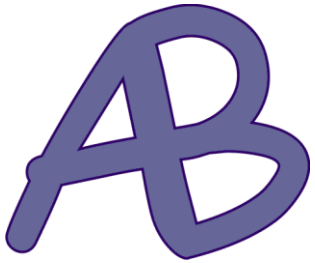
Aplikacje frontendowe



Wykład

Model zdarzeń w DOM

Model zdarzeń

Zdarzenie to obiekt reprezentujący akcję użytkownika lub przeglądarki

- ✓ Przykłady zdarzeń to:. kliknięcie myszą, naciśnięcie klawisza, załadowanie strony itp.
- ✓ Zdarzenia są rozsyłane w kontekście drzewa DOM, co umożliwia propagację.

W modelu zdarzeń DOM każde zdarzenie przechodzi przez trzy fazy:

- Capturing phase (faza przechwytywania) – zdarzenie propaguje się od korzenia dokumentu (np. document) w dół drzewa DOM do elementu docelowego.
- Target phase – zdarzenie osiąga element docelowy i uruchamiane są listenery na tym elemencie.
- Bubbling phase (faza bąbelkowania) – zdarzenie przemieszcza się z powrotem w górę drzewa DOM aż do document

Model zdarzeń

1. Zdarzenia dotyczące myszy (Mouse Events)

- `click`: Wyzwalane, gdy element zostanie kliknięty.
- `dblclick`: Wyzwalane, gdy element zostanie podwójnie kliknięty.
- `mousedown`: Wyzwalane, gdy przycisk myszy zostanie wciśnięty na elemencie.
- `mouseup`: Wyzwalane, gdy przycisk myszy zostanie zwolniony na elemencie.
- `mousemove`: Wyzwalane, gdy kursor myszy porusza się nad elementem.
- `mouseover`: Wyzwalane, gdy kursor myszy najedzie na element.
- `mouseout`: Wyzwalane, gdy kursor myszy opuści element.

Model zdarzeń

2. Zdarzenia dotyczące klawiatury (Keyboard Events)

- `keydown`: Wyzwalane, gdy klawisz zostaje wciśnięty.
- `keyup`: Wyzwalane, gdy klawisz zostaje zwolniony
- `keypress`: Wyzwalane, gdy klawisz zostaje wciśnięty i zwolniony (znak klawisza)

3. Zdarzenia formularzy (Form Events)

- `submit`: Wyzwalane, gdy formularz zostanie wysłany.
- `change`: Wyzwalane, gdy wartość elementu formularza zostanie zmieniona.
- `input`: Wyzwalane, gdy wartość pola formularza jest zmieniana.
- `focus`: Wyzwalane, gdy element otrzymuje fokus.
- `blur`: Wyzwalane, gdy element traci fokus.

Model zdarzeń

4. Zdarzenia okna (UI Events)

- `load`: Wyzwalane, gdy strona w pełni się załaduje.
- `resize`: Wyzwalane, gdy rozmiar okna zostanie zmieniony.
- `scroll`: Wyzwalane, gdy użytkownik przewija stronę.
- `unload`: Wyzwalane, gdy użytkownik opuszcza stronę

Słuchacze zdarzeń

Słuchacze zdarzeń

Metoda addEventListener() służy do dodawania obsługi zdarzeń do wybranych elementów DOM. Pozwala na przypisanie funkcji, która zostanie wywołana, gdy wystąpi określone zdarzenie na danym elemencie.

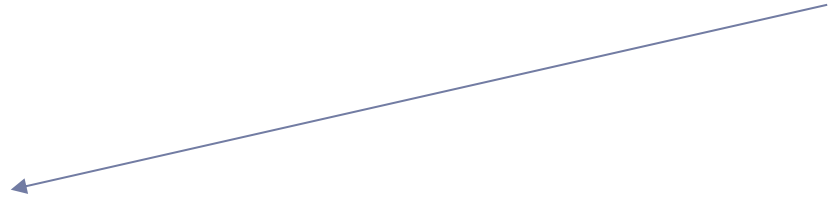
```
element.addEventListener(type, listener, options);
```

Parametry:

- type: typ zdarzenia zapisany jako ciąg znaków (nazwa zdarzenia bez prefiksu), np. 'click', 'mouseover', 'keydown’.
- listener: funkcja, która zostanie wywołana, gdy zdarzenie wystąpi.
- options (opcjonalnie): opcje logiczne lub obiekt { capture, once, passive }

Słuchacze zdarzeń

```
element.addEventListener(type, listener, options);
```



Opcje:

- once: listener zostanie automatycznie usunięty po pierwszym wywołaniu.
- capture: decyduje, czy listener działa w fazie przechwytywania (true) czy bąbelkowania (false - domyślnie).
- passive: informuje przeglądarkę, że listener nie wywoła preventDefault() (przydatne np. przy scroll).

Słuchacze zdarzeń

Obsługa kliknięcia na przycisku:

```
<button id="myButton">Kliknij mnie</button>
<script>
  const button = document.getElementById("myButton");

  if (button) {
    button.addEventListener("click", () => {
      alert("Przycisk został kliknięty!");
    });
  } else {
    console.warn("Element #myButton nie istnieje w DOM");
  }
</script>
```

Sprawdzenie, czy przycisk istnieje

Rejestrowane jest zdarzenie click na przycisku.

Słuchacze zdarzeń

Przykład: dynamiczna zmiana stylu po kliknięciu

```
<button id="btn">Zmień tło</button>
<div id="box" class="box"></div>
<style>
  .highlight {
    background-color: yellow;
  }
  .box {
    width: 100px;
    height: 100px;
    border: 1px solid #333;
  }
</style>
<script>
  const btn = document.getElementById("btn");
  const box = document.getElementById("box");
  btn.addEventListener("click", () => {
    box.classList.toggle("highlight");
  });
</script>
```

Rejestrowane jest zdarzenie click na przycisku.

Po kliknięciu wywoływana jest metoda classList.toggle('highlight') na box, która:

- doda klasę highlight, jeśli jej nie było,
- usunie klasę highlight, jeśli już była.

Słuchacze zdarzeń

Obsługa zdarzeń klawiatury:

```
<script>
  document.addEventListener("keydown", function (event) {
    console.log("Naciśnięto klawisz:", event.key); // np. "Enter"
    if (event.key === "Enter") {
      console.log("Użytkownik wcisnął Enter");
    }
  });
</script>
```

Obsługa najechania myszką na element:

```
let div = document.getElementById("myDiv");
div.addEventListener("mouseover", function () {
  div.style.backgroundColor = "yellow";
});
div.addEventListener("mouseout", function () {
  div.style.backgroundColor = "white";
});
```

Słuchacze zdarzeń

Przykład: „edytor tekstu”

```
<style>
  div {
    border: 1px solid black;
    padding: 10px;
    margin: 10px;
    font-size: 20pt;
    min-height: 200px;
  }
</style>

<div></div>
```

```
<script>
  // Przykład: pisanie wewnątrz kontenera
  const container = document.querySelector("div");

  document.addEventListener("keydown", function (event) {
    // event.key zawiera nazwę klawisza, który został naciśnięty
    let keyName = event.key;
    if (keyName === "Enter") {
      keyName = "<br>"; // nowa linia
    } else if (keyName === " ") {
      keyName = " "; // spacja
    } else if (keyName.length > 1) {
      keyName = ""; // ignorowanie klawiszy funkcyjnych
    }
    container.innerHTML += keyName;
    // dodanie do kontenera
  });
</script>
```

Słuchacze zdarzeń

Reakcja na zmianę rozmiaru okna

```
<script>
  window.addEventListener("resize", function (event) {
    console.log("Nowa szerokość:", window.innerWidth);
    console.log("Nowa wysokość:", window.innerHeight);
  });
</script>
```

Słuchacze zdarzeń

Obsługa wielu zdarzeń dla jednego elementu:

```
let input = document.getElementById("myInput");
input.addEventListener("focus", function () {
  console.log("Pole zostało zaznaczone.");
});
input.addEventListener("blur", function () {
  console.log("Pole zostało odznaczone.");
});
```

Przekazywanie danych do funkcji obsługi zdarzeń:

```
function handleEvent(event) {
  console.log(
    `Zdarzenie typu: ${event.type} na elemencie o id: ${event.target.id}`
      );
    }
    let link = document.getElementById("myLink");
    link.addEventListener("click", handleEvent);
```

Słuchacze zdarzeń

- ✓ Na ten sam element można dodać wiele słuchaczy zdarzeń dla tego samego lub różnych typów zdarzeń.
- ✓ Aby usunąć nasłuchiwanie zdarzenia, używa się metody removeEventListener, która wymaga tych samych argumentów co addEventListener.
- ✓ Wewnątrz funkcji wywoływanej przez addEventListener, this odnosi się do elementu, który wywołał zdarzenie.
- ✓ Używanie addEventListener jest preferowane w stosunku do atrybutów HTML, takich jak onclick, ponieważ pozwala na dodanie wielu słuchaczy do jednego elementu i lepszą organizację kodu.

Słuchacze zdarzeń

Metoda event.preventDefault() służy do zablokowania domyślnego zachowania przeglądarki związanego z danym zdarzeniem.

Jest ona przydatna w sytuacjach, gdy chcemy przejąć kontrolę nad reakcją interfejsu użytkownika, np. anulować wysyłanie formularza, zablokować nawigację po kliknięciu w link, czy uniemożliwić przewijanie.

```
<a href="https://bartoszewskia.github.io/uthrad/">Moja strona</a>
<script>
  const link = document.querySelector("a");
  link.addEventListener("click", function (event) {
    event.preventDefault(); // nie przechodzi do wskazanego adresu
    console.log("Link został zablokowany");
  });
</script>
```

Słuchacze zdarzeń

Metoda event.preventDefault()

```
// Zatrzymanie wysyłki formularza
form.addEventListener('submit', function(e) {
  e.preventDefault(); // bez tego strona się przeładuje
});


// Blokowanie kliknięcia w link
link.addEventListener('click', function(e) {
  e.preventDefault(); // link nie zadziała
});
```

Usuwanie słuchacza zdarzeń

Usuwanie słuchacza zdarzeń

Po dodaniu do słuchacza opcji once: true zostanie on automatycznie usunięty po pierwszym wywołaniu.

```
<button id="myButton">Kliknij mnie</button>
<script>
  const btn = document.querySelector("#myButton");

  // Przykład z użyciem opcji once
  btn.addEventListener(
    "click",
    () => {alert("Pierwszy raz kliknięto przycisk");},
    { once: true }
  );
</script>
```

Usuwanie słuchacza zdarzeń

Do usuwania słuchacza służy metoda removeEventListener()

```
element.removeEventListener(type, listener, options)
```

Aby użyć tej metody musimy dysponować referencją do listenera — nie można usunąć funkcji anonimowej

```
<button id="myButton">Kliknij mnie</button>
<script>
  const btn = document.querySelector("#myButton");

  function sluchaczKlikniecia(event) {
    alert("Kliknięto przycisk");
  }

  btn.addEventListener("click", sluchaczKlikniecia);
  // ...
  btn.removeEventListener("click", sluchaczKlikniecia);
</script>
```

Słuchacz zdefiniowany jako zwykła funkcja

Propagacja zdarzeń

Propagacja zdarzeń

Fazy zdarzenia

1. Capturing phase: z drzewa root do elementu docelowego
2. Target phase: wywołanie listenerów na elemencie docelowym
3. Bubbling phase: od elementu docelowego w górę drzewa

Propagacja zdarzeń

```
element.addEventListener(type, listener, options);
```

Opcja capture: decyduje, czy listener działa w fazie przechwytywania (true) czy bąbelkowania (false - domyślnie).

capture = true oznacza, że słuchacz uruchomi się podczas fazy przechwytywania

Ma to znaczenie gdy zdarzenia przepisane są zarówno do elementu będącego rodzicem jak i jego potomka

Propagacja zdarzeń

```
<script>
    const parent = document.querySelector("#parent");
    const child = document.querySelector("#child");

    // Listener działający w fazie przechwytywania
    parent.addEventListener(
      "click",
      () => {
        console.log("Kliknięto rodzica (capture)");
      },
      true
    ); // capture = true oznacza, że handler uruchomi się podczas fazy przechwytywania

    // Listener działający domyślnie w fazie bubbling
    child.addEventListener("click", () => {
      console.log("Kliknięto dziecko");
    });
  </script>
```

```
<style>
  div {
    border: 1px solid #333;
    padding: 10px;
    margin: 10px;
  }
</style>
<div>
  <div id="parent">
    <p>Rodzic</p>
    <div id="child">Dziecko</div>
</div>
```

Po kliknięciu na element #child, przeglądarka najpierw przechodzi przez fazę przechwytywania i wywołuje słuchacza przypisanego do rodzica (#parent).

Dopiero później słuchacza dziecka.

Jeśli opcja capture nie byłaby ustawiona, log kliknięto rodzica pojawiłby się po logu z dziecka.

Propagacja zdarzeń

Kontrola propagacji

- event.stopPropagation(): zatrzymuje dalszą propagację
- event.stopImmediatePropagation(): zatrzymuje propagację i kolejne listenery na tym samym elemencie

```
element.addEventListener('click', (e) => {
  e.stopPropagation();       // blokuje dalsze propagowanie
});
```

Literatura:

- Negrino Tom, Smith Dori, ***Po prostu JavaScriopt i Ajax, wydanie VI***, Helion, Gliwice 2007.
- Lis Marcin, JavaScript, Ćwiczenia praktyczne, wydanie II, Helion, Gliwice 2007.
- http://www.w3schools.com/JS/js_popup.asp
- Beata Pańczyk, wykłady opublikowane na stronie http://www.wykladowcy.wspa.pl/wykladowca/pliki/beatap