Лекция № 6. Введение в события

УЧЕБНЫЕ ВОПРСЫ

1. Событие и обработчик события
2. Способы добавления обработчиков событий

Вопрос №1. Событие и обработчик события

Событие – это определённый сигнал от браузера. Он сообщает нам о том, что что-то произошло, например, щелчок мыши, нажатие клавиши на клавиатуре, изменение размера области просмотра, завершение загрузки документа и т.д.

При этом **сигнал** всегда **связан** с объектом. Подавать сигналы могут различные объекты: window, document, DOM-элементы и т.д.

Список некоторых событий и их название:

- **DOMContentLoaded** – завершение загрузки DOM;
- **click** – клик (нажатие левой кнопки мыши, на устройствах с сенсорным управлением возникает при касании);
- **keydown** – нажатие клавиши на клавиатуре;
- **resize** – изменение размеров документа;
- **change** – окончание изменения значения в поле ввода.

Для того чтобы **реагировать** на события, т.е. **выполнять определённые действия**, когда они произойдут, необходимо **привязать** некоторую **функцию** к событию.

После этого эта функция будет вызываться всякий раз, когда это событие на указанном элементе будет возникать. Эту функцию в JavaScript принято называть **обработчиком события**.

Обработчик события – это функция, которая вызывается при возникновении события.

Следует отметить, что на странице постоянно возникает огромное количество событий, независимо от того назначили ли мы им обработчик или нет.

Задача разработчика – это заставить сайт или приложение реагировать **только на те события**, которые ему необходимы.

Вопрос №2. Способы добавления обработчиков событий

Назначить обработчик событию можно разными способами:

- через HTML-атрибут on{событие} (не является хорошей практикой);
- посредством свойства DOM-элемента on{событие};
- используя специальный метод addEventListener.

Инициализация обработчика через атрибут

Этот способ позволяет прописать обработчик **напрямую в разметке**. Выполняется это посредством **указания JavaScript кода** в атрибуте on{событие}. Вместо {событие} необходимо написать имя (тип) события (например: click).

Пример

```
<!-- onclick - атрибут, содержащий код, который будет выполняться всякий раз при наступлении события click на этом элементе -->
<button type="button" onclick="alert(Date())">Текущая дата</button>
```

Инициализация обработчика через атрибут

Если код, который нужно поместить в атрибут достаточно большой, то в этом случае его лучше оформить в виде функции, а в атрибут поместить её вызов

```
<button type="button" onclick="sum()">Посчитать</button>
```

При этом задавать обработчик напрямую в разметке не является хорошей практикой, т.к. это приведёт к **смешиванию JavaScript и HTML кода**.

Добавление обработчика через свойство DOM объекта

Второй способ назначить обработчик - это использовать свойство on{событие}.

Например, назначив свойству анонимную функцию

```
$btn.onclick = function() {
  alert('Вы кликнули на кнопку!');
}
```

или существующую функцию

```
document.onclick = changeBgColor;
```

Добавление обработчика через свойство DOM объекта

Внутри обработчика можно **обратиться** к текущему элементу, т.е. к тому для которого в данный момент был вызван этот обработчик. Осуществляется это с помощью ключевого слова **this**.

Пример

```
function message() {
 // this - обращаемся к кнопке для которой вызван обработчик
  alert(this.textContent);
}
```

Подписка на событие через addEventListener

Ещё один способ назначить событию обработчик — это использовать метод addEventListener.

Синтаксис addEventListener:

// $element - объект или DOM-элемент к которому нужно добавить обработчик

```
$element.addEventListener(event, handler[, options]);
```

Параметры:

- **event** - имя события (например, click);
- **handler** - функция, которая будет вызвана при возникновении этого события;;
- **options** (не обязательный) - объект, в котором можно задать дополнительные параметры.

Для прикрепления обработчиков к элементам, необходимо чтобы эти элементы на странице **были доступны**. Определить, когда они будут доступны с момента загрузки документам можно с помощью события **DOMContentLoaded**.

Данное событие возникает на document когда DOM полностью построено:

```
document.addEventListener('DOMContentLoaded', function () {
  // DOM полностью построен и доступен
  ...
});
```