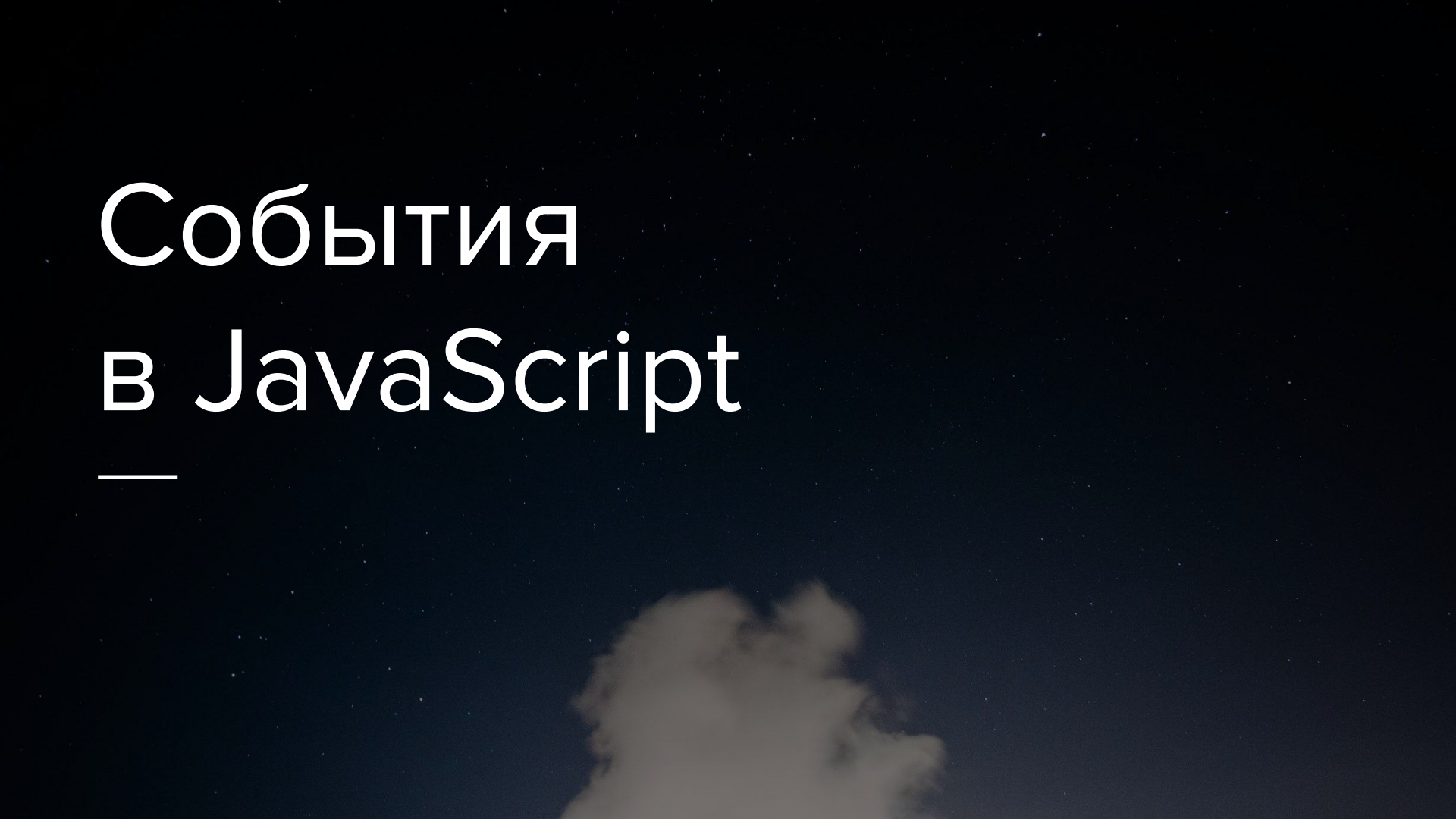
События в JavaScript



События

Для реакции на действия посетителя существуют события.

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

Типы событий:

1. события мыши
2. клавиатурные события
3. события элементов управления (формы)
4. события документа

События мыши

1) **mousedown** - кнопка мыши нажата над элементом;
2) **mouseup** - кнопка мыши отпущена над элементом;
3) **mouseover** - мышь появилась над элементом;
4) **mouseout** - мышь ушла с элемента;
5) **mousemove** - каждое движение мыши над элементом;
6) **click** - вызывается при клике мышью;
7) **contextmenu** - клик правой кнопкой мыши на элементе;
8) **dblclick** - вызывается при двойном клике по элементу.

События клавиатуры

1) **keydown** - возникает при нажатии клавиши клавиатуры и длится, пока нажата клавиша;
2) **keyup** - возникает при отпускании клавиши клавиатуры;
3) **keypress** - возникает при нажатии клавиши клавиатуры, но после события keydown и до события keyup.
   Данное событие генерируется только для тех клавиш, которые формируют вывод в виде символов. Нажатия на остальные клавиши, например, на Alt, не учитываются.

События формы

1) **submit** – посетитель отправил форму <form>;
   При отправке формы путём нажатия Enter на текстовом поле, на элементе <input type="submit"> генерируется событие click.

2) **focus** – посетитель фокусируется на элементе (например, input type=text);
3) **blur** – когда фокус исчезает (посетитель кликает на другом месте экрана);
   Любой элемент поддерживает фокусировку, если у него есть tabindex:
   - ❖ tabindex="0" - делает элемент всегда последним.
   - ❖ tabindex="-1" означает, что клавиша Tab будет элемент игнорировать.
4) **input** - срабатывает тут же при изменении значения текстового элемента;
5) **change** - происходит по окончании изменения значения элемента формы, когда это изменение зафиксировано.

События документа

Процесс загрузки HTML-документа, условно, состоит из трёх стадий.
На каждую можно повесить обработчик:

1) **DOMContentLoaded** – браузер полностью загрузил HTML и построил DOM-дерево.
2) **load** – браузер загрузил все ресурсы.
3) **beforeunload/unload** – уход со страницы.

Прокрутка

Событие **scroll** происходит, когда элемент прокручивается.

Области применения scroll:

- Показ дополнительных элементов навигации при прокрутке.
- Подгрузка и инициализация элементов интерфейса, ставших видимыми после прокрутки.

Обработка события

Способы назначения обработчиков событий:

1. как атрибут тега - **on<событие>="действие"**
2. как свойство DOM объекта **elem.on<событие> = function() {...};**
3. element.**addEventListener**(event, handler[, phase]);
   element.**removeEventListener**(event, handler[, phase]);

Обработка события (пример)

```
let elem = document.getElementById('elem');

добавление обработчика
elem.addEventListener('click', handlerFunction);

удаление обработчика
(нужно передать параметры, указанные в addEventListener)
elem.removeEventListener('click', handlerFunction);

function handlerFunction() {
    alert('Hello');
}
```

Объект события

Детали произошедшего события (координаты курсора, нажатая клавиша и тд) браузер записывает в **“объект события”**, который передаётся первым аргументом в обработчик.

Для того, чтобы посмотреть информацию о событии, надо обратиться к этому объекту.

Объект события (пример)

```
let div = document.getElementById('mouseEvents');

div.addEventListener('contextmenu', showEventObj);

function showEventObj (event) {

    event.preventDefault();  // отмена действия браузера - выделение
                                текста по двойному клику и тд)
    console.log(event);  // посмотреть информацию о событии
}
```

Порядок возникновения событий

События могут возникать:

1) по очереди;
2) по несколько событий сразу: mousedown → mouseup → click (при клике на элементе);
3) во время обработки одного события могут возникать другие.

Когда происходит **событие** (если оно инициируется пользователем), оно **попадает в очередь**.
Внутри браузера непрерывно работает цикл, который следит за состоянием очереди и обрабатывает события.

Каждое событие из очереди можно обработать отдельно от других.

Всплытие

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

Элемент, на котором произошло событие (например, на каком из вложенных элементе был клик) можно узнать с помощью **event.target**

Для **остановки всплытия** нужно вызвать метод **event.stopPropagation()**.

Перехват

При клике на вложенный элемент **событие путешествует** по цепочке родителей (начиная с того, на котором висит обработчик) **сначала вниз к элементу** («погружается»), а **потом наверх** («всплывает»), по пути **задействуя обработчики**.

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

Если в html есть **много элементов, события на которых нужно обрабатывать похожим образом**, то вместо того, чтобы назначать обработчик каждому –нужно **поставить один обработчик на их общего предка.**

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