







Frontender[1.0] JavaScript - DOM. События (ивенты). addEventListener(). Объект Event. Mouse, Window, Browser

 YouTube	https://youtu.be/stSxJZyKqKQ
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	32



полезные ссылки:

<https://learn.javascript.ru/introduction-browser-events#obekt-sobytiya>

<https://learn.javascript.ru/mouse-events-basics#otklyuchaem-vydelenie>

https://www.w3schools.com/jsref/dom_obj_event.asp



Событие – это сигнал от браузера о том, что что-то произошло.

Типы событий (некоторые из них):

- События мыши: `click`, `dblclick`, `mousedown`, `mouseup`, `mousemove`, `mouseover`, `mouseout`.
- События клавиатуры: `keydown`, `keyup`, `keypress`.
- События формы: `submit`, `change`, `focus`, `blur`.
- События окна: `resize`, `scroll`, `load`, `unload`.

Событий бывает много: https://www.w3schools.com/jsref/dom_obj_event.asp

Событию можно назначить **обработчик**, то есть функцию, которая сработает, как только событие произошло.

Именно благодаря обработчикам

JavaScript может реагировать на действия пользователя.

В основном все события в HTML:

```
<body>
  <button id="btn" onclick="click me">click me</button>
</body>
</html>
```

- `oncanplay`
- `oncanplaythrough`
- `onchange`
- `onclick`
- `oncontextmenu`
- `ondblclick`
- `ondrag`
- `ondragend`
- `ondragenter`
- `ondragleave`
- `ondragover`
- `ondragstart`
- `ondrop`
- `ondurationchange`
- `onemptied`
- `onended`
- `onerror`
- `onfocus`
- `onformchange`

Но также есть ряд чисто

браузерных событий. Список ниже для общего ознакомления и как шпаргалка:

События браузера

События, относящиеся именно к браузеру, часто связаны с состоянием документа, окна, элементов управления, или взаимодействием браузера с пользователем и сетью. Вот список некоторых чисто браузерных событий:

1. События документа и окна

- `DOMContentLoaded` — когда HTML был загружен и обработан, без ожидания загрузки стилей и изображений.
- `load` — когда все ресурсы страницы (включая изображения, стили и т.д.) были полностью загружены.
- `beforeunload` — перед тем, как пользователь покинет страницу (например, закрывает вкладку или переходит на другую страницу).
- `unload` — когда пользователь покинул страницу.

- `resize` — изменение размеров окна браузера.
- `scroll` — прокрутка страницы.
- `hashchange` — изменение части URL после символа `#` (hash).
- `popstate` — изменение состояния истории браузера (например, при навигации по истории назад или вперед).

2. События сети и соединений

- `online` — браузер переходит в режим онлайн (восстановление сети).
- `offline` — браузер переходит в режим оффлайн (сеть недоступна).
- `storage` — изменения в локальном хранилище (`localStorage` или `sessionStorage`) в другом контексте (вкладке).

3. События на устройствах

- `deviceorientation` — изменение ориентации устройства (например, поворот телефона).
- `devicemotion` — движение устройства, включающее ускорение и поворот.
- `orientationchange` — изменение ориентации экрана (например, между альбомной и портретной ориентацией).
- `pointerlockchange` — когда указатель мыши захвачен (например, в полноэкранных играх).
- `fullscreenchange` — переключение между полноэкранным и обычным режимом.

4. События, связанные с сетью и ресурсами

- `error` — ошибка при загрузке ресурса (изображения, скрипта и т.д.).
- `abort` — загрузка ресурса была отменена.
- `progress` — обновление прогресса при загрузке ресурса.
- `timeout` — сетевой запрос истёк по времени.
- `message` — получение сообщения от другого контекста (например, из `iframe` или веб-воркера).

5. События связанных с браузерными API

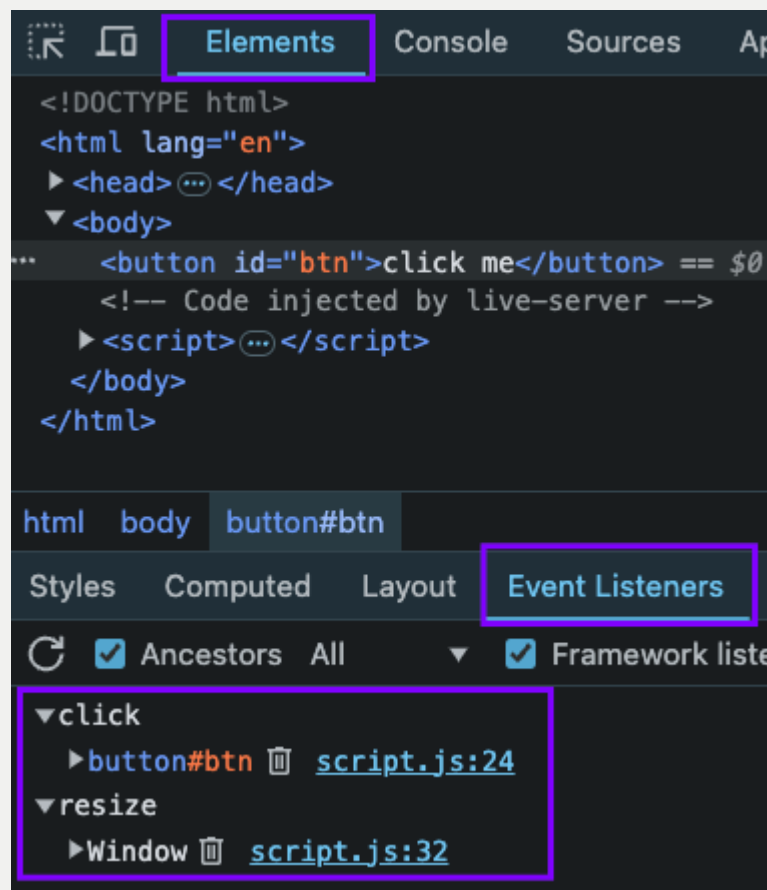
- `clipboard` — события, связанные с буфером обмена (например, копирование, вставка).
- `visibilitychange` — изменение видимости страницы (например, когда пользователь переключается на другую вкладку).
- `dragstart` / `dragend` — события перетаскивания элементов на странице.
- `pointerdown` / `pointerup` — события указателя (объединяет события мыши, касания и стилуса).

6. События WebSocket и другие

- `open` — установлено соединение WebSocket.
- `close` — соединение WebSocket закрыто.
- `message` — сообщение получено через WebSocket.
- `push` — событие `push` в контексте Push API (например, уведомление от сервера).



События можно посмотреть в **devtools**



Чтобы увидеть события на **button** нужно выделить элемент в **elements**

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="script.js" defer></script>
  <title>Events</title>
  <style>
    #btn {
      width: 150px;
      height: 30px;
      /* margin: 1rem; */
    }
    #block {
      width: 150px;
      height: 150px;
      background-color: lightsalmon;
    }
  </style>
</head>

<body>
  <button id="btn">click me</button>
  <div id="block">block</div>
  <hr>
  <input type="text" id="textField">
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    Impedit nam, inventore dolorum culpa laudantium odio doloremque?
    Laborum eum deleniti soluta, debitis fugiat sapiente et dolorem recusandae ipsam exerci
tationem assumenda quos?
```

```
    </p>
  </body>

</html>
```

```
// Events - события

const log = console.log;

const btn = document.querySelector("#btn");
const block = document.querySelector("#block");
const input = document.querySelector("#textField");
const text = document.querySelector("p");

log(btn);

// btn.onclick = () => log("кнопка была нажата");
/*
  element.onclick (любой другой event) - не использовать для добавления событий
  это не удобный способ для событий и есть большой минус что нельзя добавлять
  несколько обработчиков на одно событие, например:

  btn.onclick = () => log("1")
  btn.onclick = () => log("2")
  .....
*/

// 💡----- addEventListener -----💡

// ВАЖНО - название события без приставки "on"

const BtnListener = () => {
  log("кнопка была нажата 1");
};

btn.addEventListener("click", BtnListener)
// btn.addEventListener("click", () => {
//   log("кнопка была нажата 2");
// })

block.addEventListener("click", () => {
  log("клик по блоку");
})

input.addEventListener("input", () => {
  log("пользователь что-то ввел");
})

// вешается именно на window, не на документ
window.addEventListener("resize", () => {
  log("изменился размер окна 1");
})

// 💡----- removeEventListener -----💡

block.addEventListener("click", () => {
```

```

    btn.removeEventListener("click", constBtnListener);
  })

  // 💎----- Объект Event -----💎

  /*
  Когда происходит событие, браузер создаёт объект события,
  записывает в него детали и передаёт его в качестве аргумента
  функции-обработчику.
  */

  input.addEventListener("input", (e) => {
    // log(e);
    // log(e.type);
    // log(e.target);
    // log(e.target.nodeName);
    log(e.target.value);
  });

  btn.addEventListener("click", (e) => {
    log("X:", e.clientX, "Y:", e.clientY);
    log("X:", e.offsetX, "Y:", e.offsetY);
    log(e)
  });

  window.addEventListener("resize", (e) => {
    log(e.target.innerWidth, e.target.innerHeight)
  })

  text.addEventListener("copy", (e) => {
    e.preventDefault();
    log("КОПИРОВАНИЕ ЗАПРЕЩЕНО! 😬");
  })

  window.addEventListener("beforeunload", (e) => {
    e.preventDefault();
  })

```