

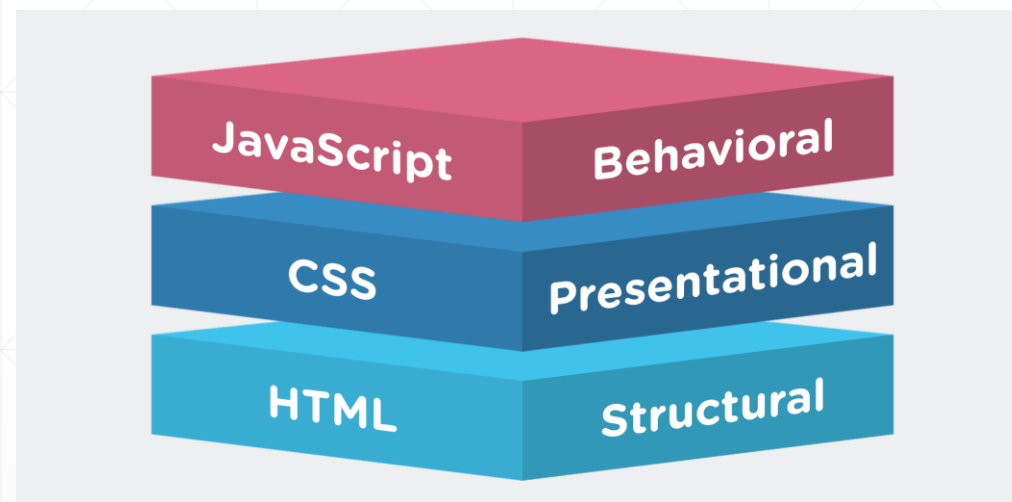
Projektowanie warstwy klienta aplikacji webowych

Sebastian Słomian

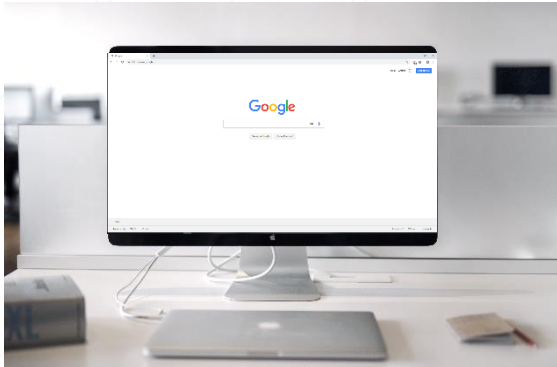
Sprawy organizacyjne

- Zaliczenie – 2 testy jednokrotnego wyboru, pierwszy dostępny na SAKE od 17.04.2023 08:00 do 07.05.2023 23:59, 10 pytań 30 minut – 50% na zaliczenie.
 - Nagrania zajęć zamieszczone będą na stronie SAKE maksymalnie do tygodnia od zakończenia zajęć
 - Główna przerwa 13.00 – 14.00
-

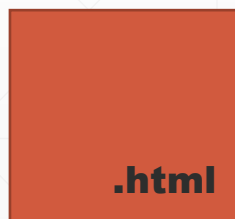
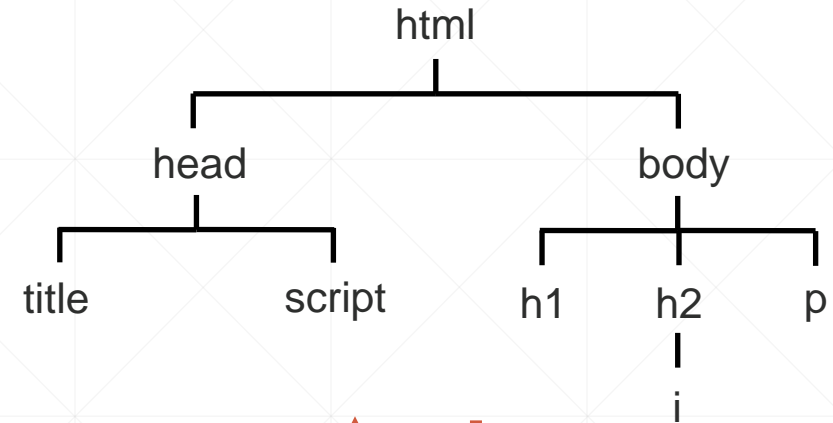
Projektowanie aplikacji webowych



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>
```

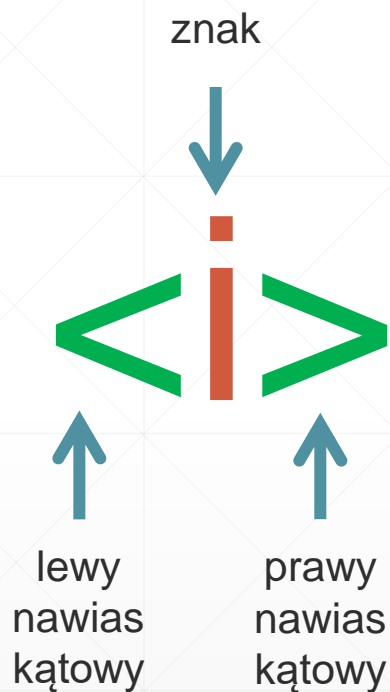
```
</head>
```

```
<body>
```

```
</body>
```

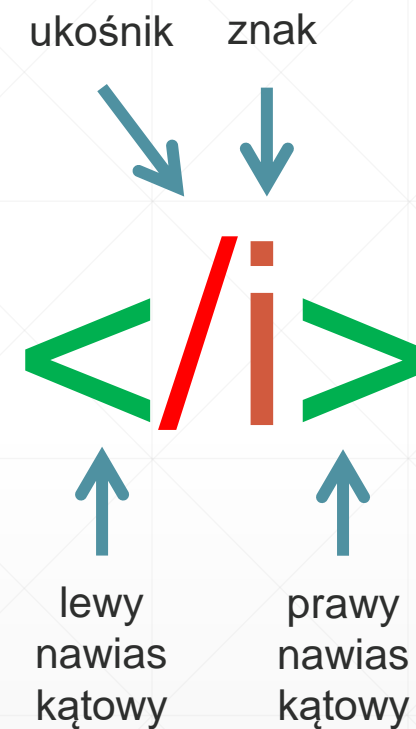
```
</html>
```

Znaczniki HTML



ZNACZNIK OTWIERAJĄCY

zawartość znacznika

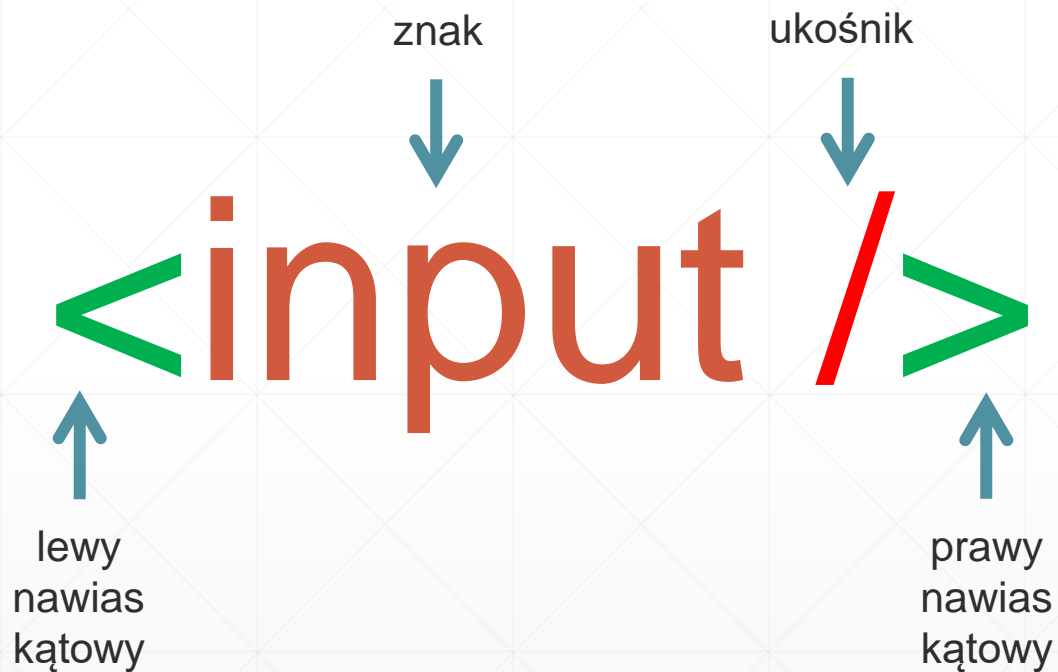


ZNACZNIK ZAMYKAJĄCY

Znaczniki HTML – empty tags

Przykładowe znaczniki samodomykające:

- input
- br
- hr
- img
- meta
- link



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>`

Nagłówek poziom 1

Nagłówek poziom 2

Nagłówek poziom 3

Nagłówek poziom 4

Nagłówek poziom 5

Nagłówek poziom 6

HTML Tekst - Akapity

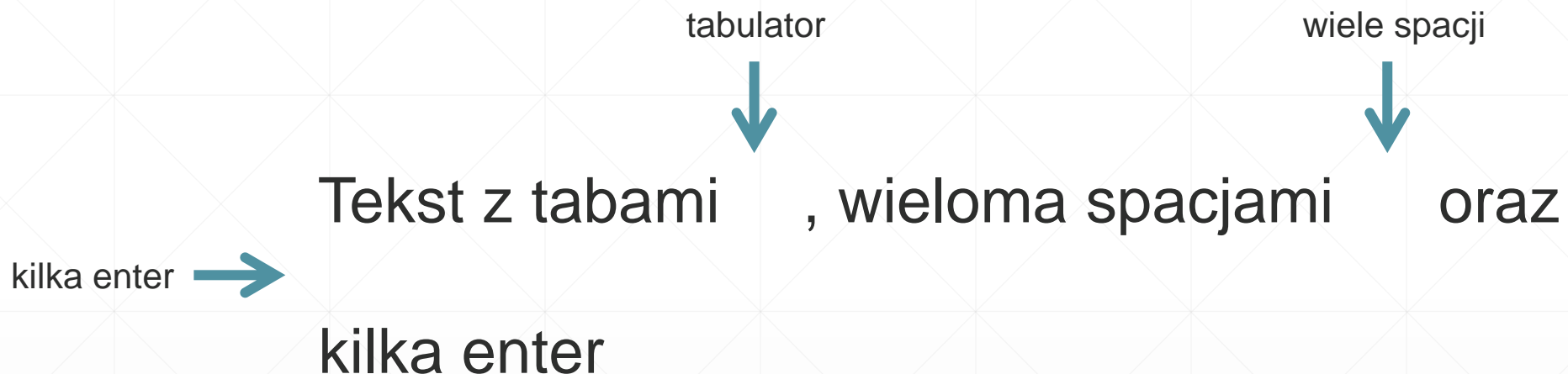
`<p>Akapit</p>`

Akapit 1

Akapit 2

HTML Tekst – Formatowanie tekstu

- Usuwanie niepotrzebnych znaków



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

```
Tekst z tabami , wieloma spacjami oraz kilka enter
Tekst z tabami , wieloma spacjami      oraz
kilka enter
```

- 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

Tekst w linii pierwszej
Tekst w drugiej linii

Tekst w linii pierwszej

Tekst w drugiej linii oddzielony linią poziomą

- Pogrubienie

To jest ``ważny tekst``



To jest ``pogrubiony tekst``

To jest **ważny tekst**
To jest **pogrubiony tekst**

- Kursywa

To jest `<i>`kursywa`</i>`



To jest ``ważna kursywa``

To jest *kursywa*
To jest *ważna kursywa*

- Akronim

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

Wykorzystuje język HTML do tworzenia stron webowych

- Elementy dodane do treści

To jest `<ins>`dodany tekst`</ins>`

To jest dodany tekst

- Elementy usunięte z treści dokumentu

To jest ``usunięty tekst``

To jest ~~usunięty tekst~~

- Podświetlenie

To jest `<mark>`podświetlony tekst`</mark>`

To jest podświetlony tekst

- Zmniejszenie tekstu

To jest `<small>`mała czcionka`</small>`

To jest mała czcionka

- Indeks górny

To jest `^{`indeks górny`}`

To jest indeks górny

- Indeks dolny

To jest `_{`indeks dolny`}`

To jest 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>
    p {
      color: red;
    }
  </style>
</head>
<body>
  <p style="color: red">Paragraf</p>
</body>
</html>
```

CSS – arkusze zewnętrzne

.html

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

.css

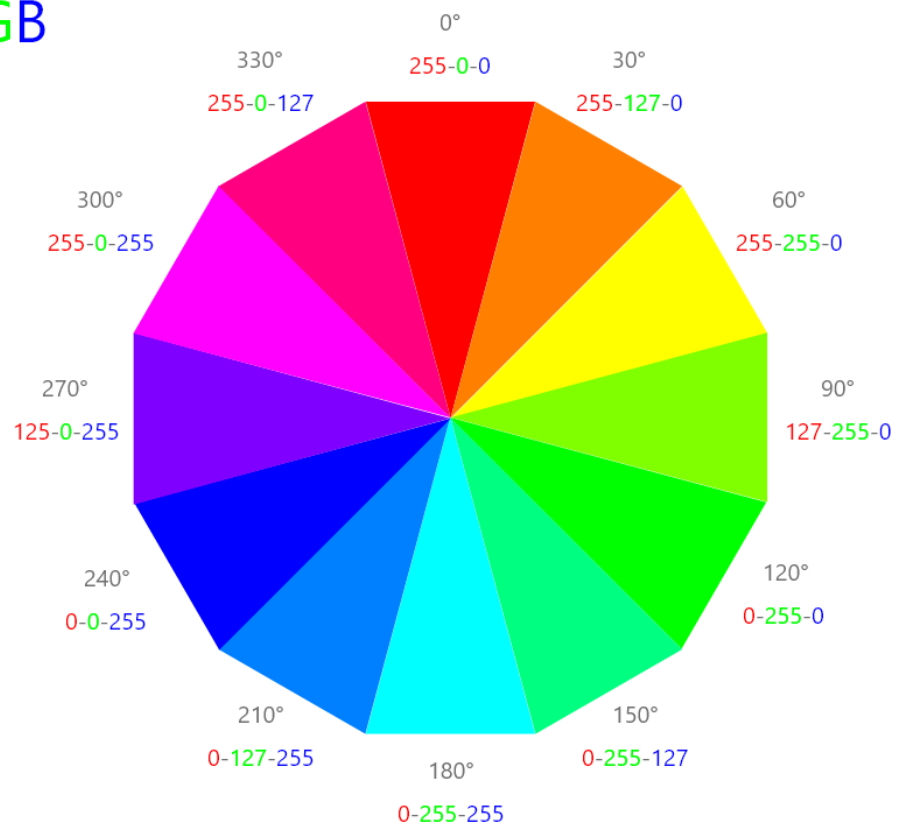
```
p {
  color: red;
}
```

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(`#000000`): czarny `#000000`, biały `#ffffff`, zielony `#00ff00`
 - RGBA(red[0-255], green[0-255], blue[0-255], alpha[0-1]): zielony `rgba(0, 255, 0, 1.0)`
 - HSL(hue[0-359], saturation[0-100%], lightness[0-100%]): zielony `hsl(120, 100%, 50%)`, czerwony `hsl(0, 100%, 50%)`
 - HSLA(hue[0-359], saturation[0-100%], lightness[0-100%], alpha[0-1]): czerwony `hsla(0, 100%, 50%, 1.0)`
-

CSS – RGB / HSL

RGB



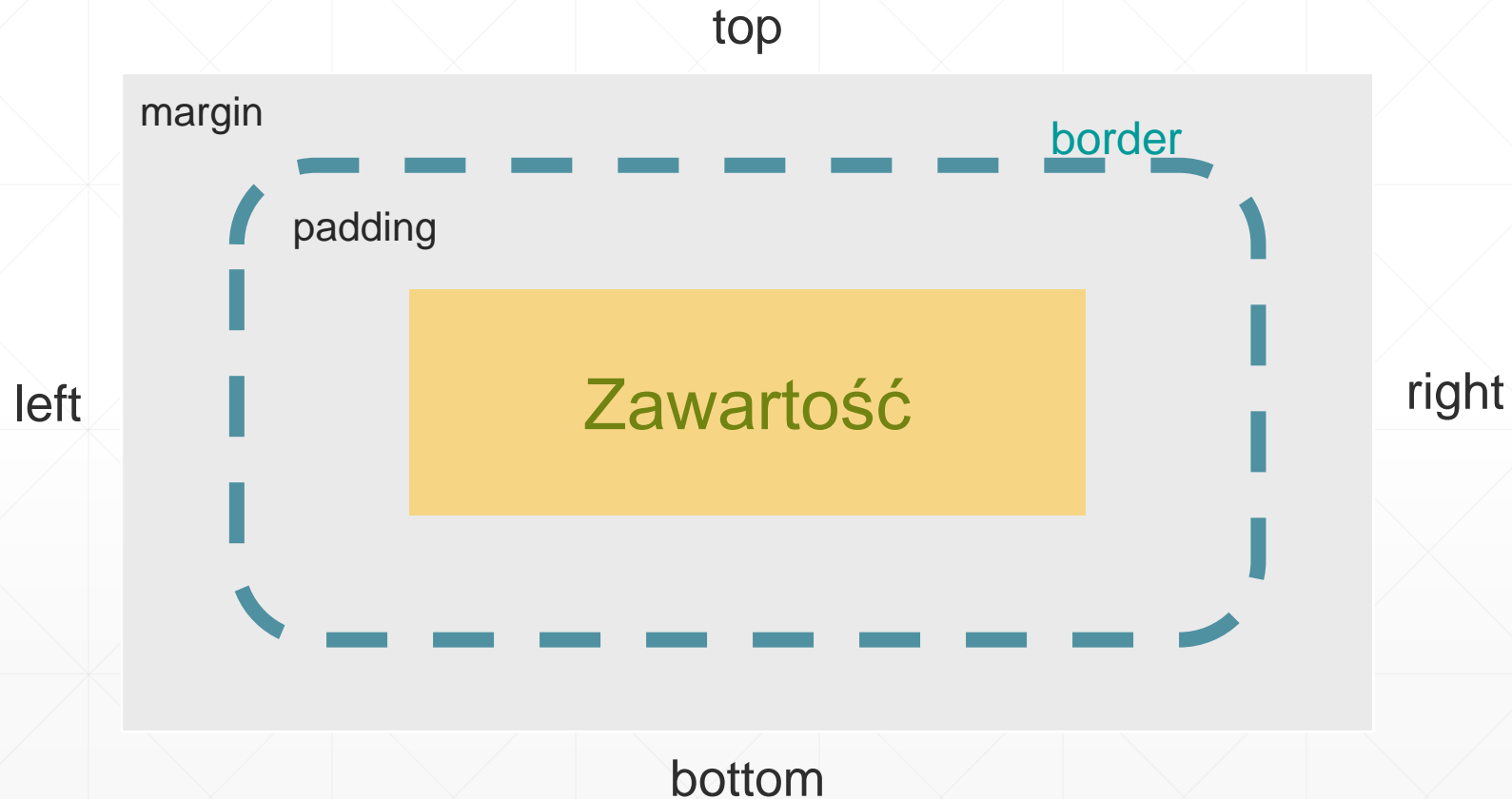
BOX MODEL

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 stacked rectangular blocks. The top block is reddish-brown and labeled 'Element blokowy 1'. The bottom block is olive green and labeled 'Element blokowy 2'. Below these blocks is a horizontal double-headed arrow, also in a reddish-brown color, with the text 'Dostępna szerokość' centered above it. A thin horizontal line is positioned at the bottom of the diagram.

Elementy wewnątrz wierszowe

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

- Element 1
 - Element 2
 - Element 3
-

Lista - uporządkowana

Element 1

Element 2

Element 3

1. Element 1
 2. Element 2
 3. Element 3
-

Lista - definicji

<dl>

<dt>Definicja 1</dt>

<dd>Opis 1</dd>

<dd>Opis 2</dd>

<dt>Definicja 2</dt>

<dd>Opis 1</dd>

<dd>Opis 2</dd>

<dd>Opis 3</dd>

</dl>

Definicja 1
Opis 1
Opis 2
Definicja 2
Opis 1
Opis 2
Opis 3

Tabele

Tabela

```
<table>
```

```
<tr>
```

```
<th>Nagłówek 1</th>
```

```
<th>Nagłówek 1</th>
```

```
</tr>
```

```
<tr>
```

```
<td>Treść 1</td>
```

```
<td>Treść 1</td>
```

```
</tr>
```

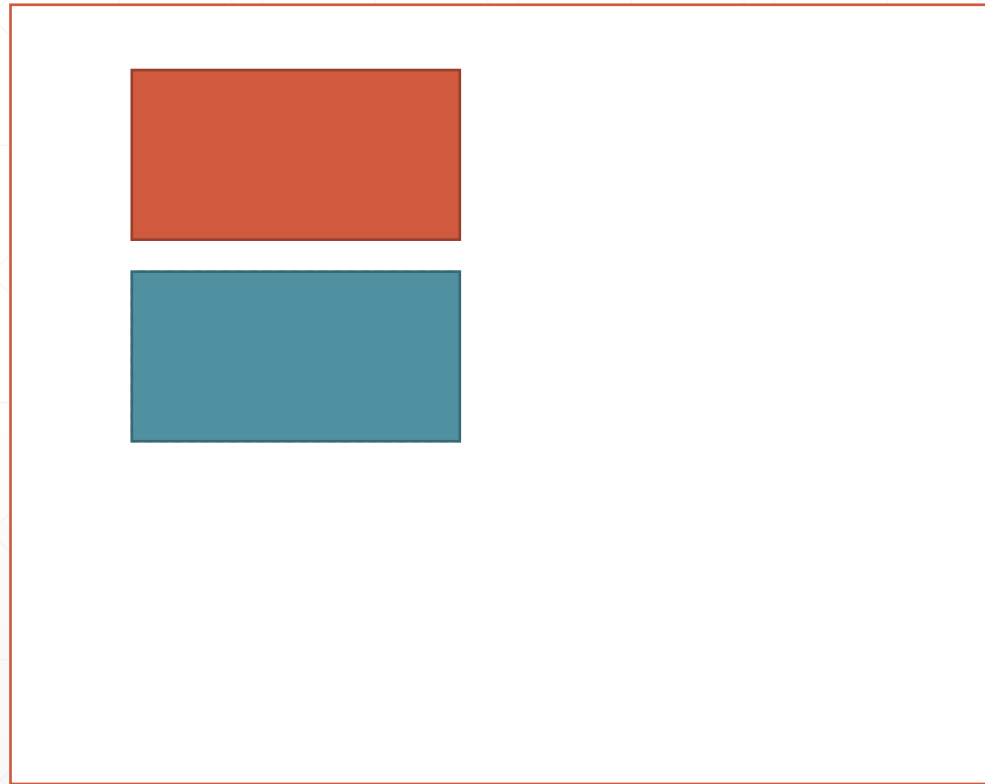
```
</table>
```

CSS - Pozycje

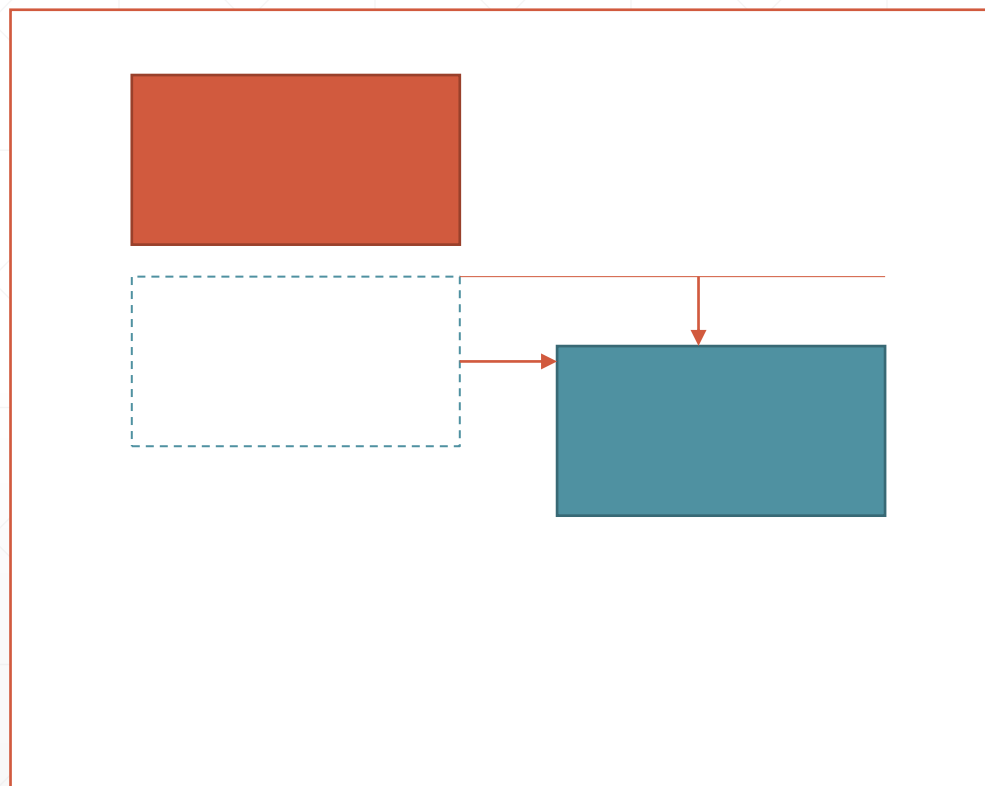
CSS - Pozycje

- Static
 - Relative
 - Fixed
 - Absolute
 - Sticky
-

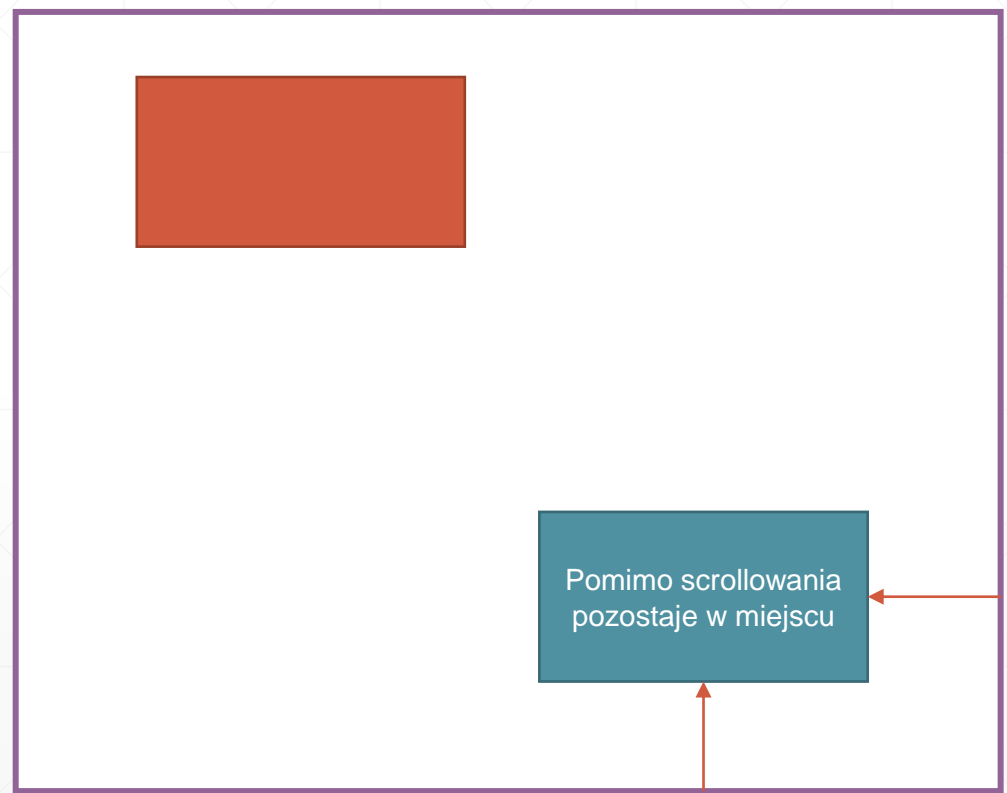
Static



Relative



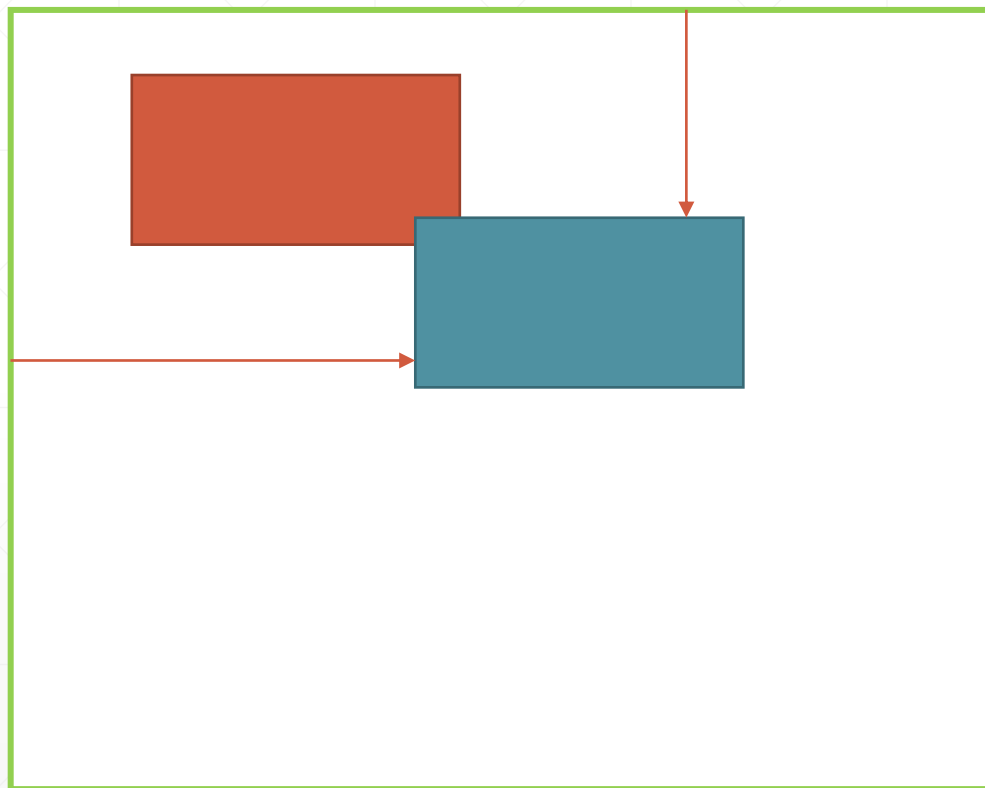
Fixed



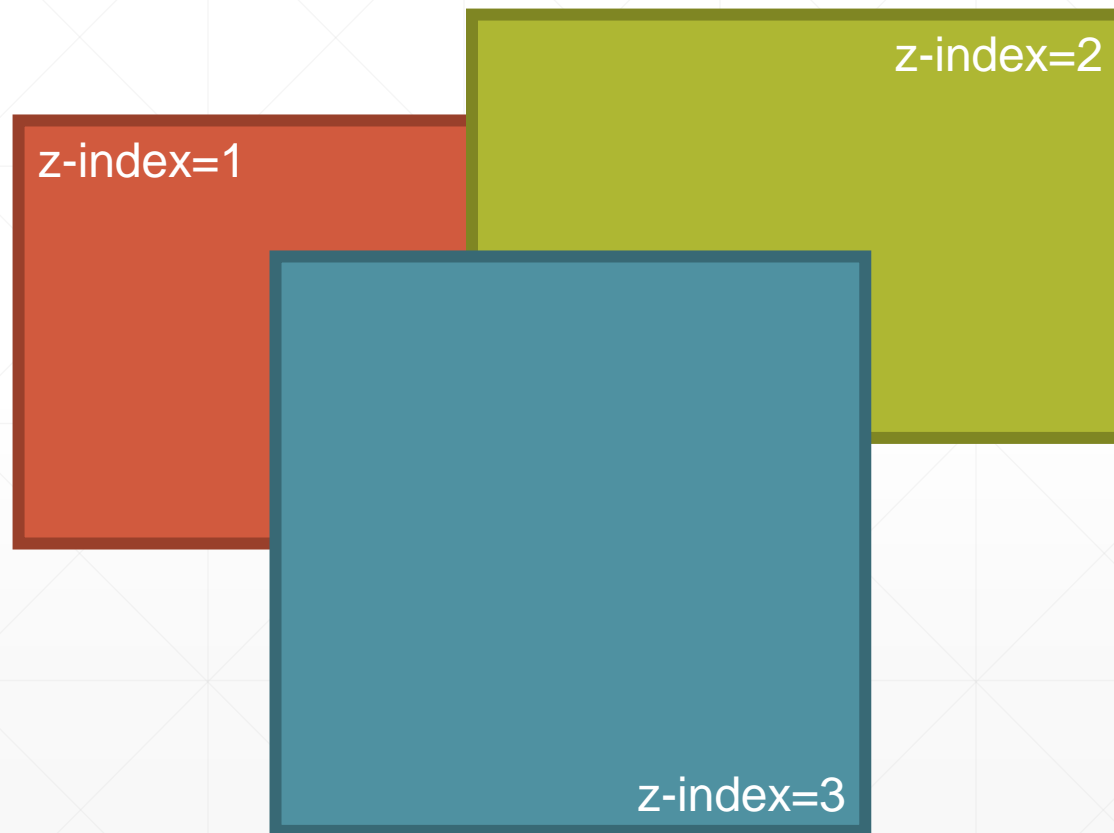
Viewport

Absolute

Rodzic



CSS - Pozycje



Flexbox

Flexbox

.css

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

kontener flexbox

class="container"



elementy flexbox

CSS - Pseudoklasy i pseudoelementy

Pseudoklasy

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

Pseudoelementy

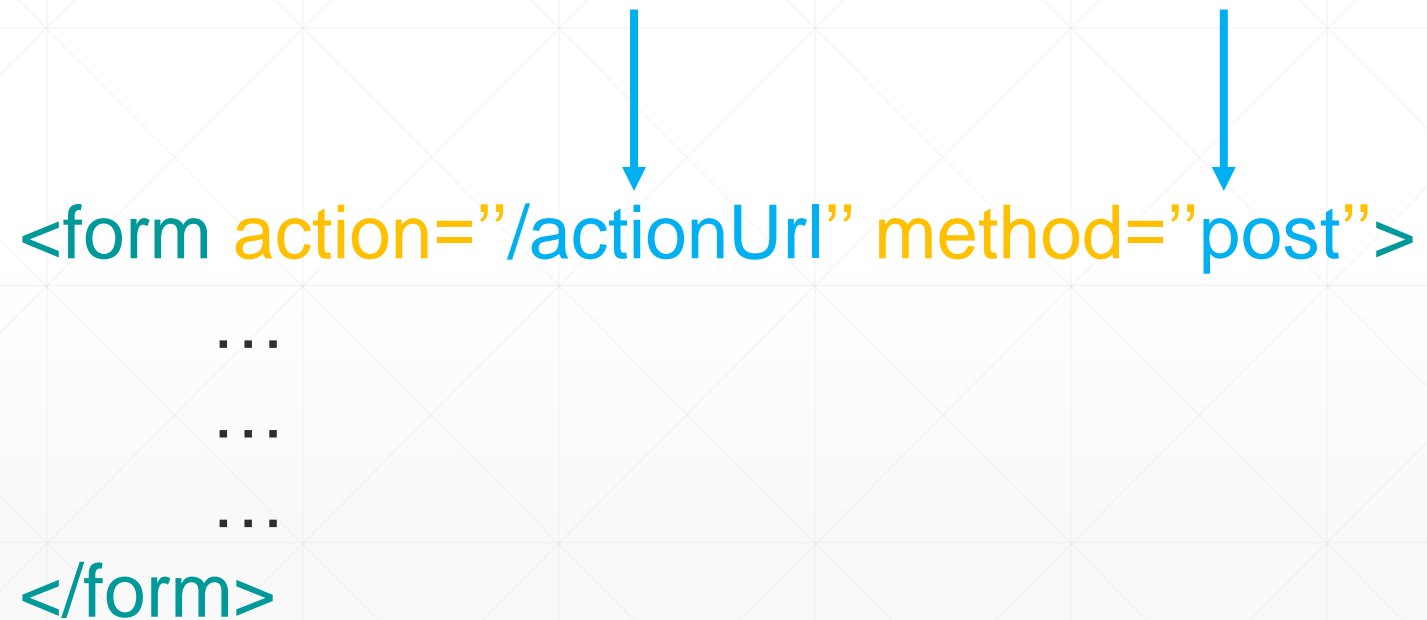
```
selektor::pseudo-element {  
    ...  
}
```

Formularze

Formularze

adres URL gdzie zostanie
wysłany formularz

metoda HTTP



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

Elementy formularzy

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

A simple rectangular text input field with a thin gray border.

```
<select>
```

```
  <option value="one">1</option>
```

```
  <option value="two">2</option>
```

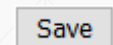
```
</select>
```

A small rectangular dropdown menu with a thin gray border, containing the number '1' and a downward-pointing arrow.

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

A rectangular text area with a thin gray border and a small 'x' icon in the bottom right corner.

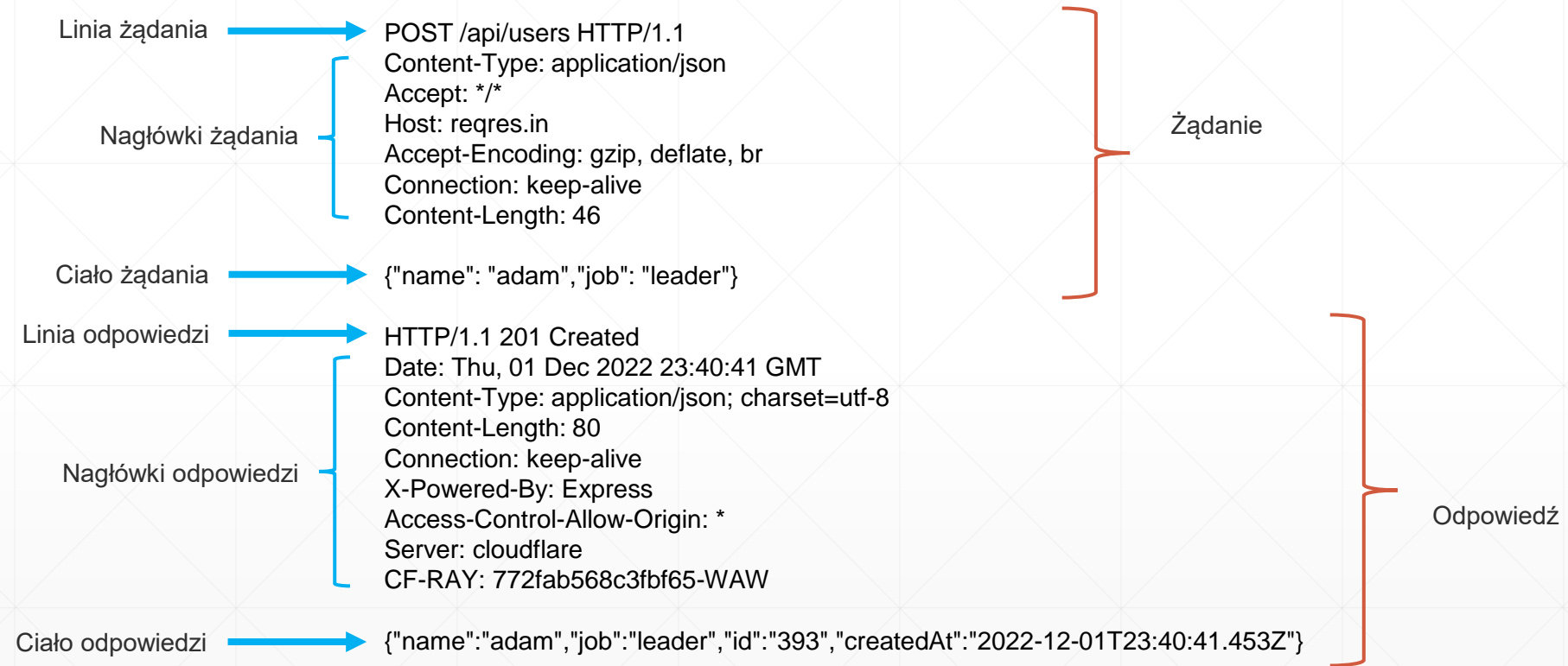
```
<button>Save</button>
```

A rectangular button with a thin gray border and the text 'Save' inside.

HTTP Get



HTTP Post



JavaScript

HTML – warstwa zawartości, szkielet strony internetowej

CSS – warstwa prezentacji strony internetowej

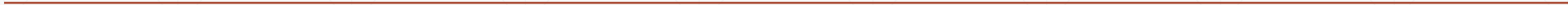
JavaScript – warstwa zachowania strony internetowej

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 w deklaracji znacznika



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

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

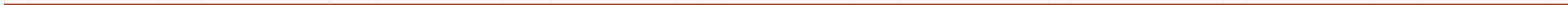
Deklaracja zmiennej

słowo kluczowe
zmiennej

średnik kończy
linię

let nazwaZmiennej;

nazwa zmiennej



Przypisanie zmiennej

znak przypisania



zmienna liczbowa

```
let nazwaZmiennej = -5.0;
```

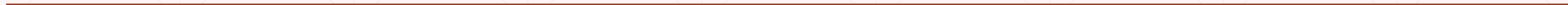
zmienna tekstowa

```
let nazwaZmiennej = 'Tekst';
```

zmienna boolowska

```
let nazwaZmiennej = true;
```

Typy proste



Tablice - deklaracja

tablica liczb

```
let nazwaZmiennej = [1, 2, 3];
```

tablica tekstowa

```
let nazwaZmiennej = ['jeden', 'dwa'];
```

tablice boolowska

```
let nazwaZmiennej = [true, false];
```

Tablice – dostęp do n-go elementu tablicy

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

`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

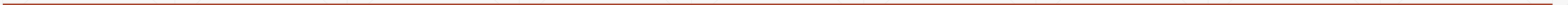
Tworzenie obiektów

- poprzez notacje literałów

```
let 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

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

Tworzenie obiektów

- poprzez klasę

konstruktor

```
class Osoba {
```

```
  constructor(imie, nazwisko) {
```

```
    this.imie = imie;
```

```
    this.nazwisko = nazwisko;
```

```
  }
```

właściwości

metoda

```
  this.idz = function() {
```

```
  }
```

```
}
```

wywołanie konstruktora

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

Obiekty - dostęp do właściwości i metod

Przypisanie do zmiennej wartości
właściwości imie

```
let imie = osoba.imie;
```

Zamiana wartości właściwości imie

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

Wywołanie metody obiektu

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

Operatory

- przypisania

zmienna = 'Przykładowy tekst';

- porównania

==

!=

===

!==

>

<

>=

<=

- arytmetyczne

+

-

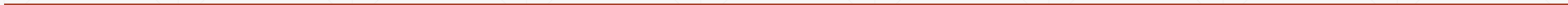
/

*

++

--

%



Operatory

- logiczne

&&

||

!

- ciągów tekstowych

zmiennaTekstowa = 'Jeden, ' + 'dwa';

Decyzje - IF

warunek

↓

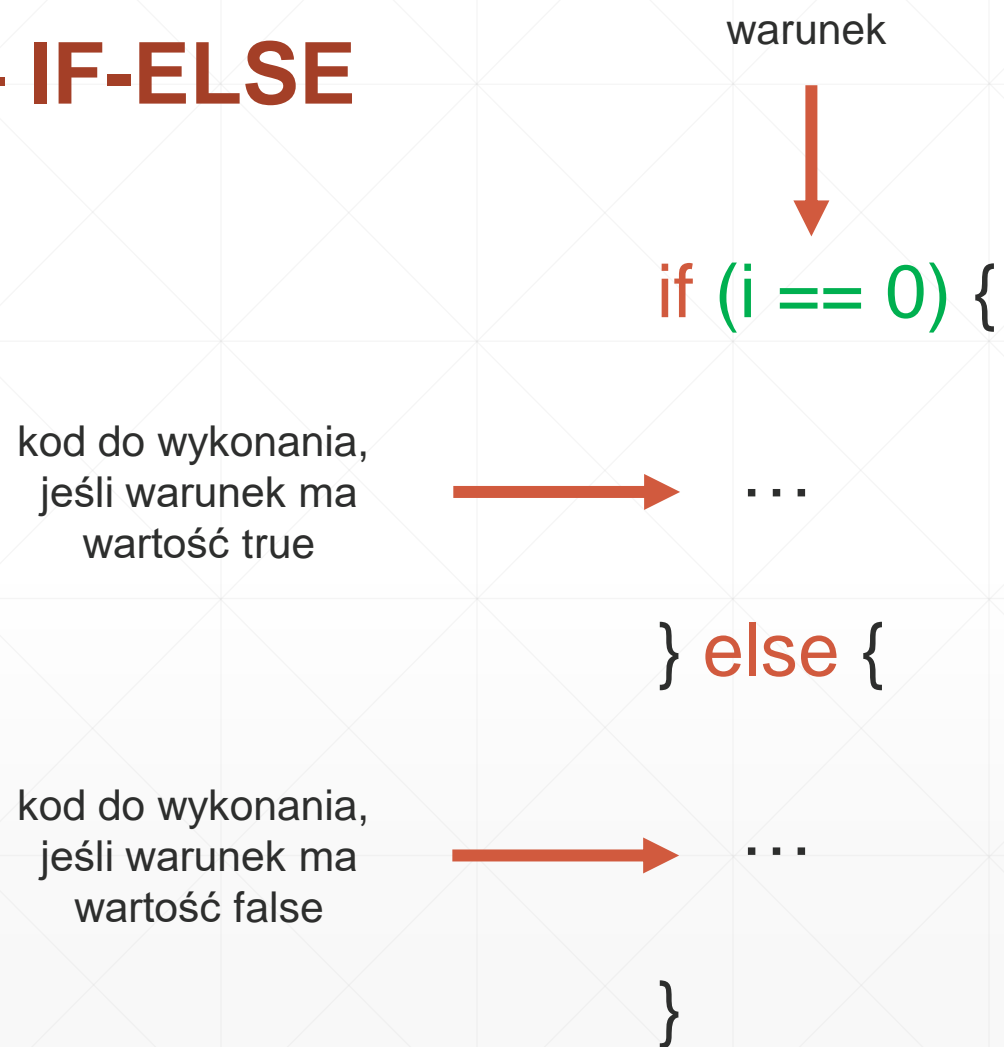
```
if (i == 0) {
```

...

```
}
```

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 – For in / For of

pojedyncza właściwość
obiektu obiekt

↓ ↓

```
for (let prop in person) {
```

...

↑

słowo kluczowe
iteracji po właściwościach

```
}
```

The diagram illustrates the 'for in' loop syntax. It shows the code snippet 'for (let prop in person) { ... }'. Red arrows point from the labels 'pojedyncza właściwość obiektu' (single object property) to 'prop' and from 'obiekt' (object) to 'person'. Another red arrow points from 'słowo kluczowe iteracji po właściwościach' (keyword for iteration over properties) to 'in'.

pojedynczy element kolekcja elementów

↓ ↓

```
for (let element of collection) {
```

...

↑

słowo kluczowe
iteracji po kolekcji

```
}
```

The diagram illustrates the 'for of' loop syntax. It shows the code snippet 'for (let element of collection) { ... }'. Red arrows point from the labels 'pojedynczy element' (single element) to 'element' and from 'kolekcja elementów' (collection of elements) to 'collection'. Another red arrow points from 'słowo kluczowe iteracji po kolekcji' (keyword for iteration over collection) to 'of'.

Pętle - While



Pętle – Do-While

kod wykonany przynajmniej
jeden raz i dopóki warunek
jest spełniony



```
do {
```

```
...
```

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

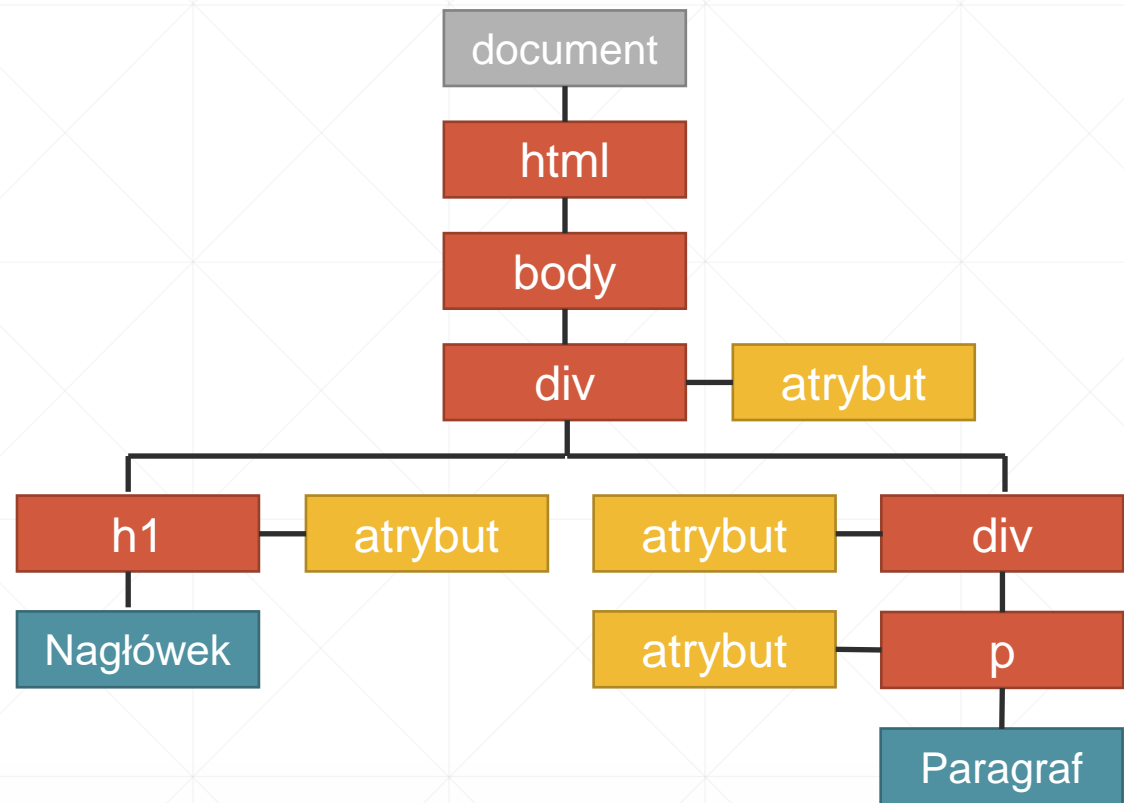


warunek

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 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');  
}
```

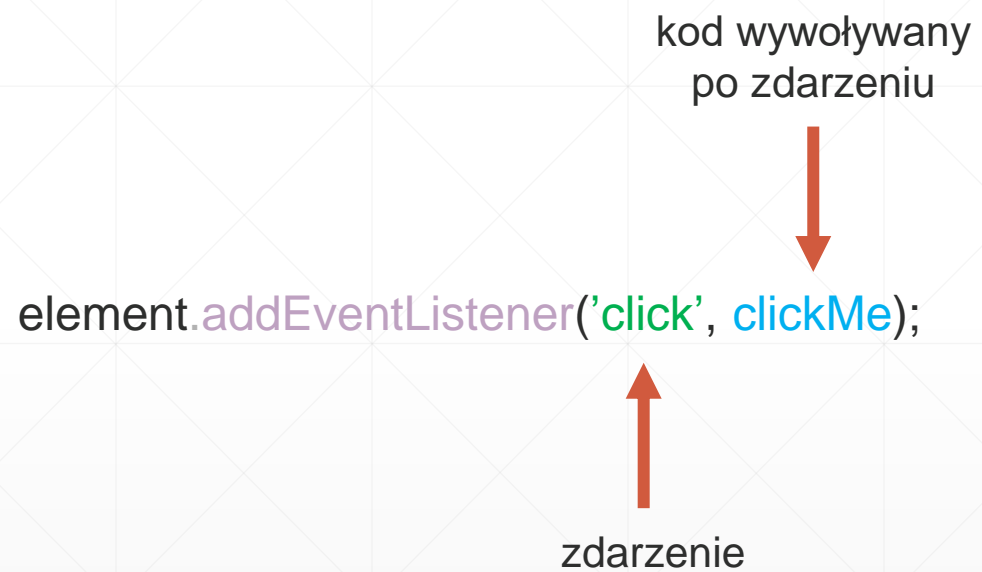
```
let element = document.getElementById('myButton');  
element.onclick = clickMe;
```



zdarzenie

Obsługa zdarzeń

- obserwator zdarzeń



Obsługa zdarzeń

- obserwator zdarzeń

.html

...

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

...

.js

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

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

Walidacja formularzy

Nazwa użytkownika*

Podaj nazwę użytkownika

Email

Password

Miasto

Wyślij zapytanie