Events

ПОДІЇ

Всі зміни, які відбуваються на Web-сторінки, пов'язані з роботою браузера або маніпуляціями користувача з клавішами миші або клавіатури, називаються подіями. Для вказівки дій, які необхідно зробити в зв'язку з появою того або іншої події, використовуються обробники подій.

Типи подій

Можна виділити декілька груп подій, у залежності від того як вони генеруються. Розглянемо деякі з них

Події документу

load	Закінчено завантаження документа (Frame, Image, Layer)
unload	Користувач залишає сторінку, наприклад, завантажуючи інший документ (Frame, window)
resize	Користувач змінив розмір вікна (Frame, window)
error	Виникла помилка підчас завантаження (Image, window)
move	Користувач перемістив вікно (Frame, window)

`DOMContentLoaded` – коли HTML завантажений і опрацьований, DOM документу повністю побудований і доступний

Події миші

click	Користувач натискає кнопку миші (Button, Checkbox, document, Link, Radio, Reset, Submit , і т.д. – майже всі об'єкти
mousedown	Користувач натиснув кнопку миші. (Button, document, Link)
mouseup	Користувач відпустив конпку миші (Button, document, Link)
dblclick	Кристувач двічі натискає на кнопку миші (Button, Checkbox, document, FileUpload, Hidden, Link, Password, Radio, Reset, Select, Submit)
dragdrop	Користувач перетягує об'єкт у вікно
mousemove	Користувач перемістив курсор миші (Виникає лише при явному заданні обробника).
mouseout	Користувач перемістив курсор за межі об'єкта(Area, Layer, Link)
mouseover	Користувач помістив курсор миші над об'єктом (Area, Layer, Link)

contextmenu – відбувається при виклику констекстного меню (права кнопка миші)

Події клавіатури

keydown	Користувач натиснув клавішу (document, Image, Link, Text, Textarea)
keyup	Користувач відпустив клавішу (document, Image, Link, Text, Textarea)
keypress	Користувач натиснув клавішу і відпустив. (document, Image, Link)

Події елемента

abort	Користувач зупиняє завантаженнязображення (Image)
blur	Користувач прибирає фокус із об'єкта (Button,Checkbox,FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window)
change	Користувач змінює зміст елемента форми (the FileUpload, Select, Text)
error	Виникла помилка підчас завантаження (Image, window)
focus	Користувач перемістив фокус на об'єкт (Button, Checkbox, FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window)
select	Користвуач виділив текст (Text, Textarea)

ОБРОБКА ПОДІЙ

Під обробкою подій розуміють виконання деякої команди або ж функції у відповідь на настання події. Обробка подій може здійснювати декількома способами

Задання обробника з використанням атрибуту елемента HTML

У цьому способі необхідно у HTML розмітці елемента додати атрибут, який починається з «on» і містить назву події (наприклад: **on**click, **on**change, **on**select,...). Значенням атрибуту може бути або деяка команда або виклик функції. Як завжди значення атрибутів вказуємо у лапках.

Загальна форма	<елемент **on**подія= "виклик_обробника">
Приклад (вказівка команди)	`<input type="button" name="name" value="Press" onclick="alert("Hello")" />`
Приклад (виклик функції)	`<input type="button" name="name" value="Press" onclick="myFunc()" />`

Приклад. Зчитування значення з текстового поля

---------- у тексті сторінки ----------

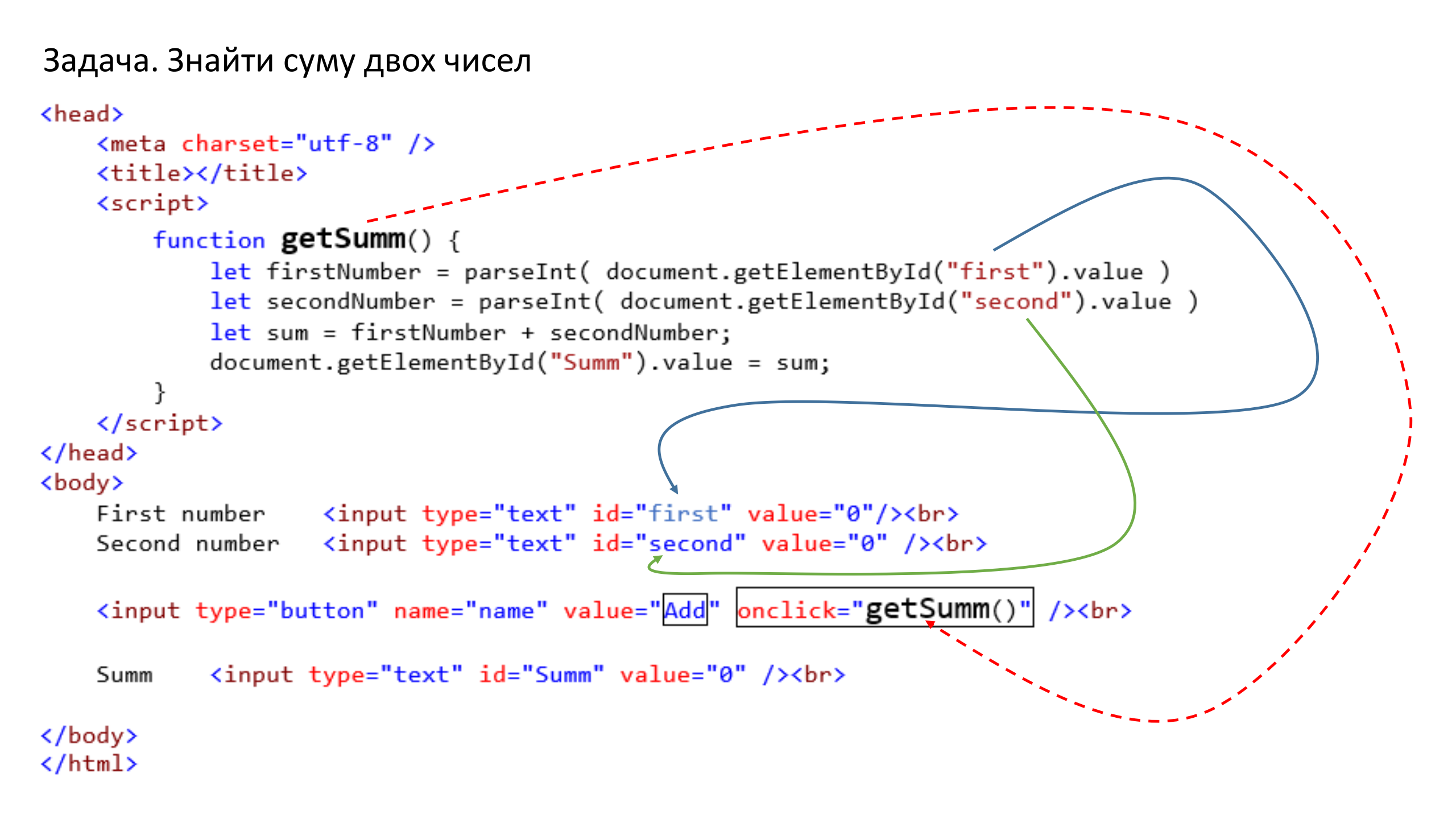
```
    <input type="text" id="MyText" value="" /><br>
```

---------- звертання до елемента -------

```
            let name = document.getElementById("MyText").value;

            alert("Hello " + name);
```

Задача. Знайти суму двох чисел

```
<head>
    <meta charset="utf-8" />
    <title></title>
    <script>
        function getSumm() {
            let firstNumber = parseInt( document.getElementById("first").value )
            let secondNumber = parseInt( document.getElementById("second").value )
            let sum = firstNumber + secondNumber;
            document.getElementById("Summ").value = sum;
        }
    </script>
</head>
<body>
    First number    <input type="text" id="first" value="0"/><br>
    Second number   <input type="text" id="second" value="0" /><br>

    <input type="button" name="name" value="Add" onclick="getSumm()" /><br>

    Summ    <input type="text" id="Summ" value="0" /><br>

</body>
</html>
```



Задання обробника з використанням властивості елемента

У цьому способі необхідно використати властивість елемента як об'єкта JavaScript. При цьому ім'я властивості, як і атрибут має вигляд **on**подія. На відміну від атрибутів елементів, які можуть бути вказані як у нижньому так і верхньому регістрах (onclick, ONCLICK, Onclick,onClick), властивості елемента вказують у нижньому регістрі (onclick).

Загальна форма	елемент . **on**подія= функція_обробника ; елемент ["**on**подія "]= функція_обробника ;
Приклад	``` ------------ Елемент в HTML ------------ <input type="button" id="myButton" value="Press" /> ------------ Встановлення обробника в JavaScript ------------ <script> //------ Варіант 1 ----- myButton.onclick = myFunc; // myFunc – функція обробник //------ Варіант 2 ----- myButton["onclick"] = myFunc; // myFunc – функція обробник </script> ```

```
function функція (){
    ...
}
window.onload = function () {
    document.getElementById("btn").onclick = функція
}
```

```
        function getSum() {
            const num1 = parseInt(document.getElementById("num1").value)
            const num2 = parseInt(document.getElementById("num2").value)
            const s = num1 + num2
            document.getElementById("res").value = s
        }

        window.onload = function () {
            document.getElementById("btn").onclick = getSum
        }

        alert("hello")
    </script>

</head>

<body>
    <label>        Number 1        <input type="number" id="num1" value="2">    </label> <br>
    <label>        Number 2        <input type="number" id="num2" value="3">    </label> <br>
    <button id="btn">        Get sum    </button> <br>
```

```html
<html lang="en">
<head>
    <title>Document</title>
    <script>
        function getSum() {
            const num1 = parseInt(document.getElementById("num1").value)
            const num2 = parseInt(document.getElementById("num2").value)
            const s = num1 + num2
            document.getElementById("res").value = s
        }

        window.onload = function () {
            document.getElementById("btn").onclick = getSum
        }

        alert("hello")
    </script>

</head>

<body>
    <label>        Number 1        <input type="number" id="num1" value="2">    </label> <br>
    <label>        Number 2        <input type="number" id="num2" value="3">    </label> <br>
    <button id="btn">        Get sum    </button> <br>
    <label>        Sum         <input type="number" name="" id="res" value="">
    </label>
</body>
</html>
```

Задача. Конвертер валют (курс і кількість валюти, що треба обміняти задаються)

Приклад. Перевірка стану чекбокса

---------- у тексті сторінки ----------

```
<input type="checkbox" id="MyCheckbox" value="10" /><br>
```

---------- звертання до елемента -------

```
if (document.getElementById("MyCheckbox").checked) {
                alert("Is checked")
            }
            else {
                alert("Is not checked");

            }
```

Задача. Знайти загальну суму обіду

---- напої ---

- чай -10 грн.
- сік – 20 грн
- кава – 35 грн

-----перше ---

- суп – 45 грн
- борщ - 37 грн.

---- друге –

- паста – 60 грн.
- картопля з катлетою – 55 грн.
- гречка з грибами – 49 грн.

Приклад. Аналіз значення з випадаючого списку

---------- у тексті сторінки ----------

```
    <select id="MySelect">
        <option value="1">text1</option>
        <option value="2">text2</option>
        <option value="3">text3</option>

    </select>
```

---------- звертання до елемента -------

```
let value = document.getElementById("MySelect").value;
```

Задача. Знайти загальну вартість поїздки у поїзді:
тип вагону :

- плацкарт – 200грн.
- купе – 800грн.

акційні пакети (вибір тільки одного)

- сніданок – акційна ціна 50 грн.
- безкоштовний чай
- безкоштовна кава

Задання і відміна обробника з використанням методів `addEventListener` та `removeEventListener`

У цьому способі для призначення обробника необхідно у методі елемента **`addEventListener`** вказати назву події та функцію обробник.

Загальна форма	елемент . **`addEventListener`**(подія, функція_обробник , фаза); фаза (true/false) – необов'язковий параметр (за замовчуванням false), який зазначає коли повинен спрацювати обробник (на нисхідному чи висхідному етапі)
Приклад	`------------ Елемент в HTML ------------` `<input type="button" id="myButton" value="Press" />` `------------ Встановлення обробника в JavaScript ------------` `<script>` `  myButton.addEventListener("click", myFunc); // myFunc – функція обробник` `</script>`

Перевагою такого підходу є те, що ми маємо можливість вказати декілька функцій обробників для однієї події

Приклад	`------------ Елемент в HTML ------------` `<input type="button" id="myButton" value="Press" />` `------------ Встановлення обробників в JavaScript ------------` `<script>` `  myButton.addEventListener("click", myFunc); // призначення обробника myFunc` `  myButton.addEventListener("click", myFunc2); //призначення ще одного обробника myFunc2` `  //при виникненні події буде викликано обидві функції` `</script>`

Для видалення обробника необхідно використати метод ***removeEventListener***

Загальна форма	елемент . ***removeEventListener*** (подія, функція_обробник)
Приклад	``` ------------- Елемент в HTML ------------- <input type="button" id="myButton" value=”Press” /> ------------- Встановлення обробників в JavaScript -------------- <script> myButton.addEventListener("click", myFunc); // призначення обробника myFunc myButton.addEventListener("click", myFunc2); //призначення ще одного обробника myFunc2 myButton.removeEventListener("click", myFunc2); // видалення обробника myFunc2 //при виникненні події буде викликано тільки функцію myFunc </script> ```

Об'єкт Event

Об'єкт `Event` зберігає інформацію про подію і об’єкт який згенерував подію. Об’єкт `Event` передається у функцію обробник першим параметром

Приклад

```
----- Зробити напис на кнопці, на яку натиснуто, червоним-----
------------- Елемент в HTML -------------
<input type="button" id="myButton" value=”Press” />
-------------  Встановлення обробника в JavaScript --------------
<script>
        function myHandler( event )
        {
            var btn = event.target;  //одержуємо адресу кнопки, на як натиснуто
            btn.style.color ="red";
        }
        myButton.onclick=myHandler;  // myHandler – функція обробник
  </script>
```

Об'єкт Event

Деякі властивості об'єкта Event

type - тип події (наприклад, "click")

target – об'єкт, якому адресована подія (який згенерував подію)

layerX, layerY – горизонтальна і вертикальна позиції курсора миші

which – номер натиснутої кнопки миші.

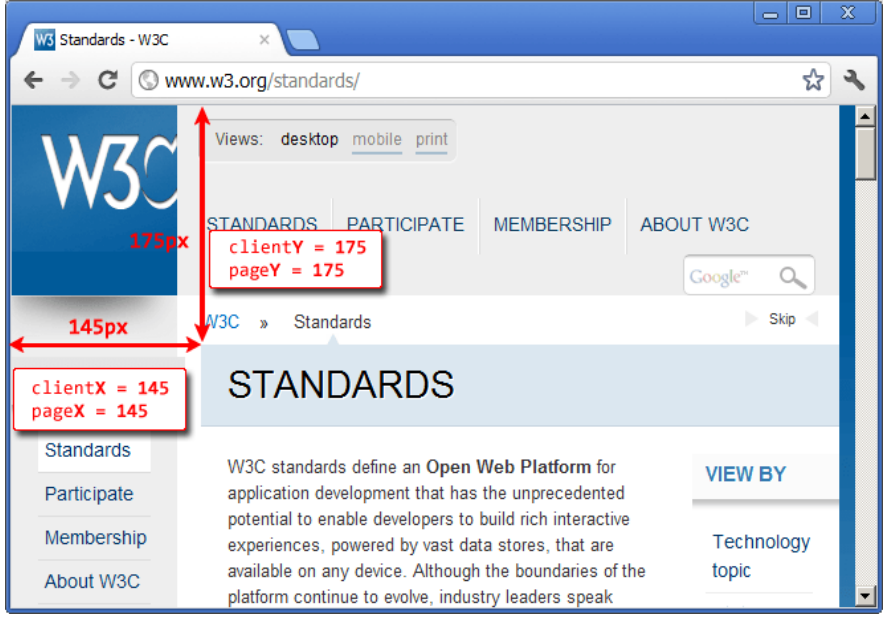
modifiers – чи натиснута клавіша-модифікатор (Alt, Ctrl, Shift). Бітова маска, одна з констант
ALT_MASK, CONTROL_MASK, SHIFT_MASK, and META_MASK

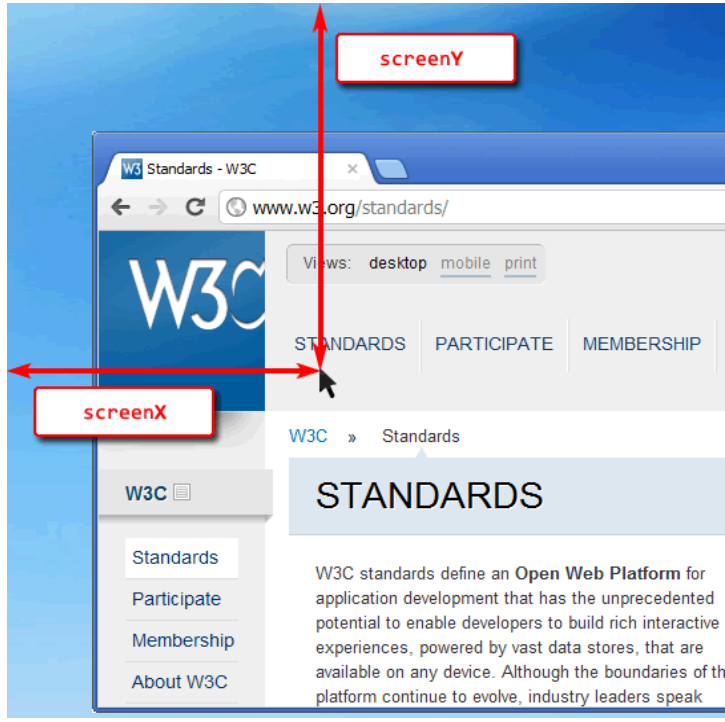
data – масив з рядків, який містить URL об'єктів, які були перетягнуті та впущені у вікно.

pageX, pageY – горизонтальна та вертикальна позиції курсора відносно веб-сторінки

screenX, screenY – горизонтальна та вертикальна позиції курсора на екрані

width, height – ширина та висота вікна після зміни розмірів.



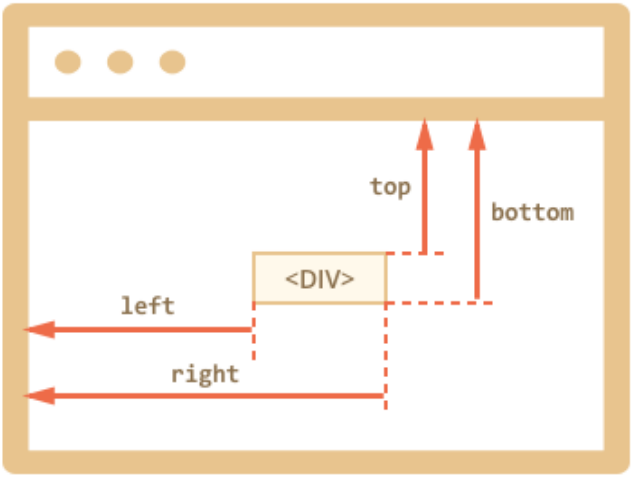


Приклад. Для двох кнопок призначити один обробник, який виводить на екран текст нажатої кнопки

Приклад. Малювання у компоненті canvas за допомогою миші.

```
<canvas id="ccc" style="border: solid 2px"></canvas> //елемент, у якому здійснюємо графічні побудови
<script>
  let isActive = false //Змінна, яка містить true/false (прапорець – малювати/не_малювати).
  let x
  let y

  ccc.onmousedown = function (e) {
    isActive = true //Увімкнути режим малювання
    x = e.clientX //Отримати клієнтські координати миші
    y = e.clientY
  }
  ccc.onmouseup = function (e) {
    isActive = false //Відімкнути режим малювання
  }
  ccc.onmousemove = function (e) {
    if (isActive) {
      //let pos = document.getElementById("ccc").getBoundingClientRect();
      let pos = this.getBoundingClientRect() //Одержуємо інформацію про положення елемента
      let cx = document.querySelector('canvas').getContext('2d')
      cx.beginPath()
      cx.moveTo(x - pos.left, y - pos.top) //Малюємо лінію від попередньої точки до поточної
      x = e.clientX
      y = e.clientY
      cx.lineTo(x - pos.left, y - pos.top)
      cx.stroke()
    }
  }
</script>
```

elem.getBoundingClientRect()



Відміна події

Для відміни дії браузера на подію у даному елементі достатньо, щоб функція обробник повернула false або скористатися методом об’єкта події `event.preventDefault()`.

Приклад	----- Відмінити перехід за посиланням при кліку ----- ------------ Елемент в HTML ------------ link

Значення `this` при виклику обробника

Всередині обробника this містить адресу елемента, для якого спрацював обробник (проводиться аналіз події). Також адресу елемента, для якого спрацював обробник можна одержати через об’єкт події event.currentTarget.

Приклад	----- Зробити напис на кнопці, на яку натиснуто, червоним----- ------------ Елемент в HTML ------------ <input type="button" id="myButton" value="press" onclick="myHandler(this)"/> ------------ Встановлення обробника в JavaScript ------------ <script> function myHandler(btn) { //через btn одержуємо адресу кнопки, на яку натиснуто btn.style.color ="red"; } </script>

Захоплення подій

Наприклад, вам може знадобитися, щоб, об'єкт window обробляв всі події MouseDown, CLICK, MOUSEUP незалежно від місця їх виникнення.

Для цього існують методи

captureEvents – захоплює події вказаного типу.

releaseEvents – ігнорує події вказаних типів.

routeEvent – перенаправляє захоплену подію вказаному об'єкту.

handleEvent – обробляє захоплену подію.

```
window.captureEvents(Event.CLICK | Event.MOUSEDOWN | Event.MOUSEUP)
```

Depricated !

1. Event.ABORT
2. Event.BLUR
3. Event.CHANGE
4. Event.CLICK
5. Event.DBLCLICK
6. Event.DRAGDROP
7. Event.ERROR
8. Event.FOCUS
9. Event.KEYDOWN
10. Event.KEYPRESS
11. Event.KEYUP
12. Event.LOAD
13. Event.MOUSEDOWN
14. Event.MOUSEMOVE
15. Event.MOUSEOUT
16. Event.MOUSEOVER
17. Event.MOUSEUP
18. Event.MOVE
19. Event.RESET
20. Event.RESIZE
21. Event.SELECT
22. Event.SUBMIT
23. Event.UNLOAD

```html
<div>Кількість кліків = <span id="click_num"></span></div>

<button id="btn1" onclick="alert('1')">btn 1</button>
<button onclick="alert('2')">btn 2</button>
<input type="text" onclick="alert('3')" />
<a href="https://www.ukr.net/">НОВИНИ</a>
<div>TEXT 1</div>

<script>
    //-------- Підраховуємо кількість кліків -----
  let num = 0
  window.captureEvents(Event.CLICK)
  window.onclick = (event) => {
    num++
    click_num.innerText = num
    //заблокувати натистення на посилання
    if (event.target.tagName === 'A') return false
    
  }
</script>
```

Кількість кліків = 3

btn 1 btn 2

НОВИНИ 1

TEXT 1

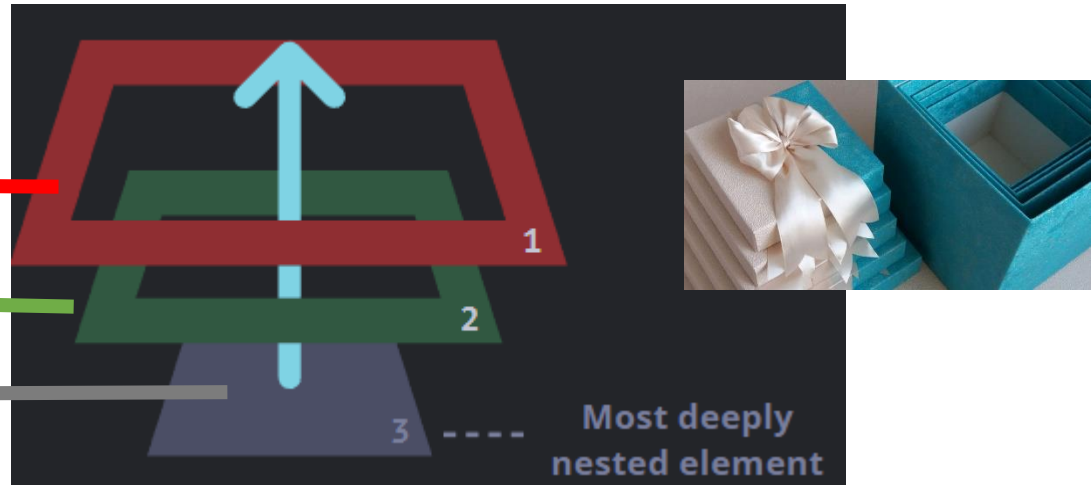
Черга подій

У процесі роботи браузера усі події, які генеруються як результат діяльності користувача, а також події, які згенеровані програмно потрапляють у чергу подій, з якої поступово ці події вибираються і опрацьовуються.

Підйом

Звичайною ситуацією є випадок, коли один елемент знаходиться всередині іншого. При настанні події для деякого внутрішнього елемента відбувається послідовний виклик обробників даної події у напрямку від внутрішнього (для якого настала подія) до усіх зовнішніх, в середині яких він знаходиться.

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



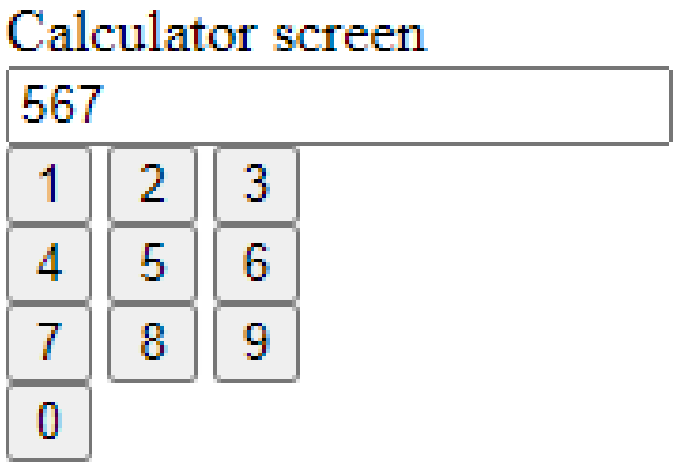
https://uk.javascript.info/bubbling-and-capturing

Зупинити підйом можна скориставшись методом об'єкта події ***event.stopPropagation()***

Делегування обробки події

Той факт, що при генеруванні події відбувається її підйом по ієрархії вкладеності (від внутрішнього до зовнішнього) часто використовують для реалізації делегування подій. Це означає, що не потрібно призначати для кожного внутрішнього елемента однотипний обробник однієї і тієї ж події. Це можна зробити для елемента, у якому вони знаходяться.

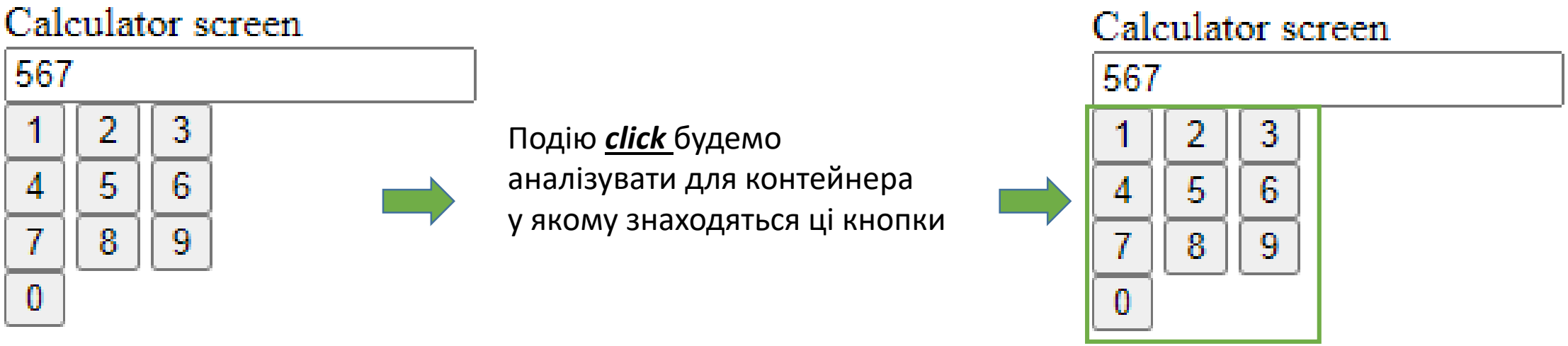
Приклад. При натисненні на кнопку текст, що розміщений на кнопці повинен додватись до поля вводу.



Делегування обробки події

Той факт, що при генеруванні події відбувається її підйом по ієрархії вкладеності (від внутрішнього до зовнішнього) часто використовують для реалізації делегування подій. Це означає, що не потрібно призначати для кожного внутрішнього елемента однотипний обробник однієї і тієї ж події. Це можна зробити для елемента, у якому вони знаходяться.

Приклад. При натисненні на кнопку текст, що розміщений на кнопці повинен додватись до поля вводу.



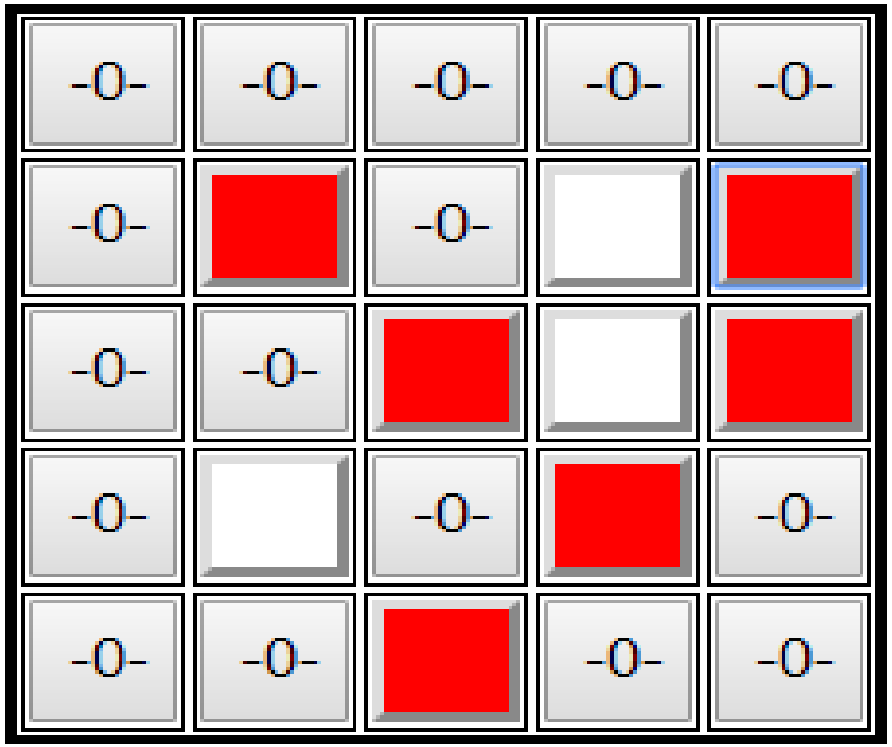
```html
<div>
  Calculator screen <input id="screen" type="text" readonly value="" />
</div>
<div id="buttons_container">
  <div>
    <button>1</button> <button>2</button> <button>3</button>
  </div>
  <div>
    <button>4</button> <button>5</button> <button>6</button>
  </div>
  <div>
    <button>7</button> <button>8</button> <button>9</button>
  </div>
  <div>
    <button>0</button>
  </div>
</div>
<script>
  buttons_container.onclick = function (event) {
    console.log(event.target.tagName)
    if (event.target.tagName === 'BUTTON') {
      document.getElementById('screen').value += event.target.innerText
    }
  }
</script>
```

Calculator screen

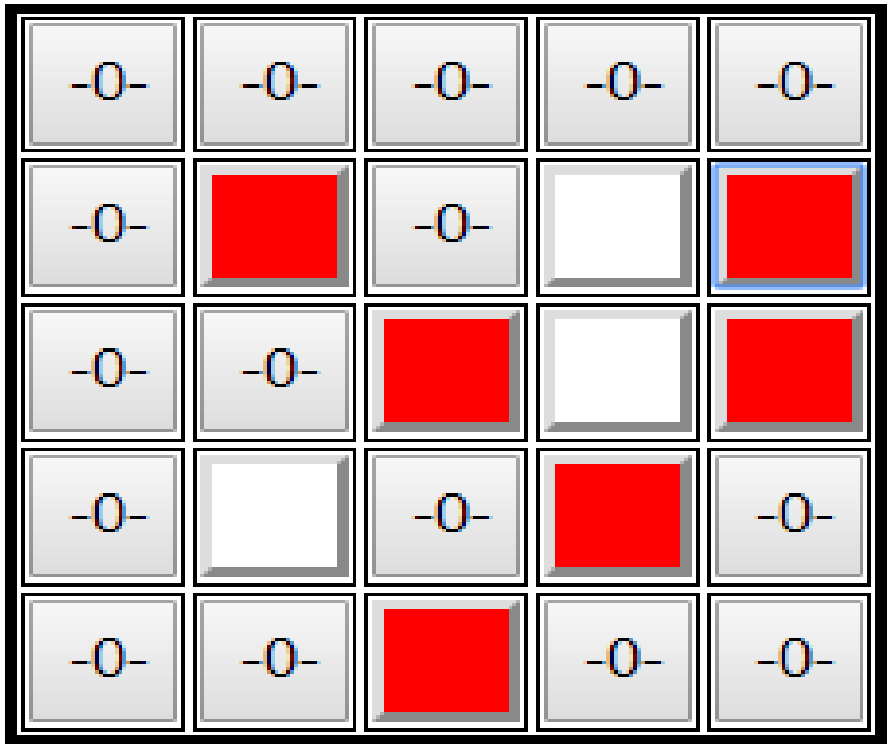
567

1 2 3
4 5 6
7 8 9
0

Приклад. Створити таблицю. При натисканні на клітинку вміст змінює на червоний колір.



Приклад. Створити клас з грою «Міни». Випадковим чином у двовимірному масиві розміщують кнопки, для кожної з яких встановлюється атрибут, що містить 0-немає міни, 1- є міна.



Події. Етапи аналізу

У сучасному стандарті окрім підняття можна проаналізувати і етап спуску. Тому аналіз події складається з 3-х етапів.

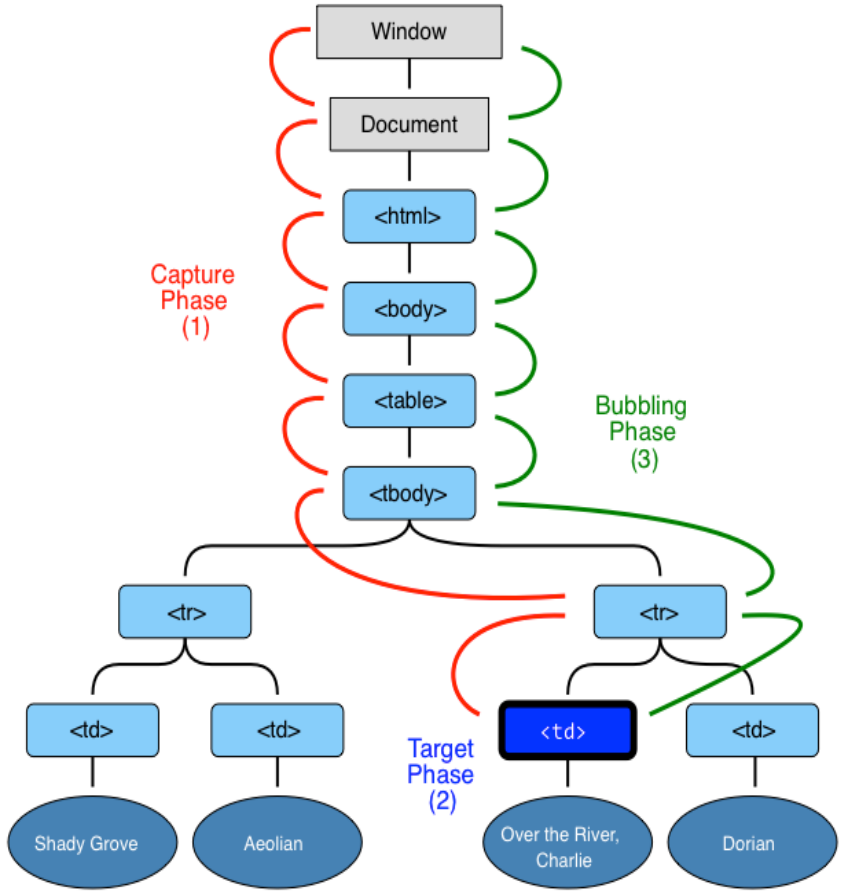
- спуск (capturing stage);
- досягнення цільового елемента (target stage);
- підняття (bubbling stage).

У випадку, якщо необхідно *проаналізувати подію*

- на етапі спуску необхідно призначати функцію обробник за допомогою методу addEventListener з останнім параметром true,
- на етапі підйому – з останнім параметром false.

Приклад.

```
<div id="divv" >
        <input type="button" id="btn" name="name" value="" />
</div>

    <script>
        function f_before(e) {
            alert("before" + this.tagName + " target=" + e.target.tagName);
        }
        function f_after(e) {
            alert("after" + this.tagName + " target" + e.target.tagName);
        }

        divv.addEventListener("click", f_before, true);
        divv.addEventListener("click", f_after, false);

    </script>
```



Визначити етап, на якому аналізується подія, можна з використанням властивості

`event.eventPhase` – (спуск =1, підняття= 3).

Генерування подій

Для генерування події спочатку треба створити об’єкт класу `Event` і ініціювати подію з використанням методу `elem.dispatchEvent(event)`

```
const myEvent = new Event('myevent', {
  bubbles: true,
  cancelable: true,
  composed: false
})
```

Створення події

Загальна форма	`let event = new Event(тип події[, опції]);`
Приклад	`let event = new Event("click");`

Тип події – може быту як власним, так і вбудованим, наприклад `"click"`.

Опції (прапорці) – об’єкт `{ bubbles: true/false, cancelable: true/false }`,

де `bubbles` визначає чи може подія підніматися (спливати), `cancelable` – чи можна відмінити, `composed` – чи може подія поширюватись за межі Shadow DOM.

Генерування події

Загальна форма	`elem.dispatchEvent(event);`
Приклад	`<button id="elem" onclick="alert('Hello!');">Say hello</button>` `<script>` `  let event = new Event("click");` `  elem.dispatchEvent(event);` `</script>`

Додавання у об’єкт події користувацьких даних ***CustomEvent***

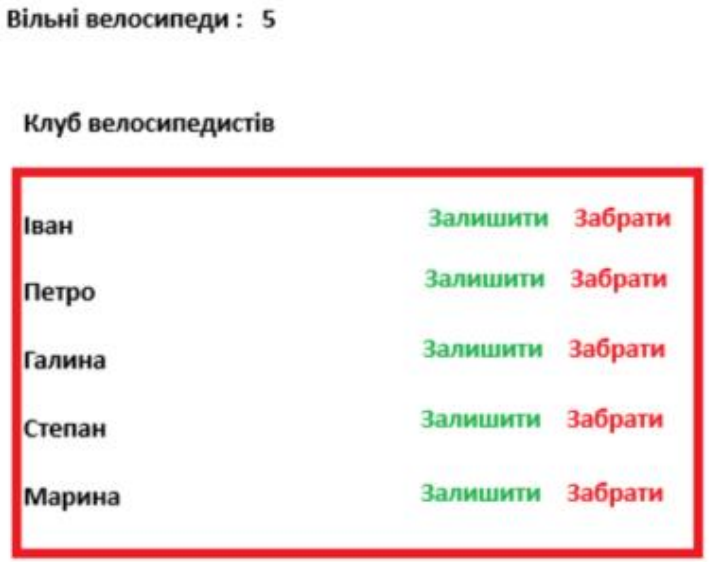
Для додавання деяких додаткових даних користувача у подію можна скористатись інтерфейсом ***CustomEvent*** і використати властивість ***`detail`***.

Створення події

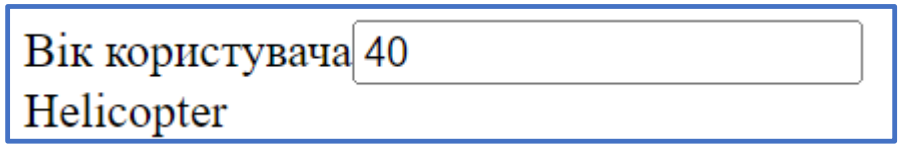
Загальна форма	`let event = new CustomEvent (тип_події, {detail:{деякі_дані}});`
Приклад	`let event = new CustomEvent (тип_події, {detail:{elId:1256}});`

```
const myEvent = new CustomEvent("myevent", {
  detail: {},
  bubbles: true,
  cancelable: true,
  composed: false,
});
```

Задача. «Вільний велосипед». Дано список учасників велосипедного клубу. Кожен з учасників може залишити вільний велосипед (тоді кількість вільних велосипедів відображається зверху списку) або забрати вільний велосипед (якщо є такий)



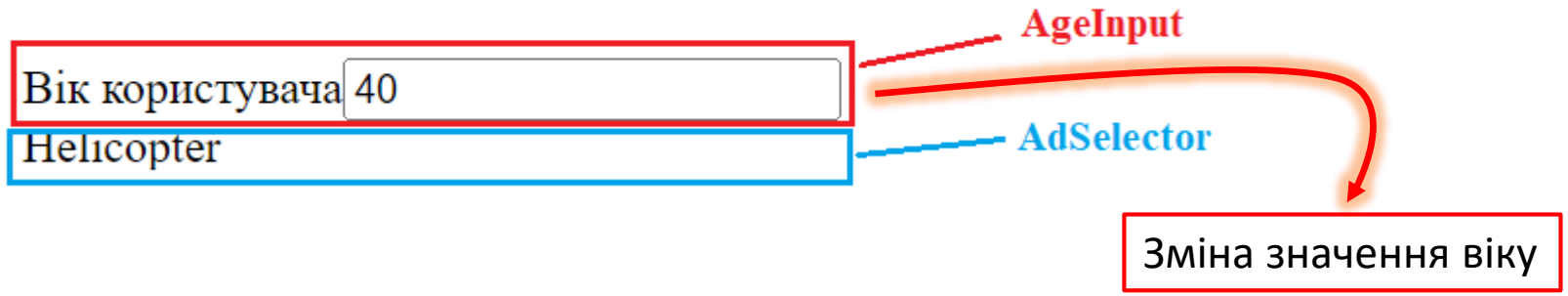
Задача. При введенні віку користувача треба виводити рекомендовану рекламу



Задача. При введенні віку користувача треба виводити рекомендовану рекламу

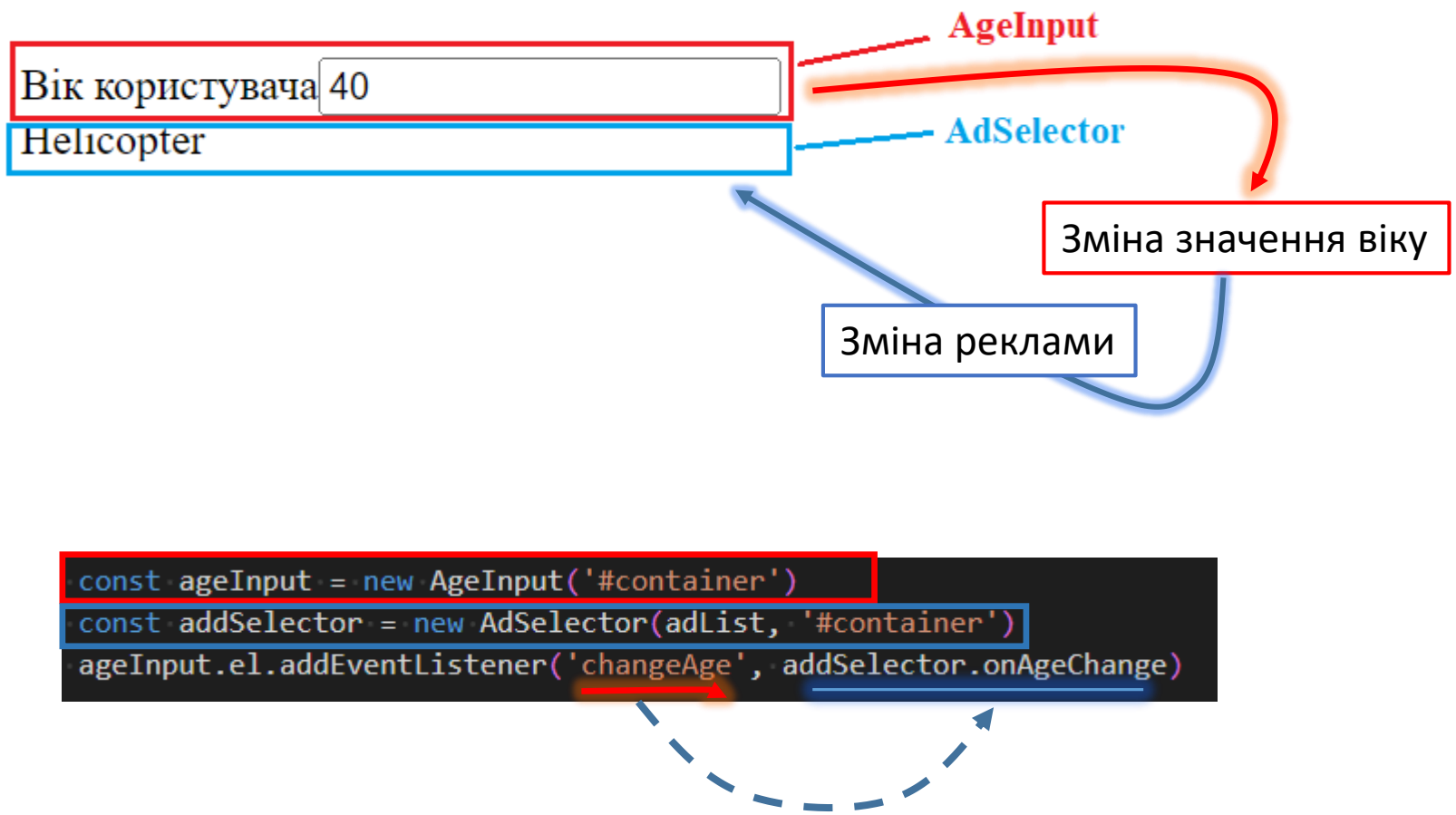

```
const adList = [
    {
        minAge: 10,
        maxAge: 15,
        message: 'Phone',
    },
    {
        minAge: 16,
        maxAge: 25,
        message: 'Car',
    },
    {
        minAge: 26,
        maxAge: 45,
        message: 'Helicopter',
    },
]
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу




```
const adList = [
    {
        minAge: 10,
        maxAge: 15,
        message: 'Phone',
    },
    {
        minAge: 16,
        maxAge: 25,
        message: 'Car',
    },
    {
        minAge: 26,
        maxAge: 45,
        message: 'Helicopter',
    },
]
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу




```
const ageInput = new AgeInput('#container')
const addSelector = new AdSelector(adList, '#container')
ageInput.el.addEventListener('changeAge', addSelector.onAgeChange)
```

```
const adList = [
    {
        minAge: 10,
        maxAge: 15,
        message: 'Phone',
    },
    {
        minAge: 16,
        maxAge: 25,
        message: 'Car',
    },
    {
        minAge: 26,
        maxAge: 45,
        message: 'Helicopter',
    },
]
```

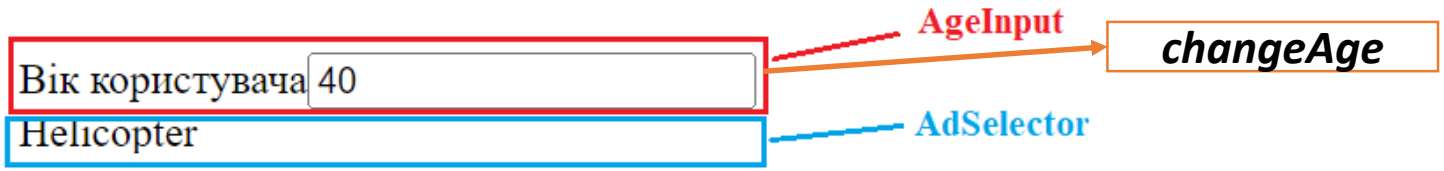
Задача. При введенні віку користувача треба виводити рекомендовану рекламу


```
class AgeInput {



  onChange() {




  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))
  }









}
```

1)При зміні значення **input** викликаємо метод ***onChange***

```
const adList = [
  {
    minAge: 10,
    maxAge: 15,
    message: 'Phone',
  },
  {
    minAge: 16,
    maxAge: 25,
    message: 'Car',
  },
  {
    minAge: 26,
    maxAge: 45,
    message: 'Helicopter',
  },
]
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

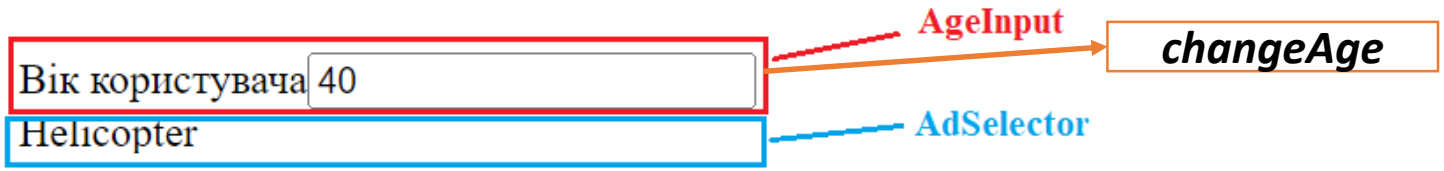



```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })

  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }









}
```

2) Створюємо об’єкт події ***changeAge***

```
const adList = [
  {
    minAge: 10,
    maxAge: 15,
    message: 'Phone',
  },
  {
    minAge: 16,
    maxAge: 25,
    message: 'Car',
  },
  {
    minAge: 26,
    maxAge: 45,
    message: 'Helicopter',
  },
]
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу




```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }









}
```

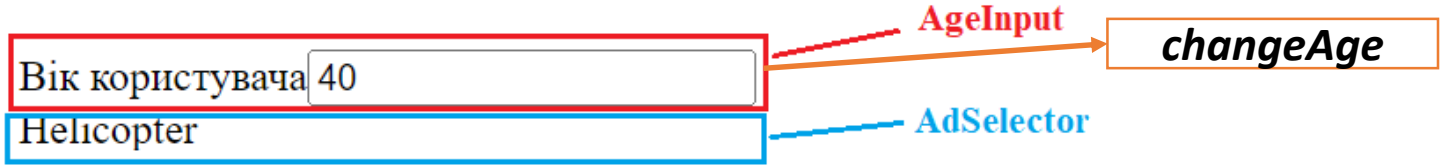
2) Створюємо об’єкт події ***changeAge***

3) Генеруємо подію ***changeAge***

```
const adList = [
  {
    minAge: 10,
    maxAge: 15,
    message: 'Phone',
  },
  {
    minAge: 16,
    maxAge: 25,
    message: 'Car',
  },
  {
    minAge: 26,
    maxAge: 45,
    message: 'Helicopter',
  },
]
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

4) Створюємо клас AdSelector




```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }

}
```

2) Створюємо об'єкт події ***changeAge***

3) Генеруємо подію ***changeAge***

Метод виведення повідомлення у залежності від віку

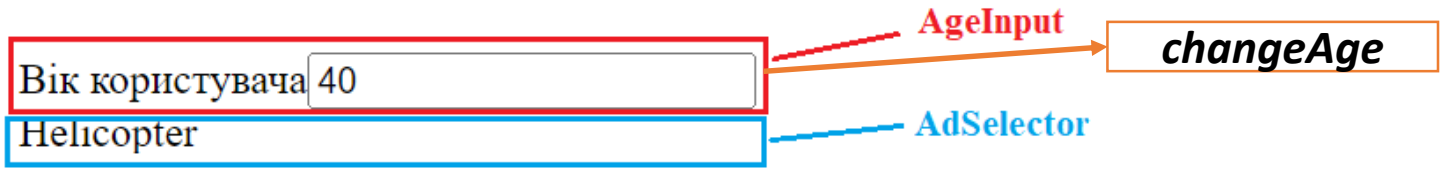
```
const adList = [
  {
    minAge: 10,
    maxAge: 15,
    message: 'Phone',
  },
  {
    minAge: 16,
    maxAge: 25,
    message: 'Car',
  },
  {
    minAge: 26,
    maxAge: 45,
    message: 'Helicopter',
  },
]
```

```
class AdSelector {









  showAd(selectedAge) {
    const adItem = this.addList.find(
      (item) => selectedAge >= item.minAge
            && selectedAge <= item.maxAge
    )
    if (adItem) this.el.innerHTML = adItem.message
    else this.el.innerHTML = 'Age not selected'
  }

}
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

```
const ageInput = new AgeInput('#container')
const addSelector = new AdSelector(adList, '#container')
```

5)Створюємо об’єкти класів ***ageInput, addSelector***




```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }


}
```

Створюємо об’єкт події ***changeAge***

Генеруємо подію ***changeAge***

```
class AdSelector {









  showAd(selectedAge) {
    const adItem = this.addList.find(
      (item) => selectedAge >= item.minAge
            && selectedAge <= item.maxAge
    )
    if (adItem) this.el.innerHTML = adItem.message
    else this.el.innerHTML = 'Age not selected'
  }


}
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

```
const ageInput = new AgeInput('#container')
const addSelector = new AdSelector(adList, '#container')
ageInput.el.addEventListener('changeAge', addSelector.onAgeChange)
```

AgeInput

AdSelector

changeAge

6) На подію ***changeAge*** об'єкта ***ageInput*** призначаємо метод ***onAgeChange*** об'єкта ***addSelector***

```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }

}
```

Створюємо об'єкт події ***changeAge***

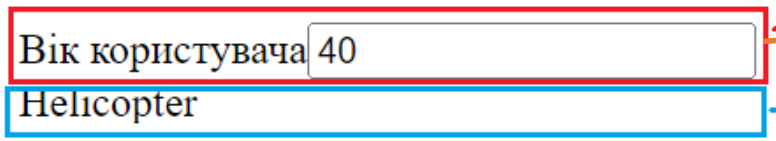
Генеруємо подію ***changeAge***

Метод аналізу зміни значення віку

```
class AdSelector {



  showAd(selectedAge) {
    const adItem = this.addList.find(
      (item) => selectedAge >= item.minAge
            && selectedAge <= item.maxAge
    )
    if (adItem) this.el.innerHTML = adItem.message
    else this.el.innerHTML = 'Age not selected'
  }
  onAgeChange(event) {
    const selectedAge = event?.detail.age
    this.showAd(selectedAge)
  }


}
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

```
const ageInput = new AgeInput('#container')
const addSelector = new AdSelector(adList, '#container')
ageInput.el.addEventListener('changeAge', addSelector.onAgeChange)
```



AgeInput

AdSelector

changeAge

```
class AgeInput {


  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }

}
```

Створюємо об’єкт події ***changeAge***

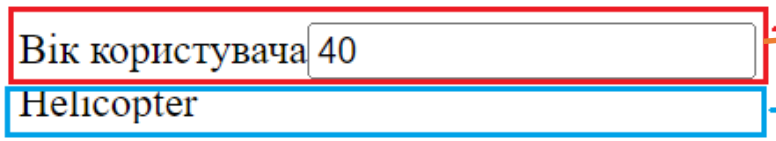
Генеруємо подію ***changeAge***

7) При настанні події ***changeAge*** викликаємо метод виведення реклами ***showAd***

```
class AdSelector {


  showAd(selectedAge) {
    const adItem = this.addList.find(
      (item) => selectedAge >= item.minAge
        && selectedAge <= item.maxAge
    )
    if (adItem) this.el.innerHTML = adItem.message
    else this.el.innerHTML = 'Age not selected'
  }
  onAgeChange(event) {
    const selectedAge = event?.detail.age
    this.showAd(selectedAge)
  }


}
```

Задача. При введенні віку користувача треба виводити рекомендовану рекламу

```
const ageInput = new AgeInput('#container')
const addSelector = new AdSelector(adList, '#container')
ageInput.el.addEventListener('changeAge', addSelector.onAgeChange)
```



AgeInput

AdSelector

changeAge

8) Виводимо рекламу

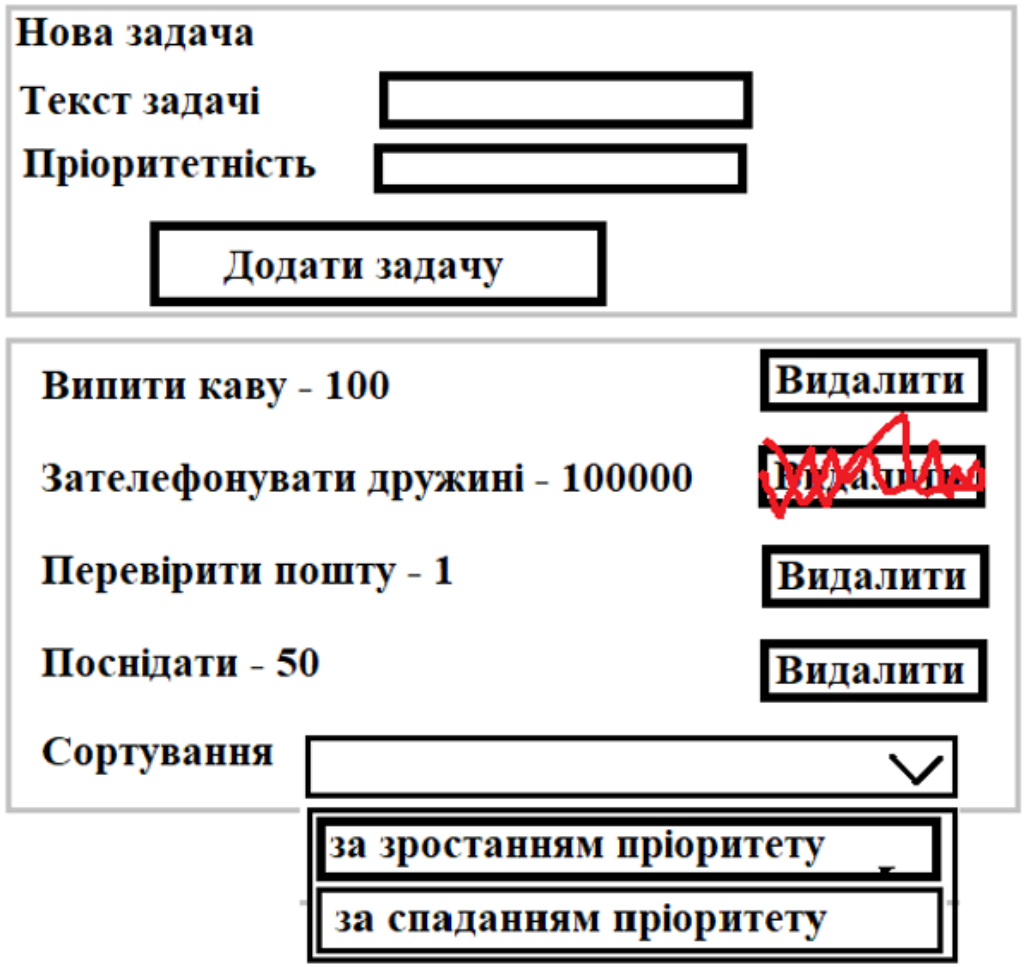
```
class AgeInput {



  onChange() {
    const changeAge = new CustomEvent('changeAge', {
      detail: {
        age: this.ageInput.value,
      },
    })
    this.el.dispatchEvent(changeAge)
  }
  createElement() {


    input.addEventListener('change', this.onChange.bind(this))

  }

}
```

Створюємо об’єкт події ***changeAge***

Генеруємо подію ***changeAge***

```
class AdSelector {



  showAd(selectedAge) {
    const adItem = this.addList.find(
      (item) => selectedAge >= item.minAge
          && selectedAge <= item.maxAge
    )
    if (adItem) this.el.innerHTML = adItem.message
    else this.el.innerHTML = 'Age not selected'
  }
  onAgeChange(event) {
    const selectedAge = event?.detail.age
    this.showAd(selectedAge)
  }


}
```

Задача. Розробити ToDo менеджер. Користувач вводить текст задачі та пріоритетність. Відображається список доданих задач з можливістю видалення задач та сортування за пріоритетністю. При видаленні елемент генерує подію зі своїм ідентифікатором.



```
class AddTaskForm {
  constructor() {
    this.el = this.createElement()
  }
  onAddTask() {
    const addEvent = new CustomEvent('add',
    {
      detail: {
        title: this.taskTitleInp.value,
        priority: this.taskPriorityInp.
        value,
      },
    })
    this.el.dispatchEvent(addEvent)
  }
  //--- функція, для побудови розмітки ---
  createElement() {
```

```
const btn = ElemensCreator.
createHTMLElement({
  tag: 'button',
  props: {
    innerText: 'Додати задачу',
  },
  events: {
    click: this.onAddTask.bind(this),
  },
})
```

Нова задача

Текст задачі task3
Пріоритетність задачі 33
Додати задачу

```
class TaskManager {
  constructor() {…
  deleteTask(event) {
  addTask(event) {
    const task = {
      id: new Date().getTime(),
      ...event.detail,
    }
    const taskObj = new Task(task)
    taskObj.el.addEventListener('delete', this.deleteTask.
    bind(this))
    this.taskList.push(taskObj)
    this.displayList()
  }

  displayList() {…

  createElement() {
    const container = document.createElement('div')

    const addForm = new AddTaskForm()
    addForm.el.addEventListener('add', this.addTask.bind
    (this))
```

Нова задача

Текст задачі task3
Пріоритетність задачі 33
Додати задачу

task1 - 1 Видалити

4.Виккається функція обробки події «add»
“detail” –містить дані про нову задачу

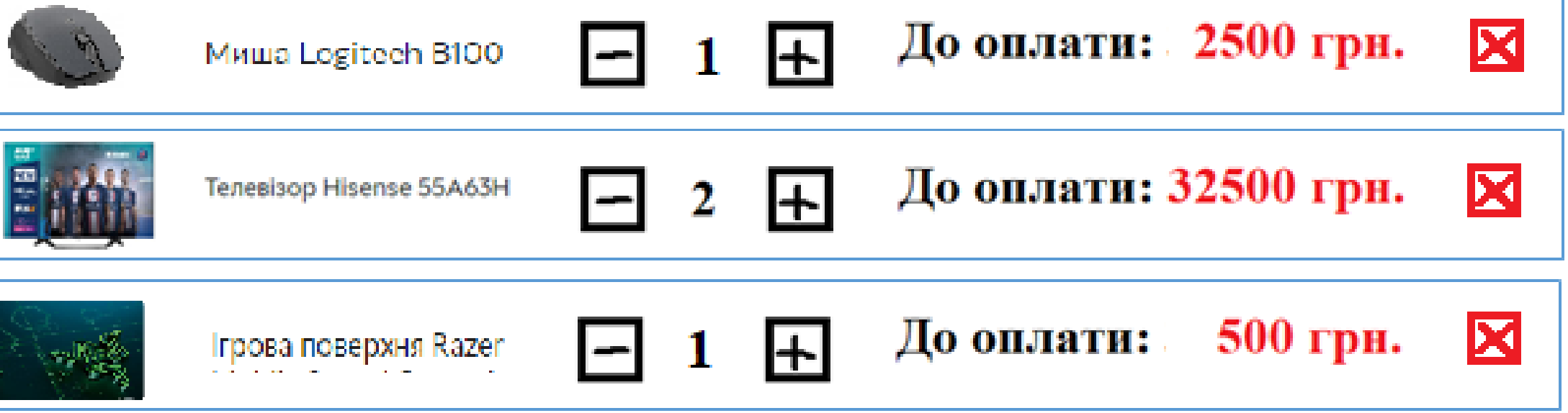
2.Викликаться функція
обробки події “click”

3. Генерується подія “add” з даними про нову задачу
“detail” – містить дані про нову задачу

Нова задача

Текст задачі task1
Пріоритетність задачі 1
Додати задачу
task1 - 1 Видалити

Приклад. Дано перелік товарів у кошику. При зміні кількості одиниць товару збільшувати загальну вартість. Створити клас Product, що призначений для маніпуляцій товаром та клас ProductManager що оперує з усіма товарами (через подію передвати ідентифікатор товару та операцію, що зроблена



Загалом до оплати: 35500 грн.

```
        taskObj.el.addEventListener('delete', this.deleteTask.
        bind(this))
        this.taskList.push(taskObj)
        this.displayList()
    }

    displayList() {···

    createElement() {
        const container = document.createElement('div')

        const addForm = new AddTaskForm()
        addForm.el.addEventListener('add', this.addTask.bind
        (this))
```

3. Генерується подія “add” з даними про нову задачу
“detail” – містить дані про нову задачу