przydatne strony

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

http://kursjs.pl/kurs/obiekty.html

http://mrzepinski.pl/70-480-tworzenie-i-wdrazanie-obiektow-i-metod.h

http://burczu-programator.pl/blog/obsluga-wyjatkow-w-jezyku-javascript

http://mrzepinski.pl/70-480-wdrazanie-narzedzi-obslugi-wyjatkow.html

JavaScript jest obiektowym skryptowym językiem programowania.

W języku JavaScript wszystkie zmienne, poza zmiennymi oznaczającymi typy podstawowe, reprezentują obiekty.

Wbudowane obiekty to między innymi:

Date, Math, Location, window, wyrażenia regularne, tablice. Obiektem jest również funkcja.

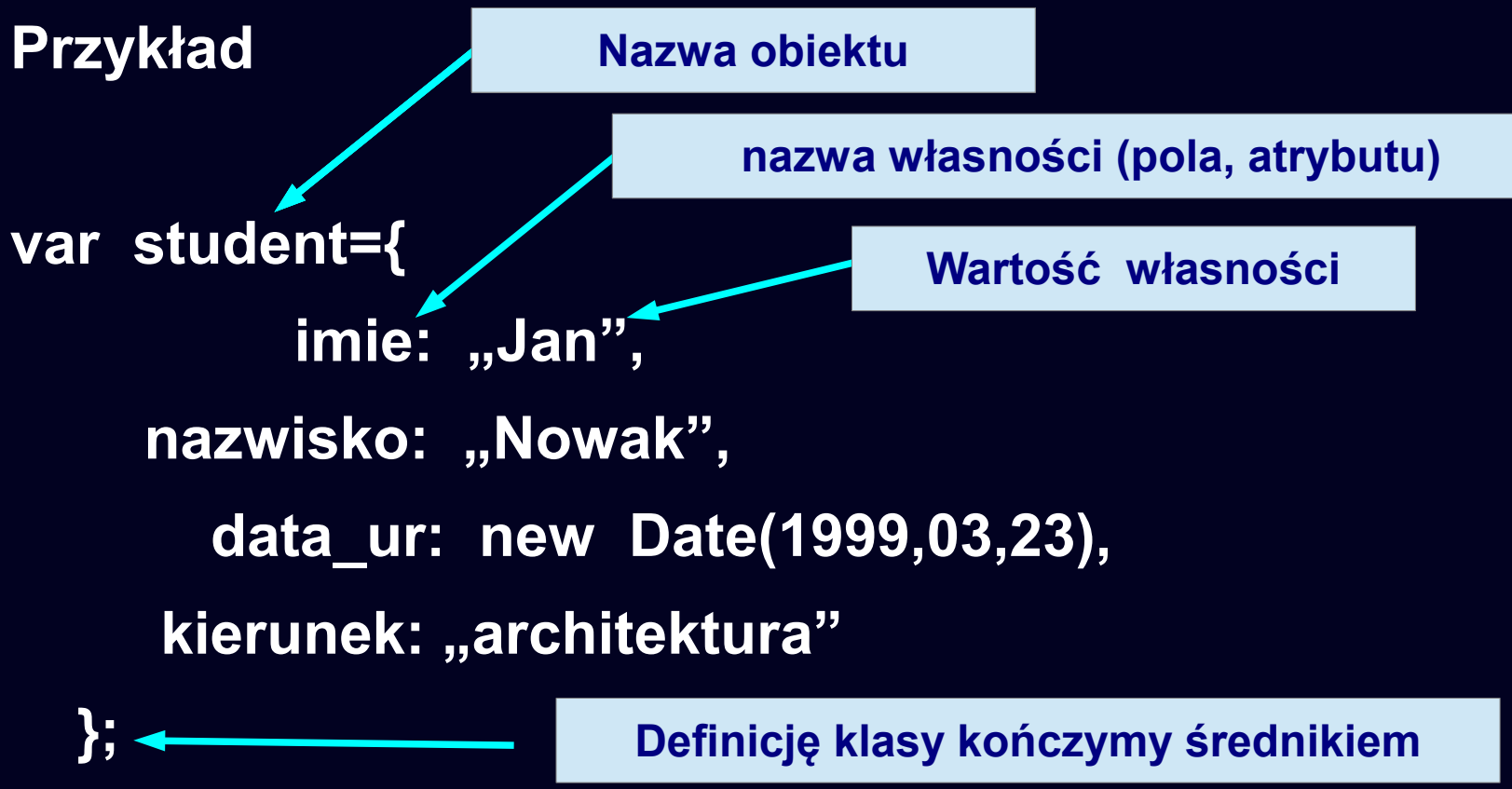
W JavaScript obiekt to zmienna zawierająca inne zmienne lub funkcje.

W JavaScript użytkownik może definiować i tworzyć własne obiekty na kilka sposobów.

Definicja obiektu- I sposób- za pomocą literału

– definiujemy i tworzymy obiekt w jednej instrukcji, podając jego nazwę, własności i wartości.

Przykład



Przykład użycia

```
<body >
<div></div>
<script >
var  student = {
            imie:"Jan",
          nazwisko:"Nowak",
           data_ur: new Date(1999,03,23),
           kierunek:"architektura"
    };
//wypiszemy dane obiektu w znaczniku div
var rok=student.data_ur.getYear()+1900;
var miesiac=student.data_ur.getMonth()+1;
  var str="dane studenta:<br>"+student.imie+"<br> "+student.nazwisko+"<br> "+rok+":"+
  miesiac+":"+student.data_ur.getDate() +" <br>"+student.kierunek;
document.getElementsByTagName("div")[0].innerHTML=str;

    </script>
</body>
```

dane studenta:
Jan
Nowak
1999:4:23
architektura

Dodanie metody

```
<body >
<div></div>
<script >
var  student = {
            imie:"Jan",
          nazwisko:"Nowak",
           data_ur: new Date(1999,03,23),
           kierunek:"architektura",
    wypiszDane: function (){
                    var rok=this.data_ur.getYear()+1900;
                    var miesiac=this.data_ur.getMonth()+1;
                   var str="dane studenta:<br>"+this.imie+
                   "<br> "+this.nazwisko+"<br> "+rok+":"+
                   miesiac+":"+this.data_ur.getDate() +" <br>"+
                   this.kierunek;
                   return str;
                }
    };
//wypiszemy dane obiektu w znaczniku div
document.getElementsByTagName("div")[0].innerHTML=
student.wypiszDane();
    </script>
</body>
```

dane studenta:
Jan
Nowak
1999:4:23
architektura

Definicja obiektu- II sposób użycie operatora new

```
var  student=new Object();
        student.imie="Jan";
      student.nazwisko="Nowak";
       student.data_ur= new Date(1999,03,23);
       student.kierunek="architektura";
```

Pierwsza i druga metoda dają taki sam efekt. Przy czym, ze względu na prostotę, przejrzystość i szybkość działania, zaleca się stosowanie I metody.

Definicja obiektu- II sposób- użycie operatora new -przykład

```
<body >
<div></div>
<script >
var  student=new Object();
            student.imie="Jan";
          student.nazwisko="Nowak";
           student.data_ur= new Date(1999,03,23);
           student.kierunek="architektura";
//wypiszemy dane obiektu w znaczniku div
var rok=student.data_ur.getYear()+1900;
var miesiac=student.data_ur.getMonth()+1;
  var str="dane studenta:<br>"+student.imie+"<br> "+student.nazwisko+"<br> "+rok+":"+ miesiac+":"
  +student.data_ur.getDate() +" <br>"+student.kierunek;
document.getElementsByTagName("div")[0].innerHTML=str;
    </script>
</body>
```

dane studenta:
Jan
Nowak
1999:4:23
architektura

Definicja obiektu- II sposób-
• użycie operatora new -przykład c.d -dodajemy metodę

```
<body >
<div></div>
<script >
var  student=new Object();
            student.imie="Jan";
            student.nazwisko="Nowak";
           student.data_ur= new Date(1999,03,23);
           student.kierunek="architektura";
        // definiujemy metodę
           student.wypiszDane= function(){
          var rok=this.data_ur.getYear()+1900;
          var miesiac=this.data_ur.getMonth()+1;
        var str="dane studenta:<br>"+this.imie+"<br> "+this.nazwisko+
        "<br> "+rok+":"+ miesiac+":"+this.data_ur.getDate() +" <br>"+
        this.kierunek;
        return str;
        }
//wypiszemy dane obiektu w znaczniku div, wywołujemy metodę wypiszDane
document.getElementsByTagName("div")[0].innerHTML=student.wypiszDane();

    </script>
</body>
```

dane studenta:
Jan
Nowak
1999:4:23
architektura

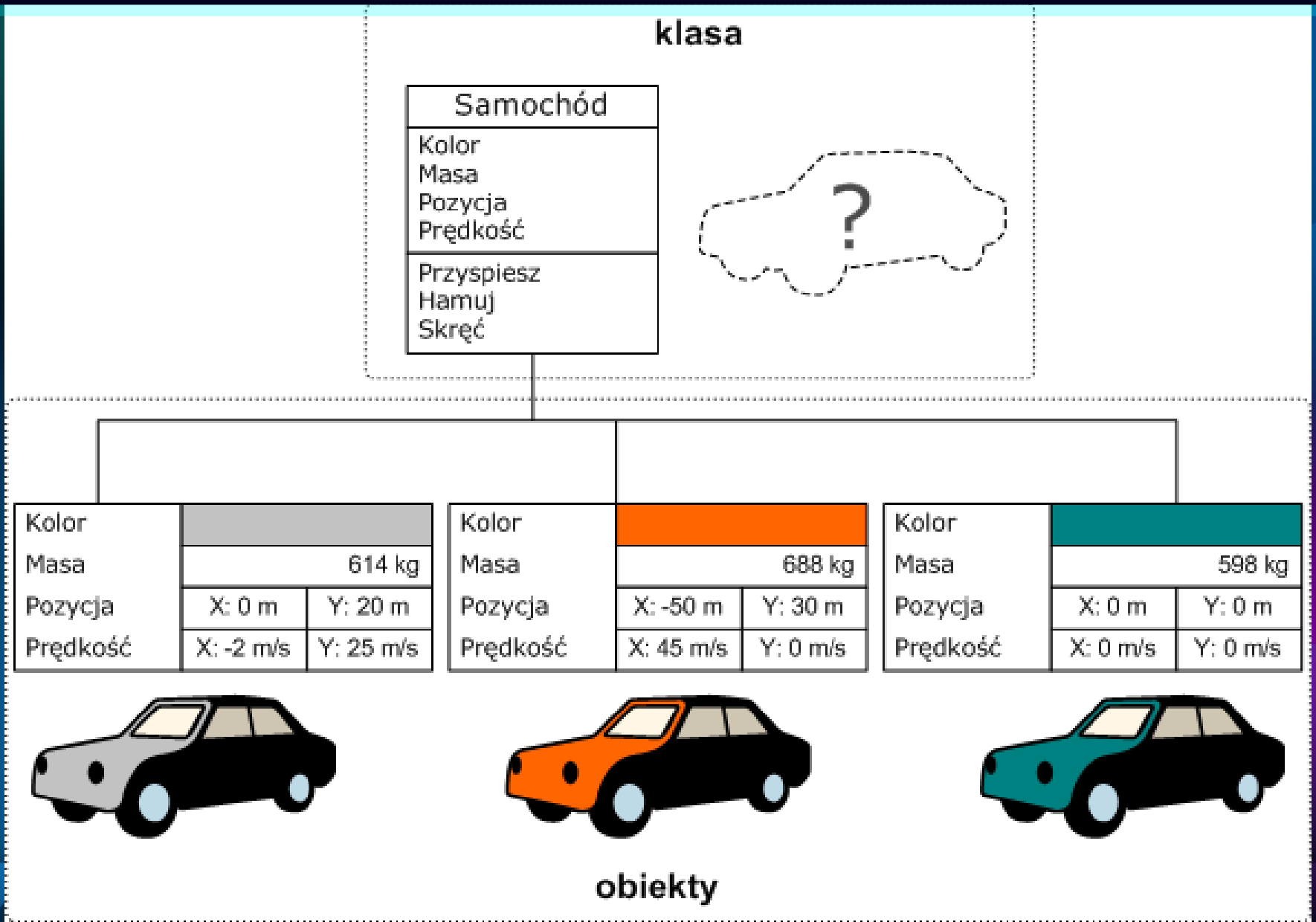
Uwaga

Pierwsze dwie metody tworzą pojedynczy obiekt.

Bardziej praktyczne jest stworzenie nowego typu danych, a następnie stworzenie wielu obiektów należących do tego typu.

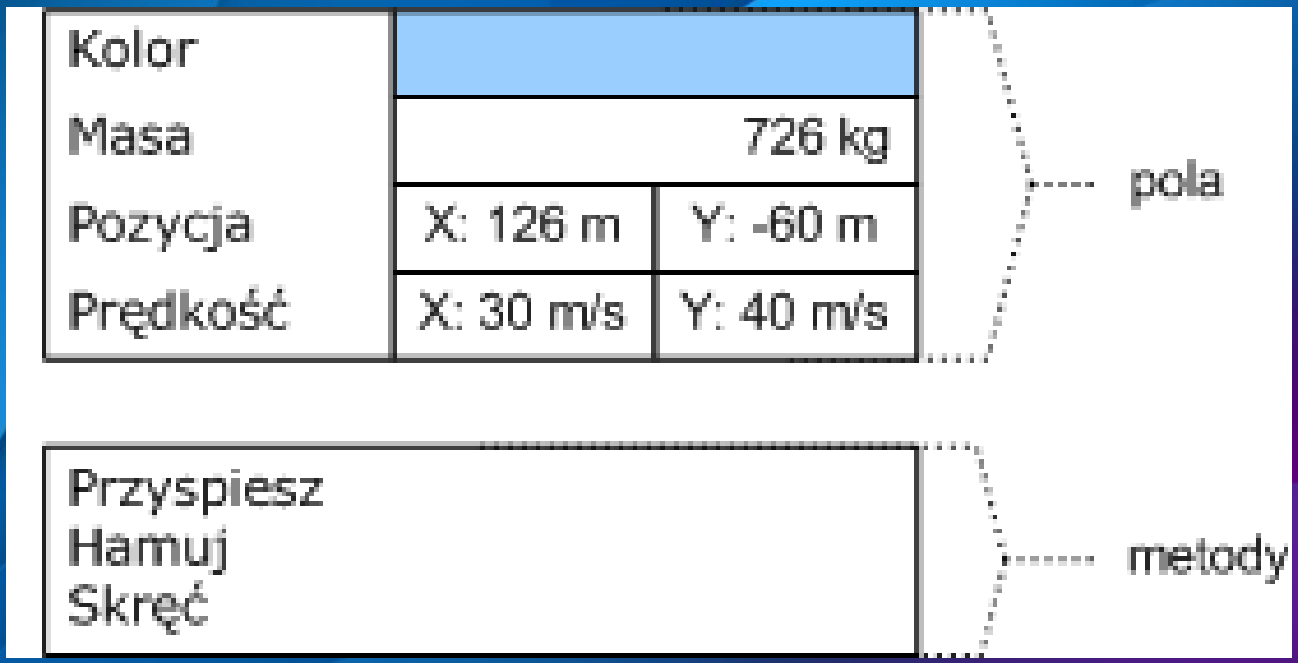
Oprócz opisanej p notacji literałowej, język JavaScript pozwala również na tworzenie obiektów za pomocą funkcji - konstruktorów, które przypominają konstruktory z innych obiektowych języków programowania.

Klasa jest czymś w rodzaju wzorca - matrycy, wedle którego "produkowane" są kolejne obiekty (instancje) w programie. Mogą one różnić się od siebie, ale tylko co do wartości poszczególnych pól; wszystkie będą jednak należeć do tej samej klasy i będą mogły wykonywać na sobie te same metody.



Załóżmy, że chcemy mieć w programie obiekt jadącego samochodu

Ustalamy więc dla niego pola, które będą go określały, oraz metody, które będzie mógł wykonywać.



Definicja obiektu-III sposób- użycie funkcji-konstruktora

– za pomocą konstruktora tworzymy klasę

Przykład

```
//konstruktor klasy student
function student(imie, nazwisko,data_ur,kierunek){
  this.imie=imie;
  this.nazwisko=nazwisko;
  this.data_ur= data_ur;
  this.kierunek=kierunek;
}
```

Obiekty (instancje) klasy tworzymy za pomocą operatora new

```
//dwa obiekty klasy student
var s1= new student("Dorota", "Kowalska", new Date(2000, 11,2), "informatyka");
var s2= new student("Paweł", "Kowal", new Date(1998, 4,12), "geologia");
```

Użycie konstruktora- • zastosowanie w programie

```
<h1>pierwszy student</h1>
<div></div>
<h1>pierwszy drugi</h1>
<div></div>
<script >
//konstruktor klasy student
function student(imie, nazwisko,data_ur,kierunek){
  this.imie=imie;
  this.nazwisko=nazwisko;
  this.data_ur= data_ur;
  this.kierunek=kierunek;
}
//dwa obiekty klasy student
var s1= new student("Dorota", "Kowalska", new Date(2000, 11,2), "informatyka");
var s2= new student("Paweł", "Kowal", new Date(1998, 4,12), "geologia");
//wypisujemy dane studentów
var rok=s1.data_ur.getYear()+1900;
var miesiac=s1.data_ur.getMonth()+1;
  var str="dane studenta:<br>"+s1.imie+"<br> "+s1.nazwisko+"<br> "+rok+":"+ miesiac+":"+
  s1.data_ur.getDate() +" <br>"+s1.kierunek;
document.getElementsByTagName("div")[0].innerHTML=str;
//i dla drugiego studenta
 rok=s2.data_ur.getYear()+1900;
 miesiac=s2.data_ur.getMonth()+1;
   str="dane studenta:<br>"+s2.imie+"<br> "+s2.nazwisko+"<br> "+rok+":"+ miesiac+":"+
   s2.data_ur.getDate() +" <br>"+s2.kierunek;
document.getElementsByTagName("div")[1].innerHTML=str;
    </script>
</body>
```

pierwszy student

dane studenta:
Dorota
Kowalska
2000:12:2
informatyka

pierwszy drugi

dane studenta:
Paweł
Kowal
1998:5:12
geologia

Konstruktor i inne metody -zastosowanie w programie

```
<body >
<h1>pierwszy student</h1>
<div></div>
<h1> drugi  student</h1>
<div></div>
<script >
//konstruktor klasy student
function student(imie, nazwisko,data_ur,kierunek){
  this.imie=imie;
  this.nazwisko=nazwisko;
  this.data_ur= data_ur;
  this.kierunek=kierunek;
  //definiujemy metodę klasy
  this.wypiszDane= function (){
var rok=this.data_ur.getYear()+1900;
var miesiac=this.data_ur.getMonth()+1;
  var str="dane studenta:<br>"+this.imie+"<br> "+this.nazwisko+"<br> "+rok+":"+ miesiac+
  ":"+this.data_ur.getDate() +" <br>"+this.kierunek;
  return str;
  }
}
//dwa obiekty klasy student
var s1= new student("Dorota", "Kowalska", new Date(2000, 11,2), "informatyka");
var s2= new student("Paweł", "Kowal", new Date(1998, 4,12), "geologia");
//wypisujemy dane studentów - wywołujemy metodę klasy
document.getElementsByTagName("div")[0].innerHTML=s1.wypiszDane();
document.getElementsByTagName("div")[1].innerHTML=s2.wypiszDane();
</script>
</body>
```

pierwszy student

dane studenta:
Dorota
Kowalska
2000:12:2
informatyka

drugi student

dane studenta:
Paweł
Kowal
1998:5:12
geologia

Obsługa wyjątków w JS

Obiekt typu Error

Obsługa błędów w JavaScript daje nam różne narzędzia. Jednym z nich jest tworzenie i rzucanie (throw) obiektów stworzonych do tego celu (Error). Obiekt taki przechowuje bardziej szczegółowe informacje dotyczące błędów.

```
<body >
<script >
function dziel(a, b) {
  if (arguments.length != 2) {
    throw new Error("funkcja dziel() wymaga dwóch argumentów.");
  }
  else if (typeof a != "number" || typeof b!= "number") {
    throw new Error("funkcja dziel()wymaga dwóch argumentów liczbowych");
  }
  else if (b == 0) {
    throw new Error("Nie dziel przez zero");
    }
  return a / b;
}
 alert(dziel(2,3));
//alert(dziel("a"));
//alert(dziel(2, 0));
//alert(dziel("a", 0));
</script>
</body>
```

Obsługa wyjątków w JS

Konstrukcja try… catch… finally

Konstrukcja tego typu występuje w wielu językach obiektowych, a służy do obsługi wyjątków.

W bloku **try** umieszczamy kod, którego działanie może zakończyć się w nieprzewidywalny sposób.

W bloku **catch** umieścimy kod obsługi wyjątku, który tu przechwycimy.

Natomiast w opcjonalnym bloku **finally** – kod, który wykona się niezależnie – zawsze po przetworzeniu bloku try i ewentualnie catch.

Obsługa wyjątków w JS

Konstrukcja try… catch… finally

Konstrukcja tego typu występuje w wielu językach obiektowych, a służy do obsługi wyjątków.

W bloku **try** umieszczamy kod, którego działanie może zakończyć się w nieprzewidywalny sposób.

W bloku **catch** umieścimy kod obsługi wyjątku, który tu przechwycimy.

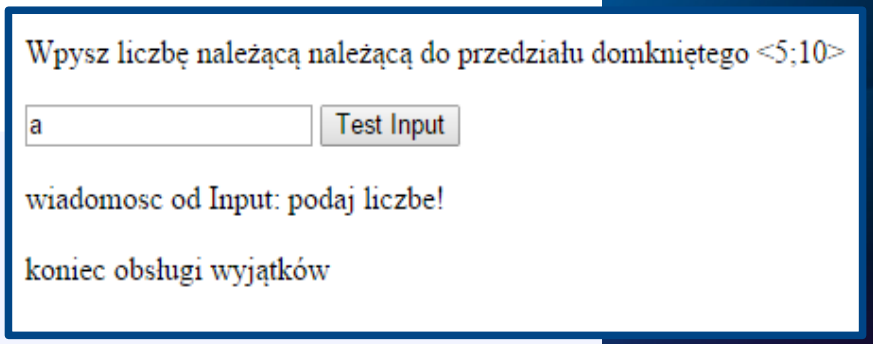
Natomiast w opcjonalnym bloku **finally** – kod, który wykona się niezależnie – zawsze po przetworzeniu bloku try i ewentualnie catch.

Konstrukcja try… catch… finally- połączenie z throw-przykład

```
<body >
<p>Wpysz liczbę należącą należącą do przedziału domkniętego <5;10></p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="message"></p>
<p ></p>
<script>

function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x =="") throw ("input jest pusty");
        if(isNaN(x)) throw ("podaj liczbe!");
        if(x > 10) throw ("liczba za duża");
        if(x < 5) throw( "liczba za mała");
    }
    catch(rodzajBledu) {
        message.innerHTML = "wiadomosc od Input: " + rodzajBledu;
    }
    finally{
    document.getElementsByTagName("p")[2].innerHTML="koniec obsługi wyjątków";
    }
}
</script>
</body>
```



Zdarzenie onerror

Zdarzenie to zachodzi m.in., gdy pojawia się błąd wykonania kodu JavaScript

Obsługa tego zdarzenia polega na wywołaniu

onerror = funkcja_obsługi_błędu

Zdarzenie onerror zwraca trzy wartości

msg – komunikat o błędzie

url – adres url strony, która spowodowała błąd

line – linia w której wystąpił błąd

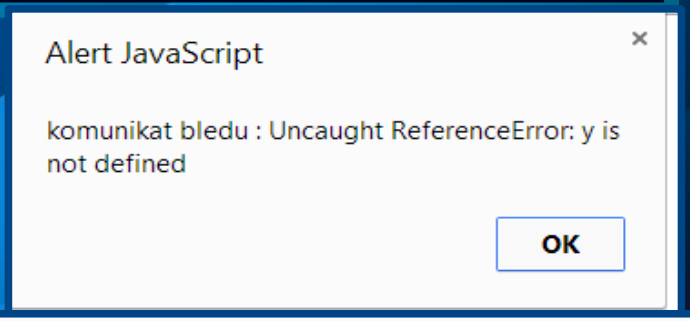
Właściwość ‚onerror' posiada wiele elementów DOM, dzięki czemu możemy dodawać do nich funkcję obsługującą zdarzenie błedu.

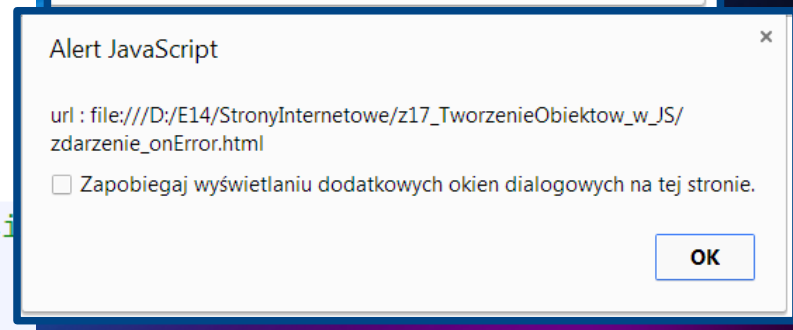
Zdarzenie onerror- przykład

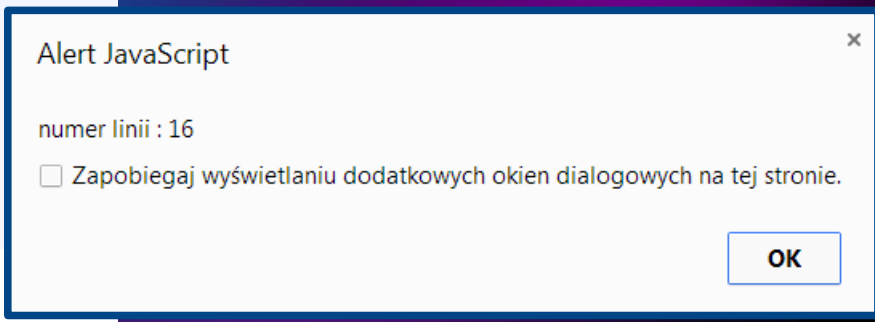
Za każdym razem gdy na stronie internetowej wystąpi błąd, wywoływane jest zdarzenie ‚error' obiektu ‚window'. Obiekt ten posiada właściwość ‚onerror', do której można przypisać funkcję obsługującą zdarzenie błędu.

```
1  <!DOCTYPE html>
2  <html lang="pl">
3    <head>
4      <title>zdarzenie onError</title>
5      <meta charset="UTF-8" />
6  </head>
7  <body >
8
9  <script>
10 // definicja funkcji anonimowej przypisanej do właściwości
   ‚onerror' obiektu ‚window'
11 window.onerror = function (msg, url, line) {
12     alert("komunikat bledu : " + msg );
13     alert("url : " + url );
14     alert("numer linii : " + line );
15 }
16 var x = y + 5; // ta linia powoduje blad
17 </script>
18 </body>
19 </html>
20
```

Funkcja anonimowa







OBIEKT STRING

Obiekt reprezentujący fragmenty tekstu

Aby utworzyć obiekt typu String wystarczy zadeklarować zmienną tekstową:

var nazwa_zmiennej = "łańcuch_znaków"

nazwa_zmiennej = "łańcuch_znaków"

nazwa = new String("łańcuch_znaków")

Przykład:

var tekst1 = "Jakiś tekst."

tekst2 = "Inny tekst !"

tekst3 = new String("Utworzenie nowego obiektu.")

Korzystanie z właściwości obiektu

nazwa_zmiennej.właściwość

Korzystanie z metod

nazwa_zmiennej.nazwa_metody(lista_argumentów)

WYBRANE METODY OBIEKTU STRING

Zmiana stylu:

- **big() –powiększona czcionka**
- **small() – pomniejszona czcionka**
- **bold() – wytłuszczona czcionka**
- **italics() – kursywa**
- **blink() – tekst migający (nie działa w IE)**
- **fixed() – czcionka o stałej szerokości**
- **strike() – przekreślona czcionka**
- **fontcolor(kolor) – zmiana koloru czcionki**
- **(kolor = nazwa koloru|rgb(n,n,n)|#xxxxxx)**
- **fontsize(rozmiar_czcionki) – zmiana rozmiaru czcionki**

WYBRANE METODY OBIEKTU STRING

Zmiana stylu:

- **toLowerCase() – małe litery**
- **toUpperCase() – wielkie litery**
- **sub() – indeks dolny**
- **sup() – indeks górny**
- **link(url) – tekst będzie wyświetlony jako hiperłącze do dokumentu o adresie url**

WYBRANEMETODY OBIEKTU STRING

Wyodrębnianie podciągu znaków:

slice(pozycja_początkowa,pozycja_końcowa) – wycinanie fragmentu tekstu od pozycja_początkowa do pozycja_końcowa (obydwa argumenty numeryczne, wartość ujemna oznacza pozycję od końca, musi być spełniony warunek pozycja_początkowa < pozycja_końcowa, drugi argument opcjonalny – brak oznacza, że pozycja_końcowa = koniec łańcucha)

substr(pozycja_początkowa, długość) - wycina z łańcucha liczbę znaków długość począwszy od pozycja_początkowa (ujemna wartość pierwszego argumentu oznacza pozycję od końca, drugi

argument opcjonalny – brak= koniec łańcucha)

WYBRANEMETODY OBIEKTU STRING

subsrtring(pozycja_początkowa,pozycja_końcowa) – podobnie jak slice

split(separator, liczba) – podzelenie łańcucha znaków na tablicę ciągów, pierwszy argument jest znakiem, który rozgranicza ciągi w łańcuchu znaków (np. "." przy podziale na zdania, " " przy podziale na wyrazy, "" przy podziale na litery, itd.) , drugi opcjonalny argument służy do określenia liczby ciągów

OBIEKT STRING

Własność length

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>własnosc length</title>
    <meta charset="UTF-8" />
</head>
<body >
<script>
var a = "ABCDEFGHIJ";
//mozna tak var  a=new String("ABCDEFGHIJ");
var n = a.length;
//zmienna a  jest tablicą znaków. do każdego znaku zmiennej string
//mozna odwołać się przez index np. a[0] jest pierwszym znakiem  zmiennej a
//-w naszym przykładzie a[0] jest literą A
for( i=0; i<n;i++)
{document.write(a[i]+"<br>");
}
</script>
</body>
</html>
```

A
B
C
D
E
F
G
H
I
J