

# HTML - multimedia

Daniel Bednarek

---

# Organizacja zajęć

- Paczka z plikami do wykorzystania w trakcie zajęć
  - Używaj linków z pliku “zrodla.md” w trakcie zajęć jako pomocy, nie bój się używać Google!
  - Każdy temat ma pliki w folderach “starter” i “final”
  - Wykonuj zadania edytując plik w folderze “starter”
  - Postaraj się nie zaglądać do pliku “final”, wykorzystaj go do powtórki w domu
  - 8h zajęć, przerwy co około 90 minut
  - Przerwa obiadowa około 12:30
  - Zadanie 3/2 => Zadanie numer 2 z rozdziału 3
-

---

# Zakres zajęć

- Box Model
  - Meta Tagi
  - SVG - Scalable Vector Graphics
  - Embedowanie multimedialnych
  - Pliki multimedialne
  - Kaskadowość oraz Specyficzność CSS
  - Wybrane selektory CSS
  - Formularze
  - Animacje CSS
-

---

# Google Color Picker

Wyszukaj frazę “Color Picker” w Google. Zostaw Color Picker w osobnej karcie na później.

---

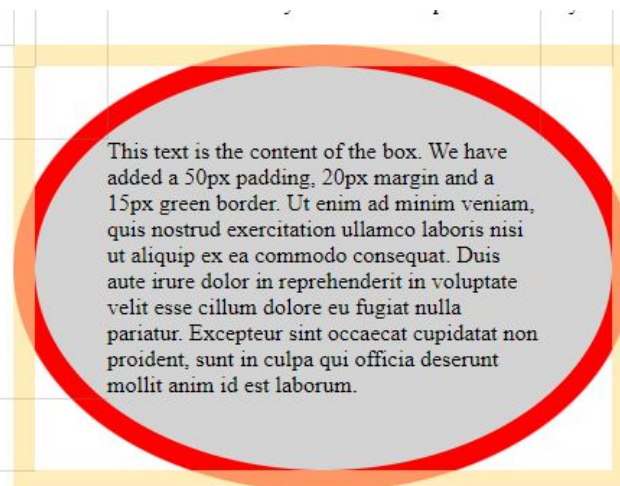
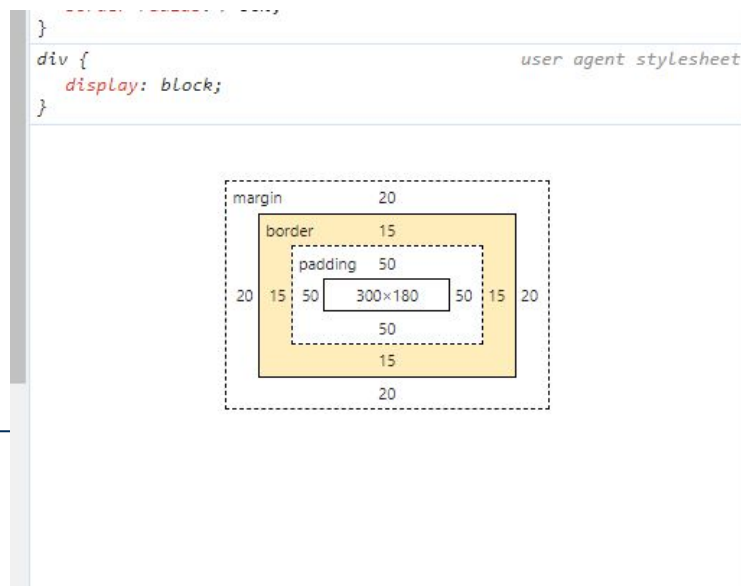
---

# Narzędzia

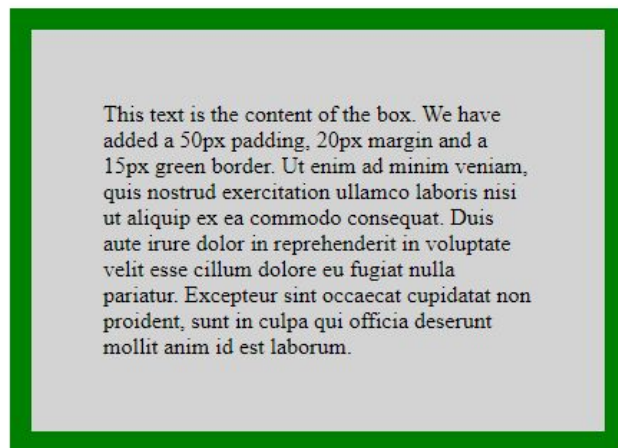
- Wyszukiwarka Google
- Przeglądarka - Chrome
- IDE lub edytor tekstu - najlepiej VS Code

# CSS Box Model

- Wszystkie elementy (tagi) HTML są “pudełkami” o kształcie prostokąta
- Rozkład i wzajemne interakcje tych elementów determinują jak wyglądać będzie nasza strona
- Każde “pudełko” składa się z kilku elementów: Zawartość (Content), Padding, Border, Margin
- DevTools po zbadaniu elementu dają możliwość sprawdzenia rozmiarów poszczególnych elementów każdego “pudełka”
- Nawet, jeżeli zmienimy kształt pudełka na owalny przy pomocy CSS, przeglądarka nadal widzi je jako prostokąt!



```
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid red;
  padding: 50px;
  margin: 20px;
}
```



Styles

Computed

Layout

Event Listeners

DOM Breakpoints

>>

Filter

:hov .cls + -

element.style {

}

div {

background-color: lightgrey;

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

<style>

div {

display: block;

}

user agent stylesheet

---

# Zadanie 0/1

Ściągnij folder **HTML\_MULTIMEDIA** i otwórz go w VSCode.

Otwórz stronę

[https://www.w3schools.com/css/tryit.asp?filename=trycss\\_boxmodel](https://www.w3schools.com/css/tryit.asp?filename=trycss_boxmodel)

(źródło 0.d), kliknij prawym przyciskiem myszy na prostokąt i poszukaj jak wygląda jego box model. Zmień rozmiar elementu, padding itp w CSS po lewej stronie, kliknij przycisk “Run >” i zobacz jak to wpłynie na wartości w DevTools

---



---

# Resetowanie Domyślnych Wartości “Pudełek”

- Przeglądarki nadają domyślne wartości dla poszczególnych elementów pudełek. Jest to niepożądane zachowanie.
- By ustrzec się przed niechcianymi zmianami rozmiarów, należy “zresetować” je przy pomocy CSS
- Jest to polecane przy tworzeniu nowego projektu
- Zazwyczaj wystarczy “zresetować” wartości padding i margin do 0.
- By zaznaczyć wszystkie elementy należy użyć selektora “\*”.

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

example1.css

---

---

# Box-sizing - zmiana kalkulacji wielkości

- Przeglądarki domyślnie obliczają całkowity rozmiar elementu jako rozmiar contentu, co może prowadzić do problemów przy stylowaniu i rozmieszczaniu elementów na stronie
- Aby temu przeciwdziałać, do selektora resetującego padding i margin, dodaje się także właściwość zmieniającą to zachowanie:



- Dzięki temu przeglądarka oblicza rozmiar elementu biorąc pod uwagę także border i padding (**ale nie margin!**)

---

# Meta Tagi

- Dane, które nie pojawiają się na stronie
- Przekazują przeglądarkom i botom informacje o Twojej stronie
- Dodajemy je wewnątrz tagu <head>

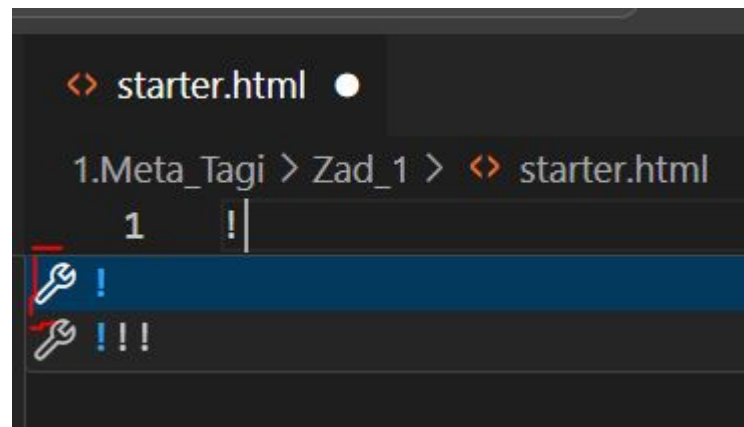
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"> <!--kodowanie znaków -->
  <meta http-equiv="X-UA-Compatible" content="IE=edge"> <!-- wsparcie IE9 -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!--
szerokość ekranu i zoom -->
  <title>Tytuł strony</title> <!-- tytuł strony -->
  <meta name="description" content="Opis Zawartości Strony"> <!-- Opis Strony -->
</head>
<body>
  <!-- TREŚĆ STRONY -->
</body>
</html>
```

---

---

## Zadanie 1/1

Przejdź do folderu **1.Meta\_Tagi/Zad\_1** i otwórz plik **starter.html**. Plik powinien być pusty. Wpisz wykrzyknik “!” na samym początku pliku. Wybierz pierwszą opcję:



Znajdź metatag odpowiedzialny za tytuł strony i zmień jego treść na swoje imię. Zapisz plik i otwórz go w przeglądarce.

Gdzie pojawiło się Twoje imię?

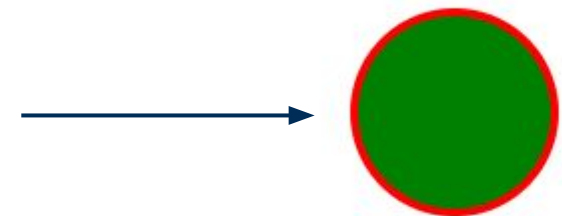
---

---

# SVG - Scalable Vector Graphics

- używany do grafiki 2D - schematów, ilustracji itp.
- pliki wektorowe, dowolnie skalowalne bez utraty jakości
- przechowywane jako wzory matematyczne, a nie piksele

```
<svg width="300" height="300">  
  <circle cx="150" cy="150" r="50"  
    stroke="red" stroke-width="4" fill="green" />  
</svg>
```



---

# Podstawowe elementy SVG

```
<svg width="300" height="300">  
  <circle cx="150" cy="150" r="50"  
    stroke="red" stroke-width="4" fill="green" />  
</svg>
```

- Całość kodu umieszczona jest w elemencie <svg> </svg>
- Atrybuty "width" i "height" określają szerokość i wysokość całego elementu obrazu
- Element <circle> określa narysowany kształt
- SVG posiada kilka gotowych do użycia kształtów: <rect>, <circle>, <ellipse>, <line>, <polyline>, <polygon>, <path> i wiele innych!
- kształty posiadają swoje atrybuty (na przykład "cx", "cy" to pozycja elementu <circle> na polu <svg>, a "r" to jego promień)
- "stroke" i "stroke-width" to kolor i szerokość (w pikselach) obramowania grafiki
- "fill" to kolor wypełnienia grafiki
- niektóre właściwości SVG można stylować za pomocą CSS (źródło 2.c)
- zwykle bardziej złożone kształty, jak na przykład logo firmy, tworzone są w programie graficznym i eksportowane do pliku SVG

---

# Zalety SVG

- Można je dowolnie zmniejszać lub powiększać bez utraty jakości
  - Zazwyczaj waży mniej niż tradycyjne ilustracje, na przykład .jpeg
  - Czytniki ekranu oraz boty mogą czytać słowa zawarte w kodzie grafiki SVG - ułatwia to czytanie osobom korzystającym z czytników ekranu oraz wyszukiwarkom przy odczytywaniu treści strony
  - Można zmienić kolor prostej grafiki bez konieczności angażowania grafika
  - Kolor grafiki można także zmieniać za pomocą CSS, zależnie od aktualnych potrzeb (na przykład jasne logo, gdy użytkownik włączy tryb ciemny, oraz ciemne logo, gdy włączy tryb jasny)
-

---

# Wady SVG

- Nadają się tylko do prostych grafik
- Kompatybilność między przeglądarkami może się różnić
- Kod SVG może sprawiać wrażenie skomplikowanego



---

## Zadanie 2/1

Otwórz plik **Zad\_1/starter/starter.html** i utwórz SVG o następujących cechach:

- okrąg
- wypełnienie niebieskie
- obramowanie zielone o grubości 6(px)
- promień okręgu 75(px)
- wycentruj okrąg na polu <svg>
- pamiętaj o odpowiednich wymiarach pola <svg> by okrąg był widoczny!
- korzystaj z materiałów 2.a, 2.b oraz Google

*czas: 10 min*

---

---

## Zadanie 2/2

Otwórz plik ***Zad\_2/starter/starter.html*** oraz ***style.css*** i zmień kolor wypełnienia figury za pomocą **CSS** na żółty. Nie edytuj pliku html!

W razie niepewności, użyj źródła **2.c** oraz Google

*czas: 5 min*

---

---

# Embedding

- Umieszczanie zewnętrznych treści na stronie
- Embedować można video z YouTube, piosenki z Spotify, posty na Twitterze, mapy Google i wiele innych!
- Aktualnie używa się elementu `<iframe>` do embedowania zewnętrznych treści, tag `<embed>` jest przestarzały
- Większość serwisów społecznościowych i multimedialnych posiada opcje generowania kodu embedującego indywidualne posty i treści

```
<!DOCTYPE html>
<html>
<body>

<iframe style="border-radius:12px"
src="https://open.spotify.com/embed/track/5tjEFjXJZOiFYfUsH1L6D3?
utm_source=generator" width="100%" height="152" frameborder="0"
allowfullscreen="" allow="autoplay; clipboard-write; encrypted-media;
fullscreen; picture-in-picture" loading="lazy"></iframe>

</body>
</html>
```



## Osadź utwór



Kolor: ● ●

Rozmiar:

Kompaktowy (152px) ▾



100%



**Guns**

Bobaflex

PODGLĄD

**Bury Me With My Guns**



Umieszczając odtwarzacz Spotify na swojej stronie internetowej, akceptujesz **Spotify's Developer Terms** (Warunki Spotify dla deweloperów) oraz **Politykę treści Spotify**.

☒ Pokaż kod

KOPIUJ

```
<iframe style="border-radius:12px"
src="https://open.spotify.com/embed/track/5tjEFjXJZ0iFYfUsh1L6D3?utm_source=generator"
width="100%" height="152" frameBorder="0" allowfullscreen="" allow="autoplay;
clipboard-write; encrypted-media; fullscreen; picture-in-picture" loading="lazy">
</iframe>
```



Umieść film



```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/
YdfNuwMhMg4" title="YouTube video
player" frameborder="0"
allow="accelerometer; autoplay;
clipboard-write; encrypted-media;
gyroscope; picture-in-picture"
allowfullscreen></iframe>
```

☐ Rozpocznij od 0:01

Kopiuuj

## Share



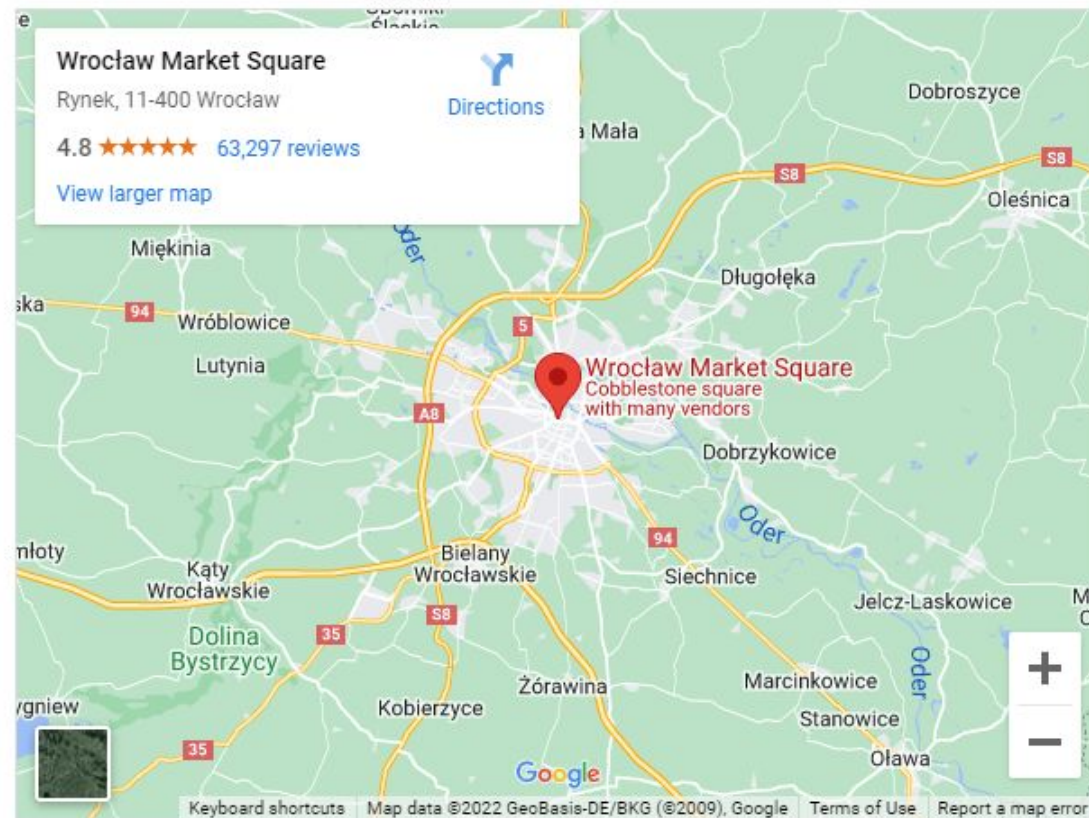
Send a link

Embed a map

Medium ▾

```
<iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12"
```

[COPY HTML](#)



By embedding this map, you agree to the [terms of service](#).

---

## Zadanie 3/1

Embeduj dowolną treść, na przykład mapę Google / wideo z YouTube / post z portalu społecznościowego, w pliku ***starter.html***

***czas: 5 min***

---



---

# Pliki multimedialne

- Przeglądarki umożliwiają umieszczanie treści multimedialnych bezpośrednio z pliku
- Zapewniają też proste UI do kontroli odtwarzania
- Wideo umieszcza się za pomocą tagu <video>, audio za pomocą tagu <audio>
- By sprawdzić obsługiwane formaty plików sprawdź źródło 4.a

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
</video>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>

<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
</audio>

</body>
</html>
```



---

# Pliki Video

- Atrybuty “width” oraz “height” określają wielkość wideo w pikselach. Dobrą praktyką jest zawsze o nich pamiętać, by uniknąć niechcianych zmian rozmiaru
- Atrybut “autoplay” powoduje automatyczne rozpoczęcie wideo przy załadowaniu strony
- Atrybut “muted” wycisza wideo
- Atrybut “controls” dodaje UI do sterowania odtwarzaniem
- Element <source> określa źródło wideo. Można dodać ich kilka i w razie nieobsługiwanego formatu, lub błędu w odczytaniu pliku, przeglądarka otworzy inne źródło
- Zwykły tekst dodany między tagami <video> oraz </video> zostanie wyświetlony tylko w przypadku, jeżeli przeglądarka nie obsługuje tagu <video>

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" autoplay muted controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Twoja przeglądarka nie obsługuje plików video
</video>

</body>
</html>
```

---

## Zadanie 4/1

Otwórz plik ***Zad\_1/starter.html*** i umieść w nim wideo zlokalizowane w tym samym folderze. Nadaj mu wysokość 480px i szerokość 640px. Włącz wyciszenie i UI do sterowania, ale nie włączaj automatycznego startu.

Posiłkuj się źródłem **4.a**

*czas: 10 min*

---

---

# Pliki Audio

- Podobnie jak z wideo, tag <audio> może mieć kilka możliwych źródeł w postaci tagów <source>
- Tu także mamy możliwość włączenia UI, automatycznego odtwarzania i wyciszenia
- Nie musimy określać “width” oraz “height”
- Zwykły tekst dodany między tagami <audio> oraz </audio> zostanie wyświetlony tylko w przypadku, jeżeli przeglądarka nie obsługuje tagu <audio>

```
<!DOCTYPE html>
<html>
<body>

<audio controls autoplay muted>
  <source src="sound.ogg" type="audio/ogg">
  <source src="sound.mp3" type="audio/mpeg">
  Twoja przeglądarka nie obsługuje elementu audio.
</audio>

</body>
</html>
```

---

## Zadanie 4/2

Otwórz plik ***Zad\_2/starter.html*** i dodaj plik audio do strony. Włącz UI kontroli, ale automatyczne odtwarzanie i wyciszenie pozostaw wyłączone. Odpal stronę i sprawdź czy działa.

czas: 10 min

---

---

## Zadanie 4/3 (Bonusowe)

Stylowanie niektórych tagów HTML może być czasami kłopotliwe. Dobrym przykładem jest tag `<audio>`. Nie dość, że różne przeglądarki nadają mu różne style domyślne, to zaznaczanie i stylowanie poszczególnych jego elementów może być kłopotliwe i przynosić niepożądane rezultaty.

Otwórz Twoje rozwiązanie poprzedniego zadania (4/2) i na podstawie artykułu z pliku ***zrodla.md*** z punktu **4.d** zmień kolory poszczególnych elementów tagu `<audio>` na na przykład różowy i żółty. Przeczytaj artykuł do końca.

---

---

# Kaskadowość CSS

- Kaskadowość oznacza, że style aplikowane są od góry do dołu pliku CSS.
- Można niechcący nadpisać wcześniejsze style

```
h1 {  
  color: ■red;  
  background-color: ■grey; /* <----- ten styl nie zostanie nadpisany */  
}  
  
h1 {  
  color: ■blue; /* kolor tekstu tagu <h1> będzie niebieski */  
}
```



# Specyficzność CSS

- Określa, który styl jest “ważniejszy” i zostanie użyty dla danego elementu
- Stylować można na kilka sposobów, z różnym specificity

```
example.html x
5. Kaskadowość i Specyficzność CSS > 5.2 Specyficzność > example.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <title>Specyficzność</title>
5      <link rel="stylesheet" type="text/css" href="example.css" /> <!-- link do
        pliku css -->
6  </head>
7  <body>
8
9      <h1 class="main-title" style="color: red;"> Hello World </h1> <!-- inline
        css -->
10
11 </body>
12
13 <style> /* css w tagu <style> */
14     .main-title {
15         color: green;
16     }
17 </style>
18
19 </html>

# example.css x
5. Kaskadowość i Specyficzność CSS > 5.2 Specyficzność > # example.css > .main-title
1  .main-title { /* css w osobnym pliku */
2      color: blue;
3  }
```

# Specyficzność

\* Specyficzność to złożony i skomplikowany temat. Jeżeli chcesz dowiedzieć się więcej, sprawdź źródła 5.2.a oraz 5.2.b

1. Style "Inline CSS"
2. Style w tagu <style>
3. Style w osobnym pliku CSS



1. Id elementu
2. Klasa elementu
3. Tag elementu

- W przypadku takiej samej specyficzności decyduje kaskadowość
- Im bardziej sprecyzowany styl, tym większa jego specyficzność
- Najwyższy priorytet nadaje *!important* którego powinno się unikać

```
h1 .main-title {  
  color: red;  
}  
  
.main-title {  
  color: green;  
}
```

```
.main-title {  
  color: green !important;  
}
```



---

## Zadanie 5/1

Nadpisz kolor tagu `<h1>` w pliku `Zad_1/starter.html` **bez edytowania zawartości tagu `<style>` i bez użycia *!important***

---

# Wybrane selektory CSS

- CSS umożliwia na stosowanie wielu selektorów do szczegółowej i warunkowej zmiany stylów
- Sprawdź źródło 6.a, aby dowiedzieć się czegoś o każdym z nich

---

# .class1, .class2

Selektor zaznaczający elementy posiadające klasę “.class1” lub “.class2”, lub obie na raz

```
<div class="class1"> Class1</div>  
<div class="class2">Class2</div>  
<div class="class1 class2">Class1 Class2</div>
```



```
.class1, .class2 {  
  color: ■ red;  
  border: solid 1px □ #000000;  
  margin: 5px;  
}
```



Class1

Class2

Class1 Class2



---

# Inne złożone selektory CSS

```
.class1.class2 {  
    /* elementy posiadające obie klasy */  
}  
  
.class1 .class2 {  
    /* element posiadający klasę class2, którego rodzicem (niekoniecznie  
    bezpośrednim) jest element o klasie class1 */  
}  
  
.class1 + .class2 {  
    /* element posiadający klasę class2 i znajdujący się bezpośrednio za  
    elementem z klasą class1 */  
}  
  
.class1 ~ .class2 {  
    /* element posiadający klasę class2 i znajdujący się za elementem z klasą  
    class1 (niekoniecznie bezpośrednio) */  
}
```

# Pseudoelementy ::before i ::after

- Używane do tworzenia pseudoelementów przed, lub za targetowanym elementem
- Przydatne w niektórych przypadkach do stylowania “opornych na stylowanie” elementów HTML, np. radio button
- Po “zaznaczeniu” pseudoelementu jest on tworzony w targetowanym elemencie, przed, lub po zawartości

```
<span class="class1"> Class1</span>
<span class="class2">Class2</span>
<span class="class1 class2">Class1 Class2</span>
<span class="class3">Class3</span>
```

Class1 Class2 Pseudoelement ::beforeClass3Pseudoelement ::after

```
.class3 {
  border: 1px solid black;
  margin: 20px;
}

.class3::before {
  content: "Pseudoelement ::before";
  background-color: #afaffe;
  color: brown;
}

.class3::after {
  content: "Pseudoelement ::after";
  background-color: #afaffe;
  color: brown;
  border: 5px dotted green;
}
```

Więcej przykładów pseudoelementów znajdziesz w źródle 6.b

---

## Zadanie 6/1

Otwórz plik ***Selektory\_CSS/example1.html*** w przeglądarce, uruchom DevTools i zbadaj strukturę elementu z klasą “class3”.  
Gdzie pojawiły się pseudoelementy?

---

---

# Pseudoklasa :hover

- Pseudoklasy definiują specjalny stan elementu
- Jest wiele pseudoklas, o których możesz dowiedzieć się więcej w źródle 6.c
- Selektor “:hover” uaktywnia się po najechaniu na dany element myszką

```
.class4 {  
    border: solid  black 1px;  
    color:  red;  
    background-color:  #bdbdbd;  
    padding: 5px;  
}  
  
.class4:hover {  
    border: solid  #a6a6a6 3px;  
    color:  white;  
    background-color:  black;  
}
```

---

## Zadanie 6/2

Otwórz plik ***Zad\_2/starter/starter.html***. Utwórz plik CSS i podlinkuj go do pliku HTML. Dodaj 2 różne klasy CSS do tagów `<span>` w pliku ***starter.html***. Zmień kolor tła obu elementów `<span>` na szary.

Napisz kod CSS, który sprawi, że pierwszy `<span>` zmieni kolor tła po najechaniu myszką na czerwony, a drugi na niebieski.



---

# Formularze

- Formularze zbierają dane od użytkownika
- Zazwyczaj wysyłają one te dane do serwera, do dalszego procesowania
- Wszystkie elementy formularza zawijamy w tag `<form>` `</form>`

```
<form>
  <label for="first-name">First name:</label><br>
  <input type="text" id="first-name" name="first-name"><br>
  <label for="last-name">Last name:</label><br>
  <input type="text" id="last-name" name="last-name"><br><br>
  <input type="submit" value="Submit">
</form>
```

## HTML Form

First name:

Last name:

Submit

- `<input>` to pole, lub przycisk, zależnie jaki atrybut `"type"` mu nadamy → Więcej typów w źródle 7.b
- `<label>` to opis pola, dodajemy atrybut `"for"` z id tagu `<input>`
- atrybut `"name"` tagu `<input>` jest obowiązkowy, inaczej dane nie zostaną przesłane

# Inne przydatne typy tagu <input>

- radio button

```
<input type="radio" id="yes" name="yes-or-no" value="yes">
<label for="yes">YES</label><br>
```

Do you like trains?

☒ YES  
☐ NO

- checkbox

```
<input type="checkbox" id="yes" name="yes-or-no" value="yes">
<label for="yes">YES</label><br>
```

Do you like trains?

☐ YES  
☒ NO

- button

```
<input
  type="button"
  class="big-button"
  value="Start Nuclear War"
  onclick="alert('BOOOM')"
/>
```

Start Nuclear War

szybkie zadanie! Otwórz plik **example4.html** i zobacz jak działa button. Następnie "odkomentuj" tag <style> i odśwież stronę

- date

```
<label for="birthday">Birthday:</label>
<input type="date" id="birthday" name="birthday" />
```

Birthday:

Otwórz plik **example5.html** i sprawdź jak działa input

- text

```
<label for="fname">First name:</label><br>
<input type="text" id="fname" name="fname" value="John">
```

First name:

...i wiele innych (sprawdź źródło 7.b)

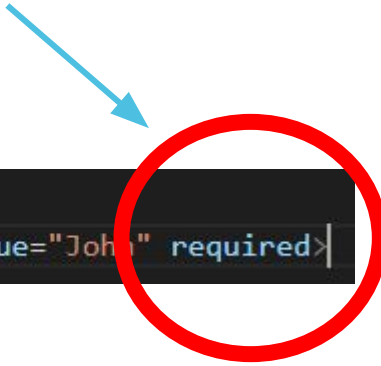
---

# Zadanie 7/1

W pliku **starter.html** stwórz formularz zbierający dane, w którym zapytasz o:

- imię
- nazwisko
- datę urodzenia
- płeć (pamiętaj o odpowiednim typie inputu!)
- email (źródło 7.b)
- akceptację regulaminu (checkbox)
- ustaw imię, nazwisko, oraz akceptację regulaminu jako obowiązkowe do wypełnienia pola

Oraz dodaj na końcu submit button.



```
<label for="fname">First name:</label><br>  
<input type="text" id="fname" name="fname" value="John" required>
```

\* Bonus: Zmień kolor tła formularza na dowolny ciemny, a kolor czcionki na jasny, dodaj padding 20px

**czas: 15 min**

---

---

# Animacje CSS

- Jest kilka sposobów na “animowanie” elementów stron internetowych przy pomocy CSS
- Najprostszy to zmiana stylu przy pomocy pseudoklasy, na przykład “:hover”
- Kolejny to Tranzycje CSS
- Bardziej zaawansowany to Animacje CSS używające zasady “@keyframes”

---

# Tranzycje CSS

- Pozwalają na gładką zmianę stylu w określonym przez nas czasie
- Aby utworzyć efekt tranzycji trzeba określić dwie rzeczy: właściwość CSS do której chcesz dodać efekt, oraz czas trwania efektu

Otwórz plik ***Animacje\_CSS/example1.html*** i sprawdź co dzieje się z elementem po najechaniu myszką.

Zmień kilka rzeczy by uzyskać inny efekt.

```
<body>
  <div>Najedź na mnie!</div>
</body>

<style>
  div {
    width: 50px;
    height: 50px;
    background-color: white;
    transition: width 2s, height 5s, background-color 8s;
  }

  div:hover {
    width: 100px;
    height: 400px;
    background-color: red;
  }
</style>
```

---

## \*@keyframes - do przerobienia w domu (źródło 8.a, 8.b)

- Animacje pozwalające na zmianę jednego stylu w drugi
- Podobne do Transitions, ale mamy więcej kontroli
- By ich używać, trzeba utworzyć najpierw kilka keyframesów (określonych momentów w czasie zawierających pożądany styl)

***Animacje\_CSS/example2.html*** -  
otwórz w przeglądarce i sprawdź  
efekty

```
<h1>CSS Animation</h1>

<div>~\_(\ツ)/~</div>

</body>

<style>
  div {
    color: ■ white;
    width: 100px;
    height: 100px;
    position: relative;
    background-color: ■ red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: infinite;
    animation-timing-function: ease-out;
  }

  @keyframes example {
    0% {
      background-color: ■ red;
      left: 0%;
    }

    50% {
      background-color: ■ blue;
      height: 200px;
      left: 80%;
    }

    100% {
      background-color: ■ red;
      left: 0%;
    }
  }
</style>
```

---

## \*Canvas - do przerobienia w domu (źródło 9.a, 9.b)

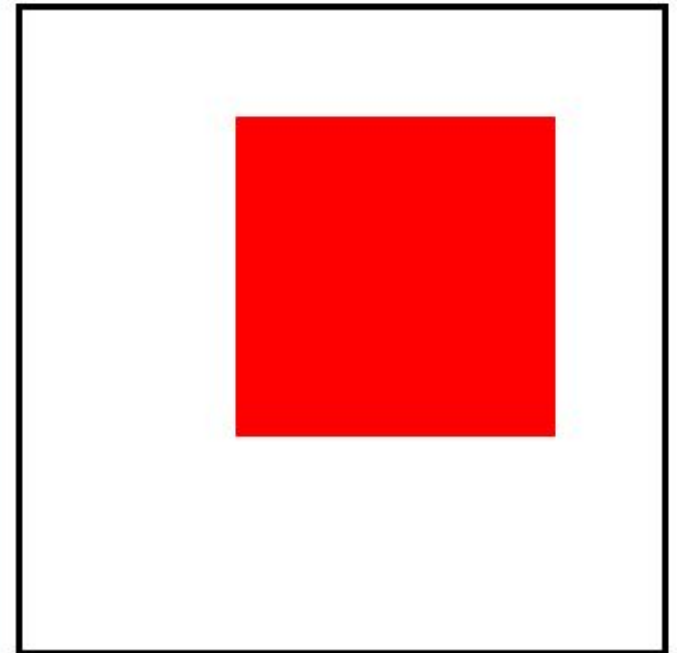
- Element <canvas> używany jest do rysowania grafiki na stronach internetowych
- Element ten jest tylko “płótnem” a rysujemy na nim za pomocą kodu w języku JavaScript
- Może być używany do rysowania dynamicznych wykresów na podstawie zmieniających się danych (na przykład cen akcji spółki)
- Może być animowany i interaktywny

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="300" height="300"
style="border:3px solid #000000;">
</canvas>

<script>
let canvas = document.getElementById("myCanvas");
let ctx = canvas.getContext("2d");
ctx.fillStyle = "#FF0000"; // kolor wypełnienia
ctx.fillRect(100,50,150,150); // pozycja x,y lewego rogu oraz wymiary w px
</script>

</body>
</html>
```





---

# Projekty

## Projekt 1:

Stwórz stronę internetową zawierającą:

- head z meta tagiem "title"
- header z SVG jako logiem - kształt i kolor dowolny
- 3 embedowane posty społecznościowe lub filmy z YouTube, każdy z tytułem umieszczonym w tagu <h2>, jeden pod drugim
- formularz zawierający pola: imię (obowiązkowe), nazwisko (obowiązkowe), płeć, data urodzenia, email(obowiązkowe), submit button
- nadaj submit buttonowi swój własny styl i zaprogramuj by zmieniał kolor gdy najedzie się na niego myszką
- footer z dowolną treścią
- kod CSS umieść w osobnym pliku CSS (pamiętaj o podlinkowaniu pliku CSS w pliku HTML!)

**\*ekstra:** dodaj do swojej strony animacje CSS (na przykład by logo mieniło się kolorami)

Używaj źródeł oraz szukaj rozwiązania w wyszukiwarce Google.

## Projekt 2:

Odwzoruj możliwie najdokładniej stronę internetową:

[Link](#)

(Link także w źródłach)