Vue. Жизненный цикл компонента. Хуки.

Этапы цикла жизни компонента vue

Инициализация

Монтирование (вставка DOM)

Обновление (Diff и повторный рендеринг)

Уничтожение (разрушение)

Создадим первый компонент

1

Создадим папку занятие и
Подключим vue в шапке сайта

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
```

2

Создадим секцию <script> на странице и создадим объект vue

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

Хуки жизненного цикла

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.

Хуки создания (инициализация)

Хуки создания — это самые первые хуки, которые запускаются в вашем компоненте. Они позволяют выполнять действия даже до добавления компонента в модель DOM. В отличие от других хуков, хуки создания также запускаются при рендеринге на стороне сервера.

Пример beforeCreate

```
beforeCreate() {
    console.log(‘Before.')
}
```

Что можно сделать в beforeCreate?

Хук `beforeCreate` выполняется прямо во время инициализации компонента. Данные ещё не стали реактивными, а события не настроены. Тут можно выполнить запрос к серверу, перед отрисовкой данных.

Реактивные данные - при любом изменении значения в data происходит автоматическое изменение выводимого значения в шаблоне.

Пример created

Хук `created` можно использовать для выполнения кода после создания экземпляра:

```
created() {
    console.log('Created.')
 }
```

Что можно сделать в created

В хуке `created` вы сможете получить доступ к data атрибутам. Шаблоны и виртуальный DOM ещё не встроены (mounted) и не отрисованы.

Монтирование

Хуки монтирования используются чаще всего. Они обеспечивают мгновенный доступ к компоненту до и после первого рендеринг. Однако они не выполняются во время рендеринга на стороне сервера.

beforeMount

Не сработает пока компонента нет в дом дереве!

Можно произвести необходимые операции с данными, пока шаблон не появился в DOM дереве

```
beforeMount(){
    console.log(`beforeMount.`)
  }
```

Что нужно для монтированная компонента?

Для того, чтобы компонент мог появится на верстке, ему нужно добавить 2 опции:

el: '#root',

- идентификатор, существующего в html элемента, для монтированная в него

template: `<div>HELLO VUE</div>`

- Шаблон (содерджимое компонента)

mounted

```
mounted() {
    console.log(`mounted.`);
    console.log(this.$el.textContent)
}
```

mounted

В хуке `mounted` вы получите полный доступ к реактивному компоненту, шаблонам и отрисованному DOM (через `this.$el`). `Mounted` — самый популярный хук жизненного цикла. Обычно его используют для извлечения данных для компонента и изменения DOM, зачастую ради интегрирования не-Vue библиотек.

Обновление

Хуки обновления вызываются, когда изменяется реактивное свойство, используемое вашим компонентом, или когда что-то еще вызывает его повторный рендеринг. Это позволяет использовать хук в цикле *watch-compute-render* для вашего компонента.

beforeUpdate

Хук `beforeUpdate` запускается после изменения данных вашего компонента и начала цикла обновления и до исправления и повторного рендеринга модели DOM.

Используйте `beforeUpdate`, если вам нужно получить новое состояние любых реактивных данных вашего компонента до фактического рендеринга:

beforeUpdate

Хук `beforeUpdate` запускается после изменения данных вашего компонента и начал цикла обновления и до исправления и повторного рендеринга модели DOM.

Используйте `beforeUpdate`, если вам нужно получить новое состояние любых реактивных данных вашего компонента до фактического рендеринга:

beforeUpdate

```
beforeUpdate() {
    console.log(`BEFORE UPDATE.`)
    console.log(this.counter)
  }
```

Обновим компонент и проверим хук

1 Добавим метод и состояние для обновления компонента

```
methods: {
    uppdateComponent:
      function (event) {
        this.name = 'React'
      }
  }
```

```
data: {
    name: 'Vue.js'
  },
```

Обновим компонент и проверим хук

2 Добавим клик на элемент в шаблон для обновления дерева

```
template: `<div v-on:click="uppdateComponent">HELLO {{name}}</div>`
```

Что произошло?

- Нажали на кнопку

- Обновилось состояние

  Состояние выводилось в шаблоне и после обновления состояния, обновился шаблон (двустороняя связь данных с шаблоном)

- После этого сработал хук beforeUpdate

updated

Хук `updated` запускается после изменения данных вашего компонента и повторного рендеринга DOM.

Используйте `updated`, если вам требуется доступ к DOM после изменения свойства:

После нажатия на заголовок, данный хук так же сработает

updated

```
updated() {
    console.log(`updated.`)
}
```

Уничтожение (разрушение)

Хуки уничтожения позволяют выполнять действия во время уничтожения компонента, например, при очистке или отправке аналитических данных. Они срабатывают, когда ваш компонент уничтожается и удаляется из DOM.

beforeDestroy

`beforeDestroy` срабатывает прямо перед уничтожением. Ваш компонент все еще присутствует и полностью функционален.

Используйте `beforeDestroy`, если вам нужно очистить события или реактивные подписки:

beforeDestroy

Добавим метод для хука

```
destroyMe: function() {
    this.$destroy();
}
```

beforeDestroy

```
beforeDestroy() {
    console.log(`Компонент разрушен!`)
 }
```

destroyed

Когда вы достигнете хука `destroyed`, от вашего компонента уже практически ничего не останется. Все, что было к нему прикреплено, будет уничтожено.

Используйте `destroyed`, если вам необходимо провести заключительную очистку или сообщить удаленному серверу об уничтожении компонента:

destroyed

```
destroyed() {
    console.log(`Destroyed`)
    console.log(this)
}
```

Другие хуки

`activated` `deactivated`

Они позволяют определит, когда компонент внутри тега `<keep-alive></keep-alive>` включается и выключается. Вы можете использовать их для доставки данных для вашего компонента или обработки изменений состояния, в результате чего они будут вести себя как `created` и `beforeDestroy` без необходимости полной перестройки компонентов.

Добавим дочерний компонент

```
Vue.component('child', {
        template: '<div>{{item}}</div>',
        props: ['item'],
        activated: function(){
          console.log('activated', this._inactive, this.item)
        },
        deactivated: function(){
          console.log('deactivated', this._inactive, this.item)
        },
        mounted: function(){
          console.log('mounted', this._inactive, this.item)
        },
        destroyed: function () {
          console.log('destroyed', this._inactive, this.item)
        }
      })
```

Добавим тестовый массив в основной компонент

```
data: {
        name: 'Vue.js',
        active: true,
        testArray: ['a', 'b', 'c']
    },
```

Добавим методы для работы с массивом

```
pushItem: function() {
            this.testArray.push('z')
          },
          popItem: function() {
            this.testArray.pop()
          }
```

Обновим шаблон основного компонента

```
<button v-on:click="pushItem()">Push Item</button>
            <button v-on:click="popItem()">Pop Item</button>
            <div v-for="(item, key) in testArray">
                <keep-alive>
                  <child :key="key" :item="item"></child>
                </keep-alive>
            </div>
```

Цикл в шаблоне

```
v-for="(item, key) in testArray"
```