# Języki znaczników i komunikacji w Internecie

Dariusz Mikułowski

Instytut Informatyki UPH W Siedlcach

9 marca 2018

# 1 Plan wykładów

- znacznikowe języki komunikacji w Internecie;
- Możliwości JAvaScript i biblioteki jQuery;
- Mechanizmy zdalnego wywoływania procedur w Java z poziomu JavaScript: Direct Web Remoting;
- Tworzenie, parsowanie, analiza i prezentowanie dokumentów XML;
- Różne zastosowania i języki bazujące na XML;
- Technologie do tworzenia warstwy prezentacji w aplikacji webowej - podstawowe i zaawansowane możliwości technologii JavaServer Faces;
- Sposoby umiędzynarodawiania aplikacji w języku Java, JSP, Javascript;
- Alternatywne rozwiązania do tworzenia zaawansowanych aplikacji internetowych

# 2 Literatura

- Giulio Zambon, Michael Sekler, Beginning JSP, JSF & Tomcat Web Development: From Novice to Professional, APRESS, 2007
- Brett D. McLaughlin, Justin Edelson Java i XML. Wydanie III, Helion Gliwice 2007
- Dariusz Mikułowski XML w programowaniu aplikacji internetowych, Oficyna Wydawnicza Akademii Podlaskiej Siedlce 2009
- jQuery `http://jquery.com/`
- Direct Web Remoting `http://directwebremoting.org/dwr/index.html`
- Materiały do wykładów i laboratorium dostępne na stronie: `http://darek.ii.uph.edu.pl` w zakładce *informacje dla studentów* -> *Technologie programistyczne i internetowe* po podaniu nazwy użytkownika student i hasła student2018

# 3 Język HTML nieco historii

HTML (Hypertext Markup Language) Hyper-tekstowy język znaczników służy do porozumiewania się w Internecie oraz tworzenia zasobów internetowych - stron WWW. Został stworzony w Szwajcarii w laboratoriach badawczych CERN-u ok roku 1990. Za pomysłodawcę HTML uważa się fizyka Tim Berners-Lee, który W 1980 r. stworzył prototyp hypertekstowego systemu informacyjnego – ENQUIRE dla organizacji CERN. Wykorzystywano go do organizowania i udostępniania dokumentów związanych z badaniami naukowymi. Rewolucyjność pomysłu polegała na tym, że użytkownik, posługując się odnośnikami, mógł z jednej lokalizacji przeglądać dokumenty fizycznie znajdujące się w innych miejscach na świecie.
Potem w latach 1990/1993 powstawały kolejne szkice specyfikacji HTML. Na początku 1994 organizacja IETF wydzieliła grupę roboczą HTML Working Group, która w 1995 stworzyła pierwszą oficjalną specyfikację tego języka: HTML 2.0.

# 4 HTML nieco Historii

Standard HTML 3.0 został przedstawiony IETF przez Dave'a Raggeta i W3C w kwietniu 1995 r. Zawierał on znaczną część funkcjonalności poprzedniej wersji i nowe elementy takie jak: obsługa tabel, oblewanie tekstem obiektów, wyświetlanie skomplikowanych wyrażeń matematycznych
W 2000 r. HTML stał się międzynarodowym standardem (ISO/IEC 15445:2000). Ostatnia długo obowiązująca specyfikacja języka HTML to opublikowana w 1999 przez W3C – HTML 4.01. W kolejnych erratach wydawanych do niej zostały poprawione najważniejsze błędy.
Obecnie zalecana jest specyfikacja HTML 5. Organizacja W3C ogłosiła ją jako rekomendację 28 października 2014.

# 5 Tworzenie i interpretacja dokumentów HTML

HTML jest językiem interpretowanym, nie jest językiem programowania. Oznacza to, że program HTML jest wykonywany od razu bez kompilacji najczęściej przez przeglądarkę. Do edycji dokumentu HTML możemy użyć dowolnego edytora tekstowego - notatnik, wordpad.... Jeśli chcemy mieć automatyczną kontrolę nad poprawnością kodu, możemy wykorzystać specjalistyczne edytory takie jak: Adobe Dreamweaver, Bluefish, ezHTML, FrontPage, kED, Notepad++, Pajączek, Sublime Text.
Ponieważ interpretacja dokumentu HTML następuje w przeglądarkach, ważne jest, aby przy pisaniu kodu strony zwracać uwagę na jej dostępność dla różnych przeglądarek.

# 6 Znaczniki i atrybuty

Dokument HTML jest plikiem tekstowym w którym polecenia dla przeglądarki zapisujemy w formie znaczników objętych nawiasami <...>. Pomiędzy nimi wpisujemy tekst przeznaczony do wyświetlenia w oknie przeglądarki użytkownika. Większość znaczników istnieje w dwóch odmianach: otwierającej: <słowo kluczowe> i zamykającej </słowo kluczowe> Wiele znaczników posiada dodatkowe atrybuty (parametry). Atrybuty wpisywane są pomiędzy słowo kluczowe a kończący znak większości znacznik. Często można wpisać kilka atrybutów do jednego znacznika, oddzielając je spacjami. Każdy atrybut ma nazwę i wartość, którą umieszczamy w znakach cudzysłowu po wcześniejszym znaku równości.

# 7 Elementy dokumentu html

Najbardziej uniwersalnym i genialnym pomysłem jaki wprowadzono w językach hypertekstowych są odnośniki, które pozwalają na przeniesienie się użytkownika do innego miejsca dokumentu albo do innego dokumentu. Za ich pomysłodawce uważa się Tima Berns Li. Są tu znane dwa pojęcia: Universal Resource Identifier (URI) czyli identyfikator zasobu i Universal Resource Locator (URL) czyli miejsce zasobu. Przy czym pod adresem URL nie musi być wcale zasobu ale pełni on rolę unikalnej nazwy zasobu w całym świecie. W HTML możemy tworzyć też takie konstrukcje jak: tabele, ramki, grafiki, multimedia oraz elementy interaktywne czyli skrypty, które zachowują się jak programy a są wykonywane w przeglądarce.

# 8 Nadawanie stylu dokumentowi

Styl dla poszczególnych znaczników wraz ze znacznikami potomnymi (znajdującymi się w jego wnętrzu) można nadać na kilka sposobów:

- Utworzenie odsyłacza do zewnętrznego arkusza stylów i zaimportowanie go.
- Użycie pary znaczników: `<STYLE>...</STYLE>` zwykle w nagłówku dokumentu. Jego parametr `TYPE=text/css` określa język definicji arkuszy stylów. `<STYLE TYPE=text/css>`
- Przedefiniowanie znacznika w części `<BODY>` przy użyciu parametru `STYLE` np. w znaczniku `<P style="...">`

Najczęściej używa się zewnętrznych arkuszy stylów. Przy określaniu stylu trzeba pamiętać, że elementy potomne dziedziczą charakterystykę elementów nadrzędnych. Oznacza to, że jeśli zdefiniujemy np. tło, kolor, czcionkę dla elementu `<body>` To wszystkie elementy podrzędne np. `<p>, <li>` itd będą miały taki sam wygląd dopóki go nie przedefiniujemy.

# 9 Arkusze stylów

Arkusz dołączamy do pliku HTMl za pomocą deklaracji `<link rel>`. Najważniejsze atrybuty css to:

- Rodzaj czcionki - `FONT-FAMILY` wraz z nazwami ogólnymi pisanymi dokładnie tak, jak są wyświetlane w systemie
- Styl czcionki - `FONT-STYLE (italic, normal)`.
- Grubość czcionki - `FONT-WEIGH` (np. `bold, bolder, lighter, normal`, lub liczby od 100 do 900, 500 to czcionka normalna)
- Wielkość czcionki - `FONT-SIZE` (np. `small, medium, large, larger, x-large, xx-large`)
- Wyrównanie tekstu - `TEXT-ALIGN: left, right, center, justify`).
- wcięcie pierwszego wiersza akapitu - `TEXT-INDENT`;
- Dekoracja tekstu - `TEXT-DECORATION (underline, line-through, overline, blink)`.
- Wielkość liter - `TEXT-TRANSFORM (capitalize, uppercase, lowercase`
- Indeks górny lub dolny - `VERTICAL-ALIGN` (sub, super.
- kolor tekstu, linii i krawędzi - `COLOR` (kolory określamy za pomocą angielskich nazw potocznych lub za pomocą parametru RGB z zakresu od 0 do 255 (w systemie dziesiętnym) lub szesnastkowo z poprzedzającym znakiem #).
- Kolor tła - `BACKGROUND-COLOR` (nazwa koloru lub RGB lub słowa TRANSPARENT).
- Obrazek w tle - `BACKGROUND-IMAGE` (argument w postaci adresu URL).
- Powtarzanie obrazka w tle - `BACKGROUND-REPEAT` repeat, repeat-x, repeat-y, norepead).
- Przewijanie tła - `BACKGROUND-ATTACHMENT` (parametry scroll, fixed).
- Pozycja obrazka w tle - `BACKGROUND-POSITION` (parametry top, center, bottom, left).
- obramowanie - `BOX` rozmiar obramowania

Dowolnemu znacznikowi html można nadać unikalny identyfikator a następnie dla niego zdefiniować styl w arkuszu css. Idantyfikator nadajemy dodając do znacznika atrybut id np.

```
<h1 id="mojnaglowek">tytuł</h1>
```

Następnie dla niego można zdefiniować styl css:

```
#mojnaglowek {FONT-SIZE:30pt; COLOR:green}
```

Użycie:

```
<h1>uzycie bez identyfikatora

<H1 id="mojnaglowek">Będzie zielony i powiększony</h1>
```

Aby jeden styl zastosować do kilku znaczników trzeba go zdefiniować jako klasę.

```
.czerw-obram {border-color: red; border-width:10px;
  padding: 5px}
```

Następnie taki styl można zastosować jako klasę do kilku elementów:

```
<H2 class ="czerw-obram"> ... </H2>

<h4 class="czerw-obram"> ... </H4>
```

# 12 Wyróżniane fragmenty tekstu

Można zdefiniować styl dla wyróżnionego tekstu w obrębie tylko określonego znacznika. Przykładowa definicja tekstu pochylonego tylko w obrębie akapitu:

```
P EM {FONT-STYLE:normal; FONT-WEIGHT:700;COLOR:GREEN}
```

Znacznik <DIV> oddziela fragment strony w którym można zastosować format specjalny. Można też do niego zastosować styl w formie klasy. lub określić styl przez parametr <STYLE>.

```
<DIV ALIGN=center> ... </DIV>
```

Znacznik <SPAN> wyróżnia fragmenty wewnątrz tekstu. Formatowanie odbywa się wyłącznie przez arkusze stylów,

```
Ten kolor jest <SPAN STYLE=color:red>czerwony</SPAN>
```

Atrybut FLOAT wyrównuje grafikę do lewej lub prawej krawędzi okna. łącznie z atrybutami MARGIN-LEFT, MARGIN-RIGHT można spowodować efekt pływania obrazu nad tekstem). Atrybut CLEAR określa potrzebne dla tekstu puste miejsce. Możliwe są cztery ustawienia:

1. none – tekst wyświetlany obok lub między pływającymi grafikami,
2. left lub right – pozostanie wolne miejsce z lewej lub prawej strony grafiki a tekst będzie kontynuowany w następnym wierszu pod pływającym obrazem,
3. both – tekst zostanie umieszczony tam, gdzie po obu stronach nie będzie żadnych elementów pływających.

# 13 Tworzenie formularzy

Formularze przeznaczone do wpisywania danych przez użytkownika tworzone są przy pomocy znacznika `<form>` Jego atrybut `method` wskazuje na metodę protokołu HTTP, jaka będzie użyta do wysłania formularza. Atrybut `ACTION` wskazuje na procedurę lub program obsługujący przesłanie formularza. W najprostszym przypadku może to yć wysłanie danych e-mailem. W jego środku używamy znaczników, które generują różne kontrolki do interakcji z użytkownikiem. są to takie znaczniki jak:

- `<INPUT>` - pole przeznaczone na wprowadzenie tekstu lub kliknięcie. jego atrybut `TYPE` określa rodzaj. Może to być
  - text - jednowierszowe pole tekstowe
  - reset - przycisk do wyczyszczenia wpisanych dotychczas danych
  - radioButton - przycisk opcji do wyboru
  - checkbox - pole wyboru do zaznaczenia
  - submit - przycisk powodujący wysłanie formularza
  - IMAGE - niestandardowy przycisk w formie obrazka. w dodatkowym parametrze SRC podaje się nazwę pliku obrazka.
- `<TEXTAREA>` - wielowierszowe pole tekstowe
- `<SELECT>` - lista opcji do wyboru
- `<OPTION>` - pojedyncza opcja z listy do wyboru
- `<BUTTON>` - przycisk

# 14 Język JavaScript

Użycie samego języka HTML bardzo szybko okazało się niewygodne. Dlatego do implementowania takich operacji jak walidacja danych w formularzach, ruchome animacje, i innych elementów strony przyciągających użytkownika wykorzystuje się język JavaScript. Kod programu JavaScript możemy umieszczać bezpośrednio w kodzie stron lub jako osobne pliki o rozszerzeniu js. W każdym przypadku będzie się on wykonywać w przeglądarce użytkownika. Podstawową zaletą tego rozwiązania jest szybkość działania programu.

# 15 JavaScript - obiekt Window

JavaScript posiada wszystkie podstawowe konstrukcje każdego języka programowania takie jak: zmienne liczbowe i znakowe, tablice, listy, pętle, konstrukcje warunkowe itd. Dokument HTML jest dostępny z poziomu konstrukcji programu w JavaScript jako struktura drzewa. Ten model nazywany jest DOM (ang. **Document Object Model**). Dzięki jego zastosowaniu mamy szybki dostęp do każdego dowolnie złożonego lub prostego elementu dokumentu takiego jak: formularz, pole formularza, zawartość paragrafu, wartość atrybutu znacznika HTML itd.
Głównym obiektem reprezentującym okno wyświetlane w przeglądarce jest objekt `Window` Jeśli strona jest zbudowana z ramek, tworzony jest główny obiekt Window dla całej strony oraz obiekty Window dla każdej ramki.
Najważniejsze właściwości obiektu window:

- `closed` - zwraca wartość logiczną True, gdy okienko jest zamknięte,
- `document` - zwraca obiekt dokumentu załadowanego w oknie.
- `frames` - zwraca listę ramek (podokien) w oknie,
- `parent` - przechowuje wskaźnik do okna nadrzędnego.
- `outerHeight i outerWidth` - przechowują całą szerokość i wysokość okna łącznie z paskami przewijania i statusu.
- `pageXOffset` i pageYOffset - przechowują ilość pikseli o którą

# 16 JavaScript - obiekt window

Najważniejsze metody obiektu `Window`:

- `alert()` - wyświetla okienko dialogowe z informacją i przyciskiem `OK`
- `focus() i blur()` odpowiednio aktywują kursor na oknie lub deaktywują go z aktualnego okna.
- `clearInterval()` i `clearTimeout()` - kasują wcześniej ustawiony przedział czasowy lub stoper, których używa się do wywoływania funkcji w określonym czasie.
- `close()` - powoduje zamknięcie okna.
- `confirm()` - Wyświetla
- `moveBy()` i `moveTo()` - przesuwają okienko do wybranej pozycji lub o wybraną ilość pikseli.
- `open()` - powoduje otwarcie nowego okna.
- `prompt()` - wyświetla okienko dialogowe z polem do wpisania informacji i przyciskami.
- `resizeBy()` i `resizeTo()` - zmieniają rozmiar okna.
- `scroll(), scrollBy()` i `scrollTo()` - przewijają zawartość okna.

# 17 JavaScript - objekt Document

Dokument HTML wyświetlany w oknie jest reprezentowany jako obiekt `Document` Ma on strukturę drzewa. Wszystkie jego elementy takie jak znaczniki html, atrybuty, zawartość tekstowa znaczników i komentarze są węzłami, które można przeglądać i modyfikować. Sam dokument jest głównym węzłem typu dokument. Najważniejsze właściwości i metody obiektu `Document`:

- `document.anchors` - zwraca listę kotwic w dokumencie.
- `document.applets`, `document.forms`, `document.images` i `document.links` - zwracają odpowiednio listy apletów, formularzy, grafik lub łączy w dokumencie.
- `document.body` - zwraca element `body`
- `document.open()` i `document.close()` - odpowiednio otwierają lub zamykają strumień wyjściowy na który wypisywana jest zawartość html.
- `document.cookie` - zwraca listę ciasteczek (par: nazwa, wartość).
- `document.createAttribute()`, `document.createComment()`, `document.createElement()` i `document.createTextNode()` - tworzą odpowiednio węzły typu atrybut,
- `document.documentURI` - ustawia lub zwraca adres dokumentu.
- `document.domain` - zwraca adres domeny w której znajduje się dokument.
- document.getElementById()+, `document.getElementsByName()`, `document.getElementsByTagName()` - zwracają odpowiednio: węzeł o podanym identyfikatorze, listę węzłów o podanej nazwie lub nazwie znacznika html.
- `document.importNode()` - importują węzeł z innego dokumentu do aktualnego.
- `document.normalize()` i `document.normalizeDocument()` - służą do normalizacji dokumentu. Polega ona na usunięciu pustych węzłów i połączeniu węzłów przecinających się.
- `document.readyState` - zwraca wartość True gdy dokument jest całkowicie załadowany do przeglądarki.
- `document.title` - ustawia lub zwraca

# 18 JavaScript - przykład

Prosty przykład użycia obiektów Window i document.
Małe zdjęcie po kliknięciu powiększy się w nowym oknie. powtórne kliknięcie w to okno spowoduje jego zamknięcie i powrót do wyświetlonego małego zdjęcia.

```
3          width="100 heigh=150"
4          onclick="javascript:powieksz('images/mojefoto.jpg')">
5  <script language="javascript">
6      function powieksz(f) {
7      okno = window.open('', 'okno', 'toolbar=no,
8              scrollbars = no, left = 0, top = 0,
9              resizable = no, width = 500, height = 600; ');
10             okno.document.write('<html><head>
11             <title>Zdjęcie</title> < /head>');
12                     okno.document.write('<body>');
13             okno.document.write('<input type=image src=' + f);
14             okno.document.write(' onClick=javascript:window.close()>');
15             okno.document.write('</body></html>');
16             }</script>
17
```

# 19 Standard DHTML

Nazwa DHTML to skrót od (Dynamic HyperText Markup Language) co oznacza Dynamiczny hypertekstowy język znaczników. Jest to umowna nazwa dla technik służących do dynamicznej zmiany treści, wyglądu, zachowania dokumentu HTML umożliwiających interakcję strony www z użytkownikiem i stosowanie efektów wizualnych. Powstał jako rozwinięcie standardowego języka HTML, który prezentował zawartość stron internetowych w sposób statyczny.

DHTML nie jest jednolitym standardem ale zbiorem różnych technik, które pozwalają na tworzenie takich elementów strony jak:

- rozwijane menu,
- powiększające się miniaturki,
- wyświetlane na żądanie dodatkowe sekcje tekstu.
- interaktywne animacje

Za elementy DHTML uważa się różne technologie działające po stronie przeglądarki. są to:

- język XHTML,
- model dokumentu DOM,
- arkusze stylów CSS,
- język grafiki SVG
- JavaScript
- php

# 20 Język XHTML

XHTML (Extensible HyperText Markup Language) to rozszerzalny język znaczników hypertekstowych do tworzenia stron www. jest on uogólnieniem HTML 4.01 do postaci XML. Najnowszą rekomendacją XHTML jest opublikowana 31 maja 2001 r. i uaktualniona jako wydanie drugie 23 listopada 2010specyfikacja XHTML 1.1. Jest ona wersją zalecaną przez W3C, ponieważ prace nad standardem XHTML 2.0 zostały zarzucone. Alternatywnie wobec XHTML2 również w ramach grupy roboczej W3C był rozwijany HTML 5,

# 21 Język XHTML

W odróżnieniu od HTML dokumenty XHTML muszą zachowywać format wymagany w XML. dzięki temu można je łatwo generować z innych dokumentów XML np. przy pomocy transformacji XSLT, lub programowo z modelu DOM. Daje to też możliwości łatwiejszego wstawiania do strony innych rodzajów zawartości np. formuł matematycznych w języku MathML czy grafiki SVG. Dokument XHTML powinien spełniać następujące wymagania:

musi rozpoczynać się od deklaracji XML:

```
<?xml version="1.0" encoding="utf-8"?>
```

Jego główny Element `<html>` musi zawierać atrybut xmlns określający przestrzeń nazw dla języka XHTML:

```
XHTML: http://www.w3.org/1999/xhtml
```

Każdy Znacznik otwierający powinien mieć znacznik zamykający:

```
<li> ... </li>)
```

Puste elementy muszą także być zamykane:
zamiast `<br>` musi być `<br/>`, albo
`<br></br>`

Elementy muszą być zagnieżdżane w odpowiedni sposób:
zamiast
`<p>Tekst z <em>wyróżnieniem</p></em>`
powinno być
`<p>Tekst z <em>wyróżnieniem</em></p>`

Nazwy elementów i atrybutów XHTML muszą być pisane małymi literami;.

Wszystkie wartości atrybutów muszą być ujęte w cudzysłów podwójny lub apostrof:
`<td rowspan="3">`, albo
`<td rowspan='3'>`

Przykład strony w XHTML:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css" href="style.css"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl">
    <head>
        <title>Przykład dokumentu zgodnego z XHTML 1.1</title>
    </head>
    <body>
        <div>To jest przykład.</div>
    </body>
</html>
```

# 23 HTML 5

HTML 5 jest uniwersalnym językiem do tworzenia i prezentowania stron www. Jest on rozwinięciem wcześniejszych języków - HTML w wersji 4 oraz XHTML.
HTML 5 Został opracowany przez konsorcjum W3C w ramach prac grupy roboczej Web Hypertext Application Technology Working Group (w skrócie WHATWG). Od października 2014 jest oficjalną rekomendacją konsorcjum W3C.

# 24 Innowacje w HTML5

HTML 5 w porównaniu do wcześniejszych wersji html wprowadza bardzo wiele nowych elementów ułatwiających tworzenie stron internetowych. Wiele funkcji, które w html4 trzeba było implementować np. przy użyciu PHP zostało wbudowanych w sam język HTML. Ponadto w wersji 5 uproszczono niektóre deklaracje i składnie. Najważniejsze wprowadzone innowacje to:

Łatwiejsza składnia znacznika <doctype>:

```
<!DOCTYPE HTML>
```

Łatwiejsze ustalanie kodowania strony:

```
<meta charset="UTF-8"/>
```

Łatwiejsza deklaracja arkuszy stylów:

```
<link rel="stylesheet" href="style.css" />
```

Łatwiejszy sposób wstawiania skryptów:

```
<script src="file.js"></script>
```

# 25 Warstwa semantycznaHTML 5

W HTML 5 wprowadzono kilka znaczników, które nie wpływają bezpośrednio na wygląd strony, ale definiują jej strukturę. Jest to potrzebne dla robotów i wyszukiwarek, które mogą łatwiej indeksować poszczególne fragmenty zawartości strony. Takie znaczniki są nazywane elementami semantycznymi, gdyż mają związek ze znaczeniem a nie wyglądem fragmentów witryny. Mają też one zastosowanie np. dla technologii udostępniających strony internetowe takie jak Screenreadery dla niewidomych ponieważ określają jak dany fragment strony ma być odczytywany głosem. Najważniejsze znaczniki semantyczne w HTML5:

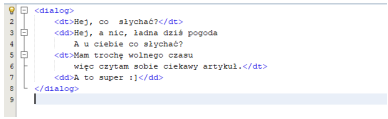
- 
- `<header>` - kontener przechowujący informacje wprowadzające (np. tytuł, logo, wyszukiwarka, nawigacja)
- `<footer>` - kontener przechowujący informacje umieszczane w stopce (np. autor, prawa autorskie)
- `<nav>` - zawiera linki nawigacyjne tworzące menu
- `<section>` - grupuje tematycznie powiązaną zawartość np.

Dalsze znaczniki semantyczne:

- `<article>` - zawiera główną treść przeznaczoną do przetworzenia (np. przez kanał RSS).
- `<aside>` - kontener przechowujący mniej ważne treści np. panel boczny.
- `<dialog>` - służy do wyróżnienia rozmowy:
- `<figure>` - służy do umieszczenia i podpisania obiektu graficznego lub multimedialnego
- `<hgroup>` - kontener dla kilku nagłówków, np dla tytułu i podtytułu

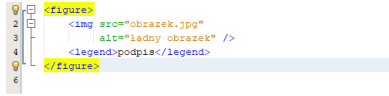
# 27 Przykłady znaczników semantycznych

Użycie znacznika dialog:



```
<dialog>
    <dt>Hej, co  słychać?</dt>
    <dd>Hej, a nic, ładna dziś pogoda
        A u ciebie co słychać?
    <dt>Mam trochę wolnego czasu
        więc czytam sobie ciekawy artykuł.</dt>
    <dd>A to super :)</dd>
</dialog>
```

Użycie znacznika figure:



```
<figure>
    <img src="obrazek.jpg"
        alt="ładny obrazek" />
    <legend>podpis</legend>
</figure>
```

# 28 Szablon strony HTML 5

Przykładowy szablon strony z wykorzystaniem znaczników semantycznych:

```
            <nav>
                <ul>
                    <li><a href="#">Kontakt</a></li>
                    <li><a href="#">Nieco więcej o mnie</a></li>
                </ul>
            </nav>
        </header>
        ....
</html>
```

```
            <nav><a href="">poprzedni wpis</a>
                <a href="">następny wpis</a></nav>
        </section>
        <footer>
            <p>Copyright all reserved.</p>
        </footer>
    </body>
</html>
```

# 29 Inne znaczniki

podświetlenie, np. wyszukiwanej frazy w tekście:

```
<marks>Podświetlony tekst</marks>
```

oznaczenie czasu:

```
<time datetime="2015-02-11">11 lutego 2015</time>
```

wartość numeryczna:

```
<meter min="0" max="100" value="55">55</meter>
```

pasek postępu:

```
<progress min="0" max="100" value="55"> </progress>
```

# 30 Tworzenie formularzy.

Wiele operacji takich jak (np. walidacja danych) w formularzach, które trzeba było implementować za pomocą JavaScriptu, w HTML 5 można zrealizować bezpośrednio przy pomocy odpowiednich znaczników.

Ponadto HTML5 udostępnia wiele nowych typów pól. Najważniejsze nowe znaczniki i atrybuty to:

`Autofocus` - ustawia wskaźnik na konkretnym polu formularza:

```
<input name="field" type="text"
  autofocus="autofocus" />
```

`required` - wskazuje, że wypełnienie pola jest wymagane, w razie nie wpisania danych nie będzie reakcji po kliknięciu na przycisk): zatwierdzający formularz:

```
<input name="field" type="text"
  required="required" />
```

nowy typ pola `number`, który wymaga wpisania liczby w określonym zakresie:

```
<input name="number" type="number"
  min="0" max="99" required="required" />
```

nowy typ pola `url` - służy do wpisania adresu internetowego:

```
<input name="url" type="url" />
```

typ pola `email` - służy do wpisania adresu e-mail:

```
<input name="email" type="email" />
```

atrybut `pattern` - służy do stworzenia własnego wyrażenia regularnego do walidacji danych:

```
<input name="telefon"
  pattern="\(\+\d\d\)-\d\d\d-\d\d\d-\d\d\d"
  title="wpisz telefon w formacie:
  (+00)-000-000-000" type="text" />
```

nowe typy pól dla daty i czasu: `date` i `time`:

```
<input name="date" type="date" />

<input name="datetime" type="datetime-local" />

<input name="time" type="time" />
```

# 31 Inne nowe konstrukcje HTML 5

Html 5 wprowadza także nowy element jakim jest suwak.

```
<form>
<input type="range" name="range" />
Value: <output onformchange="this.value = form.range.value"></output>
</form>
```

Nowa wersja HTML umożliwia również wykonanie obliczeń matematycznych:

```
<form>
(a)<input name="a" step="any" value="0" type="number" /> *
(b)<input name="b" step="any" value="0" type="number" />
<br>a+b = <output name="c" onformchange="value = a.value \item b.value">0</output> (c)
<br>c \item 2 = <output name="d" onformchange="value = c.value \item 2">0</output> (d)
<br>d - a = <output name="result" onformchange="value = d.value { a.value">0</output>
</form>
```

# 32 Listy rozwijane

W HTML 5 listy rozwijane a dokładniej pola edycyjne kombi do uzupełnienia możemy tworzyć przy pomocy nowego znacznika `<datalist>` Znacznik ,`<select>` oczywiście nadal obowiązuje.

```
2   <input list="browsers" />
3   <datalist id="browsers">
4       <option value="Safari">Safari</option>
5       <option value="Internet Explorer">Internet Explorer</option>
6       <option value="Opera">Opera</option>
7       <option value="Firefox">Firefox</option>
8   </datalist>
9
```

# 33 Dźwięk i wideo

Obsługa dźwięku i wideo wymaga odpowiednich kodeków. Problemem jest to że przeglądarki korzystają z różnych kodeków. Można to obejść odtwarzając wideo i audio jako obiekty flash.

osadzanie wideo:

```
            type="video/ogg" />
    <object type="application/x-shockwave-flash"
            width="360" height="240"
            data="player.swf?file=movie.mp4">
        <param name="movie"
            value="player.swf?file=movie.mp4" />
        <a href="movie.mp4">Pobierz plik</a>
    </object>
</video>
```

Osadzanie audio:

```
<audio controls="controls">
    <source src="audio.ogg"
            type="audio/ogg" />
    <source src="audio.mp3"
            type="audio/mpeg" />
    <object type="application/x-shockwave-flash"
            data="player.swf?soundFile=audio.mp3">
        <param name="movie"
            value="player.swf?soundFile=audio .mp3" />
        <a href="audio.mp3">Pobierz plik</a>
    </object>
</audio>
```

znacznik `<object>` zapewnia odtworzenie pliku z wykorzystaniem technologii Flash, w przypadku gdy przeglądarka nie obsługuje znaczników ¡video¿ i ¡audio¿. Ponadto warto dodawać typ mime – dzięki temu przeglądarka wie, który plik jest w stanie obsłużyć.

# 34 Rysowanie

Całkowicie nowym elementem wprowadzonym w HTML5 – jest `Canvas` Jest to prostokątny obszar grafiki rastrowej, w którym można rysować za pomocą JavaScript. Funkcjonalności Canvas:

- rysowanie prostokątów i ścieżek (linie, łuki, krzywe Beziera lub krzywe drugiego stopnia),
- wypełnianie figur kolorem (z kanałem alfa - przezroczystość), wzorem lub gradientem (liniowym lub radialnym)
- rysowanie napisów
- przekształcenia (przesuwanie, skalowanie, obracanie, pochylanie itp.)
- osadzanie obrazów rastrowych (PNG, JPEG, GIF)
- cieniowanie
- użycie stylów kompozycji które określają to, jak nowe treści są rysowane na istniejących składnikach canvas

Element Canvas daje wiele możliwości, jednakże jest dość trudny w używaniu z racji jego niskopoziomowości. Na szczęście powstały biblioteki JavaScript ułatwiające to zadanie takie jak np.: LiquidCanvas czy Rgraph.

# 35 Przykład użycia znacznika Canvas

```
<canvas id="myCanvas" width="360" height="240">
    Tekst wyświetlany gdy przeglądarka nie obsługuje canvas.
</canvas>
<script>
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    context.fillStyle = "blue";
    context.fillRect(50, 25, 150, 100);
</script>
```

# 36 Przechowywanie danych po stronie klienta

Dotychczas jedyną formą przechowywania danych po stronie klienta były ciasteczka (ang. cookies). Jednakże mogą one przechowywać tylko niewielkie porcje danych. W HTML5 istnieje nowy mechanizm nazywany web storage. Jest to API dla programistów JavaScript, które umożliwia zapis danych po stronie klienta. Istnieją dwa typy storage – localStorage przechowuje dane bez limitu czasowego. i sessionStorage przechowuje dane tylko podczas istnienia sesji

# 37 Przechowywanie danych po stronie klienta

Przykłady użycia mechanizmu web storage:

Local storage:

```
<script type="text/javascript">
    if (localStorage.pagecount)
    {
        localStorage.pagecount
                = Number(localStorage.pagecount) + 1;
    } else
    {
        localStorage.pagecount = 1;
    }
    document.write("liczba odwiedzin: "
            + localStorage.pagecount + time(s) + ".");
</script>
```

Session storage:

```
<script type="text/javascript">
    if (sessionStorage.pagecount)
    {
        sessionStorage.pagecount
                = Number(sessionStorage.pagecount) + 1;
    } else
    {
        sessionStorage.pagecount = 1;
    }
    document.write("Liczba aktualnie dostępnych użytkownikow "
            + sessionStorage.pagecount + time(s));
</script>
```

# 38 Web SQL Database

Inną możliwością przechowywania danych jest Web SQL Database. Jest to mechanizm dostępu do SQLowej bazy danych po stronie przeglądarki. Jest on jednak uważany za niebespieczny i w zasadzie jest wspierany tylko w przeglądarkach Microsoftu. Przykład użycia:

```
        return openDatabase('documents', '1.0',
                    'Offline document storage', 5 * 1024 * 1024,
                    function (db) {
                        db.changeVersion('', '1.0',
                                function (t) {
                                    t.executeSql('CREATE TABLE docids (id, name)');
                                }, error);
                    });
    }
    function showDocCount(db, span) {
    db.readTransaction(function (t) {
    t.executeSql('SELECT COUNT(*) AS c FROM docids', [],
                function (t, r) {
                span.textContent = r.rows[0].c;
                },
</script>
```

```
    prepareDatabase(function(db) {
// got database
    var span = document.getElementById('doc-count');
    showDocCount(db, span);
    },
            function (e) {
// error getting database
            alert(e.message);
            });
</script>
```

# 39 Komunikacja pomiędzy różnymi stronami

HTML 5 udostępnia mechanizm nazywany Cross Document Messaging pozwala on na przesyłanie komunikatów pomiędzy różnymi nawet odległymi dokumentami. Realizacja tej funkcjonalności odbywa się poprzez nową metodę obiektu window o nazwie postMessage. Oczekuje ona jednego argumentu typu string, w którym jest przechowywana wysyłana wiadomość. Działanie metody polega na stworzeniu zdarzenia, które zostanie wywołane na obiekcie document okna, do którego wysyłamy wiadomość.

# 40 Rządania XML

Innym mechanizmem są rządania XML. XMLHttpRequest w wersji 2 umożliwia wykonywanie żądań spoza źródła strony. Dodatkowo udostępnia on znacznie bardziej rozbudowany mechanizm zdarzeń informujących o postępach w transmisji. Możemy wykorzystać takie zdarzenia jak:

- readystatechange - gotowy do transmisji
- loadstart, - rospoczęte pobieranie pliku
- progress, - postęp pobierania
- load - trwa pobieranie
- loadend. - pobieranie zakończone
- abort, - zatrzymanie pobierania
- error, - błąd pobierania

# 41 WebSocket

WebSocket jest kolejnym nowym mechanizmem w HTML 5 o działaniu podobnym do tradycyjnych Socketów. definiuje on dwukierunkowy kanał komunikacyjny przeznaczony dla sterowanych zdarzeniami aplikacji czasu rzeczywistego. Jeśli serwer WWW implementuje ten standard, wówczas w dowolnym momencie istniejące połączenie HTTP można łatwo przekształcić w połączenie WebSocket. Wystarczy, że przeglądarka wyśle do serwera odpowiedni komunikat HTTP:

Komunikat wysłany z przeglądarki:

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4@1 46546xW
Origin: http://example.com
[8-byte security key]
```

Odpowiedź Serwera:

```
GET /demo HTTP/1.1
Host: example.com
Connection: Upgrade
Sec-WebSocket-Key2: 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol: sample
Upgrade: WebSocket
Sec-WebSocket-Key1: 4@1 46546xW
Origin: http://example.com
[8-byte security key]
```

Od tej pory komunikaty mogą być wymieniane asynchronicznie w obie strony.

Przykład implementacji połączenia Web Socket:

```
<script>
    w = new WebSocket("ws://example.com");
    w.onopen = function () {
        console.info("open");
    }
    w.onmessage = function(e) { console.info(e.data); }
    w.onclose = function() { console.info("closed"); }
    ...
            w.send("Komunikat");
            ...
            w.close();
</script>
```

# 43 Mechanizm Web Workers

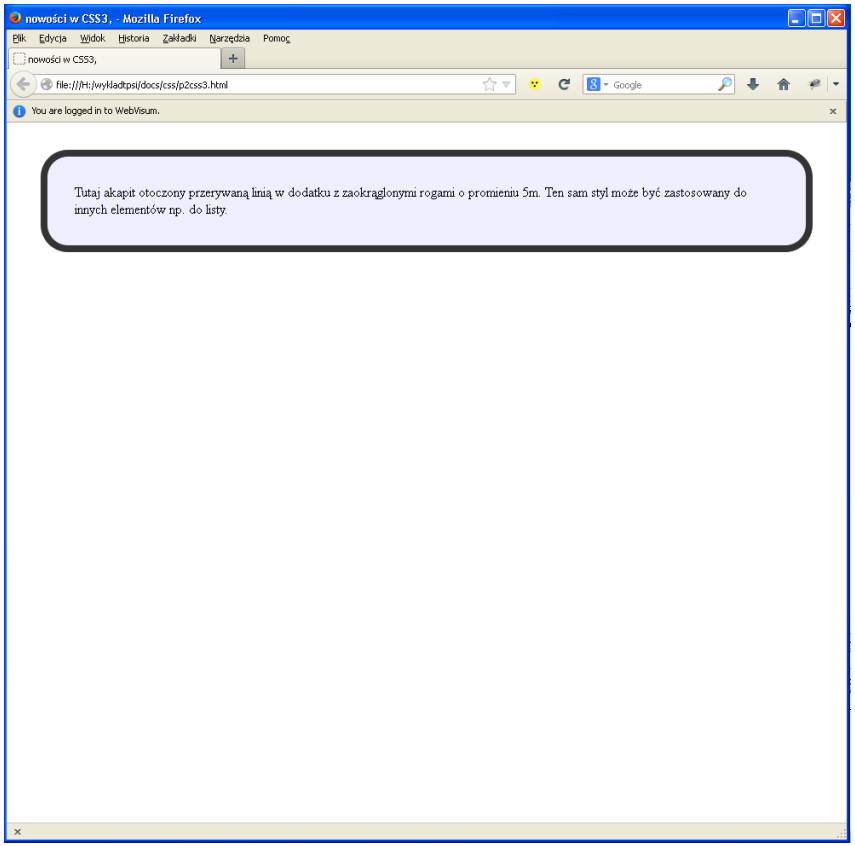
Web Workers umożliwia przetwarzanie danych w tle – na osobnych wątkach systemu operacyjnego. Powołanie nowych wątków w kodzie JavaScriptpolega na utworzeniu nowego obiektu Worker i wskazaniu pliku JS do wykonania:

```
worker = new Worker("myWorker.js");
```

Wraz z użyciem HTML 5 wykorzystuje się także nową wersję CSS v3. Posiada ona bardzo przydatne udogodnienia ułatwiające ustalanie stylu i wyglądu różnych elementów strony. Najbardziej przydatne funkcje to:

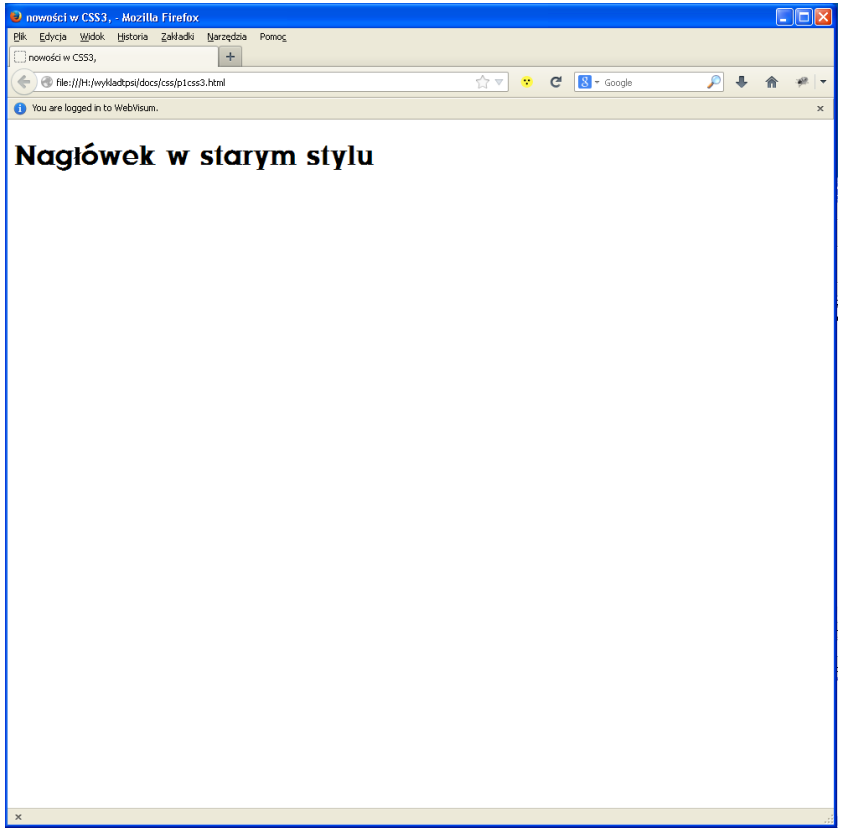
Otaczanie akapitów i innych elementów ramką z zaokrąglonymi rogami. przy pomocy atrybutu `border-radius`.

```
                -moz-border-radius: 2em;
                -webkit-border-radius: 2em;
                border-radius: 2em;
            </style>
        </head> <body>
            <p>Tutaj akapit otoczony przerywaną linią
                w dodatku z zaokrąglonymi rogami
                o promieniu 5m.</p>
        </body> </html>
```



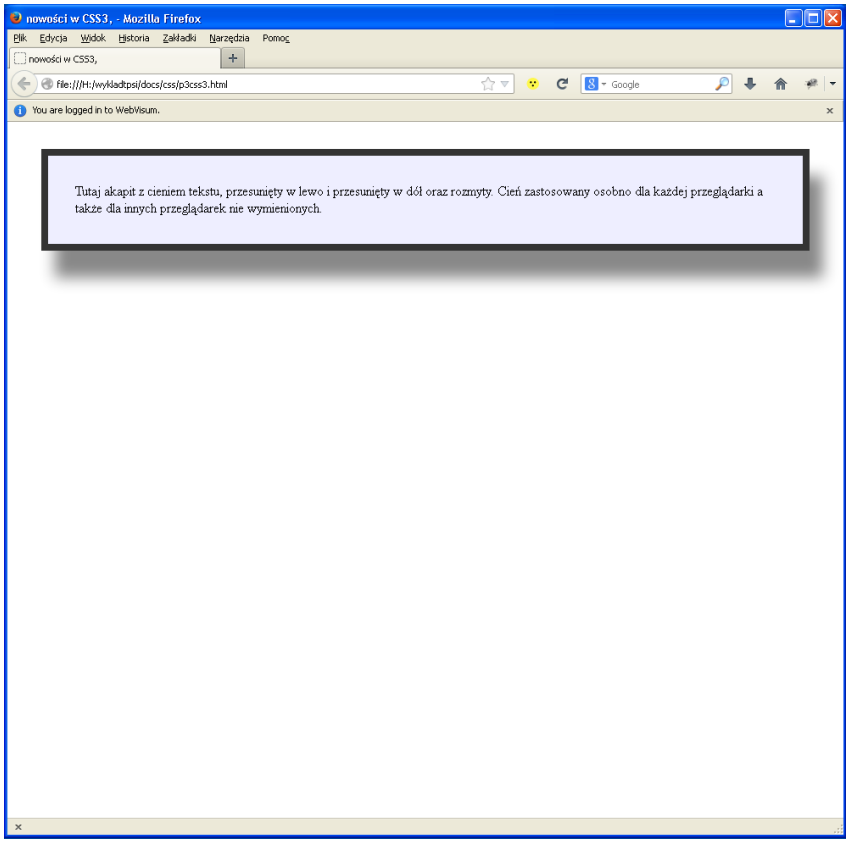
Użycie własnej czcionki przechowywanej w pliku `ttf`:

```
        <meta charset="Windows-1250"/>
        <title>nowości w CSS3,</title>
        <style>
            @font-face{
                font-family: 'mojaStaraCzcionka';
                src: url('StaraCzcionka.ttf')
                    format('truetype');
            }
            h1{
                font-family:mojaStaraCzcionka;
            }
        </style> </head> <body>
        <h1>Nagłówek w starym stylu</h1>
    </body </html>
```



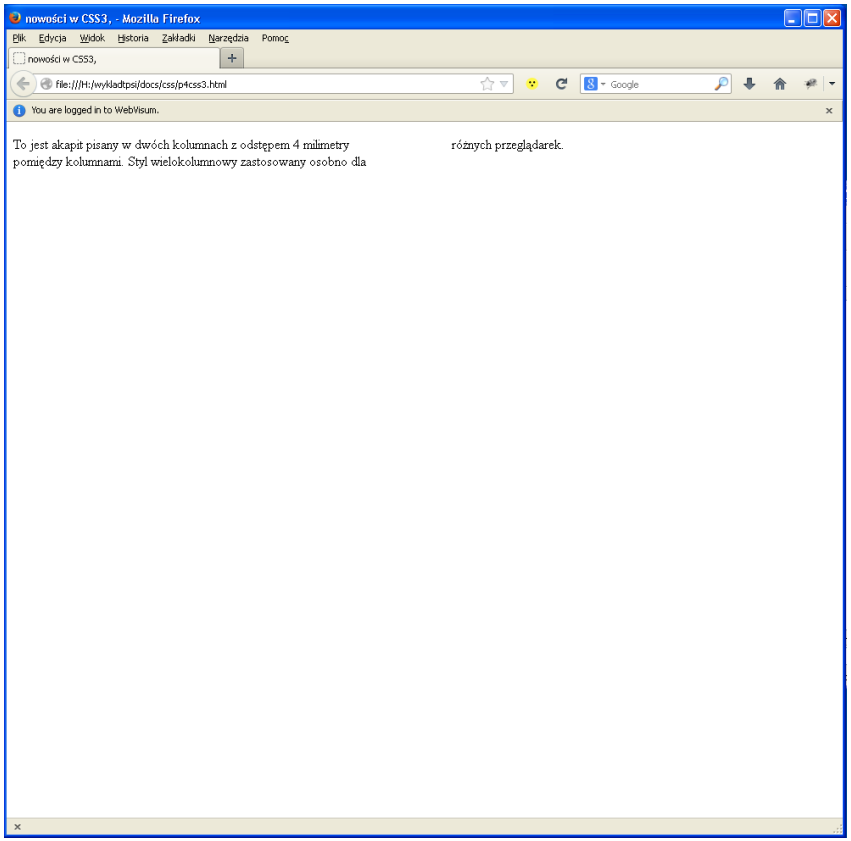
Użycie elementu `box-shadow` do uzyskania efektu cienia.

```
<script type="text/javascript">
    if (sessionStorage.pagecount)
    {
        sessionStorage.pagecount
                = Number(sessionStorage.pagecount) + 1;
    } else
    {
        sessionStorage.pagecount = 1;
    }
    document.write("Liczba aktualnie dostępnych użytkowników "
            + sessionStorage.pagecount + time(s));
</script>
```



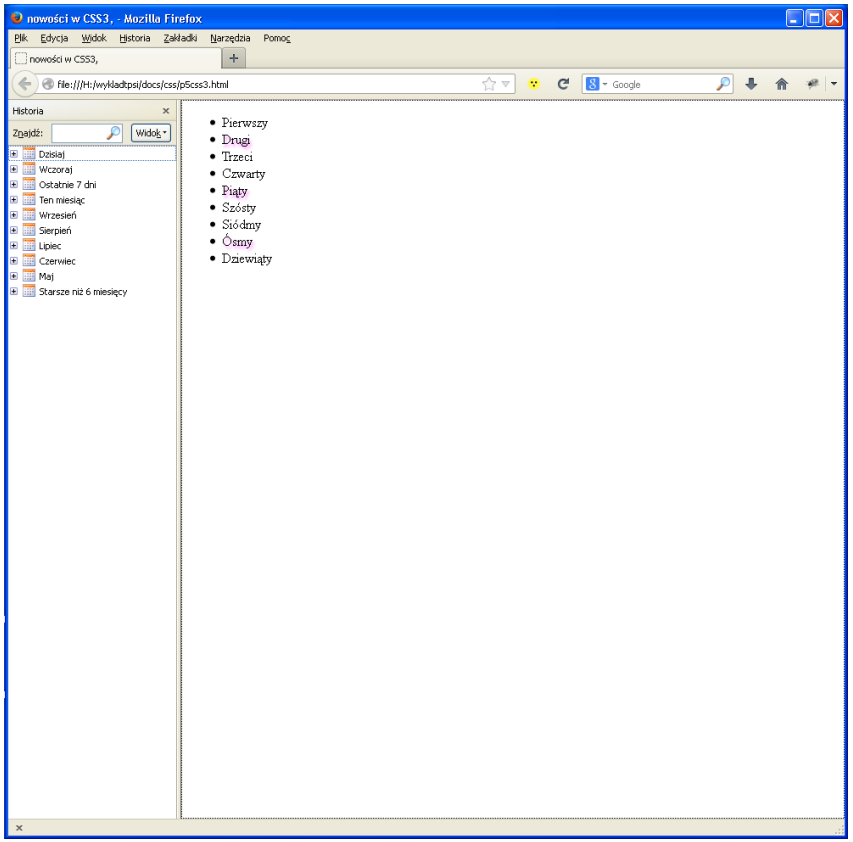
Użycie elementu `column-count` do uzyskania efektu akapitu z tekstem pisanym w kilku kolumnach.

```
                -moz-column-count: 2;
                -moz-column-gap: 4em;
            }
        </style> </head> <body>
        <p>To jest akapit pisany w dwóch kolumnach
            z odstępem 4 milimetry pomiędzy kolumnami.
            Styl wielokolumnowy zastosowany osobno
            dla różnych przeglądarek.</p>
    </body> </html>
```



Użycie atrybutu `nth-child` do uzyskania efektu podświetlania niektórych elementów listy.

```
17                  <li>Drugi</li>
18                  <li>Trzeci</li>
19                  <li>Czwarty</li>
20                  <li>Piąty</li>
21                  <li>Szósty</li>
22                  <li>Siódmy</li>
23                  <li>Ósmy</li>
24                  <li>Dziewiąty</li>
25              </ul>
26      </body> </html>
27
```



# 49 Co to jest JQuery?

Do 2005 roku JavaScript, służył najczęściej do tworzenia bardzo prostych animacji, walidacji formularzy lub projektowania rozwijanego menu. Wraz z popularyzacją technologii AJAX, i innych bogatych, interaktywnych interfejsów aplikacji internetowych, pisanie obszernych skryptów w JavaScript stało się trudne. Powstały różne biblioteki rozszerzające funkcjonalność JavaScript. można je podzielić na dwa rodzaje:

- Ułatwiające dynamizację istniejącego kodu HTML poprzez uproszczenie dostępu do DOM, CSS i AJAX. Przykładem jest właśnie JQuery.
- definiujące widgety i budujące cały interfejs użytkownika bezpośrednio w JavaScript np. Dijit, Ext-JS.

# 50 Co to jest JQuery?

JQuery to jeden z najbardziej udanych frameworków rozszerzających JavaScript. Jest łatwy w użyciu i bardzo ergonomiczny (Cały framework mieści się w pliku o wielkości 20kb. Dlatego można go bez problemu dołączać do każdej tworzonej strony. Drógą podstawową zaletą biblioteki jest ogromna zwięzłość kodu. Bardzo szybko można wykonać wiele operacji w zaledwie kilku linijkach. Aby użyć JQuery trzeba pobrać plik o rozszerzeniu .js w którym mieści się biblioteka ze strony jquery.com, a następnie dołączyć go do pliku HTML, tworzonej strony. Od tej pory w dowolnym miejscu swojego dokumentu można używać funkcji JQuery.

```
<script type="text/javascript" src="jQuery.js"></script>
```

# 51 JQuery - nieco historii

Pierwsza stabilna wersja biblioteki JQuery 1.0 została wydana 26 sierpnia 2006 Bardzo szybko tworzono kolejne:

- 14 stycznia 2007 1.1
- 10 września 2007 1.2
- 14 stycznia 2009 1.3 wprowadzono Sizzle Selector Engine do silnika
- 14 stycznia 2010 1.4
- 31 stycznia 2011 1.5
- 3 maja 2011 1.6
- 3 listopada 2011 1.7 nowe zdarzenia API:.on(),.off()
- 9 sierpnia 2012 1.8.0;
- 15 stycznia 2013 1.9;
- 18 kwietnia 2013 2.0;
- 18 grudnia 2014 - 2.1.3.
- 20 stycznia 2018 3.3.1 bardzo podobna do 3.3.0 tylko z małymi poprawkami.

# 52 Zalety JQuery

Użycie biblioteki jQuery ma wiele zalet w stosunku do standardowego JavaScript i CSS.

- kod jQuery jest bardziej zwięzły i dla wprawionych programistów bardziej czytelny.
- JQuery pozwala uzyskać separację definicji wyglądu od zasadniczej zawartości tak jak CSS ale na poziomie zachowań a nie definicji wyglądu.
- reakcje na zdarzenia kliknięć lub edycji nie zaśmiecają kodu HTML,
- kod JavaScript może być wydzielony do osobnego pliku, dzięki czemu łatwo go wielokrotnie wykorzystać.

# 53 Filozofia JQuery

jQuery udostępnia interfejs, który do złudzenia przypomina szablony stylów CSS. Różnica polega na tym, że zamiast definiować reguły wyglądu, określamy akcje do wykonania dla wybranej grupy elementów.
dzięki zastosowaniu selektorów oraz krótkiej nazwy dostępowej ($) kod jQuery jest wyjątkowo elastyczny, zwięzły i krótki.
Zwięzłość zapisu zwiększa dodatkowo możliwość tworzenia tzw. łańcuchów poleceń. Oznacza to, że na jednym obiekcie możemy rekurencyjnie wykonywać wiele poleceń (jedno polecenie przekazuje swój wynik drugiemu, drugie kolejnemu itd.). Wszystko to zapisujemy w jednej linii kodu.
Stosując jQuery nie musimy przejmować się różnicami w obsłudze zdarzeń, atrybutów i określaniu wymiarów elementów w różnych przeglądarkach, gdyż jQuery jest pod tym względem bardzo uniwersalny.

# 54 Struktura strony JQuery

W części nagłówkowej strony umieszczamy znacznik pobierający bibliotekę jQuery Możemy go także zawrzeć w odrębnym pliku .JS i odwołać się do niego w nagłówku dokumentu
Ffunkcji biblioteki możemy następnie używać w kodzie strony. Znak $ jest nazywany krótką nazwą dostępową. Używamy go zamiast przedrostka jquery przed wszystkimi metodami, obiektami i funkcjami pochodzącymi z biblioteki.
Jedną z najbardziej przydatnych funkcji jQUERY jest `ready()`

```
$(document).ready()
```

Oznacza, że dla standardowego obiektu `document` chcemy wykonać metodę jQuery o nazwie `ready()`. Zrealizuje ona funkcję przekazaną do niej jako parametr po załadowaniu w całości drzewa DOM reprezentującego kod HTML strony.
Funkcja `ready()` zastępuje zdarzenie `document.DOMContentLoaded` znane z JavaScript lub dla przeglądarek Internet Explorer zdarzenie `window.onload`. Jest to bardzo prosty sposób na wykonanie czegoś automatycznie przy wczytywaniu dokumentu.

# 55 Struktura strony JQuery

```
<!DOCTYPE html PUBLIC "-//W3C//DTD
   XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="pl-PL">
<html>  <head>
<title>Witaj, swiecie</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf8" />
<script type="text/javascript"
src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
//Cos wykonamy
});
</script>
</head>
<body>
<p>Tresc strony</p>
</body>
</html>
```

# 56 Przydatne funkcje i konstrukcje JQuery

Często używanymi funkcjami i konstrukcjami biblioteki JQuery są:

- `$("#identyfikator")` Zwraca odnośnik do elementu o atrybucie id równym identyfikator. Używamy go wówczas, gdy chcemy wykonać operacje na elemencie opatrzonym konkretnym identyfikatorem.
- `$(".klasa")` Zwraca odnośnik do elementu o atrybucie class równym klasa Używamy go, gdy chcemy wykonać operacje na wszystkich elementach danej klasy. Klasy są używane tak samo jak w CSS.
- `$(document)` Zwraca odnośnik do całego dokumentu HTML
- `$(document).ready(funkcja() {...})` Wykonuje funkcję `funkcja` od razu

- `click(kod)` Jest używany z odnośnikiem do danego elementu, wykonuje dany kod po kliknięciu tego elementu. Odpowiada zdarzeniu `onclick` najczęściej dla przycisku. Kod:
- `text("jakiś tekst")` Ustawia niesformatowany, czysty tekst dla elementu. Za jego pomocą możemy np. zmienić na nową nazwę linku lub zawartość tekstową akapitu.
- `html("jakiś tekst")` Ustawia nową zawartość html dla elementu. Parametr powinien zawierać prawidłowy tekst html ze znacznikami i zawartością.
- `toggle(szybkość)`, Jest używana wraz z odnośnikiem do danego elementu i służy do jego zwinięcia lub rozwinięcia. Argumentem jest szybkość zwinięcia która jest wyrażana

# 57 Reagowanie na kliknięcie

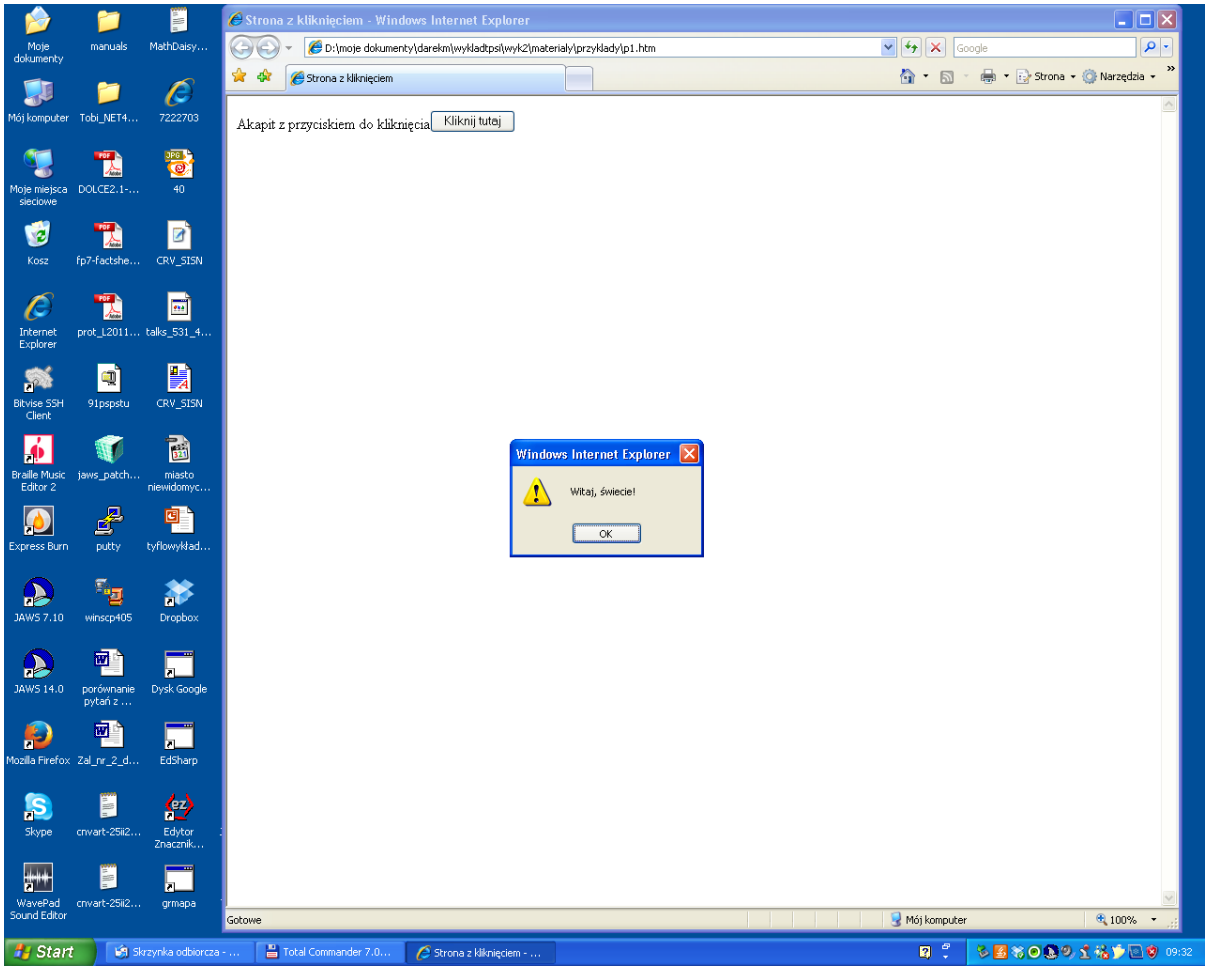
Prosty skrypt wyświetlający wiadomość po kliknięciu na łącze możemy zrealizować w następujących krokach:

- w nagłówku strony dołączamy bibliotekę JQuery
- pobieramy obiekt dokumentu DOM przy pomocy funkcji `$.(document)`
- wykonujemy funkcję `ready()` odpowiadającą zdarzeniu załadowania dokumentu.
- wewnątrz funkcji `ready` pobieramy listę przycisków z dokumentu i na pobranym przycisku wykonujemy funkcję `click` odpowiadającą zdarzeniu kliknięcia.
- wewnątrz funkcji `click` wykonujemy funkcję `alert` wyświetlającą komunikat.

# 58 Reagowanie na kliknięcie

```
<html> <head>
<title>Strona z kliknięciem</title>
<script type="text/javascript"
src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function() {
 $('button').click(function() {
   alert('Witaj, świecie!'); } );
});
</script>
</head>
<body>
<p>Akapit z przyciskiem do kliknięcia
<button>Kliknij tutaj</button></p>
</body> </html>
```

# 59 Reagowanie na kliknięcie



# 60 Ukrywanie i pokazywanie elementów

Do ukrywania i pokazywania różnych elementów strony służą trzy funkcje:

- `toggle()`, - pokazuje lub ukrywa element,
- `show()` - pokazuje element
- `hide()` - ukrywa element.

Funkcje te mogą przyjmować dwa argumenty. Pierwszym jest szybkość ukrywania lub pokazywania wyrażana przez identyfikatory `slow`, `normal`, `fast` lub przez dowolną liczbę milisekund.
`$("#more").toggle("slow");` normal - normalna szybkość przewijania
Drugi argument to funkcja, jaka ma się wykonać po wykonaniu pokazującej funkcji.
Inną pożyteczną funkcją jest `text()` Służy ona do zamiany już istniejącego tekstu na inny. Można przy jej pomocy np. link podpisany *"pokaż więcej"* po kliknięciu zmienić na *"ukryj"*.
Dodatkowo, aby wykonać działanie na obiekcie który aktualnie przetwarzamy, używamy identyfikatora `this`

# 61 Ukrywanie i pokazywanie elementów

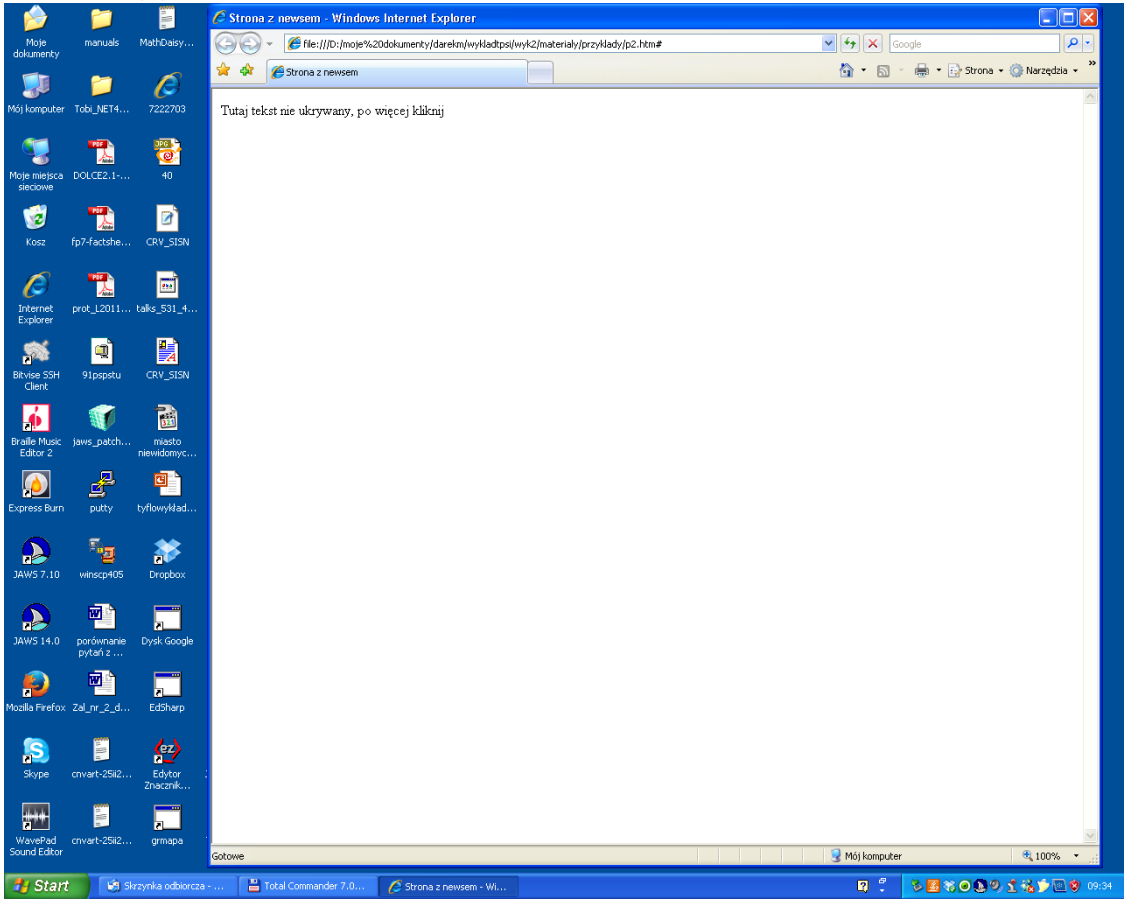
Kompletny przykład użycia funkcji `toggle` i `text`:

```
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        lang="pl-PL">
19                          }, function () {
20                              $(this).text("więcej");
21                          });
22                      });
23          </script> </head>
24      <body> <div>
25              <p>Tutaj tekst nie ukrywany, po więcej kliknij
26                  <a href="#" id="link">więcej</a></p>
27              <p id="more" style="display: none;">
28                  Tutaj tekst pokazywany i ukrywany.</p>
29          </div> </body>
</html>
31
32
```

# 62 Ukrywanie i pokazywanie elementów

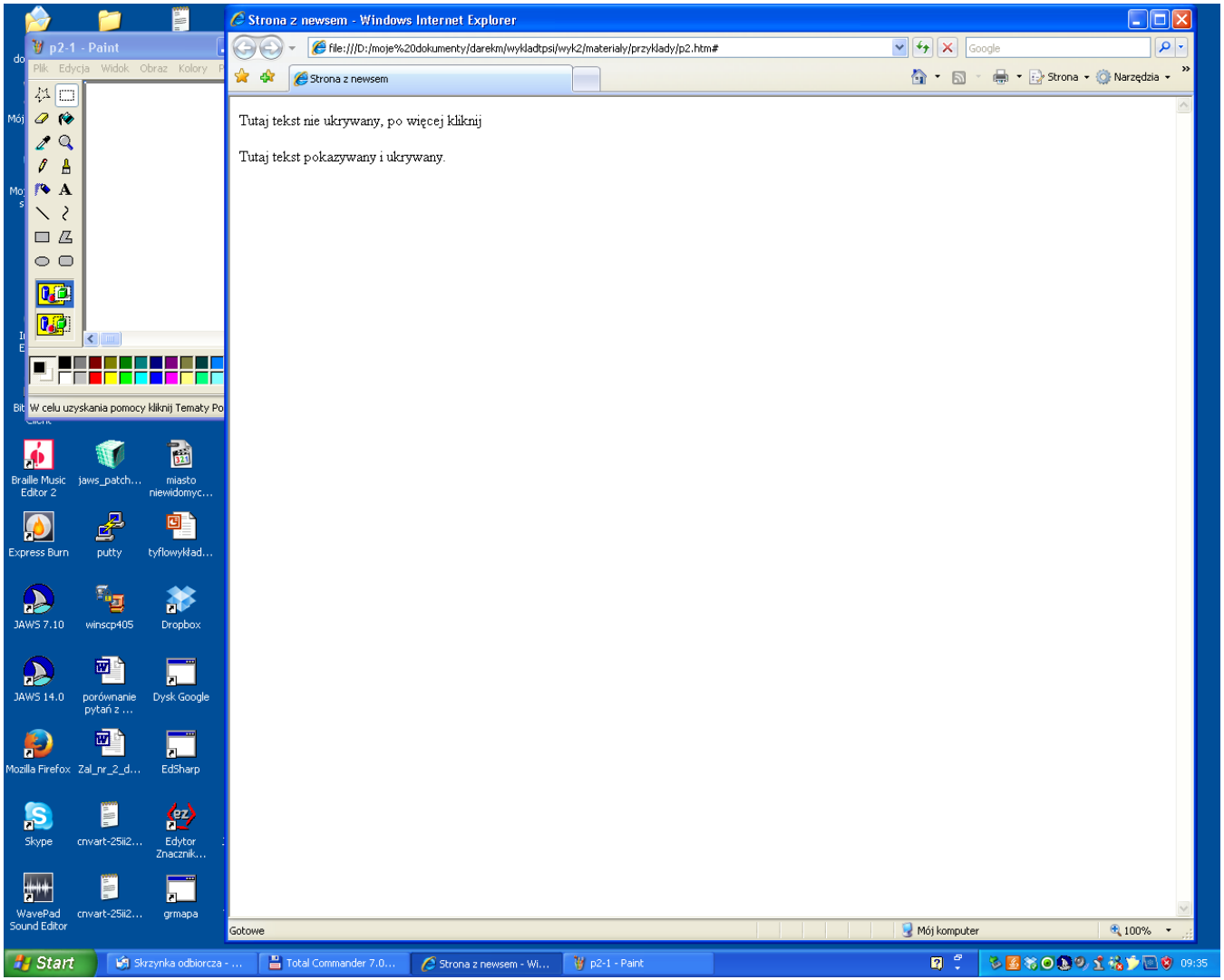
Kompletny przykład użycia funkcji `toggle` i `text`:

Przed zastosowaniem funkcji:



# 63 Ukrywanie i pokazywanie elementów

Kompletny przykład użycia funkcji `toggle` i `text`:
Po zastosowaniu funkcji:



# 64 Funkcje odd i even

Dużą zaletą JQuery jest możliwość szybkiej manipulacji na dowolnych elementach struktury dokumentu HTML. Do tego celu jest wykorzystywane drzewo dokumentu DOM (ang. Document Object Model). Dzięki odpowiednim funkcjom i odwołaniom możemy pobierać i wykonywać operacje na wybranych elementach takiego drzewa. Przykładowo, mając w dokumencie nie uporządkowaną listę elementów w znaczniku `<ul>` przy pomocy funkcji `odd()` oraz `even()` możemy wykonać operacje na parzystych bądź nieparzystych elementach takiej listy.

```
$("ul li:odd") { wykonaj coś na parzystych }
```

Wybierze każdy parzysty element `<li>` z listy `<ul>` i wykona coś na nim.

```
$("ul li:even") { wykonaj coś na nieparzystych }
```

Wykona coś na wszystkich nieparzystych elementach `<li>` z listy `<ul>`

# 65 Zmiana wyglądu elementów

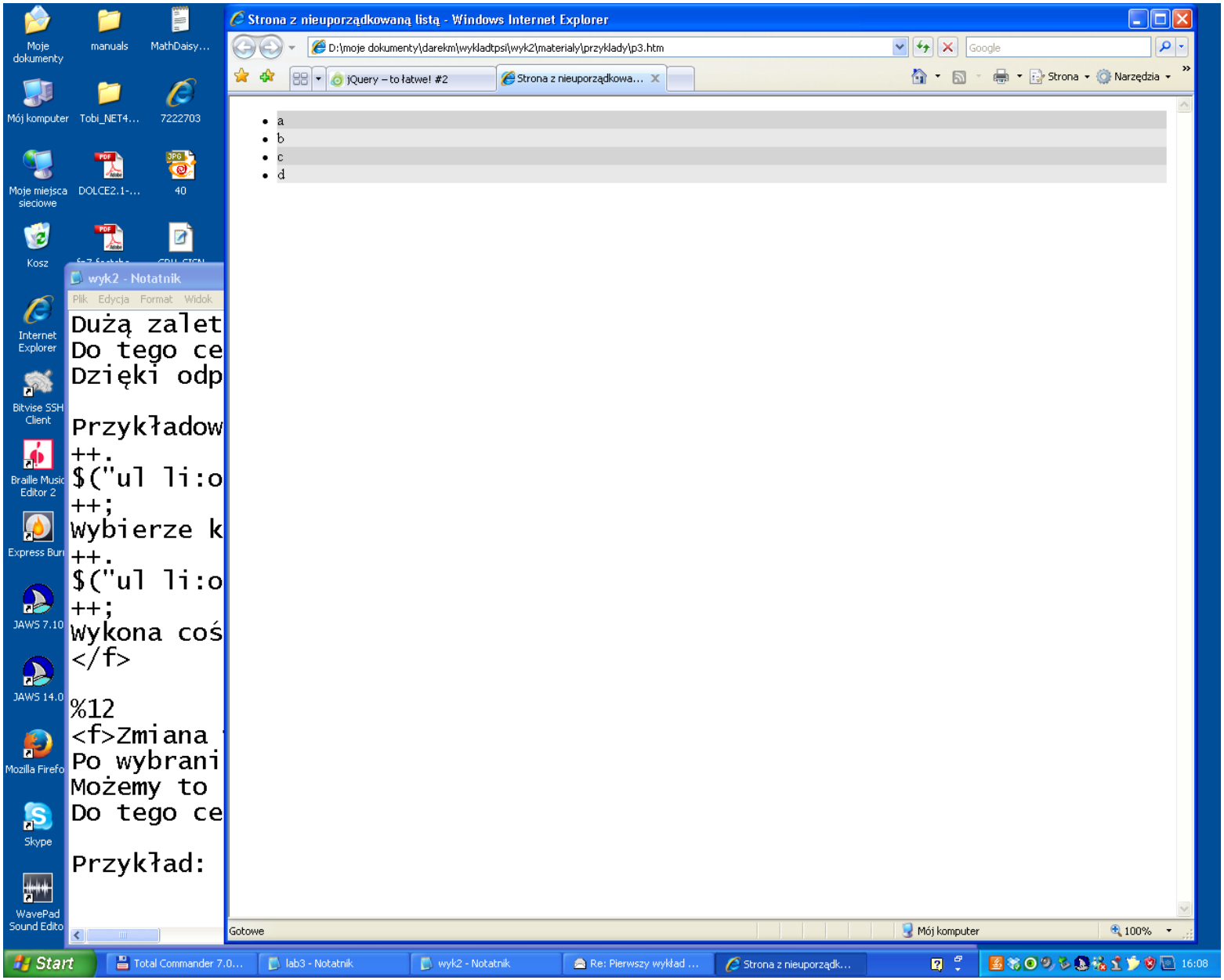
Po wybraniu odpowiednich elementów z drzewa dokumentu możemy dynamicznie zmienić ich wygląd. Możemy to zrobić definiując odpowiednie klasy dla różnych widoków przy pomocy CSS a następnie przypisać je w JQuery do tych elementów. Do tego celu służy funkcja `addclass()`
Odwrotny efekt polegający na usunięciu klasy dla danego elementu możemy osiągnąć używając funkcji `removeclass()`
Przykład użycia addclass()

```
4  <html xmlns="http://www.w3.org/1999/xhtml"
5        lang="pl-PL">
14              }
15              .tlo2
16              {
17                  background: #D4D4D4;
18              }
19          </style>
20          ...
21      </head>
22      ...
</html>
```

# 66 Zmiana wyglądu elementów

```
11          </script> </head>
12      <body>
13          <ul id="lista">
14              <li>a</li>
15              <li>b</li>
16              <li>c</li>
17              <li>d</li>
            </ul> </body> </html>
19
20
```

# 67 Zmiana wyglądu elementów



# 68 Podświetlanie elementu

W JQuery w łatwy sposób możemy osiągnąć ciekawe efekty wizualne.
Np. Podświetlanie elementu listy po najechaniu kursorem myszy.
Do tego celu możemy użyć funkcji `hover()` Jej składnia to:
`$("element").hover(funkcja1(){...}`, funkcja2()...)+
Funkcja `funkcja1()` wykona się, gdy najedziemy kursorem myszki na element, natomiast funkcja `funkcja2()` gdy przesuniemy go w inne miejsce.
Aby osiągnąć efekt podświetlania i i powrotu elementu do normalnego wyglądu możemy zdefiniować klasę CSS `podswietl` i przy pomocy funkcji `addclass()` i `removeclass()` przypisywać ją i usówać dla wybranych elementów listy.

# 69 Podświetlanie elementu - przykład

```
23                  </script> </head>
24                  <body>
25                  <ul id="lista">
26                  <li>a</li>
27                  <li>b</li>
28                  <li>c</li>
29                  <li>d</li>
30                  </ul> </body> </html>
31
32
```

# 70 operowanie na elementach drzewa dokumentu

jQUERY posiada wiele funkcji oraz selektorów, przy pomocy których możemy wykonywać różne operacje na elementach wczytanego dokumentu DOM. Najczęściej używane to:

- `(elementy).after( zawartosc)` - Wstawia zawartość podaną jako parametr bespośrednio po wybranych elementach.
- `(elementy).append( zawartosc)` - Wstawia zawartość podaną jako parametr na końcu każdego wybranego elementu.
- `(elementy).before( zawartosc )` - Wstawia zawartość podaną jako parametr bespośrednio przed każdym wybranym elementem.
- Selektor wszystkie `(\*")` - Wybiera wszystkie elementy. Nie jest zalecane jego używanie, gdyż działa wolno.
- selektor przycisku `( :button)` - Wybiera wszystkie elementy `<button>` oraz wszystkie elementy `<input>` typu `button`
- Selektor wszystkie dzieci `(\element > child")` - Wybiera wszystkie bespośrednie dzieci (tylko 1 poziomu) podanego elementu rodzica.
- `children()` - Pobiera wszystkie dzieci 1 poziomu wskazanego elementu.
- `css( "zawartosc ")` - Dodaje definicję klasy lub stylu css do wybranego elementu. Zawartość musi być poprawnym fragmentem arkusza css.

# 71 Przykład użycia: funkcja children()

- Definiujemy osobne klasy dla elementów `<div>` każdy otaczając inną ramką.
- Dla operacji ukrycia elementu potomnego o podanej klasie definiuujemy osobną funkcję.
- Funkcję tę przypisujemy do zdarzenia `onclick` przycisku.

# 72 Przykład użycia: funkcja children()

```
20              </div>
21              <div class="second">
22              </div>
23              <div class="third">
24              </div>
25              <div class="fourth">
26              </div>
27              <button onclick="hideChildElement()">Hide child element</button>
28          </div>  </body> </html>
29
```