



HTML, CSS, JavaScript

Autor: Jakub Buśkiewicz



Ramowy plan

1. Przydatne narzędzia, strony
2. HTML
3. CSS
4. JavaScript



Przydatne narzędzia, strony

1. Edytor kodu + pluginy (np. VScode, Atom, Sublime),
2. Devtools,
3. Google Fonts,
4. Optymalizacja stron/kodu:
 1. Google Lighthouse, PageSpeed,
 2. Uglify/minify
 3. Gtmetrix
 4. Google Search Console
5. WCAG - <https://www.w3.org/TR/WCAG21/>
6. <https://caniuse.com/> - tabele wsparcia przeglądarki
7. Git
8. Mockowanie danych – np. faker.js
9. Źródła wiedzy:
 1. Medium.com
 2. Youtube – overment, the net ninja
 3. Wyszukiwarka Google/StackOverflow
 4. Codewars, hackerrank



HTML



- HyperText Markup Language – język znaczników (nie jest językiem programowania), który mówi przeglądarkom jak tworzyć struktury wyświetlanych stron internetowych
- Pliki – rozszerzenie .htm/.html
- Początki datowane na rok 1991
- Rozwój od HTML2 do HTML4 na przestrzeni lat 1995 – 2000
- HTML5 – rok 2014 (5.2 – rok 2017)





HTML – anatomia znaczników i ich zagnieżdżanie

tekst

<p> - znacznik (tag) otwierający

</p> - znacznik zamykający (ten sam, co otwierający)

 - znacznik pogrubienia treści, który został zagnieżdżony w znaczniku paragrafu <p>



HTML – atrybuty znaczników

```
<p class="main-paragraph">Przykładowy tekst paragrafu</p>
```

Atrybuty zawierają dodatkowe informacje o elemencie, który nie pojawi się w treści.

W tym przykładzie atrybut **class** to nazwa identyfikująca używana do zlokalizowania elementu za pomocą informacji o stylu.

Cechy atrybutu:

1. Występuje po znaczniku (spacja),
2. Nazwa po której następuje znak równości,
3. Wartość występuje pomiędzy znakami cudzysłowu.



HTML – atrybuty znaczników - przykłady

```
<div id="main-element"></div>
```

```
<div class ="main-class"></div>
```

```
<div style ="font-size:20px"></div>
```

```
<link href="style.css" rel="stylesheet" type="text/css" />
```

```
<a href="#" title="link"></a>
```

```

```

```
<input role="search" />
```

```
<input data-visibility="hidden" />
```



HTML – struktura dokumentu

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>My test page</title>
6    </head>
7    <body>
8      <p>This is my page</p>
9    </body>
10   </html>
```



HTML – kluczowe znaczniki

`<!DOCTYPE html>` - informuje o typie dokumentu

`<html> </html>` - rozpoczyna dokument HTML

`<head> </head>` - zawiera informacje niewidoczne „gołym okiem” jak np. słowa kluczowe, tytuł strony, dane dotyczące kodowania,

`<body> </body>` - zawiera zawartość dokumentu HTML (część widoczna)



Ćwiczenie 1

Stwórz prosty szablon strony w HTML

1. Tytuł strony, opis meta
2. Kodowanie
3. Viewport – wsparcie dla urządzeń mobilnych



HTML – znaczniki cd.

<p>- paragraf treści np.

```
<body>
```

```
    <p>Cześć! Nazywam się Jakub, witaj na mojej stronie internetowej.</p>
```

```
</body>
```



HTML – znaczniki cd.

<a> - link, hiperłącze np.

```
<body>
```

```
    <a href="https://www.facebook.com/profile.php?id=1234">Link do mojego  
profilu na FB</a>
```

```
</body>
```



HTML – znaczniki cd.

<h1>, <h2>...<h6> - nagłówki strony internetowej np.

```
<body>
    <h1>Nowsy</h1>
        <h2>Wydarzenia z kraju</h2>
            <h3>News 1</h3>
        <h2>Wydarzenia ze świata</h2>
            <h3>News 2</h3>
</body>
```



HTML – znaczniki cd.

<div> - znacznik kontenera w którym umieszczamy inne znaczniki HTML np.

```
<body>
    <div>
        <p>To jest element widoczny z prawej strony</p>
        <a href="kontakt.html">Skontaktuj się ze mną</a>
    </div>
</body>
```



HTML – znaczniki cd.

- znacznik grafiki- zdjęcia, obrazka – jeden z wielu znaczników, który nie musi być "domknięty" tagiem zamykającym np.

```
<body>
    
</body>
```



HTML – znaczniki cd.

<script> - służy do wskazania pliku skryptu JavaScript (dzięki atrybutowi src) lub pozwala na wykonanie kodu znajdującego się bezpośrednio pomiędzy znacznikami np.

```
<body>
    <script src=".js/calc.js"></script>
    <script>
        console.log('Jesteś na stronie głównej');
    </script>
</body>
```



HTML – przykładowe znaczniki formatujące

**** - wzmacnianie treści; przykład semantycznego znacznika

**** - wyróżnienie treści

<sub> - indeks dolny

<sup> - indeks górny

<pre> - sformatowany tekst

**** - pogrubienie; znacznik niesemantyczny

<i> - kursywa

<s> - przekreślenie

<u> - podkreślenie



HTML – znaczniki blokowe i liniowe

Każdy element HTML ma domyślną wartość wyświetlania, w zależności od swojego typu. Wyróżniamy dwa typy: block oraz inline

```
<div>  
<p>Przykładowa sekcja zawierająca informacje na temat obecnie realizowanych <span>projektów</span></p>  
</div>
```

Elementy blokowe: div, p

Elementy liniowe: span

Przykładowa sekcja zawierająca informacje na temat obecnie realizowanychprojektów



HTML – znaczniki blokowe i liniowe cd.

Przykładowe znaczniki liniowe

```
<a>          <abbr>        <acronym>      <b>  
<code>        <dfn>         <em>           <i>  
<object>      <output>       <q>            <samp>  
<sub>         <sup>          <textarea>     <time>
```

Przykładowe znaczniki blokowe

```
<address>     <article>      <aside>        <blockquote> <canvas>  
<fieldset>    <figcaption> <figure>        <footer>     <form>  
<main>         <nav>          <noscript>     <ol>          <p>  
<ul>           <video>
```

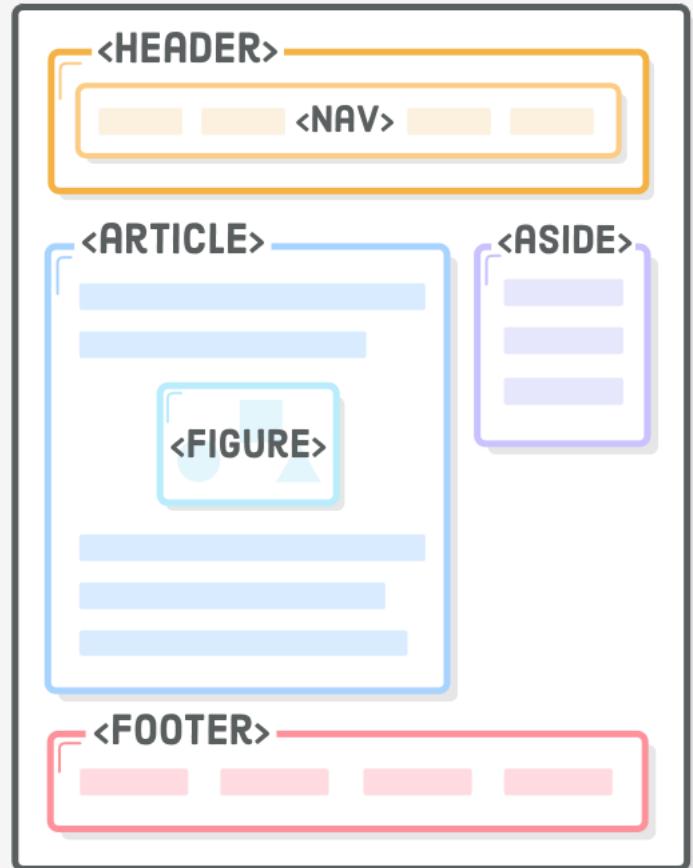


Semantyka HTML5

Nowa wersja języka HTML5 przyniosła wiele nowych znaczników, które m.in. względem na standardy dostępności są obecnie wykorzystywane w szerokim stopniu.

Semantyczny HTML oznacza użycie odpowiednich tagów zgodnie z ich przeznaczeniem.

Pozwala to m.in lepiej zrozumieć strukturę witryny robotom sieciowym oraz narzędziom ułatwiającym dostęp np. osobom niepełnosprawnym.





Semantyczny HTML5 – przykładowe tagi

<main>- kontener głównej zawartości dokumentu

<section>- wyznacza sekcję dokumentu, może zawierać nagłówek

<article>- kontener artykułu, używany np. do postów na forum, wpisów na blogu,

<header>- kontener zawierający informacje identyfikacyjne witryny (logo, nagłówek-tytuł) lub nawigację

<nav>- kontener nawigacji

<footer>- kontener stopki



Ćwiczenie 2

Rozwiń wcześniej stworzony dokument, dodaj do niego:

1. Znacznik body,
2. Kontener main wraz z odpowiednim nagłówkiem H1 oraz kilkoma zdaniami na Twój temat (możesz również wykorzystać kontenery sekcji oraz nagłówki H2).
Pamiętaj, że znaczniki paragrafów mogą posiadać wyróżnione informacje.
3. Sekcję footer.



Listy

W HTML wyróżniamy dwa typy list:

 - listy nienumerowane

 - listy numerowane

- element 1
- element 2
- element 3

1. element 1
2. element 2
3. element 3

W obu przypadkach za element listy odpowiada znacznik

```
<ul>
    <li>element 1</li>
    <li>element 2</li>
    <li>element 3</li>
</ul>
```



Ćwiczenie 3

Rozbuduj swoją stronę o kolejne elementy:

1. Dodaj znacznik header oraz nav,
2. W znaczniku nav utwórz listę nienumerowaną oraz dodaj do niej 3 elementy,
3. Do każdego elementu dodaj znacznik hiperłącza, a w nich: Home, O mnie, Kontakt,
4. Utwórz dokument style.css w folderze css oraz dodaj ten dokument w sekcji head swojej strony.



Formularze

Znacznik `<form>` osadza w dokumencie formularz. Służy do grupowania pól formularzy i przycisków np..

```
<body>  
    <form>  
        ...  
    </form>  
</body>
```



Formularze – cd.

Pola formularza (znacznik `<input />`) pozwalają użytkownikowi komunikować się z naszą aplikacją dzięki wysyłaniu potrzebnych danych.

Atrybut `type` określa rodzaj danych wejściowych:

- `type="text"` to podstawowy typ pozwalający wysyłać tekst,
- `type="number"` przyjmuje dane w postaci liczb,
- `type="checkbox"` tworzy pola wyboru,
- `type="submit"` powoduje wysłanie danych z formularza.



Formularze – cd.

W formularzach możemy grupować dane wejściowe z np. pól wyboru dzięki atrybutowi name. Dodatkowo znacznik <label> pozwala grupować opisy do poszczególnych pól wejściowych dzięki atrybutom id oraz for np.

```
<form>
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label>
  <br />
  <input type="radio" id="female" name="gender" value="female">
  <label for="female">Female</label>
</form>
```



Formularze – cd.

Odpowiednia konfiguracja naszego formularza przekaże dane do wskazanego przez nas zasobu internetowego. Dane możemy przekazać na dwa sposoby:

- get,
- post.

Dzięki atrybutowi **action** możliwe wskazanie jest zasobu w Internecie np.

```
<form action="/action_page.php" method="get">  
<form action="/action_page.php" method="post">
```



Formularze – cd.

```
<form action="/my-handling-form-page" method="post">
<ul>
  <li>
    <label for="name">Name:</label>
    <input type="text" id="name" name="user_name" />
  </li>
  <li>
    <label for="mail">E-mail:</label>
    <input type="email" id="mail" name="user_email" />
  </li>
  <li>
    <label for="msg">Message:</label>
    <textarea id="msg" name="user_message"></textarea>
  </li>
  ...

```



Ćwiczenie 4

Rozbuduj dotychczas utworzoną strukturę strony:

1. Dodaj prosty formularz, który będzie pobierał od użytkownika imię, adres mailowy oraz wiadomość,
2. Pamiętaj, że możesz wykorzystać semantyczne znaczniki HTML,
3. Wykorzystaj elementy <label> do zgrupowania pól wejściowych oraz ich opisów.



Tabele

Dane tabelaryczne możemy przedstawić w HTML przy pomocy znacznika `<table>`.

Każdy wiersz tabeli jest zdefiniowany za pomocą znacznika `<tr>`.

Nagłówek tabeli jest definiowany za pomocą znacznika `<th>`.

Dane tabeli / komórkę definiuje się za pomocą znacznika `<td>`.

```
<table>
  <tr>
    <td>John</td>
    <td>Doe</td>
  </tr>
  <tr>
    <td>Jane</td>
    <td>Doe</td>
  </tr>
</table>
```



CSS



- Cascade Style Sheets – kaskadowe arkusze stylów, pozwalają na stylizację dokumentów tekstowych np. HTML,
- Pozwalają m.in. na zmianę wielkości elementu, jego tła, koloru, marginesów, dodanie animacji,
- CSS jest oparty na regułach, które zapisujemy w określony sposób zgodnie z zasadami,
- Im bardziej dokładna reguła – tym większą ma moc (kaskada znaczników),





CSS – składnia

Kod języka CSS składa się z:

- Selektora – identyfikującego np. element HTML,
- Nawiasów klamrowych "spinających" zasadę,
- Własności (np. width),
- Wartości własności (np. 100px).

```
1 .button {  
2     background: linear-gradient(#eee, #ccc);  
3     border: 1px solid #999;  
4     color: #333;  
5     cursor: pointer;  
6     padding: 1em 1.5em;  
7  
8     &:hover {  
9         background: linear-gradient(#fff, #ddd);  
10        color: #111;  
11    }  
12 }
```



CSS – selektory

CSS zapewnia wiele możliwości identyfikowania elementów HTML np.:

- Poprzez nazwę elementu (np. div, p, nav, header, footer)
- Poprzez klasę elementu (np. .nav, .aside, .footer, .article)
- Poprzez id elementu (np. #navigation, #top, #bottom)
- Poprzez atrybuty elementu ([type=email])
- W odniesieniu do wszystkich elementów (*)

https://www.w3schools.com/cssref/css_selectors.asp



CSS – umieszczanie stylu na stronie

Trzy sposoby:

- w elemencie przy pomocy atrybutu style,
- za pomocą znacznika <style> w sekcji <head>,
- jako zewnętrzny plik – linkujemy plik w sekcji <head>.

```
<head>
    <style>
        div {
            font-size: 15px;
        }
    </style>

    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <div style="color: black"></div>
</body>
```

CSS - dziedziczenie



Niektóre właściwości CSS mogą być dziedziczone, np. ustawienie właściwości color: red dla rodzica, ustawi tą właściwość dla jego dzieci (dopóki nie nadamy dziecku jego własnego stylu).

```
<div style="color: red;">  
  <p>Przykładowy tekst</p>  
<div>
```

```
<div style="color: red;">  
  <p style="color: green">Przykładowy tekst</p>  
<div>
```



CSS – kaskada znaczników

Id > Atrybut, Klasa > Nazwa elementu

Style o wyższym priorytecie (lepszym dopasowaniu) mają pierwszeństwo w ustalaniu stylu elementu w dokumencie.

Można złamać kaskadowość poprzez użycie znacznika **!important** w deklaracji stylu dla konkretnej właściwości.



CSS – podstawowe właściwości

- margin
- font-size
- padding
- color
- background-color
- display
- border
- text-align
- font
- text-decoration
- width
- height
- font-weight
- list-style
- line-height
- letter-spacing
- box-shadow
- cursor
- position
- transform
- transition
- word-wrap
- visibility

Autor: Jakub Buśkiewicz

Prawa do korzystania z materiałów posiada Software Development Academy



Ćwiczenie 5

1. W utworzonym pliku style.css dodaj tło całej strony - **#0C2840**,
2. Zmień font-size wszystkich paragrafów na 14px, zmień kolor tekstu na biały,
3. Do sekcji head dokumentu dodaj font 'Roboto' który znajdziesz na Google Fonts,
4. Ustaw wskazany font jako domyślny dla całego dokumentu,
5. Nadaj nagłówkom H1 kolor **#A3BFBF** oraz font-weight 600,
6. Sekcji footer dodaj klase .bottom i dodaj jej ramkę w kolorze **#F2E6CE**, ustaw jej padding na 10px z gory i dołu oraz 5px z prawej i lewej strony,
7. Do sekcji header dodaj klasę .head oraz nadaj jej padding o wartości 20px.

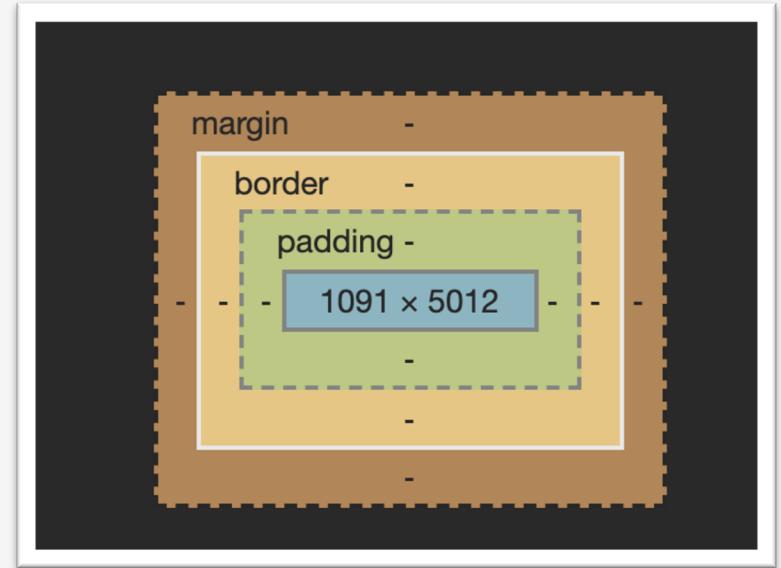


CSS – box model

Każdy element strony wyświetlany jest przez przeglądarkę w formie prostokąta, który posiada pewne właściwości tzn: wysokość, szerokość, ramkę, marginesy. Box model określa relację między zawartością elementu, a jego właściwościami.

Domyślnie właściwość "box-sizing" jest ustawiona na "content-box" co oznacza, że do wielkości elementu wlicza się tylko jego zawartość (należy doliczyć np. padding i grubość ramki).

Zmiana właściwości box-sizing na "border-box" wlicza do wielkości elementu także jego padding, margin i border (co pozwala lepiej kontrolować wielkość tego elementu).





CSS – reset

Każda przeglądarka ma wbudowany zestaw stylów, które definiują wartości właściwości np. marginesów. Ujednolicony wygląd naszej strony internetowej możemy uzyskać po "zresetowaniu" tych wartości tzn. ustawieniu wartości = 0 np.

```
* {  
    margin: 0px;  
    padding: 0px;  
}
```

Należy pamiętać, że domyślnych właściwości jest znacznie więcej (style list, tabel, nagłówków, przycisków), a do globalnego resetowania wartości można wykorzystać gotowe skrypty np.

<https://gist.github.com/DavidWells/18e73022e723037a50d6>



CSS – pseudoklasy, pseudoelementy

pseudo-classes:

```
a:link{color: #ccc}
```

```
a:hover{color: #efc}
```

```
div:first-child{border: 1px solid red}
```

pseudo-elements:

```
a::after{content:"hello"}
```

```
div::before{}
```

```
p::first-letter{color: red}
```



CSS – jednostki

px – wartość w pikselach, wartość bezwzględna np. 14px,
% - wartość procentowa względem bezpośredniego rodzica np. 50%,
em – wartość względem bezpośredniego rodzica np. 1.5em,
rem (root em) - względem korzenia tj. elementu <html>,
vw i vh – względem viewport



CSS – display

Własność display określa w jaki sposób wyświetlamy dany element HTML np.

```
div {  
  display: block | flex | none | inline-block;  
}
```

Bardzo popularną wartością dla wyświetlanych elementów blokowych jest "flex", który pozwala na ustawianie kierunku osi dla dzieci danego elementu i daje nam możliwość elastycznego zarządzania zawartością dokumentu.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>



Ćwiczenie 6

1. Stwórz pusty dokument HTML z podstawową strukturą kodu,
2. Dodaj kilka elementów div bezpośrednio w sekcji body,
3. Dodaj do dokumentu plik CSS i nadal każdemu elementowi wysokość i szerokość 200px,
4. Każdemu elementowi div nadal inny kolor tła,
5. Zmień właściwość display sekcji body na "flex",
6. Sprawdź jak zachowują się kolorowe kwadraty przy zmianach wartości właściwości flex-direction na column, row-reverse, column-reverse
7. Nadaj sekcji body wysokość 800px, a wartość align-items zmień na flex-end, flex-start, center,
8. Podobne wartości ustaw dla właściwości justify-content. Docelowo ustaw kwadraty pionowo, na środku, w odwróconej kolejności.



CSS - position

CSS pozwala użyć właściwości position aby określić pozycję elementu, który może przyjąć następujące wartości:

- static (domyślna wartość),
- relative – element staje się punktem odniesienia dla dzieci,
- absolute – element jest "wyrywany" z flow dokumentu przez co może nachodzić na inne elementy.

Pozycja jest określona względem najbliższego pozycjonowanego rodzica lub jeśli żaden nie jest pozycjonowany to jest określona względem początkowego położenia,

- fixed – element "przykleja się" do ekranu.



- Responsive web design
- Media queries
- Gracefully degradation vs progressive enhancement
- Desktop first / mobile first
- src set
- viewport



JavaScript



JavaScript – zarys

- VanillaJS, ES5, ES6...ES10 (ECMAScript 2019)
- Skryptowy (interpretowany lub komplikowany metodą JIT) język programowania,
- Język programowania wysokiego poziomu,
- Wieloplatformowy,
- Dynamiczne typowanie,
- Oparty na prototypach, wspiera programowanie funkcyjne jak i zorientowane obiektowo (OOP),
- Jeszcze kilka lat temu wykorzystywany jedynie po stronie klienta, obecnie również po stronie serwera,
- Silniki: V8, SpiderMonkey, Chakra, JavaScriptCore



Ćwiczenie

1. Utwórz nowy dokument HTML z prostym szablonem kodu,
2. Stwórz folder scripts, umieść w nim pusty plik alert.js
3. W pliku alert.js wywołaj funkcję alert() z komunikatem „To jest mój pierwszy skrypt”,
4. Dodaj skrypt do nowo utworzonej strony.



JS – typy zmiennych

- typy prymitywne – wartość prymitywna jest "niemutowalna", nie możemy zmienić wartości po jej zapisaniu w pamięci. Zmieniona jednak może być wartość przypisana do zmiennej.
- typy referencyjne – są referencją do wartości zapisanej w pamięci. Referencje prowadzą do obiektu zapisanego w pamięci. Pozwalają na mutowanie wartości.



JavaScript – typy prymitywne

- string – ciąg znaków,
- number – wartość numeryczna, w JS zarówno liczby całkowite jak i zmiennoprzecinkowe mają typ numer,
- boolean – wartość logiczna – prawda albo fałsz,
- undefined – wartość niezadeklarowana, nieistniejąca w pamięci,
- null – reprezentuje zamierzony brak wartości,
- symbol – identyfikuje anonimowe właściwości obiektu, używany jako unikalny klucz.



JavaScript – typy referencyjne

- Obiekt – jest kolekcją właściwości. Właściwości składają się z klucza i przypisanej do niego wartości. Obiekty mogą przechowywać typy prymitywne, tablice, funkcje (metody) a także inne obiekty.
- Tablice – tablice również mogą przechowywać różne typy danych, różnica między tablicą a obiektem leży w dostępie do wartości. Tablica jest iterowalna – można na niej użyć pętli for...of.
- Funkcje – blok kodu stworzony do wykonania zadania. Funkcje są wykonywane gdy coś je wywoła.



JavaScript – console.log

Konsola pozwala na wykonywanie funkcji, tworzenie zmiennych, operacje arytmetyczne, ma dostęp do obiektu Window i Document

Interpreter JS daje dostęp do metody console.log, która pozwala na wyświetlenie informacji przekazanej jako argument.

```
> console.log(2 + 1);
3
< undefined
```



JavaScript – typeof – sprawdzanie typu

Operator typeof pozwala na sprawdzenie typu danych.

Warto zwrócić uwagę, że typ dla tablicy oraz obiektu to „object” przez co nie jest jasne z jakim typem danych pracujemy.

Do sprawdzenia typu tablicy warto użyć metody statycznej isArray klasy Array:

```
Array.isArray([]) //true
```

```
> typeof 50
< "number"
> typeof 'abc'
< "string"
> typeof true
< "boolean"
> typeof []
< "object"
> typeof function() {}
< "function"
> typeof {}
< "object"
```



JavaScript – instanceof – sprawdzanie instancji

Operator instanceof sprawdza czy dany obiekt jest instancją innego obiektu (sprawdza czy constructor danego obiektu jest w łańcuchu prototypów).

```
{} instanceof Array  
true  
[] instanceof Object  
true  
Function instanceof Object  
true
```



JavaScript – index

W JavaScript możemy odwoływać się do określonych wartości za pomocą indexu w tablicach jak również w ciągach łańcucha znaków. Index zaczyna się od 0.

Istnieją metody, które sprawdzają, na którym indexie jest określony element. W przypadku gdy zwróci -1 oznacza, że zbiór nie zawiera szukanej wartości.

Do indexu odwołujemy się za pomocą kwadratowych nawiasów i podania wartości liczbowej np. tablica[2]



JavaScript – operacje logiczne

JavaScript: Logical Operators and Boolean Values

```
// Logical AND operator          // Logical OR operator
true && true; // true           true || true; // true
true && false; // false        true || false; // true
false && true; // false        false || true; // true
false && false; // false       false || false; // false
```

```
!!0
false
!!null
false
!!undefined
false
!!'false'
true
!!1
true
.
```

```
> Boolean('Hello')
< true
> Boolean('')
< false
> Boolean([])
< true
> Boolean({})
< true
> Boolean(undefined)
< false
> Boolean(null)
< false
> Boolean(42)
< true
> Boolean(0)
< false
>
```



JavaScript – String

String jest globalnym obiektem, konstruktorem dla stringów. W momencie, w którym używamy danej metody na łańcuchu znaków (np. concat), typ prymitywny jest opakowywany w obiekt String, który daje dostęp do metod przeznaczonych do pracy nad stringami.

Po wykonaniu operacji string (jako typ prymitywny) jest wypakowywany i znów jest typem prymitywnym.



JavaScript – String – popularne metody

- concat – łączy dwa lub więcejłańcuchów znaków i zwraca nowy łańcuch

```
console.log('abc'.concat('def', 'ghi'));  
abcdefghi
```

- includes – sprawdza czy jeden string zawiera drugi, zwraca boolean

```
'Wikipedia is a free online encyclopedia'.includes('free')  
true
```

- indexOf – zwraca numer indexu pierwszego wystąpienia szukanegołańcucha. Możemy także podać od którego indexu ma zacząć szukać.

```
'Wikipedia is a free online encyclopedia'.indexOf('free');  
15
```



JavaScript – String – popularne metody cd.

- match – odnajduje trafienia w stringu w stosunku do podanego regexa

```
'Wikipedia is a free online encyclopedia'.match(/ [A-Z]/g)  
▶ ["W"]
```

- repeat – powtarza i łączy string o podaną ilość razy

```
'Wikipedia is a free online encyclopedia'.repeat(2)  
"Wikipedia is a free online encyclopediaWikipedia is a free online encyclopedia"
```

- replace – zamienia część stringu pasującą do regexa lub stringu na

podaną wartość

```
'Wikipedia is a free online encyclopedia'.replace(/wikipedia/gi, 'PWN')  
"PWN is a free online encyclopedia"
```

```
'Wikipedia is a free online encyclopedia'.replace('Wikipedia', 'PWN')  
"PWN is a free online encyclopedia"
```



JavaScript – String – popularne metody cd.

- slice – wycina określony kawałek stringa i zwraca nowy łańcuch. Wymaga podania indexu od którego zaczynamy. Można także podać index w którym chcemy skończyć, w przeciwnym razie wytnie string do końca.

```
'Wikipedia is a free online encyclopedia'.slice(5, 10)  
"edia "
```

- split – dzieli string na tablicę stringów, podzielonych za pomocą podanego separatora

```
'114-44'.split('-')  
▶ (2) ["114", "44"]
```

- trim – usuwa białe znaki z początku i końca stringu

```
' test '.trim()  
"test"
```



JavaScript – Number – popularne metody

- parseInt – tworzy liczbę całkowitą, przyjmuje string i bazę (system).
- parseFloat – tworzy liczbę zmiennoprzecinkową.

Przyjmuje tylko jeden argument – string.

```
parseFloat('2.34')  
2.34  
parseFloat('2.34abc')  
2.34  
parseFloat('2abc')  
2
```

```
parseInt('0110', 2)  
6  
parseInt('15.5', 10)  
15  
parseInt('0xF', 16)  
15  
parseInt('15ac', 10)  
15
```



JavaScript – Number – popularne metody cd.

- `toFixed` – pozwala sformatować numer do określonego miejsca po przecinku podanego jako argument. Zaokrąglą w górę w razie potrzeby.
- `toString` – zamienia numer na string. Możemy podać bazę (system) w jakim ma zostać zapisany numer

```
2.34556.toFixed(3)  
"2.346"
```

```
(100).toString(2)  
"1100100"  
num.toString(8)  
"144"
```



JavaScript – Array – popularne metody

- shift – usuwa pierwszy element z tablicy i go zwraca
- unshift – dodaje element na początek tablicy
- pop – usuwa ostatni element tablicy i go zwraca
- push – dodaje ostatni element tablicy

```
const names = ['Adam', 'Robert', 'Piotr'];
names.shift();
console.log(names);
```

```
▶ (2) ["Robert", "Piotr"]
```

```
const names = ['Adam', 'Robert', 'Piotr'];
names.unshift('Roman');
console.log(names);
```

```
▶ (4) ["Roman", "Adam", "Robert", "Piotr"]
```

```
const names = ['Adam', 'Robert', 'Piotr'];
names.pop();
console.log(names);
```

```
▶ (2) ["Adam", "Robert"]
```

```
const names = ['Adam', 'Robert', 'Piotr'];
names.push('Roman');
console.log(names);
```

```
▶ (4) ["Adam", "Robert", "Piotr", "Roman"]
```



JavaScript – Array – popularne metody cd.

- concat – łączy tablice i zwraca nową, połączoną tablicę
- forEach – pozwala wykonać określoną funkcję na każdym elemencie tablicy.
- map – tworzy nową tablicę zawierającą wyniki wywoływania podanej funkcji dla każdego elementu wywołującej tablicy.
- includes – sprawdza czy tablica zawiera dany element

```
const names = ['Adam', 'Robert', 'Piotr'];
const age = [23, 40, 25];
const mergedArray = names.concat(age);
console.log(mergedArray);
```

▶ (6) ["Adam", "Robert", "Piotr", 23, 40, 25]

```
let amount = 0;
const numbers = [2, 4, 5];
numbers.forEach(number => amount += number);
console.log(amount);
```

11

```
const numbers = [2, 4, 5];
const multiplayedNumbers = numbers.map(number => number * 2);
console.log(numbers);
console.log(multiplayedNumbers);
```

▶ (3) [2, 4, 5]
▶ (3) [4, 8, 10]

```
const names = ['Adam', 'Robert', 'Piotr'];
console.log(names.includes('Piotr'));
console.log(names.includes('Stefan'));
```

true
false



JavaScript – Array – popularne metody cd.

- filter – zwraca nową tablicę z elementami, które spełniają test określony w funkcji.

```
const numbers = [2, 400, 5, 240];
const biggerThan100 = numbers.filter(number => number > 100);
console.log(biggerThan100);
▶ (2) [400, 240]
```

- indexOf – zwraca pierwszy (najmniejszy) index, w którym znajduje się podana wartość. Zwraca -1 jeśli element nie znajduje się w tablicy.

```
const numbers = [2, 400, 5, 240];
console.log(numbers.indexOf(5));
console.log(numbers.indexOf(1000));
2
-1
```

- join – łączy wszystkie elementy tablicy w string za pomocą separatora

```
const names = ['Adam', 'Robert', 'Piotr'];
console.log(names.join(', '));
Adam, Robert, Piotr
```



Ćwiczenie

1. Utwórz tablicę z nazwami miesięcy i złącz ją w jednego stringa
2. Utwórz tablicę z 5 liczbami, następnie na jej podstawie zwróć nową tablicę z liczbami mniejszymi niż 10
3. Korzystając z wcześniejszej tablicy z 5 liczbami, zwróć nową tablicę, z elementami pomnożonymi przez 3.



JavaScript – Object – tworzenie obiektów

Obiekty można tworzyć za pomocą:

- literału – wstawiając nawiasy klamrowe
- konstruktora Object
- funkcji – konstruktora - dziedziczenie
- słowa kluczowego class - dziedziczenie

```
const person = new Object();
person.name = 'Adam';
console.log(person);
▶ {name: "Adam"}
```

```
const person = {
  name: 'Adam',
  surname: 'Małysz'
};
console.log(person);
▶ {name: "Adam", surname: "Małysz"}
```

```
function Person(name){
  this.name = name
}

const adam = new Person('Adam');
console.log(adam);
▶ Person {name: "Adam"}
```

```
class Person {
  constructor(name, surname) {
    this.name = name;
    this.surname = surname;
  }
}

const am = new Person('Adam', 'Małysz');

console.log(am);
▶ Person {name: "Adam", surname: "Małysz"}
```



JavaScript – Object – popularne metody

- `entries` – tworzy tablicę tablic z kluczami i wartościami obiektu.
- `keys` – tworzy tablicę nazw właściwości (kluczy) obiektu.

Mając array kluczy możemy iterować po obiekcie.

```
const person = {  
    name: 'Adam',  
    surname: 'Małysz'  
};  
console.log(Object.entries(person));  
▼ (2) [Array(2), Array(2)] ⓘ  
  ► 0: (2) ["name", "Adam"]  
  ► 1: (2) ["surname", "Małysz"]  
    length: 2
```

```
const person = {  
    name: 'Adam',  
    surname: 'Małysz'  
};  
console.log(Object.keys(person));  
► (2) ["name", "surname"]
```

```
for (let key of keys) {  
    console.log(person[key]);  
}  
Adam  
Małysz
```



JavaScript – Object – popularne metody cd.

- **create** – tworzy nowy obiekt używając starego obiektu jako prototypu. Obiekty referują do innego miejsca w pamięci!
- **assign** – tworzy kopie wartości obiektu do nowego obiektu. Obiekty referują do innego miejsca w pamięci!

```
const person = {  
    name: 'Adam',  
    surname: 'Małysz'  
};  
  
const copiedObject = Object.create(person);  
copiedObject.name = 'Stefan';  
console.log(person.name);  
console.log(copiedObject.name);  
  
Adam  
Stefan
```

```
const person = {  
    name: 'Adam',  
    surname: 'Małysz'  
};  
  
const copiedObject = Object.assign({}, person);  
copiedObject.name = 'Stefan';  
console.log(person.name);  
console.log(copiedObject.name);  
  
Adam  
Stefan
```



JavaScript – Object – dodawanie metody

Mozemy dodać metody do obiektu w następujący sposób:

- deklarując funkcję wewnątrz obiektu lub dodając odpowiednią właściwość po stworzeniu obiektu
- używając definicji funkcji (function definition) - nowego sposobu dodawania metod, wprowadzonego w ES6

```
const dog = {  
    name: 'Burek',  
    hau: function() { console.log ('Hau! Hau!')}  
};  
  
dog.hau();  
Hau! Hau!
```

```
const dog = {  
    name: 'Burek',  
};  
dog.hau = function() { console.log ('Hau! Hau!')};  
dog.hau();  
Hau! Hau!
```

```
const dog = {  
    name: 'Burek',  
    hau() {console.log('Hau! Hau!')}  
};  
  
dog.hau();  
Hau! Hau!
```



Ćwiczenie

1. Utwórz obiekt o nazwie car, który będzie zawierał informacje o marce, modelu i roku produkcji samochodu,
2. Dodaj i wywołaj metodę "start", która wyświetli komunikat "Starting...",
3. Dodaj i wywołaj metodę "details", która wyświetli w konsoli informacje na temat marki, modelu i roku produkcji.

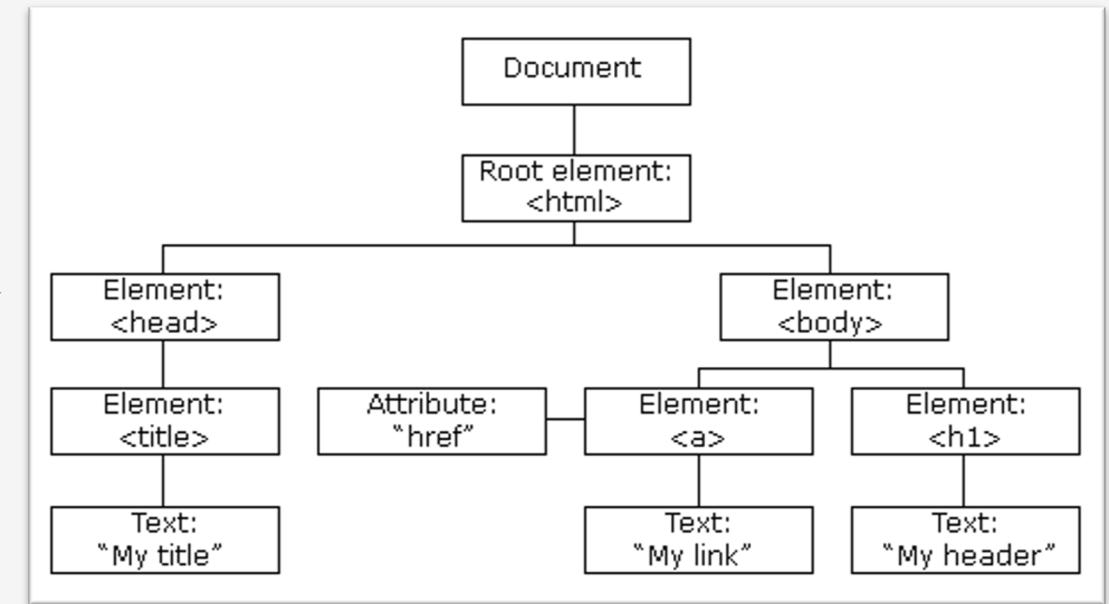


JavaScript – Operacje na DOM

DOM – Document Object Model

Kiedy HTML jest ładowany przeglądarka tworzy DOM strony

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href='http://google.pl'>
      My link
    </a>
    <h1>
      My header
    </h1>
  </body>
</html>
```





JavaScript – Cechy szczególne

- Bazowanie na prototypach,
- Hoisting – zmienna może być użyta przed jej deklaracją,
- Zasięg zmiennych,
- Globalny kontekst wywołania,
- The loop,
- Funkcja natychmiastowa – IIFE(ang. Immediately-Invoked Function Expression),
- Callback – czyli przekazywanie funkcji jako parametr do innej funkcji i użycie podczas zwracania rezultatu,
- Closures (domknięcia),



jQuery

- Biblioteka JavaScript,
- Stworzona do prostszego zarządzania elementami DOM i przeprowadzania na nich operacji,
- Bardzo duże wsparcie pluginów np. lightbox, carousel,
- Jednolitość kodu pomimo upływu czasu,
- <https://jquery.com/>
- <https://api.jquery.com/>



jQuery – funkcja ready

```
$(document).ready(function() {  
    ...  
});  
  
//lub to samo w skróconej - zalecanej wersji  
  
$(function() {  
    ...  
});
```



jQuery – przykładowe użycie i składnia

```
$("#btn").css({background : "blue"}).delay(2000).slideUp().delay(1000).fadeIn(1000);  
  
//wcale nie musisz wszystkiego pisać w jednej linii  
$("#btn")  
  .css({  
    background : "blue"  
  })  
  .delay(2000)  
  .slideUp()  
  .delay(1000)  
  .fadeIn(1000);
```



jQuery – popularne metody

```
$("div").find("span"); //znajdź spany w div  
  
$("button").eq(2); //2 element z kolekcji czyli 2 button  
  
$("span").first(); //pierwszy element w kolekcji  
  
$("div strong").last(); //ostatnie strong z tej kolekcji  
  
$(".module").not(".first"); //elementy .module który nie jest .first  
  
$("#main").is(".big"); //element #main który jest .big  
  
$("div").filter(":even"); //divy parzyste  
  
$("div").filter(function() {  
    return $(this).text() === "kot"  
}); //wszystkie divy które mają tekst "kot"  
  
const $div = $("div");  
$div.add("p"); //dodałem kolekcję p do div
```



jQuery – style i klasy

Aby zmienić style elementu skorzystamy z metody `css(właściwość, nowa_wartość*)`, którą można używać na 2 sposoby:

```
const $el = $("#test"); //wszystkie divy

$el.css("background-color", "red");
$el.css("color", "blue");
$el.css("padding", 10); //jeżeli nie podajemy jednostki, zostaną
automatycznie dodane px

//lub podając obiekt

$el.css({
    color: "red",
    "background-color": "blue",
    padding: 10 //px zostaną dodane automatycznie
});
```



jQuery – zdarzenia

Aby przypiąć nasłuch zdarzenia do pobranego lub utworzonego elementu korzystamy z metody będącej nazwą zdarzenia, lub skorzystać z metody **on()**.

```
$("button").click(function() {  
    console.log("test");  
}  
  
//lub  
  
$("button").on("click", function() {  
    console.log("test");  
})
```

Aby odwołać się do obiektu wywołującego, wskazujemy go identycznie jak w przypadku klasycznego Javascript czyli poprzez **this** lub **e.target**

```
$("button").on("click", function() {  
    this.text("Kliknięto!");  
});  
  
$("button").on("click", function(e) {  
    e.target.text("Kliknięto!");  
});
```



jQuery – manipulacja elementami DOM

Płynne pokazywanie i ukrywanie elementów

```
$("#someDiv").show();
$("#someDiv").hide();
```

Płyne zwijanie i rozwijanie elementów

```
$(".test1").slideUp();
$(".test2").slideDown(300);
$(".test3").slideToggle(function() {
    console.log("Zakończono animację");
});
```

Tworzenie nowych elementów

```
const $div = $("<div class='module'></div>");
```



jQuery – manipulacja elementami DOM cd.

Dołączanie elementów do strony

```
const $div = $(".test-append-cnt");

const $span1 = $("<span>Na końcu</span>");
$div.append($span1);

const $span2 = $("<span>Na początku</span>");
$div.prepend($span2);
```

Usuwanie elementów ze strony

```
const $btn = $(".button");
$btn.remove();
```



Sesja samodzielna - zadania

1. Rozbudowana ankieta dotycząca np. ulubionego serial/filmu
 1. Dla chętnych – ankieta podzielona na kilka części z paskiem postępu.
 2. Zadanie dodatkowe – walidacja formularza
2. Prosta strona (landing page) o wybranej tematyce (ulubiony zespół/film/serial/produkt/hobby)
 1. Dla chętnych – proste animacje w CSS, "pływające" menu nawigacyjne
3. Rozbudowana strona internetowa z kilkoma podstronami oraz poniższymi elementami
 1. Galeria,
 2. Formularz,
 3. Znacznik iframe (np. mapa / wideo),
 4. Pływające menu.
4. * Strona oparta o framework Bootstrap



Sesja samodzielna – zadania cz. 2

1. Skrypt timera,
2. Skrypt stopera,
3. * Skrypt "Pomodoro clock",
4. * Dynamiczny podgląd koloru kafelka w HEX