

Tehnici Web

CURSUL 5

Semestrul I, 2017-2018
Carmen Chirita

<https://sites.google.com/site/fmitehniciweb/>

CSS-layout : multicolorane

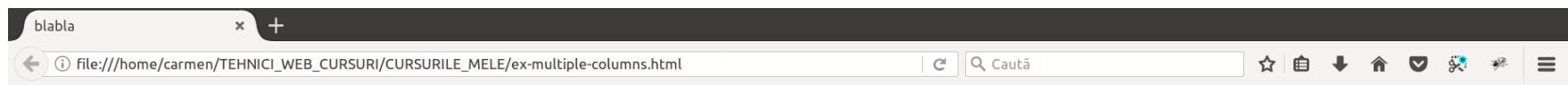
Permite aranjarea textului pe mai multe coloane.

`column-count` : nr.de coloane
`column-gap`: spațiul dintre coloane
`column-rule-style`: stilul liniei care desparte coloanele
`column-rule-width`: grosimea liniei dintre coloane
`column-rule-color`: culoarea liniei dintre coloane
`column-rule`: grosime stil culoare
`column-span`: cate coloane ocupa un element
`column-width`: latimea unei coloane

`column-count` si `column-width` se determina una pe cealalta

```
#col1{ column-width: 100px;
        column-gap: 6px;
        column-rule: 3px solid red;
}
#col2{ column-count: 4;
        column-gap: 10px;
        column-rule: 4px dotted black;
```

```
<body>
<h1>multiple colums</h1>
<p id="col1">Lorem ipsum dolor
sit amet,....</p>
<p id="col2">Aliquam lectus
odio,.....<p>
</body>
```



multiple columns

<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam tempus rhoncus ante tincidunt commodo. In in nibh vitae enim interdum lobortis. Duis euismod condimentum lacus, eget hendrerit nisi semper vel. Proin tincidunt tincidunt suscipit rhoncus. Nam at iaculis dolor. Proin at facilisis leo, non ullamcorper urna. Mauris auctor sagittis nibh, ac viverra nisl lobortis ut. Morbi et arcu eget purus tristique interdum nec, Duis ac urna leo. Maecenas sagittis rutrum enim ac suscipit porttitor. Sed turpis nunc, luctus id interdum nec, suscipit sed lectus. Nam sagittis elementum enim imperdiet ullamcorper. Fusce laoreet fermentum dui in tempor. Nam rhoncus, tellus sed rutrum porta, nisi leo euismod ante, et luctus elit sem eu mi. Duis adipiscing nibh sit amet egestas dapibus. Fusce id quam purus. Quisque fringilla ante at risus tempus, in tristique nisi accumsan. Morbi vel sollicitudin sem, non placerat ligula. Ut imperdiet, libero in iaculis egestas, eros ante pharetra lacus, a cursus nibh lacus et elit. In eget orci id metus fermentum lacinia. Sed in blandit est, at vehicula tellus. Integer vulputate elit in ligula gravida bibendum. Curabitur venenatis metus in vulputate iaculis. Quisque augue nisi, bibendum vitae nulla eget, egestas facilisis sem. Morbi malesuada, dui vel molestie congue, sem dolor suscipit risus malesuada. neque. Donec molestie felis vel mi mollis, nec suscipit risus malesuada. </p>	<p> Aliquam lectus odio, dignissim nec auctor iaculis, tristique sit amet dolor. Donec a velit consectetur, tincidunt nisl non, luctus ligula. Interdum et malesuada fames ac ante ipsum primis in faucibus. Integer consectetur justo porttitor urna suscipit, a bibendum velit semper. Vestibulum fringilla fringilla odio, id sagittis justo iaculis ac. Nunc id faucibus velit, eu pretium nisi. Pellentesque facilisis dictum augue sit amet sodales. Nunc sed ornare diam, quis blandit nulla. In ut arcu elit. Fusce molestie sagittis ipsum in ultrices. Nunc sapien nibh, malesuada eget malesuada </p>	<p> ut, elementum at nibh. Ut sodales est ut elit tincidunt congue. Fusce tincidunt nunc ut elit sollicitudin interdum ut euismod elit. Nulla molestie tincidunt lectus et viverra. Cras arcu risus, faucibus et tellus in, condimentum tincidunt risus. Vivamus imperdiet odio et nisl mattis eleifend sed sagittis ipsum. Morbi arcu dolor, ultricies eu orci ac, pharetra vulputate enim. Mauris cursus tempor rutrum. In hac habitasse platea dictumst. In ut placerat felis, ac tempor erat. Curabitur a leo eget mi semper consequat sed sit amet diam. Proin pulvinar fringilla est, ut gravida leo </p>	<p> porttitor vel. In non eros a sapien adipiscing scelerisque. Maecenas nisl leo, scelerisque nec orci suscipit, tristique pharetra ligula. Sed et eleifend felis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Integer sit amet lectus sit amet leo blandit malesuada scelerisque eu eros. In urna turpis, vehicula eu orci semper, dapibus rhoncus neque. Mauris non tortor enim. Suspendisse hendrerit purus non ipsum feugiat, quis fermentum dui ultricies. Aliquam eu sapien a elit ullamcorper venenatis. Vestibulum tincidunt odio </p>	<p> vitae quam condimentum, sit amet fringilla enim vehicula. Nulla ullamcorper mauris nec tellus rutrum, quis facilisis urna pulvinar. Nulla urna turpis, consequat eget posuere quis, dictum sed nibh. Sed tristique eros vitae facilisis pulvinar. Maecenas est risus, posuere in arcu et, auctor sagittis metus. Sed augue leo, placerat quis leo in, semper vehicula dolor. Nam sit amet consectetur velit, eget tempor purus. Cras urna purus, feugiat ut pharetra in, tempus ac augue. Integer libero risus, venenatis eu sodales at, cursus pharetra ligula. </p>
--	--	---	---	---

CSS3- gradienti

Gradient : tranzitie de la o culoare la alta

Linear gradient

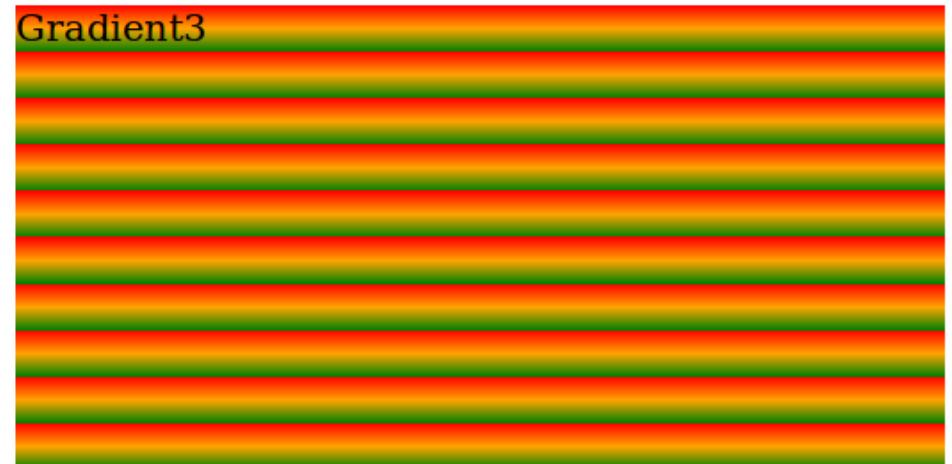
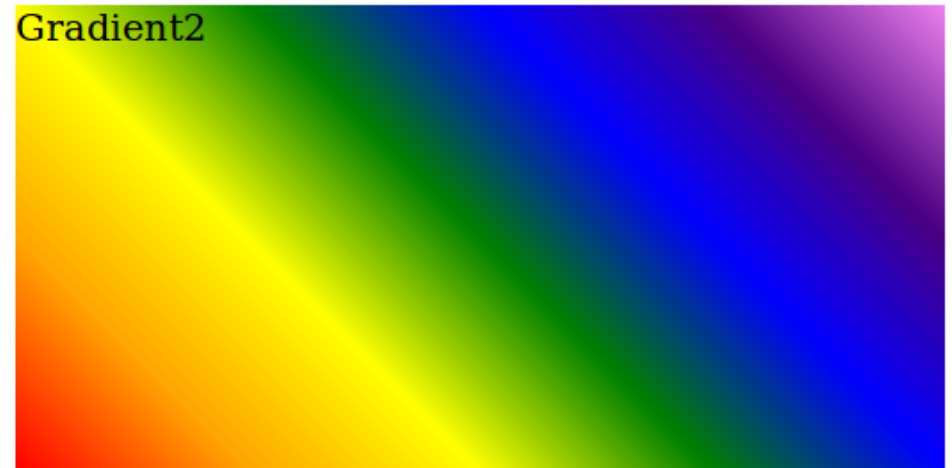
```
background: linear-gradient(directia, culoare1,culoare 2,...)
```

```
directia : to bottom (implicit) /*  
          to right ,  
          to left,  
          to top,  
          to bottom right,  
          unghi( 45deg) */
```

```
background: linear-gradient(to right, black, white);  
background: linear-gradient(45deg, black, white);  
background: repeating-linear-gradient(red, yellow 10%, green 20%);
```

```
div{ width: 400px; height:200px;}
#grad1 {
  background: linear-gradient(to right
bottom, red, orange, yellow, green,
blue, indigo, violet);
}
#grad2 {
  background: linear-gradient(45deg,
red, orange, yellow, green, blue, indigo,
violet);
}
#grad3 {
  background: repeating-linear-
gradient(red, orange 5%, green 10%);
}
```

```
<body>
<div id="grad1">
Gradient1 </div>
<hr>
<div id="grad2">
Gradient2 </div>
<hr>
<div id="grad3">
Gradient3 </div>
</body>
```



Radial gradient

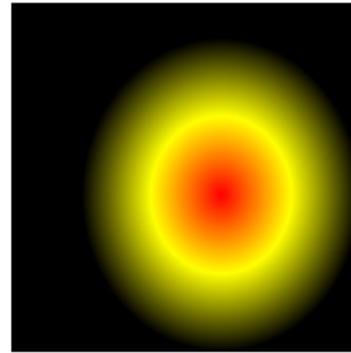
background: radial-gradient (shape size at position, start-color, ..., last-color)

shape : ellipse (implicit) /* circle*/

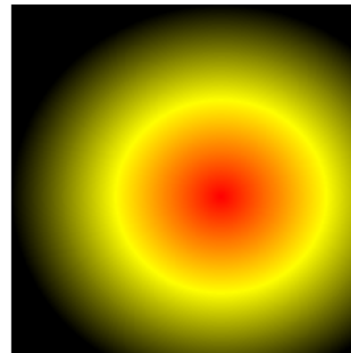
size : closest-side,
farthest-side,
closest-corner,
farthest-corner (implicit)

position: center (implicit)
10%,60%

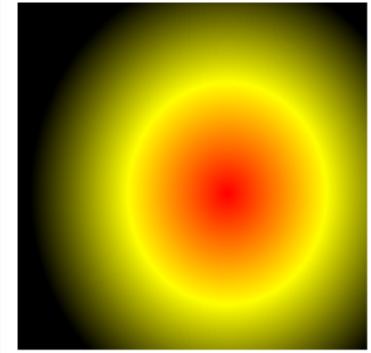
closest-side:



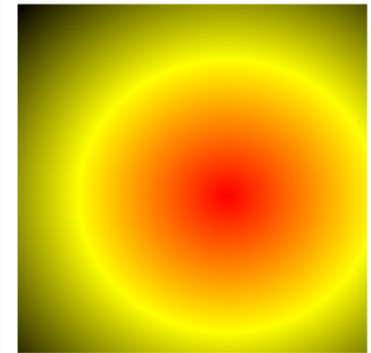
farthest-side:



closest-corner:



farthest-corner (t



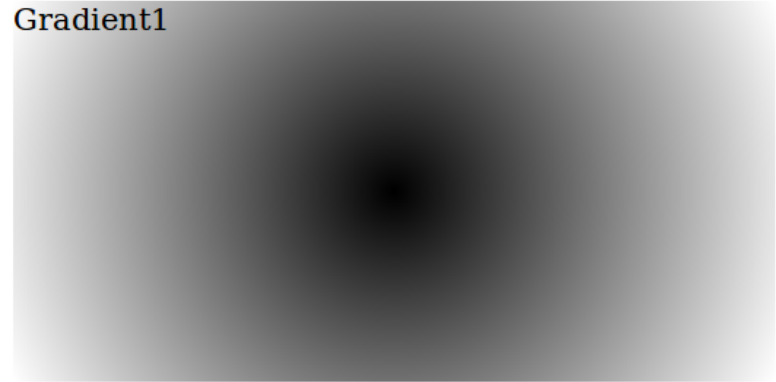
```
#gr1 {background:
radial-gradient (circle at 50% 50%,
black, white);}
```

```
#gr2 {
background: radial-gradient( ellipse,
red, blue, aqua,white);}
```

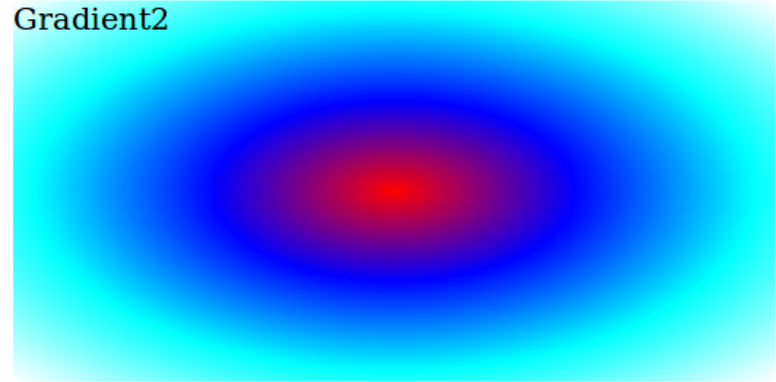
```
#gr3{ background: repeating-radial-
gradient(red, yellow 10%, green 15%);}
```

```
<body>
<div id="gr1">
Gradient1 </div>
<hr>
<div id="gr2">
Gradient2 </div>
<hr>
<div id="gr3">
Gradient3 </div>
</body>
```

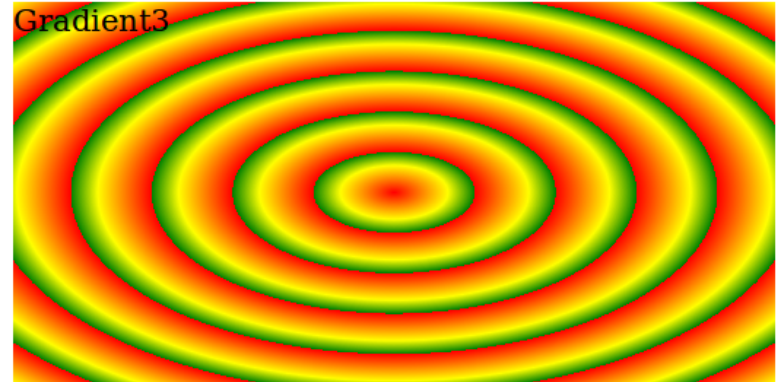
Gradient1



Gradient2



Gradient3



CSS3: border-radius (border cu colturi rotunjite)

border-radius: reuniune a proprietatilor
border-top-left-radius,
border-top-left-radius,
border-bottom-right-radius,
border-bottom-left-radius,

Patru valori: 15px 50px 30px 5px:



Trei valori: 15px 50px 30px:



Doua valori: 15px 50px:



O valoare: 30px:



JavaScript-istoric

- Inventat de Brendan Eich în 1995, la Netscape (denumit initial **Mocha** și **LiveScript**);
- Implementat de browserul Netscape Navigator sub numele de **JavaScript**;
- Adaptat de Microsoft în 1996 și denumit **Jscript**;
- Standardizat în 1997 de ECMA (European Computer Manufacturer's Association) sub numele de **ECMAScript**;

ultima versiune: **ECMAScript 2017**

JavaScript-caracteristici

- Este un limbaj de scripting pentru pagini web (pe partea de client)
- Este bazat pe prototipuri
- Este un limbaj interpretat (scriptul este executat direct, fără compilare prealabila)
- Este „loosely typed”

JavaScript și HTML

- Orice pagina Web este reprezentată în memorie ca un arbore de obiecte
(Document Object Model – DOM)
- JavaScript poate accesa elementele HTML prin intermediul DOM-ului.

element HTML → Obiect în JavaScript

Atribut al unui element HTML → Proprietate a obiectului în JavaScript

Codul JavaScript poate fi plasat:

Oriunde in documentul HTML, folosind tagul **script**

```
<head>  
<script type="text/javascript">  
/* cod JavaScript */  
</script>  
</head>
```

Într-un **fișier extern** (NumeFisier.js) care este importat in documentul HTML

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

```
<head>

<script type="text/javascript" >

var s = 0;
for (var i=1; i< 10 ; i++) s=s+i;
alert(s);
</script>

<meta charset="utf-8">
<title> JavaScript </title>
</head>
<body>

</body>
```

```
<head>

<script type="text/javascript" >

function suma(x) {
var s = 0;
for (var i=1; i< x ; i++) s=s+i;
return s;}

alert('Suma este ' +suma(10));
</script>

<meta charset="utf-8">
<title> JavaScript </title>
</head>
<body> </body>
```

JavaScript este **CASE SENSITIVE** și folosește setul de caractere **Unicode**

Identificatorii:

- denumesc variabile, cuvinte cheie, funcții, etichete;
- formați din: cifre, litere, _, \$;
primul caracter: **litera, _, \$**

; separator (ex. a = 3; b = 4;)

{ } bloc de instrucțiuni

```
/* comentariu  
   pe mai multe linii  
*/
```

```
// comentariu pe o singura linie
```

Tipuri de date:

primitive: number, string, boolean, null, undefined

tipul object

Obiecte predefinite: Array, Function, String, Number, Boolean, Math, Date,...

Instrucțiuni:

=, if/else, for, switch, while, return, { inst1; inst2;}

Funcții:

function Nume(param1,param2,..) {corpul functiei}

Variabile în JavaScript

- Pot fi declarate explicit folosind cuvântul cheie **var**; optional variabila poate fi initializata cu o valoare;

variabilele declarate astfel pot fi locale (declarate în interiorul functiilor) sau globale (declarate în afara functiilor).

```
var a = 1, b = 5;  
var r = 2.5;  
var mesaj= " acesta este un string";  
var x;
```

- Atribuirea unei valori unei variabile nedecarate înainte cu **var** creaza o variabila **globala**

```
y = "sunt globala" // variabila globala
```


Variabile în JavaScript

- Variabilele se mai pot declara folosind cuvântul cheie **let**; acesta declara o variabila locala vizibila doar în blocul, instrucțiunea sau expresia în care este folosită.

ex.1

```
var x=5;
var y=1;
var gamma = 0;
if (x > y) {
  let gamma = 12 + y;
  x = gamma * x;
}
console.log(x); // 65
console.log(gamma) // 0
```

ex.2

```
function letTest() {
  let x = 1;
  if (true) {
    let x = 2;
    console.log(x); // 2
  }
  console.log(x); //
1
}
```

In ECMAScript 6 au fost introduse declaratii de constante

```
const Max =100;
```

Variabilele au tipuri dinamice

JavaScript este "loosely typed"

Tipul variabilei nu este specificat explicit, dar poate fi aflat cu `typeof(x)`

Tipul unei variabile nedeclarate este undefined.

```
typeof(null) // "object"
```

```
typeof(undefined) // "undefined"
```

```
> x=2
2
> typeof(x)
"number"
> x="Hello"
"Hello"
> typeof(x)
"string"
> typeof(y)
"undefined"
```

Scopul variabilelor: zona din program în care sunt declarate variabilele; exista scop global și scop local

In JavaScript scopul este creat de functii;
orice functie creaza un scop.

ex.1

```
var x = "globala"; //globala
function func1() {
    var x="locala";
    return x;
}
func1();    // => "locala"
alert(x);   // => "globala"
```

ex.2

```
x = "globala"; //globala
function func1() {
    x="locala";
    return x;
}
func1();    // => "locala"
alert(x);   // => "locala"
```

```
// scopul global
function fA (){
    //scopul A
    function fB () {
        // scopul B
    }
}
```

Scop Lexical

Toate variabilele/obiectele/functiile
declarate de o functie parinte sunt
vizibile in descendenti ei

ex.3

```
var x = "globala";
function func1()
{
    var x="locala1";
    function func2()
    {var x="locala2";
    return x;
    }
    return func2();
}
func1(); // => "locala2"
```

Hoisting in JavaScript

Domeniul de vizibilitate al unei variabile coincide cu functia in interiorul careia a fost definita;

Inainte de a fi executat, codul JavaScript este parsat si "rearanjat" a.i. toate declaratiile de variabile (nu si operatiile de atribuire) sunt mutate (ridicate) la inceputul zonei de vizibilitate (adica la inceputul functiei).

```
var x = 5; //globala
function host() {
  if (x == 5) {
    var x = 10;
    x++;
  }
  alert(x);
}
func(); // va afisa undefined
alert(x); // va afisa 5
```

echivalent cu

```
var x = 5; //globala
function host() {
  var x; // variabila locala
  if (x == 5) { // x este undefined
    x = 10;
    x++;
  }
  alert(x);
}
func(); // va afisa undefined
alert(x); // va afisa 5
```

Tipul number (reprezentare binara pe 64 biti)

```
var a = 4;  
var r = 34.7;
```

Operatorii aritmetici specifici:

+ - * / % ++ --

Conversia de tip automata

```
x = "2" * 7; // 14  
y = "2" + 7; // "27"  
z = parseInt("2") + 7; // 9  
t = "2" * "7"; // 14  
u = 2 + 3 + "4"; // "54"  
z = "2" + 3 + 4; // "234"
```

Obiectul [Math](#)

Math.PI //=> 3.14

Math.pow(2,3) //=> 8

Math.round(4.7) //=>5

Math.random() // intre 0 si 1

Math.sqrt(-1) // => NaN

> x=5

5

> y=3.4

3.4

> typeof(x)

"number"

> typeof(y)

"number"

> parseInt("5hello")

5

> parseFloat("3.4Hello")

3.4

> parseInt("3.4Hello")

3

Tipul string (sir de caractere scris intre ' ' sau " ")

```
var s = "Ana Popescu";  
var t = 'Ana Popescu';  
var pnume = s.slice (0, s.indexOf(" "));  
var fnume=s.slice(s.lastIndexOf(" ")+1,s.length);
```

Proprietăți și metode: `length`, `charAt()`, `indexOf()`, `lastIndexOf()`, `replace()`, `split()`, `toLowerCase()`, `toUpperCase()`, `concat()`, ..

Concatenarea: `"numarul" + "1"`, `"id"+1`

Caractere speciale: `\'` `\"` `\n` `\t` `\v` `\b` `\\`

Accesarea unui caracter: `s[0]`, `s.charAt(0)`, `s.charAt[s.length-1]`

```
var x = "abcde";  
alert(x[0]);  
x[0] = 'v';  
alert(x[0]); // => a
```

Un string nu este un array de caractere


Tipul boolean: true si false

Orice valoare poate fi convertita explicit folosind obiectul predefinit Boolean

```
var nume = Boolean(valoare);  
  
true === Boolean("adevarat") // => true  
false === Boolean("") // => true
```

Operatori logici pentru tipuri primitive: > < <= >= && || ! == === != !==

Alte valori pot fi folosite ca si Boolene:
pentru fals: 0, "", NaN, null, undefined
pentru adevarat: orice alta valoare



verifica si tipul operanzilor

Operatorul conditional

conditie ? expr1 : expr2

```
function fact(n){  
  return (n <= 2) ? n: n*fact(n-1);  
}
```

Tipurile undefined si null

variabilele care nu au primit încă o valoare au tipul undefined

```
var x  
x == undefined // true  
typeof(x) // undefined
```

null lipsa unei valori (intentionata)

```
var x=null  
typeof(x) //object  
null == undefined // true  
null === undefined // false
```


Tipul object

Un obiect este o colectie de perechi nume-valoare.
Daca valoarea este o functie atunci proprietatea
se numeste metoda.

```
var ob = {prop1: val1, prop2: val2, ... ,prop-n: val-n};
```

Accesarea proprietatilor:

```
ob.prop1; // val1
```

```
ob["prop1"]; // val1
```

Exemple:

```
var complex = { re: 5, im: 3}; \\ complex.re, complex.im
var student = {nume: "Ionescu",
               nota1:9,
               nota2:10,
               media:function(){
                   return (this.nota1 + this.nota2)/2;
               }
            }
```

```
student.nume    \\ Ionescu
student.nota1   \\ 9
student.nota2   \\ 10
student.media() \\ 9.5
student.media   \\ functia
```

Tipul object

Toate datele din tipurile primitive in afara de tipul object sunt transmise prin valoare.

Datele de tip object sunt transmise prin referinta.

```
var a = {nume: "Ana"} // object  
var b = a; // a și b refera aceeași zona  
b.nume = b.nume + " Popescu"; // se modifica și b și a  
alert(a.nume ); // "Ana Popescu"
```

```
var s = "Ana"; // string  
var t = s; // t copiaza valoarea lui s  
t = t + " Popescu" // se modifica doar t  
alert(s) // => "Ana"
```

Crearea obiectelor

- Prin obiect literal:

proprietatile, metodele, împreună cu valorile lor sunt enumerate între acolade; se creează un singur obiect.

```
var pers= {nume:"Popescu",prenume:"Andrei",vârsta:20}
```

Crearea obiectelor

- Cu ajutorul obiectului generic

se apeleaza constructorul `new Object()` și se adauga apoi proprietatile și metodele; se creeaza un singur obiect

```
var pers= new Object();  
pers.num="Popescu";  
pers.prenume="Andrei";  
pers.varsta=20;
```

Crearea obiectelor

- Cu ajutorul unui constructor de obiecte

Se definește o funcție `constructor(parametrii)` care apoi va fi apelată cu `new constructor(parametrii)` pentru fiecare obiect care va fi creat

```
function pers(n,p,v) { this.num= n;  
                        this.prenume=p;  
                        this.varsta=v;  
                        }
```

```
var p1=new pers("Popescu","Andrei",20);  
var p2=new pers("Ionescu","Bogdan",20);
```

Proprietăți și metode globale

Pot fi folosite împreună cu orice variabilă și obiect creat în JavaScript

Proprietate	Descriere
Infinity	O valoare numerică care reprezintă infinitiv pozitiv/negativ
NaN	O valoare "Not-a-Number"
undefined	Indică o variabilă căreia nu i-a fost atribuită o valoare

Metode

```
isNaN() // Determină dacă valoarea este un număr invalid  
parseInt() //converteste un sir într-un intreg  
parseFloat() //converteste un sir într-un număr zecimal  
Number() //converteste un obiect într-un număr  
String() //converteste un obiect într-un sir
```

Obiecte predefinite

Obiecte wrapper
pentru tipurile primitive

Boolean, String, Number

se pot crea obiecte noi cu
new

Object

Array, Set, Map

Function

RegExp

Date

se pot crea obiecte noi cu
new

```
var y = new Number(123);  
typeof(y) // object
```

```
var ob = new Object();  
ob.x =1; ob.y=2; // ob ={x:1, y:2}
```

```
var d= new Date(2015,3,1);  
alert(d.getUTCDay()); // 2 (ziua din  
săptămâna (0-6))
```


Array

```
var v = new Array();  
v[0] = "a"; v[1] = "b";
```

```
var v = new Array("a","b");
```

```
var v = [6,4,7,3];
```

Proprietati si metode

```
v.length // 4  
v.push(10); // => v=[6,4,7,3,10]  
v.pop(); // => [6,4,7,3]  
v.shift(); // => [4,7,3]  
v.unshift(10); // => [10,4,7,3]  
v.sort(); // => [3,4,6,7]
```

tipul elementelor nu e fixat

```
v=["hi",2,[5,7]]
```

```
var s = "azi este joi";
```

```
var a = s.split(" ");  
// a = ["azi","este","joi"]
```

```
a.reverse();  
// a = ["joi","este","azi"]
```

```
var s = a.join('/');  
// s="joi/este/azi"
```

Set (introduces in ECMAScript6)

Metode: add(val), has(val), size(), values(), keys(), delete(val), clear()

```
var m = new Set();  
m.add(1).add(2).add(3);  
if (m.has(3)) {let s=0;  
                for (e of m.values()) s=s+e;  
                alert(s);}
```

Instructiuni: for

```
for (initializare; conditie; update) {  
  instructiuni;  
}
```

```
for (var i = 0; i < 9; i++) {  
  console.log(i);  
}
```

```
for (variabila in obiect) {  
  instructiuni  
}
```

```
var pers = {nume:"Popescu", prenume:"Andrei",  
            varsta:30};  
  
var text = " ";  
var x;  
for (x in pers) {  
  text += pers[x] + " ";  
}
```

```
for (item of iteratii) {  
  instructiuni  
}
```

```
var tablou = [10, 20, 30];  
for (var x of tablou) {  
  x += 1;  
  console.log(x);  
}
```

Instrucțiuni: while, do, if, switch

```
while (conditie) {  
    instructiuni;  
}
```

```
do {  
    instructiuni  
} while (conditie);
```

```
if (conditie) instructiune;
```

```
if (conditie) instructiune  
else instructiune;
```

```
switch (expresie)  
{  
    case 1:  
        bloc 1  
        break;  
    case 2:  
        bloc 2  
        break;  
    .....  
    default : bloc  
}
```

Funcții

Sintaxa:

```
function nume(arg1, arg2,..., argn) {  
    instructiuni;  
    return valoare; // nu neaparat  
}
```

În Java Script o funcție poate fi apelata cu un numar variabil de parametrii

```
function suma(a,b) {  
    return a+b;}  
suma(2,3); // 5  
suma(); // NaN  
suma(2); // NaN  
suma(3,4,1,5,6,7) // 7
```

```
function fun() {  
    return arguments.length;}  
fun(2,"sss", 5); // 3
```

```
function f(a,b){var z = a;  
                a = a +b.n;  
                b.n = b.n +z;}  
var x = 3;  
var y = {n:3};  
f(x,y);  
alert(y.n); //=> 6  
alert(x);   //=> 3
```

Parametrii se transmit prin valoare;
valoarea unei variabile obiect este
o referinta.

Funcții

```
function suma()  
{  
    var s=0;  
    for(var i=0; i<arguments.length; i++)  
        s+=arguments[i];  
    return s;  
}
```

```
suma(); // 0
```

```
suma(5); // 5
```

```
suma(2,3,4,5); //14
```

Funcții anonime

```
function (arg1,...,argn){  
    instructiuni;  
}  
  
var fun = function () {  
    return arguments.length;  
}
```

Funcțiile anonime
sunt expresii
care întorc valori
de tipul funcție

```
function f(x){return x+1};  
var f1 = f;  
var x=f(3);  
var x1=f1(3);
```

```
var g = function (x){return x+1};  
var y = g(3);  
typeof(g) // "function"
```

```
var h = (x)=>{x+1}
```

arrow functions
ECMAScript 6

Obiectele browserului: window

Metodele `prompt` si `alert`

`prompt(text, default-text)`: afiseaza o caseta de dialog care cere utilizatorului sa introduca informatii

`alert(mesaj)`: afiseaza o caseta de alertare care contine un mesaj și un buton OK

```
var x = prompt("nr1");  
var y = prompt("nr2");  
alert(typeof(x));  
alert(x+y);  
alert(parseInt(x)+parseInt(y));
```



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

var n=prompt("nr= ");
alert(suma(n));

function suma(x) {
var i;
var s = 0;
for (i=1; i< x ; i++) s=s+i;
return s;};

</script>
```

