

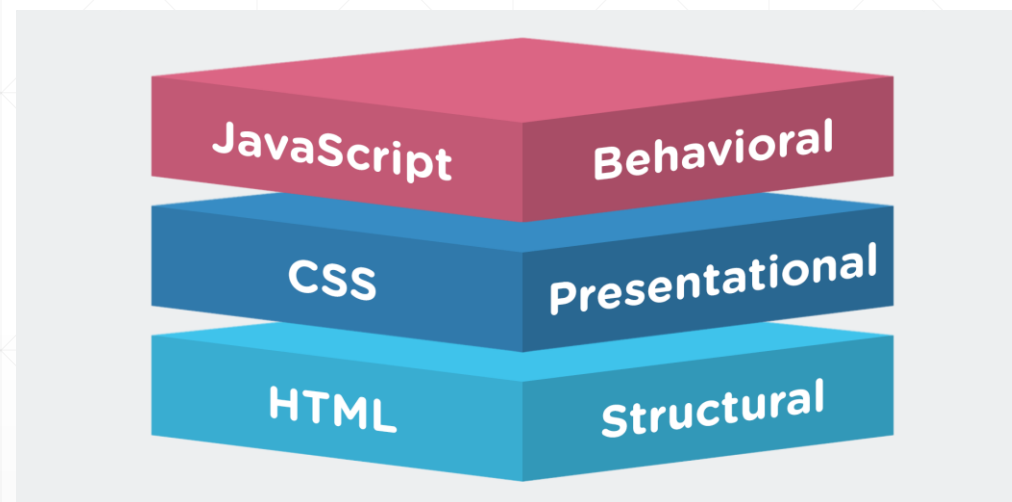
Projektowanie aplikacji webowych

Sebastian Słomian
Java Developer at WSZiB

Sprawy organizacyjne

- Zaliczenie – wykonanie strony HTML na podstawie wyznaczonych danych
 - Termin oddania zadania 22 grudnia 2019 23:59
 - Rozwiązanie zamieścić na własnym zdalnym repozytorium GitHub, link do repozytorium należy przesłać w wiadomości poprzez SUSZI
 - Nagrania zajęć zamieszczone będą na stronie SAKE maksymalnie do tygodnia od zakończenia zajęć
 - Główna przerwa 13.00 – 14.00
-

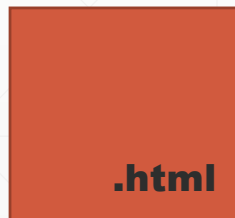
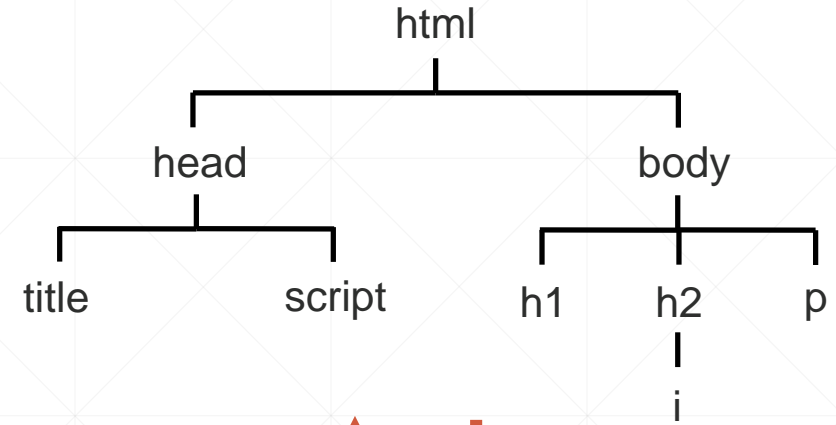
Projektowanie aplikacji webowych



Przeglądarka



Model DOM



.html



.css

Plik html i css



.js

Plik JavaScript

HTML

HYPER TEXT MARKUP LANGUAGE

Ewolucja HTML

- (1991) - Początek języka HTML
 - (1993) - HTML+
 - (1995) - HTML 2.0
 - (1997) - HTML 3.2
 - (1999) - HTML 4.01
 - (2000) - XHTML 1.0
 - (2004) - grupa WHATWG (Web Hypertext Application Technology Working Group)
 - (2014) - HTML5
 - (2015) - HTML5.1
 - (2017) - HTML5.2
-

Struktura pliku HTML

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>Domyślna struktura pliku HTML5</title>
```

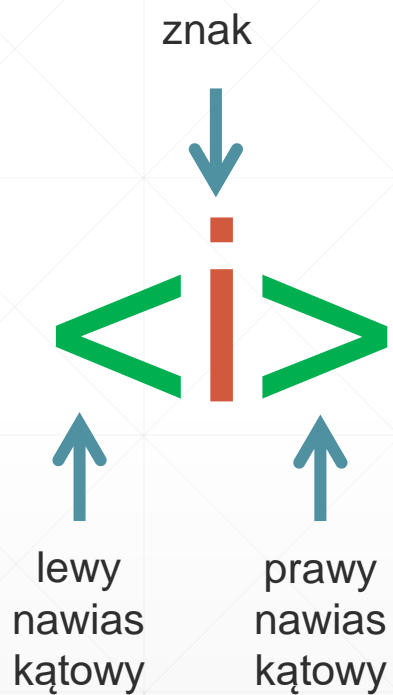
```
</head>
```

```
<body>
```

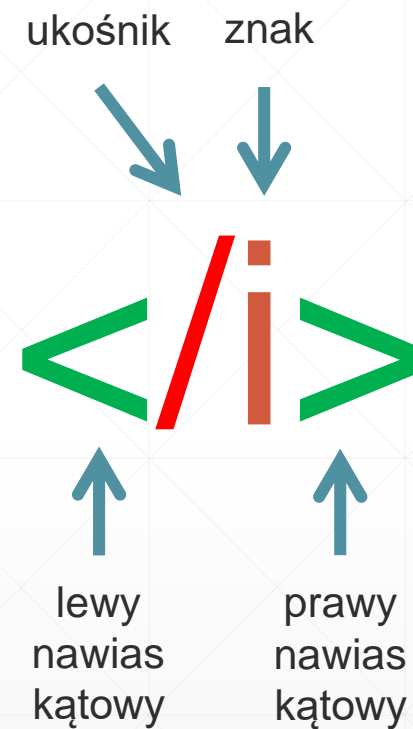
```
</body>
```

```
</html>
```

Znaczniki HTML



ZNACZNIK OTWIERAJĄCY



ZNACZNIK ZAMYKAJĄCY

Atrybuty znaczników HTML

wartość
atrybutu

↑
nazwa
atrybutu

`<i title="tytuł">Tekst</i>`

HTML LINK

nazwa pliku
do którego nastąpi
przekierowanie



```
<a href="home.html">Strona Główna</a>
```



nazwa wyświetlana
na stronie

HTML Obrazy

ścieżka do
pliku obrazu



```

```



atrybut alt
gdy nie zostanie poprawnie
wyświetlony obraz

HTML Tekst - Nagłówki

wartość od 1 do 6



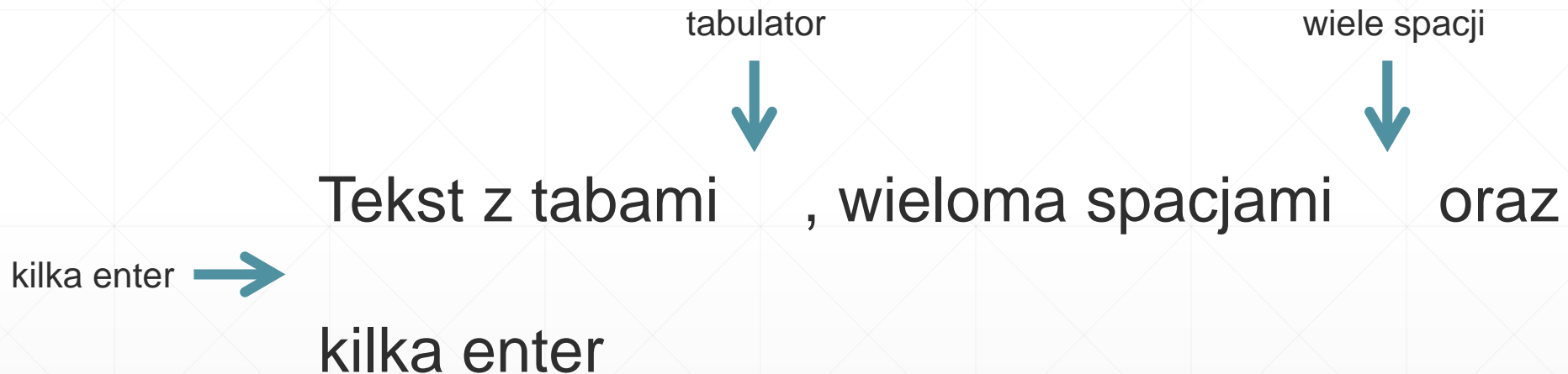
`<h...>Nagłówek</h>`

HTML Tekst - Akapity

`<p>Akapit</p>`

HTML Tekst – Formatowanie tekstu

- Usuwanie niepotrzebnych znaków



Jeśli chcemy zachować dodatkowe znaki w formatowaniu wykorzystujemy znacznik **pre**

- Rozpoczęcie od nowej linii

Tekst w linii pierwszej `
` Tekst w drugiej linii

- Rozpoczęcie od nowej linii oraz pozioma linia oddzielająca

Tekst w linii pierwszej `<hr />` Tekst w drugiej linii oddzielony linią poziomą

- Pogrubienie

`Ważny tekst`



`Pogrubiony tekst`

- Kursywa

`<i>Kursywa</i>`



`Ważna kursywa`

- Akronim

Wykorzystujemy język `<abbr title="Hyper Text Markup Language">HTML</abbr>` do tworzenia stron webowych

- Elementy dodane do treści

`<ins>`Element dodany`</ins>` do treści dokumentu

- Elementy usunięte z treści dokumentu

``Element usunięty`` z treści dokumentu

- Podświetlenie

`<mark>`Podświetlony tekst`</mark>`

- Zmniejszenie tekstu

`<small>`Mała czcionka`</small>`

- Indeks górny

Indeks`^{`górny`}`

- Indeks górny

Indeks`_{`dolny`}`

HTML Atrybut id

`<p id="akapit1">Przykład wykorzystania id</p>`

- Unikatowy na cały dokument HTML
 - Służy do odwołań z kodu CSS i JS
-

HTML Atrybut class

`<p class="akapity">`Przykład wykorzystania class`</p>`

- Każdy element może zawierać kilka atrybutów class
 - Różne elementy HTML mogą odwoływać się do jednego atrybutu class
 - Służy do odwołań z kodu CSS i JS
-

CSS

CASCADING STYLE SHEETS

CSS Składnia

selektor
↓
p {

}

właściwość wartość
↓ ↓
color: **red**;
└──────────┘
deklaracja

CSS – arkusze wewnętrzne



```
<!doctype html>
<html>
<head>
  <title>Wewnętrzny CSS</title>
  <style type="text/css">
    p {
      color: red;
    }
  </style>
</head>
<body>
  <p>Paragraf</p>
</body>
</html>
```

CSS – arkusze zewnętrzne

.html

```
<!doctype html>
<html>
<head>
  <title>Wewnętrzny CSS</title>
  <link href="styles.css" type="text/css"
    rel="stylesheet"/>
</head>
<body>
  <p>Paragraf</p>
</body>
</html>
```

.css

```
p {
  color: red;
}
```


CSS – Kolor

- Poprzez nazwę koloru np. blue, red, yellow
 - RGB(red[0-255], green[0-255], blue[0-255]): zielony `rgb(0, 255, 0)`, czerwony `rgb(255, 0, 0)`
 - Kolor w systemie szesnastkowym: czarny `#000000`, biały `#ffffff`, zielony `#00ff00`
 - RGBA(red[0-255], green[0-255], blue[0-255], alpha[0-1])
 - HSL(hue[0-359], saturation[0-100%], lightness[0-100%]): zielony `hsl(120, 100%, 100%)`, czerwony `hsl(0, 100%, 100%)`
 - HSLA(hue[0-359], saturation[0-100%], lightness[0-100%], alpha[0-1])
-

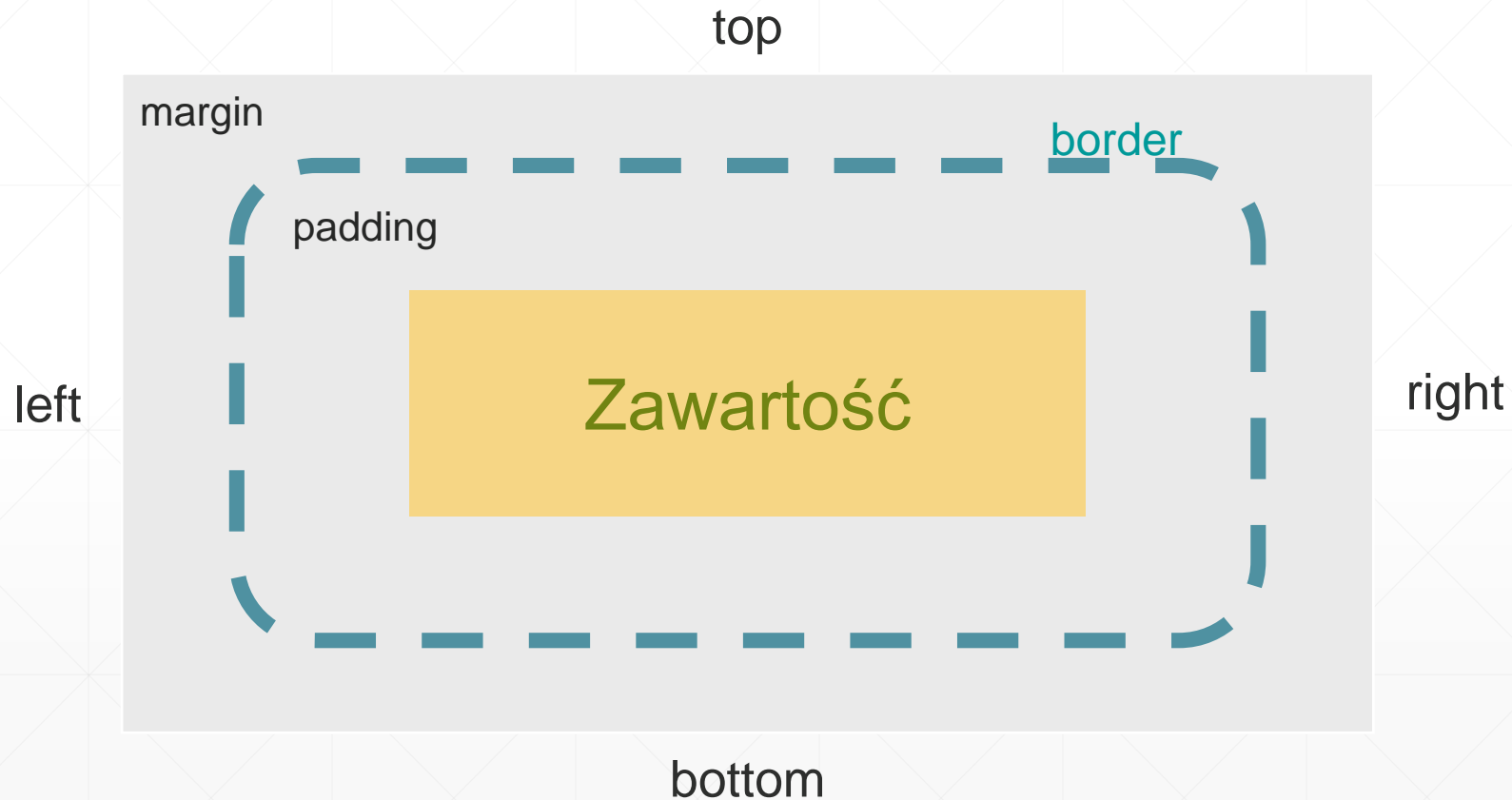
BOX MODEL

MODEL PUDEŁKOWY

Szerokość i wysokość



Obramowanie, margines i wypełnienie



CSS – Podstawowe jednostki długości

- względne:

em, ex, ch, rem

vw, vh, vmin, vmax, %

- bezwzględne:

in, cm, mm, pt, pc, px

Elementy blokowe

Element blokowy 1

Element blokowy 2

Dostępna szerokość



The diagram illustrates two block elements stacked vertically. The top element is a red rectangle labeled 'Element blokowy 1', and the bottom element is a green rectangle labeled 'Element blokowy 2'. Below these elements is a red double-headed arrow spanning the width of the container, labeled 'Dostępna szerokość'. A thin horizontal line is at the bottom of the slide.

Elementy blokowe

Szerokość elementu dostosowuje się do zawartości



Element wewnątrzwierszowy 1

Element wewnątrzwierszowy 2

Element wewnątrzwierszowy 2

Dostępna szerokość



CSS - Selektory

- Selektor uniwersalny

`* {}`

- Selektor typu

`p, i, h1 {}`

- Selektor klasy

`.klasa {}`

- Selektor identyfikatora

`#identyfikator {}`

- Selektor elementu dziecka

`div > p {}`

- Selektor elementu potomnego

`div i {}`

- Selektor elementu sąsiadującego bezp.

`p + i {}`

- Selektor elementu sąsiadującego

`p ~ i {}`

Listy

Lista - nieuporządkowana

Element 1

Element 2

Element 3

Lista - uporządkowana

Element 1

Element 2

Element 3

Lista - definicji

<dl>

<dt>Definicja 1</dt>

<dd>Opis 1</dd>

<dt>Definicja 2</dt>

<dd>Opis 1</dd>

<dd>Opis 2</dd>

<dt>Definicja 3</dt>

<dd>Opis 1</dd>

</dl>

Tabele

Tabela

<table>

<tr>

<th>Nagłówek 1</th>

<th>Nagłówek 1</th>

</tr>

<tr>

<td>Treść 1</td>

<td>Treść 1</td>

</tr>

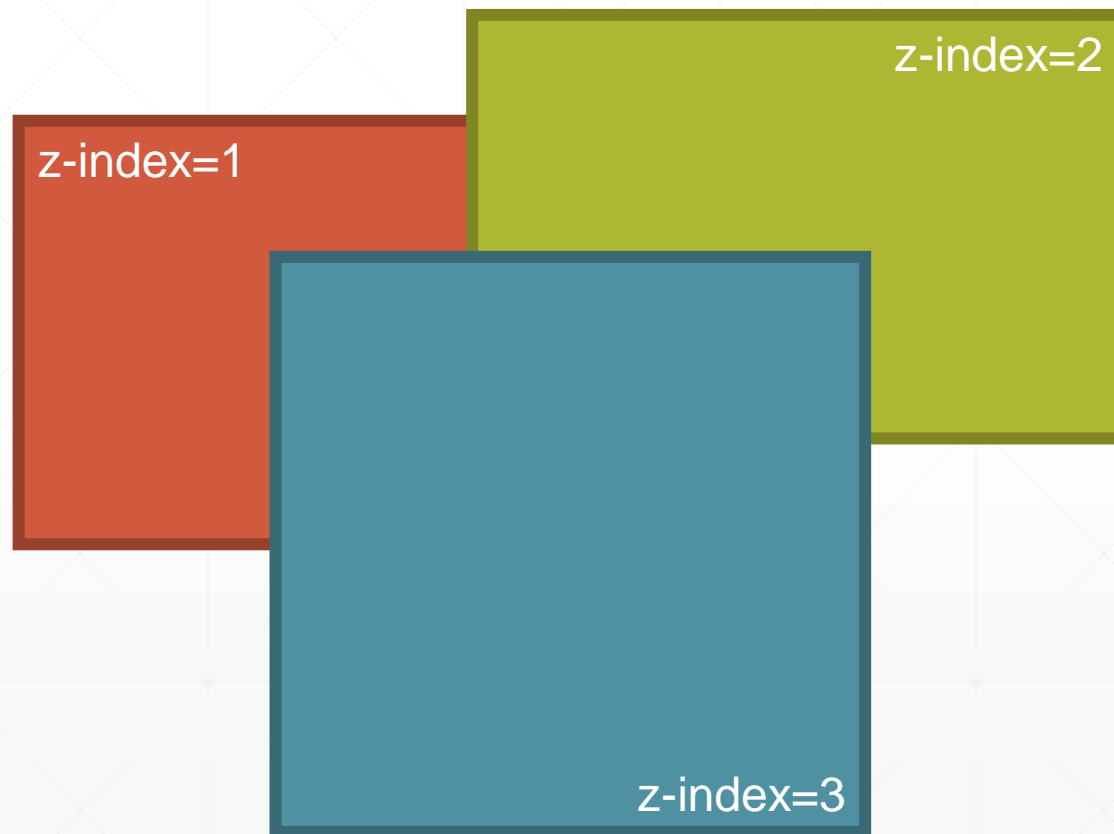
</table>

CSS - Pozycje

CSS - Pozycje

- Static
 - Relative
 - Fixed
 - Absolute
 - Sticky
-

CSS - Pozycje



Flexbox

Flexbox

.css

```
.container {  
  display: flex;  
}
```

kontener flexbox

class="container"



elementy flexbox

CSS - Pseudoklasy i pseudoelementy

Pseudoklasy

```
selektor:pseudo-klasa {  
    ...  
}
```

Pseudoelementy


```
selektor::pseudo-element {  
    ...  
}
```

Formularze

Formularze

adres URL gdzie zostanie
wysłany formularz

metoda HTTP



```
<form action="/actionUrl" method="post">  
...  
...  
...  
</form>
```

Elementy formularzy

```
<input type="text"/>
```

```
<select>
```

```
  <option value="true">Yes</option>
```

```
  <option value="false">No</option>
```

```
</select>
```

```
<textarea></textarea>
```

```
<button>Save</button>
```

JavaScript

HTML – warstwa zawartości, szkielet strony internetowej

CSS – warstwa prezentacji strony internetowej

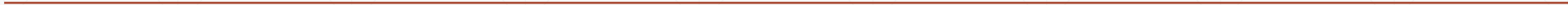
JavaScript – warstwa zachowania strony internetowej

Dołączanie kodu JavaScript

- wewnętrzne head



```
<!doctype html>
<html>
<head>
  <title>Wewnętrzny JS w head</title>
  <script>
    console.log('JavaScript');
  </script>
</head>
<body>
</body>
</html>
```



Dołączanie kodu JavaScript

- wewnętrzne body



```
<!doctype html>
<html>
<head>
  <title>Wewnętrzny JS w body</title>
</head>
<body>
  <input type="button" onclick="console.log('JavaScript');">
</body>
</html>
```

Dołączanie kodu JavaScript

- zewnętrzne

.html

.js

```
<!doctype html>
<html>
<head>
  <title>Wewnętrzny JS w body</title>
  <script src="script.js"></script>
</head>
<body>
</body>
</html>
```

```
console.log('JavaScript');
```

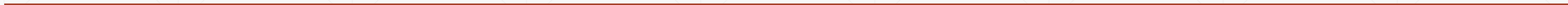
Deklaracja zmiennej

słowo kluczowe
zmiennej

średnik kończy
linię

var nazwaZmiennej;
(let)

nazwa zmiennej



Przypisanie zmiennej

znak przypisania



zmienna liczbowa

```
var nazwaZmiennej = -5.0;  
(let)
```

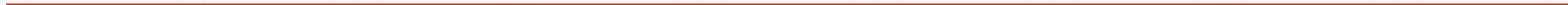
zmienna tekstowa

```
var nazwaZmiennej = 'Tekst';  
(let)
```

zmienna boolowska

```
var nazwaZmiennej = true;  
(let)
```

Typy proste



Tablice - deklaracja

tablica liczb

```
var nazwaZmiennej = [1, 2, 3];  
(let)
```

tablica tekstowa

```
var nazwaZmiennej = ['jeden', 'dwa'];  
(let)
```

tablice boolowska

```
var nazwaZmiennej = [true, false];  
(let)
```

Tablice – dostęp do n-go elementu tablicy

```
var nazwaZmiennej = 0['jeden', 1'dwa', 2'trzy'];  
(let)
```

`nazwaZmiennej[2]` element 'trzy'



wyraz tablicy
od 0 do n-1

Ilość elementów
w tablicy

`nazwaZmiennej.length`

Funkcje - deklaracja

słowo kluczowe
funkcji



nazwa funkcji



parametry funkcji



```
function nazwaFunkcji(parametr1, parametr2) {
```

ciało funkcji



...

...

...

```
return 0;
```



słowo kluczowe gdy
chcemy aby funkcja
zwróciła wartość

```
}
```

Funkcje - wywołanie

`nazwaFunkcji(parametr1, parametr2);`

Obiekty

zmienna --> właściwość obiektu

funkcja --> metoda obiektu

Tworzenie obiektów

- poprzez notacje literałów

```
var osoba = {  
  imie: 'Jan',  
  nazwisko: 'Nowak',  
}
```

właściwości

metoda {

```
  idz: function() {  
  }  
}
```

Tworzenie obiektów

- poprzez notację konstruktora

```
function Osoba(imie, nazwisko) {  
  this.imie = imie;  
  this.nazwisko = nazwisko;  
}
```

właściwości

metoda {
 this.idz = function() {
 }
}

wywołanie konstruktora

słowo kluczowe
dla nowych obiektów

```
var osoba = new Osoba('Jan', 'Kowalski');
```

Obiekty - dostęp do właściwości i metod

Przypisanie do zmiennej wartości
właściwości imię

```
var imie = osoba.imie;
```

Zamiana wartości właściwości imię

```
osoba.imie = 'Jan';
```

Wywołanie metody obiektu

```
osoba.idz();
```

Operatory

- przypisania

zmienna = 'Przykładowy tekst';

- porównania

==

!=

===

!==

>

<

>=

<=

- arytmetyczne

+

-

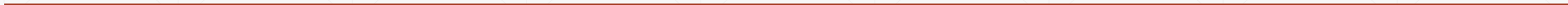
/

*

++

--

%



Operatory

- logiczne

&&

||

!

- ciągów tekstowych

zmiennaTekstowa = 'Jeden, ' + 'dwa';

Decyzje - IF

warunek

↓

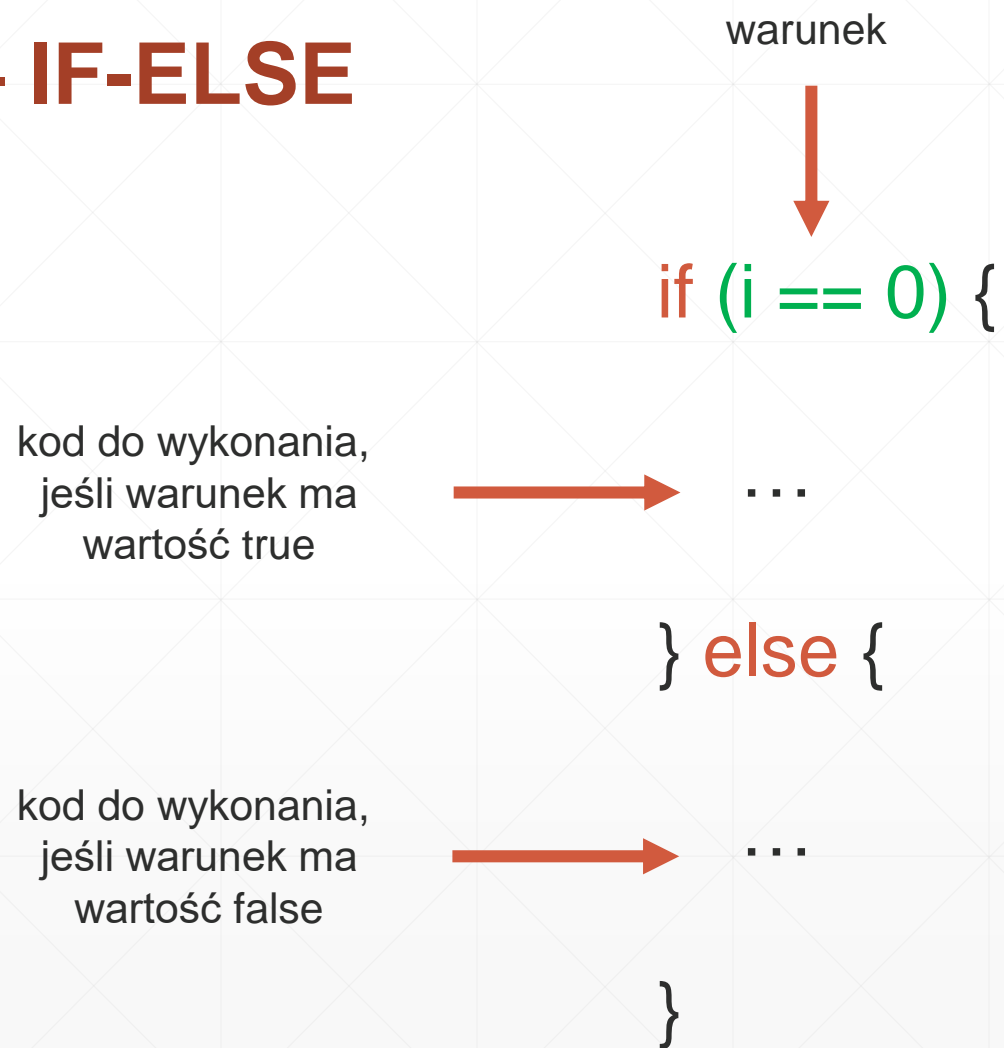
```
if (i == 0) {
```

...

```
}
```

kod do wykonania,
jeśli warunek ma
wartość true →

Decyzje – IF-ELSE



Decyzje – Switch - Case

kod do wykonania,
jeśli zmienna ma
wartość jeden



```
switch (zmienna) {
```

```
case 'jeden':
```

```
...
```

```
break;
```

kod do wykonania,
jeśli zmienna ma
wartość dwa



```
case 'dwa':
```

```
...
```

```
break;
```

kod do wykonania,
jeśli zmienna nie
ma żadnej z powyższych
wartości



```
default:
```

```
...
```

```
break;
```

```
}
```

Pętle - For

warunek (licznik)



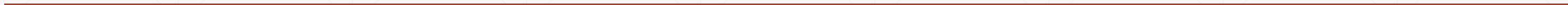
```
for (let i = 0; i < 10; i++) {
```

kod do wykonania
w trakcie pętli



...

```
}
```



Pętle - While



Pętle – Do-While

kod wykonany przynajmniej
jeden raz i dopóki warunek
jest spełniony



```
do {
```

```
...
```

```
} while (true);
```

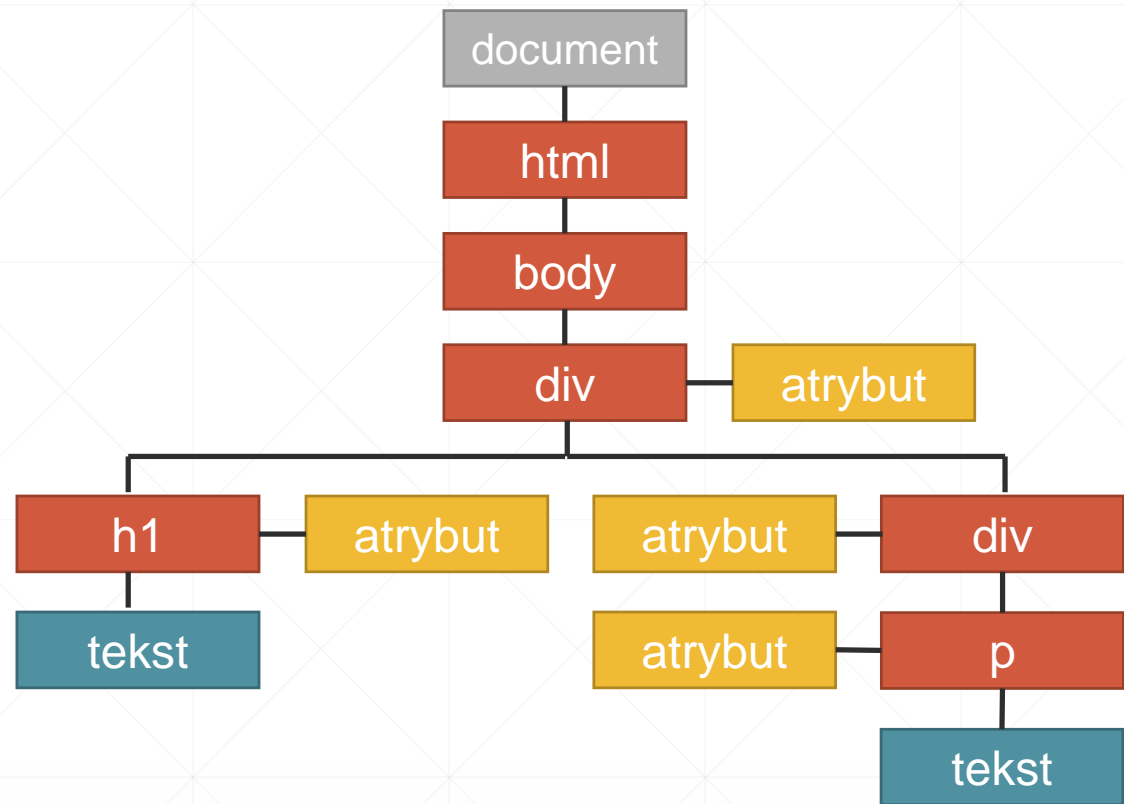


warunek

DOM

Obiektowy model dokumentu


```
<html>
<body>
  <div>
    <h1>Nagłówek</h1>
    <div>
      <p>Paragraf</p>
    </div>
  </div>
</body>
</html>
```



document	węzeł document
html	węzeł elementów
atribut	węzeł atrybutów
tekst	węzły tekstowe

Dostęp do DOM

- wybór konkretnego węzła elementu

wybór elementu za pomocą id

```
document.getElementById('idElementu');
```

wybór elementu za pomocą selektora css

```
document.querySelector('h1');
```

zwracany jest pierwszy element

Dostęp do DOM

- wybór wielu elementów

wybór elementów za pomocą klasy

```
document.getElementsByClassName('klasaElementu');
```

wybór elementów za pomocą nazwy znacznika

```
document.getElementsByTagName('nazwaZnacznika');
```

wybór wszystkich elementów za pomocą selektora css

```
document.querySelectorAll('h1');
```

Dostęp do DOM

- poruszanie się między elementami

wybór rodzica elementu

`.parentNode`

wybór poprzedniego/następnego elementu na tym samym poziomie

`.previousElementSibling`

`.nextElementSibling`

wybór pierwszego/ostatniego dziecka elementu

`.firstElementChild`

`.lastElementChild`

JS – modyfikacja CSS

document.getElementById('idEl').style.color = 'red';

↑ Uzyskanie dostępu do elementu

↓ Dostęp do stylowania elementów

↑ Właściwość css

↓ Wartość

Edycja zawartości elementów

- `textContent`

```
document.getElementById('id').textContent = 'Nowa wartość';
```

- `innerHTML`

```
document.getElementById('id').innerHTML = '<b>Wartość</b>';
```

Dodawanie elementów do DOM

1)

```
let nowyElement = document.createElement('div');
```

nazwa znacznika



2)

```
nowyElement.textContent('tekst');
```

3)

```
document.getElementById('id').appendChild(nowyElement);
```

Usuwanie elementów z DOM

1)

```
let elementDoUsuniecia = document.getElementById('idDziecka');
```

2)

```
let elementNadrzędny = document.getElementById('idRodzic');
```

3)

```
elementNadrzędny.removeChild(elementDoUsuniecia);
```

Atrybuty elementów

- dodanie atrybutu

```
document.getElementById('id').setAttribute('hidden', '');
```

nazwa atrybutu



- usunięcie atrybutu

```
document.getElementById('id').removeAttribute('hidden');
```

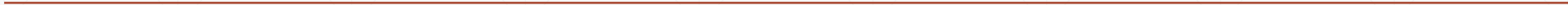
nazwa atrybutu



- sprawdzenie czy element posiada atrybut

```
document.getElementById('id').hasAttribute('hidden');
```

nazwa atrybutu



Events

Obsługa zdarzeń

Obsługa zdarzeń

- w atrybutach HTML

.html

...

```
<button onclick='clickMe()>Kliknij</button>
```

...

.js

```
function clickMe() {  
    console.log('Przycisk został naciśnięty');  
}
```

Obsługa zdarzeń

- w modelu DOM

.html

...

```
<button id='myButton'>Kliknij</button>
```

...

.js

```
function clickMe() {  
    console.log('Przycisk został kliknięty');  
}
```

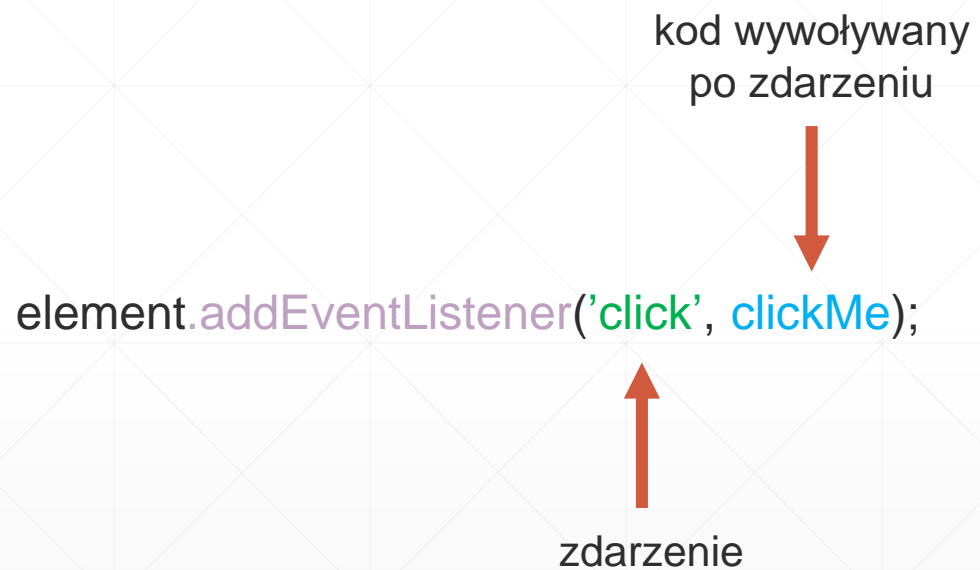
```
var element = document.getElementById('myButton');  
element.onclick = clickMe;
```



zdarzenie

Obsługa zdarzeń

- obserwator zdarzeń



Obsługa zdarzeń

- w modelu DOM

.html

...

```
<button id='myButton'>Kliknij</button>
```

...

.js

```
function clickMe() {  
    console.log('Przycisk został kliknięty');  
}
```

```
var element = document.getElementById('myButton');  
element.addEventListener('click', clickMe);
```

Walidowanie formularzy

jQuery
