# Об’єкт event. Обробка подій

*Морозов Андрій Васильович,*
*к.т.н, доц.,*
*декан факультету інформаційно-комп’ютерних технологій ЖДТУ*

Обробники подій у JavaScript можуть бути задані такими способами:

***1) в атрибуті тегу (у HTML):***

```
<тег onподія="код на JavaScript" />
```

*Наприклад:*

```
<input value="Натисніть"
       onclick="myfunc();"
       type="button" />
```

# Наприклад:

```
<!DOCTYPE HTML>
<html>
<head> <script src="script.js"></script> </head>
<body>
    <input type="button" onclick="countNumbers()"
        value="Виконати розрахунок!" id="but" />
    <div id="result"></div>
</body>
</html>
```

**Файл script.js:**

```
function countNumbers() {
    var resElem = document.getElementById("result");
    var res = 0;
    for(var i = 1; i <= 3; i++)
        res += i * 2;
    resElem.innerHTML = res.toString();
}
```

*2) через властивість DOM-елемента (у JS):*

- якщо функція оголошена як FD або FE:

```
elem.onподія = назваФункції;
```

- якщо функція оголошена як анонімна:

```
elem.onподія = function ()
{
```

# Наприклад:

```
<html>
<head> <script src="script.js"></script> </head>
<body>
    <input type="button" value="Виконати" id="but" />
    <div id="result"></div>
</body>
</html>
```

**Файл script.js:**

```
var elem;
window.onload = function() {
    elem = document.getElementById('but');
    elem.onclick = countNumbers;
}
function countNumbers() {
    var resElem = document.getElementById("result");
    var res = 0;
    for(var i = 1; i <= 3; i++)
        res += i * 2;
    resElem.innerHTML = res.toString();
}
```

*3) через перехоплювачі подій:*

*- додавання обробника події:*

```
elem.addEventListener("подія", назваФункції);
```

*- видалення обробника події:*

```
elem.removeEventListener("подія",
                         назваФункції);
```

*Переваги:*

1) *можна призначати кілька обробників події;*
2) *можна оброблювати нестандартні події*

# Наприклад:

```
<html>
<head> <script src="script.js"></script> </head>
<body>
     <input type="button" value="Виконати" id="but" />
     <div id="result"></div>
</body>
</html>
```

**Файл script.js:**

```
var elem;
window.addEventListener("load", function loadDoc() {
     elem = document.getElementById('but');
     window.removeEventListener("load", loadDoc);
     elem.addEventListenter("click", countNumbers);
}
function countNumbers() {
     var resElem = document.getElementById("result");
     var res = 0;
     ...
     resElem.innerHTML = res.toString();
}
```

У кожну функцію-обробник передається першим параметром об’єкт event, який містить інформацію про подію:

```
elem.addEventListener('click', myClick);
function myClick(event) {
 console.log(event.type + " на " + event.currentTarget);
 console.log(event.clientX + ":" + event.clientY);
}
```

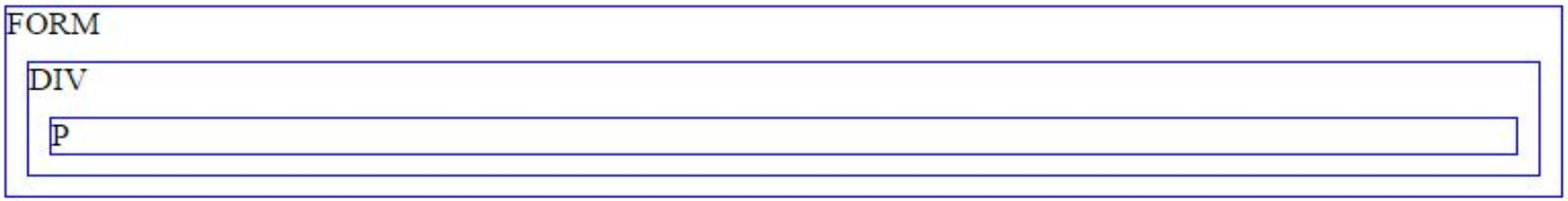
```
function myClick(event) {
    var info = event.поле;
}
```

**Поля об’єкта `event`:**

| Поле | Пояснення |
|---|---|
| **event.type** | назва події (рядок) |
| **event.currentTarget** | DOM-елемент, для якого спрацювала подія (об’єкт) |
| **event.clientX** | координати миші відносно вікна (числа) |
| **event.clientY** | |

**Спливання подій.** При спрацюванні події, вона спочатку спрацьовує для вкладеного тегу, потім для його батька, потім для батька батька і так далі поки не дійде до тегу `html`.

```
<form onclick="alert('form')">FORM
    <div onclick="alert('div')">DIV
        <p onclick="alert('p')">P</p>
    </div>
</form>
```

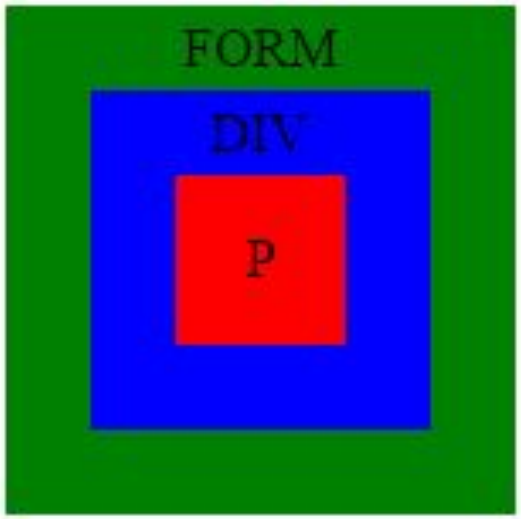



**Подія `click` спрацює для трьох тегів: `p` → `div` → `form`**

## Об’єкт event містить поля:

| Поле | Пояснення |
|---|---|
| **event.currentTarget** | DOM-елемент, для якого спрацювала подія (еквівалентний **this**, в процесі спливання події він змінюється) |
| **event.target** | DOM-елемент, в якому виникла подія (тег, який спричинив подію, в процесі спливання подій він є незмінним) |

**Наприклад:**




```
<form id="form">FORM
  <div>DIV
     <p>P</p>
  </div>
</form>
```

```
var form = document.querySelector('form');
form.onclick = function(event)
{
   event.target.style.backgroundColor = 'yellow';
   alert("target = " + event.target.tagName +
         ", this=" + this.tagName);
   event.target.style.backgroundColor = '';
};
```

4. подія click спрацює для тегу body:
   event.target вказує на p
   event.currentTatget вказує на body

5. подія click спрацює для тегу html:
   event.target вказує на p
   event.currentTatget вказує на html

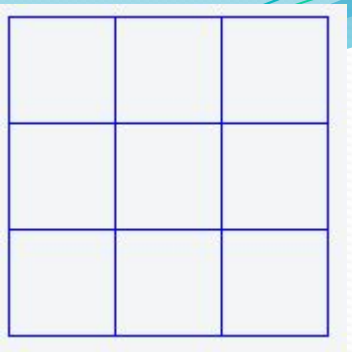
2. подія click спрацює для тегу div:
event.target вказує на p
event.currentTatget вказує на div

3. подія click спрацює для тегу form:
event.target вказує на p
event.currentTatget вказує на form

# Приклад

```
        </tr>
        <tr>
            <td></td>
            <td></td>
            <td></td>
        </tr>

        <tr>
            <td></td>
            <td></td>
            <td></td>
        </tr>
    </table>
```



```
window.addEventListener('load', function loadFunc()
{
    window.removeEventListener('load', loadFunc);
    var elem = document.getElementById('game');
    game.addEventListener('click', function (event)
    {
        var targ = event.target;
        if (targ.tagName == 'TD')
        {
            var curColor = targ.style['backgroundColor'];
            var color = (curColor == 'green')?'white':'green';
            targ.style.backgroundColor = color;
        }
    });
});
```

https://jsfiddle.net/2s7u98do/

**Для відміни спливання події:**

event.stopPropagation();

**Для відміни стандартної дії при настанні події:**

event.preventDefault();

**Події можна генерувати (примусово виконати обробник події):**

```
var event = new Event("click");
elem.dispatchEvent(event);
```

# Приклад відміни стандартної дії при натисканні на посилання

**HTML:**

```
<ul id="menu" class="menu">
    <li><a href="/php">PHP</a></li>
    <li><a href="/html">HTML</a></li>
    <li><a href="/javascript">JavaScript</a></li>
    <li><a href="/flash">Flash</a></li>
</ul>
```

**JavaScript:**

```
var menu = document.getElementById('menu');
menu.addEventListener('click', function(event)
  {
    if (event.target.nodeName != 'A') return;
    var href = event.target.getAttribute('href');
    alert( href );
    event.preventDefault();
  });
```

# Події миші

## Прості:

- **mousedown / mouseup**
- **mouseover / mouseout / mouseenter / mouseleave**
- **mousemove**

## Комплексні:

- **click** — `mousedown → mouseup → click`
- **contextmenu** — `mousedown → mouseup → contextmenu`
- **dblclick** — `mousedown → mouseup → click → mousedown → mouseup → click → dblclick`

Комплексним подіям передує чітко визначений ряд простих подій

При натисканні правою клавішею миші спочатку спрацьовує подія contextmenu, а потім з'являється контекстне меню. Для заборони відображення контекстного меню потрібно використовувати event.preventDefault():

```
<div id="block">
    Натисніть праву кнопку
    миші на цьому тезі
</div>
```

```
window.addEventListener('load', function loadFunc() {
    window.removeEventListener('load', loadFunc);
    document.getelementById('block').addEventListener(
          'contextmenu', function (event) {
        // код
        event.preventDefault();
    });
});
```

# Події миші

***Для подій миші об'єкт event містить такі***

| Поле | Пояснення |
|---|---|
| event.which | отримання натиснутої кнопки миші: 1 – ліва; 2 – середня; 3 – права |
| event.shiftKey<br>event.altKey<br>event.ctrlKey<br>event.metaKey | булеве значення, яке рівне true, якщо одночасно з натисканням клавіші миші була затиснута відповідна клавіша на клавіатурі |
| event.clientX<br>event.clientY | координати миші відносно вікна (без врахування прокрутки) |
| event.pageX<br>event.pageY | координати миші відносно документа (з врахуванням прокрутки) |

# *Події mouseover / mouseout, mouseenter / mouseleave*

| Коли виникає подія | Назва події | Подія спливає ? | Чи можна відслідкувати обидва елементи, між якими був перехід? |
|---|---|---|---|
| при введенні курсору миші в межі тегу | **mouseover** | так | так |
| | **mouseenter** | ні | ні |
| при виведенні курсору миші за межі тегу | **mouseout** | так | так |
| | **mouseleave** | ні | ні |

Для події **mouseover** об’єкт **event** містить поля:

- **event.target** – елемент, на який перейшла миша;
- **event.relatedTarget** – елемент, з якого перейшла миша

Для події **mouseout** об’єкт **event** містить поля:

- **event.target** – елемент, з якого перейшла миша;
- **event.relatedTarget** – елемент, на який перейшла миша

Якщо миша перейшла з-за меж вікна (перейшла за вікно), то **event.relatedTarget == null**

# *Приклад.*

***JavaScript:***

```
window.onload = function() {
    table.onmouseover = function(event) {
        var target = event.target;
        target.style.background = 'pink';
    };

    table.onmouseout = function(event) {
        var target = event.target;
        target.style.background = '';
    };
}
```

**Пояснення**: події, які виникають у вкладених у **table** тегах, спливають і перехоплюються
у тезі **table** обробниками подій **onmouseover** і **onmouseout.** Це дає можливість не встановлювати обробники подій для
для таблиці.

https://jsfiddle.net/morozovandriy/zxngrh8d/1/

# *Приклад.*


**Проблема:**
**при переході курсору миші на тег, розміщений всередині комірки**

**при введенні курсору миші у комірку**

https://jsfiddle.net/morozovandriy/zxngrh8d/1/

# *Приклад.*

Проблему можна вирішити, відслідковуючи, в який саме тег відбувається перехід. Якщо перехід здійснюється в тег, який знаходиться у поточній комірці, то не треба нічого робити.

```
window.onload = function() {
    table.onmouseover = function(event) {
        var target = event.target;
        while (target != this) {
            if (target.tagName == 'TD') break;
            target = target.parentNode;
        }
        if (target == this) return;
        target.style.background = 'pink';
    };
    table.onmouseout = function(event) {
        var target = event.target;
        while (target != this) {
            if (target.tagName == 'TD') break;
            target = target.parentNode;
        }
        if (target == this) return;
        target.style.background = '';
    };
}
```

https://jsfiddle.net/morozovandriy/ob6cakzw/1/

# *Події клавіатури*

- **keydown** – при натисканні клавіші
- **keyup** – при відпусканні клавіші
- **keypress** – при відпусканні клавіші для зчитування введеного символу

***Послідовність*** спрацьовування подій:
**keydown** → **keypress** → **keyup**

***Особливості:***

- події **keydown** / **keyup** дозволяють відслідкувати яка клавіша або яка комбінація клавіш була натиснута;
- подія **keypress** дозволяє отримати введений символ;
- в полях форми при виникненні подій **keydown** / **keypress** клавіша ще оброблена браузером (тобто при зверненні до **value** відповідного поля значення буде ще старе).

# *Події клавіатури*

*Об'єкт* **event** *для подій клавіатури містить поля*:

- **event.keyCode** – код кнопки на клавіатурі (не символа, а саме кнопки)
- **event.charCode, event.which** – код введеного символа;
- **event.char –** введений символ

Для подій клавіатури також можна перевіряти, чи була затиснута службова клавіша:

- **event.shiftKey**
- **event.altKey**
- **event.ctrlKey**
- **event.metaKey**

# Події клавіатури

**event.keyCode** для кнопки клавіатури можна отримати з сайту:

**http://keycode.info/ та з сайту**

***Наприклад***, можна заборонити оновлення сторінки при натисканні клавіші F5.

Для цього на сайті http://keycodes.info/ отримуємо код клавіші F5.


```
window.addEventListener('keydown',
    function myKeyDown(event)
    {
        if (event.keyCode == 116)
         event.preventDefault();
    });
```

# *Події клавіатури*

***Приклад***. Блок, що переміщується за допомогою клавіш стрілок.

**HTML-код:**

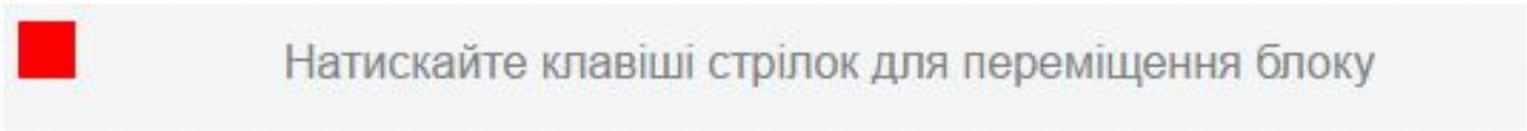
```
<p>Натискайте клавіші стрілок для переміщення блоку</p>
<div id="block"></div>
```

**CSS:**

```
#block { position: absolute; left:10px; top:10px;
width:20px; height:20px; background-color:red; }

p { font: 16px Arial; color: gray; text-align:center; }
```

**Зовнішній вигляд:**



# Події клавіатури

```
var keyCodes = {UP : 38, DOWN : 40, LEFT : 37, RIGHT : 39};
var step = 10;
function moveBlock(dx, dy)
{
    var style = getComputedStyle(block);
    var left = parseInt(style.left);
    var top  = parseInt(style.top);
    block.style.left = left + dx + "px";
    block.style.top  = top  + dy + "px";
}
window.onkeydown = function(event)
{
    switch(event.keyCode)
    {
        case keyCodes.UP:     moveBlock(0, -step); break;
        case keyCodes.DOWN:   moveBlock(0, step); break;
        case keyCodes.LEFT:   moveBlock(-step, 0); break;
        case keyCodes.RIGHT:  moveBlock(step, 0); break;
    }
}
```

https://jsfiddle.net/morozovandriy/gdbg8yaL/