Обработка браузерных событий в JS

События

- Для реакции на действия посетителя и внутреннего взаимодействия скриптов существуют *события*
- ***Событие*** *— это сигнал от браузера о том, что что-то произошло.*
- *Виды событий:*
  - DOM-события - инициируются элементами DOM.
  - События окна.
  - Другие события, например load, readystatechange. Они используются в технологии AJAX.

Часто возникающие события

- **События мыши:**
  - click – происходит, когда кликнули на элемент левой кнопкой мыши
  - contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши
  - mouseover – возникает, когда на элемент наводится мышь
  - mousedown и mouseup – когда кнопку мыши нажали или отжали
  - mousemove – при движении мыши
- **События на элементах управления:**
  - submit – посетитель отправил форму <form>
  - focus – посетитель фокусируется на элементе, например нажимает на <input>
- **Клавиатурные события:**
  - keydown – когда посетитель нажимает клавишу
  - keyup – когда посетитель отпускает клавишу
- **События документа:**
  - DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.
- **События CSS:**
  - transitionend – когда CSS-анимация завершена.

Обработчики событий

- *Событию можно назначить **обработчик**, то есть функцию, которая сработает, как только событие произошло. Именно благодаря обработчикам JavaScript-код может реагировать на действия посетителя*

Назначение обработчиков

- Обычно обработчики называют "on+имя события", например: onclick.
- JavaScript - однопоточный язык, поэтому обработчики всегда выполняются последовательно и в общем потоке.
- Способы назначать обработчик на конкретное событие элемента:
  - Через атрибут HTML-тега
  - Через свойство объекта
  - Специальные методы

Добавление обработчиков через атрибут

```
<button onclick="this.parentElement.innerHTML+=
'<span>click</span>'">Нажми меня</button>
```


```
<input id="b1" value="Нажми Меня”
 onclick="alert('Спасибо!');" type="button"/>
```

Добавление обработчиков через свойства DOM-объекта

1) получить элемент
2) назначить обработчик свойству on+имя

```
document.getElementById('myElement').onclick = function() {
    alert('Спасибо')
}
<input id="myElement" type="button" value="Нажми меня"/>
```

```
function doSomething() {
     alert('Спасибо')
}
document.getElementById('button').onclick = doSomething
```

Специальные методы

- Установка обработчика:
  - element.addEventListener(имя_события, обработчик, фаза)
- Удаление обработчика:
  - element.removeEventListener(имя_события, обработчик, фаза) – **удаляется только та же ф**

Имя события указывается без префикса "on"

`event`
Имя события, например `click`
`handler`
Ссылка на функцию, которую надо поставить обработчиком.
`phase`
Необязательный аргумент, «фаза», на которой обработчик должен сработать.

addEventListener и onclick

- **Достоинства**
- Некоторые события можно назначить только через addEventListener.
- Метод addEventListener п озволяет назначить много обработчиков на одно событие.

- **Недостатки**
- Обработчик, назначенный через onclick, проще удалить или заменить.
- Метод onclick кросс-браузерный.

Последовательность событий

Занимается выполнением JavaScript, отрисовкой

Главный поток может делать только одно дело

Потоки скачивания файлов

Дополнительные потоки

Потоки могут запускать дополнительные workes

дополнительные процессы не

Очередь событий

- Если главный поток прямо сейчас занят, то он не может срочно выйти из середины одной функции и начать выполнять другую.
- Когда происходит событие, оно попадает в очередь.
- Внутри браузера непрерывно работает «главный внутренний цикл», который следит за состоянием очереди и обрабатывает события, запускает соответствующие обработчики и т.п.
- https://learn.javascript.ru/events-and-timing-depth

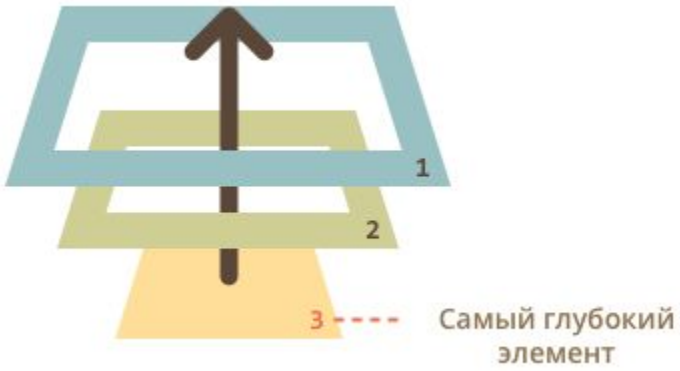
Вложенные (синхронные) события

Когда событие инициируется не посетителем, а кодом, то оно, как правило, обрабатывается синхронно, то есть прямо сейчас

Отложенный вызов через setTimeout(func, 0) используется не только в событиях, а вообще – всегда, когда мы хотим, чтобы некая функция func сработала после того, как текущий скрипт завершится.

Всплытие

- При наступлении события обработчики сначала срабатывают на самом вложенном элементе, затем на его родителе, затем выше и так далее, вверх по цепочке вложенности.




```
<form onclick="alert('form')">
FORM
  <div onclick="alert('div')“>DIV
    <p onclick="alert('p')">P</p>
  </div>
</form>
```

Объект события event

- На каком бы элементе мы ни поймали событие, всегда можно узнать, где конкретно оно произошло.
- **Самый глубокий элемент, который вызывает событие,**
  называется *«целевым»* или *«исходным»*
  элементом и доступен как event.target.

```
function(event) { alert(event.type) }
```

Общие свойства объекта event

- type - строка, содержащее имя события.
- target - DOM-элемент, который сгенерировал событие.
- currentTarget - DOM-элемент, который вызвал обработчик события.
- eventPhase - число, показывающее на каком этапе произошло событие (1 - этапе погружения (перехвата), 2 - на цели, 3 - на этапе всплытия).
- timestamp - число (дата), когда произошло событие
- bubbles - возвращает логическое значение, указывающее может ли данное событие всплывать
- defaultPrevented - проверяет можно ли вызвать метод preventDefault() для данного события.
- view - возвращает ссылку на объект window, в котором произошло событие.

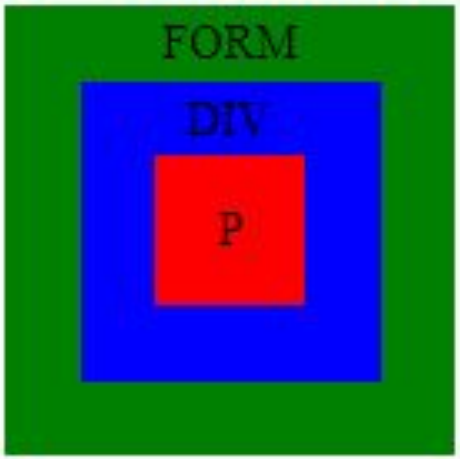
Свойства объекта event (дополнительные)

- which (для мыши) - возвращает число, указывающее, какая кнопка мыши была нажата (1 - левая кнопка, 2 - средняя кнопка, 3 - правая кнопка). Это свойство в основном используется вместе с событием mousedown.
- clientX, clientY (для мыши) - возвращают информацию о положении курсора (clientX - горизонтальная координата, clientY - вертикальная координата) относительно левого верхнего угла клиентской области.
- screenX, screenY (для мыши) - возвращают информацию о положении курсора (screenX - горизонтальная координата, screenY - вертикальная координата) относительно левого верхнего угла экрана.
- detail (для мыши) - возвращает число, указывающее сколько раз была нажата кнопка мыши в некоторой области за короткий промежуток времени.
- altKey, ctrlKey, metaKey, shiftKey (для мыши и клавиатуры) - получение дополнительной информации о том была ли нажата соответствующая клавиша alt, ctrl, meta и shift в тот момент когда произошло событие.
- relatedTarget (для мыши) - возвращает элемент, который связан с элементом, сгенерировавшим события мыши.
  Для события mouseover: свойство target - указывает на элемент, который сейчас находится под курсором; а свойство relatedTarget - на элемент с которого курсор пришёл.
  Для события mouseout: свойство target - указывает на элемент с которого курсор пришёл;
  а relatedTarget (для мыши) - на элемент, который сейчас находится под курсором.
- charCode (для клавиатуры) - возвращает код символа Unicode нажатой клавиши (для события keypress).
- keyCode, which (для клавиатуры) - возвращает код символа Unicode (для события keypress) или код ключа Unicode (для событий keydown и keyup).
- location (для клавиатуры) - возвращает число, указывающее на область клавиатуры или устройства в котором расположена нажатая клавиша (0 - основная область клавиатуры, 1 - область, в которой расположена левая клавиша CTRL или левая клавиша ALT, 2 - область, в которой расположена правая клавиша CTRL или правая клавиша ALT, 3 - область цифровой панели, которая дублирует клавиши основной области для ввода цифр и арифметических операторов).

Методы объекта event

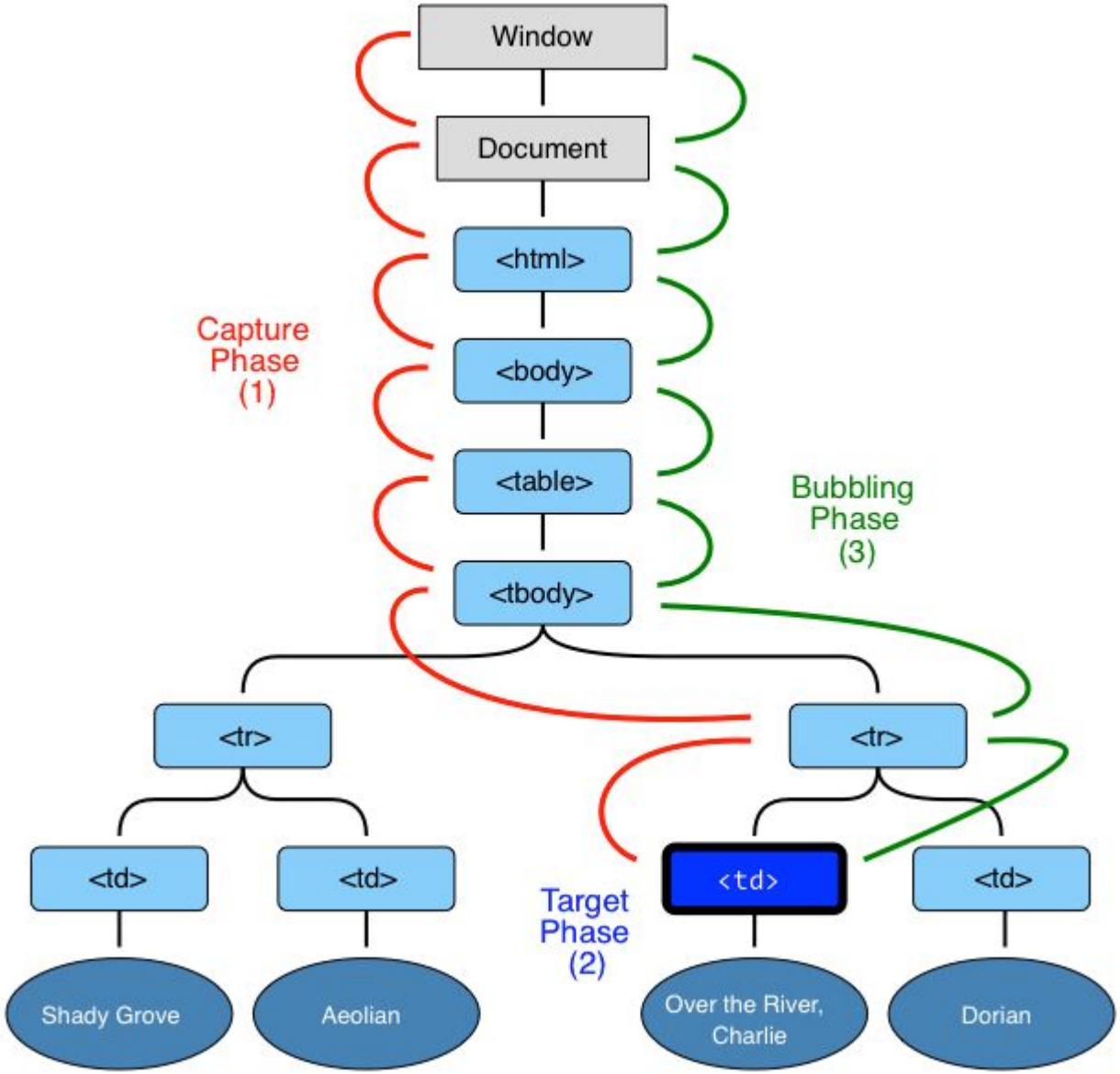
- preventDefault() - отменить стандартное действие браузера, если это конечно возможно.
- stopPropagation() - предотвратить всплытие события (пузырька)

- event.target – это **исходный элемент**, на котором произошло событие, в процессе всплытия он неизменен.
- this – это **текущий элемент**, до которого дошло всплытие, на нём сейчас выполняется обработчик.



Погружение

- Стандарт выделяет целых три стадии прохода события:
  - Событие сначала идет сверху вниз. Эта стадия называется *«стадия перехвата»* (capturing stage).
  - Событие достигло целевого элемента. Это – *«стадия цели»* (target stage).
  - После этого событие начинает всплывать. Это – *«стадия всплытия»* (bubbling stage).


Window
Document
<html>
<body>
<table>
<tbody>
<tr>
<tr>
<td>
<td>
<td>
<td>
Shady Grove
Aeolian
Over the River, Charlie
Dorian
Capture Phase (1)
Target Phase (2)
Bubbling Phase (3)

- Чтобы поймать событие на стадии перехвата, нужно использовать третий аргумент addEventListener:
  - Если аргумент true, то событие будет перехвачено по дороге вниз.
  - Если аргумент false, то событие будет поймано при всплытии.

Делегирование событий

- Один обработчик на их общего предка. Из него можно получить целевой элемент event.target, понять на каком именно потомке произошло событие и обработать его.
- Алгоритм:
  - Вешаем обработчик на контейнер.
  - В обработчике: получаем event.target.
  - В обработчике: если event.target или один из его родителей в контейнере (this) – интересующий нас элемент – обработать его.
- https://learn.javascript.ru/event-delegation