

Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомкаСобытия mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠](#) → [Браузер: документ, события, интерфейсы](#)
→ [Интерфейсные события](#) 18-го февраля 2020

Движение мыши: mouseover/out, mouseenter/leave

В этой главе мы более подробно рассмотрим события, возникающие при движении указателя мыши над элементами страницы.

События mouseover/mouseout, relatedTarget

Событие `mouseover` происходит в момент, когда курсор оказывается над элементом, а событие `mouseout` – в момент, когда курсор уходит с элемента.



Эти события являются особенными, потому что у них имеется свойство `relatedTarget`. Оно «дополняет» `target`. Когда мышь переходит с одного элемента на другой, то один из них будет `target`, а другой `relatedTarget`.

Для события `mouseover`:



- `event.target` – это элемент, на который курсор перешёл.
- `event.relatedTarget` – это элемент, с которого курсор ушёл (`relatedTarget` → `target`).


Для события `mouseout` наоборот:

- `event.target` – это элемент, с которого курсор ушёл.
- `event.relatedTarget` – это элемент, на который курсор перешёл (`target` → `relatedTarget`).

В примере ниже каждое лицо и его черты – отдельные элементы. При движении указателя по этим элементам в текстовом поле отображаются происходящие события.

Каждое из них содержит информацию о `target` и `relatedTarget`:

Результат script.js style.css index.html  



События будут показываться здесь!

Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



⚠ Свойство relatedTarget может быть null

Свойство relatedTarget может быть null.

Это нормально и означает, что указатель мыши перешёл не с другого элемента, а из-за пределов окна браузера. Или же, наоборот, ушёл за пределы окна.

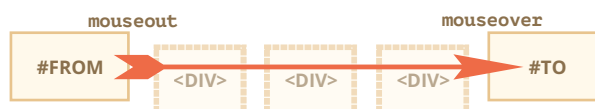
Следует держать в уме такую возможность при использовании `event.relatedTarget` в своём коде. Если, например, написать `event.relatedTarget.tagName`, то при отсутствии `event.relatedTarget` будет ошибка.

Пропуск элементов

Событие mousemove происходит при движении мыши. Однако, это не означает, что указанное событие генерируется при прохождении каждого пикселя.

Браузер периодически проверяет позицию курсора и, заметив изменения, генерирует события mousemove.

Это означает, что если пользователь двигает мышкой очень быстро, то некоторые DOM-элементы могут быть пропущены:

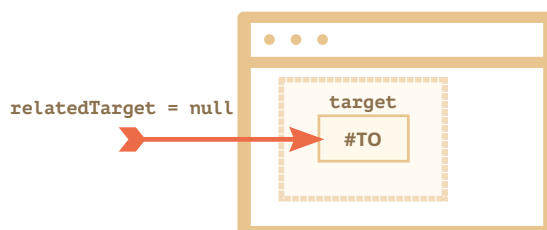


Если курсор мыши передвинуть очень быстро с элемента #FROM на элемент #TO, как это показано выше, то лежащие между ними элементы `<div>` (или некоторые из них) могут быть пропущены. Событие mouseout может запуститься на элементе #FROM и затем сразу же сгенерируется mouseover на элементе #TO.

Это хорошо с точки зрения производительности, потому что если промежуточных элементов много, вряд ли мы действительно хотим обрабатывать вход и выход для каждого.

С другой стороны, мы должны иметь в виду, что указатель мыши не «посещает» все элементы на своём пути. Он может и «прыгать».

В частности, возможно, что указатель запрыгнет в середину страницы из-за пределов окна браузера. В этом случае значение relatedTarget будет null, так как курсор пришёл «из ниоткуда»:



Вы можете проверить это «вживую» на тестовом стенде ниже.

В его HTML есть два элемента, `<div id="child">` вложен в `<div id="parent">`. Если быстро провести мышью над ними, то событие может возникнуть только на внутреннем элементе или только на внешнем, а может вообще не сгенерироваться никаких событий.

Также попробуйте поставить курсор на внутренний элемент, а затем очень быстро сделайте движение мышкой вниз через внешний элемент. Если у вас получится достаточно быстро, то на родительском элементе не будет сгенерировано никаких событий. То есть, мышь пройдёт через внешний элемент, не замечая его.

Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



parent
child



i Если был `mouseover` , то будет и `mouseout`

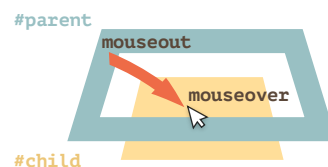
Несмотря на то, что при быстрых переходах промежуточные элементы могут игнорироваться, в одном мы можем быть уверены: элемент может быть пропущен только целиком.

Если указатель «официально» зашёл на элемент, то есть было событие `mouseover` , то при выходе с него обязательно будет `mouseout` .

Событие `mouseout` при переходе на потомка

Важная особенность события `mouseout` – оно генерируется в том числе, когда указатель переходит с элемента на его потомка.

То есть, визуально указатель всё ещё на элементе, но мы получим `mouseout` !



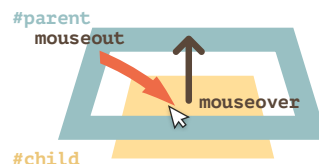
Это выглядит странно, но легко объясняется.

По логике браузера, курсор мыши может быть только над одним элементом в любой момент времени – над самым глубоко вложенным и верхним по z-index.

Таким образом, если курсор переходит на другой элемент (пусть даже дочерний), то он покидает предыдущий.

Обратите внимание на важную деталь.

Событие `mouseover` , происходящее на потомке, всплывает. Поэтому если на родительском элементе есть такой обработчик, то оно его вызовет.



Вы можете наглядно увидеть это в примере ниже: `<div id="child">` находится внутри `<div id="parent">` . На родителе определены обработчики событий `mouseover/out` , которые выводят информацию о них в текстовое поле.

При переходе мышью с внешнего элемента на внутренний, вы увидите сразу два события: `mouseout [target: parent]` (ушли с родителя) и `mouseover [target: child]` (перешли на потомка, событие всплыло).

Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Результат script.js style.css index.html



parent
child



При переходе с родителя элемента на потомка – на родителе сработают два обработчика: и mouseout и mouseover :

```
1 parent.onmouseout = function(event) {  
2   /* event.target: внешний элемент */  
3 };  
4 parent.onmouseover = function(event) {  
5   /* event.target: внутренний элемент (всплыло) */  
6 };
```

Если код внутри обработчиков не смотрит на target , то он подумает, что мышь ушла с элемента parent и вернулась на него обратно. Но это не так! Мышь никуда не уходила, она просто перешла на потомка.

Если при уходе с элемента что-то происходит, например, запускается анимация, то такая интерпретация происходящего может давать нежелательные побочные эффекты.

Чтобы этого избежать, можно смотреть на relatedTarget и, если мышь всё ещё внутри элемента, то игнорировать такие события.

Или же можно использовать другие события: mouseenter и mouseleave , которые мы сейчас изучим, с ними такая проблема не возникает.

События mouseenter и mouseleave

События mouseenter/mouseleave похожи на mouseover/mouseout . Они тоже генерируются, когда курсор мыши переходит на элемент или покидает его.

Но есть и пара важных отличий:

1. Переходы внутри элемента, на его потомки и с них, не считаются.
2. События mouseenter/mouseleave не всплывают.

События mouseenter/mouseleave предельно просты и понятны.

Когда указатель появляется над элементом – генерируется mouseenter , причём не имеет значения, где именно указатель: на самом элементе или на его потомке.

Событие mouseleave происходит, когда курсор покидает элемент.

Вот тот же пример, что и выше, но на этот раз на верхнем элементе стоят обработчики mouseenter/mouseleave вместо mouseover/mouseout .

Как вы сами можете увидеть, генерируются только события, связанные с движением курсора относительно верхнего <div> . Ничего не произойдёт при переходе на внутренний <div> и обратно. Переходы на потомки игнорируются.

Результат script.js style.css index.html



Раздел

Интерфейсные события

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



Редактировать на GitHub



parent
child



Делегирование событий

События `mouseenter/leave` просты и легки в использовании. Но они не всплывают. Таким образом, мы не можем их делегировать.

Представьте ситуацию, когда мы хотим обрабатывать события, сгенерированные при движении курсора по ячейкам таблицы. И в таблице сотни ячеек.

Очевидное решение – определить обработчик на родительском элементе `<table>` и там обрабатывать возникающие события. Но, так как `mouseenter/leave` не всплывают, то если событие происходит на ячейке `<td>`, то только обработчик на `<td>` может поймать его.

Обработчики событий `mouseenter/leave` на `<table>` срабатывают, если курсор оказывается над таблицей в целом или же уходит с неё. Невозможно получить какую-либо информацию о переходах между ячейками внутри таблицы.

Что ж, не проблема – будем использовать `mouseover/mouseout`.

Начнём с простых обработчиков, которые выделяют текущий элемент под указателем мыши:

```
1 // выделим элемент под мышью
2 table.onmouseover = function(event) {
3   let target = event.target;
4   target.style.background = 'pink';
5 };
6
7 table.onmouseout = function(event) {
8   let target = event.target;
9   target.style.background = '';
10 };
```

Вот они в действии. При переходе между элементами этой таблицы, текущий будет подсвечен:

Результат script.js style.css index.html



Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Квадрат *Bagua*: 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
Зеленый
Роман

Очистить

В нашем случае мы хотим обрабатывать переходы именно между ячейками `<td>` : вход на ячейку и выход с неё. Прочие переходы, в частности, внутри ячейки `<td>` или вообще вне любых ячеек, нас не интересуют, хорошо бы их отфильтровать.

Можно достичь этого так:

- Запоминать текущую ячейку `<td>` в переменную, которую назовём `currentElem`.
- На `mouseover` – игнорировать событие, если мы всё ещё внутри той же самой ячейки `<td>`.
- На `mouseout` – игнорировать событие, если это не уход с текущей ячейки `<td>`.

Вот пример кода, учитывающего все ситуации:

```
1 // ячейка <td> под курсором в данный момент (если есть)
2 let currentElem = null;
3
4 table.onmouseover = function(event) {
5   // перед тем, как войти на следующий элемент, курсор
6   // если currentElem есть, то мы ещё не ушли с предыдущей
7   // это переход внутри - игнорируем такое событие
8   if (currentElem) return;
9
10  let target = event.target.closest('td');
11
12  // переход не на <td> - игнорировать
13  if (!target) return;
14
15  // переход на <td>, но вне нашей таблицы (возможно при
16  // игнорировать
17  if (!table.contains(target)) return;
18
19  // ура, мы зашли на новый <td>
20  currentElem = target;
21  target.style.background = 'pink';
22 };
23
24
25 table.onmouseout = function(event) {
26   // если мы вне <td>, то игнорируем уход мыши
27   // это какой-то переход внутри таблицы, но вне <td>,
28   // например с <tr> на другой <tr>
29   if (!currentElem) return;
30
31   // мы покидаем элемент - но куда? Возможно, на потомка
32   let relatedTarget = event.relatedTarget;
```

Раздел

Интерфейсные события

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



Редактировать на GitHub



```
33
34 while (relatedTarget) {
35     // поднимаемся по дереву элементов и проверяем - вн
36     // если да, то это переход внутри элемента - игнори
37     if (relatedTarget == currentElem) return;
38
39     relatedTarget = relatedTarget.parentNode;
40 }
41
42 // мы действительно покинули элемент
43 currentElem.style.background = '';
44 currentElem = null;
45 };
```

Полный пример со всеми деталями:

Результат script.js style.css index.html



Квадрат *Vagua*: 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

Зеленый

Роман

Попробуйте подвигать курсор между ячейками и внутри них. Быстро или медленно – без разницы. В отличие от предыдущего примера выделяется только сама ячейка `<td>`.

Итого

Мы рассмотрели события `mouseover`, `mouseout`, `mousemove`, `mouseenter` и `mouseleave`.

Особенности, на которые стоит обратить внимание:

- При быстром движении мыши события не будут возникать на промежуточных элементах.
- События `mouseover/out` и `mouseenter/leave` имеют дополнительное свойство: `relatedTarget`. Оно дополняет свойство `target` и содержит ссылку на элемент, с/на который мы переходим.

События `mouseover/out` возникают, даже когда происходит переход с родительского элемента на потомка. С точки зрения браузера, курсор мыши может быть только над одним элементом в любой момент времени – над самым глубоко вложенным.

События `mouseenter/leave` в этом отличаются. Они генерируются, когда курсор переходит на элемент в целом или уходит с него. Также они не всплывают.

✓ Задачи

Улучшенная подсказка

важность: 5

Напишите JavaScript код, который показывает подсказку над элементом с атрибутом `data-tooltip`. Значение атрибута должно становиться текстом

Раздел

Интерфейсные события

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



Редактировать на GitHub

подсказки.

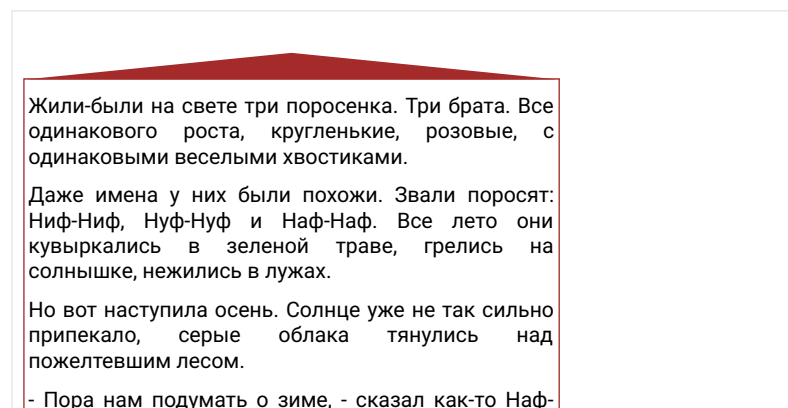
Это похоже на задачу [Поведение "подсказка"](#), но здесь элементы с подсказками могут быть вложены друг в друга. Показываться должна подсказка на самом глубоко вложенном элементе.

Только одна подсказка может быть показана в любой момент времени.

Например:

```
1 <div data-tooltip="Здесь - домашний интерьер" id="house"
2   <div data-tooltip="Здесь - крыша" id="roof"></div>
3   ...
4   <a href="https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D
5 </div>
```

Результат в iframe:



[Открыть песочницу для задачи.](#)

решение

"Умная" подсказка

важность: 5

Напишите функцию, которая показывает подсказку над элементом только в случае, когда пользователь передвигает мышь *на него*, но не *через него*.

Другими словами, если пользователь подвинул курсор на элементе и остановился – показывать подсказку. А если он просто быстро провёл курсором по элементу, то не надо ничего показывать. Кому понравится лишнее мелькание?

Технически, мы можем измерять скорость прохода курсора мыши над элементом, и если она низкая, то можно посчитать, что пользователь остановил курсор над элементом, и показать ему подсказку. А если скорость высокая, то тогда не показывать.

Создайте для этого универсальный объект `new HoverIntent(options)`.

Его настройки `options`:

- `elem` – отслеживаемый элемент.
- `over` – функция, вызываемая, при заходе на элемент, считаем что заход – это когда курсор медленно движется или остановился над элементом.
- `out` – функция, вызываемая при уходе курсора с элемента (если был заход).

Пример использования такого объекта для показа подсказки:

```
1 // пример подсказки
2 let tooltip = document.createElement('div');
3 tooltip.className = "tooltip";
4 tooltip.innerHTML = "Tooltip";
5
```


Раздел

[Интерфейсные события](#)

Навигация по уроку

События
mouseover/mouseout,
relatedTarget

Пропуск элементов

Событие mouseout при
переходе на потомка

События mouseenter и
mouseleave

Делегирование событий

Итого

Задачи (2)

Комментарии

Поделиться



[Редактировать на GitHub](#)



```
6 // объект будет отслеживать движение мыши и вызывать фу
7 new HoverIntent({
8   elem,
9   over() {
10    tooltip.style.left = elem.getBoundingClientRect().l
11    tooltip.style.top = elem.getBoundingClientRect().bo
12    document.body.append(tooltip);
13   },
14   out() {
15    tooltip.remove();
16   }
17 });
```

Демо:

12:30:00

passes: 5 failures: 0 duration: 0.60s 100%

hoverIntent

- ✓ mouseover -> immediately no tooltip
- ✓ mouseover -> pause shows tooltip
- ✓ mouseover -> fast mouseout no tooltip

Если двигать курсор над «часами» быстро, то ничего не произойдёт, а если вы замедлите движение курсора над элементом или остановите его, то будет показана подсказка.

Обратите внимание: подсказка не должна пропадать (мигать), когда курсор переходит между дочерними элементами часов.

[Открыть песочницу с тестами для задачи.](#)

решение



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



Комментарии

перед тем как писать...