

Раздел

[Введение в события](#)

Навигация по уроку


Применение делегирования:
действия в разметкеПриём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Браузер: документ, события, интерфейсы](#)
[→ Введение в события](#) 11-го августа 2020

Делегирование событий

Всплытие и перехват событий позволяет реализовать один из самых важных приёмов разработки – *делегирование*.

Идея в том, что если у нас есть много элементов, события на которых нужно обрабатывать похожим образом, то вместо того, чтобы назначать обработчик каждому, мы ставим один обработчик на их общего предка.

Из него можно получить целевой элемент `event.target`, понять на каком именно потомке произошло событие и обработать его.

Рассмотрим пример – [диаграмму Ба-Гуа](#). Это таблица, отражающая древнюю китайскую философию.

Вот она:

Квадрат <i>Bagua</i> : Направление, Элемент, Цвет, Значение		
Северо-Запад Металл Серебро Старейшины	Север Вода Синий Перемены	Северо-Восток Земля Жёлтый Направление
Запад Металл Золото Молодость	Центр Всё Пурпурный Гармония	Восток Дерево Синий Будущее
Юго-Запад Земля Коричневый Спокойствие	Юг Огонь Оранжевый Слава	Юго-Восток Дерево Зелёный Роман

Её HTML (схематично):

```
1 <table>
2   <tr>
3     <th colspan="3">Квадрат <em>Bagua</em>: Направление
4   </tr>
5   <tr>
6     <td>...<strong>Северо-Запад</strong>...</td>
7     <td>...</td>
8     <td>...</td>
9   </tr>
10  <tr>...ещё 2 строки такого же вида...</tr>
11  <tr>...ещё 2 строки такого же вида...</tr>
12 </table>
```

В этой таблице всего 9 ячеек, но могло бы быть и 99, и даже 9999, не важно.

Наша задача – реализовать подсветку ячейки `<td>` при клике.

Вместо того, чтобы назначать обработчик `onclick` для каждой ячейки `<td>` (их может быть очень много) – мы повесим «единый» обработчик на элемент `<table>`.

Он будет использовать `event.target`, чтобы получить элемент, на котором произошло событие, и подсветить его.

Код будет таким:

```
1 let selectedTd;
2
3
4
```

Раздел

Введение в события

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



Редактировать на GitHub



```
5 table.onclick = function(event) {
6   let target = event.target; // где был клик?
7
8   if (target.tagName !== 'TD') return; // не на TD? тогда
9
10  highlight(target); // подсветить TD
11 };
12
13 function highlight(td) {
14   if (selectedTd) { // убрать существующую подсветку, если
15     selectedTd.classList.remove('highlight');
16   }
17   selectedTd = td;
18   selectedTd.classList.add('highlight'); // подсветить
19 }
```

Такому коду нет разницы, сколько ячеек в таблице. Мы можем добавлять, удалять `<td>` из таблицы динамически в любое время, и подсветка будет стабильно работать.

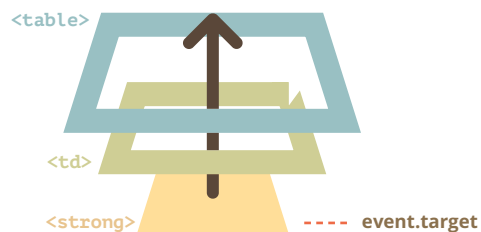
Однако, у текущей версии кода есть недостаток.

Клик может быть не на теге `<td>`, а внутри него.

В нашем случае, если взглянуть на HTML-код таблицы внимательно, видно, что ячейка `<td>` содержит вложенные теги, например ``:

```
1 <td>
2   <strong>Северо-Запад</strong>
3   ...
4 </td>
```

Естественно, если клик произойдёт на элементе ``, то он станет значением `event.target`.



Внутри обработчика `table.onclick` мы должны по `event.target` разобраться, был клик внутри `<td>` или нет.

Вот улучшенный код:

```
1 table.onclick = function(event) {
2   let td = event.target.closest('td'); // (1)
3
4   if (!td) return; // (2)
5
6   if (!table.contains(td)) return; // (3)
7
8   highlight(td); // (4)
9 };
```

Разберём пример:

1. Метод `elem.closest(selector)` возвращает ближайшего предка, соответствующего селектору. В данном случае нам нужен `<td>`, находящийся выше по дереву от исходного элемента.
2. Если `event.target` не содержится внутри элемента `<td>`, то вызов вернёт `null`, и ничего не произойдёт.



3. Если таблицы вложенные, `event.target` может содержать элемент `<td>`, находящийся вне текущей таблицы. В таких случаях мы должны проверить, действительно ли это `<td>` нашей таблицы.
4. И если это так, то подсвечиваем его.

В итоге мы получили короткий код подсветки, быстрый и эффективный, которому совершенно не важно, сколько всего в таблице `<td>`.

Применение делегирования: действия в разметке

Есть и другие применения делегирования.

Например, нам нужно сделать меню с разными кнопками: «Сохранить (save)», «Загрузить (load)», «Поиск (search)» и т.д. И есть объект с соответствующими методами `save`, `load`, `search` ... Как их состыковать?

Первое, что может прийти в голову – это найти каждую кнопку и назначить ей свой обработчик среди методов объекта. Но существует более элегантное решение. Мы можем добавить один обработчик для всего меню и атрибуты `data-action` для каждой кнопки в соответствии с методами, которые они вызывают:

```
1 <button data-action="save">Нажмите, чтобы Сохранить</button>
```

Обработчик считывает содержимое атрибута и выполняет метод. Взгляните на рабочий пример:

```
1 <div id="menu">
2   <button data-action="save">Сохранить</button>
3   <button data-action="load">Загрузить</button>
4   <button data-action="search">Поиск</button>
5 </div>
6
7 <script>
8   class Menu {
9     constructor(elem) {
10       this._elem = elem;
11       elem.onclick = this.onClick.bind(this); // (*)
12     }
13
14     save() {
15       alert('сохраняю');
16     }
17
18     load() {
19       alert('загружаю');
20     }
21
22     search() {
23       alert('ищу');
24     }
25
26     onClick(event) {
27       let action = event.target.dataset.action;
28       if (action) {
29         this[action]();
30       }
31     };
32   }
33
34   new Menu(menu);
35 </script>
```

Сохранить Загрузить Поиск

Обратите внимание, что метод `this.onClick` в строке, отмеченной звёздочкой `(*)`, привязывается к контексту текущего объекта `this`. Это

Раздел

Введение в события

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



Редактировать на GitHub



важно, т.к. иначе `this` внутри него будет ссылаться на DOM-элемент (`elem`), а не на объект `Menu`, и `this[action]` будет не тем, что нам нужно.

Так что же даёт нам здесь делегирование?

- Не нужно писать код, чтобы присвоить обработчик каждой кнопке. Достаточно просто создать один метод и поместить его в разметку.
- Структура HTML становится по-настоящему гибкой. Мы можем добавлять/удалять кнопки в любое время.

Мы также можем использовать классы `.action-save`, `.action-load`, но подход с использованием атрибутов `data-action` является более семантическим. Их можно использовать и для стилизации в правилах CSS.

Приём проектирования «поведение»

Делегирование событий можно использовать для добавления элементам «поведения» (behavior), декларативно задавая хитрые обработчики установкой специальных HTML-атрибутов и классов.

Приём проектирования «поведение» состоит из двух частей:

1. Элементу ставится пользовательский атрибут, описывающий его поведение.
2. При помощи делегирования ставится обработчик на документ, который ловит все клики (или другие события) и, если элемент имеет нужный атрибут, производит соответствующее действие.

Поведение: «Счётчик»

Например, здесь HTML-атрибут `data-counter` добавляет кнопкам поведение: «увеличить значение при клике»:

```
1 Счётчик: <input type="button" value="1" data-counter>
2 Ещё счётчик: <input type="button" value="2" data-counte
3
4 <script>
5   document.addEventListener('click', function(event) {
6
7     if (event.target.dataset.counter != undefined) { //
8       event.target.value++;
9     }
10
11   });
12 </script>
```

Счётчик: Ещё счётчик:

Если нажать на кнопку – значение увеличится. Конечно, нам важны не счётчики, а общий подход, который здесь продемонстрирован.

Элементов с атрибутом `data-counter` может быть сколько угодно. Новые могут добавляться в HTML-код в любой момент. При помощи делегирования мы фактически добавили новый «псевдостандартный» атрибут в HTML, который добавляет элементу новую возможность («поведение»).

Раздел

Введение в события

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



Редактировать на GitHub



⚠ Всегда используйте метод `addEventListener` для обработчиков на уровне документа

Когда мы устанавливаем обработчик событий на объект `document`, мы всегда должны использовать метод `addEventListener`, а не `document.on<событие>`, т.к. в случае последнего могут возникать конфликты: новые обработчики будут перезаписывать уже существующие.

Для реального проекта совершенно нормально иметь много обработчиков на элементе `document`, установленных из разных частей кода.

Поведение: «Переключатель» (Toggler)

Ещё один пример поведения. Сделаем так, что при клике на элемент с атрибутом `data-toggle-id` будет скрываться/показываться элемент с заданным `id`:

```
1 <button data-toggle-id="subscribe-mail">
2   Показать форму подписки
3 </button>
4
5 <form id="subscribe-mail" hidden>
6   Ваша почта: <input type="email">
7 </form>
8
9 <script>
10  document.addEventListener('click', function(event) {
11    let id = event.target.dataset.toggleId;
12    if (!id) return;
13
14    let elem = document.getElementById(id);
15
16    elem.hidden = !elem.hidden;
17  });
18 </script>
```

Показать форму подписки

Ещё раз подчеркнём, что мы сделали. Теперь для того, чтобы добавить скрытие-раскрытие любому элементу, даже не надо знать JavaScript, можно просто написать атрибут `data-toggle-id`.

Это бывает очень удобно – не нужно писать JavaScript-код для каждого элемента, который должен так себя вести. Просто используем поведение. Обработчики на уровне документа сделают это возможным для элемента в любом месте страницы.

Мы можем комбинировать несколько вариантов поведения на одном элементе.

Шаблон «поведение» может служить альтернативой для фрагментов JS-кода в вёрстке.

Итого

Делегирование событий – это здорово! Пожалуй, это один из самых полезных приёмов для работы с DOM.

Он часто используется, если есть много элементов, обработка которых очень схожа, но не только для этого.

Алгоритм:

1. Вешаем обработчик на контейнер.
2. В обработчике проверяем исходный элемент `event.target`.
3. Если событие произошло внутри нужного нам элемента, то обрабатываем его.

Зачем использовать:

- Упрощает процесс инициализации и экономит память: не нужно вешать много обработчиков.
- Меньше кода: при добавлении и удалении элементов не нужно ставить или снимать обработчики.
- Удобство изменений DOM: можно массово добавлять или удалять элементы путём изменения `innerHTML` и ему подобных.

Конечно, у делегирования событий есть свои ограничения:

- Во-первых, событие должно всплывать. Некоторые события этого не делают. Также, низкоуровневые обработчики не должны вызывать `event.stopPropagation()`.
- Во-вторых, делегирование создаёт дополнительную нагрузку на браузер, ведь обработчик запускается, когда событие происходит в любом месте контейнера, не обязательно на элементах, которые нам интересны. Но обычно эта нагрузка настолько пустяковая, что её даже не стоит принимать во внимание.

✓ Задачи

Спрячьте сообщения с помощью делегирования

важность: 5

Дан список сообщений с кнопками для удаления [x]. Заставьте кнопки работать.

В результате должно работать вот так:

Лошадь

Домашняя лошадь - животное семейства непарнокопытных, одомашненный и единственный сохранившийся подвид дикой лошади, вымершей в дикой природе, за исключением небольшой популяции лошади Пржевальского.

Осёл

Домашний осёл или ишак — одомашненный подвид дикого осла, сыгравший важную историческую роль в развитии хозяйства и культуры человека. Все одомашненные ослы относятся к африканским ослам.

Кошка

Кошка, или домашняя кошка (лат. *Félis silvestris catus*), — домашнее животное, одно из наиболее популярных (наряду с собакой) "животных-компаньонов". С точки зрения научной систематики, домашняя кошка —

P.S. Используйте делегирование событий. Должен быть лишь один обработчик на элементе-контейнере для всего.

[Открыть песочницу для задачи.](#)

решение

Раскрывающееся дерево

важность: 5

Раздел

[Введение в события](#)

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



[Редактировать на GitHub](#)

Создайте дерево, которое по клику на заголовок скрывает-показывает потомков:

- Животные
 - Млекопитающие
 - Коровы
 - Ослы
 - Собаки
 - Тигры
 - Другие
 - Змеи
 - Птицы
 - Ящерицы
- Рыбы
 - Аквариумные
 - Гуппи
 - Скалярии
 - Морские

Требования:

- Использовать только один обработчик событий (применить делегирование)
- Клик вне текста заголовка (на пустом месте) ничего делать не должен.

[Открыть песочницу для задачи.](#)

решение

Сортируемая таблица [↗](#)

важность: 4

Сделать таблицу сортируемой: при клике на элемент `<th>` строки таблицы должны сортироваться по соответствующему столбцу.

Каждый элемент `<th>` имеет атрибут `data-type`:

```
1 <table id="grid">
2   <thead>
3     <tr>
4       <th data-type="number">Возраст</th>
5       <th data-type="string">Имя</th>
6     </tr>
7   </thead>
8   <tbody>
9     <tr>
10      <td>5</td>
11      <td>Вася</td>
12    </tr>
13    <tr>
14      <td>10</td>
15      <td>Петя</td>
16    </tr>
17    ...
18  </tbody>
19 </table>
```

В примере выше первый столбец содержит числа, а второй – строки. Функция сортировки должна это учитывать, ведь числа сортируются иначе, чем строки.

Сортировка должна поддерживать только типы `"string"` и `"number"`.

Работающий пример:

Раздел

[Введение в события](#)

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



[Редактировать на GitHub](#)

Раздел

Введение в события

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



Редактировать на GitHub



Возраст	Имя
5	Вася
2	Петя
12	Женя
9	Маша
1	Илья

P.S. Таблица может быть большой, с любым числом строк и столбцов.

[Открыть песочницу для задачи.](#)

решение

Поведение "подсказка"

важность: 5

Напишите JS-код, реализующий поведение «подсказка».

При наведении мыши на элемент с атрибутом `data-tooltip`, над ним должна показываться подсказка и скрываться при переходе на другой элемент.

Пример HTML с подсказками:

```
1 <button data-tooltip="эта подсказка длиннее, чем элемен
2 <button data-tooltip="HTML<br>подсказка">Ещё кнопка</bu
```

Результат в ифрейме с документом:

ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя

ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя ЛяЛяЛя

Короткая кнопка Ещё кнопка

Прокрутите страницу, чтобы кнопки оказались у верхнего края, а затем проверьте, правильно ли выводятся подсказки.

В этой задаче мы полагаем, что во всех элементах с атрибутом `data-tooltip` – только текст. То есть, в них нет вложенных тегов (пока).

Детали оформления:

- Отступ от подсказки до элемента с `data-tooltip` должен быть 5px по высоте.
- Подсказка должна быть, по возможности, посередине элемента.
- Подсказка не должна вылезать за границы экрана, в том числе если страница частично прокручена, если нельзя показать сверху – показывать снизу элемента.
- Текст подсказки брать из значения атрибута `data-tooltip`. Это может быть произвольный HTML.

Для решения вам понадобятся два события:

- `mouseover` срабатывает, когда указатель мыши заходит на элемент.
- `mouseout` срабатывает, когда указатель мыши уходит с элемента.

Примените делегирование событий: установите оба обработчика на элемент `document`, чтобы отслеживать «заход» и «уход» курсора на элементы с атрибутом `data-tooltip` и управлять подсказками с их же помощью.

После реализации поведения – люди, даже не знакомые с JavaScript смогут добавлять подсказки к элементам.

P.S. В один момент может быть показана только одна подсказка.

Открыть песочницу для задачи.

решение

Раздел

[Введение в события](#)

Навигация по уроку

Применение делегирования:
действия в разметке

Приём проектирования
«поведение»

Итого

Задачи (4)

Комментарии

Поделиться



Редактировать на GitHub



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



Комментарии

перед тем как писать...

© 2007–2020 Илья Кантор | [о проекте](#) | [связаться с нами](#) | [пользовательское соглашение](#) | [политика конфи](#)

