



Web-технологии

Основы работы с Vue

Содержание

- Декларативная отрисовка
- Работа с вводом пользователя
- Условия и циклы
- Композиция приложения из компонентов
- Синтаксис шаблонов
- Вычисляемые свойства и наблюдатели
- Работа с классами и стилями
- Маршрутизация
- Vuex – единое состояние приложения

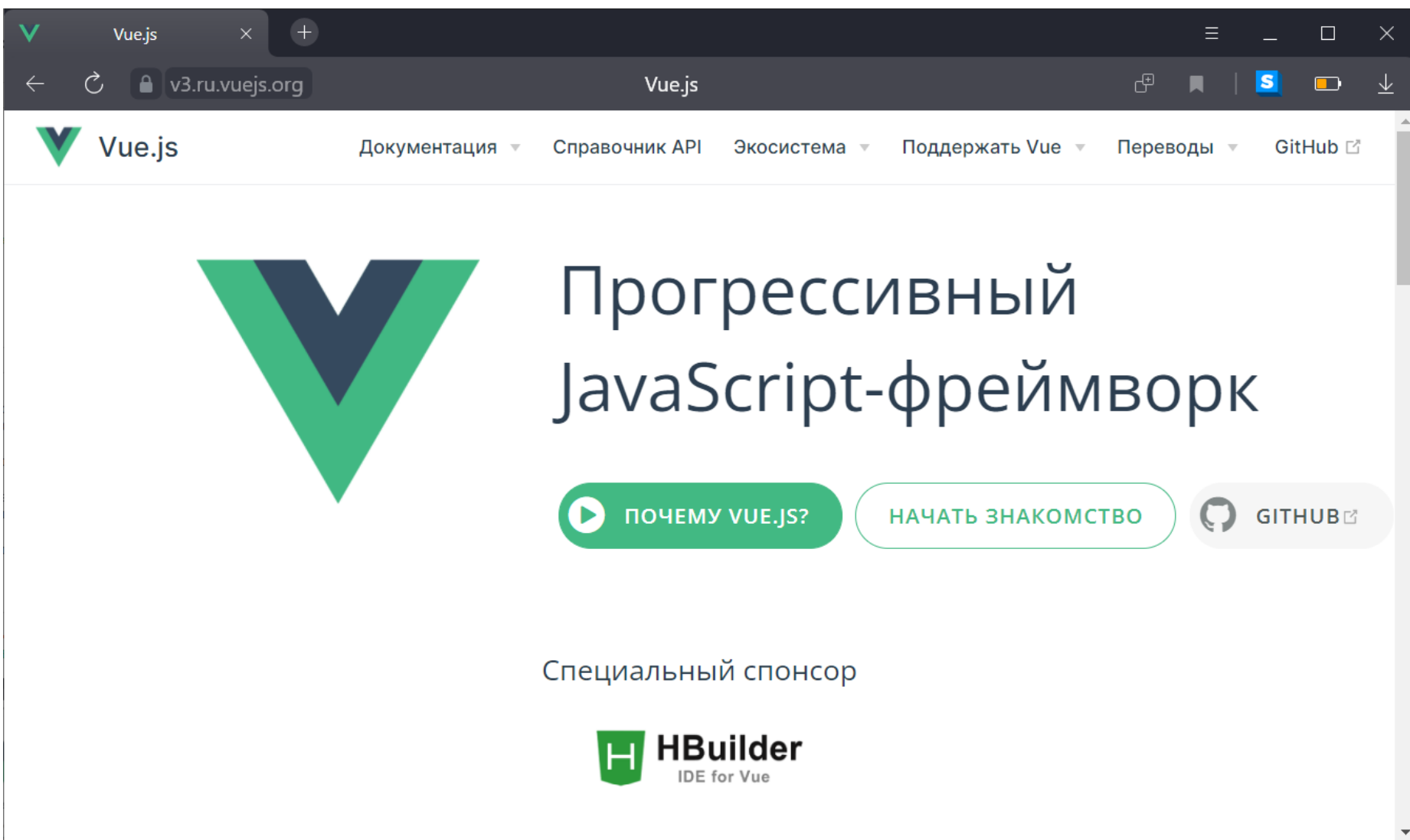
<https://v3.ru.vuejs.org/ru/>

<https://next.router.vuejs.org/>

<https://vuetifyjs.com/en/>

Vue.js – фреймворк JavaScript

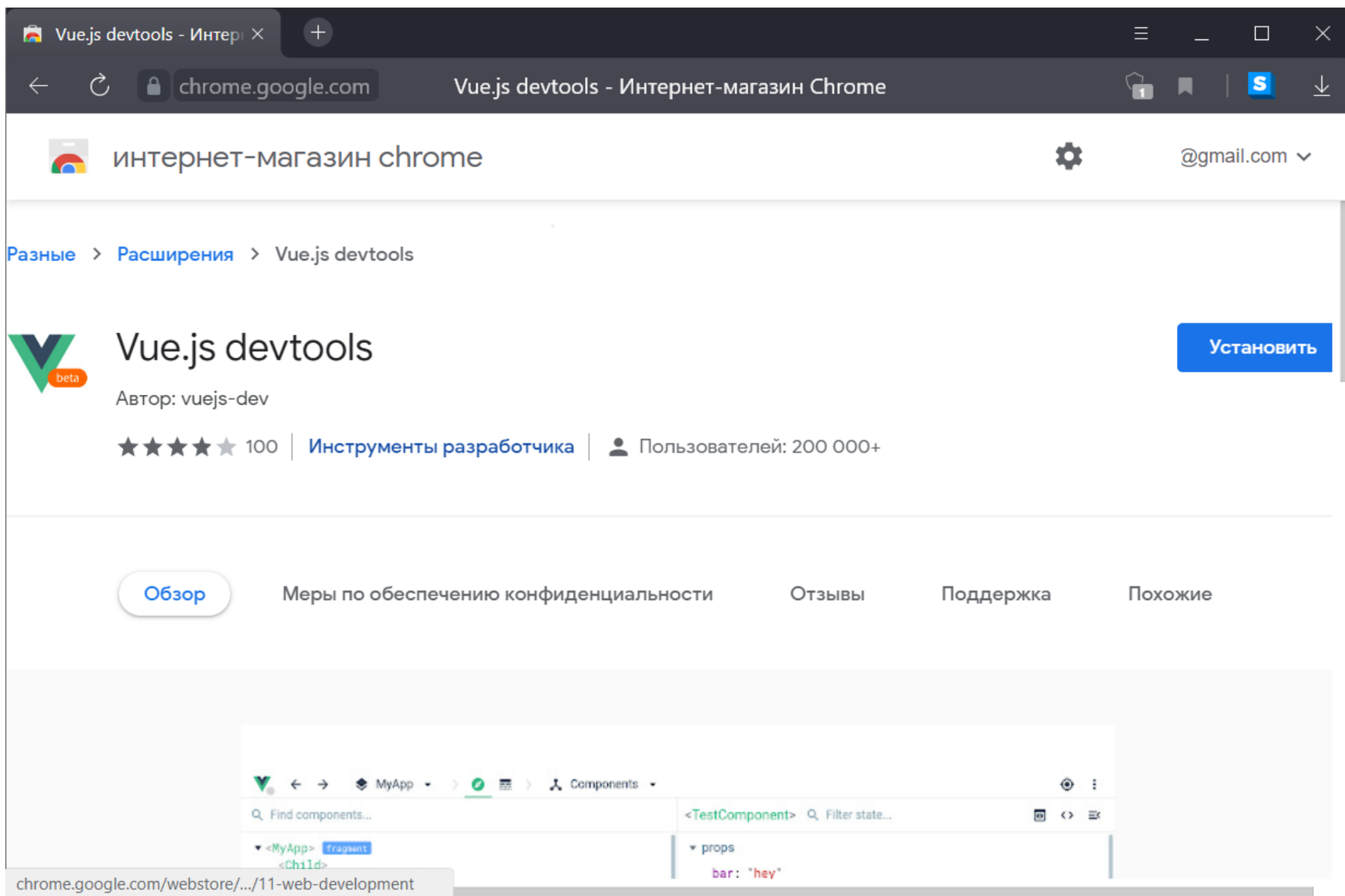
3



<https://v3.ru.vuejs.org/ru/>

Отладка: Vue.js devtools

4



Vue.js devtools - Интернет-магазин Chrome

интернет-магазин chrome

Разные > Расширения > Vue.js devtools

Vue.js devtools

Автор: vuejs-dev

★★★★★ 100 | Инструменты разработчика | Пользователей: 200 000+

Установить

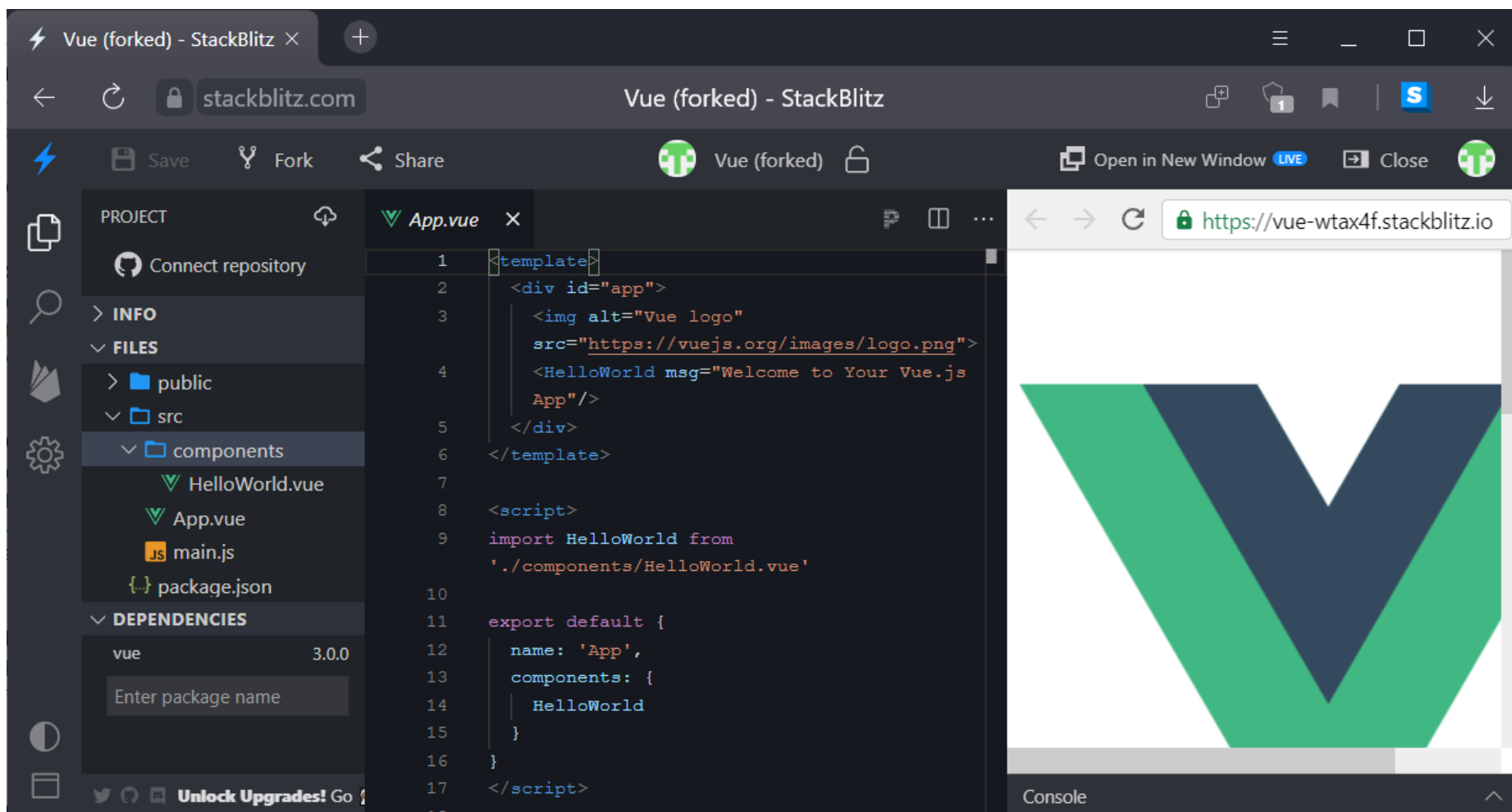
Обзор | Меры по обеспечению конфиденциальности | Отзывы | Поддержка | Похожие

chrome.google.com/webstore/.../11-web-development

<https://devtools.vuejs.org/>

Online-редакторы Vue3 (1)

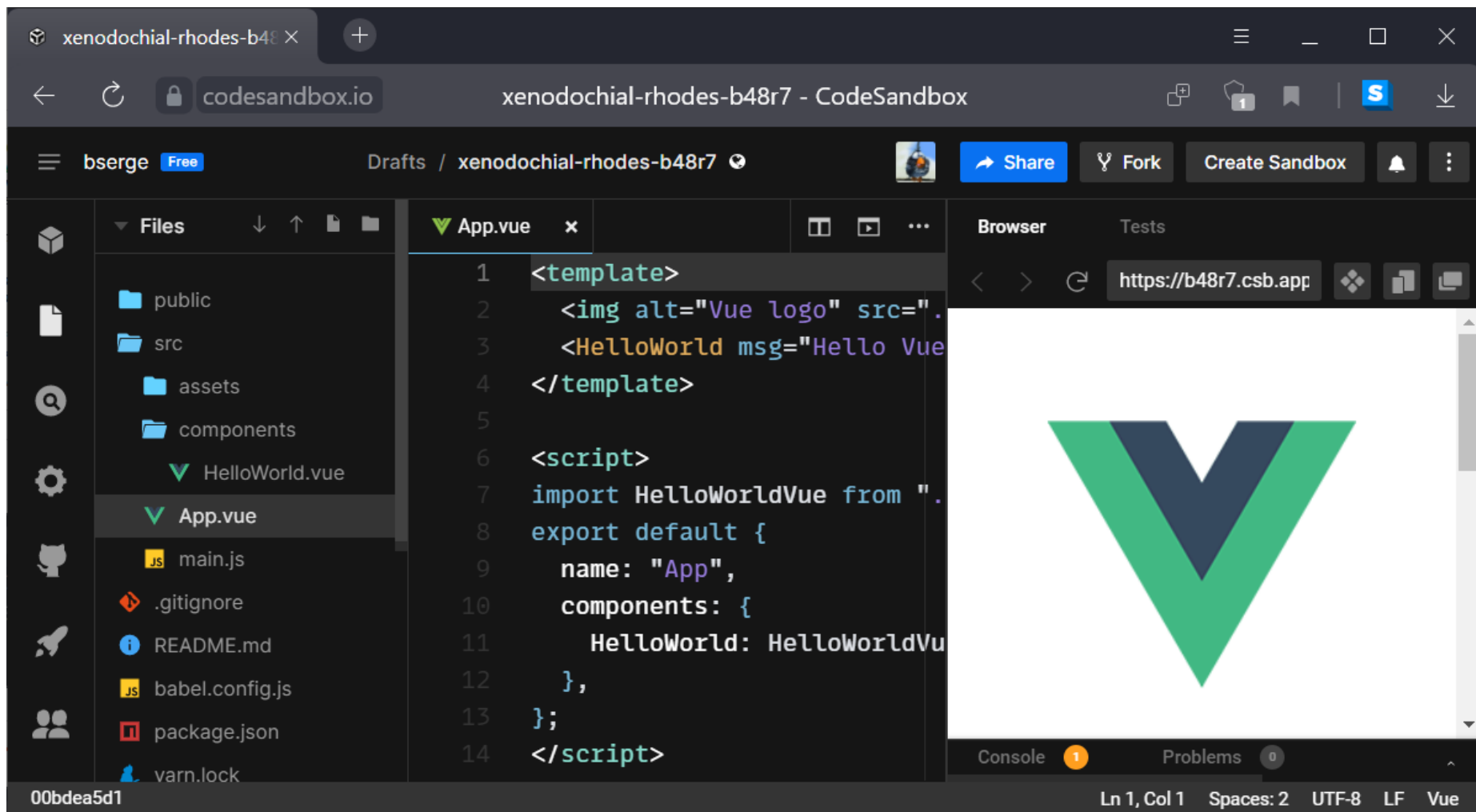
5



<https://stackblitz.com/>

Online-редакторы Vue3 (2)

6

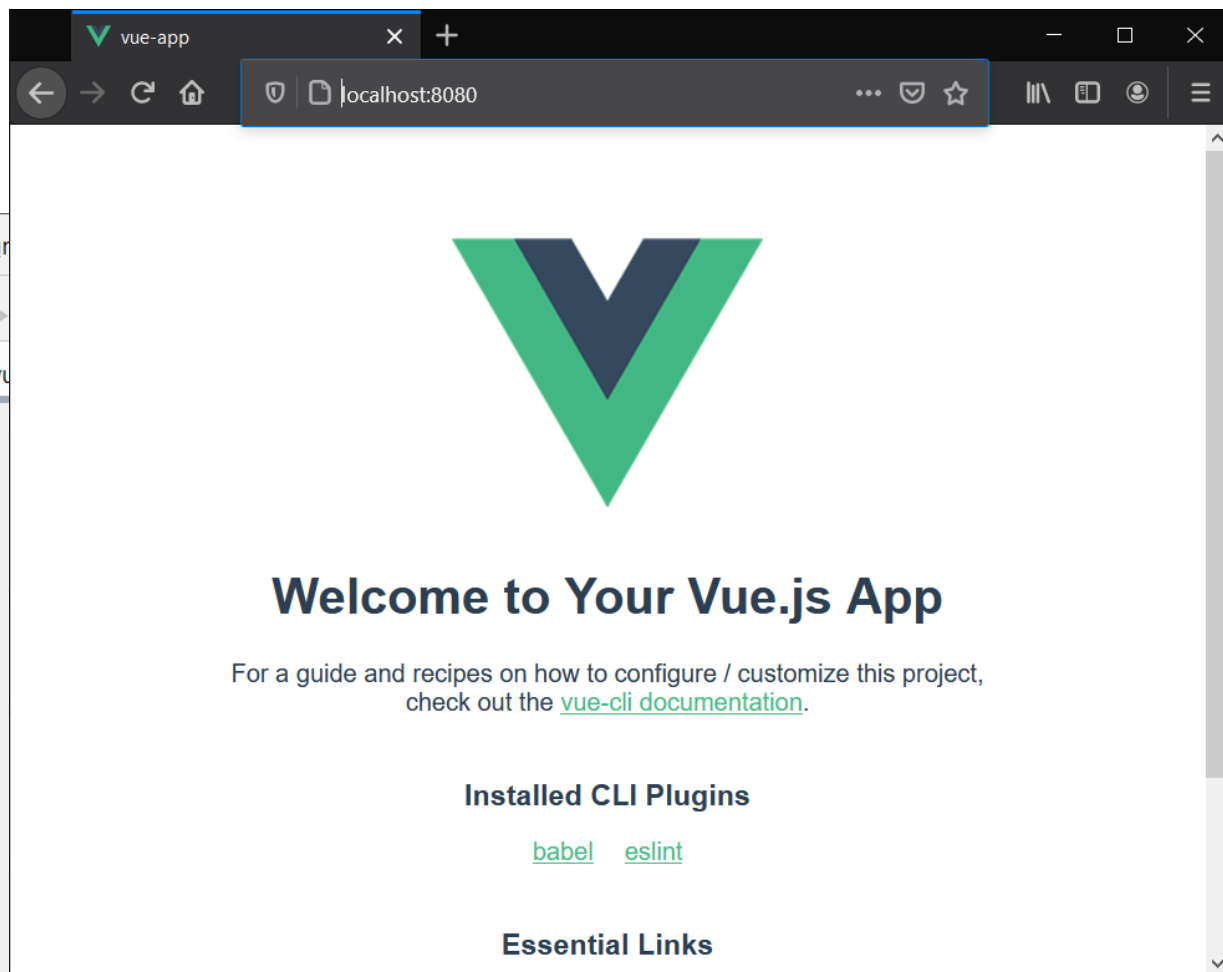
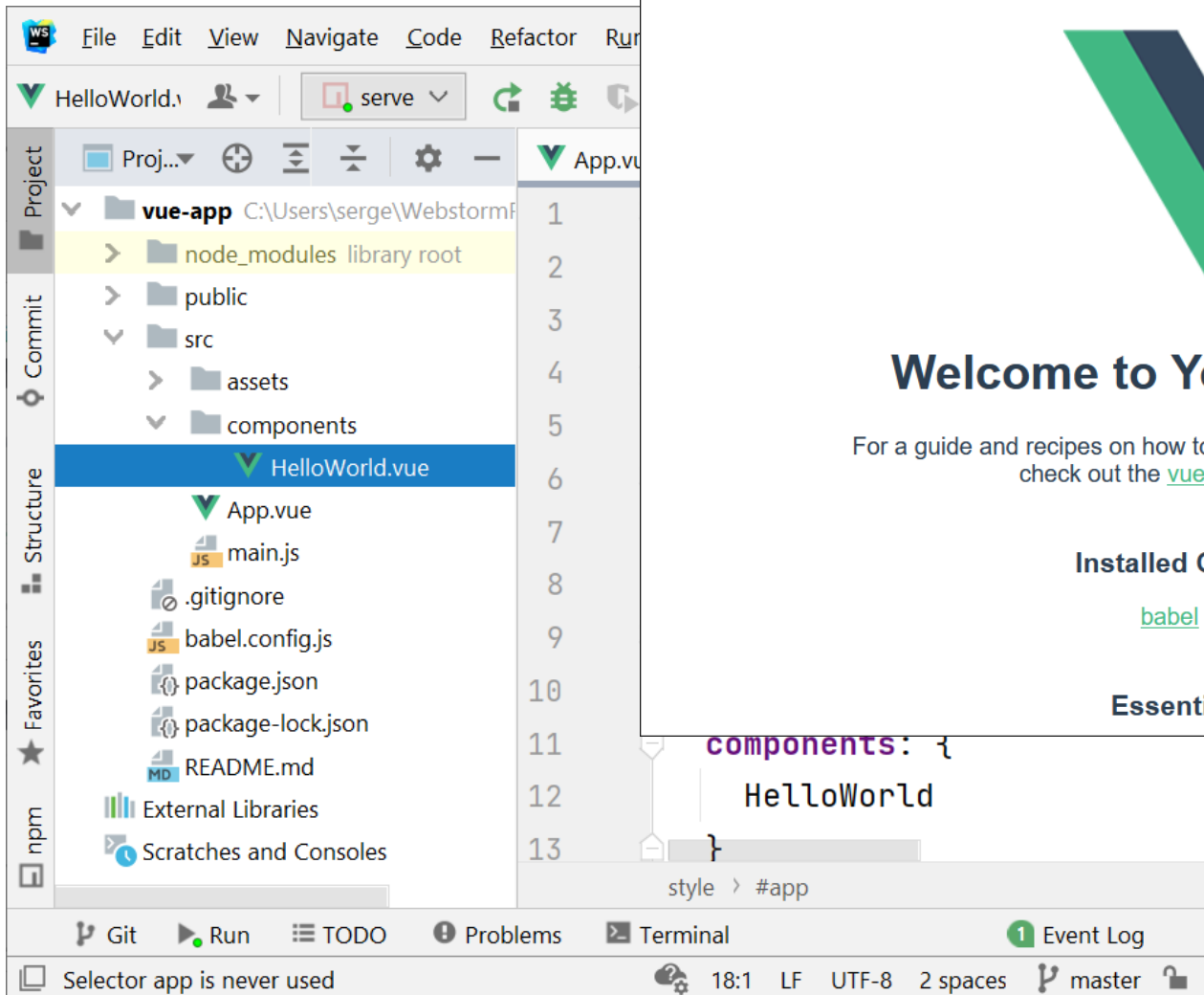


<https://codesandbox.io/>

Установка и создание Hello World

7

- `npm install -g @vue/cli`
- `vue create vue-app`



<http://localhost:8080/>

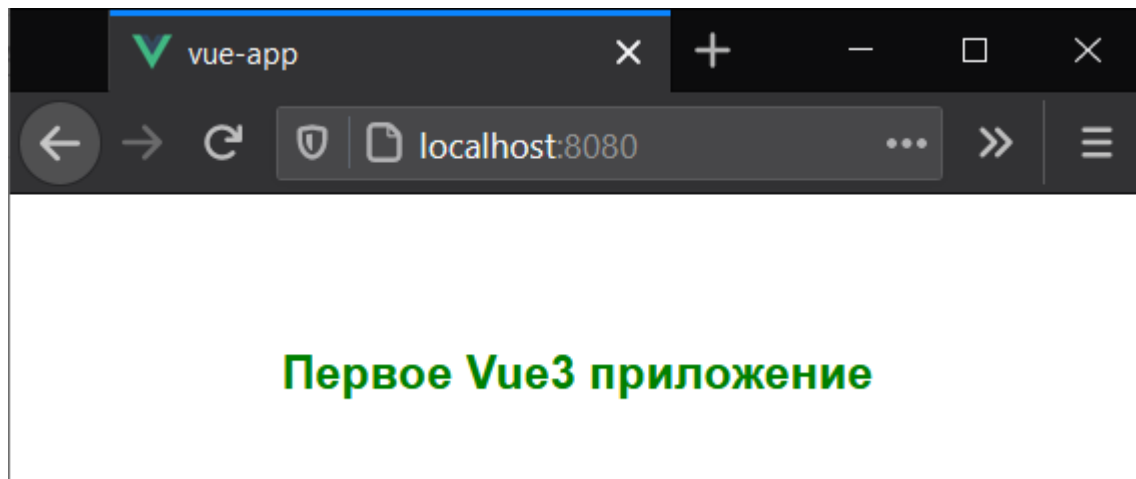
Hello world (1) index, main

index.html

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <div id="app"></div>
  </body>
</html>
```

main.js

```
import { createApp } from 'vue'
import App from './App.vue'
createApp(App).mount('#app')
```



Hello world (2) App, HelloWorld

9

App.vue

```
<template>
  <HelloWorld msg="Первое Vue3 приложение"/>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'
export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

components/HelloWorld.vue

```
<template>
  <div>
    <h3>{{ msg }}</h3>
  </div>
</template>

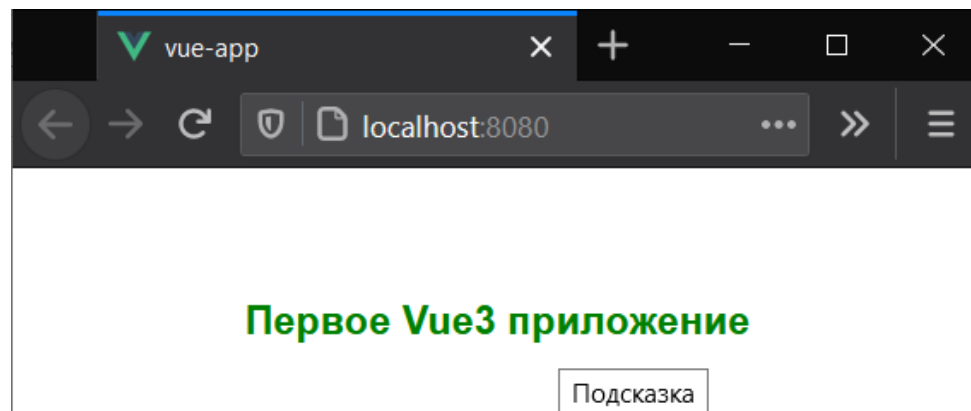
<script>
export default {
  name: 'HelloWorld',
  props: { // Получаемые параметры
    msg: String
  }
}
</script>

<!-- Настройка CSS для компонента -->
<style scoped>
h3 {
  color: green;
}
</style>
```

Директива v-bind – привязка

```
<template>
  <div>
    <h3 v-bind:title="mytitle">{{ msg }}</h3>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  props: { // Получаемые параметры
    msg: String
  },
  data() { // Параметры компонента
    return {
      mytitle: "Подсказка"
    }
  }
}
</script>
<style scoped>
h3 { color: green; }
</style>
```

Директива говорит «сохраняй значение **title** этого элемента актуальным при изменении свойства **mytitle** в текущем активном экземпляре»

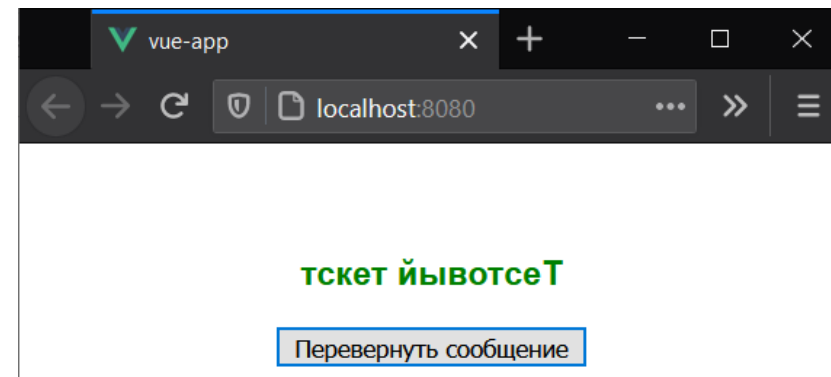
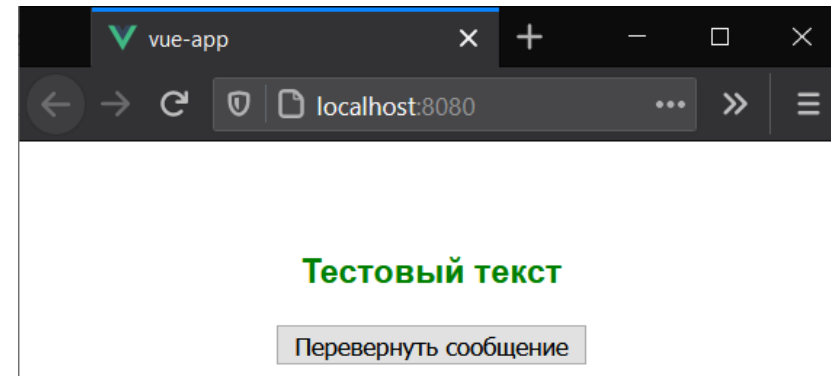


Директива v-on – ввод пользователя

11

```
<template>
  <div>
    <h3>{{ msg }}</h3>
    <button v-on:click="reverse">Перевернуть сообщение</button>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() { // Параметры компонента
    return {
      msg: "Тестовый текст"
    }
  },
  methods: { // Методы компонента
    reverse() {
      this.msg = this.msg.split("").reverse().join("")
    }
  }
}
</script>
<style scoped>
h3 { color: green; }
</style>
```

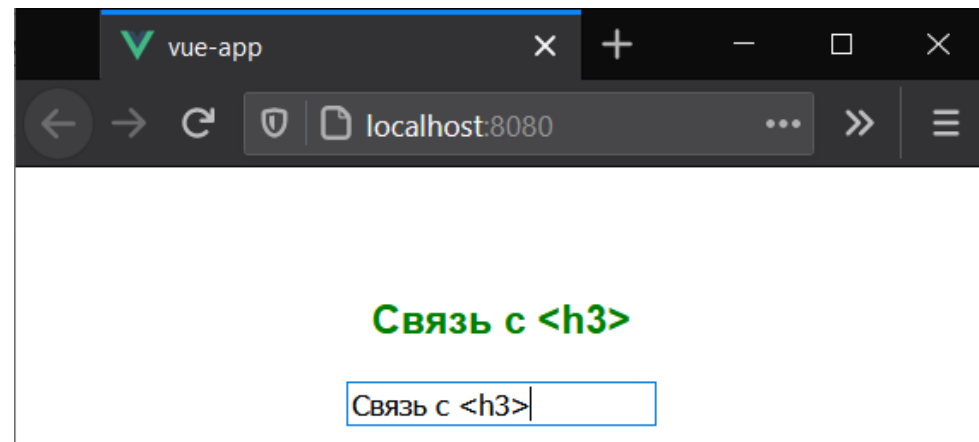
В методе не трогаем DOM и обновляем только состояние приложения



Директива v-model – двусторонняя СВЯЗЬ

12

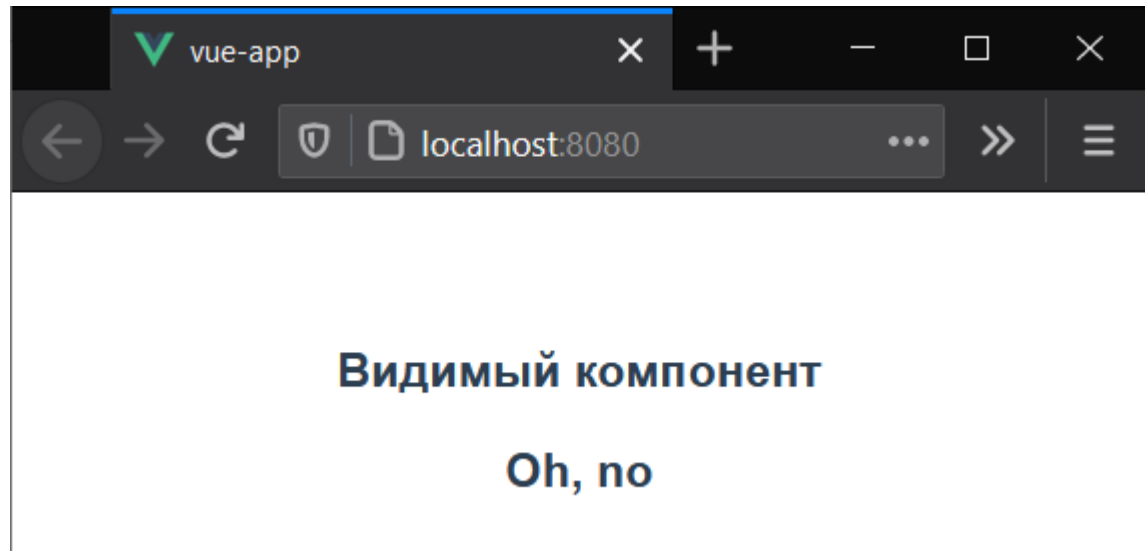
```
<template>
  <div>
    <h3>{{ msg }}</h3>
    <input v-model="msg" placeholder="Введите текст">
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() { // Параметры компонента
    return {
      msg: "Тестовый текст"
    }
  }
}
</script>
<style scoped>
h3 { color: green; }
</style>
```



Директива v-if/v-else – условие

13

```
<template>
  <div>
    <h3 v-if="seen">Видимый компонент</h3>
    <h3 v-if="!seen">Невидимый компонент</h3>
    <h3 v-else>Oh, no</h3>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      seen: true
    }
  }
}
</script>
```

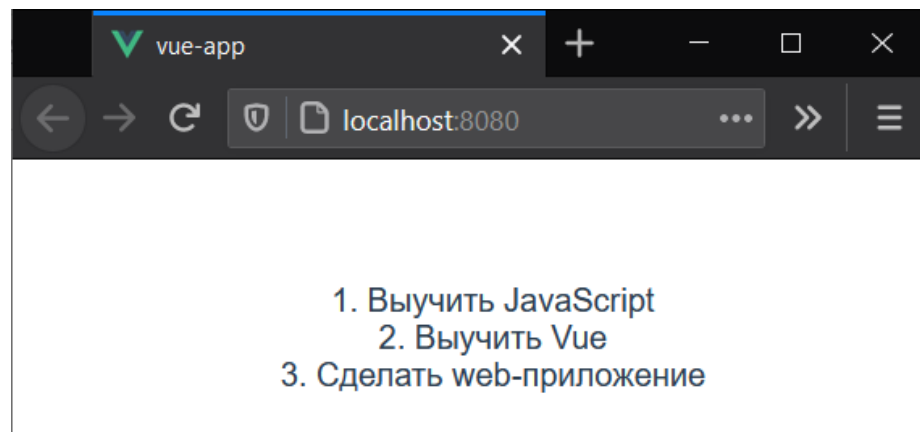


- **v-show** – сохраняет компонент в DOM, не совместим с **<template>**, **v-else**
- **v-if** – удаляет ненужные компоненты из DOM
- **v-else-if**
- **<template v-if="">**

Директива v-for – цикл

C работает в связке с v-bind:key

```
<template>
  <div>
    <ol>
      <li v-for="item in mylist" v-bind:key="item.id">
        {{item.text}}
      </li>
    </ol>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      mylist: [
        {id:1, text: "Выучить JavaScript"},
        {id:2, text: "Выучить Vue"},
        {id:3, text: "Сделать web-приложение"}
      ]
    }
  }
}
</script>
```



- Вместо **mylist** может быть
 - объект – итерация по атрибутам
 - имя функции
 - натуральное число
- Поддерживает **<template>** для отрисовки нескольких элементов

Доступ к индексу элемента

```
<li v-for="(item, index) in mylist" v-bind:key="item.id">
```

Создание экземпляра приложения main.js – три варианта

```
import Vue from 'vue'  
// Создание приложения  
const app = Vue.createApp({  
  /* опции */  
})
```

```
import Vue from 'vue'  
// Создание приложения  
const app = Vue.createApp({})  
// Регистрация глобальных компонентов, директив, плагинов  
app.component('SearchInput', SearchInputComponent)  
app.directive('focus', FocusDirective)  
app.use(LocalePlugin)
```

```
import Vue from 'vue'  
// Создание приложения - функциональный стиль  
Vue.createApp({})  
  .component('SearchInput', SearchInputComponent)  
  .directive('focus', FocusDirective)  
  .use(LocalePlugin)
```

```
// Приложение требуется примонтировать в DOM-элемент  
// mount - возвращает экземпляр корневого компонента  
const vm = app.mount('#app')
```

Свойства экземпляра компонента

16

```
import Vue from 'vue'
const app = Vue.createApp({
  data() {
    return { count: 4 }
  }
})
const vm = app.mount('#app')
// Свойства доступны в экземпляре компонента
console.log(vm.count) // => 4
// Другие опции компонента
// methods - методы
// props - свойства, получаемые от родителя
// computed - вычисляемые
// inject - встраиваемые
// setup - настройки
```


Хуки жизненного цикла (1)

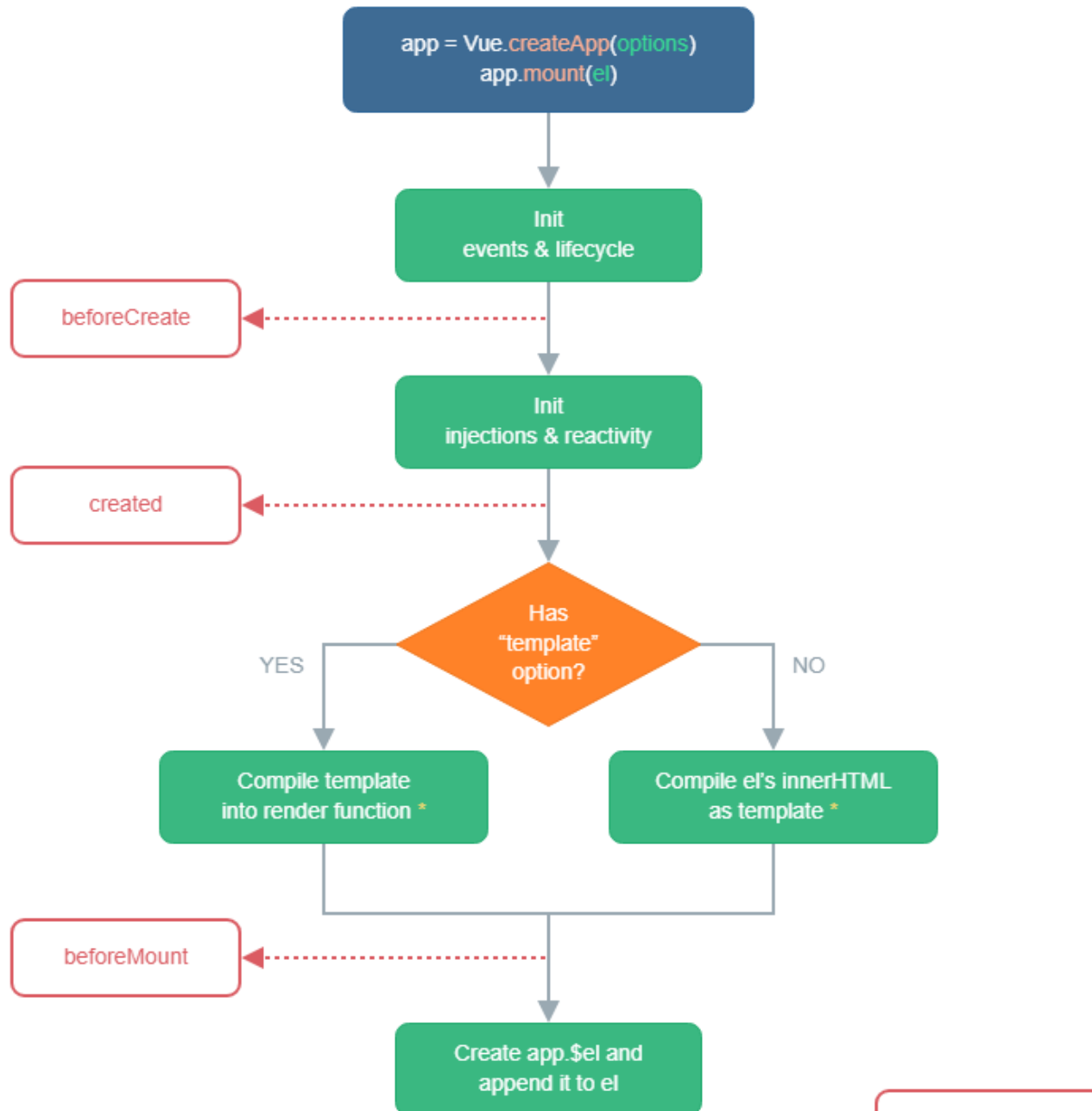
17

```
import Vue from 'vue'
Vue.createApp({
  data() {
    return { count: 1 }
  },
  created() { // Хук вызывается при создании
    // `this` указывает на экземпляр vm
    console.log('счётчик: ' + this.count) // => "счётчик: 1"
  }
})
// mounted - вызывается после монтирования
// updated - вызывается после обновления виртуального DOM
//           из-за изменения данных
// unmounted - вызывается после размонтирования

// Все хуки вызываются с контекстом this, указывающим на текущий
// активный экземпляр, который их вызвал
```

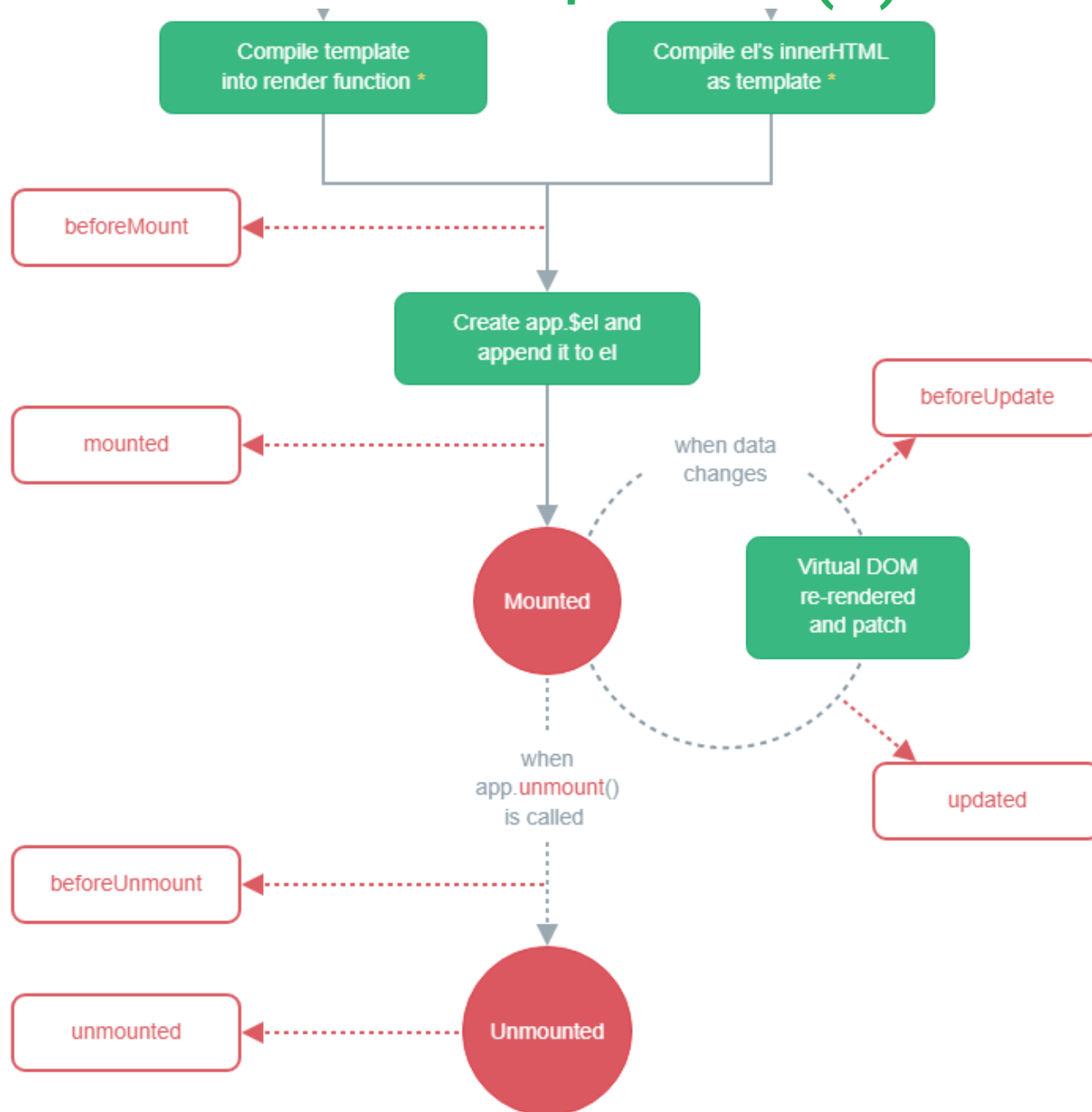
Хуки жизненного цикла (2)

18



Хуки жизненного цикла (2)

19



Интерполяция

20

****Текстовая интерполяция: {{ msg }}****

{{ number + 1 }}

{{ ok ? 'YES' : 'NO' }}

{{ message.split('').reverse().join('') }}

<!-- Привязка поддерживает ОДНО выражение -->

****Это сообщение никогда не изменится: {{ msg }}****

<p>Обрабатывается как текст: {{ rawHtml }}**</p>**

<p>Обрабатывается как html: **</p>**

<!-- Нельзя использовать {{}} в атрибутах, вместо этого v-bind

Не добавится в случаях null или undefined-->

<!-- В случае булевых атрибутов:

disabled добавится, если isMyDisabled == true || "" -->

<button v-bind:disabled="isMyDisabled">Кнопка</button>

v-once очень быстро обрабатывает статическое содержимое, затем содержимое кэшируется

Директивы / v-

<!-- v-bind - реактивное обновление атрибутов -->

**<a v-bind:href="url"> ... **

**<a :href="url"> ... ** *<!-- Сокращенная запись -->*

<!-- v-on - отслеживание событий DOM -->

**<a v-on:click="doSomething"> ... **

**<a @click="doSomething"> ... ** *<!-- Сокращенная запись -->*

*<!-- [] - динамические атрибуты,
attributeName/eventName - JavaScript-выражение -->*

**<a v-bind:[attributeName]="url"> ... **

**<a :[attributeName]="url"> ... ** *<!-- Сокращенная запись -->*

**<a v-on:[eventName]="doSomething"> ... **

**<a @[eventName]="doSomething"> ... ** *<!-- Сокращенная запись -->*

*<!-- Модификатор .prevent даёт указание директиве v-on
вызвать event.preventDefault() при обработке события -->*

<form v-on:submit.prevent="onSubmit">...</form>

Свойства data

22

```
import Vue from 'vue'
const app = Vue.createApp({
  data() {
    return { count: 4 }
  }
})
const vm = app.mount('#app')
```

```
// data() сохраняется в $data
console.log(vm.$data.count) // => 4
console.log(vm.count)      // => 4
```

```
// Присвоение значения в vm.count также обновит $data.count
vm.count = 5
console.log(vm.$data.count) // => 5
```

```
// ... и наоборот
vm.$data.count = 6
console.log(vm.count) // => 6
```

```
// Все свойства должны быть сразу объявлены в data()
```

Методы methods

23

```
import Vue from 'vue'
const app = Vue.createApp({
  data() {
    return { count: 4 }
  },
  methods: {
    increment() {
      // `this` указывает на экземпляр компонента
      this.count++
    }
  }
})
const vm = app.mount('#app')
```

```
console.log(vm.count) // => 4
vm.increment()
console.log(vm.count) // => 5
```

Использованием методов

```
<button @click="increment">Inc</button>
<!-- Параметры - доступные переменные -->
<span :title="toDate(date)">
  {{ formatDate(date) }}
</span>
```

Вычисляемый метод / computed

```
<template>
  <div>
    <span>{{ listLength }}</span>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      myList: [
        {id:1, text: "Выучить JavaScript"},
        {id:2, text: "Выучить Vue"},
        {id:3, text: "Сделать web-приложение"}
      ]
    }
  },
  computed: {
    // геттер вычисляемого метода
    listLength() {
      return this.mylist.length
    }
    // В отличие от обычного метода вычисляемые
    // свойства КЭШИРУЮТСЯ
  }
}
```


Сеттер вычисляемого значения

```

<template>
  <div><span>{{ fullName }}</span></div>
</template>
<script>
export default {
  data() {
    return {
      firstName: undefined,
      lastName: undefined
    } // Даже если неизвестны - нужно объявить
  },
  computed: {
    fullName: {
      get() { // геттер (для получения значения)
        return `${this.firstName} ${this.lastName}`
      },
      set(newValue) { // сеттер (при присвоении нового значения)
        const names = newValue.split(' ')
        this.firstName = names[0]
        this.lastName = names[1]
      }
    }
  }
}
</script>

```

Методы-наблюдатели / watch

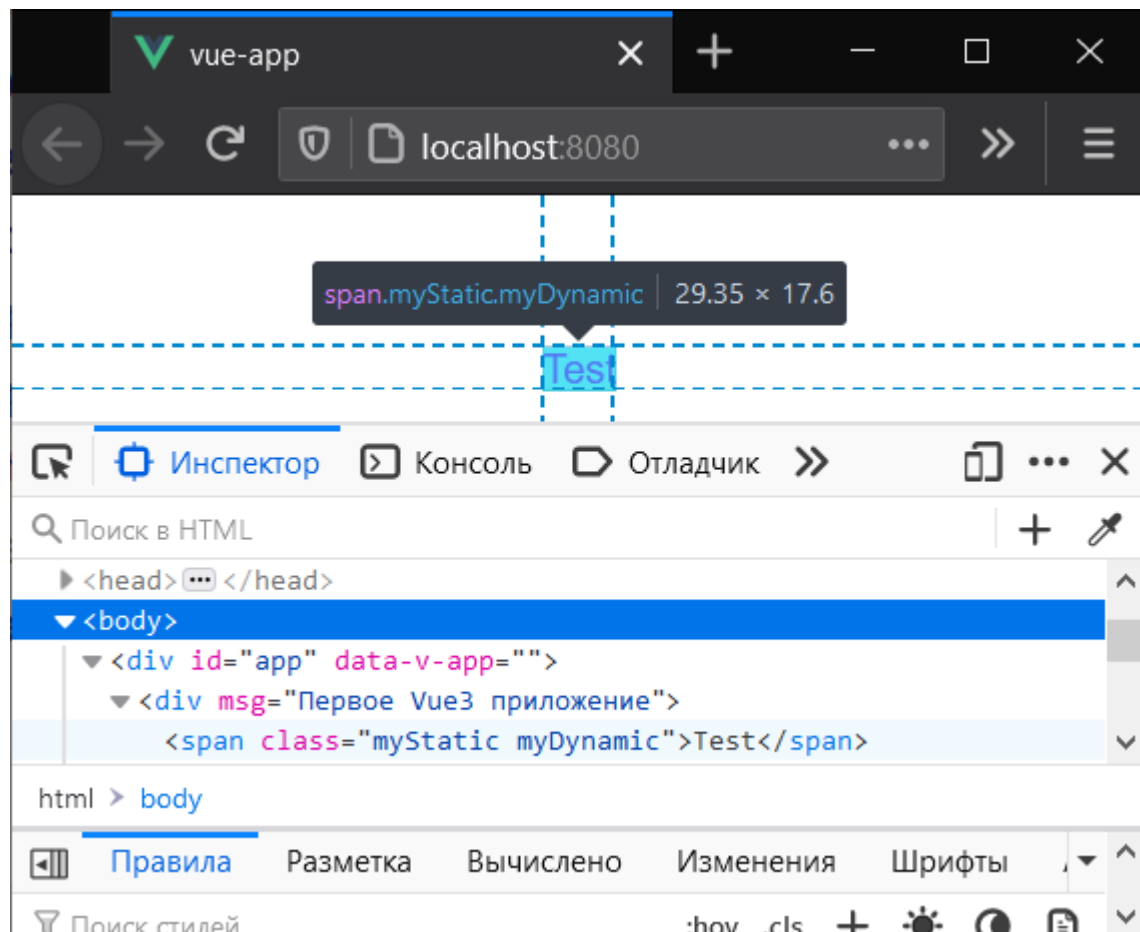
26

```
<template>
  <div><input v-model="question" /></div>
</template>
<script>
export default {
  data() {
    return {
      question: ""
    }
  },
  watch: {
    // при каждом изменении `question` эта функция будет запускаться
    question(newQuestion, oldQuestion) {
      if(newQuestion.indexOf("?") > -1)
        this.execute()
    }
  },
  methods: {
    execute() {
      console.log(this.question)
    }
  }
}
</script>
```

Часто задача может быть решена через вычисляемые методы!

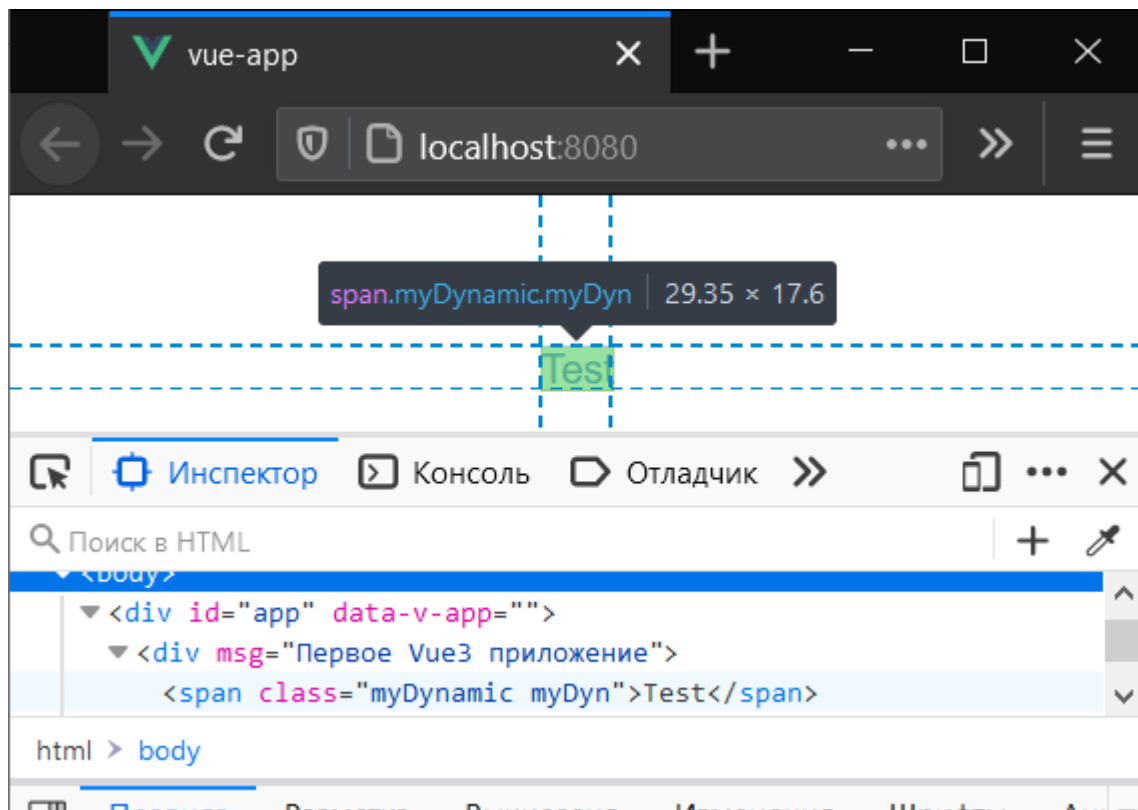
Связывание CSS-классов – объектный синтаксис / :class

```
<template>
  <div>
    <span class="myStatic" :class="{ myDynamic: dynamic }">Test</span>
  </div>
</template>
<script>
export default {
  data() {
    return {
      dynamic: true
    }
  }
}
</script>
<style>
.myStatic {
  color: blue;
}
.myDynamic {
  background-color: cyan;
}
</style>
```



Связывание CSS-классов – синтаксис с массивом / []

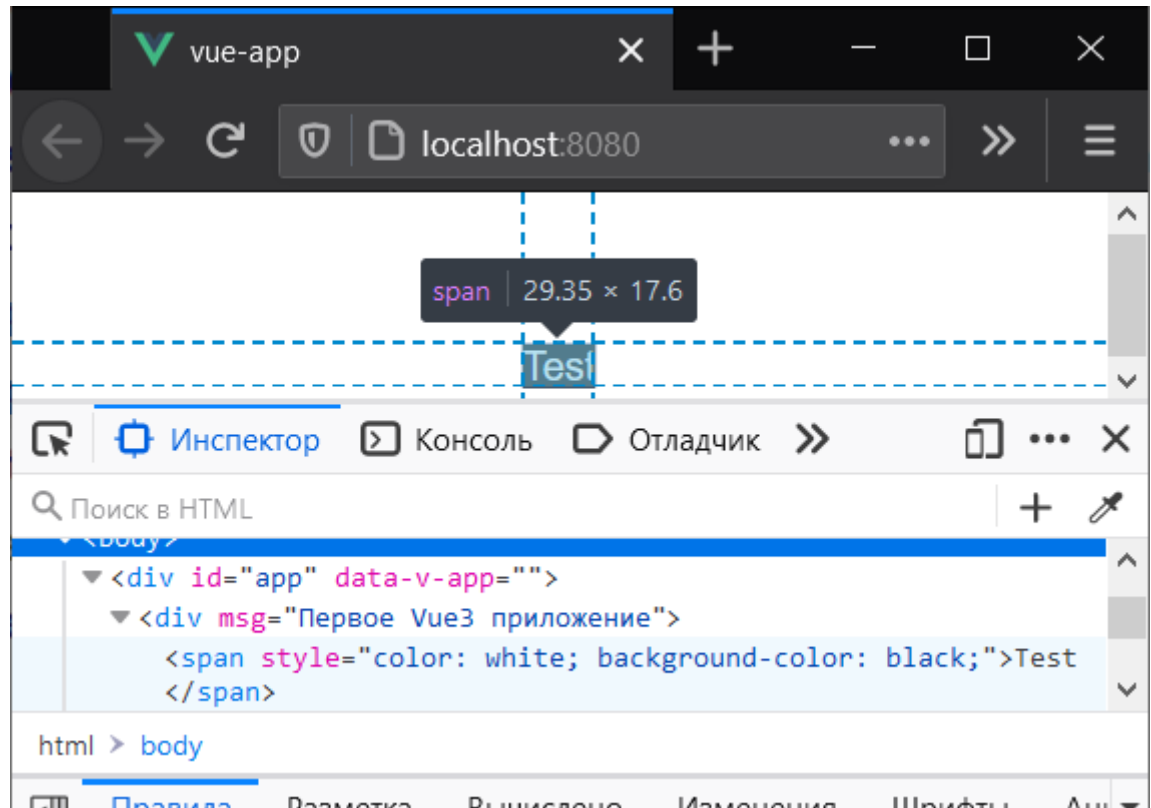
```
<template>
  <div>
    <span :class="[dynamic, dyn]">Test</span>
  </div>
</template>
<script>
export default {
  data() {
    return {
      dynamic: "myDynamic",
      dyn: "myDyn"
    }
  }
}
</script>
<style>
.myDyn {
  color: green;
}
.myDynamic {
  background-color: greenyellow;
}
</style>
```



Связывание inline-стилей

29

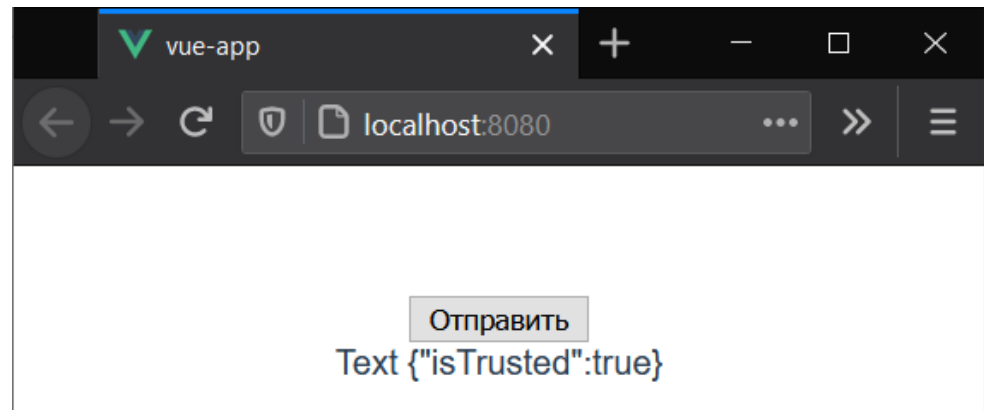
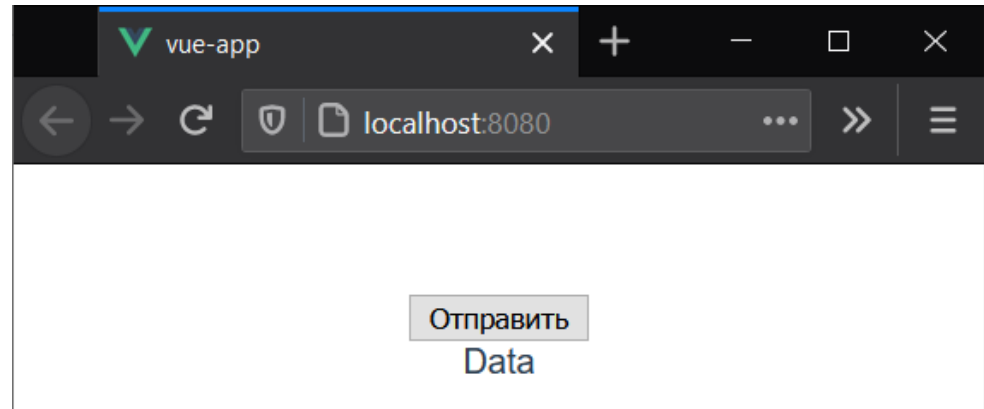
```
<template>
  <div>
    <span :style="{ color: dynColor, backgroundColor: dynBack}">Test</span>
  </div>
</template>
<script>
export default {
  data() {
    return {
      dynColor: "white",
      dynBack: "black"
    }
  }
}
</script>
```



Параметры события

30

```
<template>
  <div>
    <button @click="warn('Text', $event)">Отправить</button>
    <div>{{text}} {{myEvent}}</div>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      myEvent: undefined,
      text: "Data"
    }
  },
  methods: {
    warn(msg, event) {
      this.text = msg
      this.myEvent = JSON.stringify(event)
    }
  }
}
</script>
```



Несколько обработчиков события

```
<button @click="one($event), two($event)">
```

Модификаторы событий v-on

31

<!-- всплытие события click будет остановлено -->

<a @click.stop="doThis">

<!-- событие submit перестанет перезагружать страницу -->

<form @submit.prevent="onSubmit"></form>

<!-- модификаторы можно объединять в цепочки -->

<a @click.stop.prevent="doThat">

<!-- можно использовать без обработчиков -->

<form @submit.prevent></form>

<!-- можно отслеживать события в режиме capture, т.е. событие, нацеленное -->

<!-- на внутренний элемент, обрабатывается здесь до обработки этим элементом -->

<div @click.capture="doThis">...</div>

<!-- вызов обработчика только в случае наступления события непосредственно -->

<!-- на данном элементе (то есть не на дочернем компоненте) -->

<div @click.self="doThat">...</div>

При использовании модификаторов имеет значение их порядок

Модификаторы клавиш/мыши

32

.enter

.tab

.delete

.esc

.left

.space

.up

.down

.left

.right

.right

.ctrl

.alt

.shift

.middle

.meta

<!-- вызвать `vm.submit()` только если `key` будет `Enter` -->

<input @keyup.enter="submit" />

<!-- Alt + Enter -->

<input @keyup.alt.enter="clear" />

<!-- Ctrl + Click -->

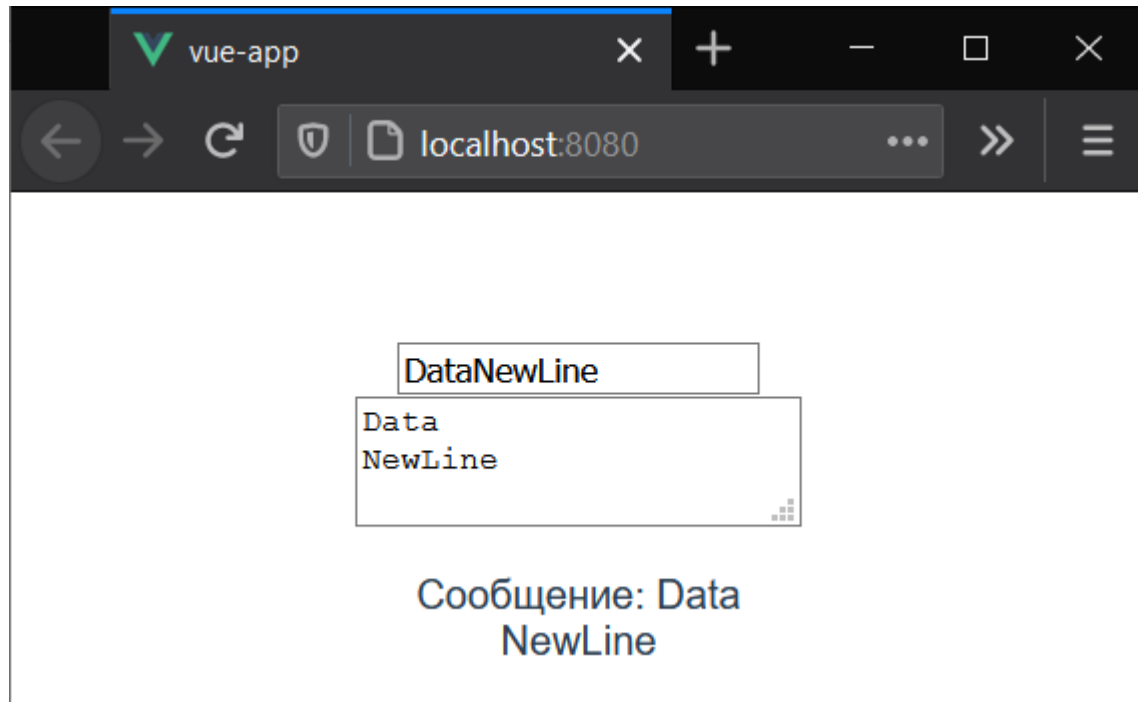
<div @click.ctrl="doSomething">Сделать что-нибудь</div>

Модификатор **.exact** позволяет контролировать точную комбинацию системных модификаторов

На клавиатурах **Apple** клавиша **meta** отмечена знаком ⌘. На клавиатурах **Windows** клавиша **meta** отмечена знаком ⌞.

Формы – работа с текстом / v-model 33

```
<template>
  <div>
    <input v-model="message"/><br>
    <textarea v-model="message"></textarea><br>
    <p style="white-space: pre-line;">Сообщение: {{ message }}</p>
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      message: "Data"
    }
  }
}
</script>
```



Модификаторы v-model

- **.lazy** – синхронизация после change
- **.number** – приведение к числу
- **.trim** – удаление пробелов

Формы – работа с checkbox

34

```
<template>
  <form>
    <input type="checkbox" id="checkbox" v-model="checked" />
    <label for="checkbox">Выбор</label> {{checked}}<br>
    <fieldset>
      <input type="checkbox" id="jack" value="jeck" v-model="checkedNames" />
      <label for="jack">Джек</label>
      <input type="checkbox" id="john" value="john" v-model="checkedNames" />
      <label for="john">Джон</label><br>
      <span>Отмеченные имена: {{ checkedNames }}</span>
    </fieldset>
  </form>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      checked: false,
      checkedNames: []
    }
  }
}
</script>
```

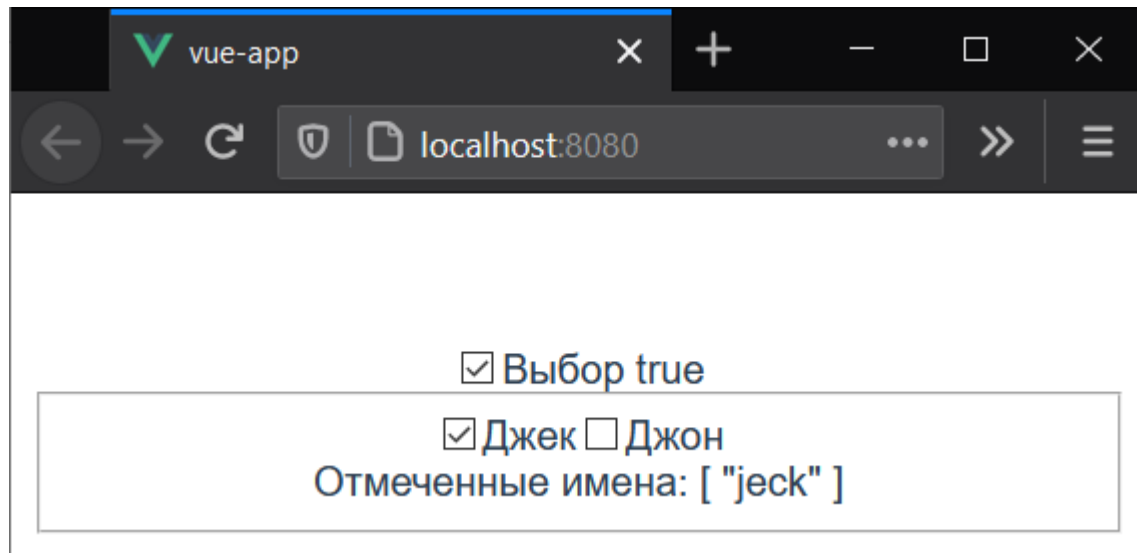
Привязка значений

radio:

- **value**="myvalue"

checkbox:

- **true-value**="да"
false-value="нет"



Аналогично для радиокнопок

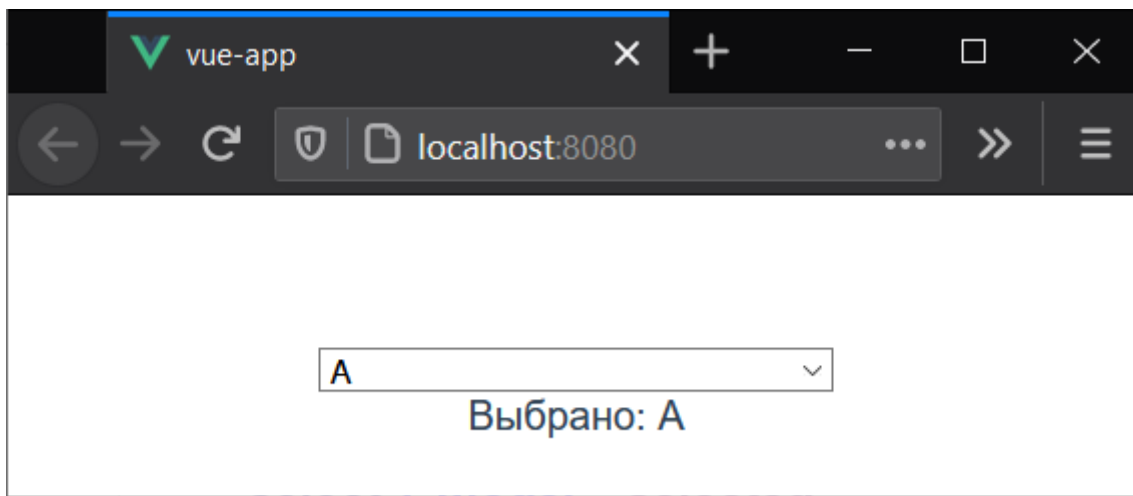
```
<input type="radio" id="jack" value="jeck" v-model="checkedName" />
```

Выпадающий список

35

```
<template>
  <form>
    <select v-model="selected">
      <option disabled value="">Выберите один из вариантов</option>
      <option>А</option>
      <option>Б</option>
      <option>В</option>
    </select><br>
    <span>Выбрано: {{ selected }}</span>
  </form>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      selected: undefined
    }
  }
}
</script>
```

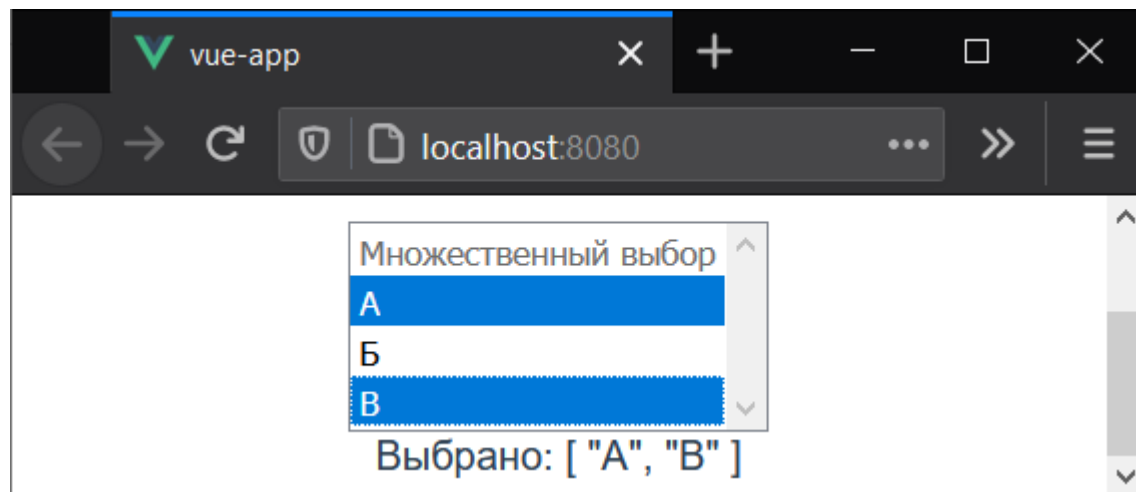
Вариант не
доступен для
выбора



Множественный выбор

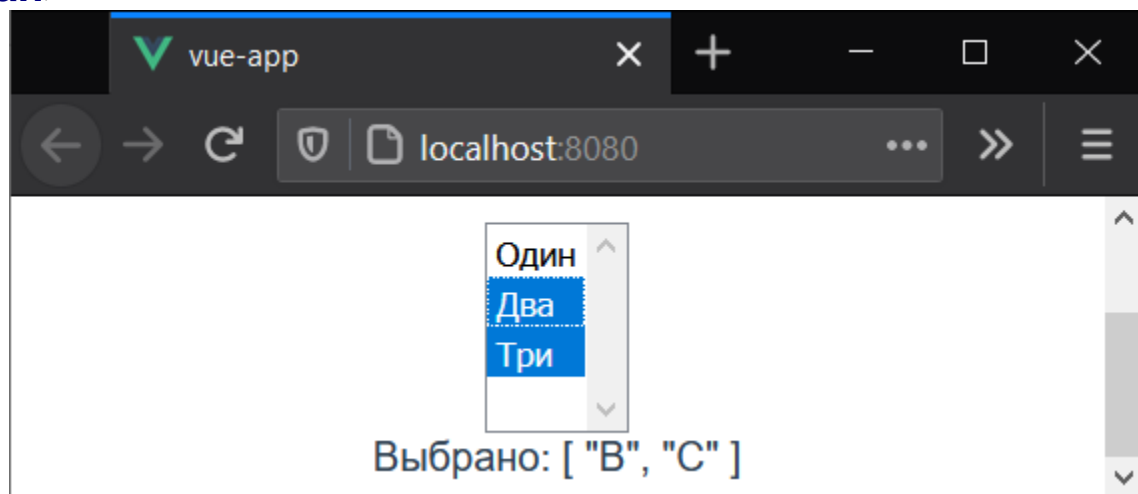
36

```
<template>
  <form>
    <select v-model="selected" multiple>
      <option disabled value="">Множественный выбор</option>
      <option>А</option>
      <option>Б</option>
      <option>В</option>
    </select><br>
    <span>Выбрано: {{ selected }}</span>
  </form>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      selected: []
    }
  }
}
</script>
```



Динамические option в select

```
<template>
  <form>
    <select v-model="selected" multiple>
      <option v-for="option in options" :value="option.value" :key="option.id">
        {{ option.text }}
      </option>
    </select><br>
    <span>Выбрано: {{ selected }}</span>
  </form>
</template>
<script>
export default {
  name: 'HelloWorld',
  data() {
    return {
      selected: [],
      options: [
        { text: 'Один', value: 'A', id: 'A' },
        { text: 'Два', value: 'B', id: 'B' },
        { text: 'Три', value: 'C', id: 'C' }
      ]
    }
  }
}
</script>
```



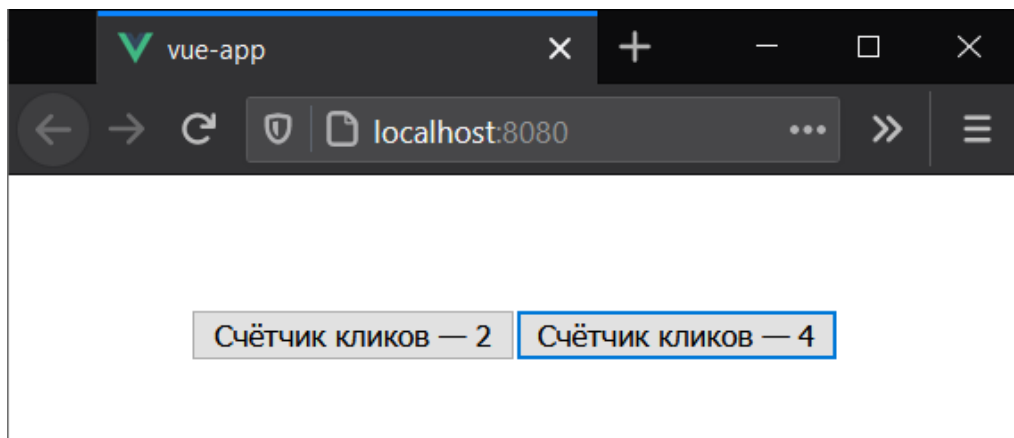
Подключение компонентов

components/HelloWorld.vue

```
<template>
  <div>
    <Button></Button>
    <Button></Button>
  </div>
</template>
<script>
import Button from "@components/Button";
export default {
  name: 'HelloWorld',
  components: {Button}
}
</script>
```

components/Button.vue

```
<template>
  <button @click="count++">
    Счётчик кликов — {{ count }}
  </button>
</template>
<script>
export default {
  name: "Button",
  data() {
    return {
      count: 0
    }
  }
}
</script>
```



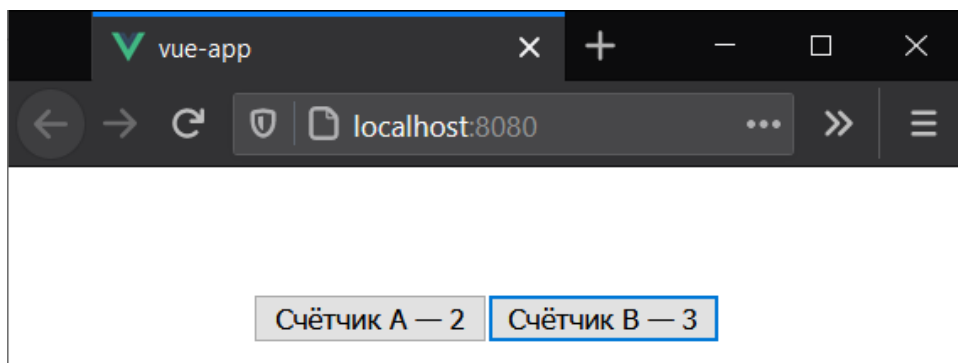
Передача параметра потомку

components/HelloWorld.vue

```
<template>
  <div>
    <Button btn-label="Счётчик А"></Button>
    <Button btn-label="Счётчик В"></Button>
  </div>
</template>
<script>
import Button from "@components/Button";
export default {
  name: 'HelloWorld',
  components: {Button}
}
</script>
```

components/Button.vue

```
<template>
  <button @click="count++">
    {{ btnLabel }} — {{ count }}
  </button>
</template>
<script>
export default {
  name: "Button",
  // Получено от родителя в camelCase
  props: ["btnLabel"],
  data() {
    return {
      count: 0
    }
  }
}
</script>
```



Передача события родителю / \$emit()

40

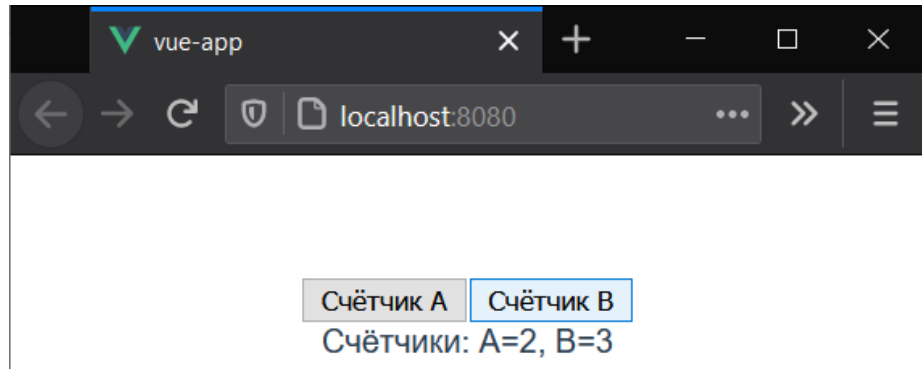
components/HelloWorld.vue

```
<template>
  <div>
    <Button btn-label="Счётчик А" idx="0" @inc-value="increment"></Button>
    <Button btn-label="Счётчик В" idx="1" @inc-value="increment"></Button>
    <div>Счётчики: A={{counter[0]}}, B={{counter[1]}}</div>
  </div>
</template>
<script>
import Button from "@components/Button";
export default {
  name: 'HelloWorld',
  components: {Button},
  data() {
    return {
      counter: [0, 0]
    }
  },
  methods: {
    increment(index) {
      this.counter[index]++
    }
  }
}
</script>
```

- btn-label → btnLabel
- inc-value → incValue
- idx – первый параметр

components/Button.vue

```
<template>
  <button @click="$emit('incValue', idx)">
    {{ btnLabel }}
  </button>
</template>
<script>
export default {
  name: "Button",
  props: ["btnLabel", "idx"]
}
</script>
```



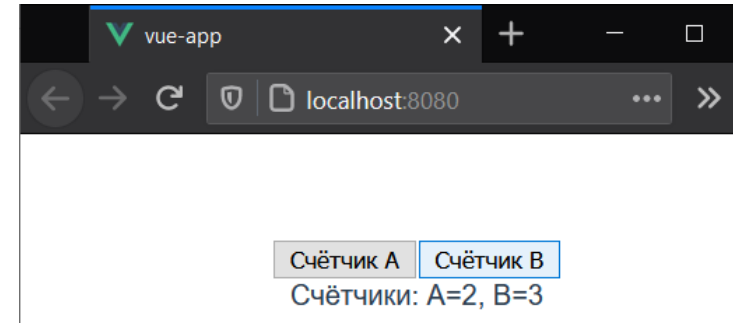
Передача потомку данных в виде <slot>

41

components/HelloWorld.vue

```
<template>
  <div>
    <Button idx="0" @inc-value="increment">Счётчик А</Button>
    <Button idx="1" @inc-value="increment">Счётчик В</Button>
    <div>Счётчики: A={{counter[0]}}, B={{counter[1]}}</div>
  </div>
</template>
<script>
import Button from "@components/Button";
export default {
  name: 'HelloWorld',
  components: {Button},
  data() {
    return {
      counter: [0, 0]
    }
  },
  methods: {
    increment(index) {
      this.counter[index]++
    }
  }
}
</script>
```

Всё в родительском шаблоне компилируется в области видимости родительского компонента; всё в шаблоне дочернего компилируется в области видимости дочернего компонента.



components/Button.vue

```
<template>
  <button @click="$emit('incValue', idx)">
    <slot></slot>
  </button>
</template>
<script>
export default {
  name: "Button",
  props: ["idx"]
}
</script>
```

Свойства с указанием типа

```
props: {
  idx: Number
  // Альтернативы:
  // String, Boolean, Array, Object, Function, Promise
}
```

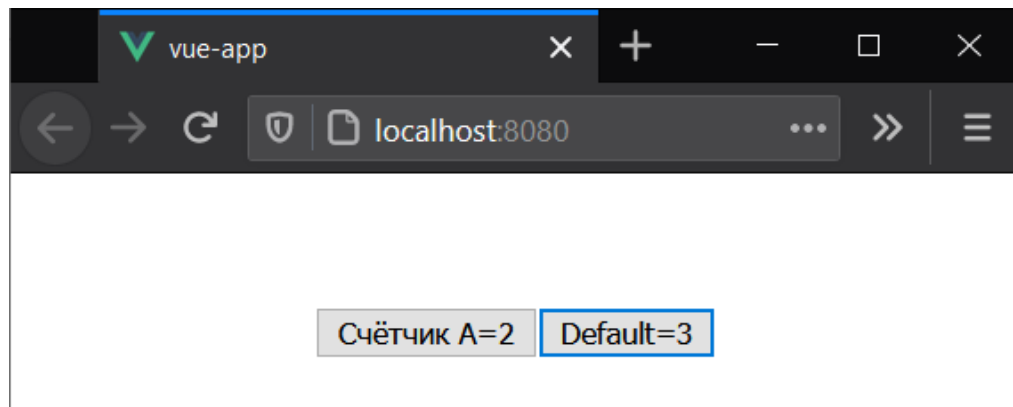
Именованные слоты / v-slot, значение по умолчанию

components/HelloWorld.vue

```
<template>
  <div>
    <Button idx="0" @inc-value="increment">
      <template v-slot:x>Счётчик A</template>
      <template v-slot:y>{{counter[0]}}</template>
    </Button>
    <Button idx="1" @inc-value="increment">
      <template v-slot:y>{{counter[1]}}</template>
    </Button>
  </div>
</template>
<script>
import Button from "@components/Button";
export default {
  name: 'HelloWorld',
  components: {Button},
  data() {
    return { counter: [0, 0] }
  },
  methods: {
    increment(index) { this.counter[index]++ }
  }
}
</script>
```

components/Button.vue

```
<template>
  <button @click="$emit('incValue', idx)">
    <slot name="x">Default</slot>
    <slot name="y"></slot>
  </button>
</template>
<script>
export default {
  name: "Button",
  props: ["idx"]
}
</script>
```



Использование v-model на компонентах

components/HelloWorld.vue

```
<template>
  <div>
    <!-- Запись с v-model идентична записи с :value/@input -->
    <custom-input v-model="textA"/>
    <custom-input :value="textB" @input="textB = $event.target.value"/>
    <div>{{textA}} {{textB}}</div>
  </div>
</template>
```

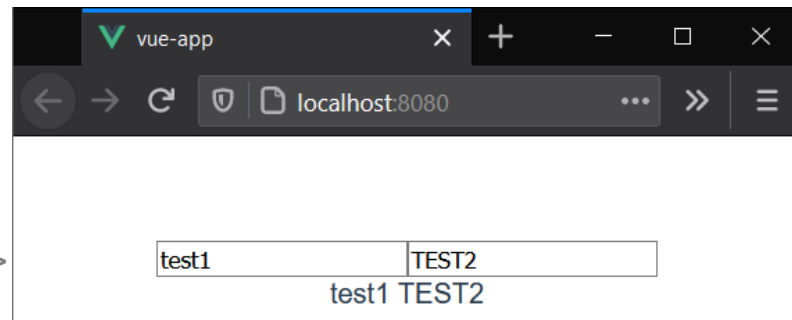
```
<script>
import CustomInput from "@components/CustomInput";
export default {
```

```
  name: 'HelloWorld',
  components: {CustomInput},
  data() {
    return {
      textA: "",
      textB: "",
    }
  }
}</script>
```

components/CustomInput.vue

```
<template>
  <input
    :value="modelValue"
    @input="$emit('update:modelValue', $event.target.value)"
  />
</template>
<script>
export default {
  name: "CustomInput",
  props: ['modelValue'],
  emits: ['update:modelValue']
}
</script>
```

modelValue – название
свойства от **v-model**



Передача статических и динамических данных

<!-- Статический параметр -->

```
<blog-post title="Как изучить Vue"></blog-post>
```

<!-- Динамически присваиваем значение: в кавычках - JavaScript -->

```
<blog-post :title="post.title"></blog-post>
```

<!-- Передача чисел только динамическая, иначе - строка -->

```
<blog-post :likes="42"></blog-post>
```

```
<blog-post :likes="post.likes"></blog-post>
```

<!-- Указание без значения будет означать `true`. -->

<!-- Но во входных - is-published: Boolean, иначе - строка -->

```
<blog-post is-published></blog-post>
```

```
<blog-post :is-published="false"></blog-post>
```

<!-- Передача массива -->

```
<blog-post :comment-ids="[234, 266, 273]"></blog-post>
```

```
<blog-post :comment-ids="post.commentIds"></blog-post>
```

<!-- Передача объекта -->

```
<blog-post
```

```
  :author="{ name: 'Veronica', company: 'Veridian Dynamics' }"
```

```
></blog-post>
```

```
<blog-post :author="post.author"></blog-post>
```

Важно!

Объекты и массивы передаются по ссылке, поэтому изменения в дочернем объекте **повлияют** на значение в родительском!

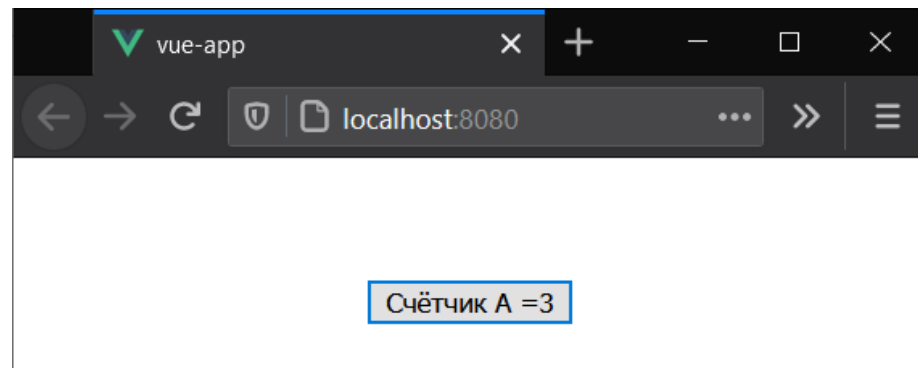
Передача данных с использованием provide & inject

components/HelloWorld.vue

```
<template>
  <div>
    <Button @inc-value="increment">
      <template v-slot:x>Счётчик A</template>
    </Button>
  </div>
</template>
<script>
import Button from "@components/Button";
import {computed} from "vue";
export default {
  name: 'HelloWorld',
  components: {Button},
  data() { return { counter: 0 } },
  provide() { // Предоставление данных потомкам
    return { // computed() - изменяемые данные
      myCounter: computed(() => this.counter)
    }
  },
  methods: { increment () { this.counter++ } }
}
</script>
```

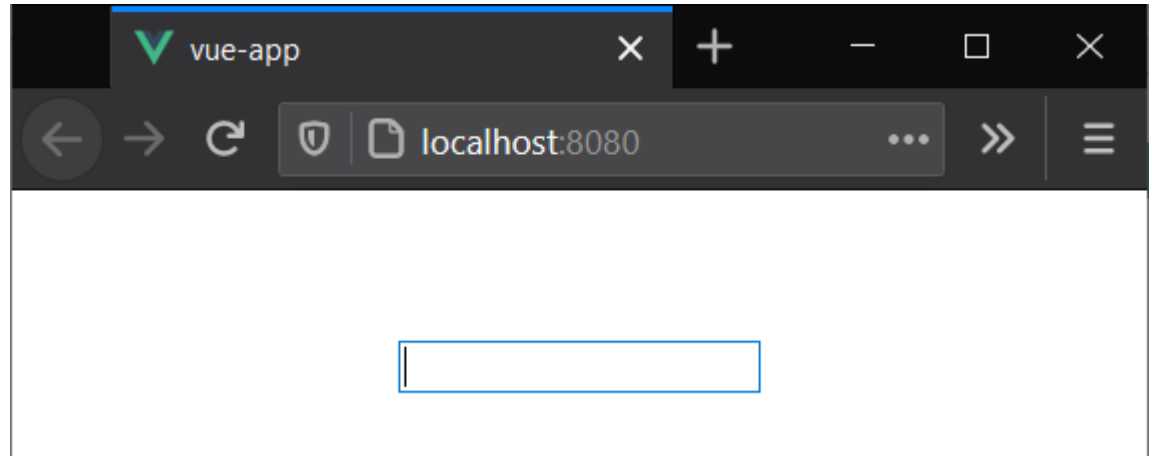
components/Button.vue

```
<template>
  <button @click="$emit('incValue', idx)">
    <slot name="x">Default</slot>
    ={{myCounter}}
  </button>
</template>
<script>
export default {
  name: "Button",
  inject: ["myCounter"]
}
</script>
```



Прямой доступ к компоненту / \$refs ⁴⁶

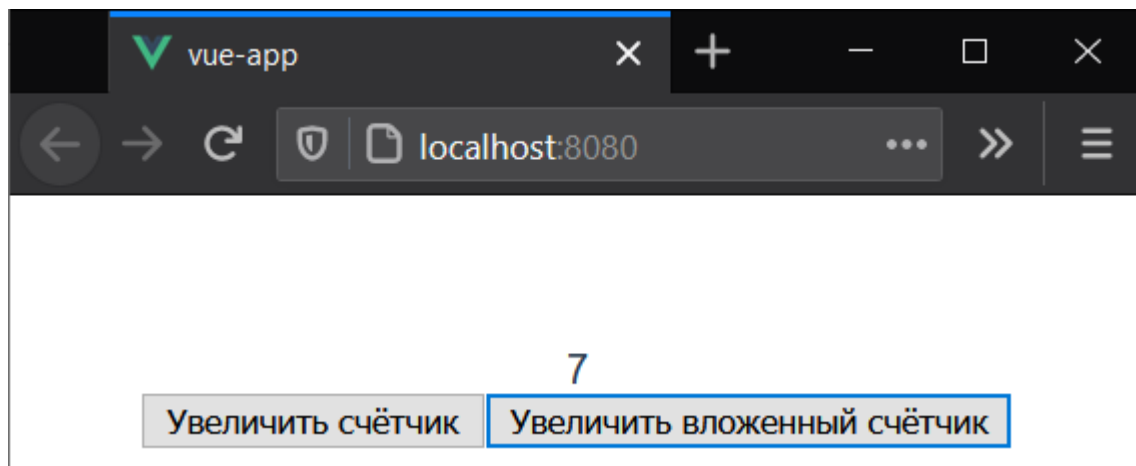
```
<template>
  <div>
    <input ref="myInput" />
  </div>
</template>
<script>
export default {
  name: 'HelloWorld',
  mounted() {
    this.focusInput()
  },
  methods: {
    focusInput() {
      // Ссылки в $refs регистрируются после отрисовки компонента
      this.$refs.myInput.focus()
    }
  }
}
</script>
```



Реактивность – автономная ссылка на примитивное значение ⁴⁷

```
<template>
  <div>
    <span>{{ count }}</span><br>
    <button @click="count++">Увеличить счётчик</button>
    <button @click="nested.count.value++">Увеличить вложенный счётчик</button>
  </div>
</template>
```

```
<script>
import { ref } from 'vue'
export default {
  setup() {
    const count = ref(0)
    return {
      count,
      nested: {
        count
      }
    }
  }
}
</script>
```



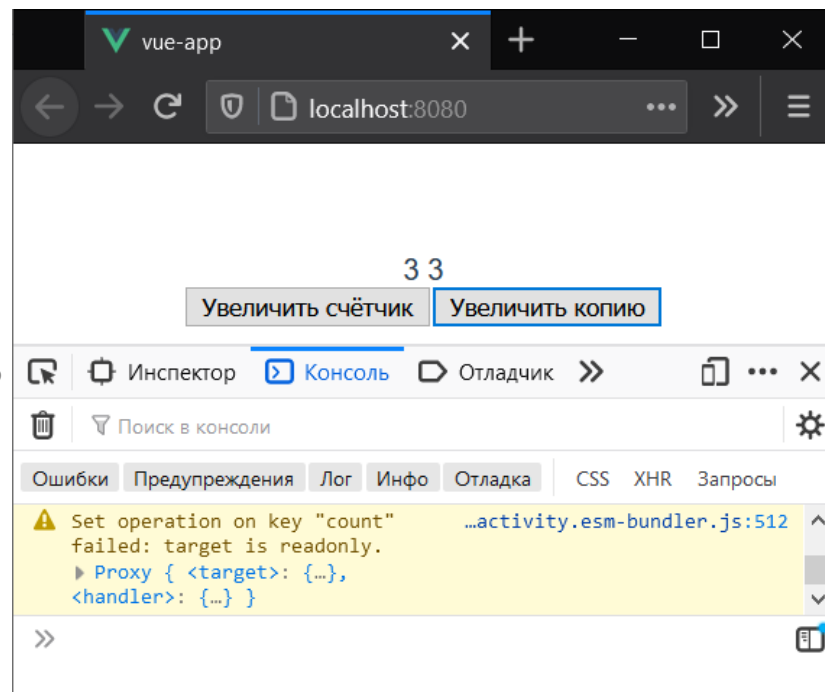
ref вернёт реактивный объект, который можно изменять и который служит реактивной ссылкой для внутреннего значения.

Реактивность – не изменяемые значения

```

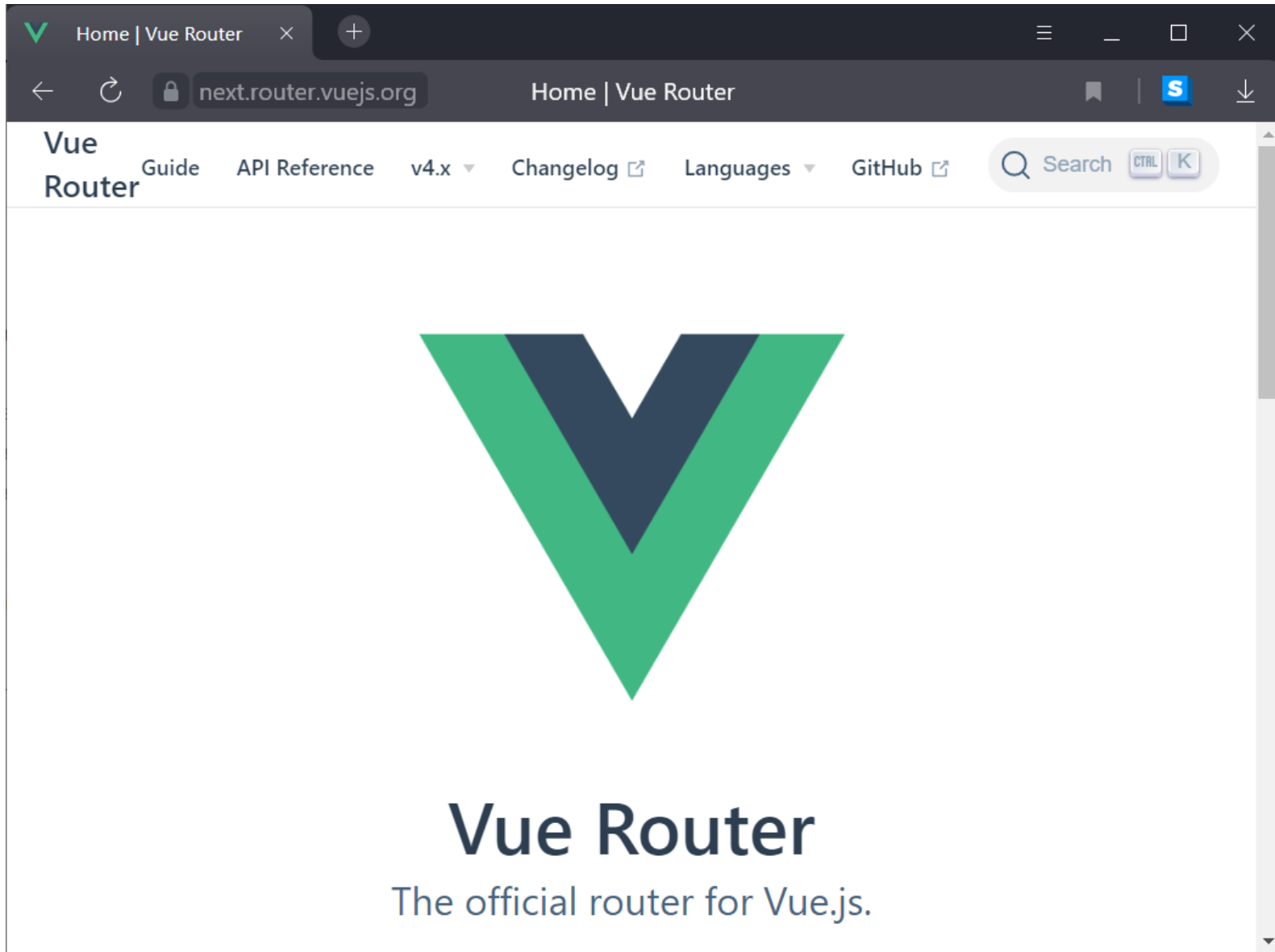
<template>
  <div>
    <span>{{ state.count }} {{ copy.count }}</span><br>
    <button @click="state.count++">Увеличить счётчик</button>
    <button @click="copy.count++">Увеличить копию</button>
  </div>
</template>
<script>
import {reactive, readonly} from 'vue'
export default {
  setup() {
    const state = reactive({count: 1})
    // Значение копии будет напрямую не изменить
    const copy = readonly(state)
    return {
      state,
      copy
    }
  }
}
</script>

```



Маршрутизация Vue Router

49



<https://next.router.vuejs.org/>

Пример простого приложения с маршрутизацией / main.js, App.vue

main.js

```
import { createApp } from 'vue'  
import App from './App.vue'  
import router from './router'  
import store from './store'
```

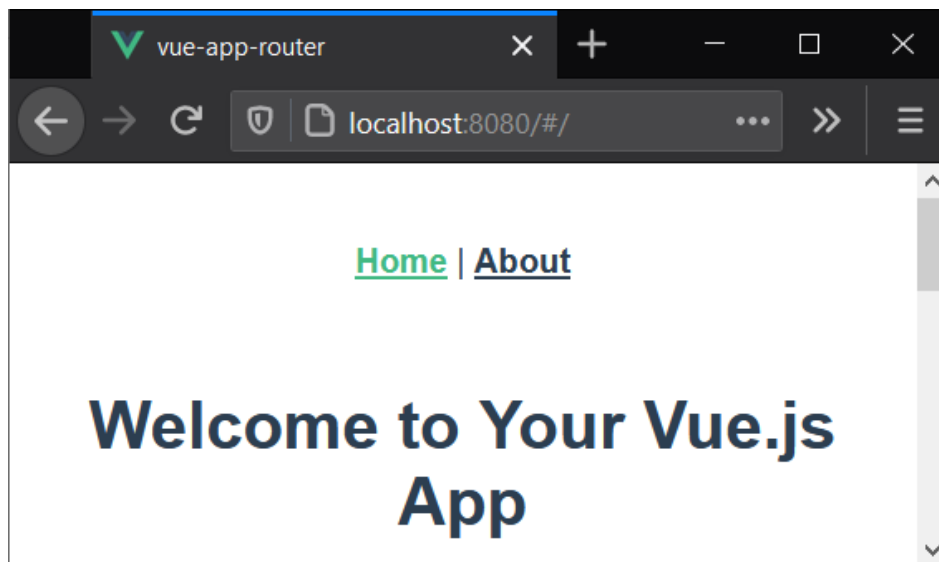
```
createApp(App)  
  .use(store)  
  .use(router)  
  .mount('#app')
```

Я создал сразу с **VUEX**, пока игнорируем **store**

App.vue

```
<template>  
  <div id="nav">  
    <router-link to="/">Home</router-link> |  
    <router-link to="/about">About</router-link>  
  </div>  
  <router-view/>  
</template>
```

<style> не приведены



Пример простого приложения с маршрутизацией / Home.vue, router.js

Home.vue

```
<template>
  <div class="home">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>
<script>
import HelloWorld from '@components/HelloWorld.vue'
export default {
  name: 'Home',
  components: {
    HelloWorld
  }
}
</script>
```

Код HelloWorld.vue не привожу,
там ничего интересного

router.js

```
import { createRouter, createWebHashHistory } from 'vue-router'
import Home from '../views/Home.vue'
const routes = [
  { path: '/', name: 'Home', component: Home },
  { path: '/about', name: 'About',
    // Создаст компонент, lazy-подгружаемый при первом обращении
    // about.[hash].js
    component: () => import(/* webpackChunkName: "about" */ '../views/About.vue')
  }
]
const router = createRouter({
  history: createWebHashHistory(),
  routes
})
export default router
```

Передача параметров в маршруте / router.js, About.vue

router.js

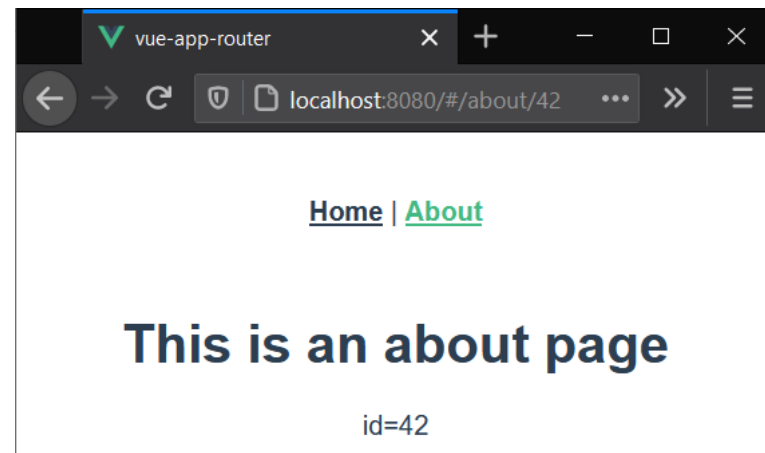
```
import { createRouter, createWebHashHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About'
const routes = [
  { path: '/', name: 'Home', component: Home },
  // динамический параметр :id
  { path: '/about/:id', name: 'About', component: About }
]
const router = createRouter({
  history: createWebHashHistory(),
  routes
})
export default router
```

About.vue

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    id={{ $route.params.id }}
  </div>
</template>
```

Маршрут в App.vue

```
<router-link to="/about/42">About</router-link>
```



Параметры \$route

- **fullPath** – полный URL
- **hash**
- **query** – перечень параметров запроса
- **name** – имя маршрута
- **params** – параметры URL
- **path** – путь

About.vue

```
<template>  
  <div class="about">  
    <h1>This is an about page</h1>  
    <div>params={{ $route.params }}</div>  
    <div>query={{ $route.query }}</div>  
    <div>path={{ $route.path }}</div>  
    <div>id={{ $route.params.id }}</div>  
  </div>  
</template>
```



http://localhost:8080/#/about/43?name=TTT

https://next.router.vuejs.org/api/#routelocationnormalized

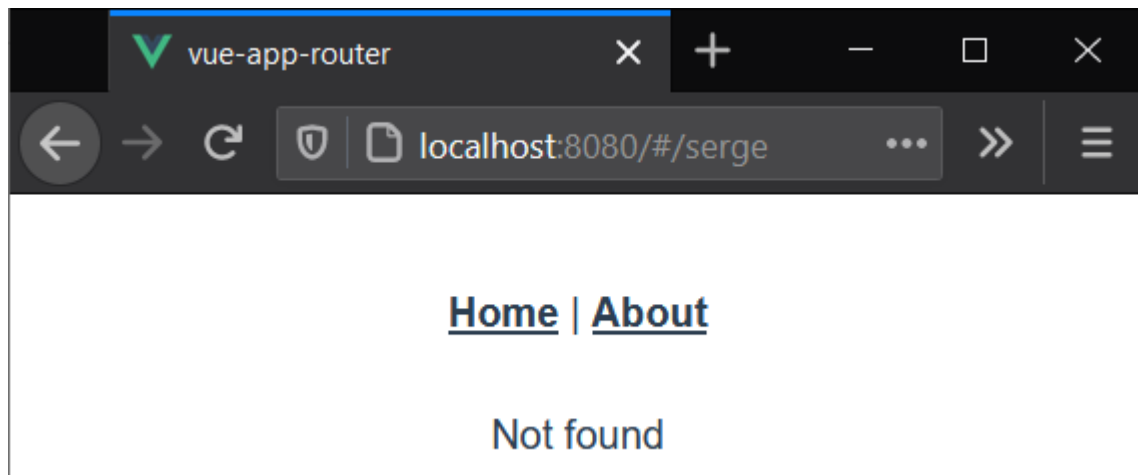
Путь не найден

router.js

```
import { createRouter, createWebHashHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About'
import NotFound from '../views/NotFound'
const routes = [
  { path: '/', name: 'Home', component: Home },
  // динамический параметр :id
  { path: '/about/:id', name: 'About', component: About },
  { path: '/*', name: 'NotFound', component: NotFound }
]
const router = createRouter({
  history: createWebHashHistory(),
  routes
})
export default router
```

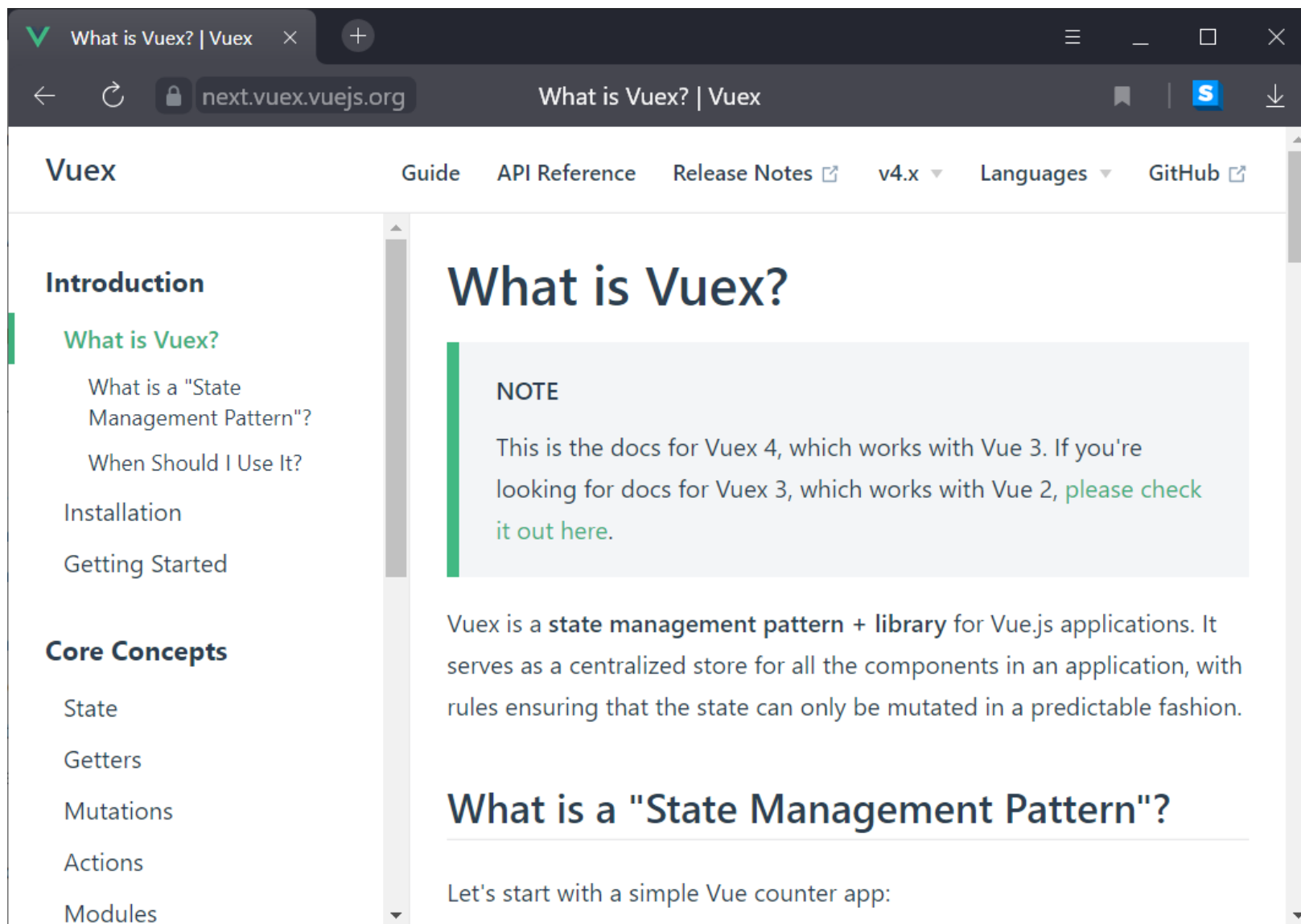
NotFound.vue

```
<template>
  <div>Not found</div>
</template>
```



http://localhost:8080/#/serge

Vuex – библиотека управления состоянием



<https://next.vuex.vuejs.org/>

Пример общего состояния / \$store

56

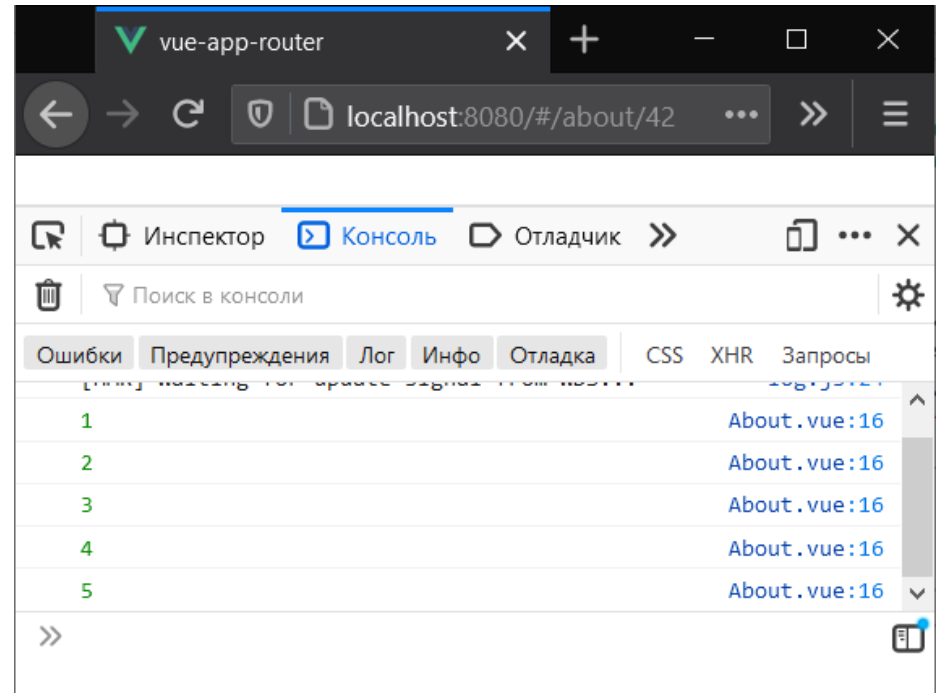
store.js

```
import { createStore } from 'vuex'

export default createStore({
  state () {
    return { // Общий счетчик
      count: 0
    }
  },
  mutations: {
    increment (state) {
      // Изменение состояния
      state.count++
    }
  }
})
```

Пример использования в компоненте

```
methods: {
  increment() {
    this.$store.commit('increment')
    console.log(this.$store.state.count)
  }
}
```



Vuex. State – состояние

```
<template>
  <div>
    <h3>VUEX</h3>
    <button @click="increment">Inc</button>
    <div>{{count}}</div>
  </div>
</template>
<script>
export default {
  methods: {
    increment() { // Увеличение значения
      this.$store.commit('increment')
    }
  },
  computed: {
    count() { // Использование значения
      return this.$store.state.count
    }
  }
}
</script>
```

Альтернатива 1:

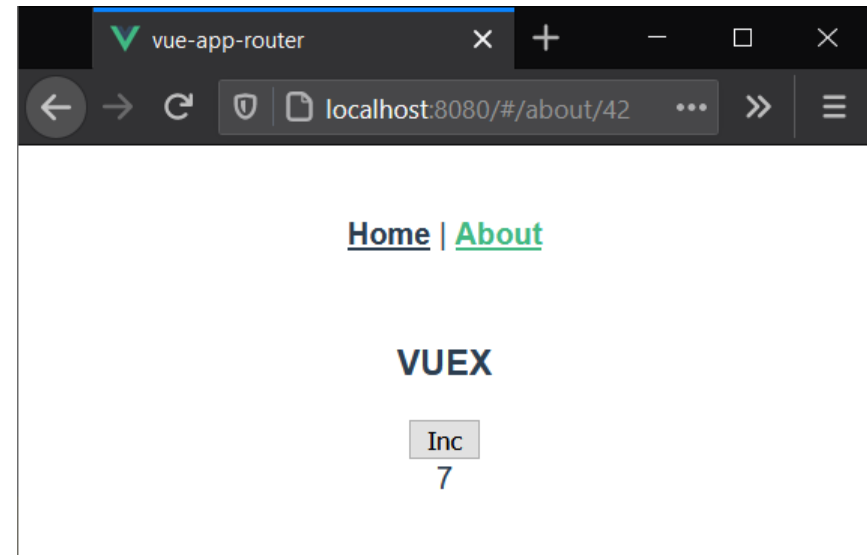
computed: *mapState*(["count"])

Альтернатива 2:

```
computed: mapState({
  count: state => state.count,
  countAlias: 'count',
  countPlusLocalState (state) {
    return state.count + this.localCount
  }
})
```

Для альтернатив нужен:

import {*mapState*} from "vuex";



Vuex. Getters – вычисление значения

58

store.js

```
import { createStore } from 'vuex'

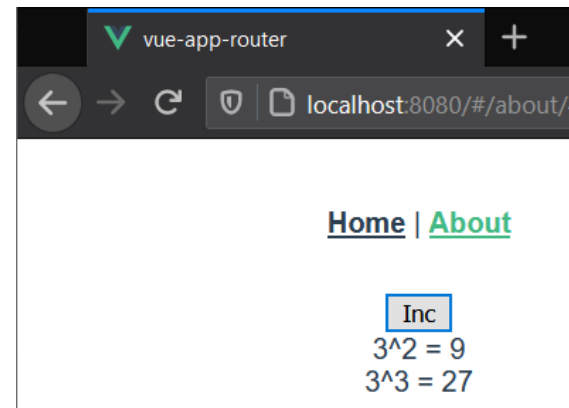
export default createStore({
  state () {
    return { // Общий счетчик
      count: 0
    }
  },
  mutations: {
    increment (state) {
      // Изменение состояния
      state.count++
    }
  },
  getters: { // Вычисление значения
    pow2(state, getters) {
      return Math.pow(state.count, 2)
    }, // С параметром
    pow: (state) => (p) => {
      return Math.pow(state.count, p)
    }
  }
})
```

Альтернатива:

```
...mapGetters({ power2: 'pow2' })
```

About.vue

```
<template>
  <div>
    <button @click="increment">Inc</button>
    <div>{{count}}^2 = {{power2}}</div>
    <div>{{count}}^3 = {{power3}}</div>
  </div>
</template>
<script>
import { mapState } from "vuex";
export default {
  methods: {
    increment() { // Увеличение значения
      this.$store.commit('increment')
    }
  },
  computed: {
    power2() { // Вычисляемое свойство
      return this.$store.getters.pow2
    },
    power3() {
      return this.$store.getters.pow(3)
    }, // Подключение count
    ...mapState(['count'])
  }
}
</script>
```



Vuex. Mutations – изменение

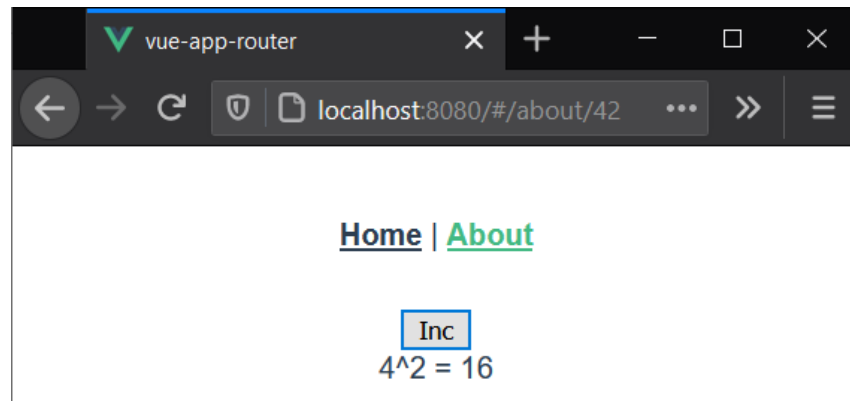
store.js

```
import { createStore } from 'vuex'

export default createStore({
  state () {
    return { // Общий счетчик
      count: 0
    }
  },
  mutations: {
    increment (state, payload) {
      // Изменение состояния
      state.count += payload.amount
    }
  },
  getters: { // Вычисление значения
    pow2(state, getters) {
      return Math.pow(state.count, 2)
    }
  }
})
```

About.vue

```
<template>
  <div>
    <button @click="increment">Inc</button>
    <div>{{count}}^2 = {{power2}}</div>
  </div>
</template>
<script>
import { mapState, mapGetters } from "vuex";
export default {
  methods: {
    increment() { // Увеличение значения
      this.$store.commit('increment', {amount: 2})
    }
  },
  computed: {
    ...mapState(["count"]),
    ...mapGetters({ power2: 'pow2' })
  }
}
</script>
```



Vuex. Actions – асинхронные действия

600

store.js

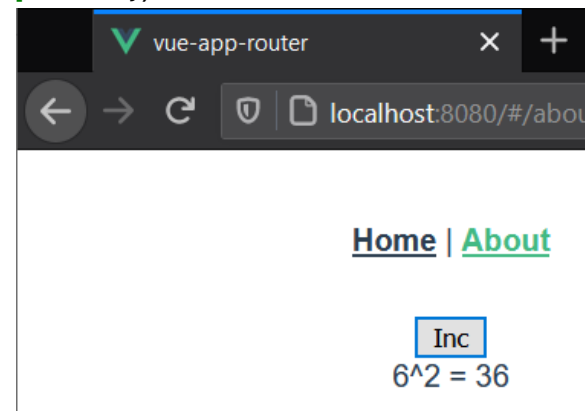
```
import { createStore } from 'vuex'

export default createStore({
  state () {
    return { // Общий счетчик
      count: 0
    }
  },
  mutations: {
    increment (state, payload) {
      state.count += payload.amount
    }
  },
  getters: { // Вычисление значения
    pow2 (state, getters) {
      return Math.pow(state.count, 2)
    }
  },
  actions: {
    // Поддержка асинхронных операций
    increment (context, payload) {
      context.commit('increment', payload)
    }
  }
})
```

About.vue

```
<template>
  <div>
    <button @click="increment">Inc</button>
    <div>{{count}}^2 = {{power2}}</div>
  </div>
</template>
<script>
import { mapState, mapGetters } from "vuex";
export default {
  methods: {
    increment() { // Увеличение значения
      this.$store.dispatch('increment', {amount: 3})
    }
  },
  computed: {
    ...mapState(["count"]),
    ...mapGetters({ power2: 'pow2' })
  }
}
</script>
```

Есть поддержка **mapActions**



Vuex. Modules – деление на модули

61

```
const moduleA = {  
  state: () => ({ ... }),  
  mutations: { ... },  
  actions: { ... },  
  getters: { ... }  
}
```

```
const moduleB = {  
  state: () => ({ ... }),  
  mutations: { ... },  
  actions: { ... }  
}
```

```
const store = createStore({  
  modules: {  
    a: moduleA,  
    b: moduleB  
  }  
})
```

`store.state.a` // -> ``moduleA`'s state`

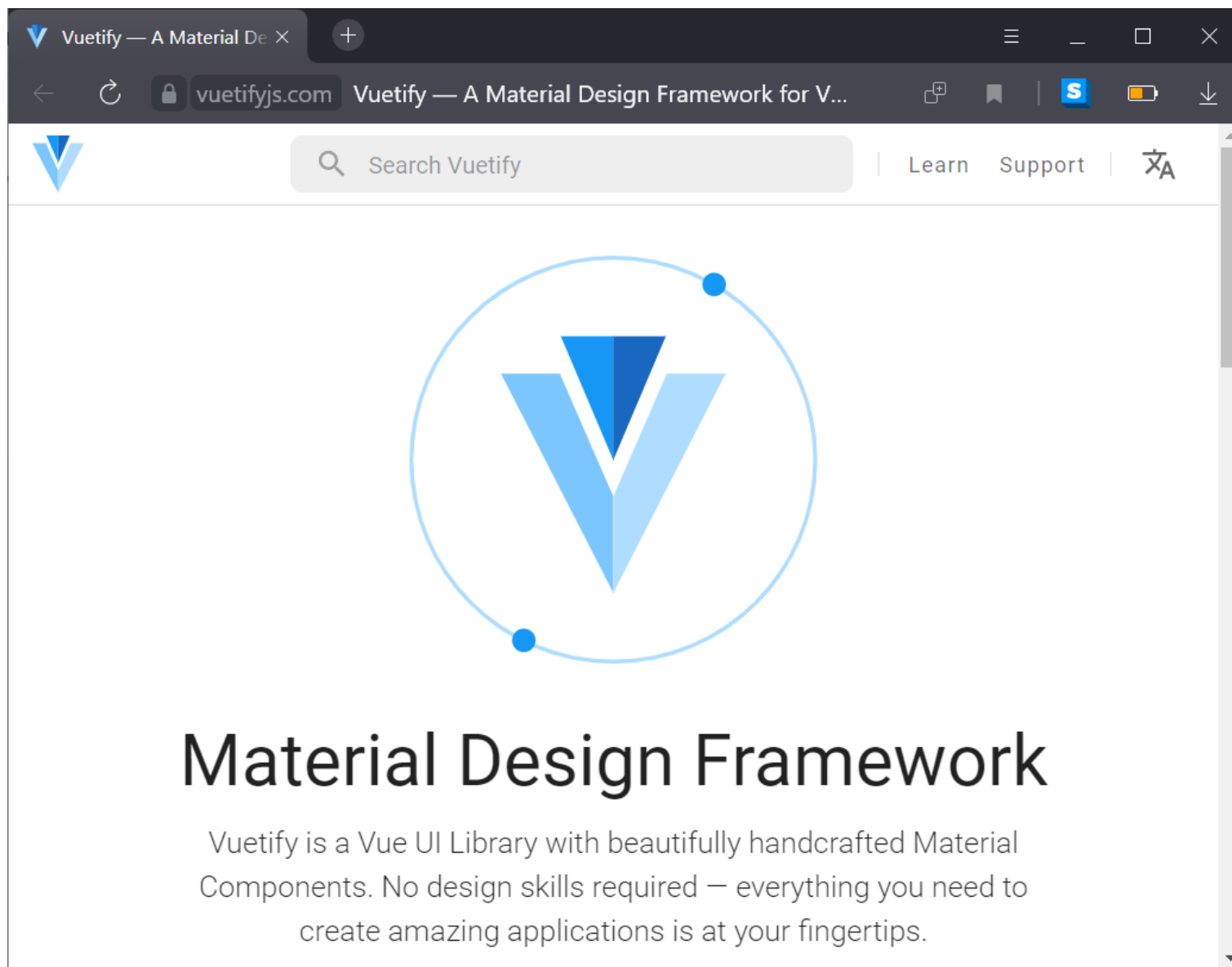
`store.state.b` // -> ``moduleB`'s state`

С поддержкой

- глобальных действий
- пространств имён
- динамической регистрацией модулей
- повторного использования модулей

Vuetify – компоненты для Vue

62

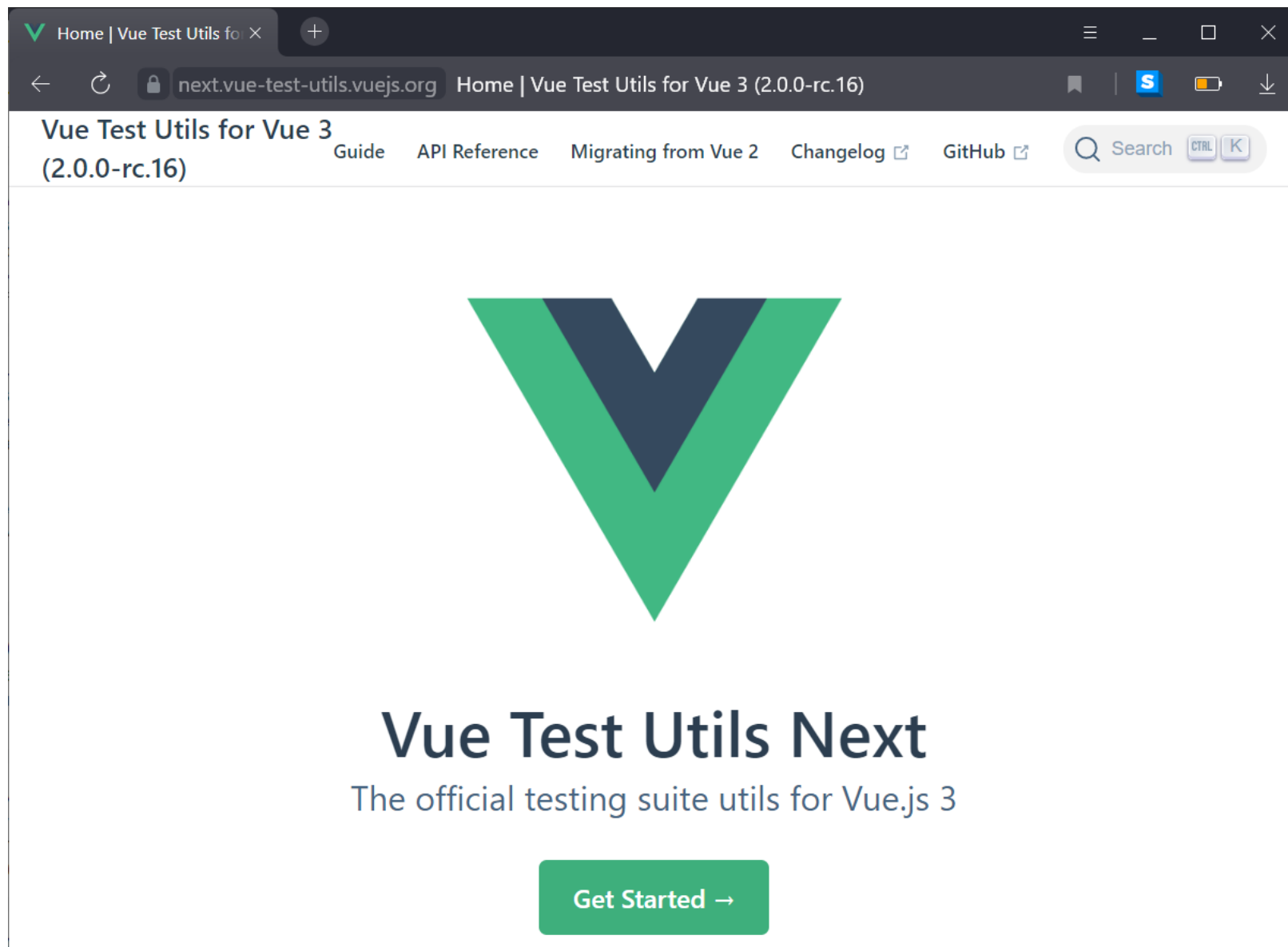


<https://vuetifyjs.com/en/>

v.3 (Titan)

Vue Test Utils – тестирование Vue

63



<https://next.vue-test-utils.vuejs.org/>

Вопросы для самопроверки

- Из каких частей состоит компонент Vue?
- В чём особенность настройки стилей для компонент Vue?
- Чем отличаются директивы `v-bind` и `v-model`? Есть короткая запись?
- Для чего используется `v-on`? Какая у него короткая запись?
- Без чего не будет работать `v-for`?
- Как создать и примонтировать приложение Vue?
- Какие хуки Vue вы знаете? Как их использовать?
- Что такое интерполяция в Vue?
- Для чего используется `data()`? Какой у неё синтаксис?
- Чем отличаются `methods` от `computed`? А при вызове в шаблоне?
- Что такое наблюдатели `watch`?
- Как работать с CSS (классами, `inline`-стилями)?
- Что такое модификаторы? (событий, директив)
- Как подключаются компоненты? Как в них передаются данные? Как данные получаются "обратно"? Что такое слоты?
- Как можно получить "прямой" доступ к компоненту?
- Как создать маршруты? Прочитать параметры? Что такое клиентская маршрутизация?
- Как работать с общим состоянием Vue-приложения? В чем отличие `getters`, `mutations` и `actions`?