HTML by BC

Spis treści

Wprowadzenie

Markup language to język komputerowy który definiuje wygląd i strukturę czystego tekstu. W html'u komputer może interpretować czysty tekst, który jest zapisany w strukturze html. *hyperText* to tekst wyświetlający się na komputerze w postaci linków(hiperłącza).

URL – uniform resource locator

Tag/znacznik(<>), wraz z atrybutem(src="") nazywany jest elementem.

Tagi

Pełna lista dostępnych tagów w HTML https://developer.mozilla.org/en-US/docs/Web/HTML/Element

- ** ** tag(znacznik), w środku taga jest jego zawartość.
- **** zawiera z reguły krótkie fragmenty tekstu. Zwykle używany żeby oddzielić większy tekst i edytować poszczególne jego fragmenty.
- podkreśla tekst
- pogrubia
-
 łamanie linii tekstu wyświetlanego w przeglądarce
- kontener na blok kodu, dzieli poszczególne elementy w kodzie
- nienumerowana lista

- numerowana lista
    - element listy
- **** (*self-closing tag*);
- **<video src="">***Video not supported***</video>;**
- ****tutaj opisujemy link**** (anchor/kotwiczenie href – hyperlink reference);
- **<nav></nav>** - container do wrzucania przycisków nawigacji po stronie;

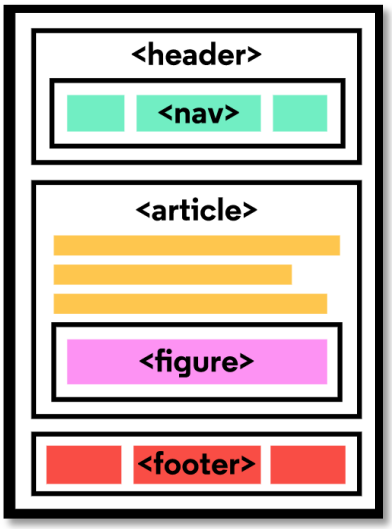
Hierarchia w HTML to inaczej zagnieżdżanie lub relacja rodzic-dziecko. Elementy dzieci mogą dziedziczyć cechy rodziców(klas nadrzędnych).

** ** *division*, kontener dzielący stronę na sekcje. (podział)

Atrybuty to dodatkowe informacje(czy zmianę stylu) zawarte w *opening tag*. Atrybuty zbudowane są z nazwy i wartości:

name="value"

Popularnym atrybutem jest id – opis na przykład div'a. nazwa ta nie wyświetla się w przeglądarce. Ułatwia natomiast późniejszą edycje jej w ***css*** lub odwoływanie się do całego tego diva.



- *<div **id**="menu"> *
- ** src tez jest atrybutem
- *Alt="" – alternative text; zamiast obrazka potrzebny, jak się nie załaduje. Pomaga tez w wyszukiwaniu, bo wyszukiwarka „widzi" opis, nie obrazek.*
- *Width="230"*
- *Height="235"*
- *Controls* – bez wartości atrybutu, dodaje tylko do video pauze i start
- *Href* – link do strony/pliku
- *Href="./index.html" relative path* – linkowanie do pliku w tym samym folderze co nasz główny plik(./ patrz w ten folder)
- *Target="_blank"* – link otwiera się w nowej karcie

Podsumowanie

- HTML to HyperText Markup Language i jest używana do postawienia struktury i zawartości strony(tekst,obrazki czy video);
- Element *html* jest zbudowany z tagu <>, w srodku tagu natomiast może zostać opisany atrybutami;

Struktura dokumentu HTML

<!DOCTYPE html> - deklaracja typu dokumentu dla przeglądarki, ten poniżej będzie html oraz jaka wersja(html5).

<html> *tutaj tworzymy całą stronę interpretowaną jako kod html* **</html>**

W *tagu html* zamieszczamy następnie:

- <head> informacje niewyświetlane w przeglądarce(dla programisty) *METADATA of webpage;*
    - *<title> - tytul strony wyświetlany u góry przeglądarki, na pasku;*
- <body> informacje widoczne w przeglądarce(dla klienta);

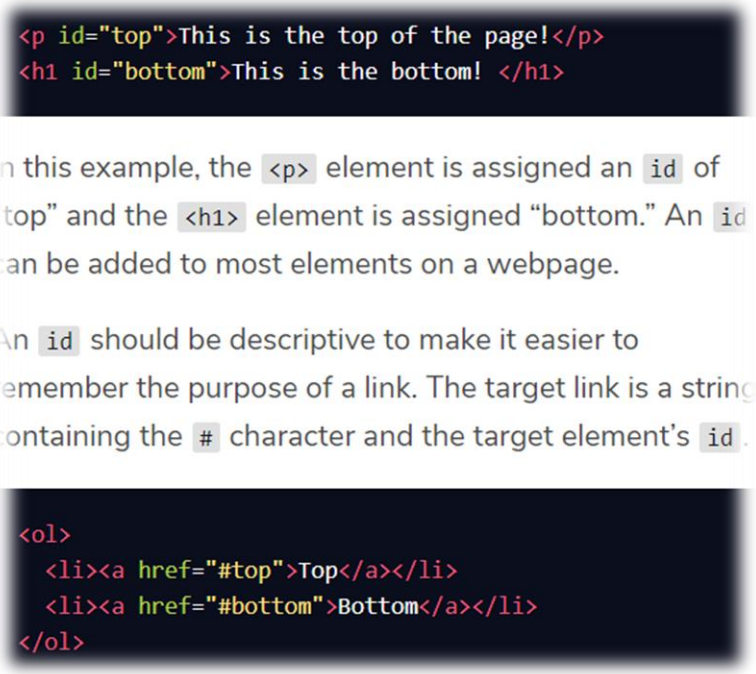
Obrazek jako link

```
<a href="https://en.wikipedia.org/wiki/Opuntia"
target="_blank"><img src="#" alt="A red prickly pear
fruit"/></a>
```

Składnia ***do zewnętrznej strony***:

Składnia ***do przejścia do innej części tej samej strony***:

Najpierw trzeba przydzielić jakiemuś elementowi na stronie **atrybut id**. Następnie odnieść się do niego jako **href=”#nazwa_id”.**



Whitespace & indentation

Poprawne pisane kodu. Nie wrzucanie do jednej linii wielu tagów po sobie. Zwiększanie czytelności kodu dla programisty.

Whitespace, to używanie spacji w edytorze, aby był on lepiej czytelny.

Indentation, to wcinanie zagnieżdżonych elementów kodu, również mające na celu poprawę czytelności kodu.

Dwie spacje są zalecane przez W3C przy zagnieżdżaniu.

Komentowanie w edytorze

<!—*tutaj komentarz* -->

Tabele

Tworzenie

<table>

<thead> tworzy głowę tabeli, tytuły

<th></th>

<th></th>

<th></th>

</thead>

<tbody> - w ciele tablicy powinny zawarte być wszelkie dane tabeli z ***wyjątkiem nagłówków/table heading(th!)***

<tr> - szkielet tabeli

<th></th> - *table heading*/tytuł; pusty *th* tworzy pustą komórkę w tabeli;

<td>*tekst***</td>** - dodawanie danych w tabeli(tekstu) *table data;*

</tr> - *table row*/ tworzenie rzędu

<tfoot>

<tr>

<td></td>

</tr>

</tfoot> - *footer*/ostatni rząd tabeli, często jako podsumowania/sumy. Warto w nim zamknąć ostatni <tr>

</tbody>

</table>

- Atrybut *scope* w table heading może być *col* lub *row*. Dajemy znać czy jest dla kolumny czy rzędu;
- Tworzenie obramowań tablic i graficzna obróbka już w CSS;
- ***Atrybut colspan*** – zwiększa rozpiętość danej **w kolumnie** o 1 lub więcej kolumn/rzędów (<td colspan=”2”>tekst</td>; tutaj zabierze 2 kolumny;
- Atrybut rowspan – zwiększa rozpiętość danej w rzędzie

Form (formularze)

Tworzenie formularza:

<form action="/example.html" method="POST">

</form>

W formularzu zamieszczamy poniższe tagi w celu utworzenia go w 100%.

Atrybut:

- ***Action*** - mówi gdzie zostanie wysłany formularz;
- ***Method*** – styl żądania np. post, czy get;

W formularzach można oczywiście dodawać elementy dzieci, jak h1 czy p.

Tagi w formularzu:

- <**input type**=”text”></input> - z typów może być też np. number. (*Input field/pole wejściowe)’; typ określa jak input zachowuje się i renderuje oraz jaki typ danych akceptuje;*
  - Atrybut **name** też jest wymagany;
  - **Value** to jest to co wpisujemy w pole *input/ można tez wpisać i to będzie się wyświetlało póki użytkownik nie zacznie pisać swojego tekstu;*
  - **Po wysłaniu *name* łączy się z *value*!** *(„name=input value”);*
  - **Input jest jedno-tagowy!**
- **<label></label>** - *etykieta*, do poprawnej identyfikacji *input field*, przeglądarka wyświetla tekst umieszczony między tagami. Aby połączyć go z *input field*, należy nadać mu **atrybut id**, a w *labelu* dać **atrybut for** o identycznej nazwie;
- **Kiedy pole *label* zostanie zaznaczone kursorem, zostanie podświetlony *input field*;**

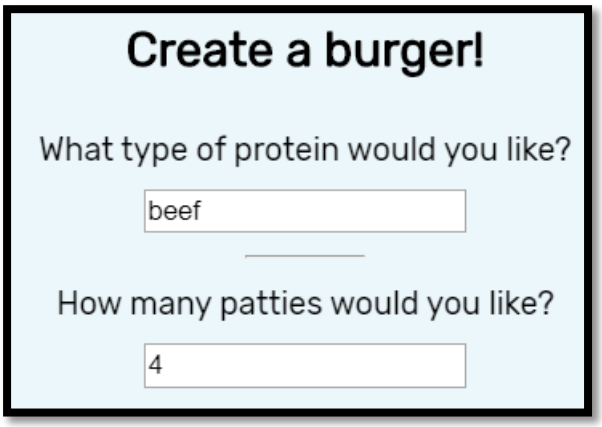
Password Input

- **Atrybut type=”password”** „kropkuje” lub „gwiazdkuje” wpisywany w *input* tekst;
- Atrybut **type=”number”** definiuje w polu możliwość wpisywania tylko numerów;
- Atrybut **step=”1”** umożliwia manualne wybieranie cyfry z pola input; tutaj o 1 skok;

```
      <form>
        <h1>Create a burger!</h1>
        <section class="protein">
          <label for="patty">What type of protein would
you like?</label>
          <input type="text" name="patty" id="patty"
value="beef">
        </section>
        <hr>

        <section class="patties">
          <label for="amount">How many patties would you
like?</label>
          <!--Add your code below-->
          <input id="amount" type="number" step="1"
 name="amount">

        </section>
      </form>
```



Utworzenie formularza, w nim nagłówka, następnie osobnej sekcji na proteiny. Potem utworzenie etykiety z pytaniem. Pod nim input field do odpowiedzi. Wymagany tekst, nie liczby.

Następnie druga etykieta z pytaniem oraz input field z wymaganą odpowiedzią w cyfrach.

Range input (suwak)

<form>

<label for="volume"> Volume Control</label>

<input id="volume" name="volume" **type="range" min="0" max="100" step="1">** *//ustawiamy suwak o granicach 0-100 gdzie mamy skok o 1.*

</form>

Checkbox input

Tworzenie checkbox’a:

<input type=”checkbox”>

Nadawanie *input fields* takich samych atrybutów *name* powoduje ze będą w jednej grupie rozpatrywane.

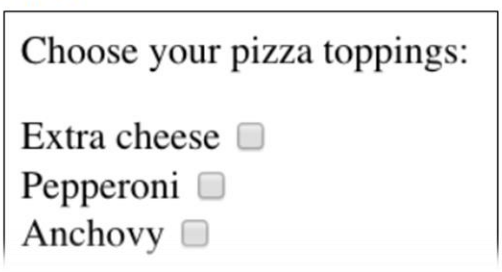
Mnogość wyborów, można wybrać wiele.

Radio Button input

Wybór tylko jednej odpowiedzi.

```
<form>
  <p>Choose your pizza toppings:</p>
  <label for="cheese">Extra cheese</label>
  <input id="cheese" name="topping" type="checkbox"
value="cheese">
  <br>
  <label for="pepperoni">Pepperoni</label>
  <input id="pepperoni" name="topping"
type="checkbox" value="pepperoni">
  <br>
  <label for="anchovy">Anchovy</label>
  <input id="anchovy" name="topping" type="checkbox"
value="anchovy">
</form>
```

Which renders:



Wymagany ten sam atrybut *name*.

```
<form>
  <p>What is sum of 1 + 1?</p>
  <input type="radio" id="two" name="answer"
value="2">
  <label for="two">2</label>
  <br>
  <input type="radio" id="eleven" name="answer"
value="11">
  <label for="eleven">11</label>
</form>
```

Dropdown List

Wybór z listy. Oszczędność miejsca jeśli jest dużo opcji wyborów.

```
<form>
  <label for="lunch">What's for lunch?</label>
  <select id="lunch" name="lunch">
    <option value="pizza">Pizza</option>
    <option value="curry">Curry</option>
    <option value="salad">Salad</option>
    <option value="ramen">Ramen</option>
    <option value="tacos">Tacos</option>
  </select>
</form>
```

Datalist input

Rozwijane menu z wyborem jednej opcji, można filtrować przy wpisywaniu znaków w pole input. W składni są **atrybut list** oraz **tag datalist**.

Pole tekstowe w *tagu option* zostawiamy puste!

```
<form>
  <label for="city">Ideal city to visit?</label>
  <input type="text" list="cities" id="city"
name="city">

  <datalist id="cities">
    <option value="New York City"></option>
    <option value="Tokyo"></option>
    <option value="Barcelona"></option>
    <option value="Mexico City"></option>
    <option value="Melbourne"></option>
    <option value="Other"></option>
  </datalist>
</form>
```

Textarea element

Tworzenie pola tekstowego o *tagu <textarea>.* Dodaje się również ilość kolumn i wierszy aby określić wielkość tegoż pola. Można umieścić w *textarea* tekst początkowy umieszczając go miedzy tagami.

```
<form>
  <label for="blog">New Blog Post: </label>
  <br>
  <textarea id="blog" name="blog" rows="5" cols="30">
  </textarea>
</form>
```

Submit form/button (wysyłanie formularza)

```
<form>
  <input type="submit" value="Send">
</form>
```

Podsumowanie formularza (form review)

https://www.codecademy.com/learn/learn-html/modules/learn-html-forms/reference

- The purpose of a <form> is to allow users to input information and send it.
- The <form>'s action attribute determines where the form's information goes.
- The <form>'s method attribute determines how the information is sent and processed.
- To add fields for users to input information we use the <input> element and set the type attribute to a field of our choosing:

- Setting type to "text" creates a single row field for text input.
- Setting type to "password" creates a single row field that censors text input.
- Setting type to "number" creates a single row field for number input.
- Setting type to "range" creates a slider to select from a range of numbers.
- Setting type to "checkbox" creates a single checkbox which can be paired with other checkboxes.
- Setting type to "radio" creates a radio button that can be paired with other radio buttons.
- Setting type to "list" will pair the <input> with a <datalist> element.
- Setting type to "submit" creates a submit button.
- A <select> element is populated with <option> elements and renders a dropdown list selection.
- A <datalist> element is populated with <option> elements and works with an <input> to search through choices.
- A <textarea> element is a text input field that has a customizable area.
- When a <form> is submitted, the name of the fields that accept input and the value of those fields are sent as name=value pairs.

Form validation (sprawdzanie formularza)

Validation:

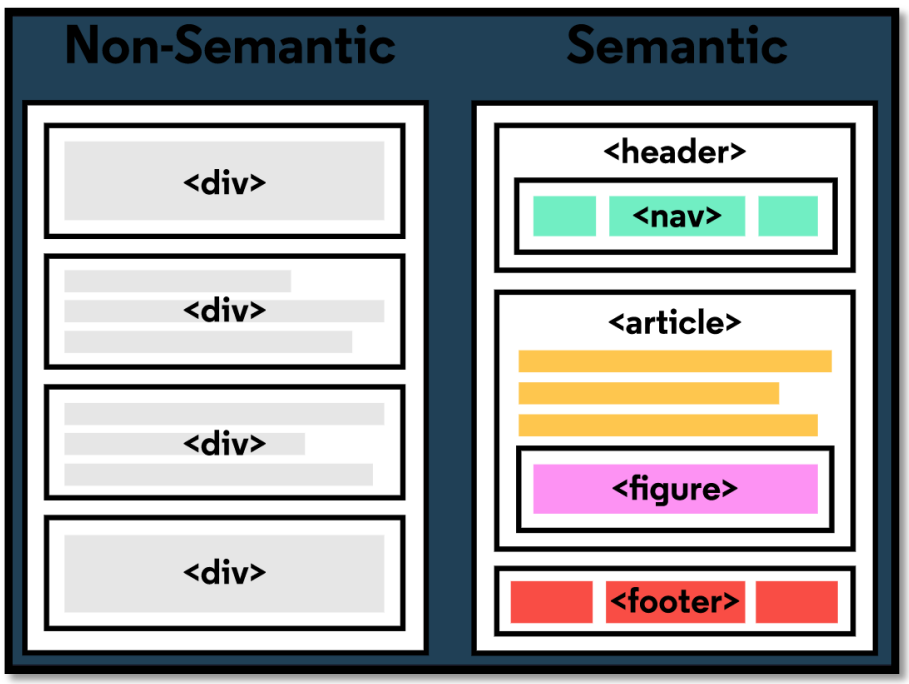
- Server-side – np.login page;
- Client-side – po stronie klienta; sprawdzanie właśnie walidacji, czy wszystko jest poprawnie wpisane, czy w ogóle jest sens wysyłać zapytanie na serwer;

Atrybuty *build-in validation*:

- **Atrybut required** umieszczony w inpucie sprawia ze to pole musi zostać wypełnione przed wysłaniem formularza. Inaczej przeglądarka wyrzuci błąd.
- **Min max value** przy type=***number*** lub ***range***; np. min=”2” -> jeśli będzie w input mniejsza wartość, to przeglądarka przy submicie wyrzuci błąd;
- *Checking length text* – **minlength / maxlength; to samo co wyżej;**
- *Pattern attribute*;
    - *regular expression* – używane w JS, przy zmiennych czy obietkach; generalnie stringi;
    - **regex** – np. **[0-9]{14,16}** – użytkownik może wpisać **tylko cyfry** od 0 do 9, oraz od 14 do 16 znaków;
    - pattern="[a-zA-Z0-9]+" -> tylko cyfry i litery, bez znaków specjalnych;

Semantic HTML

- Lepszy odbiór przez inwalidów czy urządzenia mobilne;
- Wzmacnia SEO *search engine optimization*, co przekłada się na liczbę wyświetleń strony;
- Zwiększa czytelność kodu;



Ważne znaczniki układu strony:

- Head (linki i inne/ukryte)
- **Body** *(parent tag)*
    - **Header**

    - **Nav**
    - **Main**
    - **Footer** (contact info/copyright info/terms of use/site map/reference to top of page links)
- **Dodatkowe tagi/znaczniki**:
    - Section – rozdziały, nagłówki oraz te elementy strony o tym samym motywie/znaczeniu.
    - Article - Pomaga czytać kod – gdzie kończy się i zaczyna element strony ze swoim tekstem, linkami czy obrazkami;
    - Aside – oznaczenie dodatkowej informacji, lecz niewymaganej do zrozumienia całego kontekstu. Przeznaczony do np.; bibliografii, notek kończących, komentarzy, dodatkowych informacji;
    - Figure lub figcaption – to samo co *aside*, tylko dla obrazka lub ilustracji. Figcaption to podpis pod elementem figure ;).
    - Audio – wymagany element src= oraz type=”audio/mp3” autoplay controls(dodaje play/pause buton) loop(zawija piosenkę)>.
    - Embed – video/audio gifs z zewnętrznego źródła, ale może być tez z lokalnego adresu. Działa podobnie jak znacznik video.