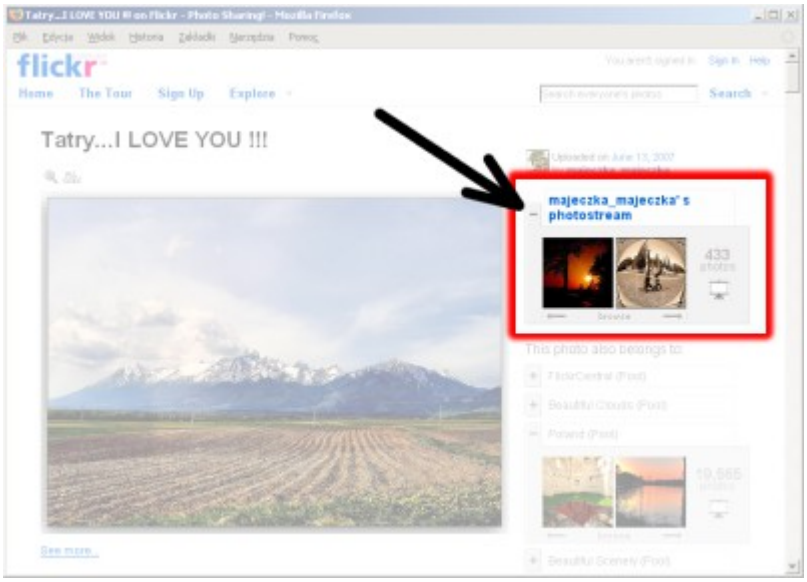
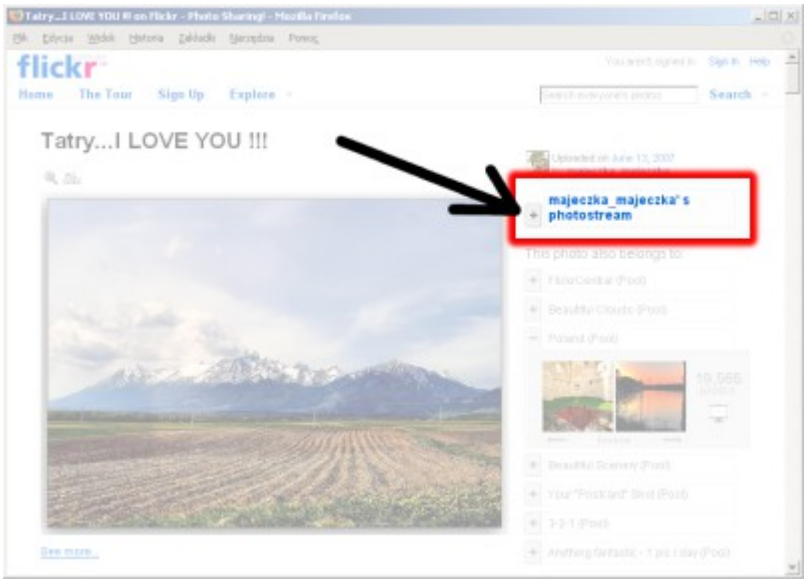
1. Ajax na stronach WWW

Strona WWW stosująca technologię Ajax zawiera dynamiczne elementy. Pod wpływem interakcji użytkownika wygląd i treść witryny ulegają zmianie. Zasadniczą cechą wyróżniającą witryny Ajax-owe jest brak przeładowania całej strony WWW. Zachowanie takie możemy zaobserwować np. w serwisie flickr (http://flickr.com).

Rysunek 1 prezentuje zdjęcie zatytułowane *Tatry... I LOVE YOU!!!*, którego autorem jest majeczka_majeczka. Po prawej stronie fotografii znajdują się kontrolki umożliwiające przeglądanie kategorii. Każda kategoria może zostać zwinięta (służy do tego ikona -) lub rozwinięta (ikona +). Rysunek 2 prezentuję witrynę z rysunku 1 po zwinięciu pierwszej kategorii.



Rysunek 1. Rozwinięty fragment strony w serwisie flickr.com



Rysunek 2. Strona z rysunku 1 po zwinięciu interaktywnego elementu

Dynamicznym fragmentem strony przedstawionej na rysunkach 1 oraz 2 jest zaznaczona na czerwono kontrolka wyświetlająca miniaturowe fotografie. Element ten jest widoczny (tj. rozwinięty) lub ukryty (tj. zwinięty).

Interakcja użytkownika polega na kliknięciu ikony + lub -. W odróżnieniu od zwykłych hiperłączy, strona nie zostaje przeładowana. Zmianie ulega tylko fragment witryny.

Zawartość rozwiniętego elementu pochodzi z serwera (są to miniaturki innych fotografii należących do danej kategorii). Zatem po rozwinięciu elementu, przeglądarka w tle pobiera z serwera dane i umieszcza je w odpowiednim obszarze strony WWW.

2. Ajax, czyli asynchroniczny JavaScript i XML

Ajax (ang. *Asynchronous JavaScript and XML — Asynchroniczny JavaScript i XML*)

Technika tworzenia aplikacji internetowych, w której interakcja użytkownika z serwerem odbywa się bez przeładowywania całego dokumentu.

źródło: Wikipedia.

Strona wykorzystująca Ajax jest zwykłym dokumentem HTML/CSS zawierającym skrypty JavaScript. Ajax nie wprowadza żadnych nowych języków. Interakcje użytkownika (np. kliknięcie ikony, wskazanie elementu kursorem myszki) jest realizowane poprzez zdarzenia zdefiniowane w specyfikacji HTML (m.in. onclick, onmouseover, onmouseout). Może to być kliknięcie elementu span:

```
<span onclick="..."></span>
```

czy wskazanie kursorem myszki obrazka img:

```
<img src="" alt="" onmouseover="..." />
```

Cała dynamiczna interakcja jest oprogramowana w języku JavaScript. Kod JavaScript możemy osadzić wewnątrz dokumentu HTML:

```
<script type="text/javascript">
...
</script>
```

lub zapisać w osobnym pliku skrypt.js:

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

Wysyłanie w tle zapytań HTTP do serwera o dodatkowe dane (np. o miniaturki widoczne na rysunku 1) odbywa się przy użyciu obiektu JavaScript o nazwie **XMLHttpRequest**.

W zależności od przeglądarki może to być obiekt wbudowany (tak jest w przypadku przeglądarek Firefox oraz Opera) lub tworzony jako kontrolka ActiveX (przeglądarki Internet Explorer).

Po odebraniu danych z serwera, modyfikujemy stronę WWW wykorzystując do tego model DOM. Innymi słowy w skrypcie JavaScript wywołujemy metody (np. getElementById(), getElementsByTagName()) by uzyskać dostęp do poszczególnych elementów HTML strony, która jest właśnie wyświetlona przez przeglądarkę.

Różnymi właściwościami (np. innerHTML, style) modyfikujemy treść (np. wstawiamy miniaturki pobrane w tle z serwera) i wygląd poszczególnych elementów HTML (np. rozwijamy/zwijamy element div).

Podsumowując, tworzenie Ajax-owych stron WWW wymaga znajomości:

- języków HTML/CSS, w szczególności zdarzeń HTML,
- języka JavaScript,
- obiektu XMLHttpRequest (jest to obiekt dostępny w JavaScript),
- protokołu HTTP, w szczególności metod GET oraz POST,
- języka XML,
- oraz modelu DOM (czyli obiektów, ich metod i właściwości dostępnych w JavaScript, które pozwalają na modyfikowanie strony WWW wyświetlanej przez przeglądarkę).

Języki przetwarzania po stronie serwera (np. PHP, ASP, JSP) nie są konieczne. Wszystkie przykłady będą napisane wyłącznie w językach HTML, CSS, JavaScript, XML bez przetwarzania po stronie serwera.

Rozwiązaniami podobnymi do Ajax są AHAH (Asynchronous HTML and HTTP — asynchroniczny HTML i HTTP) oraz AXAH (Asynchronous XHTML and HTTP — asynchroniczny XHTML i HTTP). Różnią się one od Ajax-a formatem danych. Obecnie, bez względu na format danych, rozwiązania stosujące asynchroniczną komunikację z serwerem WWW są określane terminem Ajax (nawet, jeśli nie stosują XML).

AHAH

Skrót od **Asynchronous HTML and HTTP**, (ang. Asynchroniczny HTML i HTTP). Metoda dynamicznego przeładowywania fragmentów stron WWW przy użyciu JavaScript, podobna do Ajax. Różnica polega na tym, że w przypadku AHAH odpowiedzi wysyłane przez serwer nie zawierają XML tylko tekst lub HTML. Skrypt JavaScript nie dokonuje parsingu odebranych danych, a bezpośrednio wstawia je do dokumentu.

źródło: Wikipedia.

3. Tworzenie obiektu XMLHttpRequest

Praktyczne poznawanie Ajax-a rozpoczniemy od tworzenia obiektu do komunikacji asynchronicznej. W przeglądarkach Firefox, Opera oraz Internet Explorer obiekt taki tworzymy następująco:

```
/*
 * Przeglądarki: Firefox 2+, Opera 9+, IE 7+
 */
request = new XMLHttpRequest();
```

Przeglądarka Internet Explorer 6 wymaga kodu:

```
/*
 * Przeglądarka: IE 6
 */
request = new ActiveXObject('Msxml2.XMLHTTP');
```

Uniwersalna procedura tworzenia obiektu XMLHttpRequest jest przedstawiona na listingu 1. Będzie ona działała poprawnie w większości współczesnych przeglądarek.

```
function getXMLHttpRequest()
{
  var request = false;

  try {
    request = new XMLHttpRequest();
  } catch(err1) {
    try {
      request = new ActiveXObject('Msxml2.XMLHTTP');
    } catch(err2) {
      try {
        request = new ActiveXObject('Microsoft.XMLHTTP');
      } catch(err3) {
        request = false;
      }
    }
  }
  return request;
}
```

Listing 1. Tworzenie obiektu do asynchronicznej komunikacji

Wynikiem funkcji getXMLHttpRequest() jest utworzony obiekt lub — w przypadku niepowodzenia — wartość false. Z funkcji tej korzystamy następująco:

```
var r;
r = getXMLHttpRequest();
```

Obiekt **r** jest gotowy do wysyłania zapytań protokołem HTTP do serwera WWW.

Powyższy przykład oraz kod z listingu 1 są napisane w języku JavaScript. Należy je umieścić w nagłówku strony WWW:

```
<head>
  <title>...</title>
  <script type="text/javascript">
  function getXMLHttpRequest()
  {
    ...
  }

  var r;
```

```
  r = getXMLHttpRequest();
  </script>
</head>
<body>
...
</body>
```

4. Wysyłanie żądań HTTP i odbieranie danych z serwera

Witryna wykonana w technologii Ajax składa się z dwóch komponentów: dokumentu HTML oraz danych udostępnianych przez serwer WWW. Przeglądarka w odpowiedzi na interakcje użytkownika (np. kliknięcie ikony +) wysyła zapytanie HTTP do serwera. W odpowiedzi, serwer przekazuje do strony WWW (dokładniej: do skryptu JavaScript zawartego w dokumencie HTML) dane.

W tym kroku skupimy się na wysłaniu żądania oraz odebraniu wyników.

Do wysyłania żądań HTTP obiekt XMLHttpRequest służą metody open() oraz send(). Metoda open() przygotowuje zapytanie HTTP, a send() rozpoczyna transmisję.

Funkcja open() ma trzy parametry: pierwszym jest nazwa metody protokołu HTTP, drugim — adres URL danych, zaś trzecim — flaga logiczna ustalająca, czy żądanie ma być realizowane asynchronicznie (tj. w tle, bez czekania na zakończenie).

Wywołanie:

```
r.open('GET', 'dane.txt', true);
```

przygotuje asynchroniczne żądanie GET dotyczące dokumentu dane.txt. Podany adres URL może dotyczyć nie tylko pliku tekstowego, ale pliku XML, PHP, ASP, JSP czy dowolnego innego zasobu dostępnego w ramach usługi WWW:

```
r.open('GET', 'dane.xml', true);
r.open('GET', 'skrypt.php', true);
r.open('GET', 'strona.asp', true);
r.open('GET', 'witryna.jsp', true);
r.open('GET', 'plik.jpg', true);
```

Może to również być pełny adres URL odnoszący się do innego serwera:

```
r.open('GET', 'http://www.example.net/d/data.php', true);
```

czy adres zawierający zmienne URL (tj. fragment występujący w adresie URL po znaku zapytania):

```
r.open('GET', 'http://www.example.net/d/get.php?id=123', true);
```

Żądanie przygotowane metodą open() wysyłamy wywołując metodę send():

```
r.send(null);
```

Metoda send() ma jeden parametr: dane dołączane do zapytania. Parametr ten należy wykorzystać w przypadku metody POST. Jeśli stosowaną metodą jest GET, wówczas metodę send() wywołujemy podając parametr null.

Jeśli zapytanie jest wysyłane asynchronicznie, to wykonanie skryptu JavaScript nie zostanie wstrzymane. Innymi słowy: skrypt JavaScript nie będzie czekał, aż zapytanie zostanie wysłane, a serwer zwróci wynik. Skrypt będzie wykonywany dalej, a żądanie HTTP będzie wykonywane równolegle, w tle.

W celu odebrania wyników zwracanych przez obiekt XMLHttpRequest należy przygotować funkcję, która zostanie wywołana po zakończeniu transmisji. Funkcja ta może mieć dowolną nazwę, np. processResponse(). Należy ją przypisać do obsługi zdarzenia onreadystatechange obiektu XMLHttpRequest:

```
r.onreadystatechange = processResponse;
```

W treści funkcji sprawdzamy czy nadeszła odpowiedź na wysłane żądanie oraz czy żądanie HTTP zostało poprawnie przetworzone przez serwer. Właściwość readyState o wartości 4 informuje o tym, że nadeszła odpowiedź na wysłane żądanie. Zaś właściwość status zawiera kod odpowiedzi HTTP. Wartość 200 oznacza, że serwer poprawnie przetworzył żądanie.

Funkcja odpowiedzialna za odbieranie danych z serwera przyjmuje postać:

```
function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {
      ...
    };
  };
}
```

Zwróć uwagę, że jest w niej wykorzystana zmienna globalna r. Jest to oczywiście obiekt XHMLHttpRequest.

4.1 Odbieranie danych w formacie tekstowym

Przejdźmy do wykonania kompletnego przykładu demonstrującego wysyłanie żądań HTTP i odbieranie wyników w formacie tekstowym. Takie rozwiązania są określane mianem AHAH.

Rozwiązanie składa się z dwóch plików: index.html oraz dane.txt. Plik dane.txt zawiera jedną linijkę:

```
Lorem ipsum...
```

Listing 2 przedstawia zarys strony index.html. Skrypt JavaScript zawarty w nagłówku strony rozpoczyna się od definicji funkcji getXMLHttpRequest(). Następnie funkcja ta jest wywołana, a zwrócony przez nią obiekt przypisany do zmiennej r. Kolejnym elementem jest definicja funkcji processResponse(). Funkcja ta będzie wywołana po zakończeniu transmisji danych z serwera. W jej treści odwołujemy się do zmiennej globalnej r — tj. utworzonego

wcześniej obiektu XMLHttpRequest. Skrypt kończymy wywołując metody open(), send() oraz przypisując funkcję processResponse() do obsługi zdarzenia *onreadystatechange*.

```
<head>
...
<script type="text/javascript">
function getXMLHttpRequest()
{
  ...
}

var r;
r = getXMLHttpRequest();

function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {
      alert('Tekst z serwera: ' + r.responseText);
    };
  };
}

r.open('GET', 'dane.txt', true);
r.onreadystatechange = processResponse;
r.send(null);
</script>
</head>
```

Listing 2. Wysyłanie zapytań i odbieranie danych tekstowych: zarys strony index.html

Fragmentem odpowiedzialnym za wyświetlenie danych pochodzących z serwera jest wiersz:

```
alert('Tekst z serwera: ' + r.responseText);
```

Dane (w formacie tekstowym) pochodzące z serwera (tj. z pliku dane.txt) są dostępne we właściwości responseText obiektu XMLHttpRequest.

Tak wykonana strona nie wykorzystuje asynchroniczności transferu: dane wysyłane przez serwer zostaną wyświetlone (w okienku informacyjnym alert()) natychmiast po odwiedzeniu strony index.html.

Opisany przykład wykorzystuje protokół HTTP. Nie można go więc uruchomić w wersji offline (np. z płyty CD). W celu uruchomienia przykładu trzeba dysponować zainstalowanym serwerem WWW (np. Apache). Po przekopiowaniu plików do folderu htdocs/, który jest przeznaczony na strony WWW, przykład uruchamiamy odwiedzając stronę http://localhost.

4.2 Odbieranie danych w formacie XML

Strona prezentująca wymianę danych w formacie XML w głównym zarysie wygląda podobnie do strony stosującej surowy tekst. Składa się z dwóch plików: index.html oraz dane.xml.

Użycie języka XML do transferu danych wymaga wymiany dwóch elementów. Po pierwsze plik danych dane.xml zawiera XML:

```
<?xml version="1.0" encoding="utf-8"?>
<tekst>
  Lorem ipsum...
</tekst>
```

Po drugie dane XML odebrane z serwera należy przed wyświetleniem przetworzyć.

Dostęp do danych w formacie XML zapewnia właściwość responseXML obiektu XMLHttpRequest. Do przetworzenia kodu XML służą m.in.: metoda getElementsByTagName() i właściwości childNodes oraz nodeValue.

Wewnątrz funkcji processResponse() najpierw odbieramy XML zwrócony przez serwer:

```
var x1 = r.responseXML;
```

Następnie wyszukujemy element XML o nazwie *tekst*:

```
var x2 = x1.getElementsByTagName('tekst');
```

po czym pobieramy pierwszy ze znalezionych elementów <tekst>:

```
var x3 = x2[0];
```

Teraz przechodzimy do potomków tj. elementów zawartych wewnątrz elementu <tekst>...</tekst>:

```
var x4 = x3.childNodes;
```

i pobieramy pierwszego z nich:

```
var x5 = x4[0];
```

Wartość potomka umieszczamy w zmiennej x6:

```
var x6 = x5.nodeValue;
```

i wyświetlamy w oknie informacyjnym:

```
alert('XML z serwera: ' + x6);
```

Całość możemy wykonać jedną instrukcją:

```
alert(
  'XML z serwera: ' +
  r.responseXML.getElementsByTagName('tekst')[0].childNodes[0].nodeValue
);
```

Zarys strony index.html pobierającej z serwera dane w formacie XML jest przedstawiony na listingu 3.

```
<head>
<script type="text/javascript">
function getXMLHttpRequest()
```

```
{
  ...
}

var r;
r = getXMLHttpRequest();

function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {
      alert(
        'XML z serwera: ' +
       r.responseXML.getElementsByTagName('tekst')
[0].childNodes[0].nodeValue
      );
    };
  };
}

r.open('GET', 'dane.xml', true);
r.onreadystatechange = processResponse;
r.send(null);
</script>
</head>
```

Listing 3. Wysyłanie zapytań i odbieranie danych w formacie XML: zarys strony index.html

5. Asynchroniczna wymiana treści

Zasadniczą cechą wyróżniającą strony stosujące Ajax jest asynchroniczność połączona z wymianą tylko fragmentu dokumentu. W wyniku akcji użytkownika (np. kliknięcia ikony +) następuje wysłanie żądania do serwera, odebranie danych i umieszczenie nowej treści w wybranym miejscu strony. W celu oprogramowania takiego zachowania należy poznać technikę wymiany fragmentu strony WWW oraz reakcji na zdarzenia.

5.1 Model DOM i metoda getElementById()

Do manipulacji stroną WWW wyświetlaną przez przeglądarkę służy model DOM. Cała strona WWW widoczna w bieżącej chwili jest dostępna w skryptach JavaScript za pośrednictwem zestawu obiektów, metod i właściwości.

Jeśli na stronie WWW znajduje się element o identyfikatorze #tresc:

```
<div id="tresc"></div>
```

to dostęp do niego możemy uzyskać wywołując metodę getElementById(). Jej parametrem jest identyfikator elementu HTML. Metoda ta zwraca obiekt:

```
var el;
el = document.getElementById('tresc');
```

który możemy poddać manipulacjom. Możemy wymienić jego treść:

```
el.innerHTML = '<strong>Lorem</strong> ipsum...';
```

oraz styl CSS:

```
el.style.border = '2px solid red';
el.style.background = '#fef4e0';
```

Do wymiany treści elementu służy właściwość innerHTML, a do modyfikacji stylów — właściwość style. Poszczególne właściwości CSS są dostępne po kropce, przy czym znak - zostaje zastąpiony znakiem _:

```
el.style.margin = '10px';
el.style.margin_left = '50px';
```

Listing 4 przedstawia przykładową stronę WWW, w której dynamicznie wymieniono treść i format elementu div. Pomimo tego, że element div#tresc jest pusty, jeśli odwiedzisz stronę index.html, ujrzysz tekst *Lorem ipsum* na czerwonym tle. Za wstawienie i sformatowanie tekstu odpowiada skrypt JavaScript umieszczony poniżej elementu div.

```
<body>

<div id="tresc"></div>

<script type="text/javascript">
var el;

el = document.getElementById('tresc');
el.innerHTML = '<strong>Lorem</strong> ipsum...';
el.style.border = '2px solid red';
el.style.background = '#fef4e0';
</script>

</body>
```

Listing 4. Dynamiczna wymiana treści i formatu elementu div

Przykład ten nie wykorzystuje protokołu HTTP. Może więc być uruchomiony offline.

5.2 Zdarzenia HTML

Interaktywne reakcje na zachowanie użytkownika są oprogramowane za pomocą zdarzeń HTML. Niemal każdy element HTML może być wzbogacony o zdarzenia:

```
<span onclick="...">...</span>
<p onmouseover="...">...</p>
<td onmouseout="...">...</td>
```

Treścią obsługi zdarzenia jest wybrana funkcja JavaScript. Reakcją na kliknięcie elementu może być zmiana treści oraz formatu. Najpierw przygotowujemy element HTML, który będzie poddany zmianom po wystąpieniu zdarzenia:

```
<div id="tresc">Tekst tekst tekst...</div>
```

Następnie przygotowujemy element HTML, który będzie generował zdarzenie. Kliknięcie poniższego elementu li, będzie powodowało wywołanie funkcji onclickHandler():

```
<li onclick="onclickHandler();">onclick</li>
```

Funkcja onclickHandler() odpowiada za zmianę treści (właściwość innerHTML) oraz formatu (właściwości style.border oraz style.background) elementu o identyfikatorze #tresc (dostęp do elementu uzyskujemy metodą getElementById()):

```
function onclickHandler()
{
  var el;
  el = document.getElementById('tresc');
  el.innerHTML = 'click click click...';
  el.style.border = '2px solid red';
  el.style.background = '#fef4e0';
}
```

Zarys przykładu prezentującego obsługę zdarzeń onclick, onmouseover oraz onmouseout jest widoczny na listingu 5.

```
<head>
<script type="text/javascript">
function onclickHandler()
{
  var el;
  el = document.getElementById('tresc');
  el.innerHTML = 'click click click...';
  el.style.border = '2px solid red';
  el.style.background = '#fef4e0';
}

function onmouseoverHandler()
{
  ...
}

function onmouseoutHandler()
{
  ...
}
</script>
</head>

<body>
<ul>
  <li onclick="onclickHandler();">onclick</li>
  <li onmouseover="onmouseoverHandler();">onmouseover</li>
  <li onmouseout="onmouseoutHandler();">onmouseout</li>
</ul>
<div id="tresc">Tekst tekst tekst...</div>
</body>
```

Listing 5. Przykład demonstrujący oprogramowanie zdarzeń HTML

5.3. Piosenki

Wykorzystując zdarzenie *onmouseover* przygotujmy pierwszy przykład, który będzie demonstrował asynchroniczną wymianę fragmentu strony WWW. Strona będzie zawierała menu i treść. Pozycjami menu będą tytuły piosenek. Po wskazaniu tytułu piosenki

wskaźnikiem myszy, treść wybranej piosenki będzie umieszczana na stronie WWW. Całość będzie się odbywała asynchronicznie: tekst piosenki będzie pobierany z serwera (w formacie XML) dopiero po wskazaniu wybranej pozycji menu kursorem myszy.

Przykład składa się z czterech plików: dokumentu index.html oraz trzech plików z tekstami piosenek krasnoludki.xml, misie.xml, lisek.xml. Pliki z danymi mają identyczną strukturę. Oto fragment pliku krasnoludki.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<piosenka>
  <tytul>Krasnoludki</tytul>
  <tekst>
    My jesteśmy krasnoludki,
    Hopsa sa, hopsa sa!
    ...
  </tekst>
</piosenka>
```

A tak wygląda zarys kodu HTML strony index.html:

```
<div id="pojemnik">
  <ul id="menu">
    <li>...</li>
    <li>...</li>
    ...
  </ul>
  <p id="tresc">...</p>
</div>
```

W menu znajdują się elementy li posiadające obsługę zdarzeń onmouseover oraz onmouseout:

```
<li>
  <a href='#'
    onmouseover="getText('dane/lisek.xml');"
    onmouseout="clearText();">
      Chodzi lisek koło drogi
  </a>
</li>
```

Skrypt JavaScript rozpoczynamy od definicji funkcji getXMLHttpRequest() i utworzenia obiektu r:

```
function getXMLHttpRequest()
{
  ...
}

var r;
r = getXMLHttpRequest();
```

Następnie definiujemy funkcję processResponse(), która będzie wywoływana po odebraniu danych z serwera. W treści tej funkcji sprawdzamy, czy żądanie zostało poprawnie przetworzone przez serwer (r.readyState == 4 oraz r.status == 200). Jeśli tak, to odebrany tekst w formacie XML (czyli r.responseXML) wstawiamy do elementu HTML o identyfikatorze #tresc. Dodatkowo zmieniamy kolor tła elementu div#tresc:

```
function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {
      document.getElementById('tresc').innerHTML =
        r.responseXML.getElementsByTagName('tekst')
[0].childNodes[0].nodeValue;

      document.getElementById('tresc').style.background = '#e3f5fb';
    };
  }
}
```

Pobranie danych z serwera rozpoczyna się w momencie wskazania hiperłącza kursorem myszy. Zdarzenie onmouseover jest obsługiwane przez funkcję getText(), której parametrem jest nazwa pliku XML z tekstem piosenki:

```
function getText(Dane)
{
  r.open('GET', Dane, true);
  r.onreadystatechange = processResponse;
  r.send(null);
}
```

Ostatnia z funkcji JavaScript odpowiada za wyczyszczenie akapitu, gdy myszka zostanie przesunięta poza obszar hiperłącza. Wystąpienie zdarzenia onmouseout powoduje wywołanie funkcji clearText():

```
function clearText()
{
  document.getElementById('tresc').innerHTML = 'Witaj...';
  document.getElementById('tresc').style.background = 'white';
}
```

Zarys skryptu index.html jest przedstawiony na listingu 6.

```
<head>
<script type="text/javascript">
function getXMLHttpRequest()
{
  ...
}

var r;
r = getXMLHttpRequest();

function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {
      document.getElementById('tresc').innerHTML =
        r.responseXML.getElementsByTagName('tekst')
[0].childNodes[0].nodeValue;

      document.getElementById('tresc').style.background = '#e3f5fb';
    };
  }
}
```

```
function getText(Dane)
{
  r.open('GET', Dane, true);
  r.onreadystatechange = processResponse;
  r.send(null);
}

function clearText()
{
  document.getElementById('tresc').innerHTML = 'Witaj...';
  document.getElementById('tresc').style.background = 'white';
}
</script>
</head>
<body>

<div id="pojemnik">
  ...
</div>
</body>
```

Listing 6. Piosenki: zarys pliku index.html

6. Aparaty fotograficzne

Przykład pt. *„Aparaty fotograficzne”* w pełni prezentuje możliwości Ajax-a. Strona główna zawiera listę nazw aparatów, każdy z nich jest umieszczony wewnątrz zielonego obszaru div (rysunek 3).

Rysunek 3. Witryna z aparatami fotograficznymi

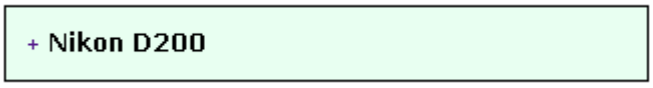
Z lewej strony nazwy każdego aparatu znajduje się ikona + pozwalająca na wyświetlenie szczegółowych danych. Rysunek 4 przedstawia witrynę po rozwinięciu danych aparatu Canon EOS 20D.

Rysunek 4. Witryna z aparatami po rozwinięciu danych aparatu Canon EOS 20D

W tym samym momencie możemy rozwinąć dane dowolnej liczby aparatów. Rysunek 5 przedstawia wygląd witryny po rozwinięciu dwóch aparatów.

Rysunek 5. Witryna z aparatami po rozwinięciu danych dwóch aparatów

Wygląd pojedynczego aparatu w formie zwiniętej i rozszerzonej jest przedstawiony na rysunkach 6 oraz 7.



Rysunek 6. Dane pojedynczego aparatu w postaci zwiniętej

Rysunek 7. Dane pojedynczego aparatu w postaci rozwiniętej

Witryna pobiera dane o aparatach w sposób asynchroniczny. Na stronie WWW znajdują się wyłącznie nazwy aparatów. Po kliknięciu ikony +, skrypt JavaScript pobiera z serwera szczegółowe dane wybranego aparatu. Po odebraniu odpowiedzi, we wnętrzu odpowiedniego zielonego prostokąta umieszczane są dane pobrane z serwera. Serwer wysyła dane aparatu w formacie XML.

Jest więc zatem:

- **asynchroniczny JavaScript**,
- **XML**,
- **modyfikacja strony przy użyciu modelu DOM**,
- **wymiana fragmentu strony WWW bez przeładowywania całego dokumentu**.

6.1 Rozwijanie i zwijanie jednej kontrolki

Pracę nad witryną *Aparaty fotograficzne* rozpoczynamy od opracowania pojedynczej zwijanej kontrolki. Kontrolka taka jest zawarta w pojemniku div#tresc i zawiera jedno hiperłącze a, tytuł *Pojemnik na treść* oraz drugi element div#minitresc:

```
<div id="tresc">
<a id="ikona" href="#" onclick="expandCollapse();">+</a>
Pojemnik na treść
<div id="minitresc"></div>
</div>
```

W obsłudze zdarzenia onclick ikony + należy zmienić wygląd całej kontrolki.

W zależności od wartości globalnej zmiennej expanded ukrywamy (style.display = 'none') lub pokazujemy (style.display = 'block') zawartość pojemnika div#minitresc. Ponadto zmieniamy ikonę oraz wstawiamy do elementu div#minitresc tekst *A B C...*:

```
var expanded = false;

function expandCollapse()
{
  if (expanded) {
    expanded = false;
    document.getElementById('minitresc').style.display = 'none';
    document.getElementById('ikona').innerHTML = '+';
  } else {
    expanded = true;
    document.getElementById('minitresc').style.display = 'block';
    document.getElementById('minitresc').innerHTML = 'A B C...';
    document.getElementById('ikona').innerHTML = '-';
  }
}
```

6.2 Rozwijanie i zwijanie wielu kontrolek

Jeśli na stronie WWW ma się znajdować seria podobnych rozwijanych kontrolek to najlepiej zrezygnować ze stosowania identyfikatorów. Cała kontrolka jest zawarta w elemencie div klasy tresc. Wewnątrz zawiera hiperłącze a, tytuł span oraz dodatkowy element div:

```
<div class="tresc">
  <a href="#" onclick="expandCollapse(this);">+</a>
  <span>Pojemnik na treść</span>
  <div></div>
</div>
```

Uwaga: w rozwiązaniu tym nie możesz umieścić białych znaków pomiędzy elementami HTML. Kod od znacznika do znacznika należy napisać w jednej linijce bez odstępów:

```
<div class="tresc"><a ...>+</a><span>...</span><div></div></div>
```

Powodem jest to, że białe znaki będą dodatkowymi dziećmi elementu div.tresc. Odwołanie Id.parentNode.childNodes[2] nie będzie dotyczyło wewnętrznego elementu div.

Zwróć uwagę, że obsługą zdarzenia onclick zajmuje się funkcja expandCollapse() wywołana z parametrem this. Parametrem this w modelu DOM jest ten węzeł drzewa, który wygenerował zdarzenie (w naszym przypadku: kliknięte hiperłącze). Takie rozwiązanie znacznie uprości treść funkcji expandCollapse():

```
function expandCollapse(Id)
{
  var n = Id.parentNode.childNodes[2];

  if (n.style.display == 'block') {
    n.style.display = 'none';
    Id.innerHTML = '+';
  } else {
    n.style.display = 'block';
    n.innerHTML = 'A B C...';
    Id.innerHTML = '-';
  }
}
```

To, czy element jest zwinięty, czy rozwinięty stwierdzamy (w warunku instrukcji if) na podstawie wartości właściwości display. Nie wprowadzamy do tego żadnych dodatkowych zmiennych. Zmienna o nazwie n jest drugim elementem div (tj. tym, który poprzednio miał identyfikator minitresc) wewnątrz bieżącej kontrolki. Docieramy do niego następująco:

- this przekazany do funkcji jest klikniętym hiperłączem a,
- parametr funkcji expandCollapse() nazywa się Id, zatem w treści funkcji zamiast this stosowany jest identyfikator Id,
- pobieramy rodzica klikniętego hiperłącza (Id.parentNode), czyli element div.tresc,

- następnie pobieramy trzecie dziecko elementu div.tresc (pierwsze dziecko: id.parentNode.childNodes[0] — hiperłącze a; drugie dziecko: id.parentNode.childNodes[1] — tytuł span; trzecie dziecko: id.parentNode.childNodes[2] — wewnętrzny element div),

- w ten sposób zmienna n odnosi się do obiektu DOM: wewnętrznego elementu div przeznaczonego na treść.

We wnętrzu instrukcji if, podobnie jak poprzednio zmieniamy widoczność wewnętrznego elementu div, ustalamy jego treść (*A B C...*) oraz zamieniamy ikonę plus na minus, a minus na plus.

6.3 Aparaty fotograficzne — kompletny przykład

Przejdźmy do połączenia wszystkich elementów. Wykorzystamy serię rozwijanych kontrolek div, oraz Ajax do pobierania szczegółowych danych konkretnego aparatu.

Opisywany przykład składa się z pliku index.html oraz danych w formacie XML, zawartych w folderze dane-xml/.

Wszystkie pliki XML mają identyczną strukturę. Każdy z nich zawiera szczegółowe dane dokładnie jednego aparatu. Na przykład plik 1.xml przedstawiony na listingu 7 zawiera szczegółowe dane Canona EOS 20D.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<aparat>
    <producent>Canon</producent>
    <model>EOS 20D</model>
    <typ>DSLR</typ>
    <megapixel>8.2</megapixel>
    <lcd>1.8</lcd>
    <matryca>COMOS</matryca>
    ...
</aparat>
```

Listing 7. Fragment pliku XML ze szczegółowymi danymi aparatu Canon EOS 20D

W treści (tj. pomiędzy znacznikami <body> i </body>) W treści strony umieszczamy serię elementów div.tresc. Jeden element div.tresc dla każdego aparatu:

```
<body>

<div class="tresc">
  <h3>
    <a href="#" onclick="expandCollapse(this, 1);">+</a>
    Canon EOS 20D
  </h3>
  <div></div>
</div>

<div class="tresc">
  <h3>
    <a href="#" onclick="expandCollapse(this, 2);">+</a>
    Canon EOS 30D
```

```
  </h3>
  <div></div>
</div>

...

</body>
```

Są to opisane wcześniej rozwijalne elementy, mające ikony plus (do rozwinięcia) oraz minus (do zwinięcia). Obsługą zdarzenia onclick zajmuje się funkcja expandCollapse(), która tym razem otrzymuje dwa parametry: obiekt DOM o nazwie this (tj. kliknięte hiperłącze) oraz liczbę, identyfikującą kliknięty aparat.

W treści funkcji expandCollapse() po pierwsze zmieniamy wygląd elementu div.tresc. Zwijamy go lub rozwijamy. Jeśli element jest rozwijany (przypadek else) dodatkowo inicjalizujemy Ajax-owy transfer danych. Parametrem metody open() jest skrypt adres URL dokumentu XML ze szczegółową specyfikacją aparatu. Nazwa pliku XML powstaje na podstawie parametru Numer, który identyfikuje kliknięty aparat:

```
function expandCollapse(Id, Numer)
{
  element = Id.parentNode.parentNode.childNodes[1];
  if (element.style.display == 'block') {
    element.style.display = 'none';
    Id.innerHTML = '+';
  } else {
    element.style.display = 'block';
    Id.innerHTML = '-';

    r.open('GET', 'dane-xml/' + Numer + '.xml', true);
    r.onreadystatechange = processResponse;
    r.send(null);

  }
}
```

Ostatnim etapem przygotowania przykładu *Aparaty fotograficzne* jest opracowanie funkcji processResponse(), która zajmie się umieszczeniem danych odebranych z serwera w rozwiniętym elemencie. Ponieważ wykorzystujemy format XML, należy użyć właściwości r.responseXML. Pobieramy wszystkie dzieci elementu o nazwie aparat:

```
var x = r.responseXML.getElementsByTagName('aparat')[0].childNodes;
```

Elementy te przetwarzamy w pętli for rozpoczynając od elementu o indeksie 2 (elementy 0 oraz 1 to nazwa firmy i nazwa modelu, które są zawarte w tytule wyświetlanego rozwijanego elementu div). Pętla for przygotowuje napis tmp, który jest wstawiony jako treść (tj. element.innerHTML) rozwiniętego elementu:

```
var element;

function processResponse()
{
  if (r.readyState == 4) {
    if (r.status == 200) {

      var x = r.responseXML.getElementsByTagName('aparat')[0].childNodes;
```

```
      var tmp = '';

      for (i = 2; i < x.length; i++) {
        tmp = tmp
          + '<strong>' + opis[i] + ':</strong> '
          + x[i].childNodes[0].nodeValue
          + '<br />';
      }

      element.innerHTML = tmp;
    };
  }
}
```

Zwróć uwagę, że zmienna element jest zmienną globalną. Po raz pierwszy pojawia się ona w funkcji expandCollapse(). Funkcja expandCollapse() umieszcza w zmiennej element rozwinięty div przeznaczony na szczegółowy opis aparatu. W ten sposób funkcja processResponse() nie musi szukać w drzewie DOM elementu, w którym należy wstawić treść. Element ten jest już przygotowany i dostępny w zmiennej element.

Etykiety podpisujące poszczególne parametry aparatu (np. *Migawka*, *Czułość*, *Autobracketing*, itd.) są zawarte w tablicy opis zadeklarowanej przed funkcją processResponse().

Zarys pliku index.html jest przedstawiony na listingu 8.

```
<head>
<script type="text/javascript">
function getXMLHttpRequest()
{
  ...
}

var r;
r = getXMLHttpRequest();

var opis= new Array(20);
opis[0]  = 'Producent';
opis[1]  = 'Model';
opis[2]  = 'Typ';
...

var element;

function processResponse()
{
  ...
}

function expandCollapse(Id, Numer)
{
  ...
}
</script>
</head>
<body>
```

```
<div class="tresc">
  <h3>
    <a href="#" onclick="expandCollapse(this, 1);">+</a>
    Canon EOS 20D
  </h3>
  <div></div>
</div>

<div class="tresc">
  <h3>
    <a href="#" onclick="expandCollapse(this, 2);">+</a>
    Canon EOS 30D
  </h3>
  <div></div>
</div>

...

</body>
```

Listing 8. Aparaty fotograficzne: zarys pliku index.html