Przydatne strony

http://alistapart.com/article/responsive-web-design

- **http://www.w3schools.com/js**
- **http://www.poradnik-webmastera.com/kursy/javascript/**
- **http://developer.mozilla.org/en/docs/JavaScript**
- **http://www.ecma-international.org/publications/**
- **standards/Ecma-262.htm**

Strona responsywna- Responsive Web Design

Po raz pierwszy sformułowania Responsive Web Design użył w swoim w artykule Ethan Marcotte. Założenie jest proste: na rynku pojawia się coraz więcej różnych urządzeń, na których przegląda się strony internetowe. Strona responsywna to taka, która płynnie dostosowuje się do rozdzielczości ekranu. Oznacza to, że zmieniać może się zarówno jej wygląd, nawigacja oraz zawartość. Jedna rzecz będzie zawsze wspólna: adres url. Niezależnie od tego, czy oglądamy ją na małym ekranie telefonu czy dużym ekranie, adres się nie zmienia. Technicznie to ta sama strona, tylko inaczej wyświetlona.

Zalety strony responsywnej:

- jeden panel CMS do zarządzania treścią
- dostosowanie do wielu różnych urządzeń
- jeden adres url niezależnie od urządzenia, na którym czytamy
- jeden kod śledzenia Google Analytics – łatwiejsza analiza danych

Strona responsywna warta jest rozważenia przy projektowaniu nowego serwisu. Ruch z urządzeń mobilnych rośnie bardzo dynamicznie z roku na rok i warto już dziś przygotować się na mobilną rewolucję. Jest to także rozwiązanie polecane właścicielom dużych, dynamicznych serwisów, gdzie konieczność aktualizowania dwóch odrębnych stron oraz pilnowanie przekierowań z wersji mobilną na desktop i vice versa mogłoby być uciążliwe.

```
all /*Wszystkie urządzenia*/
braille /*Dotykowe czytniki brailla*/
embossed /*Drukarka brailla*/
handheld /*Urządzenia przenośne*/
print /*Drukarki*/
projection /*Projektory*/
screen /*Urządzenia ekranowe*/
speech /*Czytniki, syntezatory mowy*/
tty /*Dalekopisy, terminale, itp.*/
tv /*Telewizory*/
```

Listing 3. Dostępne typy mediów

źródło: http://www.w3.org/TR/CSS21/media.html

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {
/* Styles */
}

/* Smartphones (landscape) ----------- */
@media only screen and (min-width : 321px) {
/* Styles */
}

/* Smartphones (portrait) ----------- */
@media only screen and (max-width : 320px) {
/* Styles */
}

/* iPads (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px) {
/* Styles */
}
```

@media only screen and (min-width:150px) and (max-width:700px)

```
  width:92%;
  margin: 10px auto;
  color:#FFF;
}
@media only screen and (min-width:150px) and
(max-width:700px)
{
.body {
      width: 90%;
     font-size: 95%;
}
#wybor1{
display:none;
}
.mainHeader img {
    width:100%;
```

```
 @media print
 {
#obrazek{
background-color:#ff3;
margin:  5px 5px 5px 5px;
padding: 2% 3%;
      }
.bottomcontent,  .mainHeader    {
display:none;
}
.topcontent p{
display:none;
}
asside {
display :none;
}
.mainFooter , .top-sidebar,  .middle-sidebar,
.bottom-sidebar{
display :none;
}
    }
```

CSS-ciekawe efekty graficzne- cd. (transition- czyli efekt przejścia)

CSS3 Transitions

With CSS3, we can add an effect when changing from one style to another, without using Flash animations or JavaScripts.

CSS3 transitions are effects that let an element gradually change from one style to another.

To do this, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Example

Add a transition effect on the width property, with a duration of 2 seconds:

```
div {
    -webkit-transition: width 2s; /* For Safari 3.1 to 6.0 */
    transition: width 2s;
}
```

```
transition-property: width;
transition-duration: 2s;
```

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The transition effect will start when the specified CSS property changes value. A typical CSS property change would be when a user mouse-over an element:

Example

Specify :hover for elements:

```
div:hover {
    width: 300px;
}
```

CSS-ciekawe efekty graficzne- cd. (transition- czyli efekt przejścia)

Multiple Changes

To add transition effects for more than one CSS property, separate the properties with a comma:

Example

Add transition effects on width, height, and transformation:

```
div {
    -webkit-transition: width 2s, height 2s,-webkit-transform 2s;  /* For Safari 3.1 to 6.0 */
    transition: width 2s, height 2s, transform 2s;
}
```

CSS3 Transition Properties

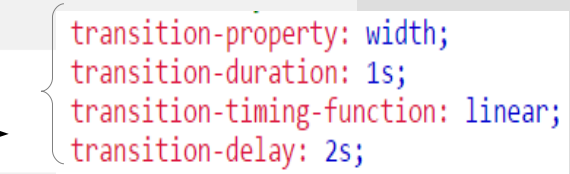
The following table lists all the transition properties:

Property	Description	CSS
transition	A shorthand property for setting the four transition properties into a single property	3
transition-delay	Specifies when the transition effect will start	3
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete	3
transition-property	Specifies the name of the CSS property the transition effect is for	3
transition-timing-function	Specifies the speed curve of the transition effect	3

The same transition effects as the example above. However, here we are using the shorthand transition property:

Example

```
div {
    -webkit-transition: width 1s linear 2s; /* For Safari 3.1 to 6.0 */
    transition: width 1s linear 2s;
}
```



CSS-ciekawe efekty graficzne-cd.-*transform property*

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

Value	Description
none	Defines that there should be no transformation
translate(x,y)	Defines a 2D translation
translate3d(x,y,z)	Defines a 3D translation
scale3d(x,y,z)	Defines a 3D scale transformation
rotate(angle)	Defines a 2D rotation, the angle is specified in the parameter
rotate3d(x,y,z,angle)	Defines a 3D rotation
skew(x-angle,y-angle)	Defines a 2D skew transformation along the X- and the Y-axis
perspective(n)	Defines a perspective view for a 3D transformed element

Przykład- „galeria3Animacja”

```css
body { background: #2F2E4E;
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
font-weight: normal;
margin: 50px; }
.galeria img{
width:300px;
height:200px;
margin:10px;
position:relative;
/* właściwości obrazków przed transformacją */
opacity:0.1;
transform:scale(0.7);
transform:rotate(20deg);
/* all oznacza, że przejściu podlegają wszystkie właściwości, które mogą podlegać
transformacji
 linear- oznacza przejście jednostajne
 1s oznacza przejście w czasie 1s*/
transition: all 1s linear ;
}
```

```css
.galeria img:hover
{
z-index:10;
position:relative;
box-shadow: 2px 2px 150px white;
left:10px;
top: 10px;
/*właściwości, które zmianiają wartości
opacity:1;
transform:rotate(360deg) ;
transform: scale(2);
```

```html
<body>
<div class="galeria">
<img src="img1.jpg" />
<img src="img2.jpg" />
<img src="img3.jpg" />
<img src="img4.jpg" />
<img src="img5.jpg" />
<img src="img6.jpg" />
</div>
</body>
```

JavaScript

Podstawowa trójka internetowa

Podstawą każdej większej strony internetowej są:

- HTML - specyfikacja zawartości strony
- CSS - specyfikacja wyglądu strony
- JavaScript - specyfikacja zachowania strony.

- **JavaScript - wysokopoziomowy, dynamiczny, nietypowany, interpretowany język programowania.**
- **Zawiera wsparcie dla obiektowego i funkcyjnego stylu programowania.**
- **Jest wspierany przez najważniejsze przeglądarki.**

JavaScript

W języku HTML za umieszczanie skryptów JS odpowiedzialny jest element<script>z atrybutami:

- type o wartości „text/javascript”
- opcjonalny atrybut language o wartości „javascript”
- SRC - umożliwia dołączenie kodu źródłowego skryptu znajdującego się w pliku z rozszerzeniem js (np. SRC="lokalizator URL pliku ze skryptem", SRC="skryptl.js"),

```
//dla HTML5
<script>
...instrukcje skryptu
</script>
```

```
//dla HTML4
<script type="text/javascript">
...instrukcje skryptu
</script>
```

atrybut type dla HTML5 jest opcjonalny

Osadzanie JavaScript- dodatkowe uwagi

- Wewnątrz dokumentu HTML

```
<SCRIPT language="JavaScript" type="text/javascript">
<!-
...tutaj umieszczamy skrypt...
//-->
</SCRIPT>
```

- Dołączenie zewnętrznego pliku

```
<SCRIPT language="JavaScript" type="text/javascript"
src="skrypt.js"></SCRIPT>
```

- Wewnątrz znaczników:

```
<A href="javascript:void(0)">Tu klikać</A>
```

JavaScript

Umieszczanie skryptów w oddzielnym pliku

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

- Język skryptowy jest językiem programowania, który umożliwia uruchamianie w specjalnym środowisku programów (skryptów).
- Z reguły te programy są interpretowane a nie kompilowane.
- Przeznaczeniem tych języków jest automatyzacja powtarzalnych zadań, są również stosowane jako języki osadzone w większych aplikacjach.
- Ze względu na swoje właściwości są czasem nazywane językami bardzo wysokiego poziomu lub nawet językami specyfikacji.

Czego nie może JavaScript

- Operacji na lokalnych dyskach użytkownika
- Wykonywania innych programów
- Połączeń z innymi komputerami oprócz ładowania innych stron HTML lub wysyłania e-mail
- Określenia działalności użytkownika w sieci

Osadzanie JavaScript

- Gdzie umieszczać?
  - jeżeli w skrypcie mamy definicje funkcji lub coś co należy wykonać przed ładowaniem strony, to należy skrypt umieścić w nagłówku,
  - jeżeli natomiast skrypt ma wykonać jakieś akcje w trakcie ładowania (np. coś napisać na ekranie) lub później, to skrypt należy umieścić wewnątrz treści dokumentu.

Instrukcja document.write

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write("Hello World!");
</script>

</body>
</html>
```

Komentarze - dwie formy

```
//komentarz 1 linijkowy

/*
------------
To  jest komentarz
kilku  linijkowy
------------
*/
```

Wewnątrz możemy używać znaczników HTML, dbając oczywiście o to, żeby je w odpowiednich miejscach otwierać i zamykać:
document.write("<h1>Strona tytułowa</h1>");
document.write("Spis treści");

Przykład

```
<HTML>
<BODY>
<script language="JavaScript">
document.write("<B>Ten tekst został napisany dzięki JavaScript</B>")
</script>
Ten zaś został napisany w HTML-u
</body>
</htML>
```

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

-**znacznik <script> możemy umieszczać wielokrotnie w dowolnym miejscu pliku HTML (najczęściej w sekcji HEAD)**

-**kod źródłowy skryptu można umieścić w osobnym pliku z rozszerzeniem .js**

Okna dialogowe

ALERT BOX

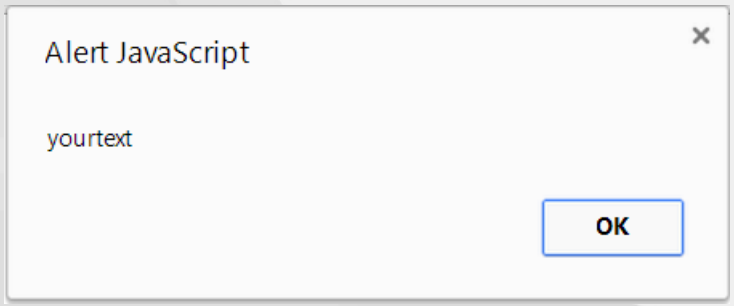
The syntax for an alert box is: **alert("yourtext");**

The user will need to click "OK" to proceed.

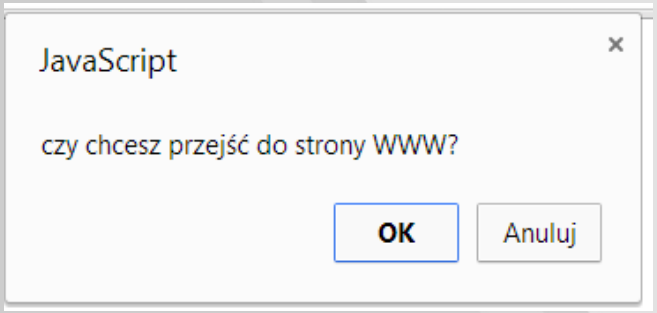
Typical use is when you want to make sure information comes through to the user.

Examples could be warnings of any kind.
(Typical examples are "Adult Content", or technical matters like "This site requires Shockwave Flash plug-in").

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>okno dialogowe Alert</title>
  </head>
<body>
 <script type="text/javascript">
alert("yourtext");
  </script>
 <p> tekst pojawiający się po kliknięciu przycisku
 OK okna informacyjnego </p>
 <p> Zadaniem okna Alert jest przekazanie
 określonej informacji. Nie ma ono wpływu na na
 dalesze działanie skryptu </p>
</body>
</html>
```



Okna dialogowe



CONFIRM BOX:

The syntax for a confirm box is: **confirm("yourtext");**

The user needs to click either "OK" or "Cancel" to proceed.

Typical use is when you want the user to verify or accept something.

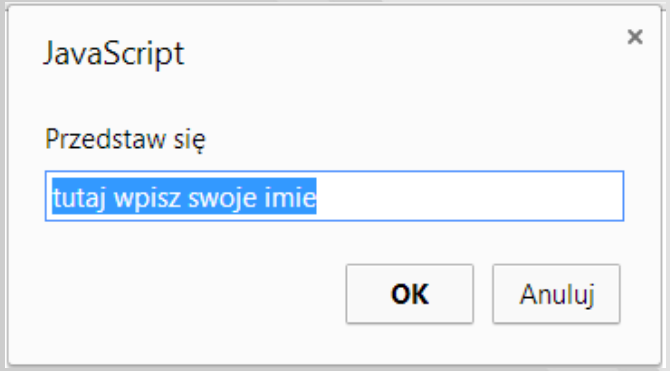
Examples could be age verification like "Confirm that you are at least 57 years old" or technical matters like "Do you have a plug-in for Shockwave Flash?"

- If the user clicks "OK", the box returns the value **true**.
- If the user clicks "Cancel", the box returns the value **false**.

```
if (confirm("Do you agree")) {alert("You agree")}
else{alert ("You do not agree")};
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>okno dialogowe ConfirmBox</title>
  <script type="text/javascript">
confirm("czy chcesz przejść do strony WWW? ");
  </script>
  </head>
<body>
 <p> Okno decyzyjne odpowiada za wyświetlenie treści komunikatu stanowiącego
 argument metody confirm. Udostępnia dwa przyciski OK oraz Anuluj, które po
 wciśnięciu zwracają wartość logiczną true lub false</p>
  <p> Ponieważ skrypt nie ma żadnej funkcji podpiętej do okna decyzyjnego,
  wcisnięcie dowolnego klawisza nie wywoła żadnej reakcji</p>
</body>
</html>
```

Okna dialogowe



PROMPT BOX:

The prompt box syntax is: **prompt("yourtext","defaultvalue");**

The user must click either "OK" or "Cancel" to proceed after entering the text.

Typical use is when the user should input a value before entering the page.

Examples could be entering user's name to be stored in a cookie or entering a password or code of some kind.

- If the user clicks "OK" the prompt box returns the entry.
- If the user clicks "Cancel" the prompt box returns **null**.

Since you usually want to use the input from the prompt box for some purpose it is normal to store the input in a variable, as shown in this example:

```
username=prompt("Please enter your name","Enter your name here");
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>okno dialogowe PromptBox</title>
  <script type="text/javascript">
  var imie;
imie=prompt("Przedstaw się  ", "tutaj wpisz swoje imie");
document.write("<p> Witaj <b style='color:blue; font-size:30px;'>"+imie+"</b> na mojej stronie</p>");
  </script>
  </head>
<body>
 <p> Okno tekstowe wyświetla treść komunikatu stanowiącego argument metody prompt oraz pole umożliwiające wprowadzanie danych przez użytkownika.W trakcie wywoływania okna tekstowego w polu może pojawić się tekst domyślny</p>
</body>
</html>
```

Typy i zmienne

JavaScript jest językiem typowanym dynamicznie

Zmienne

Deklarowanie zmiennej:

```
var zmienna = 10;
var zmienna = "to jest tekst";
var ob = document.Formularz.poleTekstowe;
```

Dla uniknięcia przypadkowych błędów, deklarację zmiennych poprzedzamy słowem **var** (nie jest wymagane).
Zmienne dzielą się na **globalne** i **lokalne**. Globalne są deklarowane poza funkcjami, a lokalne wewnątrz funkcji. Lokalne zmienne są dostępne tylko dla danej funkcji, a globalne są dostępne dla całego skryptu.

Przykład:

```
var zmienna = 69;
zmienna = "nowa wartość" // tutaj nie będzie
  błędu
x = "x = "+40; // zwraca "x = 40"
y = 69-9;     // zwraca 60
z = "69"+9;    // zwraca 699
```

Zmienne i stałe

Typy danych

Javascript udostępnia metodę **typeof()**, dzięki której możesz sprawdzać typ danych

```
var n = 3;
document.write( typeof(n) ) //wypisze się "number"

var s = "napis";
document.write( typeof(s) ) //wypisze się "string"
```

- Deklaracje zmiennych
  - przez przypisanie wartości
    `x=5;`
  - przez słowo **`var`**
  - jeśli zmiennej nie zostanie przypisana wartość to przyjmuje wartość **`undefined`**
- Deklaracja stałych
  - stała nie może zmieniać wartości lub być przedeklarowana
  - **`const wroclaw = "071";`**

Rodzaje zmiennych:

Liczbowe - czyli zawierające wartość liczbową

```
var zmienna = 10.5;
var zmienna = 100;
```

Logiczne (boolean) - zawierające wartość będącą prawdą lub fałszem

```
var zmienna = true;
var zmienna = false;
```

Znakowe - zawierające wartość będącą ciągiem znaków (stringiem)

```
var zmienna = "Moje imie to Grzegorz";
var zmienna = "Jakiś tekst";
```

undefined - zmienna niezadeklarowana. Ten typ zmiennej służy do sprawdzania, czy zmienna w ogóle istnieje.

Undefined and Null

The value of a variable with no value is **undefined**.

Variables can be emptied by setting the value to **null**.

Example

```
var cars;              // Value is undefined
var person = null;     // Value is null
```

obiekt Number

Javascript udostępnia obiekt **Number**, który zawiera takie wartości jak:

- **MAX_VALUE** - maksymalna wartość
- **MIN_VALUE** - minimalna wartość
- **NaN** - nie liczba
- **NEGATIVE_INFINITY** - specjalna wartosć nieskończoności (zwracana w przypadku overflow)
- **POSITIVE_INFINITY** - specjalna ujemna wartosć nieskończoności (zwracana w przypadku overflow)

JavaScript numbers can be written with, or without decimals:

Example

```
var x = 34.00;     // A number with decimals
var y = 34;        // A number without decimals
```

Extra large or extra small numbers can be written with scientific (exponent) notation:

Example

```
var x = 123e5;     // 12300000
var y = 123e-5;    // 0.00123
```

POSITIVE_INFINITY Property

Example

Return positive infinity:

```
Number.POSITIVE_INFINITY;
```

The result will be:

```
Infinity
```

Always Use Number.POSITIVE_INFINITY

POSITIVE_INFINITY a static property of the JavaScript Number object.

You can only use it as Number.POSITIVE_INFINITY.

Using x.POSITIVE_INFINITY, where x is a number or a Number object, will return undefined:

Example

```
var x = 100;
x.NEGATIVE_INFINITY;
```

The value of x will be:

```
undefined
```

Infinity

Infinity (or -Infinity) is the value JavaScript will return if you calculate a number outside the largest possible number.

Example

```
var myNumber = 2;
while (myNumber != Infinity) {          // Execute until Infinity
    myNumber = myNumber * myNumber;
}
```

Division by 0 (zero) also generates Infinity:

Example

```
var x =  2 / 0;          // x will be Infinity
var y = -2 / 0;          // y will be -Infinity
```

```
<script>
    var x = 2/0;
    var y = -2/0;
    document.write(x+"<br>");
    document.write(y+"<br>");
</script>
```

Infinity
-Infinity

Infinity is a number: typeOf Infinity returns number.

Example

```
typeof Infinity;        // returns "number"
```

NaN - Not a Number

NaN is a JavaScript reserved word indicating that a value is not a number.

Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number):

Example

```
var x = 100 / "Apple";  // x will be NaN (Not a Number)
```

However, if the string contains a numeric value , the result will be a number:

Example

```
var x = 100 / "10";     // x will be 10
```

You can use the global JavaScript function isNaN() to find out if a value is a number.

Example

```
var x = 100 / "Apple";
isNaN(x);               // returns true because x is Not a Number
```

Watch out for NaN. If you use NaN in a mathematical operation, the result will also be NaN.

Example

```
var x = NaN;
var y = 5;
var z = x + y;          // z will be NaN
```

NaN is a number: typeOf NaN returns number.

Example

```
typeof NaN;             // returns "number"
```

Konwersja danych

JavaScript nie wymaga od ciebie abyś deklarował typ zmiennych. Przykładowo możesz utworzyć zmienną typu liczbowego o nazwie np. zmienna:

```
var zmienna = "to jest napis " + 20; //zwróci "to jest napis 20"
```

```
var zmienna = "20" + 1; //zwróci "201"
```

Gdy od zmiennej typu znakowego w której skład wchodzą tylko znaki cyfr odejmiemy zmienną typu liczbowego wówczas wykonamy normalne równanie:

```
var zmienna = "21" - 1; //zwróci 20
```

```
var zmienna = "20a" - 1; //zwróci NaN
```

Gdy od zmiennej typu znakowego w której skład wchodzą nie tylko znaki cyfr ale i liczb odejmiemy zmienną typu liczbowego wówczas otrzymanym wynikiem będzie NaN (Not-A-Number)

Możemy też wymusić konwersję poprzez rzutowanie wartości na dany typ danych za pomocą funkcji: **parseInt()**, **parseFloat()**, **String()**