Wprowadzenie do Internetu – zajęcia 4

Zakres tematyczny zajęć

- Rozmieszczanie elementów za pomocą **CSS**;
- **JavaScript** – wprowadzenie, zagadnienia podstawowe;
- Wykorzystanie **JavaScript**-u do kontroli formularzy.

Rozmieszczanie elementów za pomocą CSS

Wykorzystując technologię **CSS** możemy każdy z elementów umieszczonych na stronie XHTML dowolnie rozmieszczać, czyli wskazywać miejsce wyświetlenia, czy element będzie wyświetlony podczas załadowania strony, czy też będzie ukryty i uwidaczniany poprzez wykonanie określonej operacji przez końcowego użytkownika.
Do dyspozycji mamy też warstwy, które możemy nakładać na siebie (przykrywanie warstwy kolejną).
Dzięki CSS możliwe jest budowanie dynamicznych elementów na stronie XHTML np. rozwijane menu, dymki pojawiające się po najechaniu na wybrany element (DHTML)

Domyślnie wszystkie elementy na stronie mają pozycję statyczną (***position:static***), czyli układają się od lewej do prawej, od góry do dołu.

Pozycjonowanie absolutne

position:absolute

Pozycjonowanie absolutne umożliwia umieszczenie obiektu na stronie **XHTML** w określonym miejscu poprzez wskazanie właściwości: **top, right, bottom, left**.
Zamieszczenie takiego obiektu (kod xhtml) może nastąpić w dowolnym momencie w sekcji **BODY** ponieważ obiekt ten jest wyjęty z biegu dokumentu i nie jest wyświetlany w miejscu wpisania go tylko tam gdzie wskazują właściwości: **top, right, bottom, left**.

Jego pozycja jest względna do elementu zawierającego, którym domyślnie jest <body>.

Przykład:

```
<html>
<head>
	<title>Pozycjonowanie absolutne</title>
</head>
<body>
	<p style="padding-top:50px;">Tytuł</p>
	<div style="position:absolute;top:20px;">Obiekt pozycjonowany
absolutnie</div>
</body>
</html>
```

Ustawienie dwóch właściwości poziomych lub pionowych rozciąga element „od-do”.

```
position:absolute; left:20px; right:20px; top: 0;
```

Pozycjonowanie relatywne

position:relative

Przesuwa obiekt o ofset ustalony we właściwościach top i left.

Obiekt pozostaje w tym samym miejscu w biegu dokumentu. Zmienia się tylko miejsce, w którym jest rysowany.

Przykład:

```
<html>
<head>
	<title>Pozycjonowanie relatywne</title>
</head>
<body>
	<p style="padding-top:50px;">Tytuł</p>
	<div style="position:relative;top:-60px;">Obiekt pozycjonowany relatywnie</div>
</body>
</html>
```

Tworzenie warstw na stronie

z-index

W celu utworzenia warstwy na stronie XHTML należy posłużyć się atrybutem z-index, który określa numer warstwy (kolejność elementu na stosie)

```
<html>
<head>
<style type="text/css">
img.x
{
position:absolute;
left:0px;
top:0px;
z-index:-1
}
</style>
</head>

<body>
<h1>Nagłówek 1</h1>
<img class="x" src="obrazek.gif" width="100" height="180">
<p>Standardowy z-index posiada wartość 0. Z-index -1 na niższy priorytet i w związku z tym jest ten tekst wyświetli sięna obrazku.</p>
</body>

</html>
```

Wygląd formularza

```
input, textarea, select
{
background-color: silver;
font-family: Verdana;
color: black;
font-size: 8pt;
border-color: black;
}

input.przycisk
{
background-color: #8f8f8f;
font-family: Verdana;
color: white;
font-size: 12pt;
}
```

Więcej informacji

- **http://algorytmy.pl**
- **http://www.strefawww.pl/cms/**
- **http://www.w3schools.com/css/default.asp**

JavaScript – wprowadzenie, zagadnienia podstawowe

Technologia **JavaScritp** jest wykorzystywana niemal na każdej stronie html, xhtml w celach: wzbogacenia funkcjonalności strony, jej wyglądu, walidowania formularzy, wykrywania przeglądarki, tworzenia ciasteczek (cookies) itp.
JavaScript obecnie jest jednym z najpopularniejszych języków skryptowych i działa niemal z wszystkimi dostępnymi przeglądarkami internetowymi Internet Explorer, Mozilla, Firefox, Netscape, Opera.

Czym jest JavaScript:

- Technologia JavaScript została stworzona w celu dodania interaktywności do stron HTML
- JavaScript jest językiem skryptowym
- Zawiera instrukcje wykonywane przez przeglądarki
- Skrypty JavaScript są dołączane do strony HTML i przesyłane do klienta
- JavaScript jest językiem interpretowanym (instrukcję języka są wykonywane bez wcześniejszej kompilacji)

UWAGA! JavaScirpt i Java są dwoma zupełnie różnymi językami.

Zamieszczanie skryptów JavaScript na stronie

Skrypty JavaScript możemy zamieścić w dowolnym miejscu na stronie, jednak zalecane jest dołączanie ich w nagłówku strony czyli sekcji <head>.
Najlepiej przyjąć zasadę, że skrypty dołączane na każdej podstronie serwisu dołączamy w sekcji <head> natomiast, jeżeli potrzebne nam są dodatkowe skrypty to możemy je zamieścić w sekcji <body>. W przypadku użycia technologii działającej po stronie serwera, możemy posłużyć się zmienną, na którą będziemy zapisywali skrypty dla poszczególnych stron i przekazywali do szablonu, który wyświetli je w odpowiednim miejscu.

Zamieszczenie skryptu JavaScript w treści strony

```
<?xml version="1.0" encoding="windows-1250"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>

        <title> JavaScript </title>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1250" />
        <script language="JavaScript" type="text/javascript">
        <!--
        Treść skryptu (instrukcje języka)
        //-->
        </script>
</head>

        <body>



        </body>

</html>
```

Uwaga!!! Skrypty te są każdorazowo ściągane przez przeglądarkę co wpływa na szybkość wczytywania strony.

Dołączanie zewnętrznego pliku JavaScript do strony XHTML

W przypadku załączania zewnętrznego pliku ze skryptami musimy po pierwsze stworzyć ten plik. Pliki JavaScript są zwykłymi plikami tekstowymi z rozszerzeniem *.js

Plik „skrypt.js”

W samym pliku nie musimy już zawierać Tagów **<script>**. Wpisujemy po prostu kolejno instrukcję języka JavaScirpt.

W dokumencie XHTML musimy zamieścić odwołanie do pliku, gdzie podajemy ścieżkę dostępu do pliku oraz jego nazwę z rozszerzeniem.

```
<?xml version="1.0" encoding="windows-1250"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>

        <title> JavaScript </title>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-1250" />
        <script language="JavaScript" type="text/javascript" src=" include/skrypt.js"></script>
</head>

        <body>



        </body>

</html
```

Uwaga!!! Skrypty te są ściągane przy pierwszym żądaniu i przechowywane w pamięci cache przeglądarki internetowej.

Podstawy JavaScript

- **Komentarze**

```
// komentarz w jednej linii


/* komentarz w
   wielu liniach */
```

- **Wyświetlenie tekstu - document.write**

```
<html>
<body>
<script type="text/javascript">
document.write("Pierszy tekst wyświetlony przez JavaScirpt!");
</script>
</body>
</html>
```

- **Definiowanie zmiennych**

  W celu zdefiniowanie zmiennej podajemy słowo kluczowe var

  ```
  var zmienna = "wartosc zmiennej";
  ```

  Można również definiować zmienne bez użycia słowa kluczowego var

  ```
  zmienna = "wartosc zmiennej";
  ```

UWAGA!!! Po każdej instrukcji w języku JavaScript podajemy średnik.

Zasięg zmiennych jest uzależniony od miejsca definicji zmiennej. Przykładowo, jeżeli deklarujemy zmienną w skrypcie jest ona widoczna na całej stronie i w funkcjach. W przypadku deklaracji zmiennych w funkcjach, są one widoczne tylko i wyłącznie w danej funkcji i ich stworzenie odbywa się podczas wywołania funkcji. Po wykonaniu się funkcji zmienne są niszczone.

Operatory

Zestawienie operatorów używanych w JavaScript:

Operatory Arytmetycze

Operator	Opis	Przykład	Wynik
+	Dodawanie	x=3 x=x+4	7
-	Odejmowanie	x=4 x=6-x	2
*	Mnożenie	x=3 x=x*5	15
/	Dzielenie	10/5 9/2	2 4.5
%	Modulo (reszta z dzielenia)	4%3 12%8 8%2	1 4 0
++	Zwiększanie o 1	x=2 x++	x=3
--	Zmniejszanie o 1	x=4 x--	x=3

Operatory przypisania

Operator	Przykład	Równoważne z
=	x=y	
+=	x+=7	x=x+7
-=	x-=3	x=x-3
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Operatory porównania

Operator	Opis	Przykład
==	jest równe	2==3 wynik:fałsz
!=	nie jest równe	2!=3 wynik:prawda
>	jest większe	25>3 wynik:fałsz
<	jest mniejsze	2<3 wynik:prawda
>=	większe lub równe	25>=3 wynik:fałsz
<=	mniejsze lub równe	2<=3 wynik:prawda

Operatory logiczne

Operator	Opis	Przykład
&&	i	x=3 y=4 (x < 9 && y > 2) wynik:prawda
||	lub	x=3 y=4 (x==8 || y==6)

		wynik:fałsz
!	zaprzeczenie	x=3 y=4 !(x==y) wynik:prawda

- **Instrukcje warunkowe if/else**

  W wielu sytuacjach podczas pisania skryptów napotykamy się z sytuacją, gdy musimy dokonać jakiegoś wyboru w zależności od jakiegoś warunku.

  Np. Jeżeli zmienna nie jest pusta dokonujemy określonej czynności

```
<html>
<body>
<script type="text/javascript">
// wykrycie czy przeglądarka obsługuję technologię DOM – document object model
var isDom = (document.getElementById ? true : false);

if(isDom)
    document.write("Twoja przeglądarka obsluguje document object model!");
else
    document.write("Twoja przeglądarka nie obsluguje document object model!");

</script>
</body>
</html>
```

  Jeżeli zmienna isDom będzie posiadała wartość true warunek będzie spełniony i zostanie wyświetlony komunikat o obsludze DOM. W innym przypadku zostanie wyświetlony tekst z instrukcji else.
  W przypadku, gdy w instrukcji zawieramy więcej niż 1 instrukcję to musimy je zawrzeć jako blok kodu poprzez dodanie nawiasów {}
  Np.

```
if(isDom){
    instrukcja 1;
    instrukcja 2;
}
else
    instrukcja 1;
```

  Jeżeli tego nie zrobimy zostanie wykonana instrukcja zawarta zaraz po instrukcji if a następna instrukcja 2, która zostanie potraktowana jako instrukcja spoza if.

```
<html>
<body>

<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
document.write("<b>Good morning</b>")
}
else if (time>=10 && time<16)
{
document.write("<b>Good day</b>")
}
else
{
document.write("<b>Hello World!</b>")
}
</script>

<p>
Ten przykład demonstuje działanie instukcji warunkowej if..else if...else.
```

```
</p>

</body>
</html>
```

- **Instrukcje switch**
  Instrukcji tej używamy, jeżeli chcemy wykonać jeden z wielu bloków kodu w zależności od wartości zmiennej.

```
switch (zmienna) {
case wartosc1:
kod wykonywany gdy zmienna=wartosc1
break
case wartosc2:
kod wykonywany gdy zmienna=wartosc2
break
// ... itd
case wartoscX:
kod wykonywany gdy zmienna=wartoscX
break
default:
kod wykonywany, gdy żadnej z powyższych wartości nie przyjmuje zmienna
}
```

```
<html>
<body>
<script type="text/javascript">
var data=new Date() // tworzony jest obiekt z datą
dzien=data.getDay() // pobieramy numer dnia
switch (dzien)
{
case 0:
    document.write("Dzisiaj jest Niedziela a Ty siedzisz na uczelni i piszesz skrypty");
case 1:
    document.write("Dzisiaj jest Poniedziałek");
case 2:
    document.write("Dzisiaj jest Wtorek");
    break;
case 6:
    document.write("Dzisiaj jest Sobota a Ty siedzisz na uczelni i piszesz skrypty");
    break;

    break;
default:
    document.write("Co to za dzień dzisiaj mamy?")
}
</script>
</body>
</html>
```

- **Operator warunkowy**

  Operator warunkowy pozwala czasami oszczędzić miejsce, gdyż doskonale zastępuje prostą konstrukcję if-else w której mamy w zależności od warunku pojedynczą reakcję:

  zmienna=(warunek)?wartoscTRUE:wartoscFALSE

  Działa to w ten sposób, że jeżeli warunek jest spełniony, to do zmiennej przypisywana jest wartość TRUE, w przeciwnym wypadku wartość FALSE

```
liczba=prompt("podaj jakąś liczbę:");
jaka=(liczba%2==0)?"parzysta":"nieparzysta";
document.write("podana liczba jest "+ jaka) ;
```

- **Pętla while**

W przypadku, gdy pewną część kodu chcemy wykonać wielokrotnie uzależniając wykonanie od pewnego warunku, możemy posłużyć się pętlami.

Kod w pętli while jest wykonywany przez cały czas, gdy warunek jest spełniony:

```
while (warunek) {
      kod pętli
}
```

Przykład

```
<?xml version="1.0" encoding="windows-1250"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>

    <body>

            <script language="JavaScript" type="text/javascript">
            <!--
            liczba=10;

            while(liczba >= 0) {
                  document.write(liczba);
                  liczba --;
            }

            //-->
            </script>

    </body>

</html>
```

- **Pętla do-while**

Pętla ta jest bardzo podobna do pętli while. Różnicą pomiędzy tymi dwoma pętlami jest sposób wykonania. Pętla while wykona się tylko wtedy, gdy zawarty w niej warunek jest spełniony. Pętla do-while wykona się przynajmniej raz, gdyż na samym jej początku wykonujemy kod pętli bez sprawdzenia warunku. Czyli może się zdarzyć sytuacja, gdy warunek nie będzie spełniony, a mimo to pętla zostanie wykonana raz. W pętli tej warunek jest sprawdzany na końcu.

```
do {
kod wykonywany w pętli
} while (warunek)
```

Przykład

```
<?xml version="1.0" encoding="windows-1250"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>

    <body>

            <script language="JavaScript" type="text/javascript">
            <!--
            liczba=0;

            do {
                    document.write(liczba);
                    liczba --;
            } while(liczba >= 0)

            //-->
            </script>

    </body>

</html>
```

- **Pętla for**

Pętla for umożliwia wykonywanie pewnego bloku kodu określoną ilość razy.

```
for(inicjalizacja_zmiennej; warunek; zmiana_zmiennej) {
    kod wykonywany w pętli
}
```

Przykład

```
<?xml version="1.0" encoding="windows-1250"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>

    <body>

            <script language="JavaScript" type="text/javascript">
            <!--

            do {
            procent=parseInt(prompt("Podaj procent zawartości (0-100):",""));
            } while (procent<0 | procent>100 | isNaN(procent))

            //parseInt() - sprawdza czy przekazane dane od uzytkownika sa liczba, jak nie
zwraca NaN (Not a Number)
            //isNaN(procent) - sprawdzenie w warunku czy przekazano liczbe (jak warunek
jest spelniony, wyjscie z petli)

            for(i=0; i < procent; i ++){

                    document.write("<span style=\"color:#FF0000;\">|</span>");
            }

            document.write(" "+procent+" % ");
            //-->
            </script>

    </body>

</html>
```

- **Obiekty JavaScript**
  Array – tablice

  Obiekt array jest używany do przechowywania zbioru wartości, które występują kolejno po sobie i do których możemy odwołać się po indeksie tablicy.

  Deklaracja tablicy

```
// 1 - sposób
var liczbySlownie=new Array("jeden","dwa","trzy");
```

```
// 1 - sposób
var liczbySlownie =new Array(3);
liczbySlownie[0]="jeden";
liczbySlownie[1]="dwa";
liczbySlownie[2]="trzy";
```

```
liczbySlownie.length // - pobranie liczby elementów w tablicy
delete.liczbySlownie[2] // - usunięcie elementu z tablicy
```

Obiekt Date

Jeżeli chcemy posłużyć się datą lub godziną musimy do tego celu wykorzystać obiekt Date.
W celu wywołania obiektu musimy posłużyć się konstruktorem

```
var data = new Date();
```

Dopiero po stworzeniu obiektu możemy odwołać się do jego metod i właściwości w celu uzyskania daty lub godziny.

Metody

nazwa	opis
.getTime()	Liczba milisekund od 1970-01-01 od godziny 00:00:00 czasu Greenwich
.getTimezoneOffset()	Zwraca różnicę czasu lokalnego i GMT
.getYear()	Określony rok minus 1900;czterocyfrowe oznaczenia dla roku 2000 i kolejnych
.getFullYear()	Rok w postaci 4-cyfrowej
.getUTCFullYear()	Rok w postaci 4-cyfrowej czasu uniwersalnego
.getMonth()	Miesiąc roku (0-11)
.getUTCMonth()	Miesiąc roku czasu uniwersalnego
.getDate()	Dzień miesiąca (1-31)
.getUTCDate()	Dzień miesiąca czasu uniwersalnego
.getDay()	Dzień tygodnia (niedziela=0) (0-6)
.getUTCDay()	Dzień tygodnia czasu uniwersalnego
.getHours()	Godzina dnia w zapisie 24-godzinnym (0-23)
.getUTCHours()	Godzina dnia czasu uniwersalnego
.getMinutes()	Minuta godziny (0-59)
.getUTCMinutes()	Minuta godziny czasu uniwersalnego
.getSeconds()	Sekunda minuty (0-59)
.getUTCSeconds()	Sekunda minuty czasu uniwersalnego
.getMilliseconds()	Milisekunda sekundy (0-999)
.getUTCMilliseconds()	Milisekunda sekundy czasu uniwersalnego

.setTime(val)	Ustawia liczbę milisekund od 1970-01-01 od godziny 00:00:00 czasu Greenwich
.setYear(val)	Ustawia rok minus 1900; czterocyfrowe oznaczenia dla roku 2000 i kolejnych
.setFullYear(val)	Ustawia rok w postaci 4-cyfrowej
.setUTCFullYear(val)	Ustawia rok w postaci 4-cyfrowej czasu uniwersalnego
.setMonth(val)	Ustawia miesiąc roku (0-11)
.setUTCMonth(val)	Ustawia miesiąc roku czasu uniwersalnego
.setDate(val)	Ustawia dzień miesiąca (1-31)
.setUTCDate(val)	Ustawia dzień miesiąca czasu uniwersalnego
.setDay(val)	Ustawia dzień tygodnia (niedziela=0) (0-6)
.setUTCDay(val)	Ustawia dzień tygodnia czasu uniwersalnego
.setHours(val)	Ustawia godzinę dnia w zapisie 24-godzinnym (0-23)
.setUTCHours(val)	Ustawia godzinę dnia czasu uniwersalnego
.setMinutes(val)	Ustawia minutę godziny (0-59)
.setUTCMinutes(val)	Ustawia minutę godziny czasu uniwersalnego
.setSeconds(val)	Ustawia sekundę minuty (0-59)
.setUTCSeconds(val)	Ustawia sekundę minuty czasu uniwersalnego
.setMilliseconds(val)	Ustawia milisekundę sekundy (0-999)
.setUTCMilliseconds(val)	Ustawia milisekundę sekundy czasu uniwersalnego
.parse()	Zwraca string z datą - jako ilość milisekund od 1 stycznia 1970 00:00:00
.toGMTString()	Zwraca stringa z datą ustawioną na GMT
.toLocaleString()	Zwraca stringa z datą lokalną
.toString()	Zwraca stringa z datą

- **Funkcje**

Funkcja jest oddzielnym blokiem kodu, który może być wielokrotnie wykonywany w danym programie, poprzez jej wielokrotne wywoływanie. Do funkcji przekazujemy przeważnie jakieś argumenty, a funkcja może nam zwracać jakąś wartość. Dobrze jest tworzyć funkcję tak, aby wykonywała jedno określone zadanie - czyli większe operacje w programie rozdzielamy na kilka wywoływanych kolejno funkcji. Dzięki temu tworzymy cegiełki, z których budujemy potem cały skrypt, a które możemy wykorzystać w innych skryptach (gdy zapiszemy je w oddzielnych plikach - o czym była mowa już wcześniej).

Funkcję możemy wywołać w dowolnym momencie pod warunkiem, że wcześniej została zdefiniowana.

Definicja funkcji

```
function nazwa_funkcji(argument1, argument2, itd) {
      kod wykonywany przez funkcję
}
```

Wywołanie funkcji

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>

        <head>
        <script language="JavaScript" type="text/javascript">
        <!--
        function nazwa_funckcji(){
                document.write("Pierwsza funkcja");
        }
        function nazwa_funckcji2(komunikat){
                document.write(komunikat);
        }
        function alert_uzytkownik(tresc){
                alert(tresc);
        }
        //-->
        </script>
        </head>
        <body>

        <script language="JavaScript" type="text/javascript">
        <!--
        nazwa_funckcji();
        nazwa_funckcji2("<br />Przykład wywołania funkcji z argumentem");
        //-->
        </script>
```

```
        <button onClick="javascript:alert_uzytkownik('Kliknołeś na mnie :)');">kliknij
mnie</button>


        </body>

</html>
```

Funkcja zwracająca wartość.

Znaczna część funkcji jakie stworzysz, będzie otrzymywała jakieś dane, przetwarzała je i zwracała wynik z powrotem do programu. Napiszmy funkcję dodającą dwie liczby:

```
function suma(liczba1, liczba2) {
sumaliczb=liczba1+liczba2; // dodaje liczby i przypisuje nowej zmiennej
wynik=sumaliczb;
return wynik; // zwraca zmienną wynik
}
```

w programie wywołamy funkcję tak:

```
a=1;
b=2;
c=suma(a,b); // funkcja zwraca wartość, która jest podstawiana do zmiennej c
document.write("c="+c+" to samo co: "+ suma(a,b));
```

Inne

- **Zaokrąglanie wyniku** w funkcji obliczWartość

  suma = Math.round(100*suma) / 100;

- **Zabezpieczenie przed nieprawidłowymi danymi**

  if (isNaN(suma))
  {
  window.alert("Proszę podać wartości będące liczbą");
  return false;
  }

- **Operacje na warstwach**

  Warstwa ukryta

  Pokaż warstwę

```
<a href="javascript:ukryjWarstwe()">Ukryj warstwę</a>

function pokazWarstwe()
{
        document.getElementById("warstwa").style.display = "block";
}

function ukryjWarstwe()
{
        document.getElementById("warstwa").style.display = "none";
}
```

- **obsługa zdarzeń myszki**

```
<div id="warstwa2" style="background-color: red; height:100px;"
onMouseOver="zmienKolor()" onMouseOut="wrocKolor()">
<br><BR><center>Warstwa druga</center>
</div>

function zmienKolor()
{
        document.getElementById("warstwa2").style.backgroundColor = "green";
}

function wrocKolor()
{
        document.getElementById("warstwa2").style.backgroundColor = "red";
}
```

- **dynamiczna podmiana zdjęć**

```
<img src="z1.jpg" id="zdjecie" onMouseOver="podmienZdjecie()"
onMouseOut="wrocZdjecie()" style="cursor: hand" alt="Zdjecie">

function podmienZdjecie()
{
        document.getElementById("zdjecie").src = "z2.jpg";
}

function wrocZdjecie()
```

```
{
        document.getElementById("zdjecie").src = "z1.jpg";
}
```

Tworzenie formularzy

Na stronie XHTML możemy zamieszczać wiele formularzy. Trzeba jednak pamiętać, że w odróżnieniu od innych elementów XHTML formularzy nie można zagnieżdżać.

Aby na stronie zdefiniować formularz należy posłużyć się znacznikiem <form>

<!-- definicja formularza -->
<form metod=”post” action=”strona.html” name=”forma” target=”_self”>
Elementy formularza
</form>

Atrybuty opcjonalne

Atrybut	Wartość
method	**get** **post**
name	**form_name**
target	**_blank** **_self** **_parent** **_top**

Sama definicja formularza to pierwszy krok. Kolejnym jest zdefiniowanie elementów formularza

Elementy formularzy:

- **pole tekstowe:**

```
<input type="text" name="nazwisko">
z parametrami:
<input type="text" name="nazwisko" value="Kowalski" size="10" maxlength="20" />
```

- **pole hasła:**

```
<input type="password" name="haslo" />
parametry jak dla pola tekstowego
```

- **lista rozwijalna zwykła:**

```
<select name="kolor">
        <option value="1">czerwony</option>
        <option value="2">zielony</option>
        <option value="3">niebieski</option>
</select>
```

- **lista rozwijalna wielowyboru:**

```
<select name="kolor" size="3" multiple>
        <option value="1">czerwony</option>
        <option value="2">zielony</option>
        <option value="3">niebieski</option>
</select>
```

- **przycisk checkbox:**

```
<input type="checkbox" name="przetwarzanie" />Tak, zagadzam się na przetwarzanie moich danych
```

- **przyciski radiowe:**

```
<input type="radio" name="monitor" value="lcd" checked />LCD
<input type="radio" name="monitor" value="crt" />CRT
```

- **pole textarea:**

```
<textarea name="opis" rows="3" cols="30">Oopppiiiisss ... </textarea>
```

- **przycisk submit (potwierdzenia):**

```
<input type="submit" value="Wyślij" />
```

- **przycisk zwykły:**

<input type="button" value="Przyciśnij..." />

- **pole ukryte:**

```
<input type="hidden" name="poleUkryte" value="jakaś wartość" />
```

- **przycisk czyszczenia/przywrócenia wartości domyślnych"**

<input type="reset" value="Wartości domyślne" />

- **button graficzny**

```
<input type="image" src="przycisk.jpg" alt="Wyslij" />
```

Wykorzystanie JavaScript-u do kontroli formularzy.

Formularze na stronie XHTML są wykorzystywane do interakcji z użytkownikami. Za ich pomocą możemy zbierać dane od użytkownika i odpowiednio na nie reagować. Sama technologia XHTML nie daje nam zbyt dużo możliwości obsługi formularzy. Dopiero połączenie ich potencjału z technologiami działającymi po stronie serwera np. PHP daje nam olbrzymie możliwości przetwarzania danych przekazanych od użytkowników i ich zapis w bazach danych, plikach tekstowych itp…

Odwołanie przy pomocy JavaScript do elementu formularza.

Aby odwołać się do elementów formularza musimy wziąć pod uwagę, iż sam formularz jest umieszczony na stronie XHTML. Odwołanie do samego dokumentu odbywa się przez obiekt „document". Następnie do formularza możemy odwołać się w dwojaki sposób:

1. Przez nazwę formularza
Np. document.nazwaFormularza

2. Przez tablicę (kolekcja formularzy na stronie)
Np. document.forms[0]

Wiemy już jak odwołać się do samego formularza. Ale co z elementami? Czy możemy odczytywać wartości tych elementów? Czy możemy je zmieniać?

Na wszystkie powyższe pytania można odpowiedzieć jednym słowem – TAK

Aby odwołać się do pola formularza mamy dwie możliwości:
1. Poprzez jego nazwę
Np. dokument.nazwaFormularza.nazwaElementu.value

2. Poprzez tablicę kolekcję(elementów formularza)
Np. dokument.nazwaFormularza.elements[0].value

Przykłady:

```
<script language="javascript">

function obliczWartosc()
{
	var suma;

	suma =  document.zamowienie.ilosc.value * document.zamowienie.cena.value;

	document.zamowienie.wartosc.value = suma;
}

</script>

<form name="zamowienie">

Ilość: <input type="text" name="ilosc" size="5" /><br />
Cena: <input type="text" name="cena" size="5" /><br />
Wartość: <input type="text" name="wartosc" size="15" value="Kliknij przycisk" readonly />
<input type="button" value="Oblicz wartość" onclick="obliczWartosc()" />

</form>
```