JS

Лабораторная работа 7.

Создадим папку lr7. Откроем ее в vsCode (файл – открыть папку…). Это будет проектом. Создадим в нем файлы Index.html и script.js.

Index.html

<html>

<head>

<meta charset="UTF-8">

    <script src="script.js"></script>

<title>Лабораторная работа 7</title>

</head>

<body>

    <form class="new\_class" name="my\_name" method="get" action="">

        <div>

            <input type="text" id="textfield1" name="textfield1" value="" size="18" />

            <input type="radio" name="radio" />

            <input type="radio" name="radio" />

            <input type="radio" name="radio" />

            <input type="submit" id="submit1" name="submit1" value="Поиск" />

        </div>

    </form>

</body>

</html>

**Выполнение.**

1. События дают возможность получить уведомление (в коде) о том, что произошло какое-либо действие.

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

События можно разделить на несколько групп:

1.1. События мыши

.**onclick** – событие, срабатывающее при клике

window.onload = function() {

    let myEdit = document.forms.my\_name.textfield1;

    myEdit.onclick = function() {

        console.log(myEdit.value);

    }

}

Событие onclick может передавать объект MouseEvent

window.onload = function() {

    let myEdit = document.forms.my\_name.textfield1;

    myEdit.onclick = function(event) {

        console.log(event);

    }

}

e.altKey – признак, была ли зажата клавиша alt

e.ctrlKey – признак, была ли зажата клавиша ctrl

e.shiftKey – признак, была ли зажата клавиша shift

e.clientX / e.clientY - отображение координат курсора мыши в локальной системе координат (DOM контент).

e.offsetX / e.offsetY - отображение координат курсора мыши относительно целевого узла.

<https://developer.mozilla.org/ru/docs/Web/API/MouseEvent>

.**onmousemove** срабатывает при проведении курсора мыши по объекту. Передает объект MouseEvent.

window.onload = function() {

    let myForm = document.forms.my\_name;

    myForm.onmousemove = function(event) {

        console.log(event);

    }

}

.**onmousedown** / .**onmouseup** – срабатывают при нажимании клавиши мыши и при отпускании. Передает объект MouseEvent

.**oncontextmenu** – срабатывают при выводе контекстного меню. Передает объект MouseEvent.

Если в функции обработчике вернуть false, то не будет отображено стандартное контекстное меню.

window.onload = function() {

    let myForm = document.forms.my\_name;

    myForm.oncontextmenu = function(e) {

        return false;

    }

}

.**onmouseover** / .**onmouseout** – срабатывает, когда курсор мыши заходит / выходит из области элемента. Передает объект MouseEvent.

.**onmousewheel** – срабатывает при прокрутке колесом мыши. Передает объект WheelEvent.

1.2. События клавиатуры

.**onkeypress** – событие срабатывает при нажатии клавиши на клавиатуре. Передает объект KeyboardEvent

window.onkeypress = function (event) {

    console.log(event);

}

<https://developer.mozilla.org/ru/docs/Web/API/KeyboardEvent>

1.3. События объекта window

**window.onload** – событие после полной загрузки страницы

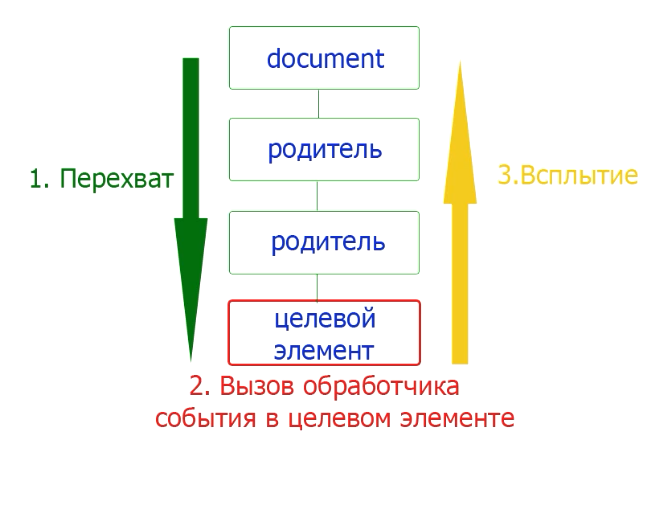
window.onload = function() {

}

2. Обработка события делится на:

* перехват – перехват обработчиком события, но срабатывание не происходит;
* обработка – вызов обработчика события в целевом элементе;

всплытие срабатывание обработчиков начинается с самого нижнего по иерархии элемента.



Обработку событий можно остановить при помощи метода event.stopPropagation() <https://developer.mozilla.org/ru/docs/Web/API/Event/stopPropagation>

Некоторые события осуществляют действия по умолчанию. Например, клик правой кнопки мыши открывает контекстное меню. Для того, чтобы отменить ействие по умолчанию можно воспользоваться методом event.preventDefault()

<https://developer.mozilla.org/ru/docs/Web/API/Event/preventDefault>

3. Задать обработчики событиям можно несколькими способами:

* в html документе, например

<input type="text" onchange="myFunc()" />

function myFunc(obj) {

    console.log(obj.value);

}

* в скрипте, получив элемент и передав ссылку на функцию в событие, например

    let myEdit = document.forms.my\_name.textfield1;

    myEdit.onclick = function() {

        console.log(myEdit.value);

    }

* при помощи метода addEventListener

addEventListener("тип события", функция - обработчик , перехват или всплытие);

*Обратите внимание, что тип события в данном случае пишется без приставки on.*

<span class="subtext" id="my\_id">

    Нажми меня

</span>

window.onload = function () {

    let block = document.getElementById('my\_id');

    block.addEventListener('click', func, false)

    block.addEventListener('click', func1, false)

}

function func() {

    console.log('Первый обработчик');

}

function func1() {

    console.log('Второй обработчик');

}

4. Пример

index.html

<html lang="ru">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>События</title>

<link rel="stylesheet" href="style.css" />

<script src="script.js"></script>

</head>

<body>

<div class="message" id="message">

<p>Ничего не выбрано</p>

</div>

<div class="field" id="field"></div>

</body>

</html>

style.css

body {

margin: 50*px* 10*%* 0*px* 10*%*;

}

*.message* {

width: 100*%*;

border: 1*px* solid gray;

border-radius: 5*px*;

margin-bottom: 5*px*;

text-align: center;

}

*.field* {

display: grid;

grid-gap: 5*px*;

grid-template-columns: repeat(5, 1*fr*);

grid-auto-rows: minmax(60*px*, auto);

}

*.item* {

align-items: center;

justify-content: center;

border: 1*px* solid gray;

border-radius: 5*px*;

cursor: pointer;

display: flex;

}

*.item:nth-child*(odd) {

background: cornsilk;

}

script.js

*const* *items* *=* *[*

*{* *key: 1, caption:* 'Поле № 1' *},*

*{* *key: 2, caption:* 'Поле № 2' *},*

*{* *key: 3, caption:* 'Поле № 3' *},*

*{* *key: 4, caption:* 'Поле № 4' *},*

*{* *key: 5, caption:* 'Поле № 5' *},*

*{* *key: 6, caption:* 'Поле № 6' *},*

*{* *key: 7, caption:* 'Поле № 7' *},*

*{* *key: 8, caption:* 'Поле № 8' *},*

*{* *key: 9, caption:* 'Поле № 9' *},*

*{* *key: 10, caption:* 'Поле № 10' *},*

*{* *key: 11, caption:* 'Поле № 11' *},*

*{* *key: 12, caption:* 'Поле № 12' *},*

*{* *key: 13, caption:* 'Поле № 13' *},*

*{* *key: 14, caption:* 'Поле № 14' *},*

*{* *key: 15, caption:* 'Поле № 15' *}*

*]*;

*let* *field*;

*let* *message*;

*window*.*onload* *=* *function* () {

*field* *=* *this*.*document*.*getElementById*('field');

*message* *=* *this*.*document*.*getElementById*('message');

*initContent*(*field*);

}

*function* *initContent*(elm) {

*for* (*let* *item* *of* *items*) {

*let* *elmItem* *=* *document.createElement(*'div'*)*;

*elmItem*.*setAttribute*('id', `i${*item*.*key*}`);

*elmItem*.*setAttribute*('class', `item`);

*elmItem*.*innerHTML* *=* `<p>${*item*.*key*}</p>`

*elmItem*.*addEventListener*('click', *setMessage*.*bind*(null, *item*))

elm.*append*(*elmItem*);

}

}

*function* *setMessage*(item) {

*message*.*innerHTML* *=* `<p>${item.*caption*}</p>`

}