



FRONT-END

## Agenda

- ❑ Wprowadzenie do Front-end
- ❑ XML, HTTP
- ❑ HTML, CSS ,Parametry URL, Narzędzia ułatwiające tworzenie stron, Bootstrap
- ❑ JavaScript, DOM, jQuery, pluginy, filtrowanie, Xpath, RSS

## Front-end wprowadzenie

- ❑ Front-end – tym pojęciem definiuje się w technologiach internetowych kod wykonywalny po stronie użytkownika. Czyli w dużym uproszczeniu to co możemy zobaczyć na ekranie.
- ❑ W ogólności do tej kategorii można zaliczyć HTML, CSS oraz JavaScript.
- ❑ W tej części kursu postaramy się nakreślić tą tematykę, ponieważ wchodzi ona w bezpośrednie relacje z programami wykonywalnymi po stronie serwera czyli ogólnie mówiąc Back-endu.

XML

## XML wprowadzenie

- ❑ XML (Extendable Markup Language) to:
  - ❑ **protokół opisu i zarządzania informacją,**
  - ❑ **metajęzyk (służący do opisywania innych języków lub danych).**
- ❑ Celem twórców XMLa było oddzielenie treści dokumentu od formatowania, zwiększenie użyteczności danych przez ich opis w czystej, strukturalnej formie; dlatego też dokument XML nie zawiera żadnych informacji formatujących.
- ❑ Przed prezentacją użytkownikowi końcowemu musi on zostać przetworzony przez zewnętrzny mechanizm. XML nie posiada (w przeciwieństwie do np. HTMLa) predefiniowanego zestawu znaczników, a interpreter XML nie posiada informacji o znaczeniu poszczególnych znaczników.
- ❑ Dokument XML ma budowę drzewiastą (drzewo elementów). Semantyka poszczególnych elementów określona jest poza dokumentem, często w nieformalny sposób. Sposób prezentacji dokumentu wymaga użycia formalnego opisu stylu (np. CSS, XSL).

## Składnia XML

- ❑ Aplikacja XML to każdy język znacznikowy, formalnie oparty o zasady XML i przeznaczony do konkretnych zastosowań (np. HTML, MathML, SVG, MusicML, DocBook, WML, itd.).
- ❑ Dokument dobrze uformowany (well-formed) to dokument XML spełniający następujące reguły:
  - ❑ Musi mieć nagłówek określający, że to jest dokument XML  
np. **`<?xml version="1.0">`**
  - ❑ Każdy znacznik otwierający musi mieć swój odpowiednik zamykający
  - ❑ Elementy puste muszą być jawnie puste **`<separator/>`**
  - ❑ Istnieje tylko jeden znacznik główny, który musi zawierać całkowicie wszystkie inne
  - ❑ Zakresy elementów nie mogą się przeplatać
  - ❑ Wartości atrybutów umieszczone są w cudzysłowach lub prawych apostrofach  
— Znaki `<` i `&` używane są tylko do otwierania znaczników i odwołania do encji

## Drzewo dokumentu XML

- ❑ Drzewo dokumentu XML tworzą jego elementy. Zaczyna się ono od korzenia i gałęzi oraz kończy się na liściach które są najniższym poziomem drzewa.

```
<root>
```

```
  <child>
```

```
    <subchild>.....</subchild>
```

```
  </child>
```

```
</root>
```

- ❑ Rodzic(parent), dziecko(child) i rodzeństwo(sibling) są to wyrażenia używane do opisanie relacji między elementami drzewa. Rodzeństwem nazywa się dzieci na tym samym poziomie (bracia i siostry).
- ❑ Wszystkie elementy mogą mieć zawartość tekstową i atrybuty (podobnie jak w HTML).

## XML przykład

- ☐ Postacie z bajek:
  - ☐ Jelonek – Bambi
  - ☐ Lew – Simba
  - ☐ Surykatka – Timon
  - ☐ Pies - Reksio



## Przykład składni XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Postacie> - element główny
```

```
    <Jelonek>Bambi</Jelonek> - element podrzędny
```

```
    <Lew>Simba</Lew>
```

```
    <Surykatka>Timon</Surykatka>
```

```
    <Pies>Reksio</Pies>
```

```
</Postacie>
```

## XML Dokument

Dlaczego nie wygląda tak:

Postacie z bajek:

Jelonek – Bambi

Lew – Simba

Surykatka – Timon

Pies - Reksio

## Nazwy elementów XML

- ☐ Mogą zawierać litery, cyfry i inne znaki
- ☐ Nie mogą zaczynać się od cyfry lub znaków interpunkcyjnych
- ☐ Nie mogą zaczynać się od XML obojętnie jak pisanego
- ☐ Nie mogą zawierać spacji

- ❑ Dobrą praktyką w każdym języku programowania jest wpisywanie komentarzy.
- ❑ W XML komentarze są zawierane w znaczniku `<!-- komentarz -->`

## Poziomy zagnieżdżenia

- ❑ Żeby móc wyszukiwać poprawnie informacje w dokumencie zazwyczaj potrzeba więcej niż jeden poziom zagnieżdżenia.
- ❑ Aby tego dokonać wystarczy dodać dodatkowy element który będzie spinał to co chcielibyśmy rozdzielić.

## Uniwersalizacja

- ☐ Czy to co stworzyliśmy jest uniwersalne?
- ☐ Czy w zastosowaniu komercyjnym to ma sens?
- ☐ Puste elementy:
  - ☐ **<nazwa\_elementu></nazwa\_elementu>**
  - ☐ **<nazwa\_elementu/>**
- ☐ Czy da się to tak zmodyfikować aby było bardziej uniwersalne?

## Atrybuty

- ❑ Czasami zamiast wprowadzania nowych znaczników do elementu można stworzyć atrybuty. Dodawanie atrybutów polega na dodaniu nazwy atrybutu i jego wartości występującej po znaku równości w cudzysłowie wewnątrz elementu otwierającego.

```
<czlowiek relacja="matka"></czlowiek>
```

- ❑ Należy pamiętać, że atrybuty są przyjemniejsze do odczytu z punktu widzenia człowieka, niestety dla maszyny i osoby która je obsługuje programowo jest to pewnego rodzaju utrudnienie.

## Elementy vs Atrybuty

- ☐ Atrybuty mogą mieć przypisaną tylko jedną wartość, elementy mogą przyjmować wiele.
- ☐ Atrybuty nie są łatwo rozszerzalne w przypadku zmian w przyszłości
- ☐ Atrybuty nie opisują struktury elementy tak
- ☐ Atrybuty są znacznie trudniejsze przy manipulowaniu nimi w programie



## Znaki Specjalne w XML

- ❑ Ze względu na to że pewne znaki są zarezerwowane dla składni języka XML należy stosować zamiennie dla nich odpowiednie encje czyli spójny ciąg znaków,
- ❑ Poniższa tabela pokazuje reprezentacje znaków specjalnych w XML:

Encje	Opis	Wizualizacja
<code>&amp;quot;</code>	cudzysłów prosty - Quot	"
<code>&amp;amp;</code>	znak and - Ampersand	&
<code>&amp;lt;</code>	znak mniejszości	<
<code>&amp;gt;</code>	znak większości	>
<code>&amp;apos;</code>	apostrof	'

## Instrukcje przetwarzania

- ❑ Zawierają dodatkowe informacje skierowane do programów odczytujących dany dokument. Składają się z nazwy (adresat instrukcji, target) oraz treści.

```
<?javascript alert("To jest przykład instrukcji przetwarzania."); ?>
```

- ❑ Nie ma ograniczeń co do składni instrukcji przetwarzania, można stosować własne instrukcje, które mogą być obsługiwane za pomocą oprogramowania napisanego przez nas. Przykładem dedykowanej instrukcji:

```
<?wyszukaj nazwa="PWN" jezyk="pl"?>
```



HTTP

## HTTP

- ❑ Hyper Text Transport Protocol – jest to protokół który wprowadza pojęcie zasobu czyli sieciowego odpowiednika pliku. np. zawierający stronę HTML, obrazek, skrypt lub cokolwiek innego. Wiele zasobów rzeczywiście reprezentuje konkretne pliki.
- ❑ Serwery i klienci WWW posługują się tym protokołem w zakresie zdalnej manipulacji zasobami co w praktyce przekłada się w przypadku przeglądarek na korzystaniu z metod pozwalających na ściąganie określonego zasobu.

## HTTP komunikacja

- ❑ Standardowo zasoby przesyłane są jako strumień bajtów. Do tego celu protokół HTTP wykorzystuje tak zwane nagłówki. Są one wysyłane z konkretnym zasobem i oddzielone od niego pustą linią.
- ❑ Kontakt jest inicjowany zawsze przez klienta, wysyła on na umówiony port maszyny docelowej (zazwyczaj 80) zapytanie o zasób.

## Zapytania HTTP

- ☐ Zapytanie składa się z:
  - ☐ Nazwy czynności, którą chcemy wykonać na zasobie
  - ☐ Części nazwy lokalnej względem serwera
  - ☐ Nazwy protokołu z którego chcemy korzystać
  - ☐ Znaku nowej linii
  - ☐ Zestawu nagłówków HTTP których koniec wyznacza pusta linijka
- ☐ Komunikat odpowiedzi serwera składa się:
  - ☐ W pierwszej kolejności z informacji o statusie połączenia
  - ☐ Następnie występują nagłówki HTTP których koniec wyznacza pusta linijka
  - ☐ Właściwe dane

## Nagłówki HTTP

- ❑ Są to sekwencje linii tekstu, z których każda definiuje jeden umówiony klucz.
- ❑ Pojedyncza linijka składa się:
  - ❑ W pierwszej kolejności zaczyna się od nazwy klucza i dwukropka
  - ❑ Następnie występuje lista wartości oddzielona przecinkami
  - ❑ Mogą występować atrybuty oddzielone średnikami

## Niektóre typy nagłówków

- ❑ Host: - informacja dla serwera o jego nazwie wpisanej w pasku adresu
- ❑ Content-type: - informacja na temat rodzaju zasobu według oficjalnej klasyfikacji MIME
- ❑ Nagłówki negocjacyjne(wysyłane przez przeglądarkę):
  - ❑ Accept: (lista typów mime),
  - ❑ Accept-Language: (określenia języków według norm ISO),
  - ❑ Accept-Charset: (określenia kodowań znaków według norm ISO)



## Przykład nagłówka HTTP

```
❑ HTTP/1.1 200 OK Server: aris Expires: Mon, 26 Jul 1997 05:00:00  
GMT Last-Modified: Wed, 26 Apr 2006 14:05:56 GMT Cache-Control:  
no-store, no-cache, must-revalidate Cache-Control: post-  
check=0, pre-check=0 Cache-Control: no-cache Pragma: no-cache  
Content-type: text/html; charset=ISO-8859-2 Set-Cookie:  
reksticket=1146060356; expires=Thu, 27-Apr-06 14:05:56 GMT;  
path=/; domain=.www.wp.pl
```

## Metody HTTP

- ❑ **GET** – najpopularniejsza z metod. Zwykle wpisanie linka w oknie adresu korzysta właśnie z niej. Parametry mogą być wysłane jako część URL-a.
- ❑ **HEAD** – Prośba o sam nagłówek
- ❑ **POST** – podobna do GET, z tym że razem z zapytaniem klient wysyła w nagłówku dodatkowe dane (w formie słownika, jak w GET, w nagłówku POST mogą być także całe pliki). Zapytanie z metodą POST mogą być generowane przez formularze.
- ❑ **PUT** – teoretycznie nagrywa plik na serwerze.
- ❑ **DELETE** - teoretycznie usuwa plik z serwera.
- ❑ **TRACE** – odsyła zapytanie bez zmian (służy do debugowania połączenia).
- ❑ **OPTIONS** – informuje o metodach zaimplementowanych w serwerze.

## Niektóre statusy HTTP

- ❑ 2.. – Sukces ○ 200 OK ○ 204 No content
- ❑ 3.. – Przekierowanie ○ 300 Multiple Choices ○ 301 Moved Permanently ○ 302 Found ○ 305 Use Proxy ○ 307 Temporary Redirect
- ❑ 4.. Błąd klienta ○ 400 Bad Request ○ 401 Unauthorized ○ 402 Payment Required ○ 403 Forbidden ○ 405 Method Not Allowed ○ 408 Request Timeout ○ 415 Unsupported Media Type
- ❑ 5.. Błąd serwera ○ 500 Internal Server Error ○ 501 Not Implemented ○ 505 HTTP Version Not Supported



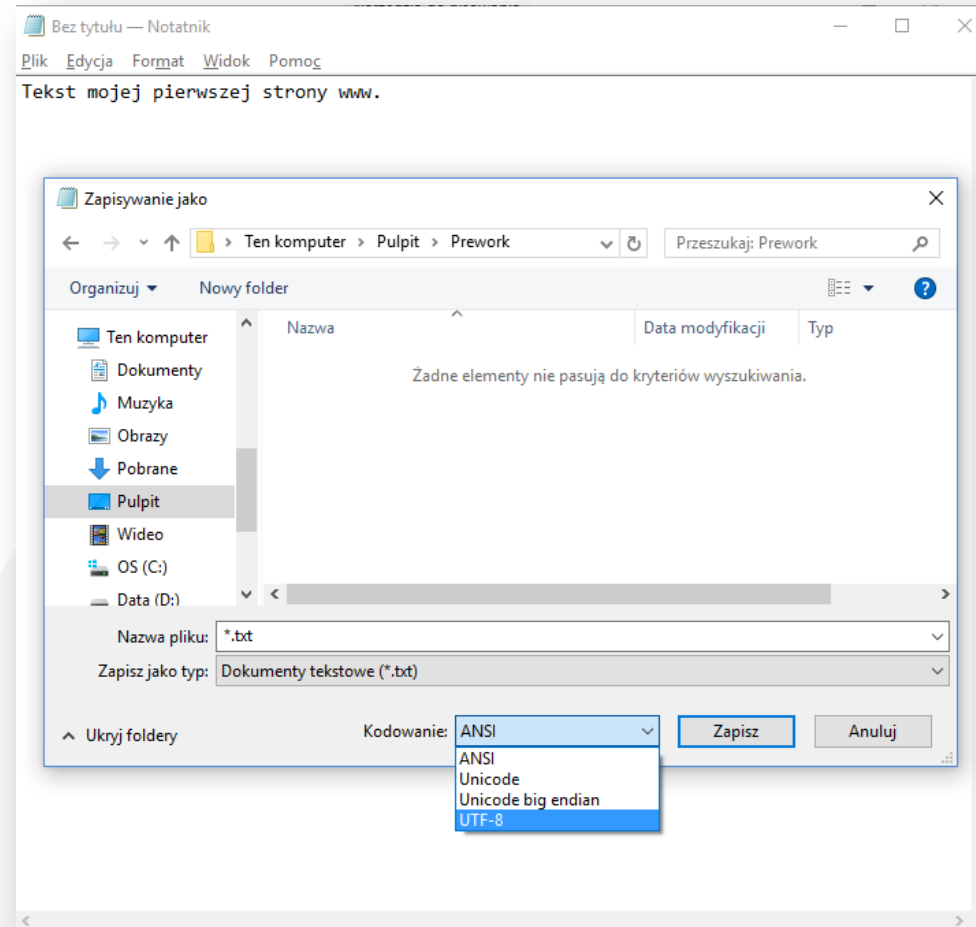
# Wprowadzenie do HTML

- ❑ HTML (ang. Hyper Text Markup Language) to najbardziej popularna technologia w sieci, absolutnie konieczna do projektowania stron internetowych.
  - ❑ Hyper Text – dokumenty są interpretowane w ramach protokołu HTTP
  - ❑ Markup – dokumenty zawierają znaczniki które określają treści
  - ❑ Language – dokument jest interpretowany przez komputer
- ❑ W tym kursie będziemy omawiali zagadnienia związane z wersją najnowszą, czyli HTML5.

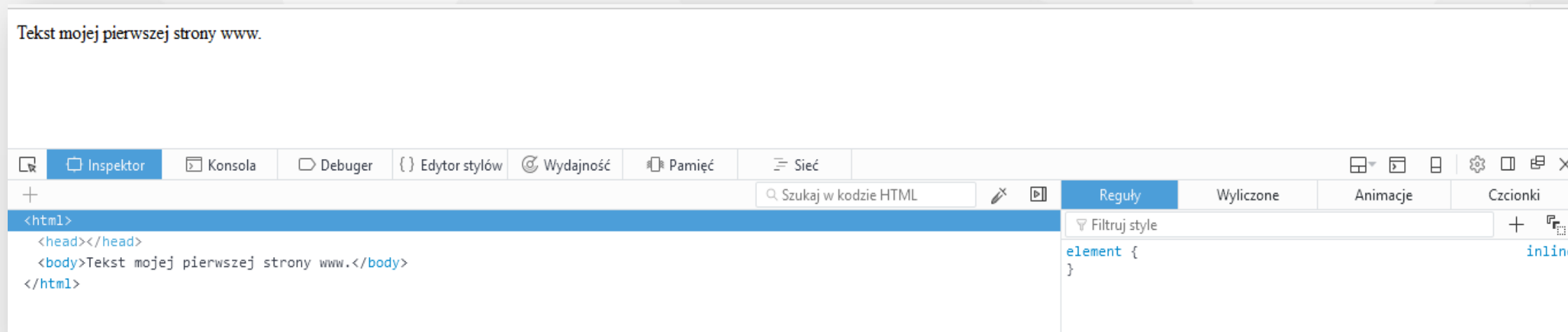
## Jak zacząć przygodę z HTML

- ❑ Otwieramy dowolny edytor tekstu np.: Notatnik dostępny w systemach Windows bez konieczności instalacji.
- ❑ Wpisujemy do niego jakąś frazę np.: Tekst mojej pierwszej strony www i zapisujemy go jako plik z rozszerzeniem .html (np.: Stronka.html). Warto zmienić kodowanie na UTF-8!
- ❑ Takie rozszerzenie jest interpretowane przez przeglądarkę internetową i w ten sposób możesz otworzyć swoją pierwszą stronę internetową.

# Jak zacząć przygodę z HTML

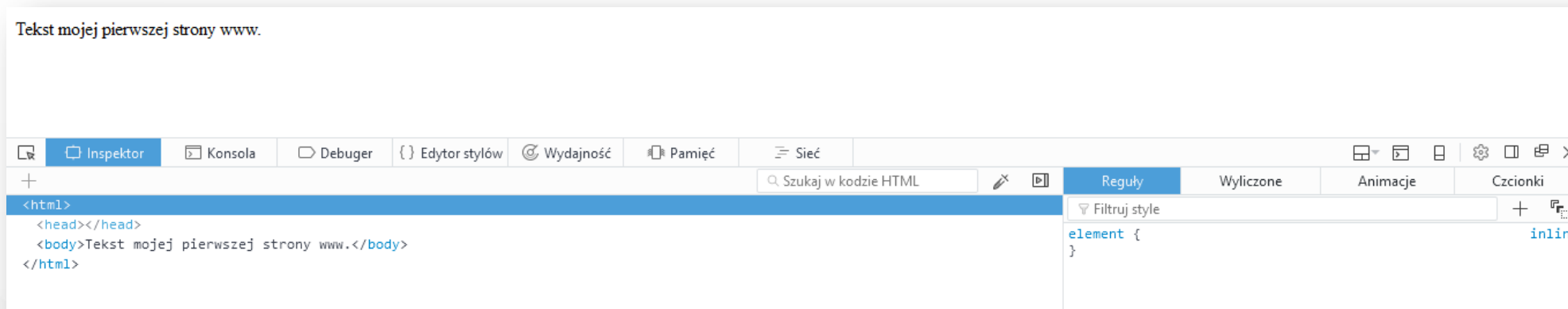


- ❑ W przeglądarce internetowej zawsze możemy zbadać źródło wyświetlanej strony internetowej klikając prawym klawiszem myszy na jej treści i wybierając polecenie: *Zbadaj*.





## Znaczniki pierwszej strony



- ❑ Widzimy, że oprócz napisu widocznego w przeglądarce pojawiły się dodatkowe elementy – znaczniki HTML.
- ❑ Przeglądarka wygenerowała domyślnie znaczniki do tekstu zawartego w pliku Stronka.html, ale oczywiście nie powinniśmy tak robić!
- ❑ Powinniśmy skorzystać ze struktury HTML po to aby wyświetlić nasz kawałek tekstu.

## Znacznik paragrafu

- ❑ Modyfikujemy tekst w pliku: Stronka.html opakowując go w znacznik `<p>` - paragraf (ang. paragraph) i `</p>` - zakończenie paragrafu.

`<p>Tekst mojej pierwszej strony www</p>`

- ❑ Zapisujemy plik otwieramy w przeglądarce i patrzymy co się stało oraz co ważniejsze badamy źródło – jak wcześniej.

# Znaczniki nagłówków

- ❑ Tym razem dodajmy sobie jakieś nagłówki do naszej Stronki.html opakowując je znacznikami typu `<h>` - nagłówek (ang. header):

```
<h1> Nagłówek typu 1 </h1>
```

```
<h2> Nagłówek typu 2 </h2>
```

```
<p>Tekst mojej pierwszej strony www</p>
```

**Nagłówek typu 1**

**Nagłówek typu 2**

Tekst mojej pierwszej strony www

- ❑ W efekcie zauważamy, że każdy z nagłówków posiada inne style!? Dlatego że nasza przeglądarka posiada wbudowane style dla tak zdefiniowanych znaczników.

**HTML**



Edytor HTML

## Wybór edytora

- ☐ Istnieje wiele różnych – mniej lub bardziej zaawansowanych edytorów HTML takich jak: Notepad ++, Sublime Text, PSPad, Aptana Studio, Brackets, czy niegdyś popularny Pajączek.
- ☐ Na tym kursie będziemy korzystać z *Notepad ++*. Dlaczego?
  - ☐ Prostota i wygodna.
  - ☐ Przejrzysty interfejs, szybkie uruchamianie i stabilna praca ułatwią tworzenie kodu źródłowego.
  - ☐ Podświetlanie składni wielu języków programowania.
  - ☐ Pozwala na wyszukiwanie i zmianę ciągów znaków za pomocą wyrażeń regularnych, tworzenie makr i własnych wtyczek.
  - ☐ Darmowy edytor oparty na licencji GNU GPL.
- ☐ Instalacja edytora jest dokładnie opisana w prezentacji dotyczącej oprogramowania potrzebnego na kursie Back-end Developer.

**HTML**



# Składnia HTML

## Dokument HTML

- ❑ HTML jest językiem służącym do opisu stron WWW opartym na XML-u.
- ❑ Inaczej mówiąc jest to taki XML, w którym sekcji nie możemy nazywać w dowolny sposób. Mamy z góry zadane nazwy sekcji, z których możemy korzystać.
- ❑ Każda sekcja ma swoje z góry zadane znaczenie.

## Składnia dokumentu HTML

☐ Zachowanie odpowiedniej składni pozwala na jego:

- ☐ Walidację
- ☐ Debugowanie
- ☐ Czytelność
- ☐ Utrzymanie

**!doctype**

**html**

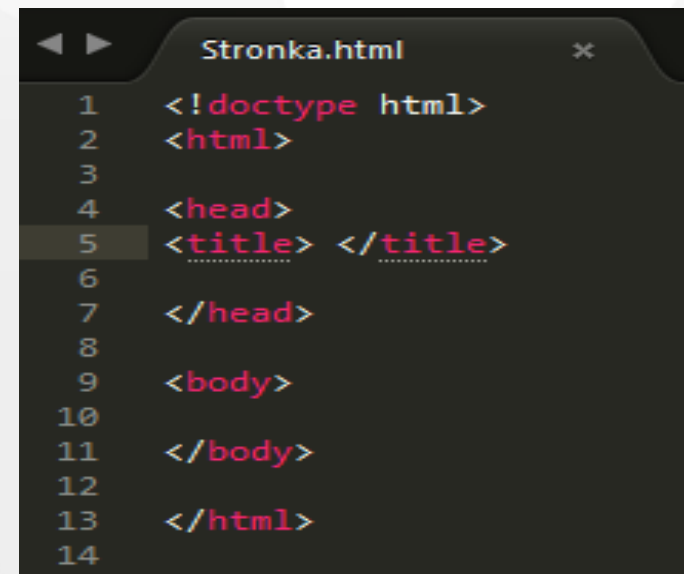
**head**

**body**



## Podstawowa składnia dokumentu

- ❑ Każdy poprawny dokument HTML powinien zawierać następujące sekcje (określone znacznikami):
  - ❑ `<!doctype html>` informuje, że dokument ma być interpretowany w języku HTML
  - ❑ `<html>` cały dokument HTML
  - ❑ `<head>` sekcja nagłówka
  - ❑ `<title>` tytuł
  - ❑ `<body>` treść dokumentu



```
Stronka.html
1  <!doctype html>
2  <html>
3
4  <head>
5  <title> </title>
6
7  </head>
8
9  <body>
10
11 </body>
12
13 </html>
14
```

## Sekcja `<head>`

- ❑ W sekcji meta występują znaczniki:
  - ❑ `<title>` - dodanie tytułu strony
  - ❑ `<meta>` - dodanie atrybutów strony
  - ❑ `<style>` - dodanie stylów bezpośrednio do HTML
  - ❑ `<link>` - dodanie stylów z zewnętrznego pliku `.CSS`

## Dodanie tytułu strony

- ❑ Za dodanie tytułu strony widocznego w zakładce przeglądarki jest odpowiedzialny znacznik title:

```
<title> NazwaStrony </title>
```

- ❑ Dobrą praktyką jest stosowanie tytułu strony zgodnego z nazwą zakładki w której aktualnie się poruszamy

## Kodowanie znaków w HTML

- ❑ Do kodowania znaków w HTML wykorzystywane jest polecenie:

```
<meta charset ="typ kodowania">
```

- ❑ W naszym przypadku wybieramy kodowanie utf-8 i znacznik ten dodajemy w pierwszej linii sekcji nagłówka (**<head>**) – wyłącznie tam on może występować:

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

## Nazwa autora strony

- Za pomocą znacznika meta możemy dodać nazwę autora strony

```
<meta name= "author" content= "nazwa autora">
```

## Słowa kluczowe strony

- ❑ Niegdyś bardzo ważne w celu pozycjonowania wyników wyszukiwania strony przez przeglądarki.

```
<meta name="keywords" content="słowa kluczowe">
```

- ❑ Aktualnie słowa kluczowe straciły na wartości.

## Opis strony

- Za pomocą znacznika meta możemy dodać opis strony, który pojawi się w przeglądarce podczas wyszukiwania (max. ok. 150znaków)

```
<meta name="description" content="opis strony">
```

Home - Reaktor - Wydawnictwo Naukowe PWN

reaktor.pwn.pl/ ▼

**Content**

Kurs w Reaktor PWN to nie tylko bardzo intensywne warsztaty pod okiem trenera. ... Dwie osoby z każdego kursu, które przygotowują najciekawsze projekty, ...

## Elementy blokowe

- ❑ Elementy blokowe to najważniejsze elementy struktury dokumentu na stronie.

Określają one ramy dokumentu:

- |                     |   |
|---------------------|---|
| ❑ Paragrafy/akapity | <code>&lt;p&gt;</code>                              |
| ❑ Listy             | <code>&lt;ul&gt;</code> i <code>&lt;ol&gt;</code>   |
| ❑ Nagłówki          | <code>&lt;h1&gt;</code> ... <code>&lt;h6&gt;</code> |
| ❑ Sekcje            | <code>&lt;section&gt;</code>                        |
| ❑ Artykuły          | <code>&lt;article&gt;</code>                        |



## Elementy liniowe

- ❑ Elementy liniowe to najczęściej elementy znajdujące się wewnątrz elementów blokowych zawierające jakiś fragment tekstu, któremu nadają jakieś konkretne znaczenie.

- ❑ Elementy
- ❑ Linki
- ❑ Ważne treści
- ❑ Pochylenie
- ❑ Pogrubienie

**<span>**

**<a>**

**<em>**

**<i>**

**<b>**

## Domykanie elementów

	Znacznik otwierający i zamykający	Samozamykanie
Elementy blokowe	<code>&lt;p&gt; &lt;/p&gt;</code> <code>&lt;ul&gt; &lt;/ul&gt;</code> <code>&lt;ol&gt; &lt;/ol&gt;</code>	Nieemożliwe
Elementy liniowe	<code>&lt;a&gt; &lt;/a&gt;</code> <code>&lt;strong&gt; &lt;/strong&gt;</code> <code>&lt;em&gt; &lt;em&gt;</code>	<code>&lt;input&gt;</code> <code>&lt;br&gt;</code> <code>&lt;img&gt;</code>

**HTML**



Treści na stronie

## Najczęściej używanie znaczniki

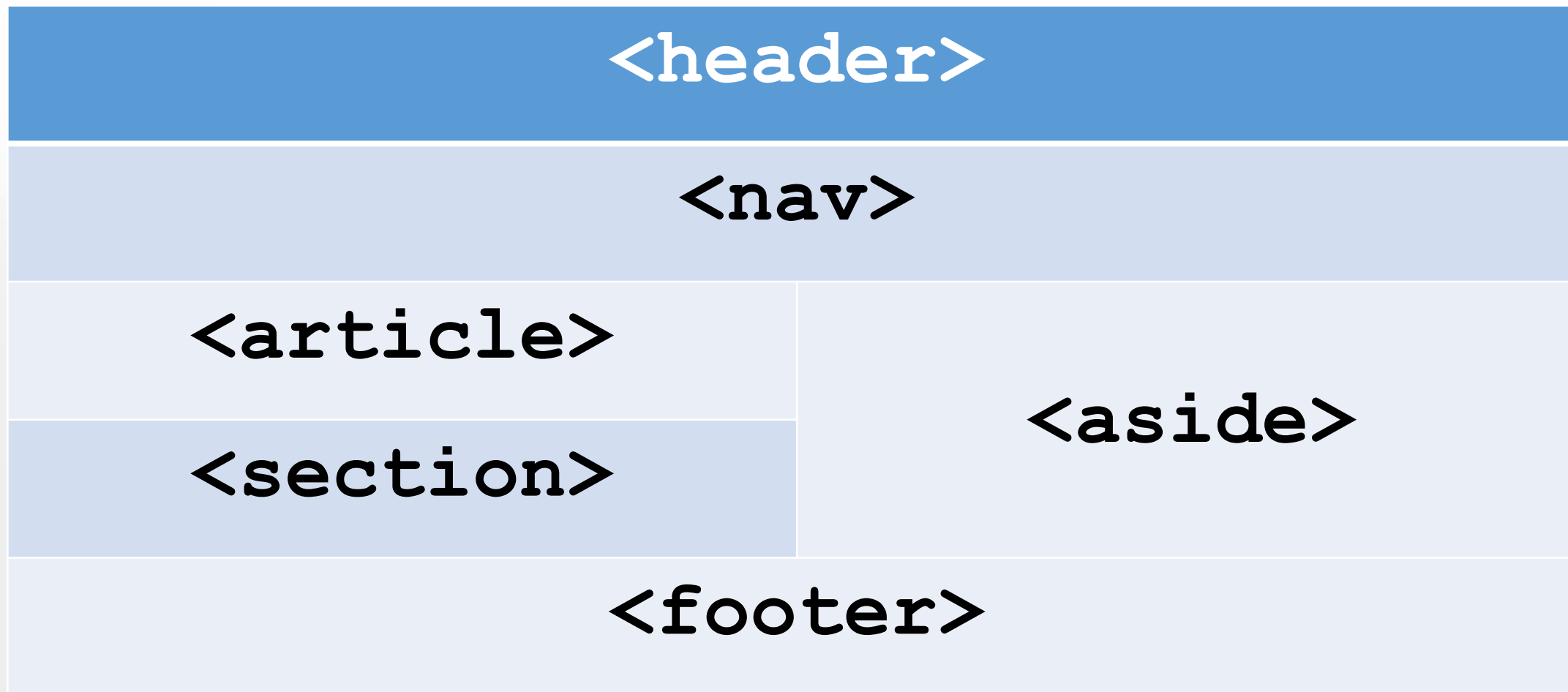
- ❑ Dokładniejszy opis znaczników HTML znajduje się pod adresem:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

- ❑ Poniższa tabela zawiera najczęściej wykorzystywane znaczniki HTML:

Struktura	Treść	Liniowe
<b>header</b> – nagłówek strony <b>h1</b> – nagłówek pierwszego poziomu <b>h2</b> – nagłówek drugiego poziomu <b>div</b> – warstwa strony <b>nav</b> – nawigacja strony <b>footer</b> – stopka strony <b>article</b> – artykuł strony <b>section</b> – sekcja strony	<b>p</b> <b>ul</b> <b>ol</b> <b>li</b> <b>blockquote</b>	<b>a</b> <b>strong</b> <b>em</b> <b>q</b> <b>abbr</b> <b>span</b>

## Semantyka znaczników struktury



## Znaczniki treści

- ☐ Otwórz plik o nazwie znaczniki treści.
- ☐ Występują tam następujące typy znaczników:
  - ☐ `<p>` akapit
  - ☐ `<ul>` lista nieuporządkowana
  - ☐ `<ol>` lista nieuporządkowana
  - ☐ `<i>` element listy
  - ☐ `<blockquote>` cytat

## Znaczniki liniowe

- ☐ Otwórz plik o nazwie znaczniki liniowe.
- ☐ Występują tam następujące typy znaczników:
  - ☐ **<a>**                      hiperłącze
  - ☐ **<strong>**                pogrubienie
  - ☐ **<em>**                     wyróżnienie semantyczne
  - ☐ **<span>**                  wyróżnienie stylistyczne
  - ☐ **<abbr>**                 zaznaczenie skróconych form wyrazów
  - ☐ **<q>**                      cytaty w cudzysłowie

## Znaki specjalne

- ❑ Aby wypisać znaki specjalne, posługujemy się encjami, czyli elementami zastępczymi, które często są skrótami od angielskiej nazwy znaku.
- ❑ W HTML-u, tak jak w XML-u, użycie niektórych znaków w zwykłym tekście jest zabronione. Chodzi oczywiście o znaki < oraz >, które służą nam do otwarcia i zamknięcia tagu.
- ❑ Znakiem specjalnym jest również &, gdyż został wybrany jako znak specjalny do przedstawienia innych znaków (encji).
- ❑ W niektórych przypadkach, np. w wartościach atrybutów, zabronione są także cudzysłowy lub apostrofy, w zależności od tego, których z nich używamy do oznaczenia początku i końca wartości danego atrybutu.



## Tabela encji

- Poniższa tabelka z encjami pokazuje kilka często stosowanych znaków. Pełna lista znaków jest znacznie dłuższa i można ją odszukać w Internecie pod hasłem „encje HTML”

Znak	Encja	Znak	Encja	Znak	Encja	Znak	Encja
&	&amp;	<sup>1</sup>	&sup1;	€	&euro;	γ	&gamma;
<	&lt;	<sup>2</sup>	&sup2;	£	&pound;	δ	&delta;
>	&gt;	<sup>3</sup>	&sup3;	§	&sect;	ε	&epsi;
©	&copy;	°	&deg;	¼	&frac14;	μ	&mu;
®	&reg;	α	&alpha;	½	&frac12;	π	&pi;
™	&trade;	β	&beta;	¾	&frac34;	Ω	&Omega;

## Polskie znaki

- ❑ Żeby prawidłowo były wyświetlane polskie znaki w przeglądarce, należy ustawić kodowanie.
- ❑ W praktyce robi się to przez element **<meta>** w gałęzi **<head>**, który potrafi narzucić przeglądarce zachowanie analogiczne jak nagłówek HTTP. Czyli pod tagiem **<head>**:
  - ❑ `<meta http-equiv="Content-type" content="text/html; charset=windows-1250"/>`
  - ❑ `<meta http-equiv="Content-type" content="text/html; charset=iso-8859-2"/>`
  - ❑ `<meta http-equiv="Content-type" content="text/html; charset=utf-8"/>`
- ❑ w zależności od użytego kodowania polskich znaków w pliku.

**HTML**



**Obrazy na stronie**

## Ścieżki bezwzględne

- Aby jakiś obraz znajdujący się na już istniejącej stronie w sieci załączyć do naszej strony należy zastosować ścieżkę bezwzględną tego obrazu w znaczniku img:

```
<img src ="adres bezwzględny obrazu">
```

Np.:

```
<img src =". ">
```

## Ścieżki względne

- ❑ Aby załączyć obraz znajdujący się w tym samym miejscu na dysku co nasz plik .html:

```
<img src ="nazwa_obrazu">
```

- ❑ Obraz jednak może się znajdować w odrębnym katalogu:

```
<img src ="nazwa_katalogu/nazwa_obrazu">
```

- ❑ Obraz również może się znajdować w katalogu wyżej w hierarchii katalogów (przed plikiem .html):

```
<img src ="nazwa_katalogu/nazwa_obrazu">
```

## Podstawowe atrybuty obrazów

- ❑ Obraz poza atrybutem źródła (src) może zawierać również inne atrybuty:
  - ❑ alt - opis obrazu
  - ❑ width - szerokość obrazu (w px lub % szerokości strony)
  - ❑ height - wysokość obrazu (w px lub % wysokości strony)
  - ❑ align - ustawienie obrazka (right – do prawej, left – do lewej, a tekst opływa obrazek)
- ❑ Gdy zdjęcie pełni jakąś ważną funkcję objaśniającą na stronie warto opakować je w znacznik **<figure>**.

## Hiperłącza wychodzące poza domenę

- ❑ Hiperłącza wychodzące:

```
<a href="adres bezwzględny linku">
```

- ❑ Hiperłącza wychodzące z wymuszeniem otwarcia nowej zakładki:

```
<a href="adres bezwzględny linku" target="blank">
```

## Hiperłącza w obrębie domeny

- ❑ Hiperłącze do identyfikatora elementu elementu strony:

```
<a href="#id_elementu">
```

- ❑ Hiperłącze do pliku znajdującego się o w miejscu pliku na którym pracujemy:

```
<a href="nazwa_pliku">
```

- ❑ Hiperłącze do pliku w katalogu znajdującym się w miejscu pliku na którym pracujemy:

```
<a href="nazwa_katalogu/nazwa_pliku">
```

- ❑ Hiperłącze do pliku w katalogu znajdującym się wyżej w strukturze katalogów niż plik na którym pracujemy:

```
<a href="../nazwa_pliku">
```





# CSS

Czyli praca z wyglądem strony internetowej.

- ❑ CSS (ang. Cascading Style Sheets) to kaskadowy arkusz definiujący style dokumentów HTML.
- ❑ Pozwala oddzielić strukturę (HTML) od jego wyglądu (warstwa prezentacji).
- ❑ CSS nie jest językiem programowania ani markupem.

- ❑ CSS składa się z reguł wyglądających następująco:



- ❑ kto      jaki element strony chcemy stylizować
- ❑ co      co takiemu elementowi chcemy przypisać
- ❑ jak      jak to chcemy zrobić (jaką chcemy przypisać wartość)

- ❑ Przypisanie do napisu nagłówka h1 czcionki czerwonej:

```
h1 { color: red }
```

- ❑ Przypisanie do identyfikatora o nazwie stopka fontu czcionki, marginesu (10px – góra i dół, 15px – prawy i lewy) i koloru czcionki:

```
#stopka {  
  
    font-family: 'Helvetica', 'Arial';  
  
    margin: 10px 15px  
  
    color: #09988  
  
}
```

Więcej znajdziesz w dokumentacji CSS:  
<https://www.w3schools.com/cssref/default.asp>  
[https://www.w3.org/standards/techs/css#w3c\\_all](https://www.w3.org/standards/techs/css#w3c_all)

## Wsparcie dla nowych technologii HTML i CSS w różnych przeglądarkach

- ❑ HTML i CSS jest cały czas rozwijany standardem i czasem jest konieczne upewnienie się czy jakiś właściwość jest wspierana w danej przeglądarce.
- ❑ W tym celu możemy wykorzystać stronę [www.caniuse.com](http://www.caniuse.com) gdzie możemy wpisać np.: flex i sprawdzić czy ta stosunkowo nowa właściwość CSS jest wspierana przez wszystkie przeglądarki.

Flexible Box Layout Module CR

Global 83.29% + 13.63% = 96.92%  
unprefixed 82.3% + 4.4% = 86.69%

Method of positioning elements in horizontal or vertical stacks. Support includes the support for the all properties prefixed with **flex** as well as **display: flex**, **display: inline-flex**, **align-content**, **align-items**, **align-self**, **justify-content** and **order**.

Current aligned Usage relative Show all

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Chrome for Android
			49					4.4	
8	13	47	51			9.2		4.4, 4	
11	14	48	52	9.1	39	9.3	all	51	51
		49	53	10	40				
		50	54	TP	41				
		51	55						



# Połączenie HTML i CSS

Czyli pracujemy nad pierwszą stroną internetową.

## Osadzanie stylów inline

- ❑ Wykorzystując atrybut style w kodzie HTML:

**`style="właściwość: wartość;"`**

Np.:

**`<p style="font-size: 50px; color: red;"> Tekst </p>`**

- ❑ Nie jest potrzebny selektor, ponieważ styl używamy wewnątrz konkretnego znacznika i nie ma sensu wyszczególniać do czego ma być zastosowany ten styl.

## Ważne klucze i ich wartości

- ❑ **border**: szerokość rodzaj kolor – odpowiada za ramkę dokoła elementu. Rodzajem może m.in. być solid (ramka ciągła), dotted (przerywana), groove (pseudotrójwymiarowa). Na przykład: border: 1em solid dotted to przerywana ramka o grubości jednego em,
- ❑ **padding**: wielkość – odległość między realną wielkością elementu a jego brzegami (watowanie),
- ❑ **font-family**: nazwa1 nazwa2, ... – określa czcionki, które mają być używane wewnątrz elementu, w kolejności preferencji autora strony,
- ❑ **font-size**: wielkość – określa wielkość czcionki wewnątrz elementu,
- ❑ **background-color**: kolor – określa kolor tła elementu,
- ❑ **background**: url("nazwaplikuzobrazkiem") – określa obrazek, z którego ma być wczytane tło elementu. Do wpisu można dodać no-repeat (obrazek będzie wyświetlony tylko raz) i pozycję obrazka (top, bottom, left, right, center i kombinacje, np. bottom left),



## Ważne klucze i ich wartości

- ❑ **color:** kolor – kolor elementu,
- ❑ **text-decoration:** styl – określa dodatki do tekstu: none (żadne), underline (podkreślenie), overline (kreska nad tekstem), line-through (przekreślenie),
- ❑ **text-align:** typ – sposób ustawienia tekstu: równanie do lewej (left), równanie do prawej (right), centrowanie (center),
- ❑ **margin:** wielkość – określa margines między brzegiem elementu a następnym elementem,
- ❑ **width:** wielkość – wyznacza szerokość obiektu,
- ❑ **display:** styl – zmienia tryb wyświetlania elementu, dostępne są m.in.: none (brak wyświetlania), block, inline.

## Ważne klucze i ich wartości

- ❑ **Kolor:** kolory określa się ich nazwą angielską bądź liczbą heksadecymalną (ta ostatnia stanowi trzy lub sześć znaków, wśród których mogą się znaleźć cyfry i litery od A do F; większość programów graficznych potrafi opisać kolor w postaci liczby heksadecymalnej). Standardowo dostępne są kolory o nazwach: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white i yellow.
- ❑ **Wielkość:** Odległości i wielkości określa się w jednej z kilku wybranych jednostek: em (wielkość zapożyczona z drukarstwa, proporcjonalna do aktualnej wielkości czcionki na stronie), % (w stosunku do wielkości pierwszego blokowego przodka) lub po prostu px – oznaczającej liczbę w pikselach.
- ❑ Więcej na stronie: <http://www.kurshtml.edu.pl/css/css.html>

## Osadzanie stylów w sekcji <head>

- ❑ Wykorzystując atrybut style w sekcji <head> dokumentu HTML:

```
<head>

  <style type="text/css">

    p { właściwość: wartość; }

  </style>

</head>
```

- ❑ Tak zdefiniowane style będą działały do każdego (w tym przypadku) akapitu <p> w dokumencie HTML, który może występować w sekcji <body>.

## Osadzanie stylów w osobnym pliku CSS

- ❑ Zaczynamy od podpięcia pliku .css do naszego dokumentu HTML poprzez dodanie w sekcji **<head>** następującej komendy:

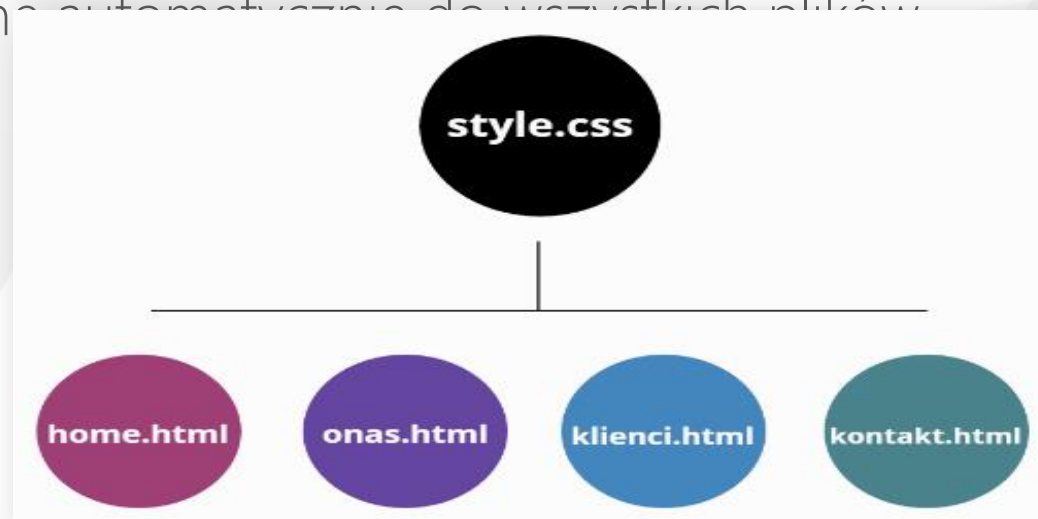
```
<link rel="stylesheet" type="text/css" href= "nazwa_pliku.css">
```

- ❑ Następnie tworzymy nowy plik z rozszerzeniem .css najlepiej w tym samym folderze co dokument HTML w którym osadzamy przykładową regułę:

```
znacznik { właściwość: wartość; }
```

## Który sposób osadzania stylów jest najlepszy?

- ❑ Oczywiście najlepszym rozwiązaniem jest trzeci sposób osadzania stylów, czyli podpięcie osobnego pliku .css wraz ze zdefiniowanymi stylami dla znaczników HTML.
- ❑ Dzięki temu możemy jeden plik .css podłączyć do wielu podstron, a ewentualne zmiany w stylach będą wprowadzane automatycznie do wszystkich plików połączonych z naszym CSS.





# Połączenie HTML i CSS

Selektory i cechy.

- ❑ Wszystkie wpisy w pliku CSS zaczynają się od selektora.
- ❑ Wyznacza on, który element HTML chcemy sformatować za pomocą CSS.

## Selektor znacznika (tagu)

- ❑ Znacznik (tag)
  - ❑ Pozwala na sformatowanie stylu dla konkretnego znacznika i wszystkie użycia tego znacznika w kodzie HTML są sformatowane identycznie.

- ❑ Plik .html

```
<znacznik> tekst </znacznik>
```

- ❑ Plik .css

```
znacznik { właściwość: wartość; }
```



## Selektor klasy

- ❑ Klasa
  - ❑ Najważniejszy i najczęściej używany typ selektora w którym określamy nazwę do konkretnego elementu HTML, czyli taką nazwę roboczą dla której następnie przypisujemy style.

- ❑ Plik .html

```
<znacznik class="nazwa"> tekst <znacznik>
```

- ❑ Plik .css

```
.nazwa { właściwość: wartość; }
```

## Selektor identyfikatora

- ❑ Identyfikator
  - ❑ Działający podobnie do selektora klasy z tą różnicą, że nazwa klasy może być taka sama dla wielu elementów, a identyfikator musi być unikatowy dla każdego elementu w którym jest zastosowany.

- ❑ Plik .html

```
<znacznik id="nazwa_id"> tekst <znacznik>
```

- ❑ Plik .css

```
#nazwa { właściwość: wartość; }
```

## Sposoby tworzenia selektorów

- ❑ Prosty:

```
znacznik { właściwość: wartość; }
```

- ❑ Złożony:

```
znacznik.nazwa_klasy { właściwość: wartość; }
```

- ❑ Zagregowany:

```
h1, h2, h3, h4 { właściwość: wartość; }
```

## Hierarchia stylów CSS



## Hierarchia stylów ogólnie

Style w pliku HTML  
w sekcji <body>

Style w pliku HTML w  
sekcji <head>

Style w pliku zewnętrznym CSS

## Dlaczego warto o hierarchii pamiętać?

- ❑ Czasem programiści chcą wprowadzić jakąś tymczasową zmianę tylko dla jednego konkretnego przypadku. Wtedy wystarczy odnaleźć element w sekcji body i nadpisać do nowym stylem.
- ❑ Dzięki temu reszta podstron posiada nadal ujednolicone style, a nie ma potrzeby dla takiej zmiany tworzyć nowego pliku CSS i go podłączać do HTML.
- ❑ Oczywiście nie jest to zbyt dobra praktyka, ale czasem bywa użyteczna.

## Konflikt w pliku CSS

- ❑ Konflikt między regułami w pliku CSS występuje wtedy gdy do tych samych selektorów przypisane zostały różne style:

Np.:

```
p { color: blue; }
```

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

- ❑ W sytuacjach konfliktowych pierwszeństwo ma reguła znajdująca się niżej w kodzie, ponieważ plik CSS jest interpretowany linijka po linijce od góry do dołu.
- ❑ Wobec tego niebieski kolor tekstu akapitu zostanie nadpisany kolorem czerwonym i ostatecznie tekst zostanie wyświetlony w tym kolorze. Ale czy zawsze tak jest?

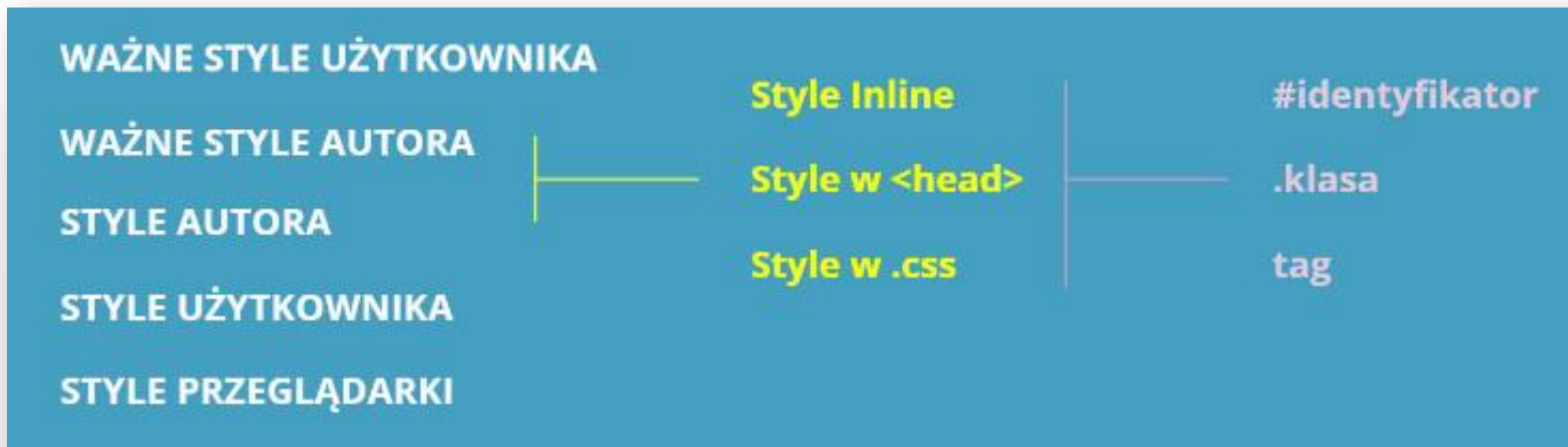
## Specyficzność

- ❑ W przypadku gdy występują specyficzne znaczniki, klasy czy identyfikatory to mamy do czynienia z następującą hierarchią ważności:





## Podsumowując hierarchie stylów CSS



## Importowanie stylów pomiędzy plikami CSS

- ❑ Aby zaimportować styl z pliku CSS (style2.css) do innego pliku CSS (style1.css), który jest podłączony do naszego dokumentu HTML, należy w pliku style1.css dodać polecenie:

```
@import url('style2.css');
```

- ❑ Teraz cała zawartość pliku style2.css została zaimportowana do pliku style1.css.

## Prosta kalkulacja hierarchii

```
#news .header .lead p { color: black; }
```

121

```
#news .lead p a span { color: black; }
```

113

```
.sekcja .historie .news .lead p a span { color: black; }
```

043

## Komentarze w CSS

- ❑ Przydatnym narzędziem są komentarze, które powodują, że zakomentowana część kodu nie jest interpretowana:

- ❑ **Komentarz jednowierszowy `//`**

Wszystko co znajduje się za znakiem `//` do końca linii nie jest interpretowane.

- ❑ **Komentarz blokowy `/* */`**

Wszystko co znajduje się pomiędzy znakami `/*` i `*/` (może być położone w wielu następujących po sobie wierszach kodu) nie jest interpretowane.

## Dobre praktyki CSS

- ☐ Pisz jak najkrótsze selektory
- ☐ Pisz nazwy klas i identyfikatorów małymi literami, a wyrazy oddzielaj znakiem \_
- ☐ Nie używaj nadmiernie identyfikatorów – są bardzo mocne w hierarchii. Lepiej jest pisać same klasy.
- ☐ Używaj intuicyjnych nazw klas
- ☐ Osadzaj style w pliku CSS
- ☐ Używaj resetowania stylów CSS ([www.csstricks.com](http://www.csstricks.com))

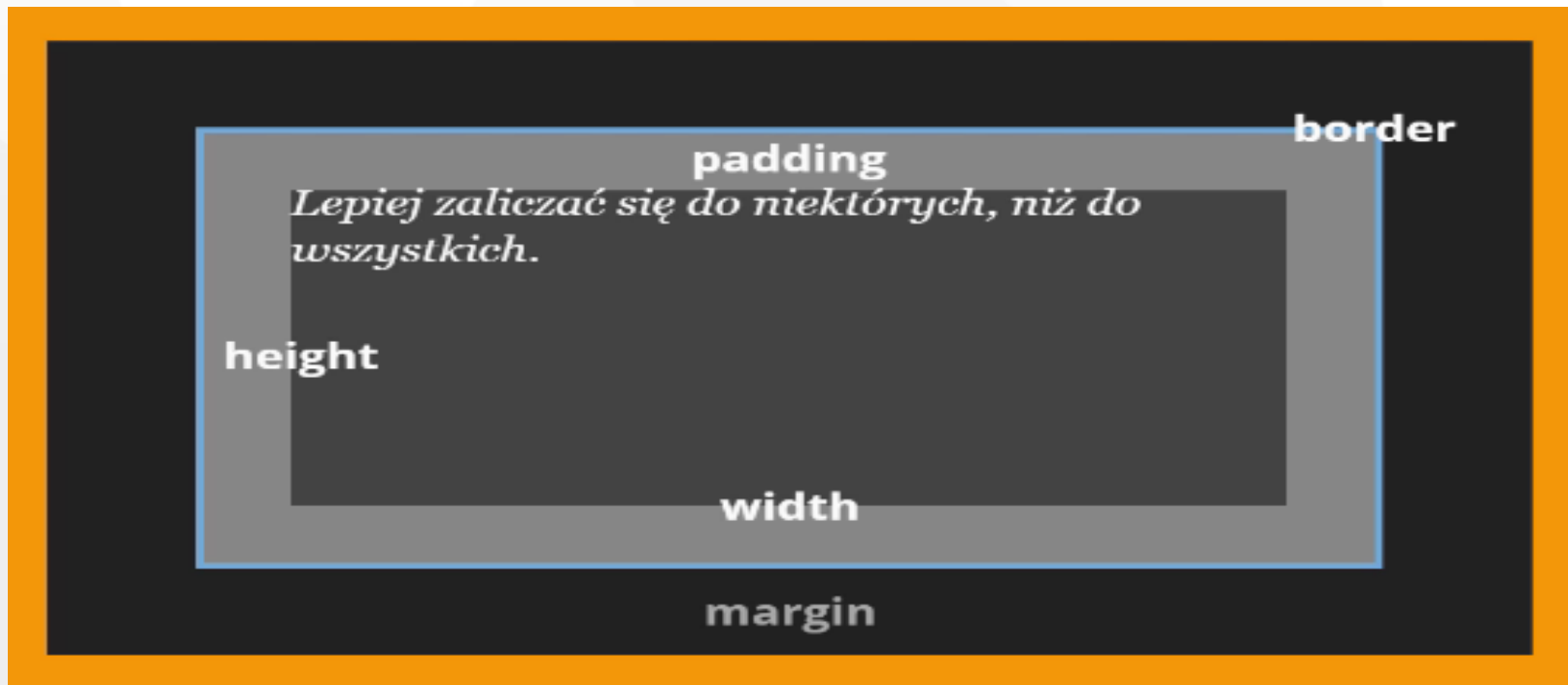


# Połączenie HTML i CSS

Model pudełkowy

## Czym jest strona internetowa w modelu pudełkowym

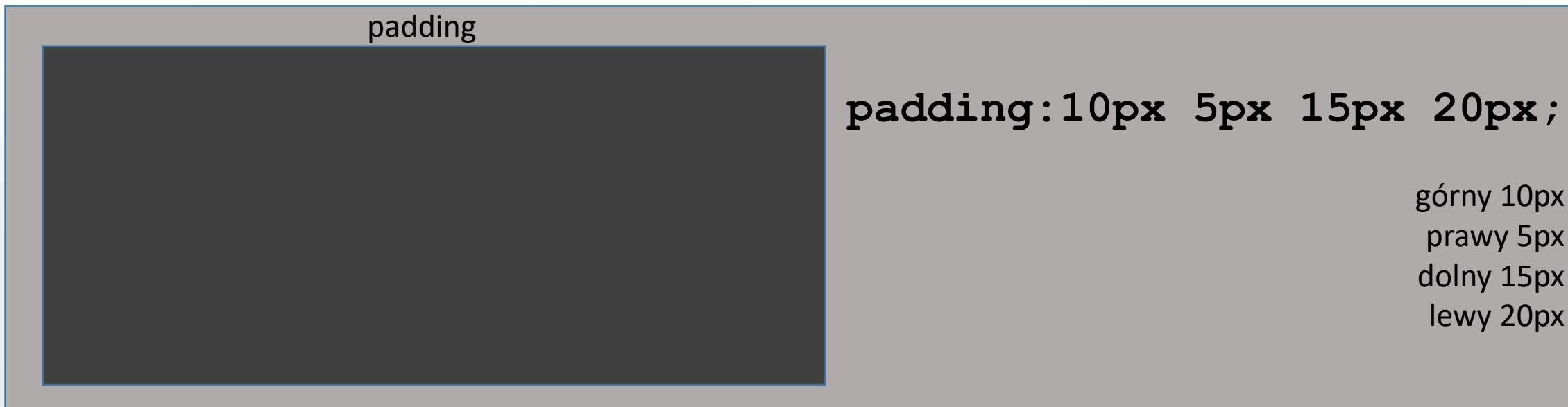
- ❑ Zestaw bloków na ekranie - każdy znacznik który dodasz tworzy taki blok.
- ❑ Za pomocą CSS możemy dowolnie ustawiać wybrane właściwości takich bloków.



- ☐ Absolutne – mają swoje fizyczne odzwierciedlenie
  - ☐ **px**
  
- ☐ Relatywne – skalowane względem jakiejś wartości odniesienia
  - ☐ **%**



## Marginesy wewnętrzne (padding)

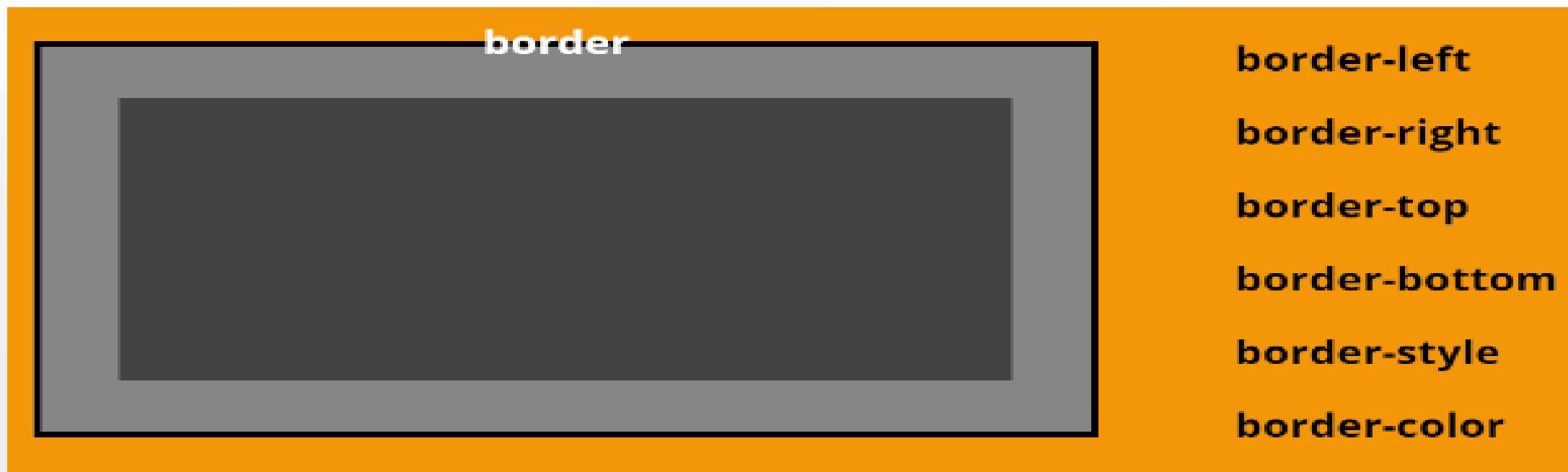


padding

```
padding:10px 5px 15px 20px;
```

górny 10px  
prawy 5px  
dolny 15px  
lewy 20px

## Obramowanie(border)



## Marginesy zewnętrzne (margin)

- ❑ Marginesy się nakładają

margin



```
margin: 25px 50px 75px 100px;  
górny 25px  
prawy 50px  
dolny 75px  
lewy 100px
```

## Pozycjonowanie elementów na stronie

- ❑ position:
  - ❑ static – nie może ulec procesowi pozycjonowania
  - ❑ relative – pozycjonowanie dokonywane jest względem krawędzi tego elementu,
  - ❑ fixed – pozycjonowanie dokonywane jest względem krawędzi okna przeglądarki internetowej
  - ❑ absolute – pozycjonowanie dokonywane jest względem zewnętrznych krawędzi padding najbliższego przodka posiadającego pozycję inną niż static

## Kolory w CSS

- ❑ Kolory w CSS są zestawem nazw (keyword) i odpowiadających im wartości (RGB hex values), które reprezentują poszczególne barwy.
- ❑ Dokładny opis znajduje się pod linkiem:

[https://developer.mozilla.org/pl/docs/Web/CSS/color\\_value](https://developer.mozilla.org/pl/docs/Web/CSS/color_value)

- ❑ Wartości kolorów w CSS są zapisane w trybie szesnastkowym, tj. #123456 z czego pierwsze dwie cyfry odpowiadają za kolor czerwony (R), kolejne dwie za kolor zielony (G) i ostatnie dwie za kolor niebieski (B). W efekcie powstaje mieszanina trzech barw podstawowych, czyli kolor wypadkowy.

- ❑ W celu pomocy w doborze odpowiednich kolorów podczas projektowania stron internetowych warto odwiedzić stronę:

<https://color.adobe.com/pl/create/color-wheel/>

## Definiowanie koloru tła

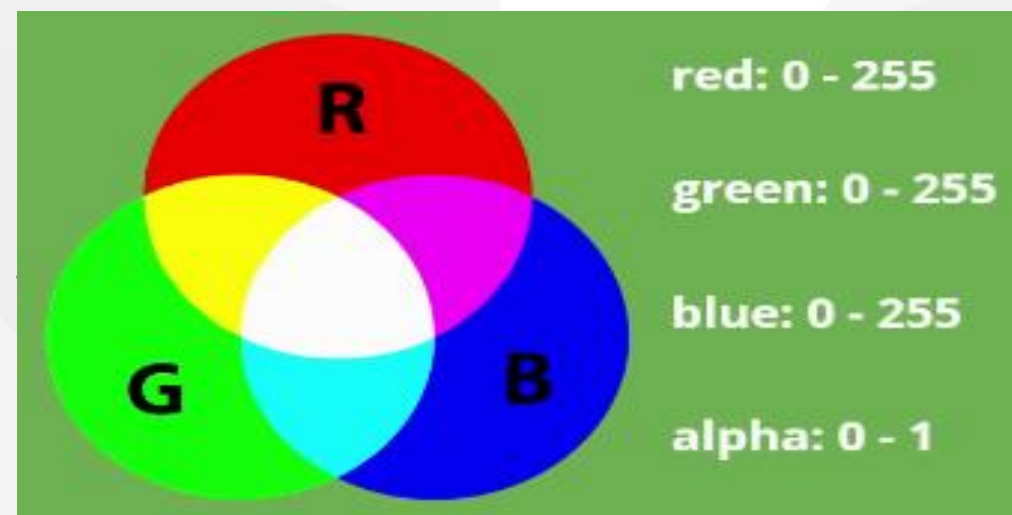
- Aby zdefiniować kolor tła dla naszej strony internetowej w pliku CSS w sekcji `<body>` wpisujemy poniższe polecenie:

```
body{  
    background-color: keyword lub wartość;  
}
```

## Efekt przezroczystości kolorów RGBA

- Aby dodać efekt przezroczystości musimy zdefiniować wartości dla kolorów w standardzie RGB oraz wartość przezroczystości A z przedziału 0-1, gdzie 1 to brak przezroczystości, a 0 to 100% przezroczystość.

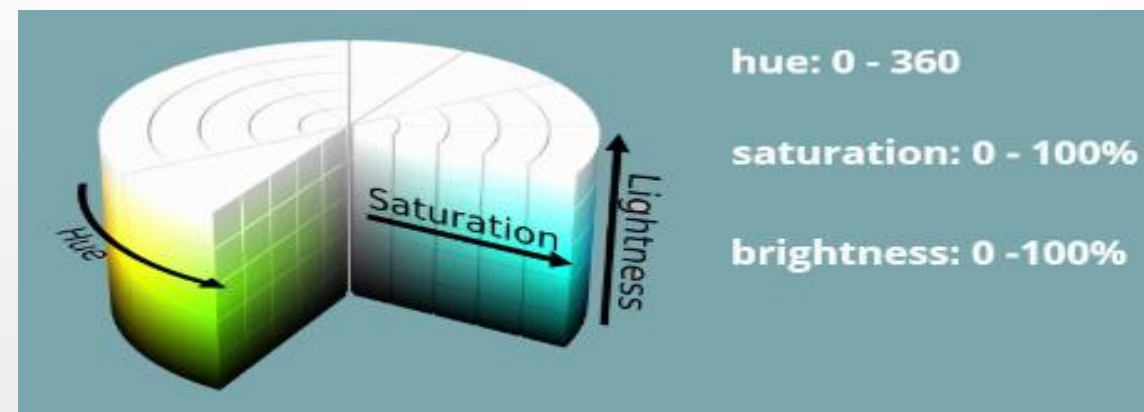
```
body{
    color: rgba(20, 20, 10, 0.5)
}
```





## Notacja HSLA

- ❑ Notacja HSLA definiuje kolory na podstawie 4 atrybutów:
  - ❑ Kolor
  - ❑ Nasycenie
  - ❑ Jasność
  - ❑ Przezroczystość



```
body{
    color: hsla(100, 10%, 50%, 0.5);
}
```

## Generyczne rodziny krojów pisma

- ❑ Typy rodzin krojów pisma:
  - ❑ Serif
  - ❑ Sans-serif
  - ❑ Monospace
  - ❑ Cursive
  - ❑ Fantasy
- ❑ Nie jest to może bardzo bogata lista wyboru, ale zapewnia poprawność wyświetlania po stronie użytkownika.
- ❑ Pewnie mniej spotykane fonty mogą nie być zainstalowane w systemie użytkownika!

## Właściwości tekstu

```
body {  
  font-family: Arial, Verdana, sans-serif;  
}
```

```
body {  
  font-size: 16px;  
}
```

```
body {  
  font-style: italic;  
}
```

```
body {  
  font-weight: bold;  
}
```

```
body {  
  line-height: 1.5;  
}
```

```
body {  
  text-indent: 1em;  
}
```

```
body {  
  text-indent: 1em;  
}
```

```
body {  
  text-align: right;  
}
```

```
body {  
  text-transform: uppercase;  
}
```

<https://www.gridlover.net/try>

## Właściwości fontów

### ☐ Właściwości fontów:

- ☐ `font-style`
- ☐ `font-variant`
- ☐ `font-weight`
- ☐ `font-size`
- ☐ `line-height`
- ☐ `font-family`

Np.:

```
body{  
    font: italic small-caps bold 12px/1.6 Arial, sans-serif;  
}
```

## Kolejność właściwości fontów

❑ /\* size | family \*/

```
font: 2em "Open Sans", sans-serif;
```

❑ /\* style | size | family \*/

```
font: italic 2em "Open Sans", sans-serif;
```

❑ /\* style | variant | weight | size/line-height | family \*/

```
font: italic small-caps bolder 16px/1.6 cursive;
```

❑ /\* style | variant | weight | stretch | size/line-height | family \*/

```
font: italic small-caps bolder condensed 16px/3 cursive;
```

## Dodawanie cienia pod tekstem

- Aby dodać cień pod tekstem w akapicie:

```
p{ text-shadow: 1px 1px 0 rgba(0,0,0,0.5); }
```

gdzie:

1px

przesunięcie cienia w poziomie

1px

przesunięcie cienia w pionie

0

rozmycie cienia

rgba()

kolor w notacji rgba

## Bezpieczne fonty

- ☐ Bezpieczne fonty możesz bez przeszkód używać na stronie i w każdej aplikacji, ponieważ są one preinstalowane w systemie użytkownika na każdej platformie, tj.:
  - ☐ Arial
  - ☐ Courier New, Courier
  - ☐ Garamond
  - ☐ Georgia
  - ☐ Lucida
  - ☐ Tahoma
  - ☐ Times New Roman, Times
  - ☐ Trebuchet
  - ☐ Verdana
  - ☐ Palatino

## Font face

- ☐ Podstawowe fonty są bardzo oklepane, ale można równie bezpiecznie wykorzystywać inne ciekawe fonty.
- ☐ Reguła font face pozwala osadzić w naszej witrynie dowolny font. Jak?
  - ☐ Generuje specjalne pliki z fontem i przechowuje na serwerze
  - ☐ Przeglądarka użytkownika nawet jeśli nie zna tych fontów będzie korzystała z plików z serwera i renderowała napisy we wskazanym foncie – bez instalacji.
  - ☐ Ważne! Muszą na to zezwalać zapisy licencyjne fonta.



## Stosujemy nietypowy font

- ❑ Pobieramy font z licencją np. ze strony: [www.fontsquirrel.com](http://www.fontsquirrel.com)
- ❑ Kopiujemy plik z fontem do katalogu gdzie znajduje się nasz plik HTML.
- ❑ Importujemy font za pomocą font face w pliku CSS:

```
@font-face{  
  
    font-family: 'MojFont';  
  
    src: url('nazwa_fonta.ttf');  
  
}
```

- ❑ Odwołujemy się do tego fonta za pomocą nadanej mu nazwy 'MojFont'

## Hostowane fonty

- ❑ Fonty znajdujące się w specjalnych serwisach ([www.fonts.google.com](http://www.fonts.google.com), [www.dafont.com](http://www.dafont.com), [www.tipekit.com](http://www.tipekit.com)), które pozwalają podłączyć konkretnego fontu do naszej strony.

- ❑ Korzystają najczęściej z fragmentów kodu JavaScript na naszej stronie.

- ❑ Plik .html:

```
<link href="https://fonts.googleapis.com/css?family=Playfair+Display" rel="stylesheet">
```

- ❑ Plik .css:

```
p { font-family: 'Playfair Display', serif; }
```

## Tło w CSS

- ❑ Wypełnienie tła

- ❑ Jednolity kolor

- background-color: kolor;**

- ❑ Gradient

- background-image: linear-gradient(kolor1, kolor2) ;**

- ❑ Obraz (uwaga na rozmiar!)

- background-image: url(adres\_obrazu) ;**

## Obrazek w tle

- ❑ Notacja obrazka w tle

- ❑ `background-image: url(adres_obrazu);`
- ❑ `height: wartość;`
- ❑ `width: wartość;`
- ❑ `background-repeat [no-repeat, repeat-x, repeat-y];`
- ❑ `background-position [center, left, right, top, bottom] lub [wartości px];`
- ❑ `background-size: [cover, contain];`

- ❑ Istnieje także skrócona notacja

❑ Odsyłam do zapoznania się z dokumentacją:

[www.colorzilla.com/gradient-editor](http://www.colorzilla.com/gradient-editor)



# Połączenie HTML i CSS

Formularze

## Parametry URL

- ❑ Dotychczas adres internetowy kończył się dla nas na nazwie pliku. Okazuje się jednak, że może w nim występować jeszcze jeden element: *parametry* (można go nazwać np. *parametrami*, *parametrami urla*, *parametrami requesta*, *parametrami wywołania*).
- ❑ Adres z parametrami może wyglądać na przykład tak:

**`http://localhost/test.php?cena=17.50&towar=telewizor&id_klienta=17432`**

- ❑ Powyższy adres oznacza dla przeglądarki: *Wejdź na stronę localhost/test.php (korzystając z protokołu http) i przekaz serwerowi listę par klucz-wartość: cena => 17.50, towar => telewizor, id\_klienta => 17432.*
- ❑ Ogólnie postać adresu zasobu HTTP (URL-a HTTP) wygląda tak:

**`http://adres_serwera/ścieżka?klucz=wartość&klucz2=wartość2&klucz3=wartość3...`**

## Parametry URL

- ❑ Wszystkie elementy licząc od adresu serwera w prawo są opcjonalne. Parametry są oddzielone od właściwego adresu znakiem `?`, poszczególne pary klucz-wartość są wydzielone znakiem `&`, natomiast klucz jest oddzielony od wartości znakiem `=`.
- ❑ W dawnych czasach było standardem, że zmienne z adresu były przepisywane do zmiennych PHP (po wejściu na stronę *test.php* z adresem jak w przykładzie powyżej, po stronie *pehapa* pojawiłaby się zmienna *\$cena* równa *'17.50'*, *\$towar* – równa *'telewizor'* i *id\_klienta* równa *'17432'*). To rozwiązanie jest teraz uznawane za nieudane, głównie z powodu potencjalnych luk bezpieczeństwa, jakie mogło wywołać.
- ❑ Adres strony z parametrami można umieścić wszędzie tam, gdzie zwykły – na przykład w linku:

`<a href='test.php?parametr=akuku'>kliknij tutaj</a>`

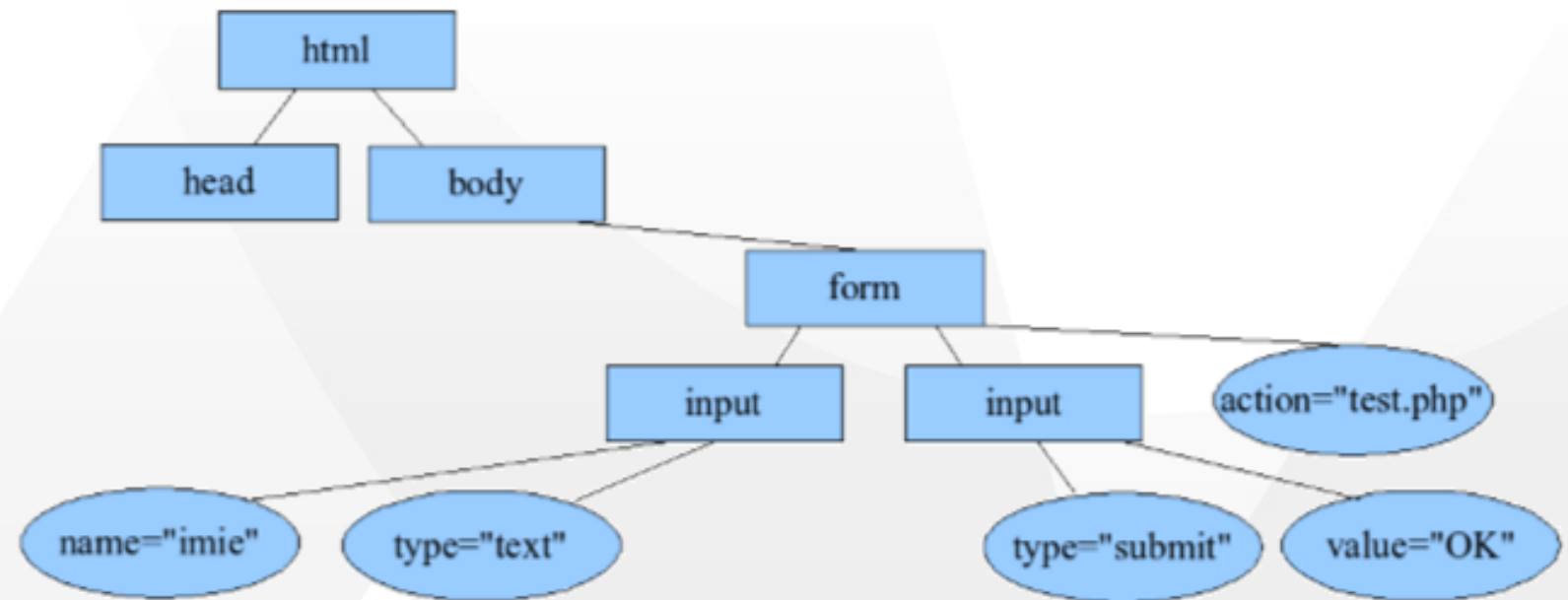
- ❑ Jednak najbardziej typowym źródłem takich URL-i są formularze.



- ❑ Służą przede wszystkim do zbierania informacji od użytkownika
- ❑ Formularz ma formę elektronicznej ankiety, którą wypełnia się wprost na stronie.
- ❑ Formularz jest generatorem zapytań http, czymś w rodzaju skomplikowanego linku.
- ❑ Wyróżnia go to, że zawartość zapytania, a ściślej: parametry dodane do URL-a, zależą od tego, jak zostały wypełnione pola formularza.

## Elementy formularza

- ☐ `<input>`
- ☐ `<select>`
- ☐ `<option>`
- ☐ `<textarea>`
- ☐ `<button>`



## Struktura formularza

- ❑ Składnia:

```
<form action="akcja" method="metoda">  
(Tutaj umieszcza się pola formularza)  
</form>
```

- ❑ Przykład:

```
<form action=''>  
  
<input name="imie" type="text" />  
  
<input type='submit' value='OK' />  
  
</form>
```

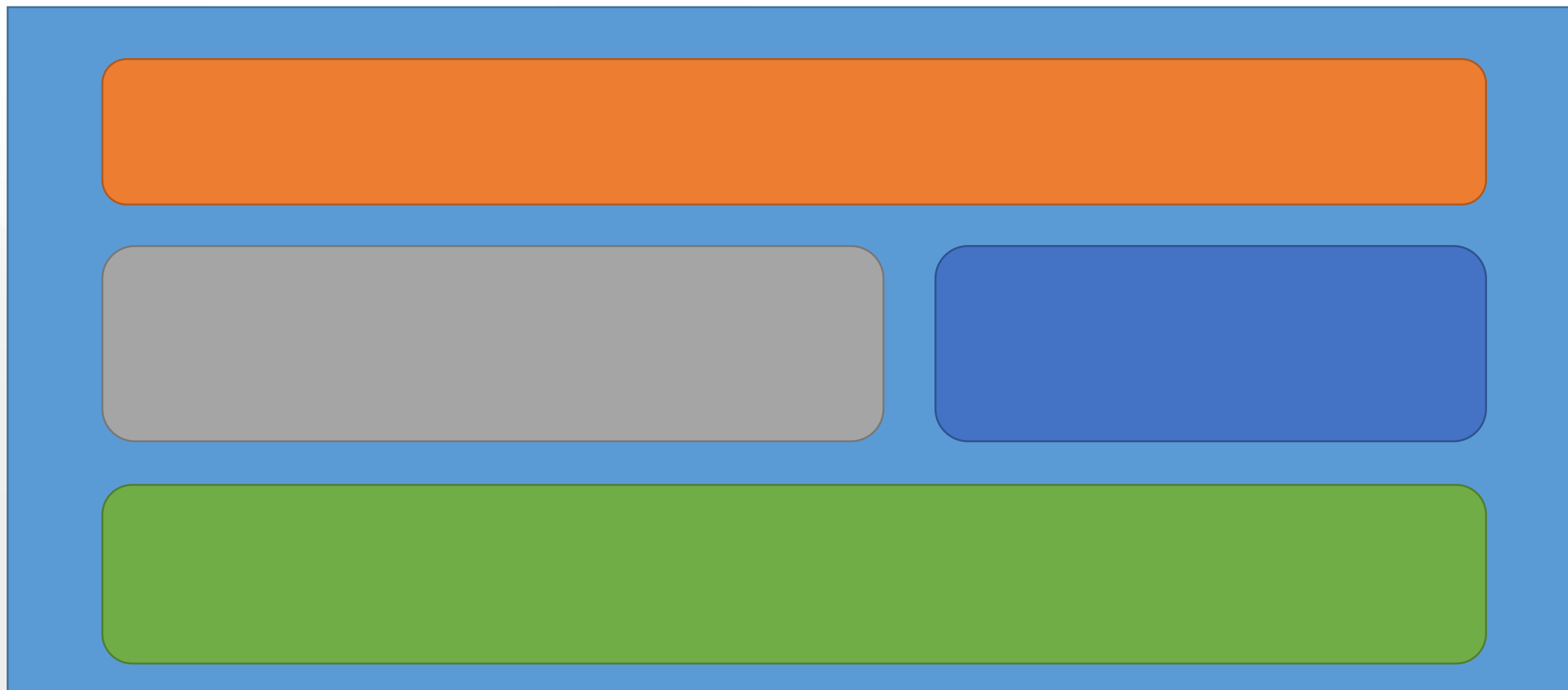
## Struktura formularza

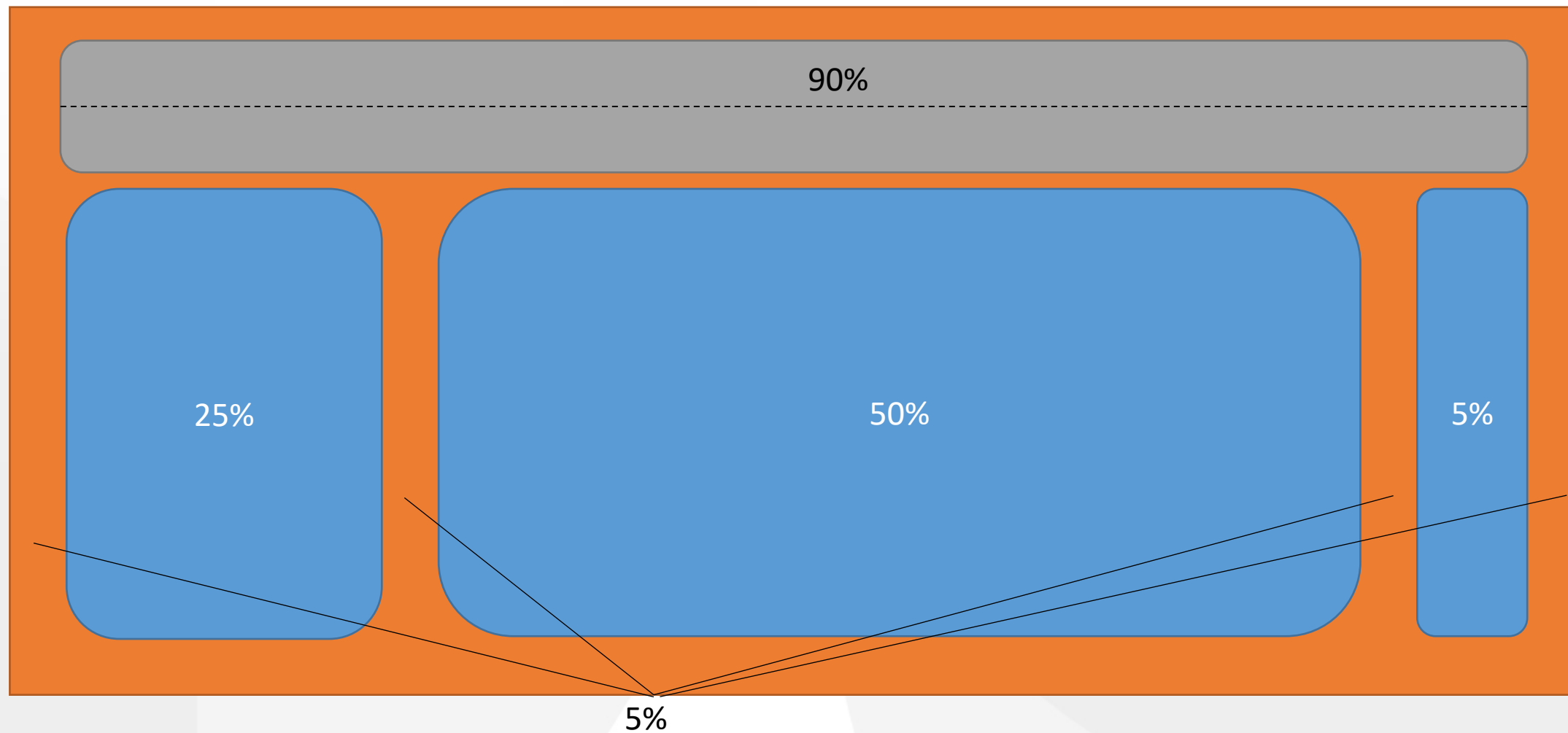
- ❑ **form: formularz** - ma zapisany adres URL (action), pod który ma się udać przeglądarka po wysłaniu formularza. Jeśli parametr action jest pusty, wysłanie formularza spowoduje powrót na ten sam adres
- ❑ **input: pole formularza** - ma nazwę (name) i typ (type). Typem najczęściej jest text (mamy wtedy do czynienia z polem tekstowym; typ text jest też typem domyślnym). Zawartość każdego pola input jest dodana do parametrów wywołania URL-a w postaci pary kluczwartość. Kluczem jest nazwa pola, a wartością – to, co wpisał w nie użytkownik.
- ❑ **input z typem submit** – specjalny rodzaj pola input. Objawia się w przeglądarce jako guzik z napisem określonym przez pole value. Jego wciśnięcie powoduje wysłanie formularza (czyli stworzenie i wysłanie przez przeglądarkę żądania HTTP z parametrami zależnymi od danych wpisanych do formularza).



# Połączenie HTML i CSS

Layout











# Bootstrap

czyli jak ułatwić sobie pracę

- ❑ Jest to framework służący do tworzenia „responsywnych” i z góry dostosowanych do obsługi na urządzeniach mobilnych stron internetowych.
- ❑ Narzędzie zapewnia dobre dostosowywanie się wyglądu strony pod różne przeglądarki.
- ❑ Cała strona budowana jest jako siatka/tabelka (ang. grid).

# Bootstrap implementacja

□ Strona projektu:

<http://getbootstrap.com/getting-started/#download>

```
<!DOCTYPE html>
<html lang="pl_PL">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap 101 Template</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body><!--tu budujemy strone -->
    <h1>Hello, world!</h1>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

## Bootstrap kontener

- ❑ Kontener o stałej szerokości:

```
<div class="container">  
  ...  
</div>
```

- ❑ Kontener szerokości całego okna przeglądarki:

```
<div class="container-fluid">  
  ...  
</div>
```

## Bootstrap rząd

- ❑ Rząd elementów – stosujemy div z klasą `.row`, który będzie mieścił jeden rząd elementów.

```
<div class="container">  
  <div class="row">  
    ...  
  </div>  
</div>
```

## Bootstrap kolumna

- ❑ Wewnątrz rzędu możemy umieścić do 12 kolumn!
- ❑ Bootstrap oferuje 4 rodzaje kolumn, kolumny dla rozdzielczości >1200px, dla rozdzielczości >992px, dla rozdzielczości >768px i dla rozdzielczości <768px

`col-lg-*`

`col-md-*`

`col-sm-*`

`col-xs-*`

- ❑ \* - wpisujemy jednostkę 1-12

```
<div class="col-xs-1">
```

...

```
</div>
```

## Bootstrap dostosowywanie wielkości kolumn

- ❑ Możliwe jest by jedna kolumna na komputerze miała rozmiar 12 jednostek, na netbooku 6, na tablecie 8 a na komórce 4.

```
<div class="col-lg-12 col-md-6 col-sm-8 col-xs-4">
```

```
...
```

```
</div>
```

## Bootstrap przestawianie kolumn

- Możemy również ustawiać kolumny w odpowiedniej kolejności bez konieczności tworzenia pustych kolumn.

**col-md-offset-\***

```
<div class="row">
```

```
  <div class="col-md-3 col-md-offset-3">...</div>
```

```
  <div class="col-md-3 col-md-offset-3">...</div>
```

```
</div>
```

- Klasa może przyjmować wartości od 1 do 11 i suma kolumn + offsetów musi wynosić 12.



## Bootstrap zagnieżdżanie kolumn

- ❑ W przypadku zagnieżdżania kolumn każdy „poziom” może zmieścić 12 kolumn.

```
<div class="row">
  <div class="col-md-12">Poziom 1: 12 kolumn
    <div class="row">
      <div class="col-md-6">Poziom 2: 6 kolumn
        <div class="row">
          <div class="col-md-6">Poziom 3: 6 kolumn</div>
          <div class="col-md-6">Poziom 3: 6 kolumn</div>
        </div>
      </div>
      <div class="col-md-6">Poziom 2: 6 columns</div>
    </div>
  </div>
</div>
```



# JavaScript

Wprowadzenie do elementów interaktywnych

## JavaScript wprowadzenie

- ❑ Jest to skryptowy język programowania, który znajduje wykorzystanie w tworzeniu interaktywnych stron WWW.
- ❑ JavaScript to program, który może być włączony do strony HTML.
- ❑ Trzeba pamiętać o tym, że użytkownik ma możliwość wyłączenia jego obsługi.
- ❑ Nie ma on możliwości generowania grafiki, jednak przy jego pomocy można manipulować istniejącymi plikami graficznymi
- ❑ JavaScript (opracowany przez Netscape) nie jest tym samym co Java (opracowana przez SunMicrosystems)

## Inicjowanie skryptów wewnątrz pliku HTML

❑ Znacznik:

```
<script> ... </script>
```

❑ Atrybut:

```
type="text/javascript"
```

❑ Zawartość:

- ❑ kod w języku JavaScript
- ❑ pojedyncze polecenia mogą kończyć się średnikiem bądź znakiem końca linii
- ❑ obiekty, metody, właściwości

## Przykładowe umieszczenie skryptu w pliku HTML

```
<html>

  <head>

    <title>Pierwszy skrypt</title>

    <script type="text/javascript">

    </script></head>

  <body>

    <script type="text/javascript">

      alert('Witaj, świecie');

    </script></body>

</html>
```

## Inicjowanie skryptów z zewnętrznego pliku

- Aby wywołać skrypt z zewnętrznego pliku należy wykorzystać znacznik wyglądający następująco:

```
<script type="text/javascript" src="nazwa_pliku.js"></script>
```

- Sam skrypt trzeba zapisać w pliku z rozszerzeniem .js

## Okna dialogowe

- ❑ alert – okienko informacyjne (message box)
- ❑ confirm – okienko pytające (zwraca true lub false)
- ❑ prompt – wprowadzanie danych (input box)

## Inicjowanie skryptu w linku

□ Składnia:

```
<a href="javascript:instrukcje;">Link</a>
```

□ Przykład:

```
<a href="javascript:alert('Hello World! ');">Execute JavaScript</a>
```



## Zmienne w JS

- ☐ Zastosowanie:
  - ☐ przechowywanie informacji
  - ☐ wartość może się zmienić w trakcie wykonywania kodu JS
- ☐ Trzeba pamiętać że JS rozróżnia wielkość liter.
- ☐ Zmienna nie może zawierać spacji.
- ☐ Zmienne mogą przyjmować następujące wartości:
  - ☐ wartości liczbowe
  - ☐ łańcuchy znaków
  - ☐ wartości logiczne
  - ☐ wartość null

## Typy zmiennych w JS

☐ Typy zmiennych:

☐ boolean

☐ char

☐ float

☐ byte

☐ double

☐ int

☐ var

## Deklaracja zmiennych w JS

- ❑ Zmienne deklaruje się w następujący sposób:

```
var nazwa_zmiennej = wartość;
```

- ❑ Wiele przeglądarek poprawnie zinterpretuje zapis:

```
nazwa_zmiennej = wartość;
```

- ❑ Pomimo tego powinno się używać pierwszego zapisu przy deklarowaniu zmiennej.

- ❑ W przypadku deklarowania zmiennej tekstowej należy pamiętać o cudzysłowach.

```
var zmienna_tekstowa = "ciąg znaków";
```

## Zasięg i czas życia zmiennych w JS

- ❑ lokalne - zadeklarowane wewnątrz funkcji – dostępne tylko wewnątrz funkcji, podczas jej wywołania od deklaracji do końca funkcji (mogą mieć te same nazwy w różnych funkcjach)
- ❑ globalne – zadeklarowane poza jakąkolwiek funkcją, dostępne we wszystkich funkcjach od chwili zadeklarowania tej zmiennej do zamknięcia strony

## Metody wbudowane typów zmiennych w JS

- ❑ `isFinite(zmienna)` - zwraca false jeżeli zmienna ma wartość +-nieskończoność i NaN
- ❑ `isNaN(zmienna)` - zwraca true jeżeli zmienna nie jest liczbą ("Not a Number")
- ❑ `Number(zmienna)` - jeżeli to możliwe to zwraca liczbę odpowiadającą obiektowi
- ❑ `parseFloat(zmienna)` - jeżeli to możliwe wykonuje konwersję zmiennej do typu float
- ❑ `parseInt(zmienna)` - jeżeli to możliwe wykonuje konwersję zmiennej do typu int
- ❑ `toString(zmienna)` - konwersja zmiennej do typu łańcuchowego
- ❑ `isArray(zmienna)` - zwraca true gdy zmienna jest tablicą
- ❑ `typeof zmienna` - zwraca typ zmiennej
- ❑ `zmienna instanceof typ` - sprawdza czy zmienna jest danego typu

## Operatory arytmetyczne

- ☐ + - dodawanie
- ☐ - - odejmowanie
- ☐ / - dzielenie
- ☐ \* - mnożenie
- ☐ % - dzielenie modulo (reszta z dzielenia)
- ☐ ++ - inkrementacja
- ☐ -- - dekrementacja

## Operatory logiczne

- ❑ == - równe
- ❑ === - identyczne (do typu zmiennej)
- ❑ != - różne
- ❑ < <= - mniejsze, mniejsze bądź równe
- ❑ > >= - większe, większe bądź równe
- ❑ ! - negacja wartości logicznej
- ❑ && - koniunkcja wartości logicznych (coś "i" coś)
- ❑ || - alternatywa wartości logicznych (coś "lub" coś)

## Operatory przypisania

- ❑ `=` - przypisanie wartości
- ❑ `+=` - powiększenie wartości zmiennej
- ❑ `-=` - pomniejszenie wartości zmiennej
- ❑ `*=` - pomnożenie wartości zmiennej
- ❑ `/= %=` - podzielenie, podzielenie modulo zmiennej



## Tablice w JS

□ Składnia:

```
var nazwa_tablicy = [element_tablicy 0, element_tablicy 1, ...,  
element_tablicyN];
```

□ Przykład:

```
var tablica = [12, 34, 35, 300]
```

## Metody tablicowe

- ❑ `nazwa_tablicy.lenght` - długość tablicy/ilość elementów tablicy
- ❑ `nazwa_tablicy.sort()` - sortowanie elementów tablicy (ale po stringach)
- ❑ `nazwa_tablicy.push(element)` - dodawanie elementu na końcu tablicy
- ❑ `nazwa_tablicy.pop()` - usuwanie i zwracanie elementu na końcu tablicy
- ❑ `nazwa_tablicy.shift()` - usuwanie elementu z początku tablicy
- ❑ `nazwa_tablicy.unshift(element)` - dodawanie elementu na początek tablicy
- ❑ `nazwa_tablicy.splice(index_gdzie_zaczyna,ile_usuwa,elementy,...)` - dodaje elementy do tablicy
- ❑ `nazwa_tablicy.concat()` - łączy dwie tablice w jedną nową
- ❑ `nazwa_tablicy.slice()` - tworzy nową tablice z wyznaczonych elementów

## Instrukcja warunkowa if w JavaScript

- ❑ Składnia if:

```
if (warunek) instrukcja;
```

- ❑ Operator warunkowy (to nie jest if):

```
warunek ? wartość : wartość
```

- ❑ lub:

```
if (warunek)  
{  
    Instrukcja;  
}  
else  
{  
    Instrukcja;  
}
```

## Przykład zastosowania instrukcji warunkowej if

```
<script language="javascript" type="text/javascript">

var a = 4;

var b = 4;

if (a == b) alert('Wartości są te same');

if (a == b && a > 2 && b > 1){

alert('złożony warunek');

} else {

alert('nie spełniony złożony warunek');

}

wynik = (a == b) ? 'Zmienna są równe' : 'Zmienne mają różne wartości';

alert(wynik);

</script>
```

# Instrukcja switch w JavaScript

## □ Składnia:

```
switch (wyrażenie)
{
  case wartość_wyrażenia1:
    instrukcja1;
    break;
  case wartość_wyrażenia2:
    instrukcja2;
    break;
  default:
    instrukcja;
}
```

## Zastosowanie instrukcji switch

```
<script language="javascript" type="text/javascript">
```

```
var a = 4; var b = 4;
```

```
switch (a+b) {
```

```
  case 5:
```

```
    alert('mamy 5'); break;
```

```
  case 8:
```

```
    alert('mamy 8'); break;
```

```
  default:
```

```
    alert('odpowieź domyślna');
```

```
}
```

```
</script>
```

## Pętla for w JavaScript

□ Składnia:

```
for(zmienna=wart_początkowa; warunek końca; zmiana wartości)
{
    instrukcje;
}
```

## Zastosowanie pętli for

```
<script language="javascript" type="text/javascript">
```

```
for (i = 1; i <= 5; i++)
```

```
{
```

```
    alert(i);
```

```
}
```

```
</script>
```



## Pętle while i do-while w JavaScript

□ Składnia WHILE:

```
while (warunek)
{
    instrukcje;
}
```

□ Składnia DO WHILE:

```
do
{
    instrukcje;
}
while (warunek)
```

## Zastosowanie pętli do-while

```
<script language="javascript" type="text/javascript">  
  
var i = 0;  
while (i < 4) {  
    i++; alert(i);  
}  
  
var j = 0;  
do {  
    alert(j); j++;  
} while (j < 2);  
  
</script>
```

# Funkcje w JavaScript

□ Składnia:

```
function nazwa_funkcji (ARGUMENTY)
{
    instrukcje;
    return wartość;
}
```

## Zastosowanie funkcji

```
<script language="javascript" type="text/javascript">
```

```
function moja_funk(a, b, c)
```

```
{
```

```
  z = a+b;
```

```
  z = z/c;
```

```
  return z;
```

```
}
```

```
alert(moja_funk(2, 4, 7));
```

```
</script>
```

## Obsługa zdarzeń w JavaScript

- ❑ **onabort** - występuje w przypadku przerwania ładowania strony/rysunku
- ❑ **onblur** - występuje, gdy dany element przestaje być aktywny
- ❑ **onchange** - występuje w przypadku zmiany zawartości elementu
- ❑ **onclick** - występuje w momencie kliknięcia na dany element
- ❑ **ondblclick** - występuje w momencie podwójnego kliknięcia na elemencie
- ❑ **onerror** - występuje w przypadku wystąpienia błędu na stronie
- ❑ **onfocus** - występuje, gdy dany element staje się aktywny
- ❑ **onload** - występuje w czasie ładowania strony
- ❑ **onmousemove** - występuje, kiedy przesuniemy kursor myszy nad elementem
- ❑ **onmouseout** - występuje, kiedy przesuniemy kursor myszy poza element
- ❑ **onmouseover** - występuje, kiedy przesuniemy kursor myszy na element
- ❑ **onsubmit** - występuje przy wciśnięciu pola typu submit formularzy

## Obiekt w JavaScript

- ❑ Obiekty w JS tak jak w życiu mają swoje parametry/pola (przymiotniki) przy pomocy których je opisujemy.
- ❑ Składnia:

```
var nazwa_obiektu = {parametr1:"wartość_parametru1",  
parametr2:"wartość_parametru2", ... ,nazwa_metody : function() {  
instrukcje }, ...};
```

## Model DOM

- ❑ DOM, czyli obiektowy model dokumentu (ang. Document Object Model) jest to sposób reprezentacji złożonych dokumentów XML i HTML w postaci modelu obiektowego.
- ❑ Model ten jest niezależny od platformy i języka programowania.
- ❑ Standard W3C DOM definiuje zespół klas i interfejsów, pozwalających na dostęp do struktury dokumentów oraz jej modyfikację poprzez tworzenie, usuwanie i modyfikację tzw. węzłów (ang. nodes).

Dokumentacja do poczytania:

- <https://www.w3.org/DOM/>
- <https://www.w3.org/TR/html5/dom.html#kinds-of-content>



## Model DOM i BOM

- ❑ BOM (Browser Object Model) to zbiór obiektów dających dostęp do przeglądarki i ekranu. Najważniejszy obiekt to `window` – obiekt globalny.

Wszystkie zmienne globalne stają się polami tego obiektu. Tworząc nową zmienną globalną `zm`, tak naprawdę tworzy się zmienną `window.zm`.

- ❑ Obiekt `window` zawiera w sobie następujące obiekty:
  - ❑ `navigator` – obiekt przechowujący informacje o przeglądarce i jej możliwościach
  - ❑ `location` – obiekt przechowujący informację o adresie URL aktualnie załadowanej strony
  - ❑ `history` – obiekt dający dostęp do stron odwiedzanych wcześniej w ramach tej samej sesji przeglądarki,
  - ❑ `frames` – przechowuje zbiór ramek z bieżącej strony (`iframes`)
  - ❑ `screen` – zawiera informacje na temat ustawień monitora (rozdzielczości, głębi kolorów) oraz o rozmiarze okna przeglądarki
  - ❑ `document` – obiekt związany z aktualnie załadowanym dokumentem. Zawiera w sobie obiekty z DOM
  - ❑ Pola i metody DOM umożliwiają dostęp i modyfikację wszystkich elementów strony



## Odwoływanie się do pól

- ❑ JavaScript to język obiektowy i pozwala w pełni na korzystanie z modelu DOM.

Dzięki czemu możemy bezpośrednio odnosić się do różnych obiektów strony.

- ❑ `document.title` - tytuł dokumentu
- ❑ `document.URL` - adres URL dokumentu

## Wywoływanie metod

- ❑ `getElementById()` - uzyskuje dostęp do elementu w dokumencie, któremu określono podany atrybut `id`
- ❑ `getElementsByClassName()` - to samo dla podanego atrybutu `class`
- ❑ `getElementsByTagName()` - uzyskuje dostęp do elementów w dokumencie, których znaczniki są podanej treści
- ❑ `document.write()` - wyświetla tekst
- ❑ `document.open()` - otwiera strumień danych w oknie przeglądarki
- ❑ `document.close()` - zamyka strumień danych



# jQuery

interaktywne interfejsy internetowe

## jQuery wprowadzenie

- ❑ Biblioteka JavaScript która jest:
  - ❑ Łatwa do nauczenia
  - ❑ Łatwa w użyciu
  - ❑ Znacznie upraszcza programowanie w JavaScript
  - ❑ Jest mnóstwo już napisanego kodu w jQuery,

## Osadzenie skryptu jQuery

- ❑ Lokalizacja lokalna:

```
<script type="text/javascript"  
src="sciezka_do_jQuery/nazwa_pliku_z_jQuery.js"></script>
```

- ❑ Lokalizacja serwer CDN

```
<script  
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>
```

lub:

```
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.2.0.min.js"></script>
```

## Składnia jQuery

- ❑ Podstawowa składnia:

```
$(selektor).akcja1().akcja2().akcjaN()
```

- ❑ Ważna konstrukcja:

```
$(document).ready(function() { ... });
```

## No conflict jQuery

- ❑ Do korzystania z funkcjonalności jQuery korzystamy z aliasu \$ czasami zdarza się że następuje konflikt gdy używamy innych bibliotek w których również wykorzystuje się ten skrót. Aby unikać tego typu konfliktu możemy użyć mechanizmu noConflict

- ❑ Przykładowa składnia:

```
<script src=„inna_lib.js"></script>
<script src="jquery.js"></script>
<script>
    $.noConflict();
    jQuery( document ).ready(function( $ ) {
// Tutaj wpisujemy kod w którym wykorzystujemy jQuery
    });
    // Tutaj kod dla innej biblioteki
</script>
```

- ❑ Służą do wyszukiwania odpowiednich węzłów w drzewie DOM
- ❑ Kilka przykładów:
  - ❑ Selektory typu element:
    - ❑ `$("p")`
    - ❑ `$("p.intro")`
    - ❑ `$("#demo")`
  - ❑ Selektory typu atrybut:
    - ❑ `$("[href]")`
  - ❑ Selektor typu CSS:
    - ❑ `$("p").css("background-color", "yellow");`



❑ Przykładowe zdarzenia:

- ❑ `$(document).ready(function)`
- ❑ `$(selector).click(function)`
- ❑ `$(selector).dblclick(function)`
- ❑ `$(selector).focus(function)`
- ❑ `$(selector).mouseover(function)`

- ❑ jQuery ma zaimplementowany szereg różnych efektów.
- ❑ Przykłady:
  - ❑ `$(selector).hide(speed, callback)`
  - ❑ `$(selector).show(speed, callback)`
  - ❑ `$(selector).slideDown(speed, callback)`
  - ❑ `$(selector).slideUp(speed, callback)`
  - ❑ `$(selector).fadeIn(speed, callback)`
  - ❑ `$(selector).fadeOut(speed, callback)`



Dziękuję za uwagę!