

ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej

Aplikacje webowe na platformę .NET

W02 – HTML5 - elementy bazowe

Syllabus

- HTML5
 - Znaki specjalne
 - Przestarzałe elementy
 - Formatowanie tekstu
 - Listy:
 - Nieuporządkowane
 - Uporządkowane
 - Definicji
 - Tabele
 - Formularze i ich pola, typy pól, część atrybutów
 - Linkowanie wewnętrzne
 - Elementy `<meta>`
 - Nowe typy pól formularza
 - Nowe elementy struktury dokumentu
 - Co jeszcze jest w HTML5



Znaki specjalne

- pewne znaki w ramach HTML5 ma specjalne znaczenie np.: `<`, ``?`, ``&`, cudzysłów itp.
- Zatem nie mogą być wprost składową danych (atrybuty, query string itp.)
- Sposobem reprezentacji takich danych są znaki specjalne (w formacie `&code;`) gdzie `code` zapisuje się jako:
 - skrót słowa
 - liczba
 - dziesiętna
 - szesnastkowa

Symbol	Opis (ang.)	Zapis
&	ampersand	<code>&amp;</code>
'	apostrofe	<code>&apos;</code>
>	greater-than	<code>&gt;</code>
<	less-than	<code>&lt;</code>
"	quote	<code>&quot;</code>

Znaki specjalne

- Również inne znaki nieobecne na klawiaturze nie powinno się w sposób sztuczny generować (ogólna zasada jest taka, że w HTML mogą być tylko symbole osiągalne z klawiatury)
- Dla nich również używany jest format `&code;`
- Tabele kodów:
 - <https://dev.w3.org/html5/html-author/charref>
 - https://www.w3schools.com/html/html_symbols.asp
 - <https://www.toptal.com/designers/htmlarrows/>
 - <https://ascii.cl/htmlcodes.htm>

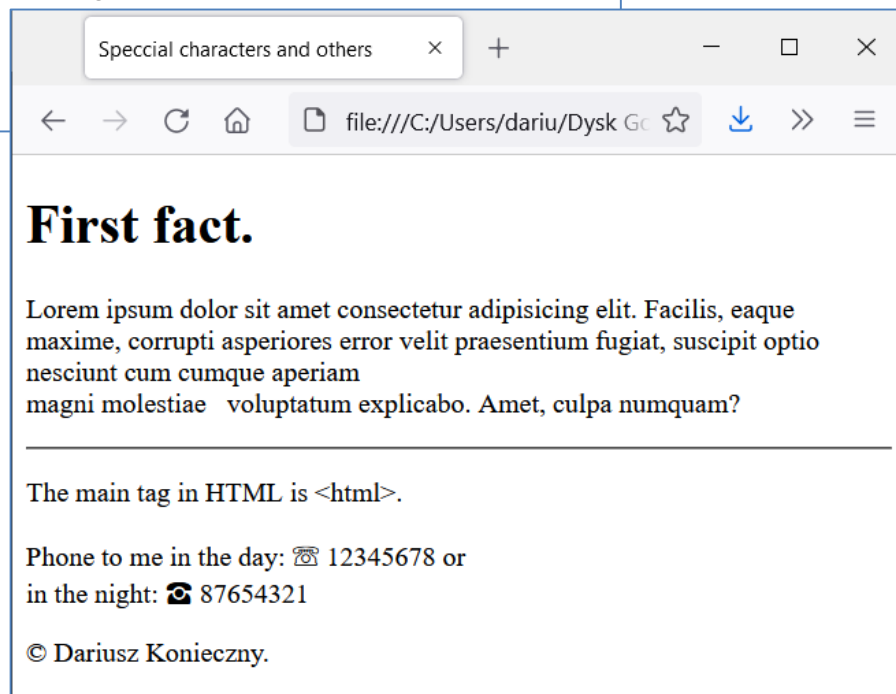
Symbol	Opis (ang.)	Zapis
©	copyright	<code>&copy;</code>
<	less-than	<code>&lt;</code>
	phone	<code>&phone;</code>
	phone	<code>&#9743;</code>
®	Registered trademark	<code>&reg;</code>
	Non-braking space	<code>&nbsp;</code>
™	trademark	<code>&trade;</code>

Pozioma linia, przełamanie tekstu

- Znaczniki HTML nie są wrażliwe na wielkość liter
 - `<html>`, `<HTML>`, `<Html>`, `<html>` - wszystkie to znacznik html
 - Raczej używa się małych liter
 - Jako rozdzielnik słów w znaczniku można używać minusa `-`
 - Np. `popup-info`
 - W standardowych znacznikach HTML5 nie ma znaczników z minusem
- Standardowo przeglądarka zamienia ciąg białych znaków (spacja, `<enter>`, `<tab>`) na jeden biały znak i w czasie renderingu w zależności od możliwości zmieszczenia wyrazu w linii zamienia ten biały znak na jedną spację lub przejście do nowej linii.
- Znak specjalny ` ` umieszcza „twardą” spację, która nie jest traktowana jako biały znak tylko jak kolejny znak wyrazu:
 - Nie nastąpi przełamanie linii w tym miejscu
 - Można zrobić większy wizualnie odstęp między wyrazami za pomocą kilku kodów ` `.
- Znacznik pusty `<hr>` oznacza poziomą linię (ang **horizontal line**) pod tekstem.
 - Jest on przestarzały, zamiast niego powinno się używać CSS
- Znacznik pusty `
` oznacza przełamanie tekstu w tym miejscu (standardowo przeglądarka decyduje, gdzie przełamać linię w ramach akapitu).
 - Też niezbyt zalecany sposób wymuszania formatowania
- Obydwa znaczniki nie definiują treści ani struktury, stąd są oznaką brzydkiego kodu.

Przykład SpecCharacters.html

- Początek kodu usunięty (oszczędność miejsca)

[illegible]

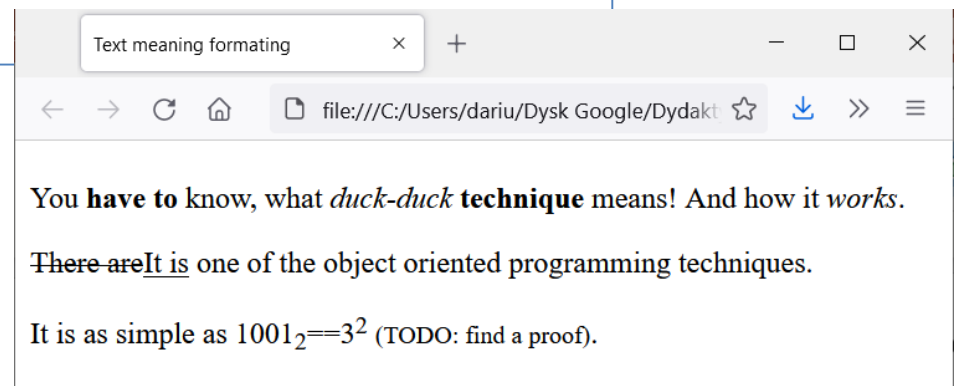
Znaczники formatujące tekst

- Znaczniki formatujące tekst można podzielić na dwa zbiory:
 - Wymuszające konkretne formatowanie (niezalecane, chyba że np. jakiś standard techniczny wymusza używanie pogrubienia lub tekstu pochyłego):
 - `` - tekst ma być pogrubiony (ang. bold)
 - `<i>` - tekst ma być pochylony (ang. italic)
 - Określające znaczenie (np. emocjonalne) tekstu, **semantyczne**, których domyślne działanie przedstawione poniżej może być zmienione przez CSS:
 - `` - ważny tekst, spowoduje pogrubienie tekstu
 - `` - akcentowany, podkreślony (ang. emphasized) tekst, spowoduje wyświetlenie pochyloną czcionką
 - `<small>` - mniejsza czcionka
 - `<mark>` - zaznaczony tekst
 - `` - tekst usunięty (będzie przekreślony)
 - `<ins>` - tekst wstawiony (gdy chcemy pokazać zmianę w stosunku do poprzedniej wersji), będzie podkreślony
 - `<sub>` - indeks dolny (ang. subscript)
 - `<sup>` - indeks górny (ang. superscript)

Kody specjalne i znaczniki formatujące – przykład użycia

- Początek kodu HTML5 usunięty (oszczędność miejsca), tak będzie też na kolejnych stronach (TextFormats.html).

```
<title>Text meaning formatting</title>
</head>
<body>
  <h1></h1>
  <p> You <strong>have to</strong> know, what <em>duck-duck</em>
    <b>technique</b> means! And how it <i>works</i>.
  </p>
  <p>
    <del>There are</del><ins>It is</ins> one of
    the object oriented programming techniques.
  </p>
  <p>
    It is as simple as  $1001_2 = 3^2$ 
    <small>(TODO: find a proof)</small>.
  </p>
</body>
</html>
```



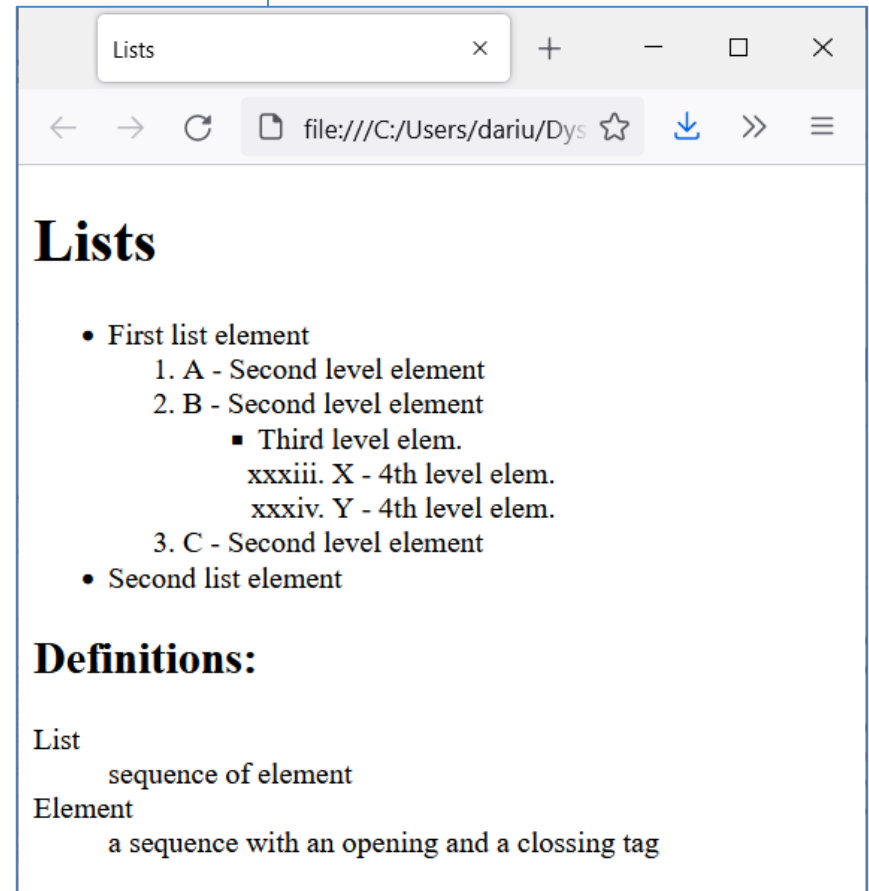
Listy

- Lista nieuporządkowana: `` (ang. unordered list)
- Lista uporządkowana: `` (ang. ordered list)
- Zagnieżdżone elementy takich list: **tylko** `` (ang. list item)
 - Dla nieuporządkowanych elementy będą poprzedzone wypunktowaniem kropką (można to zmienić przez atrybut `style="list-style-type: listStyle"`)
 - Dla uporządkowanych elementy będą poprzedzone kolejnymi liczbami naturalnymi zaczynając od 1 (można zmienić przez atrybut `start="numer"`)
- Uznawane są one jako elementy blokowe, zatem będą zawsze od nowej linii.
- Listy mogą być zagnieżdżone. Wówczas symbol wypunktowania lub format numeracji jest zmieniany, aby uwypuklić poziom listy.
- Lista definicji: `<dl>` (ang. definition list) zawiera pary:
 - Element definiowany termin: `<dt>` (ang. definition term)
 - Element definicji: `<dd>` (ang. definition description), tekst najczęściej z dużym wcięciem.

Przykład list i list zagnieżdżonych - Lists.html

```
<title>Lists</title>
</head>
<body>
  <h1>Lists</h1>
  <ul>
    <li>First list element
      <ol>
        <li> A - Second level element</li>
        <li> B - Second level element
          <ul>
            <li>Third level elem.
              <ol start="33" type="i">
                <li>X - 4th level elem.</li>
                <li>Y - 4th level elem.</li>
              </ol>
            </li>
          </ul>
        </li>
        <li> C - Second level element</li>
      </ol>
    </li>
    <li>Second list element</li>
  </ul>
  <h2>Definitions:</h2>
  <dl>
    <dt>List</dt>
    <dd>sequence of element</dd>
    <dt>Element</dt>
    <dd>a sequence with an opening and a closing tag</dd>
  </dl>

</body>
</html>
```

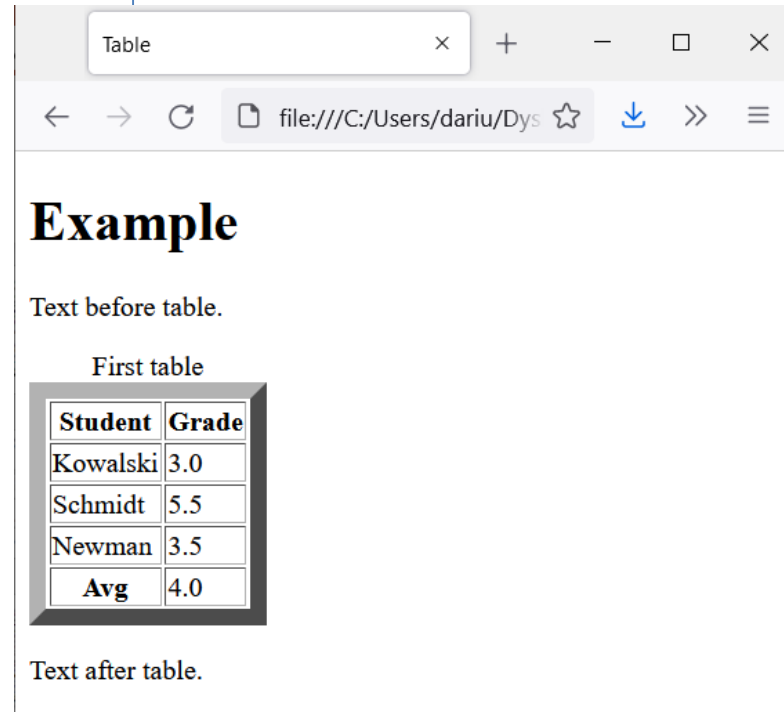


Tabele

- **Tabele** organizują dane w wiersze i kolumny. W kodzie układamy tabelę **wiersz po wierszu**, a wiersze dzielimy na **komórki**.
- Element `<table>` - definiuje tabelę HTML5.
 - Atrybut `border="X"` określa szerokość obwódki cieniującej wokół całej tabeli (**przestarzałe**)
 - Istnieje wiele atrybutów do prezentacji tabeli (użyte zostaną przy temacie CSS)
- W tabeli mogą wystąpić trzy odrębne sekcje (w tej kolejności):
 - element `<thead>` – nagłówek (ang. *head*), w który określa się nagłówki kolumn.
 - element `<tbody>` - ciało (ang. *body*), w którym będą główne dane tabeli
 - element `<tfoot>` - stopka (ang. *foot*), np. dla podsumowań kolumn
- Element `<tr>` - wiersz (ang. *row*), elementy wewnątrz w/w sekcji
- Element `<th>` - komórka nagłówkowa (ang. *header*), powinna być raczej **wewnątrz sekcji `<thead>` lub `<tfoot>`**
 - Standardowo czcionka pogrubiona, tekst wyśrodkowany
- Element `<td>` - komórka z danymi (ang. *data*), raczej w sekcji `<tbody>`
 - Standardowo czcionka zwykła, tekst dosunięty do lewej strony
- Element `<caption>` (**przed pozostałymi sekcjami**, najczęściej tuż za otwierającym znacznikiem `<table>`) określa tytuł tabeli.
 - Standardowo umieszczany nad tabelą.

Tabela – przykład - Table.html

```
<title>Table</title>
</head>
<body>
  <h1>Example</h1>
  <p>Text before table.</p>
  <table border="10">
    <caption>First table</caption>
    <thead>
      <tr>
        <th>Student</th>
        <th>Grade</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Kowalski</td>
        <td>3.0</td>
      </tr>
      <tr>
        <td>Schmidt</td>
        <td>5.5</td>
      </tr>
      <tr>
        <td>Newman</td>
        <td>3.5</td>
      </tr>
    </tbody>
    <tfoot>
      <tr>
        <th>Avg</th>
        <td>4.0</td>
      </tr>
    </tfoot>
  </table>
  <p>Text after table.</p>
</body>
</html>
```



Tabele – łączenie komórek, wierszy i kolumn

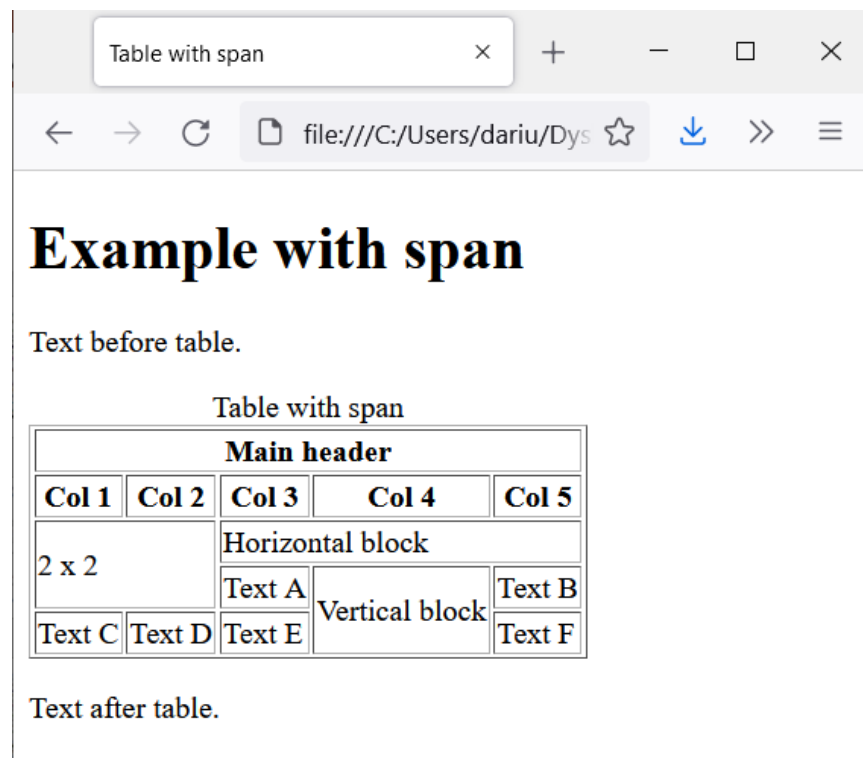
- Można scalać komórki z danymi (lub nagłówkowymi) za pomocą atrybutów `rowspan="X"` i `colspan="Y"` w elementach `<th>` lub `<td>`.
- Wartości tych atrybutów określają **liczbę wierszy** bądź **kolumn**, które mają być zajmowane przez komórkę.
- Kolejne `<td>` lub `<th>` w wierszu odnoszą się do dalszych komórek
- Jeśli użyto `colspan`, w kolejnych wierszach poprzez elementy komórek określa się zawartość **pozostałych, niescalonych** komórek.

The diagram shows a 5x3 grid representing a table. The first row is a single orange cell spanning all three columns (colspan=3). The second row consists of three white cells. The third row has a blue cell spanning two columns (colspan=2) and a light blue cell spanning one column (colspan=1). The fourth row has a white cell, a green cell spanning one column (colspan=1), and a white cell. The fifth row consists of three white cells. This visualizes how rowspan and colspan affect the layout of a table.

- Standardowy sposób renderowania tabeli dobiera wymiary wierszy i kolumn tak, aby wyświetlane dane się zmieściły.
- Poprzez atrybuty można samodzielnie ustalić rozmiary bezwzględne (np. „100px”) lub względne („30%” szerokości strony) komórki (czyli pośrednio wierszy i kolumn)
 - Ostatecznie wszystkie linie rozdzielające muszą tworzyć kratownicę (jak np. w Excelu).

Łączenie komórek – przykład - TableWithSpan.html

```
<title>Table with span</title>
</head>
<body>
  <h1>Example with span</h1>
  <p>Text before table.</p>
  <table border="1">
    <caption>Table with span</caption>
    <thead>
      <tr>
        <th colspan="5">Main header</th>
      </tr>
      <tr>
        <th>Col 1</th><th>Col 2</th><th>Col 3</th>
        <th>Col 4</th><th>Col 5</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td colspan="2" rowspan="2">2 x 2 </td>
        <td colspan="3">Horizontal block</td>
      </tr>
      <tr>
        <td>Text A</td>
        <td rowspan="2">Vertical block</td>
        <td>Text B</td>
      </tr>
      <tr>
        <td>Text C</td><td>Text D</td>
        <td>Text E</td><td>Text F</td>
      </tr>
    </tbody>
  </table>
  <p>Text after table.</p>
</body>
</html>
```



Tabele – inne informacje ogólne

- Oczywiście zawartością komórek może być wszystko: tekst, link, obrazek, inna tabela itd.
- Standardowe renderowanie dobierze tak szerokości i wysokości, aby wszystko się mieściło
- Można wymusić pewien minimalny rozmiar przez atrybuty dla wymiarów, marginesów wewnętrznych, zewnętrznych, grubości ramek itp.
- Jakikolwiek tekst, elementy, które nie będą w ramach elementów komórek zostanie zaprezentowany przez przeglądarkę PRZED tabelą.
 - Najczęściej to błąd walidacji, **nie powinno być takich elementów**.
- Tabela może służyć do podziału strony, szczególnie, gdy wyłączy się rysowanie krawędzi rozdzielających.
 - **Nie jest** to jednak **wskazane**, gdyż taką stronę często trudno zrobić *responsywną*, czyli reagującą na zmianę wymiarów (obrót ekranu komórki itp., zmniejszenie/zwiększenie okna)

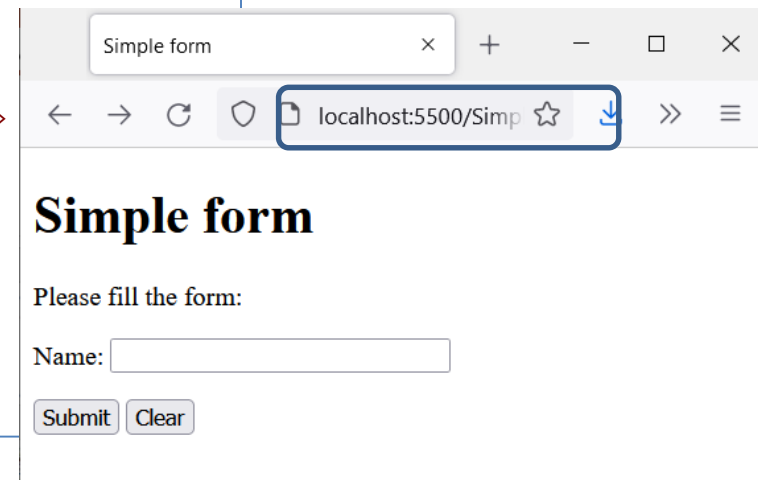
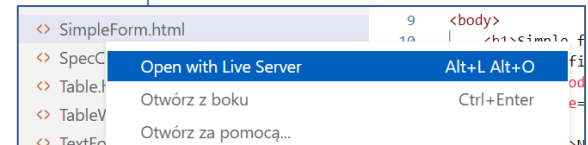
Atrybut `id` - identyfikator

- Praktycznie każdy element w ramach HTML może posiadać swój **niepowtarzalny identyfikator**.
 - Dzięki temu łatwo sięgać do takiego elementu w JS, CSS3 itp.
 - Może być wręcz wymagany w funkcjach JS, różnych frameworkach.
- Jest on przydzielany elementowi za pomocą atrybutu `id`, czyli `id="idValue"`.
- Wielkość liter w identyfikatorze jest istotna.
- W zasadzie zabronione są tylko białe znaki.
 - Aby zachować zgodność z HTML4 nie wskazane jest zaczynać identyfikator od cyfry.
- Jako rozdzielnik wyrazów można stosować minus `-` lub podkreślnik `_`.
 - Standardowo używany minus następuje potem trudności w automatycznym mapowaniu identyfikatora na właściwości w językach programowania (np. JavaScript).

Formularze i ich pola

- Formularze służą do zbierania informacji od użytkowników strony WWW.
- Poniżej kod prostego formularza z jednym polem i jego wygląd.
- Dla obserwacji działania formularza w VS Code warto zainstalować rozszerzenie Live Server, otworzyć stronę za pomocą tego serwera i w przeglądarce wpisać adres:
<http://localhost:5500/SimpleForm.html>
- Przykład: SimpleForm.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple form</title>
</head>
<body>
  <h1>Simple form</h1>
  <p>Please fill the form:</p>
  <!-- <form method="post" action="superweb.mydomain.com/userinfo.html"> -->
  <form method="post" action="http://localhost:5500/SimpleForm.html">
    <input type="hidden" name="securityCode" value="DE79W23">
    <p>
      <label for="idName">Name:</label>
      <input id="idName" name="Name" type="text" size="30" maxlength="50">
    </p>
    <p>
      <input type="submit" value="Submit">
      <input type="reset" value="Clear">
    </p>
  </form>
</body>
</html>
```



Formularz - `<form>`

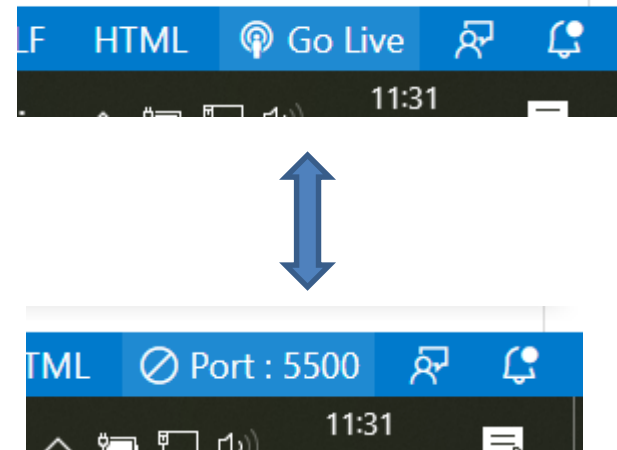
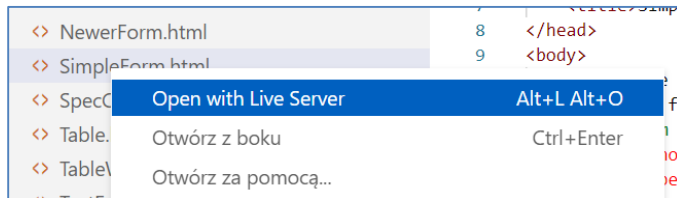
- Formularz znajduje się wewnątrz elementu `<form>`
- Atrybut `method` określa, w jaki sposób dane formularza są wysyłane na serwer WWW.
 - Wartość `"post"` powoduje przekazanie danych formularza w formacie query string w ciele żądania POST wysyłanego przez przeglądarkę.
 - Wartość `"get"` powoduje przekazanie danych formularza w formacie query string w URL żądania GET wysyłanego przez przeglądarkę.
- Atrybut `action` elementu `<form>` określa odbiorcę danych z formularza (ang. form handler) :
 - Może to być adres URL
 - Może to być metoda JavaScript
- Formularze mogą zawierać widoczne i niewidoczne składowe.
 - Widoczne komponenty to m.in. klikalne przyciski i inne komponenty graficzne, które wykorzystują użytkownicy.
 - Niewidoczne komponenty, zwane ukrytymi polami, przechowują dowolne dane tekstowe, np. specjalne kody bezpieczeństwa, dane z poprzednich stron formularza itp.
- Na stronie może być **wiele** formularzy. Są one niezależne od siebie i dane tylko z jednego są zamieniane na jedno żądanie HTTP.

Podstawowe typy pól formularza

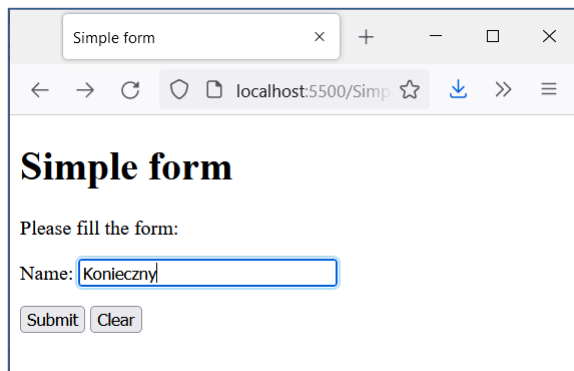
- Elementy `<input>` określają dane dostarczane do odbiorcy formularza.
- **Rodzaj/typ** elementu `<input>` jest określony przez atrybut `type`.
- Typ `hidden` zawiera ukryte elementy, które muszą posiadać również atrybut `name` i `value`. Para tych wartości jest przesyłana w query string.
- Typ `text` elementu `<input>` powoduje wstawienie do formularza pola tekstowego, aby użytkownik mógł wprowadzić dane.
 - Atrybut `size` określa liczbę znaków widoczną w polu tekstowym.
 - Opcjonalny atrybut `maxlength` to maksymalna liczba znaków możliwa do wpisania w polu tekstowym.
- Element `<label>`, czyli etykieta, dostarcza użytkownikom informacji o przeznaczeniu danego elementu `<input>`.
 - Może posiadać atrybut `for="idOfInput"`, wskazujący że jest to etykieta dla elementu `<input>` o identyfikatorze `idOfInput`.
- Typ `submit` elementu `<input>` powoduje wstawienie do formularza przycisku z napisem z atrybutu `value`.
 - W momencie przyciśnięcia przycisku typu `submit`, dane formularza są wysyłane do lokalizacji określonej w atrybucie `action` formularza.
- Typ `reset` elementu `<input>` wstawia do formularza przycisk z napisem z atrybutu `value`, którego naciśnięcie czyści wszystkie elementy formularza i przywraca je do domyślnych wartości.

Obserwacja komunikacji w VS Code – rozszerzenie Live Server

- Aby obserwować komunikację HTTP np. w przeglądarce musi być uruchomiony serwer
 - Nie wystarczy otworzyć plik z eksploratora plików, bo nie jest on serwerem HTTP.
- Np. w środowisku Visual Studio Code najprościej zainstalować rozszerzenie o nazwie Live Server (identyfikator rozszerzenia: `ritwickdey.liveserver`)
- Serwer ten nie jest przygotowany na odebranie danych z zapytania POST, ale zasymuluje próbę obsługi takiego żądania odpowiadając kodem 405 Method not Allowed.
 - Ale pozwoli to na obserwację nagłówków i ciała żądania.
- Uruchomienie można zrobić poprzez odpowiednie otwarcie pliku .html
 - Domyślnym portem jest 5500.
- Można również uruchomić i zatrzymać serwer w prawym dolnym rogu VS Code.
- Adresem URL jest wtedy np. `http://localhost:5500/SimpleForm.html`



Żądanie POST – obserwacja w przeglądarce - SimpleForm.html

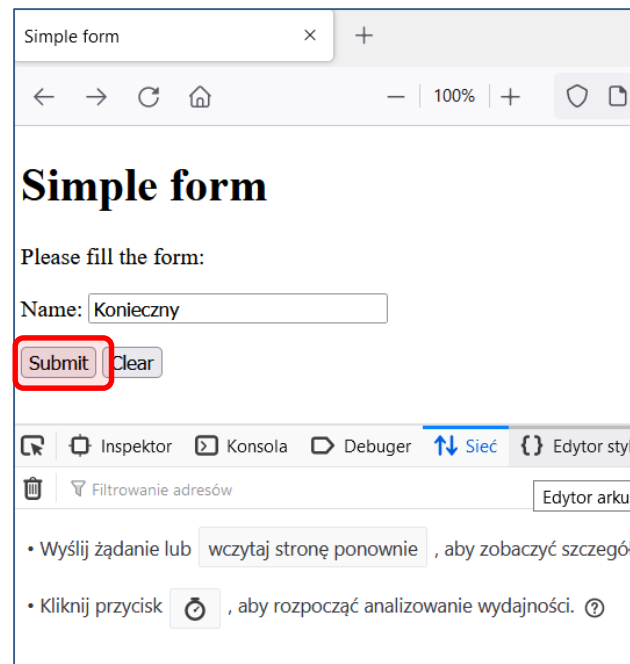


Simple form

Please fill the form:

Name:

F12




Simple form

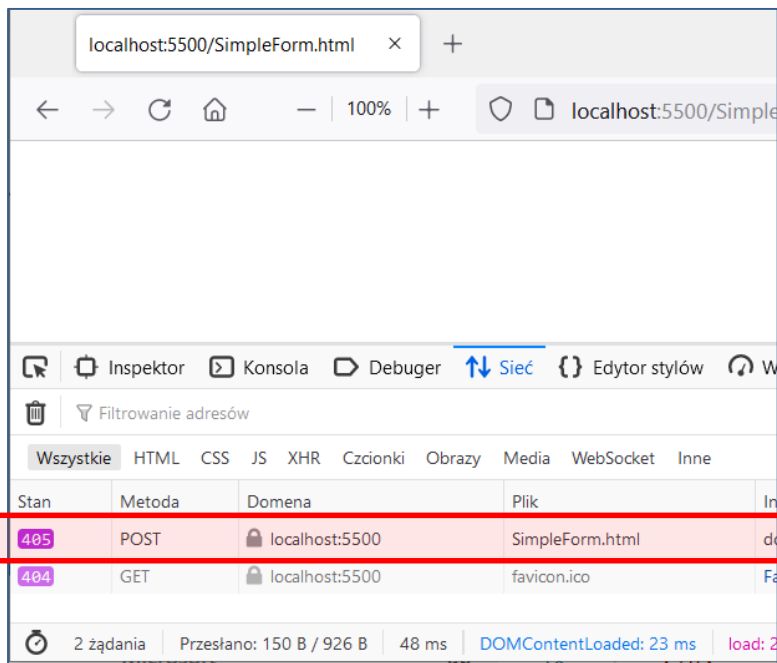
Please fill the form:

Name:

Inspektor Konsola Debugger Sieć Edytor stylów

Filtrowanie adresów Edytor arkuszy

- Wyślij żądanie lub wczytaj stronę ponownie, aby zobaczyć szczegóły
- Kliknij przycisk , aby rozpocząć analizowanie wydajności. ?



Stan	Metoda	Domena	Plik
405	POST	localhost:5500	SimpleForm.html
404	GET	localhost:5500	favicon.ico

2 żądania Przesłano: 150 B / 926 B 48 ms DOMContentLoaded: 23 ms load: 2

Ciało odpowiedzi
na żądanie POST
jest puste

Żądanie POST i odpowiedź na nie

Sta	Met	Domena	Plik	Inicja...	Ty	Przes...	Rozmiar	Nagłówki	Ciasteczka	Żądanie	Odpowiedź
40!	P...	local...	SimpleForm	docu...	htn	926 B	0 B	Filtruj nagłówki			
40!	G...	local...	favicon.ico	Favic...	htn	w pa...	150 B				
▶ POST http://localhost:5500/SimpleForm.html											
Stan		405 Method Not Allowed ?									
Wersja		HTTP/1.1									
Przesłano		926 B (o rozmiarze 0 B)									
Zasada polecającego		strict-origin-when-cross-origin									
▼ Nagłówki odpowiedzi (271 B)											
?		Access-Control-Allow-Credentials: true									
?		Access-Control-Allow-Origin: http://localhost:5500									

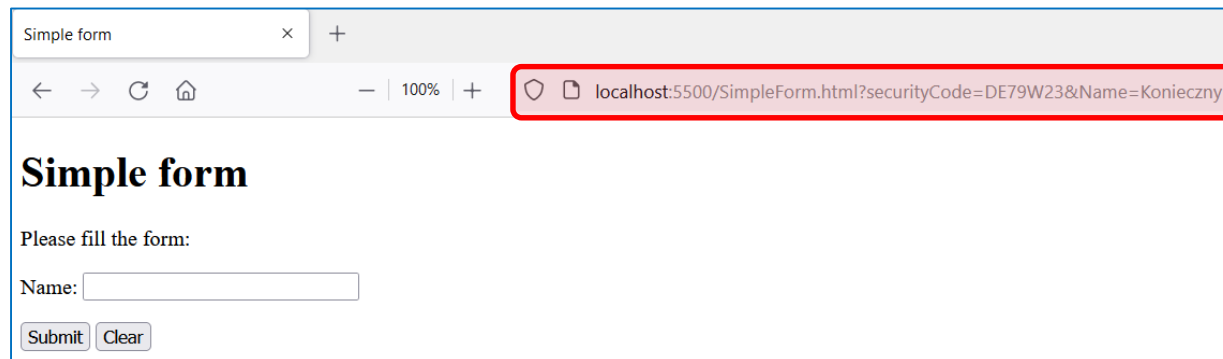
Dopiero jak poznamy
ASP .NET
będziemy mogli obsłużyć
takie żądania
Obecnie serwer zwraca błąd
405 Method not Allowed

Wszystkie HTML CSS JS XHR Czcionki Obrazy Media WebSocket Inne											
Sta	Me	Domena	Plik	Inicja...	Ty	Przes...	Rozmiar	Nagłówki	Ciasteczka	Żądanie	Odpowiedź
40!	P...	local...	SimpleFor	docu...	htn	926 B	0 B	Filtruj parametry żądania			
40!	G...	local...	favicon.ico	Favic...	htn	w pa...	150 B				
Zawartość żądania										Nieprzetworzone	🔴
1		securityCode=DE79W23&Name=Konieczny									

W samym żądaniu POST, w ciele, są pola formularza w formacie
query string.

Żądanie POST i puste ciało odpowiedzi

- Pytanie: Dlaczego normalnie nie widzimy pustej strony po wysłaniu formularza na stronach WWW za pomocą żądania POST?
- Odpowiedź: Ciało odpowiedzi jest puste, jednak przy poprawnej obsłudze otrzymujemy odpowiedź o innym kodzie, a dodatkowo w części nagłówkowej istnieje nagłówek „Location:”, który przeglądarka interpretuje jako potrzebę wykonania żądania GET na podany adres URL.
- Pod podanym adresem jest już „normalna” strona zawierająca kod HTML.
- Oznacza to, że **obsługa formularza** to najczęściej **minimum dwa żądania** wykonane przez przeglądarkę
- Przykład wysyła dane formularza z powrotem do tej samej strony (co może być poprawne, bo wysyła **inny rodzaj żądania**), ale może wysłać na całkiem inny adres. Gdybyśmy chcieli obsłużyć to żądanie należałoby napisać obsługę na serwerze (na prostym Live Server nie da się tego łatwo zrobić).
- Dane z formularza można też wysłać do metody napisanej w JavaScript, ale wtedy atrybut `action` ma inny format (nie jest to URL).
- Zmiana metody na `method="get"` spowoduje wysłanie danych w URL-u:



Formularz – pole `<textarea>` i typy pola `<input>`

- Element `<textarea>` wstawia wielolinijkowe pole tekstu do formularza.
 - Liczbę wierszy określana się atrybutem `rows` a liczbę kolumn (tj. znaków na jedną linię) atrybutem `cols`.
- Typ `password` elementu `<input>` służy do wpisywania w formularzu hasła.
 - Wpisywane dane (hasło, numer karty kredytowej itp.) jest „maskowanie” najczęściej poprzez znaki gwiazdki.
 - W żądaniu wysyłana jest właściwa wartość pola, nie zaś gwiazdki lub inne znaki maskujące.
- Typ `checkbox` elementu `<input>` pozwala użytkownikom dokonywać wyboru.
 - Kiedy checkbox jest zaznaczony, pojawia się znaczek wyboru (ang. check mark) wewnątrz checkboxa. Inaczej checkbox jest pusty.
 - Checkboxy mogą być wykorzystane pojedynczo i w grupach. Checkboxy, które są częścią tej samej grupy, mają tę samą nazwę (atrybut `name`). Wtedy ich miejsce w formularzu w zasadzie nie ma znaczenie, chociaż powinny być wizualnie blisko siebie.
 - Najczęściej formularz ma wiele checkboxów z tą samą nazwą. Dlatego należy się upewnić, że wszystkie mają inne wartości; w przeciwnym wypadku skrypt na serwerze nie będzie w stanie ich odróżnić.
 - W przypadku wybrania kilku opcji w query stringu będzie wiele par, gdzie nazwa będzie taka sama, ale różne wartości

Formularz – typy pola `<input>` i pole `<select>` z `<option>`

- Typ `radio` elementu `<input>` działa podobnie do typu `checkbox`, jednak tylko jeden `radio button` z grupy może być wybrany w tym samym czasie.
 - Wszystkie `radio buttony` w grupie mają tę samą nazwę (atrybut `name`), ale inną wartość (atrybut `value`).
- Element formularza `<select>` tworzy rozwijane listy.
 - Atrybut `name` identyfikuje listę rozwijaną.
- Element `<option>` dodaje pozycję do listy rozwijanej (musi być wewnątrz takiego elementu).
 - Poprzez atrybuty można ustawić możliwość wielu lub tylko jednej wartości z listy rozwijanej.
 - Standardowo wybrana wartość (tekst) staje się wartością dla podanego `name`.
 - Można to zmienić dodając atrybut `value` do elementu `<option>`.
- Elementy wyboru mogą posiadać atrybut `checked` (bez znaku równa się `,=` i wartości) oznaczających wstępnie wybranie opcji.
- Atrybut bez wartości to skrót notacyjny. Oznacza atrybut, którego wartość jest równa nazwie atrybutu.
 - Czyli atrybut `checked` oznacza `checked="checked"`.
- Jeśli jako wartość będą spacje lub inne specjalne znaki, będą specjalnie kodowana w query string.

Formularz – przykład ComplexForm.html (1/2)

```
<title>Complex form</title>
</head>
<body>
  <h1>Complex form</h1>
  <p>Please fill the form:</p>
  <!-- <form method="post" action="superweb.mydomain.com/userinfo.html" -->
  <form method="post" action="http://localhost:5500/SimpleForm.html">
  <p>
    <textarea name="descr" cols="30" rows="3">Short description</textarea>
  </p>
  <p>
    <label for="idPass">Password:</label>
    <input id="idPass" name="Name" type="password" size="30" maxlength="50">
  </p>
  <p>
    <label>What elements you know?:</label>
    <label><input name="knowledge" type="checkbox" value="html5" checked>HTML 5</label>
    <label><input name="knowledge" type="checkbox" value="css3">CSS 3</label>
    <label><input name="knowledge" type="checkbox" value="js">JavaScript</label>
  </p>
</body>
```

Complex form

Please fill the form:

Short description

Password:

What elements you know?: ☒ HTML 5 ☐ CSS 3 ☐ JavaScript

How would you rate your OOP knowledge?: ☐ 5 ☒ 4 ☐ 3

How would you rate this course? Excellent

Submit Clear

Please fill the form:

Probe with password "abc"

Password: ●●●

What elements you know?: ☐ HTML 5 ☒ CSS 3 ☐ JavaScript

How would you rate your OOP knowledge?: ☒ 5 ☐ 4 ☐ 3

How would you rate this course? Bad & worse

Submit Clear

1 descr=Probe+with+password+%22abc%22&Name=abc&knowledge=css3&oop=grade5&course=Bad+%26+worse

Formularz – przykład ComplexForm.html (2/2)

```
<p>
  <label>How would you rate your OOP knowledge?:</label>
  <label><input name="oop" type="radio" value="grade5">5</label>
  <label><input name="oop" type="radio" value="grade4" checked="">4</label>
  <label><input name="oop" type="radio" value="grade3">3</label>
</p>
<p>
  <label>How would you rate this course?
    <select name="course">
      <option>Excellent</option>
      <option>Good</option>
      <option value="canbe">Can be</option>
      <option>Bad & worse</option>
      <option value="0">Horrible</option>
    </select>
  </label>
</p>
<p>
  <input type="submit" value="Submit">
  <input type="reset" value="Clear">
</p>
</form>
</body>
</html>
```

Complex form

Please fill the form:

No password

Password:

What elements you know?: ☐ HTML 5 ☒ CSS 3 ☒ JavaScript

How would you rate your OOP knowledge?: ☐ 5 ☐ 4 ☒ 3

How would you rate this course? Horrible

Submit Clear

Zawartość zapytania

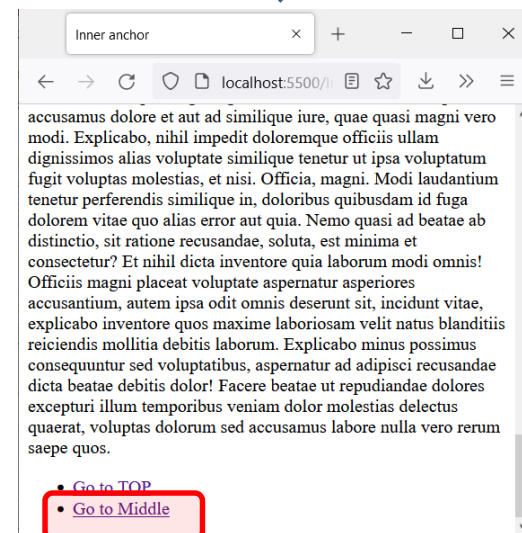
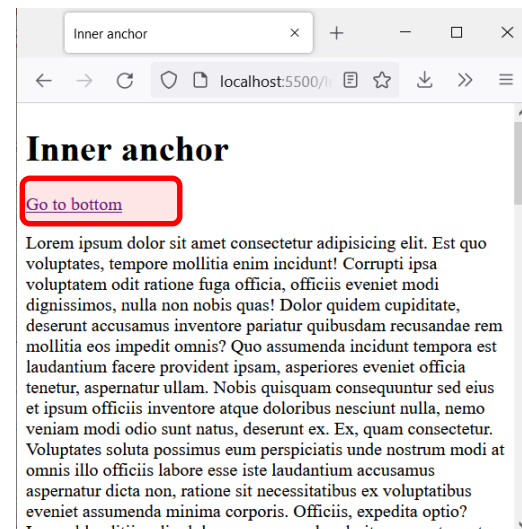
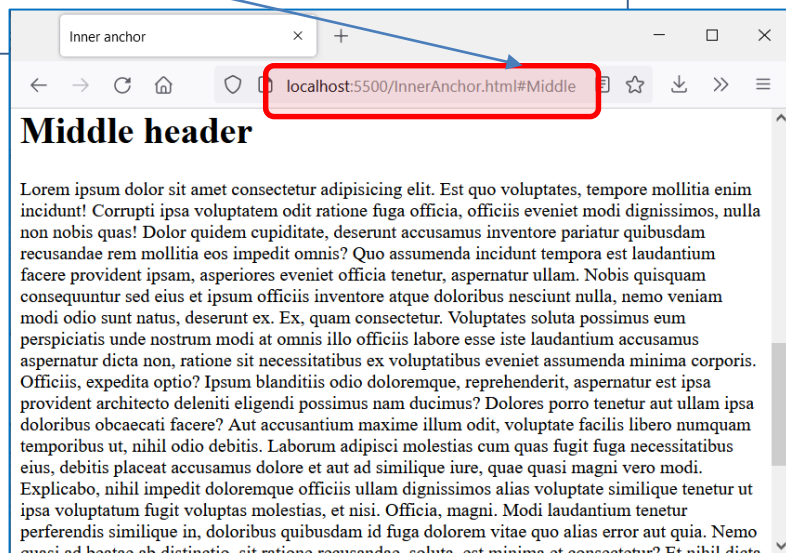
1 descr=No+password&Name=&knowledge=css3&knowledge=js&oop=grade3&course=0

Linkowanie wewnętrzne

- Znacznik `<a>` może być wykorzystany do linkowania do innych sekcji tego samego dokumentu przez podanie `id` innego elementu jako atrybutów `href`.
 - Dokładniej w cudzysłowach wartości `href` musi być znak `'#'`, po który bezpośrednio wpisana jest wartość `id` tegoż elementu.
- W adresie URL pojawia się segment opisujący dostęp do tej sekcji po znaku `'#'`, np.
 - <http://localhost:5500/InnerAnchor.html#Middle>
- Oczywiście taki adres można też wpisać z klawiatury lub podlinkować z innej strony.
- Jeśli taki identyfikator na stronie nie istnieje i ta część adresu URL jest niepoprawna, jest ona ignorowana.

Linkowanie wewnętrzne – przykład InnerAnchor.html

```
<title>Inner anchor</title>
</head>
<body>
  <h1 id="Top">Inner anchor</h1>
  <p>
    <a href="#Bottom">Go to bottom</a>
  </p>
  <!-- lorem300<TAB> -->
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Est quo voluptates, tempore mollitia enim incididunt! Corrupti ipsa voluptatem odit ratione fuga officia, officiis eveniet modi dignissimos, nulla non nobis quas! Dolor quidem cupiditate, deserunt accusamus inventore pariatur quibusdam recusandae rem mollitia eos impedit omnis? Quo assumenda incididunt tempora est laudantium facere provident ipsam, asperiores eveniet officia tenetur, aspernatur ullam. Nobis quisquam consequuntur sed eius et ipsum officiis inventore atque doloribus nesciunt nulla, nemo veniam modi odio sunt natus, deserunt ex. Ex, quam consectetur. Voluptates soluta possimus eum perspiciatis unde nostrum modi at omnis illo officiis labore esse iste laudantium accusamus aspernatur dicta non, ratione sit necessitatibus ex voluptatibus eveniet assumenda minima corporis. Officiis, expedita optio? Ipsum blanditiis odio doloremque reprehenderit, aspernatur est ipsa provident architecto deleniti eligendi possimus nam ducimus? Dolores porro tenetur aut ullam ipsa doloribus obcaecati facere? Aut accusantium maxime illum odit, voluptate facilis libero numquam temporibus ut, nihil odio debitis. Laborum adipisci molestias cum quas fugit fuga necessitatibus eius, debitis placeat accusamus dolore et aut ad similique iure, quae quasi magni vero modi. Explicabo, nihil impedit doloremque officiis ullam dignissimos alias voluptate similique tenetur ut ipsa voluptatum fugit voluptas molestias, et nisi. Officia, magni. Modi laudantium tenetur perferendis similique in, doloribus quibusdam id fuga dolorem vitae quo alias error aut quia. Nemo quasi ad beatae ab distinctio, sit ratione recusandae, soluta, est minima et consectetur? Et nihil dicta
  <h1 id="Middle">Middle header</h1>
  <!-- lorem300<TAB> -->
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Est quo voluptates, tempore mollitia enim incididunt! Corrupti ipsa voluptatem odit ratione fuga officia, officiis eveniet modi dignissimos, nulla non nobis quas! Dolor quidem cupiditate, deserunt accusamus inventore pariatur quibusdam recusandae rem mollitia eos impedit omnis? Quo assumenda incididunt tempora est laudantium facere provident ipsam, asperiores eveniet officia tenetur, aspernatur ullam. Nobis quisquam consequuntur sed eius et ipsum officiis inventore atque doloribus nesciunt nulla, nemo veniam modi odio sunt natus, deserunt ex. Ex, quam consectetur. Voluptates soluta possimus eum perspiciatis unde nostrum modi at omnis illo officiis labore esse iste laudantium accusamus aspernatur dicta non, ratione sit necessitatibus ex voluptatibus eveniet assumenda minima corporis. Officiis, expedita optio? Ipsum blanditiis odio doloremque reprehenderit, aspernatur est ipsa provident architecto deleniti eligendi possimus nam ducimus? Dolores porro tenetur aut ullam ipsa doloribus obcaecati facere? Aut accusantium maxime illum odit, voluptate facilis libero numquam temporibus ut, nihil odio debitis. Laborum adipisci molestias cum quas fugit fuga necessitatibus eius, debitis placeat accusamus dolore et aut ad similique iure, quae quasi magni vero modi. Explicabo, nihil impedit doloremque officiis ullam dignissimos alias voluptate similique tenetur ut ipsa voluptatum fugit voluptas molestias, et nisi. Officia, magni. Modi laudantium tenetur perferendis similique in, doloribus quibusdam id fuga dolorem vitae quo alias error aut quia. Nemo quasi ad beatae ab distinctio, sit ratione recusandae, soluta, est minima et consectetur? Et nihil dicta
  <ul id="Bottom">
    <li><a href="#Top">Go to TOP</a></li>
    <li><a href="#Middle">Go to Middle</a></li>
  </ul>
</body>
</html>
```

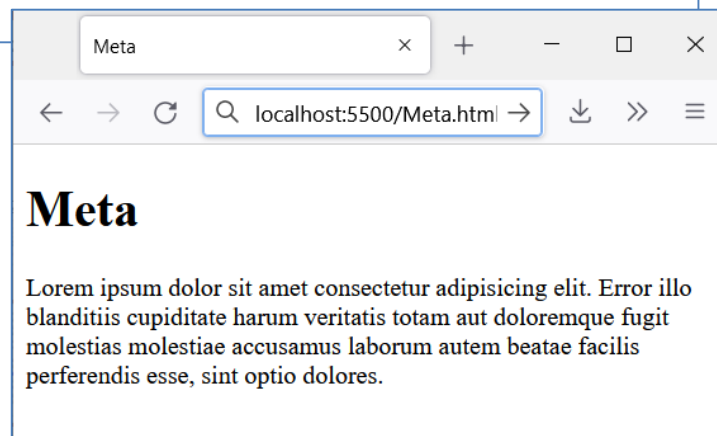


Elementy `<meta>`

- Elementy `<meta>` stosuje przeglądarka do ustawienia pewnych globalnych właściwości
 - Powinien być stosowany w nagłówku i odnosi się do całego dokumentu.
- Treści elementów `<meta>` jest często wykorzystywana do katalogowania stron WWW przez wyszukiwarki.
- Jest to element pusty
- Atrybut `name` identyfikuje typ elementu meta. W zależności od jego wartości zmienia znaczenie atrybutu `content`. Stanowią poniekąd parę nazwa-wartość.
- Znaczenie atrybutu `content` dla `name` równego (zapisanego w cudzysłowie):
 - `keywords` – słowa (wyrażenia) kluczowe opisujące stronę, rozdzielone przecinkiem
 - `description` - kilkuliniowy opis strony w formie zdań. Tekst ten jest czasem wyświetlany jako część wyników wyszukiwania i jest używany w wyszukiwarkach.
- Elementy te nie są widoczne na stronie WWW.

Elementy <meta> - przykład Meta.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="keywords" content="html5, tag meta, example">
  <meta name="description" content="The file contains an example of using a meta tag.
  inside the body the is only the lorem text. ">
  <title>Meta</title>
</head>
<body>
  <h1>Meta</h1>
  <p>Lorem ipsum dolor sit amet ...</p>
</body>
</html>
```



Nowe typy pola `<input>` dla koloru i czasu

Wszystkie dalsze typy mogą mieć różną reprezentację graficzną i funkcjonalności w różnych przeglądarkach.

- Typ `color` – do ustawienia koloru
- Typ `date` – do ustawienia daty
- Typ `datetime-local` – ustawia się datę i czas
 - Istnieje niezalecany typ `datetime`, który przeglądarki ignorują obecnie
- Typ `month` – ustawienie tylko miesiąca i roku
- Typ `time` – ustawienie czasu
- Typ `week` - ustawienie numeru tygodnia w roku

Dalej przedstawione zostanie domyślne działania, ale poprzez różne atrybuty można je zmienić:

- np. czy podać sekundy w czasie, zakres dat itp.

Przykład użycia - NewTypesForm.html

```
<title>New types form</title>
</head>
<body>
  <h1>New types form</h1>
  <p>Please fill the form:</p>
  <form method="post" action="http://localhost:5500/NewerForm.html">
    <p>
      <label>Color:</label>
      <input id="idColor" name="Color" type="color" value="#FF00FF">
    </p>
    <p>
      <label>Date:</label>
      <input id="idDate" name="Date" type="date" value="2021-09-12">
    </p>
    <p>
      <label>Datetime-local:</label>
      <input id="idDateTimeLocal" name="DateTimeLocal" type="datetime-local">
    </p>
    <p>
      <label>Month:</label>
      <input id="idMonth" name="Month" type="month">
    </p>
    <p>
      <label>Time:</label>
      <input id="idTime" name="Time" type="time">
    </p>
    <p>
      <label>Week:</label>
      <input id="idWeek" name="Week" type="week">
    </p>
    <input type="submit" value="Submit">
    <input type="reset" value="Clear">
  </form>
</body>
</html>
```

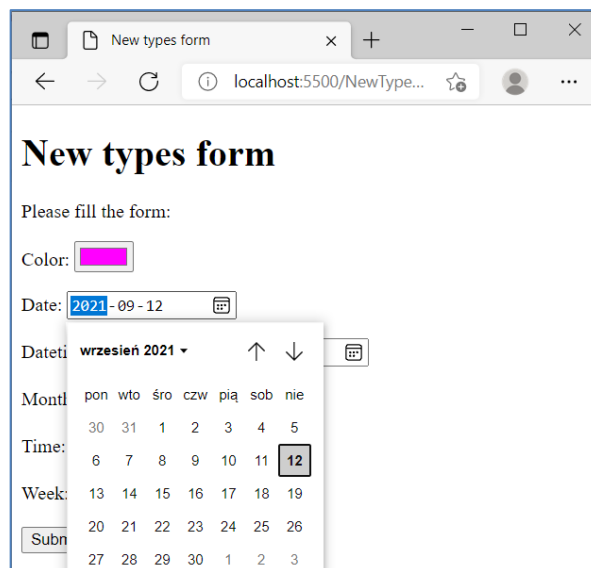
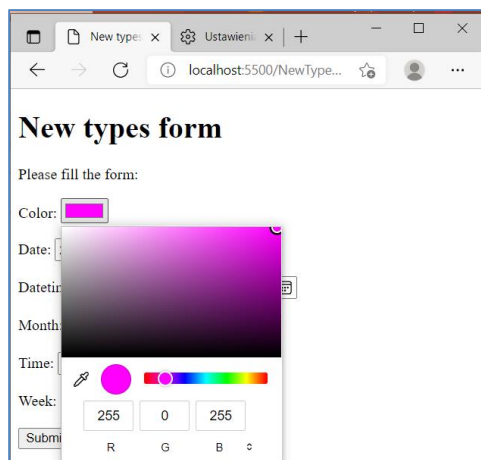
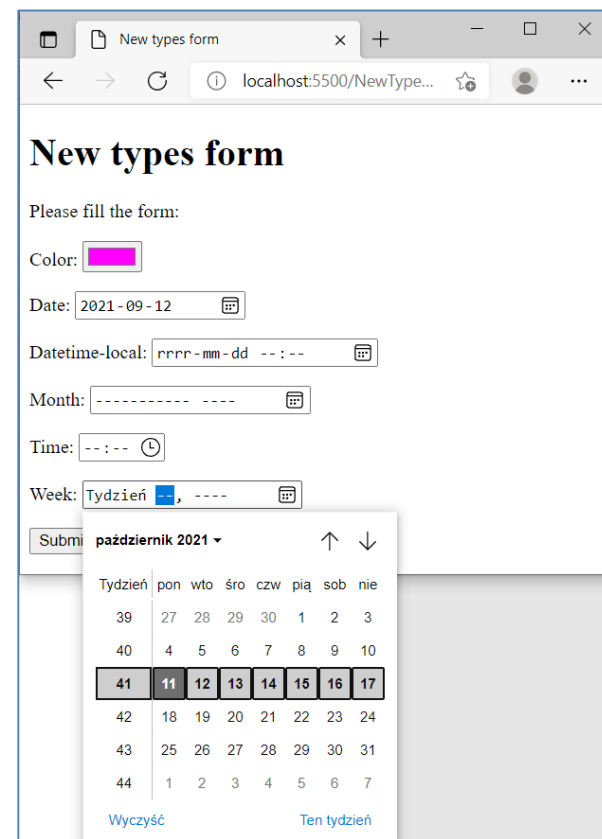
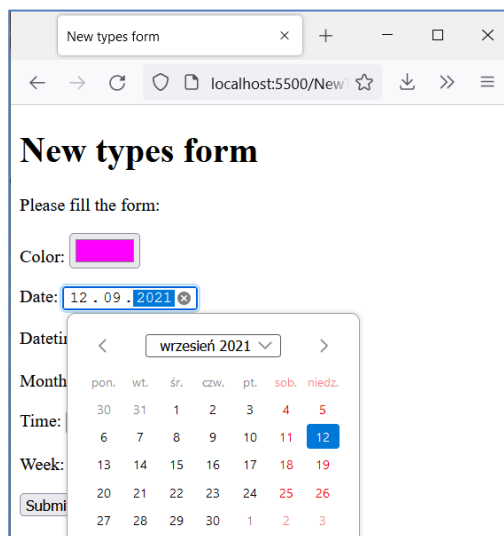
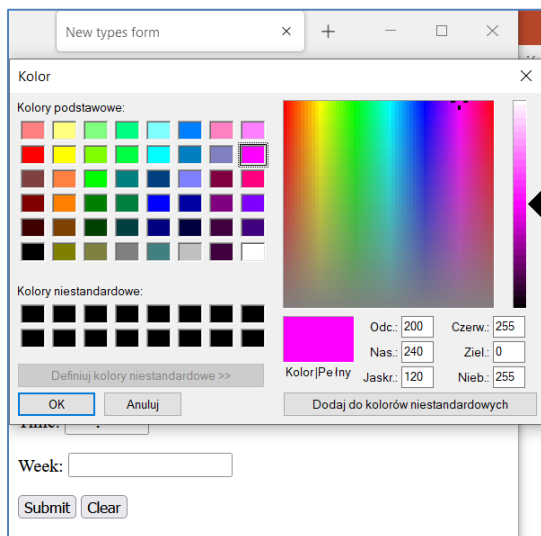
The screenshot shows the Firefox 93.0 browser window with the address bar at localhost:5500/New... The page title is "New types form". The form contains the following fields: Color (a color picker showing magenta), Date (a date picker showing 12.09.2021), Datetime-local (a datetime-local picker showing dd.mm.yyyy, --:--), Month (a month picker), Time (a time picker showing --:--), Week (a week picker), and Submit/Clear buttons.

Firefox 93.0

The screenshot shows the MS Edge 94.0 browser window with the address bar at localhost:5500/NewType... The page title is "New types form". The form contains the following fields: Color (a color picker showing magenta), Date (a date picker showing 2021-09-12), Datetime-local (a datetime-local picker showing yyyy-mm-dd --:--), Month (a month picker), Time (a time picker showing --:--), Week (a week picker showing Tydzień --, ----), and Submit/Clear buttons.

MS Edge 94.0

Różny interfejs przeglądarek - NewTypesForm.html



Nowe typy pola `<input>` dla sprawdzania formatu i inne

- Typ `email` – sprawdza czy tekst jest poprawnym emailem
- Typ `tel` - sprawdza, czy numer telefonu jest w poprawnym formacie
- Typ `url` – sprawdza, czy podany tekst stanowi poprawny URL
- Typ `number` – dla danych do wpisania, które są liczbą (całkowitą)
 - Atrybuty `min`, `max`, `step`
- Typ `range` – suwak, z zakresami, możliwą skalą, opisami na skali podziałką itp.
- Typ `search` – pojawia się ikona usunięcia wpisanego słowa
- Jeśli jakieś dane nie spełniają formatu, przeglądarka nie pozwala wysłać danych formularza i wskazuje użytkownikowi, które dane były niepoprawne
- Nie trzeba pisać kodu JavaScript dla sprawdzania.
- **Zawsze** trzeba pamiętać, że **na serwerze** i tak trzeba **ponownie walidować** otrzymane dane, bo ktoś może wysłać dowolne zapytanie z dowolnymi danymi za pomocą np. `curl`.

Inne informacje związane z polami formularza

- Atrybut `autocomplete` – pozwala skorzystać z danych użytkownika przechowywanych w przeglądarce, a wpisywanych w innych formularzach pod tym samym atrybutem `name`.
- Element `<datalist>`
 - Zestaw możliwych predefiniowanych wartości, wpisywanych jak lista elementów `<option>`
 - Możliwość/potrzeba dołączenia do innego pola za pomocą atrybutu `list`.
 - Wpisanie w pole znaków powoduje w liście filtrowanie, czyli pozostawienie na liście tylko wartości pasujących do wpisanych znaków.
 - W polu nadal można wpisać inną wartość niż w występującą w `<datalist>`
- Atrybut `placeholder` – opis (domyślnie szary) wewnątrz elementu formularza, który znika po wpisaniu pierwszego znaku.
- Dla pola `<input>` istnieje też `type="file"` do przesyłania plików.
 - Jeśli tego typu pole używane jest w formularzu, musi on mieć ustawione `method="post"` oraz dodatkowo atrybut `enctype="multipart/form-data"`.
- Można tworzyć przycisk z obrazem, gdy użyje się typu `type="image"`.
 - Wówczas w atrybucie `src` podajemy lokalizację pliku graficznego
 - Można użyć atrybutów `width`, `height`, `alt` jak dla elementu ``.
- Elementy formularza można wizualnie grupować w ramce z opisem za pomocą elementu `<fieldset>`.
 - Element ten może mieć też wewnętrzny element `<legend>` do opisu przeznaczenia tej grupy pól formularza.

Przykład użycia - Datalist.html

```
<title>Datalist</title>
</head>
<body>
  <h1>Datalist</h1>
  <form method="post" action="http://localhost:5500/SimpleForm.html">
    <p>
      <label for="flower-choice">Choose a flower:</label>
      <input list="flowers" id="flower-choice" name="FlowerChoice"
        placeholder="flower's name">

    <datalist id="flowers">
      <option value="Brugmansia"/>
      <option value="Rose"/>
      <option value="Tulip"/>
      <option value="Pansy"/>
      <option value="Azalea"/>
    </datalist>
  </p>
  <p>
    <input type="submit" value="Submit">
    <input type="reset" value="Clear">
  </p>
</form>
</body>
</html>
```

Datalist

Choose a flower:

Submit

Clear

Datalist

Choose a flower:

Submit

Clear

Brugmansia
Rose
Tulip
Pansy
Azalea

Datalist

Choose a flower:

Submit Clear

Brugmansia
Tulip

Elementy struktury strony

Elementy podziału logicznego struktury dokumentu, tworzące grupy lub wydzielone fragmenty, przydatne przy tworzenie wzorca strony.

- Element `<header>` - nagłówek strony lub sekcji (nie mylić z elementami `<h1>..<h6>`), coś w rodzaju banneru.
- Element `<main>` - główna część merytoryczna strony
- Element `<footer>` - stopka strony/sekcji
- Element `<nav>` - element nawigacyjny, menu strony. Najczęściej zawiera listę (`/`, również zagnieżdżoną) linków do tej samej strony, innych stron aplikacji, stron zewnętrznych. Często wewnątrz elementu `<header>`.
- Element `<article>` - logicznie wydzielony artykuł tekstowy
- Element `<section>` - sekcja
- Element `<aside>` - zawartość poza głównymi informacjami strony (dygresje)
- Wszystkie te elementy są szczególnie przydatne z wykorzystaniem CSS, aby wydzielić wizualnie nagłówki, stopkę, menu, artykuły itd., inaczej reagować na zdarzenia typu kliknięcie itp.
 - Bez stylowania nie różnią się niczym od np. elementu `<p>` itp.

Przykład LogicalDivision.html

```
<title>Logical division</title>
</head>
<body>
  <header>
    <nav>
      <ul>
        <a href="SimpleForm.html">Simple</a>
        <a href="ComplexForm.html">Complex</a>
      </ul>
    </nav>
  </header>
  <main>
    <h1>Logical division - main</h1>
    <article>
      <header>
        The most important article
      </header>
      Lorem ipsum dolor sit amet consectetur adipisicing elit.
    </article>
  </main>
  <footer>
    &copy; Dariusz Konieczny
  </footer>
</body>
</html>
```

[Simple](#) [Complex](#)

Logical division - main

The most important article

Lorem ipsum dolor sit amet consectetur adipisicing elit. Facere, doloremque laudantium officiis, adipisci molestiae, expedita nesciunt officia nobis. Arc. Quasi odit, rerum suscipit ipsum pariatur quibusdam repellat quaerat nisi n. aliquid recusandae rerum quasi vel rem quibusdam nobis. Ut mollitia dolor nam sint odio, possimus repellat a ducimus, accusamus iure odit quas error quidem impedit dicta ab sequi, atque modi repudiandae eveniet, veniam eri Obcaecati eum saepe ullam voluptate quam libero, nisi incidunt facere, vita
© Dariusz Konieczny

Inne nowe elementy

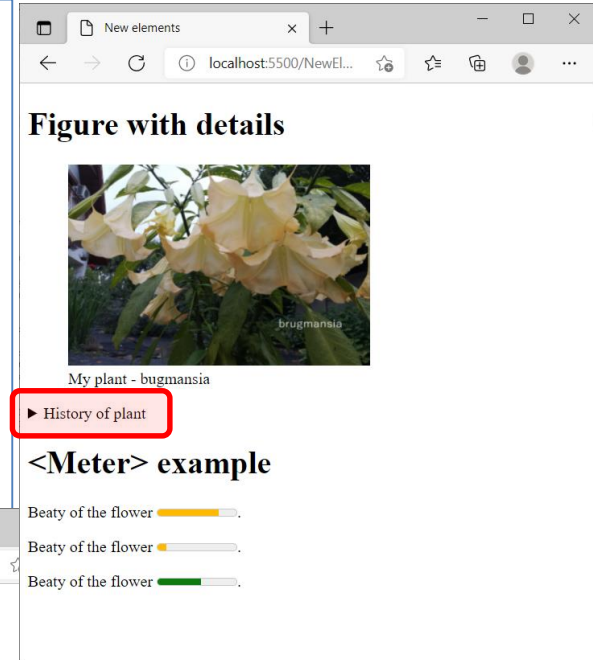
- Element `<figure>` - element najczęściej otaczający `` ale też dowolnie skonstruowaną ilustrację. Pozwala dodać tytuł/podpis do tej ilustracji
- Element `<figcaption>` - najczęściej wewnątrz elementu `<figure>`, tytuł ilustracji.
 - Standardowo powyżej ilustracji.
- Element `<details>` - Tworzy rozwijalny element ze szczegółowym opisem. Oznaczony najczęściej jako trójkąt skierowany w prawo. Po kliknięciu trójkąt skierowany jest w dół i pokazuje się treść zawartą w elemencie `<details>`
- Element `<summary>` - wewnątrz elementu `<details>` Stanowi etykietę tego elementu, czyli pojawia się po prawej stronie od trójkąta.

Użycie powyższych elementów ułatwia dostęp do nich osobom niepełnosprawnym oraz pozwala dodać opis z większą liczbą szczegółów.

- Element `<meter>` - element graficzny w rodzaju paska postępu.

Przykład NewElements.html

```
<title>New elements</title>
</head>
<body>
  <h1>Figure with details</h1>
  <figure>
    
    <figcaption>My plant - brugmansia</figcaption>
  </figure>
  <details>
    <summary>History of plant</summary>
    It was bought in "Biedronka" shop. The plant was rickety, but it was well
    fertilized and watered. When the flower buds appeared, it received
    a special fertilizer (for daturas) and it bloomed beautifully
    as you can see in the photo.
  </details>
  <h1>&lt;Meter&gt; example</h1>
  <p>Beaty of the flower <meter min="10" max="100" low="30" high="75" value="80">Very good</meter>.</p>
  <p>Beaty of the flower <meter min="10" max="100" low="30" high="75" value="20">Very good</meter>.</p>
  <p>Beaty of the flower <meter min="10" max="100" low="30" high="75" value="60">Very good</meter>.</p>
</body>
</html>
```



Czego nie było na wykładzie, a istnieje w HTML5

- Element `<picture>`, `<video>`, `<audio>` - do wpisania wielu źródeł mediów wybieranych na podstawie charakterystyki urządzenia lub przeglądarki (mniejsza rozdzielczość na komórce lub ze słabą szybkością połączenia)
- Element `<svg>` - tworzenie rysunków jako zbioru figur, krzywych itp.
- Do pisania czcionką o stałej szerokości lub bez interpretacji kodów HTML `<pre>`, `<code>`, `<kbd>`, `<var>`, `<samp>`
- Element `<template>` - element ukryty, który można wykorzystać w przyszłości
 - Z pomocą kodu Javascriptu
- Element pusty `<wbr>` - (ang. word break) miejsca łamania słowa (dla długich słów np. technicznych określeń)
- I wiele innych elementów
 - Wiele zostało też wycofanych w HTML5: `<blink>`, `<center>`, `<frame>` itp.
- Wiele atrybutów
 - Ogólne dla dużej części elementów (wysokość, szerokość, kolor tła itp.)
 - Specyficzne dla konkretnych elementów
- Można stworzyć własny znacznik w JavaScriptcie
 - Kod definicji nowego elementu:
`customElements.define('popup-info', PopUpInfo);`
 - Od tego momentu `<popup-info ...>` to nowy znacznik
 - `PopUpInfo` to jakaś nasza klasa w JavaScript z funkcjami obsługi dla tego znacznika
- Elementy formularza teoretycznie mogą być poza formularzem
 - Wtedy jednak na pewno trzeba mieć kod w JavaScriptcie z reakcją na zdarzenia pojawiające się w danym elemencie

HTML

DODATEK

Skróty i akronimy

- Element `<abbr>` z atrybutem `title` pojawiającym się po najechaniu kursorem myszki
- Element `<acronym>` - działanie podobne, z HTML4.

Inne elementy HTML

- Element `<address>` - często pisany kursywą
- Element `<s>` - oznacza informacje przestarzałą, pisany przekreśloną czcionką.

Element `<iframe>`

- Element `<iframe>` służy do osadzania innej strony wewnątrz tworzonej strony
 - Np. mapy Google
- Istotne atrybuty o standardowym znaczeniu: `src`, `width` (domyślnie 300), `height` (domyślnie 150)