Zdarzenia w DOM-ie

Zdarzenia i funkcja callback

Eventy

- Są to wydarzenia odbywające się na naszej stronie WWW. Dzięki JavaScript jesteśmy w stanie przejąć kontrolę nad zdarzeniem i odpowiednio reagować.
- Zdarzenia dzielimy wedle rodzaju interakcji np. użycie myszki czy klawiatury, edycja formularza lub okna przeglądarki itp.
- W obiekcie event są zawarte informacje dotyczące danej akcji.

Callback

- Jest to specjalna funkcja, którą przekazujemy do wywołania. Nie jest uruchamiana od razu, lecz po wystąpieniu jakiegoś zdarzenia.
- Każda obsługa zdarzenia w JavaScript jest tworzona za pomocą funkcji typu callback.

Dodawanie nasłuchiwania zdarzeń do elementów

Obsługę zdarzeń dodajemy do elementu przez użycie metody: addEventListener(eventName, callback).

W przykładzie obok znajdujemy element button i dodajemy do niego obsługę zdarzenia - kliknięcia.

Jako drugi parametr metody addEventListener() przekazujemy anonimową funkcję, która zostanie wykonana po kliknięciu.

Kod HTML

```
<button id="counter">Click me!</button>
```

Kod JavaScript

```
var button = document.querySelector("button");
var clickCount = 0;
button.addEventListener("click", function (event) {
    clickCount += 1;
    console.log("Liczba kliknięć", clickCount);
});
```

Dodawanie nasłuchiwania zdarzeń do elementów

Nie musimy za każdym razem przekazywać funkcji anonimowej jako parametr metody addEventListener() - możemy zadeklarować funkcję wcześniej a wspomnianej metodzie przekazujemy jako drugi parametr nazwę naszej funkcji.

Popatrz na funkcję clickCounter(event) - przyjmuje ona parametr event - jest to obiekt reprezentujący dane zdarzenie. Więcej o nim w dalszej części prezentacji.

Dodawanie nasłuchiwania zdarzeń do elementów

```
Kod HTML
<button id="counter">Click me!</button>

Kod JavaScript
var button = document.querySelector("button");
var clickCount = 0;
function clickCounter(event) {
    clickCount += 1;
    console.log("Liczba kliknięć", clickCount);
}
button.addEventListener("click", clickCounter);
```

Dodawanie nasłuchiwania zdarzeń do elementów

Na jednym elemencie możemy nasłuchiwać wielu zdarzeń jednocześnie.

Kod HTML

```html
<button id="counter">Click me!</button>
```

Kod JavaScript

```javascript
var button = document.querySelector("button");
var clickCount = 0, randomWords = ['Some', 'Random', 'Words'];
function clickCounter(event) {
    clickCount += 1;
    console.log('Click number', clickCount);
}
function randomWord(event) {
    var myWord = randomWords[Math.floor(Math.random() * randomWords.length)];
    console.log(myWord);
}
button.addEventListener('click', clickCounter);
button.addEventListener('click', randomWord);
```

Usuwanie nasłuchiwania zdarzeń z elementów

Możemy też usunąć nasłuchiwanie z elementu. Robimy to za pomocą metody: removeEventListener(event, callback).

Nie da się usunąć nasłuchiwań, których obsługa odbywa się za pomocą funkcji anonimowych!

```
Kod HTML
<button id="counter">Click me!</button>

Kod JavaScript
var button = document.querySelector('button');
var clickCount = 0;
function clickCounter (event) {
console.log('Click number', clickCount);
clickCount += 1;
    if(clickCount >= 10) {
        event.target.removeEventListener('click', clickCounter);
    }
}
button.addEventListener('click', clickCounter);
```

Lista eventów

mouse:

mousedown, mouseup, click, dblclick, mousemove, mouseover, mouseout

key:

keydown, keypress, keyup

touch:

touchstart, touchmove, touchend, touchcancel

control:

resize, scroll, focus, blur, change, submit

no arguments:

load, unload, DOMContentLoaded

Pełna lista eventów: https://en.wikipedia.org/wiki/DOM_events

DOMContentLoaded

- **DOMContentLoaded** jest specjalnym zdarzeniem, uruchamiającym się w momencie załadowania drzewa DOM.
- Nasz cały kod JavaScript operujący na DOM powinien znajdować się wewnątrz obsługi tego zdarzenia. Inaczej nie mamy gwarancji, że element którego szukamy, został już stworzony!
- Jeżeli wykonujesz operacje na DOM, upewnij się, że całe drzewo DOM zostało stworzone!

```
document.addEventListener("DOMContentLoaded", function () {
    console.log("DOM fully loaded and parsed!");
});
```

this w eventach

- W każdym evencie mamy możliwość odwołania się do zmiennej this.
- Jest to specjalna zmienna reprezentująca element, na którym zostało wywołane zdarzenie.
- Jest ona szczególnie przydatna, jeżeli taki sam event nastawiamy na wiele elementów.
- W przykładzie poniżej, w jednym miejscu zakładamy nasłuchiwanie zdarzeń na wszystkie przyciski.
- Zmieni się kolor tylko tego przycisku, w który klikamy - this wskazuje na element, na którym wystąpiło zdarzenie.

Kod HTML

```html
<button class="btn">Click me!</button>
<button class="btn">Click me!</button>
<button class="btn">Click me!</button>
```

Kod JavaScript

```javascript
var buttons = document.querySelectorAll(".btn");
for(var i = 0; i < buttons.length; i++) {
    buttons[i].addEventListener("click", function(event) {
        this.style.backgroundColor = "red";
    });
}
```

Obiekt event

Każde zdarzenie jest opisywane przez specjalny obiekt event. Dzięki niemu możemy dowiedzieć się wielu przydatnych rzeczy na temat zdarzenia, które wystąpiło. Np.:

event.currentTarget – zwraca element, na którym wywołane zostało zdarzenie,

event.target – zwraca element, który spowodował wywołanie eventu; w przypadku propagacji jest to element potomka,

event.timeStamp – zwraca czas, w którym został wywołany event,

event.type – zwraca typ eventu (jako ciąg znaków).

Obiekt event ma jeszcze kilka przydatnych metod:

event.preventDefault() – anuluj standardową akcję elementu,

event.stopPropagation() – anuluj wszystkie zdarzenia tego samego typu z elementów nadrzędnych,

event.stopImmediatePropagation() – anuluj wszystkie zdarzenia tego samego typu przypięte do tego elementu oraz wszystkich elementów nadrzędnych.

Propagacja zdarzeń

W DOM mamy do czynienia z propagacją zdarzeń. Polega ona na przekazywaniu informacji o zdarzeniu w drzewie DOM.

Event bubbling

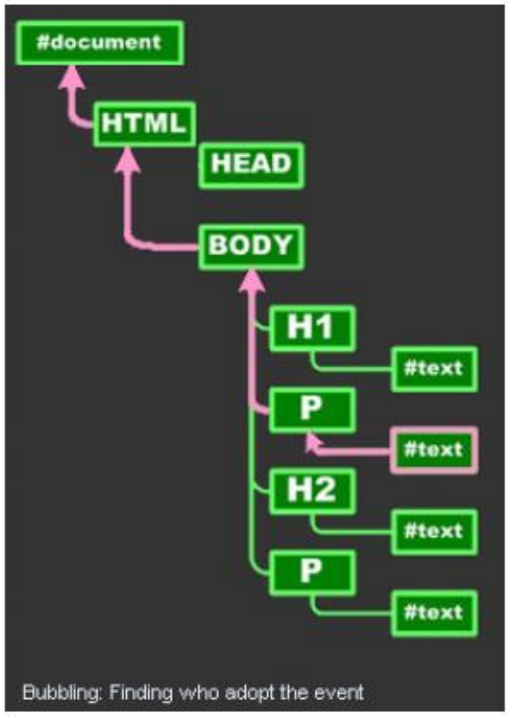
Jest to przekazywanie danych w górę drzewa DOM - domyślny rodzaj propagacji dla większości zdarzeń.

Event capturing

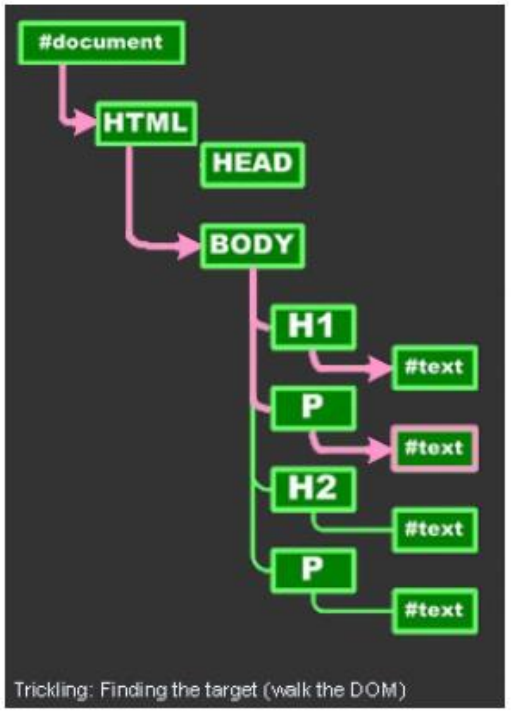
Jest to przekazywanie danych w dół drzewa DOM - opcja standardowa dla niektórych zdarzeń (np. focus). Metoda trochę szybsza dla wielkich drzew DOM.

Bubbling vs capturing

Event bubbling



Event capturing



Propagacja zdarzeń

```
Kod HTML
<div id="foo">
    <button id="bar">Click me!</button>
</div>

Kod JavaScript
document.querySelector('#foo').addEventListener('click', function () {
    console.log('Event zarejestrowany, element #foo');
});
document.querySelector('#bar').addEventListener('click', function () {
    console.log('Event zarejestrowany, element #bar');
});
```

MouseEvent

Jest to specjalny typ zdarzenia, tworzony podczas używania myszy. Rozszerza on podstawowy obiekt event o następujące właściwości:

event.button – zwraca przycisk myszki, który został naciśnięty,

event.clientX – zwraca koordynat X (horyzontalny) myszki, relatywnie do górnego, lewego rogu strony,

event.clientY – zwraca koordynat Y (wertykalny) myszki relatywnie do górnego, lewego rogu strony,

event.screenX – zwraca koordynat X (horyzontalny) myszki, relatywnie do górnego, lewego rogu okna,

event.screenY – zwraca koordynat Y (wertykalny) myszki, relatywnie do górnego, lewego rogu okna.

KeyboardEvent

Jest to specjalny typ zdarzenia, tworzony podczas używania klawiatury. Rozszerza on podstawowy obiekt event o następujące właściwości:

event.altKey – zwraca true, jeżeli alt był naciśnięty,

event.ctrlKey – zwraca true, jeżeli ctrl był naciśnięty,

event.shiftKey – zwraca true, jeżeli shift był naciśnięty.

event.charCode – zwraca znak opisujący klawisz, który wywołał event jako kod klawisza będący liczbą,

event.key – zwraca który klawisz, wywołał event (np. "d" -> "d", "Ctrl" -> "Control")

event.keyCode – zwraca wartość klawisza, który wywołał zdarzenie (np. "d" -> "68")

Czas na zadania