

Fullstack-разработка

Vue

Введение

Цели урока:

1. Создадим проект на Vue
2. Используем 3 кита Javascripta
внутри компонента

Теория

Vue

Vue – это **фреймворк** для создания пользовательских интерфейсов



- Простой
- Ускоряет разработку
- Берёт работу с DOM на себя

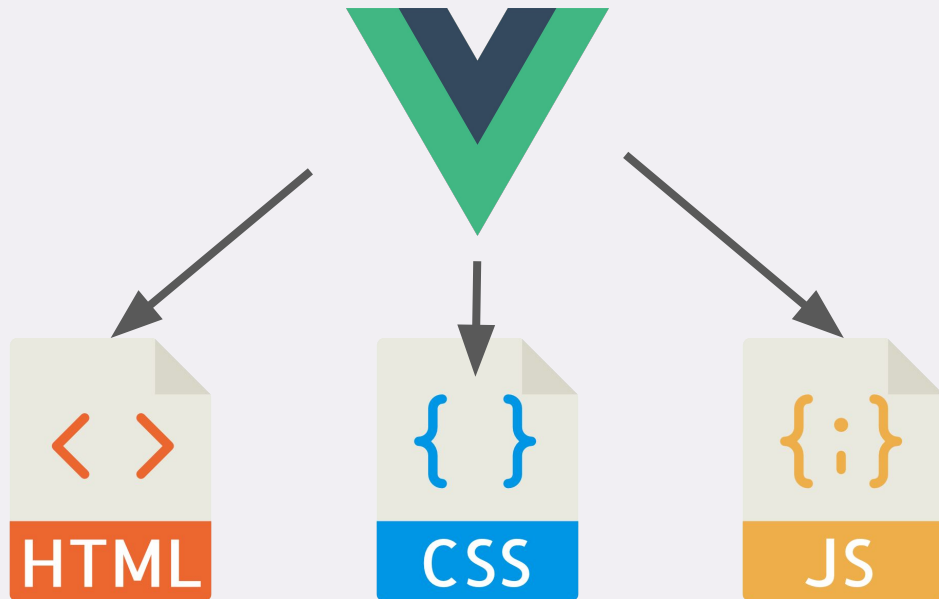
Фреймворк предлагает определённый способ писать код, что облегчает и ускоряет разработку.

Это как купить путёвку вместо планирования путешествия самостоятельно.

В основе Vue лежат
компоненты – элементы
интерфейса, которые
можно **переиспользовать**.

Navbar Home Features Pricing About

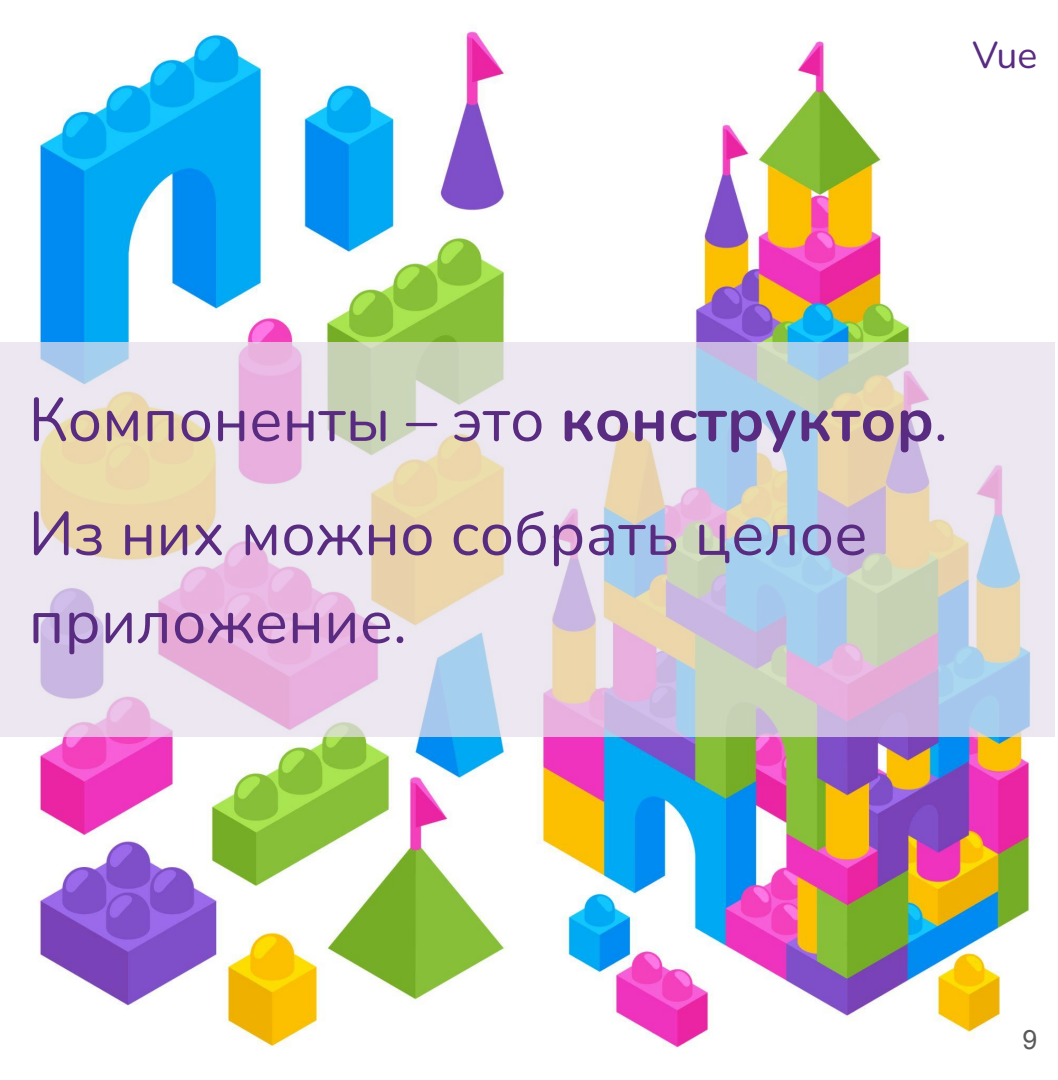
Каждый компонент
хранится в отдельном файле
с расширением .vue



HelloWorld.vue

Каждый компонент состоит из трёх секций-тегов:

```
<template>  
  <h1>Привет, мир!</h1>  
</template>  
  
<style>  
h1 { font-size: 40px; }  
</style>  
  
<script>  
...  
</script>
```

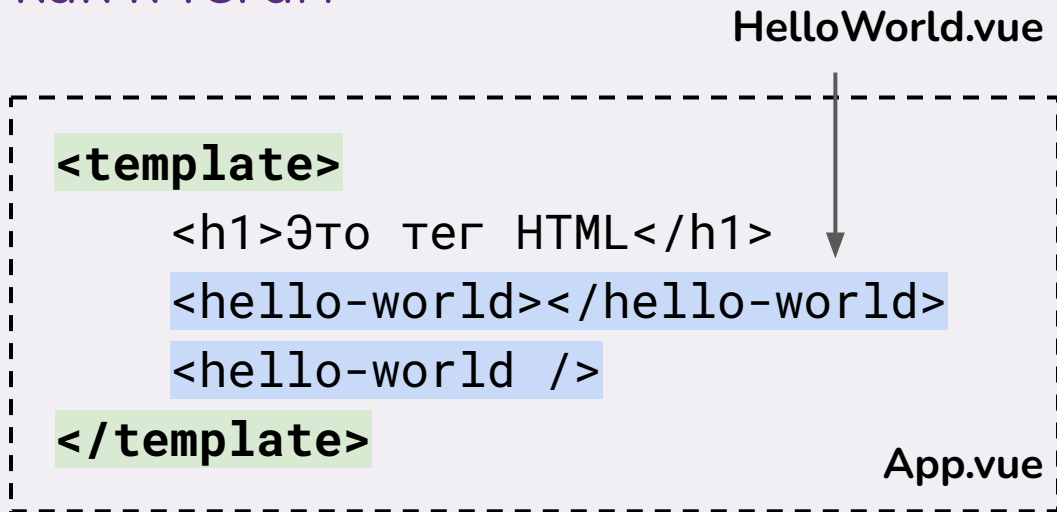
The background of the slide features a collection of colorful, isometric geometric shapes, including cubes, cylinders, and pyramids in shades of blue, green, yellow, pink, and purple. On the right side, these shapes are assembled into a complex, multi-tiered castle structure with multiple towers and flags. The text is overlaid on a semi-transparent light purple rectangular area in the center of the slide.

Компоненты – это **конструктор**.
Из них можно собрать целое
приложение.

Компонент **App.vue** –
как платформа для
конструктора компонентов

App.vue

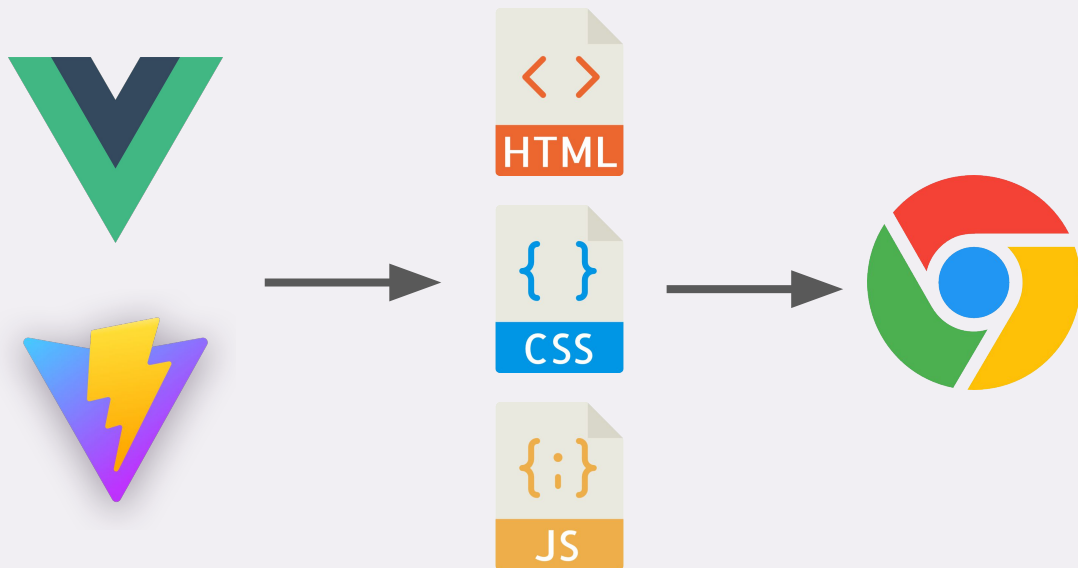
В него можно подключить другие компоненты и обращаться к ним как к тегам



Когда пользователь откроет
наш сайт, он увидит именно
App.vue

У Vue свой уникальный синтаксис,
который не понятен браузеру.

Поэтому нам нужен **сборщик Vite**.

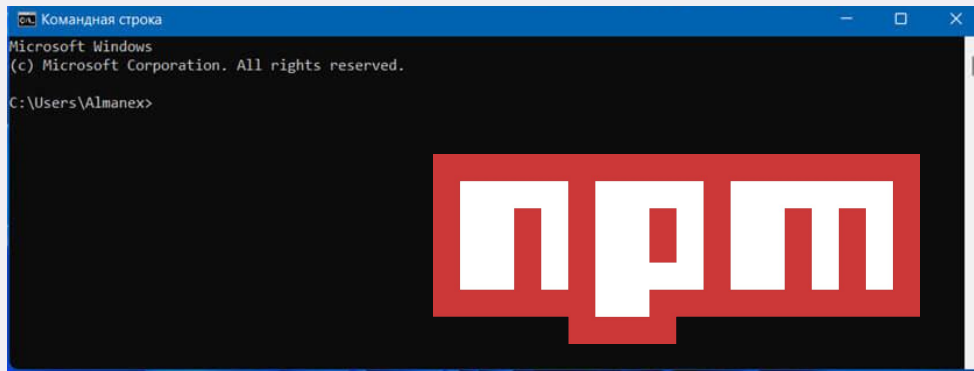


Задача сборщика – перевести компоненты Vue на понятный браузеру HTML, CSS и Javascript.

Он также запускает локальный сервер, который делает файлы доступными на localhost.

Vite и Vue это библиотеки.

Так что нам понадобится...



Теория и Практика

Создание проекта

Создание проекта

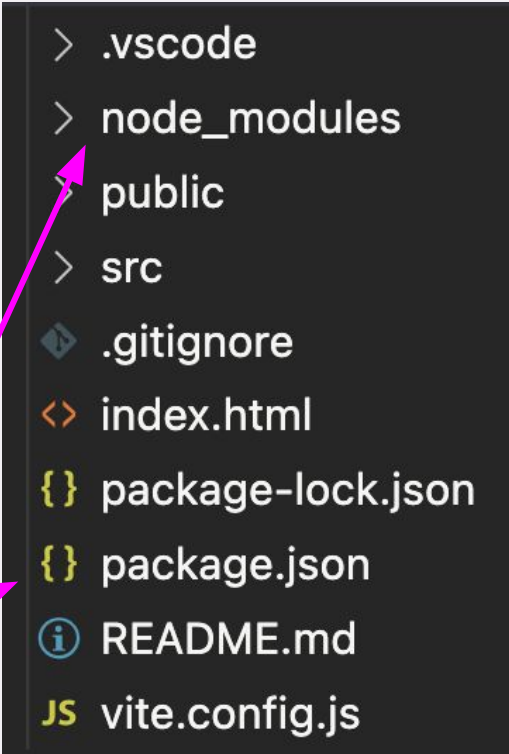
В папке со всеми проектами выполни команду:

```
npm create vue@3 название
```



Открой проект папку проекта в VSCode.
Установи пакеты:

```
npm install
```

A dark-themed file explorer view showing the contents of a project directory. The files and folders are listed with their respective icons: a folder icon for .vscode, node_modules, public, and src; a gitignore icon for .gitignore; an HTML icon for index.html; a JSON icon for package-lock.json and package.json; an info icon for README.md; and a JS icon for vite.config.js. Two pink arrows originate from the 'npm install' command box and point to the 'node_modules' and 'package.json' entries in the file explorer.

```
> .vscode  
> node_modules  
> public  
> src  
📄 .gitignore  
<> index.html  
{ } package-lock.json  
{ } package.json  
📄 README.md  
JS vite.config.js
```



Файлы проекта

The image shows a file explorer with the following structure:

- > .vscode
- > public
- ▼ src
 - ▼ assets
 - logo.svg
 - ▼ components
 - ▼ HelloWorld.vue
 - ▼ TheWelcome.vue
 - ▼ WelcomItem.vue
 - ▼ App.vue
 - JS main.js
- ◆ .gitignore
- <> index.html
- { } package.json
- i README.md
- JS vite.config.js

Annotations with arrows pointing to specific files:

- Изображения (Images) points to `assets`.
- Компоненты (с расширением *.vue) (Components (with *.vue extension)) points to `components`.
- То, что видит пользователь (What the user sees) points to `App.vue`.
- Здесь код, который подставляет **App.vue** в **index.html** (Here is the code that substitutes **App.vue** in **index.html**) points to `main.js`.
- Обычный index.html (Ordinary index.html) points to `index.html`.

Монтирование приложения

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

```
<body>  
  <div id="app">  
    <!-- Здесь будет App.vue -->  
  </div>  
  
  <script type="module" src="/src/main.js">  
  </script>  
</body>
```

main.js

index.html

Теория и Практика

Шапка сайта и подвал

Подключение компонента

Импорт

```
<script>
  import ComponentOne from './components/ComponentOne.vue';
  import ComponentTwo from './components/ComponentTwo.vue';

  export default {
    components: {
      ComponentOne,
      ComponentTwo
    }
  }
</script>
```

Использование

```
<template>
  <component-one> </component-one>
  <component-two />
</template>
```

Название компонентов

1. Состоит из двух слов
2. Записано в CamelCase
3. Не может повторять HTML-теги

Теория

Обработчики события

3 кита JavaScript

1. **Найти элемент**
`document.querySelector`
2. **Изменить элемент**
`node.innerHTML`
3. **Реагировать на событие Клик**
`button.addEventListener`

Больше мы этого не увидим)

```
1 console.log("Hello, world!");  
2  
3 let outputNode = document.querySelector(`#output`);  
4 let buttonNode = document.querySelector(`#click`);  
5  
6 buttonNode.addEventListener(`click`, function () {  
7   outputNode.innerHTML += `🔥`;  
8 });  
9
```

Жми огонёк



FireButton.vue

Создадим новый компонент с вёрсткой кнопки.

```
<template>  
  <button class="btn btn-primary">  
    Жми огонёк  
  </button>  
</template>
```

Событие

События в Vue это директива – специальный **атрибут**, который можно добавить любому тегу

@событие

- @click
- @input
- @keydown

Метод

Методы компонента описывается
в объекте **methods**.

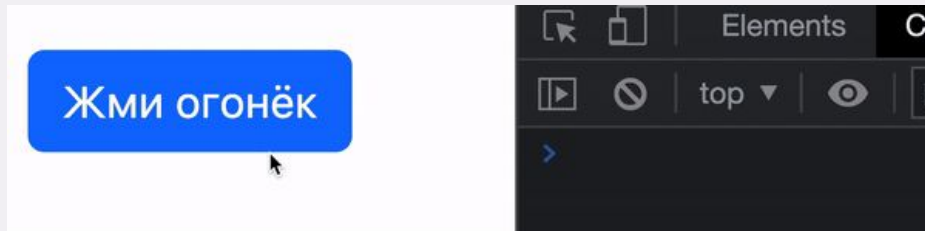
```
<script>
export default {
  methods: {
    fire() {
      console.log('огонёк!');
    },
    hello() {
      console.log('привет');
    }
  }
}
</script>
```

Подписка на событие

Событие и метод легко соединяются
внутри шаблона компонента.

```
<template>  
  <button @click="fire"  
    class="btn btn-primary">  
    Жми огонёк  
  </button>  
</template>
```

Через директиву **@click** и метод компонента устанавливается связь между событием и действием.



data

Кроме методов у компонента есть **данные**.
Это **объект** со свойствами для шаблона.

```
<script>
export default {
  data() {
    return {
      username: 'Никита',
      age: 15
    }
  }
}
</script>
```

data

Данные можно выводить в шаблон через наши любимые **фигурные скобки**.

```
<template>
```

```
<h1>
```

```
    Меня зовут {{ username }},
```

```
    мне {{ age }} лет.
```

```
</h1>
```

```
</template>
```

FireButton.vue

Допустим, у кнопки есть **data-свойство** `text`.

```
<template>  
  <button class="btn btn-primary">  
    Жми огонёк  
  </button>  
  <p> {{ text }} </p>  
</template>
```

data

С помощью метода можно обратиться к свойству через **this** и изменить его.

```
<script>
export default {
  methods: {
    fire() {
      this.text += `🔥`
    }
  }
}
</script>
```

Изменение свойства **data** автоматически приводит к перерисовке компонента.

Это называется **реактивность**.



Жми огонёк

4 кита Vue

1. Объявить data
2. Вывести data в шаблон
`{{ text }}`
3. Изменить data в методе
`fire() { this.text += "🔥"; }`
4. Связать метод и событие
`@click="fire"`

Практика

Вопрос, счётчик и галерея

Цели урока:

- ✓ Создали проект на Vue
- ✓ Использовали 3 кита Javascripta
внутри компонента

Домашнее задание