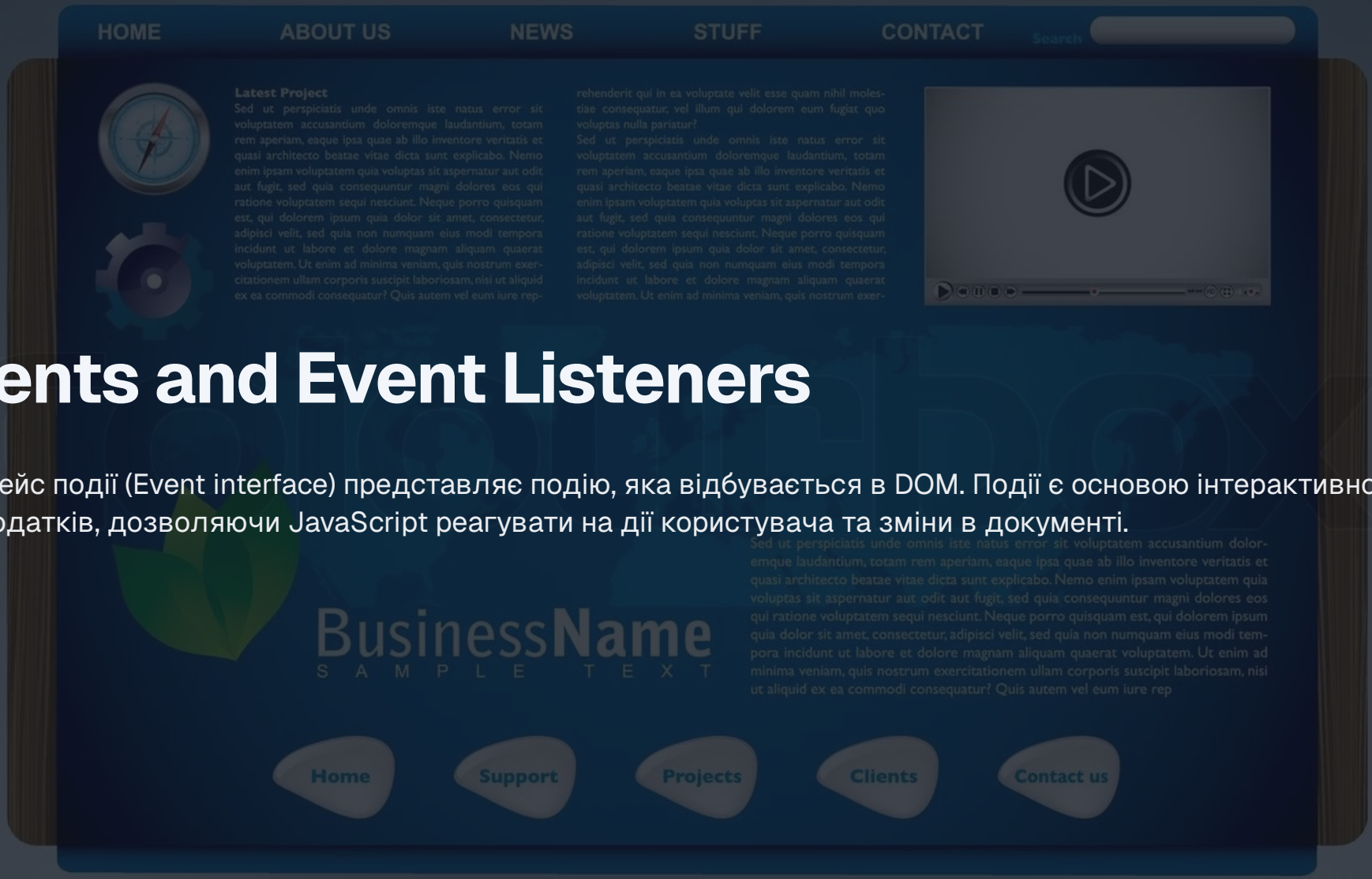


Events and Event Listeners

Інтерфейс події (Event interface) представляє подію, яка відбувається в DOM. Події є основою інтерактивності веб-додатків, дозволяючи JavaScript реагувати на дії користувача та зміни в документі.

Джерела та природа подій

Користувацькі події

Подію може викликати користувач, наприклад, натисненням кнопки миші або клавіші на клавіатурі.

Програмні події

Event також можна згенерувати програмно, наприклад, викликом методу HTMLElement.click() або визначенням події з подальшим відправленням через EventTarget.dispatchEvent().

Події можуть бути згенеровані API для відображення ходу асинхронного завдання. Існує багато типів подій, деякі використовують інші інтерфейси на основі Event interface. Сам інтерфейс Event містить властивості та методи, спільні для всіх подій.

Типи подій (Events)

Mouse Events

click: Клік на елементі

mouseover: Наведення курсора

mouseout: Курсор покидає елемент

Keyboard Events

keydown: Натискання клавіші

keyup: Відпускання клавіші

keypress: Процес натискання

Form Events

submit: Відправка форми

reset: Скидання форми

Window Events

load: Завантаження сторінки

resize: Зміна розміру вікна

scroll: Прокрутка сторінки

Focus Events

focus / **blur**: Фокусування та втрата фокусу

Media Events

play / **pause**: Відтворення медіа

Drag Events

dragstart / **drag** / **dragend**: Перетягування

Робота з Event Listeners

Багато елементів DOM можуть бути налаштовані на прийом (або "слухання") подій та виконання коду. Обробники подій (Event-handlers) зазвичай прикріплені до HTML-елементів за допомогою EventTarget.addEventListener(). Їх можна відключити через removeEventListener().

Створення події

```
// Створення події
const customEvent = new
Event('customEventName');

// Отримання елемента
const targetElement =
  document.getElementById('myElement');

// Відправлення події
targetElement.dispatchEvent(customEvent);
```

Реєстрація обробника

```
// Обробник події
function handleClick() {
 console.log('Button clicked!');
}

const button =
 document.getElementById('myButton');

// Додавання обробника
button.addEventListener('click', handleClick);

// Видалення після 3 секунд
setTimeout(() => {
 button.removeEventListener('click', handleClick);
}, 3000);
```

Бульбашковий механізм (Event Bubbling)

Коли подія відбувається на елементі, спочатку запускаються обробники на ньому, потім на його батьківському елементі, потім на інших предках і так до самого верху.

Крок 1: Внутрішній елемент

Подія спрацьовує на цільовому елементі <P>

Крок 2: Батьківський елемент

Подія спливає до <DIV>

Крок 3: Верхній рівень

Подія досягає <FORM> і document

event.target – "цільовий" елемент, який ініціював подію, він не змінюється в процесі спливання.

event.currentTarget або **this** – "поточний" елемент, на якому виконується обробник.

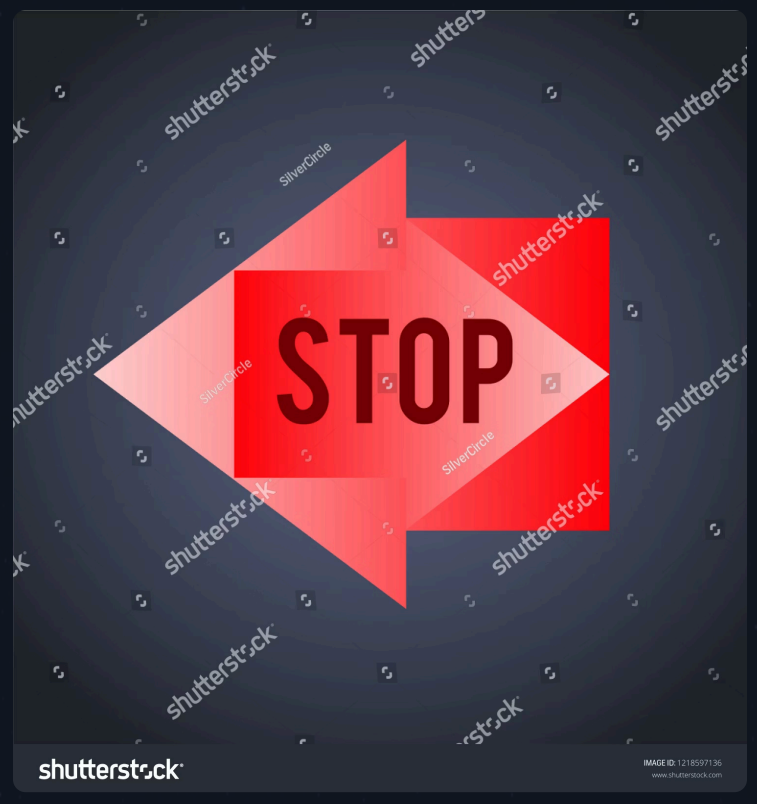
```
<form onclick="alert('form')">FORM
  <div onclick="alert('div')">DIV
    <p onclick="alert('p')">P</p>
  </div>
</form>
```

При кліку на <P> побачимо: p → div → form. Майже всі події спливають, хоча є винятки (наприклад, *focus*).

Припинення спливання

Бульбашкова подія іде від цільового елемента вгору до <html>, потім до document, і деякі події навіть досягають window, викликаючи всі обробники на своєму шляху.

Будь-який обробник може зупинити подію за допомогою методу event.stopPropagation().



Цільовий елемент

Подія виникає тут

stopPropagation()

Зупиняє спливання

Батьківські обробники

Не виконуються

```
<body onclick="alert('body')">
  <button onclick="event.stopPropagation()">
    Click me
  </button>
</body>
```

Обробник body.onclick не спрацює при натисканні на кнопку.

event.stopImmediatePropagation()

stopPropagation()

- Зупиняє подальше спливання вгору
- Інші обробники на поточному елементі виконуються
- Дозволяє множинні обробники

stopImmediatePropagation()

- Зупиняє спливання вгору
- Запобігає виконанню інших обробників
- Повна зупинка обробки події

⚠️ Не зупиняйте спливання без потреби!

Бульбашковий механізм є зручним. Не зупиняйте його без реальної потреби: це очевидно і має добре обґрунтовану архітектуру. Використання stopPropagation() може створювати сховані пастки, які в майбутньому стануть проблемою.

Пастки зупинки спливання

01

Створення вкладеного меню

Кожне підменю викликає stopPropagation, щоб зовнішнє меню не спрацьовувало

02

Додавання аналітики

Пізніше потрібно відслідковувати кліки по всьому вікну через document.addEventListener('click'...)

03

Виникнення проблеми

Аналітика не працює в області, де кліки зупинені. Виникає "мертва зона"

> Зазвичай немає справжньої потреби зупиняти бульбашковий механізм. Задачу, яка здається вимагає цього, можна вирішити іншими способами.

Також існує механізм "занурення" події (event capturing), але він рідко використовується на практиці. Це зворотний процес до спливання, коли подія спочатку опускається від document до цільового елемента.