

Tehnici Web

CURSUL 10

Semestrul I, 2018-2019
Carmen Chirita

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

Scalable Vector Graphics SVG

<http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.htm>

<https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial>

SVG

- componentele primitive ale imaginii sunt forme geometrice, care sunt descrise matematic; aceasta asigura independenta de rezolutie, scalabilitatea
- gramatica XML pentru grafica

Scalable Vector Graphics SVG

- fișierele SVG pot fi editate cu editoare specializate (Adobe Illustrator, Inkscape), dar și ca fișiere XML
- fișierele SVG pot fi incluse în fișierele HTML folosind ``
``
- imaginile (codul) SVG pot fi incluse direct în fișiere HTML

```
<svg width="450" height="500" id="elsvg">
```

```
...
```

```
</svg>
```

Exemplu

```
<body>
<h1>SVG inclus in HTML</h1>

<svg width="100" height="100"> //container pentru grafica
  <circle cx="50" cy="50" r="40" stroke="red" stroke-width="5" fill="yellow" />
</svg>

</body>
```

SVG inclus in HTML



Elemente SVG predefinite

<rect>

<circle>

<ellipse>

<line>

<polyline>

<polygon>

<path>

<text>

Elementele SVG `<defs>`, `<use>`

SVG permite ca obiectele grafice să fie definite pentru o reutilizare ulterioară.

Elementele de referință se vor defini în interiorul unui element `<defs>` și pot fi referite cu `<use>`.

De exemplu, `<defs>` poate fi utilizat la crearea gradientilor

```
<svg width="500" height="600">

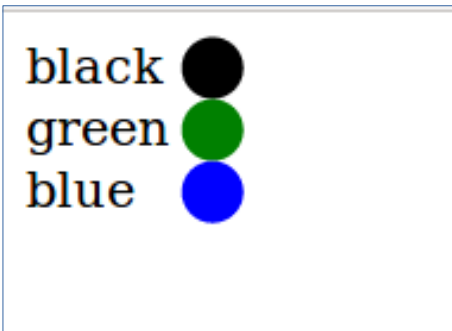
<defs>

  <circle id="Port" cx="10" cy="0" r="10"/>

</defs>

<text y="15">black</text>
<use x="50" y="10" href="#Port" style="fill: black;"/>
<text y="35">green</text>
<use x="50" y="30" href="#Port" style="fill: green;"/>
<text y="55">blue</text>
<use x="50" y="50" href="#Port" style="fill: blue;"/>

</svg>
```



SVG Rectangle - <rect>

utilizat pentru a crea un dreptunghi și variații ale unei forme de dreptunghi

```
<body>  
  
<svg width="400" height="200">  
  <rect width="200" height="100" style="fill:green; stroke-width:3; stroke:red" />  
</svg>  
  
</body>
```



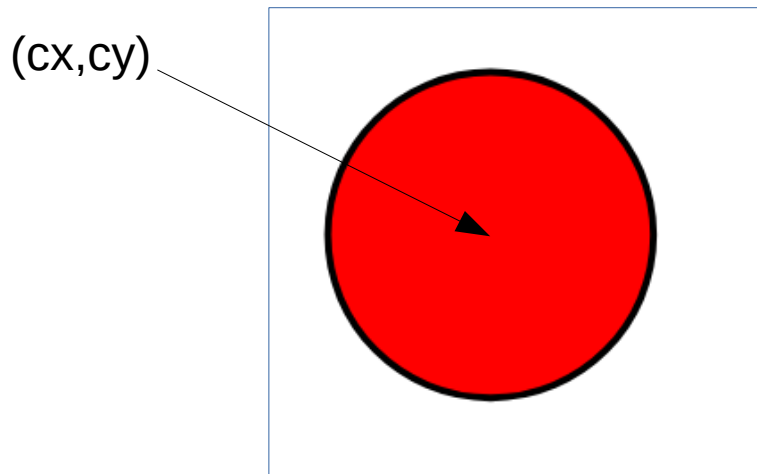
Atribute specifice: **width** , **height**, **x**, **y**
Atributele **rx** și **ry** //colturi rotunjite
Atributul **style**: defineste stil CSS

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, **stroke-opacity**:0-1,**opacity**:0-1

SVG Circle - <circle> (deseneaza un cerc)

```
<circle cx="100" cy="100" r="70"  
style="stroke:black; stroke-width:3; fill:red" />
```



Atribute specifice:

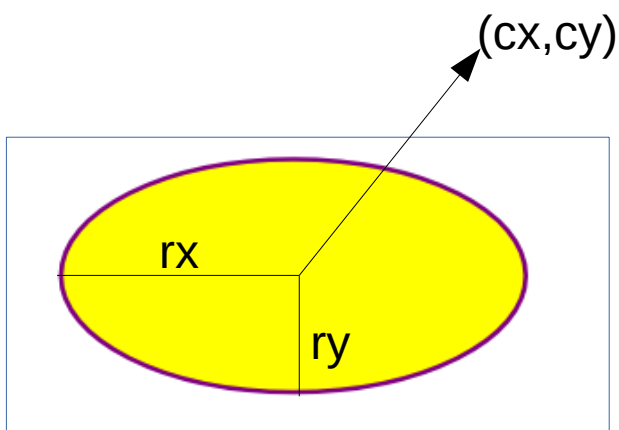
cx, cy //coordonatele centrului
r //raza cercului

Proprietati CSS:

fill: culoarea continutului
stroke: culoarea conturului
stroke-width: grosimea conturului
fill-opacity:0-1, **stroke-opacity**:0-1,**opacity**:0-1

SVG Ellipse - <ellipse> (deseneaza o elipsa)

```
<ellipse cx="200" cy="80" rx="100" ry="50"  
style="fill:yellow; stroke:purple; stroke-width:2" />
```



Atribute specifice:

cx, cy //coordonatele centrului elipsei

rx //raza orizontala

ry //raza verticala

Proprietati CSS:

fill: culoarea continutului

stroke: culoarea conturului

stroke-width: grosimea conturului

fill-opacity:0-1, **stroke-opacity**:0-1,**opacity**:0-1

SVG Line - <line> (deseneaza o linie)

```
<line x1="0" y1="0" x2="200" y2="200"  
style="stroke:rgb(255,0,0);stroke-width:2" />
```



Atribute specifice:

x1, y1 //coordonatele de inceput

x2, y2 //coordonatele de sfarsit

Proprietati CSS:

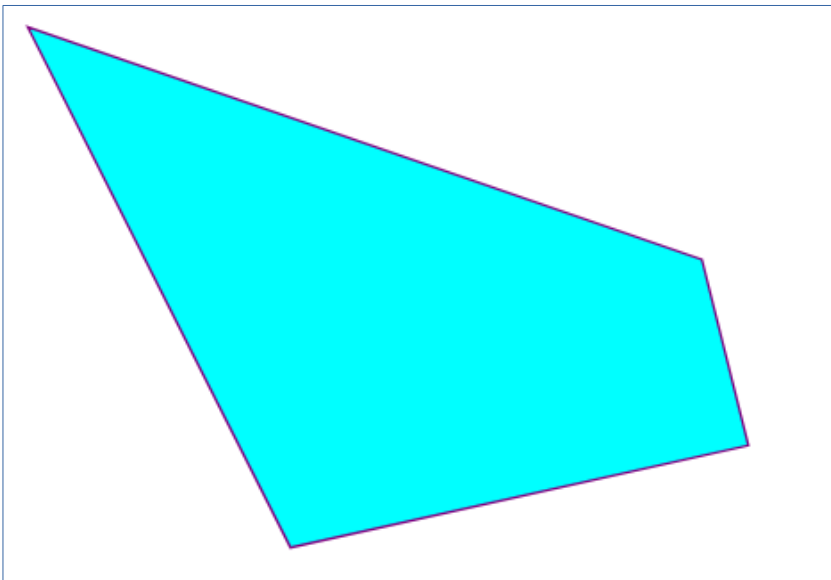
stroke: culoarea liniei

stroke-width: grosimea liniei

SVG Polygon - <polygon>

este utilizat pentru a crea o imagine care conține cel puțin trei laturi.

```
<polygon points="10,10 300,110 320,190 123,234"  
style="fill:cyan; stroke:purple; stroke-width:1" />
```

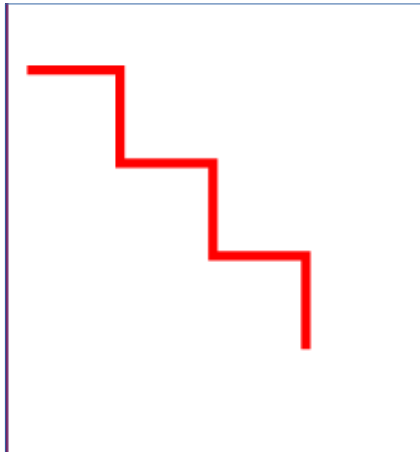


Atribute specifice:
points : //coordonatele varfurilor

SVG Polyline - <polyline>

este utilizat pentru a crea orice formă care constă numai din linii drepte

```
<polyline points="0,40 40,40 40,80 80,80 80,120 120,120 120,160" style="fill:white; stroke:red; stroke-width:4" />
```

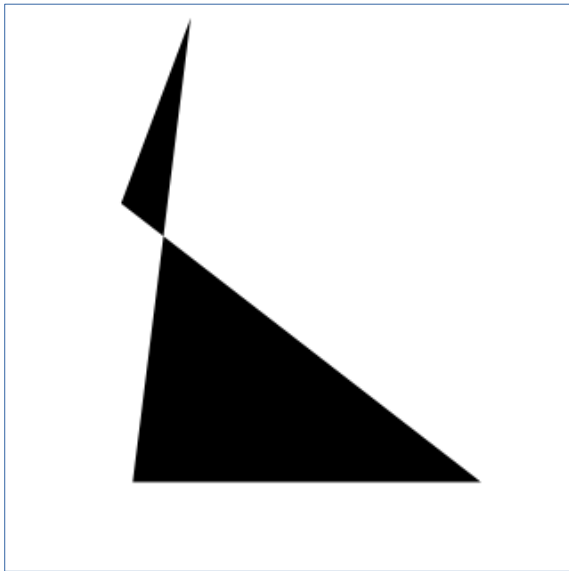


Atribute specifice:
points : //coordonatele varfurilor

SVG Path - <path>

este utilizat pentru a crea forme complexe combinand linii, arcuri, curbe, etc.

```
<path d="M100 0 L75 200 L225 200 L70 80 Z" />
```



Comenzi

M = moveto

L = lineto

Z = closepath

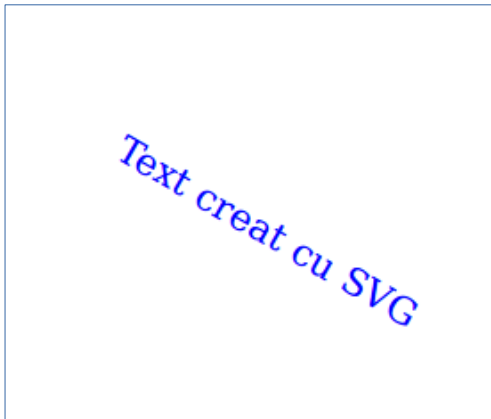
H = horizontal lineto

V = vertical lineto

A = elliptical Arc

SVG Text - <text> (defineste un text)

```
<text x="50" y="50" fill="blue" transform="rotate(30 20,40)">Text  
creat cu SVG</text>
```

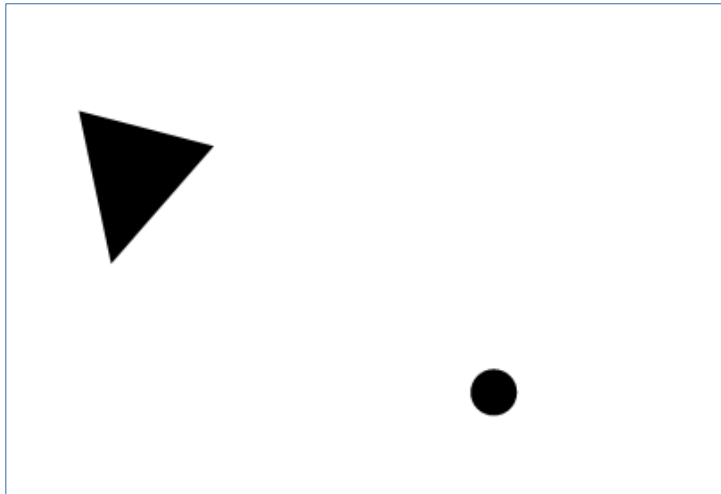


Atribute specifice:
x,y : //coordonatele de inceput

Animatie SVG

<animate>
<animateTransform>
<animateMotion>

```
<polygon points="60,30 90,90 30,90">  
  <animateTransform attributeName="transform"  
    type="rotate"  
    from="0 60 70"  
    to="360 60 70"  
    dur="10s"  
    repeatCount="4"/>  
</polygon>
```



```
<circle cx="230" cy="32" r="10" >  
  <animateMotion  
    path= "M0,0 L50,132 L-50, 132 L0,0"  
    begin="3s" dur="10s"  
    repeatCount="indefinite" />  
</circle>
```

svg.html

Gradienti in SVG- <linearGradient>

```
<svg>
<defs> //defineste elemente SVG care vor
fi utilizate ulterior

<linearGradient id="myLG"
  x1="0%" y1="0%" x2="0%" y2="100%"
  spreadMethod="pad">
  <stop offset="0%" stop-color="#00ff00" stop-opacity="1"/>
  <stop offset="100%" stop-color="#0000ff" stop-opacity="1"/>
</linearGradient>

</defs>

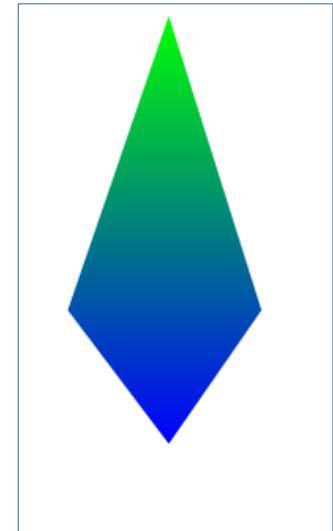
<path d="M 230,32 L181,175 L 230,240
        L 275,175 L 230,32 "
      style="fill:url(#myLG)" />
</svg>
```

Gradienti:

Pe verticala: $x1 = x2$

Pe orizontala: $y1 = y2$

Unghiulari: $x1 \neq x2, y1 \neq y2$



Gradienti in SVG -<radialGradient>

```
<svg height="150" width="500">
```

```
<defs>
```

```
  <radialGradient id="grad1" cx="20%" cy="30%" r="30%"  
  fx="50%" fy="50%">
```

```
    <stop offset="0%" style="stop-color:rgb(255,255,255);  
stop-opacity:0" />
```

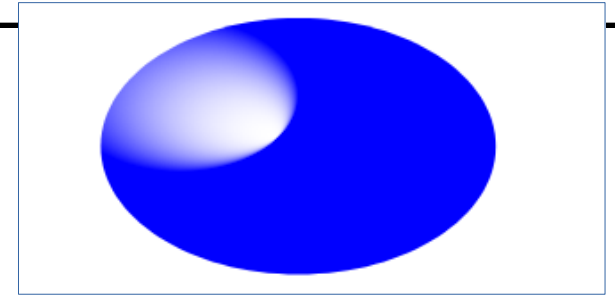
```
    <stop offset="100%" style="stop-color:rgb(0,0,255);  
stop-opacity:1" />
```

```
  </radialGradient>
```

```
</defs>
```

```
<ellipse cx="200" cy="70" rx="85" ry="55" fill="url(#grad1)" />
```

```
</svg>
```



jQuery (bibliotecă, creată de [John Resig](http://jquery.com/download/))
<http://jquery.com/download/>

Caracteristici:

- Metode eficiente pentru selecția elementelor; metode specifice de prelucrare a elementelor selectate.
- Metode pentru evenimente.
- Metode pentru efecte speciale și animații.
- Metode Ajax.

Cum se introduce JQuery?

```
<head>
```

(descărcarea de pe site-ul <http://jquery.com>)

```
<script type="text/javascript" src="jquery-3.3.1.min.js"></script>
```

sau

(includerea din CDN)

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>
```

```
</head>
```

Sintaxa jQuery

\$ sau jQuery() este unica functie globala si ea intoarce un **obiect jQuery**.

\$(selector).action()

selectarea elementelor si efectuarea unor acțiuni asupra elementelor

myjqname.js

```
$(document).ready(function(){  
    Cod JQuery  
});
```

sau

```
$(function(){  
    Cod JQuery  
});
```

Se executa codul JQuery după încărcarea paginii

jQuery

```
$(document).ready(function(){  
  
  $("p").css("background-color","gray");  
  
  $(".test").hide();  
  
  $(this).click(function(){ $(this).hide();  
  });  
});
```

`$("p")` selecteaza toate elementele de tipul "p"

`$(".test")` selecteaza toate elementele cu clasa "test"

`$(document)`, `$(this)` intoarce un obiect **jQuery** caruia
ii pot fi aplicate metodele specifice **jQuery**

jQuery

```
$(document).ready(function(){  
  
var art = $("article");  
  
$("p", art). hover(function(){ alert(this.id);});  
  
});
```

`$(“selector”,el)` intoarce multimea descendentilor
elementelor el care verifica “selector”

jQuery

```
$(document).ready(function(){  
  
var art = $("article");  
  
$("p", art). click(function(){  
    $(this).css("background-color","gray");  
});  
});
```

`$(this)` intoarce un obiect **jQuery** caruia
ii pot fi aplicade metodele specifice **jQuery**

jQuery selectors

selectorii jQuery permit selectarea și manipularea elementelor HTML.

se bazează pe selectorii CSS existenți și, în plus, există anumiți selectori personalizați.

toți selectorii din jQuery încep cu semnul dolar și paranteze: \$ ().

selectorii CSS

```
$("#div"); $("#myid"); $(".myclass"); $("input[type='text']");
```

```
$("li:first-child"); $("li:last-child"); $("p:first-of-type"); $("p:nth-of-type(2)")
```

```
 $("div > p") ; $("div p") ; $("div + p") ; $("div ~ p")
```

Selectori JQuery

<http://api.jquery.com/category/selectors/jquery-selector-extensions/>

`$(":animated");` // selecteaza elementele care sunt intr-o animație la momentul executarii selectorului

`$(":button");` // selecteaza elementele input cu type="button"
`$(":radio");` // selecteaza elementele input cu type="radio"

`$(":header");` // selecteaza elementele header h1-h6

`$(":gt(n)");` // selecteaza elementele cu index >= n

`$(":not(selector)");` //selecteaza toate elementele, cu excepția celor definite de selector

```
$("#p:not(.intro)").css("background-color", "yellow");
```


jQuery: event handlers

Metode de evenimente:

.click(), .dblclick(), .submit(), .hover(), .keydown()

```
$("#p").click(function(){ $(this).hide();});
```

```
$(window).keydown(function(event){alert(event.key);});
```

jQuery chaining

```
$("#b1").click(function(){  
    $("#article p").hide();}  
    .dblclick(function(){  
        $("#article p").show();});
```

Metoda on()

ataseaza una sau mai multe funcții handler de ev. pentru elementele selectate

```
$("#p").on("click", function(){  
    $(this).hide();  
});
```

```
$("#p").on({  
    mouseenter: function(){  
        $(this).css("background-color", "lightgray");  
    },  
    mouseleave: function(){  
        $(this).css("background-color", "lightblue");  
    },  
    click: function(){  
        $(this).css("background-color", "yellow");  
    }  
});
```

Traversare DOM cu jQuery

`children()` // copii directi ai elementului selectat

`find()` // toți descendentii elementului selectat

`parent()` //parintele elementului selectat

`parents()` //stramosii elementului selectat

`parentsUntil()` //stramosii pana la elementul
specificat

`siblings()` //toti fratii elementului selectat

`next()`, `nextAll()`, `nextUntil()`

`prev()`, `prevAll()`, `prevUntil()`

```
$("#li").parent().css("border", "3px solid red");
```

Traversare DOM cu jQuery - metode de filtrare

first(), last(), eq(), filter(), not()

first() // returneaza primul element al elementelor selectate

last() // returneaza ultimul element al elementelor selectate

```
$("#div").first(), $(".unu").last
```

eq() // returneaza elementul cu un anumit index
al elementelor selectate (index >=0)

```
$("#p").eq(1)
```

filter() // selecteaza elementele care indeplinesc o conditie

not() // opusa metodei filter()

```
$("#ul li").filter(":gt(2)"), $("#ul li").not(":gt(2)")
```

Manipulare DOM cu jQuery

creare/ stergere/modificare

`$(target).method(content)`

`append()` //inserare la sfârșitul elem. selectat
`prepend()` //inserare la începutul elem. selectat
`after()` //inserare după elem. selectat
`before()` //inserare înainte de elem. selectat
`remove()` //stergere elementului selectat
`empty()` //sterge copii elementului selectat

Manipulare DOM cu jQuery

```
$(document).ready(function(){
    $("#apend").click(function(){
        $("ol").append("<li>Element nou</li>");
    });
    $("#prepend").click(function(){
        $("ol").prepend("<li>Element nou</li>");
    });

    $("#after").click(function(){
        $("ol").after("<p>Element nou</p>");
    });
    $("#before").click(function(){
        $("ol").before("<p>Element nou</p>");
    });
    $("#empty").click(function(){
        $("ol").empty();
    });
});
```

append.html

```
<body>
<button id="apend">Append</button>
<button id="prepend">Prepend</button>
<button id="after">After</button>
<button id="before">Before</button>
<button id="empty">Empty</button>

<ol>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
<li>Item 4</li>
</ol>

</body>
```

Append

Prepend

After

Before

Empty

1. Item 1
2. Item 2
3. Item 3
4. Item 4

Selectarea și modificarea conținutului unui element

Metodele `text()`, `html()`, `val()`

pentru campurile formelor

```
<p id="myelem">  
Text vechi  
</p>  
<form> <input id="myfin" value="valoare initiala">  
</form>
```

```
$('#myelem').text(); // intoarce conținutul ca text fara marcate
```

```
$('#myelem').html(); // intoarce conținutul ca text cu marcate
```

```
$('#myelem').text("Text nou"); // modifica conținutul
```

```
$('#myelem').html("<b> Text nou </b>"); // modifica conținutul interpretand marcatele
```

```
$('#myfin').val (); // intoarce valoarea atributului value
```

```
$('#myfin').val ("valoare noua"); // modifica valoarea atributului value
```

Manipularea atributelor unui element

`attr("nume-atr"),`

`attr("nume-atr","valoare")`

`attr({atribut1:val1, atribut2:val2,...})`

`removeAttr("nume-atr")`

```
$("#flori").attr("alt", "flori de vara");
```

```
$("#fmi").attr("src","http://www.fmi.unibuc.ro");
```

```
$("#a").attr("target","_blank");
```

```
$("#a").removeAttr("target");
```

```
$("#img").attr({width:150px;height:200px;});
```


Modificarea proprietatilor CSS

css("nume-proprietate", "valoare"),
hasClass("nume-clasa"),
addClass("nume-clasa"),
removeClass("nume-clasa")

```
.vechi {color:blue;}
```

```
.nou {color:red;}
```

```
<div id="deschimbata">  
Text  
</div>
```

```
de= document.getElementById("deschimbata");  
  
function schimba(de){  
  if ($(de).hasClass("vechi"))  
    {$(de).removeClass("vechi");  
     $(de).addClass("nou");}  
}
```

jQuery effects (metode pentru efecte speciale)

show("fast", [callback]), hide(), toggle(),
fadeIn(), fadeOut(), fadeToggle(), fadeTo()

slideUp(), slideDown("slow"), slideToggle(),
animate()

jQuery effects (metode pentru efecte speciale)

Exemplul 1 (show_hide.html)

```
$(document).ready(function(){  
    $("#hide").click(function(){  
        $("p").hide(2000);  
    });  
    $("#show").click(function(){  
        $("p").show(3000,function(){alert("Hello");});  
    });  
});
```

Paragrafele se vor ascunde la click pe Hide

Paragrafele vor apărea la click pe Show

Hide

Show

<body>

<p>Paragrafele se vor ascunde la click pe Hide</p>

<p>Paragrafele vor apărea la click pe Show</p>

<button id="hide">Hide</button>

<button id="show">Show</button>

</body>

jQuery effects (metode pentru efecte speciale)

Exemplul 2 (fade.html)

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#div1").fadeToggle();  
        $("#div2").fadeToggle("slow");  
        $("#div3").fadeToggle(3000);  
    });  
});
```

```
<body>  
<button>Click</button><br><br>  
  
<div id="div1" style="width:80px;height:80px;  
background-color:red;"></div>  
<br>  
<div id="div2" style="width:80px;height:80px;  
background-color:green;"></div>  
<br>  
<div id="div3" style="width:80px;height:80px;  
background-color:blue;"></div>  
  
</body>
```

fadeToggle()

Click



jQuery effects (metode pentru efecte speciale)

Exemplul 3 (slideUp.html)

```
$(document).ready(function(){  
    $("#par").click(function(){  
        $("li").slideUp("slow");  
    });  
});
```

Click pentru a vedea efectul slideUp()

1. Item 1
2. Item 2
3. Item 3
4. Item 4

```
<p id="par">Click pentru a vedea efectul slideUp()</p>  
<ol>  
<li>Item 1</li>  
<li>Item 2</li>  
<li>Item 3</li>  
<li>Item 4</li>  
</ol>
```

jQuery effects: animate()

creaza efect de animatie simultan pentru mai multe proprietati CSS care au valori numerice

```
$(#elanim).animate(  
{ left: 200px;  
  width: "+=100"; // proprietati  
  border: "hide"}, // CSS  
  
2000, // durata animatiei  
  
function (){ ... } // callback  
)
```

proprietati CSS care au
valori numerice:

border, margin, padding,
height,width,
font size, line height,
background position,

bottom, left, right, top
(pentru elemente cu position)

jQuery chaining

```
<body>
<button>Click</button><br><br>
<div id="box"></div>
</body>
```

```
#box {
    background: red;
    height: 100px;
    width: 100px;
    position: relative;
    top: 200px;
    left: 200px;
}
```

animate.html

```
$("#button").click(function(){

    $("#box").animate({opacity: "0.1", left: "+=400"}, 1200)
    .animate({opacity: "0.4", top: "+=160", height: "20", width: "20"}, "slow")
    .animate({opacity: "1", left: "0", height: "100", width: "100"}, "slow")
    .animate({top: "0"}, "fast")
    .slideUp()
    .slideDown("slow", function() {alert("Gata")});
    return false;
});
```

AJAX se ocupa cu schimbul de date cu un server și actualizarea unor parti dintr-o pagină web fără a reîncărca întreaga pagină.

jQuery oferă mai multe metode pentru funcționalitatea AJAX.

Cu ajutorul metodelor jQuery AJAX, se pot face cereri pentru date în format text, HTML, XML sau JSON de la un server folosind atât GET, cât și POST și se pot încărca datele externe direct în elementele HTML selectate

Ajax: metoda `load()`

încarcă datele de pe un server și pune datele returnate în elementul selectat.

```
$(selector).load(URL,data,callback)
```

URL specifică adresa URL care se va încarca (parametru obligatoriu)

date reprezintă query stringul care se va trimite împreună cu cererea (optional)

callback este numele unei funcții care urmează a fi executată după ce metoda de încărcare a fost terminată (optional)

parametrii

raspuns - conține răspunsul de la server.
status - conține status-ul (starea) cererii.
xhr - obiectul XMLHttpRequest.

Exemplu

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("toload.html #doi", function(raspuns, status, xhr){
            if(status == "success")
                alert(raspuns);
            if(status == "error") alert(xhr.status);
        });
    });
});
</script>
```

```
<body>
<div id="div1"><h2>Continutul div-ului</h2></div>
<button>Load</button>
</body>
```

Continutul div-ului

Load

continut nou 2

Load

```
<!doctype html>
<html>
<article id="unu">
continut nou 1
</article>
```

```
<article id="doi">
continut nou 2
</article>
```

```
<article id="trei">
continut nou 3
</article>
</html>
```

Ok

Ajax: metoda `get()`

solicita date de la un server cu o cerere HTTP GET

```
$.get(URL,callback)
```

URL specifică URL-ul unde se va trimite cererea (parametru obligatoriu)

callback este numele unei funcții care va fi executată dacă cererea este reusita (optional)

parametrii

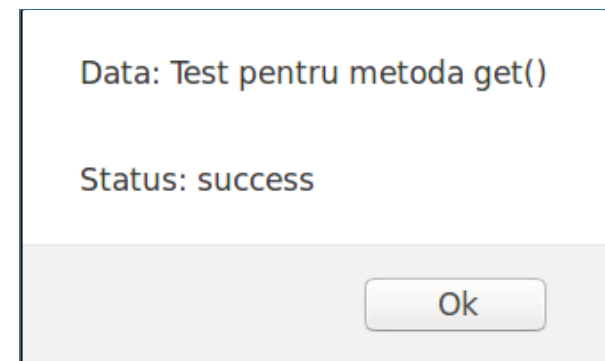
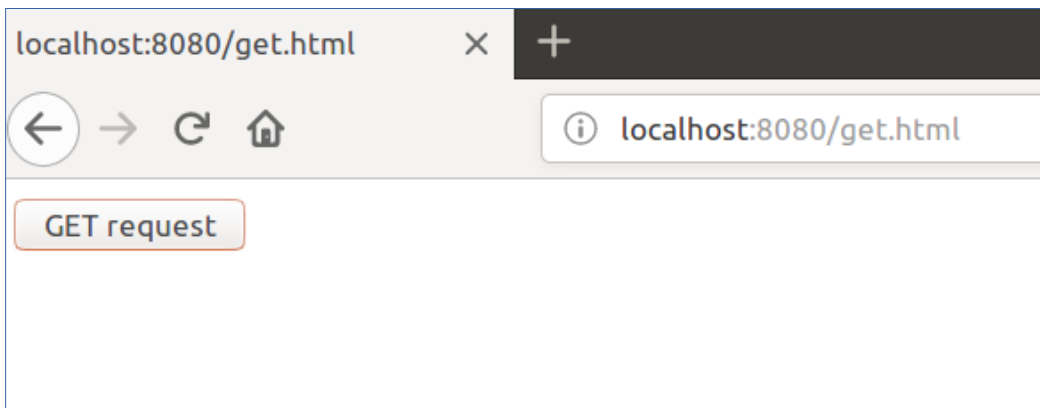
raspuns - contine raspunsul de la server.
status - contine status-ul (starea) cererii.
xhr - obiectul XMLHttpRequest.

Exemplu: metoda get()

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.get("http://localhost:8080/get.txt", function(data, status){
            alert("Data: " + data + "\nStatus: " + status);
        });
    });
});
</script>
```

get.txt

Test pentru metoda get()



Ajax: metoda `post()`

solicita date de la un server cu o cerere HTTP POST


```
$.post(URL,data,callback)
```

URL specifică adresa URL la care se va trimite cererea (parametru obligatoriu)

date reprezintă query stringul care se va trimite împreună cu cererea (optional)

callback este numele unei funcții care va fi executată dacă cererea este reusita (optional)

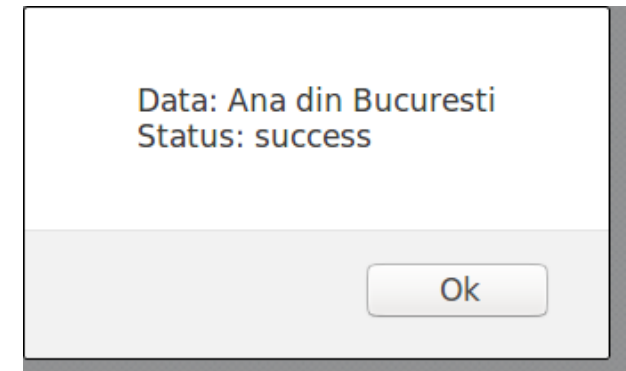
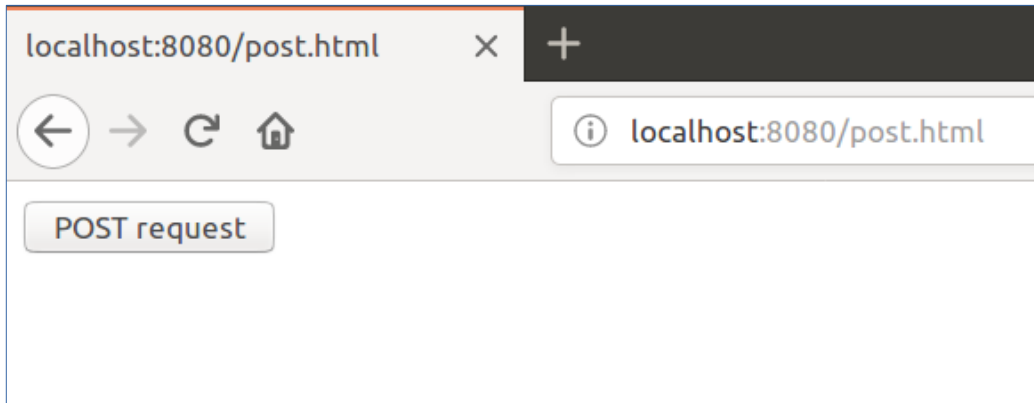
parametrii



raspuns - contine raspunsul de la server.
status - contine status-ul (starea) cererii.
xhr - obiectul XMLHttpRequest.

Exemplu: metoda post()

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.post("http://localhost:8080/pp", {nume: "Ana", city:"Bucuresti"},
            function(data, status){
                alert("Data: " + data + "\nStatus: " + status);
            });
    });
});
</script>
```



Ajax: metoda `ajax()`

Se foloseste pentru a efectua o cerere AJAX (asynchronous HTTP) la server.

Toate functiile Ajax jQuery: `load()`, `get()`, `post()` folosesc metoda `ajax()`. Aceasta contine mai multe optiuni de configurare si prelucrare a datelor, iar sintaxa ei este mai complexa.

```
$.ajax({  
  type: "GET"//"POST",  
  url:   url,  
  data:  {"nume":"valoare", "nume2":valoare2},  
                                     // string sau obiect  
  dataType: "text",  //"xml", "json"  
  success: function (raspuns, status){},  
  error:    function (raspuns, status){},  
})
```

```
<script>
var persoane;
$(document).ready(function(){
    $("button").click(function(){
        $.ajax({
            type: "GET",
            url: "date.json",
            dataType: "json",
            success: function (data){
                persoane = data;
                for (var i=0; i< persoane.length; i++)
                    $("#div1").append("<p>" + persoane[i].pers.numa + " are varsta " +
                        persoane[i].pers.varsta + "</p>");
            }
        });
    });
});
</script>
```

```
<body>
<div id="div1">
<h2>Aici se vor incarca date
noi</h2>
</div>
<button>Load</button>
</body>
```

date.json

```
[
  {
    "pers": {
      "numa": "Ion",
      "varsta": "42"
    }
  },
  {
    "pers": {
      "numa": "Maria",
      "varsta": "30"
    }
  }
]
```

Aici se vor incarca date noi

Ion are varsta 42

Maria are varsta 30

Load

<body>

<form id="testform" action="http://localhost:8080/action" method="POST">

<label>Nume:</label>

<input type="text" name="name">

<label> Varsta:</label>

<input type="text" name="age">

<label>Localitate:</label>

<select name="city">

<option value="Bucuresti" selected>Bucuresti</option>

<option value="Timisoara" selected>Timisoara</option>

</select>

<button type="submit" id="buton"> Trimite </button>

</form>

</body>

Aplicația server

```
const express = require('express')
const bodyParser = require('body-parser')
const multipart = require('connect-multiparty')
const app = express()
const multi = multipart()

app.use(bodyParser.urlencoded({extended: true}))
app.use(express.static('public'));

app.post('/action', multi, function(req, res) {res.send(req.body.name + ' din '
| + req.body.city + ' are ' + req.body.age + ' (de) ani');})

app.listen(8080, function() {console.log('listening')})
```

← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:

← → ↻ 🏠 localhost:8080/action

Andrei din Timisoara are 10 (de) ani

➤ Submiterea formelor folosind JQuery ajax()

```
$(document).ready(function(){  
  
$("#testform").submit(function (event){  
  
var dataform = $("#testform").serialize();  
  
$.ajax({  
    type: "POST",  
    url:"http://localhost:8080/action",  
    data: dataform,  
    success: function (data, textStatus){  
        $("#info").html(data);},  
    error: function (data, statusCode){  
        alert( statusCode);}  
    });  
    event.preventDefault();  
}) })
```



dataform= "name=Maria&age=20&city=Bucuresti"

← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:

← → ↻ 🏠 localhost:8080/formjquery.html

Nume:

Varsta:

Localitate:

Maria din Bucuresti are 20 (de) ani