

Лекция №10: Vue



Web-программирование / ПГНИУ

Vue

- Open-Source JavaScript Framework
- Progressive JavaScript Framework
- Фреймворк для разработки интерфейса веб-приложений на принципах реактивности и одностраничных приложений

История

- Разработан **Even You** в **2013** году
- Эван Ю был сотрудником Google и работал над **Angular.js**
- Основными фреймворками были большие **Angular.js** и **Blackbone.js**
- **React** вышел в 2013 году и был на ранней стадии
- Не было простого фремворка для быстрой разработки, прототипирования, миграций
- В **2015** вышла версия **1.0**, а в **2016** версия **2.0**, основная на сегодня
- В **2020** начал выходить **Vue 3**

Экосистема Vue

- Vue - не фреймворк в привычном смысле
- Vue масштабируется от **небольшой подключаемой к странице библиотеки** до **фреймворка** для создания SPA приложений
 - `vue-router` - роутинг в SPA
 - `vuex` - центральное Flux хранилище
 - `Vue/CLI` - прототипирование, сборка и разработка
 - `vue-dev-tools` - инструменты разработки в браузере
 - `vue-ssr` - серверный рендеринг
 - `vue-test-utils` - тестирование

Vue.js

- Реализует MVVM в компонентном подходе
- Компонент имеет данные (состояние)
- Компонент имеет шаблон
- Vue рендерит компонент по шаблону
- При изменении данные компонент автоматически ререндерится
- Эффективный ререндеринг на основе **VirtualDOM**

```
// Компонент описывается его опциями
const App = {
  // Шаблон компонента
  template: `<ul>
    <li v-for="todo in doneTodos">{{ todo.title }}</li>
  </ul>`,

  // Данные состояния приложения (реактивные)
  data: () => ({
    todos: [{ title: 'Task 1', done: true }],
  }),

  // Вычисляемые свойства, значение которых вычисляется на основе других свойств
  computed: {
    doneTodos() {
      return this.todos.filter(todo => todo.done);
    }
  },

  // Различные методы: обработчики событий и другие функции
  methods: {
    handleClick() {},
    loadTodos() {},
  },
};

// Создаём приложение на корневом компоненте и монтируем на страницу в #app
new Vue(App).$mount('#app');
```

Реактивность

- Работает за счёт переопределения свойств объектов с геттерами и сеттерами, а также патчинга прототипа массива
- Геттеры помогают определять зависимости в приложении
- Сеттеры позволяют реагировать на изменение значений

```
this.todos.push(newTodo);  
this.todos[0].title = 'New Title';  
this.todos = newTodosList;
```

Шаблон

- Описывает узлы компонента в привязке к его данным
- Вывод выражений в содержимом узлов через `{{ expression }}`
- Определение поведения узлов через директивы `v-*`
- Компилируется в render функцию с помощью `vue-template-compiler` на этапе сборки или в браузере

Выражения

```
<!-- Вывод содержимого узлов -->
```

```
<p>{{ propertyFromData }}</p>
```

```
<p>{{ todos }}</p>
```

```
<p>{{ todos[0].title + '!' }} - {{ todos[1].title + '!' }}</p>
```

```
<p>{{ new Date(todos[0].date).toLocaleDateString() }}</p>
```

Директивы ветвления

```
<div v-if="x === 1">X is 1</div>  
<div v-else-if="x === 2">X is 2</div>  
<div v-else>X is not 1 or 2</div>
```

Директива цикла

```
<p v-for="item in list">
  <b>{{ item }}</b>
</p>

<p v-for="(item, index) in list">
  <b>{{ index }}:</b> {{ item }}
</p>

<p v-for="(value, key) in object">
  <b>{{ key }}:</b> {{ value }}
</p>
```

Привязка значений

<!-- Значение атрибута привязывается к значению выражения -->

```
<a v-bind:href="todo.link">{{ todo.title }}</a>
```

<!-- Короткая форма -->

```
<a :href="todo.link">{{ todo.title }}</a>
```

<!-- JS выражение, а не просто свойство -->

```
<a :href="'/todos/' + todo.id">{{ todo.title }}</a>
```

<!-- С классами и стилями есть особые удобные формы -->

```
<div :class="['class1', todo.class]">{{ todo.title }}</div>
```

```
<div :class="{ 'todo__done': todo.done }">{{ todo.title }}</div>
```

```
<div :style="{ color: todo.color }">{{ todo.title }}</div>
```

Обработка событий

```
<!-- handler - методы компонента -->  
<button v-on:click="handler">Click Me!</button>  
  
<!-- Короткая форма -->  
<button @click="handler">Click Me!</button>  
  
<!-- Обработчик - не только метод, но и JS выражение -->  
<!-- $event - полезная нагрузка события -->  
<button @click="deleteTodo(todo.id)">Delete Todo</button>  
<input @change="todo.text = $event.target.value">Delete Todo</input>  
  
<!-- Модификаторы события -->  
<form @submit.prevent="submitForm">...</input>
```

Двусторонняя привязка - модель

```
<!-- Значение поля ввода определяется свойством text -->  
<!-- При вводе оно обновляется -->  
<input :value="text" @input="text = $event.target.value" />  
  
<!-- Коротка форма - директива модели -->  
<input v-model="text" />
```

Формы

```
<!-- Boolean -->
<input type="checkbox" v-model="todo.done"> Done?

<!-- Array -->
<input type="checkbox" v-model="selectedFruits" value="apple"> Apple
<input type="checkbox" v-model="selectedFruits" value="banana"> Banana

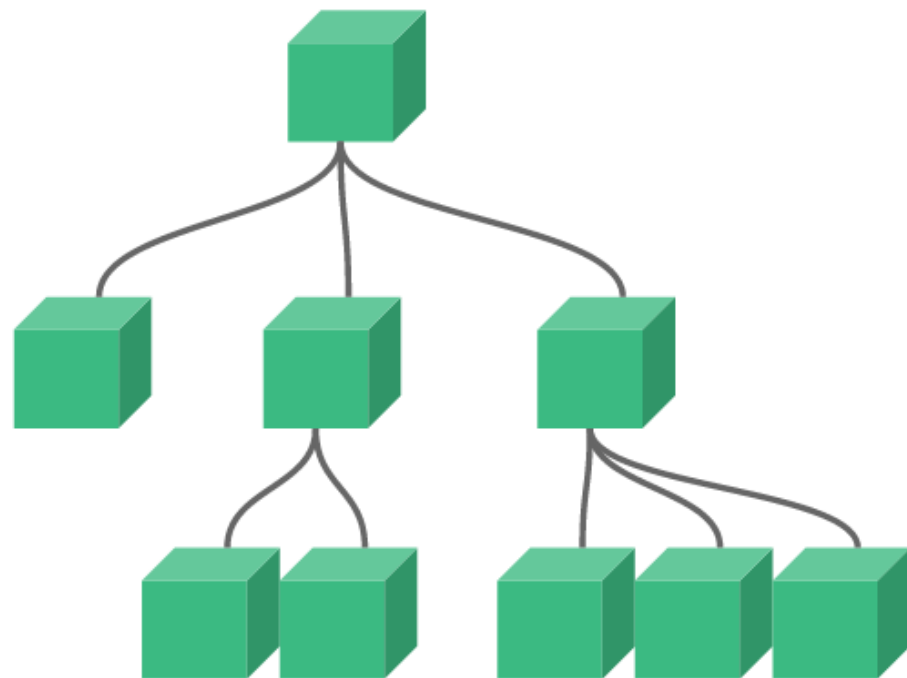
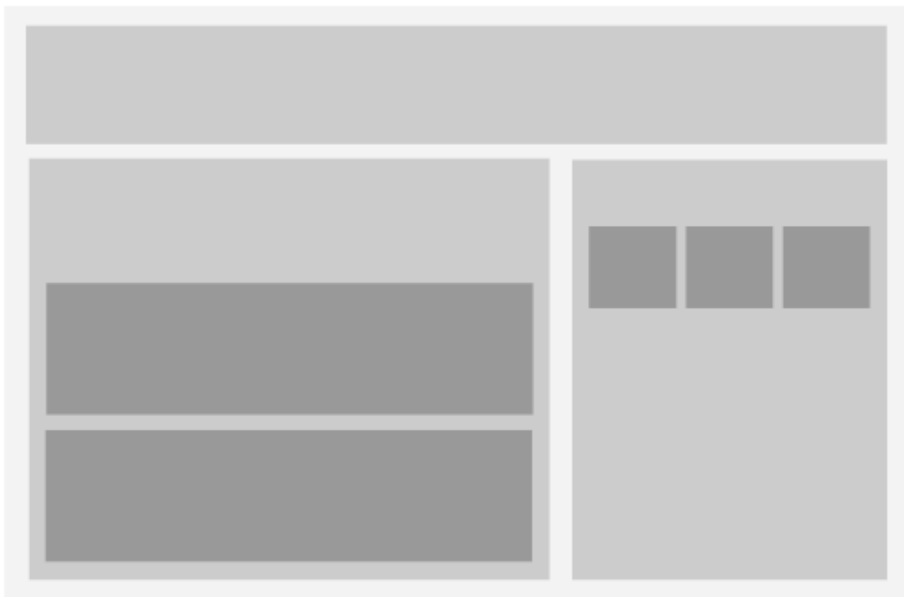
<input type="radio" v-model="selectedFruit" value="apple"> Apple
<input type="radio" v-model="selectedFruit" value="banana"> Banana

<textarea v-model="text"></textarea>

<select v-model="selectedFruit">
  <option value="apple">Apple</option>
  <option value="banana">Banana</option>
</select>
```

Компоненты

- В компонентном подходе приложение представляется как иерархия компонентов
- После регистрации компонент в шаблоне используется, как новый элемент



```
const PageTitle = {
  template: '<h1>Title!</h1>',
};

const App = {
  template: `<div>
    <main-logo />
    <the-title />
  </div>`,

  // Локальная регистрация
  components: {
    // Имя компонента : Реализация компонента
    TheTitle: PageTitle,
  },
};

// Глобальная регистрация
Vue.component('main-logo', {
  template: '',
});
```

Взаимодействие компонентов

- Компонент имеет входные параметры
- В компонент можно передавать содержимое
- Компонент может порождать события (Event Emitter)
- Родитель может подписываться на эти события
- **One-way dataflow** - данные передаются от дочернему компоненту; дочерний компонент не меняет эти, а только сообщает о такой необходимости

```
const TodoItem = {
  template: `<div class="todo" :class="{ 'todo__done': done }">
    {{ title }}
    <button @click="handleDoneClick">Done</button>
  </div>`,

  // Описываем входные параметры
  props: {
    done: {
      type: Boolean,
      default: false,
    },
    title: {
      type: String,
      required: true,
    },
  },

  methods: {
    handleDoneClick() {
      // Сообщаем родителю о том, что тудушке поменяли статус выполнения
      this.$emit('toggle-done', !this.done);
    },
  },
};
```

```
const app = {  
  template: `<<todo-item  
              :title="todo.item"  
              :done="todo.done"  
              @toggle-done="todo.done = $event"  
            />`,  
  
  components: { TodoItem },  
  
  data: () => ({  
    todo: { title: 'title', done: false }  
  }),  
};
```

Слоты

```
// В карточку можно передать содержимое
const SimpleCard = {
  template: `

<slot></slot>
  </div>`,
};

// Содержимое - любая разметка (шаблон)
const App = {
  template: `
    <h3>Title</h3>
    <p>Text</p>
  </simple-card>`,

  components: { SimpleCard },
};


```

```
// Может быть несколько слотов с разными именами
const ComplexCard = {
  template: `

Лекция №10: Vue / Курс Web-программирования 2020 / ПГНИУ



23 / 38


```

SPA: vue-router

- `vue-router` позволяет легко реализовать SPA на Vue
- Создаётся конфиг роутера, где определяются все маршруты и компоненты, которые за них отвечают
- В компоненте `<router-view>` выводится компонент текущего маршрута
- Для программного перехода между маршрутами вместо ссылок используются компоненты
`<router-link to="/todos">Todos</router-link>`
- К роутеру можно обращаться программно через `this.$router` или прямым импортом, а к текущему маршруту через `this.$route`

vue-router: другие возможности

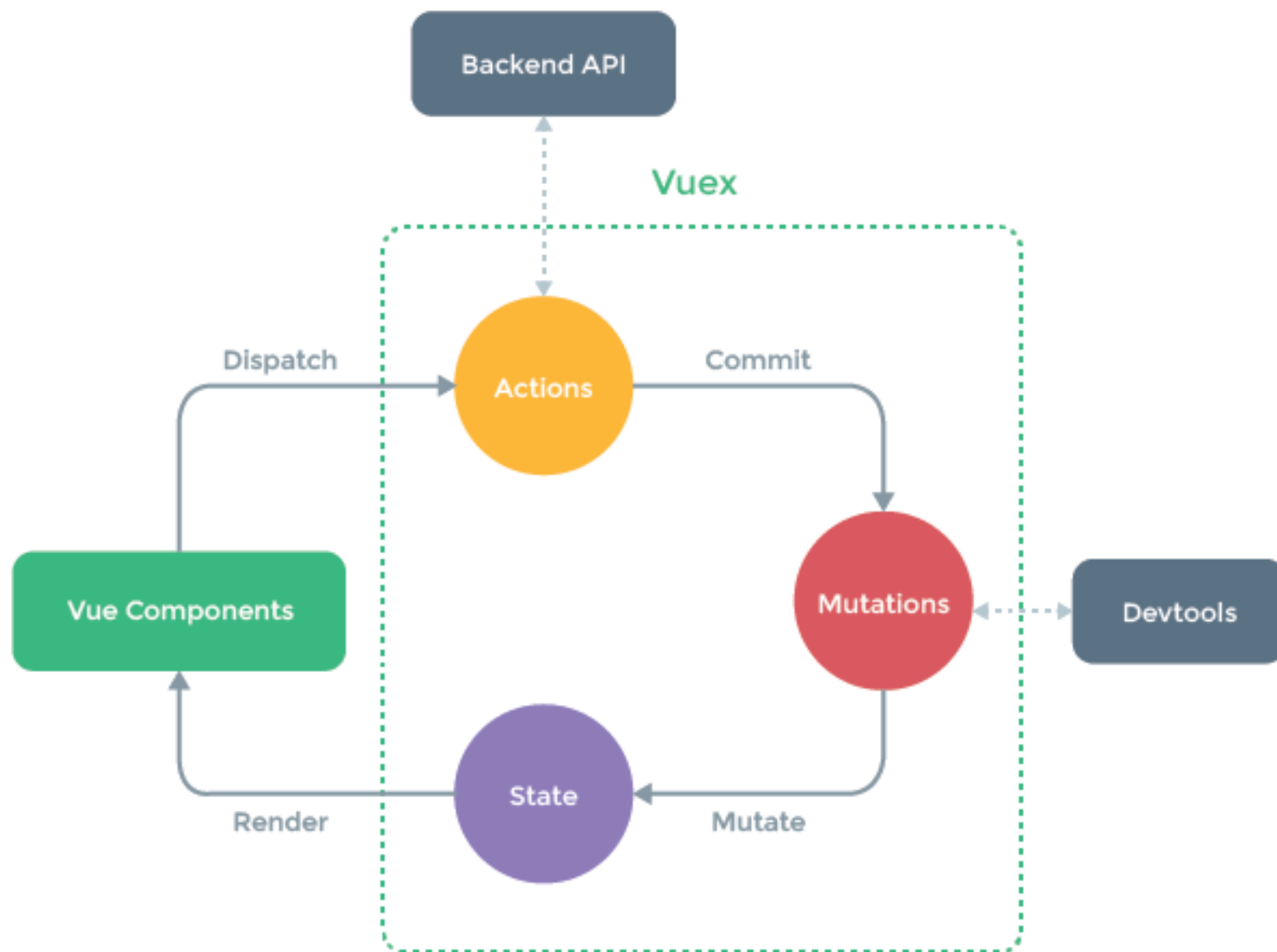
- Работает как с `hash`, так и на основе `HTML5 History API`
- Маршруты могут быть вложенными в иерархию
- У маршрута может быть несколько компонентов
- У маршрутов могут быть `guard`-ы, определяющие, можно ли переходить, и выполняющие подготовительные действия до перехода

```
const App = {  
  template: `<div>  
    <nav>  
      <router-link to="/todos">Todos List</router-link>  
      <router-link to="/todos/1">First Todo</router-link>  
    </nav>  
    <router-view />  
  </div>`,  
};
```

```
const router = new VueRouter({  
  mode: history,  
  routes: [  
    {  
      path: '/todos',  
      component: TodosPageComponent,  
    },  
    {  
      path: '/todos/:id',  
      component: TodoPageComponent,  
    },  
  ],  
});
```

Vueх

- Центральное хранилище - источник истинности состояния приложения
- Реализует Flux архитектуру
- Глобальное состояние реактивное, как и данные компонентов Vue
- Глобальное состояние меняется через диспетчер путём применения синхронных мутаций
- Хранилище описывается состоянием, мутациями, геттерами и действиями (асинхронными функциями для сложных действий)
- Хранилище разбивается на модули



vue-dev-tools

- Расширение браузера для разработки на Vue
- Дерево компонентов и их параметры
- События компонентов
- Хранилище и история его изменения
- Производительность



Ready. Detected Vue 2.6.7.



Filter components



<ConnectionStatus>

Filter inspected data



<Root>

> <StatusBar>

<ClientAddonLoader>

<LocaleLoader>

> <ConnectionStatus> = \$vm0

> <ProjectHome> = \$vm2 router-view

> <ViewNav> = \$vm1

> <ProjectQuickDropdown>

> <VueGroup>

> <ViewNavButton key='vue-project-dependencies'

> <ViewNavButton key='vue-project-tasks'>

<ResizeObserver>

> <ViewNavMore>

<ProgressScreen>

> <TopBar>

data

> \$apolloData: Object

> data: Object

loading: 0

> queries: Object

> \$route

> \$sharedData: Object

connected: true

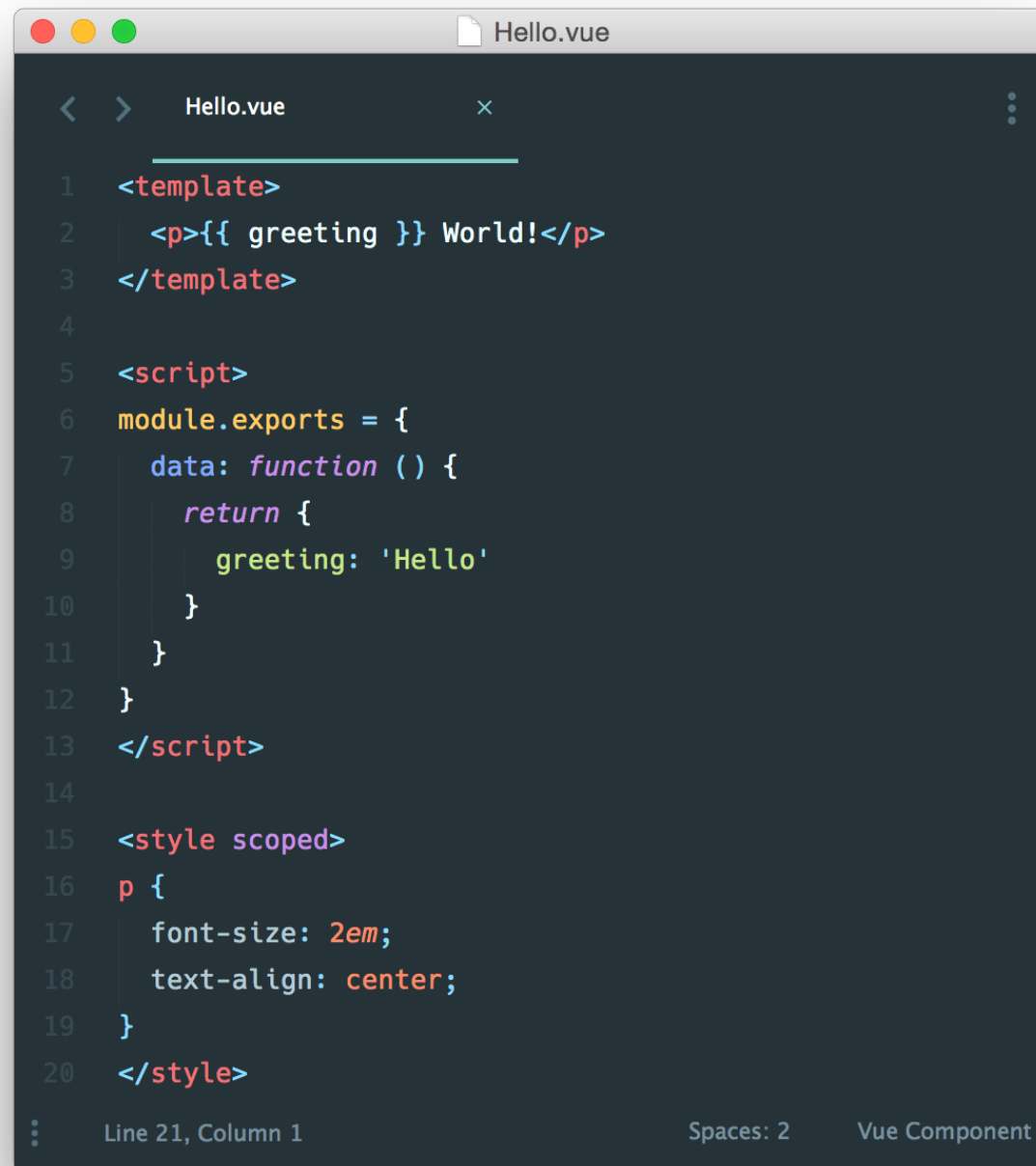
darkMode: false

Quick edit



SFC

- Single-File-Component - Однофайловые компоненты
- Возможность описать в одном файле отдельно шаблон, скрипт и инкапсулированные стили компонента
- Работает при сборке через `vue-loader`



```
1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6   module.exports = {
7     data: function () {
8       return {
9         greeting: 'Hello'
10      }
11    }
12  }
13 </script>
14
15 <style scoped>
16   p {
17     font-size: 2em;
18     text-align: center;
19   }
20 </style>
```

Line 21, Column 1 Spaces: 2 Vue Component

Vue/cli

1. Генерация новых проектов по шаблонам: не просто один шаблон проекта, а интерактивный инструмент с выбором нужных компонентов и генерация на основе плагинов, в том числе в уже созданном проекте
2. Сборка и разработка приложения:
 - `vue-cli-service` - обёртка над `Webpack` с хорошим Production-ready конфигом
 - Сборка как веб-приложения, так и библиотеки с Vue-компонентом или Web-компонентом
3. **Vue UI** - графический интерфейс для создания и сборки проектов

SSR

- Серверный рендеринг предусмотрен из коробки.
- Компоненты можно рендерить на стороне сервера
- Полная реализация приложения с SSR трудоёмкая, но есть готовые фреймворки над Vue, например, **Nuxt**

Сильные стороны

- Достаточно простой для освоения
- Эффективная и простая реактивность
- Элегантные шаблоны
- Быстрое прототипирование
- Подходит как для использования у существующих проектах и миграции, так и для разработки с нуля

Слабые стороны

- Очень плохая поддержка TypeScript
- Есть сложности с переиспользованием логики
- "Микрофреймворк". не даёт архитектуру приложения
- Разные подходы в сообществе для решения одних и тех же задач

Vue 3

- Переписан на TypeScript и должен его лучше поддерживать
- Новая эффективная реактивность на основе Proxy
- Новый эффективный рендеринг
- Новый подход к переиспользованию логики на основе Composition API

Ссылки

- Vue: <https://vuejs.org>
- Vue/Cli: <https://cli.vuejs.org>
- Vue Router: <https://router.vuejs.org>
- Vuex: <https://vuex.vuejs.org>
- vue-dev-tools: <https://github.com/vuejs/vue-devtools>
- vue-ssr: <https://ssr.vuejs.org>
- vue-test-utils: <https://vue-test-utils.vuejs.org>
- vue-loader: <https://vue-loader.vuejs.org/>
- Nuxt: <https://nuxtjs.org>