

Раздел

Веб-компоненты

Навигация по уроку

Всплытие и метод  
event.composedPath()

Свойство: event.composed

Генерация событий


Итого

Комментарии

Поделиться



Редактировать на GitHub

 → Веб-компоненты 27-го июня 2019

## Теневого DOM и события

Смысл создания теневого DOM-дерева – это инкапсуляция внутренних деталей компонента.

Допустим, клик произошёл внутри теневого DOM на компоненте `<user-card>`. Но скрипты основного документа ничего не знают о внутреннем устройстве теневого DOM-структуры, в особенности, если компонент создан сторонней библиотекой.

Поэтому, чтобы не нарушать инкапсуляцию, браузер *меняет у этого события целевой элемент*.

**События, которые произошли в теновом DOM, но пойманы снаружи этого DOM, имеют элемент-хозяин в качестве целевого элемента `event.target`.**

Рассмотрим простой пример:

```
1 <user-card></user-card>
2
3 <script>
4   customElements.define('user-card', class extends HTMLElement {
5     connectedCallback() {
6       this.attachShadow({mode: 'open'});
7       this.shadowRoot.innerHTML = `<p>
8         <button>Нажми меня</button>
9       </p>`;
10      this.shadowRoot.firstElementChild.onclick =
11        e => alert("Внутренний целевой элемент: " + e.target);
12    }
13  });
14
15 document.onclick =
16   e => alert("Внешний целевой элемент: " + e.target.tagName);
17 </script>
```



Если нажать на кнопку, то выведется следующее:

1. Внутренний целевой элемент: BUTTON – внутренний обработчик событий получает правильный целевой элемент – элемент, находящийся внутри теневого DOM.
2. Внешний целевой элемент: USER-CARD – обработчик событий на уровне документа получает элемент-хозяин в качестве целевого.

Хорошо, что браузер подменяет целевые элементы событий. Потому что внешний документ ничего не знает о внутреннем устройстве компонента. С его (внешнего документа) точки зрения, событие происходит на `<user-card>`.

**Подмена целевого элемента не происходит, если событие берёт начало на элементе из слота, который фактически находится в обычном, светлом DOM.**

Например, если пользователь кликнет на `<span slot="username">` в примере ниже – целевой элемент события будет именно этот `span` для обоих обработчиков – теневого и обычного (светлого):

```
1 <user-card id="userCard">
2   <span slot="username">John Smith</span>
3 </user-card>
4
```

Раздел

Веб-компоненты

Навигация по уроку

Всплытие и метод  
event.composedPath()

Свойство: event.composed

Генерация событий

Итого

Комментарии

Поделиться



Редактировать на GitHub



```
5 <script>
6 customElements.define('user-card', class extends HTMLElement {
7   connectedCallback() {
8     this.attachShadow({mode: 'open'});
9     this.shadowRoot.innerHTML = `<div>
10       <b>Имя:</b> <slot name="username"></slot>
11     </div>`;
12
13     this.shadowRoot.firstElementChild.onclick =
14       e => alert("Внутренний целевой элемент: " + e.target);
15   }
16 });
17
18 userCard.onclick = e => alert(`Внешний целевой элемент:
19 </script>
```

Имя: John Smith

Если клик произойдёт на "John Smith", то для обоих обработчиков – внутреннего и внешнего – целевым элементом будет `<span slot="username">`. Это элемент обычного (светлого) DOM, так что подмены не происходит.

С другой стороны, если клик произойдёт на элементе, который находится в теновом DOM, например, на `<b>Имя:</b>`, то как только всплытие выйдет за пределы теневой DOM-структуры, его `event.target` станет `<user-card>`.

## Всплытие и метод event.composedPath()

Для обеспечения всплытия событий используется развёрнутый DOM.

Таким образом, если у нас есть элемент в слоте, и событие происходит где-то внутри него, то оно всплывает до `<slot>` и выше.

Полный путь к изначальному целевому элементу, со всеми теневыми элементами, можно получить, воспользовавшись методом `event.composedPath()`. Как видно из названия, этот метод возвращает путь после композиции.

В примере выше развёрнутое DOM-дерево будет таким:

```
1 <user-card id="userCard">
2   #shadow-root
3     <div>
4       <b>Имя:</b>
5       <slot name="username">
6         <span slot="username">John Smith</span>
7       </slot>
8     </div>
9 </user-card>
```

Так что, при клике по `<span slot="username">` вызов метода `event.composedPath()` вернёт массив: `[span, slot, div, shadow-root, user-card, body, html, document, window]`. Что в точности отражает цепочку родителей от целевого элемента в развёрнутой DOM-структуре после композиции.



**Детали теневого DOM-дерева доступны только для деревьев с {mode: 'open'}**

Если теневое DOM-дерево было создано с `{mode: 'closed'}`, то после композиции путь будет начинаться с элемента-хозяина: `user-card` и дальше вверх по дереву.

Этот метод следует тем же принципам, что и остальные. Внутреннее устройство закрытых DOM-деревьев совершенно скрыто.

Раздел

[Веб-компоненты](#)

Навигация по уроку

Всплытие и метод  
`event.composedPath()`

Свойство: `event.composed`

Генерация событий

Итого

Комментарии

Поделиться



[Редактировать на GitHub](#)



## Свойство: `event.composed`

Большинство событий успешно всплывают сквозь границу теневого DOM. Но не все.

Это поведение регулируется с помощью свойства `composed` объекта события. Если оно `true`, то событие пересекает границу. Иначе, оно может быть поймано лишь внутри теневого DOM.

Если посмотреть в [спецификацию UI Events](#), то большинство событий имеют `composed: true`:

- `blur`, `focus`, `focusin`, `focusout`,
- `click`, `dblclick`,
- `mousedown`, `mouseup`, `mousemove`, `mouseout`, `mouseover`,
- `wheel`,
- `beforeinput`, `input`, `keydown`, `keyup`.

Все события курсора и сенсорные события также имеют `composed: true`.

Хотя есть и события, имеющие `composed: false`:

- `mouseenter`, `mouseleave` (они вообще не всплывают),
- `load`, `unload`, `abort`, `error`,
- `select`,
- `slotchange`.

Эти события могут быть пойманы только на элементах того же DOM, в котором находится целевой элемент события.

## Генерация событий

Когда мы генерируем своё событие, то, чтобы оно всплывало за пределы компонента, нужно установить оба свойства: `bubbles` и `composed` – в значение `true`.

Например, здесь мы создаём элемент `div#inner` в теновом DOM-дереве элемента `div#outer` и генерируем на нём два события. Только одно с флагом `composed: true` выйдет наружу, в документ:

```
1 <div id="outer"></div>
2
3 <script>
4   outer.attachShadow({mode: 'open'});
5
6   let inner = document.createElement('div');
7   outer.shadowRoot.append(inner);
8
9   /*
10  div(id=outer)
11    #shadow-dom
12      div(id=inner)
13    */
14
15  document.addEventListener('test', event => alert(event.
16
17  inner.dispatchEvent(new CustomEvent('test', {
18    bubbles: true,
19    composed: true,
20    detail: "composed"
21  }));
22
23  inner.dispatchEvent(new CustomEvent('test', {
24    bubbles: true,
25    composed: false,
26    detail: "not composed"
27  }));
28 </script>
```

## Итого

Раздел

Веб-компоненты

Навигация по уроку

Всплытие и метод  
event.composedPath()

Свойство: event.composed

Генерация событий

Итого

Комментарии

Поделиться



Редактировать на GitHub



Только те события пересекают границы теневого DOM, у которых флаг `composed` установлен в значение `true`.

У большинства встроенных событий стоит `composed: true`, это описано в соответствующих спецификациях:

- UI Events <https://www.w3.org/TR/uievents>.
- Touch Events <https://w3c.github.io/touch-events>.
- Pointer Events <https://www.w3.org/TR/pointerevents>.
- ...И так далее.

У некоторых встроенных событий всё же стоит `composed: false`:

- `mouseenter`, `mouseleave` (вообще не всплывают),
- `load`, `unload`, `abort`, `error`,
- `select`,
- `slotchange`.

Эти события могут быть пойманы только на элементах, принадлежащих тому же DOM-дереву.

Если мы генерируем своё событие `CustomEvent`, то должны явно поставить флаг `composed: true`.

Обратите внимание, что в случае вложенных компонентов теньевые DOM могут быть вложены друг в друга. События с флагом `composed` всплывают через границы всех теневого DOM. Поэтому, если событие предназначено только для ближайшего внешнего компонента-родителя, мы можем инициировать его на элементе-хозяине и установить флаг `composed: false`. Тогда оно будет уже вне теневого DOM компонента, но не выплывает наружу в «ещё более внешний» DOM.

Проводим [курсы по JavaScript и фреймворкам](#).



## Комментарии

перед тем как писать...