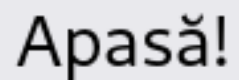


# TEHNICI WEB EVENIMENTE

*Claudia Chiriță . 2022/2023*

# ELEMENTUL HTML BUTTON



Apasă!

```
<button name="button"> Apasă! </button>
```

# ATRIBUTUL HTML ONCLICK

```
<tag onclick="cod JavaScript"></tag>
```

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
  </head>
  <body>
    <button onclick="alert('Hello, bunny!
Inline event handlers are bad practice. Do
not use them!')">
      Apasă!
    </button>
  </body>
</html>
```



# ATRIBUTUL HTML ONCLICK

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      function change(alttext){
        // event handler
        var x =
document.getElementById('bnny');
        x.src = "easter-bunny.jpg";
        x.alt = alttext;
        alert('stop using inline handlers!')
      }
    </script>
  </head>
  <body>
    
    <button id="button"
      onclick="change('easter')">
      Easterize bunny
```



# JAVASCRIPT: PROPRIETATEA ONCLICK

```
object.onclick = function() {cod JavaScript};  
object.onclick = nume-functie-js;
```

```
<!DOCTYPE html>  
<html>  
  <head>  
    <link rel="stylesheet" href="demo.css">  
    <script type="text/javascript">  
      window.onload = load;  
      function load() {  
        document.getElementById("button")  
          .onclick = function(){  
            change("easter");} }  
      function change(alttext){  
        var x =  
document.getElementById('bnny');  
        x.src = "easter-bunny2.jpg";  
        x.alt = alttext; }  
    </script>  
  </head>  
  <body>  
    
```



# EVENIMENTE

- evenimentele au loc în DOM; nu sunt elemente JavaScript
- browserul detectează un eveniment și îl anunță
- unui eveniment îi sunt asociate unele elemente JavaScript specifice: name, target, handler
- **tipuri de evenimente**: form events, window events, mouse events, key events etc.

# EVENIMENTE

- la apariția unui eveniment este creat un obiect de tip **Event** care poate fi referit prin identificatorul *event*

- referință HTML (*event*):

```
<tag onclick="f(event)"></tag>
```

- *e* - parametrul implicit al funcției apelate de eveniment

```
element.onclick = function(e) {}
```

# OBIECTE EVENT: PROPRIETĂȚI

```
event.type // numele evenimentului
```

```
event.target // ținta inițială a evenimentului
```

```
event.currentTarget /* ținta curentă a evenimentului  
                    (la capturare sau propagare) */
```

```
event.timeStamp /* numărul de milisecunde de la  
                încărcarea documentului până la  
                crearea evenimentului */
```



# OBIECTE EVENT: PROPRIETĂȚI

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        document.body.onclick = clicked;
        function clicked(event) {
          alert("event.type: " + event.type +
            '\n' +
              "event.target: " +
event.target.nodeName + '\n' +
              "event.currentTarget: " +
event.currentTarget.nodeName + '\n' +
              "event.timeStamp: " +
event.timeStamp); }
        }
      </script>
    </head>
    <body>
      <p>Easter Bunnies</p>
      
```



# MOUSE EVENTS

- evenimente determinate de mouse
- reprezentate prin obiecte de tipul MouseEvent

```
click, dblclick  
mouseover, mouseout  
mousedown, mouseup  
mouseleave
```

# MOUSE EVENTS

proprietăți speciale:

```
event.button // 0 (stânga) 1(mijloc) 2(dreapta)

event.clientX, event.clientY /* poziția în fereastră
                               (zona vizibilă) */
event.pageX, event.pageY // poziția în document
event.screenX, event.screenY // poziția în ecran

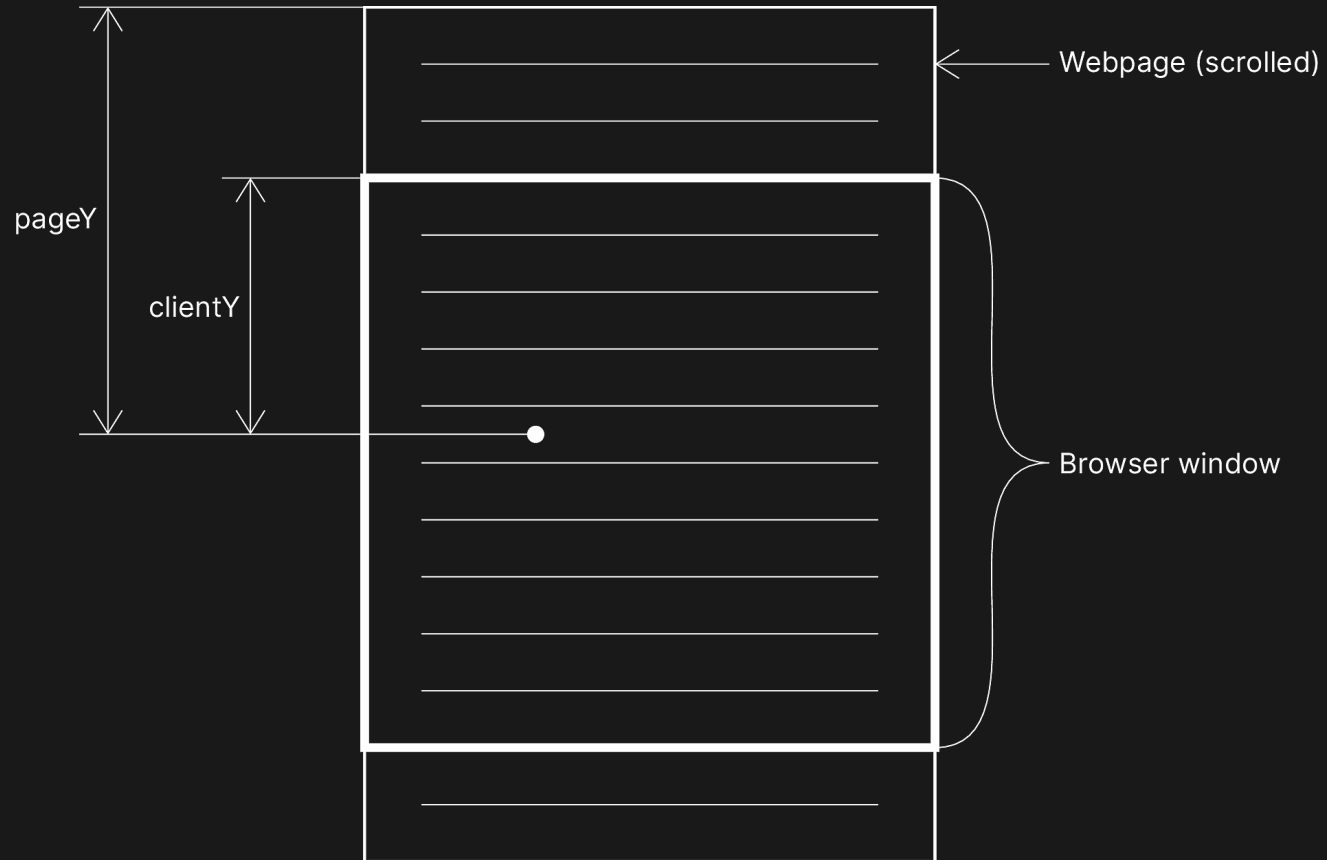
event.relatedTarget /* elementul legat de elementul care
                     a declanșat evenimentul */
```

# MOUSE EVENTS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        document.body.onclick = pozitie;
        function pozitie(event) {
          alert("event.clientX-Y " +
event.clientX + ' ' + event.clientY + '\n' +
          "event.pageX-Y: " + event.pageX
+ ' ' + event.pageY + '\n' +
          "event.screenX-Y " +
event.screenX + ' ' + event.screenY); }
        }
      </script>
    </head>
    <body>
      <p>Easter Bunnies</p>
      
    </body>
  </html>
```



# MOUSE EVENTS



# KEYBOARD EVENTS

- evenimente determinate de interacțiunea cu tastatura
- reprezentate prin obiecte de tipul `KeyboardEvent`

```
keydown, keyup  
keypress // deprecated!
```

- proprietăți:

```
event.key // tasta apasată  
event.altKey // boolean  
event.ctrlKey // boolean  
event.keyCode, event.which // deprecated!
```

# KEYBOARD EVENTS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        var par =
document.getElementById("par");
        document.body.onkeyup =
function(event) {
          switch (event.key) {
            case "p":
              par.style.color = "pink";
              break;
            case "y":
              par.style.color = "yellow";
              break;
            case "b":
              par.style.color = "lightBlue";
              break;
            default:
              alert("Altă tastă"); return; }

```



# EVENT LISTENERS

- metode care înregistrează funcții handler pentru evenimente
- un eveniment poate avea înregistrate mai multe funcții handler

```
// addEventListener(type, listener, useCapture)  
el.addEventListener("click", handleClick, false)
```

```
// removeEventListener(type, listener, useCapture)  
el.removeEventListener("click", handleClick, true)
```



# EVENT LISTENERS

```
<button id="buton">Apasă!</button>
```

```
var b = document.getElementById("buton");  
  
// b.onclick = function(){ alert("Hello, bunny!"); }  
b.addEventListener("click",  
    function(){alert("Hello, bunny!");},  
    false)
```

# HANDLERS

captarea evenimentelor: obiectul *event* este transmis ca prim argument al funcției handler

```
// el.onclick = myfct;  
el.addEventListener("click", myfct);  
  
function myfct (event) { alert(event.type); }
```

```
// el.onclick = function (event) { alert(event.type); }  
el.addEventListener("click", function (event) { alert(event.ty
```

# HANDLERS

la apariția unui eveniment, funcțiile handler sunt executate în ordinea în care sunt definite

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = main;
      function main(){
        el =
document.getElementById("button");
        el.addEventListener("click", hand1,
false);
        el.addEventListener("click", hand2,
false);
        function hand1(event) {alert(1)};
        function hand2(event) {alert(2)};
      }
    </script>
  </head>
  <body>
    <button id="button"> Apasă! </button>
  </body>
</html>
```



# EVENT BUBBLING & CAPTURE

ordinea de execuție a handlerelor în ceea ce privește nodurile imbricate poate fi controlată prin parametrul boolean *useCapture* al metodei *addEventListener* la setarea unui handler pentru un nod părinte

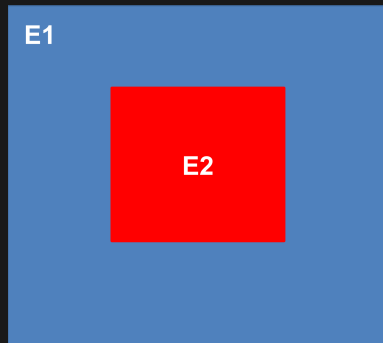
```
false // BUBBLING (implicit)
true  // CAPTURING
```

# EVENT BUBBLING & CAPTURE

modele de execuție:

- bubbling: se execută mai întâi handlerul pentru copil, apoi handlerul pentru părinte
- capturing: se execută mai întâi handlerul pentru părinte, apoi cel pentru copil

# EVENT BUBBLING & CAPTURE



E1, E2: handlers pentru onclick  
eveniment: click pe E2

modele de executie:

CAPTURING - handler E1, handler E2

BUBBLING - handler E2, handler E1

# EVENT BUBBLING & CAPTURE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <style>
      .alb{ background-color: white;
        color: black; }
      .roz{ background-color: pink; }
      .gal{ background-color: yellow; }
      div{ border: 3px solid pink;
        width: 300px; height:300px; }
      p { border: 3px solid yellow; }
    </style>
    <script type="text/javascript">
      window.onload = function() {
        var bunic = document.body;
        var parinte =
document.getElementById("parinte");
        var copil =
document.getElementById("copil");
        bunic.addEventListener("click",
function() {
```



# OBIECTE EVENT: METODE



```
event.stopPropagation()
```

oprește propagarea evenimentului în DOM



# STOPPROPAGATION()

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <style>
      .alb{ background-color: white;
        color: black; }
      .roz{ background-color: pink; }
      .gal{ background-color: yellow; }
      div{ border: 3px solid pink;
        width: 300px; height:300px; }
      p { border: 3px solid yellow; }
    </style>
    <script type="text/javascript">
      window.onload = function() {
        var bunic = document.body;
        var parinte =
document.getElementById("parinte");
        var copil =
document.getElementById("copil");
        bunic.addEventListener("click",
function() {
```



# OBIECTE EVENT: METODE

- `event.stopImmediatePropagation()`

dacă mai multe funcții listener sunt atașate  
aceluiași element, iar una conține  
`event.stopImmediatePropagation()`, funcțiile  
listener următoare nu mai sunt apelate

# EVENT BUBBLING & CAPTURE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <style>
      .alb{ background-color: white;
        color: black; }
      .roz{ background-color: pink; }
      .gal{ background-color: yellow; }
      div{ border: 3px solid pink;
        width: 300px; height:300px; }
      p { border: 3px solid yellow; }
    </style>
    <script type="text/javascript">
      window.onload = function() {
        var bunic = document.body;
        var parinte =
document.getElementById("parinte");
        var copil =
document.getElementById("copil");
        bunic.addEventListener("click",
function() {
```



# OBIECTE EVENT: METODE



`event.preventDefault()`

anulează acțiunea implicită a elementului

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        document.getElementById("link")
          .addEventListener("click",
function(event){
  event.preventDefault()});
      }
    </script>
  </head>
  <body>
    <a id="link"
href="https://gutenberg.org/cache/epub/14838/
pg14838-images.html">Peter Rabbit</a>
  </body>
</html>
```



# OBIECTUL WINDOW: TIMERS

- `window.setTimeout`

apelează o funcție sau execută un fragment de cod după un anumit timp (milisecunde)

```
vt = setTimeout(umeFuncție, intarziere, parametri);  
vt = setTimeout(function() {}, intarziere, parametri);  
  
clearTimeout(vt) // anulare funcție lansată
```

- vt este variabilă globală pentru a putea fi văzută de metoda clearTimeout
- parametrii sunt ai funcției care se va executa (umeFuncție sau anonimă)

# OBIECTUL WINDOW: TIMERS

- `window.setInterval`

execută funcția în mod repetat, la un anumit interval de timp

```
vt = setInterval(numeFuncție, interval, parametri);  
vt = setInterval(function() {}, interval, parametri);  
  
clearInterval(vt) // anulare funcție lansată
```

# OBJECTUL WINDOW: TIMERS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      /* var hello; var end = 15000;
      function sayHello() {
        alert("hello, bunny!"); }
      function sayHelloMany() {
        hello = setInterval(sayHello,
3000);}
      function stopHello(){
        clearInterval(hello); }
      sayHelloMany();
      setTimeout(stopHello, end); */
    </script>
  </head>
  <body>
  </body>
</html>
```



# PROBLEME CU VAR

```
var i = 0;
el.onclick = function() {
    alert(i); /* evaluare la click; va afisa 1 */
}
i = 1;
```

```
var i = 0;
{
    let il = i; // il e creat acum
    el.onclick = function(){
        alert(il); /* va afișa 0 */ }
} // se eliberează zona; il = 0
i = 1;
```



# PROBLEME CU VAR

la click pe o imagine din document să i se afișeze sursa

```
var imagini = document.getElementsByTagName("img");
for(var i = 0; i < imagini.length; i++)
    imagini[i].onclick = function() { alert(imagini[i].src); }
// nu funcționează
// i va fi egal cu imagini.length
```

```
// soluție
for(let i = 0; i < imagini.length; i++)
    imagini[i].onclick = function() { alert(imagini[i].src); }
```

# OBIECTUL WINDOW: TIMERS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        var pl =
document.getElementsByTagName("p");
        for(var i = 0; i < pl.length; i++)
          setTimeout(colorare, 1000*(i+1),
"red", pl[i]);
        function colorare(culoare,ob) {
          ob.style.color=culoare; }
      }
    </script>
  </head>
  <body>
    <p>Ou 1</p>
    <p>Ou 2</p>
    <p>Ou 3</p>
```



# OBIECTUL WINDOW: TIMERS

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <script type="text/javascript">
      window.onload = function() {
        var pl =
document.getElementsByTagName("p");
        for(let i = 0; i < pl.length; i++)
          setTimeout(function(){
            colorare("red",
pl[i]);},
                    1000*(i+1));
        function colorare(culoare,ob) {
          ob.style.color=culoare; }
      }
    </script>
  </head>
  <body>
```



# OBIECTUL WINDOW: GETCOMPUTEDSTYLE()

```
window.getComputedStyle(ob, pseudoel)
```

- determină stilul efectiv aplicat unui element
- întoarce un obiect read-only CSSStyleDeclaration
- *pseudoel* este opțional; pseudo-element sau null

```
var oStil = window.getComputedStyle(ob, ":after");  
var x = oStil.color; // proprietatea css ob.style.color  
  
window.getComputedStyle(ob, null).getPropertyValue("color");
```

# OBIECTUL WINDOW: COMPUTEDSTYLE

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="demo.css">
    <style>
      div{ border-radius: 50%;
          background-color:red;
          width:100px; height:150px;}
    </style>
    <script type="text/javascript">
      window.onload = function() {
        var p =
document.getElementById("stil");
        var div =
document.getElementById("ou");
        var stil =
window.getComputedStyle(div);
        p.innerHTML+= '<br>' +
          "background-color: "+
stil.getPropertyValue("background-color") ;
```



# OBIECTUL WINDOW

- fereastra browserului care conține DOM asociat documentului
- poate fi selectat prin *document.defaultView*
- dacă un document HTML conține elemente `<iframe>`, browserul creează un obiect window pentru documentul HTML și un obiect window suplimentar pentru fiecare `<iframe>`

# MULTIPLE WINDOWS

- ferestrele iframe au proprietățile

```
contentWindow // obiectul window corespunzător iframe  
parent // fereastra părinte
```

```
// colorează backgroundul primului iframe roz  
  
var x = document.getElementsByName("iframe")[0].contentWindow  
//x = obiectul window corespunzător primului iframe  
  
x.document.getElementsByName("body")[0].style.backgroundColor
```

# OBIECTUL WINDOW: LOCATION

```
window.location
```

folosit pentru a determina URL-ul paginii curente și pentru a redirecționa browserul către o pagină nouă

```
window.location.href // adresa URL a documentului  
window.location.hostname // numele de domeniu al gazdei web  
window.location.pathname /* calea și numele fișierului  
                             paginii curente */  
window.location.protocol /* protocolul folosit  
                             http: sau https: */  
window.location.hash // partea din URL care începe cu #  
window.location.assign(url) // încarcă un nou document
```



# OBIECTUL WINDOW: LOCATION

schimbarea locației

```
window.location.href="https://tinyurl.com/easter-bunny-history"
```

```
window.location="https://tinyurl.com/easter-bunny-history"
```

```
location="https://tinyurl.com/easter-bunny-history"
```

# OBIECTUL WINDOW: HISTORY

```
window.history
```

conține istoria browserului (adresele URL accesate de utilizator)

```
history.length // nr. de URL-uri din lista de istoric
history.back() /* încarcă adresa URL anterioară
               din lista de istoric */
history.forward() /* încarcă următoarea adresă URL
                  din lista de istoric */
history.go(number/url) /* încarcă o adresă URL specificată
                       din lista de istoric */
```

# OBIECTUL WINDOW: SCREEN

```
window.screen
```

conține informații despre ecranul utilizatorului

```
screen.availHeight // înălțimea ecranului (excluzând taskbarul)
screen.availWidth  // lățimea ecranului (excluzând taskbarul)
screen.height      // înălțimea totală a ecranului
screen.width       // lățimea totală a ecranului
screen.colorDepth
screen.pixelDepth /* nr. de biți folosiți pentru
                  culoarea unui pixel */
```

# OBIECTUL WINDOW: OPEN

```
window.open(URL, target, caracteristici)
```

- deschide o nouă fereastră de browser
- toți parametrii sunt opționali
- dacă nu se specifică URL se va deschide o fereastră cu about-blank

```
window.open("https://tinyurl.com/water-ship-down", "_blank",  
"toolbar=yes,scrollbars=yes,resizable=yes,top=500,left=500,  
width=400,height=400");
```

# OBIECTUL WINDOW: OPEN

```
var myWindow=window.open();  
myWindow.close();
```

Închide fereastra deschisă cu metoda open()

```
<!DOCTYPE html>  
<html>  
  <head>  
    <link rel="stylesheet" href="demo.css">  
    <script type="text/javascript">  
      var myWindow; // globală pt. a fi  
      vazută la apelul close()  
      function openWin() {  
        myWindow =  
window.open("https://tinyurl.com/watership-  
down-film"); }  
      function closeWin() {  
        myWindow.close();}  
    </script>  
  </head>  
  <body>  
    <button onclick="openWin()">Open
```



```

, @; @,
, @; @; @; @; @; @/ ) @; @;
, ; @; @; @; @; @; @ | _ / @ ' e \
( | @; @: @ \ @; @; @; @: @ (      \
' @; @; @ | @; @; @; @; ' ` " -- '
' @; @; / ; @; / ; @; '
) //      | ||
\ \ \      | ||
\ \ \      ) \ \
  ` " `      ` " ` `

```

întrebări?



