



# pasja-informatyki.pl

## Programowanie webowe

Mirosław Zelent

### Odcinek #3

ten, w którym dokonujemy przeglądu  
elementarnych znaczników języka HTML

# Spis treści

Prolog .....	3
Zapis znaczników w HTML.....	3
Hipertącze <a></a> .....	4
Obraz <img> .....	5
Paragraf <p></p> .....	7
Nagłówki <h1></h1> - <h6></h6> .....	8
Listy <ul></ul>, <ol></ol>.....	9
Tabele <table></table> .....	11
Element <span></span>.....	14
Formularz <form></form> .....	15
Pole tekstowe <input type="text"> .....	16
Etykieta <label></label> .....	16
Pole hasła <input type="password"> .....	17
Pole numeryczne <input type="number"> .....	17
Checkbox <input type="checkbox">.....	18
Pola radio <input type="radio"> .....	19
Lista wyboru <select></select> .....	19
Pole wieloliniowe <textarea></textarea> .....	21
Przycisk <input type="submit"> .....	22
Przycisk <input type="button"> .....	22
Przycisk <button></button> .....	23
Znacznik <strong></strong> .....	23
Emfaza <em></em> .....	25
Przypisy <small></small> .....	26
Indeks górny i dolny <sup></sup>, <sub></sub>.....	26
Blok cytatu <blockquote></blockquote> .....	27
Cytowanie inline <q></q> .....	28
Linia pozioma <hr> .....	28
Zmiana postaci tekstu <b></b>, <i></i>, <u></u>.....	29
Zadanie do samodzielnego wykonania .....	30

# Prolog

HTML to język opisowy służący do tworzenia dokumentów hipertekstowych. Poznajmy zatem podstawowe znaczniki tworzące ciało witryny – zapraszam Was do przeglądu popularnych tagów oraz ich atrybutów!

Kluczowe jest, aby na egzaminie wykazać się automatyzmem, który wypracowaliśmy za sprawą zrealizowanych projektów. Na końcowym etapie przygotowań powinniśmy być w stanie uporać się z całym front-endem jaki będzie do wykonania, w czasie maksymalnie 15-stu minut. Przeczytawszy treść zadania miejmy już przed oczami wyobraźni wszystkie potrzebne znaczniki oraz ułożony wstępny plan działania. A jeżeli mamy do wykonania skrypt JS lub PHP, to od razu miejmy w głowie projekt gotowego formularza. Dobra znajomość znaczników HTML uwolni twój czas, pozwalając Ci przeznaczyć go na zajęcie się właściwą, back-endową częścią zadania (mechaniką jak i zapleczem bazodanowym).

Do dzieła! Rozpocznijmy od teorii, zaś w drugiej części wykładu przejdziemy do zadań praktycznych. Jest niezwykle ważne, aby te zadania samodzielnie wykonać! I to nawet jeżeli już po samym wyjaśnieniu, wydaje Ci się że wiesz jak używać danego znacznika, wykonaj przykład praktyczny - co innego teoria, a co innego doświadczenie - myślę, że doskonale zdajemy sobie z tego sprawę ucząc się jeździć samochodem, uprawiając sport, czy grając w jakąkolwiek grę komputerową – zapewniam, tak samo jest w programowaniu - to praktyka czyni mistrza. Powodzenia!

## Zapis znaczników w HTML

Przypomnijmy ogólną konwencję – znaczniki zapisujemy w nawiasach ostrych, mogą one także posiadać wewnątrz tych nawiasów tzw. atrybuty z zapisanymi w cudzysłowie wartościami.

Znacznik pojedynczy:

```
<znacznik atrybut="wartosc">
```

Znacznik podwójny (w sensie: złożony z dwóch tagów – otwierającego i zamykającego).

```
<znacznik atrybut="wartosc"> </znacznik>
```

Jeżeli znacznik jest podwójny, to jego ewentualne atrybuty umieszczamy w tagu otwierającym. Wartość atrybutu powinna zostać zapisana w cudzysłowie (nawet jeśli jest typu liczbowego), a nie w apostrofach.

## Hipertącze <a></a>

Dlaczego znacznik definiujący link na stronie to właśnie <a>? Prawdopodobnie wzięło się to od angielskiego słowa *anchor*, które oznacza *kotwicę* (hipertącza są – jak to mówimy – *zakotwiczone w dokumencie*). Element <a> jest złożony z dwóch znaczników – posiada zatem tag otwierający oraz zamykający: <a></a>. Jest to logiczne, gdyż musimy zdefiniować obszar, który stanowi rzeczywistą kotwicę linku, namacalne obiekty możliwe do kliknięcia. W linku poniżej podlinkowany jest tekst Pasja informatyki:

```
<a href="http://pasja-informatyki.pl">Pasja informatyki</a>
```

### [Pasja informatyki](http://pasja-informatyki.pl)

Atrybut **href** to skrót od ang. *hypertext reference* – określamy tutaj adres dokumentu HTML, do którego hipertącze ma prowadzić. *Reference* oznacza z ang. *odniesienie* – i rzeczywiście czasami tak określamy linki – mówimy, że to *odnośniki* do innych dokumentów. Co ważne – atrybut href nie jest wcale wymagany. Mogą istnieć znaczniki <a>, bez podania adresu linka – np. w menu głównym strony (w górnej belce witryny). Dochodzimy tutaj do wniosku, iż tak naprawdę *hipertączem* możemy nazwać jedynie element <a>, który posiada określony wartością atrybut href – sam element <a> odnośnika jeszcze nie ustanawia.

Hipertącze może także posiadać atrybut **target** (ang. *cel*), który określa gdzie docelowo w przeglądarce ma trafić podlinkowany dokument:

```
<a href="http://pasja-informatyki.pl" target="_blank">Pasja informatyki</a>
```

Możliwe wartości atrybutu **target**:

- **target="\_self"** – otwórz stronę w tej samej karcie/ramce, w której znajduje się link (ponieważ jest to zachowanie domyślne, można ten atrybut pominąć);

- **target="\_blank"** – otwórz witrynę w nowej, nieużywanej karcie przeglądarki (uwaga: nie należy nadużywać tego mechanizmu! Kieruj się empatią wobec internauty i otwieraj nowe karty tylko tam, gdzie rzeczywiście wydaje się to pożądane – w przeciwnym wypadku gość odwiedzający naszą stronę zirytuje się i natychmiast ją opuści);
- **target="\_parent", target="\_top"**, – otworzy adres hiperłącza w odpowiedniej ramce – jest to związane z tzw. framesetem (ang. *zestaw ramek*). Wartość `_parent` otworzy witrynę w ramce o jeden poziom wyżej we framesetowej hierarchii, a `_top` w nadrzędnej ramce). Jednak budowanie witryny na ramkach to relikw przeszłości – są one fatalne zwłaszcza w kontekście SEO, jak również niewygodne dla samego internauty. W praktyce więc raczej nie zdarzy Ci się używać tych wartości atrybutu `target`.

## Obraz <img>

Znacznik `img`, z ang. *image* – obrazek. Znacznik `img` jest pojedynczy, a nie podwójny, gdyż obraz jest obiektem – nie trzeba określać gdzie się zaczyna, a gdzie kończy, ponieważ o tym decyduje rozmiar źródłowej grafiki lub ewentualnie określone przez nas właściwości CSS.

```

```

Jak wspominaliśmy w ostatnim odcinku, w HTML5 w pojedynczych znacznikach nie wstawia się już w zamknięciu kończącego znaku `/`

```

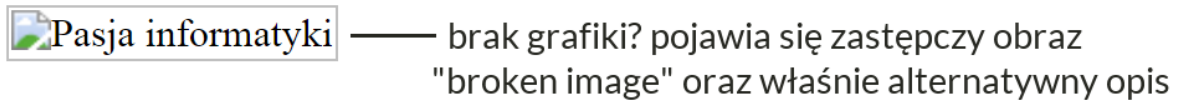
```

choć też jego obecność błędem krytycznym nie będzie – w bardzo wielu witrynach wciąż spotkamy domknięcie tagu pojedynczego znane z XHTML, czyli zawierające znak `/`.

Atrybut `src` – z ang. *source* – ścieżka dostępu do *źródłowej* grafiki. Atrybut `src` jest wymagany (ang. *required*), co wydaje się logiczne, gdyż bez źródłowego pliku graficznego ciężko mówić o istnieniu obrazu w dokumencie.

Atrybut `alt` – z ang. *alternative* – alternatywny, tekstowy opis, który dobrze oddaje czym dana grafika jest, w sensie: co na niej się znajduje. Jest to również atrybut wymagany. Dlaczego atrybut `alt` jest według standardu konieczny? Otóż jeśli grafika załaduje się nam prawidłowo, to w zasadzie można uznać taki opis za zbędny i tylko niepotrzebnie zajmujący

miejsce w kodzie. Natomiast pamiętajmy – na naszą stronę trafią także użytkownicy, którzy wyłączyli w swojej przeglądarce ładowanie obrazów, bądź są osobami słabowidzącymi, korzystającymi z czytników witryn. Wówczas taki alternatywny opis, zapewniający słowną reprezentację zawartości obrazka, pomaga im bardziej komfortowo naszą witrynę przeglądać!



Czasami spotkamy się także z zapisem pustej wartości atrybutu alt:

```
  

```



na tej stronie logo składa się z 2 plików graficznych, stąd alternatywny opis wystarczy zamieścić jeden raz

W tym przypadku nasze logo składa się z dwóch plików – wówczas tylko raz wpisujemy alternem informacje że mamy tu do czynienia z logiem. Dlaczego tylko raz? Cóż, byłoby to irytujące dla kogoś, kto np. tylko słucha naszej witryny czy ogląda ją w terminalu, żeby dwa razy słyszeć bądź zobaczyć to samo. Trzeba się po prostu (z empatią) przyjrzeć kontekstowi, w jakim dana grafika występuje. Jeśli z kontekstu wynika, że ten obraz jest naprawdę mało istotny, a alternatywny opis tylko dublował by to, co już znajduje się jako tekst w kodzie, to warto pozostawić wartość atrybutu alt pustą.

W znaczniku img możemy także zdefiniować atrybuty określające na sztywno rozmiar obrazu - *width* i *height*, czyli odpowiednio *szerokość* i *wysokość*:

```

```

Oczywiście biorąc pod uwagę, iż zazwyczaj interesuje nas rozdzielenie treści witryny od jej wyglądu, to lepiej byłoby takie wymiary (jeśli już musimy to zrobić) określić w arkuszu stylów CSS. Pamiętajmy też, iż podając szerokość i wysokość obrazu w atrybutach HTML wewnątrz tagu, nie piszemy już (jak to ma miejsce w CSS) końcówki *px* – jak widać w wartości podajemy jedynie liczbową wartość.

## Paragraf <p></p>

Akapit (paragraf, ustęp) tekstu, z ang. *paragraph*. Znacznik <p> w prosty i skuteczny sposób pozwala stworzyć pożądany podział dłuższego tekstu na łatwiejsze do strawienia akapity:

```
<p>Ósmy Światowy Kongres Futurologiczny odbył się w Costaricanie. Prawdę mówiąc, nie pojechałbym, do Nounas, gdyby nie profesor Tarantoga, który dał mi do zrozumienia, że tego się po mnie oczekuje. Powiedział też (co mnie dotknęło), że astronautyka jest dziś formą ucieczki od spraw ziemskich.</p>
```

```
<p>Każdy, kto ma ich dość, wyrusza w Galaktykę, licząc na to, że najgorsze stanie się pod jego nieobecność. Prawdą jest, że nieraz, zwłaszcza w dawniejszych podróżach, wracałem z lękiem, wypatrując przez okno Ziemi – czy nie przypomina upieczonego kartofla. Toteż zbytnio się nie opierałem, a jedynie zauważyłem, że się na futurologii nie znam.</p>
```

Ludzie stawiający swoje pierwsze kroki w HTML, bardzo często zamiast stosować akapity tekstu, wstawiają co kilka zdań podwójne znaczniki <br><br>:

```
Ósmy Światowy Kongres Futurologiczny odbył się w Costaricanie. Prawdę mówiąc, nie pojechałbym, do Nounas, gdyby nie profesor Tarantoga, który dał mi do zrozumienia, że tego się po mnie oczekuje. Powiedział też (co mnie dotknęło), że astronautyka jest dziś formą ucieczki od spraw ziemskich.<br><br>
```

```
Każdy, kto ma ich dość, wyrusza w Galaktykę, licząc na to, że najgorsze stanie się pod jego nieobecność. Prawdą jest, że nieraz, zwłaszcza w dawniejszych podróżach, wracałem z lękiem, wypatrując przez okno Ziemi – czy nie przypomina upieczonego kartofla. Toteż zbytnio się nie opierałem, a jedynie zauważyłem, że się na futurologii nie znam.
```

Tag <br> od ang. *break* – *złamanie* linii – to w HTML odpowiednik naciśnięcia klawisza Enter na klawiaturze. Zaś dwa złamanie linii, jeden po drugim, siłą rzeczy stworzą przerwę w tekście, która zachowa się podobnie jak akapit. Jednakże znacznik <p> jest dużo lepszy do tego celu, bo łatwiej stylizuje się dzięki niemu wewnętrzny tekst akapitu w CSS, z użyciem chociażby klasy (na wzór: <p class="nazwaklaszy"></p>).

Znaczników <br> wygodnie wystylizować nie sposób, gdyż stanowią po prostu złamanie linii. Paragraf to już co innego – możemy powiedzieć: *odtąd dotąd tekst ma wyglądać następująco*, ustawiając justowanie, krój, rozmiar i kolor czcionki w CSS. W kontekście SEO również

paragrafy spisują się lepiej niż <br><br>, gdyż lepiej oddają robotowi budowę wityny. Znacznika <br> można używać, ale wewnątrz paragrafu, żeby na przykład złamać linię gdy jest to rzeczywiście potrzebne. I rzeczywiście czasami jest - np. publikujemy poezję - tam złamanie linii kończy wers wiersza:

```
<p>wiatr drzewa spienia. Ziemia dojrzała.<br>
kłosa brzuch ciężki w górę unoszą<br>
i tylko chmury - palcom czy włosom<br>
podobne - suną drapieźnie w mrok.<br>
Ziemia owoców pełna po brzegi<br>
kipi sytością jak wielka miska.<br>
Tylko ze świerków na polu zwisa<br>
głowa obcięta strasząc jak krzyk.<br>
Kwiaty to krople miodu - tryskają<br>
ściśnięte ziemią, co tak nabrzmiała,<br>
pod tym jak korzeń skręcone ciała,<br>
żywcem włócznie pod ciemny strop.<br>
Ogromne nieba suną z warkotem.<br>
Ludzie w snach ciężkich jak w klatkach<br>
krzyczą.</p>
```

Zatem reasumując: jeżeli dwa fragmenty tekstu tworzą osobne wątki myślowe czy informacyjne, no to umieśćmy je w osobnych paragrafach, dla wygody czytającego. Pełne docenienie paragrafów przyjdzie z czasem, kiedy będziesz więcej uwagi poświęcać semantyce kodu, jego poprawności i czystości. Dzięki doświadczeniu docenisz tę autentyczną łatwość ostylowania tekstu akapitu i naturalnie przestawisz się na tworzenie tagów p zamiast <br><br>. Miej dla siebie cierpliwość.

## Nagłówki <h1></h1> - <h6></h6>

Sekcje w dokumencie, podobnie jak w gazecie, rozpoczynają się od nagłówków (z ang. *headings*). Przy czym od razu ustanowiono sześć rodzajów nagłówków (które różnią się rozmiarem) i dlatego mamy znaczniki od <h1> do <h6>:

```
<h1>Nagłówek rozmiar pierwszego</h1>
<h2>Nagłówek rozmiar drugiego</h2>
<h3>Nagłówek rozmiar trzeciego</h3>
<h4>Nagłówek rozmiar czwartego</h4>
<h5>Nagłówek rozmiar piątego</h5>
<h6>Nagłówek rozmiar szóstego</h6>
```



Domyślnie największą czcionkę ma nagłówek h1, po czym rozmiar czcionki zmniejsza się sukcesywnie aż do najmniejszego h6:

# Nagłówek rozmiar pierwszego

## Nagłówek rozmiar drugiego

### Nagłówek rozmiar trzeciego

#### Nagłówek rozmiar czwartego

##### Nagłówek rozmiar piątego

###### Nagłówek rozmiar szóstego

Nagłówki ładnie definiują sekcję artykułów – jesteśmy do tego przyzwyczajeni również z Worda, z gazet, z książek - stąd naturalnie mechanizm ten przenieśliśmy także do Internetu. Co do zmian wyglądu nagłówek, to wszystko oczywiście odbywa się w CSS, a nie z użyciem atrybutów HTML. W kontekście SEO szczególnie ważny jest tekst zamknięty w nagłówku `<h1></h1>`.

## Listy `<ul></ul>`, `<ol></ol>`

Lista nieuporządkowana (nienumerowana) – z ang. *unordered list* – jest definiowana przez znacznik `<ul></ul>`, zaś pojedynczy element takiej listy zamknięty jest w znacznikach `<li></li>`, od ang. *list item*:

```
<ul>
  <li>Królestwa Północy</li>
  <li>Scoia'tael</li>
  <li>Cesarstwo Nilfgaardu</li>
  <li>Skellige</li>
</ul>
```

- Królestwa Północy
- Scoia'tael
- Cesarstwo Nilfgaardu
- Skellige

**Lista uporządkowana (numerowana)** – z ang. *ordered list* – jest definiowana przez znacznik `<ol></ol>`, zaś pojedynczy element takiej listy również zamknięty jest w znacznikach `<li></li>`:

```
<ol>
  <li>Królestwa Północy</li>
  <li>Scoia'tael</li>
  <li>Cesarstwo Nilfgaardu</li>
  <li>Skellige</li>
</ol>
```

1. Królestwa Północy
2. Scoia'tael
3. Cesarstwo Nilfgaardu
4. Skellige

Oczywiście w niektórych przypadkach znajdzie potrzeba tzw. **zagnieżdżenia listy**, czyli stworzenia listy wewnętrznej wewnątrz znaczników `<li></li>` listy zewnętrznej. W poniższym przykładzie lista zewnętrzna to lista numerowana, a wewnętrzna to lista nienumerowana (choć równie dobrze można by zagnieżdżyć je odwrotnie):

```
<ol>
  <li>Królestwa Północy
    <ul>
      <li>Sigismund Dijkstra</li>
      <li>Keira Metz</li>
    </ul>
  </li>
  <li>Scoia'tael
    <ul>
      <li>Iorveth</li>
      <li>Isengrim Faol'tiarna</li>
    </ul>
  </li>
</ol>
```

1. Królestwa Północy
  - Sigismund Dijkstra
  - Keira Metz
2. Scoia'tael
  - Iorveth
  - Isengrim Faol'tiarna

Listy to ważne znaczniki HTML w kontekście egzaminu zawodowego - często pojawiają się tak w części pisemnej jak i w zadaniach praktycznych, jak również są ważne w pracy. Często na odpowiednio wystylizowanych listach buduje się menu główne witryny - pomimo, iż takie górne menu wygląda jak pozioma belka, to kiedy zajrzemy do jej trzewi, okazuje się że od strony mechaniki wszystko zbudowane jest na kanwie właśnie listy zagnieżdżonej.

## Tabele `<table></table>`

Tabele (z ang. *table*) nadają się fantastycznie do czytelnego, dwuwymiarowego przedstawiania danych. Cały obszar tabeli ograniczony jest - rzecz jasna - znacznikami `<table></table>`. Pojedynczy wiersz tabeli definiowany jest tagami `<tr></tr>` (od ang. *table row*), zaś wewnątrz wierszy definiować będziemy komórki tabeli `<td></td>` (ang. *table drawer*). I tak przypatrzmy się tabeli zawierającej 3 wiersze, a w każdym z nich po 3 komórki:

```
<style> td { border: 1px solid black; } </style>
<!-- ustawienie czarnego obramowania tabeli w CSS -->

<table>
  <tr>
    <td>1</td> <td>2</td> <td>3</td>
  </tr>
  <tr>
    <td>3</td> <td>4</td> <td>5</td>
  </tr>
  <tr>
    <td>6</td> <td>7</td> <td>8</td>
  </tr>
</table>
```

1	2	3
4	5	6
7	8	9

Kolejna ważna umiejętność w kontekście budowania tabel to scalanie komórek. Scalanie, czyli łączenie dwóch lub więcej komórek w jedną. Czasami trzeba wykonać takie złączenie w tabeli, chociażby dla przejrzystości pokazywania danych. Służą do tego dwa atrybuty - *rowspan*, jeśli scalaniu ulegają wiersze oraz *colspan*, jeżeli scalaniu ulegają kolumny. Zobaczmy to jednak na konkretnych przykładach:

Scalanie z użyciem atrybutu *colspan*:

```
<style> td { border: 1px solid black; } </style>
<!-- ustawienie czarnego obramowania tabeli w CSS -->

<table>
  <tr>
    <td colspan="2">1</td> <td>3</td>
  </tr>
  <tr>
    <td>3</td> <td>4</td> <td>5</td>
  </tr>
  <tr>
    <td>6</td> <td>7</td> <td>8</td>
  </tr>
</table>
```

1	3	
4	5	6
7	8	9

Scalanie z użyciem atrybutu *rowspan*:

```
<style> td { border: 1px solid black; } </style>
<!-- ustawienie czarnego obramowania tabeli w CSS -->

<table>
  <tr>
    <td>1</td> <td>2</td> <td>3</td>
  </tr>
  <tr>
    <td rowspan="2">4</td> <td>5</td> <td>6</td>
  </tr>
  <tr>
    <td>8</td> <td>9</td>
  </tr>
</table>
```

1	2	3
4	5	6
	7	8

Możemy także rozbudować tabelę o znaczniki `thead` (ang. *table head*), oraz `tbody` (ang. *table body*). Tak samo jak dokument HTML posiada sekcję `<head>` i `<body>`, tak tabela posiada swoją głowę i ciało – jest to ta sama tradycja, wywodząca się z budowy funkcji w programowaniu. Oczywiście ktoś zapyta - a po co w ogóle określać ten podział na głowę i ciało tabeli? No, idea jest taka, że w tabelach często mamy do czynienia ze słownym opisem, co znajduje się w danych kolumnach – i do tego celu można wykorzystać głowę `<thead>`. Do tworzenia takich „opisowych komórek nagłówkowych” stosujemy znacznik `<th>` (ang. *table header*):

```
<style> td, th { border: 1px solid black; } </style>
<!-- ustawienie czarnego obramowania tabeli w CSS -->

<table>
  <thead>
    <tr>
      <th>Przedmiot</th> <th>Nazwisko</th> <th>Ocena</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>Historia</th> <td>Nowak</td> <td>4+</td>
    </tr>
    <tr>
      <th>Historia</th> <td>Mazur</td> <td>3-</td>
    </tr>
    <tr>
      <th>Fizyka</th> <td>Nowak</td> <td>2</td>
    </tr>
    <tr>
      <th>Fizyka</th> <td>Mazur</td> <td>4</td>
    </tr>
  </tbody>
</table>
```

Przedmiot	Nazwisko	Ocena
Historia	Nowak	4+
Historia	Mazur	3-
Fizyka	Nowak	2
Fizyka	Mazur	4

W uproszczeniu można powiedzieć, że takie komórki `<th>`, zastępują nam klasyczne szufladki `<td>` wszędzie tam, gdzie zamiast wartości komórki, mamy jej opis – logiczne. W praktyce znaczniki `<th>` mogą pojawić się nie tylko w głowie tabeli `<thead>`, ale także w sekcji `<tbody>`, jako że opisy czasem znajdują się także w innym wierszu niż pierwszy – podział na głowę i ciało wykazuje więc charakter umowny.

I ostatnia, ważna sprawa przy tabelach - otóż dozwolone jest, w ściśle określonych przypadkach, pomijanie zamykających tagów. Można je pominąć w HTML5, co znacząco może odchudzić ilość kodu potrzebną do zapisania wielkich tabel:

```
<style> td, th { border: 1px solid black; } </style>
<!-- ustawienie czarnego obramowania tabeli w CSS -->

<table>
  <thead>
    <tr> <th> Przedmiot <th> Nazwisko <th> Ocena
  <tbody>
    <tr> <th> Historia <td> Nowak <td> 4+
    <tr> <th> Historia <td> Mazur <td> 3-
    <tr> <th> Fizyka <td> Nowak <td> 2
    <tr> <th> Fizyka <td> Mazur <td> 4
</table>
```

Przedmiot	Nazwisko	Ocena
Historia	Nowak	4+
Historia	Mazur	3-
Fizyka	Nowak	2
Fizyka	Mazur	4

Oczywiście tabele to super złożony temat, istnieje tutaj jeszcze wiele innych możliwości, ale to co podaliśmy to jest taki rozsądny zestaw, na pierwsze *rendez-vous* z tabelami. Osoby ciekawskie odsyłam do oficjalnej dokumentacji, tam można się wgryźć i poznać pełne oblicze tabel: [link](#).

## Element `<span></span>`

Pojemnik, na pierwszy rzut oka podobny do `<div>`, jednak w odróżnieniu od niego, element `<span>` nie zachowuje się blokowo (nie spowoduje zmian w strukturze blokowej witryny, niczego nie przesunie). Słowo *span* ma w języku angielski wiele znaczeń, jednak w tym kontekście przetłumaczymy je jako: *zasięg*, *rozpiętość*. Najczęściej element `<span>` tworzymy po to, aby zdefiniować zasięg działania jakiejś klasy w CSS. Sam znacznik nie posiada żadnej roli, poza właśnie prostym zdefiniowaniem iż *odtąd dotąd* obowiązuje dany styl. Dlatego też atrybutami znacznika `<span>` będą najczęściej *class*, albo ewentualnie *style*. Dobrym praktycznym przykładem zastosowania `<span>`, jest chęć zapisania jednego wyrazu innym kolorem czcionki, np. czerwonym:

```
<style> .cz { color:red; } </style>
<!-- ustawienie czerwonego koloru czcionki dla klasy .cz -->
Polowanie na <span class="cz">Czerwony</span> Październik
```

## Polowanie na Czerwony Październik

Takie użycie `<span>` zastąpiło nieużywane już dzisiaj tagi `<font>` z atrybutem `color`. Tagów `<font>` nie używamy, gdyż interesuje nas rozdzielenie wyglądu strony od jej zawartości. Czasami, w niektórych kodach źródłowych, zobaczysz elementy `<span>`, które zachowują się blokowo, tzn. tak jak `div`. Jest to możliwe, jeśli w klasie `spana` wymusimy takie zachowanie dopisując: `display: block`; Pełna dokumentacja: [tutaj](#).

## Formularz `<form></form>`

Formularze służą użytkownikom do wprowadzania danych wejściowych do witryny, celem ich przetworzenia (np. formularz rejestracji, który wprowadza nowego użytkownika do bazy, albo formularz logowania, który wprowadza login i hasło, których poprawność następnie sprawdzamy w bazie danych). Cały obszar formularza zamykamy w znacznikach `<form>`, zaś pomiędzy tymi tagami, układamy sobie, niczym z klocków lego, dowolny zestaw kontrolki formularza. A te bywają różne - mamy pola tekstowe, przyciski, checkboxy, listy wyboru etc.

```
<form action="login.php" method="post">
<!-- Tutaj wybrane kontrolki formularza -->
</form>
```

Atrybut **action** określa który plik zajmie się przetwarzaniem danych otrzymanych z formularza. Jako wartość podajemy ścieżkę do przygotowanego przez nas skryptu PHP (w przykładzie jest to plik `login.php`).

Atrybut **method** określa metodę przetwarzania formularza – będzie to metoda `get` (wartość domyślna) albo `post`. Dziś jednak, w przeglądzie znaczników HTML nie opowiemy jeszcze o różnicach pomiędzy tymi metodami – jeżeli już teraz masz ochotę zrozumieć różnice między nimi, to zapraszam do kursu PHP [tutaj](#). Jeszcze więcej atrybutów znajdziesz w [dokumentacji](#).

## Pole tekstowe <input type="text">

Wiele znaczników, które służą nam do wprowadzania danych, określimy przy pomocy tagu <input> (ang. *wejście*). Rodzaj elementu o jaki nam konkretnie chodzi, ustawiamy atrybutem type (typ definiowanej przez nas kontrolki). Klasyczne pole tekstowe, jakie znamy z Windows, czy ze stron internetowych wstawimy do formularza wartością text atrybutu type:

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

Uwaga: bardzo ważne w przypadku wielu kontrolek formularzy (w tym pola tekstowego) jest poprawne użycie tagu <label>, o czym przekonamy się poniżej.

## Etykieta <label></label>

Na początku przygody z tworzeniem dokumentów wszyscy zapisujemy opis kontrolki formularza jako zwykły tekst (ewentualnie dodając gdzieś po drodze złamanie linii <br>):

```
Podaj login: <input type="text">
```

Natomiast tak jak lepiej jest używać paragrafów, zamiast luźno porzrzuconych zdań rozdzielonych podwójnymi <br><br>, tak lepiej jest używać znaczników <label> do opisania pola, zamiast luźno wrzuconego tekstu:

```
<label>Podaj login: <input type="text"></label>
```

Zauważmy: wewnątrz znacznika <label> znajduje się zarówno opis pola, jak i sam <input>. Alternatywnie (równie poprawnie) można także zapisać to tak:

```
<label for="pole">Podaj login:</label>  
<input type="text" id="pole">
```

W tej wersji etykieta zamknięta jest przed znacznikiem <input>, ale pojawia się dodatkowo atrybut for (ang. *dla*) określający dla jakiego elementu jest to etykieta.



Wartością atrybutu `for` jest identyfikator, podany także w atrybucie `id` wewnątrz `<input>`. Obydwa identyfikatory rzecz jasna koniecznie muszą być identyczne. To właśnie w ten sposób następuje logiczne powiązanie pomiędzy kontrolką formularza i jego etykietą. Ustawiona prawidłowo etykieta reaguje na kliknięcie – powiązana z nią kontrolka formularza zyskuje aktywność.

Atrybut **placeholder** (ang. *rezerwujący miejsce*) powoduje wyświetlenie wewnątrz pola wyszarzonej podpowiedzi, która podpowiada użytkownikowi jakiej informacji w tym polu oczekuje witryna:

```
<label for="pole">Podaj login:</label>  
<input type="text" id="pole" placeholder="nick lub e-mail">
```

Kiedy tylko zaczniemy pisać w polu, to podpowiedź zniknie – jej miejsce zajmie wpisana przez nas wartość.

## Pole hasła `<input type="password">`

Pole tekstowe do wpisywania haseł – wartość atrybutu `type` to `password`. Ze względów bezpieczeństwa kontrolka ta automatycznie maskuje wprowadzane do niej znaki, tak aby osoba przyglądająca się monitorowi nie była w stanie zwyczajnie zobaczyć naszego hasła. Takie pole - rzecz jasna - najczęściej spotkamy w formularzach logowania czy rejestracji:

```
<label for="pass">Podaj hasło:</label>  
<input type="password" id="pass">
```

## Pole numeryczne `<input type="number">`

Jest to pole edycyjne, wprowadzone w HTML 5, które w zamyśle ma służyć do wprowadzania liczb, a w zasadzie napisów, które powinny się na liczbę bezproblemowo dać przekonwertować. Takie pole rzeczywiście pomaga użytkownikowi nie pomylić się przy wprowadzaniu liczb, jako iż nie pozwala w ogóle wprowadzić z klawiatury liter (poza literką `e`, której można używać w tzw. notacji naukowej). Oczywiście notacja naukowa najbardziej przydaje się do pokazywania naprawdę dużych lub naprawdę małych liczb, natomiast właśnie ze względu na istnienie tego zapisu, w polu numerycznym możemy wpisywać literkę `e`.

Przyjrzyjmy się przykładom zapisu numerycznego:

- $1.5e2$  – przecinek przesuwamy o 2 miejsca w prawo, gdyż po literze e znalazła się liczba dodatnia 2 – zatem jest to inaczej zapisana liczba 150.
- $1.5e-2$  – przecinek przesuwamy o 2 miejsca w lewo, gdyż po literze e znalazła się liczba ujemna -2 – jest to zatem inaczej zapisana liczba 0.015.

```
<label for="liczba">Podaj hasło:</label>  
<input type="number" id="liczba">
```

Możliwość wpisania litery e sprawia, iż użytkownik może wpisać do pola np. frazę: „eeee”. Wniosek jest prosty – łańcuch numeryczny zawsze wypada sprawdzić (zwalidować czy aby na pewno jest poprawną liczbą). Ponadto, jako element wprowadzony w HTML5, pole numeryczne może być niepoprawnie obsługiwane w przeglądarkach z rodziny IE (wówczas wyświetlone zostanie zwykłe pole `<input type="text">`).

## Checkbox `<input type="checkbox">`

Kolejna klasyczna kontrolka - checkbox, czyli małe kwadratowe pole, które można zaznaczyć (ang. *checkmark* – zaznaczenie, *box* – pudełko). Tego typu element formularza służy zazwyczaj do potwierdzania np. przeczytania regulaminu, włączenia opcji w ustawieniach konta etc.

```
<input type="checkbox" id="zazn">  
<label for="zazn">Akceptuję regulamin</label>
```

Szczególnie w przypadku checkboxa ważne jest prawidłowe użycie tagu `<label>`. Kliknięcie w prawidłowo ustawioną etykietę zmienia stan pola, zaś zwykły tekst nie spełnia tej roli.

Co do sprawdzenia wartości zaznaczenia, w sensie: czy ten checkbox był zaznaczony czy nie w momencie podsumowania formularza, to rzecz jasna dokonamy tego w pełnoprawnych językach programowania, czy to w JS czy w PHP.

Jeśli chcielibyśmy stworzyć checkboxa, który już na starcie byłby zaznaczony, to możemy mu dopisać atrybut *checked* (można to uczynić na co najmniej kilka sposobów, z różną wartością atrybutu, albo nawet bez ustawionej wartości):

```
<input type="checkbox" id="zazn" checked>  
<label for="zazn">Akceptuję regulamin</label>  
  
<input type="checkbox" id="zazn" checked="checked">  
<label for="zazn">Akceptuję regulamin</label>  
  
<input type="checkbox" id="zazn" checked="on">  
<label for="zazn">Akceptuję regulamin</label>
```

## Pola radio <input type="radio">

Pole typu radio - skąd taka nazwa? Otóż kiedy słuchamy radia, to możemy słuchać tylko jednej stacji. I tak samo w polach typu radio możemy wybrać tylko jedną opcję z możliwych:

```
<label>  
  <input type="radio" name="plec" value="m"> Mężczyzna  
</label>  
  
<label>  
  <input type="radio" name="plec" value="k"> Kobieta  
</label>
```

Aby automatyczne odznaczanie pozostałych opcji działało, każdego inputa typu radio wyposażamy w tę samą wartość atrybutu *name*. Zaś odróżnienie poszczególnych wartości w trakcie przetwarzania formularza, nastąpi za sprawą atrybutów *value*. Nie zapominajmy także, przynajmniej na dalszym etapie realizowania projektów, iż jako elementy formularza, również pola radio powinny zostać wyposażone w znaczniki <label>.

## Lista wyboru <select></select>

Do stworzenia listy wyboru użyjemy znacznika <select>. Nazwa łatwa do zapamiętania – słowo *select* oznacza oczywiście *wybór*. Zaś poszczególne opcje, możliwe do wyboru na liście, to znaczniki <option></option>. Ponownie – podobnie jak w polach radio, jakoś musimy rozróżnić później (na etapie przetwarzania formularza) poszczególne opcje, stąd możliwe jest ustawienie atrybutów *value* dla każdego możliwego. Nie zapominajmy o także o etykiecie dla całej listy <select></select>:

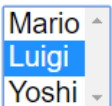
```
<label for="imie">wybierz postać:</label>
<select id="imie">
  <option value="m">Mario</option>
  <option value="l">Luigi</option>
  <option value="y">Yoshi</option>
</select>
```

Podobnie jak checkboxa można od razu zaznaczyć, tak na liście wyboru możemy od razu dokonać wyboru domyślnej opcji. Załóżmy, że chcemy, aby domyślnie wybrana była druga opcja na liście. Wówczas wystarczy do tegoż właśnie tagu option dopisać słowo **selected** – z ang. *wybrana*. Oczywiście jest ona wybrana tylko po wczytaniu strony – potem użytkownik może ten domyślny wybór, rzecz jasna, zmienić:

```
<label for="imie">wybierz postać:</label>
<select id="imie">
  <option value="m">Mario</option>
  <option value="l" selected>Luigi</option>
  <option value="y">Yoshi</option>
</select>
```

Ponadto, możemy w HTML uzyskać listę wielokrotnego wyboru – w tym celu należy do znacznika dopisać atrybut **multiple**. Dlaczego takie słowo? Z ang. *multiple* oznacza właśnie *wielokrotny [wybór]*. Listę taką niejako *pokazujemy w całości* właśnie po to, żeby móc z użyciem klawiszy Ctrl, albo Shift, zaznaczyć na niej kilka opcji jednocześnie.

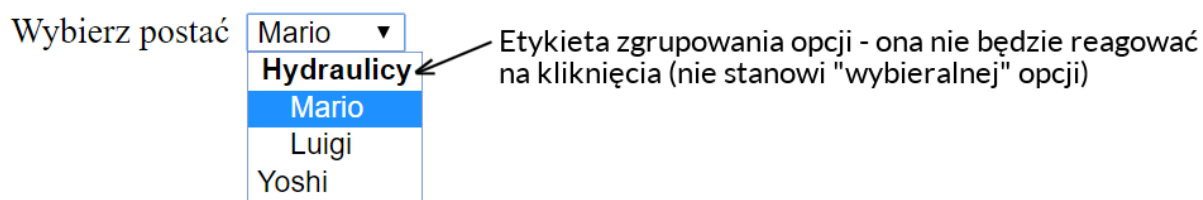
```
<label for="imie">wybierz postać:</label>
<select id="imie" multiple>
  <option value="m">Mario</option>
  <option value="l">Luigi</option>
  <option value="y">Yoshi</option>
</select>
```

Wybierz postać  **Możliwe jest wielokrotne zaznaczenie z użyciem klawiszy Shift albo Ctrl**

Taka lista wielokrotnego wyboru to element dużo rzadziej stosowany aniżeli jej klasyczny odpowiednik, niemniej jednak warto wiedzieć o jej istnieniu. I ostatnia sprawa przy liście wyboru – poszczególne opcje można dla wygody użytkownika zgrupować w sekcje, z użyciem znaczników `<optgroup></optgroup>` (nazwa mówi sama za siebie):

```
<label for="imie">wybierz postać:</label>
<select id="imie">
  <optgroup label="Hydraulicy">
    <option value="m">Mario</option>
    <option value="l">Luigi</option>
  </optgroup>
  <option value="y">Yoshi</option>
</select>
```

Pamiętajmy o atrybucie **label** (tym razem to atrybut, a nie znacznik!) dla takiej grupy, gdyż ta sama etykieta pojawia się później na liście (oczywiście jest *niewybieralna*, ale dobrze oddaje wybrane przez nas kryterium zgrupowania):



## Pole wieloliniowe <textarea></textarea>

Wieloliniowe pole zdefiniujemy w HTML znacznikami <textarea></textarea>. Słowo *area* oznacza *przestrzeń [do wpisywania tekstu]*. Zauważmy, że poznane przez nas wcześniej pola edycyjne pozwalały na wprowadzenie dokładnie jednej linii tekstu. Tylko jednej! A czasami chcemy napisać komentarz (na przykład pod filmem na YouTube), czy wysłać prywatną wiadomość, posta na forum, e-maila. Takie sytuacje wymagają wpisania już kilkuliniowego tekstu i po to właśnie mamy znacznik <textarea>:

```
<label for="komentarz">Skomentuj:</label>
<textarea id="komentarz"></textarea>
```

Jeśli chodzi o atrybuty pola wieloliniowego – możemy ustalić ile znaków ma takie pole mieścić w pojedynczej linii oraz ile linii tekstu dajemy komuś do dyspozycji (bez pojawienia się pionowego paska przewijania – ang. *scrollbar*).

Rozmiar pola wieloliniowego można w prosty sposób zdefiniować dwoma atrybutami: **rows**, czyli ilość wierszy, oraz **cols** – ilość kolumn (w praktyce minimalna ilość znaków w jednej linii).

W przykładzie poniżej zdefiniowaliśmy 5 wierszy, po około 44 znaki:

```
<label for="komentarz">Skomentuj:</label>
<textarea id="komentarz" rows="5" cols="44"></textarea>
```

## Przycisk <input type="submit">

Przycisk zazwyczaj służy w formularzu do tzw. *submita*, czyli podsumowania całego formularza dane zostają wysyłane na serwer wybraną metodą (get lub post).

```
<input type="submit" value="Kliknij">
```

Napis na przycisku określamy jak widać atrybutem **value** (ang. *wartość*). I teraz bardzo ważny wniosek: kliknięcie przycisku, który jest submitem, spowoduje próbę wysłania formularza na serwer, w przeciwieństwie do drugiego typu przycisku <input type="button">, który opisano w sekcji poniżej.

## Przycisk <input type="button">

Znacznik <input type="button"> definiuje „zwykły” przycisk – w sensie: jego kliknięcie nie *submituje* całego formularza (po kliknięciu nie zostaje podjęta próba wysłania danych formularza wybraną metodą na serwer):

```
<input type="button" value="Kliknij">
```

Czasami potrzebujemy takiego *niesubmitującego* zachowania, gdyż chcemy aby kliknięcie przycisku uruchomiło nam lokalny skrypt JS po stronie front-endu, a nie powodowało podjęcia próby komunikacji z serwerem, z PHP po stronie back-endowej.

To jest właśnie istota różnicy pomiędzy type="submit" i type="button" – ten pierwszy podsumowuje formularz, ten drugi nie robi tego.

## Przycisk <button></button>

Zamiast pojedynczego znacznika <input>, możemy zamiennie zapisać podwójny znacznik <button></button>. W tej wersji napis na przycisku umieszczamy pomiędzy tagami:

```
<button>kliknij</button>
```

Dlaczego dano nam możliwość zapisywania oprócz powyższych dwóch inputów, także znacznika button? Odpowiedź tkwi właśnie w fakcie, iż znacznik button jest podwójny - tzn. ma tag otwierający i zamykający, w przeciwieństwie do pojedynczego <input>. A to sprawia, że pomiędzy tagami <button></button>, można np. umieścić obrazek <img>, czyli niejako uatrakcyjnić wizualnie ten element:

```
<button>Wyślij</button>
```

Jeżeli chodzi o zachowanie elementu wewnątrz formularza, to kliknięcie elementu <button></button>, podobnie jak <input type="submit"> spowoduje podsumowanie formularza (zatem jedynie <input type="button"> wyróżnia się nie podejmując w ogóle próby kontaktu z serwerem).

Oczywiście na początku swoboda wyboru z aż trzech dostępnych opcji podczas tworzenia przycisku sieje mocny zamęt w głowach adeptów HTML, ale spokojnie – ty już znasz różnice pomiędzy nimi. I na tym zakończyliśmy omawianie podstawowych kontroltek, które spotkamy w formularzach.

## Znacznik <strong></strong>

Znacznik o twardo brzmiącej nazwie *strong* (ang. *silny, mocny, ostry*) podkreśla siłę danego tekstu powodując **wytłuszczenie** jego czcionki. Pytanie jakie rodzi nam się w głowach brzmi: *Kiedy powinno się używać takich znaczników?* Specyfikacja W3C podaje 3 takie okoliczności: tekst zamykamy w <strong></strong> jeżeli:

- jest z jakiegoś powodu ważny,
- przestrzega przed czymś,

- gdy warto, aby użytkownik omiatając tekst wzrokiem, najpierw przeczytał właśnie wyróżnione fragmenty (ułatwią one zrozumienie fabuły czytanego artykułu):

**Dokonujemy wyboru w boot menu - najczęściej uruchamiamy je na ekranie startowym komputera klawiszem F12.** Można też ustawić w BIOSie odpowiedni boot order. Instalator zapyta nas, czy chcemy wypróbować system czy też zainstalować go.

**Podczas mojej instalacji okazało się, że napęd DVD nie potrafił odczytać niektórych plików z płyty, co było związane z brakiem odpowiednich sterowników dla napędu.** Rozwiązałem ten problem naciskając podczas instalacji dwukrotnie klawisz F6 i dodając dodatkowy wpis o treści all\_generic\_ide ładujący inny zestaw sterowników.

**Uwaga! każdy ewentualny błąd podczas instalacji można naprawić, bo instalator zawsze zostawia nam informację co poszło nie tak.** Zatem wystarczy wygooglować ten komunikat lub ewentualnie zapytać na forum linuxowym. Dość często błędy spowodowane są pobraniem niekompletnego pliku ISO, co może się zawsze zdarzyć, gdy pobieramy duży plik przez przeglądarkę internetową.

**Dokonujemy wyboru w boot menu - najczęściej uruchamiamy je na ekranie startowym komputera klawiszem F12.** Można też ustawić w BIOSie odpowiedni boot order. Instalator zapyta nas, czy chcemy wypróbować system czy też zainstalować go.

**Podczas mojej instalacji okazało się, że napęd DVD nie potrafił odczytać niektórych plików z płyty, co było związane z brakiem odpowiednich sterowników dla napędu.** Rozwiązałem ten problem naciskając podczas instalacji dwukrotnie klawisz F6 i dodając dodatkowy wpis o treści all\_generic\_ide ładujący inny zestaw sterowników.

**Uwaga! każdy ewentualny błąd podczas instalacji można naprawić, bo instalator zawsze zostawia nam informację co poszło nie tak.** Zatem wystarczy wygooglować ten komunikat lub ewentualnie zapytać na forum linuxowym. Dość często błędy spowodowane są pobraniem niekompletnego pliku ISO, co może się zawsze zdarzyć, gdy pobieramy duży plik przez przeglądarkę internetową.

Fragmenty wytłuszczone oddają ogólną myśl w artykule, zaś te bez <strong> rozwijają ją szczegółowo. Ponadto warto rozpatrywać znacznik <strong> w kontekście SEO – robot Google zauważy, iż określone wyrazy (w odróżnieniu od pozostałych w paragrafie czy nagłówku), są uznawane za ważniejsze w danej podstronie. W trzecim akapicie podkreślona została także ważna uwaga kierowana do czytelnika.



## Emfaza `<em></em>`

Innego rodzaju wyróżnieniem tekstu są znaczniki `<em></em>`, definiujące tzw. emfazę (ang. *emphasis* – *nacisk w wymowie, emfaza*) Kiedy kładziemy na jakieś słowa emfazę wypowiadając się, to znaczy to, iż celowo wypowiadamy te słowa ze zmienioną intonacją, gestykulacją, czy natężeniem głosu - często zmieniając sens całego zdania. Od strony technicznej zaś, tekst zamknięty w znacznikach `<em></em>` zostanie zapisany pochyloną czcionką. Jako przykład przedstawmy zdanie:

Dzisiaj padał deszcz.

Jeżeli emfazę położymy na pierwszy wyraz:

`<em>Dzisiaj</em> padał deszcz.`

*Dzisiaj* padał deszcz.

to kluczowy nacisk w zdaniu sprawia, iż zmienia się jego sens: padało nie wczoraj, nie tydzień temu, tylko dziś - emfaza sprawiła, że właśnie ta informacja stała się najważniejsza. Jeśli zaś zapiszemy:

Dzisiaj `<em>padał</em>` deszcz.

Dzisiaj *padał* deszcz.

Teraz w zdaniu przekazujemy informację: już nie pada. I żeby móc oddać taki zabieg emfazy w czytanim tekście, stworzono znacznik `<em>`.

Taki nacisk w wymowie na poszczególne słowa, to jednak co innego aniżeli wcześniej poznany przez nas znacznik `<strong>` - tu nie chodzi o ważność, pilność, czy funkcję ostrzegawczą słów - nie, tu chodzi o podkreślenie nacisku położonego w wymowie na konkretne słowa.

## Przypisy

Kolejne znaczniki  (z ang. *small* – mały). W znacznikach tych powinniśmy zamieścić drobne przypisy, komentarze, podpowiedzi w formularzach – wszystko to, co ma drugorzędne, kularowe znaczenie. Od strony technicznej, zamknięcie tekstu w tych znacznikach rzeczywiście zapisze tekst zmniejszoną czcionką, tak żeby oddać marginalny charakter zapisanych słów.

```
<p>Jak ważne jest zmniejszenie rozmiarów grafik wykorzystywanych na stronie www nie trzeba myśleć nikomu tłumaczyć. Strona ładuje się szybciej przez co sprawia wrażenie "lżejszej" - najbardziej widać to na urządzeniach mobilnych lub wolnych łączach internetowych, z których każdemu czasem zdarza się korzystać gdzieś w terenie czy na wyjeździe. (Osobiście przepuszczam przez ten konwerter online całe strony internetowe. Zdarzyło mi się raz zmniejszyć łączną wagę plików graficznych całego serwisu o 68% sic!).</p>
```

Jak ważne jest zmniejszenie rozmiarów grafik wykorzystywanych na stronie www nie trzeba myśleć nikomu tłumaczyć. Strona ładuje się szybciej przez co sprawia wrażenie "lżejszej" - najbardziej widać to na urządzeniach mobilnych lub wolnych łączach internetowych, z których każdemu czasem zdarza się korzystać gdzieś w terenie czy na wyjeździe. (Osobiście przepuszczam przez ten konwerter online całe strony internetowe. Zdarzyło mi się raz zmniejszyć łączną wagę plików graficznych całego serwisu o 68% sic!).

Znamy takie przypisy – czasami mówi się, że to czy tamto jest w dokumencie zapisane *drobnym druczkiem*. Natomiast niekoniecznie trzeba to kojarzyć pejoratywnie, takie kularowe zapisy bywają dla odbiorcy naszej witryny użyteczne.

## Indeks górny i dolny ,

Kolejne dwie modyfikacje tekstu - indeks górny  oraz indeks dolny  (ang. *superscript* oraz *subscript*). Czasami musimy zapisać w dokumencie np. jednostkę m<sup>2</sup> – ta dwójka powinna zostać zapisana w indeksie górnym. Lub odwrotnie – mamy równanie chemiczne H<sub>2</sub>O - i tutaj dwójkę zapiszemy używając indeksu dolnego.

100 m<sup>2</sup>

H<sub>2</sub>O

100 m<sup>2</sup> H<sub>2</sub>O

Żeby raz na zawsze zapamiętać, który znacznik definiuje indeks dolny, a który górny, to pomyśl o piosence Beatlesów pt. *Yellow submarine* (ang. *Żółta łódź podwodna*). I stąd wiemy, że `<sub>` oznacza dolny indeks, a `<sup>`, siłą rzeczy musi definiować indeks górny.

## Blok cytatu `<blockquote></blockquote>`

Jak zapisać cytat w tekście? Możemy do tego celu użyć znacznika `<blockquote></blockquote>`. Tworzy on w tekście blok cytatu (ang. *quote* – *cytat*), który powoduje dodanie charakterystycznego wcięcia z lewej strony, a którego szczegółowy wygląd można potem ustalić w CSS. Znacznik `<blockquote>` może posiadać atrybut `cite`, którego wartością jest link, z którego cytujemy. Natomiast niestety jako atrybut jest on widoczny jedynie w kodzie strony, stąd użytkownik musiałby samodzielnie zbadać źródło, aby go ujrzeć. Zatem lepiej jest zrealizować to tak, aby oprócz atrybutu `cite`, przygotować użytkownika link do kliknięcia:

```
<blockquote cite="https://pl.wikipedia.org/wiki/Jacek_Piekara">
  <p>Nigdy nie zwyciężysz wroga, którego twoja wyobraźnia
    uczyniła niepokonanym.</p>
  <footer>
    - <a href="https://pl.wikipedia.org/wiki/Jacek_Piekara">
      Jacek Piekara - łowcy Dusz</a>
  </footer>
</blockquote>
```

Nigdy nie zwyciężysz wroga, którego twoja wyobraźnia uczyniła niepokonanym.

- [Jacek Piekara - łowcy Dusz](https://pl.wikipedia.org/wiki/Jacek_Piekara)

Ostatecznie nasz cytat zamykamy dla porządku w paragraf `<p></p>`, zaś po nim dodajemy znacznik blokowy HTML5, o nazwie `<footer>` - to będzie stopka dla naszego bloku z cytatem (ang. *footer* – *stopka*). W stopce umieszczamy hiperłącze do tej samej strony, którą podaliśmy w atrybucie `cite`. Rezultat jest na pewno wygodniejszy dla użytkownika (jedno kliknięcie dzieli go od poznania źródła cytatu).

## Cytowanie inline <q></q>

W tekście paragrafów można używać również tzw. *cytowania inline*, czyli takiego, które nie wymaga wrywania cytatu do osobnego bloku <blockquote></blockquote>, tylko oznacza cytat w ramach tekstu akapitu. Używamy do tego celu krótkiego znacznika q (ang. *quote*):

```
<p>Podszedł do telefonu, energicznie złapał słuchawkę i wykrzyknął: <q>To ja się właśnie pytam co; że się z tym tak brandzlujesz!</q>. Następnie rzucił słuchawką i szybko opuścił sekretariat.</p>
```

Podszedł do telefonu, energicznie złapał słuchawkę i wykrzyknął: „To ja się właśnie pytam co; że się z tym tak brandzlujesz!”. Następnie rzucił słuchawką i szybko opuścił sekretariat.

## Linia pozioma <hr>

Za pomocą znacznika <hr> wstawiamy linię poziomą, która ma w założeniach stanowić delikatną przerwę tematyczną, oddzielać od siebie różne mini-sekcje zgromadzone w dokumencie. Nazwa <hr> wzięta się zapewne od ang. *horizontal* – *poziomy*. Oczywiście wygląd tej poziomej linii można w razie potrzeby wygodnie ustalić w CSS

```
<p>Noc była czarna jak wewnątrz kota. Można by uwierzyć, że właśnie w taką noc bogowie przesuwają ludzi niczym pionki na szachownicy losu. Pośród tej burzy żywiołów ogień migotał pod ociekającymi krzewami kolcolistu jak obłąd w oczach łasicy.</p>
```

```
<hr>
```

```
<p>Oświeślał trzy przygarbione postacie. Kiedy zabułgotał kociótek, ktoś zajęczał przeraźliwie: <q>Rychłóż się zejdzim znów?</q> Zapadło milczenie. Aż w końcu ktoś inny odpowiedział tonem o wiele bardziej zwyczajnym: <q>Myślę, że dam radę w przyszły wtorek.</q></p>
```

Noc była czarna jak wewnątrz kota. Można by uwierzyć, że właśnie w taką noc bogowie przesuwają ludzi niczym pionki na szachownicy losu. Pośród tej burzy żywiołów ogień migotał pod ociekającymi krzewami kolcolistu jak obłąd w oczach łasicy.

Oświeślał trzy przygarbione postacie. Kiedy zabułgotał kociótek, ktoś zajęczał przeraźliwie: „Rychłóż się zejdzim znów?” Zapadło milczenie. Aż w końcu ktoś inny odpowiedział tonem o wiele bardziej zwyczajnym: „Myślę, że dam radę w przyszły wtorek.”

## Zmiana postaci tekstu <b></b>, <i></i>, <u></u>

Znaczniki podwójne, zmieniające postać tekstu: wytłuszczenie czcionki <b>, czcionka pochylona (ang. *italic*) <i>, oraz podkreślenie (ang. *underline*) <u>.

```
w studiu użyliśmy mikrofonu <b>Shure SM7B</b>.
Spojrzał w jej oczy i doświadczył mocnego uczucia <i>déjà vu</i>.
Kiedy karmisz Gizmo, upewnij się, że <u>nie minęła północ</u>.
```

W studiu użyliśmy mikrofonu **Shure SM7B**.

Spojrzał w jej oczy i doświadczył mocnego uczucia *déjà vu*.

Kiedy karmisz Gizmo, upewnij się, że nie minęła północ.

Tutaj naturalnie rodzi się nam w głowie pytanie: czym zatem różni się znacznik <b> od <strong>, oraz tag <i> od <em>? Których znaczników lepiej jest używać?

Zasada jest prosta: tagu <b> użyj tylko w przypadku, kiedy zastosowanie <strong> nie wydaje się właściwe w danym kontekście, a tekst mimo wszystko powinien być wytłuszczony. Czyli nie jest to tekst ważny, czy ostrzegawczy, a jedynie chcemy wyłuszczyć jakieś słowo kluczowe w tekście, nazwę produktu o którym jest mowa, czy istotnego w mowie czasownika.

Zaś jeśli chodzi o znacznik <i> zamiast emfazy <em>, to użyj <i></i> we wszystkich przypadkach, kiedy nie kładziemy nacisku w wymowie na zadane słowa – na przykład kiedy posługujemy się językowym idiomem.

A co do znacznika <u></u>, to zachęcam używać go na jak najmniej. Oczywiście warto czasami podkreślić jakieś słowo, ale lepiej zrobić to z użyciem <strong>, gdyż podkreślenie często sugeruje internaucie obecność nałożonego na dane słowo hiperłącza. I to jest mylące. Dlatego ze względu na specyfikę hipertekstu (tekst czytany w Internecie), wszędzie tam gdzie można zamiast podkreślenia <u> (zależnie od kontekstu) stosuj <strong>, <em>, <b> albo <i>. Podkreślenia za bardzo kojarzą się nam podświadomie z linkami.

## Zadanie do samodzielnego wykonania

Wykonaj interfejs strony internetowej polskiego klanu graczy CS:GO o nazwie *Non omnis moriar* (aktualnie jeszcze silverów), którzy chcą rozwijać pasję wspólnego rozgrywania meczy online, pragną uruchomić w sieci klanową witrynę ułatwiającą rekrutację, jak i przegląd aktualnego składu drużyny głównej i rezerwowej. Pierwsza, robocza wersja serwisu ma składać się z dwóch podstron: *index.html* (strona główna) oraz *register.html* (podstrona zawierająca formularz rekrutacyjny).

### Wymagania dotyczące witryny

1. Korzystając z wiedzy z odcinka 1 i 2 kursu, zrealizuj od zera następującą strukturę blokową serwisu (z użyciem *display: inline-block;* bądź *float:left;*) tak dla strony głównej, jak i podstrony z formularzem rejestracji. Interfejs ma zajmować 100% dostępnej w przeglądarce przestrzeni (20% nawigacja, 50% główny content, 30% tabela graczy).

[ logo serwisu ]		
[ nawigacja ]	[ główny content ]	[ tabela graczy ]
[ stopka ]		

Wymagania dotyczące obu plików (*index.html*, *register.html*):

2. Logiem serwisu ma być obraz związany z CS:GO (którego klan niestety nie dostarczył – trzeba będzie coś samodzielnie opracować), o wymiarach 800x150 px. Obraz powinien posiadać ustawiony alternatywny opis o treści: *klan non omnis moriar* przeznaczony dla osób słabowidzących (korzystających z czytników witryn), bądź odwiedzających serwis z wyłączonym pokazywaniem grafik. Pod linkami, w nowym paragrafie `<p></p>` ma znaleźć się następująca informacja:

**Uwaga! Trwa rekrutacja! Wymagana ranga: silver.**

Słowa te uznawane są za ważne informacje klanowe i będą restrykcyjnie przestrzegane.

3. Nawigacja (lewa szpalta witryny) ma zawierać trzy hiperłącza umieszczone wewnątrz listy nienumerowanej:
- do pliku `index.html`, kotwicą linku jest napis: *Strona główna*
  - do pliku `register.html`, kotwicą linku jest napis: *Dołącz do nas*
  - do serwisu `Twitch.tv` (adres: <http://bit.ly/streamy-csgo>), kotwicą linku jest napis: *Streamy z CS:GO*. Link powinien otwierać się w nowej karcie przeglądarki.
4. Obszar prawej szpalty serwisu ma zawierać aktualną tabelę graczy (w przyszłości dane będą wyjmowane z bazy, aktualnie jest to tabela 5 wierszy na 3 kolumny). Wygląd tabeli dostarczony przez klan jest następujący:

#### Aktualni gracze Non Omnis Moriar

Nick	Płeć	Ulubiona broń
Sn1p3RR	m	AWP
Gh0st256	k	PP-Bizon
K4w411	k	MP9
S0l1d\$nake	m	Desert Eagle

Legenda: szef klanu, sekretarz drużyny

Uwaga: pierwszy wiersz nie zawiera wartości, tylko słowny opis zawartości kolumn. Czcionka wiersza zawierającego dane gracza `Sn1p3RR` ma kolor czerwony, gdyż jest on założycielem (szefem) całego klanu, zaś czcionka wiersza zawierającego dane gracza `Gh0st256` ma kolor zielony, gdyż jest on sekretarzem drużyny. Pod tabelą ma znaleźć się legenda wyjaśniająca te barwy – ma zostać zapisana małą czcionką w znacznikach `<small></small>`. Poszczególne kolory zaś, mają zostać uzyskane przez zastosowanie znaczników `<span></span>`.

5. Stopka witryny ma zawierać następujący cytat:

Na wojnie się wygrywa albo przegrywa, żyje albo umiera – a decyduje się to w *mgnieniu oka*.  
- [Douglas MacArthur](#)

Słowa *Douglas MacArthur* są podlinkowane do następującego adresu (źródła cytatu):  
[https://pl.wikiquote.org/wiki/Douglas\\_MacArthur](https://pl.wikiquote.org/wiki/Douglas_MacArthur)

Słowa *w mgnieniu oka* zostały podkreślone emfazą, gdyż mają charakterystyczną wymowę w kontekście gry FPS wymagającej nad wyraz rozwiniętego refleksu.

Wymagania dotyczące jedynie strony głównej (*index.html*):

6. Obszar contentu strony głównej rozpoczyna się od nagłówka h1 o treści:

## **Witaj w witrynie klanu CS:GO Non Omnis Moriar!**

po czym następują dwa akapity cytowanego tekstu:

Counter-Strike: Global Offensive (CS:GO) – wieloosobowa gra komputerowa z gatunku first-person shooter, stworzona oraz wydana przez studia Valve Corporation i Hidden Path Entertainment, które już wcześniej pracowały nad Counter-Strike: Source. Jest to najnowsza część serii Counter-Strike.

Global Offensive wydano 21 sierpnia 2012 roku na platformę Steam. Gra jest dostępna na systemy: Microsoft Windows, Linux, Mac OS X, Xbox 360 (poprzez Xbox Live Arcade) oraz PlayStation 3 (za pośrednictwem PlayStation Network).

- źródło: [Wikipedia](#)

Link ma prowadzić pod adres:

[https://pl.wikipedia.org/wiki/Counter-Strike:\\_Global\\_Offensive](https://pl.wikipedia.org/wiki/Counter-Strike:_Global_Offensive)

Wymagania dotyczące jedynie podstrony z formularzem rejestracji (*register.html*):

7. Obszar contentu ma zawierać formularz prowadzący do pliku new.php (skrypt dodający nowego klanowicza – póki co jedynie zakładamy, że taki plik powstanie (interesuje nas jedynie ustawienie przekierowania do takiego pliku przy submitowaniu formularza). Metodą przesyłania danych formularza będzie metoda *post*.
8. Formularz ma zawierać następujące kontrolki (opatrzone odpowiednimi etykietami):
  - Tekstowe pole edycyjne do wprowadzenia nicka, z ustawioną placeholderem podpowiedzią o treści: *istniejący nick z CS*
  - Dwa pola edycyjne do wprowadzenia i powtórzenia hasła
  - Numeryczne pole edycyjne do wprowadzenia roku urodzenia
  - Dwa pola radio do wyboru płci (domyślnie zaznaczone jest pole *Mężczyzna*)
  - Checkbox *Akceptuję regulamin serwisu* (domyślnie zaznaczone)
  - Checkbox *Wyrażam zgodę na otrzymywanie newslettera* (domyślnie odznaczone)
  - Lista jednokrotnego wyboru *Ulubiona broń* z następującymi opcjami
    - Five-SeveN
    - Desert Eagle
    - PP-Bizon
    - AWP
    - MP9
    - AK-47



- Lista umożliwiająca wielokrotny wybór *Ulubione mapy*
  - Dust II
  - Mirage
  - Cache
  - Inferno
  - Nuke
  - Train
- Wielolinowe pole tekstowe *Kilka słów o sobie*, które dodatkowo opisane jest komentarzem zamkniętym w tagach `<small></small>` o treści: *Opisz swoją przygodę z CS:GO, dlaczego wybrałeś akurat nasz klan?*
- Przycisk submitujący formularz *Zarejestruj się!*

Oczywiście w zadaniu stworzymy jedynie *front-endowy interfejs formularza* (aby przećwiczyć w praktyce poznane w niniejszym przeglądzie znaczniki HTML) – zrealizowanie mechaniki przetwarzania formularza, jak i bazodanowego zaplecza to już zadanie na inne epizody serii. Powodzenia!