TIN
Techniki Internetowe
lato 2018

Grzegorz Blinowski

Instytut Informatyki

Politechniki Warszawskiej

Plan wykładów

2	Intersieć, ISO/OSI, protokoły sieciowe, IP
3	Protokół IP i prot. transportowe: UDP, TCP
4	Model klient-serwer, techniki progr. serwisów
5	Protokoły aplikacyjne: telnet, ftp, smtp, nntp, inne
6	HTTP
7	**HTML i okolice**
8	XML
9, 10, 11	Aplikacje WWW, CGI, sesje, serwery aplikacji serwlety, integracja z backended SQL
12	Aspekty zaawansowane: wydajność, przenośność, skalowalność; klastering
13	XML, RDF, SOAP, WSDL, ontologie
14	Wstęp do zagadnień bezpieczeństwa (IPSec, VPN, systemy firewall) oraz aspekty kryptograficzne (DES, AES, RSA, PGP, S/MIME), tokeny i akceleracja sprzętowa

Plan wykładów

2	Intersieć, ISO/OSI, protokoły sieciowe, IP
3	Protokół IP i prot. transportowe: UDP, TCP
4	Model klient-serwer, techniki progr. serwisów
5	Protokoły aplikacyjne: telnet, ftp, smtp, nntp, inne
6	HTTP
7	**HTML i okolice**
8	XML
9, 10, 11	Aplikacje WWW, CGI, sesje, serwery aplikacji serwlety, integracja z backended SQL
12	Aspekty zaawansowane: wydajność, przenośność, skalowalność; klastering
13	XML, RDF, SOAP, WSDL, ontologie
14	Wstęp do zagadnień bezpieczeństwa (IPSec, VPN, systemy firewall) oraz aspekty kryptograficzne (DES, AES, RSA, PGP, S/MIME), tokeny i akceleracja sprzętowa

Struktura stron WWW - HTML

HTML

- HTML – **H**yper**t**ext **M**arkup **L**anguage
- Markup:

  – *the process of marking manuscript copy for typesetting with directions for use of type fonts and sizes, spacing, indentation, etc. (The Chicago Manual Of Style).*

- HTML opisuje sposób w jaki przeglądarka wyświetli stronę WWW
- HTML **nie jest** językiem programowania

Historia

- **lata 60**: idea uporządkowanego oddzielenia treści od jej formatowania (Stanley Rice; proj. GenCode; proj. INTIME - IBM) 1969, IBM: GML – Generalized Markup Language
  (Goldfarb, Mosher, Laurie).
- **1986:** SGML – Standard Generalized Markup Language,
  ISO 8879:1986.
- **1989:** powstaje World Wide Web (Tim Berners-Lee)
- **1991:** powstaje HTML 1.0, przeglądarka „WWW” (Tim Berners-Lee)
- **1993:** Przeglądarka NCSA Mosaic, potem “Netscape”
- **1995:** HTML 2.0 zdefiniowany jako zastosowanie
  SGML-a.; Berners-Lee, T. & D. Connolly (HTML Working Group)
- **1996:** RFC1942 - HTML tables, RFC1980 Image maps
- **1997:** HTML 3.2 Reference Specification (W3C)
- **1999:** HTML 4.01 Reference Specification (W3C)

Historia

- **1998:** XML – Extensible Markup Language (W3C)
- **2000**: XHTML 1.0 - W3C „Recommended”
- **2008**: HTML5 - WHATWG (Web Hypertext Application Technology Working Group) - „First Public Draft”
- **2014:** HTML 5 – W3C „Final Recommendation”: rozszerza HTML4 i zastępuje późniejsze: XHTML v. 1 oraz HTML DOM Level 2; obejmuje m.in. bezośrednią obsługę multimediów, rozszerzenia dla aplikacji Web oraz klientów mobilnych

Formatowanie - Markup

Bułhakow Michał, Mistrz i Małgorzata, tłum. I. Lewandowska, W.Dąbrowski, Warszawa: Czytelnik 1992. W białym płaszczu z podbiciem koloru krwawnika posuwistym krokiem kawalerzysty wczesnym rankiem czternastego dnia wiosennego miesiąca nisan pod krytą kolumnadę łączącą oba skrzydła pałacu Heroda Wielkiego wyszedł procurator Judei Poncjusz Piłat. ...

Metainformacja - opis informacji

Informacja

dokument

formatowanie

Bułhakow Michał
Mistrz i Małgorzata,
tłum. I. Lewandowska, W.Dąbrowski,
Warszawa: Czytelnik 1992.

W białym płaszczu z podbiciem koloru krwawnika posuwistym krokiem kawalerzysty wczesnym rankiem ….

Wyrażone przez markup

```
<em>Bułhakow Michał</em><br/>
<h1>Mistrz i Małgorzata,</h1>
<h2>tłum. I. Lewandowska,  …
M</h2>

<p>W białym płaszczu z
podbiciem koloru krwawnika
posuwistym krokiem ...
```

HTML

- HTML opisuje wyłącznie **formatowanie**, tj. sposób prezentacji dokumentu
- Meta-informacja nie jest w (prawie) żaden sposób wyrażana w HTML
- HTML nie wyraża i nie opisuje też semantyki (**znaczenia** informacji)
- w HTML nie ma oddzielenia **znaczenia** tekstu i sposobu **jego prezentacji** (co umożliwia XML):

```
<autor>Bułhakow Michał</autor>
<tytul>Mistrz i Małgorzata</tytul>
<tlumacz>tłum. I. Lewandowska,  … </tlumacz>

<tresc>W białym płaszczu z podbiciem
  koloru<BR>krwawnika posuwistym krokiem … </tresc>
```

Inne systemy Markup

- HTML nie jest jedynym systemem format-markup,
- HTML nie jest też systemem szczególnie wyrafinowanym, uniwersalnym, spójnym lub przenośnym
- O popularności HTML przesądził rozwój Internetu i prostota tego formatu, przy obecności kilku silnych funkcjonalnie mechanizmów, głównie: linki i osadzanie grafiki

HTML	`<B>Bułhakow Michał</B>`
RTF	`{\b\f5\cf1 Bułhakow Michał}`
LaTeX	`{\bf Bułhakow Michał}`
PostScript	`/Times-BoldR 900 ff (Bułhakow Michał)W`

HTML - ekspresowy kurs - znaczniki

- Znacznik (tag):

```
<h1>
```

- Przeważająca wiekszość występuje w parach (otwierający, zamykający):

```
<h1> </h1>
```

- Znacznik może być opatrzony atrybutami:

```
<img src="pict1.jpg" alt="obrazek"/>
```

- Nieznany (nielegalny) znacznik jest ignorowany wraz z atrybutami
- Tekst otoczony nieznanymi znaczniki jest wyświetlany bez żadnych modyfikacji
- Komentarze:

```
<!-- tekst komentarza -->
```

HTML - ekspresowy kurs
Podstawowe elementy formatowania

- **Struktura dokumentu:**

```
<html>
<head>
<meta http-equiv=Content-Type content="text/html; charset=UTF-8">
<meta NAME="description" content="Znana powieść autorstwa ...">
<meta NAME="keywords" content="fikcja, proza">
<base href="http://www.mim.org/">
<title> Mistrz i Małgorzata </title>
</head>
<body>
<p>W białym płaszczu z podbiciem koloru... </body>
</html>
```

W HTML5:

– `<meta charset="utf-8”>`

HTML - ekspresowy kurs
Oznakowanie logiczne i fizyczne

- Znakowanie (markup) fizyczny
  - określa bezpośrednio cechy jakie ma przybrać formatowany dokument, np:
  - **<b>** - wytłuszczenie, **<i>** - kursywa, **<font face="arial">** - konkretny krój
- Znakowanie (markup) logiczny
  - określa znaczenie fragmentu dokumentu
  - program formatujący sam wybiera jak to znaczenie przedstawić, np.:
  - **** - emfaza, **<h*n*>-** nagłówki, **** - wyróżnienie, **<code>** - tekst "komputerowy"

Przykłady HTML

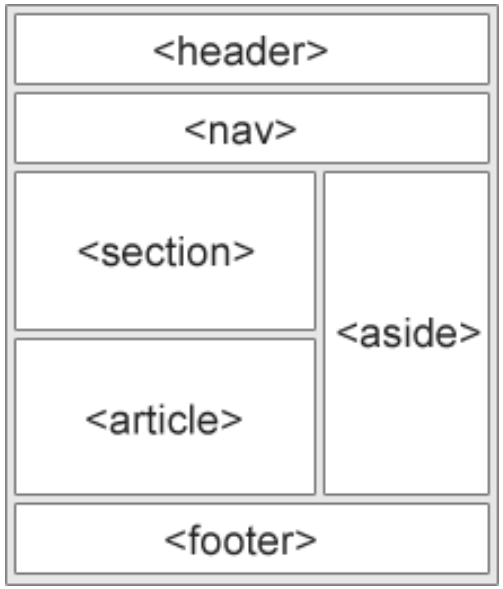
→ *zob. przykłady źródłowe*

HTML5

- Różnice w stosunku do HTML 4.01
  - Nie bazuje na SGML (syntaktycznie: zdefiniowano od nowa)
  - Można w treści umieszczać konstrukcje SVG (grafika wektorowa) oraz MathML (wzory)
  - Można osadzać obiekty multimedialne, nowe znaczniki: **`<video>`, `<audio>`, `<canvas>`**
  - Dodano wiele innych nowych znaczników w tym nowych znaczników do formularzy
  - usunięto m.in. znaczniki: **`<applet>`, `<center>`, `<font>`, `<frame>`, `<frameset>`, `<isindex>`, `<noframes>`, `<strike>`, `<tt>`**

HTML5 – istotne nowości c.d.

- Formatowanie strony: rezygnujemy z formatowania tabelami
- Nowe elementy: <header>, <article>, <section>, <nav>, <aside>, <footer>:
  - mają znaczenie semantyczne (np. nie ma znaczenia sematycznego, bo nie mówi nic o swojej zwartości)
  - **<section>** - tematycznie ujęte dane, typowo opatrzone nagłówkiem
  - **<article>** - niezależnie/autonomiczne dane (np. post na blogu)




```
<section>
  <h1>TIN</h1>
  <p>Celem przedmiotu TIN jest ... </p>
</section>
```

HTML5 – istotne nowości c.d.

- Canvas (<canvas>): API do grafiki 2D
  - Rysunki generuje się przy pomocy funkcji JavaScript
  - Dostępny jest standardowy zestaw prymitywów graficznych
  - Przykład:

```
<canvas id="pic" width="550" height="500">
<!--image for no canvas support-->
<img src="noconvas.png" alt="picture"/>
</canvas>

<script>
  var c = document.getElementById("pic");
  var ctx = c.getContext("2d");
  ctx.fillStyle = "#FF8080";
  ctx.fillRect(0,0,100,150);
</script>
```

HTML5 – istotne nowości c.d.

- Multimedia:

```
<video width="640" height="480">
  <source src="mymovie.mp4" type="video/mp4">
  <source src="mymovie.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
```

- Obsługa aplikacji offline (dostępnych gdy brak łączności z serwerem), dwa mechanizmy:
  - Lokalne kontenery danych, dostęp przez API JS:
    - podobne do cookie, ale do 5MB, związane z daną domeną
    - Przechowywane są pary klucz-wartość
    - WEBSQL – lokalne bazy SQL z API w JS
  - Application cache, “manifast cache” - jawne wskazanie w tagu <html> co podlega cachowaniu w przeglądarce, a co nie

HTML5 – istotne nowości c.d.

- Microdata: pozwala na dodanie meta-informacji do strony WWW
  - Dostępne są różne schematy meta-informacji (schemata.org): osoba, organizacja, produkt, …
  - Nie wszystkie przeglądarki obsługują
  - Przykład:

```
<section itemscope itemtype="http://schema.org/Person">
 Cześć, jestem
 <span itemprop="name">Jan iksiński</span>,
 <span itemprop="jobTitle">student</span> na
 <span itemprop="affiliation">Politechnika
     Warszawska</span>.
</section>
```

HTML5 – istotne nowości c.d.

- Edytowalne dokumenty (fragmenty dokumentów) – atrybut "*contenteditable*"
- Funkcje drag-and-drop
- "Cross-document-messaging" API:
  - Wymiana komunikatów między dokumentami
  - W poprzednich wersjach HTML zakazane ze względów bezpieczeństwa - "cross-site-scripting", ale często potrzebne
  - API pozwala na wysłanie komunikatu do innego dokumentu
- Zarządzanie historią odwiedzin
- Usystematyzowanie obsługi danych o różnym typie MIME
- Webworker – skrypt JS działający "w tle"
- Geolokalizacja (funkcje JS)
- SSE - "Server Sent Event" - zdarzenia wysyłane przez serwer aktualizujące stronę

JavaScript (ECMA Script)

Standaryzacja:
- *ECMAScript 2016*
- *ISO/IEC 16262 (2015)*

- Język interpretowany przez klienta (przeglądarkę WWW)
- Łaczy trzy modele programowania:
  - Obiektowy, imperatywny i funkcyjny (deklaratywny)
- Statycznie obiektowy:
  - obiekty odpowiadają elementom dokumentu
  - b. ograniczone możliwości tworzenia klas i dziedziczenia
- Rozluźniony mechanizm typów
  - zmienne nie muszą być jawnie deklarowane
- Dynamiczne wiązanie
  - odwołanie do obiektu ustalane w czasie wykonania
- Bezpieczny(?)
  - nie można zapisywać danych na twardy dysk i wykonywać połączeń sieciowych (poza serwerem, z którego pobrano skrypt)
  - pewne luki w bezpieczeństwie istnieją (file upload, cookie)
- Uwaga – tutaj mówimy tylko o JS klienckim (a nie serwerowym)

Zastosowania Javascript

- Dynamiczne generowanie treści dokumentu (document.write) - np. na podstawie typu i wersji przeglądarki
- Weryfikacja wartości pól formularza
- Obsługa "dynamicznych" formularzy
- "Triki"
  - podmiana grafiki (efekt "wciśnietego przycisku")
  - dynamiczne menu
  - elementy pop-up (tooltips, kalendarz)
  - wiele innych

JavaScript

Osadzanie skryptów:

```
<script language="JavaScript" type="text/javascript"
  charset="utf-8"> kod w JS </script >

<script … src="skrypt.js"/>
```

Przykład:

```
<html>
<head>
  <script language="JavaScript" type="text/javascript">
   <!-- document.write("Hello net.") -->
  </script>
</head>
<body>
bla bla
</body>
</html>
```

Podstawy JavaScript

- JS wewnątrz elementów HTML:
  - `<input type="button" VALUE="OK" onClick="myfunc('jakis tekst')">`
- JS wewnątrz atrybutów HTML:

```
<td width="&{barWidth};%" align="left">
```

- Funkcje JS

```
<script LANGUAGE="JavaScript">
  function square(i) {
     return i * i   }
</script>
```

JavaScript - język

- Wyrażenia arytmetyczno-logiczne b. podobne do C/C++ i Java
- Wbudowane funkcje - j.w.: matematyczne, stringowe, dodatkowo:
  - eval - obliczenie wartości wyrażenia stringowego,
  - parseInt, parseFloat - konwersja string na numeryczne
- Instrukcje:

```
break
comment
continue
for
for...in
```

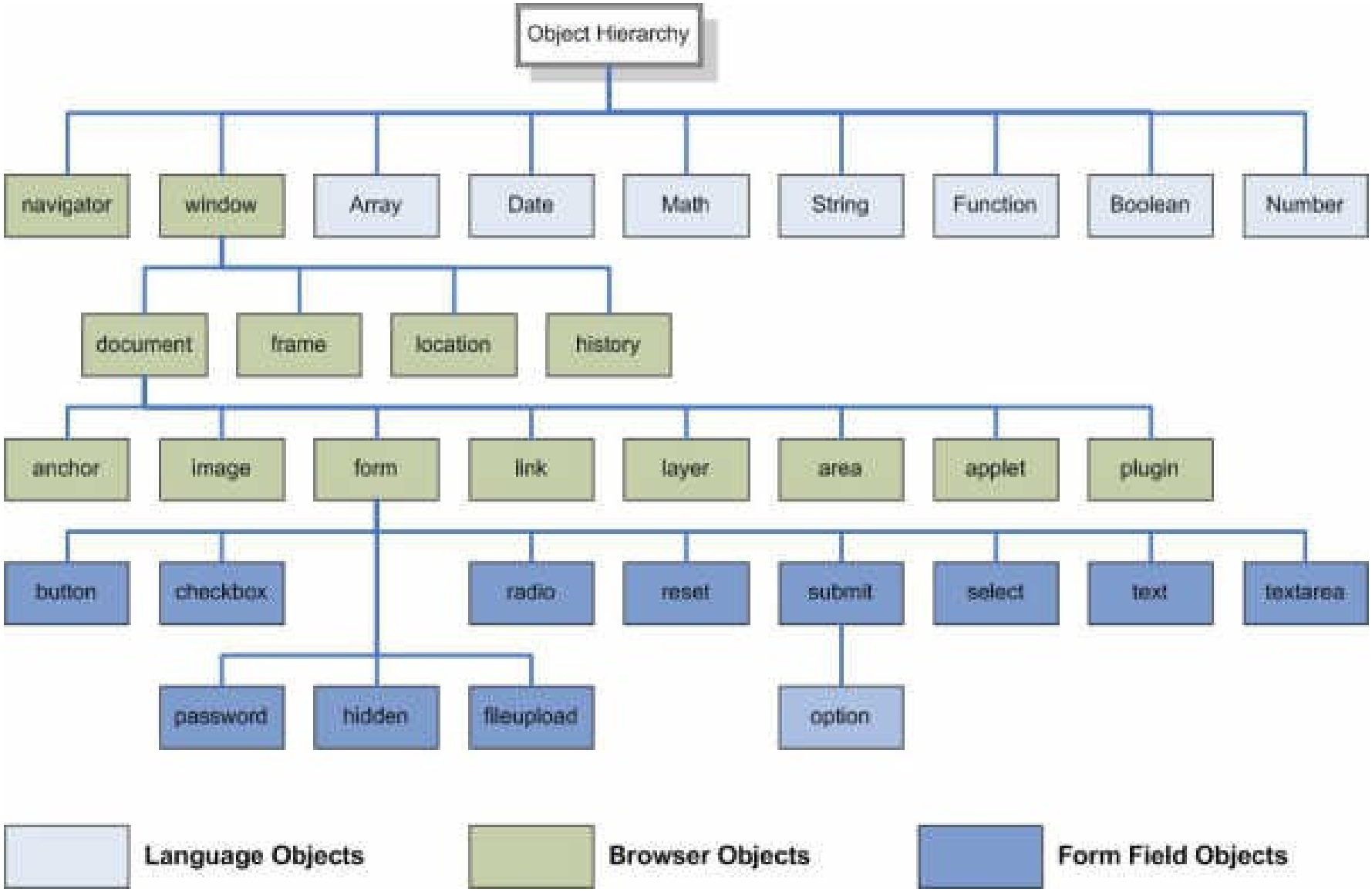
```
function
if...else
new
return
this
```

```
var
while
with
```

JavaScript - obiekty

- Możliwość tworzenia własnych obiektów poprzez konstruktory
- Główna zaleta - obiekty o typach wbudowanych odpowiadające strukturze dokumentu, w tym:
  - elementom dynamicznym
  - linkom
  - appletom Java
  - ... A także: przeglądarce, ramkom, historii odwiedzin, pluginom, cookies
- Atrybuty obiektów można odczytywać, a także (większość z nich) modyfikować

Hierarchia Obiektów JavaScript



JavaScript - dostęp do obiektów dokumentu

- Formularz:
  document.myform
- Checkbox:
  document.myform.c1
- Przycisk:
  document.myform.b1
- Odczyt wartości:
  document.myform.c1 .value
- Zmiana wartości:

document.myform.t1. value="abc"

```
<body>
<form name="myform"
action="f.cgi" method="get">
podaj wartość:
<input type="text" name="t1"
value="" size=20 >
zaznacz, jeśli chcesz:
<input type="checkbox" name="c1"
checked
onclick="update(this.form)">
option #1
<input type="button" name="b1"
value="naciśnij"
onclick="update(this.form)">
</form>
</body>
```

JavaScript - zdarzenia

- Zdarzenie związane są z akcjami użytkownika:
  - Click, DragDrop, Focus, KeyDown, KeyPress, KeyUp,
  - MouseOver, MouseMove, MouseDown, MouseUp,
  - Move, Resize - zmiany rozmiaru okna
  - Change, Select - edycja w obrębie pola tekstowego lub listy wyboru
  - Submit, Reset - przycisk

- … a także statusem dokumentu:
  - Abort - ładujące się obrazki
  - Focus, Blur - okno i formularze
  - Error
  - Load

JavaScript - zdarzenia

- Obsługa zdarzeń dla elementu HTML:

```
<tag eventHandler="JavaScript Code">
```

- Przykłady:

```
<input type="button" value="Calculate" onClick="compute(this.form)">

<input type="button" id="B1" value="Open Sesame!"
   onClick="window.open('doc.html', 'newWin')">

<a href=""
   onClick="document.form.action.value='customer_edit';document.form.
   submit();return false;">
```

- Zdarzenia mogą też być przechwytywane zanim dotrą do elementu przeznaczenia:
  - `captureEvents, releaseEvents, routeEvent, handleEvent`
  - `window.captureEvents(Event.CLICK);`

JavaScript - zaawansowane

- JS/DOM – (Document Object Model):
  - Pozwala na wygenerowanie całego dokumentu w kodzie JS
  - Konstruuje się obiekty, dołącza do nich atrybuty a następnie tworzy hierarchię obiekt macierzysty-obiekt potomny
  - Przydatne w zaawansowanych kontrolkach o silnie zmiennej zawartości, np. przy przeglądaniu struktur drzewiastych
- “Rich text” - komponent nakładany na textarea
  - Pozwala na formatowanie typu edytora tekstu - WYSIWYG

JavaScript DOM, przykłady

- Dostęp, przykłady:
- `<p id=”identyfikator paragrafu”>...</p>`
  - `document.getElementById('id paragrafu')`
  - `document.getElementsByTagName('p')[indexParagrafu]`
  - `window.document.body.childNodes[4]`
  - Atrybuty: `element.getAttribute('attribute_name')`
- Modyfikacja / budowa, przyklad:
  - ```
    var theNewParagraph = document.createElement('p');
    var theTextOfTheParagraph =
    document.createTextNode('Some content.');
    theNewParagraph.appendChild(theTextOfTheParagraph);
    document.getElementById('someElementId').appendChild(th
    eNewParagraph);
    ```
    ```

CSS
Cascading **S**tyle **S**heets

CSS - Podstawy

- Style definiują sposób formatowania elementów HTML
- Styl definiowany jest przez "**Style Sheet" - arkusz stylu**
- Style wprowadzono (W3C, HTML 4.0) aby rozwiązać problem niekompatybilności różnych przeglądarek
- Style (zwłaszcza zewnętrzne) upraszczają i przyśpieszają tworzenie stron - szczegóły wyglądu definiowane są centralnie, np. zmiana kroju i rozmiaru czcionki nagłówka H1 ma miejsce tylko w arkuszu stylu, a nie we wszystkich miejscach wszystkich dokumentów
- Definicje styli mogą być przechowywane w plikach (**.css)**
- Wiele definicji stylu kaskaduje (nakłada się) tworząc finalną definicję:
  - Styl domyślny przeglądarki
  - Style zewnętrzne (z plików)
  - Style wewnętrzne aktualnego dokumentu
  - style “inline”

CSS - Historia

- CSS1 – W3C 1996 (obecnie nieaktualne)
- CSS 2 – W3C 1998 (obecnie nieaktualne), razem z HTML4
- CSS 2.1 – W3C 1998 (obecnie nieaktualne)
- CSS 3 – W3C 2011/2012 – modularyzacja (np. css3-background, css3-box, css3-color, itd.)
- CSS4 – W3C – kilka modułów zdefiniowanych do tej pory (2015/2016)

CSS - selektor, atrybut

```
selector {property: value; ...}
Selektor, ... {deklaracja; ...}
```

Przykład:

```
body {color: black}
```

```
p {font-family: "sans serif"}
```

Cudzysłów niezbędny gdy spacja w wartości

Przykład - wiele atrybutów dla jednego selektora

```
p {text-align:center;color:red}
```

Przykład - definicja dla wielu selektorów:

```
h1,h2,h3,h4,h5,h6  {color: green}
```

Przykład - selektor klasy - różne style dla jednego elementu HTML:

```
p.right {text-align: right}
p.center {text-align: center}
```

```
<p class="right">
Wyrównanie do prawej.
</p>
```

CSS - selektory klasy i identyfikatora

Możemy też selektor zbudować z samej klasy:

```
.right {text-align:right}
```

Możemy tego stylu użyć we wszystkich elementach:

```
<h1 class="right">Nagłówek do prawej</h1>
<p class="right">Paragraf do prawej.</p>
```

Możemy też zdefiniować styl tylko dla wybranego elementu (jednego w dokumencie):

```
#wer123 {color: green}
<h1 id="wer345">Foo</h1>
```

Selektor identyfikatora odwołuje się do dokładnie jednego elementu

Ten styl pasuje tylko do elementu <P>:

```
P#wer123 {color: green}
```

Przykłady dostępnych atrybutów

Tło:

```
background-color
background-image
background-repeat
background-attachment
background-position
```

Czcionka:

```
font
font-family
font-size
font-stretch
font-style (normal, italic, …)
font-variant (small-caps, …)
font-weight (normal, bold, bolder,
        lighter, 100, 200, …, 900)
```

Tekst:

```
letter-spacing
text-align
text-decoration
        (none,
        underline,
        overline, …)
word-spacing
```

Wiele innych:

```
list-style
border-style
border-width
margin -bottom -top -right -left
        (np. akapitu)
pading -bottom -top -right -left
        (w komórce tabeli)
```

Wstawianie styli

- Styl zewnętrzny:

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

- Styl wewnętrzny:

```
<head>
<!--
<style type="text/css">
p {margin-left: 10px}
body {background-image: url("images/bg.gif")}
</style>
-->
</head>
```

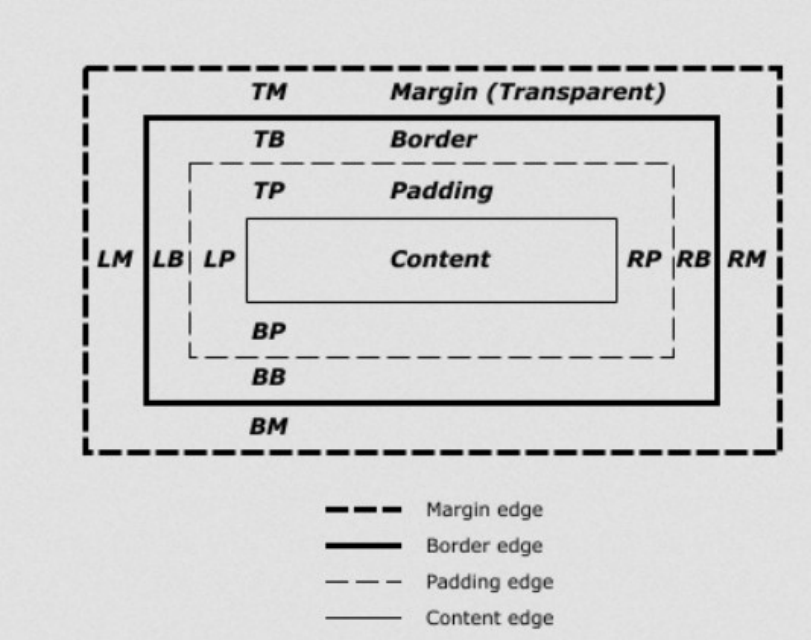
- Styl inline:

```
<h1 style="color:red;margin-left:1px;">Tytuł
 rozdziału. </h1>
```

CSS – wybrane konstrukcje

- Atrybut: `display` - określa wizualizację elementu, np.: `inline, block (<p>), list-item, table, table-cell, none, ...`
- Box model:
  (Wg. W3C)
  - Przykład:

```
mbox { width: 100px;
padding: 20px;
border: 20px;
margin: 5px; }
```



- Atrybut `position`:
  określa (względne)
  położenie elementu, dostępne opcje:
  - `static` - standardowo
  - `Relative` – względem domyślnej pozycji
  - `Fixed` – względem okna (np. zawsze na dole ekranu)
  - `absolute` – względem zewnętrznego elementu macierzystego

CSS - bibliografia

- Eric A. Meyer, "CSS. Kaskadowe arkusze stylów przewodnik encyklopedyczny", Helion (O'Reilly)

- http://www.w3schools.com/css/default.asp
- http://hotwired.lycos.com/webmonkey/reference/stylesheet_guide/