Основы интерактивности в Vue.js

ОБРАБОТКА СОБЫТИЙ

Для обработки событий элементов html используется директива v-on, после которой через двоеточие указывается тип события:

```
<div id="app">
    <button v-on:click="counter++">+</button>
    <!-- Сокращенный синтаксис обработчиков событий -->
    <button @click=";if(counter>0) counter--;">-</button>
    <div>{{counter}}</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/vue">
</script>
<script>
    let app = new Vue({
        el: '#app',
        data: {counter:0},
    });
</script>
```

ОБРАБОТКА СОБЫТИЙ

Обработчики как отдельные методы объекта Vue:

```
<div id="app">
    <button v-on:click="increase">+</button>
    <div>{{counter}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {counter:0},
        methods:{
            increase: function(){
                this.counter++;
            }
        }
    });
</script>
```

ОБРАБОТКА СОБЫТИЙ

При генерации события в его обработчике в качестве параметра мы можем получить объект, который инкапсулирует всю информацию о событии:

```
<div id="app">
    <button v-on:click="increase">+</button>
    <div>{{counter}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {counter:0},
        methods:{
            increase: function(event){
                console.log(event);
                this.counter++;
            }
        }
    });
</script>
```

ОБРАБОТКА СОБЫТИЙ

Можно передать в методы для обработки событий какие-то дополнительные значения:

```
<div id="app">
    <button v-on:click="increase(3, $event)">+</button>
    <div>{{counter}}</div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {counter:0},
        methods:{
            increase: function(n, event){
                console.log(event);
                this.counter = this.counter + n;
            }
         }
    });
</script>
```

ВЫЧИСЛЯЕМЫЕ СВОЙСТВА

```
<div id="app">
    <p>Счётчик: {{counter}}</p>
    <button v-on:click="counter++">+</button>
    <button v-on:click="counter--">-</button>
    <p>Возраст: {{age}}</p>
    <button v-on:click="age++">+</button>
    <button v-on:click="age--">-<button>
    <p>{{checkAge()}}</p>
</div><script>
    let app = new Vue({
        el: '#app',
        data: {name:'Tom', age:25},
        methods:{
            checkAge: function(){
                console.log("method");
                if(this.age > 17)
                    return "доступ разрешен";
                else
                    return "доступ запрещен";
            }
        }
    });
</script>
```

!!! Метод checkAge будет выполняться при изменении любого свойства во Vue

ВЫЧИСЛЯЕМЫЕ СВОЙСТВА

Вычисляемое свойства определяются с помощью параметра computed:

```
<div id="app">
    <p>Счётчик: {{counter}}</p>
    <button v-on:click="counter++">+</button>
    <button v-on:click="counter--">-</button>
    <p>Возраст: {{age}}</p>
    <button v-on:click="age++">+</button>
    <button v-on:click="age--">-<button>
    <p>{{checkAge}}</p>
</div><script>
    let app = new Vue({
        el: '#app',
        data: {name:'Tom', age:25},
        computed:{
            checkAge: function(){
                console.log("method");
                if(this.age > 17)
                    return "доступ разрешен";
                else
                    return "доступ запрещен";
            }
        }
    });
</script>
```

!!! Вычисляемое свойство checkAge будет пересчитываться **при изменении свойств от которых зависит** вычисляемое свойство во Vue

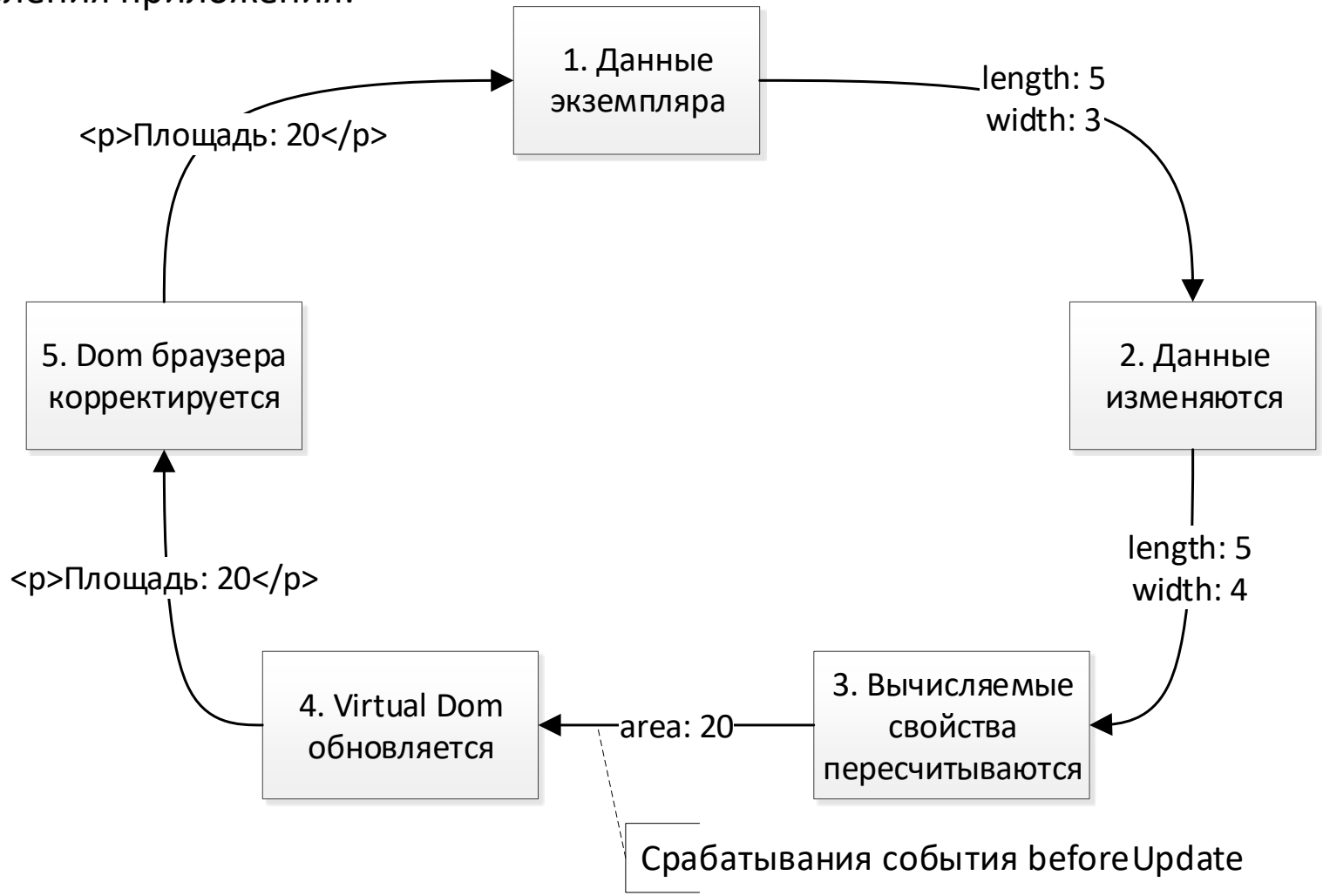
МЕТОДЫ НАБЛЮДАТЕЛИ

Отслеживание изменений свойств модели представления можно с помощью методов наблюдателей:

```
<div id="app">
    <p>Площадь: {{area}}</p>
    <button v-on:click="length++">Добавить длину</button>
    <button v-on:click="width++">Добавить ширину</button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {length:5, width:3},
        computed:{
            area: function(){
                console.log('computed method area');
                return this.length * this.width;
            }
        },
        watch:{
            width: function(newVal, oldVal){
                console.log(`Изменилось свойство width с ${oldVal} на ${newVal}`);
            }
        }
    });
</script>
```

ЦИКЛ ОБНОВЛЕНИЯ ПРИЛОЖЕНИЯ

Изменения данных экземпляра запускают последовательность событий в рамках цикла обновления приложения:



ДИРЕКТИВА V-IF

Директива **v-if** позволяет отобразить или скрыть элемент html по условию:

```
<div id="app">
    <p v-if="visible">Первый параграф</p>
    <p>Второй параграф</p>
    <button v-on:click="visible=!visible">
      {{visible?'Скрыть':'Показать'}}</button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            visible: true
        }
    });
</script>
```

ДИРЕКТИВА V-IF

В паре с директивой **v-if** может использоваться директива **v-else**:

```
<div id="app">
    <p v-if="visible">Первый параграф</p>
    <p v-else>Второй параграф</p>
    <button v-on:click="visible=!visible">
     {{visible?'К параграфу 2':'К параграфу 1'}}
    </button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            visible: true
        }
    });
</script>
```

ДИРЕКТИВА V-IF

!!! Антипримеры использования пары **v-if** и **v-else (работать не будет)**:

1)
```
<div id="app">
    <p v-if="visible">Первый параграф</p>
    <p>Промежуточный параграф</p>
    <p v-else>Второй параграф</p>
</div>
```

2)
```
<div id="app">
    <div>
        <p v-if="visible">Первый параграф</p>
    </div>
    <div>
        <p v-else>Второй параграф</p>
    <div>
</div>
```

ДИРЕКТИВА V-SHOW

Директива **v-show** аналогично **v-if** позволяет скрывать или отображать элементы по определенному условию. Отличие от **v-if** директива **v-show** не изменяет структуру DOM, а манипулирует значением стилевого свойства display:

```
<div id="app">
    <div v-show="visible">
        <h2>Заголовок</h2>
        <p>Текст</p>
    </div>
    <button v-on:click="visible=!visible">
      {{visible?'Скрыть':'Отобразить'}}</button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: { visible: true }
    });
</script>
```

МОДИФИКАТОРЫ СОБЫТИЙ

Очень часто возникает необходимость вызывать event.preventDefault() или event.stopPropagation() в обработчике события. Несмотря на то, что это легко сделать внутри метода, лучше сохранять чистоту логики и абстрагироваться от деталей реализации событий DOM.

Для решения этой задачи Vue предоставляет модификаторы событий для v-on, которые указываются как постфиксы и отделяются точкой:

- .stop
- .prevent
- .capture
- .self
- .once
- .passive

МОДИФИКАТОРЫ СОБЫТИЙ

```
<!-- событие click не будет всплывать дальше -->
<a v-on:click.stop="doThis"></a>

<!-- событие submit больше не будет перезагружать страницу -->
<form v-on:submit.prevent="onSubmit"></form>

<!-- модификаторы можно объединять в цепочки -->
<a v-on:click.stop.prevent="doThat"></a>

<!-- и использовать без указания метода-обработчика -->
<form v-on:submit.prevent></form>

<!-- можно отслеживать события в режиме capture, т.е. событие,
нацеленное на внутренний элемент, обрабатывается здесь до обработки
этим элементом -->
<div v-on:click.capture="doThis">...</div>

<!-- вызов обработчика только в случае наступления события
непосредственно на данном элементе (то есть не на дочернем
компоненте) -->
<div v-on:click.self="doThat">...</div>
```

МОДИФИКАТОРЫ СОБЫТИЙ

```
<!-- Событие click сработает только 1 раз -->
<a v-on:click.once="doThis"></a>

<! Модификатор .passive, соответствует опции passive в addEventListener. Модификатор указывает, что слушатель никогда не вызовет preventDefault(). Модификатор .passive особенно полезен для повышения производительности на мобильных устройствах. -->
<div v-on:scroll.passive="onScroll">...</div>
```

МОДИФИКАТОРЫ СОБЫТИЙ

При обработке событий клавиатуры нас часто интересуют конкретные клавиши. Vue позволяет использовать модификаторы клавиш при использовании v-on для отслеживания событий от клавиатуры:

- .enter
- .tab
- .delete (ловит как «Delete», так и «Backspace»)
- .esc
- .space
- .up
- .down
- .left
- .right

Пример: <input v-on:keyup.page-down="onPageDown">

МОДИФИКАТОРЫ СОБЫТИЙ

!!! Использование значения keyCode событий — устаревшая практика и может не поддерживаться в новых браузерах.

Использование значений keyCode:

```
<input v-on:keyup.112="help">
```

Можно также определить пользовательские псевдонимы клавиш через глобальную опцию config.keyCodes:

```
// позволит использовать `v-on:keyup.f1`
Vue.config.keyCodes.f1 = 112;
```

МОДИФИКАТОРЫ СОБЫТИЙ

Можно использовать следующие модификаторы для отслеживания событий мыши или клавиатуры с зажатой клавишей-модификатором:

- .ctrl
- .alt
- .shift
- .meta

Примечание: На клавиатурах Apple клавиша meta отмечена знаком ⌘. На клавиатурах Windows клавиша meta отмечена знаком ⊞. На клавиатурах Sun Microsystems клавиша meta отмечена символом ромба ◆. На некоторых клавиатурах, особенно MIT и Lisp machine и их преемников, таких как Knight или space-cadet клавиатуры, клавиша meta отмечена словом «META». На клавиатурах Symbolics, клавиша meta отмечена словом «META» или «Meta».

Примеры:

```
<!-- Alt + C -->
<input @keyup.alt.67="clear">

<!-- Ctrl + Click -->
<div @click.ctrl="doSomething">Сделать что-то</div>
```

МОДИФИКАТОРЫ СОБЫТИЙ

Модификатор .exact позволяет контролировать точную комбинацию системных модификаторов, необходимых для запуска события.

Примеры:

```
<!-- сработает, даже если Alt или Shift также нажаты -->
<button @click.ctrl="onClick">A</button>

<!-- сработает, только когда нажат Ctrl и не нажаты никакие другие клавиши -->
<button @click.ctrl.exact="onCtrlClick">A</button>

<!-- сработает, только когда не нажаты никакие системные модификаторы -->
<button @click.exact="onClick">A</button>
```

МОДИФИКАТОРЫ СОБЫТИЙ

Эти модификаторы ограничивают обработчик события только вызовами определённой кнопкой мыши.

- .left
- .right
- .middle

Пример:

```
<button v-on:mouseup.right="doSomething"
        v-on:contextmenu.prevent>+</button>
```

ДОПОЛНИТЕЛЬНЫЕ ССЫЛКИ

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

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