> ! Эта документация для версий v2.x и ранее. Для v3.x, **документация на русском здесь [https://v3.ru.vuejs.org/]** .

# Экземпляр Vue

## Создание экземпляра Vue [#Создание-экземпляра-Vue]

Каждое приложение начинается с создания нового **экземпляра Vue** с помощью функции `Vue` :

```JS
var vm = new Vue({
  // опции
})
```

Хоть Vue и не реализует **паттерн MVVM [https://ru.wikipedia.org/wiki/Model-View-ViewModel]** в полной мере, архитектура фреймворка им во многом вдохновлена. Поэтому переменную с экземпляром Vue традиционно именуют `vm` (сокращённо от ViewModel).

При создании экземпляра Vue необходимо передать **объект опций**. Большая часть этого руководства посвящена описанию, как можно использовать эти опции для достижения желаемого поведения. Полный список опций можно посмотреть в **справочнике API [../api/#Опции-—-данные]** .

Приложение Vue состоит из **корневого экземпляра Vue**, создаваемого с помощью `new Vue` , опционально организованного в дерево вложенных, повторно используемых компонентов. Например, дерево компонентов для приложения TODO-списка может выглядеть так:

```
Корневой экземпляр
└─ TodoList
   ├─ TodoItem
   │  ├─ TodoButtonDelete
   │  └─ TodoButtonEdit
   └─ TodoListFooter
      ├─ TodosButtonClear
      └─ TodoListStatistics
```

Подробнее о **системе компонентов [components.html]** мы поговорим позднее. А сейчас запомните, что все компоненты Vue также являются экземплярами Vue и поэтому принимают такой же объект опций (за исключением нескольких специфичных для корневого).

## Данные и методы [#Данные-и-методы]

Когда экземпляр Vue создан, он добавляет все свойства, найденные в опции `data` , в **систему реактивности** Vue. Поэтому представление будет «реагировать» на их изменения, обновляясь в соответствии с новыми значениями.

```JS
// Наш объект data
var data = { a: 1 }

// Объект добавляется в экземпляр Vue
var vm = new Vue({
  data: data
})

// Получение свойства из экземпляра
// возвращает то же значение из исходных данных
vm.a === data.a // => true

// Изменение свойства экземпляра
// влияет на оригинальные данные
vm.a = 2
data.a // => 2

// ... и наоборот
data.a = 3
vm.a // => 3
```

Когда значения изменяются, представление будет переотрисовано. Но обратите внимание, свойства в `data` будут **реактивными**, только если они существовали при создании экземпляра. Это значит, если добавить новое свойство, например:

```JS
vm.b = 'hi'
```

То изменения в `b` не будут вызывать никаких обновлений. Если вы знаете, что свойство вам понадобится позже, но изначально оно пустое или несуществующее, нужно просто установить начальное значение. Например:

```JS
data: {
  newTodoText: '',
  visitCount: 0,
  hideCompletedTodos: false,
  todos: [],
  error: null
}
```

Единственным исключением здесь является использование `Object.freeze()`, который предотвращает изменение существующих свойств, что также означает невозможность *отслеживать* изменения системой реактивности.

```JS
var obj = {
  foo: 'bar'
}

Object.freeze(obj)

new Vue({
  el: '#app',
  data: obj
})
```

```HTML
<div id="app">
  <p>{{ foo }}</p>
  <!-- мы теперь не можем обновить `foo`! -->
  <button v-on:click="foo = 'baz'">Изменить</button>
</div>
```

Кроме свойств объекта `data`, экземпляры Vue предоставляют ряд служебных свойств и методов экземпляра. Их имена начинаются с префикса `$`, чтобы отличаться от пользовательских свойств. Например:

```JS
var data = { a: 1 }
var vm = new Vue({
  el: '#example',
  data: data
})

vm.$data === data // => true
vm.$el === document.getElementById('example') // => true

// $watch — это метод экземпляра
vm.$watch('a', function (newValue, oldValue) {
  // Этот коллбэк будет вызван, когда изменится `vm.a`
})
```

С полным списком свойств и методов экземпляра Vue можно ознакомиться в **справочнике API [../api/#Свойства-экземпляра]** .

## Хуки жизненного цикла экземпляра [#Хуки-жизненного-цикла-экземпляра]

Посмотрите бесплатный урок на Vue School

Каждый экземпляр Vue при создании проходит через последовательность шагов инициализации — например, настраивает наблюдение за данными, компилирует шаблон, монтирует экземпляр в DOM, обновляет DOM при изменении данных. Между этими шагами вызываются функции, называемые **хуками жизненного цикла**, с помощью которых можно выполнять свой код на определённых этапах.

Например, хук **`created`** **[../api/#created]** можно использовать для выполнения кода после создания экземпляра:

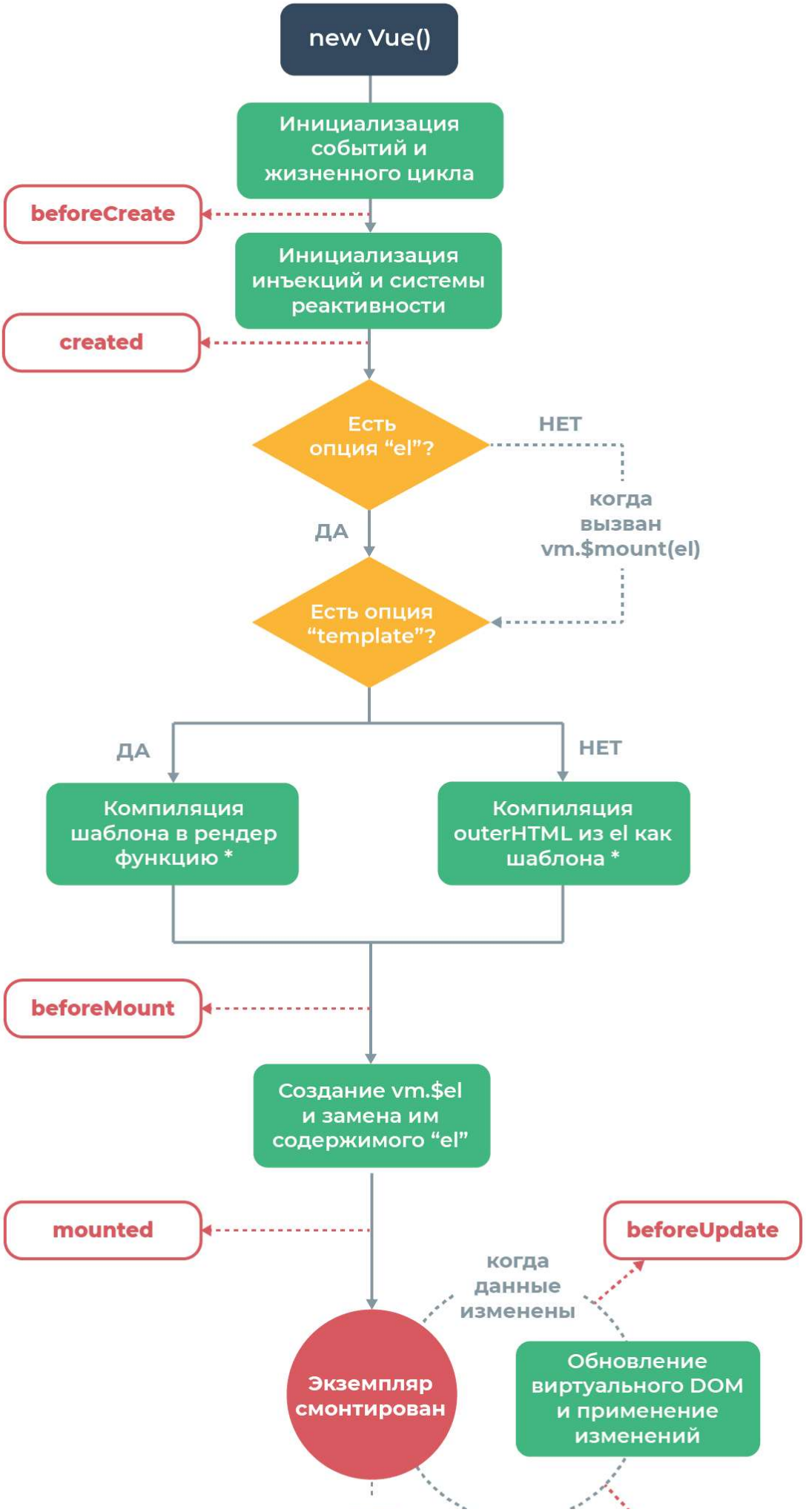
```js
new Vue({
  data: {
    a: 1
  },
  created: function () {
    // `this` указывает на экземпляр vm
    console.log('Значение a: ' + this.a)
  }
})
// => "Значение a: 1"
```

Существуют и другие хуки, вызываемые на различных стадиях жизненного цикла экземпляра, например `mounted` [../api/#mounted], `updated` [../api/#updated] и `destroyed` [../api/#destroyed]. Все хуки вызываются с контекстной переменной `this`, ссылающейся на вызывающий экземпляр Vue.

> ! Не используйте **стрелочные функции** [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions/Arrow_functic в свойствах экземпляра и в коллбэках, например `created: () => console.log(this.a)` или `vm.$watch('a', newVal => this.myMethod())`. Так как стрелочные функции не имеют собственного `this`, то `this` в коде будет обрабатываться как любая другая переменная и её поиск будет производиться в областях видимости выше до тех пор пока не будет найдена, часто приводя к таким ошибкам, как `Uncaught TypeError: Cannot read property of undefined` или `Uncaught TypeError: this.myMethod is not a function`.

## Диаграмма жизненного цикла [#Диаграмма-жизненного-цикла]

Ниже представлена диаграмма жизненного цикла экземпляра. Необязательно понимать её полностью прямо сейчас, но по мере изучения и практики разработки к ней полезно будет обращаться.


new Vue()
Инициализация событий и жизненного цикла
beforeCreate
Инициализация инъекций и системы реактивности
created
Есть опция “el”?
НЕТ
когда вызван vm.$mount(el)
ДА
Есть опция “template”?
ДА
НЕТ
Компиляция шаблона в рендер функцию *
Компиляция outerHTML из el как шаблона *
beforeMount
Создание vm.$el и замена им содержимого “el”
mounted
beforeUpdate
когда данные изменены
Экземпляр смонтирован
Обновление виртуального DOM и применение изменений


* компиляция шаблона выполняется заранее при наличии шага сборки, например при использовании однофайловых компонентов

← Введение [/v2/guide/index.html] Синтаксис шаблонов [/v2/guide/syntax.html] →