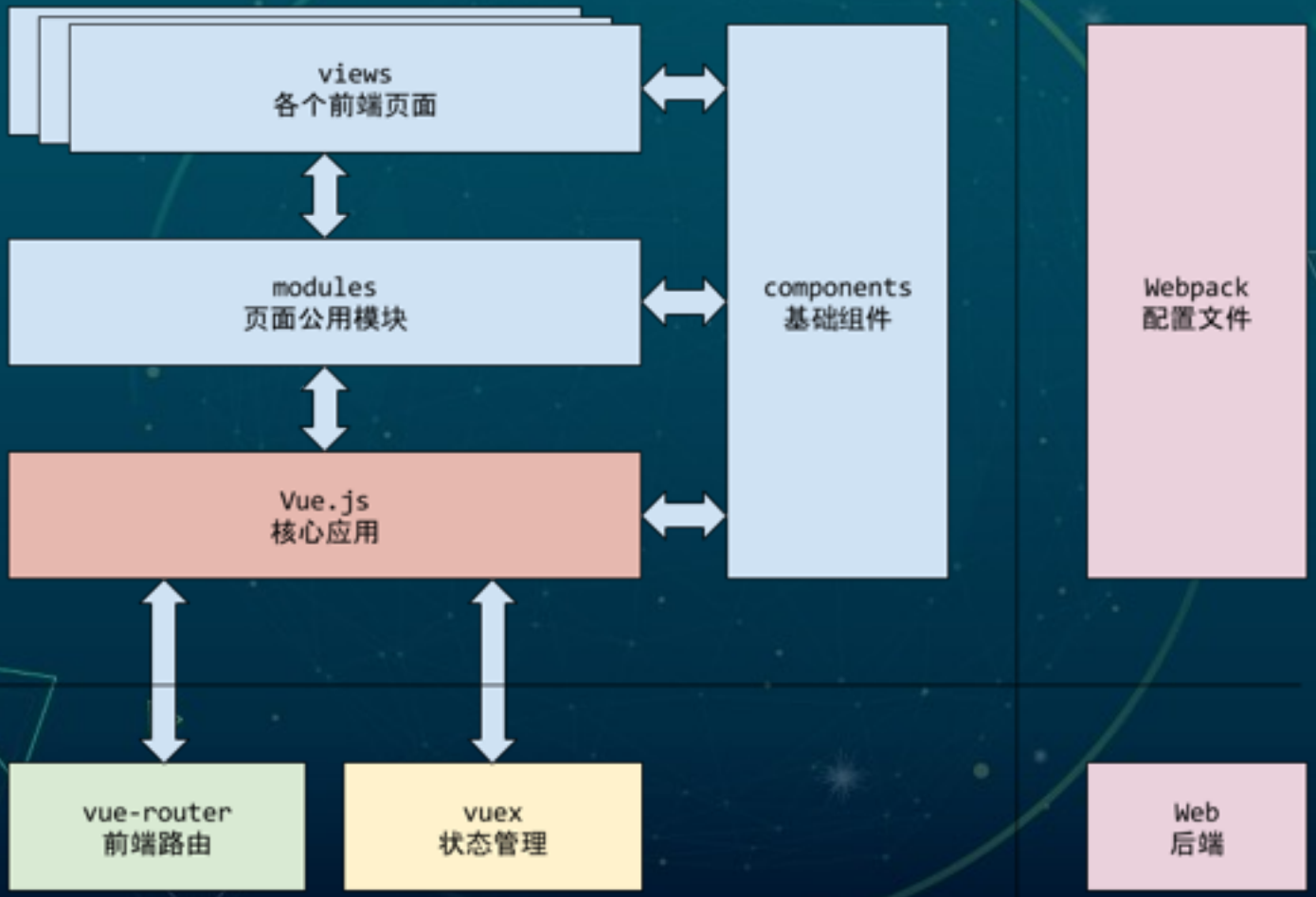
- 一个组件独立为一个 `.vue` 文件
- 内容：`<template>`
- 样式：`<style>`
- 业务：`<script>`

- 通过 JavaScript 原生的 `import` / `export` 来引用
- webpack 等工具自动打包

Vue.js 实现复杂应用的工具

- 脚手架：vue-cli
- 组件化：vue-loader
- 路由：vue-router
- 状态管理：vuex
- Ajax：vue-resource / vue-axios
- 前端开发工具：vue-devtools
- 前端组件库：vux / vue-mdl / mint-ui / element

Vue.js 掘金架构



OSC源创会
年终盛典 2016

Vue.js 2.0 新版主要变化

语法精简

Virtual DOM / Render Function

后端渲染

语法升级

- 循环
  - v-for="item in items", :key
- 绑定事件
  - v-on:click.prevent / @click.prevent
- 数据绑定、传递
  - v-bind:user / :user
- 动画
  - <transition> <transition-group>

语法精简

- 表格元素

```
<input
  type=“checkbox" v-model=“toggle"
  :true-value=“a" :false-value=“b”
>
<input v-model.lazy="msg" >
<input v-model.number="age" type="number">
<input v-model.trim="msg">
```

- Class
  - `<div :class=“[activeClass, errorClass]">`

Virtual DOM

- Vue.js 2.0 在 DOM 渲染上完全适用 Virtual DOM

```
<ul id='myId'>
  <li>Item 1</li>
  <li>Item 2</li>
<ul>
```

```
// Pseudo-code of a DOM node
represented as Javascript
Let domNode = {
  tag: 'ul'
  attributes: { id: 'myId' }
  children: [
    // where the LI's would go
  ]
};
```

```
domNode.children.push('<ul>Item 3</ul>');
sync(originalDomNode, domNode);
```

Render Function

- 用 Render Function 来生成 DOM

```
new Vue({
  el: '#app',
  data: {
   message: 'hello world'
  },
  render() {
    var node = this.$createElement;
    return node(
      'div',
      { attrs: { id: 'myId' } },
      this.message
    );
  }
});
```

```
<div id='app'>
  <div id='myId'>
   hello world
   </div>
</div>
```

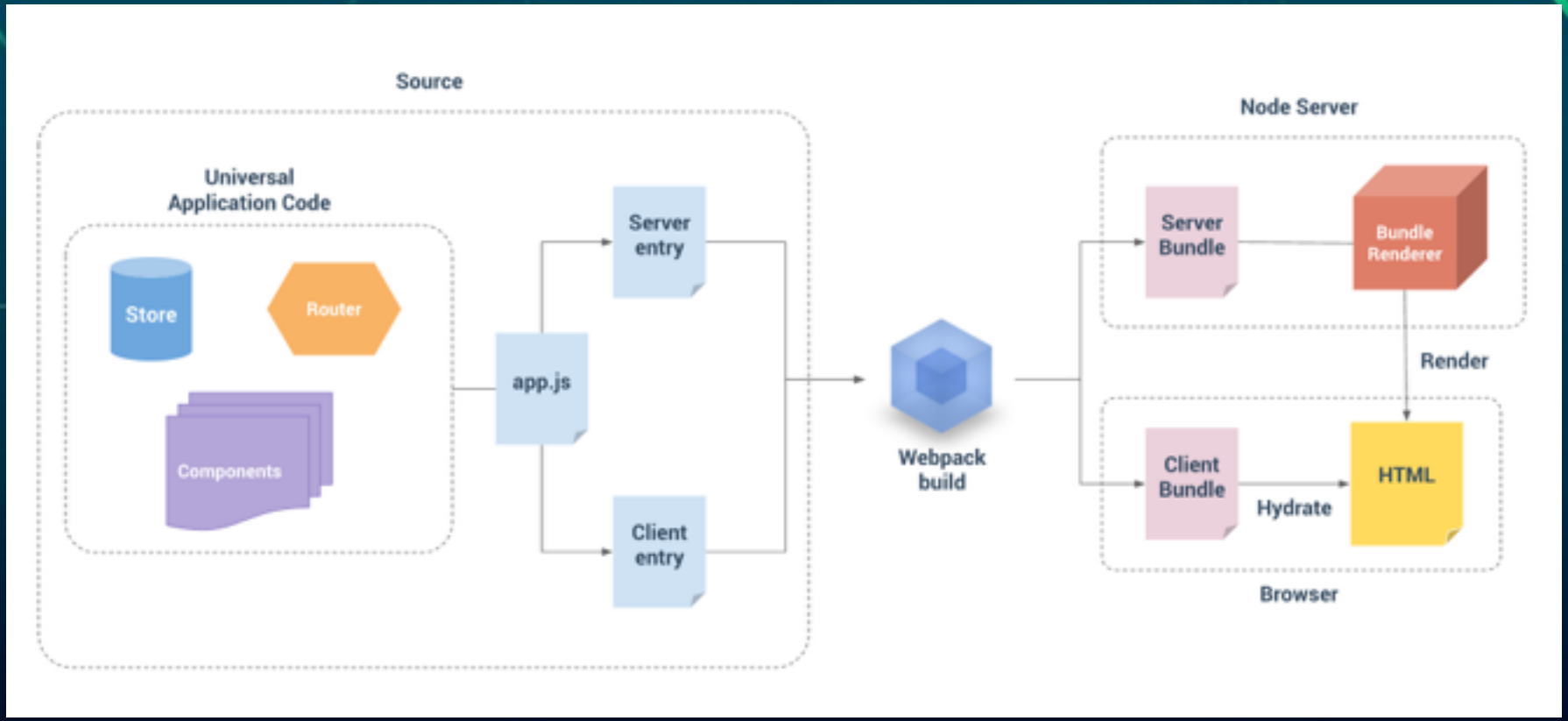
Render Function

- createElement
- jsx

```
// @returns {VNode}
createElement(
  // {String | Object | Function}
  // An HTML tag name, component options, or function
  // returning one of these. Required.
  'div',
  // {Object}
  // A data object corresponding to the attributes
  // you would use in a template. Optional.
  {
    // (see details in the next section below)
  },
  // {String | Array}
  // Children VNodes. Optional.
  [
    createElement('h1', 'hello world'),
    createElement(MyComponent, {
      props: {
        someProp: 'foo'
      }
    }),
    'bar'
  ]
)
```

服务器端渲染

- Render Function 可以在后端 Node 环境运行



服务器端渲染

- Example
- nuxt.js

```
// Step 1: Create a Vue instance
var Vue = require('vue')
var app = new Vue({
  render: function (h) {
    return h('p', 'hello world')
  }
})

// Step 2: Create a renderer
var renderer = require('vue-server-renderer')
               .createRenderer()

// Step 3: Render the Vue instance to HTML
renderer.renderToString(app, function (error, html) {
  if (error) throw error
  console.log(html)
  // => <p server-rendered="true">hello world</p>
})
```

服务器端渲染

- 只要在 Vue 业务包在一个 function call 中并连接上 `window context`
- 服务器 `renderer` 拿到相关业务 js 文件吐出内容
- Vue.js 2.0 支持 Stream 后端流式数据
  - 在 HTML 完整生成之前向前端吐数据
  - `renderer.renderToStream( … )`


掘金
阴明
Thx with 💕💕💕💕💕💕💕💕💕💕💕