

01.09.2021

## Dział 1 Praca z oknami

### Zaliczenie zadań

Uwaga1:

Najpierw wykonaj Zadanie1 a potem zajmij się budową strony zaliczeniowej.

Uwaga2:

Do Twojego nazwiska 2cm użyj liniowego arkusza styli, wycentrum Twoje nazwisko.

Zaliczenie nastąpi z użyciem strony o następującej budowie:

<b>Twoje nazwisko wielkością czcionki 2 cm</b>					
Przycisk Przykład1a Działanie	Przycisk Przykład1a Listing	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....

Np.

<div>Kowalski</div> <div><div>Przykład 1a działanie</div><div>Przykład 1a listing</div><div>Zadanie 1a działanie</div></div>		
------------------------------------------------------------------------------------------------------------------------------	--	--

Uwaga3:

Okienka uruchamiane przyciskami powinny mieć przyciski Zamknij.

Uwaga4:

Do wyświetlania kodu użyj pola formularza z HTML **typu textarea** ze zdefiniowaną odpowiednio do listingu ilością kolumn i wierszy (tak aby widoczny był cały kod, bez powiększa okna textarea myszką.)

### **Zadanie 1. (Przykład 1a)**

Temat: Demonstracja pracy z oknami.

Wykonaj:

- 1)Utwórz folder Zadanie1
- 2) Dokonaj kopiowania plików:

Plik o nazwie przyklad1a.html

```
<html>
  <head>
    <title>Przykład nr tutaj wpisz odpowiedni nume </title>
    <script language="JavaScript">
      function WinOpen()
      {
window.open("obraz.html","okienko","toolbar=no,directories=no,menubar=no,height=280,width=160,top=200,left=200");
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" name="przycisk" value="Nowa Strona" onclick="WinOpen(' ')">
    </form>
  </body>
</html>
```

Plik o nazwie obraz.html

```
<html>
  <head>
    <script type="text/javascript">
      function okno_zamknij()
      {
        window.close()
      }
    </script>
  </head>
  <body>
    opis obrazka /* tekst -->opis obrazka<-- zastąp odpowiedziami/patrz poniżej Teoria */
    
    <input type="button" value="zamknij okno" onclick="okno_zamknij()"/>
  </body>
</html>
```

3)dograj animowanego gifa o nazwie obraz1.gif

4)

Zmień nazwy funkcji tak aby było tam nazwisko:

function WinOpen() →na WinOpen\_kowalski()

function okno\_zamknij() →okno\_zamknij\_kowalski()

5)Dopisz teorię zamiast „**opis obrazka**”

**Teoria:**

**a)Uwagi na temat JS**

-kiedy powstała

-dlaczego jest to język skryptowy

-gdzie jest wykonywany JS (klient lub serwer)

-dlaczego jest to bezpieczne narzędzie

-czy jest to język obiektowy , jeśli tak to dlaczego.

**b)Przepisz linie kody pod nimi wytłumaczenie:**

<input type="button" name="przycisk" value="Nowa Strona" onclick="WinOpen(' ')">

.....wytłumaczenie.....

window.open("obraz.html","okienko","toolbar=no,directories=no,menubar=no,height=280,width=160,top=200,left=200");

.....wytłumaczenie.....

toolbar=no                   →wytłumaczenie

directories=no,               →wytłumaczenie

menubar=no,               →wytłumaczenie

height=280,               →wytłumaczenie

width=160,               →wytłumaczenie

top=200,               →wytłumaczenie

left=200               →wytłumaczenie

window.close()

.....wytłumaczenie.....

## Zadanie 2.(Zadanie 1a)

Temat: Praca z oknami.

Uzupełnij przykład tak aby:

- wyświetlały się, co najmniej cztery pliki graficzne z opisami odpowiadającymi grafice. Grafiki otwierają się w nowych oknach, opis umieść pod grafiką w tym samym oknie. Całość ma tworzyć śmieszna przyzwoita historyjkę. Pliki pobierz z Internetu. Opisy wymyśl sam.
- na stronie głównej wykonaj przyciski do każdej grafiki z odpowiednim napisem na przycisku.
- okna z grafiki powinny być tak rozmieszczone aby po otwarciu wszystkich nie nachodziły na siebie.

- Funkcje muszą mieć nazwy z Twoim nazwiskiem

WinOpen\_kowalski\_1()

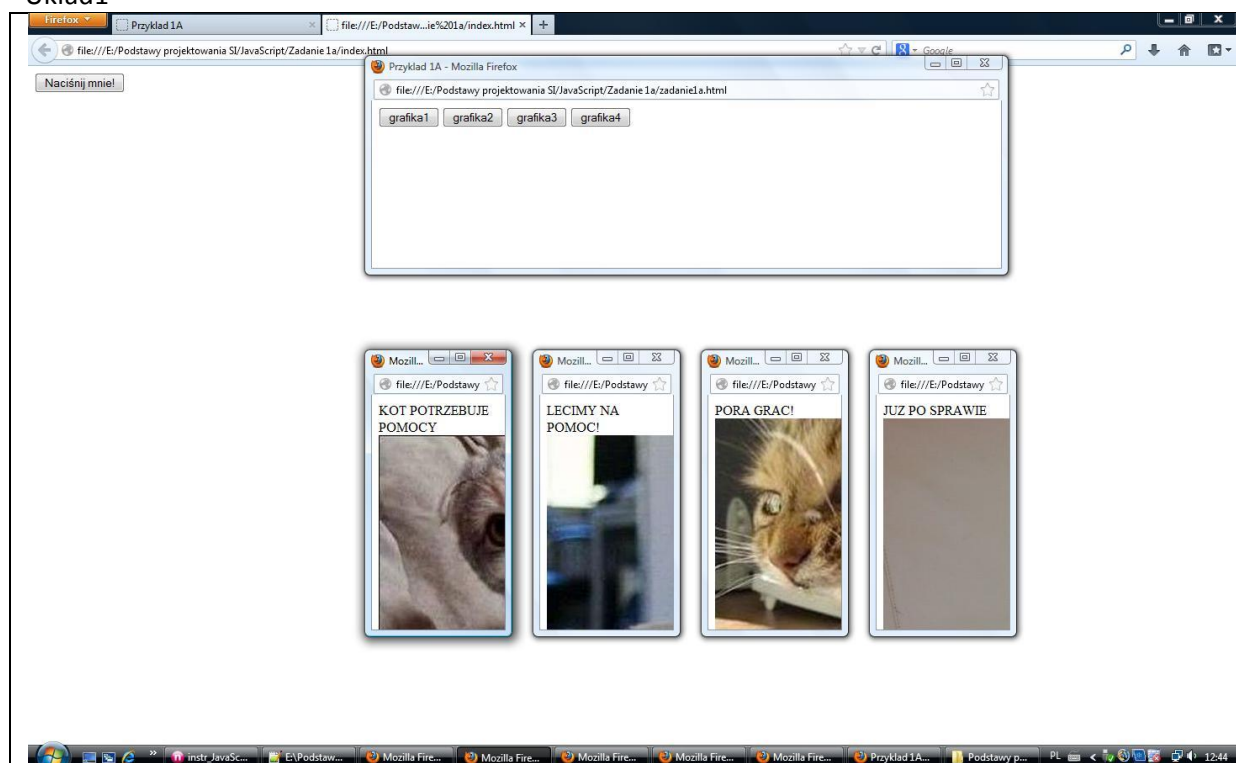
okno\_zamknij\_kowalski\_1()

- pamiętaj o tym, że okna muszą mieć różne nazwy
- podczas zaliczania zmniejsz okno przeglądarki po uruchomieniu index.html, tak aby otwierane okna z garfikami i napisami nie były zasłaniane przez okno nowych index.html.

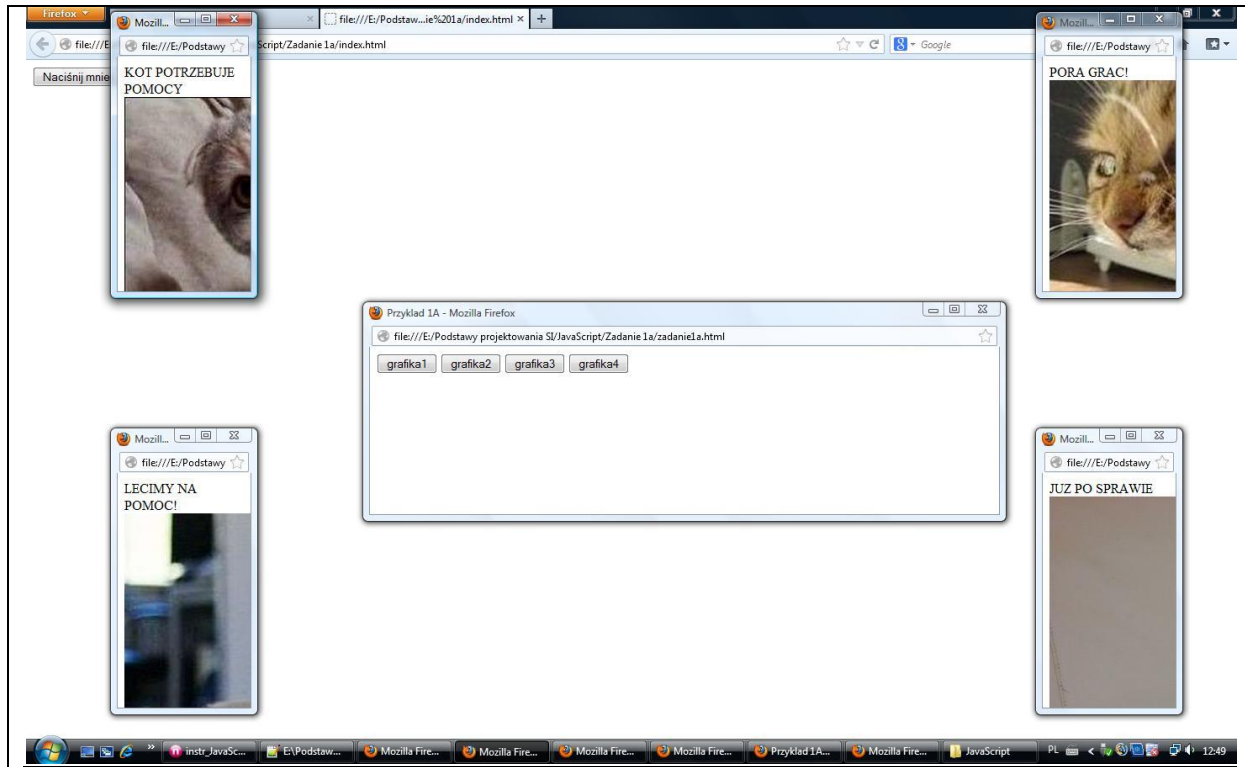
Wykonujesz układ jak układach zademonstrowanych poniżej.

Obliczenie	Który układ
(Numer_w_grupie) mod 4=1	Układ1
(Numer_w_grupie) mod 4=2	Układ2
(Numer_w_grupie) mod 4=0	Układ3
(Numer_w_grupie) mod 4=3	Układ4

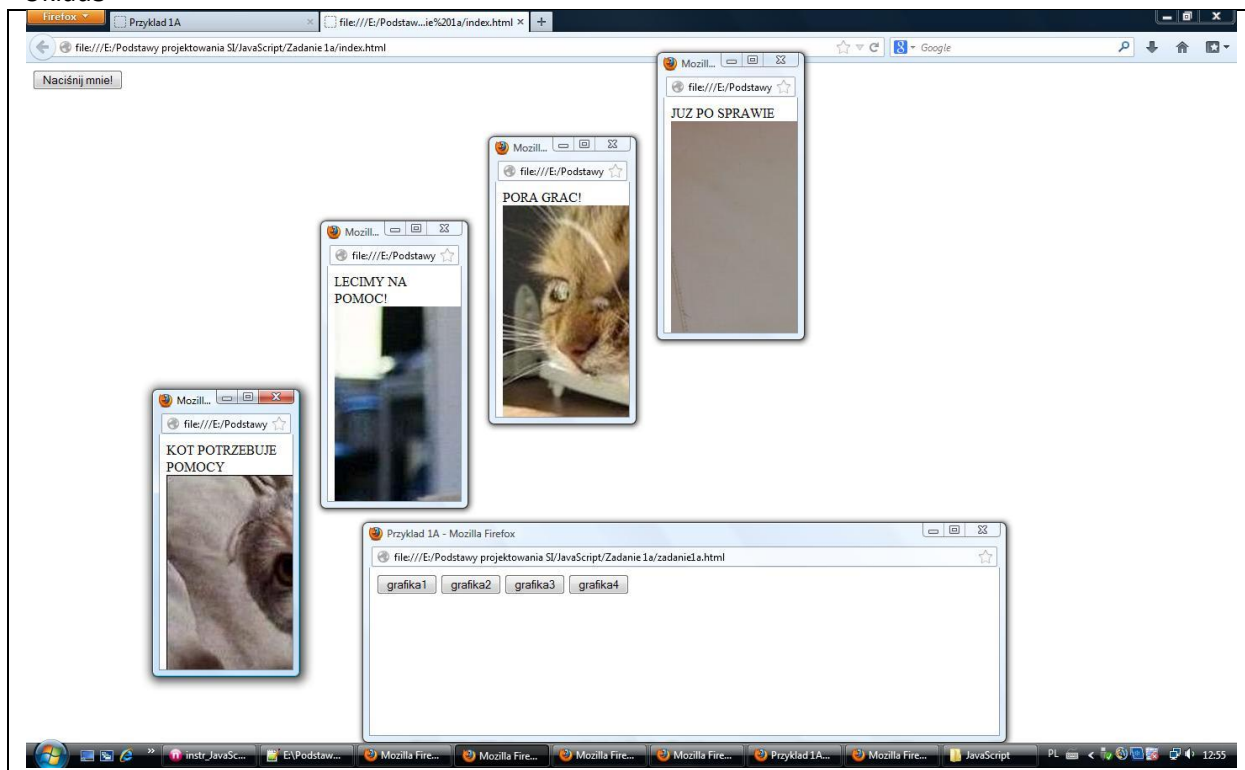
### Układ1



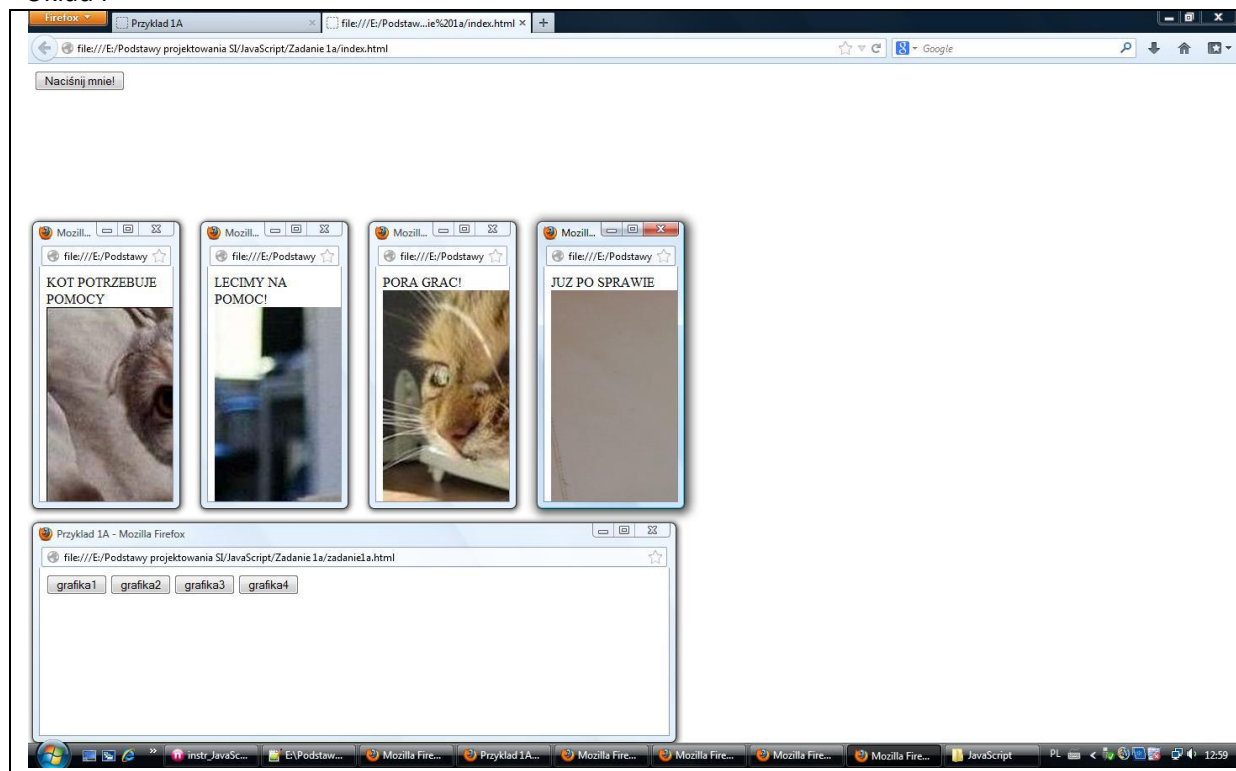
## Układ2



## Układ3



## Układ4



Narzędzia	
Polecenie	Skrót
Pobieranie	Ctrl + J
Dodatki	Ctrl + Shift + A
Otwieranie narzędzi twórców witryn	F12    Ctrl + Shift + I
Konsola WWW	Ctrl + Shift + K
Inspektor	Ctrl + Shift + C
Debugger	Ctrl + Shift + S
Edytor stylów	Shift + F7
Profilery	Shift + F5
Sieć	Ctrl + Shift + Q
Pasek programisty	Shift + F2
Widok responsywny	Ctrl + Shift + M
Brudnopis	Shift + F4
Źródło strony	Ctrl + U
Konsola przeglądarki	Ctrl + Shift + J

Postępowanie:

- 1) wybierz F12 → Otwieranie narzędzi twórców witryn
- 2) Uzyskasz dostęp do:

Inspektor	Konsola	Debugger	Edytor stylów	Wydajność	Sieć
-----------	---------	----------	---------------	-----------	------

## Dział 2 Sterowanie wydrukami. Obliczenia. Obiekty math

Kowalski-przykład1a

Kowalski-przykład1b

Kowalski-przykład1c

Kowalski-przykład1d

Przyciski zawierają Twoje nazwisko.

Poproś nauczyciela o treść zadania na tą ocenę.

Link w postaci przyciski do zadania:

Kowalski-zadanie na 3

1. Po wybraniu przycisku wyświetli się treść zadania
2. Działające rozwiązanie zadania
3. Kod zadania w polu textarea

### **Zadanie 3.**

**Temat:** Zadanie teoretyczne.

Wykonaj przycisk na stronie zaliczeniowej omawiający następujące zagadnienia teoretyczne:

- 1) Rodzaje okienek,
- 2) Konwersji liczb.

w oparciu o opis teoretyczne znajdujący się poniżej (wytluczenia oraz podkreślenia tekstu konieczne).

#### **Rodzaje okienek:**

- Okienko **alert** – wyświetlająca okienko dialogowe z informacją i przyciskiem OK,
- Okienko **prompt** – generująca okno, w którym użytkownik może wpisać wartość, zwracaną następnie do programu.
- Okienko **confirm** służy do tego, aby użytkownik coś potwierdził (stąd nazwa confirm).

#### **Konwersji liczb.**

Wczytanie zmiennej z użyciem **prompt** wykonywane jest jako ciąg znaków dlatego konieczna jest konwersja z ciągu znaków na wartości liczbowe aby wykonywać obliczenia.

parseFloat(zmienna) - konwersja zmiennej do typu float

parseInt(zmienna) - konwersja zmiennej do typu int (liczbowego)

toString(zmienna) - konwersja zmiennej do typu łańcuchowego

### **Zadanie 4.(Przykład 1a)**

Temat: **Alert Box**

Wykonaj przycisk uruchamiający następujący przykład.

#### **Zastosowanie:**

Jest używany do przekazywania informacji użytkownikowi. Użytkownik musi nacisnąć ok. aby potwierdził jej przeczytanie.

```
<html>
  <head>
    <script type="text/javascript"> function show_alert() { alert("Uwaga!"); } </script>
  </head>
  <body>
    <input type="button" onclick="show_alert()" value="Pokaż Alert Box" />
  </body>
</html>
```

### **Zadanie 5. (Przykład 1b)**

Temat: **Confirm Box**

Wykonaj przycisk uruchamiający następujący przykład.

#### **Zastosowanie:**

Jest używany, jeśli chcemy zapytać użytkownika o weryfikację, lub akceptację jakichś danych. Użytkownik musi nacisnąć „OK.” w celu akceptacji bądź „Cancel” w celu odrzucenia. Jeżeli zostanie wybrane „OK.” metoda zwraca true w przeciwnym wypadku false.

```
<html>
<head>
<script type="text/javascript">
  function show_confirm()
  {
    var r=confirm("Naciśnij klawisz");
    if (r==true)
      { document.write("Nacisnąłeś OK!"); }
    else
      { document.write("Nacisnąłeś Cancel!"); }
  }
</script>
</head>
<body>
  <input type="button" onclick="show_confirm()" value="Pokaż confirm box" />
</body>
</html>
```

### **Zadanie 6. (Przykład 1c)**

Temat: **Prompt Box**

Wykonaj przycisk uruchamiający następujący przykład.

#### **Zastosowanie:**

Jest używany kiedy użytkownik ma wpisać coś np. zanim wejdzie w dany moduł strony. Np. hasło. Użytkownik po wpisaniu wartości może nacisnąć „OK.” albo „Cancel” Jeżeli użytkownik kliknie „OK.” metoda zwraca wpisaną wartość, a w przeciwnym wypadku zwraca null.

```
<html>
<head>
<script type="text/javascript">
  function show_prompt()
  {
    var name=prompt("Napisz imię", "Imię Nazwisko");
    if (name!=null && name!="")
    {
      document.write("Witaj " + name + "! jak się masz?");
    }
  }
</script>
</head>
```



```
<body>
  <input type="button" onclick="show_prompt()" value="Pokaż prompt box" />
</body>
</html>
```

### **Zadanie 7. (Przykład 1d)**

Temat: Wczytanie dwóch liczb oraz obliczenie sumy tych liczb.

Wykonaj przycisk uruchamiający następujący przykład.

```
<html>
<body>
  <div style="color:red;font-size:50px;">
    <script language="JavaScript">
      var liczba1=prompt("liczba1=", "");
      var liczba2=prompt("liczba2=", "");
      var suma;
      var a=parseFloat(liczba1);
      var b=parseFloat(liczba2);
      suma=a+b;
      document.write("suma="+a+" + "+b+" = "+suma+"<br>");
    </script>
  </div>
</body>
</html>
```

### **Zadanie 8. (Zadanie 1a1)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Ile sekund.

Wczytać dwa numer dni z miesiąca stycznia D1\_kow, D2\_kow (trzy litery nazwiska ucznia) i obliczyć ile sekund minęło między tymi dniami dla północy obu dni. Program nie sprawdza poprawności wczytania danych czyli  $D1 \geq D2$   $1 \leq D1, D2 \leq 31$  D1,D2 należy do naturalnych. Kolor wyświetlania zielony wielkość znaków  $30 + \text{numer\_dnia urodzenia}$ .

### **Zadanie 9. (Zadanie 1a2)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Ile sekund → modyfikacja zadania 1a1.

Wczytać dwa numer dni z miesiąca stycznia D1\_kow, D2\_kow (trzy litery nazwiska ucznia) oraz dwie godziny (G1\_kow, G2\_kow) obliczyć ile sekund minęło między tymi dniami i godzinami. Program nie sprawdza poprawności wczytania danych czyli  $D1 \geq D2$   $1 \leq D1, D2 \leq 31$   $1 \leq G1, G2 \leq 24$  D1,D2,G1,G2 należy do naturalnych. Kolor wyświetlania zielony wielkość znaków  $30 + \text{numer\_dnia urodzenia}$ .

### **Zadanie 10. (Zadanie 1b1)**

Temat: Rozkład ułamka egipskiego na dwa ułamki egipskie.

Wykonaj przycisk uruchamiający następujące zadanie.

Użyj instrukcji document.write i prompt do wczytania wartości zmiennej m\_kow.

### Opis teoretyczny.

Ułamki egipskie są to inaczej ułamki proste- ułamki postaci

$$\frac{1}{m}$$

gdzie m jest liczbą naturalną większą od 0.

Ułamek ten jest rozkładany na ułamek (ułamki), które mają w liczniku też jeden. Stosujemy wzór:

$$\frac{1}{m} = \frac{1}{(m+1)} + \frac{1}{m*(m+1)}$$

### Przykład

Teraz przedstawimy liczbę 1 za pomocą sumy różnych ułamków.

Otrzymujemy:

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{6}$$

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{42}$$

$$1 = \frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{43} + \frac{1}{1806} \text{ itd.}$$

### **Zadanie 11. (Zadanie 1b2)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Rozkład ułamka egipskiego na trzy ułamki egipskie.

Wskazówka:

Rozłóż najpierw na dw ułamki i następnie drugi lub pierwszy rozłóż na dwa a otrzymasz rozkład na dwa.

### **Zadanie 12. (Zadanie 1b3)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Rozkład ułamka egipskiego na dwa takie same ułamki egipskie.

### **Zadanie 13. (Zadanie 1)**

Wykonaj przycisk uruchamiający następujące zadanie.

Użyj instrukcji document.write i prompt. Napisz program obliczając, który po podaniu dwóch liczb naturalnych wykona obliczenia według wzoru:  $(A + B)^3 = A^3 + 3A^2B + 3AB^2 + B^3$

Kolor liter granatowy a ich wielkość tak aby druga linia wydruków skryptu zajmowała całą szerokość ekranu.

### Wygląd ekranu dla okienek prompt

Podaj pierwszą liczbę

A=

Podaj drugą liczbę

B=

### Wygląd ekranu

wczytane liczby:

A=1 B=2

wygląd ekranu dla A=1 B=2 np.

$$(1 + 2)^3 = 1^3 + 3 \cdot (1^2) \cdot (2) + 3 \cdot (1) \cdot (2^2) + 2^3 = 1 + 6 + 12 + 8 = 27$$

uwaga: przy innych danych A i B liczby w napisie będą się zmieniać

### **Zadanie 14. (Zadanie 2)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Napisz program obliczający ilość atomów pierwiastka promieniotwórczego N [sztuk] po czasie t [rok] oraz czasu połowicznego  $T_{1/2}$  rozpadu danego pierwiastka.

[Przejdźcie do obiektów Math.](#)

Dane wejściowe:

No, t,  $T_{1/2}$

Dane wyjściowe

$\lambda$ , N

Najpierw oblicz  $\lambda$  i wyprowadź na ekran ze wzoru:

$$\lambda = \frac{\ln(2)}{T_{1/2}}$$

Następnie oblicz N

$$N = N_o * e^{-\lambda * t}$$

gdzie:

e - liczba Eulera

ln(x) - logarytm naturalny

$T_{1/2}$  – okres połowicznego zaniku, ( czas po jakim pozostanie połowa atomów)

$T_{1/2}$ - dla izotopu węgla C14 wynosi 5700 lat

$\lambda$  [1/ rok], stała rozpadu promieniotwórczego (określa prawdopodobieństwo zajścia rozpadu jednego jądra w jednostce czasu),

$N_o$  [sztuk]-ilość początkowa atomów pierwiastka rozpadu.

1)Wczytaj dane→prompt.

2)Wypisz wczytane dane oraz wyniki z jednostkami, dobierz kolor oraz wielkość( nie stosuj standardowych).

3)Użyj formatowania poprzez toFixed(2)( patrz przykład następny) danych dwa miejsca po przecinku dla lambdy siedem miejsc.

4)Użyj komentarzy→podaj autora.

5)do testowania użyj  $T_{1/2}$ dla węgla, wielkość masz w treści zadania oraz dla radu (odszukaj w necie)

6) Użyj obiekty **Obiekty Math** ( patrz opis w dalszej części instrukcji) np. e to Math.E podnoszenie do potęgi  $x^y$  to Math.pow(x,y)

$$N_0 = 1 [\text{sztuk}]$$

$$t = 1 [\text{lat}]$$

$$T_{1/2} = 5700 [\text{lat}]$$

$$\Lambda = 0.0001216 [1/\text{rok}]$$

$$N = 0.9998784 [\text{sztuk}]$$

### Zadanie 15. (Zadanie 2a1)

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Obliczenia z użyciem Math

Wykonaj:

1) Wczytaj dwie liczby  $x_1$  i  $x_2$  z użyciem prompt

2) Wypisz wczytane dane:

Wygląd ekranu

wczytane liczby:  $x_1=1$   $x_2=2$

3) Oblicz i wypisz

$$y = 3 * x_1^{\frac{\text{liczba liter imienia}}{\text{liczba liter imienia}+2}}$$

$$y = 3 * x_1^{\frac{\text{liczba liter imienia}}{\text{liczba liter imienia}+2}}$$

Uwaga:  $x_1$  jest podnoszone do potęgi.

4) Oblicz i wypisz

$$y_1 = 2 * \cos x_1 + 2 \sin x_1$$

$$y_1 = 2 * \cos x_1 + 2 \sin x_1$$

Pamiętaj o wczytaniu  $x_1$  w stopniach ( przy sprawdzaniu tego punktu zadania) i zamienianie na radiany

Wygląd ekranu

wczytane liczby:

w stopniach  $x_1=30$  [stopniach]  $x_1=???$  [radianach]

5) Oblicz i wypisz

$$y_3 = \sqrt[3]{(x_1)^5 + 2 * (x_2)^4} * |x_1 - x_2|$$

$$y3 = \sqrt[3]{(x1)^5 + 2 * (x2)^4 * |x1 - x2|}$$

### Zadanie 16.

Temat: Zadanie teoretyczne.

Wykonaj przycisk na stronie zaliczeniowej :

Cztery sposoby wyprowadzania danych w JS oraz innerHTML

Cztery sposoby wyprowadzania danych na stronę:

1. z użyciem dokument.write()
2. z użyciem okna alert()
3. do okienek formularza
4. do tabeli

Właściwości **innerHTML** umożliwia umieszczenie za pomocą jednej instrukcji dowolnie złożonej zawartości HTML we wskazanym elemencie np. w tabeli.

Opis → `getElementsByName`

### Zadanie 17.

Funkcje → opis, przykład

### Zadanie 18. (Przykład 2)

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Użycie formularza do obliczeń pola powierzchni oraz objętości walca.

Uruchom zadanie.

```
<html>
<head>
<title>
  Przykład obliczenia V i Pc walca
</title>
<script language="JavaScript">
function V(r,h)
{
  r_liczba=parseFloat(r)
  h_liczba=parseFloat(h)
  V_wynik=Math.PI*r_liczba*r_liczba*h_liczba
  document.getElementsByName('obj')[0].value=V_wynik.toFixed(2); /* na dokument */
  alert("V="+V_wynik.toFixed(2)+" cm^3"); /*do okna Alert */
  pokaz1.innerHTML="V="+V_wynik.toFixed(2)+" cm^3"<br>; . /* do tabeli */
}
function Pc(r,h)
{
  r_liczba=parseFloat(r)
  h_liczba=parseFloat(h)
```

```

Pc_wynik=2*Math.PI*r_liczba*r_liczba+2*Math.PI*r_liczba*h_liczba
document.getElementsByName('pole')[0].value=Pc_wynik.toFixed(2);
alert("Pc="+Pc_wynik.toFixed(2)+" cm^2");
pokaz2.innerHTML="Pc="+Pc_wynik.toFixed(2)+"cm^2";
}
</script>
</head>
<body>
  Pobiczanie V i Pc walca
<br> <br>
<form name="formularz" action="..." onsubmit="V(this.promien.value,this.wysokosc.value);
Pc(this.promien.value,this.wysokosc.value);return false">
  Podaj r=
<input size="6" name="promien">
  Podaj h=
<input size="6" name="wysokosc">
<br>
<br>
<input TYPE="submit" Value="oblicz" >
<br>
<br>
V=
<input size="6" name="obj">
Pc=
<input size="6" name="pole">
</form>
<table>
<tr id="pokaz1"></tr>
<tr id="pokaz2"></tr>
</table>

</body>
</html>

```

### **Zadanie 19. (Zadanie 2a2)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Obliczenie pola oraz obwodu prostokąta.

Napisz skrypy, który obliczy z użyciem funkcji

- a) pole,
  - b) obwód,
- Prostokąta

Nazwy funkcji:

Pole\_kowalski(a\_kow,b\_kow)

obwod\_kowalski(a\_kow,b\_kow)

nazwy pól do wczytywania danych:

a\_dana\_kow

b\_dana\_kow

nazwy pól tabeli

wyprowadz\_pole\_kow

wyprowadz\_obwod\_kow

nazwy pól Input do wyprowadzania pola i obwodu

pole

obw

### **Zadanie 20. (Zadanie 2a3)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Obliczenie pola oraz obwodu trapez równoramiennego po podaniu górnej i dolnej podstawy oraz wysokości.

Uwagi:

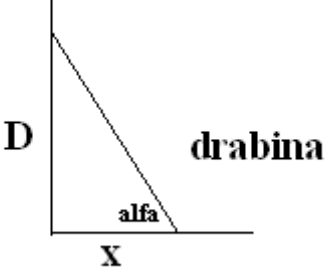
- 1)Użyj dwóch funkcji z odpowiednią ilością parametrów,
- 2)Nazwy funkcji to **Pow\_kowalski(parametry)** oraz **Obw\_kowalski(parametry)**,
- 3)W celu uniknięcia błędu, która podstawa jest dłuższa użyj wartości bezwzględnej,
- 4)ramie oblicz z twierdzenia pitagorasa,
- 5)wprowadzaj dane z użyciem formularza,
- 5)Wyprowadzanie danych z użyciem document.write oraz formularza.

### Zadanie 21. (Zadanie 3)

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Używając formularzy do wpisywania danych oraz do wypisywania rozwiązań zadanie jak poniżej.

Drabina stoi oparta o mur

	<p><u>Wczytaj:</u> D-wysokość oparcia drabiny X- odległość oparcia drabiny</p> <p><u>Oblicz:</u> długość drabiny [m] kąt nachylenia drabiny w [stopniach]</p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Do obliczenia długości drabiny użyj twierdzenia Pitagorasa. Do obliczenia kąta użyj funkcji arcustangens. Pamiętaj o zamianie radianów na stopnie. Do obliczeń użyj definiowanych własnych dwuparametrowych funkcji.

<b>Pobliczanie odległości od drabiny i kąta</b>	
Podaj d= <input type="text" value="3"/>	Podaj x= <input type="text" value="4"/>
<input type="button" value="oblicz"/>	
Drabina= <input type="text" value="5.00"/>	Alfa= <input type="text" value="36.87"/>

### Zadanie 22. (Zadanie 3a1)

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Używając formularzy do wpisywania danych oraz do wypisywania rozwiązań zadanie jak poniżej.

Artur i Dawid stoją w odległości odl\_kow ( taka zmienna do, której wczytujemy daną). Artur jest na lotnisku z którego startuje samolot pod pewnym kątem (kat\_kow). Samolot gdy znajduje się pionowo nad Dawidem przebył drogę s\_kow ( taka zmienna do, której wczytujemy daną). Oblicz na jakiej wysokości h\_kow oraz jaki był kąt startu samolotu. Do obliczeń użyj definiowanych własnych dwuparametrowych funkcji.



### **Zadanie 23. (Przykład 3)**

Wykonaj przycisk uruchamiający następujące zadanie.

Temat: Zastosowanie pliku zewnętrznego javascript \*.js do określenia ostatniej daty modyfikacji.

Uwaga: Do zaliczenia zadania wykonaj trzy przyciski:

- Przycisk1 → działanie programu
- Przycisk2 → listing do programu
- Przycisk3 → listing do pliku zewnętrznego.

1) Plik \*.html o nazwie **przyklad3.html**

```
<HTML>
<HEAD>
<TITLE>SKRYPT trzeci --> zewnętrzny</TITLE>
</HEAD>
<BODY>
    <SCRIPT type="text/javascript" src="przyklad3_zewnetrzny.js"></SCRIPT>
</BODY>
</HTML>
```

2) Plik \*.js o nazwie przyklad3\_zewnetrzny.js

```
document.write("ostatnia modyfikacja strony".fontcolor("red").bold().fontSize(7)+"<br>");
document.write(document.lastModified);
```

## **Dział 3 Programowanie obiektowe**

### **Zadanie 24.**

Udziel odpowiedzi na następujące pytania teoretyczne:

1. Rodzaje „filozofii” programowania.
2. Opisz paradygmat programowania strukturalnego. Założenia programowania strukturalnego.
3. Programowanie obiektowe definicja(angielski skrót, obiekty, stan, zachowanie, końcowa konkluzja czym jest programowanie obiektowe).
4. Dlaczego stosujemy programowanie obiektowe?
5. Omów następujące pojęcia ( w krótkiej formie),
  - Klasa
  - Obiekt
  - Właściwość
  - Metoda
  - Konstruktor
  - Dziedziczenie
  - Hermetyzacja
  - Polimorfizm

Koniec pytań teoretycznych

### **Rodzaje „filozofii-paradygmatów” programowania:**

- 1)Programowanie strukturalne
- 2)Programowanie obiektowe

**Programowanie strukturalne** – paradygmat programowania opierający się na podziale kodu źródłowego programu na procedury i hierarchicznie ułożone bloki z wykorzystaniem struktur kontrolnych w postaci instrukcji wyboru i pętli.  
Programowanie strukturalne zwiększa czytelność i ułatwia analizę programów, co stanowi znaczącą poprawę w stosunku do trudnego w utrzymaniu „*spaghetti code*” często wynikającego z użycia *instrukcji goto*.

#### **Założenia programowania strukturalnego**

1. Podzielone bloki kodu mają jeden punkt wejścia (mogą mieć wiele punktów wyjścia),
2. Wykonywanie wyrażeń w określonej kolejności,
3. Używanie instrukcji warunkowych (if, if else),
4. Używanie pętli (for, while, do while),
5. Unikanie instrukcji skoku (goto),
6. Unikanie instrukcji break, continua,
7. W programowaniu proceduralnym (strukturalne) funkcje i zmienne opisujące dany przedmiot nie są ze sobą powiązane.

#### **Opis programowania obiektowego**

*Krótki opis.*

Programowanie obiektowe jest to podejście bardziej naturalne dla ludzi, bardziej zgodne z rzeczywistością. W pamięci komputera tworzona jest wirtualna rzeczywistość. Definiowane (powoływane są do wirtualnego życia w pamięci komputera) **obiekty** na podstawie wcześniej zdefiniowanych **klas**. Obiekty mają zdefiniowane **metody** oraz **pola**.

#### Programowanie obiektowe definicja:

Programowanie obiektowe (ang. object-oriented programming) — sposób programowania, w którym programy definiuje się za pomocą:

- **obektów** (obiekty powoływane są do życia wirtualnego przez programistę, mogą być kasowane przez programistę) np. okienka w Windows → okienko jest obiektem,
- obiekt ma jakiś **stan** (czyli dane, w programowaniu obiektowym nazywane najczęściej **polami**) np. wielość okienka w Windows,
- obiekt wykazuje pewne **zachowanie** (czyli posiada pewne procedury w programowaniu obiektowym nazywane metody). Np. jest zdefiniowana metoda do zmiany wielkości okienka.

Czyli **obiektowy program komputerowy** wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.

#### **Dlaczego stosujemy programowanie obiektowe:**

- Jest to bardziej naturalne podejście przez programistę podczas tworzenia programów ponieważ otaczający nas świat to różnego rodzaju przedmioty. Tworząc program komputerowy dokonujemy pewnego ich odwzorowania rzeczywistości np. definiujemy obiekt samochód. Definiujemy pola czyli jego stan np. rok produkcji, marka, kolor. Definiujemy również metody czyli co umie np. hamowanie, przyspieszanie. Definicja obiektu jego metod i pól znajduje się w klasie. Następnie na podstawie zdefiniowanego obiektu (może mieć pola oraz metody) powołujemy obiekt w pamięci wirtualnej
- Programowanie obiektowe pozwala na lepsze gospodarowanie pamięcią komputera, ponieważ gdy obiekt staje się zbędny możemy go usunąć co zwolni pamięć zajmowaną przez ten obiekt. W programowaniu strukturalnym aby zwalniać pamięć konieczne było stosowanie dynamicznych struktur danych np. stos czy kolejka.
- W programowaniu obiektowym dobrze napisana i przetestowana klasa pozwala na stosowaniu przez innych programistów. Zaoszczędza to czas i środki pieniężne.

W przypadku programowania zorientowanego na obiekt (programowania obiektowego) dokonuje się powiązania metod (funkcji programu) z danymi (zmiennymi) definiującymi przedmiot. Wszystko to grupuje się w tzw. klasie zawierającej zarówno dane (pola) definiowanego przedmiotu, jak i funkcje (metody). W ten sposób definicje przedmiotu i jego właściwości znajdują się w jednym miejscu programu. W języku JavaScript, opis klasy jest dokonywany za pomocą pól (zmienne) i metod(funkcje).

#### **Terminologia (podsumowanie)**

##### **Klasa** (ang. "class")

Klasa rodzaj foremki, która opisuje nam jak będą wyglądać tworzone na jej podstawie nowe obiekty. Klasa jest to złożony typ będący opisem (definicją) obiektu/obektów wraz z definicją pól i metod obiektów.

##### **Obiekt** (ang. "object")

Instancja (byt, twór) który powstał na podstawie klasy. Czyli Obiekty są konstrukcjami programistycznymi mającymi tak zwane właściwości (inaczej pola obiektu), którymi mogą być zmienne lub inne obiekty. Z obiektami powiązane są funkcje wykonujące operacje na ich właściwościach, nazywane metodami.

**Właściwość** (ang. "property")

Własność obiektu, np. kolor.

**Metoda** (ang. "method")

Zdolność (czynność) którą umie wykonywać obiekt, np. chodzenie (idź). Realizowane w postaci konstrukcji podobnej do funkcji.

**Konstruktor** (ang. "constructor")

Metoda wywoływana w momencie inicjalizacji obiektu. np. tworzymy obiekt a wraz z tworzeniem obiektu uruchomi się metoda, która nada np. Imię i Nazwisko.

**Dziedziczenie** (ang. "inheritance")

Klasa może dziedziczyć własności od innej klasy. Tworzymy klasę na podstawie innej klasy pobierając od „rodzica” właściwości i metody, może dopisać nowe pola i nowe metody.

**Hermetyzacja** (lub enkapsulacja - ang. "encapsulation")

Wewnątrz ciała klasy znajdują się pola i metody. Część pól i metod można odpowiednio ukryć przed "zewnętrznym światem" klasy tak jak to ma miejsce z przedmiotami ze świata rzeczywistego.

**Polimorfizm** (ang. "polymorphism")

Polimorfizm czyli wielopostaciowość. Oznacza to, że dany obiekt może zmieniać swoją postać w zależności od potrzeb. Może to być realizowane w formie metody wywoływanej z różną ilością parametrów. Tak więc utworzony obiekt może być traktowany polimorficznie bo zachowuje się różnie w zależności od sposobu uruchamiania jego metody.

## Zadanie 25.

Udziel odpowiedzi na następujące pytania teoretyczne:

1. W jaki sposób można określić właściwość danego obiektu ( dwa sposoby wraz z przykładami)? (odpowiedź poniżej)
2. W jaki sposób można odwołać się do pól (właściwości) i metod? (odpowiedź poniżej)
3. Wymień cechy charakterystyczne konstruktora. (odpowiedź poniżej)
4. Konstruowanie obiektu na podstawie przykładu → obiekt ma trzy pola oraz jedną metodę.
5. Znaczenie słowa **this**. Jak oddzielamy pola oraz metody? (odpowiedź poniżej)
6. Konstruktor w JS na podstawie przykładu wraz z opisem.
7. Znaczenie słowa kluczowego **New** wraz z przykładem.
8. Znaczenie słowa kluczowego **prototype** wraz z przykładem i opisem ( **szczególnie ważny tekst pogrubiony**).

Koniec pytań teoretycznych

=====

Właściwość danego obiektu można określić (wpisać daną do pola odpowiedzialnego za właściwość)

### Sposób 1

Używając zapisu postaci:

a)

**nazwa\_obiektu.nazwa\_właściwości="tekst";** → dla właściwości typu tekstowego  
np.

auto.marka="Opel Omega";

b)

**nazwa\_obiektu.nazwa\_właściwości=liczba;** → dla właściwości typu liczbowego  
np.

auto.rok=1996;

auto.cena=25000;

### Sposób 2

używając zapisu postaci:

```
nazwa_obiektu[nazwa_właściwości];  
np. auto[cena]=25000;
```

### Odwołanie do pól (właściwości) i metod.

Do właściwości i metod można się odwołać podobnie jak do zwykłych zmiennych i funkcji, trzeba tylko przed ich nazwą umieścić nazwę obiektu, którego są elementami (np. zmienną, która przechowuje dany obiekt), i kropkę.

np. console.log(samochod1.kolor); → wyświetla kolor obiektu samochod

### Cechy charakterystyczne konstruktora:

- jest wywoływany (uruchamiany) automatycznie w chwili tworzenia obiektu danej klasy - na rzecz tego obiektu,
- nazywa się tak samo jak klasa,
- nie zwraca żadnej wartości ,
- może przyjmować argumenty - tak jak zwykła funkcja czy metoda, aby następnie powołać do życia obiekt za pomocą operatora **new**.

### **Przykład**

Temat: Konstruowanie obiektów→obiekt ma trzy pola oraz jedną metodę.

```
var osoba_1 =  
{  
  nazwisko      : 'Kowalski',  
  imie          : 'Jan',  
  zawod         : 'piekarz',  
  wyswietl      : function ()  
    {  
      Document.write(this.nazwisko + ' ' + this.imie)  
    }  
}
```

Uwaga1:

Słowo kluczowe **this** → pozwala odwołać się do pola lub metody obiektu z wnętrza tego obiektu.

Uwaga2:

Metody i pola muszą być oddzielne przecinkami w obrębie jednego obiektu.

### **Definiowanie klasy w JS**

Zamiast klas, JavaScript stosuje funkcje. Zdefiniowanie klasy ogranicza się do prostej czynności, jaką jest zdefiniowanie funkcji, która pełni funkcję klasy. Tak działa JS jest ponieważ JavaScript jest językiem opartym na **prototypie**, w którym nie występuje pojęcie klasy, w przeciwieństwie do języków takich, jak C++ czy Java. Fakt ten bywa dezorientujący dla programistów przyzwyczajonych do języków z pojęciem klasy

### **Przykład: Tworzenie konstruktora.**

Zostanie utworzony konstruktor o nazwie **klient** z właściwościami nazwisko, imie, zawod oraz

metoda **wypisz()**. Właściwościom obiektu zostały przypisane wartości parametrów. Użyte słowo kluczowe **this** odnosi się do aktualnego obiektu i pozwala na przypisanie wartości parametru do odpowiedniego pola tego obiektu.

```
function klient(nazwisko_k, imie_k, zawod_k)
{
  this.nazwisko=nazwisko_k;
  this.imie=imie_k;
  this.zawod=zawod_k;

  wypisz = function ()
  {
    alert(this.nazwisko + ' ' + this.imie)
  }
}
```

### **Słowo kluczowe new**

Do utworzenia nowego obiektu na podstawie konstruktora stosowane jest **słowo kluczowe new**.

Przykład

```
var osoba1 = new klient('Kowalski', 'Jan', 'kierowca');
var osoba2 = new klient('Nowak', 'Anna', 'sekretarka');
```

Powstały dwa nowe obiekty osoba1 i osoba2 należące do klasy klient.

### **Właściwość prototype**

Do definiowania metod i właściwości dla obiektu jest wykorzystana właściwość **prototype**.

Przykład

```
function klient()
{
  this.nazwisko='Bielski';
  this.imie='Paweł';
}
klient.prototype.pisz_dane = function ()
{
  document.write(this.nazwisko + ' ' + this.imie)
}
klient.prototype.zawod = 'kierowca';
var osoba1 = new klient();
osoba1.pisz_dane();
```

Opis do przykładu powyżej:

W definicji konstruktora nie zostały zadeklarowane żadne metody i właściwości. Dopiero po użyciu właściwości prototype została dodana metoda pisz\_dane oraz właściwość zawod. Od tej pory każdy nowo tworzony obiekt na podstawie konstruktora klient będzie posiadał tę dodatkową właściwość i metodę.

**Właściwość prototype może być również wykorzystana do dodawania dodatkowych metod lub właściwości do istniejących obiektów.**

=====

### Przykład

Temat: Nadawanie nowych właściwości i metod.

```
var osoba_1 =
{
  nazwisko    : 'Kowalski',
  imie        : 'Jan',
  zawod       : 'piekarz',
  wyświetl    : function ()
    {
      Document.write(this.nazwisko + ' ' + this.imie)
    }
}
czlowiek.wiek = 25;
czlowiek.wypisz_wiek = function() {alert('wiek: ' + this.wiek + 'lat')}
czlowiek.wypisz();
```

Przykład:

Temat: Tworzenie pojedynczego obiektu

Obiekt posiada dwie ustalone właściwości

- name i height
- jedną metodę, która wypisuje jego imię.

```
var myObject = {
  name: "Marcin",
  height: 184,
  print : function() {
    console.log(this.name)
  }
}
myObject.print(); //wypisze w konsoli "Marcin"
console.log(myObject.height); //wypisze 184
```

### Przykład

Temat: Nadanie nowej własności number i obliczenie kwadratu.

Aby odwołać się do danego obiektu z jego wnętrza stosujemy instrukcję this. Dzięki temu możemy w łatwy sposób wywoływać z wnętrza inne metody danego obiektu lub korzystać z jego właściwości:

```
var myObject = {
  number: 100,
  square : function() {
    return this.number * this.number
  }
}

obiekt2.number = 200;
console.log(obiekt2.square())
```

### Przykład

Temat: Definiowanie obiektów z użyciem konstruktora oraz tworzenie nowego obiektu.

W poniższym przykładzie wygenerujemy więc dwa nowe obiekty – „auto” i „osoba”, wykorzystamy je do zgromadzenia odpowiednich informacji, po czym te ostatnie wyświetlimy na ekranie komputera:

```
function auto(marka,rok,cena,wlasciciel)
{
    this.marka=marka;
    this.rok=rok;
    this.cena=cena;
    this.wlasciciel=wlasciciel;
}

function osoba(imie,nazwisko)
{
    this.nazwisko=nazwisko;
    this.imie=imie;
}
posiadacz=new osoba("Jan","Kowalski");
bryka=new auto("Ferrari",2003,200000,posiadacz);
document.write("Marka: " + bryka.marka + " rocznik: " + bryka.rok + " cena: " + bryka.cena);
document.write("<br>Wlasciciel: " + bryka.wlasciciel.imie + " " + bryka.wlasciciel.nazwisko);
```

Uwaga1

**Konstruktor** obiektu przypomina zwykłą funkcję.

Uwaga2

Słowo kluczowe **new** służy do tworzenia obiektu na podstawie konstruktora.

Przykład

Temat: Operowanie metodami → definiowanie metod wyświetlających właściwości obiektów.

```
function pokaz_auto()
{
    dane="Marka: " + this.marka + " Rocznik: " + this.rok + " Cena: " + this.cena + "<br>";
    document.write(dane);
    this.wlasciciel.pokaz()           // metoda pokaz obiektu osoba
}

function pokaz_osoba()
{
    dane="imie: " + this.imie + " nazwisko: " + this.nazwisko + "<br>";
    document.write(dane);
}

function auto(marka,rok,cena,wlasciciel)
{
    this.marka=marka;
    this.rok=rok;
    this.cena=cena;
    this.wlasciciel=wlasciciel;
    this.pokaz=pokaz_auto           // dodajemy metodę pokazującą dane naszego auta
}
```



```
function osoba(imie,nazwisko)
{
    this.nazwisko=nazwisko;
    this.imie=imie;
    this.pokaz=pokaz_osoba           // dodajemy metodę pokazującą naszą osobę
}

posiadacz=new osoba("Jan","Kowalski");
bryka=new auto("Ferrari",2003,200000,posiadacz);
bryka.pokaz()                       // pokazuje nam wszystkie właściwości naszego obiektu
```

Przykład

Temat: Właściwości i metody możemy deklarować dla nowych obiektów:

```
var myObject = {
    name: "Marcin",
    height: 184,
    print : function() {
        alert(this.name)
    }
}

myObject.weight = 73; //dodaliśmy nową właściwość
myObject.printDetail = function() {
    return {
        height : this.height,
        name :
    }
}
myObject.printDetail()
```

Przykład

Temat: Usuwanie właściwości

Aby usunąć właściwość obiektu, skorzystamy z operatora delete:

```
var myObject = {
    color: 'zielony',
    size: 'duzy',
    price: 'wielka'
}

console.log(price);

delete myObject.price; //Usuwamy właściwość cena

console.log(price); //wypisze błąd
```

**Przykład**

Temat: Tworzenie obiektu za pomocą konstruktora.

Chcemy utworzyć ich kilka. Każdy obiekt powinien posiadać jakieś właściwości i metody np. width, height i wypisz().

Aby utworzyć kilka podobnych obiektów posłużymy się klasą obiektu.

W JS w przeciwieństwie do innych języków nie mamy mechanizmu konstruktora, ale samą klasę możemy stworzyć za pomocą zwykłej funkcji:

```
function SuperObjectClass(_width,_height) {
  this.width = _width;
  this.height = _height;
  this.print = function() {
    console.log(this.width + 'x' + this.height)
  }
}

var myObject1 = new SuperObjectClass(200, 100);
var myObject2 = new SuperObjectClass(300, 200);

myObject1.print(); //wypisze 200x100

myObject2.width = 600;
myObject2.print() //wypisze 600x200
```

### Zadanie 26.

Na podstawie Przykładu1 (poniżej), wykonaj program z użyciem programowania obiektowego. Zdefiniuj obiekt, trzy właściwości z wpisanymi danymi, jedną metodę wyświetlającą wartości pól właściwości (nazwa metody to nazwa obiekt\_wyświetl\_nazwisko\_ucznia np. samochod\_wyświetl\_kowalski).

Nr w tabeli	Nazwa obiektu	Właściwości	Metoda
1	samochod		samochod_wyświetl_nazwisko
2	ciezarowka		ciezarowka
3	Dom		dom
4	pudelko		pudelko
5	Wazon		wazon
6	Pociąg		pociąg
7	Mebel		mebel
8	autobus		autobus
9	komputer		komputer
10	monitor		monitor
11	Router		router
12	Telefon		telefon
13	Spodnie		spodnie
14	Owoc		owoc
15	wiezowiec		wiezowiec

Przykład 1→dotyczący programowania obiektowego.

Temat: Utwórz obiekt o nazwie **człowiek**, polach **nazwisko**, **imie**, **zawod** i metodzie **pokaz**, która wyświetla nazwisko i imię.

```
<html>
<head>
  <script type="text/javascript">
    var czlowiek = {
```

```

        nazwisko: 'Nowacki',
        imie: 'Marek',
        zawod: 'informatyk',
        pokaz: function ()
        {
            document.write(this.nazwisko + ' ' + this.imie)
        }
    }
    czlowiek.pokaz();
</script>
</head>
<body>
<br>
    programowanie obiektowe
</body>
</html>

```

Uwagi:

- 1) Przy definiowaniu obiektu deklarowane właściwości i funkcje muszą być oddzielone przecinkami.
- 2) Słowo kluczowe **this** pozwala odwołać się do właściwości lub metod danego obiektu z jego wnętrza tego obiektu.

### Zadanie 27.

Na podstawie Przykładu 2 (poniżej), uzupełnij program z zadania poprzedniego (każdy uczeń miał inny obiekt) z użyciem programowania obiektowego.

Dopisz jedną właściwość (poza ciałem obiektu) z wpisaną daną, jedną metodę wyświetlającą wartości pola właściwości (dopisaną w zadaniu poprzednim),

nazwa metody to:

nazwa obiekt\_pokaz\_nazwisko\_ucznia

np. samochod\_pokaz\_kowalski (metoda dopisana poza ciałem obiektu).

### Przykład 2

Temat: Dla istniejących już obiektów można deklarować nowe właściwości i metody (poza klasą/funkcją).

```

<html>
<head>
    <script type="text/javascript">
        var czlowiek = {
            nazwisko: 'Nowacki',
            imie: 'Marek',
            zawod: 'informatyk',
            pokaz: function ()
            {
                document.write(this.nazwisko + ' ' + this.imie)
            }
        }
        czlowiek.pokaz();
        czlowiek.wiek=19;
        czlowiek.wypisz=function() {alert('Wiek: ' + this.wiek + 'lat')}
    </script>
</head>
<body>
    programowanie obiektowe
</body>
</html>

```

```
        czlowiek.wypisz();
    </script>
</head>
<body>
<br>
    programowanie obiektowe
</body>
</html>
```

#### Przykład 4

##### Temat: **Tworzenie obiektu za pomocą konstruktora**

Chcemy utworzyć ich kilka. Każdy obiekt powinien posiadać jakieś właściwości i metody np. width, height i wypisz(). Aby utworzyć kilka podobnych obiektów posłużymy się klasą obiekt. W JS w przeciwieństwie do innych języków nie mamy mechanizmu konstruktora, ale samą klasę możemy stworzyć za pomocą zwykłej funkcji:

```
function SuperObjectClass(_width,_height) {
    this.width = _width;
    this.height = _height;
    this.print = function() {
        console.log(this.width + 'x' + this.height)
    }
}
```

```
var myObject1 = new SuperObjectClass(200, 100);
var myObject2 = new SuperObjectClass(300, 200);
```

```
myObject1.print(); //wypisze 200x100
```

```
myObject2.width = 600;
myObject2.print() //wypisze 600x200
```

**Konstruktor** jest wywoływany w momencie instancjalizacji (moment, w którym instancja obiektu zostaje utworzona). Konstruktor jest metodą klasy. W JavaScript, funkcja służy za konstruktor obiektu. Nie ma jednak wyraźnej potrzeby definiowania konstruktora. Każda akcja zadeklarowana w konstruktorze zostanie wykonana w momencie utworzenia obiektu.

**Konstruktor** jest używany do ustawienia właściwości obiektu lub do wywołania metod przygotowujących obiekt do użytku.

W języku JavaScript istnieje możliwość tworzenia wielu obiektów posiadających podobne właściwości. W tym celu można posłużyć się konstruktorem obiektu. Konstruktor przypomina zwykłą funkcję.

Do właściwości obiektu można się do nich odwoływać na dwa sposoby:

```
nazwa_obiektu.nazwa_właściwości
np.
klient.nazwisko,

nazwa_obiektu["nazwa_właściwości"]
```

```
np.  
klient["nazwisko"].
```

Przykład

Temat: instrukcja Prototype

Przypuśćmy, że mamy już stworzone jakieś obiekty. Po jakimś czasie chcielibyśmy dodać do nich nową metodę. Dodajemy więc nową metodę do naszej foremki (klasy), ale okazuje się, że dostaną ją dopiero nowe obiekty stworzone na podstawie tej formy. Aby zmienić prototyp (a co za tym idzie i wszystkie obiekty stworzone na jego podstawie) posłużymy się instrukcją prototype.

```
function SuperObjectClass() {  
    this.name = 'Marcin';  
    this.height = 183  
}  
  
//dodaję właściwość i metodę do prototypu  
SuperObjectClass.prototype.weight = 73;  
SuperObjectClass.prototype.showInfo = function () {  
    alert(this.name + ' ma ' + this.height + 'cm wzrostu.')  
}  
var myObject = new SuperObjectClass();  
myObject.showInfo();  
console.log(myObject.weight); //wypisze sie 73
```

### Zadanie 28.

Na podstawie Przykładu3 (poniżej), uzupełnij program z dwóch zadań poprzednich z użyciem programowania obiektowego.

Program powinien zawierać:

- Definiowanie konstruktora (klasy). Należy zauważyć, że w JS w wersji przed 2015 nie ma możliwości definiowania klasy (tak jak w innych językach obiektowych). Dlatego definiowanie klasy odbywa się z użyciem konstruktora (słowo kluczowe function).
- dodawanie metody do konstruktora z użyciem prototypu, (z czterema polami),
- powoływanie obiektów, (powołanie 3 obiektów),
- dostęp do właściwości (pól) obiektów, (wyświetlenie danych o wszystkich obiektach),
- użycie zdefiniowanej metody. (wyświetlenie danych o wszystkich obiektach).

Dopisz jedną właściwość (poza ciałem obiektu) z wpisaną daną, jedną metodę wyświetlającą wartości pola właściwości (dopisaną w zadaniu 2??), nazwa metody to nazwa obiekt\_pokaz\_nazwisko\_ucznia np. samochod\_pokaz\_kowalski (metoda dopisana poza ciałem obiektu).

### Przykład 3

Temat: Definiowanie konstruktora, dodawanie metody do konstruktora z użyciem prototypu, powoływanie obiektów, dostęp do właściwości (pól) obiektów, użycie zdefiniowanej metody.

```
<html>  
  <head>  
    <script type="text/javascript">  
      function klient(nazwisko_k,imie_k,zawod_k)//początek konstruktora  
      {  
          this.nazwisko=nazwisko_k;  
          this.imie=imie_k;
```



```

this.name = 'Marcin';
this.height = 183
this.button = null;

this.init = function() {
  this.button = document.createElement('input');
  this.button.addEventListener('click', function() {
    this.value = this.height ??????
  });
  document.querySelector('body').appendChild(this.button);
}
this.init();
}

var someObject = new SuperObjectClass();

```

Po wywołaniu metody init tworzymy nowy guzik. Po jego kliknięciu powinien on ustawić swoje value na height SuperObjectClass. Jak jednak to zrobić, skoro instrukcja this wewnątrz obiektu guzika wskazuje na niego samego?

Są na to dwa sposoby: posłużenie się dodatkową zmienną that:

```

var SuperObjectClass = function() {
  this.name = 'Marcin';
  this.height = 183
  this.button = null;

  this.init = function() {
    var that = this;

    this.button = document.createElement('input');
    this.button.value = 'Kliknij';
    this.button.type = 'button';
    this.button.addEventListener('click', function() {
      alert('To jest ' + this.nodeName); //przycisk
      alert('Wzrost Marcina: ' + that.height); //obiekt SuperObjectClass
    });
    document.querySelector('body').appendChild(this.button);
  }
  this.init();
}

var someObject = new SuperObjectClass();

```

lub skorzystanie z instrukcji bind(), za pomocą której możemy przekazać kontekst do danej funkcji:

```

var SuperObjectClass = function() {
  this.name = 'Marcin';
  this.height = 183
  this.button = null;

  this.init = function() {
    this.button = document.createElement('input');
    this.button.value = 'Kliknij';

```

```

    this.button.type = 'button';
    this.button.addEventListener('click', function(e) {
        //tutaj już this wskazuje na SuperObjectClass
        //dlatego dany przycisk musimy pobrać za pomocą e.target
        alert(e.target.value = this.height);
    }.bind(this));
    document.querySelector('body').appendChild(this.button);
}
this.init();
}
var someObject = new SuperObjectClass();

```

Bardzo dużo osób neguje stosowanie dodatkowej zmiennej do przekazywania kontekstu obiektu. Pamiętać należy jednak, że stosując instrukcję bind() tracamy dostęp do this danego pod obiektu.

Przykład

Temat: Metoda Watch oraz Unwatch

Javascript udostępnia nam metodę:

**watch("nazwaWlasciwosci", funkcja(id, staraWartosc, nowaWartosc),**

która służy do podglądu właściwości obiektów. Jej działanie jest takie samo jak w innych językach.

Gdy podglądana wartość się zmieni, wówczas watch wywoła funkcję podaną w drugim atrybucie.

Przekazujemy jej 2 parametry - pierwszy określa nazwę obserwowanej właściwości, drugi to funkcja z 3 atrybutami: id, oldValue, newValue. Parametry te są wypełniane automatycznie. Gdy obserwowana wartość się zmieni, funkcja podana w 2 atrybucie zostanie wywołana:

```

var obiekt = {p:1}

//śledzenie właściwości "p" obiektu "obiekt"
obiekt.watch("p", function (id, oldValue, newValue) {
    console.log("o." + id + " zmieniło się z " + oldValue + " na " + newValue)
    return nowaWart;
});

//teraz przy każdej zmianie wartości obiektu wywołany zostanie callback
//który śledzi zmiany w obiekcie
obiekt.p = 2
obiekt.p = 3

```

Aby zakończyć obserwowanie, korzystamy z metody unwatch("nazwa\_obserwowanej\_wlasciwosci").

```

obiekt.unwatch('p')

```

## Zadanie 29.

1)Stwórz nowy konstruktor "**User**", którego argumentami będą:

- adres e-mail,
- hasło
- adres url awatara

2)Stwórz obiekt **userList**, który będzie posiadał metodę "**addUser**", przyjmującą jako argument instancję User i zapamiętującą ją w obiekcie, oraz metodę "login" z argumentami "email" oraz "password", zwracającą true w przypadku podania prawidłowych danych dla któregoś z użytkowników, oraz "false" w przypadku podania nieprawidłowych danych.

3)Stwórz prosty formularz, zawierający pola "login" oraz "hasło", który wyświetli komunikat z informacją, czy udało się zalogować



**Zadanie 30.**

1) Stwórz nowy konstruktor "UserView", którego argumentem będzie instancja User

W konstruktorze UserView wygeneruj element li, zawierający obrazek z awatarem użytkownika, oraz jego adres e-mail umieszczony jako link z protokołem "mailto:" (nadaj tym elementom jakieś ładne style :)). Przypisz element dom oraz obiekt User jako własności tworzonego obiektu.

W prototypie UserView utwórz metodę refresh odświeżającą dane użytkownika w elemencie dom.

## Dział 3a Programowanie obiektowe-definiowane w przeglądarce

Przykład

Kod	Wytłumaczenie
var napisz="Witaj w szkole";	utworzenie obiektu <b>napisz</b> i przypisanie wartości "Witaj w szkole"
document.write(napisz.toUpperCase());	UpperCase() jest metodą, która działa na obiekt <b>napisz</b> , → zamienia małe litery na duże
document.write(napisz.length);	napisz.length jego właściwością obiektu <b>napisz</b> → podaje długość tekstu zapisanego w obiekcie <b>napisz</b>
Wynikiem będzie wypisanie tekstu: "WITAJ W SZKOLE15"	

### **Zadanie 31.**

Udziel odpowiedzi na następujące pytania teoretyczne:

Wymień oraz opisz obiekty przeglądarki

- window
- navigator

Koniec pytania teoretycznego

=====

Przeglądarka internetowa wykonana jest z użyciem programowania obiektowego.

### **Obiekty przeglądarki**

Dla każdej strony internetowej zdefiniowane są następujące obiekty:

- window
- navigator
- document
- location
- history

### **Obiekt window**

1. Opisuje bieżące okno przeglądarki.
2. Jest najważniejszym obiektem w hierarchii, dlatego odwołanie do jego właściwości lub metod nie wymaga podania nazwy obiektu.
3. Tworzony jest automatycznie podczas otwierania okna przeglądarki.
4. Posiada wiele właściwości przydatnych podczas tworzenia strony.

### **Obiekt navigator**

1. Pozwala na dostęp do informacji dotyczących przeglądarki.
2. Służy do ustalenia wersji przeglądarki używanej przez użytkownika.
3. Właściwości tego obiektu mogą być tylko odczytywane nie mogą być zmieniane.
4. Najczęściej używa się go do sprawdzenia, czy przeglądarka jest odpowiednia do zastosowanej wersji języka JavaScript.
5. Szczególnie przydatnym jest window.navigator.userAgent – to tekst identyfikujący przeglądarkę.

### **Zadanie 32.** (dawniej Zadanie 32a)

Uruchom poniższy przykład:

Przykład1 do zadania 32a

Temat:

## Rozpoznawanie typu przeglądarki przy użyciu **navigator** i jej właściwości **userAgent**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="pl">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <title>Moja strona WWW</title>
  </head>
  <body>
    <script type="text/javascript">
      var agent = navigator.userAgent.toLowerCase();
      var nazwa = "nieznany";
      if(agent.indexOf('firefox') != -1){nazwa = "Firefox";}
      if(agent.indexOf('opera') != -1){nazwa = "Opera";}
      else if(agent.indexOf('konqueror') != -1){nazwa = "Konqueror";}
        else if(agent.indexOf('netscape') != -1){nazwa = "Netscape Navigator";}
          else if(agent.indexOf('msie') != -1){nazwa = "Internet Explorer";}
            document.write("Twoja przeglądarka to: " + nazwa);
    </script>
  </body>
</html>
```

### Opis do przykładu powyżej:

Tekst znajdujący się we właściwości **userAgent** jest konwertowany za pomocą metody **toLowerCase**, tak aby zawierał wyłącznie małe litery (co ułatwia dalsze operacje), i jest zapisywany w pomocniczej zmiennej **agent**. Następnie za pomocą metody **indexOf** jest sprawdzane, czy w ciągu zapisanym w **agent** znajduje się któreś ze słów charakterystycznych dla jednej z rozpoznawanych przeglądarek. Jeśli tak, zmiennej **nazwa** jest przypisywana nazwa rozpoznanego produktu. Nazwa ta jest wyświetlana na ekranie za pomocą instrukcji **document.write**.

### *koniec przykładu*

=====

### **Zadanie 33.** (dawniej przykład 32b )

Uruchom poniższy przykład (dopisując część kodu aby całość działała):

Uruchom dwa poniższe przykłady w jednym skrypcie aby otrzymać wygląd ekranu jak poniżej.

```
pierwszy sposób
mam przeglądarkę= Netscape
drugi sposób
Masz niewłaściwą wersję przeglądarki!!!
```

Przykłady do uruchomienia:

```
document.write("mam przeglądarkę= "+navigator.appName);
```

inny przykład

```
if((navigator.appName=="Firefox") && (parseInt(navigator.appVersion)>=3))
  { document.write("mam przeglądarkę="+navigator.appName);}
```

```
else
{document.write("Masz niewłaściwą wersję przeglądarki!!!");}
```

*Opis właściwość użytych w przykładzie*

appName → zawiera nazwę przeglądarki,

appVersion → zawiera numer wersji przeglądarki,

funkcja parseInt() → zamienia dowolny łańcuch na liczbę.

*koniec przykładu*

### **Zadanie 34.** (dawniej przykład 32c )

Uruchom poniższy przykład:

Przykład3 do zadania 32c → do **navigator** i jej właściwości **userAgent**

Uruchom dwa poniższe przykłady w jednym skrypcie aby otrzymać wygląd ekranu jak poniżej.

## Rozpoznawanie typu systemu operacyjnego

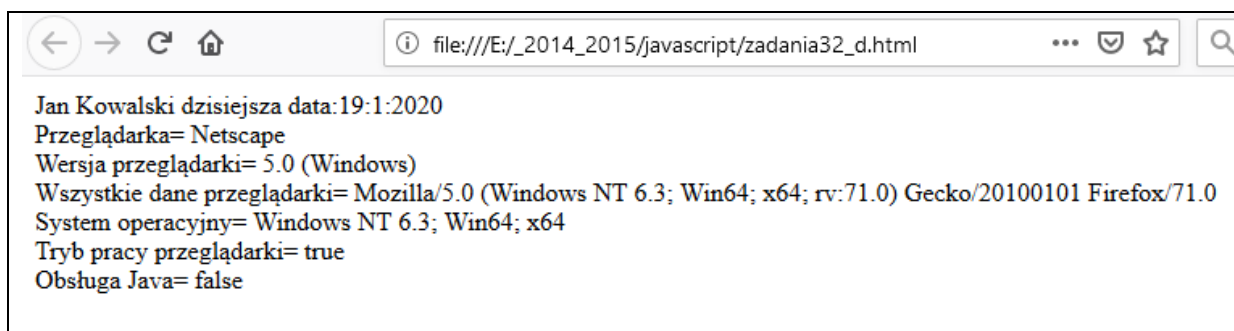
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="pl">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <title>Moja strona WWW</title>
  </head>
  <body>
    <script type="text/javascript">
      var agent = navigator.userAgent.toLowerCase();
      var nazwa = "nieznany";
      if(agent.indexOf('windows') != -1) {nazwa = "Windows";}
      if(agent.indexOf('linux') != -1) {nazwa = "Linux";}
      else if(agent.indexOf('mac') != -1) {nazwa = "MacOS";}
      document.write("Twój system to: " + nazwa);
    </script>
  </body>
</html>
```

*koniec przykładu*

### **Zadanie 35.** (dawniej Zadanie 32d )

Treść zadania:

Na podstawie przykładów oraz tabeli wygeneruj następujący tekst w oknie przeglądarki:



W celu rozwiązania zadania:

**1)Użyj przykład, który pokazuje jak uzyskać datę:**

```
<html>
<!-- metatagi -->
<head>
<script type="text/javascript">
function wyswietlDate(){
    var Today = new Date();
    var Month = (Today.getMonth()+1);
    var Day = Today.getDate();
    var Year = Today.getFullYear();
    if(Year <= 99)    Year += 1900
    return Day + "-" + Month + "-" + Year;
}
</script>
</head>
<body>
<script type="text/javascript">
    document.write("Dzisiaj jest " + wyswietlDate());
</script>
</body>
</html>
```

**2) Podpowiedzi na temat rozwiązania zadania:**

Uczeń [tutaj twoje imię i nazwisko] ustalił w dniu [dzisiejsza data→ ustalona z użyciem odpowiedniego obiektu JS, patrz przykład powyżej] następujące dane:

Wykrywana cecha	Wygląd ekranu
Typ przeglądarki z użyciem <b>appName</b>	Przeglądarka=.....
Wersja przeglądarki z użyciem <b>navigator.appVersion</b>	Wersja przeglądarki=.....
Wszystkie dane dotyczące przeglądarki z użyciem <b>userAgent</b>	Wszystkie dane Przeglądarki=.....
Określenie systemu operacyjnego, na którym jest uruchomiona przeglądarka <b>oscpu</b>	System operacyjny =.....
W jakim trybie pracuje przeglądarka <b>onLine</b>	Przeglądarka pracuje w trybie=.....
Czy przeglądarka obsługuje Java. Użyj metody <b>javaEnabled()</b>	Obsługa Java=.....

**Tabela: Właściwość obiektu navigator**

Nazwa	Znaczenie	Dostępność
appCodeName	Nazwa kodowa przeglądarki.	FF, IE, NN, OP
appName	Oficjalna nazwa przeglądarki.	FF, IE, NN, OP
appVersion	Wersja kodowa przeglądarki.	FF, IE, NN, OP
appMinorVersion	Podwersja przeglądarki	IE, OP
cookieEnabled	Określa, czy w przeglądarce jest włączona obsługa plików cookie (true — tak, false — nie).	FF, IE, NN, OP
cpuClass	Rodzina procesorów urządzenia, na którym jest uruchomiona przeglądarka.	IE

language	Kod języka przeglądarki.	FF, NN, OP
mimetypes	Tablica zawierająca listę typów mime obsługiwanych przez przeglądarkę. W niektórych przypadkach ta właściwość jest wartością pustą.	FF, IE, NN, OP
onLine	Określa, czy przeglądarka pracuje w trybie online.	FF, IE
oscpu	Określa system operacyjny, na którym jest uruchomiona przeglądarka.	FF, NN
platform	Określa platformę systemową, dla której jest przeznaczona przeglądarka.	FF, IE, NN, OP
plugins	Tablica zawierająca odniesienia do rozszerzeń zainstalowanych w przeglądarce.	FF, IE, NN, OP
product	Nazwa produktowa przeglądarki (np. Gecko).	FF, NN
productSub	Wersja produktowa przeglądarki.	FF, NN
systemLanguage	Język systemu operacyjnego, w którym jest uruchomiona przeglądarka.	IE
userAgent	Ciąg wysyłany przez przeglądarkę do serwera jako nagłówek HTTP_USER_AGENT. Z reguły pozwala na dokładną identyfikację przeglądarki.	FF, IE, NN, OP
userLanguage	Język użytkownika (z reguły wersja językowa przeglądarki).	IE, OP
vendor	Dostawca (producent) przeglądarki.	FF, NN
vendorSub	Numer produktu według producenta (np. 5.1, 6.0).	FF, NN

## Metody obiektu navigator

### Metoda *javaEnabled*

Wywołanie: `javaEnabled()`

Metoda `javaEnabled` zwraca wartość `true`, jeżeli dana przeglądarka obsługuje Javę, lub `false` w przeciwnym razie.

=====

### **Zadanie 36.** (dawniej Zadanie 33 -T )

Opisz teorię dotyczącą Obiekt dokument:

#### Obiekt `document`

- Zawiera informacje o bieżącym dokumencie HTML.
- Poprzez jego właściwości `document` mamy wpływ na elementy strony (np. kolor tła, kolor czcionki).
- Metody `document` umożliwia wyświetlenie np. tekstu w oknie przeglądarki.

Koniec teorii

=====

### **Zadanie 37.** (dawniej Zadanie 33 P )

Uruchom następujący przykład a następnie dopisz i wyświetlanie dwie właściwości z tabeli poniżej.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="pl">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

```

<meta http-equiv="Content-Script-Type" content="text/javascript">
<title>Moja strona WWW</title>
</head>
<body>
  <script type="text/javascript">
    document.write("Podstawowe informacje o dokumencie:<br />");
    document.write("Tryb kompatybilności: ");
    document.write(document.compatMode + "<br />");
    document.write("URL: " + document.URL + "<br />");
    document.write("Liczba apletów: ");
    document.write(document.applets.length + "<br />");
    document.write("Liczba obrazów: ");
    document.write(document.images.length + "<br />");
    document.write("Liczba formularzy: ");
    document.write(document.forms.length + "<br />");
    document.write("Tytuł: " + document.title + "<br />");
    document.write("Data ostatniej modyfikacji: ");
    document.write(document.lastModified + "<br />");
  </script>
</body>
</html>

```

koniec przykładu

=====

Tabela Wybrane właściwości obiektu document

Nazwa	Znaczenie	Dostępność
all	Obiekt zawierający odniesienia do wszystkich elementów dokumentu, charakterystyczny dla przeglądarki Internet Explorer.	IE, OP
alinkColor	Kolor aktywnego odnośnika.	FF, IE, NN, OP
anchors	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Anchor.	FF, IE, NN, OP
applets	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Applet.	FF, IE, NN, OP
bgColor	Kolor tła dokumentu.	FF, IE, NN
body	Obiekt zawierający treść dokumentu HTML.	FF, IE, NN, OP
characterSet	Ciąg określający kodowanie znaków w dokumencie.	FF, NN
compatMode	Ciąg określający tryb kompatybilności dokumentu ze standardami HTML.	FF, IE, NN, OP
cookie	Ciąg znaków zawierający cookies danego dokumentu.	FF, IE, NN, OP
docType	Obiekt określający typ dokumentu (DTD).	FF, NN, OP
domain	Nazwa domenowa serwera, z którego pochodzi dokument.	FF, IE, NN, OP
embeds	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów zagnieżdżonych.	FF, IE, NN, OP
fgColor	Kolor tekstu dokumentu.	FF, IE, NN, OP
fileCreatedDate	Data utworzenia pliku zawierającego aktualnie otwarty dokument, w formacie mm/dd/yyyy. Właściwość charakterystyczna dla przeglądarek z rodziny Internet Explorer.	IE
fileModifiedDate	Data ostatniej modyfikacji pliku zawierającego aktualnie otwarty dokument, w formacie mm/dd/yyyy. Właściwość charakterystyczna dla przeglądarek z rodziny Internet Explorer. Należy wziąć pod uwagę fakt, że nie każdy serwer WWW dostarcza taką informację.	IE
fileSize	Rozmiar pliku zawierającego aktualnie otwarty dokument.	IE

	Właściwość charakterystyczna dla przeglądarek z rodziny Internet Explorer. Należy wziąć pod uwagę fakt, że nie każdy serwer WWW dostarcza taką informację.	
forms	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów formularzy.	FF, IE, NN, OP
height	Wysokość dokumentu.	FF, NN
images	Tablica zawierająca odniesienia do znajdujących się w dokumencie obrazów.	FF, IE, NN, OP
implementation	Obiekt typu DOMImplementation pozwalający stwierdzić, które elementy modelu DOM są implementowane przez bieżące środowisko.	
lastModified	Zwiera datę i czas ostatniej modyfikacji dokumentu.	FF, IE, NN, OP
linkColor	Definiuje kolor odnośnika.	FF, IE, NN, OP
links	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Link (odnośników).	FF, IE, NN, OP
location	Obiekt przechowujący URL bieżącego dokumentu.	FF, IE, NN, OP
plugins	Tablica zawierająca odniesienia do znajdujących się w dokumencie obiektów typu Plugin.	FF, IE, NN, OP
referrer	Zawiera URL dokumentu, z którego nastąpiło odwołanie do bieżącego dokumentu.	FF, IE, NN, OP
styleSheets	Zawiera wszystkie style zdefiniowane w dokumencie.	FF, IE, NN, OP
title	Zawiera tytuł dokumentu zdefiniowany za pomocą znacznika <title>.	FF, IE, NN, OP
URL	Ciąg zawierający URL bieżącego dokumentu.	FF, IE, NN, OP
vLink	Kolor odwiedzonego odnośnika.	FF, IE, NN, OP
width	Szerokość dokumentu.	FF, NN

=====

### **Zadanie 38.** (dawniej Zadanie 34 T )

Opisz teorię dotyczącą Obiekt location.

#### **Obiekt location**

Obiekt ten zawiera informacje o bieżącym adresie URL. Jego właściwości odpowiadają kolejnym członom adresu.

Ogólna postać lokalizatora URL to:

protocol://host:port/path#fragment?query

#### **Właściwości obiektu location**

protocol → to łańcuch znakowy określający protokół, zgodnie z którym

należy nawiązać łączność z podanym serwerem (np. http:, ftp:, file:),

host → to łańcuch znakowy określający nazwę serwera z bieżącego adresu, port

to łańcuch znakowy odpowiadający portowi, przez który należy połączyć się z serwerem,

pathname → to łańcuch znakowy określający ścieżkę dostępu do dokumentu na serwerze,

hash → to łańcuch znakowy odpowiadający nazwie zakotwiczenia (przypisanie tu jakiejś wartości spowoduje przewinięcie dokumentu do wskazanego punktu),

serach → to człon lokalizatora URL.

Właściwość obiektu location w formie tabeli.

Nazwa	Znaczenie	Dostępność
hash	Część adresu URL znajdująca się za znakiem #.	FF, IE3, NN2, OP7
host	Część adresu URL zawierająca nazwę hosta oraz numer portu.	FF, IE3, NN2, OP7



hostname	Część adresu URL zawierająca nazwę hosta.	FF, IE3, NN2, OP7
href	Pełny adres URL.	FF, IE3, NN2, OP7
pathname	Część adresu URL zawierająca ścieżkę dostępu i nazwę pliku.	FF, IE3, NN2, OP7
port	Część adresu URL zawierająca numer portu.	FF, IE3, NN2, OP7
protocol	Część adresu URL zawierająca nazwę protokołu (np. http, ftp).	FF, IE3, NN2, OP7
search	Część adresu URL znajdująca się za znakiem zapytania.	FF, IE3, NN2, OP7

## Metody obiektu location

### 1)Metoda assign

Wywołanie: location.assign(url)

Metoda assign wczytuje dokument o adresie wskazanym przez argument url.

### 2)Metoda reload

Wywołanie: location.reload(force)

Metoda reload wymusza ponowne wczytanie bieżącej strony. Jeśli obecny jest argument force i ma on wartość true, wymusza to ponowne wczytanie zawartości witryny z serwera. W pozostałych przypadkach przeglądarka może wczytać ją z pamięci cache.

### 3)Metoda replace

Wywołanie: location.replace(url)

Metoda replace zastępuje bieżący dokument przez wczytany spod adresu wskazanego przez argument url.

#### Różnica między assign a replace

W przeciwieństwie do metody assign, po zastosowaniu replace bieżąca strona nie zostanie zapisana w historii przeglądanych witryn, nie będzie więc można się do niej ponownie odwołać poprzez wciśnięcie przycisku Wstecz bądź też wywołanie history.go(-1).

Tę metodę można wykorzystać np. w sytuacji, kiedy witryna została przeniesiona pod inny adres, a pod starym ma być umieszczona stosowna informacja oraz automatyczne przekierowanie użytkownika (po zadany czasie).

### 4)Metoda toString

Wywołanie: location.toString()

Metoda toString przekształca zawartość obiektu location w ciąg znaków reprezentujący przechowywany przezeń adres URL.

1. window.location.href →przechowuje pełny adres
2. window.location.hostname →wyłącznie domenę.
3. window.location.reload() → odświeża stronę,
4. window.location.assign() → przechodzi na podany adres,
5. window.location.replace() → przechodzi na podany adres i nie tworzy nowego wpisu w historii przeglądania.

## **Zadanie 39.** (dawniej Zadanie 34 P )

Na podstawie przykładów zapisanych poniżej oraz zapisów teoretycznych wykonaj stronę z czterema przyciskami, przycisk powinien mieć odpowiedni opis np. Kowalski-strona CKE (nazwisko ucznia):

1. przycisk 1 →strona CKE uruchamiana po 4 sekundach
2. przycisk 2 → strona oficjalna Gdańska uruchamiana po 6 sekundach
3. przycisk 3 → strona zaproponowana przez ucznia uruchamiana po 10 sekundach
4. przycisk 4 → strona odświeżenie aktualnej strony uruchamiana po 8 sekundach

Przykład:

Aby zmienić adres strony wyświetlanej w oknie, wystarczy zmienić lokalizator URL.

```
window.location = https://www.google.pl/search?q=warszawa+lotnisko
```

```
window.location = "http://helion.pl/ksiazki/cwjas2.htm";
```

Przykład:

Wyświetlanie strony po naciśnięciu PRZYCISKU (button)

```
<html>
  <head>
    <title>Przykład nr tutaj wpisz odpowiedni numer oraz nazwisko </title>
    <script language="JavaScript">
      function ZSE_Open()
      {
        window.location = "http://zse.gda.pl";
      }

      function OKE_Open()
      {
        window.location = "http://oke.gda.pl";
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" name="przycisk1" value="Strona ZSE" onclick="ZSE_Open(' ')">
      <input type="button" name="przycisk2" value="Strona OKE" onclick="OKE_Open(' ')">
    </form>
  </body>
</html>
```

Przykład:

Przekierowanie użytkownika na nowy adres

### **setTimeout(fn, time)**

Służy ona wywołania z opóźnieniem funkcji przekazanej w pierwszym parametrze. W drugim parametrze podajemy czas w milisekundach po jakim zostanie ta funkcja wywołana.

fn → wywoływana funkcja

time → czas opóźnienia w milisekundach

```
window.setInterval()
```

funkcję co określoną liczbę milisekund, wiele razy.

```
function dot()
{
  console.log('.');
}
```

```
}
setTimeout(dot,2000);    //po 2 sekundach na konsoli pojawi się kropka
```

```
function dot()
{
console.log('.');
}
var funId=setTimeout(dot,2000);
clearTimeout(funId);           //zanim upłynie 2 s, odwołujemy funkcję,
var id=setInterval(dot, 2000); //po upływie każdych 2 s, wykona się funkcja
clearInterval(id);             //od tego momentu funkcja przestanie się wykonywać
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html lang="pl">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta http-equiv="Content-Script-Type" content="text/javascript">
  <title>Moja strona WWW</title>
  <script type="text/javascript">
    function js_zmienStrone()
    {
      window.location.replace("http://helion.pl/ksiazki/jscpk.htm");
    }
    setTimeout("js_zmienStrone();", 5000);
  </script>
</head>
<body>
  <p>Witryna została przeniesiona pod nowy adres:<br />
    http://helion.pl/ksiazki/jscpk.htm<br />
    Za 5 sekund zostaniesz przeniesiony do nowej lokalizacji.
  </p>
</body>
</html>
```

=====

#### Zadanie 40.

Opisz teorię dotyczącą Obiekt history.

#### Obiekt history

1. Zawiera historię stron odwiedzanych w bieżącej sesji.
2. Posiada dwie właściwości:
  - current → zawiera bieżący lokalizator URL,
  - length → zawiera długość listy historii.

#### window.history→wszystkie możliwości

**window.history.length** → możesz sprawdzić ile stron odwiedził użytkownik zanim wszedł na Twoją, ze względów bezpieczeństwa nie da się jednak pobrać ich adresów.

**window.history.forward()** → Działa to jak przyciski "w przód" w przeglądarce.  
**window.history.back()** → Działa to jak przyciski "w tył" w przeglądarce.  
**window.history.go(n)** → pozwala przejść o kilka stron do przodu lub do tyłu naraz.  
**window.history.go(-2);** → przejdzie dwie strony wstecz  
**window.history.go(0);** → odświeży aktualną stronę

=====

#### **Zadanie 41.**

##### **Zadanie na history**

#### **Zadanie 42.**

##### **Zadanie teoretyczne na przyszły rok screen, close, windowo itp**

###### **Obiekt screen**

window.screen

**window.screen.colorDepth** → odpowiada za głębię kolorów.  
**window.screen.width** → szerokość ekranu,  
**window.screen.height** → wysokość ekranu,  
**window.screen.availWidth** → tylko dostępna szerokość ekranu, to znaczy odejmuje wszelkie systemowe paski menu,  
**window.screen.availHeight** → analogicznie jak wyżej,

=====

###### **Metoda .open()**

window.open() → to metoda, która pozwala na otworezenie nowego okna.

Jej parametrami są:

**URL** – adres strony, która ma zostać załadowana w nowym oknie,

**name** – określa sposób otwarcia nowego okna,

**specs** – lista wartości rozdzielonych przecinkami, które dotyczą takich właściwości okna jak możliwość zmiany rozmiaru, wysokość, szerokość, ...,

**replace** – parametr, który mówi czy nowe okno ma dodać nowy wpis do historii przeglądarki czy zastąpić aktualny.

###### **Metoda .close()**

**window.close()** → służy do zamknięcia aktualnego okna.

###### **Metoda → window.moveTo(), window.moveBy(), window.resizeTo()**

**window.moveTo()** → metody służą do zmiany położenia okna przeglądarki na pulpicie. Przyjmują parametry określające pozycję w poziomie i pionie

**window.moveBy()** → metody służą do zmiany położenia okna przeglądarki na pulpicie. Przyjmują parametry określające pozycję w poziomie i pionie

**window.resizeTo()** → Trzecia metoda zmienia rozmiar okna przeglądarki. Przyjmuje nową szerokość i wysokość.

#### **Zadanie 43.**

- **Zadanie praktyczne na przyszły rok screen, close, windowo itp**

=====

Ciekawostka

/\*jeśli nie jesteśmy pewni czy dana funkcjonalność jest w danej przeglądarce, robimy tak:\*/

```
if( typeof window.addEventListener==='function' ){  
  //kod jeśli funkcjonalność JEST wspierana  
}else{  
  //kod jeśli funkcjonalność NIE jest wspierana  
}
```

console.log(answer); → gdzieś to umieścić

Ten obiekt odnosi się do aktualnie załadowanego dokumentu czyli strony WWW. Wszystkie jego pola i metody są związane z modelem DOM. Zostaną omówione w kolejnej lekcji.

#### **Zadanie 44.**

##### Wykonaj:

1. Zapisz w „szkielecie”, co to jest model DOM.
2. Na podstawie przykładu schematu/drzewa (umieszczonego poniżej ) narysuj „szkielecie”schemat DOM dla kodu ( musisz wybrać odpowiedni numer schematu/kodu).  
Metoda rysownia schematu/drzewa dowolna.

Wykonujesz układ jak układach zademonstrowanych poniżej.

Obliczenie	Który kod
(Numer_w_grupie) mod 4=1	kod1
(Numer_w_grupie) mod 4=2	kod2
(Numer_w_grupie) mod 4=0	kod3
(Numer_w_grupie) mod 4=3	kod4

#### **Model DOM**

DOM (Document Object Model) to sposób opisu złożonych dokumentów XML i HTML w postaci hierarchii modeli (niektóre obiekty są nadrzędne nad innymi, cała strona tworzy drzewo obiektów) modelu obiektowego.

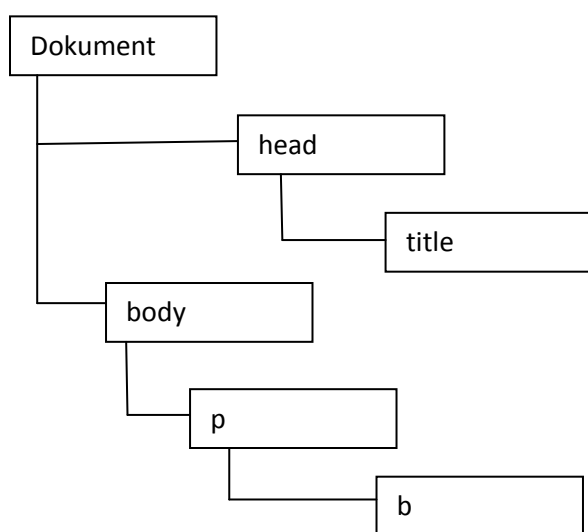
Mechanizm działania DOM →Kod HTML zamieniany jest przez przeglądarkę na stronę WWW.

Przeglądarka przechowuje informację o stronie WWW w postaci modelu obiektowego strony. DOM ma hierarchię obiektów, udostępniane są metody oraz właściwości obiektów.

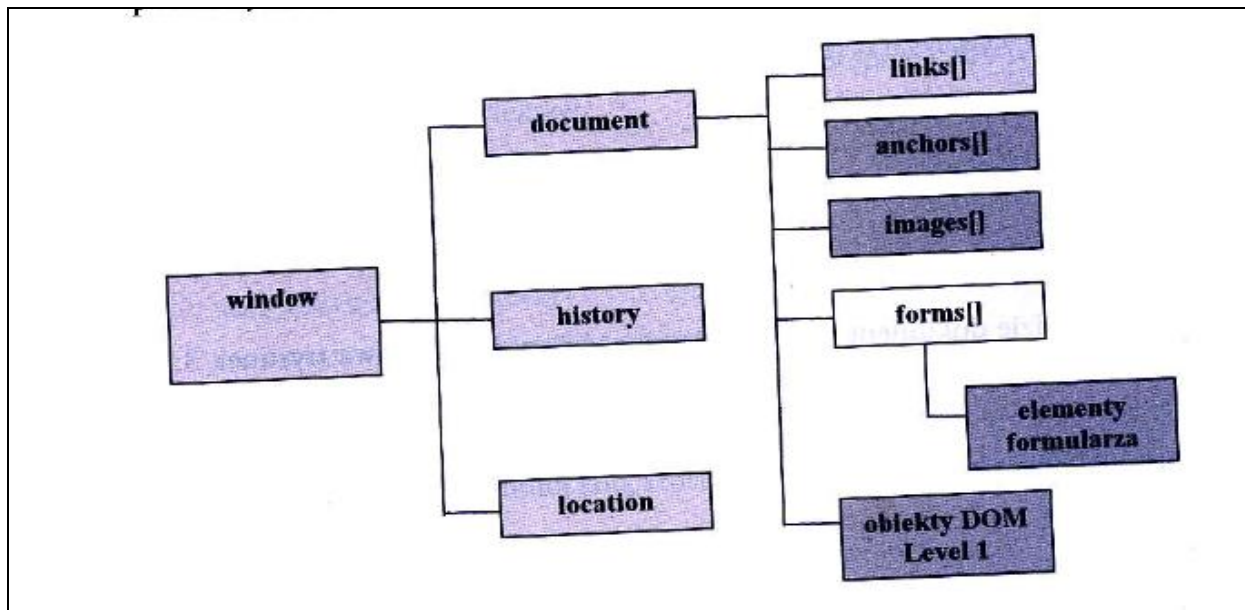
Przykład kodu + schemat/drzewo DOM

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Obiekty</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <p>Strona DOM<b>pokaz</b></p>
  </body>
</html>
```

Drzewo/Hierarchia obiektów z **kodu** poprzedniego (przykład)



Wycinek hierarchii DOM → (inny przykład) oraz sposób odwołania do obiektu links1.



W celu odwołania się do obiektu należy podać nazwy obiektów w hierarchii oddzielonych kropkami.

np. `window.document.link1`

#### Zadanie 45.

Będzie strona i trzeba narysować schemat hierarcji Dom

#### Zadanie 46.

Teoria

#### Dodatkowe elementy DOM:

- form — formularz,
- anchor — zakotwiczenie,
- link — odsyłacz,
- image — obrazek,
- embed — dodatek,
- applet — aplet Javy,
- frame — ramka,
- area — mapa graficzna.

#### Predefiniowane obiekty w JavaScript

- String — łańcuch tekstowy. Posiada własność **length** i metody: **slice()**, **split()**.
- Array — tablica. Posiada własność **length** i metody: **concat()**, **pop()**, **push()**.
- Date — data. Posiada metody: **getMonth()**, **getDay()**.
- Match — obiekt matematyczny.



## Zadanie 47.

### Zadanie teoretyczne

#### Dodatkowe elementy DOM:

- form — formularz,
- anchor — zakotwiczenie,
- link — odsyłacz,
- image — obrazek,
- embed — dodatek,
- applet — aplet Javy,
- frame — ramka,
- area — mapa graficzna.

## Obiekt String

### Zadanie 48.

Zadanie na praktyczne na

- String — łańcuch tekstowy. Posiada własność **length** i metody: **slice()**, **split()**.
- **Obiekt String**
- Obiektem zawsze występującym w języku JavaScript jest obiekt String. Posiada on jedną właściwość,
- length, określającą długość łańcucha.
- 
- Przykład
- 

```
tekst = "Obiekty języka JavaScript"  
dl = tekst.length
```

- 
- Zmiennej **dl** przypisana zostanie wartość 25, określająca długość tekstu.
- 
- Obiekt **String** posiada dwa typy metod.
- 
- Typ1
- **metoda substring()**, która zwraca podzbiór tego łańcucha. Jej parametry określają położenie początku i końca podzbioru.
- 
- Przykład

```
x = tekst.substring (15,19)
```

- 
- Zostanie zwrócony łańcuch Java.
- 
- Typ2
- Metoda **toUpperCase()** zamienia wszystkie litery ciągu **na duże**,
- Metoda **toLowerCase()** zamienia wszystkie litery ciągu **na małe**.
- 
- 
- Przykład

```
tekst = "Obiekty języka JavaScript"  
x = tekst.toLowerCase()
```

```
/*Zostanie zwrócony łańcuch obiekty języka javascript */
```

- 
- Przykład

```
tekst = "Obiekty języka JavaScript"
```

```
x = tekst.toUpperCase()
```

```
/*Zostanie zwrócony łańcuch OBIEKTY JĘZYKA JAVASCRIPT*/
```

- 
- Przykład
- Temat: Jak wykonać aby pierwsza litera ciągu znaków była pisana dużą literą.
- 
- Wykonanie:
- Do istniejących obiektów można dodawać nowe właściwości i można definiować dla nich nowe metody. Do obiektu String dodamy dodatkową funkcjonalność, dzięki której pierwsza litera
- łańcucha będzie pisana dużą literą.

```
String.prototype.duzaLitera = function() { return this.charAt(0).toUpperCase() + this.substr(1);}
```

- *Opis działania zdefiniowanej powyżej metody*
- Metoda **charAt()** zwraca znak z pierwszej pozycji łańcucha znaków. Metoda **substr()** zwraca podzbiór łańcucha znaków. Jako parametr tej metody został podany indeks pierwszego znaku podzbioru. Zadeklarowana metoda została dołączona do obiektu **String** i od tej pory każdy nowy tekst będzie posiadał metodę, która zamieni jego pierwszą literę na dużą.
- 
- Przykład
- Temat: Zamiana pierwszej litery na dużą.

```
var tx1 = 'komputer';
```

```
document.write(tx1.duzaLitera())
```

- W wyniku wykonania skryptu wyświetli się tekst 'Komputer'.
- 
- 
- **Obiekt Date**
- Obiekt specjalny Date, który służy do przechowywania wartości daty i czasu. Przy jego pomocy można odczytać wartość daty i czasu, można też rozłożyć te wartości na części, odczytując oddzielnie dzień, miesiąc, rok itp. Można również te części niezależnie modyfikować.
- 
- Przykład
- Temat: Odczytanie bieżącej datę i czas do zmiennej **dat\_cz**.
- 

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

- 
- Przykład
- Temat:Utworzenie obiektu, który będzie zawierał ściśle określoną wartość daty i godziny.

```
var data = new Date(2018,2,27);
```

- Opis:
- Można utworzyć obiekt z określoną liczbą parametrów. Tych parametrów może być od dwóch do siedmiu (**rok, miesiąc, dzień, godzina, minuty, sekundy, milisekundy**).
- W języku JavaScript wartości daty i czasu są przechowywane w formacie **timestamp**, czyli jako liczba milisekund, które upłynęły od północy 1 stycznia 1970 roku.
- 
-

- Do konwersji obiektu Date na tekst służy kilka funkcji:
- **toString()** — zwraca datę, czas oraz informacje o strefie czasowej w języku angielskim,
- **toLocaleString()** — zwraca datę i czas dla bieżących ustawień regionalnych,
- **toUTCString()** — zwraca datę, czas oraz informacje o strefie czasowej dla formatu UTC (Universal Coordinated Time),
- **toGMTString()** — działa jak funkcja toUTCString(),
- **toDateString()** — zwraca tylko datę w języku angielskim,
- **toLocaleDateString()** — zwraca tylko datę dla bieżących ustawień regionalnych,
- **toTimeString()** — zwraca tylko czas w języku angielskim,
- **toLocaleTimeString()** — zwraca tylko czas dla bieżących ustawień regionalnych.
- 
- UWAGA
- Dla metod **toString()** i **toLocaleString()** różne przeglądarki zwracają wyniki w różny sposób.

## porównywanie tekstów ułożyć zadanie

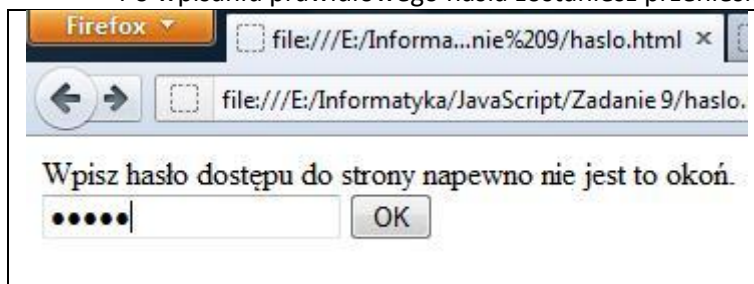
### Hasła

#### Zadanie 49.

Temat: [Hasło na stronę wersja 1.](#)

Wykonaj:

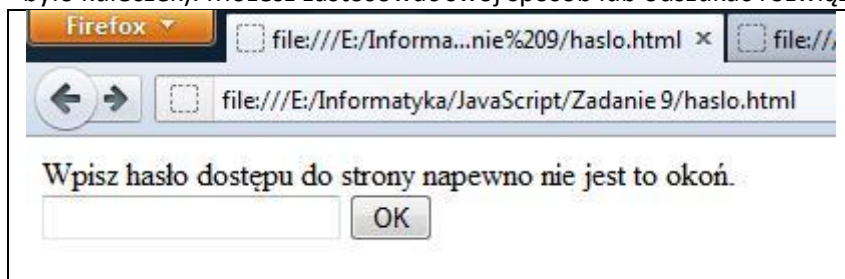
- Posługując się opisem dotyczącym wykonanie hasła do strony zabezpiecz hasłem stronę z zadania powyżej (kliknij link zawarty w tytule).
- Wykonaj stronę na, której będzie tylko pytanie o hasło w celu uzyskania dostępu do strony.
- Po wpisaniu prawidłowego hasła zostaniesz przeniesiony do strony



#### Zadanie 50.

Temat: Hasło na stronę wersja 1 → modyfikacja.

Zmień zadanie poprzednie tak aby przy ponownym logowaniu nie było wpisanego hasła (czyli nie było kuleczek). Możesz zastosować swój sposób lub odszukać rozwiązanie w instrukcji.

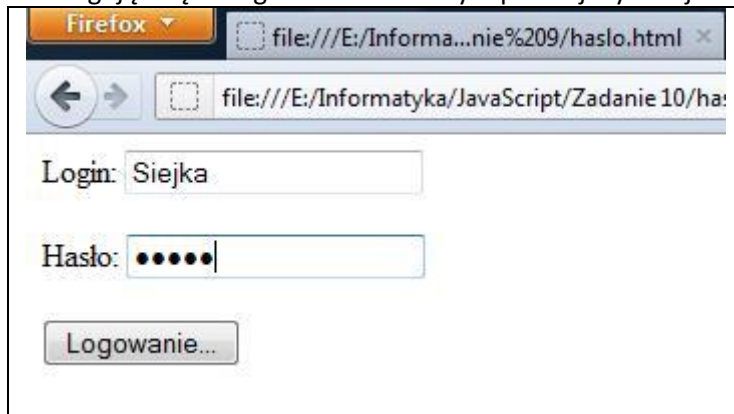


### Zadanie 51.

Temat: Hasło na stronę wersja 2

Wykonaj:

Posługując się listingiem umieszczonym poniżej wykonaj zabezpieczenie strony z zadania8.



Zmień:

login → na Twoje nazwisko

hasło → na Twoje imię

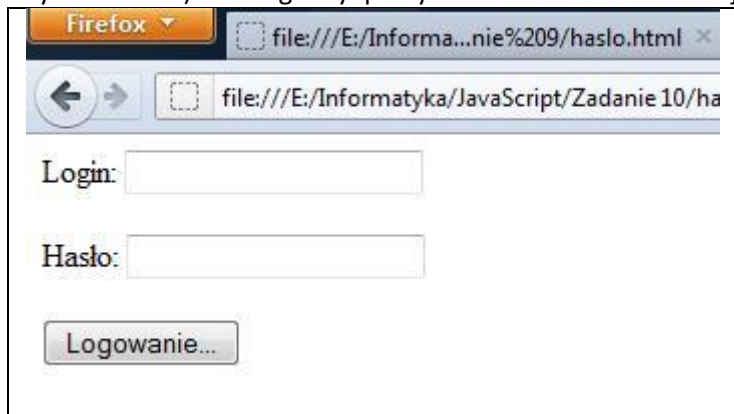
parametr → funkcji Twoje inicjały (bez polskich liter)

```
<html>
<head>
<script type="text/javascript">
function logowanie(dane)
{
    if(dane.login.value=="k"&dane.haslo.value=="k")
    {
        window.location.replace("obliczenia.html");
    }
    else
    {
        window.location.replace("");
    }
}
</script>
</head>
<body>
<form name="formularz">
    Login: <input type="text" name="login"/><br><br>
    Hasło: <input type="password" name="haslo"/><br><br>
    <input type="button" value="Logowanie..." onClick="logowanie(this.form)"/><br><br>
</form>
</body>
</html>
```

### **Zadanie 52.**

Temat: Hasło na stronę wersja 2→modyfikacja.

Zmień zadanie poprzednie tak aby przy ponownym logowaniu nie było wpisanego hasła (czyli nie było kuleczek) oraz login był pusty. Możesz zastosować swój sposób lub wzorować się na zadaniu 9a.



### **Zadanie 53.**

Temat: Hasło na stronę wersja 3

Zmień:

Zmień tak zadanie11 aby nieistotne było login i hasło pisane małymi czy dużymi literami (całość lub część) np. KOwaLski było prawidłowe.

Skomplikowana funkcja sprawdzająca hasło

```
<script language='JavaScript' type='text/JavaScript'>

function weryfikacja_hasla()
{
var prob = 1;
var haslo = prompt('Podaj hasło','');
if (haslo != null)
{
while (prob < 3)
{
if (!haslo) history.go(-1);
//Metoda toLowerCase zwraca wartość łańcucha znaków skonwertowanego na
małe litery
if (haslo.toLowerCase() == 'asdf')
{
alert('Hasło OK!');
window.open('skrypt1.html','_top');
break;
}
prob+=1;
var haslo = prompt('Złe hasło. Spróbuj ponownie','Hasło');
}
}
if (haslo !=null) {
if (haslo.toLowerCase()!='Hasło' + prob ==3) history.go(-1);return ' ' ;}}
</script>
```

=====

### Zadanie 54.

Zadanie na praktyczne na ( może z obliczeń przenieść na tablice)

- Array— tablica. Posiada własność **length** i metody: **concat()**, **pop()**, **push()**.

=====

### Zadanie 55.

Zadanie na praktyczne na ( może z obliczeń przenieść na tablice)

- Date— data. Posiada metody: **getMonth()**, **getDay()**.

=====

### Tworzenie obiektów

Aby utworzyć nowy obiekt w języku JavaScript, należy skorzystać z konstrukcji (funkcji)i, która zdefiniuje:

- nazwę obiektu
- utworzy właściwości
- utworzy metody.

### Obiekty wbudowane JavaScript

Przykład

Temat: Wykorzystania JavaScript na stronie WWW do wyświetlania daty i czasu jako elementu strony internetowej.

```
<html>
<head>
<title>JavaScript - Data i czas</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<body>
<h2>Wyświetlam bieżącą datę i czas</h2>
<p>
<script type="text/javascript">
data_n = new Date();
data_l = data_n.toString();
data_u = data_n.toGMTString();
data_r = data_n.toLocaleString();
document.write("<b>Czas lokalny:</b> " + data_l + "<br>");
document.write("<b>Czas uniwersalny:</b> " + data_u + "<br>");
document.write("<b>Czas regionalny:</b> " + data_r + "<br>");
</script>
</p>
</body>
</html>
```

Opis działania przykładu powyżej:

Utworzonej zmiennej o nazwie `data_n` przypisany został obiekt `Date`. Zmienne `data_l`, `data_u` i `data_r` zawierają tekstową postać czasu lokalnego, uniwersalnego i regionalnego, odpowiadającego wartości przechowywanej w zmiennej `data_n`. Do wyświetlenia wartości otrzymanych zmiennych została użyta funkcja `document.write`.

### Zadanie 56.

Zdefiniuj funkcję, która będzie wyświetlała datę i czas po polsku

## Zadanie 57.

Utwórz stronę internetową, na której data i czas będą wyświetlane w postaci kartki z kalendarza.

### Obiekt Array

W języku JavaScript do pracy z tabelami można używać wbudowanego obiektu Array. Posiada on metody do manipulowania tablicami zmiennych.

Aby utworzyć nową tablicę, należy zadeklarować obiekt Array w postaci:

```
var NazwaTablicy = new Array()  
lub  
var NazwaTablicy = []
```

Jeżeli w nawiasach[] zostanie podana liczba n, to zostanie utworzona tablica zawierająca n pustych elementów.

Przykład

```
var Tab1 = new Array(10)  
var Tab2 = [15]
```

Przykład

Tablicę można również tworzyć, wstawiając do niej konkretne wartości.

```
var Tab3 = new Array('Anna', 'Adam', 'Piotr', 'Ewa')  
var Tab4 = ['Paweł', 'Marcin', 'Ela']
```

Żeby uzyskać **dostęp do elementów tablicy**, należy podać numer indeksu danego elementu. Elementy są **indeksowane od zera**.

Przykład

```
document.write(Tab3[2])
```

W wyniku wyświetli się wartość 'Piotr'.

Przykład

Aby **dodać nową wartość** do tablicy, należy przypisać tę wartość do odpowiedniego indeksu tablicy.

```
var Tab5 = new Array('kot', 'pies', 'koń')  
Tab5[3] = 'mysz';  
Tab5[4] = 'chomik';  
document.write(Tab5[0] + 'i' + Tab5[3])
```

W wyniku wyświetli się tekst 'kot i mysz'.

Dzięki właściwości **length** można określić, z ilu elementów składa się tablica. Jest to bardzo przydatna właściwość, szczególnie gdy chcemy utworzyć pętlę odczytującą wszystkie elementy tablicy.

Przykład

W wyniku zostaną wyświetlone po kolei wszystkie elementy tablicy.

```
var Tablica_N = new Array('Anna', 'Adam', 'Piotr', 'Ewa', 'Paweł', 'Marcin', 'Ela')
```

```
for (i = 0; i<tablica_N.length; i++)  
{  
  document.write(Tablica_N[i] + "<br>");  
}
```

### Zadanie 58.

Zmień zapis skryptu podanego w przykładzie powyżej, tak aby wyświetlone zostały elementy tabeli od ostatniego do pierwszego. Pamiętaj, że indeksowanie elementów rozpoczyna się od zera.

### Tablice wielowymiarowe

W języku JavaScript można również tworzyć tablice wielowymiarowe. Wtedy element tablicy jest opisywany za pomocą indeksu określającego jego położenie w wierszu i kolumnie.

Przykład

```
var Tablica_Z = [];  
Tablica_Z[0] = ['Anna', 'Nowak'];  
Tablica_Z[1] = ['Adam', 'Kowal'];  
Tablica_Z[2] = ['Piotr', 'Ogórek'];  
Tablica_Z[3] = ['Ewa', 'Lisowska'];  
document.write('imię: ' + Tablica_Z[0][0] + 'nazwisko: ' + Tablica_Z[0][1] + "<br>");  
document.write('imię: ' + Tablica_Z[1][0] + 'nazwisko: ' + Tablica_Z[1][1] + "<br>");  
document.write('imię: ' + Tablica_Z[2][0] + 'nazwisko: ' + Tablica_Z[2][1] + "<br>");  
document.write('imię: ' + Tablica_Z[3][0] + 'nazwisko: ' + Tablica_Z[3][1]);
```

### Łączenie elementów tablicy

Za pomocą metody **join()** można łączyć elementy tablicy w jeden tekst. W metodzie tej można opcjonalnie podać parametr, który określi znak oddzielający kolejne elementy tablicy. Jeżeli nie zostanie podana wartość tego parametru, domyślnym znakiem będzie przecinek.

Przykład

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');  
document.write(Tablica.join() + "<br>"); /*odzielający znak , (przecinek) */  
document.write(Tablica.join(" - ") + "<br>"); /*odzielający znak -(minus) */
```

### Odwracanie kolejności elementów tablicy

Za pomocą metody **reverse()** można odwrócić kolejność elementów tablicy.

Przykład

```
var Tablica = new Array('Anna', 'Adam', 'Piotr');  
document.write(Tablica.join() + "<br>");  
Tablica.reverse()  
document.write(Tablica.join() + "<br>");
```

### Sortowanie

Do sortowania elementów tablicy służy metoda **sort()**.

Przykład

```
var Tablica = new Array('Paweł', 'Anna', 'Maria', 'Adam', 'Piotr');  
Tablica.sort()
```



```
document.write(Tablica.join());
```

### Zadanie 59.

łączenie, odwracanie sortowani → napisać zadanie

Domyślnie tablica jest sortowana leksykograficznie. Powoduje to, że liczba 12459 będzie mniejsza od 4567, ponieważ cyfra na pierwszej pozycji jest mniejsza. Aby temu zaradzić, można sortować tablicę według własnych kryteriów. Należy skorzystać z dodatkowego parametru metody `sort()`. Parametrem będzie własna funkcja sortująca. Tworząc taką funkcję, należy pamiętać o trzech zasadach, które muszą być spełnione:

- 1) jeżeli funkcja(a,b) zwróci wartość mniejszą od 0, to wartości a zostanie nadany indeks mniejszy od indeksu przyznanego wartości b,
- 2) jeżeli funkcja(a,b) zwróci wartość równą 0, to wartości indeksów pozostaną bez zmian,
- 3) jeżeli funkcja(a,b) zwróci wartość większą od 0, to wartości a zostanie nadany indeks większy od indeksu przyznanego wartości b.

Stosując się do tych zasad, można tworzyć własne metody sortowania w tablicy.

Przykład

```
function porownaj(a,b)
{
  return a - b
}
var Tablica = new Array(27, 100, 10, 450, 1654, 320);
document.write('Bez sortowania: ' + Tablica.join() );
document.write('Sortowanie domyślne: ');
Tablica.sort()
document.write(Tablica.join());
document.write('Sortowanie poprawne: ');
Tablica.sort(porownaj)
document.write(Tablica.join() );
```

W podanym przykładzie została zdefiniowana funkcja `porownaj(a,b)`, która zwróci wartość mniejszą od zera, równą zero lub większą od zera. W zależności od zwróconej wartości elementy tablicy zostaną uporządkowane od wartości najmniejszej do największej.

### Zadanie 60.

Utwórz tablicę, która będzie zawierała elementy zawierające cyfry i litery. Zdefiniuj funkcję, która pozwoli uporządkować elementy tabeli w ten sposób, że najpierw będą umieszczone litery w kolejności alfabetycznej, a następnie cyfry w kolejności od wartości najmniejszej do największej.

→sprawdzić

### Zadanie 61.

Utwórz tabelę o wymiarze „trzy na trzy”.

```
var tablica = new Array(1, 2, 3);
document.write(tablica[1] + "<br>");
tablica[1] = 123;
document.write(tablica[1] + "<br>");
```

Umieść w niej obrazki. Zdefiniuj kod, który spowoduje zmianę wyświetlanych na stronie obrazków co 2 sekundy.

→sprawdzić, rozwiązać

## Dział 4 Instrukcje warunkowe Pętle

### Zadanie 62. (Przykład 3a)

Temat: Użycie instrukcji if dla określenia znaku liczby wczytanej z klawiatury.

```
<html>
<body>
  <div style="color:red;font-size:50px;">
    <script language="JavaScript">
      var x=prompt("podaj liczbę x=", "");
      var a=parseFloat(x);
      document.write("czytana liczba x="+a+"<br>");
    if(a>0)
      {document.write("czytana liczba jest większa od 0"+"<br>");}
    // Dokończ dwa pozostałe przypadki a==0 i a<0
    // polskie litery mają działać(uzupełnij w kodzie strony)
    </script>
  </div>
</body>
</html>
```

### Zadanie 63.(Zadanie 3a)

Wczytać kolor wyświetlania tekstu.

Jeżeli:

wczytamy liczbę 1 **lub** słowo czerwony to tekst uzyskamy tekst *"wczytałeś kolor czerwony"* wyświetli się na czerwono czcionką wielkość o numerze 7 (size)

wczytamy liczbę 2 **lub** słowo niebieski to tekst uzyskamy tekst *"wczytałeś kolor niebieski"* wyświetli się na niebiesko wielkość o numerze 1 (size)

Wielkość liter wczytywanych danych nie będzie miało znaczenia. Użyj metody x.toLowerCase() [ dla zmiennej x ]

Poniżej są umieszczone dwie możliwości wczytywania danych:

przykład1 → przycisk i do niego podłączona funkcja,

przykład2 → użycie formularza.

Wykonaj menu:

MENU:

1-CZERWONY

2-NIEBIESKI

(napis kolorem zielonym)

(napis kolorem czerwonym)

(napis kolorem niebieskim)

Przykład 1	Przykład 2
<p><b>MENU</b></p> <p><b>1-CZERWONY</b></p> <p><b>2-NIEBIESKI</b></p> <p>Oblicz</p>	<p>Menu</p> <p>1 - Czerwony</p> <p>2 - Niebieski</p> <p><input type="text"/></p> <p>Sprawdź</p>

Użycie instrukcji document.write do wykonania instrukcji HTML na przykładzie formatowania tekstu

```
document.write("<font color=silver size=4>wpisałeś kolor czerwony</font>"+"<br>");
```

Użycie instrukcji document.write do wykonania instrukcji CSS na przykładzie formatowania tekstu

Uwaga:

Konieczne jest użycie \" ponieważ zastosowanie samego znaku " bez \ spowoduje, że interpreter Javascript będzie chciał zakończyć instrukcję document.write

```
document.write ("<p style=\"font-size: 40.px; color: black;\"><br>Dziki melanz<br></p>");
```

#### **Zadanie 64. (Zadanie 3a1)**

Wczytać liczbę i stwierdzić czy jest parzysta czy nieparzysta:

Gdy jest parzysta to: czerwony napis „*liczba jest parzysta*” wielkością 3 (size)

Gdy jest nieparzysta to: srebrny napis „*liczba jest nieparzysta*” wielkością 5 (size)

#### **Zadanie 65. (Zadanie 3a2)**

Użyj instrukcji if.... else ..... do sprawdzenia czy użytkownik wie od, którego roku Javascript jest w użyciu.

Wczytać rok do zmiennej rok\_kow i są dwie możliwości:

Brawo znasz rok od kiedy działa Javascript (niebieski, wielkość 4)

Źle musisz jeszcze poczytać (czerwony, wielkość 7)

Wyświetli się również napis „wczytany rok=.....”

### **Zadanie 66.(Zadanie 3b)**

Temat: Użycie pętli **for** do wypisywania Twojego nazwiska 10 razy.

```
<html>
<body>
    <div style="color:red;font-size:50px;">
        <script language="JavaScript">
            for(i=1;i<=10;i++)
            {
                document.write("wyświetlam "+i+" raz"+" tutaj wpisz Twoje nazwisko"+"<br>");
            }
        </script>
    </div>
</body>
</html>
```

Zapewnij wyświetlanie polskich liter w standardzie Utf8

### **Zadanie 67.(Zadanie 3c1)**

Temat: Użycie jednej pętli **for** do wypisywania:

Wyświetlaj Twoje **nazwisko** lub **imię** na przemian.

Gdy wyświetlasz nieparzyste raz to wyświetli się Twoje **nazwisko**

Gdy wyświetlasz parzyste raz to wyświetli się Twoje **imie**

Ilość wyświetleń (imienia lub nazwiska) to:

Ile\_razy\_wyświetlić=[( liczba\_liter\_nazwiska) mod 4]+10 (oblicz w pamięci i wstaw do pętli)

Dla określenia czy liczba jest parzysta czy nie użyj if.... else..... oraz reszty z dzielenia (%)

Wielkość liter:

Wielkość\_liter=[( liczba\_liter\_nazwiska) mod 3]+40px (oblicz w pamięci i wstaw do programu)

Zapewnij wyświetlanie polskich liter w standardzie Utf8

### **Zadanie 68.(Zadanie 3c2)**

Temat: Użycie jednej pętli **for** do wypisywania twojego nazwiska lub imienia.

Wczytujemy jedną zmienną w okienku.

I gdy:

→ wczytana dana jest parzysta. Wypisujemy Twoje **nazwisko** ( ile razy patrz poniżej ) dla danej wczytanej w okienku. Kolor wyświetlania to niebieski.

→ wczytana dana jest nieparzysta. Twoje **imie** ( ile razy patrz poniżej ) dla danej wczytanej w okienku Kolor wyświetlania to zielony.

Ilość wyświetleń (imienia lub nazwiska) to:

Ile\_razy\_wyświetlić=[( liczba\_liter\_nazwiska) mod 4]+10 (oblicz w pamięci i wstaw do pętli)

Dla określenia czy liczba jest parzysta czy nie użyj if.... else..... oraz reszty z dzielenia (%)

Wielkość liter:

Wielkość\_liter=[( liczba\_liter\_nazwiska) mod 3]+40px (oblicz w pamięci i wstaw do programu)

Zapewnij wyświetlanie polskich liter w standardzie Utf8

### **Zadanie 69.(Zadanie 3d1)**

Temat: Użycie pętli **for** do rysowania w trybie tekstowym

Z użyciem 3 trzech pętli narysuj prostokąt z gwiazdek o zadanych ( obliczonych) wymiarach a i b.

```

          b
*****
*               *
*               *           a
*               *
*****
```

Pierwsza pętla to górne poziome gwiazdki, środkowe gwiazdki czyli pionowe gwiazdki to druga pętla, trzecia pętla to dolne poziome gwiazdki.

**Oblicz** a i b ze wzorów zapisanych poniżej:

$a = [( \text{liczba\_liter\_nazwiska}) \bmod 4] + 5$

$b = [( \text{liczba\_liter\_imienia}) \bmod 6] + 20$

Uwaga: Pierwsza osoba ( oraz zadania indywidualne), które rozwiążą zadania z użyciem będą punktowane +1 punkt.

### **Zadanie 70.(Zadanie 3d2 26)**

Temat: Użycie pętli **for** do rysowania w trybie tekstowym

Z użyciem 3 trzech pętli narysuj prostokąt z gwiazdek o zadanych ( obliczonych) wymiarach a i b.

```

          b
*****
*               *
*               *           a
*               *
*****
```

Pierwsza pętla to górne poziome gwiazdki, środkowe gwiazdki czyli pionowe gwiazdki to druga pętla, trzecia pętla to dolne poziome gwiazdki.

Program będzie pytał się o **a** i **b** i następnie rysował.

Uwaga: Pierwsza osoba ( oraz zadania indywidualne), które rozwiążą zadania z użyciem będą punktowane +1 punkt.

### **Zadanie 71.(Zadanie 3d3)**

Temat: Użycie pętli **for** do rysowania w trybie tekstowym

Z użyciem 3 trzech pętli narysuj prostokąt z gwiazdek o zadanych ( obliczonych) wymiarach a i b.

```

          b
*****
*               *
*               *           a
*               *
*****
```

Pierwsza pętla to górne poziome gwiazdki, środkowe gwiazdki czyli pionowe gwiazdki to druga pętla, trzecia pętla to dolne poziome gwiazdki.

Program będzie pytał się o **imię i nazwisko** następnie **obliczał** ( z użyciem odpowiednich formuł)

**a** i **b** ze wzorów zapisanych poniżej:

$a = [( \text{liczba\_liter\_nazwiska}) \bmod 4] + 5$   
 $b = [( \text{liczba\_liter\_imienia}) \bmod 6] + 20$

Uwaga: Pierwsza osoba ( oraz zadania indywidualne), które rozwiążą zadania z użyciem będą punktowane +1 punkt.

### **Zadanie 72.(Zadanie 3e 26)**

Temat: Użycie pętli **for** do zmiany wielkości tekstu

Program pyta się o:

- 1)dwie liczby z przedziału <20,60> wczytane do zmiennych o nazwach l1\_t\_kow, l2\_t\_kow, zmienne zawierają trzy litery nazwiska ucznia (bez polskich liter),
- 2)wartość początkową wielkości tekstu wczytaną do zmiennej o nazwie pocz\_kow, zmienna zawiera trzy litery nazwiska ucznia (bez polskich liter),
- 3)dowolny tekst, wczytany do zmiennej o nazwie tekst\_kow, zmienna zawiera trzy litery nazwiska ucznia (bez polskich liter),

Wczytany tekst zostanie wyświetlony określaną ilość razy:

- Dla zwiększania liter przerywamy wyświetlanie gdy tekst jest większy od 100px
- Dla zmniejszania liter przerywamy wyświetlanie gdy tekst jest mniejszy od 0px

Gdy zachodzi zależność  $\text{liczba1} < \text{liczba2}$  to tekst zwiększa się o 5 pikseli

Gdy zachodzi zależność  $\text{liczba2} < \text{liczba1}$  to tekst zmniejsza się o 5 pikseli

Program sprawdza wczytane dane i informuje o tym np. „liczba l1 nie należy do przedziału <20,60>”.

Wskazówki:

- 1) Ilość powtórzeń pętli trzeba obliczyć z wartości początkowej wielkości tekstu oraz uwzględnić czy następuje zwiększanie czy zmniejszanie tekstu,
- 2) Stosujemy następującą metodę użycia wartości zmiennych dla np. wyświetlania wielkości tekstu zdefiniowane w zmiennej.

<code>document.write("&lt;p style=\"font-size:\"" + i + "px; color: black;\""&gt;&lt;br&gt;" + tekst_kow + "&lt;br&gt;&lt;/p&gt;");</code>
--------------------------------------------------------------------------------------------------------------------------------------------

Wiekść w zmiennej → i

Tekst w zmiennej → tekst\_kow

### **Zadanie 73.(Zadanie 3f 27)**

Temat: Obliczanie liczb pierwszych podanych wzorem.

Istnieje wzór wielomianu o postaci:

$$w(i) = i^2 + i + 41$$

którego wartości dla  $i = 0, 1, 2, \dots, 39$  są liczbami pierwszymi Napisz program znajdujący czterdzieści liczb pierwszych z użyciem pętli, korzystając ze wzoru. Zatrzymanie ekranu po pięciu liniach.

Przykładowy wydruk zadania

i=0      w(0)=41

i=1      w(1)=43

..... // zatrzymanie po pięciu liniach  
Aby kontynuować, naciśnij dowolny klawisz → okno confirm

#### **Zadanie 74.(Zadanie 3g 28)**

Temat:Napisać program drukujący liczby, ich pierwiastki drugie oraz pierwiastki czwartego stopnia od numeru w dzienniku do numeru w dzienniku+15. Zatrzymanie ekranu po trzech liniach.

Wskazówki:

- Pierwiastek czwartego stopnia obliczamy jako pierwiastek z pierwiastka drugie stopnia.
- W celu uzyskania dwóch miejsc po przecinku zastosuj metodę .toFixed(2) np. x1. .toFixed(2)
- W celu uzyskania pierwiastka zastosuj metodę Math.sqrt(liczba)

Przykładowy wydruk zadania

l=4      pierw\_2(4)=1.41                      pierw\_4(4)=1.19  
l=5      pierw\_2(5)=2.34                      pierw\_4(5)=1.49

..... // zatrzymanie po trzech liniach  
Aby kontynuować, naciśnij dowolny klawisz → okno confirm

#### **Zadanie 75.(Zadanie 3h)**

Temat: Wypisz wszystkie liczby parzyste od 2 do liczby wczytanej w okienku.

#### **Zadanie 76.(Zadanie 3i)**

Temat: Wczytanej w osobnych okienkach dwie liczby. Wypisz wszystkie liczby nie parzyste zawarte między tymi liczbami. Gdy pierwsza liczba jest mniejsza od drugiej to liczby wypisywane są od najmniejszej do największej kolorom zielonym. Gdy pierwsza liczba jest większa od drugiej to liczby wypisywane są od największej do najmniejszej kolorom niebieskim czcionka dwa razy mniejsza niż czcionka w pierwszy przypadku. Gdy pierwsza liczba wczytana liczba jest równa drugiej to komunikat i brak obliczeń w okienku confirm.

#### **Zadanie 77.Zadanie 3? (25)**

Temat: Wyprowadzanie liczb od 1 do 10 przesuniętych o jedną spację w prawo

```
<html>
<head>
<title>
    Przykład petli
</title>
<script language="JavaScript">
    for ( var i=1; i<=10; i++)
    {
        for ( var k=1; k<=i; k++)
        {
            document.write("&nbsp;");
        }
    }
</script>
</html>
```

```

        document.write(i+"<br>");
    }
</script>
</head>
<body>

</body>
</html>

```

### **Zadanie 78.(Zadanie 29)**

Temat: Obliczanie ilości punktów kratowych z użyciem pętli podwójnej.

Punkt kratowy to punkt, którego współrzędne w układzie kartezjańskim są liczbami całkowitymi.

Przykłady punktów kratowych:

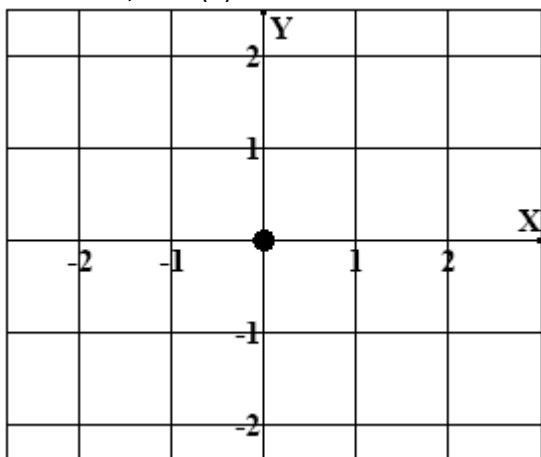
$(-100, 101)$ ,  $(1, 1)$ ,  $(0, 0)$ ,  $(-1, -3)$ .

Rozważamy koło o środku w początku układu współrzędnych. Dla nieujemnej liczby rzeczywistej  $R$  przez  $K(R)$  oznaczmy koło o promieniu  $R$  (brzeg koła należy do koła).

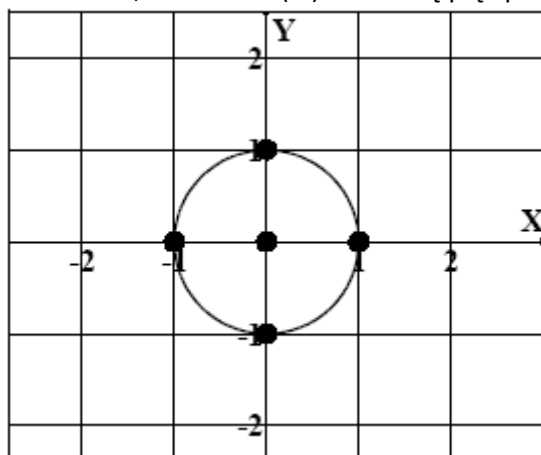
Niech  $N(R)$  będzie liczbą punktów kratowych zawartych w kole  $K(R)$ .

Przykłady:

Jeżeli  $R = 0$ , to  $N(R) = 1$ .

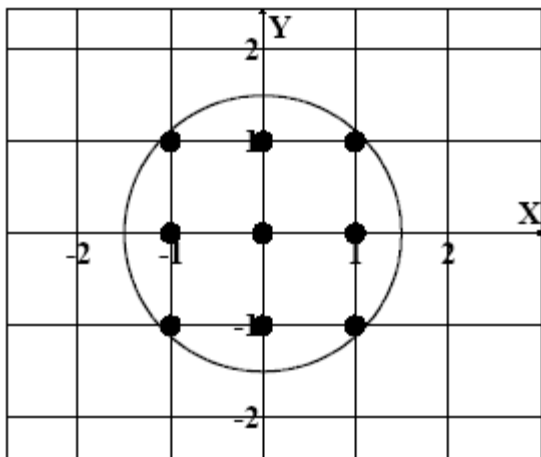


Jeżeli  $R = 1$ , to w kole  $K(R)$  mieści się pięć punktów kratowych, czyli  $N(R) = 5$ .



Jeżeli  $R = 1,5$ , to w kole  $K(R)$  mieści się dziewięć punktów kratowych, zatem  $N(R) = 9$ .





### Do wykonania

a) Zapisz specyfikację w zeszycie.

Specyfikacja:

**Dane:** R – promień koła o środku znajdującym się w początku układu współrzędnych (0,0);  
liczba całkowita nieujemna.

**Wynik:** liczba całkowita N(R) – liczba punktów kratowych zawierających się w kole o środku (0,0) i promieniu R

b) Napisz program obliczający liczbę punktów kratowych zawierających się w kole o promieniu R. Użyj zmiennych sterujących i\_trzy\_pierwsze\_litery\_nazwiska\_ucznia oraz j\_trzy\_pierwsze\_litery\_nazwiska\_ucznia np.i\_kow oraz j\_kow.

c) Uzupełnij i zapisz w zeszycie poniższą tabelę używając napisany program:

Promień koła R	Liczba punktów kratowych N(R)
2,01	
4,50	
100+numer_w_dzienniku	
1000+numer_w_dzienniku	

### Rozwiązanie:

Należy przeszukać obszar w postaci kwadratu w układzie XOY  $(-1-R) \leq x \leq (1+R)$

$(-1-R) \leq y \leq (1+R)$ . W tym celu należy stworzyć dwie pętle (jedna w drugiej, czyli pętla podwójna).

Pętla podwójna będzie generować współrzędne wszystkich punktów o zmiennych całkowitych wewnątrz tego kwadratu. Należy zwrócić uwagę, że zmienna R jest liczbą rzeczywistą a nie całkowitą dlatego przed użyciem w pętli należy zamienić ją na całkowitą. np.  $(\text{int})(-R\_kow-1)$ . Dla wszystkich wygenerowanych punktów sprawdź czy odległość wygenerowanego punktu od początku układu współrzędnych (punkt (0,0)) jest mniejsza lub równa od R. Jeśli tak to zwiększ licznik (na początku zadania musi być zerowany). Po wykonaniu zadania wyświetl licznik.

Jeśli wygenerowany punkt ma współrzędne  $(i\_kow, j\_kow)$  to warunek ma postać:

$$\sqrt{(i\_kow)^2 + (j\_kow)^2} \leq R\_kow$$

**Zadanie** → dla osób nie uczestniczących w zajęciach

Temat: Określanie punktów kratowych z użyciem pętli podwójnej.

### Wstęp teoretyczny

**Punkt kratowy** to punkt, którego współrzędne w układzie kartezjańskim są liczbami całkowitymi.

Przykłady punktów kratowych:

$(-100, 101)$ ,  $(1, 1)$ ,  $(0, 0)$ ,  $(-1, -3)$

### Wykonanie zadania:

Przeczytaj poprzednie zadanie. Podpowiedzi do jego rozwiązania mogą być pomocne przy rozwiązaniu tego zadania.

1) Wczytaj cztery liczby

$x\_MAX \rightarrow$  maksymalna wartość osi x

$x\_MIN \rightarrow$  minimalna wartość osi x

$y\_MAX \rightarrow$  maksymalna wartość osi y

$y\_MIN \rightarrow$  minimalna wartość osi y

2) Wypisać punkty kratowe

$x\_MAX = -1$

$x\_MIN = 2$

$y\_MAX = -2$

$y\_MIN = 1$

$(-1, -2)$   $(-1, -1)$   $(-1, 0)$   $(-1, 1)$

$(0, -2)$   $(0, -1)$   $(0, 0)$   $(0, 1)$

$(1, -2)$   $(1, -1)$   $(1, 0)$   $(1, 1)$

$(2, -2)$   $(2, -1)$   $(2, 0)$   $(2, 1)$

3) Oblicz maksymalną odległość punktu kratowego od początku układu współrzędnych (kilka punktów może mieć spełniać warunki zadania).

Jeśli wygenerowany punkt ma współrzędne  $(x, y)$  to wzór ma postać:

$$d = \sqrt{(x)^2 + (y)^2}$$

Dla danych z punktu 2) wydruk będzie następujący:

Punkt  $(2, -2)$   $d=2.83$

Dwa miejsca po przecinku

4) Istnieją dwie możliwe strategie obliczenia  $d$  (odległości).

- Maksymalna odległość znajduje się w punktach narożnych prostokąta obszaru punktów kratowych.
- Można obliczać odległość dla wszystkich punktów kratowych, znajdować  $d\_max$  (przy zapamiętanych aktualnych iteracyjnych „x” i „y”).

### **Zadanie 79. (Zadanie 30)**

Temat: Użycie obiektów-tablic do rozwiązania układu dwóch równań liniowych.

Wykonaj:

I.)Przeczytaj teorię o [tablice obiekty](#).

II.)Wykonaj praktycznie zadanie

Temat: Rozwiązać układ dwóch równań liniowych w postaci. Do rozwiązania użyj tablic i pętli. Zastosuj wyznaczniki.

### Przydatna teoria.

np.

$$A[0][0] * X + A[0][1] * Y = A[0][2]$$

$$A[1][0] * X + A[1][1] * Y = A[1][2]$$

$$\begin{cases} 2 * x + 3 * y = 5 \\ -4 * x + 8 * y = 0 \end{cases}$$

$$\text{i tak} \quad \begin{matrix} A[0][0]=2 & A[0][1]=3 & A[0][2]=5 \\ A[1][0]=-4 & A[1][1]=8 & A[1][2]=0 \end{matrix}$$

Tak więc w celu rozwiązania układu równań konieczne będzie użycie tablicy o dwóch wierszach i trzech kolumnach czyli konieczna będzie rezerwacja tablicy A[2][3].

W przypadku tablic dwuwymiarowych pierwsza liczba jest ilością wierszy, a druga ilością kolumn.

Wyznaczniki obliczamy stosując wzory:

$$W = \begin{vmatrix} A[0][0] & A[0][1] \\ A[1][0] & A[1][1] \end{vmatrix} = A[0][0] * A[1][1] - A[0][1] * A[1][0]$$

$$W_x = \begin{vmatrix} A[0][2] & A[0][1] \\ A[1][2] & A[1][1] \end{vmatrix} = A[0][2] * A[1][1] - A[1][2] * A[0][1]$$

$$W_y = \begin{vmatrix} A[0][0] & A[0][2] \\ A[1][0] & A[1][2] \end{vmatrix} = A[0][0] * A[1][2] - A[1][0] * A[0][2]$$

Rozwiązywalność układu równań (możliwe przypadki A) lub B) lub C):

A) gdy (**W<>0**) układ ma jedno rozwiązanie

gdzie: **X=W<sub>x</sub>/W** **Y=W<sub>y</sub>/W**

**W**—wyznacznik główny układu równań

**W<sub>x</sub>**—wyznacznik dla zmiennej **X**

**W<sub>y</sub>**—wyznacznik dla zmiennej **Y**

B) gdy (( **W=0** ) i (( **W<sub>x</sub><>0** ) lub ( **W<sub>y</sub><>0** ))) układ sprzeczny

C) gdy (( **W=0** ) i ( **W<sub>x</sub>=0** ) i ( **W<sub>y</sub>=0** )) układ ma nieskończenie wiele rozwiązań sposób obliczania wyznaczników

### **Wykonaj na stronie:**

1)Na stronie zapisz ile wynoszą elementy tablicy dla układu równań.

$$\begin{cases} nr\_z\_dziennika * x + miesiaki\_urodzenia * y = dzien\_urodzenia \\ (nr\_z\_dziennika - 40) * x + liczba\_liter\_nazwska * y = liczba\_liter\_imienia \end{cases}$$

czyli A[0][0]=..... wypisz wszystkie sześć elementów tak aby tworzyły układ równań, łącznie klamrą ( powiększ zwykłą klamrę).

2) Do operacji na tablicach użyj metody **.length**

Ilość wierszy uzyskuje się przez **tablica.length**

a liczbę kolumn w danym wierszu uzyskuje się przez **tablica[numerWiersza].length**.

```
for(int i = 0; i < tablica.length ; i++)
{
    for(int j = 0; j < tablica[i].length ; j++) //sprawdzić dlaczego nie działa
    {
        // tutaj dostępny jest element tablica[i][j]
    }
}
```

Wczytywanie danych

<input type="text"/>	* x +	<input type="text"/>	* y =	<input type="text"/>
<input type="text"/>	* x +	<input type="text"/>	* y =	<input type="text"/>
<input type="button" value="Oblicz"/>				

Obliczenia → dwa miejsca po przecinku metoda `.toFixed(2)` `+x.toFixed(2);`

Na poszczególnych ekranach ma wartości do testowania.

<input type="text" value="1"/>	* x +	<input type="text" value="2"/>	* y =	<input type="text" value="-1"/>
<input type="text" value="2"/>	* x +	<input type="text" value="3"/>	* y =	<input type="text" value="2"/>
<input type="button" value="Oblicz"/>				
Równanie ma jedno rozwiązanie:				
X = 7.00				
Y = -4.00				

<input type="text" value="1"/>	* x +	<input type="text" value="1"/>	* y =	<input type="text" value="1"/>
<input type="text" value="1"/>	* x +	<input type="text" value="1"/>	* y =	<input type="text" value="1"/>
<input type="button" value="Oblicz"/>				
Równanie ma nieskończenie wiele rozwiązań.				

<input type="text" value="1"/>	* x +	<input type="text" value="1"/>	* y =	<input type="text" value="1"/>
<input type="text" value="1"/>	* x +	<input type="text" value="1"/>	* y =	<input type="text" value="0"/>
<input type="button" value="Oblicz"/>				
Sprzeczne równanie				

3) Program pisz etapami :

#### ETAP 1

Zapewnić wczytywanie danych do tablicy :

Rozwiązanie Etapu 1:

1)Wczytywanie danych czyli współczynników układu równań z formularza

2)utwórz obiekt tablicy o nazwie A\_pierwsza\_litera\_nazwiska  
np.

```
float $A[][] = new float [?][?];
```

w miejsce znaków zapytania wpisz odpowiednie liczby zgodne z treścią zadania.

3)Wpisz danych do tablicy, pamiętaj o Twojej nazwie tablicy (innej niż w przykładzie poniżej.  
np.

```
$A[1][1]= dana z formularza
```

7)Wypisz dane z tablicy czytanej w podpunkcie 6)

## ETAP 2

Obliczyć Wx, Wy, W, X, Y oraz wydrukować Wx, Wy, W, X i Y z dwoma miejscami po przecinku.

np. String.format("%.2f",obwod)

Potrzebne zmienne zadeklaruj jako double.

## ETAP 3

Sprawdzenie warunków czy układ jest

- oznaczony
- sprzeczny
- ma nieskończenie wiele rozwiązań  
wraz z wyprowadzeniem stosownego komunikatu na monitorze

układy do testowania:

- oznaczony  
wpisz swoje dane z układu
$$\begin{cases} nr\_z\_dziennika * x + miesi\acute{a}i\_urodzenia * y = dzien\_urodzenia \\ (nr\_z\_dziennika - 40) * x + liczba\_liter\_nazwska * y = liczba\_liter\_imienia \end{cases}$$
- sprzeczny
$$\begin{cases} 1 * x + 2 * y = 2 \\ 1 * x + 2 * y = -5 \end{cases}$$
- ma nieskończenie wiele rozwiązań
$$\begin{cases} -3 * x + 4 * y = 2 \\ -3 * x + 4 * y = 2 \end{cases}$$

## ETAP 4

Zapewnienie powtarzalności obliczeń

Wypisanie na monitorze układu równań w postaci np.

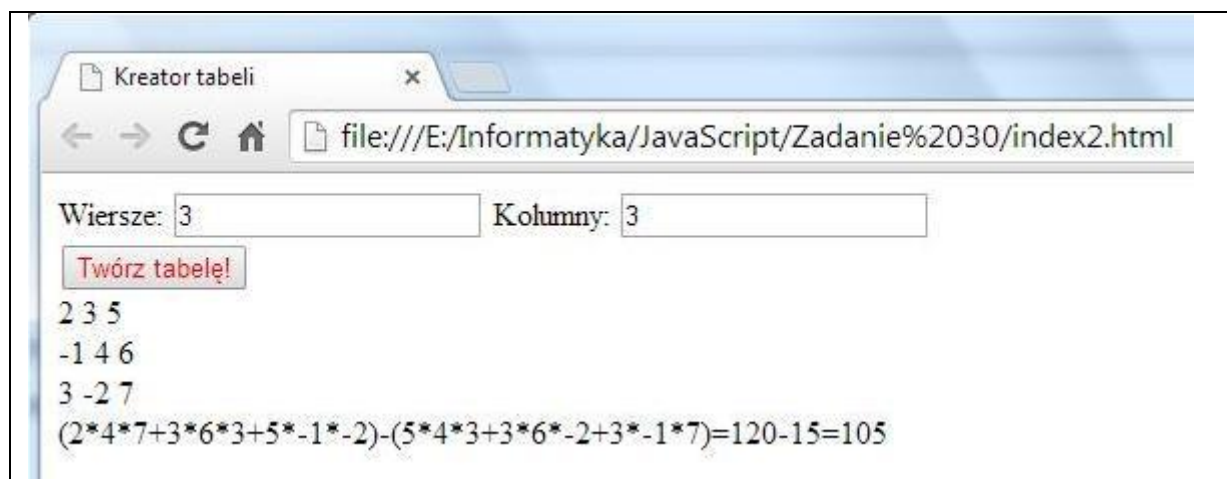
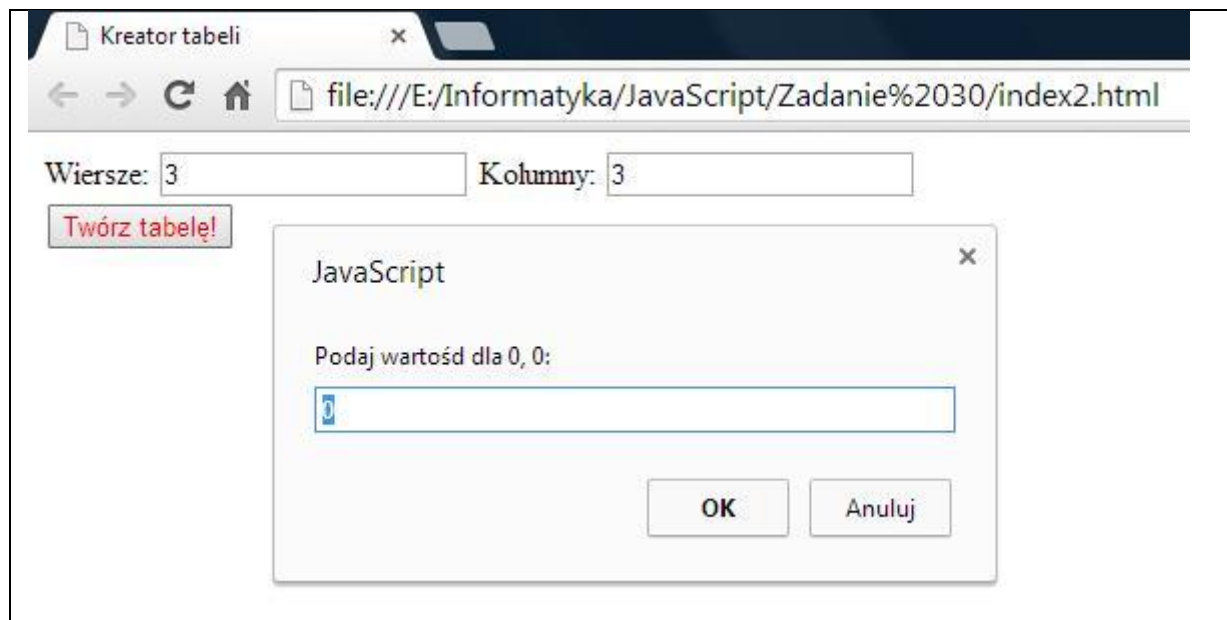
$$\begin{aligned} 2 * X + 3 * Y &= 5 \\ -4 * X + 8 * Y &= 0 \end{aligned}$$

Wykonaj specyfikację problemu.

### Zadanie 80.(ZADANIE 31)

Napisz program obliczający wyznacznik 3x3 metodą Sarussa. Wczytaj macierz oraz wydrukuj ją po wczytaniu wierszami oraz wartość wyznacznika.

Znajdź w Internecie jak obliczamy wyznacznik 3x3 metodą Sarussa.



## **Dział 6 Data i czas**

### **Zadanie 81. (Przykład 4)**

Temat: Strona wyświetlająca:

- ostatnią datę modyfikacji,
- komunikat Cześć na wciśnięcie klawisz,
- aktualną datę oraz czas.

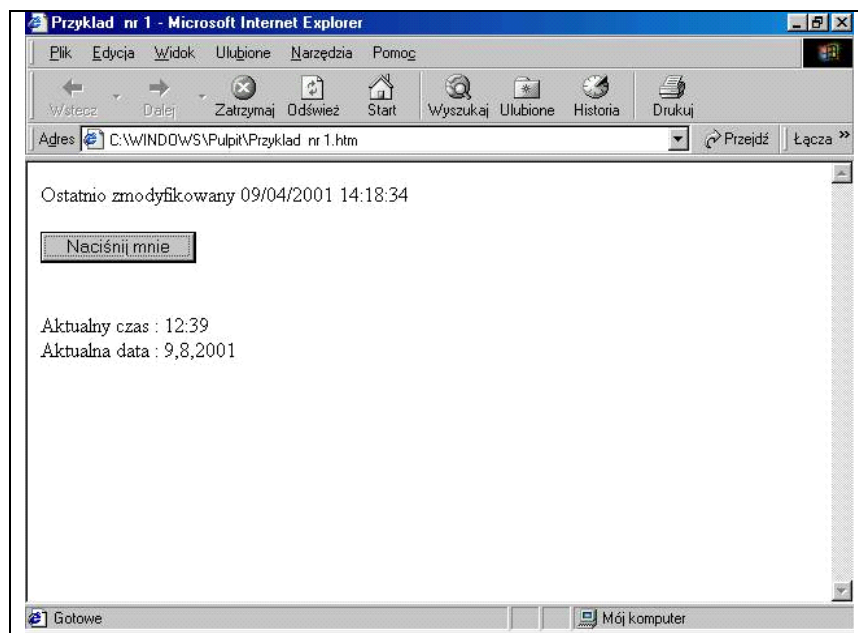
#### Uwaga:

Do miesiąca należy dodać 1 ponieważ funkcja odczytuje wartości od 0 do 11.

```
<html>
<head>
<title> Przykład nr 4</title>
  <script language="JavaScript">
    function pushbutton() {
      alert("Czesc !");}
  </script>
</head>
<body>
  Ostatnio zmodyfikowany
  <script language="LiveScript">
    document.write(document.lastModified,"<br>")
  </script>
  <form>
    <input type="button" name="Button1" value="Naciśnij mnie" onclick="pushbutton()">
  </form>
  <script language="JavaScript">
    dzis = new Date()
    document.write("<br>Aktualny czas : ",dzis.getHours(),":",dzis.getMinutes())
    document.write("<br>Aktualna data : ",dzis.getMonth()+1,"",dzis.getDate(),"",dzis.getFullYear());
  </script>
</body>
</html>
```



Strona po wyświetleniu powinna wyglądać mniej więcej tak:



### **Zadanie 82.(Zadanie 4a)**

Temat: Wykonaj modyfikację przykładu poprzedniego.

Użyj jedno liniowych CSS.

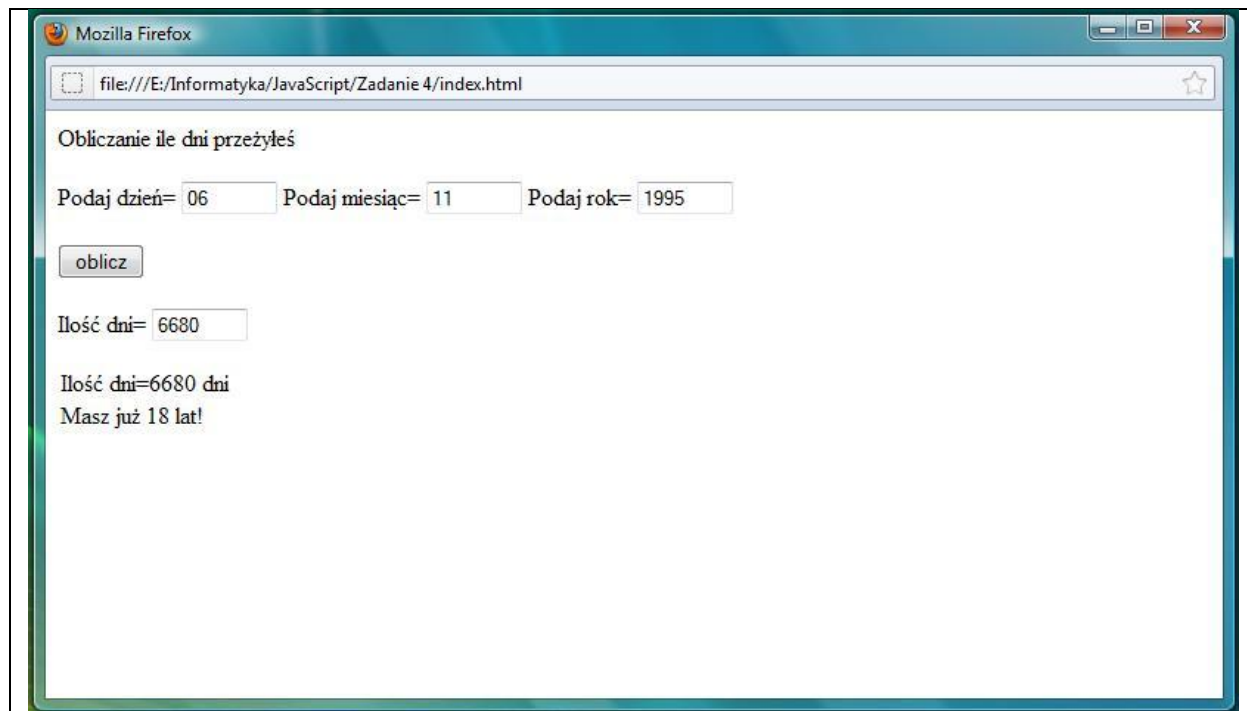
- 1)Ostatnio zmodyfikowano→kolor czerwony, wielkość 1 cm, Arial
- 2)Aktualny czas→kolor zielony, wielkość 1,5 cm, Calibri
- 3)Aktualna data→kolor niebieski, wielkość 2 cm, Times New Roman
- 4)Dodatkowy przycisk, który po naciśnięciu uruchomi okienko z imieniem ucznia.

### **Zadanie 83.(Zadanie 4)**

Temat: Utwórz teraz dokument, w którym po podaniu dnia, miesiąca i roku twoich urodzin program stwierdzi:

- ile dni już przeżyłeś
- czy jesteś pełnoletni czy też nie a może właśnie masz osiemnaste urodziny.

Wczytywanie danych z formularza. Wyniki na stronie.



### Etapy wykonania zadania:

Etap1→wykonanie formularza:

- wczytywanie daty czyli **dzień miesiąc rok**
- wykonanie okienka do **wypisywania** obliczeń (obliczona ilość dni)
- wykonanie tabeli do **wypisywania** obliczeń (obliczona ilość dni oraz czy masz więcej czy mniej niż 18 lat)

```
<form name="formularz">
    Podaj dzień=
    <input size="6" name="dzien">
    Podaj miesiąc=
    Tutaj miesiąc i rok
    <br>
    <br>
    <input TYPE="submit" Value="oblicz" >
    <br>
    <br>
    Ilość dni=
    <input size="6" name="ilosc">
</form>
<table>
    <tr id="pokaz1"></tr>
    Jeszcze jeden wiersz tabeli
</table>
```

Etap2→wykonanie funkcji:

Umieść w kodzie programu funkcję javascript, pamiętaj aby funkcję osadzić w <head>  
w <script language="JavaScript">

```

function dni_nazwisko(dzien,miesiac,rok) // function dni_kowalski(dzien,miesiac,rok)
{
    d_liczba=parseFloat(dzien) // zamiana tekstu na liczbę, tekst który był parametrem funkcji
    m_liczba=parseFloat(miesiac)
    r_liczba=parseFloat(rok)

    var dzis=new Date(); // utworzenie obiektu z dzisiejszą datą
    var urodziny=new Date(r_liczba,m_liczba-1,d_liczba);
    // utworzenie obiektu z datą urodzin użytkownika
    // m_liczba-1 musi być -1 bo komputer styczeń=0
    var czyRok; // zmienne pomocnicze
    var czyMiesiac;
    var czyDzien;

    ilosc_wynik=Math.floor((dzis-urodziny)/((????????????????????)));
    // Math.floor- metoda matematyczna zaokrąglająca
    // różnica (dzis- urodziny) podaje wynik w milisekundach
    // musisz przeliczyć na dni, zapisz w miejsce znaków zapytania.

    document.getElementsByName('ilosc')[0].value=ilosc_wynik;
    // wyprowadzanie do okienka formularza

    alert(????????????????????);
    // wyprowadzanie do okienka alert

    pokaz1.innerHTML="Ilość dni="+ilosc_wynik+" dni"+"<br>";
    // wyprowadzenie do tabeli

    czyRok = dzis.getFullYear();
    czyMiesiac = dzis.getMonth();
    czyDzien = dzis.getDate();
    // przypisanie dzisiejszej daty do zmiennych

    // teraz sprawdzanie warunków czy masz więcej czy mniej niż 18 lat

    if ((czyRok==(r_liczba+18))&&(czyMiesiac==(m_liczba-1))&&(czyDzien==d_liczba))
    {
        pokaz2.innerHTML="Masz dziś 18 urodziny!";
    }
    if (czyRok<(r_liczba+18))
    {
        pokaz2.innerHTML="Nie masz jeszcze 18 lat!";
    }

    .....
    Tutaj pozostałe warunki

    .....
}

```

Etap3→wykonanie akcji dla funkcji:

```
<form name="formularz" action="..."  
onsubmit="dni(this.dzien.value,this.miesiac.value,this.rok.value);return false">
```

### **Zadanie 84. (Zadanie 5)**

Wykonaj program stoper o następującym wyglądzie:



#### Zasada działania:

przyciskamy START stoper → następuje początek pomiaru czasu  
przyciskamy STOP starter → uzyskujemy czas w sekundach i jej tysięcznych.

#### Algorytm rozwiązania zadania

1)wstaw formularz z dwoma przyciskami:

```
<input type="button" name="Button1" value="START stoper" onclick="kowalski_start()">
```

nazwa funkcji → nazwisko\_ucznia\_start

nazwa funkcji → nazwisko\_ucznia\_stop

2)wyświetl Aktualny czas:

poprzez skrypt w <body>

```
dzis_kow = new Date();
```

```
document.write("<br>Aktualny czas : ",dzis_kow.getHours(),":",dzis_kow.getMinutes());
```

uwaga: obiekt Date() z trzema literami nazwiska ucznia

3)Wykonaj dwie funkcje w head

nazwa funkcji → nazwisko\_ucznia\_start

nazwa funkcji → nazwisko\_ucznia\_stop

poniżej wewnątrz funkcji nazwisko\_ucznia\_start

```
{
    start_kow = new Date();
    poczatek_kow=start_kow.getTime()
}
```

poniżej wewnątrz funkcji nazwisko\_ucznia\_stop

```
{
    stop_kow = new Date();
    koniec_kow=stop_kow.getTime()

    roznica_kow=koniec_kow-poczatek_kow;
    pokaz1.innerHTML=roznica_kow/1000+" s";
}
```

4)wstaw tabelę na koniec body

```
<table tutaj obramowanie>
  <tr id="pokaz1"></tr>
</table>
```

### **Zadanie 85. (Zadanie 6)**

Wykonaj program wypisujący dzień tygodnia na podstawie aktualnej daty.  
Użyj instrukcji switch case oraz getDay().

### **Zadanie 86. Zadanie 6a**

Wykonaj program który będzie miał trzy [aktywne przyciski](#) wykonane z grafik.

Wykonaj;

grafiki ściągnij z Internetu. Będą to trzy komplety przycisków. Komplet składa się z dwóch grafik różniących się od siebie np. kolorem liter lub cieniowaniem.

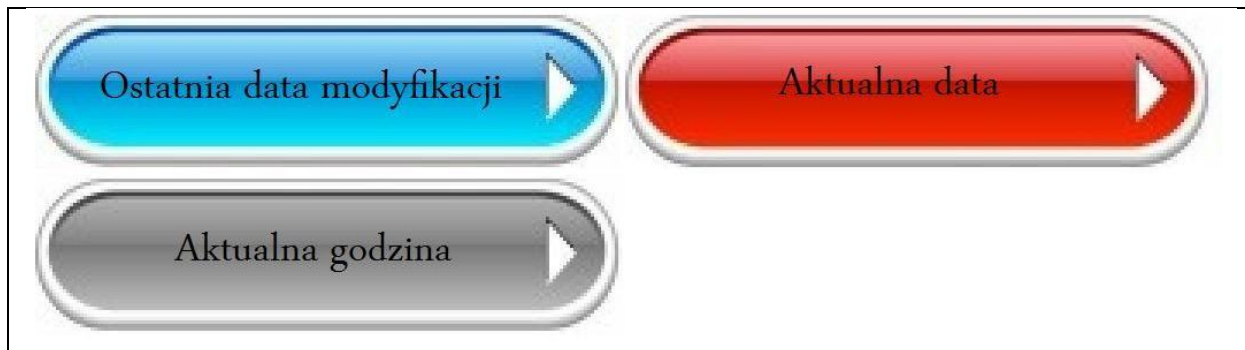
do przycisków dodaj akcje:

akcja 1 → ostatnia data modyfikacji

akcja 2 → aktualna data

akcja 3 → aktualna godzina

Widok1



Widok2



### **Funkcje:**

Nazwy

```
function Przycisk1_nazwisko_ucznia()
```

```
function Przycisk2_nazwisko_ucznia()
```

```
function Przycisk3_nazwisko_ucznia()
```

np.

```
function Przycisk1_nowak()
```

```
{
pokaz1.innerHTML="Ostatnio zmodyfikowany : "+document.lastModified;
}
```

### **Kod**

Wyświetlanie wyników z użyciem właściwości tabeli

```
<form>
<a onclick="Przycisk1_nowak(' ')"></a>
.....
</form>
<table>
.....
<tr id="pokaz2"></tr>
.....
</table>
```



## Dział 4 Formularze

### Zadanie 87. (Przykład 5)

Temat: Pokaz działania formularza.

W formularzu sprawdzana jest zawartość pól:

- czy pole nie jest puste
- czy w adresie email jest znak @

W formularzu ustawiane jest aktywne pole:

- tekstowe → wpisywania nazwiska
- radio → środkowa opcja

Na co zwrócić uwagę;

- `document.jeden.txt1.focus();` → uruchomienie metody `focus()` czyli aktywny element formularza,
- `<body onLoad="setfocus()">` → uruchomienie funkcji w momencie załadowania dokumentu do przeglądarki,
- `<input name="choice" type="radio" value="2" checked>`Może być<BR> → zaznaczenie aktywnego radio,
- `form.txt2.value.indexOf('@', 0) == -1` → wartość wpisaną w polu tekstowym traktujemy jako tablicę znaków, sprawdzamy czy w tablicy znaków występuje znak '@', jeśli nie to metoda `indexOf` zwraca -1, zero (0) w parametrach oznacza, że przeszukiwanie jest od zerowego elementu tablicy.

Wykonaj:

- zmień domyślne pole tekstowe (czyli miejsce gdzie jest kursor wpisywania danych po uruchomieniu programu),
- zmień domyślne pole radio (czyli zaznaczone pole radio po uruchomieniu programu),
- dopisz takie sprawdzanie aby nie można było wpisać nazwiska krótszego niż 4 znaki → użyj własności tekstu `length`.

```
<html>
```

```
<head>
```

```
<title>
```

```
Przykład -> pokaz formularza
```

```
</title>
```

```
<script language="JavaScript">
```

```
function setfocus()
```

```
{
    document.jeden.txt1.focus(); return;
}
```

```
function test1(form)
```

```
{
    if (form.txt1.value == "")
    {
        alert("Podaj ciąg znaków!")
    }
    else
    {
        alert("Cześć "+form.txt1.value+" ! Informacja poprawna!");
    }
}
```

```

function test2(form)
{
    if (form.txt2.value == "" || form.txt2.value.indexOf('@', 0) == -1)
    {
        alert("Niepoprawny adres poczty elektronicznej!");
    }
    else
    {
        alert("OK!");
    }
}

</script>
</head>
<body onLoad="setfocus()">

<form name="jeden">
    Wpisz swoje nazwisko:<br>
    <input type="text" name="txt1">
    <input type="button" name="button1" value="Test" onClick="test1(this.form)">
</form>

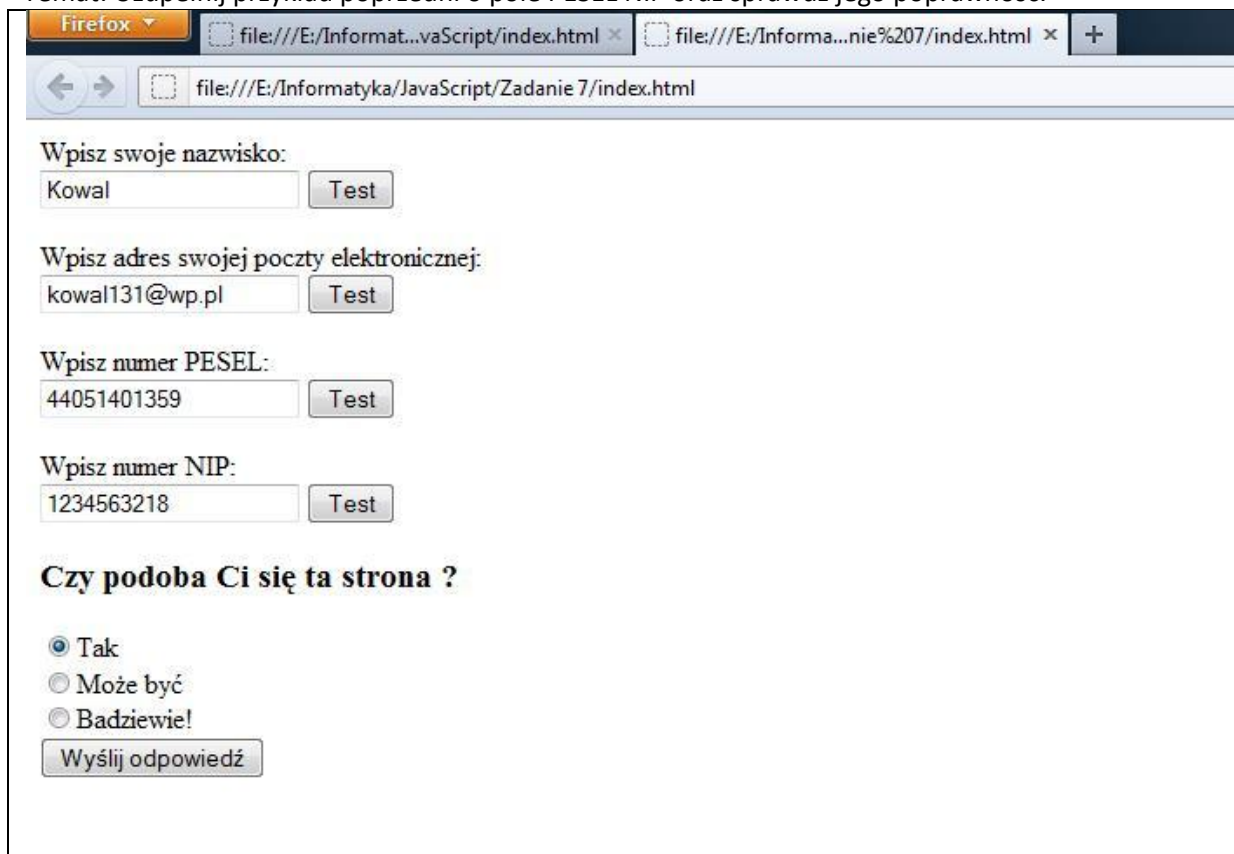
<form name="dwa">
    Wpisz adres swojej poczty elektronicznej:<br>
    <input type="text" name="txt2">
    <input type="button" name="button2" value="Test" onClick="test2(this.form)">
</form>

<form method="post" action="mailto:dowolny@mail.pl">
    <h3>Czy podoba Ci się ta strona ?</h3>
    <input name="choice" type="radio" value="1">Tak<BR>
    <input name="choice" type="radio" value="2" checked>Może być<BR>
    <input name="choice" type="radio" value="3">Badziewie!<BR>
    <input name="submit" type="submit" value="Wyślij odpowiedź">
</form>
</body>
</html>

```

## Zadanie 88.(Zadanie 7)

Temat: Uzupełnij przykład poprzedni o pole PESEL NIP oraz sprawdź jego poprawność.



Firefox

file:///E:/Informat...vaScript/index.html x file:///E:/Informa...nie%207/index.html x +

file:///E:/Informatyka/JavaScript/Zadanie 7/index.html

Wpisz swoje nazwisko:  
Kowal Test

Wpisz adres swojej poczty elektronicznej:  
kowal131@wp.pl Test

Wpisz numer PESEL:  
44051401359 Test

Wpisz numer NIP:  
1234563218 Test

**Czy podoba Ci się ta strona ?**

☒ Tak  
☐ Może być  
☐ Bardziej

Wyślij odpowiedź

Rozwiązanie:

1) Zapewnij wpisywanie PESEL w formularzu.

```
<form name="pesel">  
  Wpisz numer PESEL:<br>  
    <input type="text" name="pesel_kowalski">  
    <input type="button" name="button3" value="Test" onClick="test_pesel(this.form)">  
</form>
```

2) Zapewnij wpisywanie NIP w formularzu.

```
.....  
    <input type="text" name="nip_kowalski">  
.....
```

3) wykonaj funkcję sprawdzania (przeczytaj algorytm poniżej)

- sprawdzenie czy Pesel ma 11 znaków

- podział oraz zamiana na wartość Int 11 znaków PESEL i zapisanie do osobnych zmiennych l1..l11

Konwersja → parsellnt

Wycinanie znaków z tekstu → substring(?,?)

4) obliczenie lk sumy kontrolnej w zmiennej lk

5) obliczenie reszty z dzielenia przez 10 w zmiennej ck (cyfra kontrolna)

6) odejmowanie od 10 ck i zapis w zmiennej ckk (cyfra kontrolna końcowa)

7) Sprawdzenie warunku czy cyfra kontrolna końcowa jest równa ostatniemu znakowi Pesel lub czy cyfra końcowa jest równa zero i czy ostatni znak jest równy zero.

8) Wykonaj podobnie dla NIP

```
function test_pesel(form)
{
    if (form.txt3.value.length == 11)
    {
        var l1= parseInt (form. pesel_kowalski.value.substring(0,1));
        var l2=parseInt(form. pesel_kowalski.value.substring(1,2));
        .....

        var lk=(1*l1+3*l2+7*l3+.....);
        var ck=.....;
        var ckk=.....;
        if (????????????????????)
        {
            alert("PESEL prawidłowy!");
        }
        else
        {
            alert("PESEL nieprawidłowy!");
        }
    }
    else
    {
        alert("PESEL nieprawidłowy!");
    }
}
```

### Numer PESEL → algorytm sprawdzania poprawności.

Numer PESEL jest to 11-cyfrowy, stały symbol numeryczny, jednoznacznie identyfikujący określoną osobę fizyczną.

Zbudowany jest z następujących elementów:

- zakodowanej daty urodzenia
- liczby porządkowej
- zakodowanej płci
- cyfry kontrolnej

Przykładowa postać:

4	4	0	5	1	4	0	1	4	5	8
---	---	---	---	---	---	---	---	---	---	---

cyfry [1-6] – data urodzenia z określeniem stulecia urodzenia

cyfry [7-10] – numer serii z oznaczeniem płci

cyfra [10] – płeć

cyfra [11] – cyfra kontrolna

### Data urodzenia w formacie

RRMMDD

RR      -rok

MM -miesiąc  
DD -dzień

dla osób urodzonych w latach 1900 do 1999 – miesiąc zapisywany jest w sposób naturalny  
dla osób urodzonych w innych latach niż 1900 – 1999 dodawane są do numeru miesiąca następujące wielkości:

dla lat 1800-1899 - 80

dla lat 2000-2099 - 20

dla lat 2100-2199 - 40

dla lat 2200-2299 - 60

Przykładowo osoba urodzona 14 lipca 2002 roku ma następujący zapis w numerze ewidencyjnym:

0	2	2	7	1	4
---	---	---	---	---	---

### **Płeć**

Informacja o płci osoby, zawarta jest na 10. (przedostatniej) pozycji numeru PESEL.

cyfry 0, 2, 4, 6, 8 – oznaczają płeć żeńską

cyfry 1, 3, 5, 7, 9 – oznaczają płeć męską

### **Cyfra kontrolna i sprawdzanie poprawności numeru**

Jedenasta cyfra jest cyfrą kontrolną, służącą do wychwytywania przekłamań numeru. Jest ona generowana na podstawie pierwszych dziesięciu cyfr. Aby sprawdzić czy dany numer PESEL jest prawidłowy należy, zakładając, że litery **a-j** to kolejne cyfry numeru od lewej, obliczyć wyrażenie

$$1*a + 3*b + 7*c + 9*d + 1*e + 3*f + 7*g + 9*h + 1*i + 3*j$$

Następnie należy odjąć ostatnią cyfrę otrzymanego wyniku od 10. Jeśli otrzymany wynik nie jest równy cyfrze kontrolnej, to znaczy, że numer zawiera błąd.

Uwaga implementacyjna - jeśli ostatnią cyfrą otrzymanego wyniku jest 0, w wyniku odejmowania otrzymamy liczbę 10, podczas gdy suma kontrolna jest cyfrą. Oznacza to tyle, że cyfra kontrolna winna być równa 0 (stąd dobrze jest wykonać na wyniku odejmowania operację modulo 10). W wyniku niezbyt precyzyjnego opisu na stronie MSW ten aspekt jest często pomijany i prowadzi do błędów w implementacji sprawdzania poprawności numeru PESEL.

Przykład dla numeru PESEL 44051401358:

$$1*4 + 3*4 + 7*0 + 9*5 + 1*1 + 3*4 + 7*0 + 9*1 + 1*3 + 3*5 = 101$$

Wyznaczamy resztę z dzielenia sumy przez 10:

$$101:10 = 10 \text{ reszta} = 1$$

Jeżeli reszta = 0, to cyfra kontrolna wynosi 0. Jeżeli reszta  $\neq$  0, to cyfra kontrolna będzie uzupełnieniem reszty do 10, czyli w podanym przykładzie jest to cyfra 9.

$$10 - 1 = 9$$

Wynik 9 nie jest równy ostatniej cyfrze numeru PESEL, czyli 8, więc numer jest błędny.

**Metoda równoważna → algorytm drugi poprawności zapisu.**

Powyższa metoda sprowadza się do obliczenia wyrażenia:

$$a+3b+7c+9d+e+3f+7g+9h+i+3j+k$$

gdzie litery oznaczają kolejne cyfry numeru, i sprawdzeniu czy reszta z dzielenia przez 10 jest zerem (ostatnia cyfra powstałej liczby jest zerem) i jeśli nie jest, to numer jest błędny.

Cechy specyficzne algorytmu sprawdzania

#### **Wada algorytmu**

Algorytm ma pewną wadę w przydziale wag do poszczególnych elementów, która powoduje, że gdy zamienimy rok z dniem (zamieniając zapis z rr-mm-dd na dd-mm-rr) otrzymamy identyczną sumę kontrolną jak w numerze z poprawnym zapisem.

W praktyce zdarzają się (a przynajmniej zdarzały i wciąż istnieją) numery PESEL z błędami. Błędy w dacie zwykle były zauważane i poprawiane od razu, lecz zdarzały się też powtórzenia numeru porządkowego, błędy w określeniu płci i błędne cyfry kontrolne, które zostały wychwycone po latach przy okazji wprowadzania numeru PESEL do komputerowych baz danych. W związku z tym nie można zakładać, że wynik sprawdzania jednoznacznie określa istnienie bądź nieistnienie podanego numeru PESEL.

#### **Sprawdzanie poprawności numeru NIP**

NIP czyli Numer Identyfikacyjny Płatnika to numer przyznawany każdemu podatnikowi przez właściwy Urząd Skarbowy, służący do identyfikowania osoby lub podmiotu gospodarczego przy prowadzeniu rozliczeń podatkowych. NIP składa się z dziesięciu cyfr. Pierwsze dziewięć cyfr to część identyfikacyjna. Dziesiąta cyfra jest cyfrą kontrolną. Sprawdzanie poprawności NIP-u można przeprowadzić za pomocą następujących czynności:

A. Wymnożyć kolejne cyfry (od pierwszej do dziewiątej) przez odpowiednie wagi. Jako wagi stosujemy kolejno liczby  
6 5 7 2 3 4 5 6 7

B. Zsumować wyniki uzyskane w punkcie A

C. Wynik otrzymany w punkcie B należy podzielić modulo przez 11

Jeśli NIP jest poprawny to liczba uzyskana w punkcie C będzie równa dziesiątej cyfrze.

#### **Zadanie 89. (Przykład 6)**

Temat: Demonstracja obliczeń z użyciem formularza.

```
<html>
<head>
<script type="text/javascript">
function obliczanie(dane)
{
    wynik.innerHTML="";
    cena_koncowa=0;
    if ( dane.wybor[0].checked)
```

[illegible]

```

przedłużona gwarancja 28%<input type="checkbox" name="gwarancja" value=0 /><br><br>
<!-- <input type="button" value="Oblicz....." onClick="obliczanie(this.form)"/><br><br> -->
<input type="button" value="Oblicz....." onClick="obliczanie(this.form)"/><br><br>
</form>
<table>
    <tr id="wynik"></tr>
</table>
</body>
</html>

```

## Zadanie 90. (Zadanie 8a)

Temat: Wykonaj formularz ALKOTESTU

## Zadanie 91. (pobrane z Internetu)

Sprawdzanie czy pola pozostały puste – sposób pierwszy.

```

<script type="text/javascript">
    function f(w,i){
        var pola=['imie','nazwisko','adres','kod_pocztowy'];
        dane=w.elements;
        for(i=0;i<pola.length;i++){
            if(dane[pola[i]].value==""){
                dane[pola[i]].focus(); alert("Musisz wypełnić wymagane pola"); return false;
            }
        }
        return true;
    }
</script>
<form onsubmit="[b]return [/b]f(this)">
<input type="text" name="imie">
<input type="text" name="nazwisko">

```



```
<input type="text" name="kod_pocztowy">
</form>
```

### **Zadanie 92. (pobrane z Internetu)**

Temat: Sprawdzanie czy wszystkie pola formularza są wypełnione-przykład2.

```
<script>
function sprawdz(formularz)
{
    for (i = 0; i < formularz.length; i++)
    {
        var pole = formularz.elements[i];
        if (!pole.disabled && !pole.readonly && (pole.type == "text" || pole.type == "password" ||
        pole.type == "textarea") && pole.value == "")
        {
            alert("Proszę wypełnić wszystkie pola!");
            return false;
        }
    }
    return true;
}
</script>

<form action="mailto:adres e-mail?subject=temat" method="post" enctype="text/plain"
onsubmit="if (sprawdz(this)) return true; return false">
<div>
    <input type="text" name="a"><br>
    <input type="password" name="b"><br>
    <textarea name="c" cols="20" rows="10"></textarea><br>
    <input type="submit" value="OK">
</div>
</form>
```

Metaznak	Znaczenie	Przykład wyrażenia	Zgodne ciągi z wyrażeniem	Niezgodne ciągi z wyrażeniem
^	początek wzorca	^za	zapałka zadra zapłon zarazek	kazanie poza bazar
\$	koniec wzorca	az\$	uraz pokaz	azymut pokazy
		^.arka\$	barka warka	parkan
.	dowolny pojedynczy znak	.an.a	panda Wanda panna kania	rana konია
[...]	dowolny z wymienionych znaków; możemy podawać kolejne znaki lub wpisywać zakres - na przykład [a-z] oznacza wszystkie małe litery. Wymieniając escapowane znaki (następna tabela) nie musimy ich poprzedzać ukośnikiem	[a-z]an[nd]a	pana panda wanna	Wanda kania
		[a-z][a-zA-Z0-9.-]	pas mAs p2p	Bas bal balu mp3
[^...]	dowolny z niewymienionych znaków	kre[^st]	krew krem	kres kret
	dowolny z rozdzielonych znakiem ciągów	[nz]a pod przed	na za pod przed	
		trzynasty 13-ty 13	trzynasty 13-ty 13	
(...)	zawężenie zasięgu	g(ż rz)eg(ż rz)(u ó)łka	gżegżółka gżegrzółka gżegrzułka grzegrzułka	
		(ósm 8-my 8)(maj maja)	ósm 8-my 8-maj 8 maja	
?	zero lub jeden poprzedzający znak lub element; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	ro?uter	router ruter	
		(ósm 8(-my)?)maja?	ósm 8-my ósm 8-maj 8-maj	

			8-my maj 8 maja 8 maj	
+	jeden lub więcej poprzedzających znaków lub elementów; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	[0-9]+[abc]	10a 1b 003c 42334b	a b c z 14 03 12d 1231z
		pan+a	pana panna pannnna	
		(tam)+	tam tamtam tamtamtam	paa panda ta tamta mat
*	zero lub więcej poprzedzających znaków lub elementów; elementem może być na przykład wyrażenie umieszczone wewnątrz nawiasów (...)	[0-9]*[abc]	10a 1b 003c 42334b a b c	k 2335
		pora*n*a*	por poa poranna poraannnaa pornnna	porada panna
{4}	dokładnie 4 poprzedzające znaki lub elementy	[0-9]{4}	8765 8273 2635	12345 234 2123456
{4,}	4 lub więcej poprzedzających znaków lub elementów	[ah]{4,}	haha haaaaahaha ahaaa	haa ha hehe aha
{2,4}	od 2 do 4 poprzedzających znaków lub elementów	p.{2,4}a	piana pola polana	psa poranna

Jeżeli chcemy w wyrażeniu zawrzeć znaki, które normalnie mają specjalne znaczenie, musimy poprzecić taki znak ukośnikiem.

\.	znak kropki	[0-9]{,3}\.[0-9]{,3}\.[0-9]{,3}	128.0.0.2	128-0-0-2
----	-------------	---------------------------------	-----------	-----------

\*	znak *	\*.*+	*nic	nic* nic
\	znak /	^\/\/\$	//	
\?	znak ?	^.*\?\$	Czy to jest kot?	Czy to jest kot
\:	znak :	^.*\:\$	Oto one:	:nic
\.	znak .	\.*	.....	
\^	znak ^	.*\^	To jest ^	To jest &
\+	znak +	[0-9]+\ [0-9]+\	928374+29832	23873-32787 238738278
\\	znak \	c\:\	c:\	
\=	znak =	[0-9]+\ [0-9]+\ [0-9]+\	11+12=23	11-12=23
\	znak	x \  y	x \  y	x -- y

### Flagi

Poza wymienionymi meta znakami istnieją specjalne parametry (flagi), które oddziałują na wyszukiwanie wzorców.:

const reg = /[a-z]\*/mg

const reg = new RegExp("[a-z]\*", "g")

znak Flagi	znaczenie
i	powoduje niebranie pod uwagę wielkości liter
g	powoduje zwracanie wszystkich psujących fragmentów, a nie tylko pierwszego
m	powoduje wyszukiwanie w tekście kilku liniowym. W trybie tym znak początku i końca wzorca (^\$) jest wstawiany przed i po znaku nowej linii (\n).

### Klasy znaków

Dodatkowo Javascript udostępnia specjalne klasy znaków. Zamiast wyszukiwać wszystkie litery za pomocą [a-zA-Z\_] możemy skorzystać z klasy znaków \w.

Klasa znaków	znaczenie
\s	znak spacji, tabulacji lub nowego wiersza
\S	znak nie będący spacją, tabulacją lub znakiem nowego wiersza
\w	każdy znak będący literą, cyfrą i znakiem _
\W	każdy znak nie będący literą, cyfrą i znakiem _
\d	każdy znak będący cyfrą
\D	każdy znak nie będący cyfrą

**Interfejs programistyczny aplikacji** (ang. Application Programming Interface, **API**) – sposób, rozumiany jako ściśle określony zestaw reguł i ich opisów, w jaki programy komputerowe komunikują się między sobą. API definiuje się na poziomie kodu źródłowego dla takich składników oprogramowania jak np. aplikacje, biblioteki czy system operacyjny. Zadaniem API jest dostarczenie odpowiednich specyfikacji podprogramów, struktur danych, klas obiektów i wymaganych protokołów komunikacyjnych. Przykładem API jest POSIX, czy też Windows API.

**Windows API, lub krócej: WinAPI** – interfejs programistyczny systemu Microsoft Windows – jest to zbiór niezbędnych funkcji, stałych i zmiennych umożliwiających działanie programu w systemie operacyjnym Microsoft Windows.

## Dział 5 Przyciski

### Zadanie 8

Temat: Różne funkcje przycisków (button).

Wykonaj:

stronę o następującej budowie(możesz wykorzystać stronę, którą już robiłeś)

<p>MENU</p> <p>Opcja1 Opcja2 Opcja3</p>	<p>Otwierane strony zaproponowanych przez ucznia</p>
<p>Przyciski</p> <p>[strona do przodu]      [drukuj stronę]      [wstecz strona]      [odśwież stronę]</p>	

**Działanie MENU pionowego, (patrz schemat strony powyżej);**

Opcje menu wybierają (otwierają) strony zaproponowane przez ucznia . Wybrana opcja wgra stronę w dużym oknie.

**Działanie MENU poziomego, ( zbudowane z przycisków, patrz schemat strony powyżej);**

Wybór jednego z Przycisków powoduje akcja na stronie w dużym okna (strona do przodu, wstecz, odśwież).

**Pomocna teoria:**

a) do przodu strona

```
function przod()
{
    history.forward()
}
```

b) wstecz strona  
`history.back()`

c) odświeżanie strony  
`location.refresh()`

d) drukowanie strony  
`window.print()`

## **Dział 7 Galerie**

### **Zadanie 12**

Temat: Klasyczna galeria zdjęć

[Wykonanie miniatur.](#)

[Wykonaj klasyczna galerię zdjęć:](#)

sześć zdjęć i miniatur związanych z:

Numer z dziennika	Temat
1,17	Parki narodowe
2,18	Gdańsk
3,19	Warszawa
4,20	Samochody wyścigowe
5,21	Tatry
6,22	Kultyryści
7,23	Morze
8,24	Wyspy
9,25	Piłka nożna
10,26	Antyczna Grecja
11,27	Słynne obrazy
12,28	Słynne mosty
13,29	Zamki polskie
14,28	Arbuzy
15,29	Sady
16,30	Motocykle

Dla numerów nieparzystych ułożenie miniatur


Dla numerów parzystych ułożenie miniatur


Po uzyskaniu całego zdjęcia powinien być przycisk wróć do miniatur.



## Dział 8 Skrypty z internetu

### Zadanie 12a

Temat: Dołączenie 5 grafik uzyskanych z Internetu.

Wykonaj:

- 1) Do strony na, której będzie Twoje nazwisko (zielone, 3cm → formatowanie napisu zewnętrznym arkuszem stylu CSS o nazwie twoje\_nazwisko.css) dokomponuj pięć skryptów JS z Internetu.
- 2) Wykonaj przyciski (buttony) po, których naciśnięciu wyświetli się kod skryptu JS. Przycisk ma napis z nazwą skryptu.

KÓŁKO I KRZYŻYK

0

Komunikat:

Stopień trudności: Medium

Computer wykonuje ruch jako pierwszy

Restart

Data

Zegar tekstowy

Dodaj do zakładek

Kółko i Krzyżyk

Śnieg

Chodorowski

Po naciśnięciu przycisku Data uzyskujemy kod w okienku np.

```
<SCRIPT LANGUAGE="JavaScript">
<!-- // skrypt pobrano z http://szablony.freeware.info.pl
miesiac = new Array(12)
miesiac[0] = "stycznia "
miesiac[1] = "lutego "
miesiac[2] = "marca "
miesiac[3] = "kwietnia "
miesiac[4] = "maja "
miesiac[5] = "czerwca "
miesiac[6] = "lipca "
miesiac[7] = "sierpnia "
miesiac[8] = "wrzenia "
miesiac[9] = "padziernika "
miesiac[10] = "listopada "
miesiac[11] = "grudnia "
dzien = new Array(7)
dzien[0] = "niedziela "
dzien[1] = "poniedziałek "
dzien[2] = "wtorek "
dzien[3] = "sroda "
dzien[4] = "czwartek "
dzien[5] = "piątek "
dzien[6] = "sobota "
function podaj_date() {
var Dzisiaj = new Date()
var Tygodnia = Dzisiaj.getDay()
var Miesiac = Dzisiaj.getMonth()
var Dnia = Dzisiaj.getDate()
var Rok = Dzisiaj.getFullYear()
if(Rok <= 99) Rok += 1900
return dzien[Tygodnia] + "," + " " + Dnia + " " +
miesiac[Miesiac] + ", " + Rok + "r." }
//-->
</SCRIPT>
<SCRIPT>document.write("Dzis jest " + podaj_date())</SCRIPT>
```

## Dział 9 HTML5 canvans

Element **canvas** umożliwia rysowanie **grafiki rastrowej** na stronach internetowych.

**Canvas** jest prostokątnym obszarem, który po umieszczeniu na stronie **HTML5** umożliwia kontrolowanie każdego punktu za pomocą **JavaScript**.

Technologia ta pozwala on na:

- rysowanie grafiki, (proste figury geometryczne)
- różnego rodzaju krzywe,
- gradienty,
- wykresy,
- tworzenia galerii zdjęć,
- tworzenia animacji 2D ( poprzez cykliczne przerysowywanie całego obszaru),
- gry.

<http://msdn.microsoft.com/pl-pl/library/kurs-html5--canvas--grafika.aspx>

### **Zadanie 14**

Temat: Narysuj żółty kwadrat z mniejszym czerwonym kwadratem pośrodku.

#### Etap1

Szkielet strony:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Canvas</title>
  </head>
  <body>
  </body>
</html>
```

#### Etap2

Utworzenie elementu `<canvas>` → nowy znacznik HTML5

W celu określenia obszaru rysowania, pomiędzy znacznikami `<body></body>`, dodaj kod wg schematu:

```
<canvas id="nazwa" width="" height="">
</canvas>
```

gdzie:

id – unikalny identyfikator,  
width – szerokość elementu canvas (domyślnie 300px),  
height – wysokość elementu canvas (domyślnie 150px).

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Canvas</title>
  </head>
  <body>
    <canvas id="obrazek" width="300" height="300">
    </canvas>
  </body>
</html>
```

Definiowanie koloru tła obszaru do rysowania definiowanego w canvans.

```
<canvas
  id="plaszczyna" width="800" height="600" style='background-color: lightgreen;'>
</canvas>
```

### Etap3

Po utworzeniu nowego elementu, nie jest on jeszcze widoczny w oknie przeglądarki. Należy skorzystać z metody `getContext`, umożliwiającej wykorzystanie funkcji rysujących.

Metodę `getContext` umieść w funkcji, którą wykorzystasz do narysowania prostokąta.

Wykonaj:

Do kodu z etapu 2 dodaj:

Pod `</title>` wpisz skrypt Javascript:

```
<script type="text/javascript">
function rysuj()
{
  var rysowanie = document.getElementById('obrazek');
  var rodzaj = rysowanie.getContext('2d');
}
</script>
```

Dodaj atrybut w znaczniku `<body>`, który uruchomi funkcję `rysuj()` po załadowaniu strony:

```
<body onload="rysuj();">
```

**Metoda `getContext`** obiektu `canvas` powoduje pobranie kontekstu płaszczyzny, który jest nam potrzebny, abyśmy mogli wywoływać funkcje rysujące.

Pierwszy parametr metody `getContext` to identyfikator kontekstu. Jeśli podamy „**2d**”, pobierzemy kontekst implementujący interfejs `CanvasRenderingContext2D`.

Możliwe są też inne wartości. Na przykład, jeśli chcemy rysować grafikę 3D, odpowiednio użylibyśmy wartości „**3d**”. Z kolei, aby zastosować **webgl**, używać będziemy identyfikatora „webgl”.

Powstanie kod strony

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Canvas</title>
    <script type="text/javascript">
function rysuj()
{
    var rysowanie = document.getElementById('obrazek');
    var rodzaj = canvas.getContext('2d');
}
</script>

    </head>
    <body>
        <canvas id="obrazek" width="300" height="300">
        </canvas>
    </body>
</html>
```

#### Etap 4

##### **Rysowanie w zdefiniowanym obiekcie do rysowania.**

Uwaga1:

Punkt (0,0) nie znajduje się jak na układzie kartezjańskim pośrodku obrazka, ale w jego **lewym górnym rogu**.

Metody do rysowania:

**fillRect(x,y,s,w)** – wypełnia obszar z lewym górnym rogiem, znajdujący się w punkcie (x,y) o szerokości (s) i wysokości (w), podawanych w pikselach,

**fillStyle** = 'kolor' – zmienia kolor wypełnienia.

Możliwe formaty koloru:

- 'red',
- postać hexadecymalna, np. '#aabbcc'
- format rgb, np. `rgb(123,123,123)` – podobnie jak w CSS,

**strokeRect(x,y,s,w)** – rysuje kontur prostokąta z lewym górnym rogiem, znajdującym się w punkcie (x,y), o szerokości (s) i wysokości (w), podawanych w pikselach,

**strokeStyle** = 'kolor' – zmienia kolor konturu prostokąta; używa się go tak samo jak przy `fillStyle()`,

**clearRect(x,y,s,w)** – czyści obszar (rysuje przeźroczysty kwadrat) na obszarze z lewym górnym rogiem, znajdującym się w punkcie (x,y), o szerokości (s) i wysokości (w), podawanych w pikselach.

Utwórz nowe linie w funkcji rysuj().

po linii var rodzaj = canvas.getContext('2d'); wpisz:

```
rodzaj.fillStyle = 'yellow';
rodzaj.fillRect(0,0,100,100);
rodzaj.strokeRect(0,0,100,100);
rodzaj.fillStyle = 'red';
rodzaj.fillRect(30,30,30,30);
```

### **Obsługa przeglądarek, które nie wspierają elementu canvas:**

Wykonać to można poprzez umieszczenie tekstu komunikatu pomiędzy znacznikami <canvas></canvas>. Ukaże się on w przypadku, gdy przeglądarka nie będzie wspierała elementu canvas, np. :

```
<canvas id="image" width="300" height="300">
```

Twoja przeglądarka nie wspiera Canvas! Zainstaluj Internet Explorer 9.

```
</canvas>
```

### **Strona końcowa**

```
<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="utf-8">
  <title>Canvas</title>
  <script type="text/javascript">
    function rysuj() {
      var rysowanie = document.getElementById('obrazek');
      var rodzaj = rysowanie.getContext('2d');
      rodzaj.fillStyle = 'yellow';
      rodzaj.fillRect(0, 0, 100, 100);
      rodzaj.strokeRect(0, 0, 100, 100);
      rodzaj.fillStyle = 'red';
      rodzaj.fillRect(30, 30, 30, 30);
    }
  </script>
</head>
<body onload="rysuj();">
  <canvas id="obrazek" width="300" height="300">
    Twoja przeglądarka nie wspiera Canvas!
    Zainstaluj Internet Explorer 9.x
  </canvas>
</body>
</html>
```

### Zadanie 15

Temat: Narysuj zielony kwadrat rogami do góry.

Wykonaj:

1) Dodaj kod do poprzedniego programu:

```
rodzaj.beginPath();
rodzaj.moveTo(50,50);
rodzaj.lineTo(100,0);
rodzaj.lineTo(150,50);
rodzaj.lineTo(100,100);
rodzaj.lineTo(50,50);
rodzaj.fillStyle = 'green';
rodzaj.fill();
```

2) zmień położenie zielonego kwadratu tak aby nie nakładał się elementami z zadania poprzedniego. Współrzędne górnego rogu będą następujące:

$$x=200+(\text{numer\_miesiaca\_rodzenia})*10$$
$$y=300+(\text{numer\_miesiaca\_rodzenia})*11$$

**Podczas tworzenia ścieżek i linii w Canvas, wykorzystaj następujące metody:**

**beginPath()** – rozpoczyna tworzenie ścieżki,

**closePath()** – kończy tworzenie ścieżki,

**moveTo(x,y)** – ustawia punkt, od którego rozpocznie rysować (x,y) (domyślnie jest on ustawiony w miejscu zakończenia poprzedniego rysunku),

**lineTo(x,y)** – rysuje linię do punktu (x,y) z miejsca oznaczonego metodą moveTo() lub punktu, w którym zakończone zostało poprzednie rysowanie,

**fill()** – wypełnia ścieżkę kolorem podanym w parametrze fillStyle(). Warto zwrócić uwagę, że metoda ta sama domyka ścieżki, dlatego nie wymaga użycia metody closePath(),

**fillStyle='kolor'** – zmienia kolor wypełnienia metody, gdzie kolor może być nazwą angielską, formatem szesnastkowym lub RGB,

**stroke()** – rysuje kontur ścieżki,

**strokeStyle='kolor'** – zmienia kolor konturu ścieżki, gdzie kolor może być nazwą angielską, formatem szesnastkowym lub rgb.

**lineWidth = size** – grubość linii w pixelach.

### Zadanie 16

Temat: Narysuj kółka olimpijskie.

Wykonaj:

1. Skasuj rysowanie z poprzedniego zadania.
2. Dodaj kod do poprzedniego programu.
3. Wykonaj rysowanie kółek olimpijskich (kolory mają być identyczne i wielkości kółek odpowiedniej wielkości).

```
rodzaj.beginPath();  
rodzaj.arc(200,50,30,Math.PI,0,false);  
rodzaj.closePath();  
rodzaj.lineWidth = 10;  
rodzaj.strokeStyle = "black";  
rodzaj.stroke();
```

**Rysowanie łuku** → przy rysowaniu łuków wykorzystujesz metody:

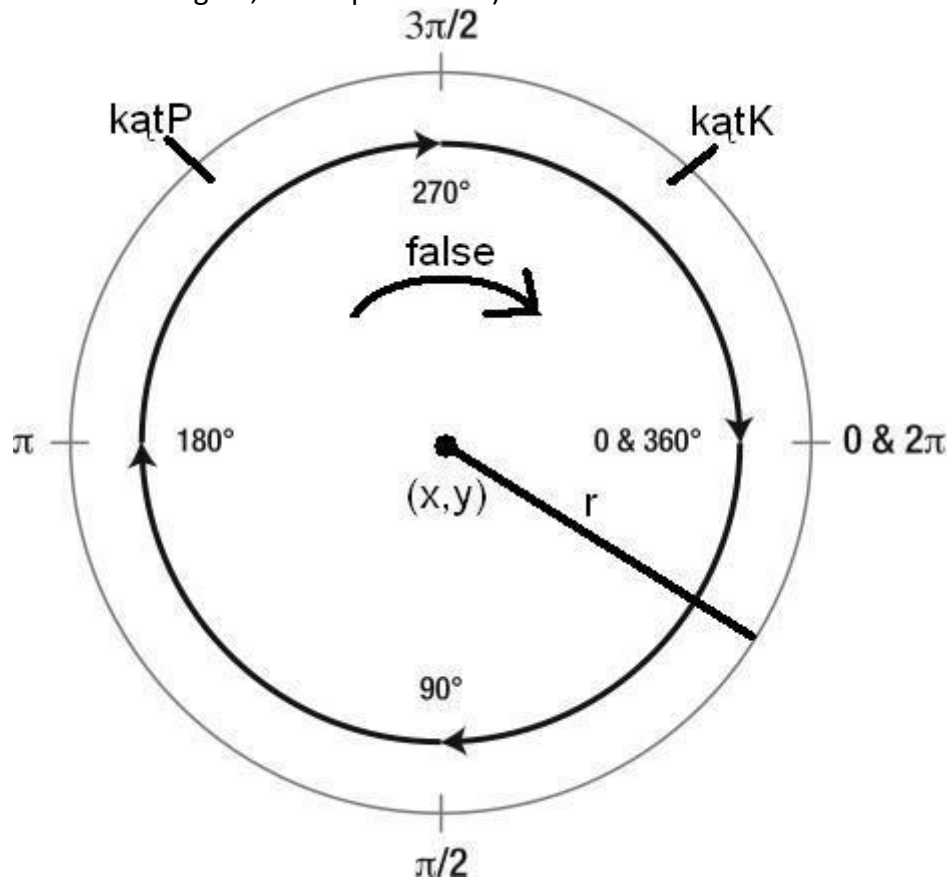
**arc(x, y, r, kątP, kątK, kierunek)**

**x, y** to współrzędne środka okręgu, na którym rysowany jest łuk;

**r** to promień okręgu,

**kątP, kątK** określają odpowiednio kąt początkowy i końcowy łuku (wartość w radianach);

**kierunek** to wartość boolowska, określająca kierunek rysowania (false – zgodny z ruchem wskazówek zegara, true – przeciwnie).



Zamiana stopni na radiany jest możliwa na pomocą następującego równania:

$miara\_w\_radianach = miara\_w\_stopniach * (Math.PI / 180)$

Math.PI to wartość liczby  $\pi$ , która określona jest w bibliotece Math języka JavaScript:



### Zadanie 17

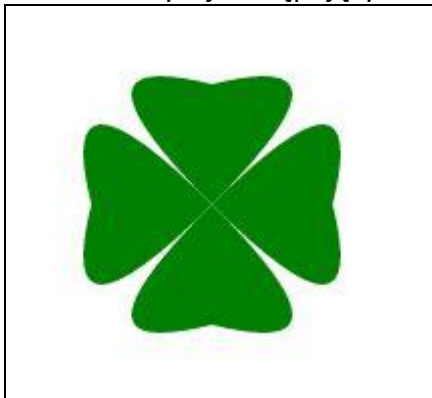
Temat: Rysowanie koniczynki z użyciem krzywych kwadratowej.

#### Wykonaj:

1. Skasuj rysowanie z poprzedniego zadania.
2. Dodaj kod do poprzedniego programu.

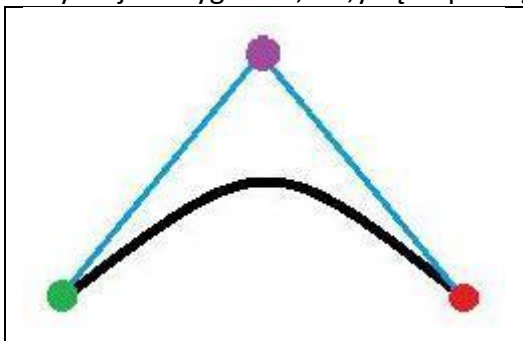
```
rodzaj.beginPath();  
rodzaj.moveTo(50,250);  
rodzaj.quadraticCurveTo(100,200,180,280);  
rodzaj.closePath();  
rodzaj.strokeStyle = "red";  
rodzaj.stroke();
```

3. Narysuj następujący obrazek:



### Rysowanie krzywej kwadratowej

**quadraticCurveTo(PktX, PktY, x, y)** – PktX, PktY są współrzędnymi punktu, wg którego krzywa jest wyginana, a x,y są współrzędnymi punktu końcowego.



fioletowe kółko to punkt o współrzędnych (PktX, PktY),  
czerwone kółko to punkt o współrzędnych (x, y),  
zielone kółko to punkt o współrzędnych ustawianych w metodzie moveTo().

## Stylowanie linii

### Zadanie 18

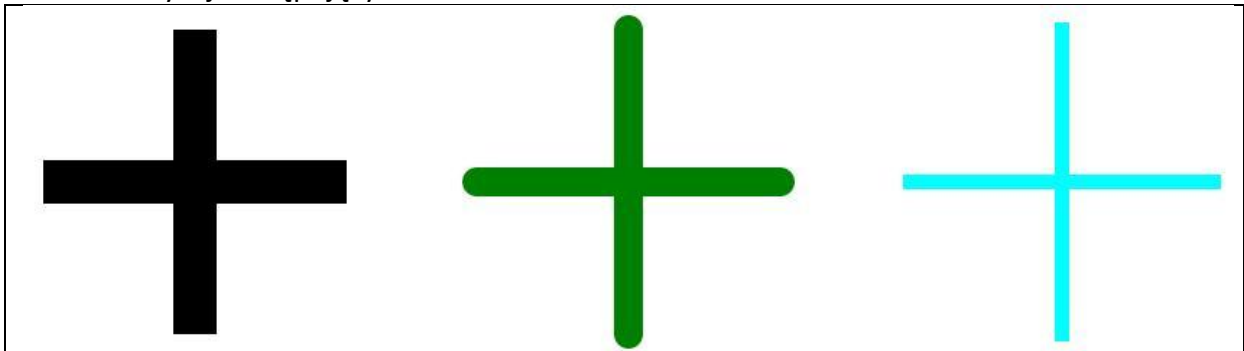
Temat: Rysowanie stylizowanych linii

#### Wykonaj:

1. Skasuj rysowanie z poprzedniego zadania.
2. Dodaj kod do poprzedniego programu.

```
rodzaj.strokeStyle = 'black';  
rodzaj.lineWidth = 20;  
rodzaj.lineCap = 'butt';  
rodzaj.beginPath();  
rodzaj.moveTo(30, 250);  
rodzaj.lineTo(30, 400);  
rodzaj.stroke();  
rodzaj.lineCap = 'round';  
rodzaj.beginPath();  
rodzaj.moveTo(70, 250);  
rodzaj.lineTo(70, 400);  
rodzaj.stroke();  
rodzaj.lineCap = 'square';  
rodzaj.beginPath();  
rodzaj.moveTo(110, 250);  
rodzaj.lineTo(110, 400);  
rodzaj.stroke();
```

3. Narysuj następujący obrazek:



### Zadanie 19

Temat: Rysowanie stylizowanych linii

#### Wykonaj:

1. Skasuj rysowanie z poprzedniego zadania.
2. Dodaj kod do poprzedniego programu.

```
rodzaj.beginPath();
rodzaj.moveTo(200, 400);
rodzaj.lineTo(250, 300);
rodzaj.lineTo(300, 400);
rodzaj.lineWidth = 20;
rodzaj.lineJoin = "miter";
rodzaj.stroke();
```

```
rodzaj.beginPath();
rodzaj.moveTo(350, 400);
rodzaj.lineTo(400, 300);
rodzaj.lineTo(450, 400);
rodzaj.lineWidth = 20;
rodzaj.lineJoin = "round";
rodzaj.stroke();
```

```
rodzaj.beginPath();
rodzaj.moveTo(500, 400);
rodzaj.lineTo(550, 300);
rodzaj.lineTo(600, 400);
rodzaj.lineWidth = 20;
rodzaj.lineJoin = "bevel";
rodzaj.stroke();
```

Zamień w ostatniej linijce parametr round na bevel, a następnie na miter i sprawdź, jak wyglądają połączenia wielokąta.

3. Narysuj następujący obrazek:  
Pagon(jeden) sierżanta.



Czyli



### Zadanie 20

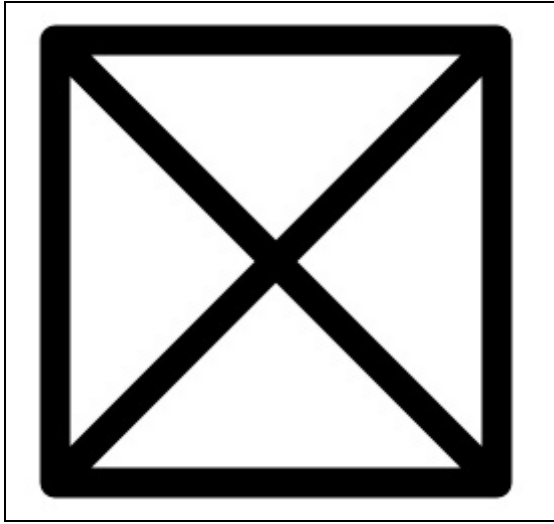
Temat: Rysowanie stylizowanych linii

Wykonaj:

1. Skasuj rysowanie z poprzedniego zadania.
2. Dodaj kod do poprzedniego programu.

```
rodzaj.beginPath();  
rodzaj.moveTo(200, 300);  
rodzaj.lineTo(300, 350);  
rodzaj.lineTo(300, 450);  
rodzaj.lineTo(200, 500);  
rodzaj.lineTo(100, 450);  
rodzaj.lineTo(100, 350);  
rodzaj.closePath();  
rodzaj.lineWidth = 20;  
rodzaj.lineJoin = "round";  
rodzaj.stroke();
```

3. Narysuj następujący obrazek:



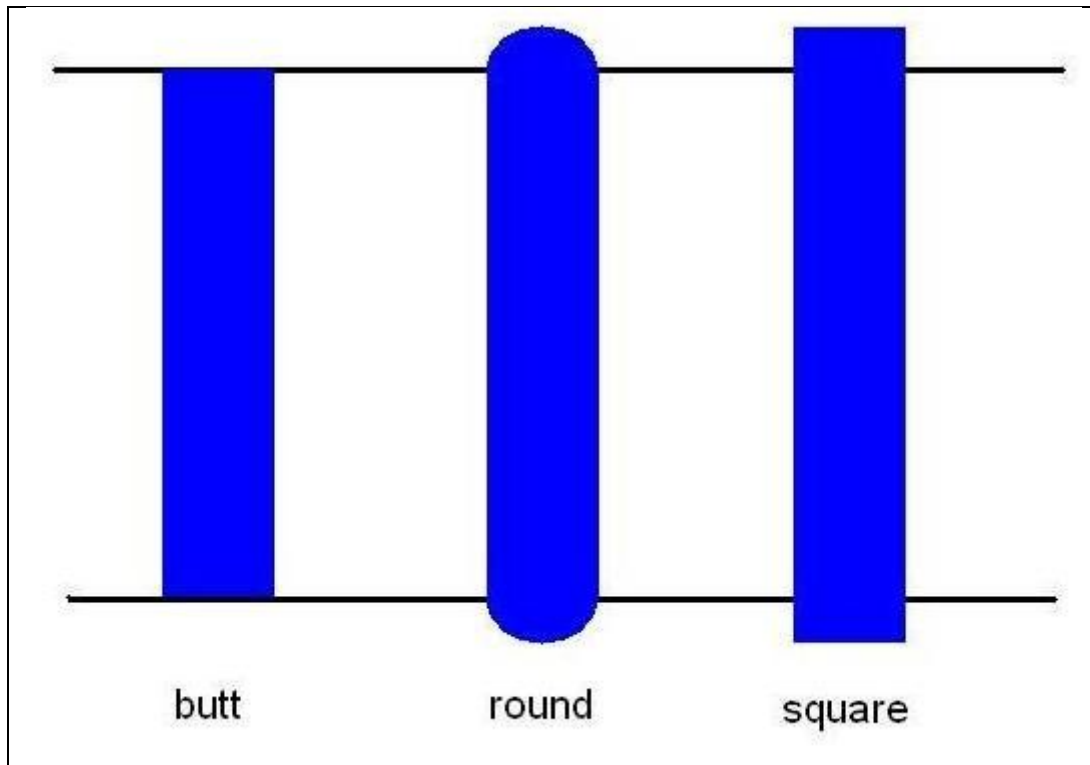
### Stylowanie linni opis teoretyczny

Canvas umożliwia określenie właściwości linii za pomocą 4 metod:

**lineWidth** = x – wartość parametru x podawana w pikselach określa grubość linii,

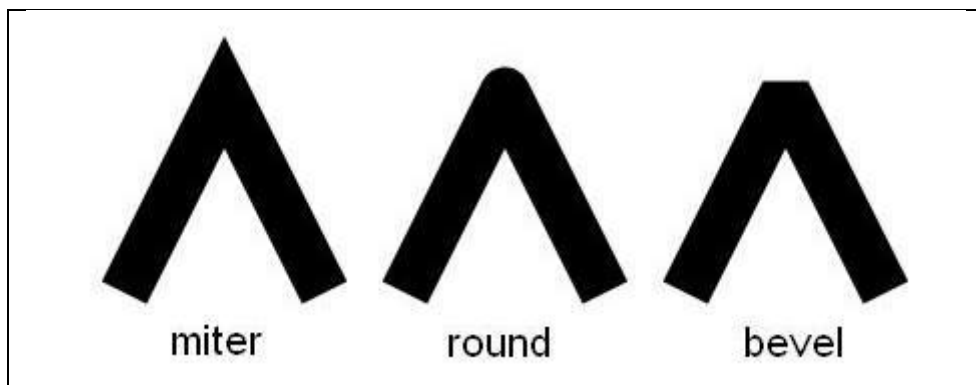
**lineCap** = typ – określa typ zakończenia linii; parametr typ przyjmuje jedną z trzech wartości:

- butt,
- round,
- square,



**lineJoin** = typ – określa styl połączenia 2 linii; parametr typ przyjmuje jedną z trzech wartości:

- round,
- bevel,
- miter,



## Import i wyrysowanie grafik

Zaczynając importowanie obrazków warto wiedzieć, że źródłem grafiki do importu może być jedynie inny element canvas lub obiekt obrazu. Jeśli masz już odnośnik do obrazka, narysuj go, używając metody `drawImage()`, którą możesz wywołać z różnymi parametrami:

**`drawImage(nazwa_obiektu, x, y),`**

gdzie:

- `nazwa_obiektu` – nazwa obiektu przechowującego obrazek,
- `x,y` – współrzędne lewego, górnego wierzchołka obrazka,

**`drawImage(nazwa_obiektu, x, y, w, h),`**

gdzie:

- `nazwa_obiektu` – nazwa obiektu przechowującego obrazek,
- `x,y` – współrzędne lewego, górnego wierzchołka obrazka,
- `w,h` – szerokość i wysokość obrazka.

**`drawImage(nazwa_obiektu, sx, sy, sw, sh, dx, dy, dw, dh),`**

gdzie:

- `nazwa_obiektu` – nazwa obiektu przechowującego obrazek,
- `sx, sy` – współrzędne lewego, górnego wierzchołka obrazka źródłowego,
- `sw, sh` – szerokość i wysokość źródłowego obrazka,
- `dx, dy` – współrzędne lewego, górnego wierzchołka nowego obrazka,
- `dw, dh` – szerokość i wysokość nowego obrazka.

Obiekt obrazka możesz utworzyć za pomocą następującego kodu:

```
var nazwa_obiektu = new Image();
nazwa_obiektu.src = "adres_obrazka.jpg";
image.onload = function () { rodzaj.drawImage(nazwa_obiektu, 0, 0); };
```

gdzie:

pierwsza linia odpowiedzialna jest za utworzenie nowego obiektu (metoda `Image()`) i przypisaniu mu nazwy "`nazwa_obiektu`",  
druga linia odpowiedzialna jest za ustawienie adresu do obrazka (`adres_obrazka.jpg`), dzięki wykorzystaniu właściwości `src` wcześniej utworzonego obiektu,  
trzecia linijka umożliwia wyświetlenie obrazka po jego pobraniu, brak `image.onload` spowoduje, że większość przeglądarek nie wyświetli poprawnie obrazka.



```

<!DOCTYPE html>
<html lang="pl">
<head>
  <meta charset="utf-8">
  <title>Canvas</title>
  <script type="text/javascript">
    function rysuj() {
      var rysowanie = document.getElementById('obrazek');
      var rodzaj = rysowanie.getContext('2d');
      var obraz_arbuz = new Image();

      obraz_arbuz.src = "arbuz.jpg";
      obraz_arbuz.onload = function () { rodzaj.drawImage(obraz_arbuz, 100,100); }
      rodzaj.font = "30pt Times New Roman";
      rodzaj.fillStyle = "blue";
      rodzaj.fillText("Am-am Arbuz", 10, 50);
    }
  </script>
</head>
<body onload="rysuj();">
  <canvas id="obrazek" width="600" height="800">
    Twoja przeglądarka nie wspiera Canvas! Zainstaluj Internet Explorer 9.x
  </canvas>
</body>
</html>

```

### Zadanie 21

Wykonaj stronę wyświetlającą, co najmniej cztery pliki graficzne z opisami odpowiadającymi grafice. Grafiki możesz ściągnąć o różnych wymiarach ( szerokość, wysokość), ale poprzez skalowanie uzyskaj stosunek długości do szerokości 16/9 oraz grafiki powinny być tej samej wielkości po skalowaniu, dostosuj większe grafiki do najmniejszej( wymiary). Opis umieść pod grafiką. Opisy różnymi czcionkami, różną wielkością czcionek. Całość ma tworzyć śmieszną przyzwoita historyjkę. Pliki pobierz z Internetu. Opisy wymyśl sam. Obok grafik wyświetl miniatury grafik wykonane z użyciem instrukcji (drawImage(nazwa\_obiektu, x, y, w, h→skalowanie obrazka).

o wymiarze 50+miesiąc\_urodzenia\*50+miesiąc\_urodzenia.

Do miniatur dodaj cienie a do dwóch grafik przeźroczystość.

### Ułożenie obrazków i miniatur:

1)(Dzień\_urodzenia mod 4=0)→ Mintury po prawej zdjęć nie styka się ze zdjęciem


2) Dzień\_urodzenia mod 4=1)→ Mintury po prawej zdjęć nie styka się ze zdjęciem


3)(Dzień\_urodzenia mod 4=2)→ Mintury po lewej zdjęć nie styka się ze zdjęciem


4) Dzień\_urodzenia mod 4=3)→ Mintury po lewej zdjęć nie styka się ze zdjęciem


### **Skalowanie obrazka**

Do zmiany rozmiaru obrazka, wykorzystaj drugą z metod drawImage, podanych powyżej:

drawImage(nazwa\_obiektu, x, y, w, h).

Skalowanie obrazka rozpocznij od wstawienia go na stronę za pomocą znacznika <img>. Następnie pobierz jego id i wyświetl go na elemencie canvas, zmniejszonego do rozmiaru 100x100 pikseli.

Ustaw kursor za znacznikiem body i dodaj nową linię.

Wpisz następujący kod:

```

```

Następnie ustaw kursor za znacznikiem z metodą getContext() i dodaj nową linię.

Wpisz następujący kod:

```
var iWielblad = document.getElementById("wielblad");  
rodzaj.drawImage(iWielblad, 350, 0, 100, 100);
```

### **Krojenie grafiki**

Do przycięcia grafiki korzystaj z trzeciej z zaprezentowanych metod drawImage():

drawImage(nazwa\_obiektu, sx, sy, sw, sh, dx, dy, dw, dh).

Ustaw kursor za znacznikiem z metodą `getContext()` i dodaj nową linię.

Wpisz następujący kod:

```
rodzaj.drawImage(iWielblad, 215, 10, 50, 45, 500, 0, 250, 225);
```

### **Przezroczystość**

Do określenia przezroczystości możesz wykorzystać następujące metody:

`globalAlpha = x` – gdzie `x` to liczba z zakresu `<0,1>` gdzie `0` – całkowita przezroczystość, `1` – nieprzezroczysty,

`rgba(r,g,b,a)` – gdzie `"r"`, `"g"`, `"b"` to wartości kolorów w formacie `rgb`, natomiast `"a"` oznacza przezroczystość i przyjmuje wartości jak w `globalAlpha`.

Ustaw kursor za znacznikiem z metodą `getContext()` i dodaj nową linię.

Wpisz następujący kod:

```
var image = new Image();
image.src = "wielblad.jpg";
image.onload = function () {
    rodzaj.drawImage(image, 0, 0);
    rodzaj.fillRect(0,0,100,100);
    rodzaj.globalAlpha = 0.5;
    rodzaj.fillRect(110,0,100,100);
//przezroczystość
};
//linie
```

Teraz, gdy potrafisz już użyć metody `globalAlpha()`, spróbuj wykonać trudniejszą grafikę, która wykorzystuje przezroczystość.

```
rodzaj.drawImage(image, 350, 0);
rodzaj.fillStyle = "white";
for (var i = 0; i < 1; i += 0.04) {
    rodzaj.globalAlpha = i;
    rodzaj.fillRect(350, i * 225, 350, 10); }
```

### **Cień w Cavas**

Do określenia cienia możesz wykorzystać następujące właściwości:

`shadowColor` – określa kolor cienia, taką samą wartość jak dla kolorów w CSS.

shadowBlur – określa rozmycie cienia w pikselach, im niższa wartość rozmycia, tym ostrzejsze są cienie. Standardowa wartość wynosi 0.

shadowOffsetX i shadowOffsetY – określa przesunięcie cienia w pikselach, wartości dodatnie przesuwają cień na dół i prawo, ujemne do góry i w lewo. Standardowa wartość wynosi 0.

W funkcji draw zastąp pierwsze wystąpienie polecenia rodzaj.drawImage(image, 0, 0); za pomocą następującego kodu:

```
rodzaj.shadowOffsetX = 10;
rodzaj.shadowOffsetY = 10;
rodzaj.shadowBlur = 4;
rodzaj.shadowColor = 'rgba(0, 0, 0, 0.5)';
rodzaj.drawImage(image, 0, 0);
rodzaj.shadowOffsetX = 0;
rodzaj.shadowOffsetY = 0;
rodzaj.shadowBlur = 0;
```

Tworząc bardziej skomplikowane rysunki użyteczne są następujące dwie metody:

**Save()** – zapisuje stan canvas,

**Restore()** – przywraca stan canvas z momentu zapisu za pomocą metody save().

Metody te zostaną użyte w poniższych przykładach w celu ukazania różnic przed i po transformacjach.

### Zmiana początku układu współrzędnych

Zmiana początku układu współrzędnych, względem którego odbywa się rysowanie (domyślnie lewy-górny wierzchołek elementu canvas) jest bardzo prosta, wystarczy tylko wykorzystać metodę:

**Translate(x,y)** – zmienia początek układu współrzędnych na punkt (x,y).

1. Ustaw kursor za znacznikiem z metodą getContext() i dodaj nową linię.

2. Wpisz następujący kod:

```
rodzaj.save();
rodzaj.translate(150,150);
rodzaj.fillRect(0,0,100,100);
rodzaj.restore();
rodzaj.fillStyle = "red";
rodzaj.fillRect(0,0,100,100);
```

Metoda save() zapisuje na stosie stan początkowy canvas, następnie musisz zmienić punkt, względem którego nastąpi rysowanie o wektor [150,150]. Najpierw musisz narysować kwadrat, przywrócić ustawienia domyślne i ponownie narysować kwadrat, który znajduje się

już w innym miejscu, gdyż metoda `restore()` przywraca punkt początku układu współrzędnych z punktu (150,150) na punkt (0,0).

### Zmiana jednostki

Canvas umożliwia zmianę wielkości domyślnej jednostki z jednego piksela na większą lub mniejszą. W tym celu użyteczna jest kolejna z metod transformacji:

**scale(x,y)** – gdzie x i y zwiększają/zmniejszają skalę odpowiednio w poziomie i pionie, np. `scale(2,2)` zwiększy domyślny rozmiar jednostki w pionie i poziomie dwukrotnie.

1. Ustaw kursor za ostatnio dodanym poleceniem i dodaj nową linię.
2. Wpisz następujący kod:  

```
rodzaj.save();  
rodzaj.fillStyle = "green";  
rodzaj.fillRect(150,0,100,100);  
rodzaj.scale(0.5,0.5);  
rodzaj.fillStyle = "yellow";  
rodzaj.fillRect(300,0,100,100);  
rodzaj.restore();
```
3. Zapisz plik `index.html`. Otwórz go w przeglądarce i sprawdź, czy wygląda jak na Rys. 2.

### Zmiana kąta rysowania

W celu narysowania obiektu obróconego o podany kąt, wykorzystaj metodę:

**Rotate(kąt)** – gdzie argument kąt podawany jest w radianach. W celu zrozumienia, w jaki sposób podawać kąt w radianach lub jak zamienić radiany na stopnie, zajrzyj do samouczka pt. `Kształty w Canvas`.

1. Ustaw kursor za ostatnio dodanym poleceniem i dodaj nową linię.
2. Wpisz następujący kod:  

```
rodzaj.save();  
rodzaj.rotate(Math.PI/4);  
rodzaj.fillStyle = "green";  
rodzaj.fillRect(130, -40, 100, 100);  
rodzaj.restore();  
rodzaj.fillStyle = "yellow";  
rodzaj.fillRect(0,150,100,100);
```
3. Zapisz plik `index.html`. Otwórz go w przeglądarce i sprawdź, czy wygląda tak, jak na

### Podstawowe kompozycje

Tworzenie kompozycji wiąże się z jedną metodą:

`globalCompositeOperation=typ` – gdzie typ może przybierać jedną z wartości zaprezentowanych na Rys. 2.

Ustaw kursor za znacznikiem z metodą `getContext()` i dodaj nową linię.

Wpisz:

```
rodzaj.globalCompositeOperation = "source-over";
rodzaj.fillStyle = "blue";
rodzaj.fillRect(0, 0, 100, 100);
rodzaj.fillStyle = "red";
rodzaj.fillRect(50, 50, 100, 100);
rodzaj.fillStyle = "blue";
rodzaj.globalCompositeOperation = "destination-over";
rodzaj.fillRect(200, 0, 100, 100);
rodzaj.fillStyle = "red";
rodzaj.fillRect(250, 50, 100, 100);
```

### **Łączenie kompozycji**

Wiesz już jak stosować kompozycje, teraz spróbuj dodać inne kompozycje.

Ustaw kursor za ostatnio dodanym kodem i dodaj nową linię.

Wpisz:

```
rodzaj.globalCompositeOperation = "source-in";
rodzaj.fillStyle = "blue";
rodzaj.fillRect(0, 200, 100, 100);
rodzaj.fillStyle = "red";
rodzaj.fillRect(50, 250, 100, 100);
```

### **Informacja**

Kompozycja typu "source-in" pokaże element przykrywający tylko w obszarze elementu przykrywanego. Zmieniając typ kompozycji w canvas, musisz pamiętać, że część z nich usuwa fragmenty obrazu.

Zmień "source-in" na "lighter".

Ustaw kursor za ostatnio dodanym kodem i dodaj nową linię.

Wpisz:

```
rodzaj.globalCompositeOperation = "xor";
rodzaj.fillStyle = "blue";
rodzaj.fillRect(200, 200, 100, 100);
rodzaj.fillStyle = "red";
rodzaj.fillRect(250, 250, 100, 100);
```

## Umieszczanie tekstu w Canvas

### Zadanie 22

Temat: Teksty.

Wykonaj:

1. Wykonaj następujące teksty.



Do tworzenia podstawowych napisów wykorzystaj następujące metody:

`fillText("napis", x, y)` – umieszcza napis o treści określonej jako pierwszy argument, rozpoczynając pisanie od punktu (x,y),  
`fillStyle = "kolor"` – określa kolor tekstu,  
`strokeText("napis", x, y)` – umieszcza obramowanie napisu o treści określonej w pierwszym argumencie w punkcie (x,y), wykorzystując `strokeStyle`,  
`strokeStyle = "kolor"` – określa kolor obramowania czcionki,  
`font = "Xpt FontFamily"` – określa wielkość i font. X jest wartością liczbową, określającą wielkość fontu, a FontFamily nazwą fontu np. "40pt Consolas".

Ustaw kursor za znacznikiem z metodą `getContext()` i dodaj nową linię.

Wpisz:

```
rodzaj.font = "30pt Times New Roman";
```

```
rodzaj.fillStyle = "blue";
rodzaj.fillText("Witaj", 10, 50);
rodzaj.font = "30pt Arial";
rodzaj.strokeStyle = "red";
rodzaj.strokeText("CANVAS", 110, 50);
```

### Pomiar narysowanego tekstu

Na pewno zauważyłeś w poprzednim przykładzie, że aby połączyć style – zastosować różne fonty, kolory, wielkości, należy ręcznie określać położenie kolejnych fragmentów tekstu. Jeśli chcesz zrobić to automatycznie, musisz wykorzystać metodę `measureText`. Umożliwia ona pomiar tekstu przy aktualnych ustawieniach:

`measureText("napis").width` – zwraca długość napisu w pikselach przy aktualnych ustawieniach.

Ustaw kursor za ostatnio dodanym kodem i dodaj nową linię.

Wpisz:

```
rodzaj.font = "30pt Times New Roman";
rodzaj.fillStyle = "blue";
rodzaj.fillText("Yo", 10, 100);
var len = rodzaj.measureText("Yo").width;
rodzaj.font = "30pt Arial";
rodzaj.strokeStyle = "red";
rodzaj.strokeText("CANVAS", len + 20, 100);
```

### Wyrównanie tekstu

Tekst możesz umiejscowić dowolnie, względem punktu odniesienia. Do tego celu służą dwie metody:

**`textAlign=[X]`** – określa wyrównanie tekstu w poziomie względem punktu (x,y). Wartość elementu X można ustawić na: "start", "end", "left", "right", "center" (domyślnie: "start"),

**`textBaseline=[X]`**; – określa wyrównanie tekstu w pionie. Wartość elementu X można ustawić na: "top", "hanging", "middle", "alphabetic", "ideographic", "bottom" (domyślnie: "alphabetic").

Ustaw kursor za ostatnio dodanym kodem i dodaj nową linię.

Wpisz:



```
rodzaj.arc(100, 150, 2, 2 * Math.PI, 0, true);
rodzaj.textAlign = "left";
rodzaj.fillStyle = "blue";
rodzaj.fillText("left", 100, 150);
rodzaj.arc(100, 200, 2, 2 * Math.PI, 0, true);
rodzaj.textAlign = "right";
rodzaj.fillText("right", 100, 200);
rodzaj.arc(100, 250, 2, 2 * Math.PI, 0, true);
rodzaj.textAlign = "center";
rodzaj.fillText("center", 100, 250);
rodzaj.arc(100, 300, 2, 2 * Math.PI, 0, true);
rodzaj.textBaseline = "middle";
rodzaj.fillText("middle", 100, 300);
rodzaj.arc(100, 400, 2, 2 * Math.PI, 0, true);
rodzaj.textBaseline = "bottom";
rodzaj.fillText("bottom", 100, 400);
rodzaj.textBaseline = "top";
rodzaj.fillText("top", 100, 400);
rodzaj.fillStyle = "black";
rodzaj.fill();
```

## Strona końcowa

Poniżej możesz zobaczyć końcowy kod strony HTML:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Tekst w Canvas</title>
    <script type="text/javascript">
      function draw() {
        var canvas = document.getElementById('image1');
        var rodzaj = canvas.getContext('2d');
        rodzaj.font = "30pt Times New Roman";
        rodzaj.fillStyle = "blue";
        rodzaj.fillText("Witaj", 10, 50);
        rodzaj.font = "30pt Arial";
        rodzaj.strokeStyle = "red";
        rodzaj.strokeText("CANVAS", 110, 50);
        rodzaj.font = "30pt Times New Roman";
        rodzaj.fillStyle = "blue";
        rodzaj.fillText("Yo", 10, 100);
        var len = rodzaj.measureText("Yo").width;
        rodzaj.font = "30pt Arial";
        rodzaj.strokeStyle = "red";
        rodzaj.strokeText("CANVAS", len + 20, 100);
        rodzaj.arc(100, 150, 2, 2 * Math.PI, 0, true);
        rodzaj.textAlign = "left";
        rodzaj.fillStyle = "blue";
        rodzaj.fillText("left", 100, 150);
        rodzaj.arc(100, 200, 2, 2 * Math.PI, 0, true);
        rodzaj.textAlign = "right";
        rodzaj.fillText("right", 100, 200);
        rodzaj.arc(100, 250, 2, 2 * Math.PI, 0, true);
        rodzaj.textAlign = "center";
        rodzaj.fillText("center", 100, 250);
        rodzaj.arc(100, 300, 2, 2 * Math.PI, 0, true);
        rodzaj.textBaseline = "middle";
        rodzaj.fillText("middle", 100, 300);
        rodzaj.arc(100, 400, 2, 2 * Math.PI, 0, true);
        rodzaj.textBaseline = "bottom";
```

```
        rodzaj.fillText("bottom", 100, 400);
        rodzaj.textBaseline = "top";
        rodzaj.fillText("top", 100, 400);
        rodzaj.fillStyle = "black";
        rodzaj.fill();
    }
</script>
</head>
<body onload="draw();">
    <canvas id="image1" width="300" height="450">
        Twoja przeglądarka nie wspiera elementu canvas!
        Pobierz Internet Explorer 9 lub nowszą wersję!
    </canvas>
</body>
</html>
```

## Zadanie 23

Temat: Animowana piłka (tor lotu góra dół) zmieniającą kolor.

### Teoria

#### Uwaga1

**Animacja** w elemencie canvas polega na cyklicznym przerysowywaniu całości obrazka z uaktualnionymi obiektami.

Animacja składa się z następujących kroków:

1. usunięcie poprzedniego rysunku,
2. zapisanie stanu elementu canvas (jest to używane w przypadku, kiedy w kolejnych krokach zmieniamy style lub transformujemy element canvas),
3. wykonanie operacji przeliczenia nowych współrzędnych elementu i ponowne narysowanie wszystkich elementów na stronie,
4. przywrócenie stanu elementu canvas (jak w pkt. 2.).

#### Uwaga2

Do cyklicznego przerysowywania obrazka możesz wykorzystać dwie funkcje języka JavaScript:

**setInterval(funkcja, czas)** - funkcja jest nazwą funkcji, która ma być wywołana co czas podany w milisekundach (1s=1000ms); możesz zatrzymać jej cykliczne wywoływanie za pomocą funkcji clearInterval(),

zastosowanie: Funkcja setInterval jest używana w sytuacji, kiedy zachodzi potrzeba wykonania operacji w dokładnych odstępach czasowych, ponieważ nie czeka ona na zakończenie operacji, a jedynie wykonuje cyklicznie zadanie. Zapewne bardzo często spotkasz się z potrzebą zapewnienia cykliczności wykonywanych operacji. Wówczas użyjesz funkcji setInterval.

**setTimeout(funkcja, czas)** - funkcja jest nazwą funkcji, która ma być wywołana po czasie określonym w parametrze czas, podanym w milisekundach (1s=1000ms).

zastosowanie: Funkcja setTimeout jest wywoływana na końcu operacji i na czas pomiędzy kolejnym wywołaniem funkcji wpływa czas jej realizacji.

### Przygotowanie strony

#### Uwaga1:

Przygotuj szablon i obramuj element canvas kolorem niebieskim, tak, aby zaznaczyć obszar elementu. W edytorze HTML utwórz nowy plik pod nazwą index.html i wklej do niego szkielet strony:

```

<!DOCTYPE html>
<html lang="pl">
  <head>
    <meta charset="utf-8">
    <title>Animacja w Canvas</title>
    <script type="text/javascript">
      var canvas
      var rodzaj;
      function draw(){
        canvas = document.getElementById('image1');
        rodzaj = canvas.getContext('2d');
      }
    </script>
    <style type="text/css">
      canvas { border: 1px solid blue; }
    </style>
  </head>
  <body onload="draw();">
    <canvas id="image1" width="400" height="400">
      Twoja przeglądarka nie wspiera elementu canvas!
      Pobierz Internet Explorer 9 lub nowszą wersję!
    </canvas>
  </body>
</html>

```

### Uwaga2

Przygotuj:

- zmienne przechowujące aktualne współrzędne piłki (x,y),
- przesunięcie zmiennej y (dy),
- paletę kolorów (kolory)
- aktualny kolor (i).

Za linią var rodzaj; dodaj:

```

var x = 200;
var y = 20;
var dy = 10;
var kolory = new
Array("blue","red","yellow","green");
var i = 0;

```

Zapisz zmiany w pliku index.html i otwórz go w przeglądarce wspierającej element canvas,

## Przygotowanie animacji

1)Utwórz nową funkcję, umieszczoną w funkcji draw, oraz cykliczne jej wywołanie za pomocą setInterval. Czyli ustaw kursor za znacznikiem </script>i dodaj nową linię.

Wpisz:

```
setInterval(anim, 100);
```

2)Ustaw kursor przed znacznikiem </script>i dodaj nową linię. Wpisz:

```
function anim() {  
    rodzaj.clearRect(0, 0, canvas.width, canvas.height);  
}
```

3)Narysuj piłkę – koło o promieniu 20 pikseli w punkcie określonym przez zmienne x i y. Koło ma zostać wypełnione kolorem z listy (kolory) na pozycji określonej przez wartość zmiennej i. Poniżej metody clearRect wpisz:

```
rodzaj.beginPath();  
rodzaj.fillStyle = kolory[i];  
rodzaj.arc(x, y, 20, 0, Math.PI * 2, true);  
rodzaj.closePath;  
rodzaj.fill();
```

## Animacja piłki

Teraz, gdy wiesz już jak narysować piłkę, nadszedł czas na jej animację. Animacja będzie polegała na ruchu piłki w dół, następnie odbiciu od krawędzi elementu canvas i ruchu do góry, aż do krawędzi, gdzie piłka ponownie się odbije. Przy każdym odbiciu piłka będzie cyklicznie zmieniała kolor.

Dodaj zmianę pozycji piłki.

Poniżej metody clearRect wpisz:

```
y += dy;
```

Zapisz stronę index.html i odśwież ją w przeglądarce. Sprawdź, jak zachowuje się piłka.

## Informacja

Piłka przesuwa się z góry na dół, ale nie odbija się od krawędzi. Aby mogła się odbić, musisz ograniczyć jej ruch do pola elementu canvas. Pole to ma wysokość określoną przez wartość canvas.height. Należy również uwzględnić promień piłki. Aby aby nie „wychodziła” za pole, musisz ograniczyć jej ruch od 20 do (canvas.height-20) pikseli.

Ogranicz ruch piłki do od 20 do (canvas.height-20) pikseli.

Za poprzednio dodaną linijką wpisz:

```
if (y > (canvas.height-20)) {  
    dy = -10;  
}
```

```
if (y < 20) {  
    dy = 10;  
}
```

Zapisz stronę index.html i odśwież ją w przeglądarce. Sprawdź, jak zachowuje się piłka.

### Informacja

Piłka przesuwa się z góry na dół i odbija się od krawędzi. Aby możliwa była zamiana koloru, musisz po każdym odbiciu od górnej lub dolnej krawędzi zmienić kolor wypełnienia – wartość zmiennej i od 0 do ilości elementów w tablicy kolorów.

Zmień cyklicznie kolor po odbiciu od krawędzi.

Za linijką `dy = -10;` dodaj :

```
i = ++i % kolory.length;
```

Za linijką `dy = 10;` dodaj :

```
i = ++i % kolory.length;
```

Zapisz stronę index.html i odśwież ją w przeglądarce. Sprawdź, jak zachowuje się piłka.

### Strona końcowa

Poniżej możesz zobaczyć końcowy kod strony HTML:

```
<!DOCTYPE html>  
<html lang="pl">  
  <head>  
    <meta charset="utf-8">  
    <title>Animacja w Canvas</title>  
    <script type="text/javascript">  
      var canvas  
      var rodzaj;  
      var x = 200;  
      var y = 20;  
      var dy = 10;  
      var kolory = new Array("blue", "red", "yellow", "green");  
      var i = 0;
```

```

function draw() {
    canvas = document.getElementById('image1');
    rodzaj = canvas.getContext('2d');
    setInterval(anim, 100);
}

function anim() {
    rodzaj.clearRect(0, 0, canvas.width, canvas.height);
    y += dy;
    if (y > (canvas.height - 20)) {
        dy = -10;
        i = ++i % kolory.length;
    }
    if (y < 20) {
        dy = 10;
        i = ++i % kolory.length;
    }
    rodzaj.beginPath();
    rodzaj.fillStyle = kolory[i];
    rodzaj.arc(x, y, 20, 0, Math.PI * 2, true);
    rodzaj.closePath();
    rodzaj.fill();
}
</script>
<style type="text/css">
    canvas { border: 1px solid blue; }
</style>
</head>
<body onload="draw();">
    <canvas id="image1" width="400" height="400">
        Twoja przeglądarka nie wspiera elementu canvas!
        Pobierz Internet Explorer 9 lub nowszą wersję!
    </canvas>
</body>
</html>

```



**Zadanie 24**

Temat: Gra zaproponowana przez jednego z uczniów o nazwie Karol\_game.

Zasady gry:

Odbijamy piłkę do góry, każde odbicie przyspiesz ruch piłki. Licznik zlicza ilość odbić.

## Zdarzenia

<http://webmaster.helion.pl/index.php/kursjs-obsługa-zdarzeń-i-elementów-strony/kursjs-zdarzenia>

Z JavaScriptem i DHTML-em nieodłącznie związane jest pojęcie zdarzeń. Zdarzenie to przesunięcie kursora myszy, kliknięcie, wczytanie strony, opuszczenie strony itp. Każdy element strony ma przypisany zestaw obsługiwanych przez niego zdarzeń, a każdemu z nich można przypisywać własne procedury obsługi, napisane w JavaScriptcie.

Procedurę obsługi można przypisać do danego zdarzenia na dwa sposoby.

Jeśli jest ona bardzo krótka, np. składa się tylko z jednej, dwóch instrukcji, i może być zapisana w jednej linii kodu, można ją przypisać bezpośrednio do znacznika obsługującego dane zdarzenie:

### Sposób 1

`<znacznik zdarzenie="instrukcja;">`

### Sposób 2

W przypadku dłuższych procedur obsługi zdarzeniu należy przypisać jedynie wywołanie funkcji JavaScript:

`<znacznik zdarzenie="nazwa_funkcji();">`

natomiast w kodzie strony (z reguły w sekcji head) umieścić samą funkcję. Całość miałaby zatem schematyczną postać:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta http-equiv="Content-Script-Type" content="text/javascript">
    <title>Moja strona WWW</title>
    <script type="text/javascript">
      function nazwa_funkcji()
      {
        //tutaj kod funkcji
      }
    </script>
  </head>
  <body>
    <znacznik zdarzenie="nazwa_funkcji();">
  </body>
</html>
```

Przykładowy znacznik obsługujący zdarzenie miałby postać:

`<p onclick="pClicked();">`

### Lista typowych zdarzeń.

Nazwa zdarzenia	opis	Elementy HTML obsługujące zdarzenie
onAbort	Zdarzenie, które powstaje, kiedy zostanie przerwane ładowanie obrazu.	img
onBlur	Zdarzenie, które powstaje, kiedy element traci fokus.	większość elementów strony

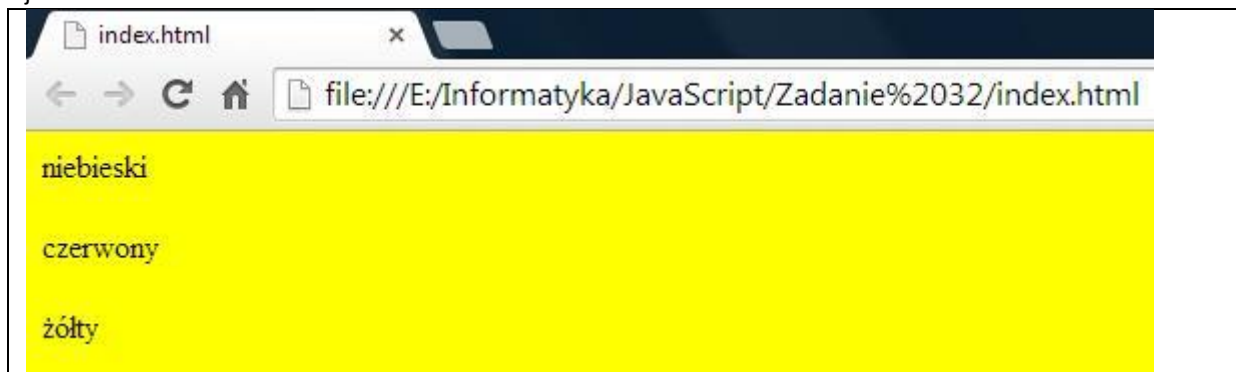
onChange	Zdarzenie, które powstaje, kiedy element traci fokus i jednocześnie zmienia się wartość zawarta w tym elemencie (np. polu tekstowym).	input, select, textarea
onClick	Zdarzenie, które powstaje, kiedy element został kliknięty	większość elementów strony
onDblclick	Zdarzenie, które powstaje, kiedy element został dwukrotnie kliknięty.	większość elementów strony
onError	Zdarzenie, które powstaje, kiedy w trakcie ładowanie obrazu wystąpi błąd.	Img
Onfocus	Zdarzenie, które powstaje, kiedy element otrzymuje fokus.	większość elementów strony
onkeydown	Zdarzenie, które powstaje, kiedy naciśnięty zostanie klawisz klawiatury.	większość elementów strony
onkeypress	Zdarzenie, które powstaje, kiedy klawisz klawiatury zostanie naciśnięty i puszczony.	większość elementów strony
Onkeyup	Zdarzenie, które powstaje, kiedy klawisz klawiatury zostanie puszczony.	większość elementów strony
onload	Zdarzenie, które powstaje, kiedy przeglądarka zakończy ładowanie strony lub ramki.	body, frameset
onmousedown	Zdarzenie, które powstaje, kiedy klawisz myszy zostanie naciśnięty nad elementem.	większość elementów strony
onMouseMove	Zdarzenie, które powstaje, kiedy kursor myszy jest przesuwany nad elementem.	większość elementów strony
onMouseOut	Zdarzenie, które powstaje, kiedy kursor myszy opuści obszar elementu.	większość elementów strony
onMouseOver	Zdarzenie, które powstaje, kiedy kursor myszy wejdzie w obszar elementu.	większość elementów strony
onmouseup	Zdarzenie, które powstaje, kiedy klawisz myszy zostanie zwolniony nad elementem.	większość elementów strony
on reset	Zdarzenie, które powstaje podczas resetowania (wywołanie w kodzie metody reset, kliknięcie przycisku reset) formularza.	Form
Onresize	Zdarzenie, które powstaje, gdy zmieni się rozmiar okna.	body, frameset
Onselect	Zdarzenie, które powstaje podczas zaznaczania fragmentu tekstu.	input (text, textarea)
onsubmit	Zdarzenie, które powstaje podczas wysyłania (wywołanie w kodzie metody submit, kliknięcie przycisku submit) formularza.	Form
onUnload	Zdarzenie, które powstaje, kiedy przeglądarka usuwa bieżący dokument.	body, frameset

### Zadanie 32

Napisać stronę używając zdarzenia (onMouseOver) oraz zdefiniowanych funkcji, która będzie zmieniać kolor tła. Wykorzystaj metodę .bgColor=kolor; dla obiektu dokument.

Nazwa funkcji:

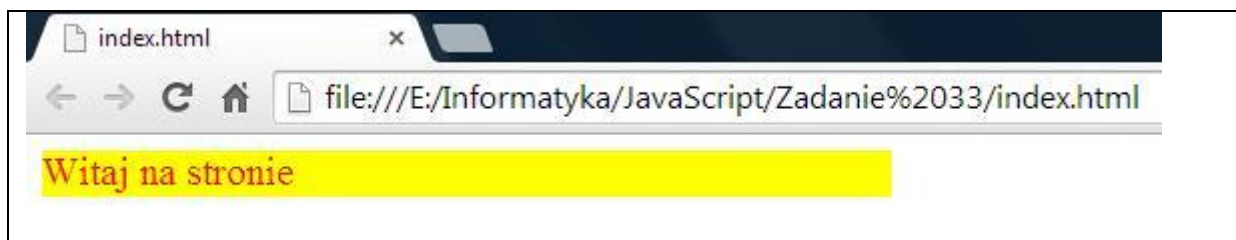
```
function kolor_tla_nazwisko_ucznia(kolor)
{
    treść funkcji;
}
```



Dopisać dokładne rozwiązanie

### Zadanie 33

Napisać stronę używając zdarzenia onMouseMove oraz onMouseOut, która będzie po najechaniu myszką zmieniać kolor tła i napis (kolor i wielkość czcionki).

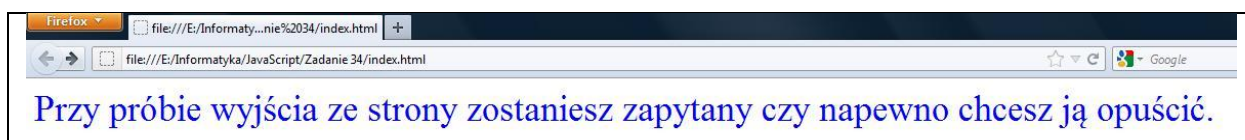


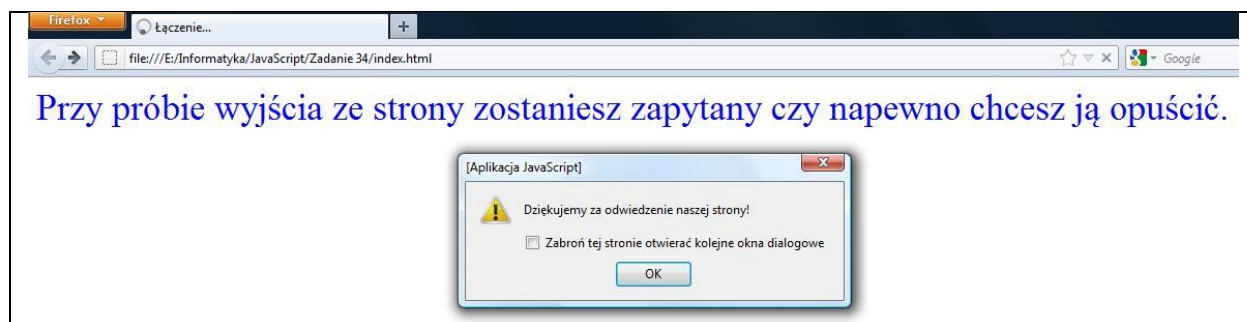
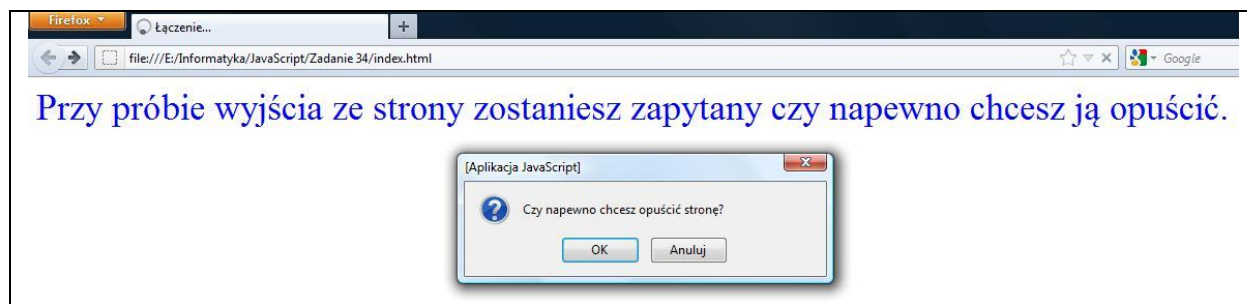
Dopisać dokładne rozwiązanie

### Zadanie 34

Napisać stronę używając zdarzenia onUnload która przy próbie wyjścia ze strony pytało się o potwierdzenie. Gdy nie będziesz chciał opuścić strony do użyj funkcji history.back();

Po opuszczeniu uruchom jeszcze okno Alert z napisem Dziękujemy za odwiedzenie naszej strony.





Dopisać dokładne rozwiązanie

## Uruchamianie program Windows ze strony

Nasza strona tematycznie nie powinna tak bardzo odbiegać jednak wątek uruchamiania programów środowiska Windows za pomocą Apletów Java stanowi dość spore zainteresowanie.

Alternatywnym rozwiązaniem dla rozszerzenia zasięgu przeglądarki internetowej (realizacji możliwości uruchamiania w/w procesów) może być wykorzystanie użycie JavaScript.

Nasz przykład zawiera funkcję odczytu typu przeglądarki oraz skrypty dla uruchomienia konkretnych programów systemu operacyjnego wprowadzone za pomocą parametru:

```
<html>
<head>
  <title></title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<p>
Test uruchomienia aplikacji za pomocą JavaScript.</p>
Skrypty testowane na IE oraz FireFox.</p>
Polecenie : <input id = "polecenie" type="text" value="C:windowsnotepad.exe"></input><br>
<input type="button" width="15" value="Uruchom" onclick="RunExe();" /></input></p>
<script type="JavaScript">
</script>
<script type="text/javascript">
function RunExe()
{
  var bname = navigator.appName;
  var pol = document.getElementById('polecenie');
  if (bname == "Netscape")
  {
    netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
    var exe =
window.Components.classes['@mozilla.org/file/local;1'].createInstance(Components.interfaces.nsILocalFile);
    exe.initWithPath(pol.value);
    var run =
window.Components.classes['@mozilla.org/process/util;1'].createInstance(Components.interfaces.nsIProcess);
    run.init(exe);
    var parameters = [""];

    run.run(false, parameters,parameters.length);

  }
  else
  {
    var oShell = new ActiveXObject("WScript.Shell");
    var prog = pol.value;
    oShell.Run('"' + prog + '"',1);
  }
}
```

```
</script>  
<a href="/ http://www.gmmobile.pl">Strona główna</a>  
</body>  
</html>
```

<http://gaidaw.pl/ajax/jquery-tutorial/p1.html>

<http://jquery.com>

<https://mansfeld.pl/programowanie/jquery-co-to-jest-czy-warto-uzywac/>

## Biblioteka jQuery

### Dołączanie biblioteki.

Biblioteka jQuery jest dołączana do stron WWW jako zewnętrzny plik JavaScript, a zatem nie wymaga instalacji żadnego oprogramowania. Po dołączeniu do dokumentu HTML pliku jquery-1.2.3.js uzyskujemy dostęp do obiektów i metod biblioteki jQuery.

### Znaczenie metoda ready() klasy jQuery.

Zasadniczą rolę odgrywa metoda ready() klasy jQuery. Jest ona wywoływana po kompletnym załadowaniu strony, a zatem w jej treści możemy operować pełnym modelem DOM dokumentu. Wewnątrz funkcji ready() definiujemy anonimową funkcję, która zostanie wywołana po zakończeniu ładowania dokumentu. W treści anonimowej funkcji możemy umieścić dowolny kod, np. wywołanie okna informacyjnego alert():

```
<html>
<head>
  <title>Przykład 1-1</title>
  <script type="text/javascript" src="jquery-1.2.3.js"></script>
  <script type="text/javascript">

    $(document).ready(function(){

      alert('Witaj');

    });

  </script>
</head>
<body>

<p>Zespół Szkół Energetycznych</p>

</body>
</html>
```

### **Obsługa zdarzenia onclick elementu p na dwa sposoby:**

Separacja zachowania (tj. JavaScript) od struktury (tj. HTML) polega na tym, że wewnątrz elementu body nie występują żadne zdarzenia JavaScript.

#### Tradycyjne podejście:



Zdarzenie onclick akapitu p, które w tradycyjnym dokumencie HTML/JavaScript opisalibyśmy jako:

```
<p onclick="alert('Witaj!')">Lorem</p>
```

#### Podejście z użyciem jQuery

przy użyciu jQuery definiujemy stosując selektor 'p' oraz metodę click(), w której umieszczamy wywołanie funkcji alert():

```
<html>
<head>
  <title>Przykład 1-2</title>
  <script type="text/javascript" src="jquery-1.2.3.js"></script>
  <script type="text/javascript">

    $(document).ready(function(){

      $('p').click(function(){
        alert('Witaj!');
      });

    });

  </script>
</head>
<body>

<p> Zespół Szkół Energetycznych </p>

</body>
</html>
```

#### **Składnia jQuery**

\$ w kodzie powyższych przykładów, jest aliasem do funkcji o nazwie jQuery(). W odróżnieniu od Pascal-a czy C++, w języku JavaScript znak \$ jest dozwolony w identyfikatorach funkcji.

#### **Zadanie 35**

Powyższy przykład możemy zapisać jako:

```
<html>
<head>
  <title>Przykład 1-3</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
<script type="text/javascript">
  jQuery(document).ready(function(){
    jQuery('p').click(function(){
      alert('Witaj!');
```

```
});  
});  
</script>  
</head>  
<body>  
<p> Zespół Szkół Energetycznych </p>  
</body>  
</html>
```

Analizując składnię powyższego przykładu, powiemy, że w skrypcie występuje funkcja `jQuery()` wywołana z parametrem `document`:

```
jQuery(document)
```

Funkcja ta zwraca obiekt, na rzecz którego wywołujemy metodę `ready`:

```
jQuery(document).ready(  
    ...  
);
```

Parametrem metody `ready()` jest funkcja anonimowa, tj. taka, która nie otrzymała nazwy. Jej definicja występuje w miejscu wywołania:

```
function(){  
  
}
```

W treści funkcji może występować dowolny kod. W powyższym przykładzie było to wywołanie funkcji `jQuery()` z parametrem `'p'`. Zwróconemu obiektowi wywołaliśmy metodę `click()`, której parametrem jest funkcja anonimowa wywołująca okno dialogowe `alert()`:

```
jQuery('p').click(function(){  
    alert('Witaj');  
});
```

### Zadanie 36

Wykonaj program w jQuery który, po kliknięciu myszką na napis „Podaj pierwszą liczbę” uruchomi okienko **prompt** → nastąpi wczytanie liczby do zmiennej `a_trzy_litery_nazwiska_ucznia` oraz po najechaniu myszką na napis „Podaj drugą liczbę” uruchomi okienko **prompt** → nastąpi wczytanie liczby do zmiennej `b_trzy_litery_nazwiska_ucznia`. Komputer obliczy resztę z dzielenia liczby `a` i `b`. Użyj znacznika `<p>` oraz `<span>`.  
I wypisze wynik:

Np.

a=7

b=4

$7 \% 4 = 3$

<p>Podaj pierwszą liczbę (wczytywanie po kliknięciu myszką)</p> <p>Podaj drugą liczbę (wczytywanie po najechaniu myszką)</p> <p><input type="button" value="Oblicz resztę z dzielenia"/></p>	<div>JavaScript <span>×</span></div> <p>Podaj pierwszą liczbę</p> <input type="text" value="3"/> <p><input type="button" value="OK"/> <input type="button" value="Anuluj"/></p>
<p>Podaj pierwszą liczbę (wczytywanie po kliknięciu myszką)</p> <p>Podaj drugą liczbę (wczytywanie po najechaniu myszką)</p> <p><input type="button" value="Oblicz resztę z dzielenia"/></p>	<div>JavaScript <span>×</span></div> <p>Podaj drugą liczbę</p> <input type="text" value="6"/> <p><input type="button" value="OK"/> <input type="button" value="Anuluj"/></p>
<p>Podaj pierwszą liczbę (wczytywanie po kliknięciu myszką)</p> <p>Podaj drugą liczbę (wczytywanie po najechaniu myszką)</p> <p><input type="button" value="Oblicz resztę z dzielenia"/></p> <p><math>3 \% 6 = 3</math></p>	

## Selektory CSS

### Element

Do wyboru wszystkich elementów zadanego typu służy selektor będący nazwą elementu. Akapity wybierzemy selektorem `p`, sekcje `div` — selektorem `div`, tabele — selektorem `table`.

Skrypt:

```
$(document).ready(function(){
  $('p').click(function(){
    alert('kliknięto p');
  });
});
```

-----  
<p> Zespół Szkół Energetycznych </p>

przypisze obsługę zdarzenia onclick wszystkim akapitom **<p>** występującym w dokumencie.

Uwaga:

W powyższym przykładzie fragment znajdujący się przed linią jest kodem JavaScript, zaś dolny fragment — kodem HTML. Wszystkie kolejne przykłady zostały przedstawione w podobnej, skróconej formie.

Uwaga:

Technologia	oddzielenie	Od
CSS	Kodu HTML	prezentacji-wyglądu
jQuery	Kodu HTML	działania-akcji

### Praktyczne oddzielenie kodu HTML od **prezentacji-wyglądu** dla CSS

```
p {
  color: blue;
}
```

to wszystkie akapity staną się niebieskie.

### Praktyczne oddzielenie kodu HTML od **działania-akcji** dla CSS

Podobnie, selektor **p** w jQuery:

```
$('p').click(function(){
  alert('kliknięto p');
});
```

spowoduje, że wszystkie elementy `p` otrzymają obsługę zdarzenia onclick.

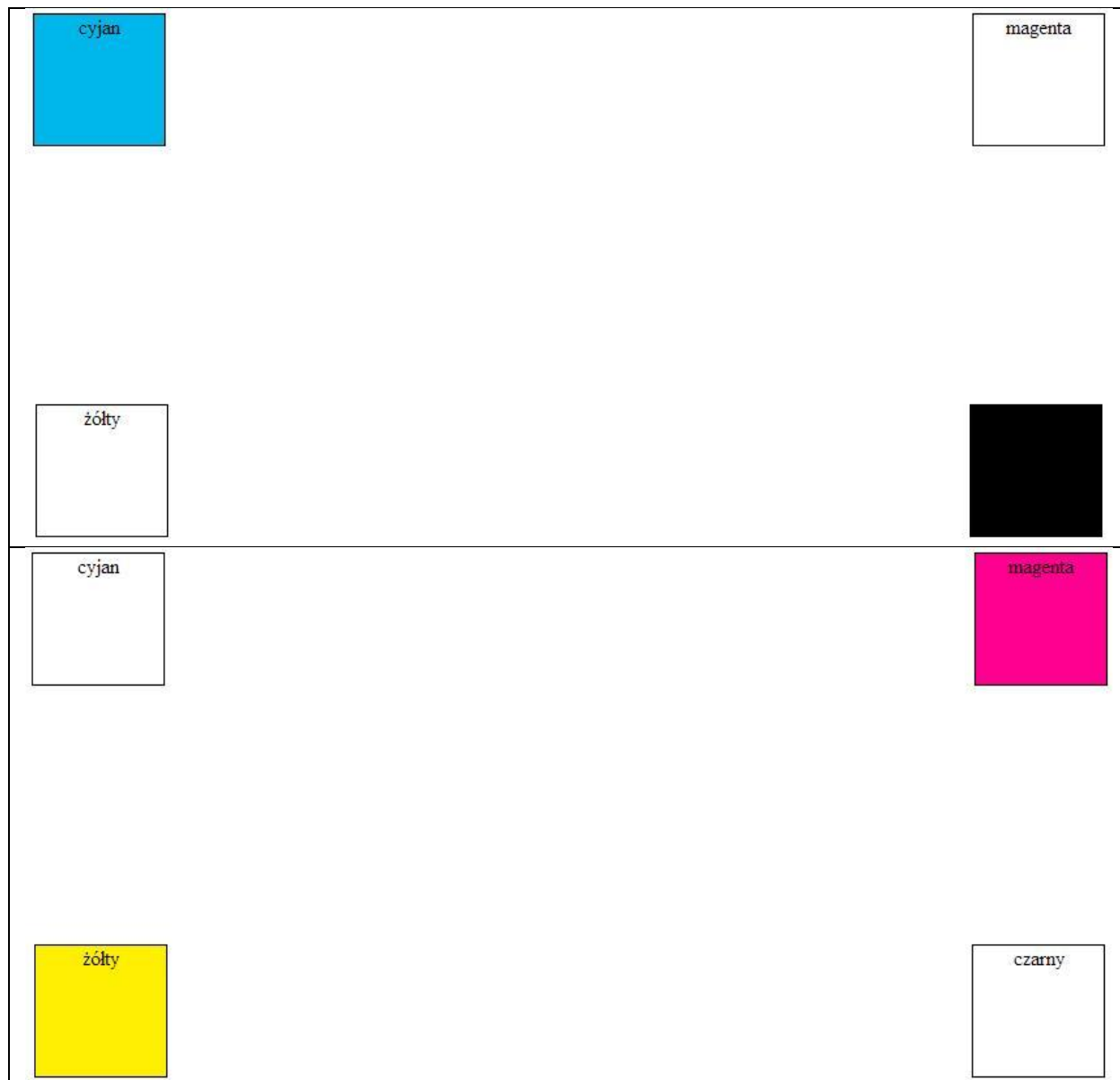
### Identyfikator

Jeśli element HTML posiada identyfikator id, wówczas stosujemy selektor składający się ze znaku # oraz identyfikatora:

```
$(document).ready(function(){  
  $('#tresc').click(function(){  
    alert('kliknięto #tresc');  
  });  
});  
-----  
<p id="tresc"> Zespół Szkół Energetycznych </p>
```

### Zadanie 37

Wykonaj program w jQuery który, po kliknięciu myszką na kwadraty będzie gasił i zapalał w schemacie kolorów CMYK. Użyj Div i id.



## **Klasa**

Selektor elementów wzbogaconych o klasę składa się z kropki i nazwy klasy:

```
$(document).ready(function(){
  $('.inny').click(function(){
    alert('kliknięto .inny');
  });
});
-----
<p class="inny"> Zespół Szkół Energetycznych </p>
```

## **Wartość atrybutu**

Selektor:

`a[href="ipsum.html"]`

odnosi się do elementu `a`, którego atrybut `href` przyjmuje wartość `ipsum.html`:

```
<a href="ipsum.html">ipsum</a>
Zatem kod:
$(document).ready(function(){
  $('a[href="ipsum.html"]').click(function(){
    alert('kliknięto ipsum.html');
  });
});
-----
<ol>
<li><a href="lorem.html">lorem</a></li>
<li><a href="ipsum.html">ipsum</a></li>
<li><a href="dolor.html">dolor</a></li>
</ol>
```

będzie ustalał obsługę zdarzenia `onclick` wyłącznie środkowego hiperłącza.

## **Zmiana treści i wyglądu**

### **Dostęp do treści elementu**

Dostęp do treści elementu zapewnia metoda `html()`. Wywołana bez parametru zwraca bieżącą treść elementu. Wywołana z parametrem — ustala nową treść.

Jeśli metodę `html()` wywołamy na rzecz klikniętego akapitu, wówczas kliknięcie akapitu będzie powodowało zmianę treści elementu:

```
$(document).ready(function(){
  $('p').click(function(){
    $(this).html('one, two, ...');
  });
});
```

-----  
<p> Zespół Szkół Energetycznych </p>

Dostęp do klikniętego akapitu wewnątrz metody click umożliwia this — referencja do obiektu, który wygenerował zdarzenie.

### Zmiana stylu

Styl elementu zmienimy metodą `css()`, która pobiera dwa parametry: nazwę właściwości oraz wartość:

```
$(document).ready(function(){
  $('p').click(function(){
    $(this).css('color', 'yellow');
  });
});
```

-----  
<p> Zespół Szkół Energetycznych </p>

Wadą powyższego rozwiązania jest połączenie dwóch języków: CSS oraz JavaScript. Wygodniej będzie do zmiany formatu wykorzystać klasy CSS oraz metody `addClass()` i `removeClass()`.

### Dodanie i usunięcie klasy

Każdy element wybrany selektorem możemy dynamicznie wzbogacić o klasę. Umożliwia to metoda **`addClass()`**. Poniższy przykład jest rozbity na trzy języki: CSS, JavaScript oraz HTML:

```
.inny {
  color: yellow;
}
```

```
$(document).ready(function(){
  $('p').click(function(){
    $(this).addClass('inny');
  });
});
```

-----  
<p> Zespół Szkół Energetycznych </p>

W celu usunięcia klasy wywołujemy metodę **`removeClass()`**:

```
.inny {
```

```
color: yellow;
}
$(document).ready(function(){
  $('p.inny').click(function(){
    $(this).removeClass('inny');
  });
});
-----
<p> Zespół Szkół Energetycznych </p>
```

---

## Zdarzenia

Biblioteka jQuery zapewnia dostęp do pełnego zestawu zdarzeń HTML. Na przykład obsługę zdarzeń onmouseover oraz onmouseout zdefiniujemy metodami mouseover() oraz mouseout():

```
.inny {
  color: yellow;
}

$(document).ready(function(){
  $('p').mouseover(function(){
    $(this).addClass('inny');
  });
  $('p').mouseout(function(){
    $(this).removeClass('inny');
  });
});
-----
<p> Zespół Szkół Energetycznych </p>
```

zaś zdarzenie ondblclick metodą dblclick():

```
$(document).ready(function(){
  $('p').dblclick(function(){
    $(this).css('text-decoration', 'underline');
  });
});
-----
<p> Zespół Szkół Energetycznych </p>
```

Duża część funkcji odpowiedzialnych za zdarzenia może być wywoływana na dwa sposoby.

Pierwszym sposobem jest wywołanie z parametrem, którym jest funkcja:



```
$('#p').click(function(){  
    ...  
});
```

Takie wywołanie definiuje funkcję obsługi zdarzenia.

Drugi sposób wywołania to wywołanie bezparametrowe:

```
$('#p').click();
```

Powoduje ono uruchomienie zdarzenia (ang. trigger).

Zestawienie części zdarzeń HTML i odpowiadających im metod klasy jQuery.

Zdarzenie HTML	Metoda jQuery
onchange	change()
onclick	click()
ondblclick	dblclick()
onkeydown	keydown()
onkeypress	keypress()
onkeyup	keyup()
onload	load()
onmousedown	mousedown()
onmousemove	mousemove()
onmouseout	mouseout()
onmouseover	mouseover()
onmouseup	mouseup()
onsubmit	submit()

## Zagadnienia dalsze

### Automatyczna iteracja

Jak już zostało powiedziane, selektor jQuery obejmuje swoim działaniem wszystkie wybrane elementy. Jeśli użyjemy selektora li:

```
$(document).ready(function(){  
  $('li').css('color', 'blue');  
});
```

```
-----  
<ul>  
  <li>one</li>  
  <li>two</li>  
  <li>three</li>  
  <li>four</li>  
</ul>
```

to przetwarzaniu będą podlegały wszystkie elementy listy. Każdy z nich stanie się niebieski. Nie wpisujemy ręcznie żadnej iteracji (np. pętli for). Jest ona wykonywana podobnie jak w CSS automatycznie dla wszystkich elementów pasujących do selektora.

### Kaskada

Większość metod jQuery zwraca w wyniku obiekt jQuery. Pozwala to na tworzenie kaskadowych wywołań metod:

```
$(document).ready(function(){  
  $('li').css('color', 'yellow').append(' - read more').css('background', 'black');  
});
```

```
-----  
<ul>  
  <li>one</li>  
  <li>two</li>  
  <li>three</li>  
  <li>four</li>  
</ul>
```

Powyższy kod jQuery możemy równoważnie zapisać jako trzy osobne wywołania:

```
$('li').css('color', 'yellow');  
$('li').append(' - read more');  
$('li').css('background', 'black');
```

## Odczyt atrybutów

Dostęp do atrybutów zapewnia metoda `attr()`, której parametrem jest nazwa atrybutu HTML:

```
$(document).ready(function(){
  $('input').focus();
  $('input').keyup(function(){
    $('div').html($('input').attr('value'));
  });
});

-----

<form action="#" method="post">
  <input type="text" name="imie" value="" />
</form>
<div></div>
```

Wywołanie w jQuery:

```
$('input').attr('value')
```

powoduje odczytanie atrybutu `value` elementu `input`:

```
<input name="" value="" />
```

### wytłumaczenie

Pierwsza instrukcja:

```
$('input').focus();
```

uruchamia zdarzenie `onfocus` dla elementu `input`. Dzięki niej po bezpośrednio po otwarciu dokumentu w przeglądarce kursor będzie znajdował się w polu edycyjnym formularza.

## Ajax, czyli asynchroniczne ładowanie pliku

Do załadowania zewnętrznego pliku służy metoda `load()`:

```
$(document).ready(function(){
  $('#content').load('lorem.html');
});

-----

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

Działa ona w oparciu o obiekt `XMLHttpRequest`. Dokument, którego adres URL jest podany jako parametr jest pobierany asynchronicznie w tle (Ajax).

---

## Menu

### Ładowanie treści po kliknięciu

Wykonanie menu, które nie będzie przeładowywało całej strony rozpoczynamy od załadowania zewnętrznego dokumentu (tj. pliku fragment-abba.html) do elementu o identyfikatorze #content po wystąpieniu zdarzenia onclick. Zadanie to zrealizuje następujący kod jQuery:

```
$('#a1').click(function(){  
    $('#content').load('fragment-abba.html');  
});
```

```
<ol>  
    <li id="a1">ABBA</li>  
</ol>  
<div id="content"></div>
```

Najpierw wybieramy element o identyfikatorze #a1, któremu definiujemy zdarzenie onclick:

```
$('#a1').click(function(){  
    ...  
});
```

Wewnątrz obsługi zdarzenia występuje ładowanie zewnętrznego dokumentu fragment-abba.html do elementu o identyfikatorze #content:

```
$('#content').load('fragment-abba.html');
```

### 7.2 Kilka opcji menu

Jeśli menu zawiera kilka opcji:

```
<ol id="menu">  
    <li id="a1">ABBA</li>  
    <li id="a2">AC DC</li>  
    ...  
</ol>  
<div id="content"></div>
```

wówczas definiujemy zdarzenie onclick każdej opcji z osobna:

```
$(document).ready(function(){  
  
    $('#a1').click(function(){  
        $('#content').load('fragment-abba.html');  
    });
```

```

$('#a2').click(function(){
    $('#content').load('fragment-ac_dc.html');
});

...

});

```

### 7.3 Menu ol/li/a

Opcje menu należy przekształcić w hiperłącza a:

```

<ol>
  <li><a href="fragment-abba.html">ABBA</a></li>
  <li><a href="fragment-ac_dc.html">AC DC</a></li>
  ...
</ol>
<div id="content"></div>

```

W tym przypadku stosujemy selektor wybierający element o zadanej wartości atrybutu. W ten sposób kod jQuery przyjmie postać:

```

$(document).ready(function(){

    $('a[href="fragment-abba.html"]').click(function(){
        $('#content').load('fragment-abba.html');
        return false;
    });

    $('a[href="fragment-ac_dc.html"]').click(function(){
        $('#content').load('fragment-ac_dc.html');
        return false;
    });

    ...

});

```

Instrukcje return zapewniają, że kliknięcie hiperłącza nie będzie powodowało przeładowania strony.

### 7.4 Ajaksowe menu działające po wyłączeniu JavaScript

Witryna, z menu, które nie przeładowuje strony, wymaga przygotowania każdej podstrony w dwóch wersjach: pełnej i okrojonej. Strona w wersji okrojonej zawiera przedrostek fragment-. Jeśli na przykład witryna zawiera dwie opcje menu, które mają ładować pliki abba.html oraz ac\_dc.html, to należy przygotować cztery pliki:

abba.html  
fragment-abba.html

ac\_dc.html  
fragment-ac\_dc.html

W menu witryny, w atrybutach href, stosujemy nazwy pełnych stron (tj. bez przedrostka fragment-): Dzięki temu witryna będzie działała po wyłączeniu JavaScript:

```
<ol>
  <li><a href="abba.html">ABBA</a></li>
  <li><a href="ac_dc.html">AC DC</a></li>
  ...
</ol>
<div id="content">
```

W powyższym kodzie ujawnia się cały urok rozwiązania stosującego jQuery. Jest to bowiem zwykłe menu, jakie byśmy przygotowali wykonując statyczne pliki HTML.

Modyfikację hiperłączy menu, w takie, które nie przeładowują strony zapewnia kod zawarty w pliku menu.js (plik ten należy oczywiście dołączyć do każdej pełnej podstrony):

```
$(document).ready(function(){

  $('a').click(function(){
    $('#content').load(
      'fragment-' + $(this).attr('href')
    );
    return false;
  });

});
```

Kolejno:

wybieramy wszystkie hiperłącza \$('a'),  
ustalamy obsługę zdarzenia onclick,  
w obsłudze zdarzenia onclick do elementu o identyfikatorze #content ładujemy zewnętrzny plik,  
adres url pobieranego pliku zewnętrznego powstaje z adresu odczytanego z atrybutu href klikniętego hiperłącza this przez dodanie prefiksu fragment-,  
kliknięcie hiperłącza nie może powodować przeładowania strony (return false).

---

## 8. Wyszukiwarka

Ajaksowa wyszukiwarka, której użycie nie przeładowuje strony wykorzystuje formularz oraz element #wyniki:

```
<form action="index.php" method="get">
  <input id="sz" name="szukaj" />
  <input type="submit" value="szukaj" />
</form>
<div id="wyniki"></div>
```

Działanie wyszukiwarki definiuje następujący kod JavaScript:

```
$(document).ready(function(){

  $('#wyniki').hide();

  $('form').submit(function(){
    $('#wyniki').load('server.php?co=' + $('#sz').attr('value'));
    $('#wyniki').show();
    return false;
  });

});
```

Kolejno:

- zaraz po załadowaniu dokumentu ukrywamy element #wyniki (metoda hide()),
- ustalamy obsługę zdarzenia onsubmit elementu form,
- obsługa zdarzenia polega na załadowaniu zewnętrznego dokumentu do elementu #wyniki (metoda load()) i pokazaniu elementu #wyniki (metoda show()),
- adres url pobieranego elementu powstaje przez dodanie na końcu adresu sever.php?co= tekstu wprowadzonego w formularzu do elementu o identyfikatorze #sz.
- instrukcja return false zakazuje przeładowania strony.

Wykorzystując odwołania kaskadowe, obsługę zdarzenia onsubmit można zakodować następująco:

```
$('#wyniki').load('server.php?co=' + $('#sz').attr('value')).show();
```

Podana wyszukiwarka, podobnie jak opisane wcześniej menu, działa także po wyłączeniu obsługi JavaScript.

## Co to jest JavaScript?

Ten wszechobecny dziś język skryptowy został stworzony – początkowo jako LiveScript – na potrzeby firmy Netscape i przeglądarki Navigator. Zamierzeniem jego projektantów było rozszerzenie możliwości języka HTML w taki sposób, aby strony stały się bardziej atrakcyjne pod względem sposobu prezentacji danych oraz możliwości interakcji z użytkownikiem. W 1995 roku, na mocy porozumienia z firmą Sun Microsystems, w zamian za włączenie do przeglądarki Netscape obsługi technologii Java, język otrzymał oficjalną nazwę JavaScript. W grudniu tego samego roku został zaimplementowany w najnowszej wówczas wersji Navigatora i rozpoczął triumfalny pochód przez Sieć.

JavaScript jest przede wszystkim **językiem skryptowym – to znaczy interpretowanym**. Nie musi zostać skompilowany do kodu maszynowego, aby można było zobaczyć efekty jego działania. Wystarczy nam do tego przeglądarka internetowa, która ten język obsługuje – czyli w zasadzie każda z liczących się na rynku aplikacji. Ze względów bezpieczeństwa JavaScript ma znacznie ograniczone uprawnienia dostępu do zasobów komputera, przy użyciu którego przeglądana jest dana strona, a wszelkie odwołania do funkcji i obiektów wykonywane są w trakcie wykonywania programu.

## JavaScript a Java

Zbieżność nazw obu języków może być dla początkującego użytkownika myląca. Przyjęcie takiej, a nie innej nazwy dla produktu Netscape'a najprawdopodobniej podyktowane było względami marketingowymi, ale zasadność łączenia obu pakietów do dziś jest przedmiotem kontrowersji. Istnieją tutaj dwie szkoły: jedni uważają, że JavaScript sięga swoimi korzeniami środowiska Java, wskazując na przykład na podobieństwo składni, inni zaś dowodzą, że poza nazwą oba języki niewiele mają ze sobą wspólnego. Jakiegokolwiek miałyby być rozstrzygnięcie tego sporu, debiutujący programiści powinni pamiętać o tym, że mają do czynienia z zupełnie odrębnymi mechanizmami.

## Zalety JavaScriptu

Największą zaletą JavaScriptu jest niewątpliwie jego prostota. Język ten uważa się więc za relatywnie łatwy do opanowania i rzeczywiście – nic nie stoi na przeszkodzie, aby zacząć używać go już po kilku minutach od rozpoczęcia nauki. Ponieważ jest to obecnie najbardziej rozpowszechniony język skryptowy, w Internecie można znaleźć miliony praktycznych przykładów – witryn wykorzystujących różnorodne funkcje JavaScriptu.

O olbrzymiej popularności języka zadecydowały z pewnością jego uniwersalność oraz fakt, iż pozwala on na odciążenie serwerów i ograniczenie ilości zbędnych danych przesyłanych pomiędzy nimi a przeglądarką klienta. Najprostszym przykładem tych zalet mogą być choćby sklepy internetowe, których nieodzownym elementem są formularze wypełniane przez użytkownika. Dzięki JavaScriptowi można sprawdzić poprawność wprowadzonych danych na komputerze klienta, unikając – przy ewentualnym błędzie – urzeczywistnienia scenariusza, w którym dane są wysyłane, a serwer zużywa moc obliczeniową na ich sprawdzenie, wygenerowanie strony z opisem błędu i odesłaniem jej z powrotem.

JavaScript jest nieustannie rozwijany. Podstawowe funkcje z biegiem czasu uzupełniane są o nowe narzędzia, jak chociażby obiekt Image i tablica document.images, które pojawiły się w JavaScriptcie 1.1 i wzbogaciły język o opcje manipulacji plikami graficznymi. Wraz z wersją 1.2, koncepcją dynamicznego HTML-a i zastosowaniem warstw możliwości projektowania interaktywnych stron zwiększyły się wręcz zdumiewająco. Obecnie przy użyciu odpowiednich bibliotek – jak na przykład script.aculo.us czy mootools – tworzyć można aplikacje sieciowe o dużym stopniu zaawansowania i atrakcyjnym interfejsie.



## Umieszczanie skryptu na stronie

Skrypt JavaScript można umieścić w czterech różnych miejscach:

W treści strony

JavaScript

```
<body>
  <script type="text/javascript">
    <!--
      alert("Hello World!");
    //-->
  </script>
</body>
```

```
<body>
```

```
  <script type="text/javascript">
    <!--
      alert("Hello World!");
    //-->
  </script>
</body>
```

W nagłówku strony wewnątrz znaczników <head>

JavaScript

```
<head>
  <script type="text/javascript">
    <!--
      alert("Hello World!");
    //-->
  </script>
</head>
```

```
<head>
```

```
  <script type="text/javascript">
    <!--
      alert("Hello World!");
    //-->
  </script>
</head>
```

Skrypty w nagłówku HTML nie wpływają od razu na dokument HTML, lecz mogą się do nich odwoływać inne skrypty. Nagłówek często wykorzystywany jest do umieszczania funkcji – grup instrukcji JavaScriptu, które mogą być użyte jako jedna całość.

W znaczniku HTML

JavaScript

```
<body onLoad="window.alert('Okno komunikatu')">
```

1

```
<body onLoad="window.alert('Okno komunikatu')">
```

Nosi to nazwę funkcji obsługi zdarzenia (ang. event handler) i pozwala na pracę skryptu z elementami HTML. Używając JavaScriptu w funkcjach obsługi zdarzeń, nie musimy używać znacznika <script>.

W osobnym pliku

JavaScript

<head>

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

</head>

<head>

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

</head>

JavaScript pozwala korzystać z plików o rozszerzeniu .js zawierających skrypty; można je dołączać do dokumentu HTML, wskazując nazwę pliku w atrybucie src.

Kod źródłowy zagnieżdżony w HTML-u

Kod JavaScript musi być zawarty pomiędzy znacznikami HTML <script> i </script>:

Przykład

Temat: Zagnieżdżanie skryptu

<script>

    kod naszego skryptu

</script>

Znaczniki takie można umieszczać w dowolnym miejscu dokumentu, ale dobrą praktyką jest osadzanie kodu na początku strony, w sekcji HEAD, która przez przeglądarkę jest wczytywana jako pierwsza. Pozwala to na uniknięcie sytuacji, kiedy użytkownik aktywuje element strony powiązany z kodem, który jeszcze nie został załadowany. Oczywiście możemy w obrębie strony używać znaczników <script> wielokrotnie – na przykład w nagłówku i sekcji BODY.

Znacznik <script> ma **atrybut type** – nadając mu odpowiednią właściwość, definiujemy język, w którym napisany będzie nasz kod. W wypadku JavaScriptu wartością atrybutu będzie oczywiście: „text/javascript”.

Przykład

Temat: Kod HTML strony używającej JavaScript.

<html>

    <head>

        <script type="text/javascript">

            kod skryptu

        </script>

        <script type="text/javascript">

            przykładowy kod

        </script>

    </head>

<body>

    <script type="text/javascript">

        przykładowy kod

    </script>

</body>

</html>

Przykład

Temat: Kod HTML strony używającej JavaScript.

## Kod źródłowy zamieszczony w oddzielnym pliku

Bardzo dobrą i ułatwiającą pracę programisty praktyką jest wielokrotne wykorzystywanie napisanego wcześniej kodu. Aby uniknąć każdorazowego przeszukiwania dokumentów, otwierania, kopiowania i wklejania, kod źródłowy skryptu możemy umieścić w osobnym pliku. Będzie to plik tekstowy o rozszerzeniu .JS, zawierający kod pisany już bezpośrednio, bez użycia znaczników <script>. O tym, że kod źródłowy jest w pliku zewnętrznym, informujemy przeglądarkę, wykorzystując atrybut src:

Składnia:

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

Przykład

Temat: Zewnętrzny pliku zawierający kod JavaScript. Wyświetlanie datę ostatniej modyfikacji pliku html.

**Plik \*.html**

```
<HTML>
  <HEAD>
    <TITLE>PIERWSZY SKRYPT</TITLE>
  </HEAD>
<BODY>
  <SCRIPT type="text/javascript" src="przyklad0_zewnetrzny.js"></SCRIPT>
</BODY>
</HTML>
```

**Plik \*.js** o nazwie przyklad0\_zewnetrzny.js

```
document.write(document.lastModified);
```

Dzięki takiej metodzie postępowania możemy wkrótce stworzyć własną kolekcję najczęściej używanych skryptów, co zaowocuje oszczędnością czasu potrzebnego na tworzenie nowych rozwiązań od podstaw.

## Jak zadbać o przeglądarki nieobsługujące JavaScriptu?

Mimo że w zasadzie wszystkie używane dziś przeglądarki bez problemu interpretują kod JavaScript, nie zaszkodzi, jeżeli zadbamy o użytkowników, którzy na przykład wyłączyli obsługę języka w ustawieniach przeglądarki. Zaoszczędzimy im trudnych do przewidzenia zachowań aplikacji lub komunikatów o błędach, umieszczając kod w HTML-owych znacznikach komentarza. Znacznik zamykający komentarz HTML powinniśmy przy tym wzbogacić o sekwencję komentarza JavaScript – `//`. Unikniemy w ten sposób konfliktów wynikających z faktu, że symbole tego znacznika są w obrębie składni JavaScriptu nieprawidłowe.

Przykład

Temat: Gdy przeglądarka nie obsługuje JavaScript. Kod JavaScript umieszczony pomiędzy znacznikami komentarza

```
<script type="text/javascript">
    <!--
        kod skryptu
    //-->
</script>
```

Dobłą praktyką jest także poinformowanie użytkowników, że strona zawiera skrypty, które nie zostały wykonane przez ich przeglądarkę. W tym celu stosuje się **znaczniki <noscript>**. Pomiędzy nimi powinniśmy umieścić element (np. odpowiedni komunikat), który zostanie wyświetlony zamiast interpretacji kodu. Uwzględniając wszystkie powyższe wskazówki, szablon naszej strony HTML będzie wyglądał następująco:

Przykład

Temat: Kod strony zawierającej JavaScript i użycie znacznika <noscript>

```
<html>
    <head>
        <script type="text/javascript">
            <!--
                kod skryptu
            //-->
        </script>
    </head>
    <body>
        <noscript>
            Twoja przeglądarka nie obsługuje JavaScriptu lub wyłączyłeś jego obsługę.
        </noscript>
        kod HTML strony
    </body>
</html>
```

## Komentarze do kodu

Do wyboru mamy dwa typy komentarzy:

- **Komentarz liniowy** zaczyna się od dwóch ukośników, a kończy przy przejściu do następnej linii. Oznacza to, że przeglądarka zignoruje wszystko za znacznikiem // aż do końca linii, w której znacznik ten występuje.  
np. // to jest komentarz liniowy
- **Komentarz blokowy rozpoczyna** się z kolei od sekwencji: /\*, a kończy sekwencją: \*/. Może on więc ciągnąć się przez wiele linii – pamiętajmy jednak, że niemożliwe jest jego zagnieżdżanie, czyli stosowanie jednego komentarza zawartego w innym.  
np. /\* komentarz blokowy \*/

Dzięki komentarzom możemy poinformować użytkownika, że jego przeglądarka nie obsługuje skryptów, nie używając przy tym znacznika <noscript>. Wymieniony w listingu nr 5 przykład zmienia więc swoją postać na:

Przykład

Temat: Kod strony zawierającej w JavaScript z komentarzami.

```

<html>
  <head>
    <script type="text/javascript">
      // Twoja przeglądarka nie obsługuje JavaScriptu.
      /* Aby zobaczyć stronę w pełnej funkcjonalności, zainstaluj inną przeglądarkę
      Internet Explorer, Netscape Navigator, Mozilla, Opera... */

      <!--
        kod skryptu
      // -->
    </script>
  </head>
  <body>
    kod HTML strony
  </body>
</html>

```

## Wyświetlanie informacji na stronie i działania na zmiennych

### Kilka podstawowych zasad programowania

- Po pierwsze, interpreter kodu rozróżnia tekst **pisany wielkimi i małymi literami**. Musimy być więc niezwykle ostrożni, aby nie popełnić błędu – na przykład wywołując wcześniej zdefiniowaną funkcję „program()” za pomocą instrukcji „Program()”. W takim wypadku przeglądarka zwróci błąd, a początkujący programista może mieć duże problemy ze zidentyfikowaniem jego przyczyny.
- Czynnością należącą już raczej do katalogu dobrych praktyk jest ponadto umieszczanie na **końcu każdej instrukcji średnika** – głównie w sytuacji, gdy w jednej linii tekstu chcemy zawrzeć kilka poleceń. Ma to często miejsce na przykład podczas konstruowania pętli. Przejrzystości kodu służy także umiejętne stosowanie spacji. Odstępy nie są „widoczne” dla interpretera, ale zdecydowanie ułatwiają naszą pracę ze skryptem lub też zapoznanie się innych użytkowników z jego budową.

### Instrukcja document.write

**Document** to obiekt JavaScript, który reprezentuje akurat wyświetlaną stronę, **write** to natomiast jego metoda, czyli funkcja wykonująca określone działania na obiekcie – w tym wypadku wpisująca tekst. Nasz tekst umieszczamy w nawiasach jako argumenty wywołania metody. Postępujemy więc zgodnie z ogólną regułą składni, która wygląda następująco:  
 obiekt.metoda(argumenty metody);

Przykład

Temat: Zastosowanie metody write

```
document.write("Witaj, świecie!");
```

Przykład

Temat: Metoda write – użycie znaczników HTML

```
document.write("<h1>Strona tytułowa</h1>");
document.write("<b>Spis treści</b>");
```

Jako składników tekstu możemy także używać znaczników HTML, dbając oczywiście o to, żeby w odpowiednich miejscach otwierać je i zamykać:

### Uwaga:

#### Metoda write – użycie cudzysłowów w łańcuchu znaków.

W tym miejscu powinniśmy zwrócić naszą uwagę na pewien istotny fakt: otóż łańcuch znaków przekazywany metodzie write ograniczony jest z obu stron podwójnym cudzysłowem. Co zrobić zatem w wypadku, gdy w tym łańcuchu będziemy chcieli umieścić znaczniki HTML, których atrybuty również używają cudzysłowu? Jeżeli użyjemy symbolu `"`, popełnimy błąd, gdyż interpreter JavaScriptu przyjmie, że w tym miejscu zakończyliśmy wprowadzanie łańcucha znakowego do metody write. Aby uniknąć wynikających z tego problemów, musimy stosować zamiennie znaki `"` oraz `'`. Jeżeli będziemy pamiętali o tym, że powinny się one parami otwierać i zamykać, możemy wielokrotnie zagnieżdżać jedno w drugim, stosując je przemiennie: `"1 '2 "3 – 3" 2' 1"`.

Nieco łatwiej przyjdzie nam rozstrzygnąć problem ewentualnego użycia cudzysłowu w samym tekście, który chcemy umieścić na stronie. Zarówno cudzysłów, jak i inny znak specjalny powinniśmy w takim wypadku poprzedzić backslashem – czyli znakiem `\` lub specjalnymi sekwencjami podstawienia HTML.

```
document.write("Witamy na naszej stronie filmowej.");
document.write("Polecamy film \"Władca Pierścieni\"");
document.write("<b><a href='spis.htm'>Spis treści</a></b>");
```

## Zmienne

Typy zmiennych:

- **string** → czyli łańcuch znakowy – dlatego jest on umieszczony w cudzysłowie.
- **integer** → liczbę całkowitą
- **float** → zmiennoprzecinkową

W JavaScriptcie nie musimy deklarować, jakiego typu zmiennej będziemy używali.

Ponadto typ zmiennej elastycznie dopasowuje się do naszych potrzeb, dlatego nic nie stoi na przeszkodzie, aby przypisywać na zmianę typ łańcuchowy i liczb całkowitych.

Jedyne, co musimy zrobić, to poinformować interpreter o korzystaniu ze zmiennej.

```
var nazwa zmiennej = wartość zmiennej;
```

Przykład

Temat: Zastosowanie instrukcji var

```
var imie="Janek"                // zmienna typu string
var wiek=20                     // zmienna typu integer
document.write("Nasz gość ma na imie "+imie+".");
document.write(imie+" ma "+wiek+" lat");
```

Nazwy zmiennych zaczynamy od litery i konstruujemy je w całości z liter, cyfr lub znaku podkreślenia (`_`). Powinniśmy też zadbać, aby nazwa była zrozumiała dla osoby czytającej kod – na przykład wiązała się określonym elementem czy funkcją. Wszystko to oczywiście po to, byśmy w przyszłości nie musieli się zastanawiać, z jakich przyczyn daną zmienną w kodzie umieściliśmy.

## Operatory

Operatory służą dokonywaniu działań na wartościach zmiennych. Już w powyższym przykładzie zastosowania funkcji `var` użyliśmy operatora łączenia łańcuchów tekstu – `+`. Stosując go, możemy wpisywać do ciągu złożone zdania, zmieniające się w zależności od wartości wprowadzanych zmiennych. Należy przy tym pamiętać, że jeśli przy łączeniu różnych typów zmiennych występuje łańcuch znakowy (string) i operator łączenia `+`, pozostałe zmienne są również przekształcane na typ string.

Ważniejszym jeszcze od manipulacji tekstem zastosowaniem operatorów będą oczywiście działania matematyczne. Możemy przy tym wykorzystać szeroką ich gamę – JavaScript oddaje nam do dyspozycji nie tylko operatory arytmetyczne, ale także logiczne oraz operatory przypisania i porównania. Poniższe tabele w sposób wyczerpujący prezentują wszystkie dostępne nam możliwości.

### Operatory arytmetyczne

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

### Operatory przypisania

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

### Operatory porównania

Operator	Opis	Przykład
==	jest równe	<code>2==3    wynik:fałsz</code>
!=	nie jest równe	<code>2!=3    wynik:prawda</code>
>	jest większe	<code>25&gt;3    wynik:prawda</code>
<	jest mniejsze	<code>2&lt;3    wynik:prawda</code>
>=	większe lub równe	<code>25&gt;=3    wynik:prawda</code>
<=	mniejsze lub równe	<code>2&lt;=3    wynik:prawda</code>

### 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



## KONWERSJA TYPÓW ZMIENNYCH

isFinite(zmienna) - zwraca true jeżeli zmienna ma wartość nieskończoność

isNaN(zmienna) - zwraca true jeżeli zmienna nie jest liczbą ("Not a Number")

Number(zmienna) - sprawdza czy zmienna jest typu liczbowego (zwraca NaN jeżeli nie, zwraca wartość jeżeli tak)

parseFloat(zmienna) - konwersja zmiennej do typu float

parseInt(zmienna) - konwersja zmiennej do typu int (liczbowego)

toString(zmienna) - konwersja zmiennej do typu łańcuchowego

```
var txt = "Hello World!";
```

```
document.write("The original string: " + txt);  
document.write("<p>Big: " + txt.big() + "</p>");  
document.write("<p>Small: " + txt.small() + "</p>");  
document.write("<p>Bold: " + txt.bold() + "</p>");  
document.write("<p>Italic: " + txt.italics() + "</p>");  
document.write("<p>Fixed: " + txt.fixed() + "</p>");  
document.write("<p>Strike: " + txt.strike() + "</p>");  
document.write("<p>Fontcolor: " + txt.fontcolor("green") + "</p>");  
document.write("<p>FontSize: " + txt.fontSize(6) + "</p>");  
document.write("<p>Subscript: " + txt.sub() + "</p>");  
document.write("<p>Superscript: " + txt.sup() + "</p>");  
document.write("<p>Link: " + txt.link("http://www.w3schools.com") + "</p>");  
document.write("<p>Blink: " + txt.blink() + " (works only in Opera)</p>");
```

## Funkcje i obiekty

### Co to jest funkcja?

Funkcja jest oddzielnym blokiem kodu, który może być wielokrotnie wykonywany w danym programie przez jego wywoływanie. W większości wypadków do funkcji przekazujemy odpowiednie argumenty, aby otrzymać wartość wyjściową wynikającą z zaprogramowanych w funkcji działań. Dobrze jest tworzyć funkcję tak, by wykonywała ona jedno określone zadanie – większe operacje zaplanowane w ramach algorytmu powinniśmy więc rozdzielać pomiędzy kilka, wywoływanych kolejno, funkcji. Dzięki temu stworzymy swego rodzaju „cegiełki”, z których zbudujemy następnie cały skrypt, a które (gdy zapiszemy je w oddzielnych plikach – o czym była mowa już wcześniej) będziemy także mogli później wykorzystać w zupełnie innych konfiguracjach czy programach.

Funkcję powinniśmy zdefiniować na początku kodu strony – czyli w sekcji HEAD. Wywołać ją możemy w dowolnym miejscu poniżej, gdy tylko zajdzie taka potrzeba. Dzięki takiemu rozplanowaniu struktury kodu będziemy pewni, że funkcja zostanie załadowana, zanim nastąpi jej wywołanie.

### Jak zdefiniować funkcję → składnia?

Po słowie function wpisujemy zatem nazwę funkcji, a w nawiasach argumenty, jakie chcemy przekazać. Kod wykonywany przez funkcję umieścimy pomiędzy klamrą otwierającą i zamykającą – równoznaczną z końcem funkcji.

Przykład

Temat: Przykładowa funkcja bezargumentowa będzie więc wyglądała tak:

#### **definiowanie**

```
function pomoc()
{
    document.write("<a href=\"pomoc.html\"><img src=\"help.gif\" width=\"15\" height=\"10\"
    alt=\"help\" /></a>");
}
```

#### **wywołanie**

```
document.write("Jeżeli nie wiesz, co zrobić dalej, zobacz pomoc: ");
pomoc() // to jest wywołanie funkcji w programie
```

#### **Funkcja zwracająca wartość**

Przykład

Temat: Funkcję dodającą dwie liczby, których wartość będzie przekazywana w argumentach, w trakcie wywołania.

#### definiowanie funkcji

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

#### wywołanie funkcji

Aby wywołać funkcję w dalszej części programu, wystarczy jedynie wcześniej przypisać zmiennym odpowiednie wartości:

```
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));
```

#### **Zasięg zmiennych**

Zasięg zmiennej nazywany jest inaczej, obrazowo, czasem jej życia. Określa on, w jakich miejscach kodu interpreter JavaScript zmienną „widzi” i może z niej korzystać. Pod tym względem zmienne dzielimy więc na **lokalne i globalne**.

Zmienne lokalne to takie, które są deklarowane wewnątrz funkcji i które „giną” wraz z zakończeniem jej działania. Zatem zmiennej „sumaliczba”, utworzonej w przykładzie, nie będziemy mogli użyć nigdzie poza funkcją „suma”.

Inny charakter mają zmienne globalne – deklarowane poza funkcją, są dostępne od momentu ich pierwszego użycia aż do końca kodu HTML.

## **Funkcje predefiniowane JavaScriptu**

Okienka dialogowe

Javascript udostępnia nam trzy typy okien dialogowych:

Z jednej strony okienka takie są łatwe we wdrożeniu i skuteczne (bo gdy się pojawiają, nie pozwalają kliknąć niczego innego na stronie), z drugiej strony bardzo ładnie odciągają uwagę odwiedzającego od zawartości strony. Nie odradzam ich używania. Po prostu czasami trzeba się zastanowić, czy nie ma lepszej metody, by poinformować użytkownika o danym zdarzeniu. Przykładowo zamiast pokazywać alerta z informacją o źle wpisanym mailu, lepiej jest wyświetlić obok takiego pola czerwony znaczek czy coś symbolizującego, że ów pole jest źle wypełnione. Użytkowanie strony będzie wtedy o wiele przyjemniejsze.

Formatowanie tekstu w okienkach dialogowych

W oknach dialogowych można formatować tekst za pomocą znaków:

\n - służący do łamania linii - jak użycie ENTERA w edytorze tekstu.

\t - służący do wstawiania znaku tabulacji.

Jedynym wymogiem jest to, że znaki te muszą znajdować się między podwójnymi cudzysłowami ("...") a nie między apostrofami ('...') - na przykład "To jest łamany\ntekst."

okienko typu alert()

Metoda alert ma postać

```
alert('treść komunikatu');
```

Po wywołaniu metody alert() zostaje wyświetlone okienko z komunikatem i przyciskiem OK. Po kliknięciu przycisku nie zostaje zwrócona żadna wartość, tak więc okno to nadaje się tylko do informowania użytkownika o pewnych zdarzeniach. Zaznaczyć trzeba, że nie ma możliwości zmiany tytułu tego okna (jest zależny od przeglądarki), a tylko treści jakie to okienko wyświetla.

```
<input type="button" id="alert" value="Okienko Alert">
```

```
<script type="text/javascript">
  function oknoAlert() {
    alert('To jest okienko alert');
  }

  document.getElementById('alert').onclick = function() {
    oknoAlert()
  }
</script>
```

Za pomocą metody getElementById pobraliśmy input i przypisaliśmy do niego zdarzenie onclick, które odpala funkcję pokazującą okienko alert

okienko typu confirm()

Metoda confirm ma postać:

```
confirm('treść komunikatu');
```

Można powiedzieć, że okienko confirm służy do tego, aby użytkownik coś potwierdził (stąd nazwa confirm). Do metody confirm() przekazywany jest tylko jeden parametr zawierający treść jaka będzie

wyświetlana w okienku. Okienko to wyświetla dwa guziki: [OK] i [ANULUJ] (CANCEL). W zależności od tego który guzik został kliknięty przez użytkownika, metoda ta zwróci true lub false.

```
<input type="button" id="confirm" value="Okienko Confirm">
```

```
<script type="text/javascript">
    function oknoConfirm() {
        if (confirm('Czy jesteś pewien, że chcesz kontynuować?')) {
            alert('No to kontynuuj...');
        } else {
            alert('Przykro mi, że nie chcesz kontynuować...');
        }
    }

    document.getElementById('confirm').onclick = function() {
        oknoConfirm()
    }
</script>
```

okienko typu prompt()

```
prompt('treść komunikatu', 'domyślna wartość');
```

Do metody prompt przekazujemy dwa parametry - jeden jest treścią, która będzie wyświetlana w okienku, a drugi jest domyślną wartością, która będzie wyświetlana w polu, w które wpisujemy tekst. Okienko to wyświetla dwa guziki: [OK] i [ANULUJ] (CANCEL). Jeżeli użytkownik kliknie [OK], to zostanie zwrócona wartość z pola tekstowego znajdującego się w tym okienku (lub też 'domyślna wartość', jeżeli użytkownik nie zmieniał zawartości tego pola). Jeżeli użytkownik kliknie [ANULUJ] (CANCEL), to zostanie zwrócona wartość null.

```
<input type="button" id="prompt" value="Okienko Prompt" />
```

```
<script type="text/javascript">
    function oknoPrompt() {
        var imie = prompt('Podaj swoje imię:', 'Kartofel');
        if (imie != null) {
            alert('Witaj ' + imie);
        } else {
            alert('Anulowałeś akcję');
        }
    }

    document.getElementById('prompt').onclick = function() {
        oknoPrompt()
    }
</script>
```

Okienko **alert** – wyświetlająca okienko dialogowe z informacją i przyciskiem OK,

Okienko **prompt** – generująca okno, w którym użytkownik może wpisać wartość, zwracaną następnie do programu.

Okienko **confirm** służy do tego, aby użytkownik coś potwierdził (stąd nazwa confirm).

Kod przedstawiony na poniższym przykładzie wyświetli więc komunikat powitalny, a po podaniu przez użytkownika imienia umieści w zawartości strony odpowiednią treść:

```
alert("Witam na mojej stronie");  
document.write("Miło mi Cię gościć "+ prompt("Podaj imię:") +" na mojej stronie");
```



## Instrukcje warunkowe i pętle

### Instrukcja warunkowa if

Instrukcje warunkowe są jednym z podstawowych elementów tworzących skomplikowane systemy interakcji z użytkownikiem na potrzeby współczesnych aplikacji online. Zasadą ich działania jest to, że polecenia w nich zawarte będą wykonywane, jeżeli umieszczony na wstępie warunek okaże się prawdziwy. Składnia jest więc w tym wypadku następująca: if (warunek) {kod wykonywany, gdy warunek zostanie spełniony}.

Przykład

Temat: Wykorzystanie instrukcji warunkowej if

W oparciu o ten schemat możemy na przykład stworzyć funkcję zamykającą dostęp do strony osobom niepełnoletnim:

```
wiek=prompt("W jakim jesteś wieku?");
if (wiek>18) {
document.write("Jesteś pełnoletni, więc możesz wejść dalej.");
document.write("<br><a href=\"adult.html\">Wejście dla dorosłych</a>");
}
```

### Instrukcja warunkowa if ... else

Składnia zyskuje więc postać:

```
if (warunek)
    {kod wykonywany, gdy warunek zostanie spełniony}
    else
    {kod wykonywany, gdy warunek nie zostanie spełniony}.
```

Przykład

Temat: Wykorzystanie instrukcji warunkowej if ... else

Aby sprawdzić działanie tej instrukcji w praktyce, możemy na przykład stworzyć kod, który skontroluje stan naszej pamięci:

```
odpowiedz=prompt("W którym roku narodził się JavaScript?");

if (odpowiedz=="1995")
{
document.write("Brawo! Masz dobrą pamięć!");
}
else
{
alert("Źle!!!");
document.write("Nie zapamiętałeś dokładnie pierwszej części kursu?");
}
```

### Konstrukcja switch

Jeśli chcemy mieć jeszcze więcej możliwości określenia zachowania programu przy zróżnicowanych wartościach wejściowych, użyjemy konstrukcji switch. Pozwala ona wykonać jeden z wielu bloków kodu, w zależności od różnych wartości zmiennej. Jej składnia ma postać:

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

Przykład:

Temat: Wykorzystując obiekt Date zwracający datę, możemy uzależnić wygląd tekstu wyświetlanego na stronie od dnia tygodnia:

```
var dzis=new Date()           // tworzony jest obiekt z datą
dzien=dzis.getDay()           // wiemy, jaki jest dzień, na podstawie daty
switch (dzien);
{
    case 5:
        document.write("Wreszcie piątek!");
        break;
    case 6:
        document.write("Imprezowa sobota");
        break;
    case 0:
        document.write("Śpiąca niedziela");
        break;
    default:
        document.write("Kiedy wreszcie będzie weekend?!");
}
```

### **Operator warunkowy**

Operator ten pozwala wielokrotnie zaoszczędzić czas programisty i kilka linijek skryptu, gdyż doskonale zastępuje prostą konstrukcję if ... else. W wypadku operatora warunkowego w zależności od zrealizowania warunku uzyskamy pojedynczą reakcję.

Jeśli zostanie on spełniony, zmiennej przypisana będzie jedna wartość, jeżeli nie – druga:

```
zmienna=(warunek)?wartoscTRUE:wartoscFALSE.
```

Przykład

Temat: Sprawdzanie, czy podana przez użytkownika liczba jest parzysta:



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

### Pętla while

Zastosowanie pętli pozwala wykonywać kod przez określoną liczbę powtórzeń, aż do spełnienia zdefiniowanego warunku.

W wypadku pętli while polecenia są wykonywane przez cały czas, gdy warunek jest spełniony:

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

#### Przykład

Temat: W poniższym przykładzie zostaną wyświetlone po kolei wszystkie litery od „a” do „y”. Litera „z” nie ukaże się, gdyż warunek straci status spełnionego i wykonywanie kodu z wnętrza pętli while zostanie zatrzymane:

```
litera="a";
while(litera!="z")
{
    document.write(litera);
    kodLitery=litera.charCodeAt() // pobierany jest kod ASCII litery
    kodLitery++ // zwiększamy kod o 1
    litera=String.fromCharCode(kodLitery) // tworzymy literę z kodu ASCII
}
```

### Pętla do ... while

Konstrukcja tej pętli jest bardzo podobna do poprzednio opisanej while, z tym że kod w niej umieszczony zawsze **musi być wykonany co najmniej raz** – nawet gdy warunek nie zostanie spełniony. Jest to związane z tym, że warunek interpreter sprawdza dopiero po wykonaniu poleceń – składnia ma bowiem postać

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

#### Przykład

Sprawdzanie czy konwersja wczytanej liczby nastąpiła poprawnie.

```
do
{
    var liczba_konc = prompt('Podaj liczbę całkowitą końcową=', '');
    k=parseInt(liczba_konc);
}
while (isNaN(p)==true);
```

### Pętla for

Pętla for może być wykorzystana do wykonania pewnego kodu przez określoną liczbę powtórzeń.

Jej składnia ma postać:

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

Określana jest więc początkowa postać zmiennej, wartość, do czasu osiągnięcia której pętla będzie wykonywana, oraz „stopień” zmiany zmiennej w każdej iteracji.

#### Przykład

Temat: W poniższym przykładzie posłużymy się pętlą do ... while, aby „zmusić” użytkownika do podania liczby z przedziału od 1 do 100, a następnie wyświetlimy graficzny, różnokolorowy słupkę, obrazujący stosunek pomiędzy częścią przez niego wybraną a pozostałą:

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

/\* **parseInt** zamienia wartość, którą podał użytkownik, na liczbę całkowitą; pętla wykonuje się dotąd, aż użytkownik wpisze liczbę z przedziału 0-100; jeżeli użytkownik wpisze inny znak lub ciąg znaków, wówczas operacja parseInt zamienia taki ciąg na wartość NaN (Not a Number) - co jest również sprawdzane w jednym z warunków działania pętli \*/

```
slupek="<font color=\"#00FF00\">";
```

```
for (i=0;i<=procent;i++)
{
    slupek+="|";
}
```

```
slupek+="</font><font color=\"#0000FF\">";
```

```
for (i=procent+1;i<=100;i++)
{
    slupek+="|";
}
```

```
slupek+="</font>";
```

```
document.write("<br>" + slupek + " = " + procent + "%");
```

## Rdzenne Obiekty JavaScriptu

### Obiekt Array

Array to po prostu tablica – wykorzystując ją, możemy przechowywać zbiory wartości, kolejno ułożonych, do których odwołujemy się przez nazwę tablicy i indeks.

Tablicę tworzymy, korzystając z operatora new.

Do dyspozycji mamy kilka sposobów:

```
var owoce=new Array("Gruszka","Banan","Ananas");
```

```
var owoce=new Array(3);
owoce[0]="Gruszka";
owoce[1]="Banan";
owoce[2]="Ananas";
```

```
sok=new Array();
sok.smak="Pomidorowy";
sok.pojemnosc="1 litr";
sok.producent="Pomidorek";
sok.length // zwrócony wynik=3;
```

Dana jest tablica myArray i pięciu elementach. W trakcie wykonywania programu konieczne może się okazać usunięcie niektórych elementów z tablicy – tak jak na poniższym przykładzie. Stosując właściwość length, zobaczymy jednak, że usunięcie elementu z tablicy **nie** powoduje skrócenia jej długości:

```
myArray.length // wynik 5
delete.myArray[2]
myArray.length // wynik 5
myArray[2] // wynik: undefined
Listing 27 – usuwanie elementów z tablicy
```

Posługiwanie się tablicami w sposób bardziej zaawansowany wymaga użycia określonych właściwości i metod. Umieszczona poniżej lista pozwoli z pewnością zrealizować wszystkie działania dotyczące przechowywania i modyfikowania danych, jakie tylko mogą przyjść na myśl początkującemu programiście JavaScriptu:

### Właściwości

nazwa	opis
constructor	Zawiera funkcję tworzącą prototyp obiektu.
length	Zwraca liczbę elementów w tablicy.
prototype	Pozwala na dodanie właściwości do tablicy.

### Metody

nazwa	opis
.concat(obiektArray2)	Łączy dwie lub więcej tablic i zwraca nową.
.join("separ")	Zwraca łańcuch znaków elementów tablicy, przedzielonych separatorem.
.pop()	Usuwa i zwraca ostatni element tablicy.
.push("el1",["el2"])	Dodaje element lub więcej na koniec tablicy i zwraca nową długość.
.reverse()	Zwraca tablicę z elementami w odwrotnej kolejności.
.slice(indexPocz [,indexKońc])	Zwraca tablicę utworzoną z części danej.

<code>.sort([funkcjaPorównawcza])</code>	Zwraca tablicę posortowaną.
<code>.splice(index,ile [,el1, el2])</code>	Dodaje i/lub usuwa elementy tablicy.
<code>.toSource()</code>	Zwraca łańcuch znaków, który reprezentuje źródło tablicy.
<code>.toString()</code>	Zwraca łańcuch znaków, który reprezentuje daną tablicę.
<code>.unshift("el1","el2")</code>	Dodaje jeden lub kilka elementów na początek tablicy i zwraca nową długość.
<code>.valueOf()</code>	Zwraca wartość tablicy.

### Obiekt Boolean

Obiekt ten, zgodnie ze swoją nazwą, służy do tworzenia i przechowywania wartości logicznych „prawda” lub „fałsz”. Podobnie jak w wypadku konstruowania tablicy, mamy do wyboru kilka możliwości stworzenia obiektu, którego wartość logiczna wynosi „fałsz” bądź „prawda”:

*//tworzymy obiekt, którego wartość logiczna to fałsz*

```
var b1=new Boolean();
var b2=new Boolean(0);
var b3=new Boolean(null);
var b4=new Boolean("");
var b5=new Boolean(false);
var b6=new Boolean(NaN);
```

*//tworzymy obiekt, którego wartość logiczna to prawda*

```
var b1=new Boolean(true);
var b2=new Boolean("true");
var b3=new Boolean("false");
var b4=new Boolean("prawda");
var b5=new Boolean("fałsz");
var b6=new Boolean("Janek");
```

### Obiekt Date

Posługiwanie się datą i czasem w wypadku wielu skryptów okazuje się niezbędne. Przydatność tego typu danych zdążyliśmy już wykazać w jednym z poprzednich przykładów. Teraz przyjrzymy się dokładniej obiektowi Date, który w JavaScriptcie odpowiada za całość operacji dotyczących czasu. Konstruując go, możemy użyć określonych parametrów lub też tego zaniechać – wówczas zostanie zwrócona aktualna data:

*// bez parametrów – zostanie zwrócona aktualna data*

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

*//liczba milisekund od 1 stycznia 1970 roku*

```
var someDate=new Date(milisekundy);
```

*//data w formacie miesiąc dd, rrrr gg:mm:ss*

```
var someDate=new Date("Jan 13, 1982 21:12:45");
```

*//data w formacie miesiąc dd, rrrr*

```
var someDate=new Date("Jun 23, 1998");
```

*//data w formacie rrrr,mm(od 0 do 11),dd,gg,mm,ss*

```
var someDate=new Date(1981,11,28,14,15,45);
```

*//data w formacie rrrr,mm,dd*

```
var someDate=new Date(1985,10,11);
```

Skuteczne i wydajne operowanie obiektem Date wymaga jeszcze zapoznania się z metodami, za pomocą których będziemy mogli manipulować jego zawartością. Należy także pamiętać o tym, że w wypadku obliczeń dokonywanych na datach najlepiej posługiwać się wartościami milisekund.

## Metody

nazwa	opis
.getTime()	Liczba milisekund od 1970-01-01 od godziny 00:00:00 czasu Greenwich.
.getTimezoneOffset()	Zwraca różnicę czasu lokalnego i GMT.
.getFullYear()	Określony rok minus 1900 – czterocyfrowe oznaczenia dla roku 2000 i kolejnych.
.getFullYear()	Rok w postaci czterocyfrowej.
.getUTCFullYear()	Rok w postaci czterocyfrowej 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 czterocyfrowej.
.setUTCFullYear(val)	Ustawia rok w postaci czterocyfrowej 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 łańcuch znaków z datą – jako liczbę milisekund od 1 stycznia 1970 00:00:00.
.toGMTString()	Zwraca łańcuch znaków z datą ustawioną na GMT.
.toLocaleString()	Zwraca łańcuch znaków z datą lokalną.
.toString()	Zwraca łańcuch znaków z datą.

## Obiekt Math

przykład dla wylosowania pewnej liczby powinniśmy użyć polecenia:

```
liczba=Math.random();
```

### Stałe

symbol	opis
E	Podstawa logarytmu naturalnego – liczba e.
LN2	Logarytm naturalny z 2.
LN10	Logarytm naturalny z 10.
LOG2E	Logarytm z e przy podstawie 2.
LOG10E	Logarytm z e przy podstawie 10.
PI	Liczba Pi.
SQRT1_2	Odwrotność pierwiastka kwadratowego z 2.
SQRT2	Pierwiastek kwadratowy z 2.

### Metody

nazwa	opis
.abs(x)	Wartość bezwzględna liczby.
.acos(x)	Arcus cosinus liczby.
.asin(x)	Arcus sinus liczby.
.atan(x)	Arcus tangens liczby.
.atan2(y,x)	Zwraca kąt od osi x do punktu y.
.ceil(x)	Zwraca najbliższą liczbę całkowitą większą lub równą x.
.exp(x)	Zwraca e do potęgi x.
.floor(x)	Zwraca najbliższą liczbę całkowitą mniejszą lub równą x.
.log(x)	Logarytm naturalny z x.
.max(x,y)	Zwraca większą z dwóch liczb.
.min(x,y)	Zwraca mniejszą z dwóch liczb.
.pow(x,y)	Zwraca x do potęgi y.
.random()	Zwraca losową liczbę rzeczywistą z przedziału 0–1.
.round(x)	Zaokrągla x do liczby całkowitej.
.sqrt(x)	Pierwiastek liczby.
.sin(x)	Sinus liczby.
.cos(x)	Cosinus liczby.
.tan(x)	Tangens liczby.

## Obiekt Number

Obiekt Number pozwala, na wyższym stopniu wtajemniczenia, na bardziej precyzyjne operowanie na liczbach – ich przechowywanie i wyświetlanie. Oprócz przypisywanych wartości obiekt ten zawiera kilka predefiniowanych właściwości, określających na przykład nieskończoność i ujemną nieskończoność. Aby stworzyć obiekt Number, używamy polecenia: `new Number(liczba)`.

Stosując obiekt Number oraz odnoszące się do niego metody, możemy używać zapisu:

- dziesiętnego,
- binarnego,
- ósemkowego
- szesnastkowego.

Musimy jednak pamiętać, że

- wartości dziesiętne nie mogą zaczynać się od zer wiodących,
- szesnastkowe powinny zaczynać się od „0x” lub „0X”,
- ósemkowe natomiast rozpoczynają się zerem wiodącym, a przy ich opisywaniu należy korzystać z cyfr od 0 do 7.
- Liczby możemy także zapisywać z wykładnikiem, na przykład „1e6 = 1\*10^6”.

Poniższy przykład obrazuje możliwość konwersji liczby z pierwotnego formatu na ciąg binarny lub szesnastkowy:

```
var x=30;
var y=x.toString(16)           // wynik="1e"
var z=x.toString(2)            // wynik="11110"
```

### Właściwości

nazwa	opis
MAX_VALUE	Określa największą możliwą do przedstawienia liczbę.
MIN_VALUE	Określa najmniejszą możliwą do przedstawienia liczbę.
NaN	Specjalna wartość określająca, że dane wyrażenie nie jest liczbą.
NEGATIVE_INFINITY	Specjalna wartość określająca nieskończoność.
POSITIVE_INFINITY	Specjalna wartość określająca ujemną nieskończoność.

### Metody

nazwa	opis
.toExponential(precyzja)	Zwraca łańcuch znaków określających liczbę w notacji wykładniczej. Opcjonalny argument precyzja oznacza liczbę miejsc po przecinku.
.toFixed(precyzja)	Zwraca łańcuch znaków określających liczbę w notacji dziesiętnej. Opcjonalny argument precyzja oznacza liczbę miejsc po przecinku.
.toPrecision(precyzja)	Zwraca łańcuch znaków określających liczbę w notacji dziesiętnej, z dokładnością wyznaczoną argumentem precyzja.
.toString(system)	Zwraca łańcuch znaków reprezentujący daną liczbę. Argument system określa w tym wypadku rodzaj zapisu – dziesiętny (domyślny), dwójkowy (2), ósemkowy (8) lub szesnastkowy (16).
.valueOf	Zwraca pierwotną wartość określonego obiektu.

### Obiekt String

Obiekt String to po prostu ciąg znaków. Przy jego tworzeniu możemy więc wykorzystać poznane wcześniej metody „tradycyjne” lub skorzystać z konstruktora postaci: new String("łańcuch znaków"). Podobnie jak w wypadku obiektu Number, przypisane do String metody pozwalają z dużą precyzją operować na danych – tym razem łańcuchach znaków. Wykorzystując je, możemy nie tylko zmieniać wartości ciągów (patrz: „Metody rozbioru łańcuchów”), ale także dodawać do nich określone znaczniki formatujące (patrz: „Metody formatujące łańcuchy”).

### Znaki specjalne

\"	podwójny cudzysłów
\'	pojedynczy cudzysłów (apostrof)
\\	ukośnik lewy
\b	backspace
\t	znacznik tabulacji
\n	znak nowej linii
\r	znak powrotu karetki
\f	znak przewinięcia strony

## Właściwości

nazwa	opis	uwagi
length	Zwraca długość łańcucha (int).	tylko odczyt
prototype	Obiekt – prototyp.	odczyt i zapis;

## Metody rozbioru łańcuchów

[illegible]

`.charCodeAt(n)` Zwraca kod znaku znajdującego się na pozycji (w indeksie) `n` w ciągu.

String.fromCharCode(kod1[,kodn])      Zwraca znaki, których kody zostały przekazane w argumentach.

`.concat(napis2)` Zwraca połączony łańcuch znaków, np.  
`"abc".concat("def")` da wynik `"abcdef"`.

**.indexOf(szukane[,indexPocz])** Zwraca numer indeksu łańcucha, w którym występuje ciąg szukane – gdy łańcuch nie zostanie odnaleziony, zwracane jest -1. Opcjonalny parametr indexPocz określa początkowe miejsce w indeksie, od którego zaczyna się szukanie.

`.lastIndexOf(szukane[,indexPocz])` Przeszukiwanie w tym wypadku rozpoczyna się od końca, także od podanego indeksu w kierunku początku łańcucha, na przykład polecenie `"banany".lastIndexOf("a",2)` da wynik równy 1.

<code>.match(wyrReg)</code> kryteria.	Zwraca tablicę łańcuchów spełniających określone w wyrażeniu regularnym kryteria.
------------------------------------------	-----------------------------------------------------------------------------------

<code>.replace(wyrReg, nowy_lancuch)</code>	Podmienia nowym łańcuchem ciągu odpowiadające wyrażeniu regularnemu.
---------------------------------------------	----------------------------------------------------------------------

<code>.search(wyrReg)</code>	Zwraca liczbę całkowitą określającą pozycję wyrażenia regularnego w łańcuchu.
------------------------------	-------------------------------------------------------------------------------

`.slice(indexPocz[, indexKońc])` Działa podobnie jak `substring()` – polecenia `łańcuch.substring(4, (łańcuch.length-2))` oraz `łańcuch.slice(4, -2)` dadzą ten sam wynik. NN4 IE4

`.split("znakOddzielający" [,maksymalnaLiczba])`      Zwraca tablicę oddzielonych od siebie elementów.

`.substr(początek[,długość])`      Podobnie jak `substring` – drugi parametr określa długość łańcucha.



.substring(indexA, indexB)      Zwraca łańcuch znaków od indexA do indexB.  
 .toLowerCase() Zmienia litery na małe.  
 .toUpperCase() Zmienia litery na wielkie.

#### Metody formatujące łańcuchy

sposób użycia	wynik
napis.anchor("abc")	<A NAME="abc">napis</A>
napis.blink()	<BLINK>napis</BLINK>
napis.bold()	<B>napis</B>
napis.fixed()	<TT>napis</TT>
napis.fontcolor("red")	<FONT COLOR="red">napis</FONT>
napis.fontSize(2)	<FONT SIZE="2">napis</FONT>
napis.italics()	<I>napis</I>
napis.link("http://www.w3c.org")	<A HREF="http://www.w3c.org">napis</A>
napis.big()	<BIG>napis</BIG>
napis.small()	<SMALL>napis</SMALL>
napis.strike()	<STRIKE>napis</STRIKE>
napis.sub()	<SUB>napis</SUB>
napis.sup()	<SUP>napis</SUP>
escape("Ala ma\nkota")	Ala%20ma%0Akota
unescape("Ala%20ma%20kota")	Ala ma kota

## Zdarzenia elementów HTML-a

**Zdarzenia** są to akcje podejmowane przez przeglądarkę w odpowiedzi na działania użytkownika bądź ogólnie – na zmianę stanu dokumentu w danym momencie (np. załadowanie się obrazka czy kodu strony). JavaScript daje nam możliwość kontrolowania i odpowiadania na większość zdarzeń, z jakimi możemy się spotkać w trakcie korzystania z naszej witryny. Najczęściej wykorzystywane zdarzenia przedstawione są poniżej. Należy jedynie pamiętać, że dla prawidłowego korzystania z tych dobrodziejstw

JavaScriptu należy dobrze poznać tak zwany obiektowy **model dokumentu HTML – czyli DOM**.

Określa on sposoby dostępu oraz zmiany zawartości i parametrów poszczególnych elementów struktury strony. Dzięki niemu do dokumentów HTML mogą mieć dostęp wszystkie typy przeglądarek i języków programowania. Zamieszczanie w tym miejscu pełnego opisu elementów DOM mija się z celem. Wszystkie potrzebne informacje zainteresowani programiści JavaScriptu mogą znaleźć pod adresami:

<http://www.w3schools.com/html/dom/default.asp>

[http://www.w3schools.com/js/js\\_obj\\_htmlDOM.asp](http://www.w3schools.com/js/js_obj_htmlDOM.asp)

## Zdarzenia JavaScript

zdarzenie	opis	z obiektami
onclick	Zachodzi, gdy klikamy obiekt.	button, checkbox,
image, layer, link, radio, reset, submit		
onmouseover	Zachodzi, gdy przesuwamy wskaźnik myszy nad obiekt.	image, layer, link
onmouseout	Zachodzi, gdy przesuwamy wskaźnik myszy znad obiektu.	image, layer, link
onmousedown	Zachodzi, gdy nie zwalnimy przycisku myszy, a jej wskaźnik znajduje się nad obiektem.	image, layer
onmouseup	Zachodzi, gdy zwalnimy przycisk myszki, a jej wskaźnik znajduje się nad obiektem.	image, layer
onblur	Zachodzi, gdy opuszczamy obiekt (przejdzie na inny obiekt).	password, select, text,
textarea		

onfocus	Zachodzi, gdy aktywujemy obiekt.	password, select, text, textarea
onchange	Zachodzi, gdy opuszczamy obiekt, którego zawartość została zmieniona.	password, select,
text, textarea		
onselect	Zachodzi, gdy zaznaczamy tekst wewnątrz pola tekstowego.	password, text,
textarea		
onsubmit	Zachodzi, gdy wysyłamy formularz.	form
onreset	Zachodzi, gdy resetujemy formularz.	form
onload	Zachodzi w momencie załadowania dokumentu do przeglądarki.	document, frame
onunload	Zachodzi w momencie opuszczania strony.	document, frame
onabort	Zachodzi, gdy przerwane zostaje ładowanie strony.	document, frame
onerror	Zachodzi, gdy przeglądarka nie może załadować obrazu.	image

## Okna i okienka

W przeszłości bardzo często w linkach stosowało się atrybut `target="_new"`, który nakazywał otwierać strony w nowym oknie. W3C na szczęście stwierdziło, że nie ma podstawy używania tego atrybutu. Dzisiejsze wszystkie [dobre] przeglądarki pozwalają na otwieranie stron w zakładkach i to od użytkownika zależy, co wybierze - nową kartę, okno czy ten sam widok.

Za pomocą Javascriptu możemy otwierać nowe okna, formatować ich wygląd, przysyłać między nimi informacje itp.

Pamiętaj, że większość przeglądarek domyślnie blokuje wszelkie próby otwarcia nowego okna, dlatego lepiej nie uzależniać działania naszej strony od okienek. O wiele lepszym pomysłem jest używanie dynamicznie pozycjonowanych warstw (np DIV).

Otwieranie nowego okna

Standardowa konstrukcja otwierająca okno ma postać:

```
window.open('url do strony','nazwa strony','ustawienia nowego okna')
```

W miejscu url do strony wstawiamy adres do strony która ma być umieszczona w nowo otwartym oknie. W miejscu nazwa strony podajemy nazwę strony, którą będziemy wykorzystywać przy korzystaniu z atrybutu `target` (mało użyteczne). Ma ona znaczenie gdy chcemy korzystać z atrybutu `target`. W miejscu ustawienia nowego okna umieszczamy opcjonalnie ustawienia nowo powstałego okna. Ustawienia te mogą wyłączać nam menu standardowe, ustawiać wysokość i szerokość naszego okna itp.

Poniżej zamieszczam wszystkie możliwe ustawienia okna:

Directories	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa przyciski katalogów
Location	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa pasek adresowy
Menubar	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa menu przeglądarki
Resizable	- wartość yes lub no (1 lub 0): określa czy okno może zmienić rozmiar
Scrollbars	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa paski przewijania
Status	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa pasek statusu
Toolbar	- wartość yes lub no (1 lub 0): pokazuje lub ukrywa standardowy pasek narzędzi
Height	- wartość w pixelach (1 lub 0): ustawia wysokość okna
Width	- wartość w pixelach (1 lub 0): ustawia szerokość okna
Channelmode	tylko IE - wartość yes lub no (0 lub 1): pokazuje lub ukrywa listę kanałów CDF
Fullscreen	tylko IE - wartość yes lub no (0 lub 1): ustala czy okno ma mieć standardowy rozmiar czy też ma być rozciągnięte na cały ekran
Top	- wartość w pixelach : ustawia położenie okna względem góry ekranu
Left	- wartość w pixelach : ustawia położenie okna względem lewej strony ekranu

Wszystkie powyższe właściwości są typu boolean. Oznacza to, że jeżeli nie podamy ich wartości (0 lub 1), to domyślnie przypisana zostanie im wartość `true` (1). W niektórych przeglądarkach trzeba zwracać uwagę, by przy wypisywaniu kolejnych właściwości nie używać spacji. Może to powodować błędy.

Kod funkcji, która otworzy nowe okno może mieć postać:

```
function okno()
{
var NoweOkienko = window.open('strona.html', 'strona',
'toolbar=0,menubar=0,scrollbars=0,resizable=0,status=0,location=0,directories=0,top=20,left=20,
height=300,width=400');
}
```

```
<input type="button" value="Pokaz Okno" id="guzikOkna" />
```

```
<script type="text/javascript">
document.getElementById('guzikOkna').onclick = function() { okno('strona.html') }
</script>
```

### Modyfikacja okna

Dzięki przypisaniu nowego okna pod zmienną, możemy nim w łatwy sposób manipulować. Zasady są identyczne jak dla skryptów w głównym oknie przeglądarki - jedyna różnica polega na odwołaniu:

window.document //dokument w oknie przeglądarki

NoweOkienko.document //dokument w nowym oknie

NoweOkienko.document.getElementsByTagName('body')[0] //pobieramy sekcję body w nowym okienku

NoweOkienko.document.getElementsByTagName('p').style.fontSize = '20px' //ustawiam wielkość czcionki dla wszystkich akapitów w nowym oknie

NoweOkienko.width = '400'; //ustawiam szerokość okienka na 400px

NoweOkienko.moveTo(100,100); //przesuwam okienko do pozycji 100x100

NoweOkienko.close() //zamyka okienko

Jak widać powyżej otwarte okno możemy dodatkowo formatować za pomocą metod i właściwości. Okna on-the-fly

Dzięki temu, że możemy odwoływać się do dokumentu w nowym oknie, możemy dynamicznie generować jego treść:

```
function noweOkno() {
var noweOkno = window.open('', 'okno', 'toolbar=no,location=no,width=200,height=200');
with (noweOkno) {
document.writeln('<html>');
document.writeln('<head>');
document.writeln('<title>Tytuł okienka</title>');
document.writeln('<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />');
document.writeln('</head>');
document.writeln('<body>');
document.writeln('<p>To jest okno utworzone dynamicznie <strong>On-The-Fly</strong></p>');
document.writeln('</body>');
document.writeln('</html>');
document.close() //kończymy zapis do dokumentu
}
}
```

Właściwość opener

Jedną z ciekawszych właściwości otwartych okien jest właściwość `opener`. Dzięki niej możemy się odwoływać do okna, z którego utworzyliśmy nowe okno.

```
opener.location.href = '#rozdzial_1';
```

Powyższy kod przeniesie stronę w głównym oknie do kotwicy `rozdzial_1`.

Metody i właściwości obiektu `WINDOW`

Poniżej przedstawiam metody i właściwości z których możemy korzystać przy pracy z okienkami. Oczywiście nie ze wszystkich jest sens korzystać, ale korzystanie z większości z nich daje dość ciekawe efekty:

Właściwość:    Opis:

- `closed`    - zwraca wartość czy okno jest zamknięte czy nie
- `defaultStatus`    - ustawia/zwraca domyślny status okna
- `document`    - zwraca odwołanie do dokumentu w oknie
- `event`    - zwraca obiekt event (1, 2)
- `frames`    - zwraca odwołanie do obiektu frames
- `history`    - zwraca odwołanie do historii skoków danego okna
- `innerHeight`    - zwraca/ustawia wysokość zawartości okna
- `innerWidth`    - zwraca/ustawia szerokość zawartości okna
- `length`    - zwraca ilość ramek dla danego okna
- `location`    - zwraca/ustawia adres strony w aktualnym oknie
- `locationbar`    - zwraca obiekt locationbar, którego widoczność można ustawić dla danego okna
- `menubar`    - zwraca obiekt menubar, którego widoczność można ustawić dla danego okna
- `name`    - zwraca/ustawia nazwę okna
- `navigator`    - zwraca obiekt navigator
- `opener`    - zwraca odwołanie do okna, z którego utworzono dane okno
- `outerHeight`    - zwraca wysokość zewnętrzną okna przeglądarki
- `outerWidth`    - zwraca szerokość zewnętrzną okna przeglądarki
- `pageXOffset`    - zwraca wielkość kawałka strony, który został ukryty po przesunięciu strony w prawo
- `pageYOffset`    - zwraca wielkość kawałka strony, który został ukryty po przesunięciu strony w dół
- `parent`    - zwraca dokument ze zdefiniowanymi ramkami
- `personalbar`    - zwraca obiekt personalbar, którego widoczność można ustawić dla danego okna
- `screen`    - zwraca obiekt screen (np. `screen.colorDepth`)
- `screenX`    - zwraca lewe położenie okna na ekranie
- `screenY`    - zwraca górne położenie okna na ekranie
- `scrollbars`    - zwraca obiekt scrollbars, którego widoczność można ustawić dla danego okna
- `self`    - zwraca odwołanie do aktualnego okna
- `status`    - zwraca/ustawia tekst na statusie danego okna
- `statusbar`    - zwraca obiekt statusbar, którego widoczność można ustawić dla danego okna
- `toolbar`    - zwraca obiekt toolbar, którego widoczność można ustawić dla danego okna
- `top`    - zwraca odwołanie do okna położonego najwyżej w hierarchi

Metoda:        Opis:

- `alert()`    - wywołuje okno alert
- `back()`    - wraca do strony poprzedniej w historii skoków
- `blur()`    - ustawia aktualne okno pod innymi oknami
- `close()`    - zamyka aktualne okno. Okno główne przeglądarki pyta o potwierdzenie zamknięcia
- `confirm()`    - wywołuje okno confirm
- `focus()`    - ustawia aktualne okno na pierwszym planie (ponad innymi oknami)

forward() - skacze do następnej strony w historii skoków  
home() - wraca do strony startowej  
moveBy(x,y) - przesuwa aktualne okno o dane wartości x,y  
moveTo(x,y) - przesuwa aktualne okno do x,y  
open() - otwiera nowe okno  
print() - drukuje zawartość strony  
prompt() - otwiera okno dialogowe prompt  
resizeBy(x,y) - zmienia rozmiar okna o dane wartości x,y  
resizeTo(x,y) - zmienia rozmiar okna do x,y  
scroll(x,y) - przewija zawartość strony do x,y  
scrollBy(x,y) - przesuwa zawartość strony o dane wartości x,y  
scrollTo(x,y) - przewija zawartość strony do x,y  
stop() - zatrzymuje ładowanie zawartości strony (to samo co po naciśnięciu przycisku STOP w przeglądarce)

## Kilka okien

Możliwe jest otwarcie jednocześnie kilku nowych okien. Należy wtedy oddzielić średnikami (";") kolejne polecenia `window.open('adres', 'nazwa')`:

```
<body onload="window.open('adres1', 'nazwa1'); window.open('adres2', 'nazwa2')">...</body>
```

Trzeba wtedy jednak pamiętać, aby nazwy okien w kolejnych poleceniach były inne (albo ich wcale nie podawać). Wpisanie takich samych nazw, spowoduje otwarcie tylko jednego okna i załadowanie do niego strony ostatniej w kolejności podawania.

## Strona na hasło

Wykorzystując pole typu "password" można w prosty sposób zabezpieczyć wybraną stronę hasłem. W tym celu wystarczy na stronie głównej, do której wszyscy mają normalny dostęp, wstawić następujący kod:

```
<form action="?" onsubmit="window.location.href = this.password.value + '.html'; return false">  
    <input type="password" name="password" />  
    <input type="submit" value="OK" />  
</form>
```

Hasłem w tym przypadku jest nazwa strony bez rozszerzenia, którą chcemy zabezpieczyć.

## Galeria zdjęć

Galeria tworzona jest w ten sposób, że na głównej stronie umieszcza się pomniejszone kopie obrazków oraz odsyłacze, po kliknięciu których następuje wczytanie obrazka w pełnych rozmiarach. Pozwala to uchronić się od wczytywania wszystkich dużych obrazków jednocześnie (użytkownik powiększa tylko te, które mu odpowiadają), a także zachowuje estetykę strony.

Uwaga! Miniaturki grafik muszą być pomniejszone fizycznie, tzn. nie mogą to być oryginalne duże obrazki ze zmniejszonymi rozmiarami wyświetlania za pomocą atrybutów WIDTH oraz HEIGHT. Tylko wstawienie naprawdę pomniejszonych obrazków uchroni od niepotrzebnego wczytywania dużych plików.

Jak wykonać:

- Najczęściej ustala się takie same rozmiary dla wszystkich miniatur, ponieważ ułatwia to utrzymanie estetyki strony.
- Obrazki w galerii zwykle ustawia się w komórkach tabeli (najczęściej bez obramowania), dzięki czemu można je dokładnie ustawić w rzędach i kolumnach.
- Jeżeli zamierzasz umieścić na swojej stronie obszerniejszą galerię grafik, zaleca się podzielenie jej na kilka części i stworzenie kilku podstron z miniaturkami.
- Wczytanie oryginalnego dużego obrazka następuje najczęściej po kliknięciu bezpośrednio jego miniaturki. Aby zastosować taki efekt, należy użyć odsyłacza obrazkowego (podstawowego). Prawie zawsze stosuje się również atrybut `target="_blank"`, który powoduje otwarcie nowego okna przeglądarki.

```
<a target="_blank" href="ścieżka do dużego obrazka">
```

```
</a>
```

## Wykonywanie miniatur w programie gimp

1)Wczytaj pliku którego chcesz wykonać miniatur (opcja Plik → Otwórz).

2)Opcja menu Obraz→Skaluj Obraz...→Okno dialogowe „Skalowanie obrazu” umożliwia wybór jednostki, w jakiej podawane są wymiary obrazu, podanie nowej szerokości obrazu lub wysokości oraz zmianę współczynników X i Y. Domyślnie Szerokość i Wysokość oraz Współczynniki X oraz Y są połączone, co powoduje, że obraz jest skalowany proporcjonalnie (zmiana wysokości z 600 na 300 spowoduje automatycznie zmianę szerokości z 800 na 400). Jeśli chcemy zmieniać niezależnie szerokość i wysokość lub skalowanie obrazu należy rozłączyć klikając na przycisk o wyglądzie spinacza. Na koniec Skaluj

## Aktywne przyciski obrazkowe

Aby wprowadzić na stronę przyciski obrazkowe, które zmieniają swój wygląd, po wskazaniu ich myszką, wystarczy dodać do znacznika `<img />` dwa dodatkowe atrybuty: `onmouseover="..."` i `onmouseout="..."`:

```
<a href="adres"></a>
```

## Ochrona strony

Czasami zdarza się, że chcemy opublikować w sieci jakieś ważne informacje. Zależy nam jednak, aby można się było z nimi zaznajomić tylko bezpośrednio na naszej stronie WWW i nie chcemy udostępniać takich materiałów do dalszej publikacji innym osobom. Chodzi tutaj głównie o opracowania typu: praca dyplomowa, ważny referat lub obszerny artykuł, unikalna grafika nad którą długo pracowaliśmy czy wyniki badań doświadczalnych. Niestety zwykle jedyną ochroną przed plagiatami jest podanie wyraźnej informacji: "Wszelkie prawa zastrzeżone". Jak wiadomo taki zapis nie może uchronić przed podkradaniem naszej własności intelektualnej, na co najlepszym dowodem jest konieczność istnienia serwisów i organizacji zajmujących się plagiatami w sieci, jak np. BOWI - Biuro Ochrony Witryn Internetowych.

Istnieją pewne metody utrudniające kradzież materiałów ze strony WWW. Od razu chciałbym wyraźnie podkreślić, że nie są to prawdziwe zabezpieczenia, a tylko pewne przeszkody, mogące zatrzymać raczej osoby początkujące, które jednak stanowią większość wśród użytkowników sieci. Według mnie nie warto w ten sposób zabezpieczać każdej strony internetowej, ponieważ irytuje to tylko internautów, a i tak prawdopodobnie nie zatrzyma osób bardzo zdeterminowanych, znających w jakimś stopniu język HTML i JavaScript. Ponadto większość z zabezpieczeń przedstawionych na tej stronie działa tylko w Internet Explorerze 5.0 lub nowszym. Pewnym pocieszeniem w tej sytuacji może być fakt, że przeglądarka ta zdobyła zdecydowaną większość rynku (nie dyskutując w tej chwili: słusznie czy nie).

Dalej w tym rozdziale znajdziesz wybrane formy blokady niedozwolonych działań na stronie. Oczywiście można je ze sobą łączyć.

### UWAGA!

Metody ochrony stron WWW opisane w tym rozdziale w większości przypadków działają tylko w Internet Explorerze 5.0 lub nowszym!

#### **Blokada prawego klawisza myszki**

Zabezpiecza przed wybraniem "Pokaż źródło" z menu kontekstowego.

```
<body oncontextmenu="return false">  
...  
</body>
```

Uwaga! Źródło dokumentu nadal będzie można podejrzeć, wybierając odpowiednią opcję z górnego menu przeglądarki. Poza tym nie wszystkie przeglądarki interpretują to polecenie. Częściowo można zlikwidować ten problem, otwierając stronę w nowym oknie bez paska menu. Można również próbować otwierać stronę zawsze w ramkach - wtedy z menu będzie można zobaczyć jedynie źródło "mało ciekawej" strony startowej.

#### **Blokada zaznaczania i kopiowania tekstu**



```
<body onselectstart="return false" onselect="return false" oncopy="return false">
...
</body>
```

### **Blokada przeciągania elementów strony**

Zabezpieczana m.in. przed przeciąganiem obrazków do innego okna lub programu w celu ich zapisania na dysku.

```
<body ondragstart="return false" ondrag="return false">
...
</body>
```

### **Blokada drukowania**

Zaznacz kod Wypróbuj kod

```
<body onbeforeprint="document.body.style.visibility = 'hidden'; alert('Wydruk jest niedostępny!')"
onafterprint="document.body.style.visibility = 'visible'">
...
</body>
```

### **Blokada pamięci podręcznej przeglądarki**

Przeciwdziała zapisywaniu strony i jej elementów (np. obrazów) na dysku użytkownika w tzw. cache'u przeglądarki - zobacz: Cache.

### **Blokada zapisu wybranych zdjęć**

```

```

[Zobacz: Obrazek]

Blokada paska narzędziowego obrazów

Pasek narzędziowy obrazów to zestaw ikon pojawiających się nad dużymi zdjęciami po "najechnaniu" myszką (zwykle oba wymiary grafiki - szerokość i wysokość - muszą wynosić co najmniej 200 pikseli), co umożliwia m.in. wydruk lub zapisanie grafiki na dysku (Internet Explorer 6)

Dla wszystkich zdjęć na stronie (wstaw w nagłówku dokumentu - <head>...</head> - poniższy kod):

```
<meta http-equiv="Imagetoolbar" content="no" />
```

Tylko dla wybranych zdjęć:

```

```

### **Blokada klawisza Print Screen**

Klawisza Print Screen umożliwia wykonanie prostego zrzutu ekranu i późniejszego wklejenia do programu graficznego. Należy wstawić do dokumentu specjalny kod w następujący sposób:

```
<html>
<head>
<script type="text/javascript">
// <![CDATA[
var browser = navigator.userAgent;
var ie = 0;
if (browser.indexOf("MSIE") != -1 && browser.indexOf(" ") == -1) ie =
parseFloat(browser.substring(browser.indexOf("MSIE")+4));

var id_status_blink = 0;
function status_blink(txt)
{
    window.status = txt;
    if (!txt) id_status_blink = setTimeout('status_blink("KLIKNIJ WEWNĄTRZ OKNA
PRZEGLĄDARKI !!!!!")', 250);
    else id_status_blink = setTimeout('status_blink("")', 1500);
    return true;
}

function blur_ie()
{
    document.all["body"].style.visibility = "hidden";
    clipboardData.clearData();
    status_blink("");
}

function focus_ie()
{
    document.all["body"].style.visibility = "visible";
    if (id_status_blink) clearTimeout(id_status_blink);
    window.status = "";
}
```

```
        return true;
    }

    if (ie >= 5)
    {
        window.onblur = blur_ie;
        window.onfocus = focus_ie;
    }
// ]]>
</script>
</head>
<body>
<div id="body">
```

Treść dokumentu

```
</div>
</body>
</html>
```

Oczywiście w nagłówku dokumentu mogą - a nawet powinny (!) - znaleźć się również inne znaczniki (<meta />, <title>...</title> itp.). Nie można zapomnieć o deklaracji DTD. Nic nie stoi również na przeszkodzie, aby dodać atrybuty do znacznika BODY, określające np. kolor tekstu i tła strony. Ważne jest jedynie, aby skrypt został wstawiony w ramy dokumentu tak jak pokazano.

Podsumowanie

Niestety takie rozwiązania zwykle nie są idealne i zawsze znajdzie się droga, aby je obejść. Mogą one natomiast utrudnić życie początkującym "hakerom". Powtarzam jeszcze raz: w większości przypadków stosowanie tych poleceń nie ma dużego sensu, ponieważ nie taka jest idea Internetu. Nie widzę większego celu w zabezpieczaniu w ten sposób wszystkich stron zwykłego serwisu. Irytuje to tylko użytkowników, a jeśli ktoś naprawdę będzie chciał podejrzeć źródło dokumentu, skopiować tekst lub zdjęcie, prawdopodobnie i tak znajdzie sposób, żeby to zrobić. A poza tym zastanów się, czy na Twojej stronie rzeczywiście są aż tak tajne dane, że naprawdę nikt nie może mieć do nich dostępu? Jeśli tak, to uźmysłów sobie, że powyższe sposoby stanowią tylko utrudnienie, a nie prawdziwe i pewne zabezpieczenie.



## Sprawdzanie formularzy w javascript

### formularz

```
<form id="formularz" action="skrypt.php" method="post"
  onsubmit="return sprawdz_formularz()" ">
  <div>
    Podaj swoje imię: <input type="text" name="imie"><br>
    Podaj swoje nazwisko: <input type="text" name="nazwisko"><br>
    <input type="submit" value="Wyślij">
  </div>
</form>
```

### funkcja sprawdzająca w javascript

```
function sprawdz_formularz()
{
    // przypisanie obiektu formularza do zmiennej
    var f = document.forms['formularz'];
    // sprawdzenie imienia
    if (f.imie.value == "")
    {
        alert('Musisz wpisać imię!');
        f.imie.focus();
        return false;
    }
    // sprawdzenie nazwiska
    if (f.nazwisko.value == "")
    {
        alert('Musisz wpisać nazwisko!');
        f.nazwisko.focus();
        return false;
    }
    // formularz jest wypełniony poprawnie
    return true;
}
```

### inny przykład

JavaScript - obsługa formularzy w JavaScript, input, textarea, submit

DZIAŁANIE:

1 Liczba :	<input type="text" value="3"/>
2 Liczba :	<input type="text" value="4"/>
Wynik :	<input type="text" value="12"/>
<input type="button" value="Pomnóż liczby!"/>	

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

```
<!--
```

```
function obslugaFormularza()
```

```
{
```

```
    var liczba1 = window.document.kalkulator.licz1.value;
```

```

        var liczba2 = window.document.kalkulator.licz2.value;
        var iloczyn = liczba1*liczba2;
        window.document.kalkulator.wynik.value = iloczyn;
    }
    //-->
</script>
<form name="kalkulator" method="post">
1 Liczba : <input type="text" name="licz1"/><br/>
2 Liczba : <input type="text" name="licz2"/><br/>
Wynik : <input type="text" name="wynik"/><br/>
<input type="submit" value="Pomnóż liczby!" onclick="obsługaFormularza(); return false;"/>
</form>

```

Popatrzmy na przykład - mamy formularz, dwa pola input gdzie wpisujemy liczby, jedno pole gdzie ma pojawić się wynik iloczynu no i wiadomo przycisk submit, który po kliknięciu wywołuje napisaną przez nas funkcję obsługiFormularza , czyli :

- zmiennej liczba1 jest przypisana wartość z pola formularza o nazwie licz1, czyli uzyskujemy tą wartość poprzez : window.document.nazwa\_formularza.nazwa\_pola.value
- podobnie odczytujemy drugą liczbę
- obliczamy iloczyn
- do pola input o nazwie wynik wpisujemy wynik iloczynu

Zwróć uwagę, że po wywołaniu funkcji mamy return false, po to aby strona się nam nie przeładowała po kliknięciu, tylko żeby wykonał się JS i nic więcej. Należy pamiętać aby nadawać nazwy polom formularza oraz samemu formularzowi - w przypadku PHP zazwyczaj nazywanie samego formularza nie było konieczne.

Jeśli chodzi o inne elementy formularzy HTMLowych działamy podobnie, na przykład dopisanie wartości do jakiegoś elementu textarea :

- window.document.formularz.nazwa\_pola.value = "Tutaj jakiś tekst, który sobie dopisujemy do pola typu textarea";

No i analogicznie można odczytać przez : var odczyt = window.document.formularz.nazwa\_pola.value;

dobra strona o formularzach:

<http://www.doman.art.pl/kursjs/kurs/formularze/formularze.html>

## jQuery

jQuery to jedna z najbardziej popularnych bibliotek/frameworków do javascript. Jej popularność oczywiście znikąd się nie bierze. Dzięki tej bibliotece jesteśmy w stanie o wiele szybciej i sprawniej pisać nasze skrypty, nie martwiąc się przy tym o kompatybilność z różnymi przeglądarkami. Główne zalety tej biblioteki:

Mała objętość skryptów, prosta składnia

Mała objętość samej biblioteki

Bardzo dużo dodatkowych pluginów

Minusy? Pisanie z wykorzystaniem jQuery rozleniwiają ^^.

<http://webmaster.helion.pl/starocie/skrypt/skrypt.htm>

### **Zadanie bez numeru**

Temat: Obliczanie wyznacznika trzeciego stopnia metodą Sarussa. Wczytywanie tablicy danych wg pomysłu ucznia Patryka z 3F.

Wykonaj:

a) uruchom program do wczytywania danych do tablicy( pamiętaj aby po kopiowaniu przenieść złamane linie, [są takie dwie] tak aby tworzyły jedną linię kodu),

```
<html>
<head>
<title>Kreator tabeli</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
<script LANGUAGE="JavaScript">
    function generuj_tabele(kolumny,wiersze)
    {
        var tabela = document.createElement('table');
        for(var x = wiersze;x>0;x--)
        {
            var wiersz = tabela.appendChild(document.createElement('tr'));
            for(var y = kolumny;y>0;y--)
            {
                var komorka = document.createElement('td');
                komorka.appendChild(document.createTextNode(prompt('Podaj
wartość dla '+((wiersze-x)+1)+' ', '+(kolumny-y)+1)+':','0')));
                wiersz.appendChild(komorka);
            }
        }
        document.getElementById('wstaw').appendChild(tabela);
    }
</script>
</head>
<body>
    Wiersze:      <input type="text" name="rows" maxlength="10" value='3' id="rows" />
    Kolumny:      <input type="text" name="cols" maxlength="10" value='3' id="cols" />
    <p>
onClick="generuj_tabele(document.getElementById('cols').value,document.getElementById('rows').v
alue);" style="color: red;">Twórz tabelę!</p>
    <div id="wstaw"></div>
</body>
</html>
```

```
document.write("<p style=\"color:red; \">jjjj</p>");
```

**Konstruktor** metoda w klasie, która nazywa się tak samo jak klasa oraz nie zwraca żadnej wartości, nazywa się konstruktorem i jest wywoływana automatycznie w chwili tworzenia obiektu tej klasy.