Создание экземпляра Vue.js

ПОДКЛЮЧЕНИЕ

Все необходимые материалы для загрузки находятся: https://ru.vuejs.org/v2/guide/installation.html

CDN:

- версия для разработки, включает отображение полезных предупреждений в консоли:

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

- production-версия, оптимизированная в размере и скорости:

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

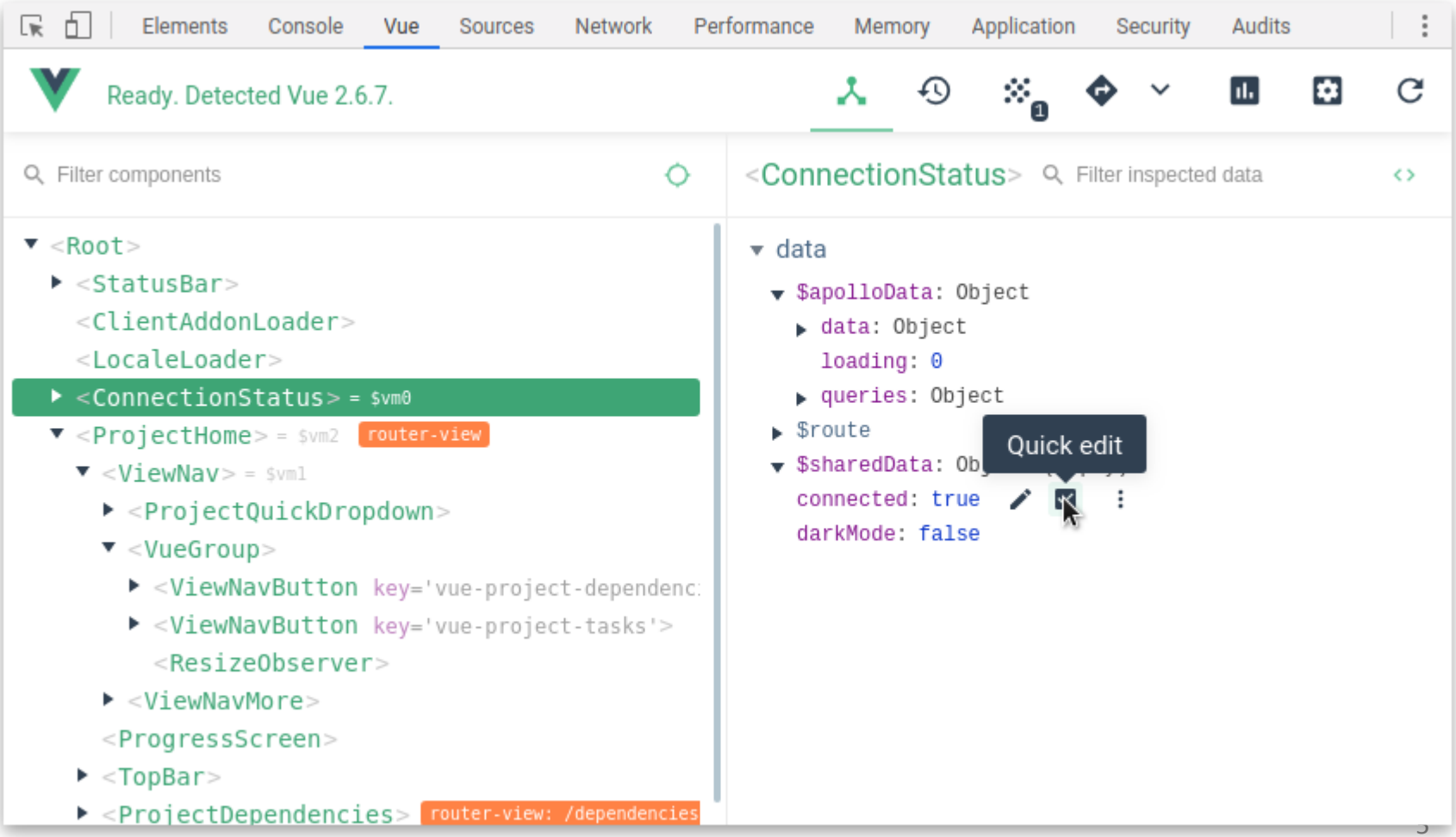

vk.com/CommitStrip
Hello-oo World
Давай же! Скажи это!
Hello...
...Woooorld...
Wo?
Да, да, давай!
Hellooo
Эй?! Что ты там говоришь?
М?
Эээ.. Ничего...
«Дядя», давай же, скажи «дядя»..
Ну-ну..
CommitStrip.com

ПРИЛОЖЕНИЕ «HELLO VUE»

```
<div id="app">
    <input type="text" v-on:input="setMessage" />
    <p>{{message}}</p>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
<script>
    let app = new Vue({
        el: '#app', //корневой элемент приложения
        data: { //используемые данные
            message: 'Hello Vue!'
        },
        methods: {//действия выполняемые в приложении
            setMessage: function(event){
                this.message = event.target.value;
            }
        }
    })
</script>
```

VUE-DEVTOOLS

vue-devtools - Chrome и Firefox DevTools расширение для отладки Vue.js приложений (https://github.com/vuejs/vue-devtools).



ОБЪЕКТ VUE

Создание приложения в Vue.js с передачей в конструктор параметров инициализации приложения:

```
let vm = new Vue({
  // параметры
})
```

Определение в конструкторе html области на веб-странице для приложения Vue.js с которой она будет работать:

```
<div id="app"></div>
<script>
    let app = new Vue({
        el: '#app' //CSS селектор
    })
</script>
```

ОБЪЕКТ VUE

Определение данных, которые должно хранить приложение (то есть данные которые определяют его состояние):

```
let app = new Vue({
    el: '#app',
    data: {
        name: 'Tom',
        age: 25
    }
});
```

ПРИВЯЗКА ДАННЫХ

Vue.js позволяет устанавливать привязку между элементами веб-страницы и данными объекта Vue.

Есть различные формы привязки. Рассмотрим:

- интерполяцию текста,
- интерполяцию HTML-кода,
- привязку к атрибуту.

Интерполяция - вставка в текст слов или фраз, отсутствовавших в оригинале.

ПРИВЯЗКА ДАННЫХ (ИНТЕРПОЛЯЦИЯ ТЕКСТА)

Значение, к которому выполняется привязка, заключается в двойные фигурные скобки (стиль шаблонизатора Mustache) – удобно использовать если нужно вывести значение свойства посреди длинной строки:

```
<div id="app">
    <p>{{name}} - {{age}}</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {name: 'Tom', age:25}
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ИНТЕРПОЛЯЦИЯ ТЕКСТА)

В интерполяции могут использоваться любые валидные выражения JavaScript.

```
<div id="app">
    <p>{{age > 25?'Больше 25 лет':'25 лет или меньше' }}</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {name: 'Tom', age:99}
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ИНТЕРПОЛЯЦИЯ ТЕКСТА)

Интерполяцию можно задать через директиву v-text - выводит текстовое представление свойства, на которое она ссылается в качестве содержимого тега.

```
<div id="app">
    <p v-text="name"></p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {name: 'Tom', age:25}
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ИНТЕРПОЛЯЦИЯ HTML-КОДА)

Для привязки содержимому элемента к коду html применяется директива **v-html:**

```
<div id="app">
    <div v-html="message"></div>
    <div>{{message}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {message:'<h2>Hello</h2>'}
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ИНТЕРПОЛЯЦИЯ HTML-КОДА)

Ещё пример, что в привязывании можно использовать любые валидные выражения JavaScript :

```
<div id="app">
    <div v-html="message.toUpperCase()"></div>
    <div>{{message}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {message:'<h2>Hello</h2>'}
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ПРИВЯЗКА К АТРИБУТАМ)

Для привязки к атрибутам html-элементов предназначена директива **v-bind:**

```
<div id="app">
    <a v-bind:href="link">{{text}}</a>
<!-- <a :href="link">{{text}}</a> краткая форма -->
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            text: 'Google',
            link:'https://google.com'
         }
    });
</script>
```

ПРИВЯЗКА ДАННЫХ (ОДНОКРАТНАЯ ПРИВЯЗКА)

Если необходимо, чтобы значение было привязано к элементу html только один раз и впоследствии не изменялось, то применяется директива **v-once:**

```
<div id="app">
    <p v-once>{{message}}</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {message:'Hello'}
    });
</script>
```

ФИЛЬТР ВЫВОДА

Для распространённых задач форматирования текста во Vue используются фильтры. Они находят своё применение в двух местах: в интерполяциях и в выражениях с директивой v-bind. Фильтры добавляются в конце выражения JavaScript и отделяются вертикальной чертой:

- Интерполяция

```
{{ message | capitalize }}
```

- v-bind

```
<div v-bind:id="link | formatLink"></div>
```

Функция фильтра всегда принимает значение выражения (результат предыдущей цепочки) в качестве первого аргумента. Фильтры можно объединять в цепочки:

```
{{ message | filterA | filterB }}
```

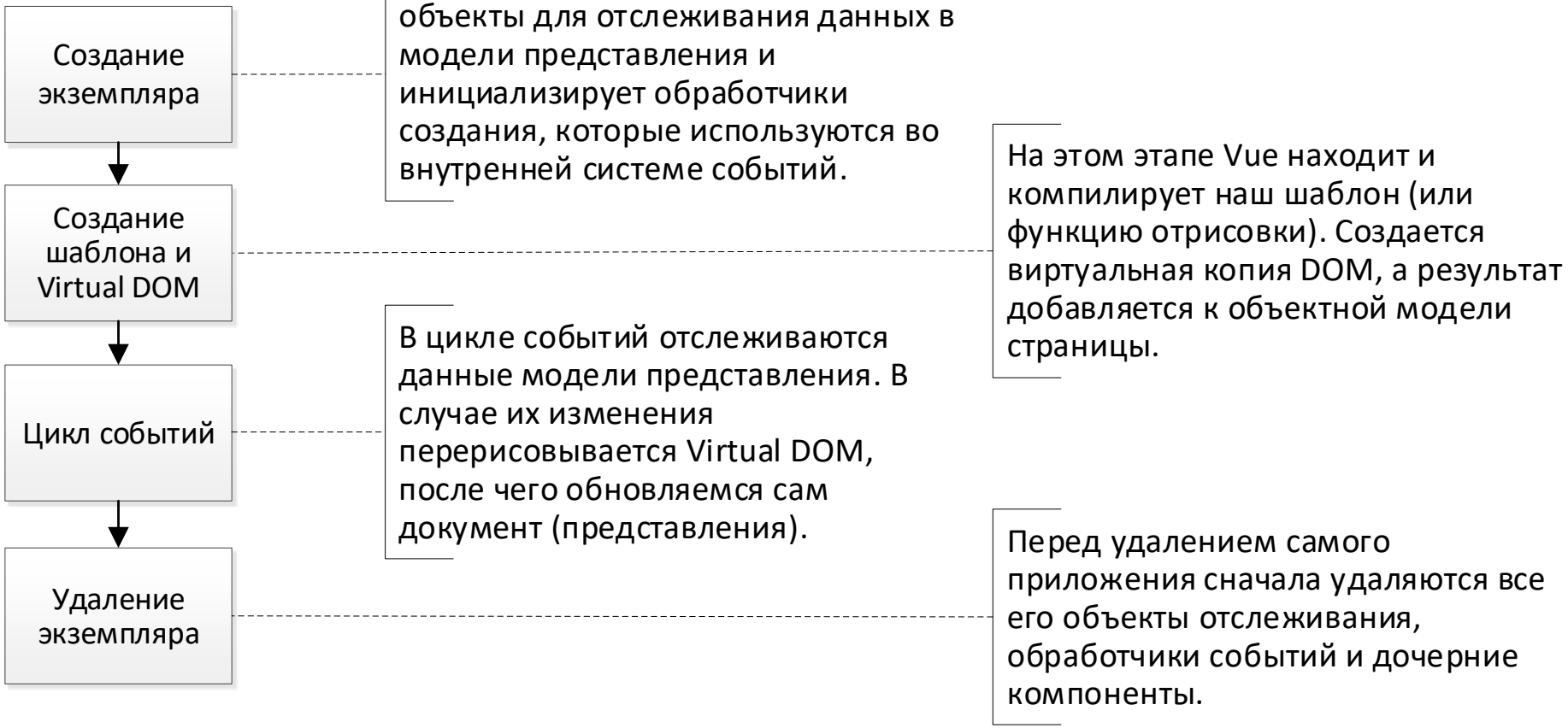
ФИЛЬТР ВЫВОДА

Пример фильтра:

```
<div id="app">
    <div>{{message | capitalize}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {message:'hello'},
        filters: {
            capitalize: function (value) {
                if (!value) return '';
                value = value.toString();
                return  value.charAt(0).toUpperCase()
                    + value.slice(1);
            }
        }
    });
</script>
```

ЖИЗНЕННЫЙ ЦИКЛ VUE

Процесс запуска Vue-приложения состоит из последовательных этапов, совокупность которых называется жизненным циклом.



ЖИЗНЕННЫЙ ЦИКЛ VUE

Хук (hook — буквально «крючок») — функция, вызываемая в определённый момент жизненного цикла.

Примеры Хуков жизненного цикла Vue.js:

- beforeCreate – Вызывается синхронно сразу после инициализации экземпляра, до настройки наблюдения за данными, механизмов слежения и событий.
- created - Вызывается синхронно сразу после создания экземпляра. На этом этапе экземпляр закончил обработку опций и настроил наблюдение за данными, вычисляемые свойства, методы, коллбэки слежения и событий. Однако, фаза монтирования ещё не начата, и свойство $el ещё не доступно.
- beforeMount - Вызывается перед началом монтирования, перед первым вызовом функции render.

ЖИЗНЕННЫЙ ЦИКЛ VUE

Примеры Хуков жизненного цикла Vue.js:

- mounted – Вызывается сразу после монтирования экземпляра, когда взамен el создан vm.$el. Если корневой экземпляр смонтирован на элемент документа, vm.$el тоже будет элементом документа.
- beforeUpdate - Вызывается при изменении данных перед тем, как DOM будет обновляться.
- updated - Вызывается после того, как виртуальный DOM был обновлён из-за изменения данных.
- beforeDestroy - Вызывается перед уничтожением экземпляра Vue. На этом этапе экземпляр всё ещё полностью функционален.
- destroyed - Вызывается после уничтожения экземпляра Vue.

ДОПОЛНИТЕЛЬНЫЕ ССЫЛКИ

Официальное руководство по Vue.js:
https://ru.vuejs.org/v2/guide/

Руководство по Vue.js (онлайн учебник)
https://metanit.com/web/vuejs/