

Paweł Rajba

pawel@cs.uni.wroc.pl

<http://pawel.ii.uni.wroc.pl/>

HTML/XHTML

Agenda

- Wprowadzenie
- Podstawy
 - Podstawy składni, szablon dokumentu
 - Wsparcie przeglądarek, nowości HTML5
- Nagłówki, struktura dokumentu i elementy grupujące
- Formatowanie i elementy struktury
 - Formatowanie tekstu, odnośniki
 - Obrazki i mapy, listy, tabele, ramki
- Formularze
- Nowości w HTML5
 - Geolokalizacja, Web workers, Web storage, Canvas, Drag & Drop, Dane strukturalne

Wprowadzenie

- HTML/XHTML jest językiem do prezentacji treści
- Organizacja World Wide Web Consortium (W3C):
 - <http://www.w3c.org/>
 - HTML i CSS: <http://www.w3.org/standards/webdesign/htmlcss>
 - „Best practices” dla HTML: http://www.w3.org/standards/techs/htmlbp#w3c_all
 - Walidator HTML: <http://validator.w3.org/>
- Warte uwagi zasoby: <http://www.w3schools.com/>
- XHTML aplikacja HTML w XML
 - Można stosować narzędzia do XML-a
- Różnice pomiędzy HTML i XML
 - Składniowe, interpretacja (np. pionowe centrowanie treści)

Wprowadzenie

- Problemy z przeglądarkami: utworzona strona może wyglądać
- inaczej w każdej przeglądarce
 - Teoria: utworzone strony powinny tak samo wyglądać w każdej przeglądarce
 - Praktyka: nie w każdej przeglądarce, nie we wszystkich wersjach (kwestia kosztów utworzenia i utrzymywania serwisu)
- Tzw. wiodące przeglądarki:
 - Microsoft Edge, Microsoft Internet Explorer
 - Mozilla Firefox
 - Google Chrome
 - Opera
 - Safari
- Przeglądarki wspierają tworzenie stron:
 - Narzędzia dla programistów w Chrome
 - Narzędzia dla twórców witryn w Firefox
 - Narzędzia programistyczne w IE

Wprowadzenie

- DOM
 - Document Object Model
- Specyfikacja
 - <https://www.w3.org/DOM/DOMTR>
- Co to jest i do czego?

```
...<!DOCTYPE html> == $0
<!-- $Id: DOMTR.html,v 1.80 2012/05/02 23:14:54 plehegar Exp $-->
<html lang="en" xml:lang="en">
<head>...</head>
▶<body onload="init()">
▼<p align="right" class="navbar">
  ▼<a href="/">
    
    </a>
  ▼<a href="/Architecture/" rel="in-domain">
    
    </a>
  ▶<small>...</small>
</p>
<p>
<h1>Document Object Model (DOM) Technical Reports</h1>
▼<p>
  "
  See also "
  <a href="/TR/">other W3C Technical Reports</a>
  "
  "
  </p>
▶<div id="TableOfContents">...</div>
▶<div id="dom4">...</div>
▶<div id="dom3">...</div>
▶<div id="dom2">...</div>
▶<div id="dom1">...</div>
▶<div id="others">...</div>
<hr>
▶<address>...</address>
▶<p class="copyright">...</p>
</body>
</html>
```

Podstawy składni

- Do budowy struktury służą znaczniki
 - parzyste, np. `<div></div>`
 - nieparzyste, np. `<br />`
- Znaczniki można parametryzować atrybutami
 - np. `<image src="img.jpg" alt="Image" />`
- Kilka zasad dotyczących znaczników i atrybutów:
 - nazwy piszemy małymi literami
 - znaczniki zawsze muszą być zamykane
 - znaczniki muszą być poprawnie zagnieżdżane
 - atrybuty są zawsze postaci nazwa="wartość"
 - wartości atrybutów zawsze muszą być w cudzysłowie
- Atrybuty dostępne dla wszystkich (prawie) znaczników:
 - class, id, style, title

Wersje HTML/XHTML

- HTML 5
- HTML 4.01 Strict
- HTML 4.01 Transitional
- HTML 4.01 Frameset
- XHTML 1.0 Strict
- XHTML 1.0 Transitional
- XHTML 1.0 Frameset
- XHTML 1.1

(więcej: https://www.w3schools.com/tags/tag_doctype.asp)

Szablon dokumentu HTML5

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
<meta charset="UTF-8">
</head>

<body>
The content of the document.....
</body>

</html>
```


Szablon dokumentu XHTML5

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Title of the document</title>
<meta charset="UTF-8"/>
</head>

<body>
The content of the document.....
</body>

</html>
```

HTML5 vs. XHTML5

- XHTML5 oznacza, że składnia jest zgodna z XML
- Powinna zostać dołączona przestrzeń nazw XML-a
- Serwer powinien wysłać typ `application/xhtml+xml` zamiast `text/html`
- Przewaga HTML5
 - Drobne błędy składniowe mogą nie wpłynąć na prezentację dokumentu (parser XML jest dużo bardziej rygorystyczny)
- Przewaga XHTML5
 - Można łączyć z innymi standardami XML jak MathML czy SVG
- Do poczytania:
<https://blog.whatwg.org/xhtml5-in-a-nutshell>

Wsparcie przez przeglądarki

- Test na stronie: <http://html5test.com/>

	Chrome	Opera	Firefox	Edge	Safari
Upcoming	67 528		60 497	18 496	11.2 477
Current	66 528	45 518	59 491	17 492	11.1 471
Older	65 528	37 489	58 486	16 476	11 452
	64 528	30 479	57 486	15 473	10.1 406
	63 528	12.10 309	56 478	14 460	10.0 383
	62 528		55 478	13 433	9.1 370
	61 526		54 474	12 377	9.0 360
	60 523		53 474	Internet Explorer	8.0 354
				11 312	

- na 555 możliwych do zdobycia punktów
- Sporo przykładów i przegląd możliwości przeglądarek: <http://html5demos.com/>

Co nowego w HTML5?

- Nowe znaczniki do budowy dokumentu
- Nowe znaczniki formularzy (wraz z nowymi typami)
- Renderowanie grafiki 2D, czyli słynny canvas
- Obsługa audio i video
- Mechanizm przeciągnij i upuść
- Aplikacje w trybie offline
- Przechowywanie danych po stronie klienta (web storage)
- Dane strukturalne (MicroData, data-*)

Co nowego w HTML5?

- Poniższe nie jest częścią standardu, ale jest wspierane przez przeglądarki (i często kojarzone z HTML5)
 - Obsługa geolokalizacji
 - Procesy działające w tle
 - WebSQL, Indexed Data API
 - File API, File Writer
 - Web Sockets

Zawartość nagłówka

- W nagłówku możemy umieścić znaczniki:
 - title – tytuł strony
 - meta – metainformacje
 - link – powiązanie między dokumentami
 - base – adres bazowy dla relatywnych odwołań ze stron

Zawartość nagłówka

- Znacznik link
 - Typowe argumenty:
 - href: adres zasobu
 - type: typ zawartości
 - rel: typ wskazywanego dokumentu
 - Np. Alternate, StyleSheet, Start, Next, Prev, Index, Content, Glossary, Copyright, Appendix, Help
 - Typowe użycie:
 - `<link rel="stylesheet" type="text/css" href="style.css" />`
 - Więcej do poczytania:
 - http://www.w3schools.com/TAGS/tag_link.asp
- Znacznik base, typowe użycie:
 - `<base href="http://pawel.ii.uni.wroc.pl/">`

Zawartość nagłówka

- Znacznik meta
 - atrybuty name i content – informacje o dokumencie
 - atrybuty http-equiv i content – nagłówki protokołu HTTP
- Przykładowe użycia znacznika meta
 - `<meta name="Author" content="Paweł Rajba">`
 - `<meta name="Keywords" content="xhtml,css,php">`
 - `<meta name="Description" content="Strona domowa">`
 - `<meta http-equiv="Content-Type,"
content="text/html; charset=utf-8">`
 - `<meta http-equiv="Content-Language" content="pl">`
 - `<meta http-equiv="Refresh" content="10">`
 - `<meta http-equiv="Refresh" content="10;
URL=http://www.onet.pl/">`

Zawartość nagłówka a wyszukiwarki

- Do sterowania zachowaniem wyszukiwarki mamy dwa mechanizmy:
 - Plik *robots.txt* w katalogu głównym serwisu
 - Odpowiednie wpisy w nagłówku dokumentu

Zawartość nagłówka a wyszukiwarki

- Plik robots.txt

- Słowa kluczowe:

- User-agent – określa wyszukiwarke
 - Disallow – określa wykluczony zasób

- Prosty przykład:

```
User-agent: googlebot
User-agent: slurp
Disallow: /js/
Disallow: /webservicess/
User-agent: *
Disallow: /
```

- Dla zainteresowanych adresy zasobów:

- <http://tools.seobook.com/robots-txt/generator/>

Zawartość nagłówka a wyszukiwarki

- Nagłówek robots w dokumencie
 - Możliwe argumenty: index, follow, noindex, follow, index, nofollow, noindex, nofollow, all
 - Najczęstsze użycia:
 - `<meta name="robots" content="noindex">`
 - `<meta name="robots" content="nofollow">`
 - `<meta name="robots" content="noindex, nofollow">`
- (nie trzeba dodawać index i follow, ponieważ to jest domyślne zachowanie wyszukiwarki)

Zawartość nagłówka a wyszukiwarki

- Warto poczytać co nagłówkach sądzi Google

Other content considerations

Meta tags that Google understands

Keep a simple URL structure

Use canonical URLs

Use rel="nofollow" for specific links

Indicate paginated content

Tag site for child-directed treatment

https://support.google.com/webmasters/topic/4617741?hl=en&ref_topic=4598733

Podstawowe elementy grupujące

- Znacznik `<div>`
 - element typu block
- Znacznik `<span>`
 - element typu inline

Struktura dokumentu HTML5

- Article
 - blog post, user comment
- Aside
- Audio, Video
- Canvas
- Summary i details
- Embed
- Figure, Figcaption
- Header, Footer
- Address
- Section
- Mark
- Meter
- Output
- Progress
- Time
- Wbr
- Nav

Więcej:

https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Document_and_website_structure#Enter_HTML5_structural_elements
<https://stackoverflow.com/questions/7549561/section-vs-article-html5>

Demo

- 01-simplepage.html
 - przegląd w różnych przeglądarkach

Formatowanie tekstu

- Określanie czcionki
 - , , <dfn>, <code>
 - <tt>, <i>, , <u>, <big>, <small>
- Indeksy
 - <sub>, <sup>
- Cytowania
 - <blockquote> – typu block
 - <q> – typu inline
 - dla obu możemy określić atrybut cite="URL"

Formatowanie tekstu

- Elementy blokowe

- `<h1>-<h6>`,
- `<address>`,
- `<p>`,
- `<pre>`,
- `<br />`,
- `<hr />`
 - atrybuty `noshade`, `size`, `width`
- `<ins>`, `<del>`
 - atrybut `cite="URL"`, `datetime="datetime"`

Odnosińiki

- Tworzymy za pomocą znacznika `<a>`. Atrybuty
 - `href="adres"`
 - `target=" blank" | " parent" | " self" | " top"`
- Zakotwiczenia – mechanizm pozwalający nawigować wewnątrz dokumentu
 - Możemy napisać `<a href="d.html#kotwica">Kotwica</a>` lub `<a href="#kotwica">Kotwica</a>`
 - Kotwica jest wtedy dowolny element o `id="kotwica"`, np. `<a name="kotwica"></a>` lub `<div id="kotwica"></div>`
- Adresowanie
 - względne, np. `<a href="../index.html">Home</a>`
 - bezwzględne, np. `<a href="http://home.pl/">Home</a>`

Obrazki i mapy

- Obrazek umieszczamy za pomocą znacznika `<img>`.
Atrybuty:
 - `src="URI", alt="opis", name="nazwa", height="140", width="200", usemap="#mapa"`
- Co to jest mapa?
- Mapę tworzymy za pomocą znacznika:
 - `<map id="nazwa">`
- W skład mapy wchodzi jeden więcej obszarów, które definiujemy znacznikiem `<area>`. Atrybuty
 - `shape="rect | circle | poly | default"`
 - `coords="1,2,3,4", alt="tekst"`
 - `href="URI", nohref="nohref"`

Obrazki i mapy

- Atrybut coords zależy od atrybutu shape:

- rect – lewy-x, góra-y, prawy-x, dół-y
- circle – środek-x, środek-y, promień
- poly – x1,y1,x2,y2,...,xN,yN

- Przykład:

```

<map id="mapa">
  <area href="guide.html" alt="Access Guide"
    shape="rect" coords="0,0,118,28" />
  <area href="search.html" alt="Search"
    shape="rect" coords="184,0,276,28" />
  <area href="shortcut.html" alt="Go"
    shape="circle" coords="184,200,60" />
  <area href="top10.html" alt="Top Ten" shape="poly"
    coords="276,0,276,28,100,200,50,50,276,0" />
</map>
```

Listy

- Mamy trzy rodzaje list
 - Lista numerowana
 - Lista nienumerowana
 - Lista definicji

Lista nienumerowana

- Tworzymy znacznikiem `<ul>`. Atrybuty:
 - `type="disc" | "circle" | "square"`
 - `compact` — większy stopień upakowania
- Elementy listy tworzymy znacznikiem `<li>`
- Przykład:

```
<ul>
<li>Warszawa</li>
<li>Wrocław</li>
<li>Kraków</li>
</ul>
```

Lista numerowana

- Tworzymy znacznikiem `<ol>`. Atrybuty:
 - `start="liczba"`
 - `type="1" | "A" | "a" | "i" | "I"`
 - `compact` — większy stopień upakowania
- Elementy listy tworzymy znacznikiem `<li>`
 - Mamy dodatkowo atrybut `value`
- Przykład:

```
<ol start="5" type="A">
<li>Procesor</li>
<li>Pamiec</li>
</ol>
```

Lista definicji

- Tworzymy znacznikami <dl>, <dt>, <dd>
- Przykład:

```
<dl>
<dt>Afrodyta</dt>
<dd>bogini miłości i piękna</dd>
<dt>Nemesis</dt>
<dd>uosobienie gniewu bogów i kary spadającej na ludzi
    przekraczających wyznaczone im granice</dd>
<dt>Posejdon</dt>
<dd>bóg morza, opiekun żeglarzy i rybaków; syn Kronosa
    i Rei, brat Zeusa i Hadesa, mąż Amfitryty</dd>
</dl>
```


Tabele

- Tworzymy z wykorzystaniem znaczników:
 - `<table>`, `<th>`, `<tr>`, `<td>`, `<caption>`, `<colgroup>`, `<col>`, `<thead>`, `<tbody>`, `<tfoot>`
- Struktura tabeli jest następująca:
 - tabela składa się z wierszy,
 - wiersze składają się z kolumn
- Główny znacznik to `<table>`. Atrybuty:
 - `summary="tekst"`
 - `width="50%"` j `"500"`
 - `border="2"`, `cellpadding="4"`, `cellspacing="1"`

(więcej pod adresem: <http://www.w3.org/TR/html401/struct/tables.html#margins>)

Tabele, prosty przykład

```
<table>
<tr><th>Nr indeksu</th><th>Ocena</th></tr>
<tr><td>91044</td><td>5.0</td>
<tr><td>91057</td><td>5.0</td>
<tr><td>91088</td><td>5.0</td>
<tr><td>91092</td><td>5.0</td>
</table>
```

Łączenie wierszy i kolumn

- Łączenie komórek realizuje się w znacznikach `<th>` i `<td>`
- Do łączenia służą atrybuty
 - `colspan="3"`
 - `rowspan="2"`
- Utwórzmy kod dla poniższej tabelki:

Indeks	Ocena	Grupa
91044	5.0	Gr. 1
91055	5.0	
Razem		

Grupowanie wierszy

- Mamy trzy rodzaje grup: head, body i foot
- Każda grupa musi mieć co najmniej 1 wiersz
- Sekcja tfoot powinna być przed tbody,
- Znacznik <tbody> jest obowiązkowy, chyba że występuje sam (wtedy można go pominąć)
- Szablon tabeli z wykorzystaniem grup:

```
<table>
<thead>...</thead>
<tfoot>...</tfoot>
<tbody>...</tbody>
<tbody>...</tbody>
...
</table>
```

Grupowanie kolumn

- Realizowany poprzez znacznik `<colgroup>`.
Atrybuty
 - `span="4"`
 - `width="50"` (dotyczy każdej kolumny w grupie)
- Do stosowania wspólnego formatowania przydatny jest znacznik `<col>`
 - Nie grupuje kolumn
 - Atrybuty `span` i `width`

Grupowanie kolumn

Przykład:

```
<table>
<colgroup>
<col width="30" />
</colgroup>
<colgroup>
<col width="30" />
<col width="0*" />
<col width="2*" />
</colgroup>
<colgroup align="center">
<col width="1*" />
<col width="3*" />
</colgroup>
<thead>
...dalsza czesc tabelki...
</table>
```

Linie w tabeli

- Mamy dwa atrybuty znacznika <table>
 - frame – określa sposób obramowania; wartości:
 - void – z żadnej strony
 - above, below – na górze, na dole
 - lhs, rhs – z lewej, z prawej
 - hside – na górze i na dole, d – d
 - vside – z lewej i z prawej
 - box, border – z każdej strony
 - rules – określa sposób wyświetlania linii wewnętrznych
 - wartości: none, all, groups, rows, cols

Ramki

- Przede wszystkim nie należy ich stosować
 - mówimy o nich, ponieważ są jednym ze sposobów na zbudowanie szablonu strony
- Realizowane poprzez znaczniki: `<frameset>`, `<frame>` i `<noframes>`
- Okna (ramki) reprezentuje znacznik `<frame>`.
Atrybuty
 - `name="nazwa", src="URI", frameborder="1 | 0",`
 - `marginwidth="pixele", marginheight="pixele",`
 - `scrolling="yes | no | auto"`

Ramki

Przykład:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Frameset//EN">
<HTML>
<HEAD><TITLE>Przykładowy dokument z ramkami</TITLE></HEAD>
<FRAMESET cols="20%,80%">
<FRAMESET rows="100,*">
<FRAME src="frame1.gif">
<FRAME src="frame2.html">
</FRAMESET>
<FRAME src="frame3.html">
<NOFRAMES>
<P>Ten dokument zawiera:
<UL>
<LI><IMG src="frame1.gif" alt="Obrazek">
<LI><A href="frame2.html">Dokument 2</A>
<LI><A href="frame3.html">Dokument 2</A>
</UL>
</NOFRAMES>
</FRAMESET>
</HTML>
```

Ramki pływające

- Co to jest ramka pływająca?
- Do utworzenia takiej ramki używamy znacznika `<iframe>`
- Atrybuty znacznika `<iframe>`
 - `name`, `src`, `frameborder`, `marginwidth`, `marginheight`, `scrolling`, `height`, `width`, `align`="left | right | middle | top | bottom"
- Przykład:

```
<iframe src="ramka.html" width="400" height="500"
scrolling="auto" frameborder="1" align="right">
Twoja przeglądarka aktualnie nie pokazuje ramek.
Zawartość tej ramki można obejrzeć na stronie pod <a
href="ramka.html">tym</a> adresem.
</iframes>
```

Formularze

- Jakiego jest przeznaczenie formularzy?
- Kontrolki, za pomocą których możemy budować formularze:
 - przyciski (`<button>`, `<input>`)
 - checkbox-y (`<input>`)
 - przyciski radio (`<input>`)
 - listy (`<select>` + `<option>` + `<optgroup>`)
 - pola tekstowe (`<input>`, `<textarea>`)
 - wybór pliku (`<input>`)
 - pole ukryte (`<input>`)

Utworzenie i właściwości formularza

- Formularz tworzymy znacznikiem <form>. Główne atrybuty:
 - id="ID", name="nazwa"
 - action="URI"
 - method="POST | GET" (domyślnie GET)
 - enctype="typ zawartości" (ma sens w przypadku metody POST).
Wartości:
 - application/x-www-form-urlencoded (domyślnie)
 - multipart/form-data (przy wysyłaniu plików)
- Atrybuty wspólne dla większości kontrolek:
 - name="nazwa" (obowiązkowo)
 - readonly="readonly"
 - disabled="disabled"
 - tabindex=

Pola tekstowe

- Za pomocą znacznika `<input>`
 - Znacznik `<input>` przyjmuje wtedy atrybuty:
 - `type="text"` lub `type="password"`
 - `size="20", maxlength="40"`
 - Przykład:
`<input id="txtLoginName" type="text" size="20" maxlength="50" />`
- Za pomocą znacznika `<textarea>`
 - Znacznik `<textarea>` ma atrybuty `rows="10", cols="40"`
 - Przykład:
`<textarea id="txtDesc" rows="20" cols="80">`
Pierwszy wiersz tekstu początkowego.
Drugi wiersz tekstu początkowego.
`</textarea>`

Listy

- Tworzymy za pomocą znaczników `<select>` i `<option>` (opcjonalnie znacznika `<optgroup>`)
- Atrybuty znacznika `<select>`
 - `size="3"`
 - `multiple="multiple"`
- Atrybuty znacznika `<option>`
 - `selected="selected"`
 - `value="wartosc"`
 - `label="tekst"`
- Atrybuty znacznika `<optgroup>`
 - `label="opis"`

Listy

Przykłady

```
<select id="miasto" name="'miasto'">
<option value="0">Wrocław</option>
<option value="1">Kraków</option>
<option value="2">Poznań</option>
</select>
```

```
<select id="Linux" name="'Linux'">
<option selected="selected" value="none">None</option>
<optgroup label="SUSE">
<option value="opensuse">openSUSE 10.3</option>
<option value="sles">SUSE Linux Enterprise Server
10</option>
</optgroup>
<optgroup label="Ubuntu">
<option value="ubuntud71">Ubuntu Desktop 7.10</option>
<option value="ubuntus71">Ubuntu Server 7.10</option>
</optgroup>
</select>
```

Checkbox i radio

- Tworzymy je za pomocą znacznika `<input>`
- Znacznik przyjmuje wtedy atrybuty:
 - `name="nazwa"`
 - `checked="checked"`
 - `value="wartość"`
- W przypadku radio, grupa elementów, w której wybrać można tylko jeden element, ma wspólna wartość atrybutu `name` (ale wartości atrybutów `id` muszą mieć różne)

Checkbox i radio

Przykłady:

```
<input name="c1" type="checkbox" checked="checked" value="0" /> Sport  
<input name="c2" type="checkbox" checked="checked" value="1" /> Muzyka  
<input name="c3" type="checkbox" checked="checked" value="2" /> Polityka  
  
<input type="radio" checked="checked" name="sex" value="m"/> Mężczyzna  
<input type="radio" name="sex" value="k"/> Kobieta
```

Przyciski

- Możemy utworzyć na kilka sposobów:
 - `<input type="submit" value="napis" />`
 - Wciśnięcie spowoduje wysłanie danych z formularza
 - `<input type="image" src="przycisk.jpg" />`
 - Wciśnięcie spowoduje wysłanie danych z formularza
 - Dodatkowo wysyłane są współrzędne miejsca kliknięcia w obrazek
 - `<input type="button" value="napis" />`
 - Wciśnięcie nie spowoduje wysłanie danych z formularza

Przyciski

- `<input type="reset" value="Domyślne" />`
 - Wciśnięcie spowoduje do kontrolek formularza wartości domyślnych
- `<button></button>`
 - Atrybuty znacznika `<button>`
 - `value="wartosc"` (wysyłane do serwera)
 - `type="button | submit | reset"`
 - Przykład:
 - `<button name="reset" type="reset"> Reset</button>`

Pozostałe

- Pole wyboru pliku
 - `<input name="filename" type="file" size="30" />`
- Wartość ukryta
 - `<input type="hidden" name="viewstate" value="X" />`
- Obramowanie
 - Realizowane przez znaczniki `<fieldset>` i `<legend>`
 - Przykład:

```
<fieldset>
<legend>Dane</legend>
Zawartosc formularza
</fieldset>
```

Kontrolki formularzy w HTML5

- color
- datalist
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Demo

- o2-newmarkup.html
 - przegląd w różnych przeglądarkach

Walidacja formularzy w HTML5

- Określana przez atrybuty kontrolek
- Podpowiedź
 - placeholder
- Mamy do dyspozycji:
 - required
 - min, max
 - pattern
 - maxlength
 - ...
- Wyłączenie walidacji
 - `<form novalidate ...>`
 - `<input type="submit" formnovalidate="formnovalidate"...>`
- Domyślne zachowanie przeglądarki vs. pseudo klasy CSS3

Demo

- `o3-simplevalidation1.html`
- `o3-simplevalidation2.html`

Walidacja formularzy w HTML5

- Manualne sterowanie komunikatami błędów, czyli użycie „built-in validation rules”
 - valueMissing
 - typeMismatch
 - patternMismatch
 - tooLong
 - rangeUnderflow i rangeOverflow
 - stepMismatch
 - valid
- Metoda checkValidity() i właściwość validity
 - <https://developer.mozilla.org/en-US/docs/Web/API/ValidityState>

Demo

- o4-bettervalidation1.html
- o4-bettervalidation2.html

Nowości HTML5

- Dalej przejrzymy wybrane nowości HTML5
 - Geolokalizacja
 - Web workers
 - Web storage
 - Canvas
 - Drag & Drop
 - Dane strukturalne

Geolokalizacja

- Główna metoda: `getCurrentPosition()`
- Dwa parametry:
 - `showPosition` – callback, do którego przekazywany jest obiekt lokalizacji o następujących właściwościach:
 - `coords.latitude`, `coords.longitude`, `coords.accuracy`
 - `coords.altitude`, `coords.altitudeAccuracy`
 - `coords.heading`
 - `coords.speed`
 - `Timestamp`
 - `showError` – callback, do którego przekazywany jest obiekt błędu, jeśli wystąpił, o następujących polach:
 - `error.PERMISSION_DENIED`:
 - `error.POSITION_UNAVAILABLE`:
 - `error.TIMEOUT`:
 - `error.UNKNOWN_ERROR`:

Geolokalizacja

- Są też

- metody `watchPosition()`
- `clearWatch()`

do śledzenia pozycji (sens przy smartfonie, tablecie, ewentualnie laptopie)

Demo

- 05-geolocation.html
 - odpalamy na serwerze

Web Workers

- Mechanizm uruchamiania procesów w tle
- Mamy dwa rodzaje web worker'ów
 - Dedicated – powiązany z konkretnym oknem
 - Shared – wspólny dla całej domeny
- Worker nie ma dostępu do
 - DOM
 - Okna (Window)
 - Strony, z której został wywołany (host page)
- Worker ma dostęp do
 - Przeglądarki (appName, userAgent, ...)
 - setTimeout, setInterval, clearTimeout, clearInterval
 - XMLHttpRequest

Web Workers

- Czego potrzebujemy?
 - skrypt procesu w osobnym pliku
 - metoda `postmessage(dane)` do wysyłania komunikatów
 - Uwaga: przy wysyłaniu dane są kopiowane, więc wielkich danych może pojawić się kłopot
 - zdarzenie `worker.onmessage` do obsługi wysłanych komunikatów
- Zastosowania
 - Obliczenia
 - Generowanie grafiki
 - Kryptografia (szyfrowanie, ...)
- Wsparcie
 - Chrome, FF, Opera, Safari (czyli brakuje IE)

Demo

- o6-worker.html
- o7-jsonworker.html
- o8-queueworker.html
 - działa pod Opera i Chrome, ale Chrome dopiero po opublikowaniu

WebStorage

- Mamy do dyspozycji:
 - localStorage - bezterminowe
 - sessionStorage – per sesja (do zamknięcia okna przeglądarki)
- Bezpieczeństwo: per sesja i per domena
- Pojemność: 2-10MB, zwykle ok. 5MB
- WebStorage vs. Cookies
 - Tylko po stronie klienta
 - Większa pojemność
- W obu repozytoriach
 - dane są pamiętane jako pary klucz/wartość
 - wszystko jest typu string
- Wsparcie we wszystkich przeglądarkach
 - Chrome 4, FF 3.5, Opera 10.6, Safari 4, IE 8 (i wyższych)

WebStorage

- API
 - `storage.clear`
 - `storage.getItem`
 - `storage.key`
 - `storage.length`
 - `storage.remainingSpace`
 - `storage.removeItem`
 - `storage.setItem`
 - `event` (tylko w IE 9)

WebStorage

- Dostęp na różne sposoby:
 - `localStorage.zmienna = "wartosc"`
 - `localStorage["zmienna"] = "wartosc"`
 - `localStorage.setItem("zmienna", "wartosc")`
- `alert(localStorage.zmienna)`
- `alert(localStorage["zmienna"])`
- `alert(localStorage.getItem("zmienna"))`

WebStorage

- Limit miejsca (quota)
 - Na każdej przeglądarce inny
 - Dodanie nowej wartości warto opakować w try..catch..

```
try {  
    window.localStorage.setItem(key,value) ;  
}  
catch(e) {  
    // obsługa błędu  
}
```
 - W przypadku przekroczenia limitu pojawi się błąd
 - DOM Exception: QUOTA_EXCEEDED_ERR (22)
 - storage.remainingSpace działa tylko w IE 9

WebStorage

- Można także śledzić zmiany:
 - podpinamy zdarzenie:
`window.addEventListener("storage", handle_storage, false)`
 - tworzymy funkcję obsługi
`function handle_storage(e) { }`
 - gdzie e ma pola
key, oldValue (wartość null dla setItem), newValue, url lub uri
- Niektóre przeglądarki wspierają także Web SQL Database, co daje interfejs SQL, a nie tylko key/value.
- Fajny artykuł o web storage:
<http://diveintohtml5.info/storage.html>

Demo

- 09-webstorage.html
- Przeglądamy
 - <http://wroclawski.eu/09-webstorage.html>
 - <http://wroclawski.eu/storage/09-webstorage.html>
 - <http://spacerownik.wroclawski.eu/09-webstorage.html>
 - <http://itcourses.eu/w3/09-webstorage.html>

Canvas

- ... czyli obrazki i rysowanie
- Tworzymy płótno:
 - `<canvas id="myCanvas" width="200" height="100"></canvas>`
- a potem kod do rysowania:
 - ```
<script type="text/javascript">
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.fillStyle="#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```



# Demo

- Oglądamy stronę  
<http://www.html5canvastutorials.com/>

# Drag & Drop

- Miły dla oka efekt, dostępny dotychczas tylko przez spory zestaw skryptów JavaScript
- Jak to robimy w HTML?
  - Ustawiamy content jako „przesuwalny”
    - `draggable=true`
    - dla `img` jest to domyślne
  - Podpinamy funkcje pod zdarzenia
    - **dragstart**, **drag**, **dragenter**, **dragleave**, **dragover**, **drop**, **dragend**
  - Tam, gdzie obiekt ma być „upuszczony”, trzeba go wstawić
    - Wykorzystujemy tzw. `data transfer` object
- Dokumentacja
  - [https://developer.mozilla.org/en-US/docs/Web/API/HTML\\_Drag\\_and\\_Drop\\_API](https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API)
- Bardzo dobry artykuł
  - <http://www.html5rocks.com/en/tutorials/dnd/basics/>

# Demo

---

- 10-drag&drop1.html
- 10-drag&drop2.html

# Dane strukturalne

- Dane mogą być prezentowane w formacie przyjaznym dla
  - Człowieka (HTML5+CSS3)
  - Maszyny (XML, JSON)
- MicroData czy data-\* umożliwiają połączenie obu światów
- W ramach dokumentu HTML5 dodajemy informacje, które mogą zostać przetworzone przez maszyny
- Zastosowanie
  - MicroData – korzysta z tego np. wyszukiwarka Google
  - data-\* - łatwe do przetwarzania w JavaScript

# DEMO

- Oglądamy:

- [https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use\\_data\\_attributes](https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes)
- <https://schema.org/docs/gs.html>

- Wklejamy przykład 1d z

<https://schema.org/docs/gs.html>

do

<https://search.google.com/structured-data/testing-tool/u/o/>

- Odpalamy

<https://search.google.com/structured-data/testing-tool/u/o/#url=https%3A%2F%2Fwww.w3.org%2FTR%2Fmicrodata%2F>

# Dane strukturalne

## ■ Narzędzie do testowania

- <https://search.google.com/structured-data/testing-tool>

## ■ Do poczytania:

- <https://www.w3.org/TR/microdata/>
- <https://developer.mozilla.org/en-US/docs/Web/HTML/Microdata>
- <https://schema.org/>
- <https://developers.google.com/search/docs/guides/intro-structured-data>
- [https://en.wikipedia.org/wiki/Microdata\\_\(HTML\)](https://en.wikipedia.org/wiki/Microdata_(HTML))
- [https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using\\_data\\_attributes](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_data_attributes)
- <http://www.hongkiat.com/blog/better-ux-html-data-attributes/>
- <https://www.sitepoint.com/use-html5-data-attributes/>

# Pozostałe możliwości

- Bardzo dużo nowych zdarzeń
  - [http://www.w3schools.com/tags/ref\\_eventattributes.asp](http://www.w3schools.com/tags/ref_eventattributes.asp)
- Wsparcie dla grafiki wektorowej SVG
  - ```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" height="190">  
  <polygon points="100,10 40,180 190,60 10,60 160,180"  
    style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;">  
</svg>
```
- Wsparcie dla tworzenia wzorów w MathML
- Video & audio
- Server-Sent Events
- Aplikacje offline (app cache)
 - Główny bohater: plik manifest
- Niezłe tutoriale
 - <http://www.w3schools.com/html/>
 - <http://www.tutorialspoint.com/html5/>