Wykład 7
Skrypty typu JavaScript

Technologie internetowe

Zofia Kruczkiewicz

http://www.w3schools.com/js/default.asp

JavaScript:

- JavaScript **wprowadza interaktywność** do stron HTML w celu:
  - **walidacji** danych przed wysłaniem do serwera www (ogranicza ruch w sieci w przypadku wprowadzanie danych o niewłaściwym formacie)
  - **obługuje zdarzenia** – czyli reaguje na „klikanie” na elementy strony
  - **wzbogaca funkcjonalność** stron www - może dostosować prezentację do możliwości rozpoznanej przeglądarki
  - wpływa na **prezentację elementów** strony uzupełniając kaskadowe arkusze stylów
  - umożliwia zarządzanie **cookies.**
- JavaScript jest **językiem skryptowym** – wprowadza lekki styl programowania
- Skrypty JavaScript zazwyczaj **są wbudowane** do stron HTML
- JavaScript jest **językiem interpretowanym**
- Można używać JavaScript **bez zakupu licencji**
- **Java i JavaScript są różnymi językami** pod względem koncepcji i implementacji

http://www.w3schools.com/js/default.asp

- JavaScript jest implementacją standardu języka **ECMAScript. ECMA-262** jest oficjalnym standardem języka JavaScript.
- **JavaScript** został zdefiniowany przez Brendan Eich i zastosował w przeglądarce **Netscape Navigator 2.0**, i został wprowadzony do wszystkich przeglądarek od 1996.
- Oficjalny standard został przyjęty przez organizację **ECMA w 1997**.
- **ECMA-262** definiuje semantykę języka oraz standardowe typy danych (String, Boolean, Number, Object itp.) i obiekty (np. Math, Array). Elementy dotyczące modelu dokumentu, funkcji wejścia-wyjścia, elementów strony są definiowane są przez standard W3C DOM lub autorów konkretnych implementacji.
- **Standard ECMAScript-262** został wprowadzony jako międzynarodowy standard **ISO/IEC 16262** w 1998.
- Standard jest w ciągłym rozwoju

1. Podstawowe elementy języka JavaScript

1.1. Komentarze

- Komentarz blokowy: między znakami /* */

```
/*  to
        jest treść komentarza
*/
```

- Komentarz liniowy:

od znaku // do znaku końca linii;

```
// to jest komentarz liniowy
```

1.2. Operatory

http://www.programmershelp.co.uk/docs/javascript/ops.html#1003191

Operatory przypisania

Operatory łańcuchowe

Operatory arytmetyczne

- % (reszta z dzielenia)
- ++ (Inkrementacja)
- -- (Dekrementacja)
- - (Negacja jednoargumentowa)

Operatory bitowe

- Logiczne operatory bitowe
- Operatory przesunięcia bitowego

Operatory porównania

Operatory logiczne

Operatory specjalne:

- ?: (Operator alternatywy)
- , (Operator przecinka)
- delete
- function
- in
- instanceof
- new
- this
- typeof
- void

1.3. Definiowanie zmiennych – instrukcja var

var *nazwazmiennej1* [= wartosc] [..., *nazwazmiennej2* [= wartosc]]

Zmienne:

- Należy definiować za pomocą liter dużych lub małych oraz cyfr, jednak pierwszym znakiem musi być litera
- Rozróżnia się duże i małe litery

Zasięg zmiennych

- zmienne definiowane wewnątrz funkcji są **lokalne**
- definiowane poza funkcją są **globalne** (usuwane są po zamknięciu strony)
- zmienne zadeklarowane bez słowa var stają się **globalnymi**
- zmienne niezadeklarowane po przypisaniu wartości stają się **globalnymi**

Przykład

```
var num_1 = 0, num_2 = 0
name ="Kowalski";
```

1.4. Definiowanie stałej

Deklarowanie stałej bez możliwości zmiany jej wartości

const ***nazwa1*** **[= wartosc] [...,** ***nazwa2*** **[= wartosc]]**

Przykład

```
const a = 7;
document.writeln("a jest równa " + a );  // a jest równa 7
```

1.5. Instrukcje warunkowe

Instrukcja if

```
if (warunek)
    { instrukcje; }
```

Instrukcja if ..else

```
if (warunek)
    { instrukcje; }
else
    { instrukcje;}
```

Instrukcja if...else if...else

```
if   (warunek)
    { instrukcje; }
else if
    { instrukcje;}
else
    { instrukcje;}
```

Instrukcja switch

Składnia

```
switch (wyrażenie)
  {
    case etykieta1 :     instrukcje;
          break;
    case etykieta2 :      instrukcje;
          break;
    ...
    default : instrukcje;
  }
```

Przykład:

```
switch (i)
{   case "Ksiazki" :          document.writeln("Wybrano ksiazki.");
        break;
    case "Czasopisma" :  document.writeln("Wybrano czasopisma.");
        break;
    default :                   document.writeln("Taki wybór nie istnieje " + i + ".");
}
    document.writeln(" Czy dokonałeś właściwego wyboru?");
```

1.6. Okienko typu alert

Składnia

```
alert("sometext");
```

1.7. Okienko dialogowe z przyciskami Ok i Cancel

Składnia

```
var input1 = confirm("sometext");
```

input równe true, gdy naciśnięto przycisk OK.

input równe false, gdy nacisnięto przycisk Cancel

1.8. Okienko dialogowe z polem wejściowym z przyciskami Ok i Cancel (wartość pola wejściowego równa null)

Składnia

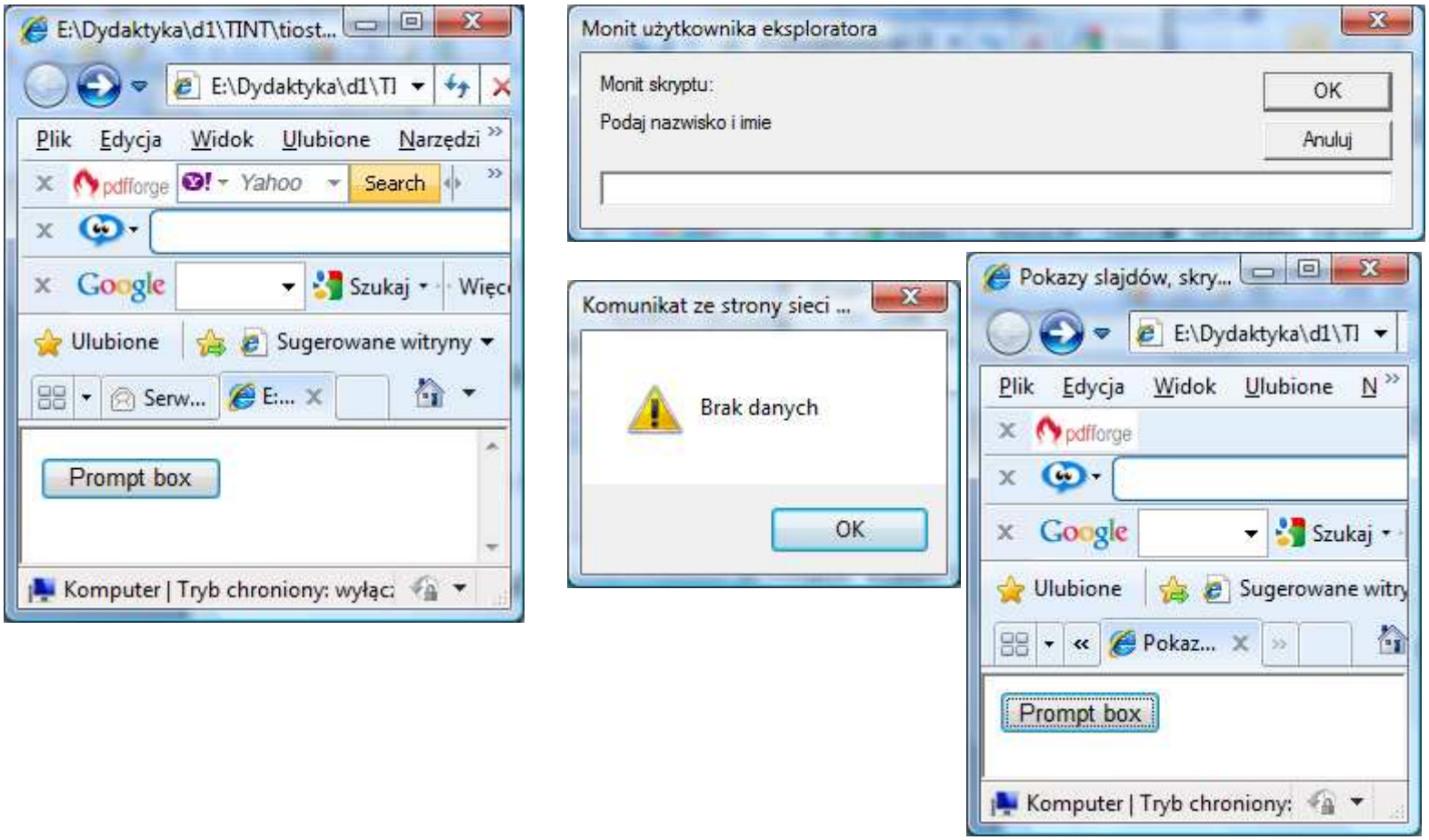
```
var input1 = prompt("sometext","defaultvalue");
```

Przykład 1: zastosowanie okienek: alert, prompt i confirm

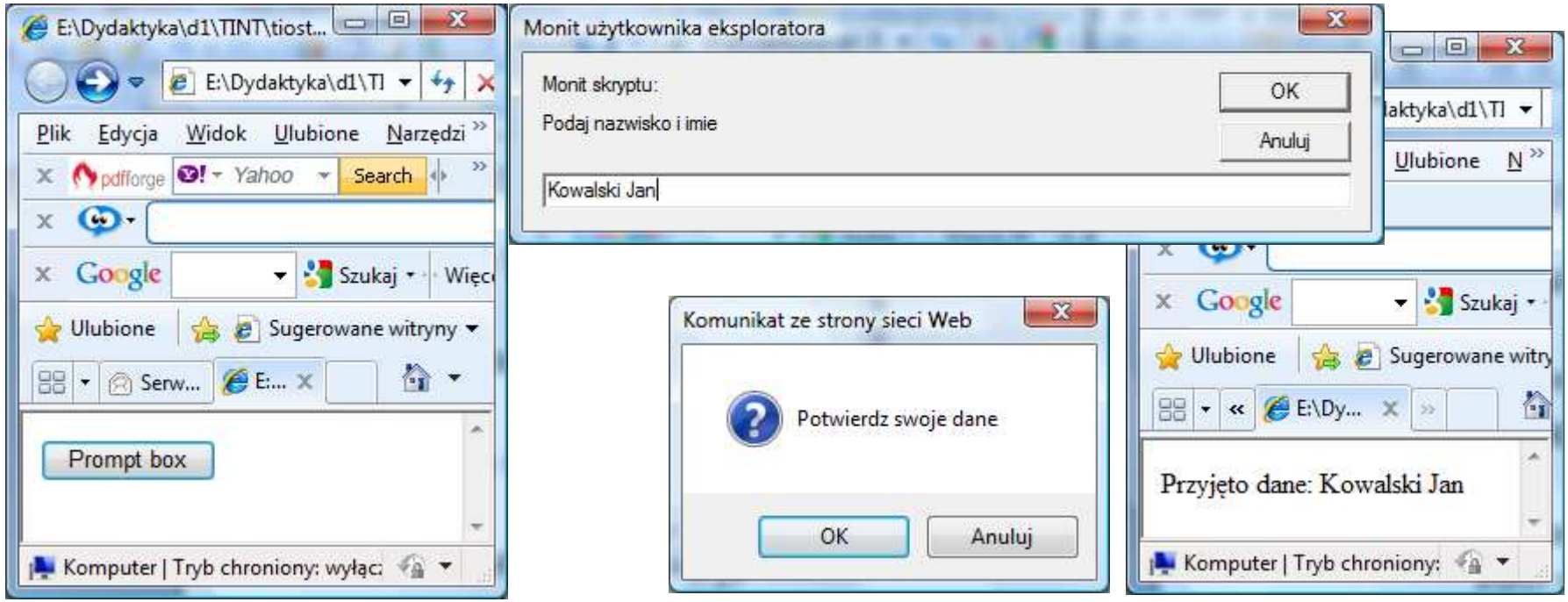
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>
<title>Skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
//<![CDATA[
function alert_prompt_confirm()
{
    var name=prompt("Podaj nazwisko i imie","");
    if (name==null || name=="")
       {   alert("Brak danych ");  }                    //1
    else
       {  var ok = confirm("Potwierdz swoje dane");                          //2, 3
          if (ok ==true)      document.write("Przyjęto dane: "+name); //2
          else                document.write("Anulowano dane");       //3
        }
}
//]]>
</script>
</head>
```

```
<body>
    <input type="button" onclick="alert_prompt_confirm()"
           value="Prompt box" />
</body>
</html>
```

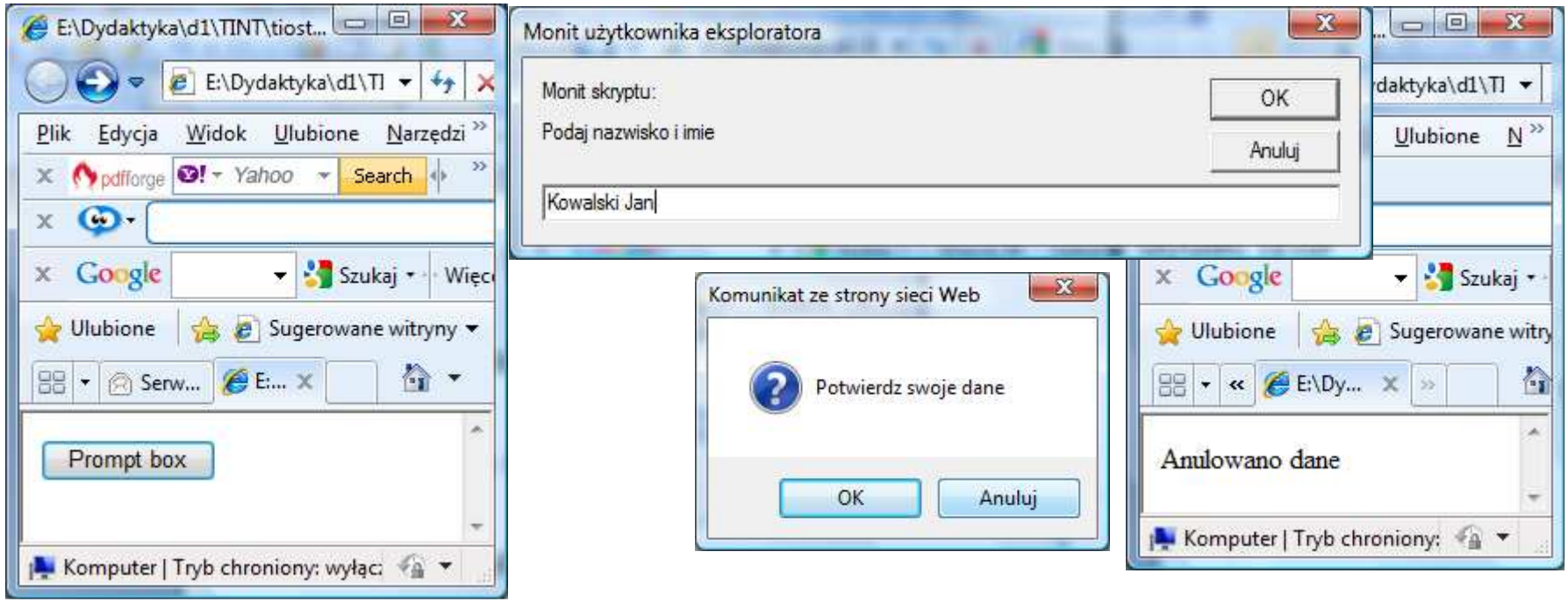
1



2


E:\Dydaktyka\d1\TINT\tiost...
Plik Edycja Widok Ulubione Narzędzi
Yahoo Search
Google Szukaj
Ulubione Sugerowane witryny
Prompt box
Komputer | Tryb chroniony: wyłąc:
Monit użytkownika eksploratora
Monit skryptu:
Podaj nazwisko i imie
OK
Anuluj
Kowalski Jan
Komunikat ze strony sieci Web
Potwierdz swoje dane
OK
Anuluj
Przyjęto dane: Kowalski Jan
Komputer | Tryb chroniony:

3


E:\Dydaktyka\d1\TINT\tiost...
Plik Edycja Widok Ulubione Narzędzi
Yahoo Search
Google Szukaj
Ulubione Sugerowane witryny
Prompt box
Komputer | Tryb chroniony: wyłąc:
Monit użytkownika eksploratora
Monit skryptu:
Podaj nazwisko i imie
OK
Anuluj
Kowalski Jan
Komunikat ze strony sieci Web
Potwierdz swoje dane
OK
Anuluj
Anulowano dane
Komputer | Tryb chroniony:

1.9. function

Definiowanie funkcji ze specyfikacją parametrów typu łańcuchowego, liczbowego oraz obiektowego.

Składnia

function *nazwa*([*param*] [, *param*] [..., *param*])
{ *instrukcje* }

Przykład 1 – deklarowanie funkcji

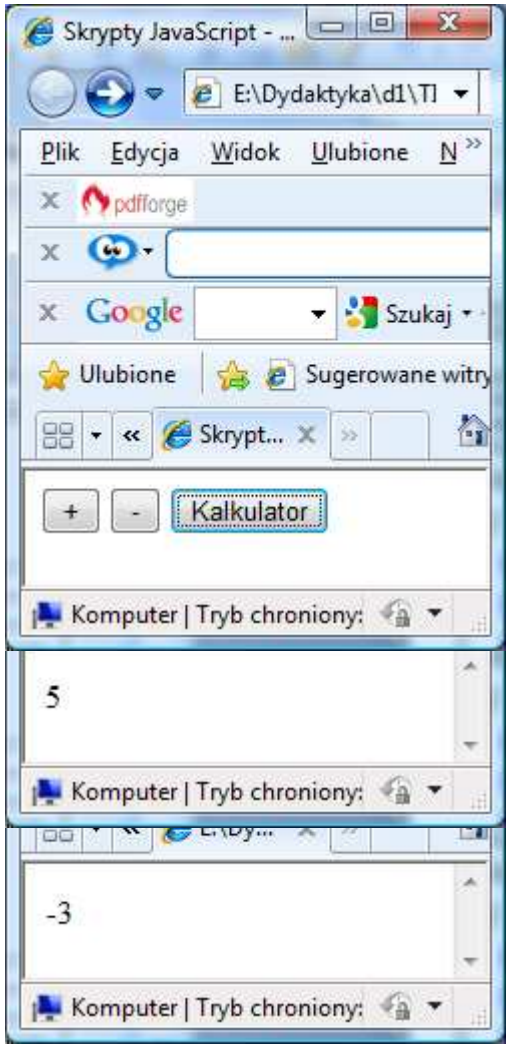
```
function cena_brutto(cena, podatek)
   { return cena*(1+podatek/100); }
```

Przykład 2 - W JavaScript operuje się na referencjach do funkcji:

```
function f1(a,b)       { return a+b;}
function f2(a,b)       { return a-b; }
function kalk(){
    switch(i)
     {  case  '+' : g=f1; break;
        case   '-' : g=f2; break;
     }
   document.writeln(g(1, 4)); }
```

Przykład 2: Zastosowanie referencji do funkcji oraz instrukcji switch

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
// <![CDATA[
  function f1(a,b)       { return a+b;}
  function f2(a,b)       { return a-b; }
  function kalk(co){
       switch(co)
        {  case  '+' : g = f1; break;
           case   '-' : g = f2; break;
        }
      document.writeln(g(1, 4)); }
// ]]>
</script>
</head>
<body>
<input type="button" onclick="i='+'" value="+" />
<input type="button" onclick="i='-'"  value="-" />
<input type="button" onclick="kalk(i)" value="Kalkulator" />
</body>
</html>
```



1.20. Pętle for, while, do while, for in

Instrukcja pętli for

Składnia

```
for ([instrukcja-początkowe]; [warunek]; [instrukcja-inkrementacji])
   { instrukcje;  }
```

Przykład

```
for (var i = 0; i < 5; i++)
        { document.write("i= "+ i +" "); }
```

Instrukcja pętli while

Składnia

```
while (warunek)          //pętla działa, dopóki warunek jest równy true
        { instrukcje;  }
```

Instrukcja pętli do...while

Składnia

```
      do {    instrukcje  }
      while (warunek);     //pętla działa, dopóki warunek jest równy true
```

Przykład

```
do
 { i+=1;
   document.write(i);
 } while (i<5);
```

Instrukcja pętli for..in

Pętla ta przechodzi przez wszystkie pola danego obiektu (np. elementy tablicy)

Składnia

```
for (zmienna in obiekt)
    { instrukcje; }
```

Przykład

```
function pokaz_obiekt(obiekt, NazwaObiektu)
 {
       var rezultat = "";
       for (var i in obiekt)
        {
            rezultat +=  NazwaObiektu+ "." + i + " = " + obiekt[i] + "\n";
        }
          document.write(rezultat);
}
```

Przykład 3: Zastosowanie instrukcji **for .. in**

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
// <![CDATA[
  function pokaz_obiekt(obiekt, NazwaObiektu)
    {
     var rezultat = "";
     for (var i in obiekt)
      rezultat +=  NazwaObiektu+ "." + i + " = " + obiekt[i] + "\n";
     document.write(rezultat);
   }
   a=new Array(1,2,3);
   b="Date";
// ]]>
</script>
</head>
<body>
    <input type="button" onclick="pokaz_obiekt(a, b)" value="for..in" />
</body>
</html>
```

1.21. Instrukcja label

Blok instrukcji, które mogą być wykonane podczas instrukcji break lub continue

Składnia

label:
instrukcja

Przykład

```
Miejsce1:
   document.writeln(("To jest instrukcja wskazana przez instrukcję label");
```

1.22. Instrukcja break

Jest używana do przerwania pętli, instrukcji **switch** lub etykiety.

Składnia

```
break [etykieta]
```

Przykład

```
function testBreak()
{   var i = 0;
    while (i < 6)
      { if (i == 3)
           break;
        i++; }
}
```

1.23. Instrukcja continue

Wznawia dowolną instrukcję pętli oraz etykiety

Składnia

continue [*label*]

Przykład

```
1)  i = 0;
    while (i <= 9)
        {  i++;
           if ((i % 2) == 0)
              continue;
            i += 2;  //po zakończeniu pętli i jest równe  11=1;3;4;5;7;8;9;11
    }
```

Przykład

```
 2) i = 0;
    poczatek:
    i+=1;
    while (i <= 9)
       {  i++;
          if ((i % 5) == 0)
              continue poczatek;
          i +=2;    //po zakończeniu pętli i jest równe  11=1;2;4;5;6;7;9;10;11
       }
```

1.24. Zdarzenie – obiekt Event (http://www.w3schools.com/jsref/dom_obj_event.asp)

Attribute	The event occurs when...	IE	F	O	W3C
onblur	An element loses focus	3	1	9	Yes
onchange	The content of a field changes	3	1	9	Yes
onclick	Mouse clicks an object	3	1	9	Yes
ondblclick	Mouse double-clicks an object	4	1	9	Yes
onerror	An error occurs when loading a document or an image	4	1	9	Yes
onfocus	An element gets focus	3	1	9	Yes
onkeydown	A keyboard key is pressed	3	1	No	Yes
onkeypress	A keyboard key is pressed or held down	3	1	9	Yes
onkeyup	A keyboard key is released	3	1	9	Yes
onload	A page or image is finished loading	3	1	9	Yes
onmousedown	A mouse button is pressed	4	1	9	Yes
onmousemove	The mouse is moved	3	1	9	Yes
onmouseout	The mouse is moved off an element	4	1	9	Yes
onmouseover	The mouse is moved over an element	3	1	9	Yes
onmouseup	A mouse button is released	4	1	9	Yes
onresize	A window or frame is resized	4	1	9	Yes
onselect	Text is selected	3	1	9	Yes
onunload	The user exits the page	3	1	9	Yes

Mouse / Keyboard

Property	Description	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	6	1	9	Yes
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	6	1	9	Yes
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

Pozostałe atrybuty obiektu Event

Property	Description	IE	F	O	W3C
bubbles	Returns a Boolean value that indicates whether or not an event is a bubbling event	No	1	9	Yes
cancelable	Returns a Boolean value that indicates whether or not an event can have its default action prevented	No	1	9	Yes
currentTarget	Returns the element whose event listeners triggered the event	No	1	9	Yes
eventPhase	Returns which phase of the event flow is currently being evaluated				Yes
target	Returns the element that triggered the event	No	1	9	Yes
timeStamp	Returns the time stamp, in milliseconds, from the epoch (system start or event trigger)	No	1	9	Yes
type	Returns the name of the event	6	1	9	Yes

1.25. Obsługa wyjątków: try… catch

Instrukcja try..catch
Składnia

```
try {
      instrukcje
   }
   [catch (wyjatek_var if wyrazenie)
     {instrukcje}] . . .
   [catch (wyjatek_var)
     {instrukcje}]
   [finally
      {instrukcje}]
```

Przykład

```
   try
    {   throw "moj_Wyjatek";  }              //  generowanie wyjatku
   catch (e)                                 // obsługa wyjątku
    { logMyErrors(e);  }   // przekazanie obiektu wyjątku do obsługi błędu
```

Przykład

```
try
 {  jakasfunkcja();   }                  //  może wygenerować dwa wyjątki
catch (e if e instanceof Type)
{/*  instrukcje obsługujące błędy typu wyjątku*/ }
catch (e if e instanceof  Format)
   { /* instrukcje obsługujące błędy  Format wyjątku*/ }
 catch (e)
 { /*  instrukcje obsługujące dowolny typ wyjątku*/
    logMyErrors(e);  }        //  przekazanie obiektu wyjatku do obsługi błędu
 finally
{ /* instrukcje zawsze wykonywane*/
}
```

Instrukcje w bloku **finally** zawsze są wykonywane, również wtedy, gdy wystąpi wyjątek.

1.26. Instrukcja throw

Definiowanie wyjątków użytkownika

Składnia

throw *wyrażenie;*

Przykład

```
throw "Blad1";  //  generowanie wyjątku z wartością łańcuchową
throw 42;       // generowanie wyjątku z wartością 42
throw true;     // generowanie wyjątku z wartością true
```

1.27. Instrukcja return

Specyfikuje wartość zwracaną przez funkcję.

Składnia

return *expression;*

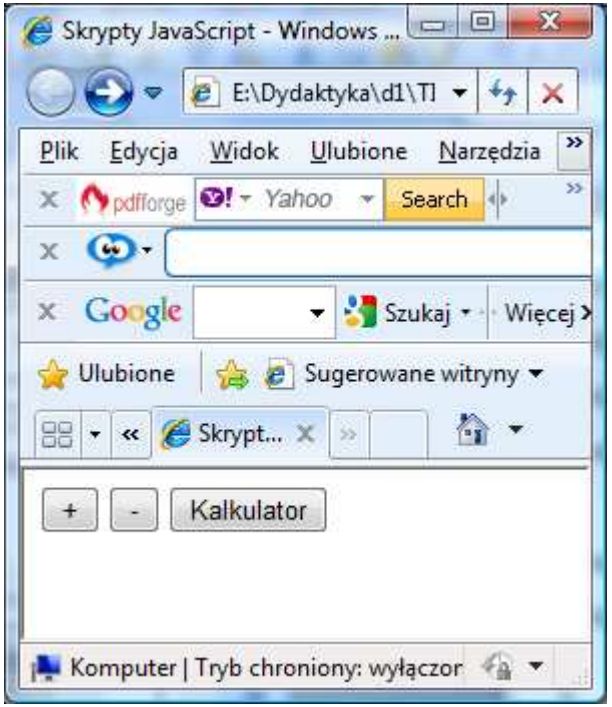
Przykład

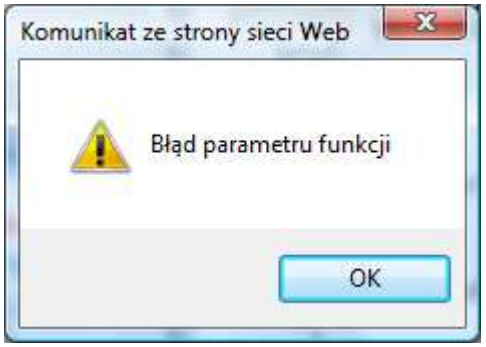
```
function sqr(x)
   { return x*x; }
```

Przykład 4: Zastosowanie obsługi wyjątków

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
// <![CDATA[
    function f1(a,b)       { return a+b;}
    function f2(a,b)       { return a-b; }
    function kalk(co) {
      try {
          switch(co)
           {  case  '+' : g = f1; break;
              case   '-' : g = f2; break;
           }
            document.writeln(g(1, 4));
       }
       catch(err)
           {    alert("Błąd parametru funkcji");   }
     }
// ]]>
</script>
</head>
```

```
<body>
<input type="button" onclick="i='+'" value="+" />
<input type="button" onclick="i='-'"  value="-" />
<input type="button" onclick="kalk()" value="Kalkulator" />
                    <!– brak parametru w wywoływanej funkcji kalk()-->
</body>
</html>
```





Przykład 5: Generowanie wyjątku z generowaniem obiektu.

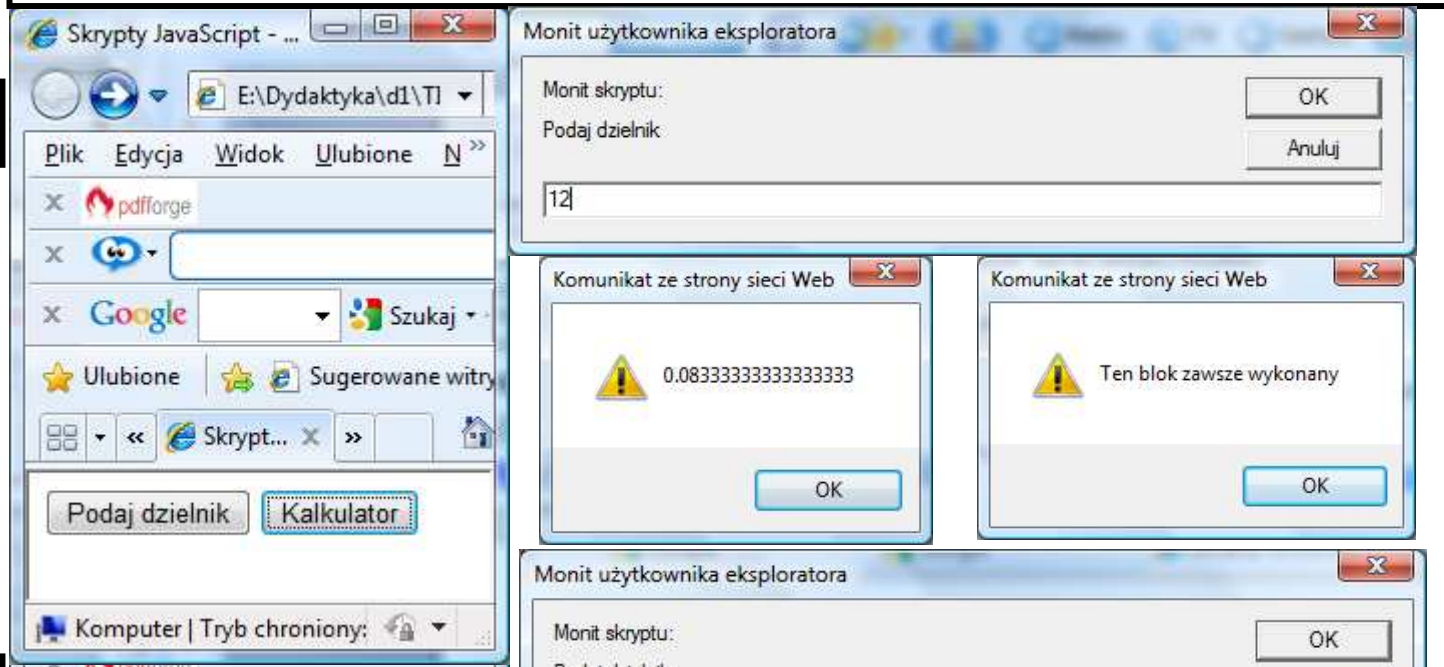
Należy zdefiniować obiekt, zanim zostanie przekazany do generowanego wyjątku. W bloku catch można odwoływać się do właściwości przekazanego obiektu przez wyjątek.
W przykładzie generowany jest obiekt moj_Wyjatek typu Wyjatek_uzytkownika i przekazany w instrukcji generowania wyjątku.

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

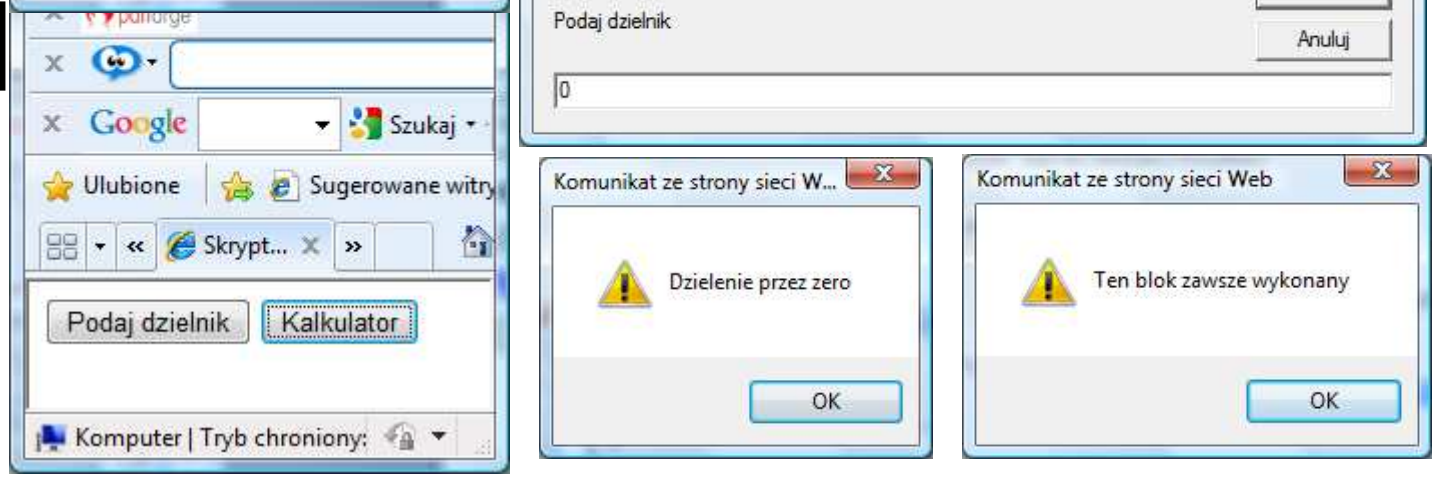
```
<script type="text/javascript">
// <![CDATA[
  function Wyjatek_uzytkownika(wiadomosc)
   {  this.message=wiadomosc; }
  function Podziel (dzielnik)
  {  if (dzielnik != 0)
        { alert(1/dzielnik); }            //1
    else                                  //2
      { moj_Wyjatek = new Wyjatek_uzytkownika ("Dzielenie przez zero");
       throw  moj_Wyjatek; }
  }
 function Dzielenie(liczba)
   {   try
        { odwrotnosc=Podziel(liczba);  }
      catch (e)
       {  alert(e.message); }  // 2 przekazanie obiektu wyjątku do obsługi błędu
      finally
       {  alert("Ten blok zawsze wykonany"); }
   }
    var a="Podaj dzielnik";
    var b="";
// ]]>
</script>
</head>
```

```
<body>
<input type="button" onclick="i=prompt(a, b)" value="Podaj dzielnik" />
<input type="button" onclick="Dzielenie(i)" value="Kalkulator" />
</body>
</html>
```

1



2



1.27. Znaki specjalne w łańcuchach tekstowych

Kod	**Wyjście**
\'	apostrof
\"	cudzysłów
\\	ukośnik lewy
\n	nowa linia
\r	Powrót karetki
\t	Tabulacja
\b	Backspace
\f	wysunięcie strony (w drukarce)

2. Obiekty standardowe

- Podstawowym typem obiektowym jest typ Object – wszystkie obiekty dziedziczą po typie Object.
- Array
  Boolean
  Date
  Function
  java
  JavaArray
  JavaClass
  JavaObject
  JavaPackage
  Math
  netscape
  Number
  Object
  Packages
  RegExp
  String
  sun

2.1. Definiowanie obiektów

2.1.1. Obiekty JavaScriptu są tablicami asocjacyjnymi.

Dostęp do pól obiektów jest możliwy przy użyciu równoważnych notacji:

obiekt.pole
obiekt["pole"]
obiekt[nr].

Obie notacje z nawiasami kwadratowymi zwyczajowo stosuje się jednak przy korzystaniu z tablic powstałych jako obiekt Array.

2.1.2. Definiowanie typu obiektu

Aby zdefiniować własny typ obiektu, należy utworzyć funkcję konstruktora:

```
function Obiekt1(wlasciwosc_1)
  {  this.wlasciwosc_1 =  wlasciwosc_1;
     function metoda_1()
        { alert("Obiekt1::metoda_1()"); }
     this.metoda_1 = metoda_1;
  }
```

2.1.3. **Aby utworzyć obiekt typu Obiekt1**, należy skorzystać z operatora new:

```
var obiekt1 = new Obiekt1(2);
```

Nowe obiekty w JavaScript tworzone są na stercie. W przypadku, kiedy do danego obiektu nie istnieje już żadna referencja, mechanizm garbage collector usuwa dany obiekt z pamięci.

2.1.4. **Metody obiektu (funkcje)** są jego składowymi, dostęp do nich jest możliwy przy użyciu notacji z kropką i notacji z nawiasami kwadratowymi. Poniższe dwie linie kodu są równoważne:

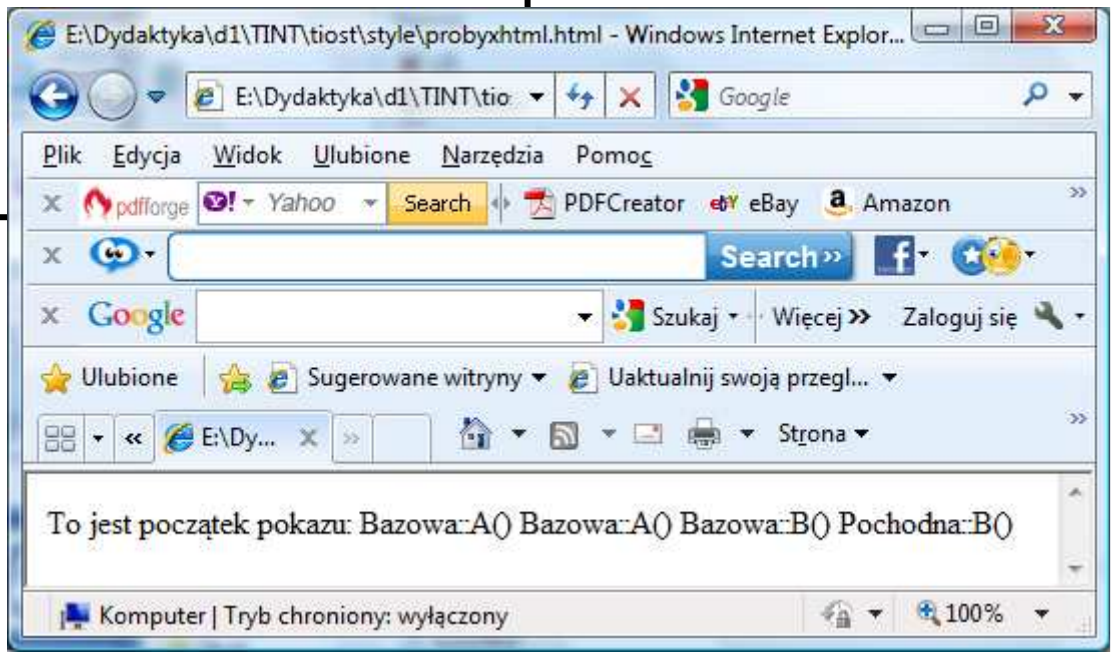
```
obiekt1.metoda_1();

obiekt1["metoda_1"]();
```

Przykład 6: Pierwszy sposób tworzenia dziedziczenia:

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Pokazy slajdów, skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="styl10.css" rel="stylesheet" type="text/css" />
 <script type="text/javascript">
  // <![CDATA[
  function Bazowa() {
    this.A = function()
      { document.write("Bazowa::A() ");    }
    this.B = function()
       { document.write("Bazowa::B() ");}
  }
 function Pochodna()
    {       // B przeciąża odpowiednią metodę B z klasy Bazowa:
       this.B = function ()
        { document.write("Pochodna::B() ");    }
    }
function Logo()
   {   document.write("To jest początek pokazu: ");  }
```

```
function proby()
{ Logo();
  Pochodna.prototype = new Bazowa();
  bazowy = new Bazowa();
  pochodny = new Pochodna();
  bazowy.A();                    // wyświetla: "Bazowa::A()"
  pochodny.A();                  // wyświetla: "Bazowa::A()"
  bazowy.B();                    // wyświetla: "Bazowa::B()"
  pochodny.B();                  // wyświetla: "Pochodna::B()"
}
  // ]]>
</script>
</head>
<body onload= "proby()">
 </body>
</html>
```

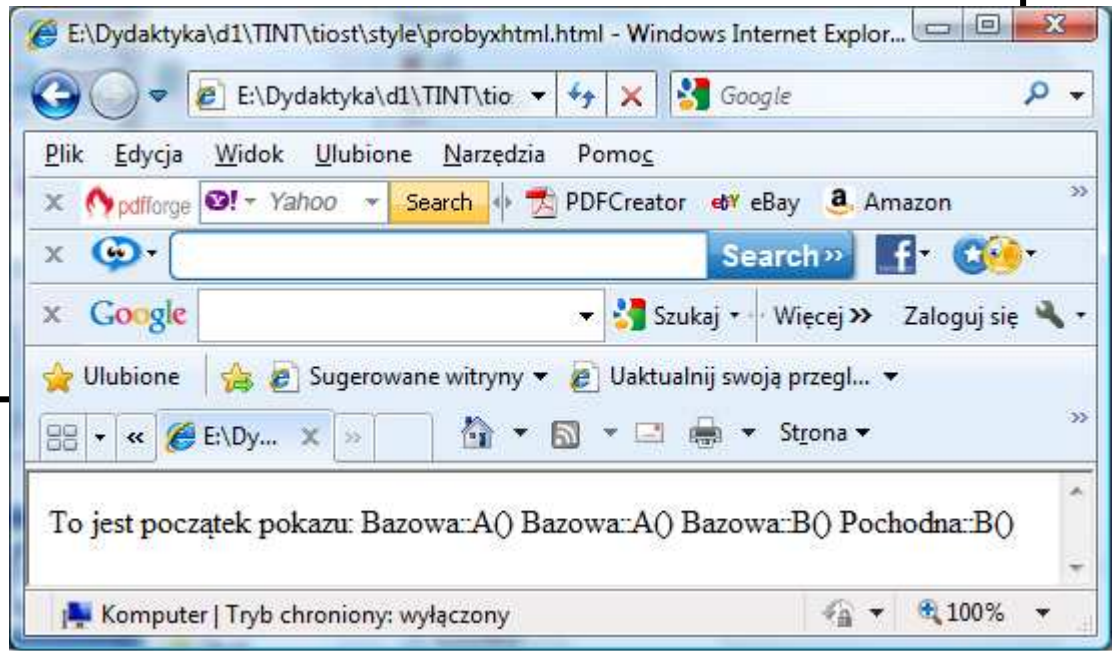


Przykład 7: Drugi sposób tworzenia dziedziczenia:

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Pokazy slajdów, skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="styl10.css" rel="stylesheet" type="text/css" />
<script type="text/javascript">
 // <![CDATA[
   Object.prototype.extending = function (supClass)
     {  tempObj = new supClass();
        for (property in tempObj)
          {  this[property] = tempObj[property];  }
      }
  function Bazowa()
  { this.A = function ()
     { document.write("Bazowa::A() ");  }
    this.B = function ()
     { document.write("Bazowa::B() "); }
  }
  function Pochodna()
   {  this.extending(Bazowa);
     this.B = function ()
      { document.write("Pochodna::B() "); }
   }
```

```
function Logo()
 {   document.write("To jest początek pokazu: ");}

function proby()
  { Logo();
    bazowy = new Bazowa();
    pochodny = new Pochodna();
    bazowy.A();              // wyświetla: "Bazowa::A()"
    pochodny.A();            // wyświetla: "Bazowa::A()"
    bazowy.B();              // wyświetla: "Bazowa::B()"
    pochodny.B();   }        // wyświetla: "Pochodna::B()"
 // ]]>
</script>
 </head>
  <body onload= "proby()">
 </body>
</html>
```

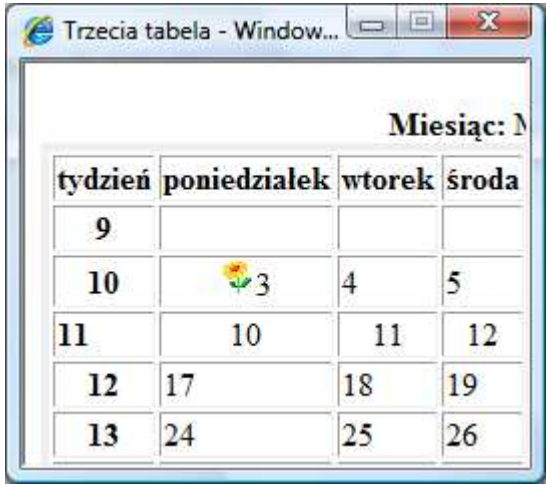


2.2. Ważne właściwości i funkcje niektórych typów obiektów

decodeURI
decodeURIComponent
encodeURI
encodeURIComponent
eval
Infinity
isFinite
isNaN
NaN
Number
parseFloat
parseInt
String
undefined

3. Przykłady - Okna

Przykład 8: Tworzenie i zamykanie nowego okna



```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Pokazy slajdów, skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="styl10.css" rel="stylesheet" type="text/css" />
</head>
<body>
<p>
    <input type="button" value="Otworz okno 2"
       onclick="msgWindow=
          window.open('','okno2','resizable=no,width=200,height=200')"/>
</p>
<p><a href="b2.html" target="okno2">Ładuj plik do okna 2</a></p>
<p><input type="button" value="Zamknij okno 2"
        onclick="msgWindow.close()"/>
</p>
</form>
</body>
</html>
```

Przykład 9: Liczba odwiedzeń okna, zastosowanie URL, zmiana rozmiarów i położenia głównego okna oraz okna potomnego

```
<!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" xml:lang="pl" lang="pl">
<head>
<title>Pokazy slajdów, skrypty JavaScript</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="styl10.css" rel="stylesheet" type="text/css" />
<script type="text/javascript">
  // <![CDATA[
  var p1=0;
  function okno()
  { msgWindow=window.open('b2.html','okno2',
     'resizable=no,width=300,height=300,toolbars=yes,location=yes, scrollbars=yes');
     msgWindow.moveTo(300,100);
     setTimeout('msgWindow.scroll(0,100)',1000);
     msgWindow.focus();
    ++p1; }
  function okno1()
   { msgWindow.location="b2.html";
     msgWindow.moveTo(300,100);
     msgWindow.focus();
     ++p1; }
    // ]]>
</script>
</head>
```

```
<body onload="self.resizeTo(350,250), self.moveTo(0,0)">

<p><input type="button" value="Otworz okno 2" onclick="okno()"/>
</p>
<p><a href="javascript:okno1()" >Ładuj plik do okna 2</a>
</p>
<p> <input type="button" value="Zamknij okno 2"
    onclick="msgWindow.close(), alert('liczba odwiedzeń: '+p1)"/>
</p>
</form>
</body>
</html>
```

Okienko to pokazuje się po naciśnięciu przycisku „Zamknij okno 2”, jeśli naciśnięto raz przycisk „Otwórz okno 2” oraz link „Ładuj plik do okna 2”

Okno to pokazuje się po naciśnięciu przycisku „Otwórz okno 2” – zawartość okna jest przesuwana w górę po upływie 1 s

Okno to pokazuje się po naciśnięciu linku „Ładuj plik do okna 2”