infoShare ACADEMY

JavaScript obsługa zdarzeń i callback

infoShare Academy

infoShareAcademy.com

HELLO

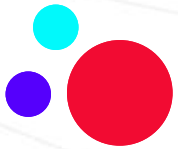
Maciej Mikulski

Senior Front End Developer
@JIT.Team @Dialecticanet.com

infoShare ACADEMY

Interfejs EventTarget

infoShareAcademy.com



Materiały referencyjne

Introduction to Events (MDN) - poradnik wprowadzający do tematyki Eventów

Events (MDN) - opis eventów w JS

Bubbling and capturing (javascript.info) - jak propagowane są eventy w przeglądarce

Mouse Events (Flavio Copes) - informacje o Mouse Events

keycode.info - pomocniczna stronka do weryfikacji keycode dla Keyboard Events

Event listener

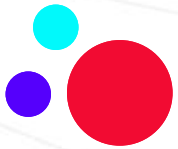
```
<script>
    const button = document.querySelector(".btn");
    const handler = function () {
      console.log("click");
    };

    button.addEventListener("click", handler);
</script>
```

Event listener i Event

```
<script>
    const button = document.querySelector(".btn");
    const handler = function (event) {
      console.log("event details: ", event);
    };

    button.addEventListener("click", handler);
</script>
```



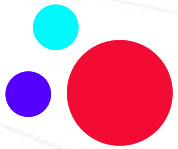
Event listeners options

Jako trzeci argument możemy przekazać obiekt options

```
el.addEventListener(‘click’, function, {
 capture: true,
 once: true,
 passive: true
})
```

lub Boolean wskazujący na atrybut ‘useCapture’ – jest to domyśln

```
el.addEventListener(‘click’, function, false)
```



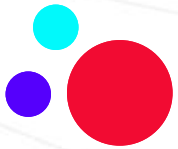
onclick kontra addEventListener('click')

```
el.onclick = function1
el.onclick = function2
```

Powyżej następuje nadpisanie poprzedniej funkcji.

```
el.addEventListener('click', function1)
el.addEventListener('click', function2)
```

Element otrzymuje wiele handlerów, a zostaną one wykonane w kolejności rejestracji. (zobacz listę Event handlerów w Chrome Dev Tools)

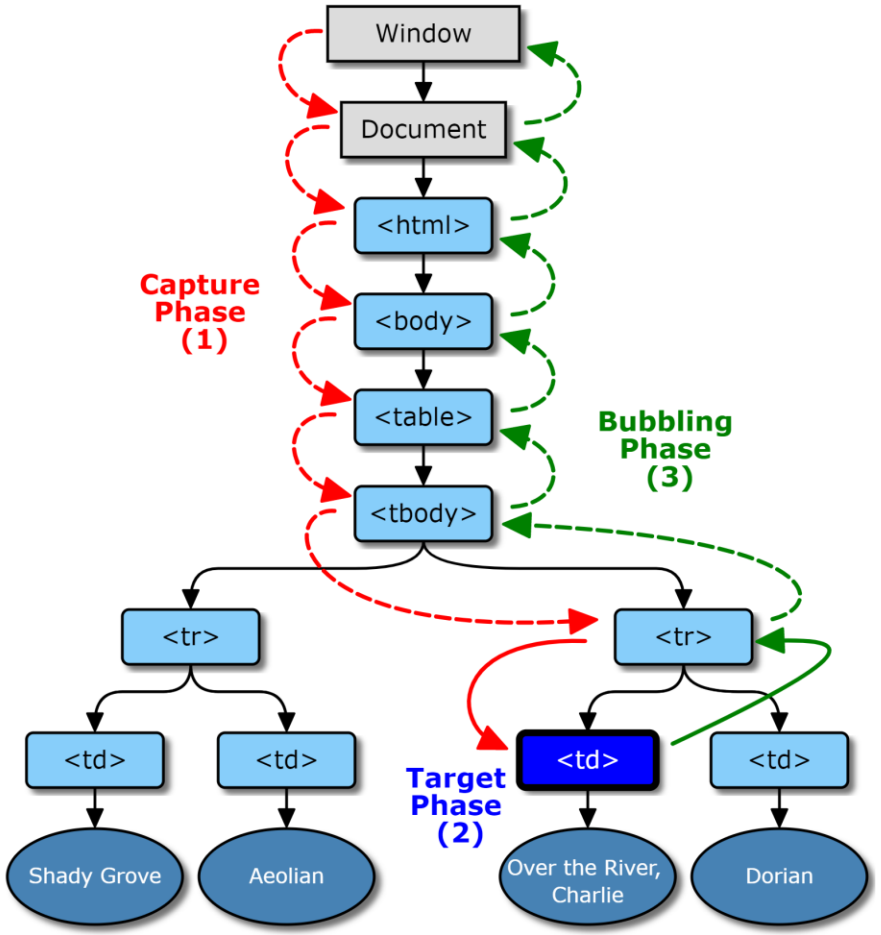


Event – wybrane atrybuty i metody

```
Event.target
Event.currentTarget
Event.type

Event.preventDefault()
Event.stopImmediatePropagation()
Event.stopPropagation()
```

Bubbling i propagacja



preventDefault()

Jeśli chcemy powstrzymać domyślne zachowanie elementów możemy wywołać metodę `event.preventDefault()`

Przykłady użycia

- “submit” formularza
- Blokowanie wybranych znaków w inputach
- Blokowanie inputów, np. Checkbox

`Event.preventDefault()` działa tylko na eventy, które mają atrybut `cancelable: true`

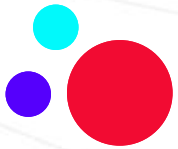
stopPropagation()

`event.stopPropagation()`

Powstrzymuje dalszą propagację eventu w ramach bubbling.

`event.stopImmediatePropagation()`

Powstrzymuje propagację do innych handlerów danego typu eventu na wybranym elemencie.



Usuwanie listenerów

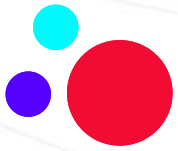
```
const handler = () => {}

el.addEventListener('click', handler)

el.removeEventListener('click', handler)
```

Nie można w ten sposób usunąć anonimowych handlerów!

infoShare ACADEMY



dispatchEvent – tzw. synthetic events

```
const event = new Event(‘zonk');

// Listen for the event.
elem.addEventListener(‘zonk', (e) => { /* … */ }, false);

// Dispatch the event.
elem.dispatchEvent(event);
```

CustomEvent

```
const event = new CustomEvent(‘zonk’, {detail: {bramka: 2}});

// Listen for the event.
elem.addEventListener(‘zonk', (event) => {
  console.log(‘zonk w bramce: ’, event.detail.bramka);
});

// Dispatch the event.
elem.dispatchEvent(event);
```

Obsługa myszy

https://developer.mozilla.org/en-US/docs/Web/API/Element#mouse_events

Przykładowe eventy

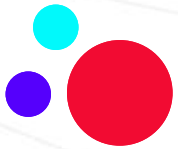
- `click`
- `contextmenu`
- `dblclick`
- `mousedown`
- `mouseenter`
- `mouseleave`
- `mousemove`
- `mouseout`
- `mouseover`
- `mouseup`

Obsługa klawiatury

keydown – naciśnięcie guzika
keyup – zwolnienie guzika
keypress Deprecated – nie używać

Przykład:

```
window.addEventListener("keydown", (event) => {
  switch (event.key) {
    case "Down": // IE/Edge specific value
    case "ArrowDown":
      // Do something for "down arrow" key press.
      break;
    default:
        return;
  }
});
```



Wzorzec: event delegation

Zamiast tworzyć event listener na wielu elementach, tworzymy jeden na wspólnym przodku (ancestor element), który analizuje szczegóły eventu i jego target, a następnie podejmuje właściwe działanie.

- Pozwala zebrać w jednym miejscu powiązany ze sobą kod
- Może mieć korzystny wpływ na wydajność strony
- Ułatwia współpracę z dynamicznie dodawanymi elementami DOM
- Wymaga precyzyjnego określenia elementów/atrybutów na które należy reagować – mogą w tym pomóc atrybuty data-* i dostęp do nich poprzez element.dataset.

```
<button data-action="save">Save</button>
[…]
(event) { let action = event.target.dataset.action;}
```

Dziękuję za uwagę!

infoShareAcademy.com