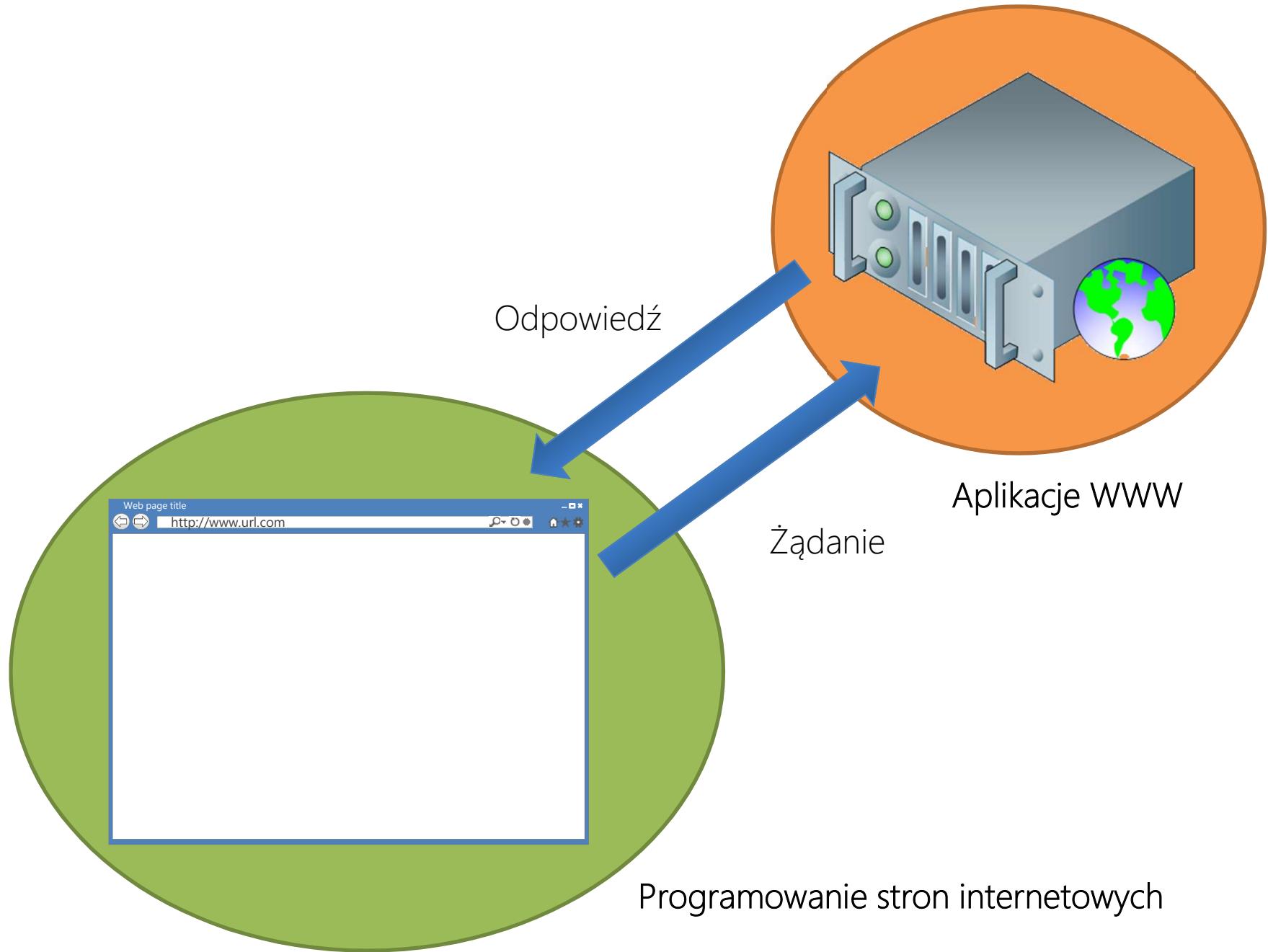


Programowanie stron internetowych

Łukasz Bartczuk

1. WPROWADZENIE



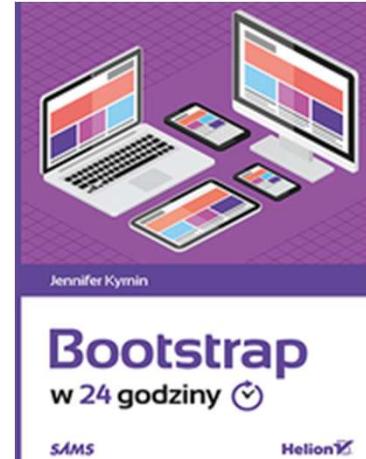
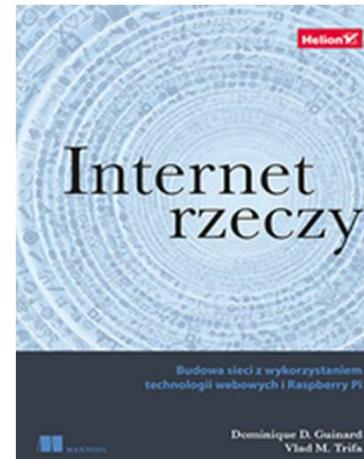
Zakres przedmiotu

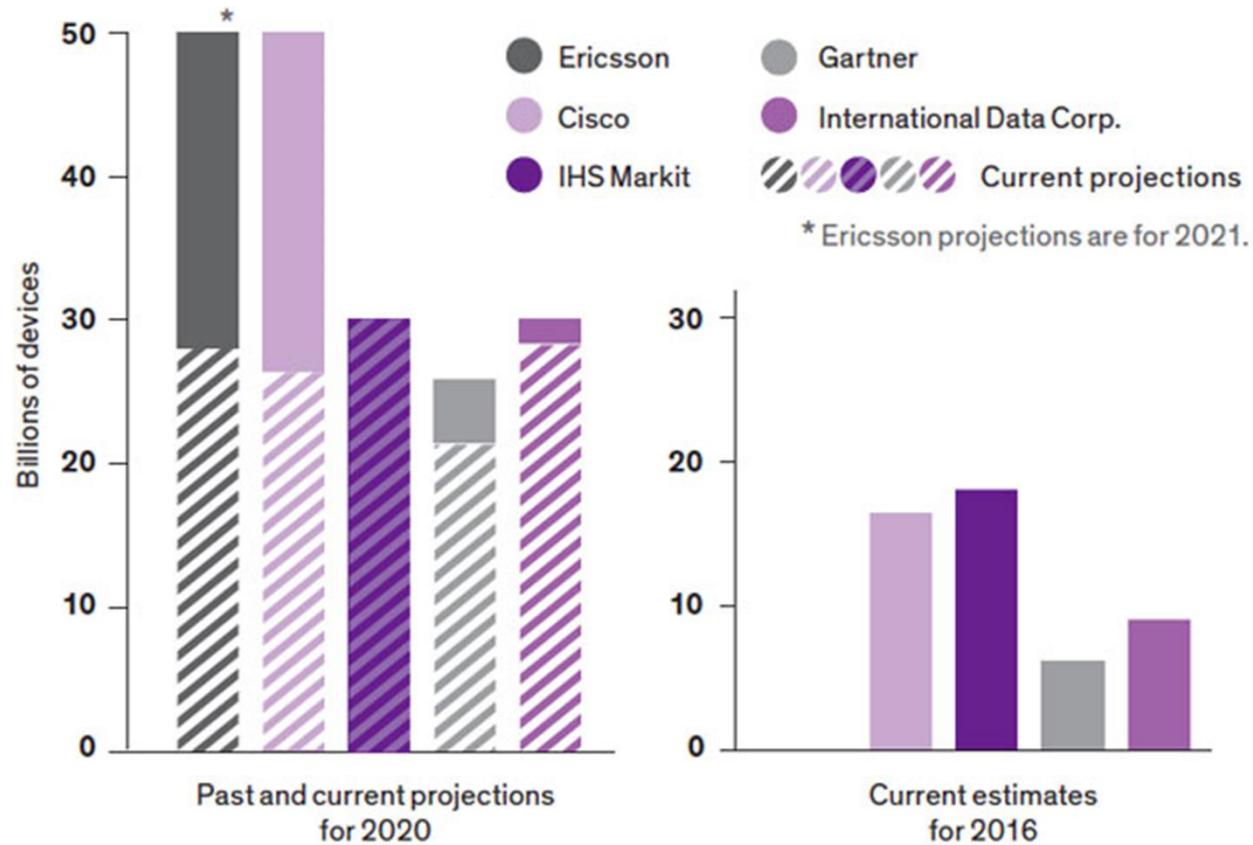
1. Wprowadzenie
2. Tworzenie stron internetowych w języku HTML
3. Ustalanie wyglądu strony internetowej za pomocą języka CSS
4. Wprowadzenie do języka JavaScript
5. Tworzenie aplikacji internetowych działających całkowicie w przeglądarce
6. Omówienie wybranych bibliotek ułatwiających tworzenie aplikacji w języku JavaScript
7. Tworzenie asynchronicznych aplikacji internetowych
8. Zastosowanie API języka HTML 5
9. Inne technologie i języki przydatne podczas tworzenia stron i aplikacji internetowych

Zaliczenie wykładu

KOLOKWIUM w formie testowej

Literatura





Amy Nordrum, *The Internet of Fewer Things*, IEEE Spectrum, October 2016

Protokoły Internetu

WEBSOCKET

FTP

HTTP

BITTORRENT

POP

SOAP

DHCP

RSTP

DNS

TELNET

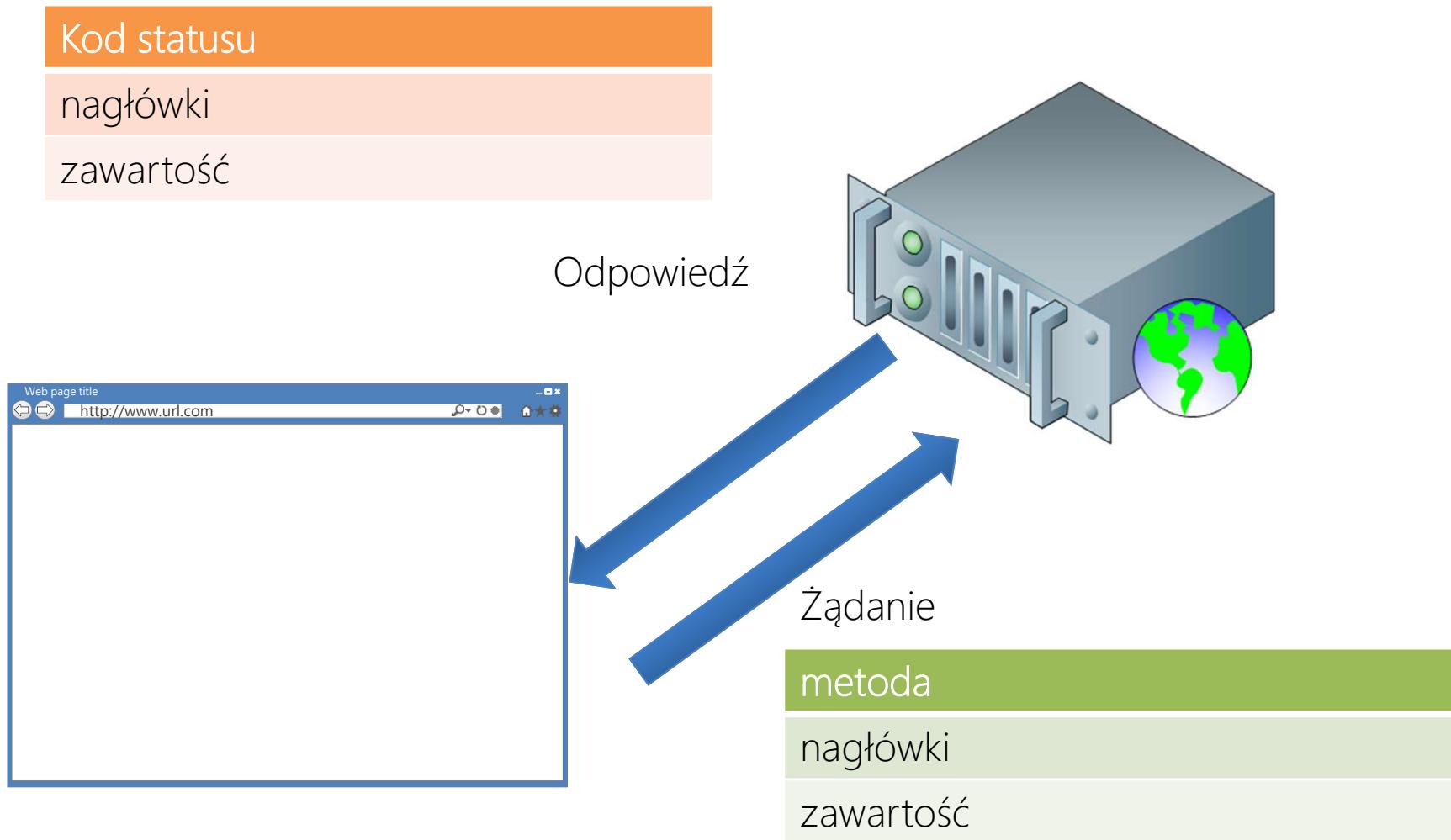
SMTP

RDP

BITCOIN

TSL/SSL

Protokół HTTP



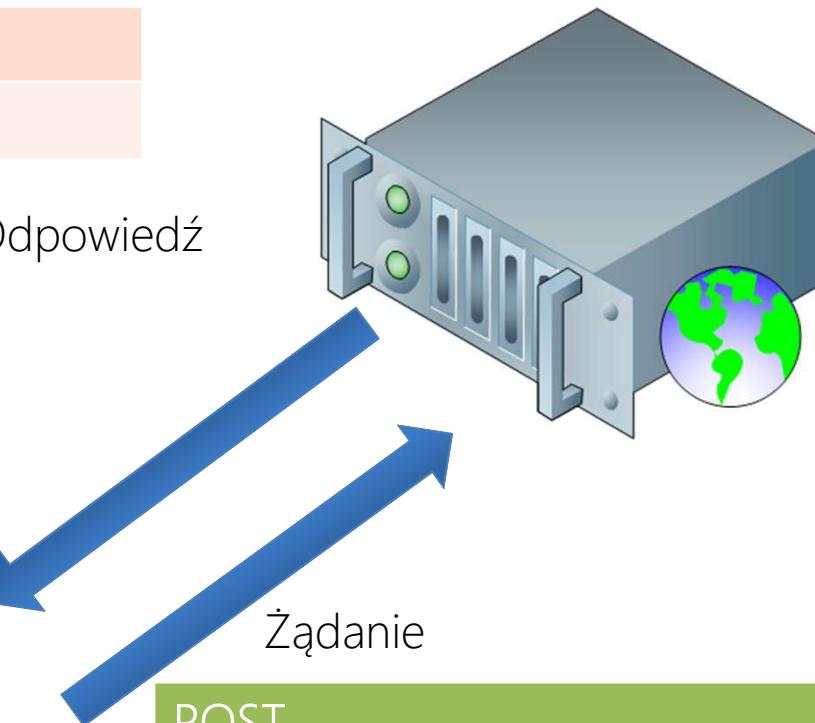
Protokół HTTP

200 OK

Content type:text/html

Ala ma kota

Odpowiedź



POST

Content type: application/x-www-form-urlencoded

imie=Ala&nazwisko=Kowalska

Żądanie protokołu HTTP

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; pl; rv:1.8.1.7)
Gecko/20070914 Firefox/2.0.0.7
Accept: text/xml,application/xml,application/xhtml+xml,text/html;
q=0.9,text/plain;q=0.8
Accept-Language: pl,en-us;q=0.7,en;q=0.3
Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

Metody protokołu HTTP

Metoda	Opis
GET	pobranie zasobu wskazanego przez URI , może mieć postać warunkową jeśli w nagłówku występują pola warunkowe takie jak "If-Modified-Since"
POST	przyjęcie danych przesyłanych od klienta do serwera (np. wysyłanie zawartości formularzy)
PUT	przyjęcie danych przesyłanych od klienta do serwera, najczęściej aby zaktualizować wartość encji
DELETE	żądanie usunięcia zasobu, włączone dla uprawnionych użytkowników

Metody protokołu HTTP

Metoda	Opis
HEAD	pobiera informacje o zasobie, stosowane do sprawdzania dostępności zasobu
OPTIONS	informacje o opcjach i wymaganiach istniejących w kanale komunikacyjnym
TRACE	diagnostyka, analiza kanału komunikacyjnego
CONNECT	żądanie przeznaczone dla serwerów pośredniczących pełniących funkcje tunelowania
PATCH	aktualizacja części danych

Odpowiedź HTTP

HTTP/1.1 200 OK

Date: Thu, 20 Dec 2001 12:04:30 GMT

Server: Apache/2.0.50 (Unix) DAV/2

Set-Cookie: PSID=d6dd02e9957fb162d2385ca6f2829a73; path=/

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate

Keep-Alive: timeout=15, max=100

Connection: Keep-Alive

Transfer-Encoding: chunked

Content-Type: application/xhtml+xml; charset=utf-8

Kody statusu

Kod	Opis	Kod	Opis
200	OK	400	Bed request
201	Created	401	Not authorized
204	Accepted	403	Forbidden
302	Found	404	Not found
304	Not Modified	405	Method not allowed
307	Temp redirect	409	Conflict
308	Perm redirect	412	Precondition Failed
		500	Internal Error

Status odpowiedzi - 100

Kody informacyjne

kod	opis słowny	znaczenie/zwrócony zasób
100	Continue	Kontynuuj – prośba o dalsze wysyłanie zapytania
101	Switching Protocols	Zmiana protokołu
110	Connection Timed Out	Przekroczono czas połączenia. Serwer zbyt długo nie odpowiada.
111	Connection refused	Serwer odrzucił połączenie

Status odpowiedzi - 200

Kody powodzenia

kod	opis słowny	znaczenie/zwrócony zasób
200	OK	Zawartość żądanego dokumentu
201	Created	Utworzono – wysłany dokument został zapisany na serwerze
202	Accepted	Przyjęto – zapytanie zostało przyjęte do obsłużenia, lecz jego zrealizowanie jeszcze się nie skończyło
203	Non-Authoritative Information	Informacja nieautorytywna – zwrócona informacja nie odpowiada dokładnie odpowiedzi pierwotnego serwera, lecz została utworzona z lokalnych bądź zewnętrznych kopii

Status odpowiedzi - 200

Kody powodzenia

kod	opis słowny	znaczenie/zwrócony zasób
204	No content	Brak zawartości – serwer zrealizował zapytanie klienta i nie potrzebuje zwracać żadnej treści
205	Reset Content	Przywróć zawartość – serwer zrealizował zapytanie i klient powinien przywrócić pierwotny wygląd dokumentu
206	Partial Content	Część zawartości – serwer zrealizował tylko część zapytania typu GET, odpowiedź musi zawierać nagłówek Range informujący o zakresie bajtowym zwróconego elementu

Status odpowiedzi - 300

Kody przekierowania

kod	opis słowny	znaczenie/zwrócony zasób
300	Multiple Choices	Wiele możliwości – istnieje więcej niż jeden sposób obsłużenia danego zapytania, serwer może podać adres zasobu, który pozwala na wybór jednoznacznego zapytania spośród możliwych
301	Moved Permanently	Trwale przeniesiony – żądany zasób zmienił swój URI i w przyszłości zasób powinien być szukany pod wskazanym nowym adresem
302	Found	Znaleziono – żądany zasób jest chwilowo dostępny pod innym adresem a przyszłe odwołania do zasobu powinny być kierowane pod adres pierwotny
303	See Other	Zobacz inne – odpowiedź na żądanie znajduje się pod innym URI i tam klient powinien się skierować. To jest właściwy sposób przekierowywania w odpowiedzi na żądanie metodą POST.

Status odpowiedzi - 300

Kody przekierowania

kod	opis słowny	znaczenie/zwrócony zasób
304	Not Modified	Nie zmieniono – zawartość zasobu nie podległa zmianie według warunku przekazanego przez klienta
305	Use Proxy	Użyj serwera proxy – do żądanego zasobu trzeba odwołać się przez serwer proxy podany w nagłówku Location odpowiedzi
306	Switch Proxy	Kod nieużywany, aczkolwiek zastrzeżony dla starszych wersji protokołu
307	Temporary Redirect	Tymczasowe przekierowanie – żądany zasób znajduje się chwilowo pod innym adresem URI, odpowiedź powinna zawierać zmieniony adres zasobu, na który klient zobowiązany jest się przenieść
310	Too many redirects	Zbyt wiele przekierowań.

Status odpowiedzi - 400

Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
400	Bad Request	Nieprawidłowe zapytanie – żądanie nie może być obsłużone przez serwer z powodu nieprawidłowości postrzeganej jako błąd użytkownika (np. błędna składnia zapytania)
401	Unauthorized	Nieautoryzowany dostęp – żądanie zasobu, który wymaga uwierzytelnienia
402	Payment Required	Wymagana opłata – odpowiedź zarezerwowana na przyszłość
403	Forbidden	Zabroniony – serwer zrozumiał zapytanie lecz konfiguracja bezpieczeństwa zabrania mu zwrócić żądanego zasobu
404	Not Found	Nie znaleziono – serwer nie odnalazł zasobu według podanego URL ani niczego co by wskazywało na istnienie takiego zasobu w przeszłości

Status odpowiedzi - 400

Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
405	Method Not Allowed	Niedozwolona metoda – metoda zawarta w żądaniu nie jest dozwolona dla wskazanego zasobu, odpowiedź zawiera też listę dozwolonych metod
406	Not Acceptable	Niedozwolone – zażądany zasób nie jest w stanie zwrócić odpowiedzi mogącej być obsłużonej przez klienta według informacji podanych w zapytaniu
407	Proxy Authentication Required	Wymagane uwierzytelnienie do serwera pośredniczącego (ang. proxy) – analogicznie do kodu 401, dotyczy dostępu do serwera proxy
408	Request Timeout	Koniec czasu oczekiwania na żądanie – klient nie przesłał zapytania do serwera w określonym czasie

Status odpowiedzi - 400

Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
409	Conflict	Konflikt – żądanie nie może być zrealizowane, ponieważ występuje konflikt z obecnym statusem zasobu, ten kod odpowiedzi jest zwracany tylko w przypadku podejrzewania przez serwer, że klient może znaleźć przyczyny błędu i przesyłać ponownie prawidłowe zapytanie. Odpowiedź serwera powinna zawierać informację umożliwiającą klientowi rozwiązywanie problemu, jednak nie jest to obowiązkowe.
410	Gone	Zniknął (usunięto) – zażądany zasób nie jest dłużej dostępny i nieznany jest jego ewentualny nowy adres URL; klient powinien już więcej nie odwoływać się do tego zasobu
411	Length required	Wymagana długość – serwer odmawia zrealizowania zapytania ze względu na brak nagłówka Content-Length w zapytaniu; klient może powtórzyć zapytanie dodając doń poprawny nagłówek długości

Status odpowiedzi - 400

Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
412	Precondition Failed	Warunek wstępny nie może być spełniony – serwer nie może spełnić przynajmniej jednego z warunków zawartych w zapytaniu
413	Request Entity Too Large	Encja zapytania zbyt długa – całkowita długość zapytania jest zbyt długa dla serwera
414	Request-URI Too Long	Adres URI zapytania zbyt długi – długość zażdanego URI jest większa niż maksymalna oczekiwana przez serwer
415	Unsupported Media Type	Nieznany sposób żądania – serwer odmawia przyjęcia zapytania, ponieważ jego składnia jest niezrozumiała dla serwera

Status odpowiedzi - 400

Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
416	Requested Range Not Satisfiable	Zakres bajtowy podany w zapytaniu nie do obsłużenia – klient podał w zapytaniu zakres, który nie może być zastosowany do wskazanego zasobu
417	Expectation Failed	Oczekiwana wartość nie do zwrócenia – oczekiwanie podane w nagłówku Expect żądania nie może być spełnione przez serwer lub – jeśli zapytanie realizuje serwer proxy – serwer ma dowód, że oczekiwanie nie będzie spełnione przez następny w łańcuchu serwer realizujący zapytanie
418	I'm a teapot	"Jestem czajnikiem" – tzw. easter egg. Zdefiniowany w 1998. Obecnie nie jest implementowany do serwerów HTTP, ale znane są takie przypadki
451	Unavailable For Legal Reasons	Zawartość niedostępna z powodów prawnych – strona lub zasób zostały zablokowane z powodów naruszenia prawa, w tym także z powodu ocenzurowania zawartości przez władze

Status odpowiedzi - 500

Kody błędu aplikacji serwera

kod	opis słowny	znaczenie/zwrócony zasób
500	Internal Server Error	Wewnętrzny błąd serwera – serwer napotkał niespodziewane trudności, które uniemożliwiły zrealizowanie żądania
501	Not Implemented	Nie zaimplementowano – serwer nie dysponuje funkcjonalnością wymaganą w zapytaniu; ten kod jest zwracany, gdy serwer otrzymał nieznany typ zapytania
502	Bad Gateway	Błąd bramy – serwer – spełniający rolę bramy lub pośrednika – otrzymał niepoprawną odpowiedź od serwera nadziednego i nie jest w stanie zrealizować żądania klienta
503	Service Unavailable	Usługa niedostępna – serwer nie jest w stanie w danej chwili zrealizować zapytania klienta ze względu na przeciążenie

Status odpowiedzi - 500

Kody błędu aplikacji klienta

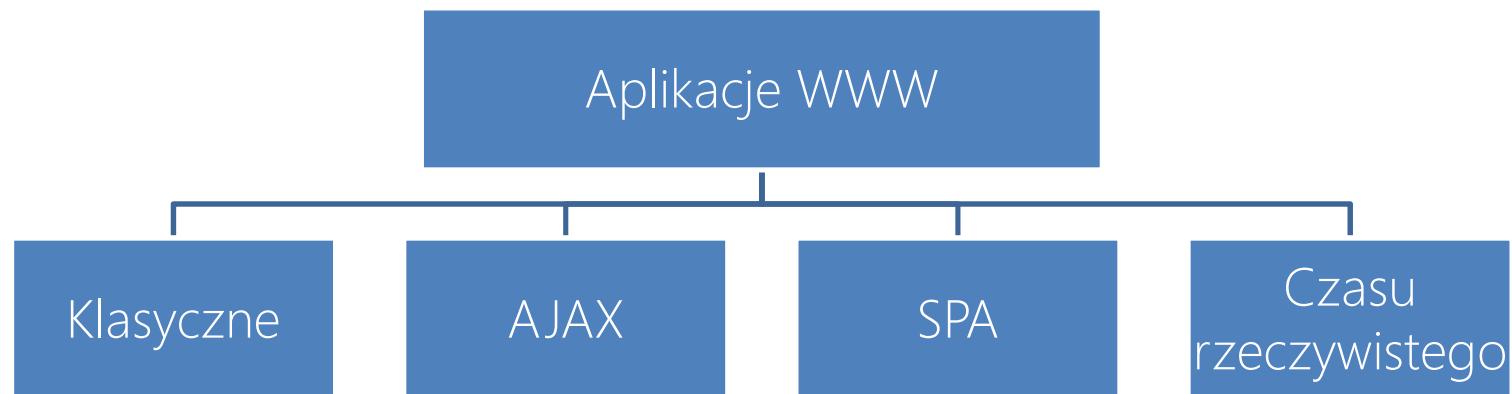
kod	opis słowny	znaczenie/zwrócony zasób
504	Gateway Timeout	Przekroczony czas bramy – serwer – spełniający rolę bramy lub pośrednika – nie otrzymał w ustalonym czasie odpowiedzi od wskazanego serwera HTTP, FTP, LDAP itp. lub serwer DNS jest potrzebny do obsługi zapytania
505	HTTP Version Not Supported	Nieobsługiwana wersja HTTP – serwer nie obsługuje bądź odmawia obsługi wskazanej przez klienta wersji HTTP
506	Variant Also Negotiates	
507	Insufficient Storage	Serwer nie jest w stanie zapisać danych związanych z wykonaniem zapytania

Status odpowiedzi - 500

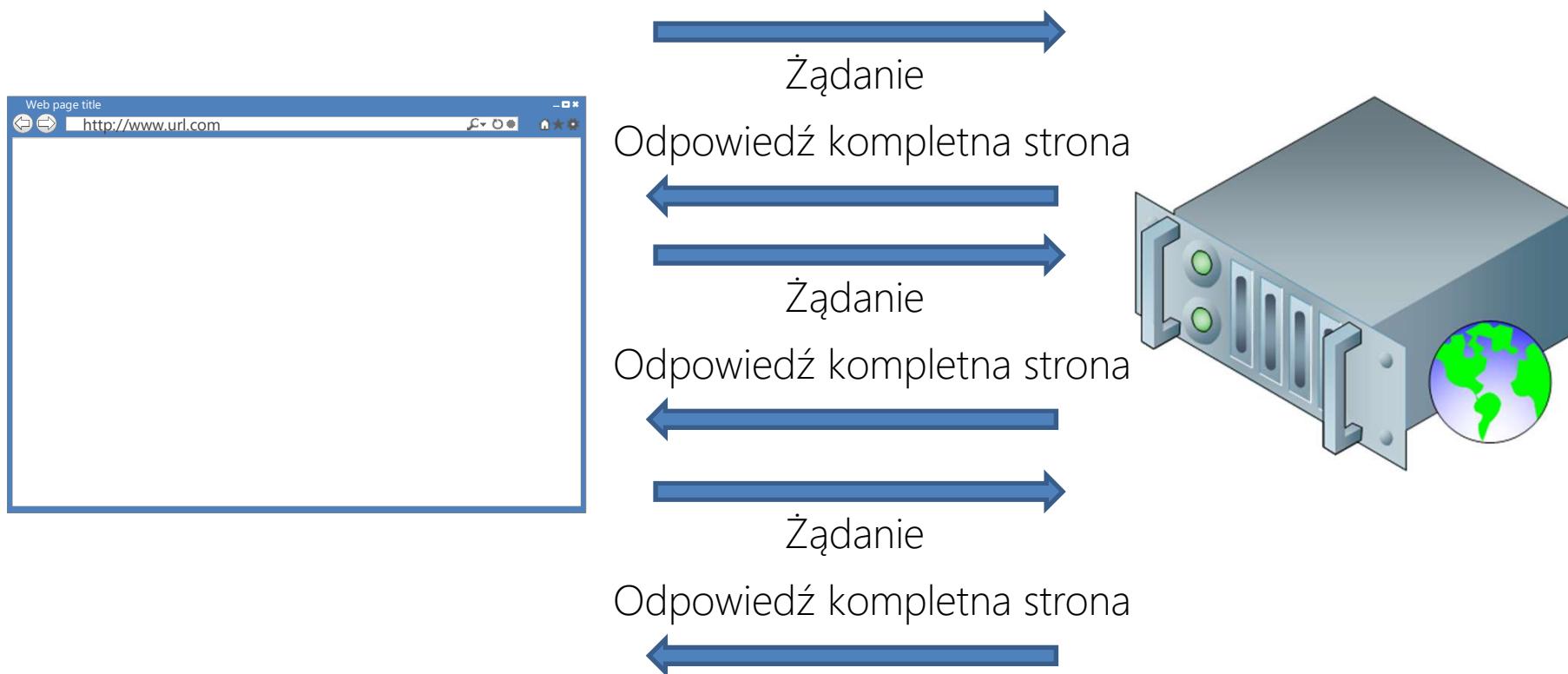
Kody błędu aplikacji klienta

kod	opis słowny	znaczenie/zwrócony zasób
508	Loop Detected	Serwer wykrył nieskończoną pętlę w trakcie przetwarzania zapytania
510	Not Extended	Brak rozszerzenia HTTP koniecznego do obsługi danego zapytania
511	Network Authentication Required	Wymagane uwierzytelnienie przed otrzymaniem dostępu do sieci. W zamyśle wykorzystywane przez pośredników kontrolujących dostęp do sieci (np.: wymaganie potwierdzenia zasad użytkowania przed udostępnieniem połączenia).

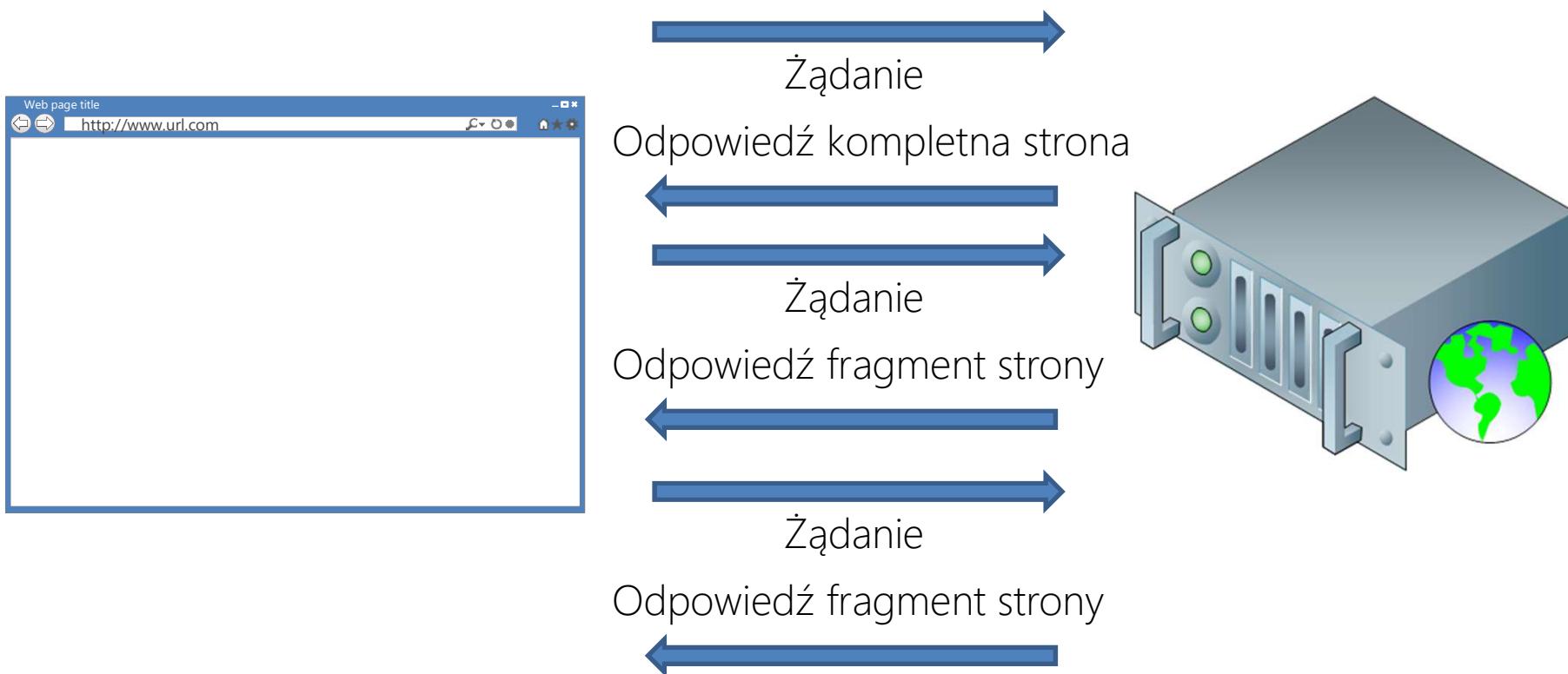
Aplikacje WWW



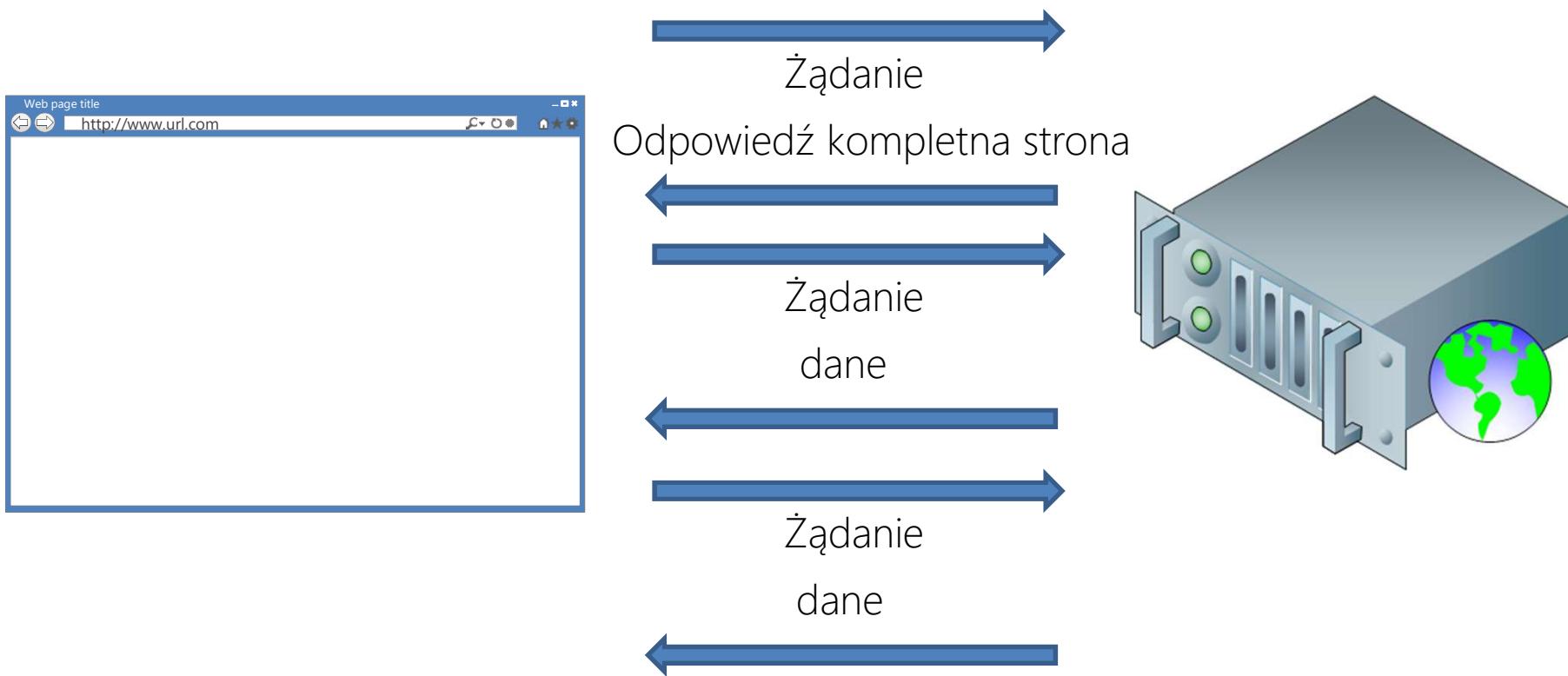
Klasyczne aplikacje WWW



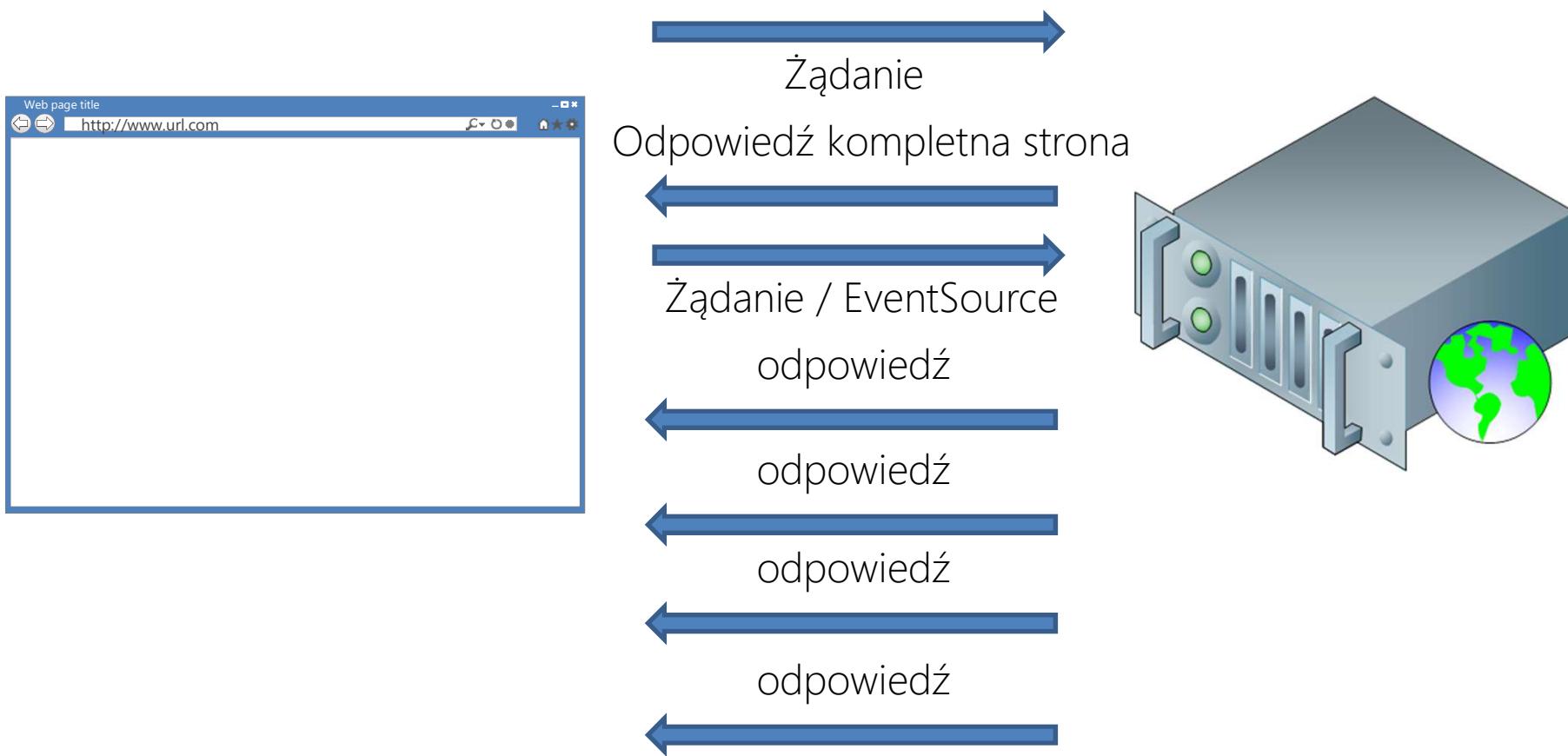
Aplikacje AJAX



Aplikacje SPA

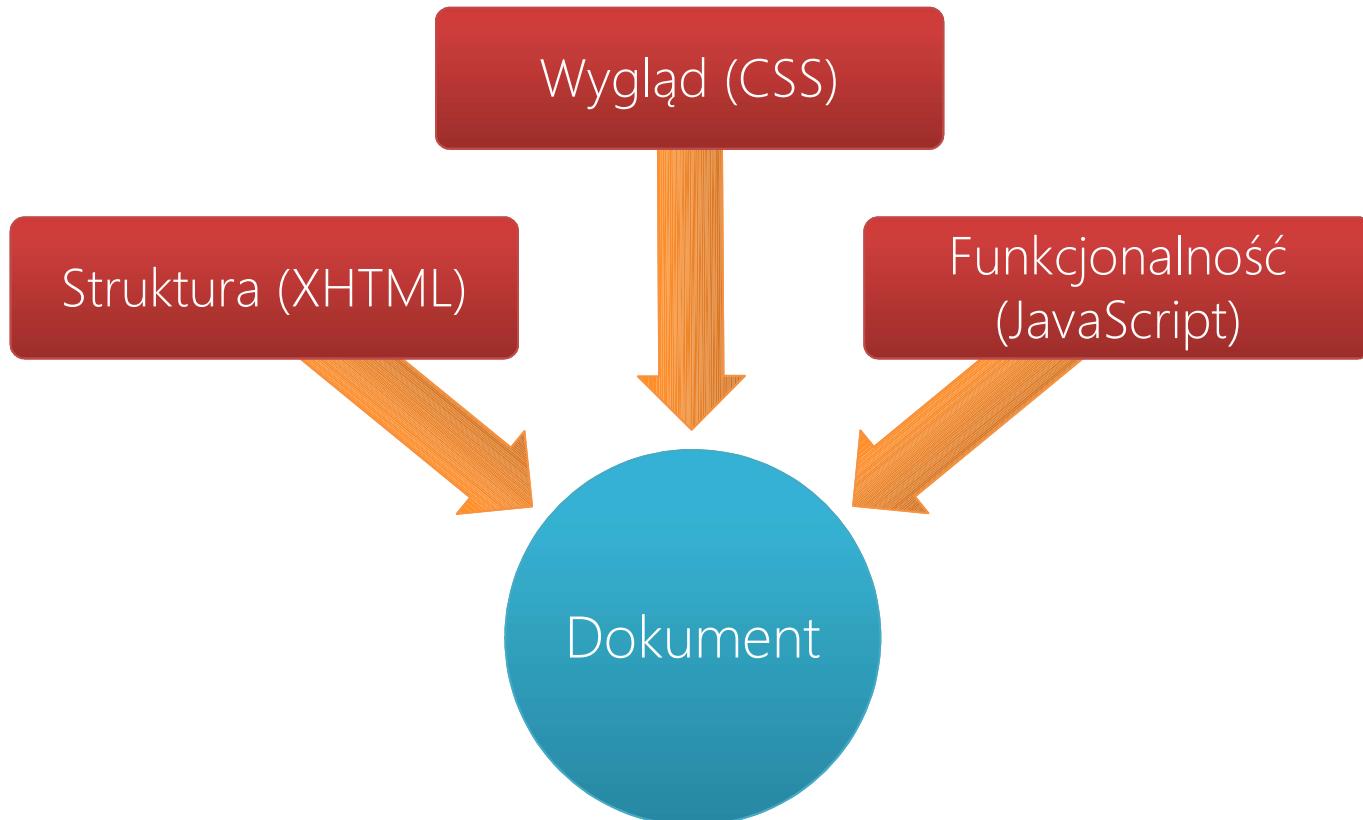


Aplikacje real-time



PODSTAWY TWORZENIA STRON INTERNETOWYCH

Składowe strony internetowej



Czym jest HTML 5?



- Jest to nowa wersja języka HTML
- Stanowi połączenie HTML + CSS 3.0 + API JavaScript

Znaczniki (1)

- Język HTML składa się z znaczników
- Znacznik zawarty jest pomiędzy parą nawiasów < i >
- Każdy znacznik musi być zamknięty.
- Każdy znacznik posiada swoje znaczenie

```
<div>
  <p>lorem ipsum... </p>
</div>
```

Znaczniki (2)

- Znaczniki mogą być zagnieżdżane
- Każdy znacznik może posiadać atrybuty
- Atrybuty umieszczamy w znaczniku otwierającym

```
<div>
    <p id="tresc">lorem ipsum....
        </p>
</div>
```

Dokument HTML

```
<!DOCTYPE html>

<html>
  <head>
    <title>Tytuł strony</title>
  </head>
  <body>
    Zawartość strony.....
  </body>
</html>
```

Deklaracja DOCTYPE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<!DOCTYPE html>
```

Nagłówek strony

Zawiera meta informacje nt. strony

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

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

Elementy HTML (1)

Podstawowe

- <!DOCTYPE>,<html>,<body>, <!-- -->

Struktura

- <h1>-<h6>,<p>,<div>,,
,<hr/>

Formatowania

- <acronym>,<abbr>,<address>,,<bdo>,<big>,<blockquote>,<code>,,<dfn>,,<i>,<ins>,<kbd>,<pre>,<q>,<samp>,<small>,,<sub>,<sup>,<tt>,<u>,<var>

Formularze

- <form>,<input>,<textarea>,<button>,<select>,<optgroup>,<option>,<label>,<fieldset>,<legend>

Tabele

- <table>,<caption>,<th>,<tr>,<td>,<thead>,<tbody>,<tfoot>,<col />,<colgroup>

Elementy HTML (2)

Listy

- , , , <dl>, <dt>, <dd>

Programowanie

- <script>, <noscript>, <object>, <param />

Obrazy

- , <map>, <area />

Metadane

- <head>, <title>, <meta>

Style

- <style>

Linki

- <a>, <link />

Elementy strukturalne

Do elementów strukturalnych języka HTML zaliczamy:

- div-y
 - paragrafy
 - nagłówki
 - span-y
-
- elementy blokowe
- element liniowy

Elementy te mogą zawierać zarówno tekst jak i inne elementy HTML

Nowe elementy strukturalne

HTML 5 wprowadza następujące nowe elementy

- Article
- Section
- HGroup
- Header
- Footer
- Section
- Address
- Nav
- Aside

```
<!DOCTYPE html>
<html>
<body>
  <header>
    <h1>Nagłówek</h1>
    <nav><ul><li>nawigacja</li></ul></nav>
  </header>
  <section>
    <article><p>Jakaś zawartość</p></article>
  </section>
  <footer><p>informacje w stopce</p></footer>
</body>
</html>
```

Po co?

```
<!DOCTYPE html>
<html>
<body>
  <header>
    <h1>Nagłówek</h1>
    <nav>
      <ul>
        <li>nawigacja</li>
      </ul>
    </nav>
  </header>
  <section>
    <article>
      <p>Jakaś zawartość</p>
    </article>
  </section>
  <footer>
    <p>informacje w stopce</p>
  </footer>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
  <div id="header">
    <h1>Nagłówek</h1>
    <div id="nav">
      <ul>
        <li>nawigacja</li>
      </ul>
    </div>
  </div>
  <div class="section">
    <div class="article">
      <p>Jakaś zawartość</p>
    </div>
  </div>
  <div id="footer">
    <p>informacje w stopce</p>
  </div>
</body>
</html>
```

Linki (1)



Linki (2)

- zewnętrzne

```
<a href="adres_strony.html">link</a>
```

- wewnętrzne

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

...

```
<div id="nazwa">Zawartość do linku</div>
```

- do określonego miejsca na stronie

```
<a href="adres_strony.html#nazwa">link</a>
```

Rysunki

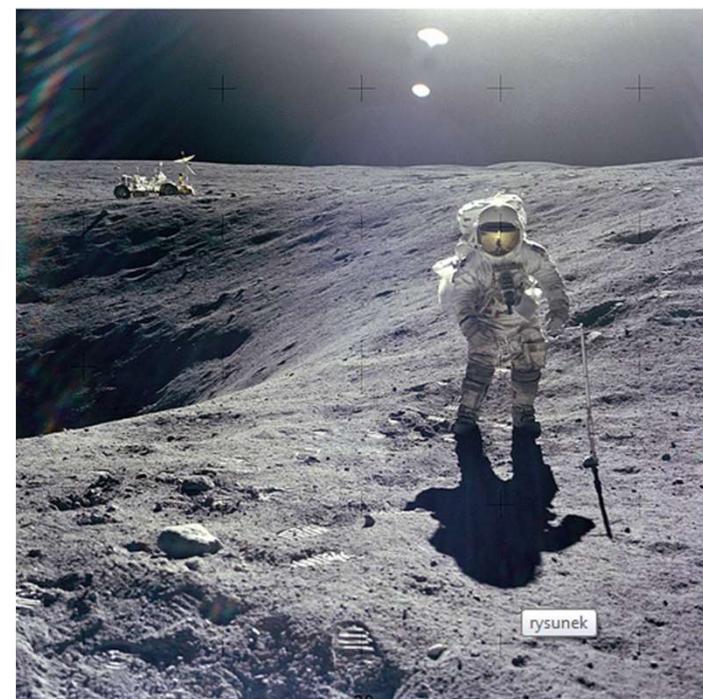
- Znacznik ``
- Brak znacznika zamykającego

```

```

```

```



Tabele

```
<table>
<caption>Podpis</caption>
<thead>
  <tr><th>Nagłówek 1</th><th>Nagłówek 2</th><th>Nagłówek 3</th></tr>
</thead>
<tbody>
  <tr><td>11</td><td>12</td><td>13</td></tr>
  <tr><td rowspan="2">21</td><td>22</td><td>23</td></tr>
  <tr><td colspan="2">32</td></tr>
</tbody>
</table>
```

Podpis

Nagłówek 1	Nagłówek 2	Nagłówek 3
11	12	13
21	22	23
	32	

Listy

- Wyróżniamy trzy rodzaje list:
 - Punktowane ()
 - Numerowane ()
 - Definicyjne (<dl>)
- Mogą zawierać jeden lub więcej pozycji ()

Formularze

```
<form action="akcja.php" method="post" enctype="multipart/form-data">
<div>
    <label for="imie">Imię i nazwisko</label>
    <input name="imie" id="imie" maxlength="20" />
</div>
<div>
    <label for="kategoria">Kategoria</label>
    <select name="kategoria" id="kategoria">
        <option value="-1">Wybierz</option>
        <option value="1">Student</option>
        <option value="2">Pracownik</option>
    </select>
</div>
<div>
    <button type="submit">Dodaj</button>
    <button type="reset">Wyczyść formularz</button>
</div>
</form>
```

Formularze

Znacznik	Typ	Opis
<form>		Wyznacza ramy formularza
<input>	<code>type="text"</code>	Pole do wprowadzania jednej linii tekstu
	<code>type="password"</code>	Pole do wprowadzania hasła
	<code>type="checkbox"</code>	Pole wielokrotnego wyboru
	<code>type="radio"</code>	Pole jednokrotnego wyboru
	<code>type="hidden"</code>	Pole ukryte
	<code>type="file"</code>	Pole wyboru pliku
	<code>type="button"</code>	Przycisk
	<code>type="submit"</code>	Przycisk zatwierdzania formularza
	<code>type="reset"</code>	Przycisk resetowania formularza
	<code>type="date"</code>	Pole do wprowadzania daty
	<code>type="datetime"</code>	Pole do wprowadzania daty i czasu
	<code>type="email"</code>	Pole do wprowadzania adresu email

Formularze

Znacznik	Typ	Opis
<input>	<code>type="time"</code>	Pole do wprowadzania czasu
	<code>type="url"</code>	Pole do wprowadzania adresu url
	<code>type="number"</code>	Pole do wprowadzania liczb
	<code>type="color"</code>	Pole wyboru koloru
<select>		Lista rozwijana
	<code>multiple</code>	Lista wielokrotnego wyboru
<option>		Pozycja na liście
<textarea>		Pole do wprowadzania wielu linii tekstu
<button>		Przyciski
<fieldset>		Grupowanie pól
<label>		Etykieta

Walidacja danych

HTML 5 pozwala na walidację danych wprowadzonych do formularzy.

```
<input type="text" pattern="\d\d-\d\d\d\d" />  
<input type="text" required />  
<input type="number" min="10" max="20" />
```

Atrybut	Atrybut
novalidate (F)	min/max (I)
formnovalidate (I)	required (I)
step (I)	minlength, maxlength (I)

Formularze

```
<form action="akcja.php" method="post" enctype="multipart/form-data">
<div>
    <label for="imie">Imię i nazwisko</label>
    <input name="imie" id="imie"
        maxlength="30" required placeholder="Podaj imię i nazwisko"/>
</div>
<div>
    <label for="kategoria">Kategoria</label>
    <select name="kategoria" id="kategoria">
        <option value="-1">Wybierz</option>
        <option value="1">Student</option>
        <option value="2">Pracownik</option>
    </select>
</div>
<div>
    <button type="submit">Dodaj</button>
    <button type="reset">Wyczyść formularz</button>
</div>
</form>
```

Formularze

Imię i nazwisko	<input type="text" value="Podaj imię i nazwisko"/>
Kategoria	<input type="text" value="Wybierz..."/> ! Wypełnij to pole.
<button>Dodaj</button> <button>Wyczyść formularz</button>	

HTML 5 – video/audio

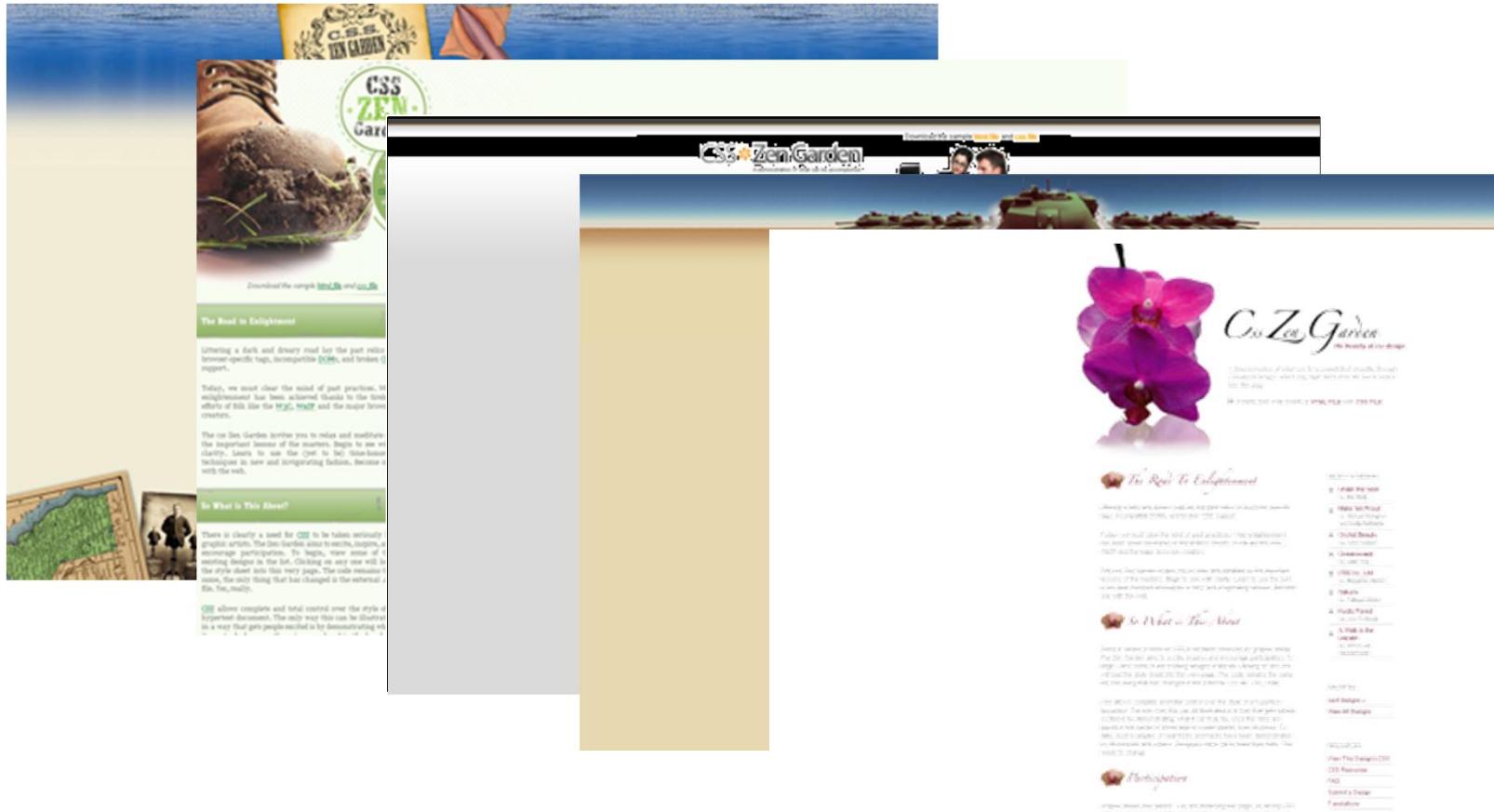
```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Hobbit trailer</title>
  </head>
  <body>
    <video src="The-Official-Hobbit-Movie-Trailer.mp4"
           controls poster="poster.png" muted>
      Plik video nie może być wyświetlony.
    </video>
    <audio src="cicho.mp3" controls muted>
      Plik audio nie może być odtworzony.
    </audio>
  </body>
</html>
```

HTML 5 – video / audio

Atrybut	Opis
autoplay	Automatyczne odtwarzanie pliku po załadowaniu
preload	Czy załadować plik automatycznie? none, metadata, auto
controls	Czy wyświetlać kontrolki do obsługi czy nie?
loop	Czy zapętlić wyświetlanie?
poster	Określa rysunek, który będzie wyświetlany przed odtworzeniem pliku
height, weight	Określają wymiary obrazu.
muted	Czy wyłączyć dźwięk?
src	źródło pliku video

Czym są kaskadowe arkusze stylów

Kaskadowe arkusze stylów (ang. Cascade Style Sheets) pozwalają na zdefiniowanie wyglądu i układu strony internetowej.



Składnia CSS

- Informacje o wyglądzie strony są zapisywane w formie reguł
- Reguła zapisywana jest w postaci:

```
selektor {  
    właściwość : wartość;  
    właściwość : wartość;  
    właściwość : wartość;  
}
```

- Język jest czuły na wielkość znaków.

Miejsce umieszczania CSS

miejsce umieszczenia reguł CSS

w tagu HTML

w części HEAD

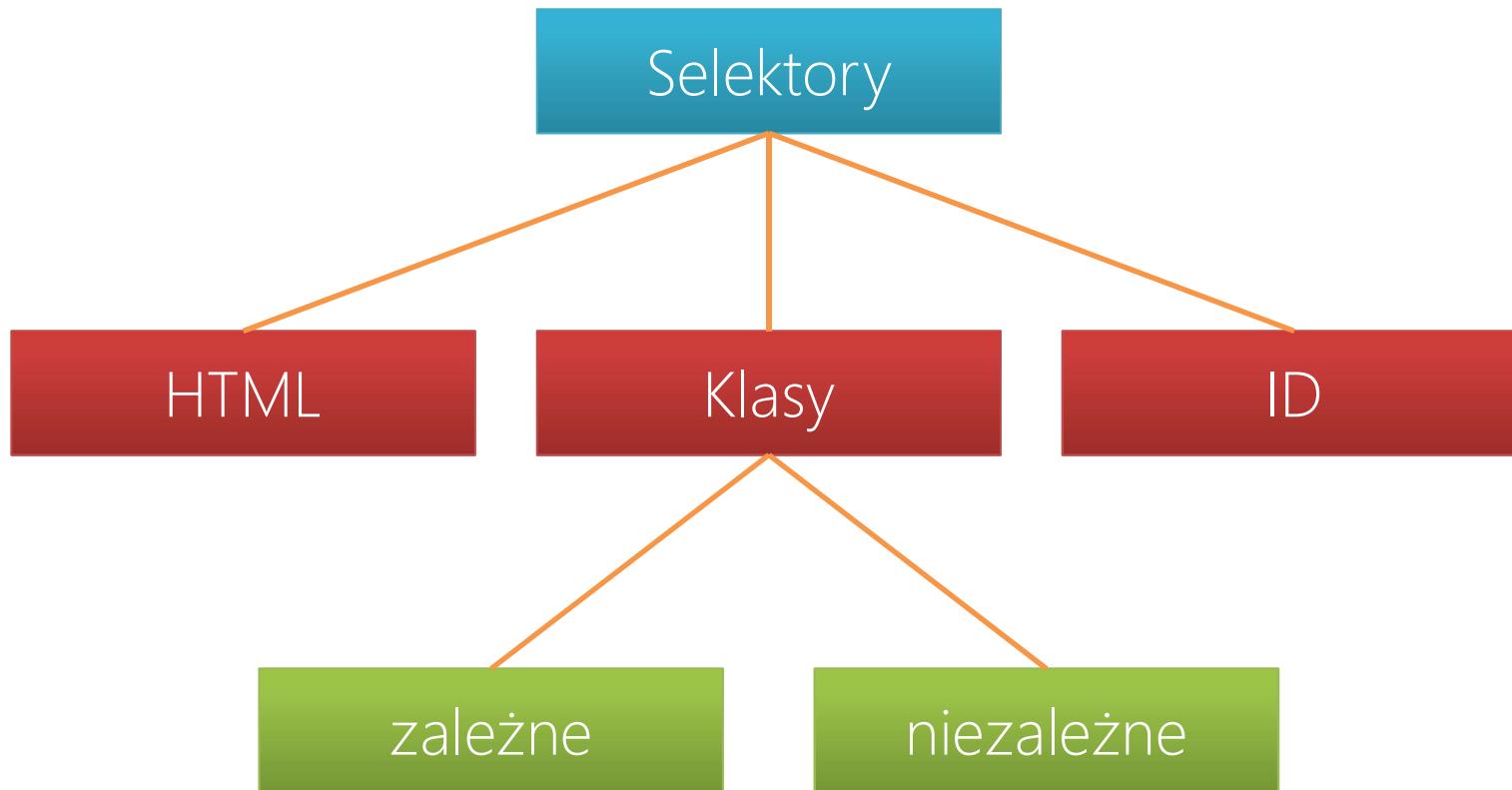
w osobnym pliku

```
...  
<p style="color:blue;">  
...  
</p>  
...  
<p style="color:blue;">  
...  
</p>  
...
```

```
<head>  
<style type="text/css">  
    p {color: blue;}  
</style>  
</head>
```

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

Selektory proste



Selektory proste - HTML

Selektory HTML dopasowują się do określonego rodzaju elementu HTML

```
h1 {color:blue; text-decoration:underline;}  
p {color:black; text-style:italic;}
```

Selektory proste – klasy niezależne

Każdy element HTML może mieć przypisaną klasę, która pozwala łączyć kilka różnych elementów w grupę.

Selektory klasy niezależne dopasowują się do grupy elementów z przypisaną określoną klasą

```
.news {color:blue; text-decoration:underline;}
```

```
...
```

```
<h1 class="news">Jakiś tytuł 1</h1>
<p class="news">Jakiś paragraf 1</p>
<h1>Jakiś tytuł 2</h1>
<p>Jakiś paragraf 2</p>
```

```
...
```

Selektory proste – klasy zależne

Selektory klasy zależne dopasowują się do grupy elementów tego samego typu z przypisaną określona klasą.

```
h1.news {color:blue; text-decoration:underline;}  
p.news {color:black; text-style:italic;}
```

...

```
<h1 class="news">Jakiś tytuł 1</h1>  
<p class="news">Jakiś paragraf 1</p>  
<h1>Jakiś tytuł 2</h1>  
<p>Jakiś paragraf 2</p>
```

...

Selektory proste – ID

Każdy element HTML może mieć przypisany identyfikator, który pozwala jednoznacznie ten element na stronie.

Selektory ID dopasowują się do tylko do elementu z określonym identyfikatorem.

```
#newsSuperWazny {color:blue; text-decoration:underline;}  
#news {color:black; text-decoration:underline;}
```

...

```
<h1 id="newsSuperWazny">Jakiś tytuł 1</h1>  
<p id="news">Jakiś paragraf 1</p>  
<h1>Jakiś tytuł 2</h1>  
<p>Jakiś paragraf 2</p>
```

...

Grupowanie selektorów

```
sel1,sel2,... {  
    właściwość : wartość;  
    właściwość : wartość;  
    właściwość : wartość;  
}
```

```
h1,h2,h3 {color:blue; text-decoration:underline;}
```

Selektory pochodne

```
p span {color:orange; text-decoration:underline;}
```

...

```
<p id="news">  
Jakiś paragraf 1 <span> bardzo <span> interesujące  
</span> cytowanie </span> Dalsza część paragrafu 1  
<span>kolejne interesujące cytowanie</span> Dalsza  
część paragrafu 1  
</p>
```

...

Jakiś paragraf 1 bardzo interesujace cytowanie Dalsza część
paragrafu 1 kolejne interesujace cytowanie Dalsza część
paragrafu 1

Selektory dzieci

```
p span {color:orange;}  
p>span {color:red;}
```

...

```
<p id="news">  
Jakiś paragraf 1 <span> bardzo <span> interesujące  
</span> cytowanie </span> Dalsza część paragrafu 1  
<span>kolejne interesujące cytowanie</span> Dalsza  
część paragrafu 1  
</p>
```

...

Jakiś paragraf 1 **bardzo interesujące cytowanie** Dalsza część
paragrafu 1 **kolejne interesujące cytowanie** Dalsza część
paragrafu 1

Selektory dzieci

```
p span {color:orange;}  
p>span {color:red;}
```

...

```
<p id="news">  
Jakiś paragraf 1 <span> bardzo <span> interesujące  
</span> cytowanie </span> Dalsza część paragrafu 1  
<span>kolejne interesujące cytowanie</span> Dalsza  
część paragrafu 1  
</p>
```

...

Jakiś paragraf 1 **bardzo interesujące cytowanie** Dalsza część
paragrafu 1 **kolejne interesujące cytowanie** Dalsza część
paragrafu 1

Selektory atrybutów (1)

Są dopasowywane do elementów posiadających określone atrybuty lub ich wartości

- **[atrybut]** – reguła jest dopasowana, gdy element ma przypisaną wartość dla atrybutu "atrybut"
- **[atrybut=wartość]** – reguła jest dopasowywana, gdy atrybut "atrybut" ma dokładnie wartość "wartość".
- **[atrybut~=wartość]** – reguła jest dopasowana, gdy atrybut "atrybut" ma przypisaną listę wartości oddzielonych spacjami, z których jedna jest równa "wartość"
- **[atrybut|=wartość]** – reguła jest dopasowana, gdy atrybut ma wartość "wartość" lub, gdy rozpoczyna się od wartości "wartość" po której występuje znak "-".

Selektory atrybutów (2)

```
input[readonly] {color:blue;}
```

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

```
a[href="http://onet.pl"] {color:blue;}
```

```
<a href="http://onet.pl">onet.pl</a>
```

```
img[alt~="opis"] {border:1px solid red;}
```

```
<img alt="jakiś opis" />
```

Pseudo-klasy

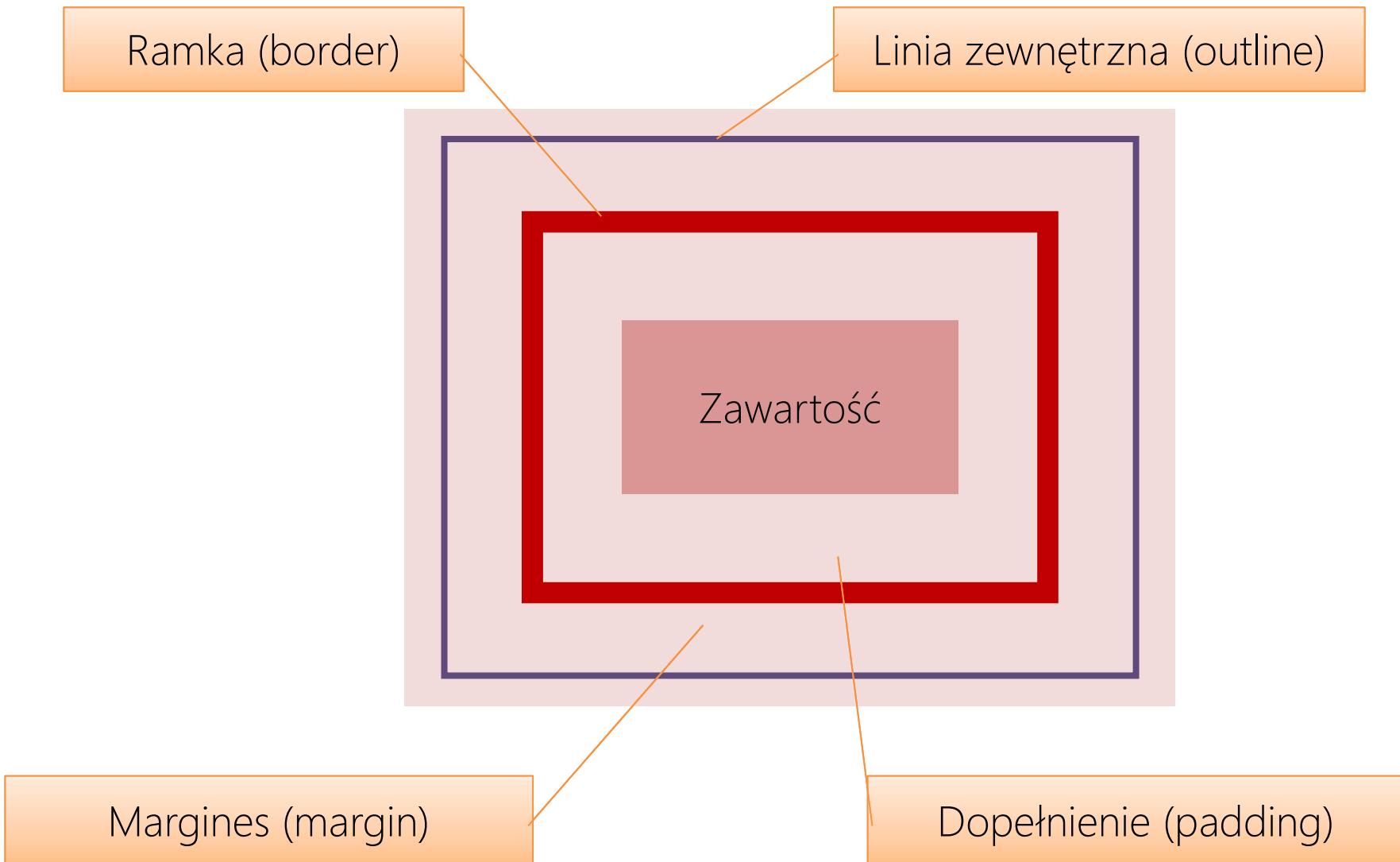
```
a:link {color:blue;}  
a:visited {color:orange;}  
a:hover {color:red;}  
a:active {color:green;}
```

...

```
<a href="http://onet.pl/">onet.pl</a>
```

...

Model pudełkowy CSS (1)



Przykład (1)

 Company Name

Moja pierwsza fajna strona

Menu-1 Menu-2 Menu-3 Menu-4 Menu-5 Menu-6

Kolumna 1

Text content: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac. Fusce congue, sapien at iaculis aliquet, ligula mi commodo nulla, pretium tempor erat leo eget ipsum. Sed eu orci feugiat, dapibus nibh eu, scelerisque mi.

Text content: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

Kolumna 2



Text content: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

Kolumna 3

Text content: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

Najfajniejsza stopka świata

Przykład (2)

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css" />
        <meta charset="utf-8" />
    </head>
    <body>
        <header> </header>
        <div class="part columns">
            <div class="column"> </div>
            <div class="column"> </div>
            <div class="column"> </div>
        </div>
        <footer>
            </footer>
    </body>
</html>
```

Przykład (3)

Moja pierwsza fajna strona

- [Menu-1](#)
- [Menu-2](#)
- [Menu-3](#)
- [Menu-4](#)
- [Menu-5](#)
- [Menu-6](#)

```
<header>
    <div class="part">
        <h1>Moja pierwsza fajna strona</h1>
        <nav>
            <ul>
                <li><a href="#">Menu-1</a></li>
                <li><a href="#">Menu-2</a></li>
                <li><a href="#">Menu-3</a></li>
                <li><a href="#">Menu-4</a></li>
                <li><a href="#">Menu-5</a></li>
                <li><a href="#">Menu-6</a></li>
            </ul>
        </nav>
    </div>
</header>
```

Przykład (4)



```
header {background-color:#ddd; }

header>div
{
    height:110px; padding:25px 0 0 160px;
    background: transparent url(logo.png) no-repeat 0 8px;
    box-sizing: border-box;
}

.part { width:1024px; margin:auto; }
```

Model pudełkowy – ustalanie wielkości elementu

Właściwości width i height pozwalają na ustalenie wielkości elementu

Właściwość nie jest dostępna dla niektórych rodzajów pudełek

Jednostki i kolory w CSS

Jednostki CSS

względne	bezwzględne
em	mm
ex	cm
px	in
	pt
	pc

Kolory CSS

notacja heksadecymalna:
`color: #808000;`

notacja dziesiętna:
`color: rgb(128,128,0);`

słowa kluczowe:
`color:olive`

Podstawowe właściwości CSS

Dekorowanie tekstu

`text-decoration: underline;`

`none | line-through | underline | overline`

Kolor tekstu

`color: #FF0000;`

Podstawowe właściwości CSS

Tło

background: background-color background-image
background-repeat background-attachment
background-position

```
background: yellow url(tlo.jpg) repeat-x fixed 0 left;
```

Ramka

border: border-width border-style border-color

```
border: 1px dotted red;
```

Przykład (5) / Model pudełkowy - display



```
nav li {display:inline-block; width:70px;}
```

Właściwość display pozwala na zmianę domyślnego sposobu traktowania elementu przez przeglądarkę

Przykładowe wartości dla display:

`none, inline, block, list-item, inline-block`

Przykład (6) / padding, margin



```
nav>ul {margin:0; padding:0;}
```

```
nav>ul a { text-decoration:none; color:white;  
background-color:green; padding:5px;  
border-radius: 3px;  
}
```

```
nav>ul a:hover {background-color:orange;}
```

Dopełnienie (padding)

```
padding:10px;  
padding:10px 5px;  
padding:10px 5px 10px;  
padding:10px 5px 10px 5px;
```

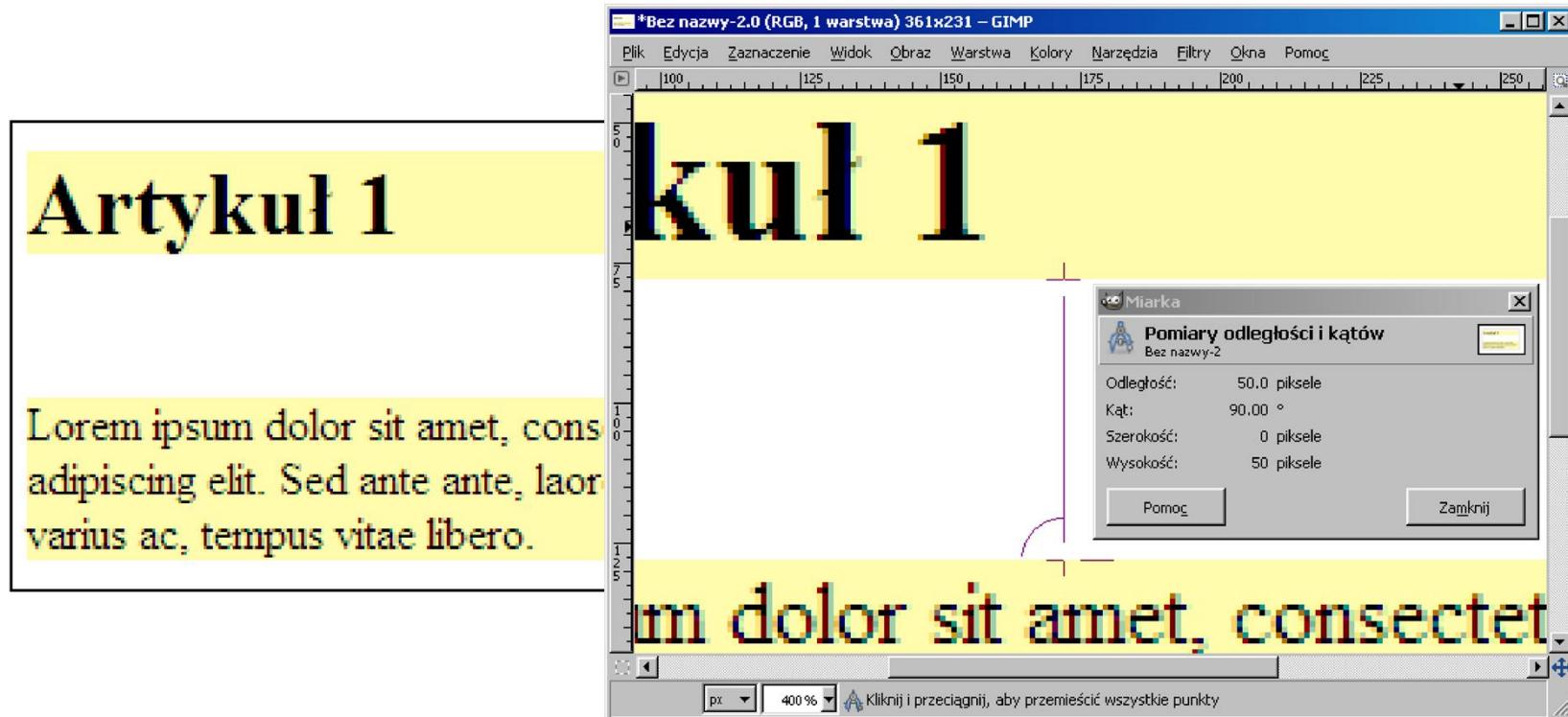
Margines (margin)

```
margin:10px;  
margin:10px 5px;  
margin:10px 5px 10px;  
margin:10px 5px 10px 5px;
```

Łączanie marginesów

```
<!DOCTYPE html>
<html>
<head>
    <style>
        h1 {margin:10px 5px 40px 5px;}
        p  {margin: 50px 5px 10px 5px;}
    </style>
</head>
<body>
    <h1>Artykuł 1</h1>
    <p>
        Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Sed ante ante, laoreet
        porttitor varius ac, tempus vitae libero.
    </p>
</body>
</html>
```

Łączenie marginesów



Przykład (6) / padding, margin



```
.part { width:1024px; margin:auto; position:relative; }  
nav {position:absolute; bottom:9px; right:0px; }
```

Właściwość `position` pozwala na zmianę domyślnego sposobu pozycjonowania elementu.

Ustalenie pozycji dokonujemy za pomocą właściwości:
`top`, `left`, `bottom`, `right`

Wartości dla `position`:

`static`, `relative`, `absolute`, `fixed`

position

```
position: static;
```

```
DIV #1  
position:static;
```

```
DIV #2  
position:static;
```

```
DIV #3  
position:static;
```

```
DIV #2  
position:absolute  
left:0px; top:0px;
```

```
position: static;
```

```
DIV #1  
position:static;
```

```
DIV #3  
position:static;
```

position

```
position: static;
```

```
    DIV #1  
    position:static;
```

```
    DIV #2  
    position: relative  
    left: 30px;  
    top: 30px;
```

```
    DIV #3  
    position: static;
```

```
position: static;
```

```
    DIV #1  
    position:static;
```

```
    DIV #3  
    position:static;
```

```
    DIV #2  
    position: fixed  
    left: 30px;  
    top: 160px;
```

position

DIV #2
position: absolute;
left:0px; top:0px;

position: static;

DIV #1
position:static;

DIV #3
position:static;

position: relative;

DIV #1
position:static;

DIV #3
position:static;

DIV #2
position: absolute;
left:30px;
top:130px;

Przykład (7)

The screenshot displays a website template with a header, two columns of content, and a large central dark area.

Header:

- Logo: A blue circle containing a white icon of a book or document.
- Text: "Company Name" next to the logo.
- Page Title: "Moja pierwsza fajna strona"
- Menu: A horizontal menu bar with six items: "Menu-1", "Menu-2", "Menu-3", "Menu-4", "Menu-5", and "Menu-6".

Kolumna 1 (Left Column):

Kolumna 1

Placeholder text for Kolumna 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac. Fusce congue, sapien at iaculis aliquet, ligula mi commodo nulla, pretium tempor erat leo eget ipsum. Sed eu orci feugiat, dapibus nibh eu, scelerisque mi.

Placeholder text for Kolumna 1: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż więcej](#)

Kolumna 2 (Right Column):

Kolumna 2

A large, solid dark brown rectangular area, likely a placeholder for additional content or a background image.

Przykład (8)

```
<div class="part columns">
    <div class="column">
        <article>
            <h2>Kolumna 1</h2>
            <p>Lorem ipsum dolor sit amet, ...</p>
            <div><a href="#">Pokaż więcej</a></div>
        </article>
    </div>
    <div class="column">
        <article>
            <h2>Kolumna 2</h2>
            
            <p>Lorem ipsum dolor sit amet, ...</p>
            <div><a href="#">Pokaż więcej</a></div>
        </article>
    </div>
    <div class="column">
        <article>
            <h2>Kolumna 3</h2>
            <p>Lorem ipsum dolor sit amet, ...</p>
            <div><a href="#">Pokaż więcej</a></div>
        </article>
    </div>
</div>
```

Przykład (9) / float, clear



```
.column {width:31%; float:left; padding:0px 10px;  
border-right:1px solid black;}  
.column:last-of-type {float:right; border-right:none;}  
.columns {overflow:hidden; padding:20px 0;}
```

Właściwość float pozwala na zmianę sposobu opływania elementów.

Właściwość clear wyłącza efekt opływania

Przykład (10) / float, clear

```
.column {width:31%; float:left; padding:0px 10px;  
         border-right:1px solid black;}  
.column:last-of-type {float:right; border-right:none;}  
.columns {overflow:hidden; padding:20px 0;}
```

Właściwość float pozwala na zmianę sposobu opływania elementów.

Właściwość clear wyłącza efekt opływania

float: left, right, none

`clear:left, right, both, none`

Przykład (11) / float, clear

```
.column {width:31%; float:left; padding:0px 10px;  
         border-right:1px solid black;}  
.column:last-of-type {float:right; border-right:none;}  
.columns {overflow:hidden; padding:20px 0;}
```

Właściwość float pozwala na zmianę sposobu opływania elementów.

Właściwość clear wyłącza efekt opływania

float: left, right, none

`clear:left, right, both, none`

float, clear

```
<body>
  <div id="pierwszy">
    
    <span>Lorem ipsum dolor sit amet, consectetur
    adipiscing elit. Sed ante ante, laoreet porttitor
    varius ac, tempus vitae libero.</span> Aliquam erat
    volutpat.
  </div>
  <div id="drugi">
    Lorem ipsum dolor sit amet, consectetur adipiscing
    elit. Sed ante ante, laoreet porttitor varius ac,
    tempus vitae libero. Aliquam erat volutpat. Ut
    ornare, augue sed mollis aliquam, sem tortor semper
    nisi, quis egestas sem ...
  </div>
</body>
```

float, clear



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur. Cras sagittis, orci a imperdiet posuere, orci leo accumsan justo, id lobortis ipsum sapien ut libero. Vestibulum euismod, nunc at consequat pellentesque, lacus ligula aliquet orci, at fringilla augue dui a nulla.

float, clear

```

```



• Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur.

looreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur. Cras sagittis, orci a imperdiet posuere, orci leo accumsan justo, id lobortis ipsum sapien ut libero. Vestibulum euismod, nunc at consequat pellentesque, lacus ligula aliquet orci, at fringilla augue dui a nulla.

float, clear



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur. Cras sagittis, orci a imperdiet posuere, orci leo accumsan justo, id lobortis ipsum sapien ut libero. Vestibulum euismod, nunc at consequat pellentesque, lacus ligula aliquet orci, at fringilla augue dui a nulla.

float, clear

```
<div style="clear:both"/>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed ante ante, laoreet porttitor varius ac, tempus vitae libero. Aliquam erat volutpat. Ut ornare, augue sed mollis aliquam, sem tortor semper nisi, quis egestas sem massa eget magna. Quisque feugiat, diam sit amet venenatis condimentum, nulla diam ultricies nunc, non bibendum ante sapien nec enim. Vestibulum commodo aliquam turpis, quis volutpat dui porta ac. Phasellus fermentum sollicitudin mi non venenatis. Suspendisse potenti. Nam suscipit sagittis consectetur. Cras sagittis, orci a imperdiet posuere, orci leo accumsan justo, id lobortis ipsum sapien ut libero. Vestibulum euismod, nunc at consequat pellentesque, lacinia ligula aliquet orci, at fringilla augue dui a nulla.

overflow

`overflow`

`overflow-x`

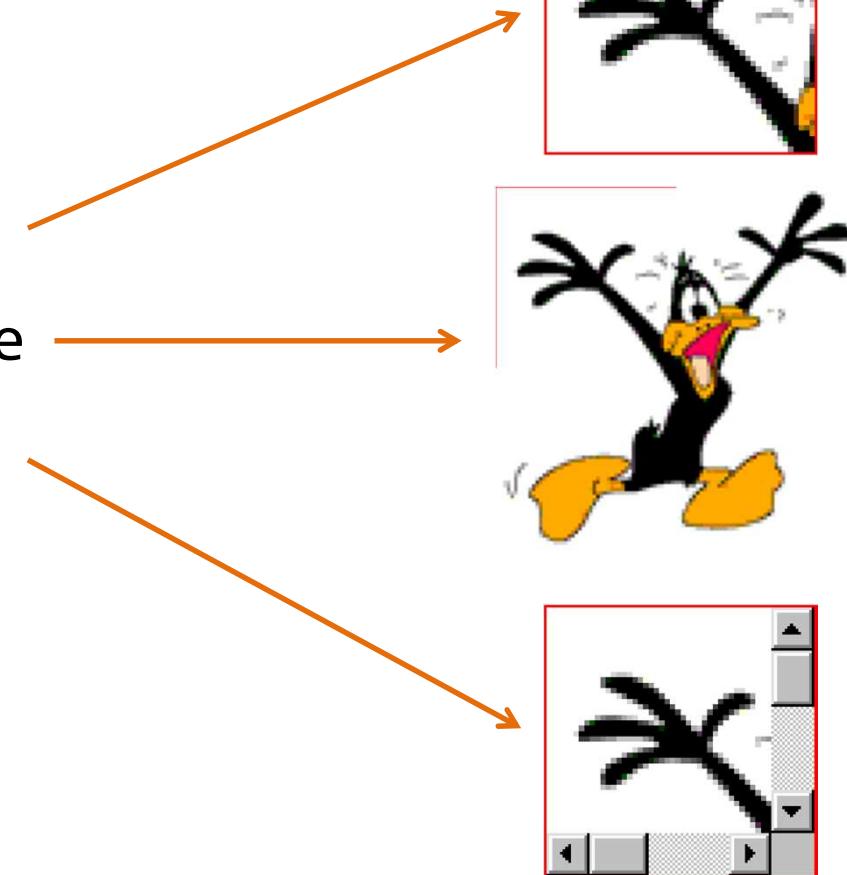
`overflow-y`

- `hidden`

- `visible`

- `scroll`

- `auto`



Przykład (12)

Kolumna 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac. Fusce congue, sapien at iaculis aliquet, ligula mi commodo nulla, pretium tempor erat leo eget ipsum. Sed eu orci feugiat, dapibus nibh eu, scelerisque mi.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż więcej](#)

Najfajniejsza stopka świata

Kolumna 2



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż więcej](#)

Kolumna 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż więcej](#)

```
article img {width:100%;}  
footer {height:50px;}  
footer {background-color:#ddd;}
```

Przykład (13)



Company Name

Moja pierwsza fajna strona

Menu-1 Menu-2 Menu-3 Menu-4 Menu-5 Menu-6

Kolumna 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac. Fusce congue, sapien at iaculis aliquet, ligula mi commodo nulla, pretium tempor erat leo eget ipsum. Sed eu orci feugiat, dapibus nibh eu, scelerisque mi.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

Najfajniejsza stopka świata

Kolumna 2



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

Kolumna 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ut eleifend tortor. Duis sed euismod eros. Morbi quis nunc sem. Quisque at malesuada ante, ac euismod leo. Fusce vel felis scelerisque, faucibus nulla in, ultrices elit. Vivamus ultricies aliquam tellus. Aenean suscipit porttitor turpis, sed convallis eros dignissim ac.

[Pokaż wiecej](#)

```
@font-face{  
    font-family: Raleway;  
    src: url(Raleway-Regular.ttf);  
}  
body {  
    font-family:Raleway;  
    margin:0;  
    padding:0;  
}
```

Podstawowe właściwości CSS

Ustawienia czcionek

`font: <font-weight> <font-style> <font-variant>
<font-size>/<line-height> <font-family>`

`p {font: normal italic 2em/1.5 times,serif;}`

Wyrównanie tekstu w poziomie:

`text-align:left; left|center|right|justify`

flexbox

Screenshot of a web browser displaying a flexbox layout example. The page title is "Moja pierwsza fajna strona". The header includes a logo, company name, and a menu bar with six items: Menu-1 through Menu-6.

The content is organized into three columns:

- Kolumna 1**: Contains two blocks of placeholder text (Lorem ipsum) and a "Pokaz wiecej" link.
- Kolumna 2**: Displays a close-up image of several colored pencils.
- Kolumna 3**: Contains two blocks of placeholder text (Lorem ipsum) and a "Pokaz wiecej" link.

A footer bar at the bottom contains the text "Najfajniejsza stopka świata".

przyklad2.html

file:///F/Dydaktyka/Czestochowa/2016-17/psi/przykłady/przyklad2.html

Company Name Moja pierwsza fajna strona

Menu-1 Menu-2 Menu-3 Menu-4 Menu-5 Menu-6

Kolumna 1

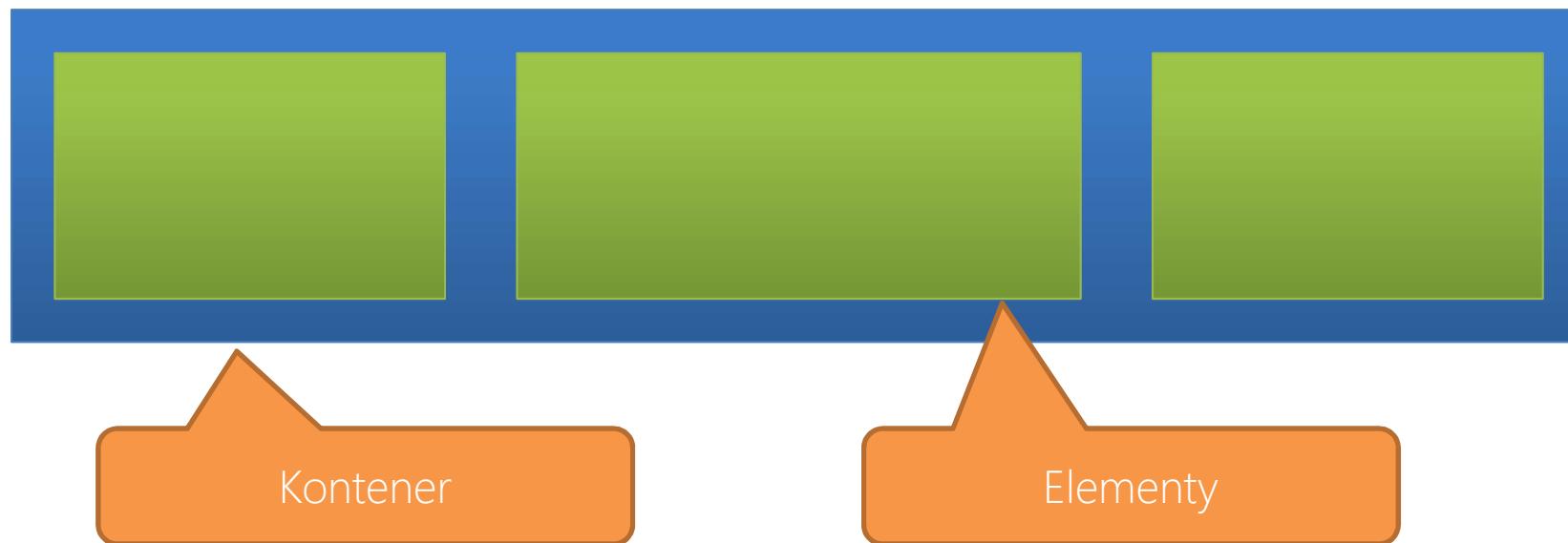
Kolumna 2

Kolumna 3

Najfajniejsza stopka świata

flexbox

Flexbox jest nowym bardziej efektywnym sposobem układania elementów wewnątrz kontenera nawet jeżeli ich rozmiar jest nieznany i/lub dynamiczny



flexbox

```
body {  
    font-family:Raleway;  
    margin:0;  
    padding:0;  
    min-height: 100%;  
    display: flex;           └─ Stworzenie kontenera dla  
                            flexbox  
    flex-direction: column;  └─ Ustalenie kierunku elementów  
    position:absolute;  
    top:0; bottom:0; left:0; right:0;  
}
```

flexbox

```
header {  
    flex-grow: 0;  
    background-color:#ddd;  
    position:relative;  
}  
  
footer {  
    flex-grow: 0;  
    height:50px;  
    background-color:#ddd;  
}  
  
.columns {  
    flex-grow:1;  
    display:flex;  
    flex-direction: row;  
    width:100%;  
    overflow:auto;  
}  
  
.column {width:33%;padding:0 10px; border-right:1px solid black;}
```

Określa zdolność elementu do zmiany swojej wielkości (jeżeli jest to konieczne)

PODSTAWY JAVASCRIPT

Czym jest JavaScript

- Język skryptowy – najbardziej rozpowszechniony jako język programowania stron internetowych po stronie klienta
- Język dynamiczny
- Wspiera programowanie funkcyjne i obiektowe
- Pozwala na manipulowanie elementami strony internetowej i komunikację z serwerem
- Składnia języka podobna do języków C++ i C#

Podstawowe typy danych

- JavaScript zawiera pięć podstawowych typów danych:
 - Numeryczny,
 - Logiczny
 - Łańcuchowy
 - Pusty (null)
 - Niezdefiniowany (undefined)
- Ścisłe porównywanie (operatorы `==` і `!=`)

Zmienne

- Deklarowanie zmiennych – słowo kluczowe var

```
var imie = "Łukasz";
var liczbaStudentow;
liczbaStudentow = 12;
```

- Operator typeof – sprawdzanie typu

```
typeof imie; //"string"
typeof liczbaStudentow == "number"
```

Tablice (1)

- Tablice możemy tworzyć na dwa sposoby:
 - za pomocą operatora []
 - za pomocą funkcji Array

```
var tablica = [];
var tablicaLiczb = [12,13,14];
var tablicaImion = ["Jan","Adam","Ewa"];
var tablicaWartosciWszelakich = [12,"Adam",13.4];
```

- Dostęp do elementów tablicy możliwy jest za pomocą operatora [], przy czym tablice są numerowane od 0

```
console.log(tablicaLiczb[0])
```

Tablice (2)

Sprawdzanie długości tablicy: nazwaTablicy.length

funkcja	Deklaracja
concat	Array.concat(wart1, wart2, ...)
every	Array.every(funkcja)
filter	Array.filter(funkcja)
forEach	Array.forEach(funkcja)
isArray	Array.isArray(wartosc);
indexOf	Array.indexOf(wartość, indeks)
join	Array.join(ogranicznik)
lastIndexOf	Array.lastIndexOf(wartość, indeks)
map	Array.map(funkcja)

Tablice (3)

funkcja	Deklaracja
pop	Array.pop()
push	Array.push(wartość1, wartość2, ...)
reverse	Array.reverse()
shift	Array.shift()
slice	Array.slice(początek, koniec)
some	Array.some(funkcja)
sort	Array.sort(funkcja)
splice	Array.splice(początek, ile, w1, w2, ...)
toString	Array.toString()
unShift	Array.unShift(wartość1, wartość2, ...)

Funkcje (1)

- Deklarowanie funkcji

```
function pokazKomunikat() {  
    alert("komunikat");  
}
```

```
pokazKomunikat();
```

- Parametry funkcji

```
function pokazKomunikat(tresc) {  
    alert(tresc);  
}
```

```
pokazKomunikat("komunikat");
```

- Słowo kluczowe arguments

Funkcje (2)

- Rezultat działania funkcji określa się za pomocą słowa kluczowego return

```
function tresp1() {  
    return "Ala ma kota";  
}
```

```
function pokazKomunikat(tresp) {  
    alert(tresp);  
}
```

```
pokazKomunikat(tresp1());
```

Funkcje (3)

- Funkcje są wartościami

```
function pokazKomunikat() {  
    alert("komunikat");  
}
```

```
var info = pokazKomunikat;  
info();
```

Funkcje (4)

- Funkcje można przekazywać jako parametr

```
function pokazKomunikat(generujTresc) {
```

```
    alert(generujTresc());
```

```
}
```

```
function tresc1() {
```

```
    return "Ala ma kota";
```

```
}
```

```
pokazKomunikat(tresc1);
```

Funkcje (5)

- Funkcje mogą nie mieć nazwy

```
function pokazKomunikat(generujTresc) {  
    alert(generujTresc());  
}
```

```
pokazKomunikat(function() {return "Ala";});
```

```
var info = function() {return "Ala";}
```

```
pokazKomunikat(info);
```

Funkcje (6)

- Funkcje mogą być deklarowane wewnątrz funkcji ...

```
function pokazKomunikat() {  
    var trecs1 = function() {return "Ala";};  
    function trecs2() { return "ma kota";}  
  
    alert(trecs1()+" "+trecs2());  
}  
  
pokazKomunikat();
```

Funkcje (7)

- ... i zwracane

```
function pokazKomunikat(generujTresc) { alert(generujTresc()); }
```

```
function tresc1() { return "Ala ma kota"; }
```

```
pokazKomunikat(tresc1);
```

```
function tresc2(imie) {  
    return function() { return imie+"ma kota"; }  
}
```

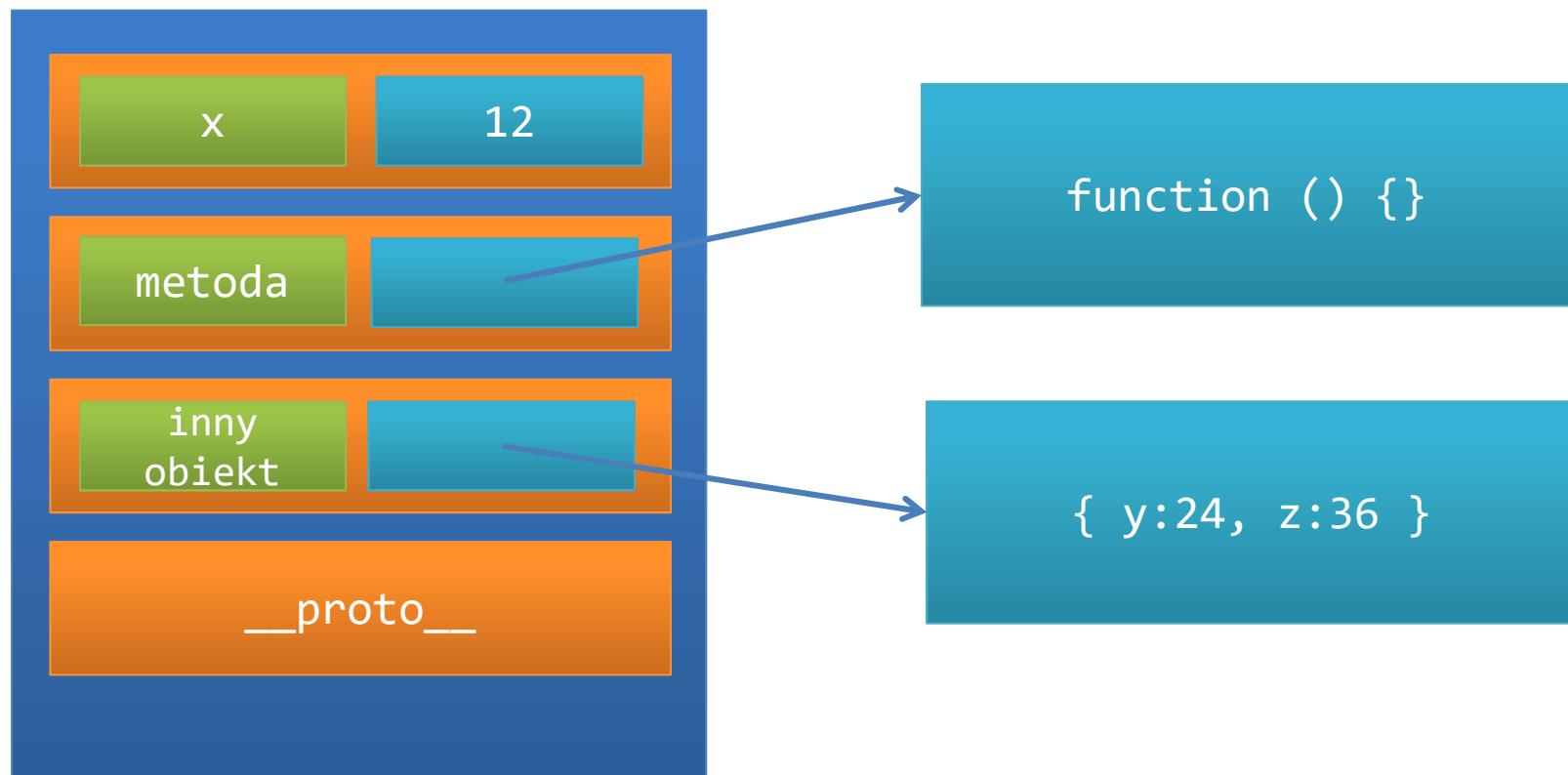
```
pokazKomunikat(tresc2("Tomek"));
```

Zakres widoczności zmiennych

- Zakresem widoczności zmiennych jest funkcja

```
function przyklad() {  
    var a=2, b=3;  
    if(a!==0) {  
        const y = b/a;  
        y = 12;  
    }  
    alert(y);  
}
```

Obiekty



Obiekty (1)

- Obiekty mogą zawierać właściwości (dane/stan) oraz metody (zachowanie)
- Obiekty tworzymy za pomocą słowa kluczowego new lub metody literalnej

```
var obiekt = new Object();
```

```
obiekt.wlasciwosc = 12;  
obiekt.metoda = function() {  
    ...  
}
```

```
var obiekt = {};
```

```
var obiekt = {  
    wlasciwosc : 12,  
    metoda : function() {  
        ...  
    }  
}
```

Obiekty (2)

- Obiekty mogą być przekazywane do i zwracane przez funkcje

```
function Test(artykul)
{
    return {
        pelnyTytul: artykuł.Tytul +" "+ artykuł.podtytul
    };
}
```

```
function Test(artykul)
{
    return
    {
        pelnyTytul: artykuł.Tytul +" "+ artykuł.podtytul
    };
}
```

UWAGA! Będzie błąd!!!



Obiekty (3)

- Słowo kluczowe this

```
var obiekt = {};
```

```
var obiekt = {
    wlasciwosc : 12,
    metoda : function() {
        alert(this.wlasciwosc);
    }
}
```

Obiekty (4)

- Konstruktory

```
function Artykul(tytul, podtytul, trecs) {  
    this.tytul = tytul;  
    this.podtytul = podtytul;  
    this.trecs = trecs;  
    this.pelnyTutul = function() {  
        return this.tytul + " " + this.podtytul;  
    }  
}
```

```
var art1 = new Artykul("Programowanie", "jest", "fajne");  
var art2 = new Artykul("Bardzo", "lubie", "JavaScript");
```

Obiekty (5)

- Konstruktory

```
function Artykul(tytul, podtytul, trecs) {  
    this.tytul = tytul;  
    this.podtytul = podtytul;  
    this.trecs = trecs;  
    this.pełnyTytul = function() {  
        return this.tytul + " " + this.podtytul;  
    }  
}
```

```
var art1 = new Artykul("Programowanie", "jest", "fajne");  
var art2 = new Artykul("Bardzo", "lubie", "JavaScript");
```

Obiekty (6)

- Prototypy

Pozwalają na definiowanie dziedziczenia

```
function Artykul(tytul, podtytul, tresc) {  
    this.tytul = tytul;  
    this.podtytul = podtytul;  
    this.tresc = tresc;  
}
```

```
Artykul.prototype.pelnyTytul = function() {  
    return this.tytul+" "+this.podtytul;  
}
```

```
var art1 = new Artykul("Programowanie", "jest", "fajne");  
var art2 = new Artykul("Bardzo", "lubie", "JavaScript");
```

Obiekty (7)

- Łańcuch prototypów

```
function Artykul() {}
```

```
Artykul.prototype.publikuj = function() {...}
```

```
function Komentarz(artykul, login, data, tresc ) {...}
```

```
Komentarz.prototype = new Artykul();
```

```
function ArtykulGlowny(tytul, podtytul, tresc) {...}
```

```
ArtykulGlowny.prototype = new Artykul();
```

JAVASCRIPT W PRZEGŁĄDARCE I JQUERY

Gdzie umieszczać skrypty?

- Skrypty JavaScript można umieszczać:
 - w elementach HTML (przypisując skrypt bezpośrednio do zdarzenia w elemencie)
 - w znaczniku script wewnętrz strony internetowej
 - w osobnym pliku

Znacznik <script>

- Znacznik, w którym można umieszczać skrypty JavaScript

```
<script type="text/javascript">
    alert("Ala ma kota");
</script>
```

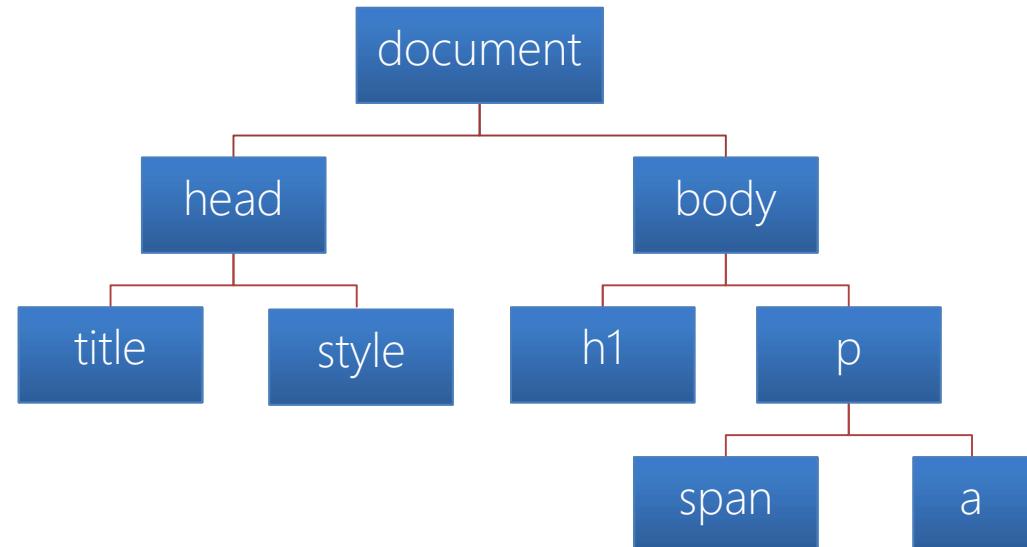
- Może być umieszczony w dowolnym miejscu strony
- Renderowanie strony zostanie zatrzymane do momentu wykonania skryptu

Czym jest model DOM?

- DOM (ang. Document Object Model)
- Model umożliwiający obiektowy opis strony internetowej oraz podstawowe API pozwalające na wykonywanie operacji na strukturze dokumentu
- W modelu tym strona przedstawiana jest w postaci drzewa, gdzie:
 - Węzłami wewnętrznymi, są zawsze tagi HTML
 - Liśćmi drzewa są tagi HTML, lub elementy tekstowe

Czym jest model DOM?

```
<html>
<head>
  <title>przyklad</title>
  <style>
    h {color:red}
  </style>
</head>
<body>
  <h1>Naglowek</h1>
  <p>Jakis tekst
    <span>cos</span>
    <a href="adres">
      link
    </a>
  </p>
</body>
</html>
```



Dostęp do elementów modelu DOM możliwy jest dopiero po jego załadowaniu i wygenerowaniu

Czym jest model DOM?

Każdy węzeł posiada m. in następujące właściwości i metody

Właściwości	Metody
attributes[]	appendChild
childNodes[]	cloneChild
children[]	hasAttribute / hasAttributes
className	getAttribute / getAttributes
id	hasChildNodes
innerHTML	insertBefore
nodeName	removeAttribute
nodeType	removeChild
nodeValue	replaceChild
style	

Pobieranie elementów DOM

- Do elementów modelu DOM możemy dostać się za pomocą jednej z trzech funkcji:
 - `document.getElementById`
 - `document.getElementsByTagName`
 - `document.getElementsByClassName`

```
<h1 id="naglowek">Zawartosc</h1>

<script type="text/javascript">
    var element = document.getElementById("naglowek");
</script>
```

- Dostęp do elementów modelu DOM możliwy jest dopiero po jego załadowaniu i wygenerowaniu

Czym jest jQuery?

- Powstała w 2006 roku,
- Dostępna na stronie: www.jquery.com
- JQuery jest to obecnie jedna z najpopularniejszych bibliotek upraszczających programowanie w języku JavaScript – przede wszystkim operacje na modelu DOM,
- Wspiera wiele różnych przeglądarek
- Aktualna wersja to 3.2.1

Podstawowe zasady jQuery

Filozofia "pobierz i działaj"

Podstawą jest obiekt jQuery (oznaczany również \$)

jQuery - selektory

- selektory pozwalają na wybranie elementów strony HTML
- wspierane są pojedyncze elementy lub wiele elementów

`$(selektor)`

`jQuery(selektor)`

Selektory jQuery (1)

- Biblioteka jQuery posiada bardzo bogaty zestaw selektorów, bazujących na składni CSS
 - wszystkie elementy - `$("*")`
 - selektory HTML - `$("a")`
 - selektory ID - `$("#id")`
 - selektory klasy - `".klasa"`
 - kontekst elementy pochodne - `("div a.menu")`
 - kontekst dzieci - `("div>a")`
 - kontekst rodzeństwo - `("span~a") , ("span+a")`
 - wartość atrybutu:
 - `("a[rel]")`
 - `("a[rel=menu]")`
 - `("a[href^=http])`
 - `("a[href$=.com"])`
 - `("a[href*=onet"])`

Selektory jQuery (2)

- Biblioteka jQuery posiada bardzo bogaty zestaw selektorów, bazujących na składni CSS
 - selektory formularzy - `$("input:radio")`, `$(":button")`
 - selektory wg. pozycji - `:first`, `:last`, `:first-child`,
`:last-child`, `:only-child`, `:nth-child(n)`,
`:nth-child(even|odd)`, `:nth-child(An+B)`,
`:even`, `:odd`, `:eq(n)`, `:gt(n)`, `:lt(n)`
- To są tylko przykładowe selektory (jest ich więcej)
- można stworzyć własne selektory

Iteracja po węzłach

- funkcja `.each((index, element) => {})`

```
$("div").each( (index, element) => {  
    console.log(`#${index} ${$(element).text()}`);  
});
```



element HTML

Model DOM i jQuery (1)

- Biblioteka jQuery pozwala na przechodzenie po elementach drzewa DOM oraz dokonywanie ich modyfikacji

Funkcja	Opis
size()	Zwraca liczebność zbioru
get(index)	Pobiera jeden lub wszystkie elementy
index(element)	Zwraca indeks elementu
add(wyrażenie)	Dodaje elementy w wyrażeniu do zbioru
not(wyrażenie)	Usuwa elementy w wyrażeniu ze zbioru
filter(wyrażenie)	Filtruje elementy zbioru na podstawie wyrażenia
slice(pocz, kon)	Zwraca fragment zbioru
end()	Wykorzystane w ciągu metod jQuery przywraca zbiór elementów do poprzedniego zbioru
find(selektor)	Zwraca listę elementów pochodnych zgodnych z selektorem

Model DOM i jQuery (2)

- Biblioteka jQuery pozwala na przechodzenie po elementach drzewa DOM oraz dokonywanie ich modyfikacji

funkcja	Opis
each(funkcja)	Wywołuje funkcję przekazaną jako parametr na wszystkich elementach zbioru
attr(nazwa)	Zwraca wartość atrybutu
attr(nazwa, wartość)	Ustawia wartość atrybutu
removeAttr(nazwa)	Usuwa atrybut
addClass(nazwy)	Dodaje określone klasy do obiektów zbioru
removeClass(nazwy)	Usuwa określone klasy z obiektów zbioru
css(nazwa, wartość)	Ustawia właściwość CSS
css(nazwa)	Zwraca właściwość CSS
html(tekst), html()	Ustawia/zwraca zawartość wszystkich elementów zbioru
text(tekst), text()	Ustawia/zwraca zawartość tekstową we wszystkich elementach zbioru.

Model DOM i jQuery (3)

- Biblioteka jQuery pozwala na przechodzenie po elementach drzewa DOM oraz dokonywanie ich modyfikacji

funkcja	Opis
append(zawartość)	Dołącza zawartość przekazaną jako parametr do wszystkich elementów zbioru.
wrap(element)	Obejmuje obiekty w zbiorze elementem przekazanym jako parametr
wrapAll(element)	Obejmuje obiekty w zbiorze – jako całość - elementem przekazanym jako parametr
remove()	Usuwa wszystkie obiekty w zbiorze z drzewa DOM
empty()	Usuwa zawartość wszystkich elementów w zbiorze
val()	Zwraca wartość właściwości "value"
val(wartość)	Ustawia właściwość "value"
val(wartości)	Powoduje, że wszystkie checkbox-y, radiobutton-y i obiekty option są wybrane (checked), jeżeli wartość atrybutu value przyjmuje jedną z wartości przekazanych jako parametr

Model DOM i jQuery (4)

- Przykładowe operacje na zbiorze elementów

```
<!DOCTYPE html>
<html lang="en">
<body>
<ul id="lista">
  <li></li>
  <li>
    <ul>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </li>
  <li></li>
  <li></li>
</ul>
</body>
</html>
```

```
$(function () {
  $('#lista').
    .find('> li')
    .filter(':first')
    .addClass('specjalny')
  .end()
  .find('ul')
    .css('border', '1px solid red')
    .find('li:last')
      .addClass('specjalny')
    .end()
  .end()
  .end()
  .find('li')
    .append('każdy li');
});
```

Modyfikowanie elementów

```
$( "div" ).each( (i,e)=> {
    e.style.backgroundColor = "#ddd";
});
```

```
$( "div" ).each( function (i,e) {
    this.style.backgroundColor = "#ddd";
});
```

Dostęp do atrybutów

- funkcja `.attr()`

```
$("#obraz").attr("src");
```

odczyt jednego atrybutu

```
$("#obraz").attr("src", "plik.jpg");
```

modyfikacja
jednego atrybutu

```
$("#obraz").attr({  
    src: "plik.jpg",  
    alt: "opis obrazu  
});
```

modyfikacja wielu
atrybutu

Modyfikowanie stylów

- funkcja `.css()`

odczyt jednej właściwości

```
$("#obraz").css("background-color");
```

modyfikacja jednej
właściwości

```
$("#obraz").css("background-color", "red");
```

```
$("#obraz").css({  
    backgroundColor: "red",  
    border: "1px solid black"  
});
```

modyfikacja wielu
właściwości

Modyfikowanie klas

- do pracy z klasami w jQuery istnieją cztery funkcje
 - `addClass()`
 - `hasClass()`
 - `removeClass()`
 - `toggleClass()`

Tworzenie elementów DOM

- Za pomocą funkcji \$ można tworzyć również nowe elementy HTML
 - `$("<div>Witaj świecie</div>")`
 - `$("<div>")`
 - `$("<div/>")`

Dodawanie i usuwanie elementów

- Nowe elementy możemy dodawać do strony za pomocą:
 - **append()**
 - **appendTo()**
 - **prepend()**
 - **prependTo()**
- Istniejące elementy możemy usuwać za pomocą funkcji:
 - **remove()**

wstawianie elementu
jako ostatni element

wstawianie elementu
jako pierwszy element

Otaczanie elementu

- funkcja `.wrap()`

```

```

```
$( "img" ).wrap( '<div class="obraz" />' );
```

```
<div class="obraz">  
      
</div>
```

Zdarzenia

- Zdarzenie - umożliwia obiektowi powiadomić inne obiekty o zmianie swego stanu.
- Procedura obsługi zdarzenia - funkcja, której kod ma się wykonać w momencie wystąpienia danego zdarzenia.
- Akcja domyślna - działanie, które przeglądarka normalnie wykonuje, w momencie wystąpienia danego zdarzenia.

Akcje domyślne przypisane są do takich zdarzeń, jak:

- naciśnięcie linku przez użytkownika
- zatwierdzenie formularza
- przeniesienie "focus" do następnego elementu, gdy użytkownik naciśnie klawisz TAB.

jQuery i zdarzenia

- Najczęściej stosowane zdarzenia posiadają własne funkcje pozwalające je przypisywać do elementów:

`blur()`, `change()`, `click()`, `dblclick()`, `focus()`,
`keydown()`, `keyup()`, `keypress()`, `load()`, `mousedown()`,
`mouseenter()`, `mouseleave()`, `mousemove()`, `mouseout()`,
`mouseover()`, `mouseup()`,
`ready()`, `resize()`, `select()`, `submit()`, `toggle()`,
`unload()`

Parametry zdarzeń

Funkcje obsługi zdarzeń mogą posiadać parametry

```
$("a").click(function (e) {  
    console.dir(e);  
});
```

Zdarzenia

Obiekt *event* umożliwia dostęp do dodatkowych informacji dotyczących się danego zdarzenia.

Właściwości	Metody
altKey, ctrlKey, shiftKey	preventDefault()
Bubbles	stopPropagation()
button*	
cancelable	
charCode	
clientX, clientY	
keyCode	
eventPhase	
target	
type	

jQuery i zdarzenia

Przypisywanie dowolnego zdarzenia

.on(zdarzenia, funkcja)

```
$("div").on("click", function () {  
    console.log("click");  
});
```

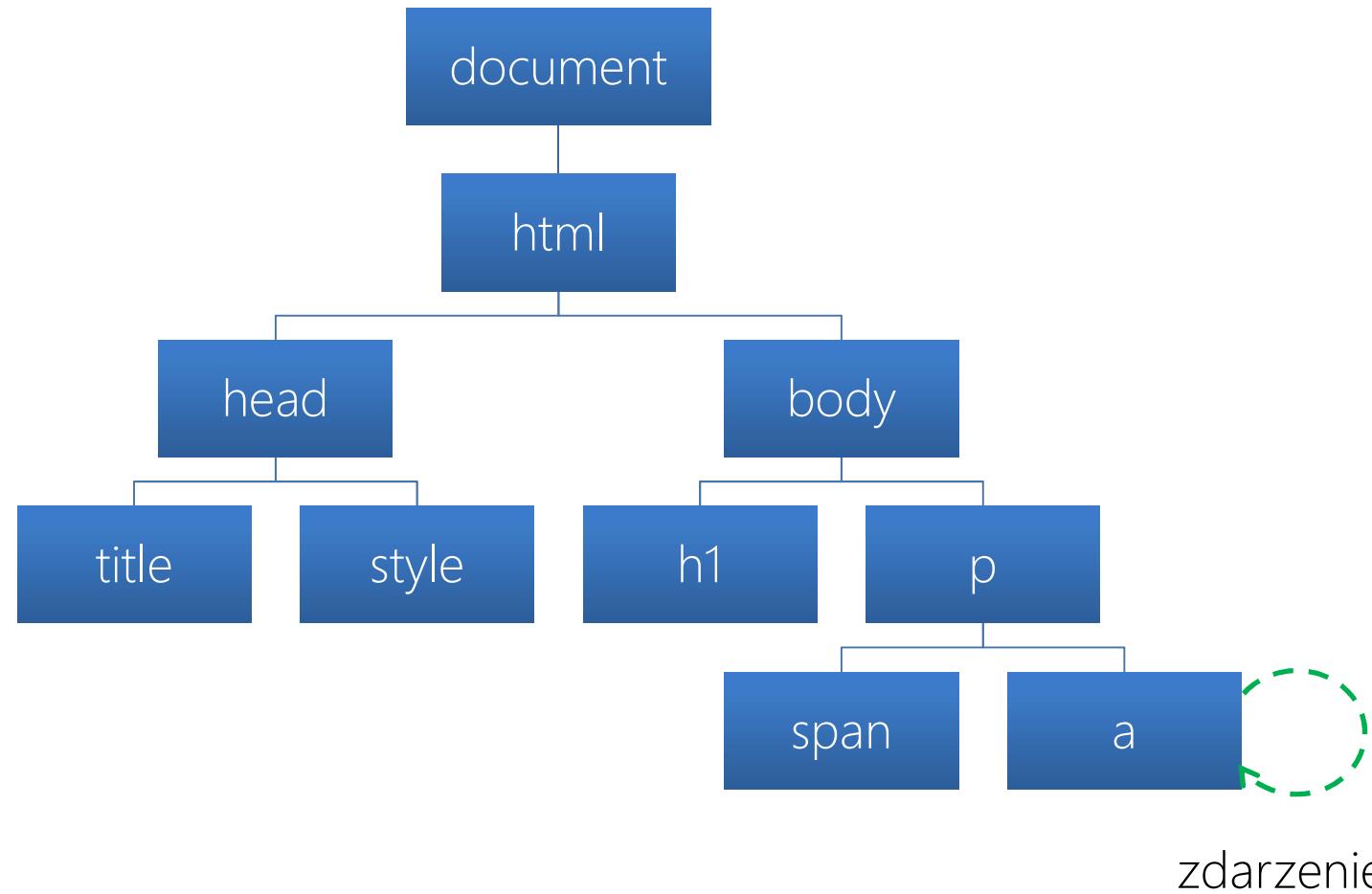
jQuery i zdarzenia

Usunięcie dowolnego zdarzenia

`.off(zdarzenie)`

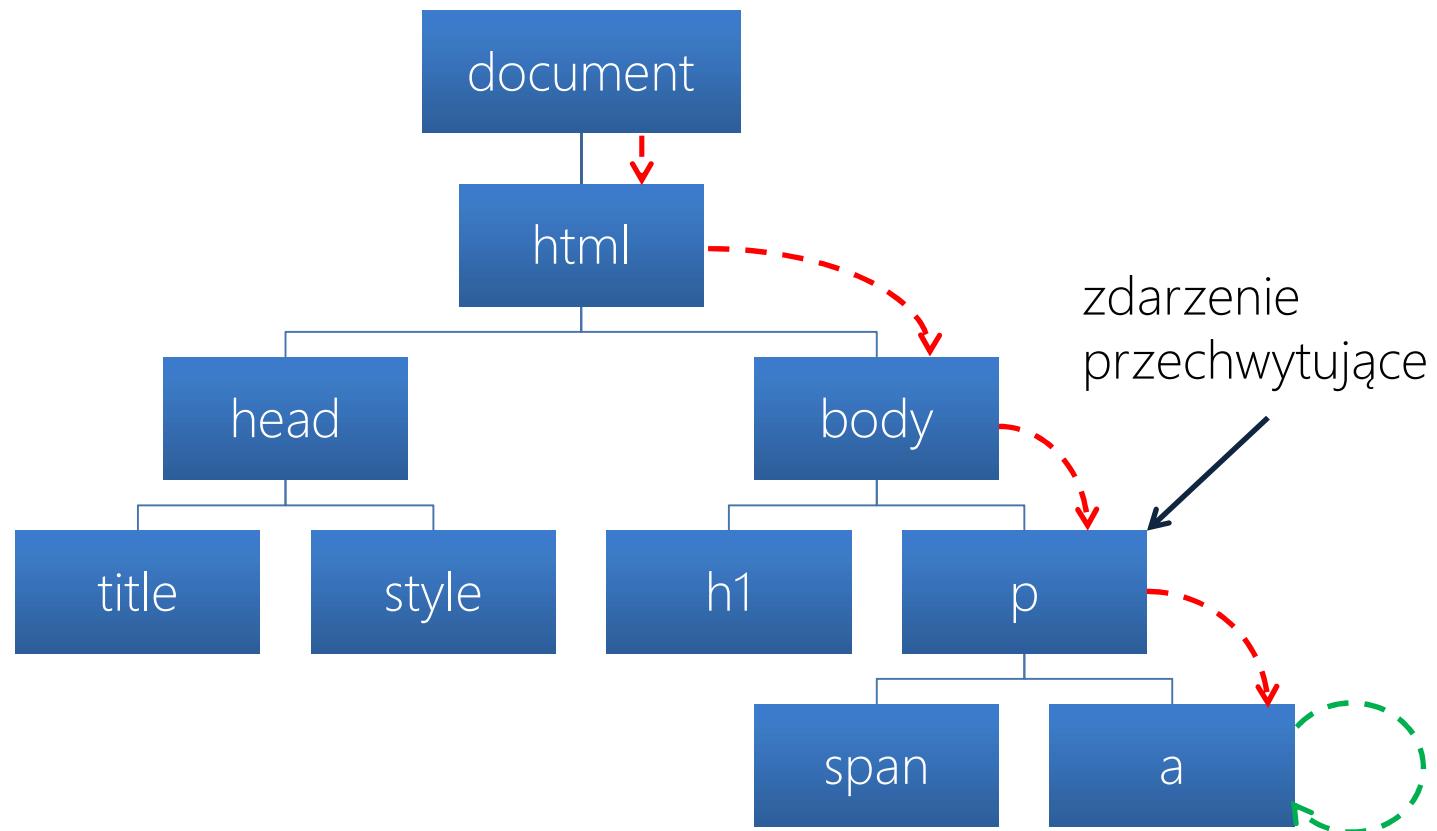
```
$("div").off("click");
```

Zdarzenia – propagowanie zdarzeń



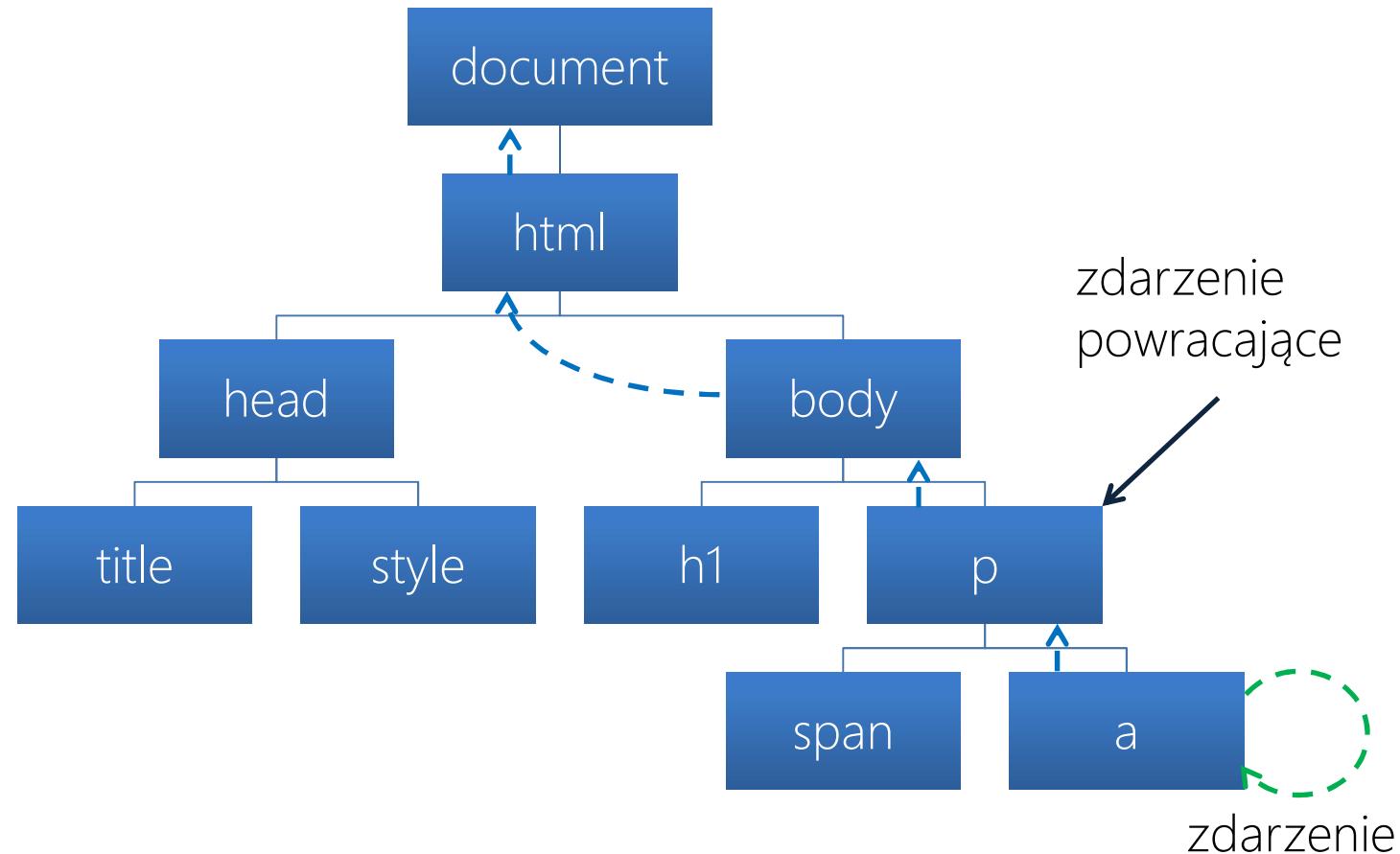
Zdarzenia – propagowanie zdarzeń

Model Netscape – „przechwytywanie”



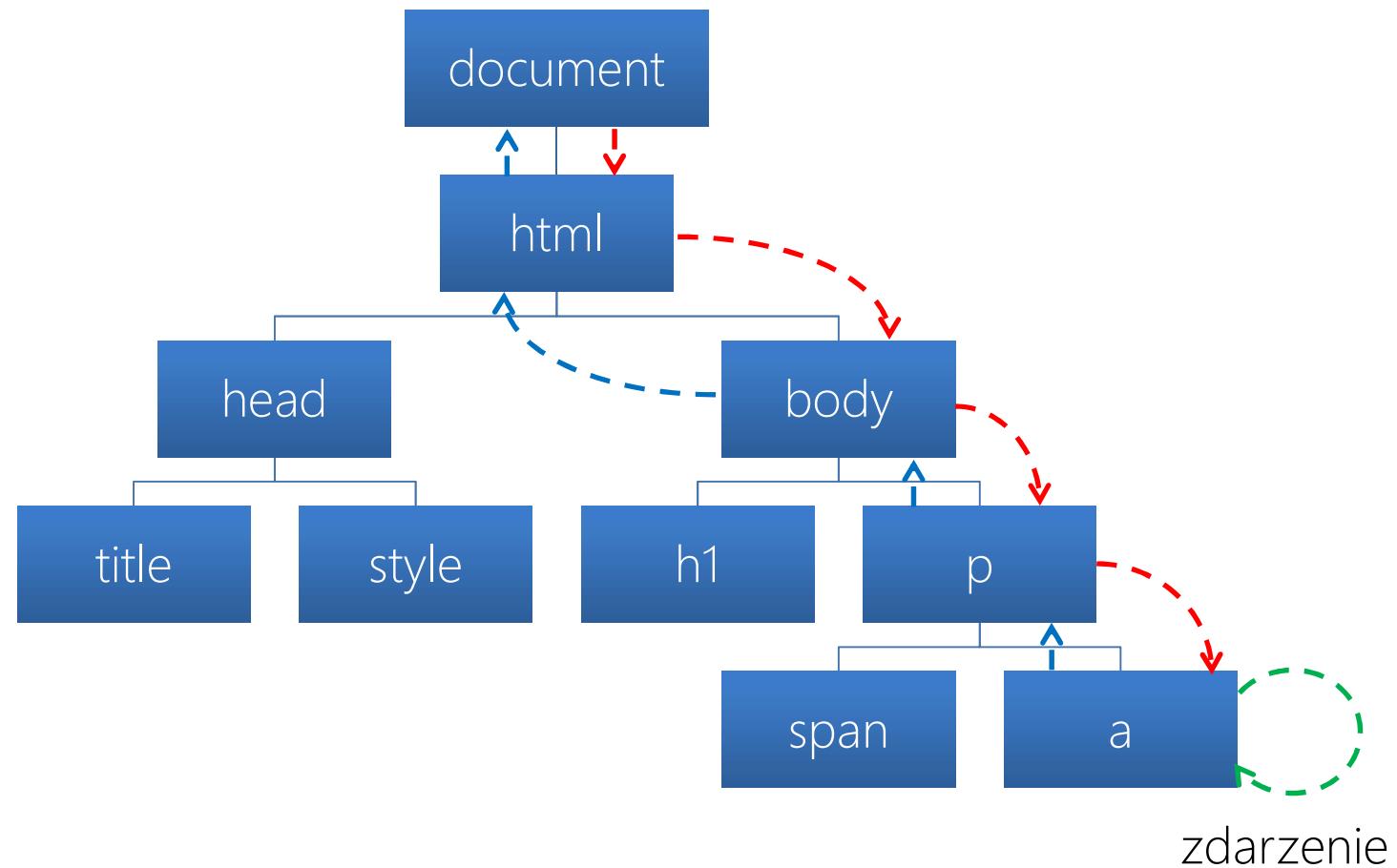
Zdarzenia – propagowanie zdarzeń

Model Internet Explorer – „bąbelkowanie”



Zdarzenia – propagowanie zdarzeń

Model W3C



jQuery i zdarzenia

Przypisywanie dowolnego zdarzenia do elementu pochodnego

```
.on( zdarzenia, selektor, funkcja)
```

```
$("div").on("click", "a", function () {  
    console.log("click");  
});
```

Zdarzenia

Zdarzenia load, DOMContentLoaded pozwalają na wykonywanie kodu w momencie całkowitego załadowania strony, załadowania drzewa DOM.

```
<!DOCTYPE html>
<html><head>
    <script type="text/javascript">
        window.addEventListener("load", function () {
            console.log("onLoad");
        }, false);
        document.addEventListener("DOMContentLoaded", function() {
            console.log("onDOMLoaded");
        }, false);
    </script>
</head>
<body>
</body>
</html>
```

Zdarzenie ready

- Zdarzenie ready – punkt startowy aplikacji

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

- Może występować w uproszczonej wersji

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

Przykład 1

```
<!DOCTYPE html>
<html>
  <body>
    <form>
      <div>
        <label for="imie">Imię: </label>
        <input id="imie" />
      </div>
      <div>
        <label for="nazwisko">Nazwisko: </label>
        <input id="nazwisko" />
      </div>
      <div><button>Zatwierdź</button></div>
    </form>
    <div id="rezultat"></div>
    <script src="jquery-3.2.1.min.js" ></script>
    <script src="przyklad.js" ></script>
  </body>
</html>
```

Przykład 1 - js

```
$(function () {
    $("button").click(function (e) {
        var imie = $("#imie").val();
        var nazwisko = $("#nazwisko").val();

        $("<span>").text(imie)
            .appendTo("#rezultat");
        $("<span>").text(nazwisko)
            .appendTo("#rezultat");
    })
});
```

Przykład 2

```
<!DOCTYPE html>
<html>
  <body>
    <form>
      <div>
        <label for="adres">adres: </label><input id="adres" />
      </div>
      <div><button>Zatwierdź</button></div>
    </form>
    <div id="rezultat"></div>
    <script src="jquery-3.2.1.min.js" ></script>
    <script src="przyklad.js" ></script>
  </body>
</html>
```

Przykład 2 - js

```
function linkKlikniecie(event) {
    if(!confirm("Czy na pewno chcesz przejść do tej strony"))
        e.preventDefault();
}

$(function () {
    $("button").click(function (e) {
        var adres = $("#adres").val();

        $("<span><a></a></span>")
            .find("a")
            .text(adres)
            .attr("href", adres)
            .attr("target", "_blank")
            .click(linkKlikniecie)
            .wrap("<div />")
            .parent()
            .appendTo("#rezultat");
    });
});
```

Przykład 2 - js

```
$function () {
    $("#rezultat").on("click", "a", function (event) {
        if(!confirm("Czy na pewno chcesz przejść do tej strony"))
            event.preventDefault();
    });

    $("button").click(function (e) {
        var adres = $("#adres").val();

        $("<span><a></a></span>")
            .find("a")
            .text(adres)
            .attr("href", adres)
            .attr("target", "_blank")
            .wrap("<div />")
            .parent()
            .appendTo("#rezultat");
    });
});
```

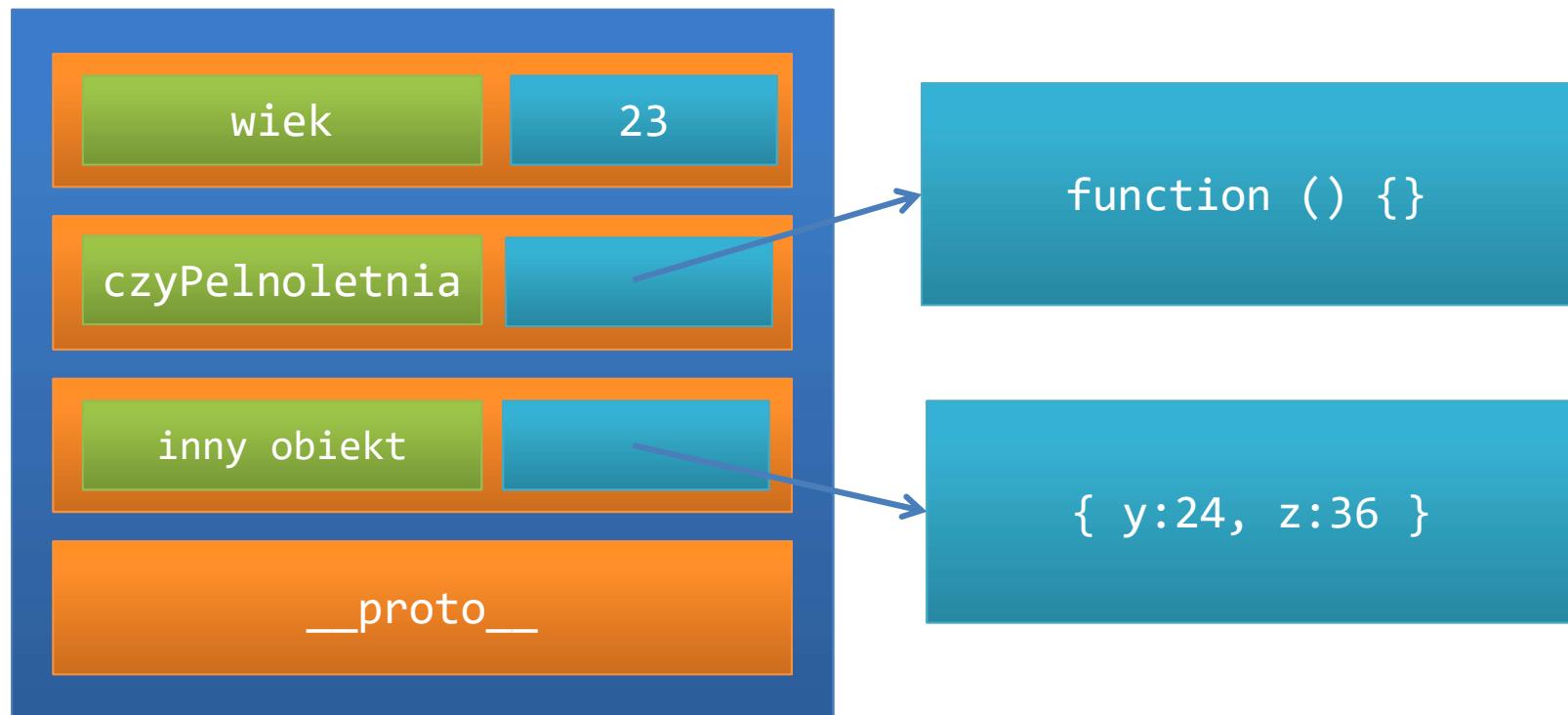
PROGRAMOWANIE OBIEKTOWE W JAVASCRIPT

Tworzenie obiektów w JavaScript

Za pomocą funkcji konstruktora

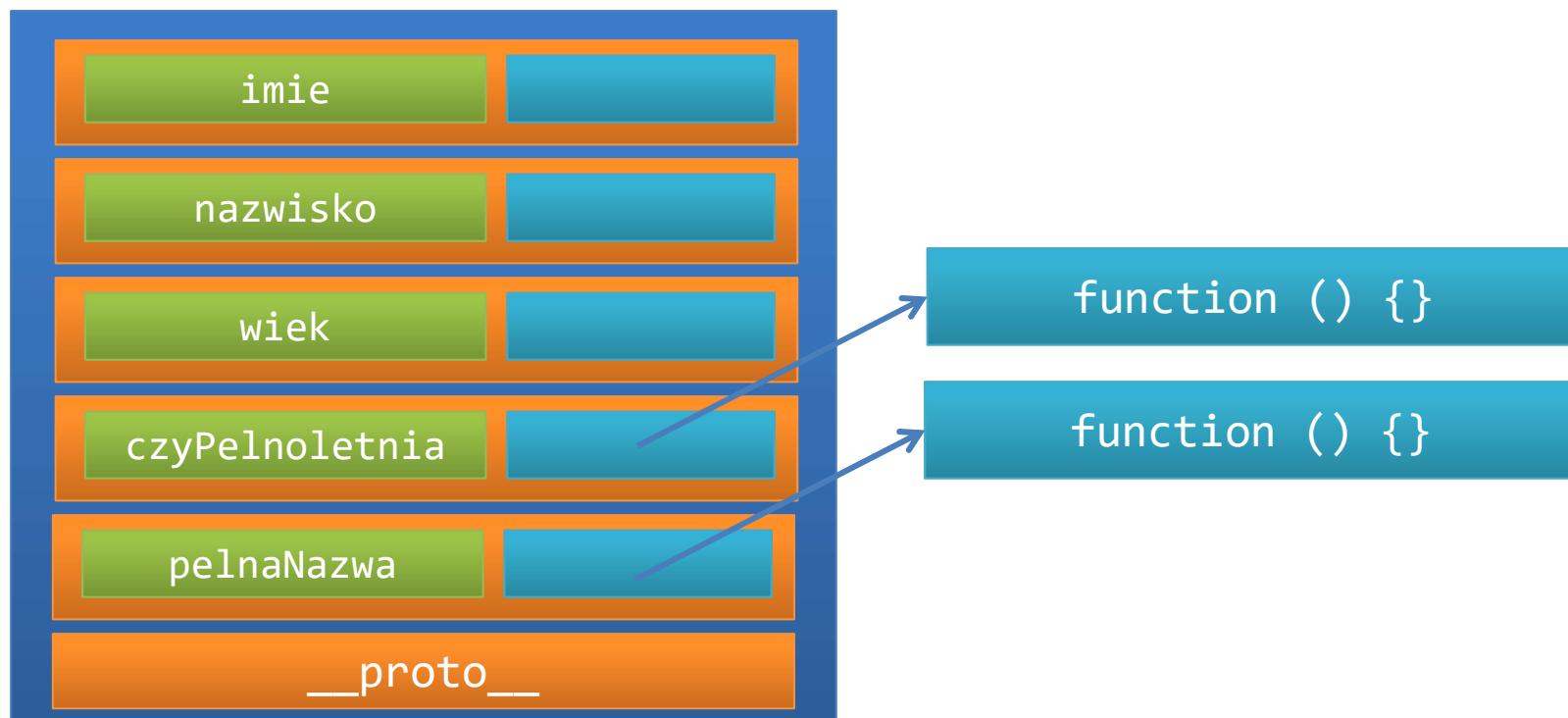
```
function Osoba(imie, nazwisko, wiek) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
    this.wiek = wiek;  
    this.pelnaNazwa = function() {  
        return this.nazwisko +" "+this.imie;  
    }  
  
    this.czyPelnoletnia = function() {  
        return this.wiek>18;  
    }  
}  
  
var osoba1 = new Osoba("Ala", "Kowalska", 23);
```

Obiekt w JavaScript



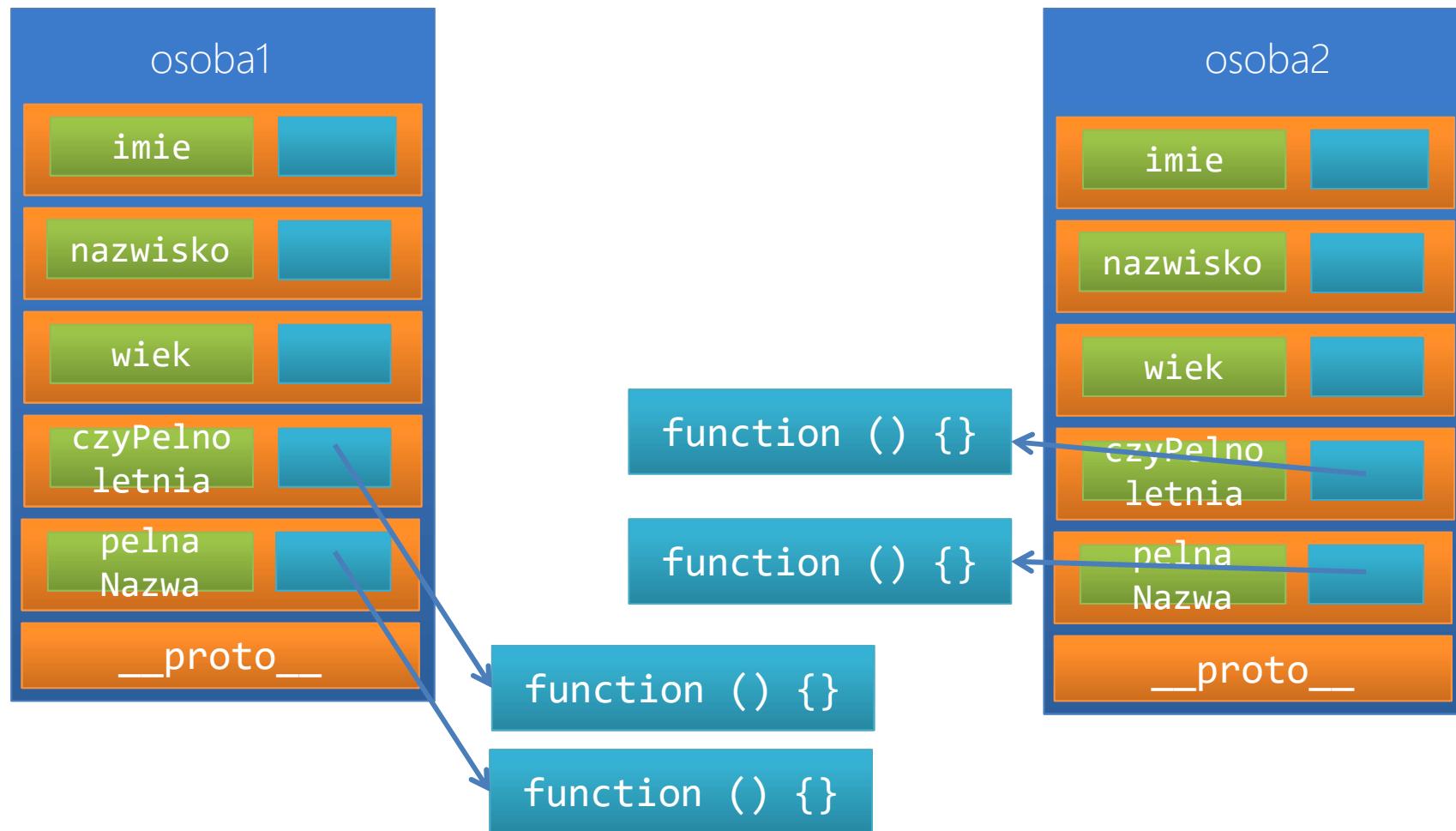
Obiekty w JavaScript

```
var osoba1 = new Osoba("Ala", "Kowalska", 23);
```

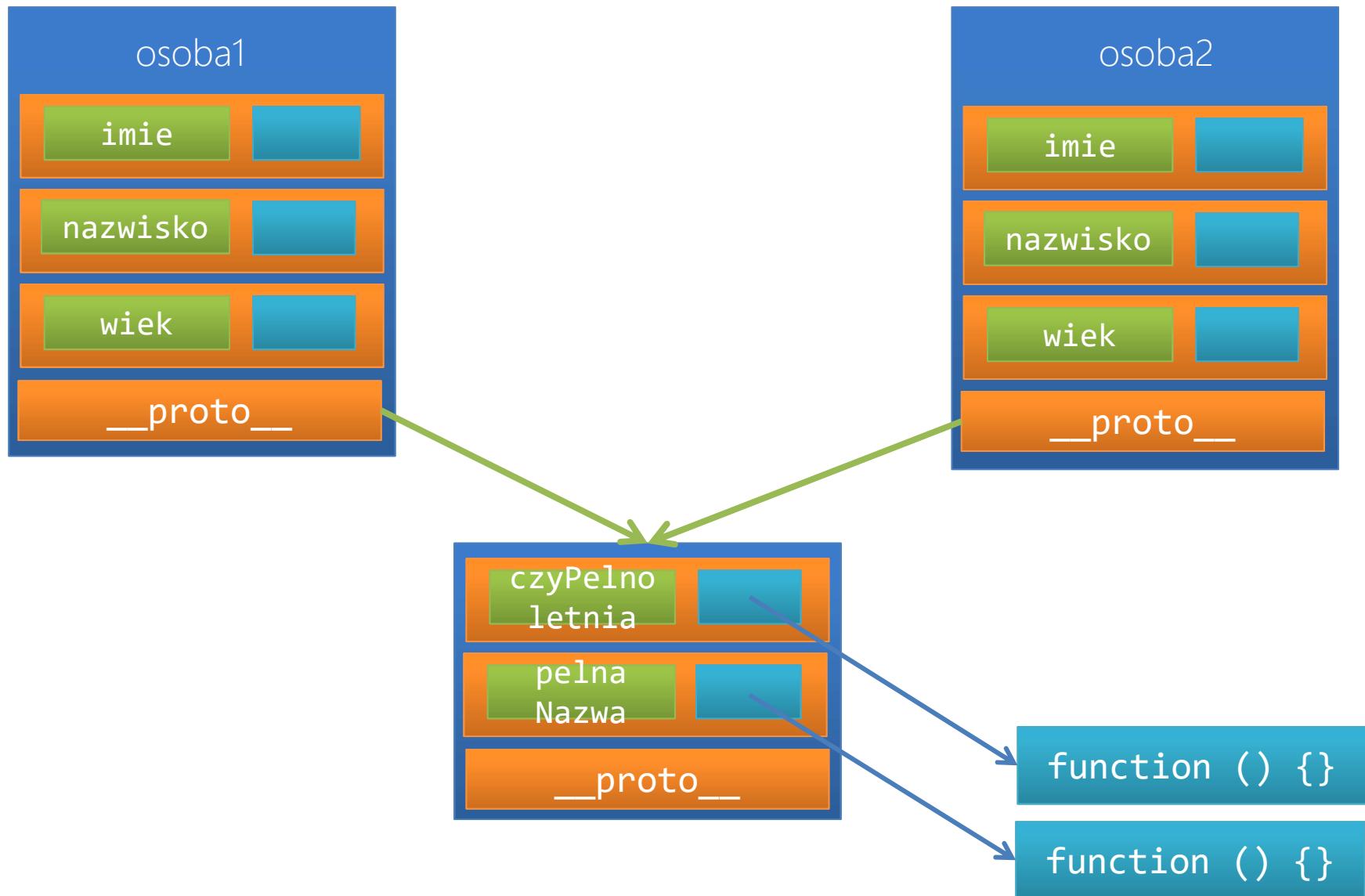


Obiekty w JavaScript

```
var osoba1 = new Osoba("Ala", "Kowalska", 23);  
var osoba2 = new Osoba("Tomasz", "Nowak", 18);
```



Obiekty w JavaScript

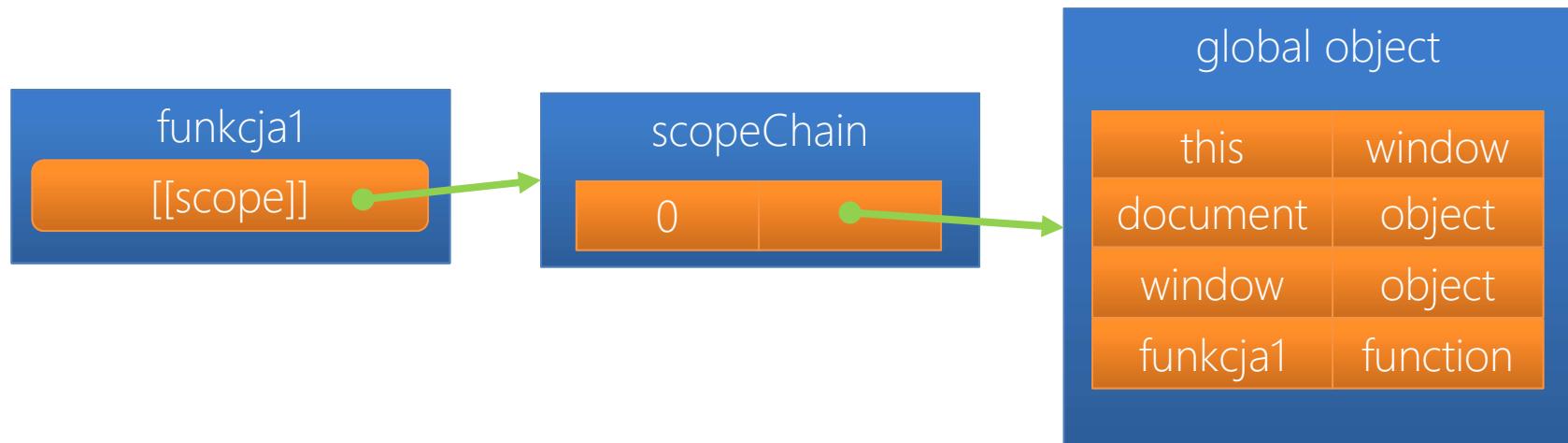


Prototype

```
function Osoba(imie, nazwisko, wiek) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
    this.wiek = wiek;  
}  
  
Osoba.prototype.pelnaNazwa = function() {  
    return this.nazwisko +" "+this.imie;  
}  
  
Osoba.prototype.czyPelnoletnia = function() {  
    return this.wiek>18;  
}  
  
var osoba1 = new Osoba("Ala", "Kowalska", 23);  
var osoba2 = new Osoba("Tomasz", "Nowak", 18);
```

Definiowanie funkcji

```
function funkcja1(param1) {  
  
    var lokalna1 = 12;  
    console.log(param1);  
    console.log(lokalna1);  
  
}
```



Wywołanie funkcji

```
funkcja1(12);
```



activationObject

this	window
arguments	[]
param1	12
lokalna1	12

global object

this	window
document	object
window	object
funkcja1	function

Dopełnienia

```
function funkcja1(param1) {  
    var lokalna1 = 12;  
  
    function funkcja2() {  
        var lokalna2 = 24;  
        console.log(lokalna1);  
    }  
  
    funkcja2();  
    console.dir(funkcja2);  
}
```

Dopełnienia

```
function funkcja2() {  
    var lokalna2 = 24;  
    console.log(lokalna1);  
}  
  
function funkcja1(param1) {  
    var lokalna1 = 12;  
  
    funkcja2();  
    console.dir(funkcja2);  
}
```

Będzie błąd!

Dopełnienia

```
function funkcja1(param1) {  
    var lokalna1 = 12;  
  
    function funkcja2() {  
        var lokalna2 = 24;  
        console.log(lokalna1);  
    }  
  
    funkcja2();  
    console.dir(funkcja2);  
}
```



activationObject	
this	window
arguments	[]
param1	12
lokalna1	12

global object	
this	window
document	object
window	object
funkcja1	function

Dopełnienia

```
function funkcja1(param1) {  
    var lokalna1 = 12;  
  
    function funkcja2() {  
        var lokalna2 = 24;  
        console.log(lokalna1);  
    }  
  
    funkcja2();  
    console.dir(funkcja2);  
}
```



scopeChain	
0	
1	



scopeChain	
0	
1	
2	

activationObject (dopełnienie)

this	window
arguments	[]
param1	12
lokalna1	12

activationObject (funkcja)

this	window
arguments	[]
lokalna2	24

global object

this	window
document	object
window	object
funkcja1	function

Pola i metody prywatne

```
function Osoba(imie, nazwisko, wiek) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
    this.pelnaNazwa = function() {  
        return this.nazwisko +" "+this.imie;  
    }  
  
    this.czyPelnoletnia = function() {  
        return wiek>18;  
    }  
}  
  
var osoba1 = new Osoba("Ala", "Kowalska", 23);
```

Tu będzie utworzone
dopełnienie

Pola i metody statyczne

```
function Osoba(imie, nazwisko, wiek) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
    var wiek = wiek;  
    this.pelnaNazwa = function() {  
        return this.nazwisko +" "+this.imie;  
    }  
  
    this.czyPelnoletnia = function() {  
        return wiek>18;  
    }  
}  
  
Osoba.stworzPusta = function () {  
    return new Osoba("", "", 0);  
}
```

Dziedziczenie

```
function Osoba(imie, nazwisko, wiek) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
    this.wiek = wiek;  
}  
  
Osoba.prototype.pelnaNazwa = function() {...}  
  
Osoba.prototype.czyPelnoletnia = function() {...}  
  
function Student(imie, nazwisko, wiek, nrIndeksu) {  
    Osoba.call(this, imie, nazwisko, wiek);  
    this.nrIndeksu = nrIndeksu;  
}  
  
Student.prototype = Object.create(Osoba.prototype);
```

Wywołanie funkcji Osoba

Utworzenie prototypu

Klasy w JavaScript

Standard EcmaScript 6 wprowadza do JavaScript idee klasy

```
class Osoba {  
  
    constructor(imie, nazwisko, wiek) {  
        this.imie = imie;  
        this.nazwisko = nazwisko;  
        this.wiek = wiek;  
    }  
  
    pelnaNazwa() {  
        return this.nazwisko +" "+this.imie;  
    }  
  
    czyPelnoletnia() {  
        return this.wiek > 18;  
    }  
}
```

Klasy w JavaScript - właściwości

```
class Osoba {  
  
    constructor(imie, nazwisko, wiek) {  
        this.imie = imie;  
        this.nazwisko = nazwisko;  
        this.wiek = wiek;  
    }  
  
    get wiek() {  
        return this._wiek;  
    }  
  
    set wiek(wartosc) {  
        this._wiek = wartosc;  
    }  
}
```

Klasy w JavaScript - static

```
class Osoba {  
  
    constructor(imie, nazwisko, wiek) {  
        this.imie = imie;  
        this.nazwisko = nazwisko;  
        this.wiek = wiek;  
    }  
  
    static stworzPusta () {  
        return new Osoba("", "", 0);  
    }  
}
```

Klasy w JavaScript - dziedziczenie

```
class Student extends Osoba {  
    constructor(imie, nazwisko, wiek, nrIndeksu) {  
        super(imie, nazwisko, wiek);  
        this.nrIndeksu = nrIndeksu;  
    }  
}
```

Wyjątki - throw

Instrukcja throw pozwala na rzucanie wyjątków zdefiniowanych przez użytkownika

```
throw "Ala ma kota"  
throw 12;  
throw true;  
throw Error("komunikat o błędzie");
```

Wyjątki – try/catch/finally

Instrukcja try/catch/finally oznacza blok instrukcji do wypróbowania i określa reakcję na błędy

```
try {  
    //instrukcje do wywołania  
}  
catch(wyjatek) {  
    //instrukcje reagujące na błąd  
}  
finally {  
    //instrukcje wykonane bez względu na wystąpienie błędu  
}
```

ASYNCHRONICZNA JAVASCRIPT

Model wykonania JavaScript

Javascript jest językiem jednowątkowym.

Uruchamiane jest jedno zadanie i dopóki się ono nie zakończy nie jest uruchamiane inne zadanie

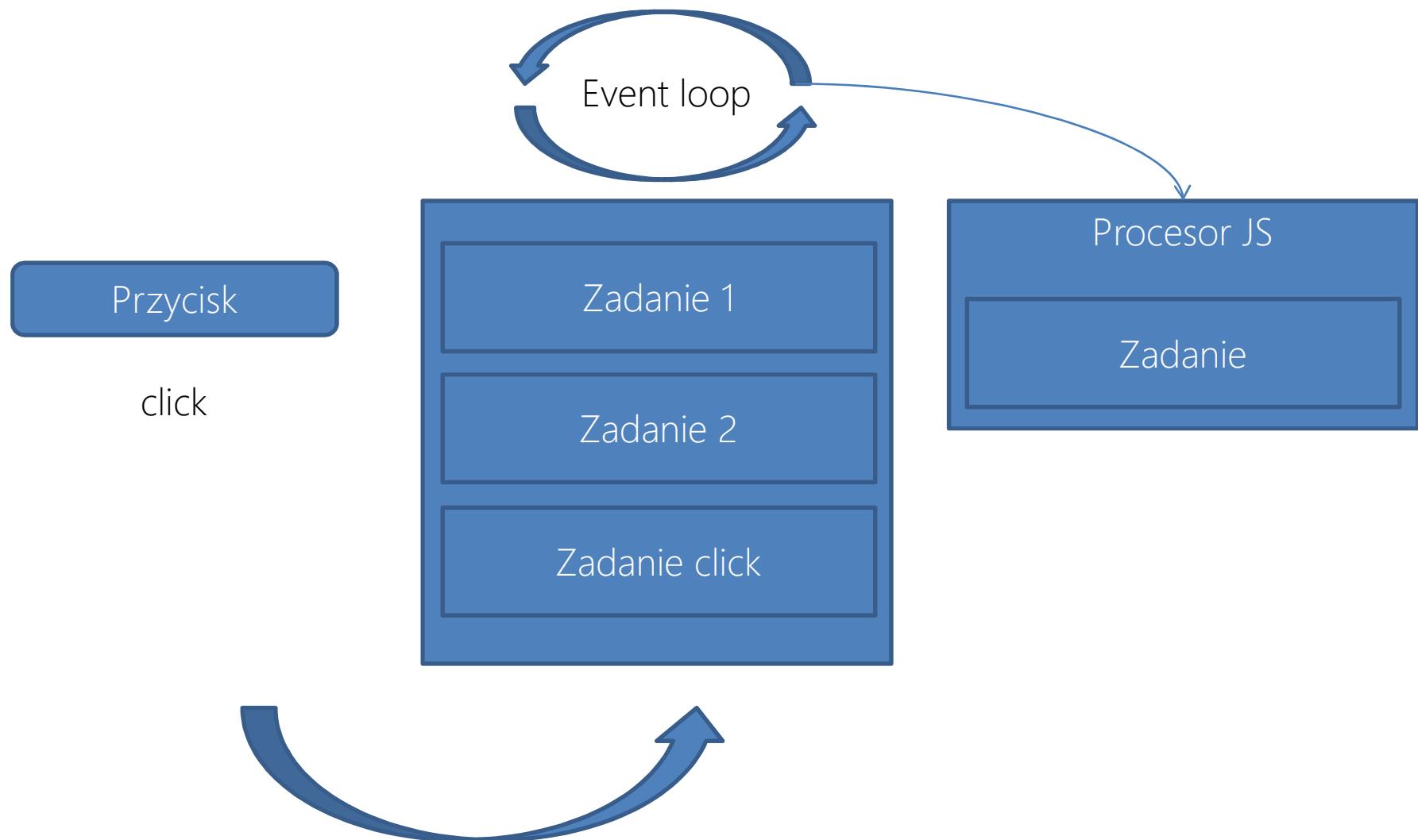
Zadania

Zadania są to wywołania funkcji

Ale funkcji, które zostały uruchomione nie z poziomu Javascript,
a ze "środowiska zewnętrznego" np. naciśnięcie guzika

Typowo środowisko zewnętrzne dzieli ten sam wątek, który Javascript wykorzystuje
dla innych elementów, które przetwarza

Model wykonania Javascript



Model wykonania Javascript

- Każde zadanie jest niezależne
- Każde zadanie przechowuje swoje dane w ramce stosu
- Dane pomiędzy zadaniami można przekazywać za pomocą dopełnień
- Podczas pracy z zadaniami można posłużyć się następującymi technikami:
 - wywołania zwrotne
 - obietnice (promises)
 - obiekty obserwowalne
 - async/await

Wywołania zwrotne

```
let funkcja = (fz) => {
    console.log("funkcja");
    fz();
    console.log("no i koniec");
}

funkcja(()=>{console.log("callback");});
```

```
let funkcja1 = ()=>{console.log("callback");};

funkcja(funkcja1);
```

To nie są funkcje zwrotne asynchroniczne

Wywołania zwrotne asynchroniczne

```
setTimeout(() => {
    console.log("funkcja wykonywana asynchronicznie");
}, 2000);

console.log("czekaj!");
```

Wywołania zwrotne asynchroniczne

```
let funkcja = () => {
  let dane = 12;
  let fz = () => {console.log(dane);}

  console.dir(fz);
  setTimeout(fz, 2000);
};

funkcja();
console.log("czekaj!");
```

setTimeout / setInterval

dwie podstawowe funkcje wykonywane asynchronicznie

setTimeout:

- parametry funkcja i przedział czasu
- funkcja jest wywoływana po upłynięciu podanego przedziału czasu

setInterval:

- parametry funkcja i przedział czasu
- funkcja jest wywoływana jest ciągle za każdym razem po upłynięciu podanego przedziału czasu

```
setTimeout(() => {
    console.log("funkcja wykonywana asynchronicznie");
}, 0);

console.log("czekaj!");
```

Asynchroniczność, a błędy

- Try/catch nie działa przy wywołaniach asynchronicznych
- Funkcja zwrotna może być zdefiniowana wewnątrz bloku try, ale kiedy jest uruchamiana wykonuje się w innym zadaniu poza leksykalnym obejmowaniem bloku try
- Try/catch musi być zakodowania wewnątrz funkcji callback

```
try {  
    setTimeout(() => {  
        throw Error("Błąd");  
    }, 2000);  
}  
catch(err) {  
    console.log(err);  
}
```

```
setTimeout(() => {  
    try {  
        throw Error("Błąd");  
    }  
    catch(err) {  
        console.log(err);  
    }  
}, 2000);
```

Wzorce wywołań asynchronicznych

- wywołania sekwencyjne
- wywołania równoległe
- wyścig

Przykład

```
setTimeout(() => {
    console.log("funkcja A");
}, 1000);

setTimeout(() => {
    console.log("funkcja B");
}, 2000);

setTimeout(() => {
    console.log("funkcja C");
}, 500);

console.log("czekaj!");
```

Wywołania sekwencyjne



Wywołania sekwencyjne

```
setTimeout(() => {  
  
    console.log("funkcja A");  
    setTimeout(() => {  
  
        console.log("funkcja B");  
        setTimeout(() => {  
  
            console.log("funkcja C");  
  
        }, 500);  
  
    }, 2000);  
  
}, 1000);  
  
console.log("czekaj!");
```

Piramida śmierci

Wywołania równoległe



Wywołania równoległe

```
setTimeout(() => { console.log("funkcja A"); }, 1000);  
setTimeout(() => { console.log("funkcja B"); }, 2000);  
setTimeout(() => { console.log("funkcja C"); }, 500);  
  
function PoWszystkim() {  
    console.log("czekaj!");  
}  
PoWszystkim();
```

Wywołania równoległe

```
var licznik = 0;
licznik++;
setTimeout(() => {
    console.log("funkcja A"); licznik--;
    if(licznik == 0) PoWszystkim();
}, 1000);

licznik++;
setTimeout(() => {
    console.log("funkcja B"); licznik--;
    if(licznik == 0) PoWszystkim();
}, 2000);

licznik++;
setTimeout(() => {
    console.log("funkcja C"); licznik--;
    if(licznik == 0) PoWszystkim();
}, 500);

function PoWszystkim() { console.log("czekaj!"); }
```

Wyścig



© Darrell Urnski

Wyścig

```
var pierwszy = true;

setTimeout(() => {
    console.log("funkcja A");
    if(pierwszy) { Połyszystkim(); pierwszy = false; }
}, 1000);

setTimeout(() => {
    console.log("funkcja B");
    if(pierwszy) { Połyszystkim(); pierwszy = false; }
}, 2000);

setTimeout(() => {
    console.log("funkcja C");
    if(pierwszy) { Połyszystkim(); pierwszy = false; }
}, 500);

function Połyszystkim() { console.log("czekaj!"); }
```

Funkcje zwrotne

- Zastosowanie funkcji zwrotnych dla jednej operacji asynchronicznej działa dobrze
- Wykorzystanie funkcji zwrotnych do zarządzania wieloma, połączonymi operacjami asynchronicznymi może być problematyczne
- Obsługa błędów jest również problematyczna

Obietnice

Obietnice są obiektami, które opisują wartość, która obecnie nie jest jeszcze znana, ale mamy pewność, że zostaniemy o niej poinformowani jeżeli zostanie poznana.

Obietnice



Obietnice

```
let p = new Promise((resolve, reject) => {
    let rozwiazac = true;
    setTimeout(()=> {
        if(rozwiazac) {
            resolve("udało się");
        }
        else {
            reject("błąd");
        }
    }, 2000);
});
```

Wykorzystanie obietnic

```
p.then(rezultat => {
  console.log(rezultat);
});
```

```
p.then(
  rezultat => {
    console.log(rezultat);
  },
  rezultat => {
    console.log("Błąd "+rezultat);
  },
);
```

```
p.then( rezultat => {
  console.log(rezultat);
})
.catch( rezultat => {
  console.log("Błąd "+rezultat);
});
```

Obietnice

- Obietnice mogą być w trzech stanach:
 - oczekująca (pending) – utworzona i oczekująca na rozwiążanie lub odrzucenie,
 - spełniona – (wywołana resolve),
 - odrzucona – (wywołane reject)
- Obietnica raz spełniona lub odrzucona nie może zmienić stanu

Obietnice – sekwencja

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
  new Promise((rozwiaz, odrzuc) => {
    licznik++;
    setTimeout(()=>{
      if(czyRozwiazac) {
        console.log(licznik);
        rozwiaz({wiadomosc:"Udało się", wartosc: licznik});
      }
      else
        odrzuc({wiadomosc:"Błąd", wartosc: licznik});
    }, czas);
  });

wykonaj(true, 1000)
  .then(()=>wykonaj(true, 2000))
  .then(()=>wykonaj(true, 500))
  .then(()=>console.log("koniec"));
```

Obietnice – sekwencja

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
  new Promise((rozwiaz, odrzuc) => {
    licznik++;
    setTimeout(()=>{
      if(czyRozwiazac) {
        console.log(licznik);
        rozwiaz({wiadomosc:"Udało się", wartosc: licznik});
      }
      else
        odrzuc({wiadomosc:"Błąd", wartosc: licznik});
    }, czas);
  });

wykonaj(true, 1000)
  .then((wynik)=>{console.log(wynik); wykonaj(true, 2000);})
  .then((wynik)=>{console.log(wynik); wykonaj(true, 2000);})
  .then((wynik)=>{console.log(wynik); console.log("koniec");});
```

Obietnice – sekwencja

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
    new Promise((rozwiaz, odrzuc) => {
        licznik++;
        setTimeout(()=>{
            if(czyRozwiazac) {
                console.log(licznik);
                rozwiaz({wiadomosc:"Udało się", wartosc: licznik});
            }
            else
                odrzuc({wiadomosc:"Błąd", wartosc: licznik});
        }, czas);
    });

wykonaj(true, 1000)
    .then((wynik)=>{console.log(wynik); return wykonaj(true, 2000);})
    .then((wynik)=>{console.log(wynik); return wykonaj(true, 2000);})
    .then((wynik)=>{console.log("koniec"); console.log(wynik);})
    .catch((wynik)=>{console.log("Błąd"); console.log(wynik);});
```

Obietnice – operacje równoległe

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
  new Promise((rozwiaz, odrzuc) => {
    licznik++;
    setTimeout(()=>{
      if(czyRozwiazac)
        rozwiaz({wiadomosc:"Udało się", wartosc: licznik});
      else
        odrzuc({wiadomosc:"Błąd", wartosc: licznik});
    }, czas);
  });

let p1 = wykonaj(true, 1000);
let p2 = wykonaj(true, 2000);
let p3 = wykonaj(true, 500);

Promise.all([p1,p2,p3]).then(wynik=>{console.log("Koniec", wynik);});
```

Obietnice – operacje równoległe

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
  new Promise((rozwiaz, odrzuc) => {
    let lokalna = licznik++;
    setTimeout(()=>{
      if(czyRozwiazac)
        rozwiaz({wiadomosc:"Udało się", wartosc: lokalna});
      else
        odrzuc({wiadomosc:"Błąd", wartosc: lokalna});
    }, czas);
  });

let p1 = wykonaj(true, 1000);
let p2 = wykonaj(true, 2000);
let p3 = wykonaj(true, 500);

Promise.all([p1,p2,p3]).then(wynik=>{console.log("Koniec", wynik);});
```

Obietnice – wyścig

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
    new Promise((rozwiaz, odrzuc) => {
        licznik++;
        setTimeout(()=>{
            if(czyRozwiazac)
                rozwiaz({wiadomosc:"Udało się", wartosc: licznik});
            else
                odrzuc({wiadomosc:"Błąd", wartosc: licznik});
        }, czas);
    });

let p1 = wykonaj(true, 1000);
let p2 = wykonaj(true, 2000);
let p3 = wykonaj(true, 500);

Promise.all([p1,p2,p3]).then(wynik=>{console.log("Koniec", wynik);});
```

Obietnice – wyścig

```
let licznik = 0;

let wykonaj = (czyRozwiazac, czas) =>
    new Promise((rozwiaz, odrzuc) => {
        let lokalna = licznik++;
        setTimeout(()=>{
            if(czyRozwiazac)
                rozwiaz({wiadomosc:"Udało się", wartosc: lokalna});
            else
                odrzuc({wiadomosc:"Błąd", wartosc: lokalna});
        }, czas);
    });

let p1 = wykonaj(true, 1000);
let p2 = wykonaj(true, 2000);
let p3 = wykonaj(true, 500);

Promise.race([p1,p2,p3]).then(wynik=>{console.log("Koniec", wynik);});
```

BOOTSTRAP

Bootstrap



Najpopularniejszy w sieci framework HTML, CSS
i Javascript do tworzenia responsywnych projektów
typu mobile first *.

wg. strony bootstrapa (getbootstrap.com)

Dlaczego Bootstrap?

- Tworzenie odpowiednich stylów CSS bywa czasami kłopotliwe
- Wspieranie różnych przeglądarek jest wyzwaniem
- Wspieranie różnych urządzeń jest również wyzwaniem
- Pozwala na proste tworzenie układów stron

Responsywne projektowanie stron



One framework to rule them all



Wsparcie dla przeglądarek

	Chrome	Firefox	Internet Explorer	Opera	Safari
Android	✓ Supported	✓ Supported	N/A	✗ Not Supported	N/A
iOS	✓ Supported	N/A		✗ Not Supported	✓ Supported
Mac OS X	✓ Supported	✓ Supported		✓ Supported	✓ Supported
Windows	✓ Supported	✓ Supported	✓ Supported	✓ Supported	✗ Not Supported

abc.go.com

abc

shows watch live search menu ≡

trending shows

MARVEL
AGENT CARTER

2-HOUR PREMIERE TONIGHT 9|8c

MARVEL
AGENT CARTER

TUESDAYS 9|8c

abc

AMERICAN CRIME

WEDNESDAYS 10|9c

The BACHELOR

MONDAYS 8|7c

GALAVANT

SUNDAYS 8|7c

GREY'S ANATOMY

RETURNS THU FEB 11 8|7c

how to get away with Murder

RETURNS THU FEB 11 10|9c

JIMMY KIMMEL LIVE!

WEEKNIGHTS 11:35|10:35c

modernfamily

WEDNESDAYS 9|8c

QUANTICO

WATCH ALL FULL EPISODES

SCANDAL

RETURNS THU FEB 11 9|8c

www.washtimesherald.com/

About Us | Contact Us | App Downloads | e-Edition | Login

Washington Times Herald

With Find&Save it's all on sale.

FIND LOCAL SALES

FIND & SAVE

Home Subscribe News Sports Classifieds Community Opinion Obituaries Shop Public Notices

Featured Stories

LOCAL NEWS

Run on guns

Posted: 23 hours ago

Just after the first of the year President Barack Obama issued an executive order to try and tighten some of the regulations for the purchase of guns. The result has been a run on weapons and concealed carry licenses in Washington.

Comments (0)

Local News

Police Report | Business | Submit News

Winter storm warning issued

The National Weather Service has declared a Winter Storm Warning for five southwest Indiana counties and a Winter Weather Advisory for an additional 29 southern Indiana counties. Two to 4 inch...

EARLY POLICE REPORT Jan 19

High schoolers can score cash for college at STEM Challenge Jan 19

Police Reports Jan 18

Early Police Reports Jan 18

Football Experts Club

SIGN UP NOW!

The 2015 college football season is heating up, and now is your chance to get in on the action with the other members of the College Football Experts Club game. No drafts, rosters or trades. Simply pick the winners of each game and earn points. Play to win great prizes and a chance to reign as the champ!

Washington, IN 19°F

CLICK FOR MORE WEATHER

Davies Community Hospital

Advanced Search

Search this site...

MAY JUNE JULY

27 28 29 30 31 29 30 31

Tu We Th Fr Sa Su Mo Tu We Th Fr Sa

1 2 1 2 3 4 5 6 7 8 9 10 11

12 13 14 15 16 17 18 12 13 14 15 16 17 18

19 20 21 22 23 24 25 26 27 19 20 21 22 23 24 25

26 27 28 29 30 28 29 30 26 27 28 29 30 31

SEPTEMBER OCTOBER NOVEMBER

www.thetimes.co.uk/

THE TIMES THE SUNDAY TIMES TIMES+ Login | Subscribe | Contact us Search

THE TIMES

400,000 MEMBERS

JOIN NOW 12 WEEKS FOR £12

News | Opinion | Business | Money | Sport | Life | Arts | Puzzles | Papers | Irish news | Wednesday, January 20 | Max 6C

Latest News Thousands could miss out as pension exit fees scrapped

Apartheid of the asylum seekers on British streets

Andrew Norfolk Chief Investigative Reporter Last updated at 12:01AM, January 20 2016

A secret apartheid policy brands hundreds of asylum seekers in England's poorest town by housing them in properties with red front doors, it can be revealed. The doors make asylum homes easy to identify and are blamed for numerous attacks in which people were victims of harassment and abuse. A former local MP compared the red paint, used on terraced streets in deprived areas of Middlesbrough, to the yellow stars that Jews were forced to wear in Nazi Germany. The properties are owned by Stuart Monk, who is paid millions of pounds a year to provide accommodation for thousands of asylum seekers, some of whom fled war and oppression in Syria. Mr Monk is worth an estimated £175 million, according to The Sunday Times Rich List. His property company,...

- Red Alert
- Mark of shame
- No escape from badge
- Record numbers


£70,000 a month is plenty for ex-wife'

Divorcee's killer 'may have been rival in love triangle'


Police investigating the murder of a businesswoman while her partner was on bail in investigation into the

Miliband 'not to blame' for failure

Ed Miliband has ducked most of the responsibility for Labour's crushing election defeat in an official report that instead blames the Conservatives, the Liberal Democrats, Scottish nationalists and the media. The party's long-awaited review, overseen by...

Last updated at 12:01AM, January 20 2016

LATEST NEWS

Poland: gets troops from Britain after backing migrant curbs

High street: explain why you are charging women more

Nude activist: makes an exhibition of herself

Bramall: competence of Metropolitan police is questioned after fiasco

Billionaire divorce: £70,000 a month is plenty for ex-wife

Mussolini: stampede to print his war diaries

Times Woman
Don't miss our new column: Eat, Pray, Sleep - surviving with a newborn

Sign up to our weekly email out on Thursday afternoon

www.nytimes.com

≡ SECTIONS SEARCH

U.S. INTERNATIONAL 中文

SUBSCRIBE NOW LOG IN ⚙

The New York Times

Wednesday, January 20, 2016 | Today's Paper | Video | 22°F | DAX -3.02%

Curiosity is the world's most vital resource.

TRY A DIGITAL SUBSCRIPTION, JUST €1 FOR 4 WEEKS.

World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Style Food Travel Magazine T Magazine Real Estate ALL

Curiosity is the world's most vital resource.

€1 FOR A 4-WEEK DIGITAL SUBSCRIPTION

GET STARTED

DEVELOPING

Gunmen Kill at Least 19 at University in Pakistan

By ISMAIL KHAN 3:37 AM ET
Militants stormed Bacha Khan University in northwestern Pakistan, where students, a faculty member and guards were among the dead. All four attackers were killed, the police said.

• Live Updates: What We Know



Coffins holding the bodies of victims at a hospital after an attack at Bacha Khan University, in Charsadda, Pakistan, on Wednesday.
Bilawal Arbab/European Pressphoto Agency

Governor Says of Flint Water Crisis, 'I Let You Down'

By JULIE BOSMAN and MITCH SMITH
Gov. Rick Snyder of Michigan issued a sweeping apology to

ids-trunk-tennessee-tutor-s-car-n499186

Your Wednesday Briefing

By ADEEL HASSAN 19 minutes ago
Here's what you need to know to start your day.

- New York Today: Diners in the Rough



Max Whittaker for The NYT

The Opinion Pages

The Supreme Court, the Nativists and Immigrants

President Obama's action to protect undocumented parents from deportation was well within his authority.

- Bruni: Rethinking College Admissions
- Friedman: What If?
- Edsall: The Price of Republican Orthodoxy

OP-ED CONTRIBUTOR
Can Iran Change?
By ADEL BIN AHMED AL-JUBEIR
Saudi Arabia's foreign minister on the danger Iran poses to the Middle East.

- Op-Ed: Is It Better to Die in America or in England?
- Op-Ed: Hope for L.G.B.T. Rights in India
- Join us on Facebook »

Watching

An elementary school tutor in Tennessee was arrested after the authorities found nine young children in her car.

NBC News ▾

3h



System siatkowy

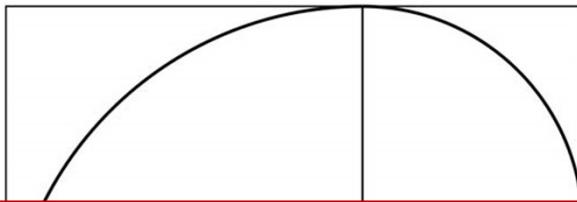
„Siatka jest instrumentem do uporządkowania elementów tekstowych i multimedialnych”

„There has to be a mathematical explanation for how bad your tie is ”

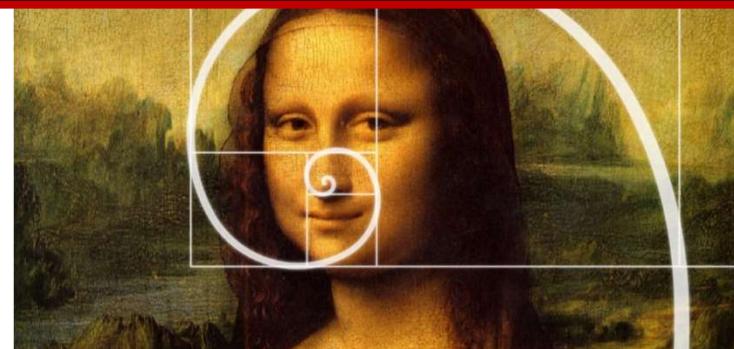


cytat z filmu "Piękny umysł"

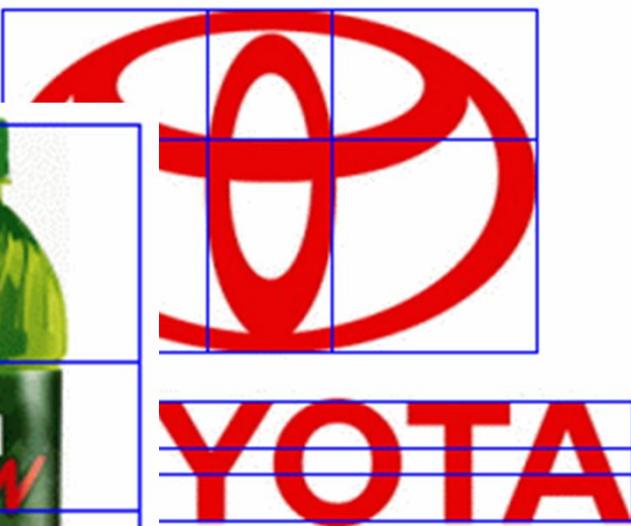
Złoty podział



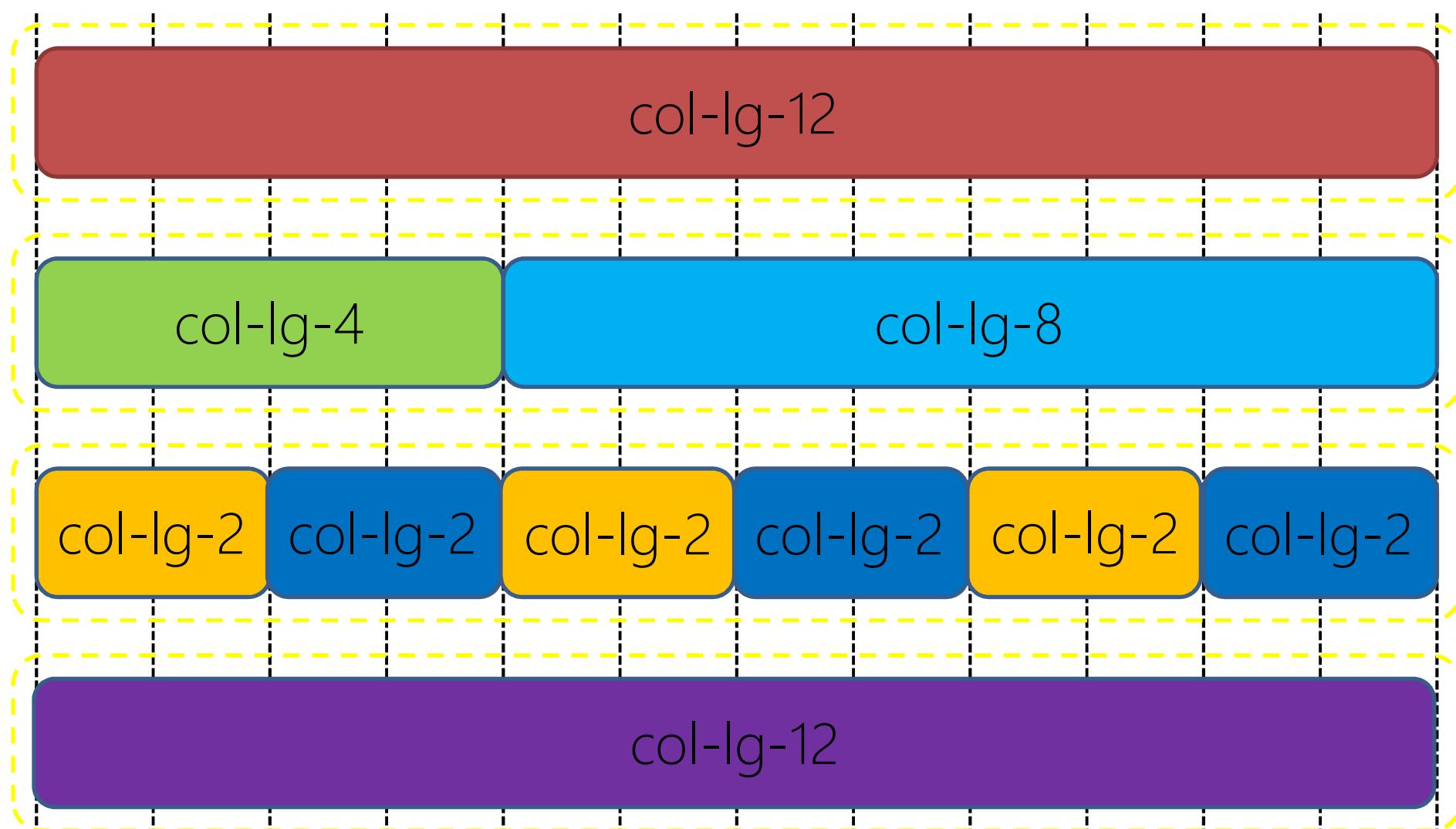
$\varphi \approx 1,618033\dots$



Złoty podział



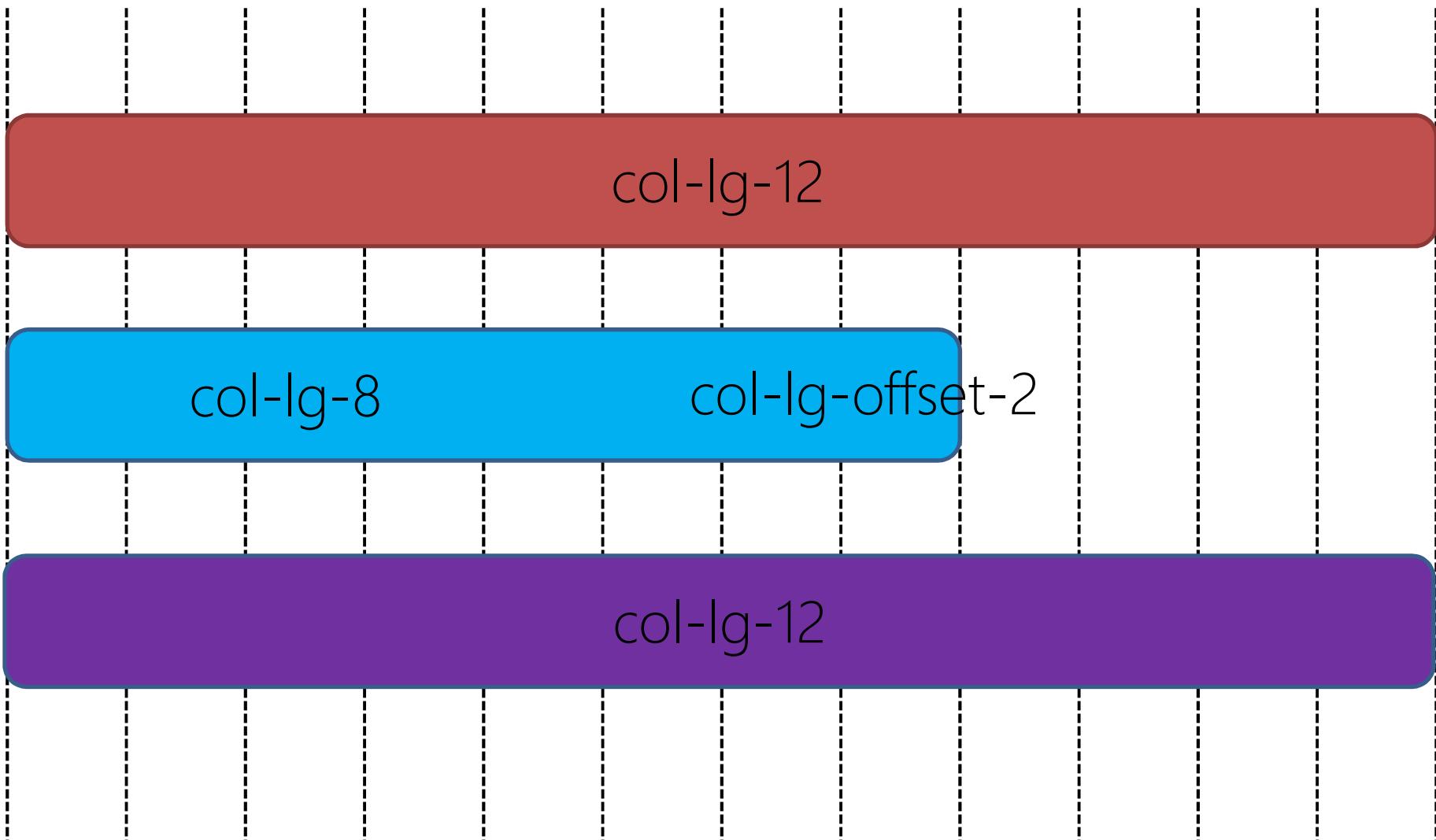
System siatkowy



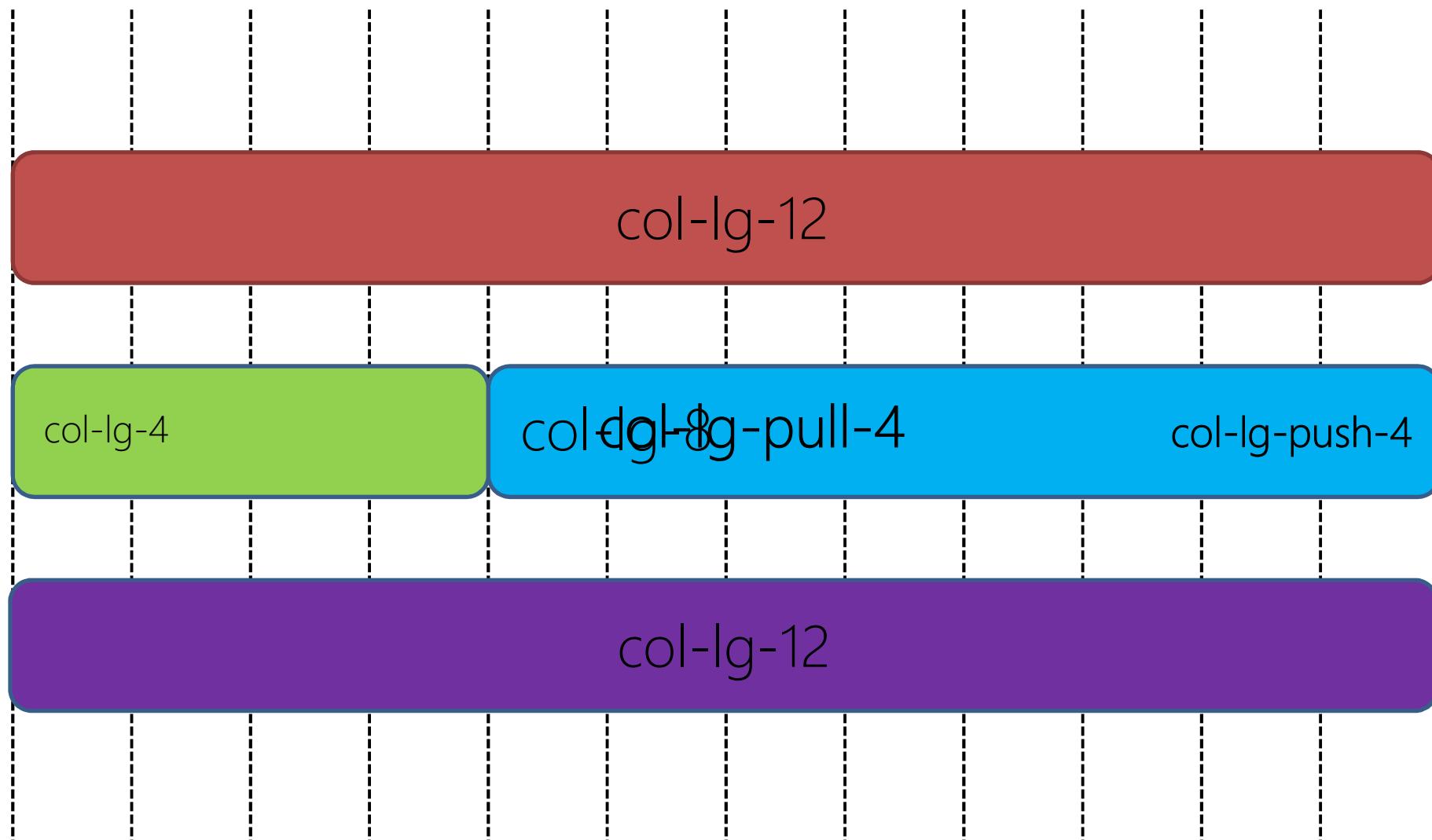
System siatkowy

```
<div class="container">
  <div class="row">
    <div class="col-lg-12">Nagłówek</div>
  </div>
  <div class="row">
    <div class="col-lg-4">Kolumna 1</div>
    <div class="col-lg-8">Kolumna 2</div>
  </div>
  ...
  <div class="row">
    <div class="col-lg-12">Stopka</div>
  </div>
</div>
```

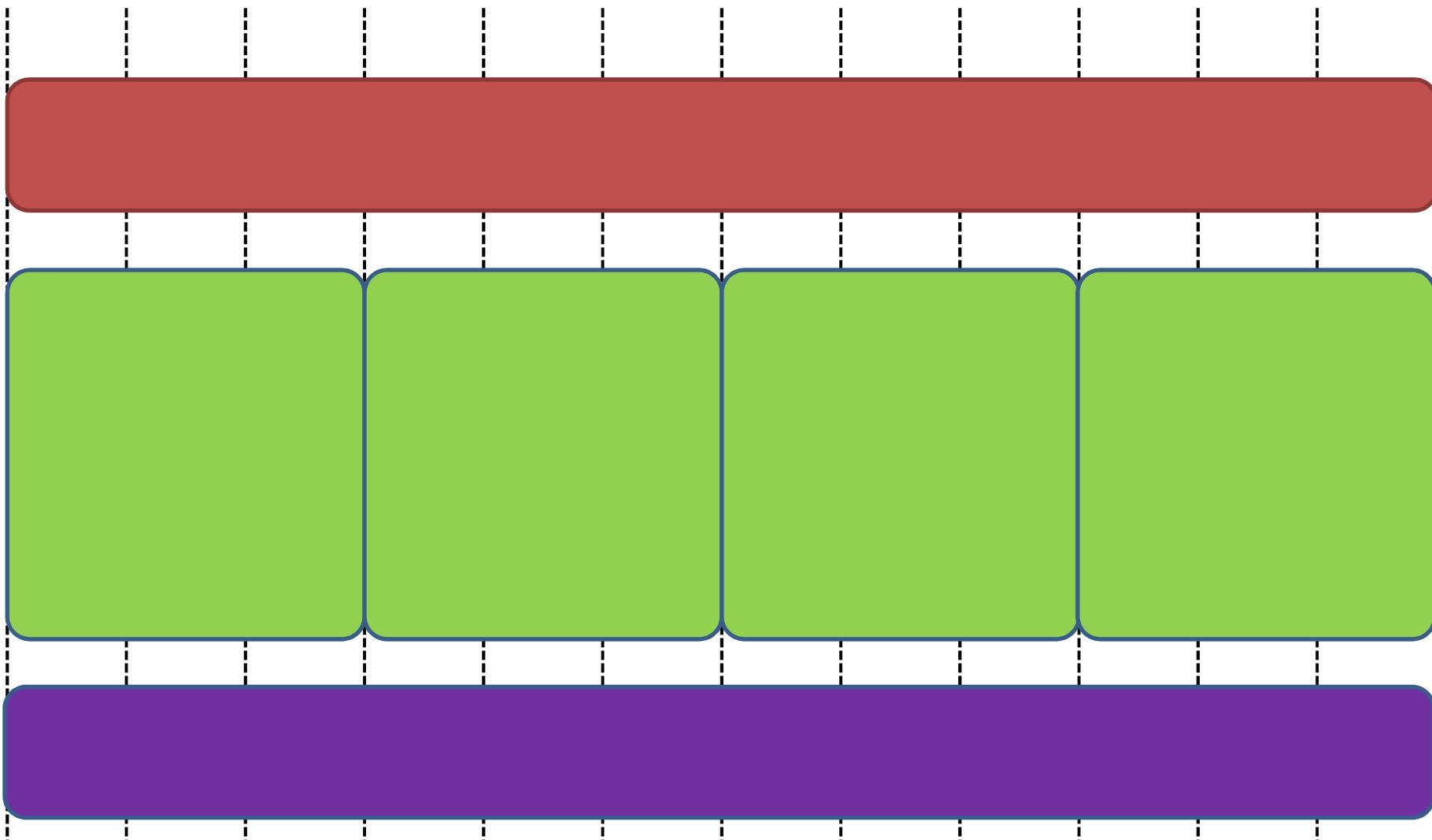
System siatkowy



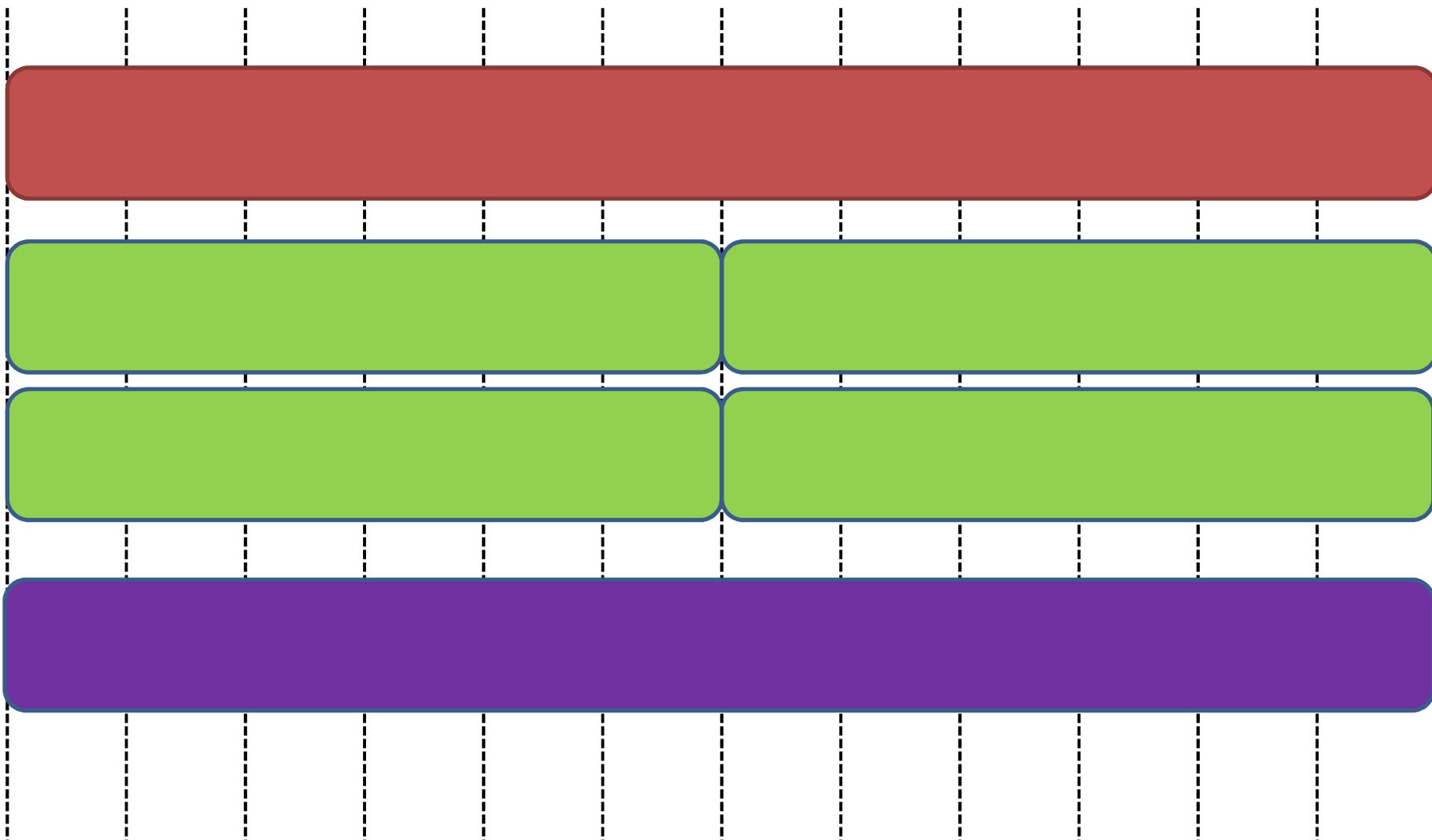
System siatkowy



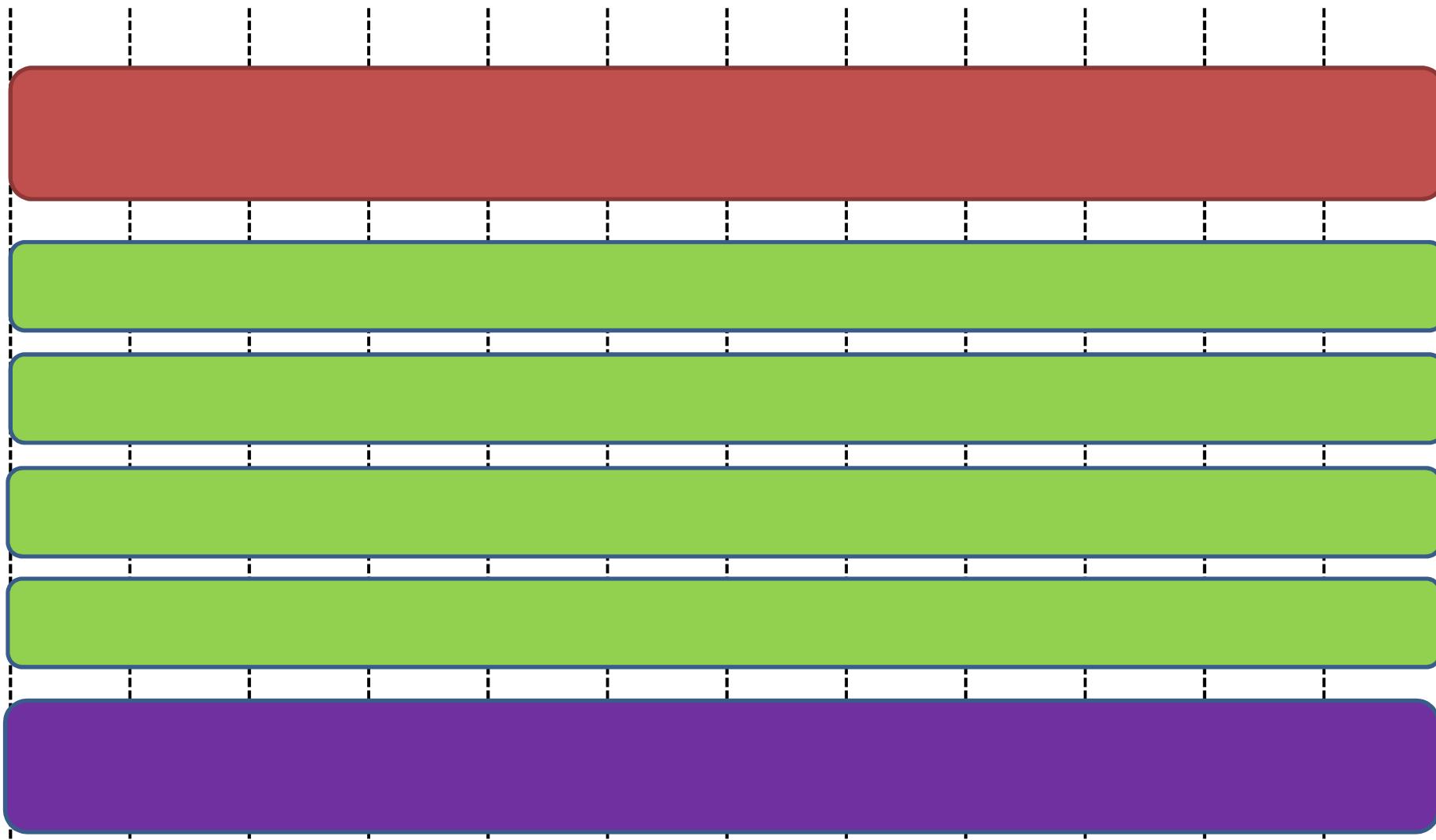
System siatkowy



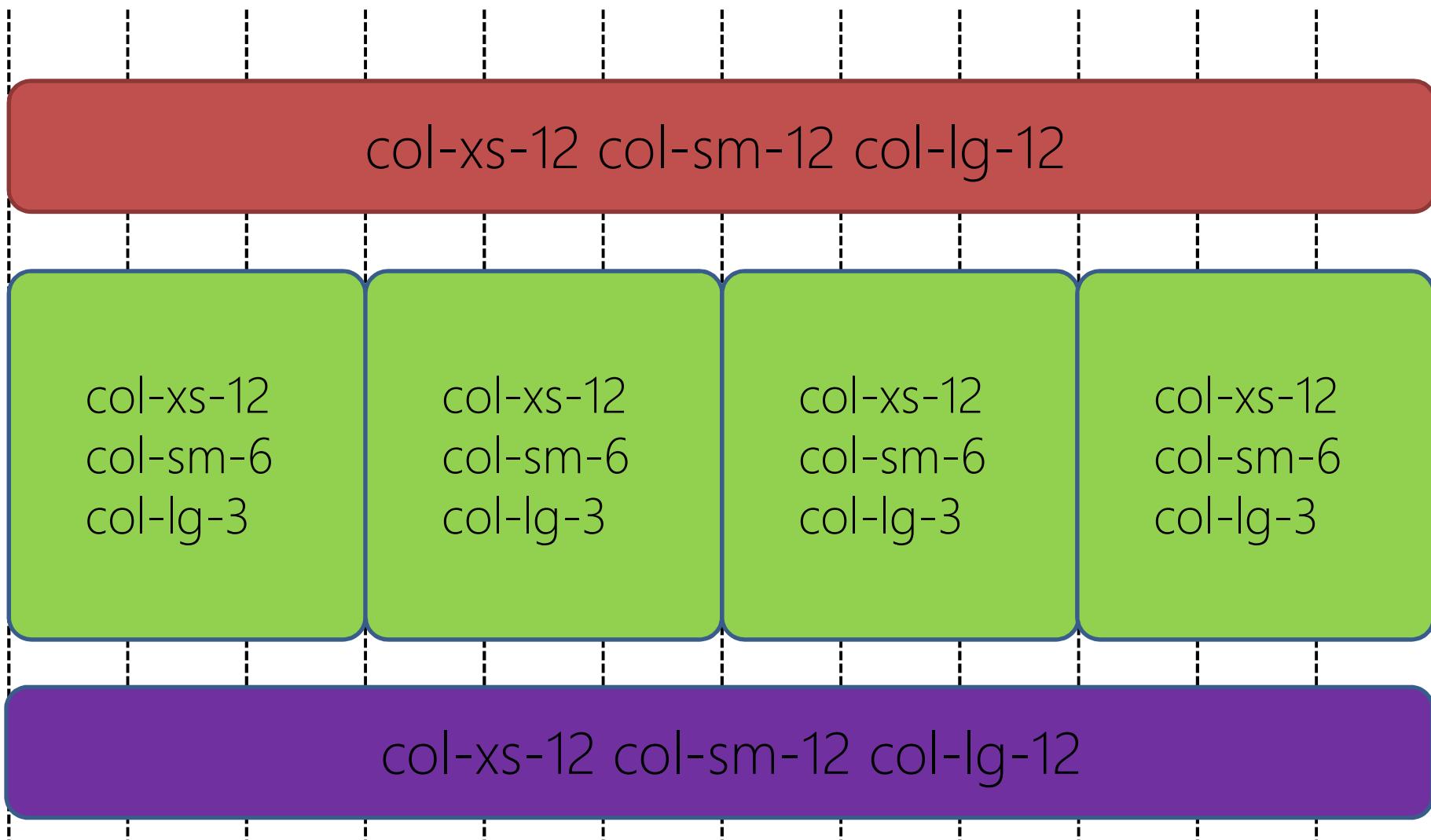
System siatkowy



System siatkowy



System siatkowy



System siatkowy

	Bardzo małe urządzenia / telefony (<768px)	Małe urządzenia / tablety ($\geq 768px$)	Średniej wielkości urządzenia / Desktopy ($\geq 992px$)	Dużej wielkości urządzenia / Desktopy ($\geq 1200px$)
Zachowanie siatki	Horizontal at all times		Collapsed to start, horizontal above breakpoints	
Container width	Automatyczna	750px	970px	1170px
Class prefix	.col-xs-	.col-sm-	.col-md-	.col-lg-
# of columns	12			
Column width	Automatyczna	60px	78px	95px
Gutter width	30px (15px on each side of a column)			

Komponenty

Default Primary Success Info Warning Danger

Home 42 Profile Messages 3

« 1 2 3 4 5 »

Tooltip on right Tooltip on right

Home Profile Messages

300x200

Thumbnail label

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Button Button

300x200

Thumbnail label

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Button Button

300x200

Thumbnail label

Cras justo odio, dapibus ac facilisis in, egestas eget quam. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Button Button

Well done! You successfully read this important alert message.

Heads up! This alert needs your attention, but it's not super important.

Warning! Better check yourself, you're not looking too good.

Oh snap! Change a few things up and try submitting again.

Collapsible Group Item #1

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

Collapsible Group Item #2

Collapsible Group Item #3

Modal title

One fine body...

Close Save changes

Komponenty - nagłówek

```
<div class="container">
  <h1 class="page-header">
    Moja pierwsza strona w Bootstrap
  </h1>
</div>
```

Moja pierwsza strona w Bootstrap

Komponenty - Jumbotron

```
<div class="container">
  <div class="jumbotron">
    <h1>Moja pierwsza strona w Bootstrap</h1>
    <p>Pierwsza informacja w jumbotron</p>
  </div>
</div>
```



Moja pierwsza strona
w Bootstrap

Pierwsza informacja w jumbotron

Komponenty - przyciski

```
<div class="btn">  
    To jest przycisk  
</div>
```

To jest przycisk

```
<div class="btn btn-default">  
    To jest przycisk  
</div>
```

To jest przycisk

btn-primary

btn-success

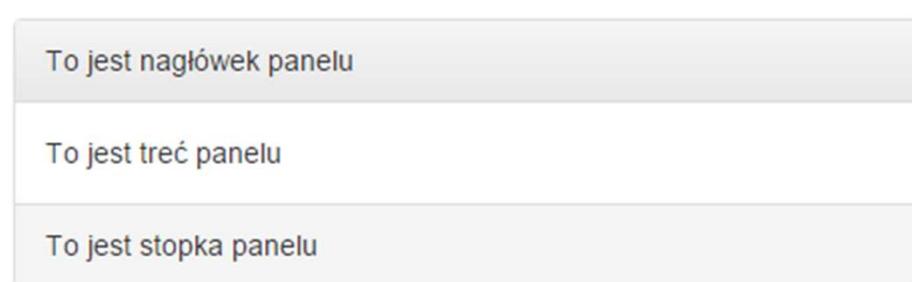
btn-warning

btn-info

btn-danger

Komponenty - panele

```
<div class="panel panel-default">
  <div class="panel-heading">
    To jest nagłówek panelu
  </div>
  <div class="panel-body">
    To jest treść panelu
  </div>
  <div class="panel-footer">
    To jest stopka panelu
  </div>
</div>
```



Komponenty - miniatury

```
<div class="col-lg-3">
  <div class="thumbnail">
    
    <div class="caption">
      <h3>Kuter torpedowy</h3>
      <p>Lorem ipsum...</p>
      <p><a href="#" class="btn btn-info">Przycisk</a></p>
    </div>
  </div>
</div>
```



Kuter torpedowy

Lorem ipsum...

Przycisk



Kuter torpedowy

Lorem ipsum...

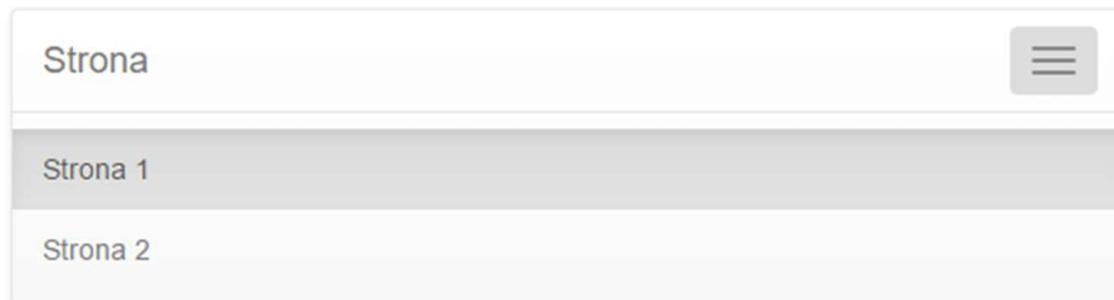
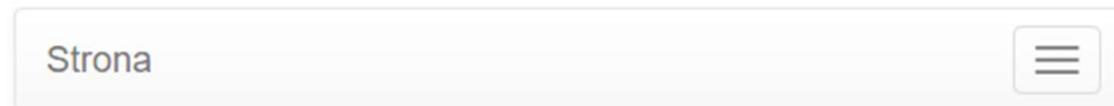
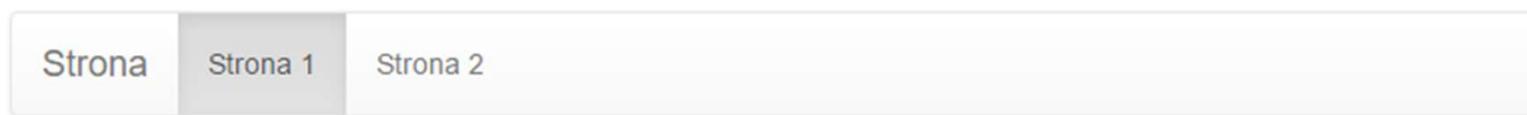
Przycisk

Komponenty - menu

```
<div class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <button class="navbar-toggle" data-toggle="collapse"
             data-target="#navbar-content">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Strona</a>
    </div>

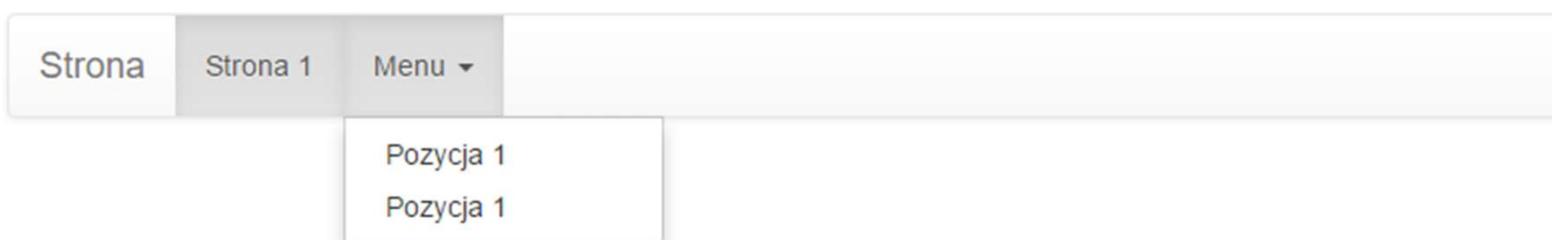
    <div class="collapse navbar-collapse" id="navbar-content">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Strona 1</a></li>
        <li class="active"><a href="#">Strona 2</a></li>
      </ul>
    </div>
  </div>
</div>
```

Komponenty - menu



Komponenty - menu

```
<li class="dropdown">
  <a href="#" class="dropdown-toggle" data-toggle="dropdown">
    Menu <b class="caret"></b>
  </a>
  <ul class="dropdown-menu">
    <li><a href="#">Pozycja 1</a></li>
    <li><a href="#">Pozycja 2</a></li>
    <li><a href="#">Pozycja 3</a></li>
    <li><a href="#">Pozycja 4</a></li>
  </ul>
</li>
```



Glyphikony

 glyphicon glyphicon-asterisk	 glyphicon glyphicon-plus	 glyphicon glyphicon-euro	 glyphicon glyphicon-minus	 glyphicon glyphicon-cloud	 glyphicon glyphicon-envelope	 glyphicon glyphicon-pencil	 glyphicon glyphicon-glass
 glyphicon glyphicon-music	 glyphicon glyphicon-search	 glyphicon glyphicon-heart	 glyphicon glyphicon-star	 glyphicon glyphicon-star-empty	 glyphicon glyphicon-user	 glyphicon glyphicon-film	 glyphicon glyphicon-th-large
 glyphicon glyphicon-th	 glyphicon glyphicon-th-list	 glyphicon glyphicon-ok	 glyphicon glyphicon-remove	 glyphicon glyphicon-zoom-in	 glyphicon glyphicon-zoom-out	 glyphicon glyphicon-off	 glyphicon glyphicon-signal
 glyphicon glyphicon-cog	 glyphicon glyphicon-trash	 glyphicon glyphicon-home	 glyphicon glyphicon-file	 glyphicon glyphicon-time	 glyphicon glyphicon-road	 glyphicon glyphicon-download-alt	 glyphicon glyphicon-download
 glyphicon glyphicon-upload	 glyphicon glyphicon-inbox	 glyphicon glyphicon-play-circle	 glyphicon glyphicon-repeat	 glyphicon glyphicon-refresh	 glyphicon glyphicon-list-alt	 glyphicon glyphicon-lock	 glyphicon glyphicon-flag

PRZECHOWYWANIE DANYCH PO STRONIE KLIENTA

Local Storage

- Rozszerzenie Cookies – większy rozmiar danych
- Local Storage / Session storage
- API:
 - storage.setItem(klucz, wartosc)
 - storage.getItem(klucz)
 - storage[klucz]
 - storage.key(index)
 - storage.length
 - storage.clear()

JSON

- JavaScript Object Notation
- Lekki format wymiany danych, niezależny od języka i łatwy do zrozumienia

```
{  
    "studenci": [  
        {"imie": "Jan", "nazwisko": "Nowak"},  
        {"imie": "Anna", "nazwisko": "Kowalska"},  
        {"imie": "Piotr", "nazwisko": "Janicki"}  
    ]  
}
```

JSON

- `JSON.Parse()` – zamienia łańcuch znaków zawierający zapis JSON na obiekt JavaScript
- `JSON.Stringify()` – zamienia obiekty JavaScript na ich zapis tekstowy w formacie JSON

IndexedDB

Jest to system przechowywania danych typu NoSQL pozwalający na umieszczanie dowolnych danych w przeglądarce internetowej użytkownika

IndexedDb

Baza danych

- Najwyższy poziom w IndexedDB
- Zawiera magazyny obiektów

Magazyn obiektów

- Koszyk do przechowywania danych
- Odpowiednik tabel z SQL
- Typowo występuje jeden dla każdego "typu danych"

Index

- Jest rodzajem obiektu do organizowania danych w innym magazynie danych poprzez określoną właściwość danych
- Jest wykorzystywany do dostępu do danych w magazynie

Operacja

- Interakcja z bazą danych

IndexedDb

Transakcja

- Jest wywoływana w celu zabezpieczenia jednej lub wielu operacji bazodanowych

Cursor

- Mechanizm iterowania po wielu rekordach w bazie danych

Otwarcie bazy danych

```
indexedDB.open(nazwaBazy, wersja, callback)
```

```
var obietnica = indexedDB.open("test",1)
```

Tworzenie magazynów danych

```
magazyn = createObjectStore(nazwaMagazynu)
```

- tworzenie magazynów jest możliwe tylko w funkcji callback przekazanej jako trzeci parametr funkcji open

```
var obietnica = indexedDB.open("test",1,  
    function(db) {  
        if(!db.objectStoreNames.contains('osoba'))  
            var magazyn = db.createObjectStore('osoba');  
    });
```

Definiowanie klucza głównego

```
var obietnica = indexedDB.open("test",1,  
    function(db) {  
        if(!db.objectStoreNames.contains('osoba'))  
            var magazyn = db.createObjectStore('osoba',  
                {  
                    keyPath:'id',  
                    autoIncrement: true  
                }  
            );  
    });
```

Definiowanie indeksu

```
magazyn.createIndex(nazwaIndeksu, nazwawlasciwosci, opcje)
```

- unique – indeks nie pozwala na duplikowanie wartości dla pojedynczego klucza
- multiEntry – (true) dodaje wpis w indeksie dla każdego elementu tablicy
(false) dodaje jeden wpis dla całej tablicy

```
var obietnica = indexedDB.open("test",1,
  function(db) {
    if(!db.objectStoreNames.contains('osoba')) {
      var magazyn = db.createObjectStore('osoba',
        { keyPath:'id', autoIncrement: true });
      magazyn.createIndex("nazwisko", "nazwisko", {unique:false})
    }
});
```

Praca z danymi

- Wszystkie operacje CRUD są asynchroniczne i wykorzystują obietnice
- Wszystkie operacje CRUD są wykonywane wewnątrz transakcji
- Każda operacja wymaga następującej formy:
 1. pobrania obiektu bazy danych
 2. otwarcia transakcji wewnątrz bazy
 3. otwarcia magazynu wewnątrz transakcji
 4. wykonania operacji na magazynie

Transakcje

```
var tr = db.transaction([magazyny i indeksy], tryb);
```

- Rozpoczęcie transakcji
- Tryb transakcji: "readonly", "readwrite"

```
transaction.complete
```

- Zwraca obietnicę
- jest ona rozwiązywana jeśli wszystkie operacje wykonane w ramach transakcji zakończą się poprawnie
- jest ona odrzucana, jeśli którakolwiek operacja wykonana w ramach transakcji zakończy się błędem

Tworzenie danych

```
magazyn.add(obiekt)
```

```
obietnica.then(function (db) {
    var trans = db.transaction('osoba', 'readwrite');
    var magazyn = trans.objectStore('osoba');
    var osoba = {
        imie: 'Ala',
        nazwisko: 'Kowalska',
        wiek: 23
    };
    magazyn.add(osoba);
    return trans.complete;
}).then(function () {
    console.log('dodałem wpis do bazy');
})
```

Pobieranie danych

```
magazyn.get(wartośćKluczaGłównego)
```

```
obietnica.then(function (db) {  
    var trans = db.transaction('osoba', 'readwrite');  
    var magazyn = trans.objectStore('osoba');  
    return magazyn.get(12);  
}).then(function (osoba) {  
    console.dir(osoba);  
})
```

Modyfikowanie danych

```
magazyn.put(obiekt)
```

```
obietnica.then(function (db) {
    var trans = db.transaction('osoba', 'readwrite');
    var magazyn = trans.objectStore('osoba');
    var osoba = {
        id: 12,
        imie: 'Ala',
        nazwisko: 'Kowalska',
        wiek: 23
    };
    magazyn.put(osoba);
    return trans.complete;
}).then(function () {
    console.log('dodałem wpis do bazy');
})
```

Usuwanie danych

```
magazyn.delete(wartośćKluczaGłównego)
```

```
obietnica.then(function (db) {  
    var trans = db.transaction('osoba', 'readwrite');  
    var magazyn = trans.objectStore('osoba');  
    return magazyn.delete(12);  
}).then(function (osoba) {  
    console.dir(osoba);  
})
```

Pobieranie wszystkich danych

```
magazyn.getAll()
```

```
obietnica.then(function (db) {
  var trans = db.transaction('osoba', 'readwrite');
  var magazyn = trans.objectStore('osoba');
  return magazyn.getAll();
}).then(function (osoby) {
  console.dir(osoby);
})
```

Praca z kursorami

```
magazyn.openCursor(opcjonalnyZakres, opcjonalnyKierunek)
```

- kierunek: "next", "nextunique", "prev", "prevunique"

```
kursor.continue()
```

- przesuwa kursor na następną pozycję lub zwraca undefined, jeżeli nie ma już następnego obiektu

Praca z kursorami

```
obietnica.then(function (db) {
    var trans = db.transaction('osoba', 'readwrite');
    var magazyn = trans.objectStore('osoba');
    return magazyn.openCursor();

}).then(function przetworz(kursor) {
    if(!cursor) return;
    console.dir(osoba);
    return kursor.continue().then(przetworz);

}).then(function () {
    console.log("Koniec");
})
```

Wykorzystanie indeksów

Indeksy pozwalają na wybór zakresu danych względem jednej z właściwości danych.

Wykorzystują obiekt IDBKeyRange

Obiekt IDBKeyRange posiada 4 metody

- `IDBKeyRange.upperBound(x, exclusive)`
- `IDBKeyRange.lowerBound(x, exclusive)`
- `IDBKeyRange.bound(x,y, exclusiveX, exclusiveY)`
- `IDBKeyRange.only(x)`

Wykorzystanie indeksów

```
obietnica.then(function (db) {
    var trans = db.transaction('osoba', 'readwrite');
    var magazyn = trans.objectStore('osoba');
    var indeks = magazyn.index('nazwisko');
    var zakres = IDBKeyRange.lowerBound('Kowalska');
    return magazyn.openCursor(zakres);

}).then(function przetworz(kursor) {
    if(!kursor) return;
    console.dir(osoba);
    return kursor.continue().then(przetworz);

}).then(function () {
    console.log("Koniec");
})
```

ARCHITEKTURA APLIKACJI JAVASCRIPT

Prosta aplikacja

id	imie	nazwisko	
1	Ala	Nowak	<button>edytuj</button> <button>usuń</button>
2	Tomasz	Nowak	<button>edytuj</button> <button>usuń</button>

id

imie

nazwisko

Zapisz **Anuluj**

Prosta aplikacja - html

```
<body>
  <div>
    <table>
      <thead>
        <tr><th>id</th><th>imie</th><th>nazwisko</th></tr>
      <thead>
        <tbody></tbody>
      </table>
    </div>
    <div>
      <form id="formularz">
        <div><label>id</label><input id="id" /></div>
        <div><label>imie</label><input id="imie" /></div>
        <div><label>nazwisko</label><input id="nazwisko" /></div>
        <div><a href="#" id="zapisz">Zapisz</a>
          <a href="#" id="anuluj">Anuluj</a></div>
      </form>
    </div>
  </body>
```

Prosta aplikacja – JS (1)

```
$(function() {
    Inicjalizuj();
    $("#zapisz").click(Zapisz);
    $("#anuluj").click(Wyczysc);
    $(document).on("click", ".usun", Usun);
    $(document).on("click", ".edytuj", Edytuj);
});
```

Prosta aplikacja – JS (2)

```
function Inicjalizuj() {
    for(var i =0; i<window.localStorage.length; i++) {
        var klucz = window.localStorage.key(i);
        var obiekt = JSON.parse(window.localStorage.getItem(klucz));
        GenerujWiersz(klucz, obiekt.imie, obiekt.nazwisko)
            .appendTo("table");
    }
}

function GenerujWiersz(id, imie, nazwisko) {
    return $(<tr data-id="' + id + '"><td>' + id + '</td>' +
        '<td>' + imie + '</td>' +
        '<td>' + nazwisko + '</td>' +
        '<td>' +
        '<a href="#" class="edytuj" data-id="' + id + '">' + 'edytuj' + '</a>' +
        '</td><td>' +
        '<a href="#" class="usun" data-id="' + id + '">' + 'usuń' + '</a>' +
        '</td></tr>')
}
```

Prosta aplikacja – JS (3)

```
function Usun() {
    var id = $(this).data("id");
    $("tr[data-id='"+id+"']").remove();
    window.localStorage.removeItem(id);
}

function Edytuj() {
    dodawanie = false;
    var obj = JSON.parse(
        window.localStorage.getItem($(this).attr("data-id")));
    $("#id").val($(this).attr("data-id"));
    $("#imie").val(obj.imie);
    $("#nazwisko").val(obj.nazwisko);
}

function Wyczysc() {
    $("#formularz")[0].reset();
}
```

Prosta aplikacja – JS (4)

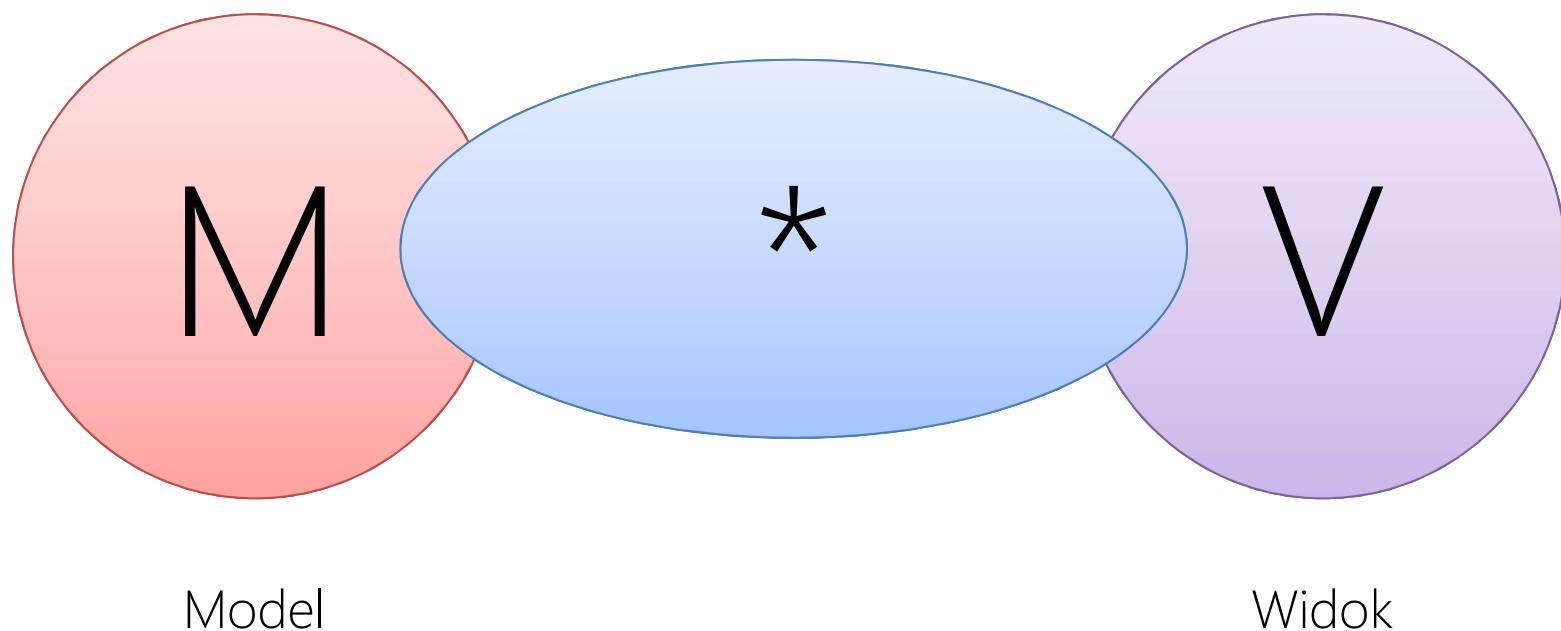
```
function Zapisz() {
    var id = $("#id").val();
    var imie = $("#imie").val();
    var nazwisko = $("#nazwisko").val();

    if(dodawanie)
        GenerujWiersz(id, imie, nazwisko).appendTo("table");
    else
        GenerujWiersz(id, imie, nazwisko).replaceAll("tr[data-id='"+id+"']");

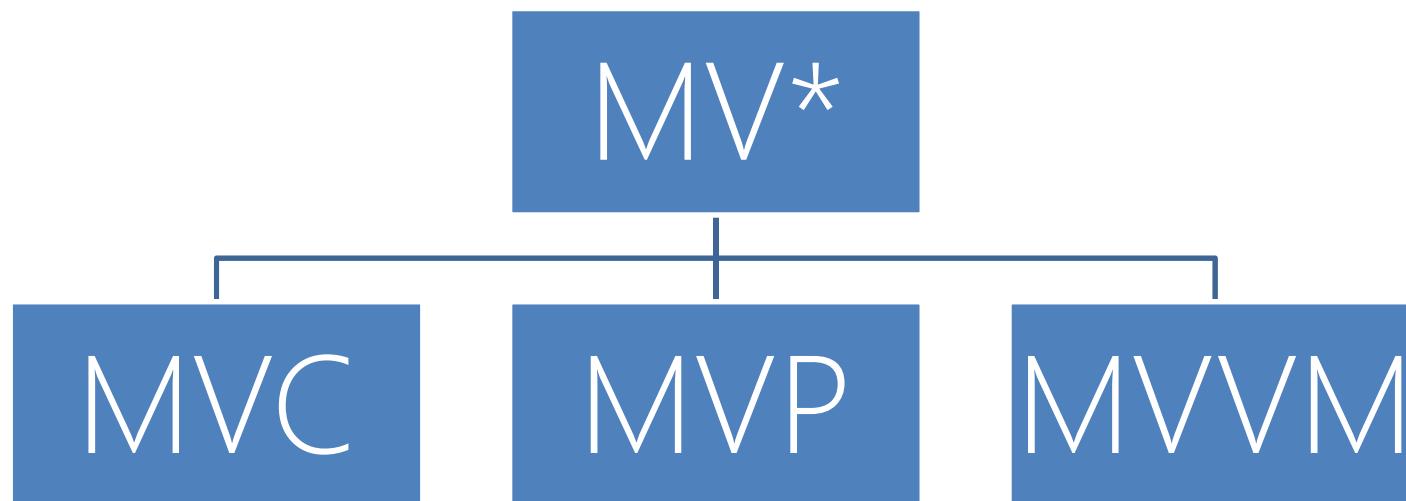
    window.localStorage.setItem(id,
        JSON.stringify({ imie : imie, nazwisko : nazwisko}));
    $("#formularz")[0].reset();
    dodawanie = true;
}
```

Wzorce architektoniczne MV*

Wzorce architektoniczne mają za zadanie uprościć tworzenie i poprawić organizację aplikacji poprzez rozdzielenie danych i procesów biznesowych od interfejsu użytkownika.



Wzorce architektoniczne MV*



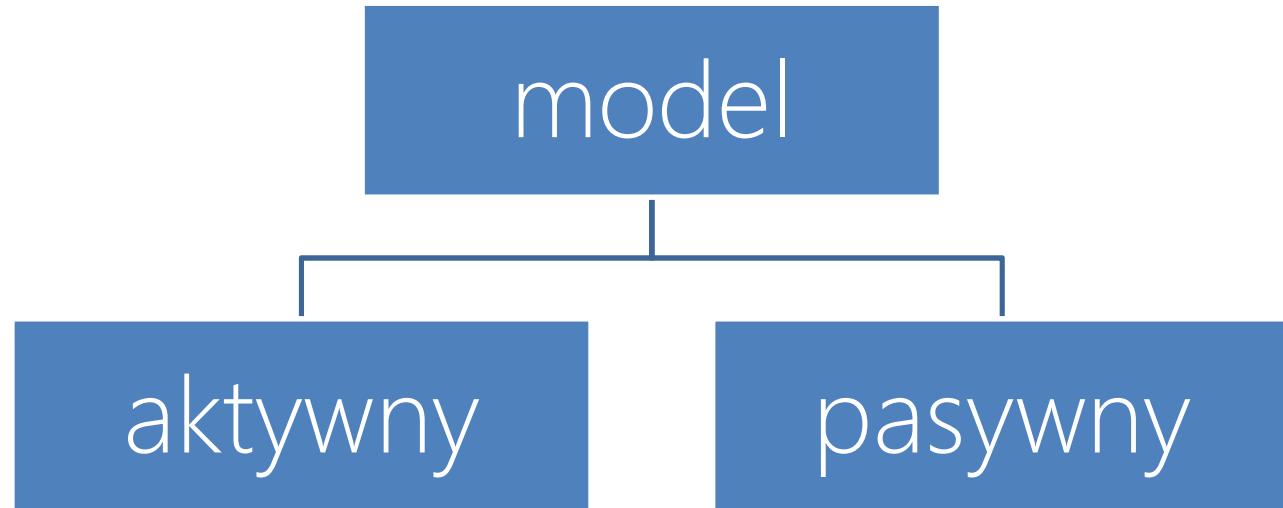
Wzorzec MVC (1)

Wzorzec MVC (Model – View – Controller)

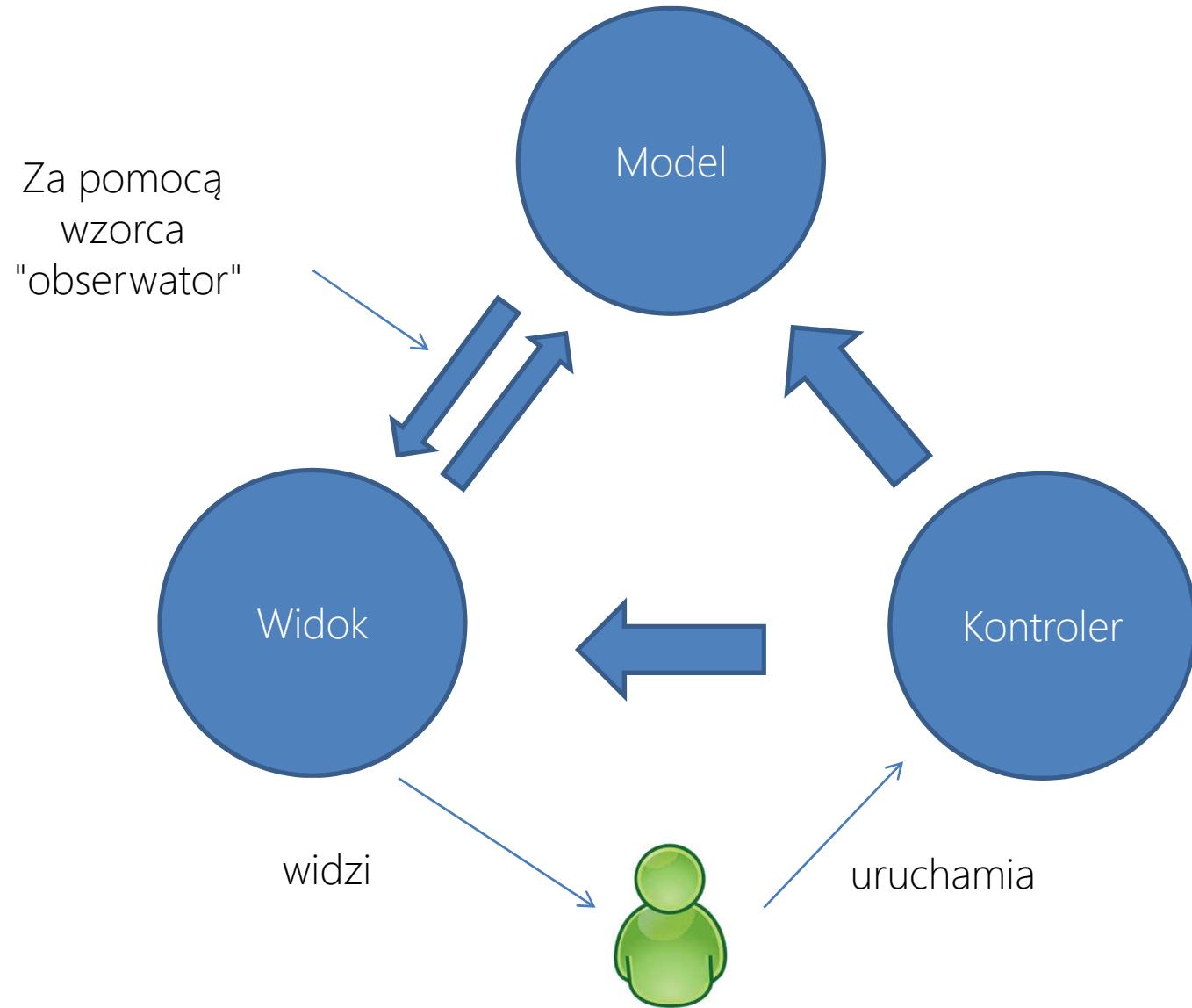
Pozwala podzielić aplikację na trzy osobne części:

- Model – zawiera całą logikę biznesową aplikacji
- Widok – zawiera informacje jak wyświetlić model
- Kontroler – obsługuje żądania użytkownika (zarządza zmianami modelu i widoków)

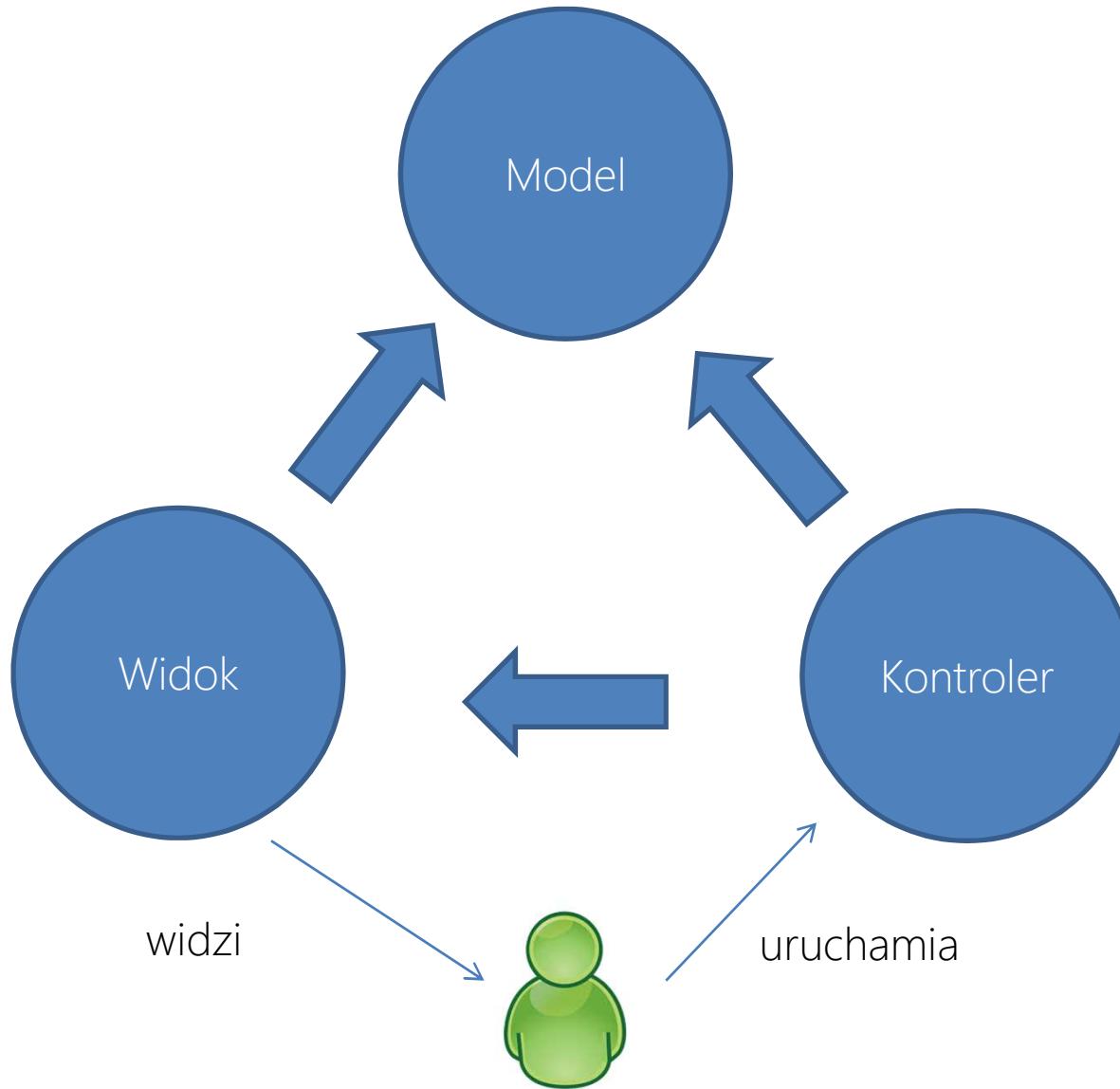
Wzorzec MVC (2)



Wzorzec MVC (3) – model aktywny



Wzorzec MVC (4) – model pasywny



Wzorzec MVP (1)

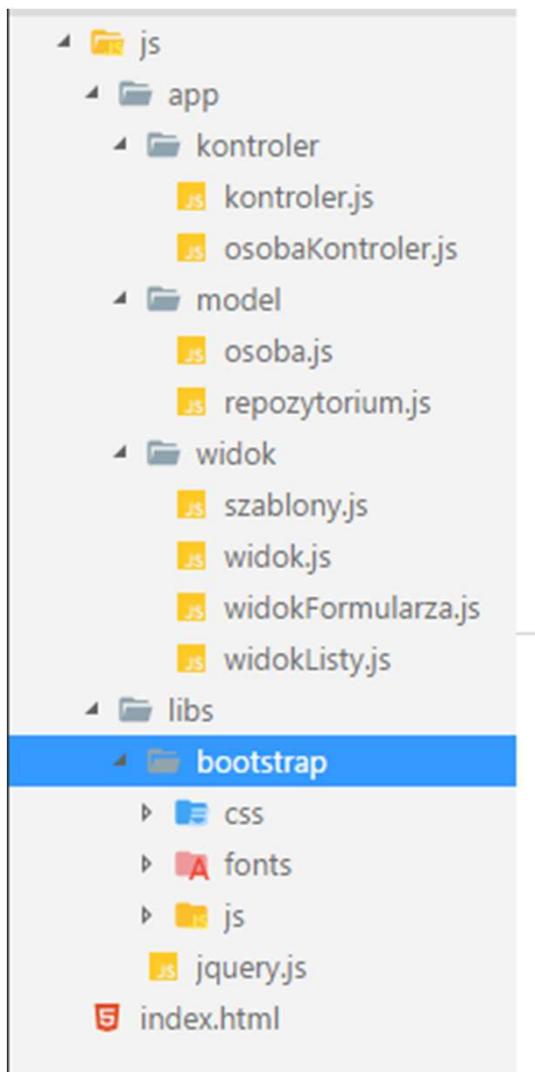
Wzorzec MVP (Model – View – Presenter)

Modyfikacja wzorca MVC.

Pozwala podzielić aplikację na trzy osobne części:

- Model – zawiera całą logikę biznesową aplikacji
- Widok – zawiera informacje jak wyświetlić model
- Prezenter – łączy widok z modelem (przy czym model nie może bezpośrednio modyfikować widoku)

Przykład prostej aplikacji



Przykład prostej aplikacji (1)

```
class Osoba {  
  
    constructor(id, imie, nazwisko) {  
        this.id = id;  
        this.imie = imie;  
        this.nazwisko = nazwisko;  
    }  
  
    JakoJSON() {  
        return JSON.stringify(this);  
    }  
  
    static StworzZJson(json) {  
        var obj = JSON.parse(json);  
        return new Osoba(obj.id, obj.imie, obj.nazwisko);  
    }  
}
```

Przykład prostej aplikacji (2)

```
class Repozytorium {
    constructor(model) {
        this.storage = window.localStorage;
        this.model = model;
    }

    PobierzWszystkie() {
        return new Promise((resolve, reject) => {
            var data = [];
            for(var i=0; i<this.storage.length; i++) {
                var klucz = this.storage.key(i);
                var osoba = this.model.StworzZJson(this.storage[klucz]);
                data.push(osoba);
            }
            resolve(data);
        });
    }
}
```

Przykład prostej aplikacji (3)

```
PobierzWgId(id) {
    return new Promise((resolve, reject) => {
        var osoba = this.storage[id];
        if(osoba) resolve(this.model.StworzZJson(osoba));
        else reject(id);
    });
}

Dodaj(obiekt) {
    return new Promise((resolve, reject) => {
        this.storage.setItem(obiekt.id, obiekt.JakoJSON());
        resolve();
    });
}
```

Przykład prostej aplikacji (4)

```
Edytuj(obiekt) {
    return new Promise((resolve, reject) => {
        this.storage.setItem(obiekt.id, obiekt.JakoJSON());
        resolve();
    });
}

Usun(id) {
    return new Promise((resolve, reject) => {
        this.storage.removeItem(id);
        resolve();
    })
}
```

Przykład prostej aplikacji (5)

```
class Widok
{
    GenerujSzablon(szablon, obiekt) {
        for(var klucz in obiekt)
            szablon = szablon.replace(
                new RegExp("\\{" + klucz + "\\}", "g"), obiekt[klucz] || "");
        return szablon;
    }
}
```

Przykład prostej aplikacji (6)

```
class WidokListy extends Widok
{
    constructor(elementGlowny, szablonTabeli, szablonWiersza) {
        super();
        this.elementGlowny = elementGlowny;
        this.szablonTabeli = szablonTabeli;
        this.szablonWiersza = szablonWiersza;
    }

    Generuj(dane) {
        var element = $(this.szablonTabeli);
        var tabela = element.find("table tbody");
        for(var i = 0; i<dane.length; i++) {
            var wiersz = $(this.GenerujSzablon(
                this.szablonWiersza, dane[i]));
            wiersz.appendTo(tabela);
        }
        $("#" + this.elementGlowny).html(element);
        this.element = element;
        this.wiadomosc = element.find("div");
        this.wiadomoscTresc = this.wiadomosc.html();
    }
}
```

Przykład prostej aplikacji (7)

```
PokazKomunikat(id) {
    this.wiadomosc.html(
        this.GenerujSzablon(this.wiadomoscTresc, {id: id})
    );
    if(!this.timer)
        this.wiadomosc.show(200);
    else
        clearTimeout(this.timer);
    this.timer = setTimeout(()=>{
        this.wiadomosc.hide(500);
        this.timer = null;
    }, 2000);
}
```

Przykład prostej aplikacji (8)

```
class WidokFormularza extends Widok {  
    constructor(elementGlowny, szablonFormularza) {  
        super();  
        this.elementGlowny = elementGlowny;  
        this.szablonFormularza = szablonFormularza;  
    }  
  
    Generuj(dane) {  
        $("#" + this.elementGlowny).html(  
            this.GenerujSzablon(this.szablonFormularza, dane));  
    }  
  
    PobierzDane(obiekt) {  
        for(var klucz in obiekt) {  
            if(obiekt.hasOwnProperty(klucz)) {  
                var nazwa = "#" + klucz;  
                var wartosc = $(nazwa).val();  
                obiekt[klucz] = wartosc;  
            }  
        }  
        return obiekt;  
    }  
}
```

Przykład prostej aplikacji (9)

```
class Kontroler {
    constructor() {
        this.akcje = {};
    }

    RejestrujAkcje(nazwa, funkcja) {
        this.akcje[nazwa] = funkcja;
    }

    InicjujAkcje() {
        for(var klucz in this.akcje) {
            var nazwa = "."+klucz;
            var funkcja = this.akcje[klucz];
            let k = klucz;
            $(document).on("click", "."+klucz, e=>{
                e.preventDefault();
                console.log(k);
                this.akcje[k]($(e.target))
            })
        }
    }
}
```

Przykład prostej aplikacji (10)

```
class OsobaKontroler extends Kontroler {
    constructor() {
        super();
        this.repozytorium = new Repozytorium(Osoba);
        this.widokFormularza = new WidokFormularza("main",
            Szablony.SzablonFormularza());
        this.widokListy = new WidokListy("main",
            Szablony.SzablonListy(), Szablony.SzablonOsoby());
        this.dodawanie = true;
        this.init();
    }

    dodaj() {
        this.dodawanie = true;
        var osoba = new Osoba();
        this.widokFormularza.Generuj(osoba);
    }

    usun(nadawca) {
        var id = nadawca.data("id");
        this.repozytorium.Usun(id).then( () => this.index() );
    }
}
```

Przykład prostej aplikacji (11)

```
edytuj(nadawca) {
    this.dodawanie = false;
    var id = nadawca.data("id");
    this.repozytorium.PobierzWgId(id)
        .then((osoba)=>{ this.widokFormularza.Generuj(osoba); })
        .catch((id)=> { this.widokListy.PokazKomunikat(id); });
}

zatwierdz() {
    var obietnica;
    if(this.dodawanie) {
        var osoba = this.widokFormularza.PobierzDane(new Osoba());
        obietnica = this.repozytorium.Dodaj(osoba);
    } else {
        var osoba = this.widokFormularza.PobierzDane(new Osoba());
        obietnica = this.repozytorium.Edytuj(osoba);
    }
    obietnica.then( ()=>this.index() );
}
```

Przykład prostej aplikacji (12)

```
anuluj() { this.index(); }

index() {
    this.repozytorium.PobierzWszystkie().then( (osoby) => {
        this.widokListy.Generuj(osoby);
    });
}

init() {
    this.RejestrujAkcje("Index", (e) => this.index());
    this.RejestrujAkcje("Zatwierdz", (e) => this.zatwierdz());
    this.RejestrujAkcje("Dodaj", (e) => this.dodaj());
    this.RejestrujAkcje("Usun", (e) => this.usun(e));
    this.RejestrujAkcje("Edytuj", (e) => this.edytuj(e));
    this.RejestrujAkcje("Anuluj", (e) => this.anuluj());
}

Inicjalizuj() {
    this.InicjujAkcje();
    this.akcje.Index();
}
}
```

Przykład prostej aplikacji (13)

```
var Szablony = {};  
  
Szablony.SzablonFormularza = function () {  
    return `  
        <form class="form-horizontal">  
            <div class="form-group">  
                <label class="control-label col-xs-3">Id</label>  
                <div class="col-xs-9">  
                    <input type="text" name="Id" id="id"  
                        class="form-control" value="{id}" />  
                </div>  
            </div>  
            <div class="form-group">  
                <label class="control-label col-xs-3">Imię</label>  
                <div class="col-xs-9">  
                    <input type="text" name="Imie" id="imie"  
                        class="form-control" value="{imie}" />  
                </div>  
            </div>  
        </form>  
    `};
```

Przykład prostej aplikacji (13)

```
<div class="form-group">
    <label class="control-label col-xs-3">Nazwisko</label>
    <div class="col-xs-9">
        <input type="text" name="Nazwisko" id="nazwisko"
               class="form-control" value="{nazwisko}" />
    </div>
</div>
<div>
    <a href="#" class="Zatwierdz btn btn-success">OK</a>
    <a href="#" class="Anuluj btn btn-warning">Anuluj</a>
</div>
</form>`;
}
```

Przykład prostej aplikacji (14)

```
Szablony.SzablonListy = function() {
    return `
        <div>
            <a href="#" class="Dodaj btn btn-xs btn-primary">Nowa osoba</a>
            <div class="alert alert-danger" style="display:none">
                Osoba o identyfikatorze {id} nie istnieje w bazie
            </div>
            <table class="table table-striped">
                <thead>
                    <tr><th>Id</th><th>Imię</th><th>Nazwisko</th><th></th></tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>`;
}
```

Przykład prostej aplikacji (15)

```
Szablony.SzablonOsoby = function() {
    return `<tr>
        <td>{id}</td><td>{imie}</td><td>{nazwisko}</td>
        <td><a href="#" class="Edytuj btn btn-xs btn-default"
            data-id="{id}">Edytuj</a>
            <a href="#" class="Usun btn btn-xs btn-warning"
            data-id="{id}">Usuń</a>
        </td>
    </tr>`;
}
```

Przykład prostej aplikacji (16)

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="js/libs/bootstrap/css/bootstrap-theme.css"/>
    <link rel="stylesheet" href="js/libs/bootstrap/css/bootstrap.css"/>
</head>
<body>
    <div class="container">
        <div class="row">
            <div id="main" class="col-xs-12"></div>
        </div>
    </div>
    <script type="text/javascript" src="js/libs/jquery.js"></script>
    <script type="text/javascript" src="js/libs/bootstrap/js/bootstrap.js">
    </script>
```

Przykład prostej aplikacji (17)

```
<script type="text/javascript" src="js/app/Model/Osoba.js"></script>
<script type="text/javascript" src="js/app/Model/repozytorium.js"></script>
<script type="text/javascript" src="js/app/Widok/Widok.js"></script>
<script type="text/javascript" src="js/app/Widok/WidokFormularza.js">
    </script>
<script type="text/javascript" src="js/app/Widok/WidokListy.js"></script>
<script type="text/javascript" src="js/app/Widok/Szablony.js"></script>
<script type="text/javascript" src="js/app/Kontroler/Kontroler.js"></script>
<script type="text/javascript" src="js/app/Kontroler/OsobaKontroler.js">
    </script>
<script type="text/javascript">
$(function() {
    var kontroler = new OsobaKontroler();
    kontroler.Inicjalizuj();
});
</script>
</body>
</html>
```

Knockout.

WZORZEC MVVM I BIBLIOTEKA
KNOCKOUT

Prosta aplikacja

id	imie	nazwisko		
1	Ala	Nowak	edytuj	usuń
2	Tomasz	Nowak	edytuj	usuń

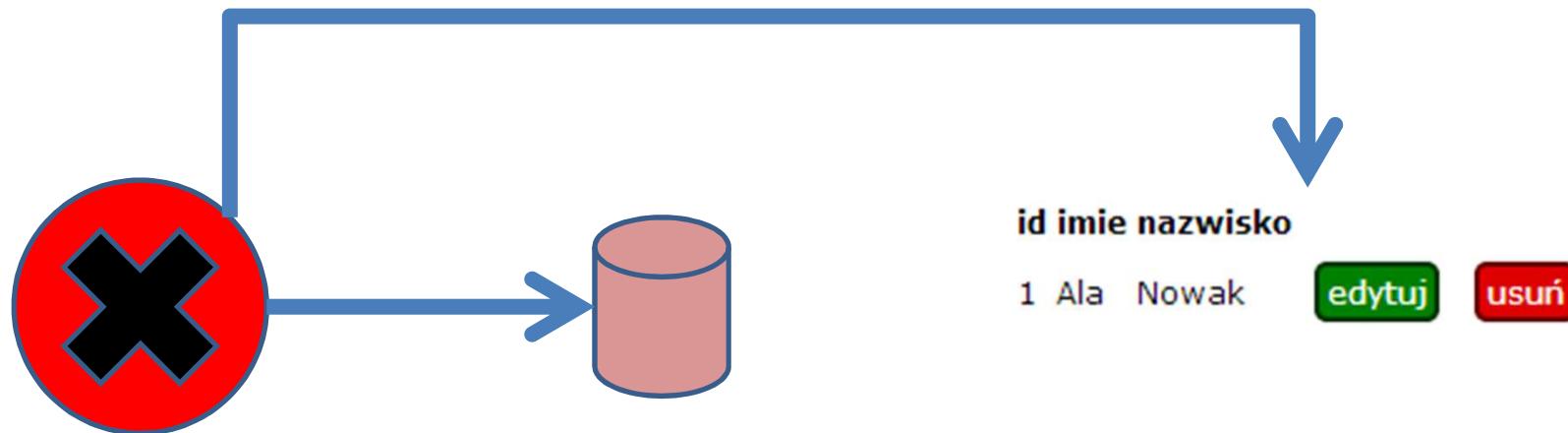
id

imie

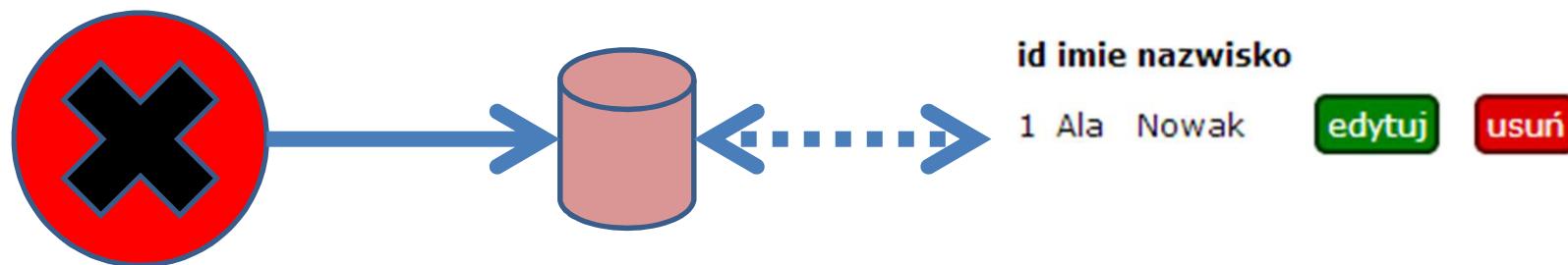
nazwisko

[Zapisz](#) [Anuluj](#)

Usuwanie danych



Usuwanie danych



Wzorzec MVVM

Kolejna wersja wzorca MVC

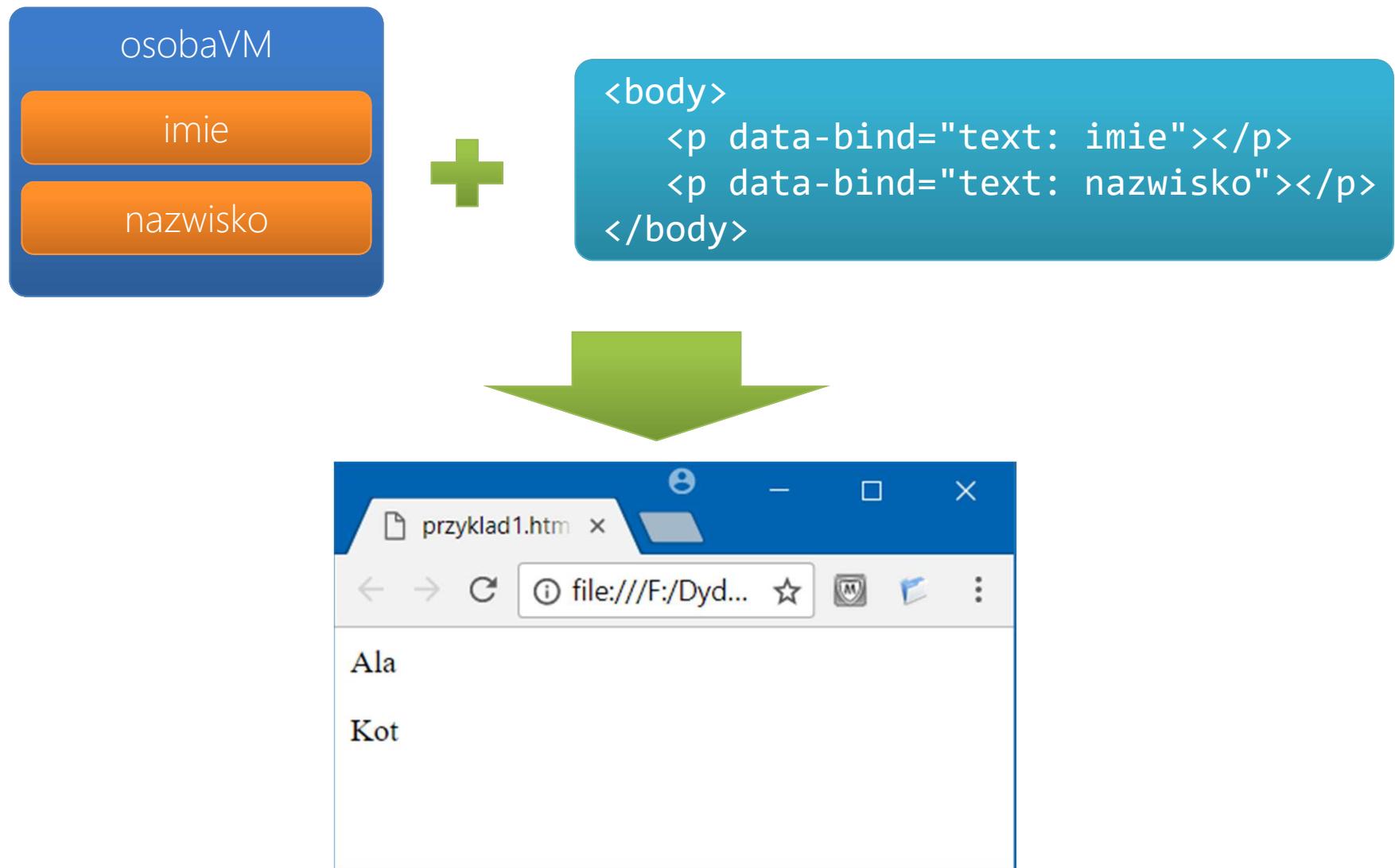
- Model (*Model*) – zawiera dane aplikacji i reguły biznesowe jakie działają na tych danych
- Widok (*View*) – sposób przedstawienia danych
- Widok Modelu (*ViewModel*) – reprezentacja danych koniecznych do wyświetlenia widoku oraz metody konieczne do jego obsługi

Knockout JS

Biblioteka dla języka JavaScript umożliwiająca prostą realizację aplikacji zgodnie z wzorcem MVVM

- Observables – właściwości obiektu typu VlewModel, które pozwalają bibliotece Knockout śledzić zmiany modelu i odświeżać widok.
- Bindings – mechanizm pozwalający połączyć model widoku z widokiem. Dzięki połączeniu elementu HTML z właściwością typu observables, Knockout może automatycznie oświeżyć widok.

Najprostszy model widoku



Najprostszy model widoku

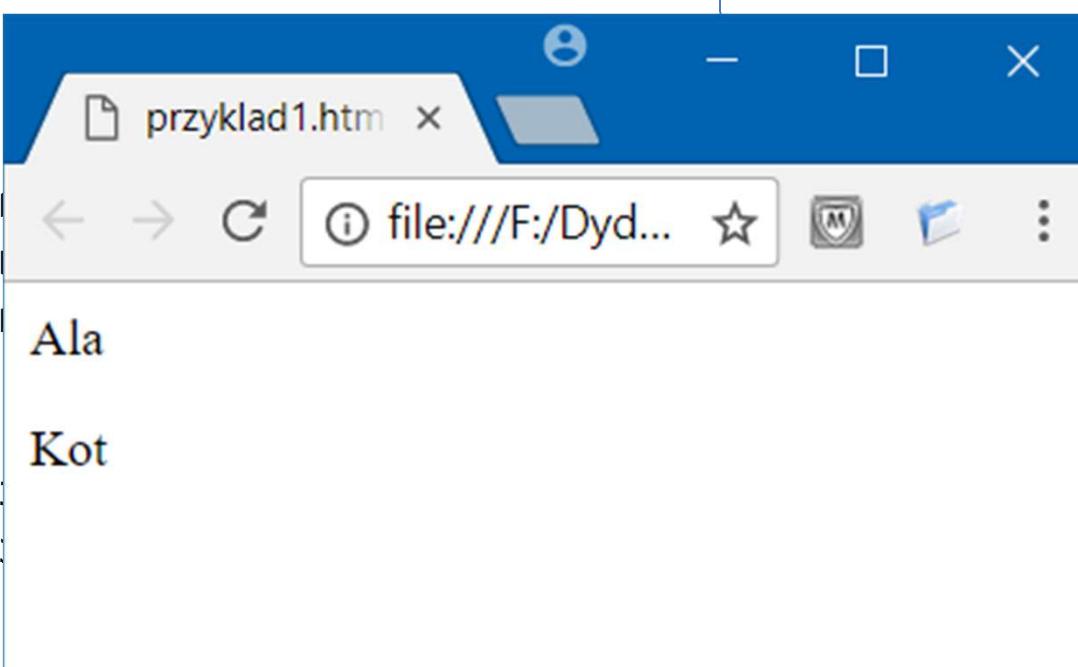
```
<html>
<head>
  <script src="jquery.js"></script>
  <script src="knockout.js"></script>
  <script src="app1.js"></script>
</head>
<body>
  <p data-bind="text: imie"></p>
  <p data-bind="text: nazwisko"></p>
</body>
</html>
```

```
var osobaVM = {
  imie : "Ala",
  nazwisko: "Kot"
};

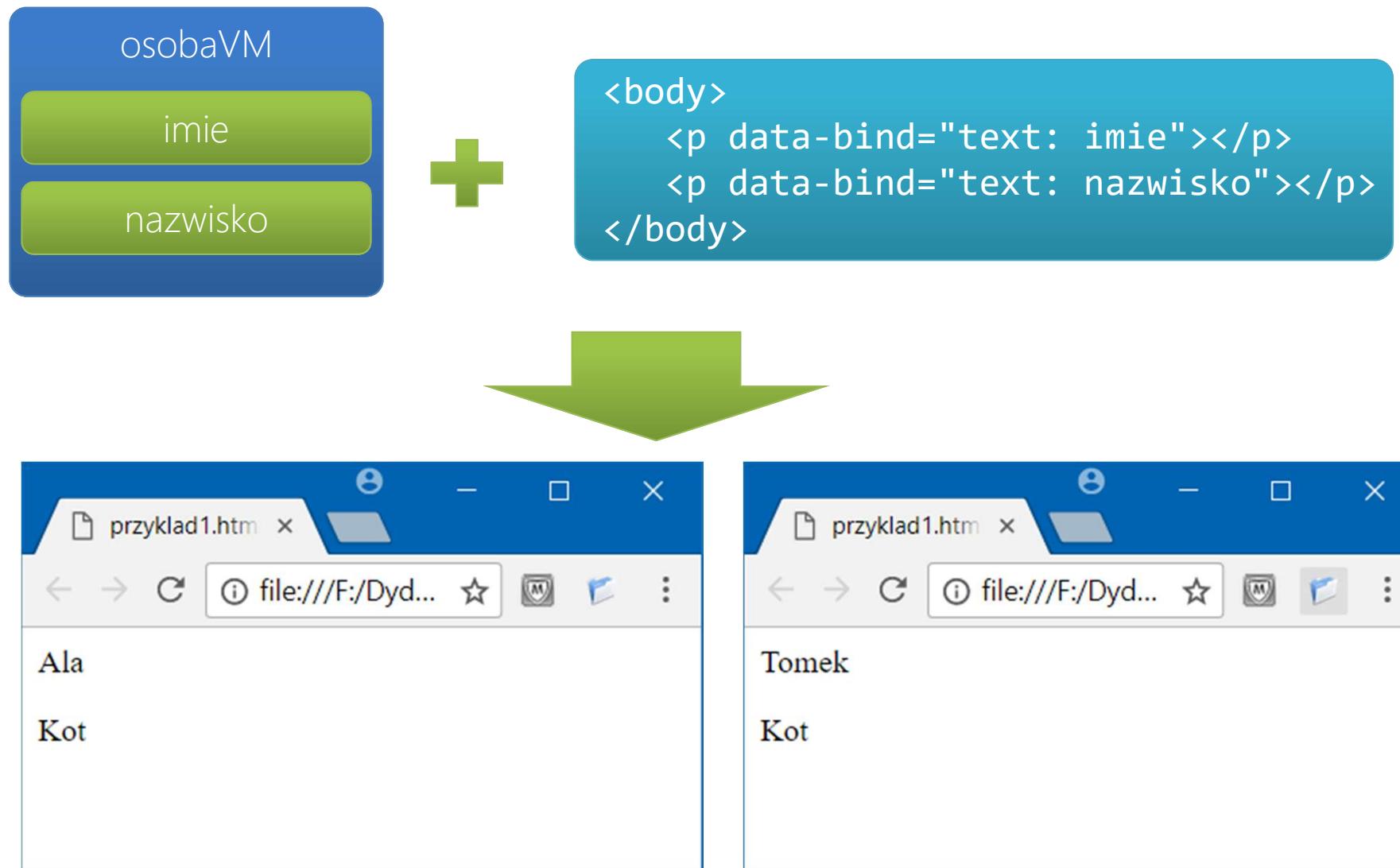
$(function () {
  ko.applyBindings(osobaVM);
});
```

Najprostszy model widoku

```
<html>
<head>
    <script s= {
        "Ala",
        "Kot"
    }
    <bindings(osobaVM);
    mie = "Tomasz";
</body>
</html>
```



Najprostszy model widoku



Najprostszy obserwowalny model widoku

```
<html>
<head>
  <script src="jquery.js"></script>
  <script src="knockout.js"></script>
  <script src="app2.js"></script>
</head>
<body>
  <p data-bind="text: imie"></p>
  <p data-bind="text: nazwisko"></p>
</body>
</html>
```

```
var osobaVM = {
  imie : ko.observable("Ala"),
  nazwisko: ko.observable("Kot")
};

$(function () {
  ko.applyBindings(osobaVM);
  osobaVM.imie("Tomasz");
});
```

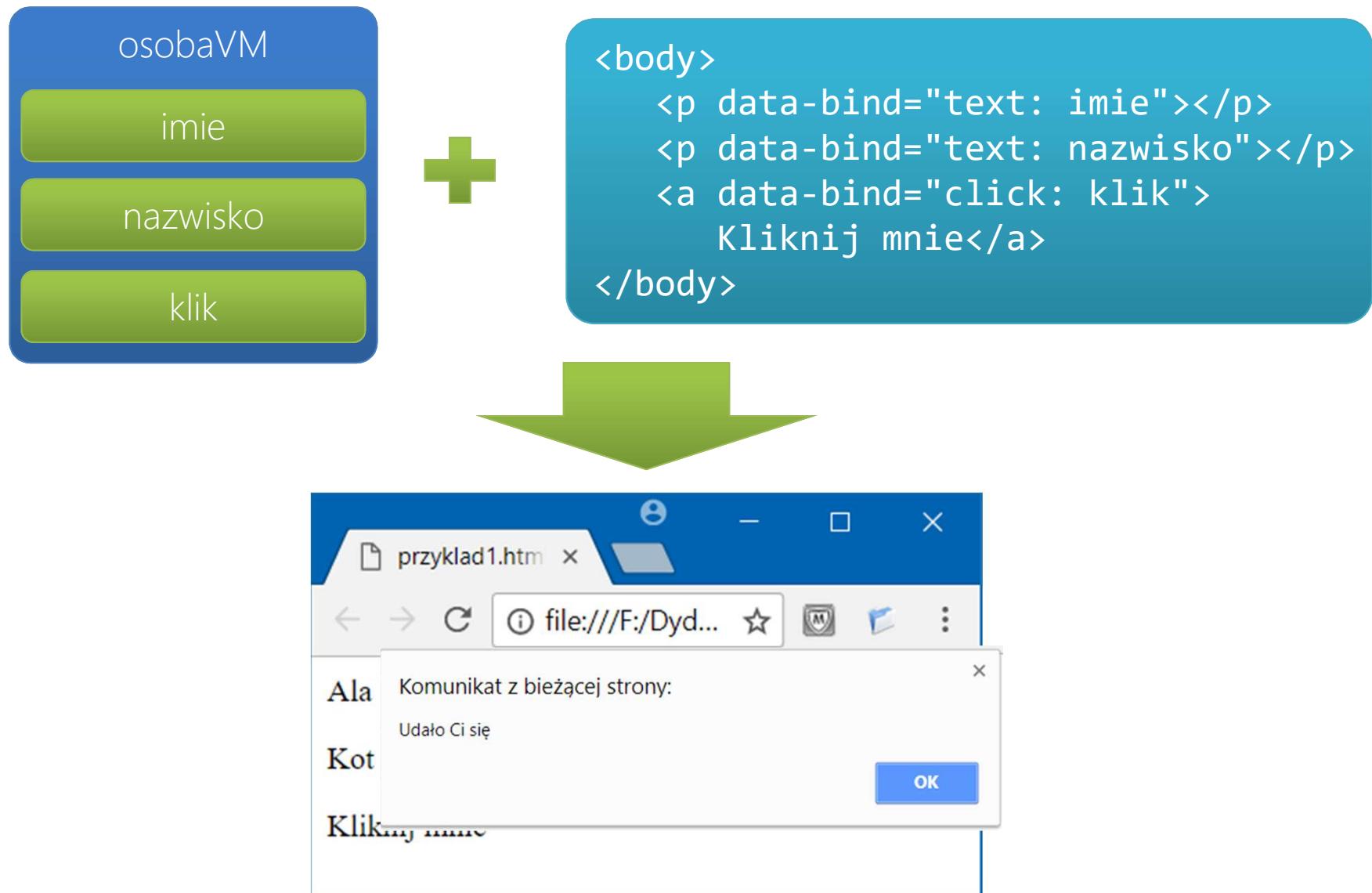
Najprostszy obserwowalny model widoku

```
...
<body>
  <p data-bind="text: imie">
    </p>
  <p data-bind="text: nazwisko">
    </p>
  <a data-bind="click: klik">
    Kliknij mnie</a>
</body>
...
```

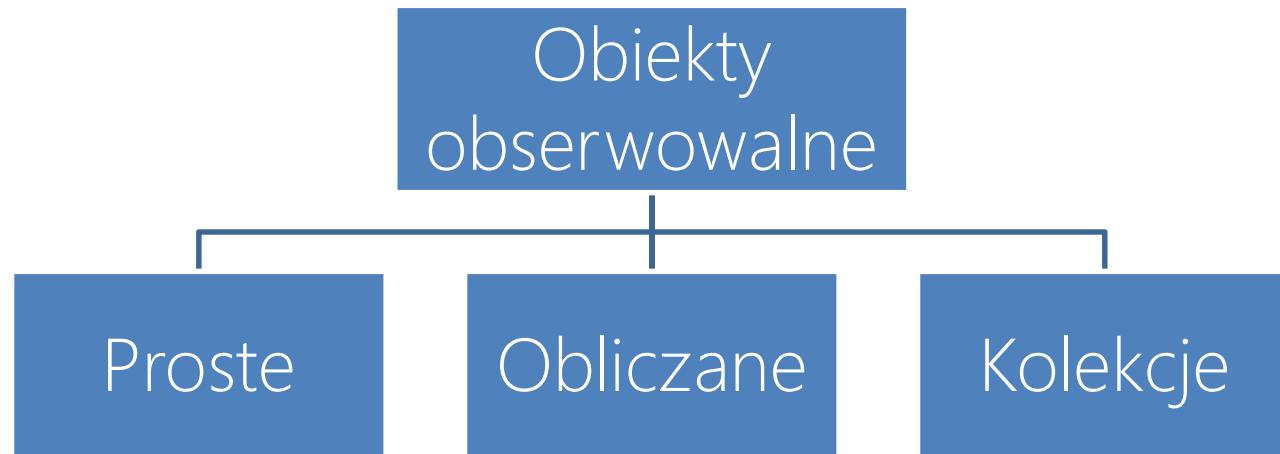
```
var osobaVM = {
  imie : ko.observable("Ala"),
  nazwisko: ko.observable("Kot"),
  klik: function() {
    alert("Udało Ci się");
  }
};

$(function () {
  ko.applyBindings(osobaVM);
});
```

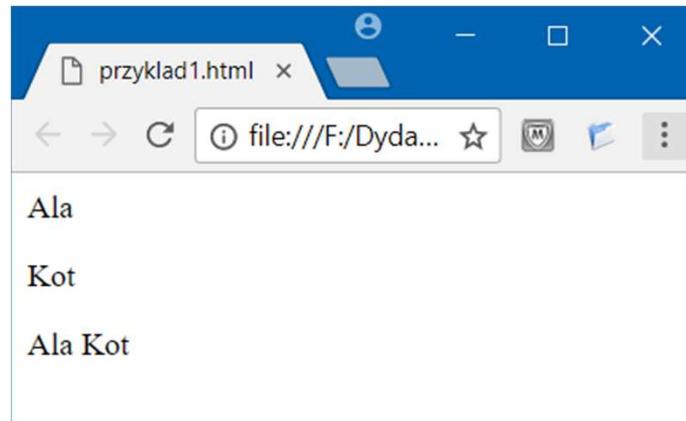
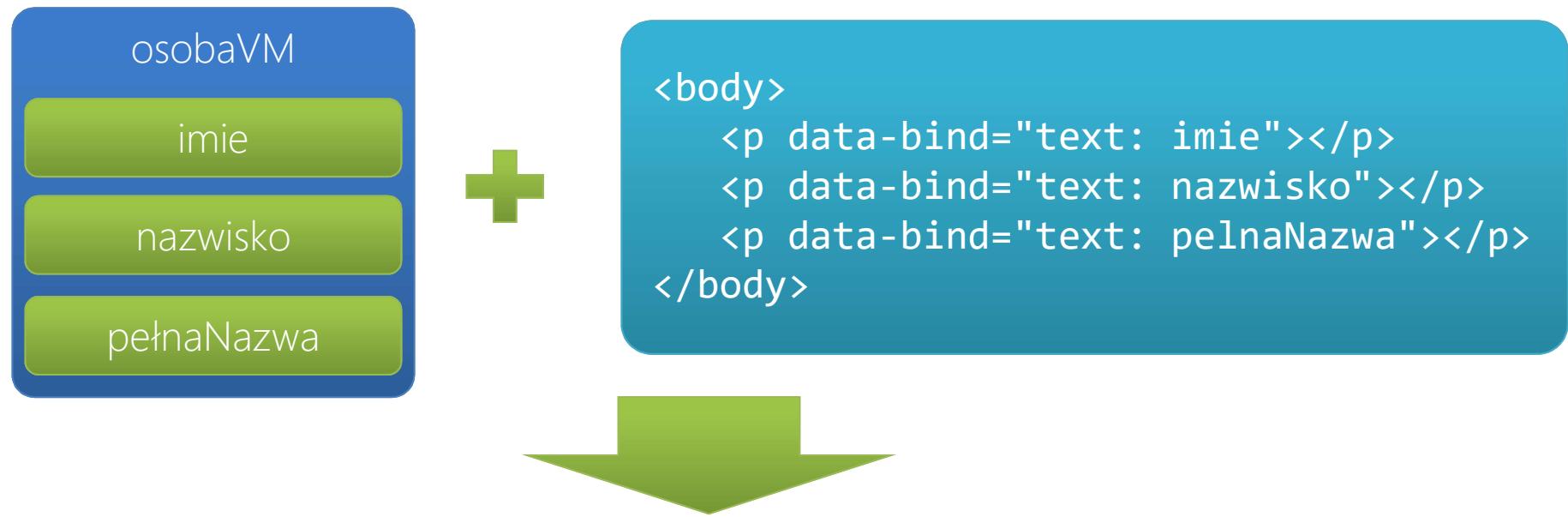
Najprostszy obserwowalny model widoku



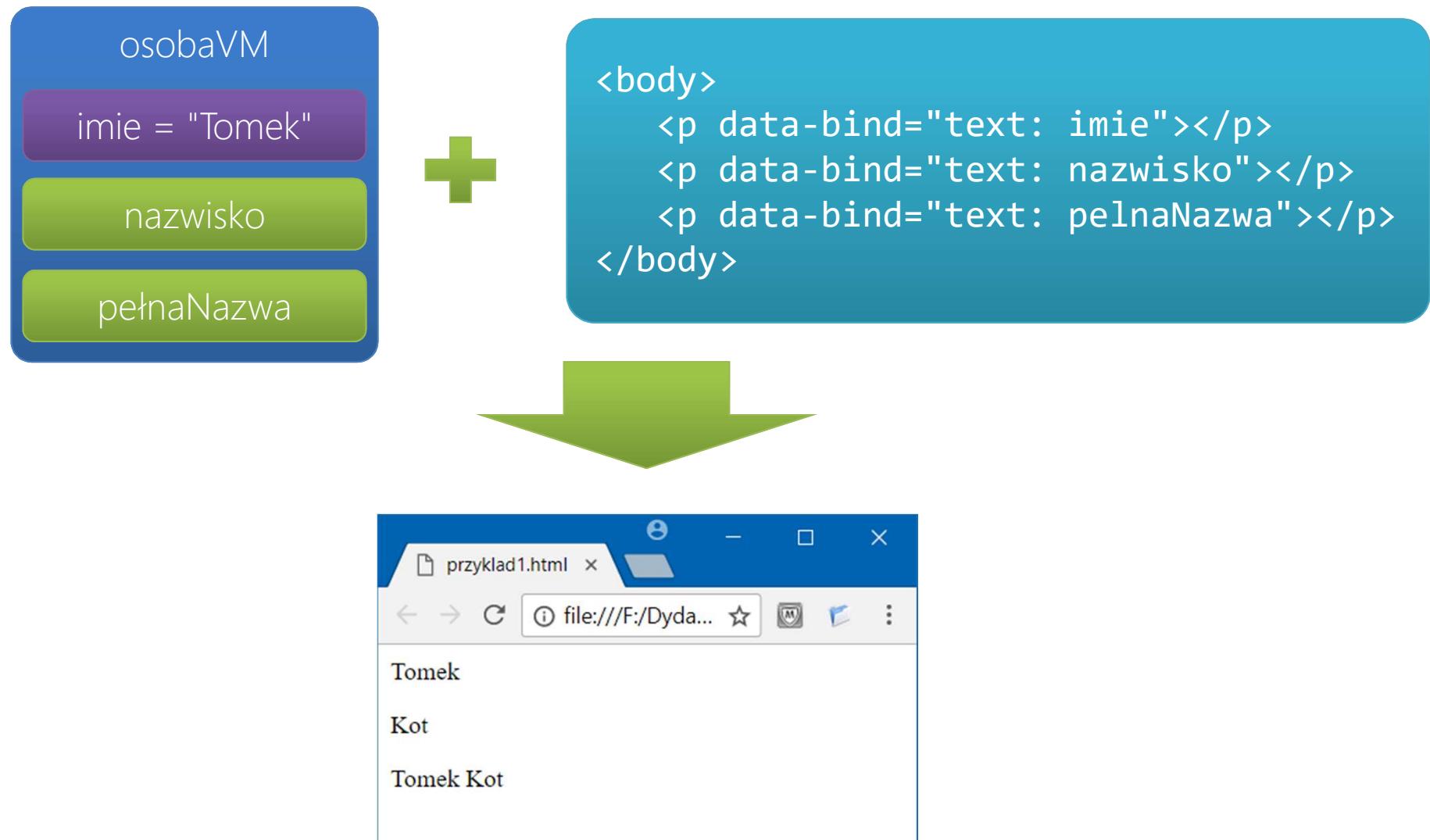
Rodzaje obiektów observables



Rodzaje obiektów observables



Rodzaje obiektów observables



Rodzaje obiektów observables

Obliczane obiekty observables

```
<html>
  <head>
    <script src="jquery.js"></script>
    <script src="knockout.js"></script>
    <script src="app4.js"></script>
  </head>
  <body>
    <p data-bind="text: imie"></p>
    <p data-bind="text: nazwisko"></p>
    <p data-bind="text: pelnaNazwa"></p>
  </body>
</html>
```

Rodzaje obiektów observables

Obliczane obiekty observables

```
function osobaVM()
{
    var that = this;
    this.imie = ko.observable("Ala"),
    this.nazwisko = ko.observable("Kot"),
    this.pelnaNazwa = ko.computed(
        function() {
            return that.imie()+" "+that.nazwisko();
        }
    );
}

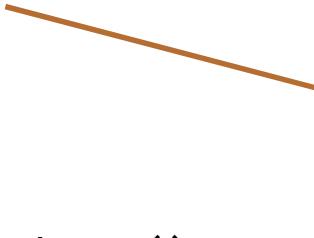
$(function () {
    var obj = new osobaVM();
    ko.applyBindings(obj);
    obj.imie("Tomek");
});
```

Rodzaje obiektów observables

Obliczane obiekty observables

```
function osobaVM()
{
    var that = this;
    this.imie = ko.observable("Ala"),
    this.nazwisko = ko.observable("Kot"),
    this.pelnaNazwa = ko.computed(
        function() {
            return that.imie()+" "+that.nazwisko();
        }
    );
}

$(function () {
    var obj = new osobaVM();
    ko.applyBindings(obj);
    obj.imie("Tomek");
});
```



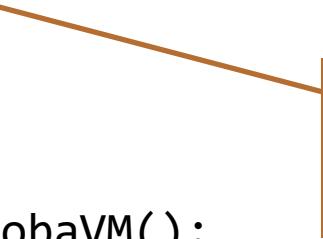
Problem z this

Rodzaje obiektów observables

Obliczane obiekty observables

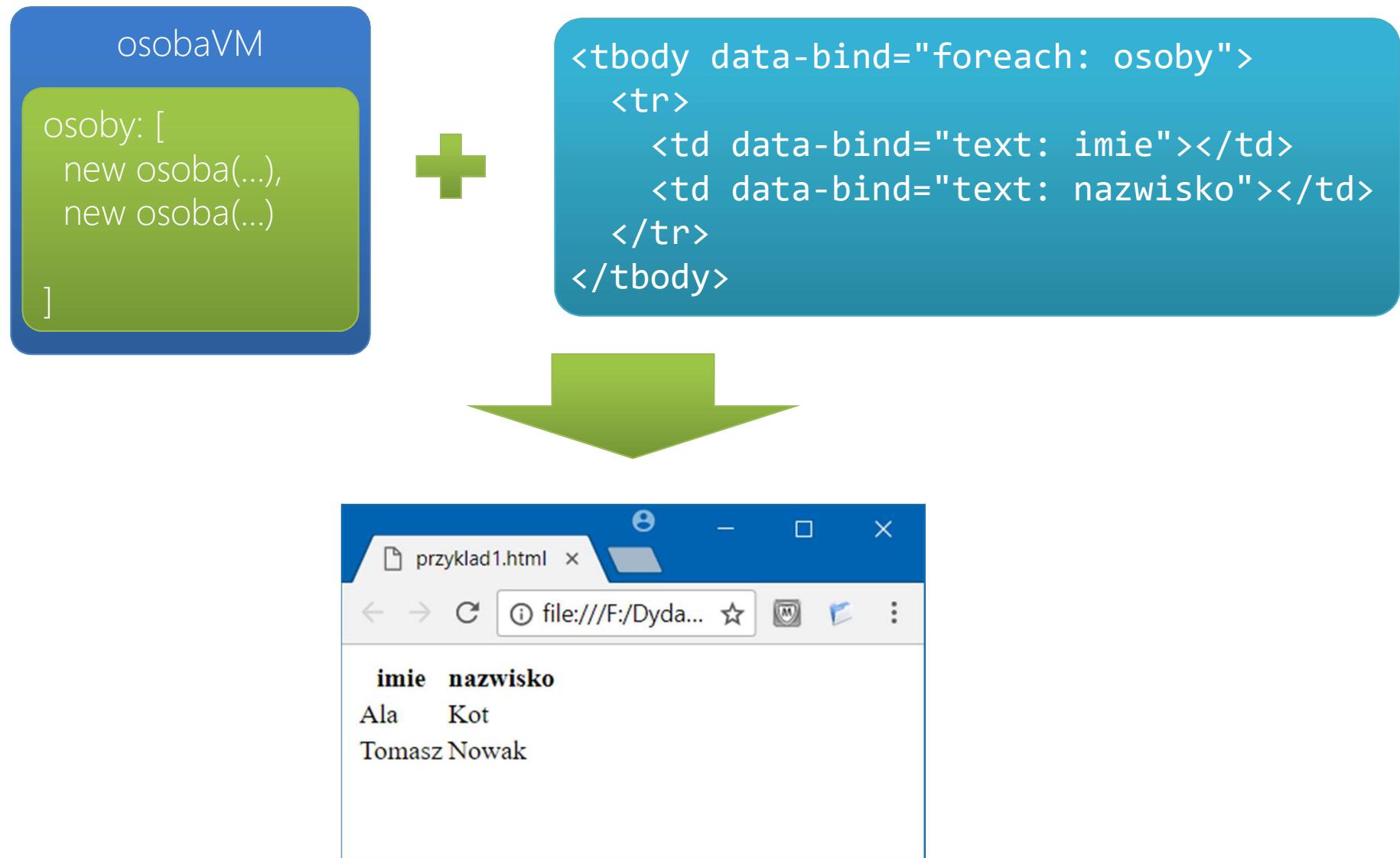
```
function osobaVM()
{
    this.imie = ko.observable("Ala"),
    this.nazwisko = ko.observable("Kot"),
    this.pelnaNazwa = ko.computed(
        () => this.imie()+" "+this.nazwisko()
    );
};

$(function () {
    var obj = new osobaVM();
    ko.applyBindings(obj);
    obj.imie("Tomek");
});
```



Funkcja lambda

Rodzaje obiektów observables



Rodzaje obiektów observables

Obserwowalne kolekcje

```
<body>
  <table>
    <thead><tr><th>imie</th><th>nazwisko</th></tr></thead>
    <tbody data-bind="foreach: osoby">
      <tr>
        <td data-bind="text: imie"></td>
        <td data-bind="text: nazwisko"></td>
      </tr>
    </tbody>
  </table>
</body>
```

Rodzaje obiektów observables

Obserwowlne kolekcje

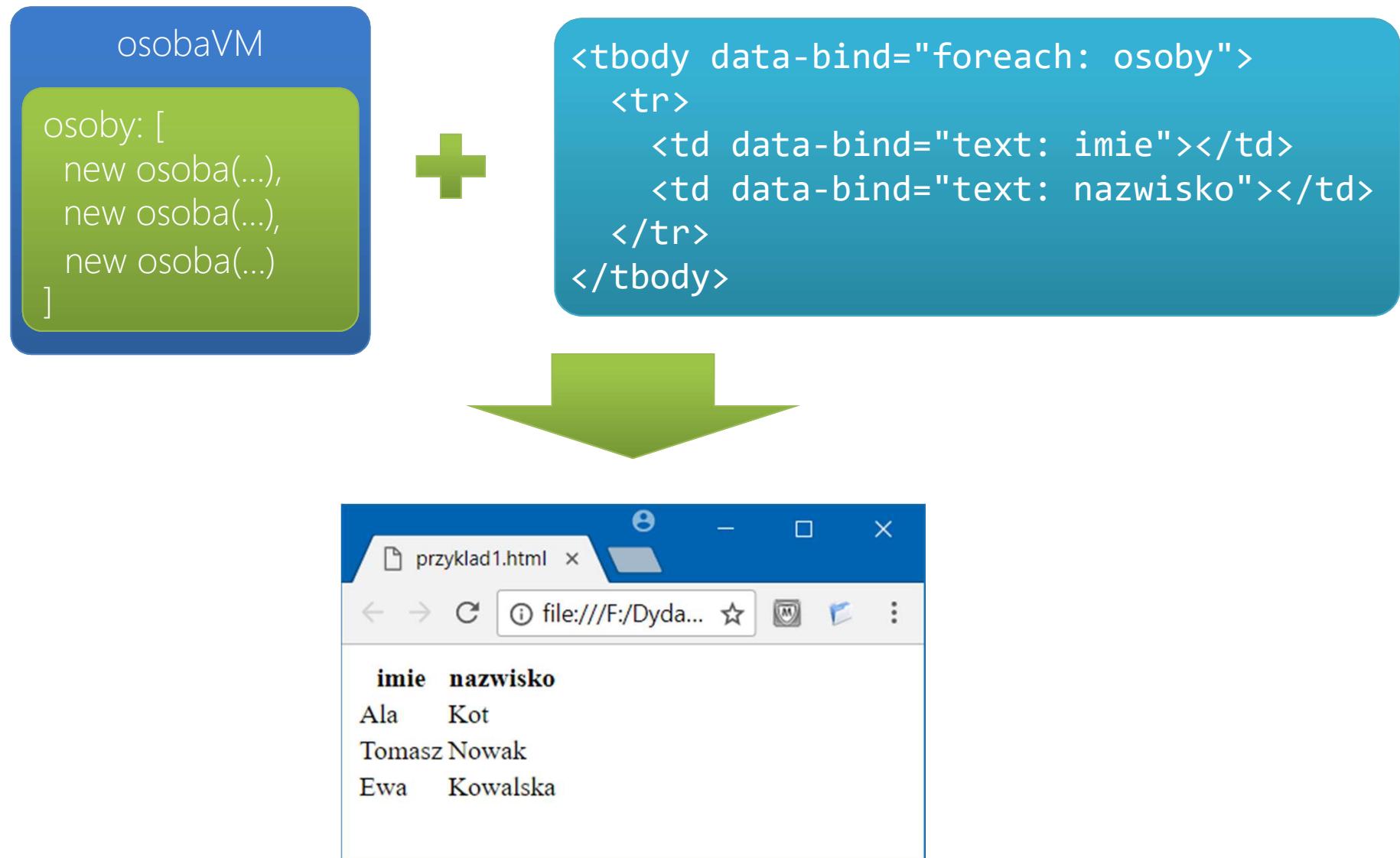
```
function osoba(imie, nazwisko) {  
    this.imie = imie;  
    this.nazwisko = nazwisko;  
};  
  
function osobyVM() {  
    this.osoby = ko.observableArray([  
        new osoba("Ala", "Kot"),  
        new osoba("Tomasz", "Nowak")  
    ]);  
}  
  
$(function () {  
    ko.applyBindings(osobyVM);  
});
```

Rodzaje obiektów observables

Obserwowalne kolekcje – dodawanie elementu

```
<body>
  <a data-bind="click:dodaj">Dodaj nową osobę</a>
  <table>
    <thead><tr><th>imie</th><th>nazwisko</th></tr></thead>
    <tbody data-bind="foreach: osoby">
      <tr>
        <td data-bind="text: imie"></td>
        <td data-bind="text: nazwisko"></td>
        <td><a data-bind="click: $root.usun">Usuń</a></td>
      </tr>
    </tbody>
  </table>
</body>
```

Rodzaje obiektów observables



Rodzaje obiektów observables

Obserwowalne kolekcje – dodawanie / usuwanie elementu

```
function osobyVM()
{
    this.osoby = ko.observableArray([
        new osoba("Ala", "Kot"),
        new osoba("Tomasz", "Nowak")
    ]);
    this.dodaj = () => {
        this.osoby.push(new osoba("Kolejny", "Nowak"));
    };
    this.usun = (osoba) => {
        this.osoby.remove(osoba);
    }
}
```

Kontekst wiązania

Określa dane, które aktualnie podlegają wiązaniu:

- **\$root** – zawsze odnosi się do obiektu najwyższego kontekstu
- **\$data** – zawsze odnosi się do obiektu kontekstu aktualnego
- **\$index** – zawiera indeks aktualnego elementu kolekcji
- **\$parent** – odnosi się do obiektu kontekstu o jeden poziom wyżej od obecnego

Rodzaje obiektów observables

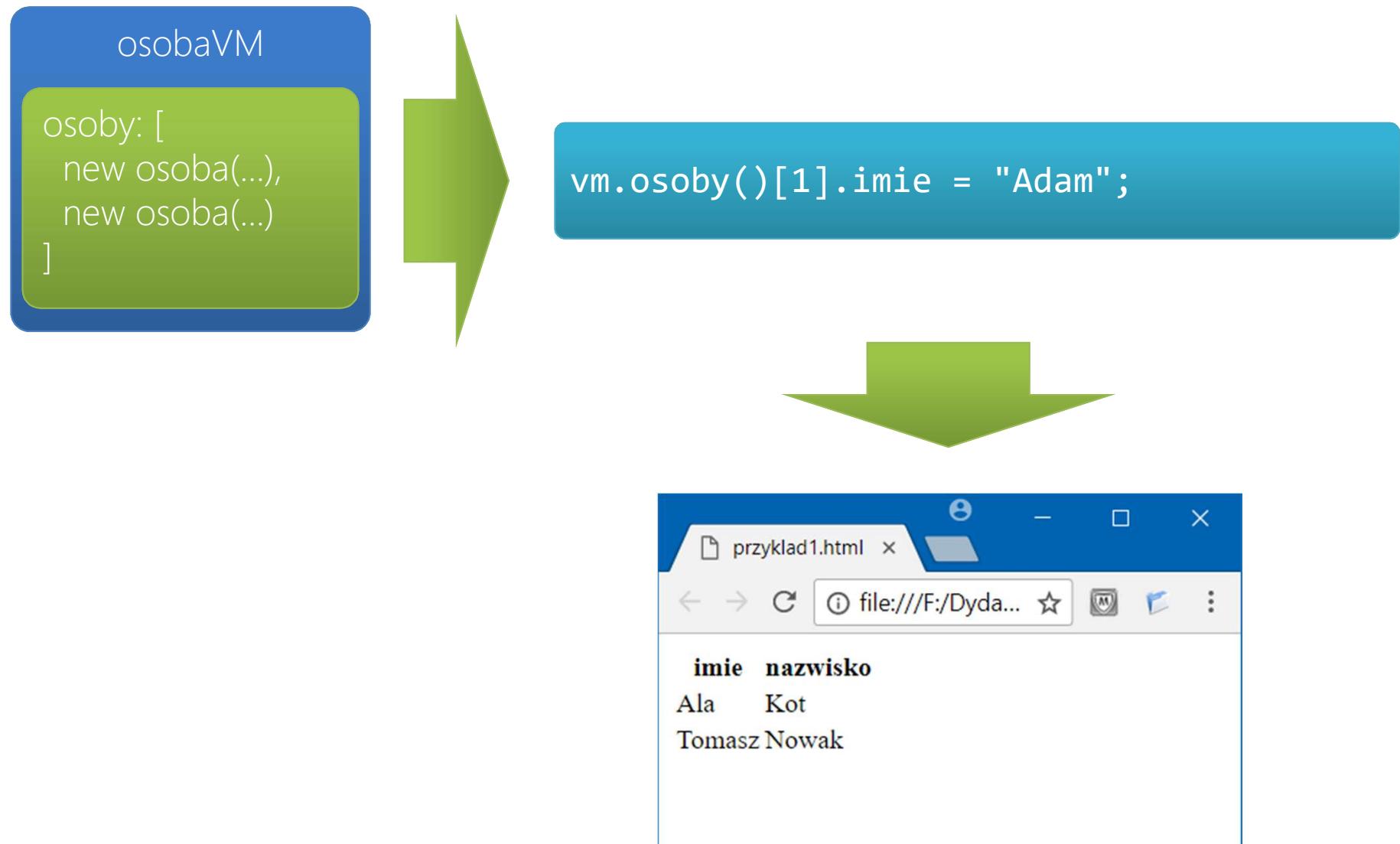
Obserwowańe kolekcje

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

function osobyVM() {
    this.osoby = ko.observableArray([
        new osoba("Ala", "Kot"),
        new osoba("Tomasz", "Nowak")
    ]);
}

$(function () {
    ko.applyBindings(osobyVM);
});
```

Rodzaje obiektów observables



Rodzaje obiektów observables

Obserwowlne kolekcje

```
function osoba(imie, nazwisko) {
    this.imie = ko.observable(imie);
    this.nazwisko = ko.observable(nazwisko);
};

function osobyVM() {
    this.osoby = ko.observableArray([
        new osoba("Ala", "Kot"),
        new osoba("Tomasz", "Nowak")
    ]);
}

$(function () {
    ko.applyBindings(osobyVM);
});
```

Wiązania warunkowe

```
<body>
  <a data-bind="click:dodaj">Dodaj nową osobę</a>
  <table>
    <thead><tr><th>imie</th><th>nazwisko</th></tr></thead>
    <tbody data-bind="foreach: osoby">
      <tr data-bind="if: zaliczyl()">
        <td data-bind="text: imie"></td>
        <td data-bind="text: nazwisko"></td>
        <td><a data-bind="click: $root.usun">Usuń</a></td>
      </tr>
    </tbody>
  </table>
</body>
```

Wiązania warunkowe

```
function osoba(imie, nazwisko, zaliczyl) {  
    this.imie = ko.observable(imie),  
    this.nazwisko = ko.observable(nazwisko),  
    this.zaliczyl = ko.observable(zaliczyl)  
};  
  
function osobyVM() {  
    this.osoby = ko.observableArray([  
        new osoba("Ala", "Kot", true),  
        new osoba("Tomasz", "Nowak", false)  
    ]);  
}  
  
$(function () {  
    ko.applyBindings(new osobyVM());  
});
```

Wiązania połączone z wyglądem

Knockout udostępnia kilka typów wiązań odnoszących się do wyglądu widoku:

- **text: <wartość>** - ustawia zawartość elementu
- **html: <wartość>** - ustawia zawartość HTML elementu
- **visible: <warunek>** - ustawia widoczność elementu
- **css: <obiekt>** - dodaje klasę css do elementu
- **style: <obiekt>** - definiuje atrybut style elementu
- **attr: <obiekt>** - dodaje dowolny atrybut do elementu

Wiązania interaktywne

Knockout udostępnia kilka typów wiązań umożliwiających interakcję z użytkownikiem

- **click: <metoda>** - wywołuje metodę obiektu ViewModel po kliknięciu
- **value: <właściwość>** - łączy element formularza do właściwości ViewModel
- **event: <obiekt>** - wywołuje metodę, gdy wystąpi zdarzenie zainicjowane przez użytkownika
- **submit: <metoda>** - wywołuje metodę, gdy formularz jest zatwierdzany
- **enable: <właściwość>** - aktywuje element formularza na podstawie podanego warunku

Wiązania interaktywne

Knockout udostępnia kilka typów wiązań umożliwiających interakcję z użytkownikiem

- **disable:** <właściwość> - deaktywuje element formularza na podstawie podanego warunku
- **checked:** <właściwość> - łączy element typu radiobutton lub checkbox z właściwością obiektu ViewModel
- **options <tablica>** - łączy elementy <select> z właściwością obiektu ViewModel
- **selectedOptions <tablica>** - definiuje zaznaczone elementy na liście
- **hasFocus <właściwość>** - definiuje czy, element posiada fokus

AJAX

Czym jest AJAX?

AJAX = Asynchronous JavaScript and XML

AJAX pozwala na tworzenie aplikacji internetowych w taki sposób aby odczucia użytkownika były jak najbardziej zbliżone do aplikacji desktopowych.

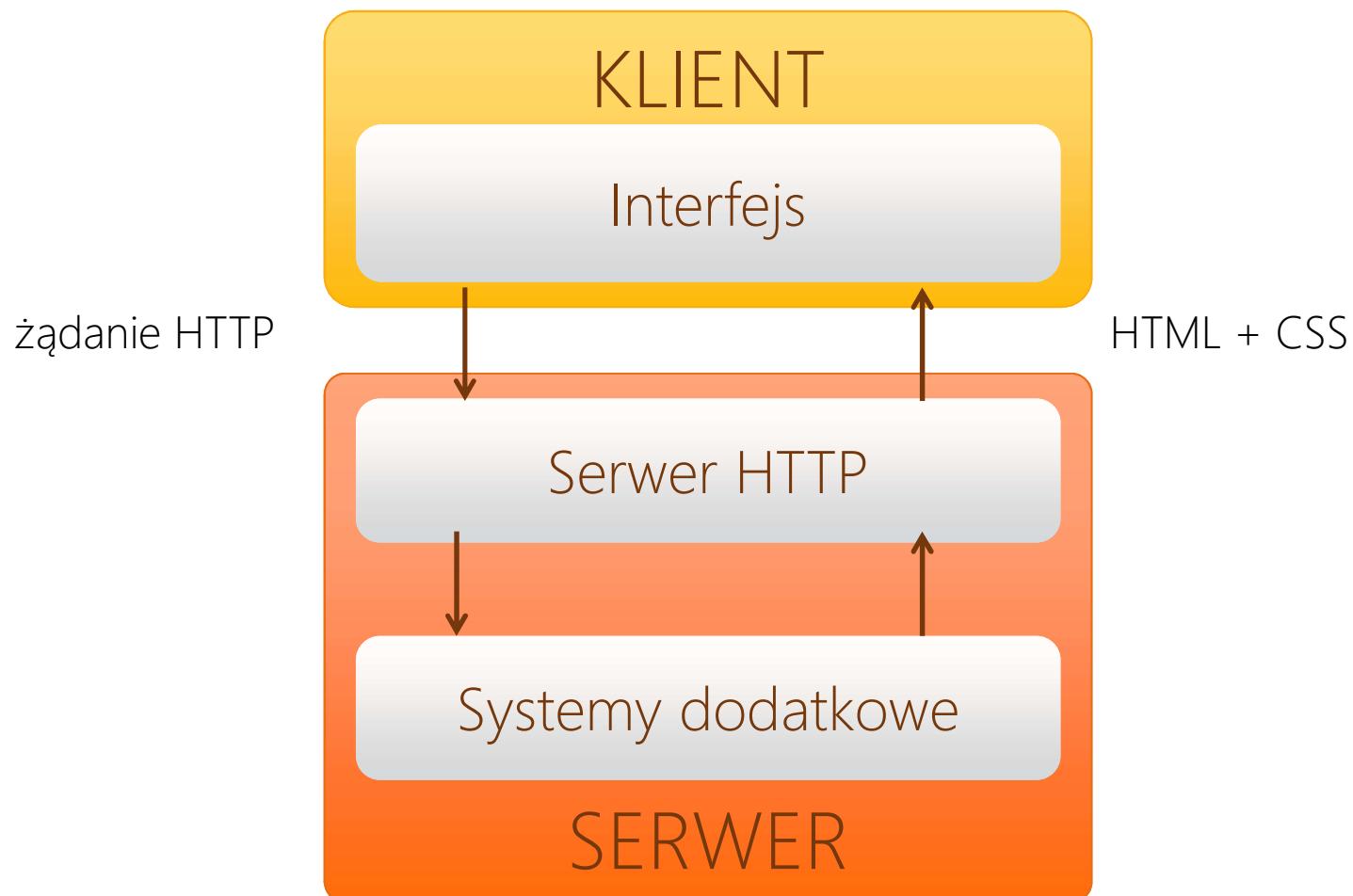
Czym jest AJAX? (1)

W skład AJAX wchodzą:

- Standardowa reprezentacja internetowa HTML + CSS
- Document Object Model
- XML / JSON
- JavaScript
- XMLHttpRequest

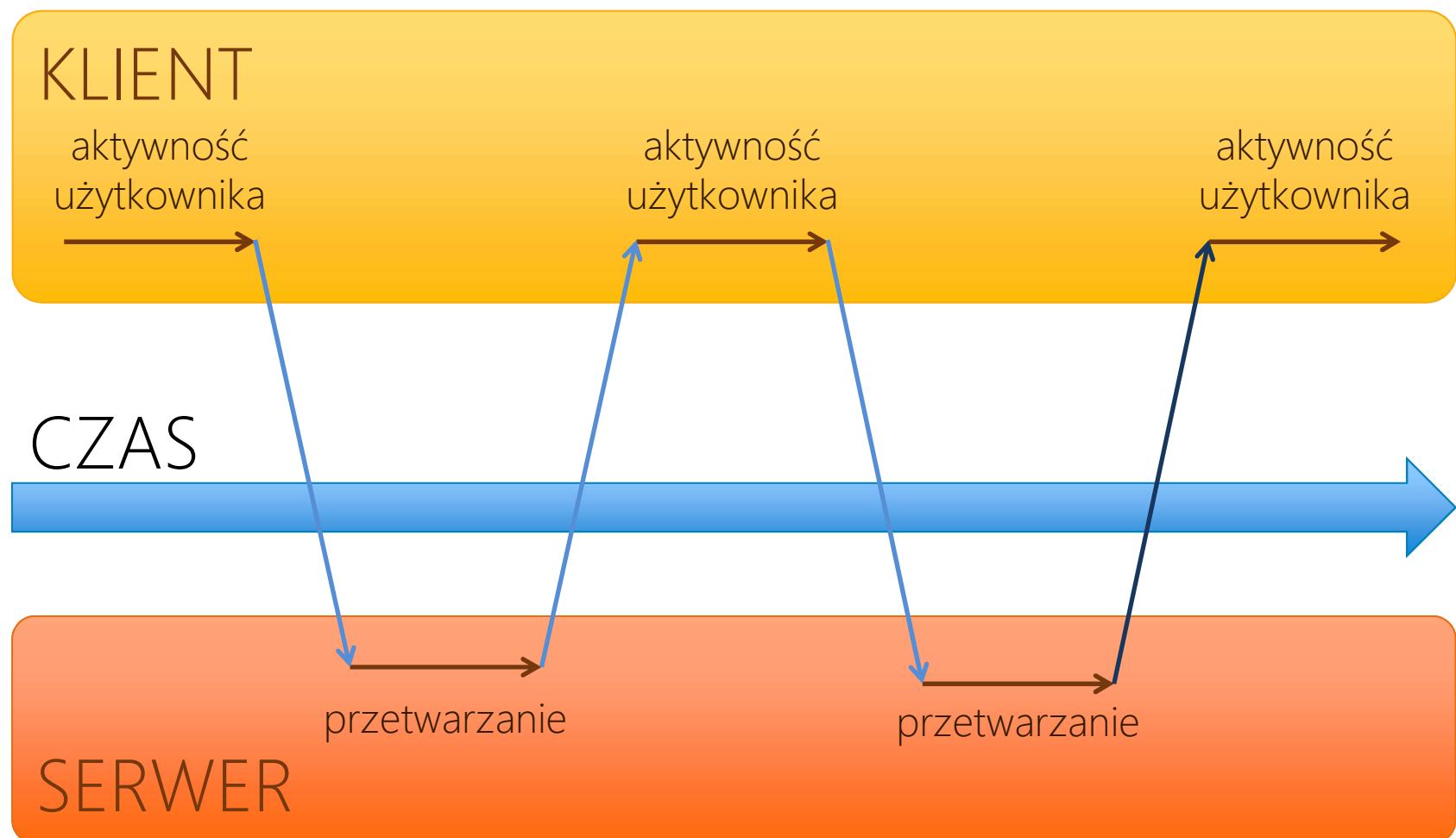
Czym jest AJAX? (2)

Klasyczny model aplikacji WWW



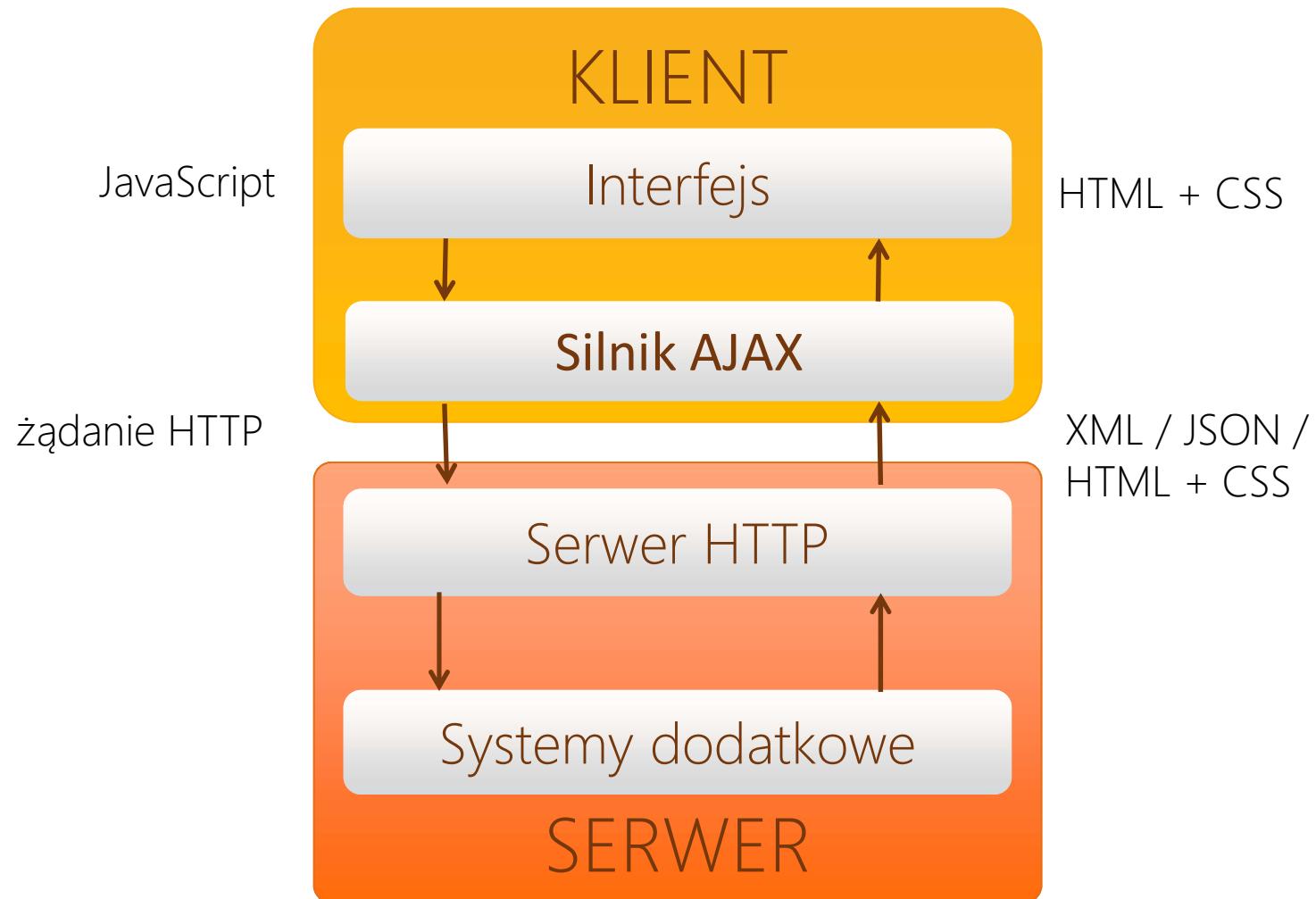
Czym jest AJAX? (3)

Interakcja synchroniczna



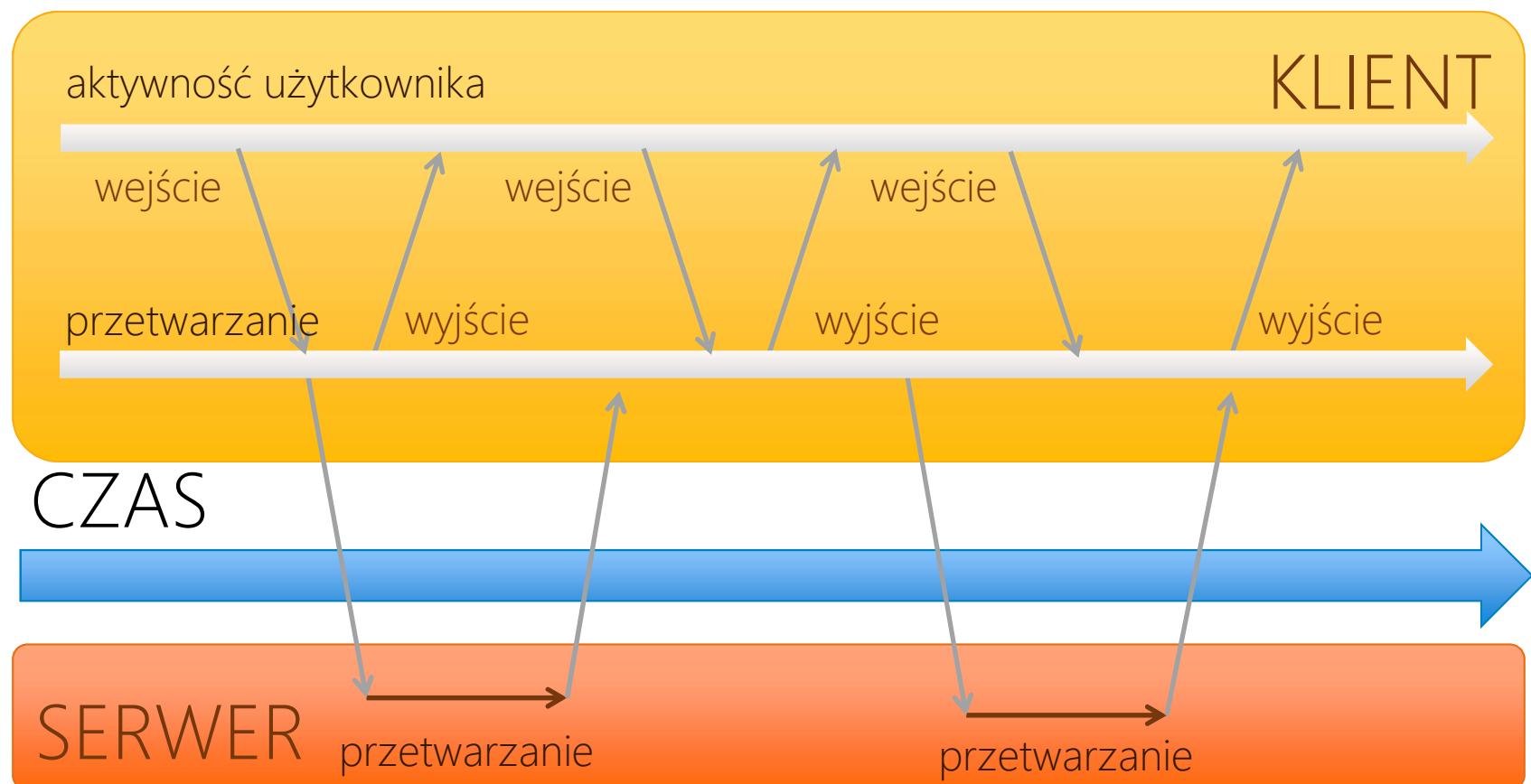
Czym jest AJAX? (4)

Model aplikacji AJAX



Czym jest AJAX? (5)

Interakcja asynchroniczna



XMLHttpRequest (1)

- Obiekt XMLHttpRequest jest rdzennym elementem AJAX-u odpowiedzialnym za obsługę komunikacji klienta z serwerem.
- Pozwala na wysyłanie żądania HTTP lub HTTPS bezpośrednio do serwera HTTP, z poziomu języka skryptowego, oraz odbierania odpowiedzi również z poziomu języka skryptowego.
- XMLHttpRequest jest dostępny jako obiekt JavaScript.

XMLHttpRequest (2)

- Właściwości i zdarzenia

Nazwa	Opis
status	Kod odpowiedzi serwera
statusText	Opis statusu
readyState	Wartość liczbową wskazującą obecny stan obiektu
responseText	Odpowiedź serwera w postaci łańcucha tekstowego
responseXML	Odpowiedź serwera w postaci łańcucha XML
upload	Obiekt reprezentujący proces ładowania danych
timeout	Czas w milisekundach do automatycznego przerwania

Nazwa	Opis
onreadystatechange	Funkcja obsługi zdarzenia wywoływana, gdy zmieni się stan obiektu

XMLHttpRequest (3)

- Metody

Nazwa	Opis
abort()	Anuluje obecne żądanie HTTP
getAllResponseHeaders()	pobiera wartości nagłówków HTTP z odpowiedzi serwera
getResponseHeader("nazwa")	zwraca wartość nagłówka HTTP o podanej nazwie
open("metoda", "URL", asyncFlag, "login", "hasło")	Inicjalizuje żądanie
send(treść)	wysyła żądanie HTTP do serwera i odbiera odpowiedź
setRequestHeader("nazwa", "wartość")	Ustawia wartość nagłówka żądania HTTP

Metody protokołu HTTP

Metoda	Opis
GET	pobranie zasobu wskazanego przez URI , może mieć postać warunkową jeśli w nagłówku występują pola warunkowe takie jak "If-Modified-Since"
POST	przyjęcie danych przesyłanych od klienta do serwera (np. wysyłanie zawartości formularzy)
PUT	przyjęcie danych przesyłanych od klienta do serwera, najczęściej aby zaktualizować wartość encji
POST	przyjęcie danych przesyłanych od klienta do serwera (np. wysyłanie zawartości formularzy)
DELETE	żądanie usunięcia zasobu, włączone dla uprawnionych użytkowników

Metody protokołu HTTP

Metoda	Opis
HEAD	pobiera informacje o zasobie, stosowane do sprawdzania dostępności zasobu
OPTIONS	informacje o opcjach i wymaganiach istniejących w kanale komunikacyjnym
TRACE	diagnostyka, analiza kanału komunikacyjnego
CONNECT	żądanie przeznaczone dla serwerów pośredniczących pełniących funkcje tunelowania
PATCH	aktualizacja części danych

XMLHttpRequest (4)

- Właściwość readyState

Kod	Nazwa	Opis
0	Niezainicjalizowany	obiekt nie jest zainicjalizowany (open nie została wywołana)
1	Otwarty	obiekt otwarty (open została wywołana)
2	Załadowane nagłówki	otrzymano nagłówki i status (send wywołana)
3	Ładowany	dane są pobierane (możliwa jest interakcja z obiektem, choć może on nie być w pełni zainicjalizowany)
4	Gotowy	obiekt jest w pełni zainicjalizowany

XMLHttpRequest – transmisja synchroniczna

```
var xmlhttp = new XMLHttpRequest();  
  
if(xmlhttp)  
{  
    xmlhttp.open('GET', 'adresStrony', false);  
    xmlhttp.send(null);  
    document.getElementById('wyniki').innerHTML =  
        xmlhttp.responseText;  
}
```

XMLHttpRequest – transmisja asynchroniczna

```
var xmlhttp = StworzXMLHttpRequest();

if(xmlhttp) {
    xmlhttp.open('GET', 'adresStrony', true);
    xmlhttp.onreadystatechange = zmianaStanu;
    xmlhttp.send(null);
}

function zmianaStanu() {
    if(xmlhttp.readyState == 4)
        if(xmlhttp.status == 200)
            document.getElementById('wyniki').innerHTML =
                xmlhttp.responseText;
}
```

XMLHttpRequest – transmisja asynchroniczna

```
var xmlhttp = StworzXMLHttpRequest();  
  
if(xmlhttp) {  
    xmlhttp.open('POST', 'adresStrony', true);  
    xmlhttp.onreadystatechange = zmianaStanu;  
    xmlhttp.send("pierwsze=ala&drugie=ma&trzecie=kota");  
}  
  
function zmianaStanu() {  
    if(xmlhttp.readyState == 4)  
        if(xmlhttp.status == 200)  
            document.getElementById('wyniki').innerHTML =  
                xmlhttp.responseText;  
}
```

AJAX, a przesyłanie plików

- Przesyłanie plików przy pomocy przeglądarki jest proste

```
<form action="skrypt" method="post"  
      enctype="multipart/form-data" id="formularz">  
  
    <div>Imię: <input type="text" name="imie" /></div>  
    <div>Nazwisko: <input type="text" name="nazwisko" /></div>  
    <div>Zdjecie: <input type="file" name="zdjecie" /></div>  
    <div><button type="submit">Zatwierdź</button></div>  
  
</form>
```

- Przesyłanie plików przy pomocy AJAX do HTML5 było dość skomplikowane

AJAX, a przesyłanie plików

- Obiekt FormData

```
var $ = function(id) { return document.getElementById(id); }

window.addEventListener("load", function() {
    var form = $("#formularz");
    form.addEventListener("submit", function(e) {
        e.preventDefault();
        var formData = new FormData(form);
        var ajax = new XMLHttpRequest();
        ajax.open("post", "adres", true);
        ...
        ajax.send(formData);
    });
});
```

Ajax i jQuery

- Biblioteka jQuery umożliwia również komunikację z serwerem za pomocą AJAX

```
$ .load(url [,dane] [,ukonczono] )  
$ .ajax(opcje)  
$ .get(url [,dane] [,ukonczono] [,typWyniku])  
$ .getJSON(url, [,dane] [,ukonczono])  
$ .post(url [,dane] [,ukonczono] [,typWyniku])
```

- Podstawowe parametry funkcji \$.ajax to:
 - **type** - "GET" | "POST"
 - **url**
 - **dataType** - "xml" | "json" | "script" | "html"
 - **success**