INVEST IN POMERANIA ACADEMY

Podstawy HTML, CSS, Bootstrap, JS i Ajax

Część 2

Rzeczpospolita
Polska

URZĄD MARSZAŁKOWSKI
WOJEWÓDZTWA POMORSKIEGO

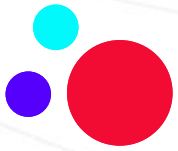
Unia Europejska
Europejski Fundusz
Rozwoju Regionalnego

HELLO

Krzysztof Drzymalski

Frontend Developer

infoShare ACADEMY



Agenda

1. Layout przy użyciu Bootstrap’a
2. Przydatne komponenty Bootstrap’a
3. Podstawy JavaScript
4. Typy danych i zmienne
5. Pętle i funkcje
6. Operatory i instrukcje warunkowe

01.

Layout przy użyciu Bootstrapa

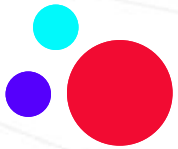
Responsywny design

infoShareAcademy.com

Responsywność

Pojęcie responsywności oznacza dostosowywanie się strony do różnych rozdzielczości ekranu. Często można spotkać skrót **RWD** znaczący **Responsive Web Design**

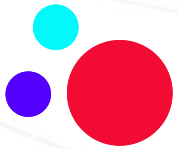
Źródło: s90.pl



Kontenery

- **container** – ustawia nam max-width na każdym responsywnym breakpoincie.
- **container-{breakpoint}** - szerokość 100% do konkretnego **breakpointa**, po czym nakładany jest max-width dla każdego z wyższych breakpointów. Przykład: **.container-sm** posiada width: 100% do 576px, po czym nakładany ma max width każdego z kolejnych breakpointów.
- **container-fluid** – zawsze 100% szerokości niezależnie od breakpointów.

Oficjalne demo: https://getbootstrap.com/docs/5.2/examples/grid/#containers



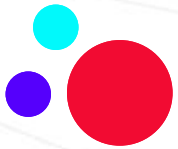
Lista breakpointów

Responsive Breakpoint – jest to punkt, w którym strona internetowa dostosuje się w pewien sposób do rozmiarów ekranu

Bootstrap udostępnia nam kilka predefiniowanych breakpointów

Breakpoint	Class infix	Dimensions
Extra small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

https://getbootstrap.com/docs/5.2/layout/grid/#all-breakpoints



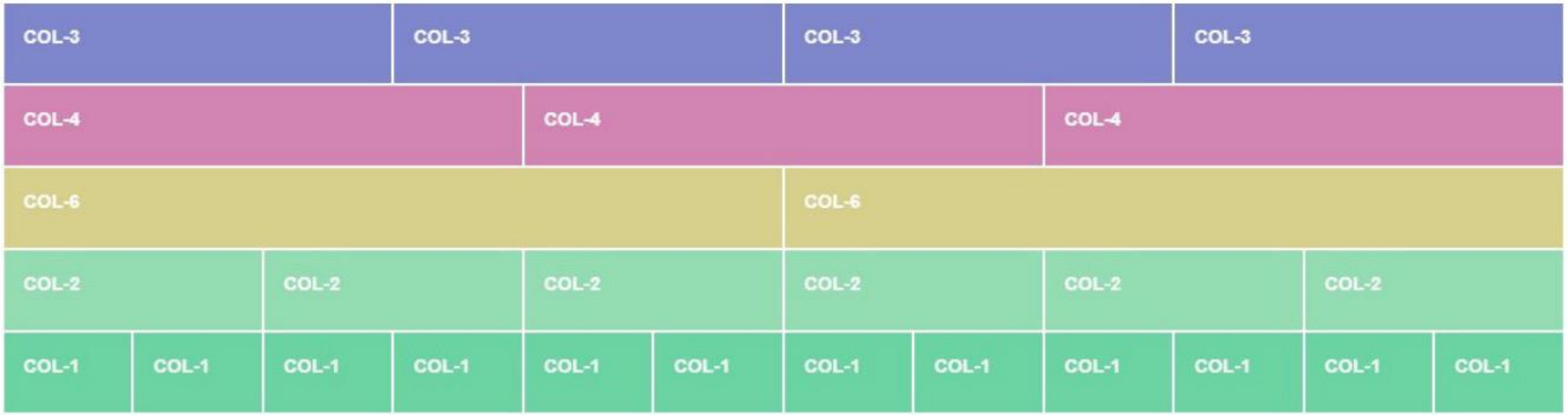
Grid system

Jest to system stworzony w celu wspierania w budowaniu layoutów typu **mobile first.**
Opiera się na kilku założeniach:

- Kontenery są wyśrodkowane
- Responsywne działanie opiera się na 6 breakpointach (opisane slajd wcześniej)
- Kolumny są zawarte w wierszach
- Rozmiar kolumn sumuje się do 12
- Jesteśmy w stanie dodać przerwy między kolumnami (gutters)

Grid w dokumentacji: https://getbootstrap.com/docs/5.2/layout/grid/

Grid system



Grid w dokumentacji: https://getbootstrap.com/docs/5.2/layout/grid/

Podstawowy Layout

Całość zbudowana tylko na divach ale z odpowiednimi klasami i w odpowiedniej kolejności zagnieżdżeń:

- container
- row
- col

```
<div class="container-fluid bg-primary">
  <h1>Nagłówek</h1>
</div>
<div class="container bg-success">
  <div class="row bg-secondary">
    <div class="col bg-info">Kolumna 1</div>
    <div class="col bg-danger">Kolumna 2</div>
    <div class="col bg-warning">Kolumna 3</div>
  </div>
  <div class="row">Rząd 2</div>
</div>
```

Predefiniowane kolory: https://getbootstrap.com/docs/5.2/utilities/background/#background-color

Zadanie Layout

- Otwórz plik zadanie6.html
- Za pomocą grid systemu jaki oferuje nam bootstrap odwzoruj strukturę jak z obrazka poniżej
- Wartości w nawiasach podpowiadają jaką wielkość ma dana komórka
- Nie dodawaj własnych styli/klas korzystaj jedynie z tego co oferuje nam bootstrap

Czas: 15min

Przydatne komponenty

Czyli jak nie wynajdywać koła na nowo

infoShareAcademy.com

Buttons

Gotowy komponent, możliwy do użycia na zasadzie kopiuj wklej, jego kilka cech:

- Posiada predefiniowane style dla przycisków
- Duże możliwości modyfikacji i dostosowania do swoich celów
- Klasy stworzone z myślą o elemencie <button></buton> ale jesteśmy w stanie także je zaaplikować do takich elementów jak <a> lub <input />

Oficjalne demo: https://getbootstrap.com/docs/5.2/components/buttons/

Formularze

- **Formularze** tworzone są z elementów typu form control, select, checks & radios, range
- **Form control** to nic innego jak input I textarea, czyli krótkie i dłuższe pola do wpisywania treści
- **Select** to komponent stworzony by wybrać wartość z predefiniowanej listy elementów
- **Check & radios** pola wyboru wartości, jedno lub wielokrotnego wyboru
- **Range** pole wyboru wartości z określonego zakresu (swego rodzaju slider)

Ogólna sekcja i demo: https://getbootstrap.com/docs/5.2/forms/overview/

Spacing

Bootstrap przychodzi nam z pomocą jeśli chodzi o wszelkiego rodzaju odstępy czyli marginesy i paddingi. Opracowali notację ułatwiającą szybkie dodawanie takich styli do naszych komponentów:

- Litera **m** lub **p** ustala czy to margines czy padding
- Litery **t s e b x y** ustalają kierunek
- Cyfry od **1 – 5** ustalają wielkość

Ogólna sekcja i demo: https://getbootstrap.com/docs/5.2/utilities/spacing/#margin-and-padding

Helpers & Utilities

Jest to zbiór klas służących do szybszego dodawania podstawowych właściwości CSS

Przykładowo nie musimy tworzyć własnej klasy oraz dodawać do niej konkretnych właściwości. Wystarczy że użyjemy predefiniowanych klas:

Skrócony zapis za pomocą Heleprs & Utilities

```
<div class="d-flex bg-dark text-white">Flex dark div</div>
```

Tradycyjne stylowanie

```
<div class="div-styles">Flex dark div</div>
```

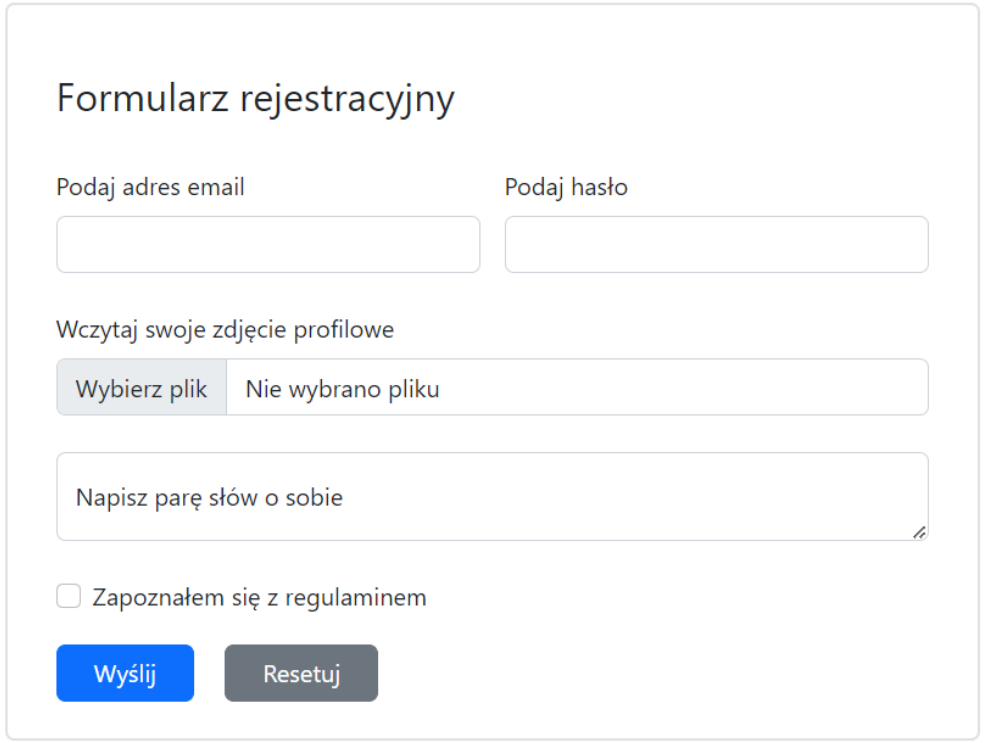
```
.div-styles {
    display: flex;
    color: white;
    background-color: black;
}
```

Przykładowy helper: https://getbootstrap.com/docs/5.2/helpers/color-background/

Zadanie formularz z gotowych komponentów

W repozytorium znajdziesz plik zadanie7.html z prawidłowo zaimportowanym Bootstrapem oraz zdjęcie formularza do odwzorowania:

1. Postaraj się odwzorować formularz ze zdjęcia
2. Nie dodawaj żadnych własnych klas ani stylowań. Korzystaj jedynie z tego co oferuje Bootstrap
3. Pola email oraz hasło muszą być wymagane
4. Dla ekranów poniżej 768px pola email oraz hasło powinny być jedno pod drugim



Czas: 15min

infoShare ACADEMY

Modale

- Są komponentami będącymi dialogami pływającymi na szczycie dokumentu HTML
- Posiadają tło za komponentem modelu, będące swego rodzaju overlayem (czyli takim przykrywającym, półprzezroczystym tłem)
- Stworzone są z użyciem position: fixed.

Oficjalne demo: https://getbootstrap.com/docs/5.2/components/modal/

Tabele

Służą do przedstawiania danych tabelarycznych i dają wiele wariantów konfiguracyjnych
Tabelki są w pełni responsywne (bardzo ciężko to osiągnąć samemu)

Oficjalne demo: https://getbootstrap.com/docs/5.2/content/tables/

Carousel

- Slider dla treści przełączającej się w sposób cykliczny
- Można w nich umieszczać obrazki, tekst lub inne elementy.
- Pozwala dodać przełączniki, podpisy itd.

Wizualne demo: https://getbootstrap.com/docs/5.2/components/carousel/#with-controls

Nawigacje

- Ostylowany pasek nawigacyjny
- Przystosowany do rozmieszczania za pomocą flexboxa
- Spełnia semantyczne znaczenie (używa odpowiednich znaczników w odpowiednim momencie)

Oficjalne demo: https://getbootstrap.com/docs/5.2/components/navbar/

Dropdowns

- Rozwijalne kontenery, pozwalające na wyświetlenie listy elementów.
- Otwierają się na kliknięcie.
- Działają w oparciu o bibliotekę Popper.
- Stworzone z myślą o dostępności dla osób niepełnosprawnych (accesibility)

Oficjalne demo: https://getbootstrap.com/docs/5.2/components/dropdowns/

Obrazki

- W pełni responsywne
- Mają szereg predefiniowanych klas
- Są przystosowane do użycia jako miniaturki

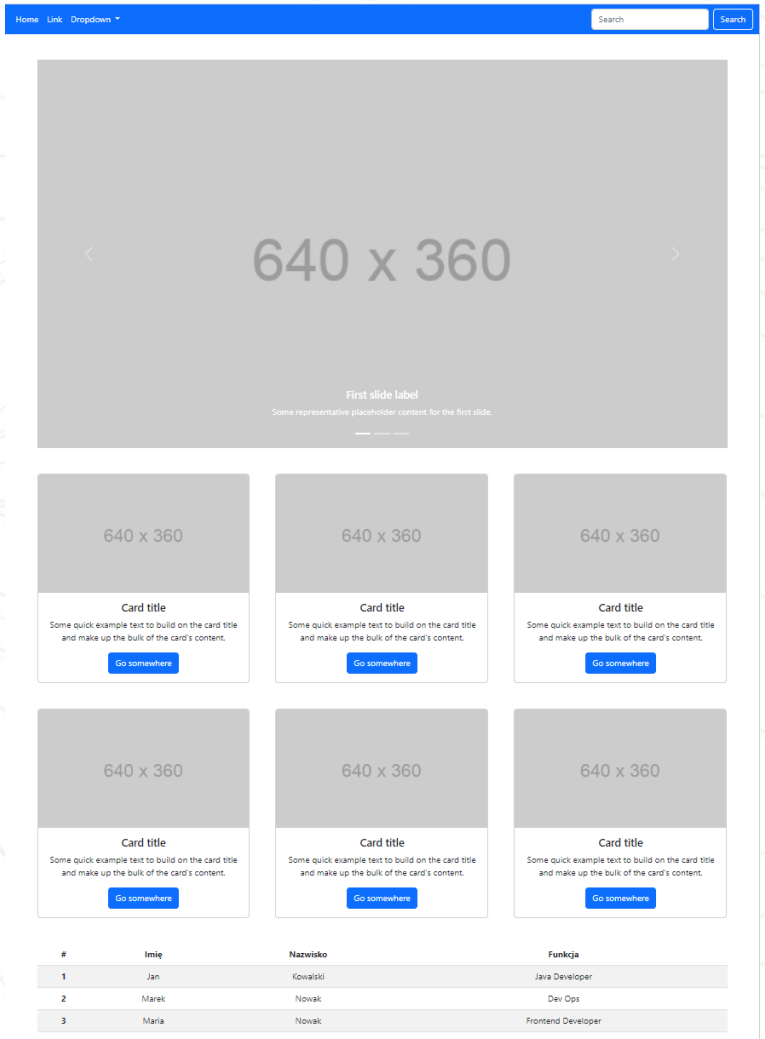
Skąd można brać testowe obrazki: https://loremipsum.io/21-of-the-best-placeholder-image-generators/

Oficjalna dokumentacja: https://getbootstrap.com/docs/5.2/content/images/#responsive-images

Zadanie szablon strony z gotowych komponentów

W repozytorium znajdziesz plik zadanie8.html z prawidłowo zaimportowanym Bootstrapem oraz zdjęcie formularza do odwzorowania:

1. Postaraj się odwzorować stronę ze zdjęcia (tekst nie musi być taki sam, grunt żeby pozycja elementów się zgadzała)
2. Nie dodawaj żadnych własnych klas ani stylowań. Korzystaj jedynie z tego co oferuje Bootstrap
3. Na samej górze trzeci element nawigacji to lista rozwijalna
4. Karuzela niech na każdym slidzie ma inny obrazek i tekst
5. Karty powinny być ustawione w rzędach po 3, jednakże jeśli szerokość ekranu jest mniejsza niż 576px niech każdy rząd ma tylko 2 karty
6. Możecie wykorzystać te obrazki które wrzuciłem do repozytorium
7. Niech pasek nawigacyjny będzie przyklejony do góry podczas scrollowania



Czas: 20min

infoShare ACADEMY

03. Podstawy JavaScript

Wprowadzenie do JS'a

infoShareAcademy.com

Kawałek historii

JavaScript został wynaleziony w roku 1995 przez amerykańskiego programistę i hakera Brendana Eicha pracującego dla Netscape Communications. Pierwotnym celem nowego języka było tworzenie interaktywnych stron WWW.

Parę faktów:

- Jest to język typowany słabo i dynamicznie
- Dzięki JS'owi jesteśmy w stanie uzyskać dostęp do elementów HTML
- Obecnie w JS jesteśmy w stanie stworzyć aplikację webową od A do Z (Frontend + Backend)

Przydatne metody

Przydatne metod, których możemy używać bez dodawania żadnych bibliotek:

console.log(„mój komunikat w konsoli”) – wypisze nam wynik w konsoli

alert(„mój alert”) – wyskoczy nam komunikat w przeglądarce

prompt(„moje polecenie”) – wyświetli nam pole w którym użytkownik może udzielić odpowiedzi na zadane pytanie

confirm(„moje pytanie”) – otwiera okno z pytanie/informacją, na które użytkownik może odpowiedzieć twierdząco lub przecząco

Wersje Java Script

Do 2015 roku nie było bardzo dużych zmian w JS'ie. W 2015 roku wyszła bardzo duża aktualizacja zwana ECMAScript 2015 lub ES6. Od tego momentu co roku wychodzi nowa aktualizacja dodająca nowe funkcje. Każda kolejna wersja jest nazywana

Zbiór propozycji dodania nowych funkcji do JS'a https://www.proposals.es/

Dokładna lista wersji: https://www.w3schools.com/js/js_versions.asp

Dodawanie JS'a do strony

Mamy dwa zasadnicze sposoby aby dodać skrypt do naszej strony:

- Inline pomiędzy tagami <script></script>
- Jako zewnętrzny plik

Zewnętrzny plik

```
<script defer src="myExternalFile.js"></script>
```

Osadzenie skryptu inline

```
<body>
  <script>
    console.log("inline script")
  </script>
</body>
```

Atrybuty **async** oraz **defer**:

Ich zadaniem jest odroczenie wykonania skryptu. Pobierają go w tle, a następnie wykonują gdy cały HTML już się załaduje dzięki czemu pobieranie skryptu nie blokuje parsowania HTML

defer dba o kolejność skryptów, a **async** działa na zasadzie kto pierwszy ten lepszy

Ciekawostka Bootstrap: https://github.com/twbs/bootstrap/discussions/36462

04. Typy danych i zmienne

Zmienne wartości i zakresy

Rodzaje typowań

Typowanie słabe oznacza system typów, w którym typ wyrażenia może być automatycznie zmieniony jeśli kontekst tego wymaga

Typowanie dynamiczne polega na przypisywaniu typów do wartości przechowywanych w zmiennych w trakcie działania programu

Typowanie słabe

```
const myVariable = 1

// słabe typowanie (weak typing)
if ("1" == myVariable) { // true
    // mimo że porównujemy dwa różne typy, słabe typowanie dopuszcza konwersję "w locie"
}
```

Typowanie dynamiczne

```
// dynamic typings

let dynamicVariable
dynamicVariable = 11
dynamicVariable = "some text"
```

Typy danych w JS

W JS mamy łącznie 9 typów, dzielą się one także na dwie grupy:

Typy prymitywne
- number (zarówno int jak i zmienno przecinkowe)
- string
- boolean (true/false)
- null
- undefined
- bigint (liczby większe niż 2^53 -1)
- symbol

```
// typy prymitywne
const myNumber = 16
const myString = "my text"
const myBoolean = true
let myUndefined
const myNull = null
const myBigInt = 12345678901234567890123456n
const mySymbol = Symbol("mySymbol")
```

Typy złożone:
- object
- function

```
// typy złożone
const myObject = {
    property1: 'myProperty',
    secondProperty: 125
}

3 usages
const myFunction = () => {
    console.log("myFunction")
}

1 usage
function classicFunction () {
    console.log("classicFunction")
}
```

infoShare ACADEMY

Określenie jakiego typu jest wartość

Do określenia jakiego typu jest wartość przechowywana w zmiennej służy nam operator **typeof**

```
console.log(typeof myNumber) // "number"
console.log(typeof myString) // "string"
console.log(typeof myBoolean) // "boolean"
console.log(typeof myUndefined) // "undefined"
console.log(typeof myNull) // "object" ⇒ historyczny błąd w języku JavaScript
console.log(typeof myBigInt) // "bigint"
console.log(typeof mySymbol) // "symbol"
console.log(typeof myObject) // "object"
console.log(typeof myFunction) // "function"
console.log(typeof classicFunction) // "function"
```

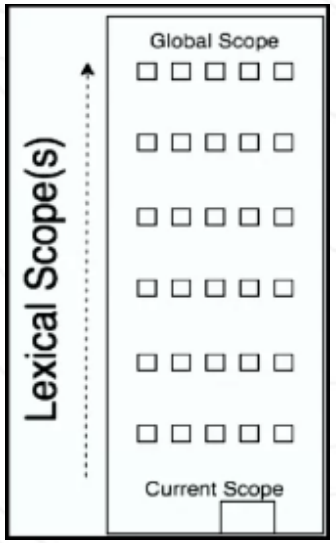
W JS to nie zmienna jest jakiegoś typu tylko wartość do niej przypisana!!!!

Zakresy

W prostych słowach zakres to przestrzeń w której JS szuka zmiennych oraz ma do nich dostęp

W języku JavaScript są 3 rodzaje zakresów:

- **Globalny** – jest utworzony domyślnie i wszystkie tworzone zakresy się w nim znajdują
- **Funkcyjny** – jest tworzony wraz z utworzeniem każdej funkcji
- **Blokowy** – jest tworzony wraz z użyciem {} np. w instrukcji if, pętli for



JS szuka referencji najpierw w najbliższym scope, a dopiero później idzie w górę, może to przypominać szukanie po piętrach bloku bez windy. Idziemy od parteru na samą górę

Zmienne

W wersjach JavaScript po ES6 mamy 3 opcje na deklarację zmiennej:

- **Var** – zmienna o zakresie globalnym oraz funkcyjnym
- **Let** – zmienna o zakresie globalnym funkcyjnym oraz blokowym
- **Const** – zmienna o takich samych zakresach jak let. Różnicą jest to, że do consta nie można PONOWNIE PRZYPISAĆ wartości.

Var to zmienna której identyfikator może być tworzony ponownie w tym samym zakresie bez błędu, w pozostałych przypadkach będzie błąd **Uncaught SyntaxError: Identifier <zmienna> has already been declared**

05. Pętle i funkcje

Jak nie duplikować naszego kodu

Pętle

Są to mechanizmy służące do kontrolowanego wykonywania kodu określoną ilość razy.
Podstawowe rodzaje pętli:

- For
- While
- Do while

```
const numbers = [1, 2, 3, 4, 5]

for (let i = 0; i < numbers.length; i++) {
    console.log(numbers[i])
}
```

```
let j = 0
while (j < numbers.length) {
    console.log(numbers[j])
    j++
}
```

```
let k = 0
do {
    console.log(numbers[k])
    k++
} while (k < numbers.length)
```

Funkcje

O funkcji można powiedzieć że jest to jakiś re-używalny blok kodu który możemy parametryzować oraz wywoływać w dowolnym miejscu dowolną ilość razy. Jest kilka sposobów na stworzenie funkcji:

- Function Declaration
- Arrow Function
- Function Expression

Function Declaration

```
function myFunction() {
  console.log("Function Declaration")
}
```

Arrow Function

```
const arrowFunction = () => {
  console.log("Arrow Function")
}
```

Function Expression

```
(function () {
  console.log("Immediately Invoked Function Expression")
})();
```

Zadanie Pętle i Funkcje

- Otwórz plik zadanie9.html
- Utwórz funkcję o nazwie printNumbers, która będzie przyjmowała jeden argument
- Zdefiniuj tablicę liczb od 11 – 15
- Przekaż ją jako argument do funkcji printNumbers
- Wewnątrz funkcji printNumbers za pomocą dowolnej pętli stwórz mechanizm wypisywania po kolei liczb przekazanych do funkcji

Czas: 15min

06.

Operatory i instrukcje warunkowe

Sterowanie naszym programem

Operatory

Operatory służą do wykonywania operacji na wartościach, możemy je podzielić na kilka grup:

- Arytmetyczne
- Przypisania
- Porównania
- Logiczne
- Jednowartościowe

Operatory arytmetyczne

Ich zadaniem jest operowanie na wartościach w sposób matematyczny. Możem dzięki nim wykonywać takie działania jak dodawanie, odejmowanie, mnożenie, dzielenie itp.

Pamiętaj o kolejności działań jak na matematyce!

Operatory przypisania

Są to operatory służące przypisaniu wartości do zmiennych, może to być zwykły znak = lub kombinacja znaku = i działania arytmetycznego np. *=, +=, -= itp.

Operatory porównania

Są to operatory takie jak: ==, ===, !==, !=, >=, <= itp.

W przypadku == i === bardzo ważną informacją jest fakt iż operator === porównuje wartości gdy są tych samych typów i nie dopuszcza do ich konwersji. Operator == jeśli widzi że wartości są różnych typów najpierw próbuje je przekonwertować na ten sam typ

Dobrą praktyką jest używanie wszędzie operatora === gdyż jest on bardziej przewidywalny

Operatory logiczne

Najczęściej wykorzystywane są w instrukcjach warunkowych:

- && (AND -> logiczne i) – oba składniki muszą być truthy
- || (OR -> logiczne lub) – przynajmniej jeden ze składników musi być truthy
- ! (NOT -> negacja) – odwraca wartość z true na false lub z false na true
- ?? (Nullish coalescing operator) – zwraca prawą wartość, gdy lewa jest null lub undefined

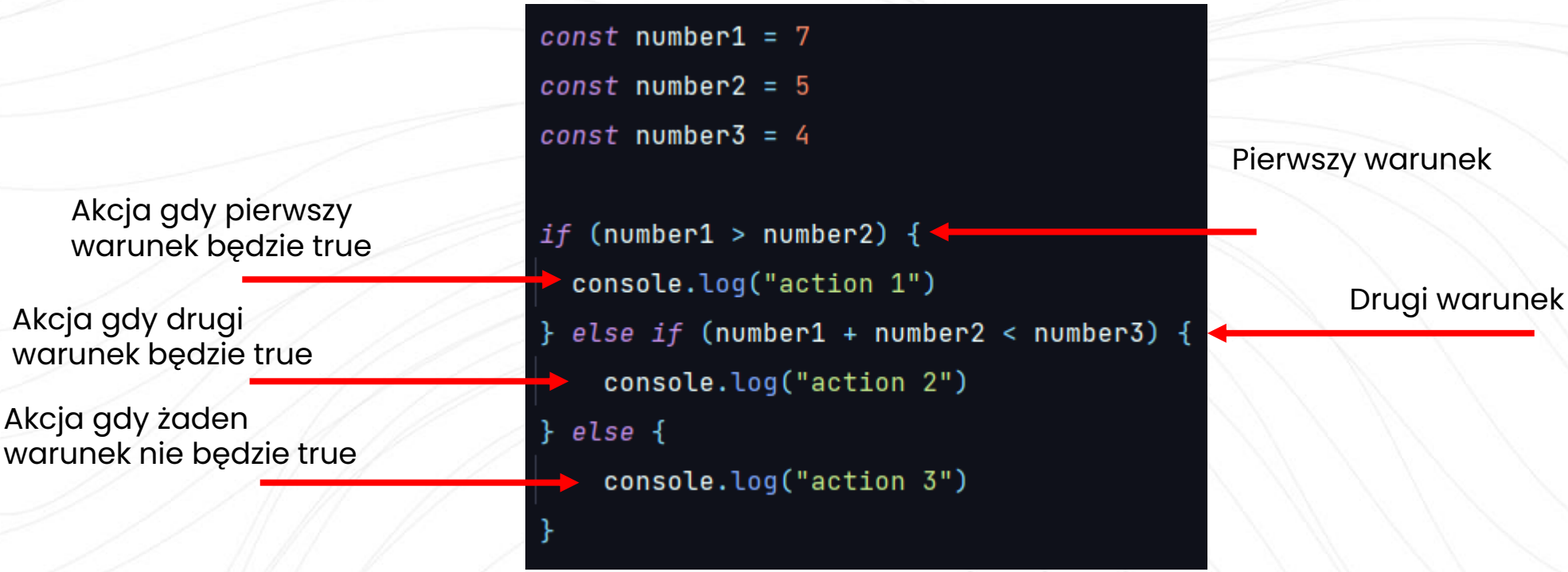
Operator ?? https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Nullish_coalescing

Operatory jednowartościowe

Przyjmują jedną wartość i wykonują na niej jakieś działanie, przykładem takiego operatora jest np. **delete** lub **typeof**

Instrukcje warunkowe

Jest to element programowania, który pozwala nam sterować przebiegiem działania programu. Warunek instrukcji if zawsze zostanie przekonwertowany przez Java Script do wartości logicznej boolean (true/false)



Ternary operator

Jedyny operator w JS przyjmujący trzy argumenty. Potrafi zastąpić klasyczną instrukcję warunkową i dzięki swojej prostej strukturze często jest stosowany jako prostsza alternatywa

Wzór

```
condition ? exprIfTrue : exprIfFalse
```

Przykład użycia

```
const age = 18;
const beverage = age ⩾ 18 ? "Beer" : "Juice";
```

Dokumentacja: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator

Falsy Values

Falsy values to wartości które w przypadku konwersji na wartość Boolean przyjmą wartość **false.** W JS są to:

- False
- Null
- Undefined
- NaN
- ””
- 0
- -0
- 0n
- document.all (tylko w ramach ciekawostki, bo jest to deprecated)

Sprawdzenie czy wartość jest falsy

```
const myVariable = 12
console.log(Boolean(myVariable))
```

Dokumentacja: https://developer.mozilla.org/en-US/docs/Glossary/Falsy

Zadanie sprawdzenie warunku

- Otwórz plik zadanie10.html
- Za pomocą **prompt** zadaj użytkownikowi pytanie: „Ile masz lat?”
- Wykorzystując instrukcje warunkowe i metodę console.log() spraw, że:
- Gdy odpowiedź użytkownika to 18 lub więcej w konsoli wypisz komunikat: „Jesteś pełnoletni/a”
- Gdy odpowiedź użytkownika to mniej niż 18 wypisz w konsoli „Nie jesteś pełnoletni/a”

Czas: 10min

infoShare ACADEMY

Invest in Pomerania ACADEMY

Do zobaczenia

Fundusze Europejskie Program Regionalny

Rzeczpospolita Polska

URZĄD MARSZAŁKOWSKI WOJEWÓDZTWA POMORSKIEGO

Unia Europejska Europejski Fundusz Rozwoju Regionalnego