INVEST IN POMERANIA ACADEMY

Podstawy HTML, CSS, Bootstrap, JS i Ajax

Część 3

Rzeczpospolita
Polska

URZĄD MARSZAŁKOWSKI
WOJEWÓDZTWA POMORSKIEGO

Unia Europejska
Europejski Fundusz
Rozwoju Regionalnego

HELLO

Krzysztof Drzymalski

Frontend Developer

infoShare ACADEMY

Agenda

1. Elementy i DOM
2. Eventy
3. Ajax
4. Podsumowanie
5. Dodatkowe zagadnienia
    - Przydatne funkcje tablicowe
    - Bootstrap Icons

Elementy i DOM

Ingerowanie w HTML

infoShareAcademy.com

DOM

Document Object Model (DOM) jest to model, który reprezentuje HTML'a w formie obiektu. Często można się spotkać z określeniem, że jest to taki interfejs/API do HTML'a. Umożliwia on połączenie HTML'a z warstwą JS'a i manipulowanie nim.

Za pomocą DOM'a jesteśmy w stanie dodawać, usuwać, zmieniać itp. Elementy HTML z poziomu JS'a

Dostęp do całego DOM mamy poprzez **window.document**

Referencja do elementu

Aby pobrać referencję do danego elementu musimy go jakoś „znaleźć”. Mogą nam do tego posłużyć wbudowane w JS metody, jedne z częściej używanych to:

- getElementById
- getElementsByClassName
- getElementsByTagName

Odczytywanie danych z elementów

Aby odczytać dane z elementu najpierw pobieramy do niego referencję a następnie jesteśmy w stanie odczytać jego zawartość:

- innerHTML – odczytuje cały HTML
- innerText – odczytuje jedynie tekst z wnętrza elementu

```
<div id="myDiv">
  My div
</div>

<script>
  document.getElementById("myDiv").innerText
</script>
```

```
<div id="myDiv">My div</div>

<script>
  document.getElementById("myDiv").innerHTML
</script>
```

Manipulowanie DOM'em

Tworzenie elementu

```
<div id="container"></div>
<script>
  const paragraph = document.createElement("p");
  paragraph.innerHTML = "Lorem ipsum"
  document.getElementById("container").appendChild(paragraph);
</script>
```

Zmiana zawartości elementu

```
<p id="paragraph">Lorem ipsum</p>
<script>
  document.getElementById("paragraph").innerHTML = "New tekst"
</script>
```

Usuwanie elementu

```
<div id="container">
  <p id="paragraph">Lorem ipsum</p>
</div>
<script>
  document.getElementById("container").remove()
</script>
```

Eventy

Wydarzenia w naszej aplikacji

infoShareAcademy.com

Eventy

Mamy zasadniczo 2 sposoby jak dodać obsługę zdarzenia do elementu HTML:

- Inline do elementu w HTML
- Z poziomu JavaScript

Z poziomu JS

```
<a target="_blank" id="stackLink" href="https://stackoverflow.com/">Link do stacka</a>
<script>
    const linkRef = document.getElementById('stackLink');

    linkRef.addEventListener("click", (event) => {
        console.log(event)
    });
</script>
```

Inline

```
<button onclick="myEventHandler(event)">Inline event</button>
```

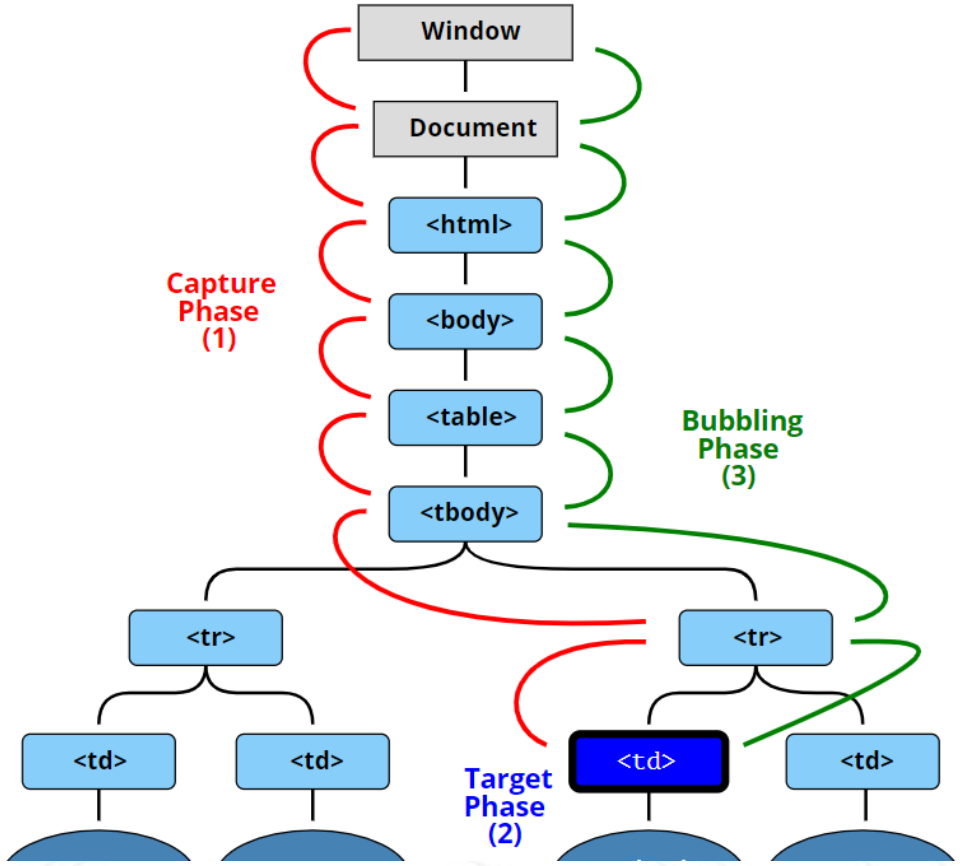
Bubbling & Capturing

Każdy standardowy UI Event posiada 3 fazy przebiegu:

- **Capturing (Przechwytywanie)** – posiada cyfrę 1 jako oznaczenie i jest to faza, w której event podróżuje „w głąb" drzewa DOM aż do elementu który jest inicjatorem zdarzenia
- **Target (Cel)** – posiada cyfrę 2 jako oznaczenie i jest to faza w której event dotarł do właściwego elementu inicjatora
- **Bubbling (bąbelkowanie)** – posiada cyfrę 3 jako oznaczenie i jest to faza w której event podróżuje „wzwyż" drzewa DOM aż do elementu html. Event będzie się bąbelkował tylko jeśli ma właściwość bubbling ustawiona na true.

Interaktywne przykłady: https://www.w3schools.com/js/js_events.asp

Bubbling & Capturing



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

Zadanie TODO list

- Otwórz plik zadanie11.html
- Utwórz input typu tekstowego
- Utwórz przycisk i dodaj do niego obsługę zdarzenia **click**
- Po kliknięciu w przycisk w kontenerze poniżej ma się dodać tekst z inputa jako nowy element (TODO list)
- Po dodaniu nowego elementu wartość inputa ma się wyczyścić

Ajax

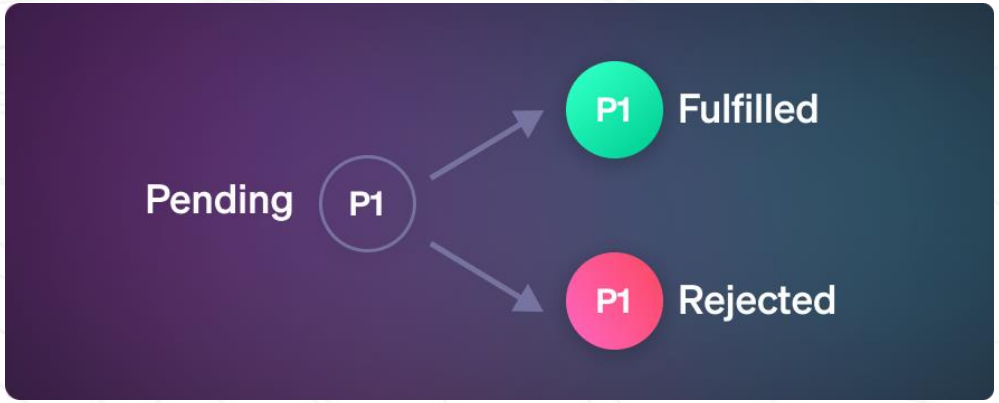
Asynchroniczne akcje

Promise API

Obiekty Promise (inaczej obietnice) reprezentują **wynik** działania operacji asynchronicznej

Każdy Promise będzie dostępny w jednym z 3 stanów:

- **Pending** – stan inicjacji, nie jest ani zakończone sukcesem ani porażką
- **Fulfilled** – operacja jest zakończona sukcesem
- **Rejected** – operacja jest zakończona porażką



Źródło: Tabnine

Dokumentacja Promise https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Obsługa obietnic

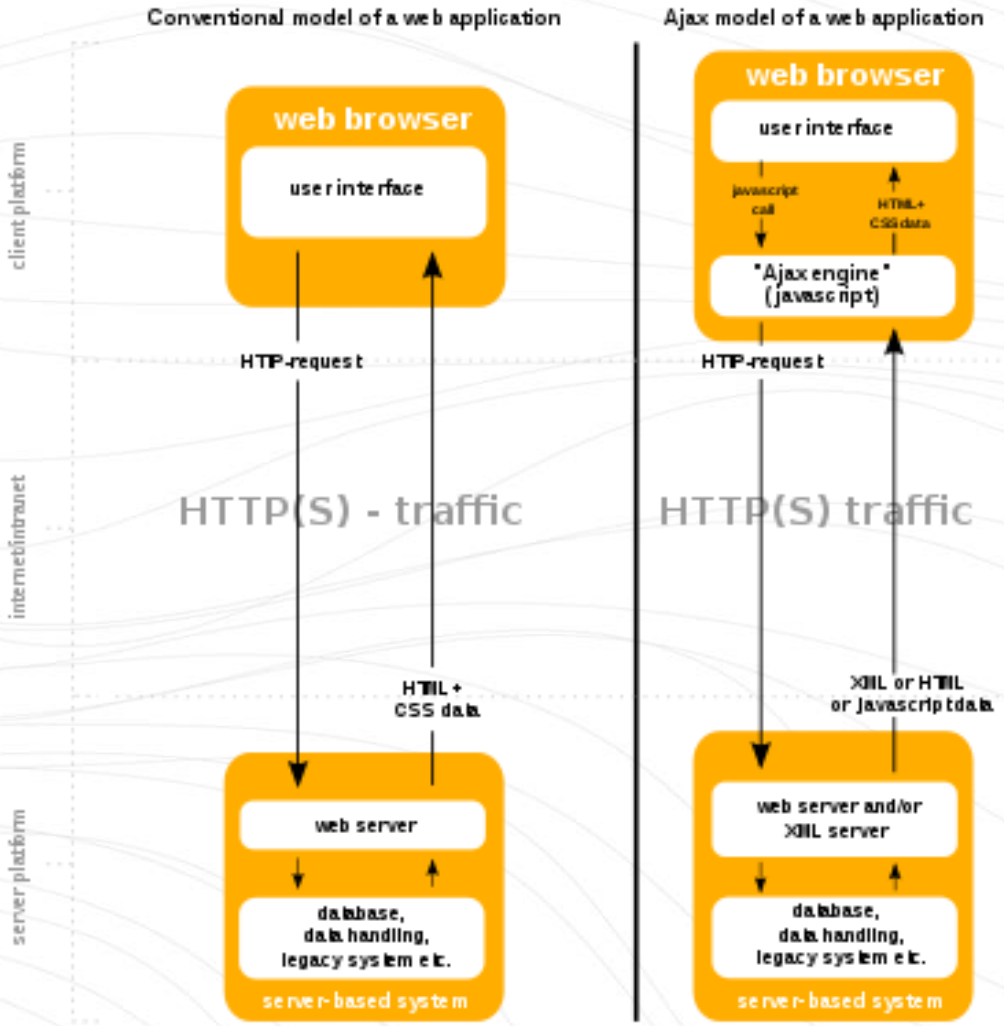
Aby odczytać dane za pomocą obietnic należy skorzystać z dostępnych metod:

- **Then** – Funkcja zostanie wykonana gdy obietnica zostanie spełniona (Fulfilled)
- **Catch** - Funkcja zostanie wykonana gdy obietnica zostanie odrzucona (Rejected)
- **Finally** - Funkcja zostanie wykonana niezależnie od wyniku obietnicy

Ajax

Asynchronous JavaScript and XML (AJAX)

Streszczając i przekładając to na język ludzki AJAX jest zbiorem technik służących do dynamicznej komunikacji czyli ściągania i wysyłania danych bez przeładowywania strony



Źródło: Wikipedia

Stary sposób na obsługę AJAX wrzucam jako ciekawostkę dla chętnych: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

Dokumentacja Ajax: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

Fetch API

Obecnie powszechnie do komunikacji AJAX stosuje się **Fetch API** czyli nowy interfejs programistyczny, który jest bardziej intuicyjny oraz usprawnia pracę stosując w swoich mechanizmach **Promise API.**

```
<script>
  fetch('https://jsonplaceholder.typicode.com/todos/10')
          .then(response => response.json())
          .then(json => console.log(json))
</script>
```

API użyte w przykładzie: https://jsonplaceholder.typicode.com/

Zadanie pobieranie danych

- Otwórz plik zadanie12.html
- Utwórz przycisk i dodaj mu obsługę zdarzenia **click**
- Po kliknięciu, w przycisk pobierz dane z tego adresu url: https://swapi.dev/api/planets
- Z pobranych danych wybierz tylko 3 pola: **name, climate, population**
- Dane wyświetl wierszami jedne pod drugimi

04. Podsumowanie

Zwięzła powtórka całego materiału

Podsumowanie

- **HTML** jest językiem znaczników, który opisuje znaczenie poszczególnych treści, pozwala linkować do innych stron i umieszczać multimedia
- **Elementy** blokowe zabierają całe miejsce dostępne w linii a liniowe zajmują tylko tyle miejsca ile potrzebują
- **Formularze** służą do zebrania danych od użytkownika
- **CSS** jest językiem arkuszy stylów, które odpowiadają jak dany element będzie wyświetlany na stronie
- **Selektory** CSS mają określoną hierarchię. Przeglądarka wybiera style z wyższym specificity (siłą selektora)
- **Pozycjonowanie** elementów możemy osiągnąć przy pomocy właściwości **position** lub **flexboxa**
- **JS** to kompletny język programowania, za jego pomocą wprowadzamy interaktywność do naszej aplikacji
- **Bootstrap** to framework CSS, dzięki któremu za pomocą gotowych styli CSS jesteśmy w stanie budować naszą aplikację „jak z klocków”
- **Bootstrap** oferuje zestaw gotowych komponentów, których możemy używać na zasadzie „kopiuj wklej”
- **Responsywność** to dostosowywanie się strony do różnych rozdzielczości ekranu

Podsumowanie

- **Inspektor** jest częścią narzędzi programistycznych wbudowanych w przeglądarkę. Pozwala podejrzeć kod danej strony oraz wprowadzić zmiany w nim oraz zasymulować rozdzielczość urządzenia.
- **JS** jest językiem typowanym **słabo** oraz **dynamicznie**
- **Operatory** służą do wykonywania operacji na wartościach i dzielą się na:
  - Arytmetyczne
  - Przypisania
  - Porównania
  - Logiczne
  - Jednowartościowe
- **Event handlery** pozwalają obsługiwać wydarzenia, które mają miejsce w naszej aplikacji
- **Operacje asynchroniczne** jesteśmy w stanie obsługiwać dzięki Promise API
- **Ajax** to zbiór technik służących do dynamicznej komunikacji czyli ściągania i wysyłania danych bez przeładowywania strony
- **Fetch API** to nowoczesny sposób na komunikację **AJAX**

05. Dodatkowe zagadnienia

Rozszerzenie materiału dla chętnych

Bootstrap icons

Zestaw gotowych ikonek, których możemy używać w naszych aplikacjach. Jesteśmy w stanie:

- Wstawiać bezpośrednio do kodu jako SVG
- Wstawiać do kodu za pomocą tagu z odpowiednią klasą
- Pobrać na dysk i wstawiać za pomocą tagu <img/>

Link do dokumentacji: https://icons.getbootstrap.com/

Przydatne metody tablicowe

W przypadku zaawansowanych metod należy do wnętrza wywołania przekazać funkcję jako argument (callback) jest on wywoływany dla każdego z elementów tablicy na której używamy metody. Niektóre z takich metod:

- ForEach – iteruje po tablicy (można powiedzieć że jest takim innym typem pętli for) co ważne pomimo że iteracja odbywa się jak w zwykłej pętli to nie działają w niej słowa kluczowe break i continue
- Map – również iteruje po tablicy jednakże zwraca wynik dla każdego z elementów
- Filter – iteruje po tablicy i zwraca nową tablicę w której znajdują się tylko elementy pozytywnie przechodzące test podany jako callback
- Some - zwraca wartość true jeśli przekazany warunek ma wartości true dla przynajmniej jednego elementu tablicy, w przeciwnym razie dostaniemy false
- Every - zwraca wartość true jeśli przekazany warunek ma wartości true dla każdego elementu tablicy, w przeciwnym razie dostaniemy false

Array: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Przydatne linki

- Jak sprawdzić wsparcie przeglądarek -> https://caniuse.com/
- Świetna książka o JS -> https://github.com/getify/You-Dont-Know-JS
- Zbiór darmowych API w Internecie: https://github.com/public-apis/public-apis
- Apka która pokazuje kod klawisza po naciśnięciu -> https://keycode.info/

infoShare ACADEMY

Invest in Pomerania ACADEMY

Do zobaczenia

Fundusze Europejskie Program Regionalny

Rzeczpospolita Polska

URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO

Unia Europejska Europejski Fundusz Rozwoju Regionalnego