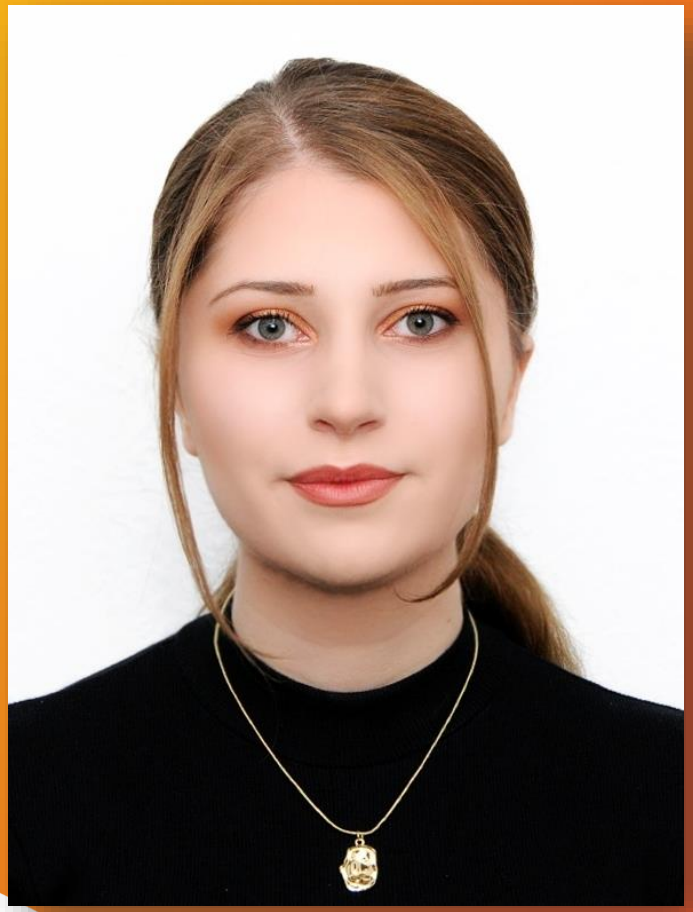
TEL-RAN
by Starta Institute

Frontend Basic
LESSON 12

Anna Khachaturyan



TEL-RAN
by Starta Institute

Анна Хачатурян

Front End/Gen Tech Teacher

- Since 2018 in IT
- Full Stack Developer at Web Magnat
- QA Engineer/Web Developer at Central Bank of RA
- Lecturer at Plekhanov Russian University of Economics
- TA at Picsart Academy
- Teacher at Tel-Ran

https://t.me/anny_khachaturyan

TEL-RAN
by Starta Institute

ВАЖНО:

- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение изученного
2. Вопросы по повторению
3. DOM – работа с браузерными событиями
4. DOM – работа со стилями
5. Практика
6. CallBack функции

TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ

TEL-RAN
by Starta Institute

2 ВОПРОСЫ ПО ПОВТОРЕНИЮ

TEL-RAN
by Starta Institute

3

DOM – РАБОТА С БРАУЗЕРНЫМИ СОБЫТИЯМИ

Введение в браузерные события

Событие – это сигнал от браузера о том, что что-то произошло. Все DOM-узлы подают такие сигналы.

Вот список самых часто используемых DOM-событий, пока просто для ознакомления:

<u>События мыши:</u>

- click – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).
- contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши.
- mouseover / mouseout – когда мышь наводится на / покидает элемент.
- mousedown / mouseup – когда нажали / отжали кнопку мыши на элементе.
- mousemove – при движении мыши.

<u>События на элементах управления:</u>

- submit – пользователь отправил форму <form>.
- focus – пользователь фокусируется на элементе, например нажимает на <input>.

<u>Клавиатурные события:</u>

- keydown и keyup – когда пользователь нажимает / отпускает клавишу.

Обработчики событий

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

Именно благодаря обработчикам JavaScript-код может реагировать на действия пользователя.

addEventListener, removeEventListener

Метод **addEventListener**() присоединяет обработчик события к определенному элементу. При этом новый обработчик события не переписывает уже существующие обработчики событий.

Таким образом, вы можете добавлять сколько угодно обработчиков событий к одному элементу. При этом это могут быть обработчики событий одного типа, например, два события нажатия мышкой.

Вы можете добавлять обработчики событий к любому объекту DOM, а не только к HTML элементам, например, к объекту окна.

Метод addEventListener() позволяет легко контролировать то, как обработчик реагирует на, так называемое, "всплывание" события.

Когда используется метод addEventListener(), JavaScript отделяется от разметки HTML, что улучшает читаемость скрипта и позволяет добавлять обработчики событий даже тогда, когда вы не можете контролировать разметку HTML.

Синтаксис:

элемент.addEventListener(событие, функция, useCapture);

Первый параметр — тип события (например, "click" или "mousedown").

Второй параметр — функция, которая будет вызываться при возникновении события.

Третий параметр — логическое значение (true/false), определяющее следует ли отправить событие дальше ("всплывание") или нужно закрыть это событие. Этот параметр необязателен.

В следующем примере при нажатии пользователем на элемент появляется окно с сообщением "Hello World!":

```
элемент.addEventListener("click", function(){ alert("Hello World!"); });
```

Также, можно задать и внешнюю "именованную" функцию:

```
элемент.addEventListener("click", myFunction);

function myFunction() {
    alert ("Hello World!");
}
```

Метод addEventListener() позволяет добавлять несколько обработчиков событий к одному и тому же элементу не переписывая уже существующие обработчики событий:

```
элемент.addEventListener("click", myFunction);
элемент.addEventListener("click", mySecondFunction);
```

Также, можно добавлять обработчики событий разных типов:

```
элемент.addEventListener("mouseover", myFunction);
элемент.addEventListener("click", mySecondFunction);
элемент.addEventListener("mouseout", myThirdFunction);
```

Для **удаления** обработчика следует использовать **removeEventListener**:

Синтаксис:

элемент. removeEventListener(событие, функция, useCapture);

Удаление требует именно ту же функцию

Для удаления нужно передать именно ту функцию-обработчик, которая была назначена.

```
function handler() {
    alert( 'Спасибо!' );
  }

  input.addEventListener("click", handler);
  // ....
  input.removeEventListener("click", handler);
```

Обратим внимание – если функцию обработчик не сохранить где-либо, мы не сможем её удалить. Нет метода, который позволяет получить из элемента обработчики событий, назначенные через addEventListener.

Где можно найти список событий JS:

https://developer.mozilla.org/ru/docs/Web/Events

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

TEL-RAN
by Starta Institute

4 DOM – РАБОТА СО СТИЛЯМИ

Свойство Style

До того, как начнём изучать способы работы со стилями и классами в JavaScript, есть одно важное правило. Надеемся, это достаточно очевидно, но мы всё равно должны об этом упомянуть.

Как правило, существует два способа задания стилей для элемента:

- Создать класс в CSS и использовать его:
- Писать стили непосредственно в атрибуте style: .

JavaScript может менять и классы, и свойство style.

Element style

Свойство elem.style – это объект, который соответствует тому, что написано в атрибуте "style". Установка стиля elem.style.width="100px" работает так же, как наличие в атрибуте style строки width:100px.

Для свойства из нескольких слов используется camelCase:

```
let button = document.querySelector(".click");
button.style.backgroundColor = "black";
button.style.color = "white";
button.style.width = "100px";
button.style.height = "60px";
button.style.fontSize = "20px";
```

Сброс стилей

Иногда нам нужно добавить свойство стиля, а потом, позже, убрать его.

Например, чтобы скрыть элемент, мы можем задать elem.style.display = "none".

Затем мы можем удалить свойство style.display, чтобы вернуться к первоначальному состоянию. Вместо delete elem.style.display мы должны присвоить ему пустую строку: elem.style.display = "".

TEL-RAN
by Starta Institute

5 ПРАКТИКА

Задание: Повесить событие клика на кнопку таким образом, чтобы при нажатии на нее в консоль выводилась строка “Hello world”.

Решение:

```
<body>
    <button class="click">Click!</button>
    <script src="script.js"></script>
</body>
```

```
let button = document.querySelector(".click");

function onclick(){
    console.log('Hello world!');
}

button.addEventListener("click", onclick);
```

Задание: Переделать кнопку таким образом, чтобы при нажатии на нее в параграфе в интерфейсе значение увеличивалось на 1. В начале рассмотрите процесс, при котором мы каждый раз считываем число из параграфа, меняем и записываем обратно, а потом переделайте процесс так, чтобы значение хранилось в переменной, изменялось при клике и записывалось в параграф.

Решение:
Первый этап:

```
let button = document.querySelector(".click");

function onclick(){
    document.querySelector('.num').innerText = Number(document.querySelector('.num').innerText) + 1;
}

button.addEventListener("click", onclick);
```

Второй этап:

```
let button = document.querySelector(".click");
let num = document.querySelector('.num');

function onclick(){
    num.innerText = Number(num.innerText) + 1;
}

button.addEventListener("click", onclick);
```

Задание: Написать программу, которая создает две кнопки и вешает на них событие нажатия. При нажатии на первую выводится в консоль “минус”, а при нажатии на вторую ‘плюс’.

Решение:

```
let button1 = document.createElement('button');
button1.innerText = "Plus";
button1.classList.add("plus");

let button2 = document.createElement('button');
button2.innerText = "Minus";
button2.classList.add("minus");
document.body.append(button1, button2);

function plus(){
    console.log("+");
}

function minus(){
    console.log("-");
}

button1.addEventListener('click', plus);
button2.addEventListener('click', minus);
```

Задание: Доработать процесс таким образом, чтобы при нажатии на плюс в консоль выводилось число, увеличившись на 1, а при нажатии на минус уменьшившись на 1. Для решения этой задачи понадобится создать переменную и изменять ее при нажатии на ту или иную кнопку.

Решение:

```
let num = 5;

function plus(){
    console.log(++num);
}

function minus(){
    console.log(--num);
}

button1.addEventListener('click', plus);
button2.addEventListener('click', minus);
```

Попробовать так же изменить значение параграфа.

Задание: Создать квадратный div с рамкой и при наведении курсора на него цвет div-а поменять на введённый пользователем цвет (все шаги сделать с помощью JS).

```
function changecolor(){
    let newcolor = prompt("Enter the new color for block");
    block.style.backgroundColor = newcolor;
}

let block = document.createElement('div');
block.style.width = "100px";
block.style.height = "100px";
block.style.border = "1px solid black";
block.style.backgroundColor = "black";
block.addEventListener("mouseover", changecolor)
document.body.append(block);
```

TEL-RAN
by Starta Institute

6 CALLBACK ФУНКЦИИ

CallBack функция

Простыми словами: коллбэк — это функция, которая должна быть выполнена после того, как другая функция завершила выполнение (отсюда и название: callback — функция обратного вызова).

Чуть сложнее: В JavaScript функции — это объекты. Поэтому функции могут принимать другие функции в качестве аргументов, а также возвращать функции в качестве результата. Функции, которые это умеют, называются функциями высшего порядка. А любая функция, которая передается как аргумент, называется callback-функцией.

```
function myFunc(callback){
    let a = [4, 5, 6]; // список
    let element = document.querySelector('.p1'); //параграф
    callback(element, a); // вызов функции out, с аргументами element(параграф), a(список).
}

function out(elem, arr){
    elem.innerText = arr;
}

myFunc(out); // вызов функции с аргументом функции (out)
```

WISHING YOU ALL THE

GOOD LUCK

IN THE WORLD!