PROGRAMMAZIONE WEB

JAVASCRIPT - APPROFONDIMENTI

SINTASSI DI BASE, ARRAY, STRINGHE E ALTRI OGGETTI

Operatori aritmetici

Somma (+) somma due numeri oppure concatena due stringhe (o concatena numeri a stringhe)

Sottrazione (-) sottrae il secondo numero dal primo

Prodotto (*) moltiplica tra loro due numeri

Divisione (/) divide il primo numero per il secondo

Modulo (%) restituisce il resto della divisione intera del primo operando per il secondo

```
5 % 2 = 1
```

Incremento/Decremento (++/--) l'operatore ++ (--) aumenta (diminuisce) di una unità il valore di una variabile (attenzione all'uso post-fisso e pre-fisso)

```
i = i + 1;
```
è analogo a `i++;`

```
i = i - 1;
```
è analogo a `i--;`

Assegnamento (= += -= *= /= %=) esegue l'operazione ed effettua l'assegnamento del valore dell'espressione che sta a destra dell'operatore alla variabile che sta a sinistra

```
i += 10;
```
è analogo a `i = i + 10;`

Operatori di confronto

Sono operatori che verificano una relazione tra due operandi e restituiscono un valore booleano (true o false) a seconda che la relazione sia o meno verificata

Sono:

Uguaglianza (==)

Uguaglianza esatta (===), cioè con stesso valore e stesso tipo

Disuguaglianza (!=)

Minore di (<)

Maggiore di (>)

Minore o uguale a (<=)

Maggiore o uguale a (>=)

```
<script type="text/javascript">
var firstname;
firstname="5";
document.write(firstname==5);
document.write("<br>");
document.write(firstname===5);
</script>
```

```
true
false
```

Operatori logici

AND (&&) restituisce un valore `true` se e solo se il primo operando e il secondo sono entrambi veri. Se uno o entrambi gli operandi sono falsi restituisce `false`

OR (||) restituisce un valore `true` se il primo operando o il secondo o entrambi sono veri. Se entrambi gli operandi sono falsi restituisce `false`

NOT (!) è un operatore unario, ossia si applica ad un solo operando. Restituisce il valore `false` se l'operando è vero, viceversa restituisce `true` se l'operando è falso

Operatore condizionale

In Javascript esiste un operatore condizionale che assegna ad una variabile un valore a seconda di una particolare condizione

`nome_variabile=(condizione)?valore1:valore2`

```
<script type="text/javascript">
var destinatario = "Prof";
var mail = (destinatario=="Prof")?"Gentile professore":"Senti pirla..."
document.write(mail);
</script>
```

Operatori - Esempi

`var s = "tre " + 2;` la stringa `s` vale "tre 2"

`s += " uno";` la stringa `s` vale "tre 2 uno"

`s > "ciao";` l'espressione vale `true`, in quanto il valore di `s` è lessicograficamente successivo a "ciao"

`typeof(s);` restituisce "string"

`eval("3+1");` restituisce 4

`eval("f(x)");` esegue lo script, chiamando `f(x)` e restituendo il valore di ritorno della chiamata

`eval("var s=1");` dichiara la variabile s e le assegna il valore 1

`void(f(x));` esegue `f(x)` ed ignora il suo valore di ritorno

Il costrutto if …else


```
if (espressione)
      istruzione1;
else
      istruzione2;
```

- ✓ il costrutto **`if`** permette di decidere quale istruzione eseguire a fronte del valore dell'espressione racchiusa tra parentesi
- ✓ se il valore di **`espressione`** è vero allora si esegue **`istruzione1`**, altrimenti si esegue **`istruzione2`**
- ✓ è possibile avere un **`if`** senza la parte **`else`**

Il costrutto if … else - Esempio


```
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
document.write("<b>Good morning</b>");
}
else if (time>=10 && time<16)
{
document.write("<b>Good day</b>");
}
else
{
document.write("<b>Hello World!</b>");
}
</script>

<p>
This example demonstrates the if..else if...else statement.
</p>
```

Il costrutto if … else - Esempio


```
<
v
v
i
{
o
}
e
{
o
}
e
{
o
}
<
<
T
<
```

Good day

This example demonstrates the if..else if...else statement.

Il costrutto switch

Javascript dispone del costrutto switch con la stessa sintassi di Java:

```
switch (espressione) {
   case v1: istruzioni; break;
   case v2: istruzioni; break;
   default: istruzioni
}
```

L'espressione viene valutata e confrontata con i valori dei diversi case:

- ✓ vengono quindi eseguite le istruzioni a partire dal primo case con lo stesso valore dell'espressione
- ✓ se nessun case è selezionato, vengono eseguite le istruzioni del default, se presenti
- ✓ se si desidera limitare l'esecuzione a un gruppo di istruzioni, è necessario introdurre la parola chiave break

Il costrutto switch - Esempio


```
<script type="text/javascript">
var d=new Date();
var theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("<b>Finally Friday</b>");
  break;
case 6:
  document.write("<b>Super Saturday</b>");
  break;
case 0:
  document.write("<b>Sleepy Sunday</b>");
  break;
default:
  document.write("<b>I'm really looking forward to this weekend!</b>");
}
</script>

<p>This JavaScript will generate a different greeting based on what day it
is. Note that Sunday=0, Monday=1, Tuesday=2, etc.</p>
```

Il costrutto switch - Esempio

<script type="text/javascript">

Sleepy Sunday

This JavaScript will generate a different greeting based on what day it is. Note that Sunday=0, Monday=1, Tuesday=2, etc.

Il ciclo for


```
for (inizializza; test; incremento)
    istruzione;
```

Il ciclo **`for`** permette di ripetere **`istruzione`** in modo ciclico fino a quando **`test`** risulta essere vero

```
for (var i = 0; i < 10; i++)
    istruzione;
```

- ✓ Inizializza la variabile **`i`** con il valore zero, esegue **`istruzione`** e quindi incrementa il valore di **`i`**. A questo punto esegue il test **`i<0`** e se questo è vero inizia da capo eseguendo di nuovo **`istruzione`**
- ✓ Per tutti i costrutti ciclici vale l'uso classico di **`break`** e **`continue`**

Il ciclo for - Esempio


```
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>

<p>Explanation:</p>

<p>This for loop starts with i=0.</p>

<p>As long as <b>i</b> is less than, or equal to 5, the loop will continue
to run.</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p>

</body>
</html>
```

Il ciclo for - Esempio


```
<script type="text/javascript">
f
{
c
c
}
<

<

<

t

<


<
<
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

Explanation:

This for loop starts with i=0.

As long as **i** is less than, or equal to 5, the loop will continue to run.

i will increase by 1 each time the loop runs.

Il ciclo while


```
while (espressione)
    istruzione;
```

Il ciclo **while** permette di ripetere **istruzione** in modo ciclico fino a quando **espressione** risulta essere vero

- ✓ se espressione non è mai vera è possibile che **istruzione** non venga mai eseguita
- ✓ per fare in modo che **istruzione** non venga eseguita all'infinito è necessario che **espressione** prima o poi diventi **false**

Il ciclo while - Esempio


```
<script type="text/javascript">
i=0;
while (i<=5)
{
document.write("The number is " + i);
document.write("<br />");
i++;
}
</script>

<p>Explanation:</p>
<p><b>i</b> is equal to 0.</p>
<p>While <b>i</b> is less than , or equal to, 5, the loop will continue to
run.</p>
<p><b>i</b> will increase by 1 each time the loop runs.</p>

</body>
</html>
```

Il ciclo while - Esempio


```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

Explanation:

i is equal to 0.

While i is less than , or equal to, 5, the loop will continue to run.

i will increase by 1 each time the loop runs.
```

Il ciclo do-while


```
do
      istruzione
while (espressione);
```

Il ciclo **do-while** permette di ripetere **istruzione** in modo ciclico fino a quando **espressione** risulta essere vero

- ✓ **istruzione** viene eseguita almeno una volta
- ✓ per fare in modo che **istruzione** non venga eseguita all'infinito è necessario che **espressione** prima o poi diventi **false**

Il ciclo do-while - Esempio


```
<body>

<script type="text/javascript">
i = 0;
do
{
document.write("The number is " + i);
document.write("<br />");
i++;
}
while (i <= 5)
</script>

<p>Explanation:</p>

<p><b>i</b>  equal to 0.</p>

<p>The loop will run</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p>
```

Il ciclo do-while - Esempio


```
<body>

<s
i
do
{
do
do
i+
}
wh
</
<p
<p
<p>The loop will run</p>

<p><b>i</b> will increase by 1 each time the loop runs.</p>
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

Explanation:

i equal to 0.

The loop will run

i will increase by 1 each time the loop runs.

Oggetti predefiniti – window (I)

Quando Javascript è usato all'interno del browser, sono disponibili alcuni oggetti particolari, relativi al browser stesso e alla pagina visualizzata

L'oggetto **`window`** è il punto di accesso a tutti gli oggetti esposti dal browser

Si tratta dell'oggetto predefinito per lo scripting, il che significa che tutte le sue proprietà e i suoi metodi sono accessibili a livello globale, senza bisogno di specificare esplicitamente l'oggetto **`window`**

Oggetti predefiniti – window (II)

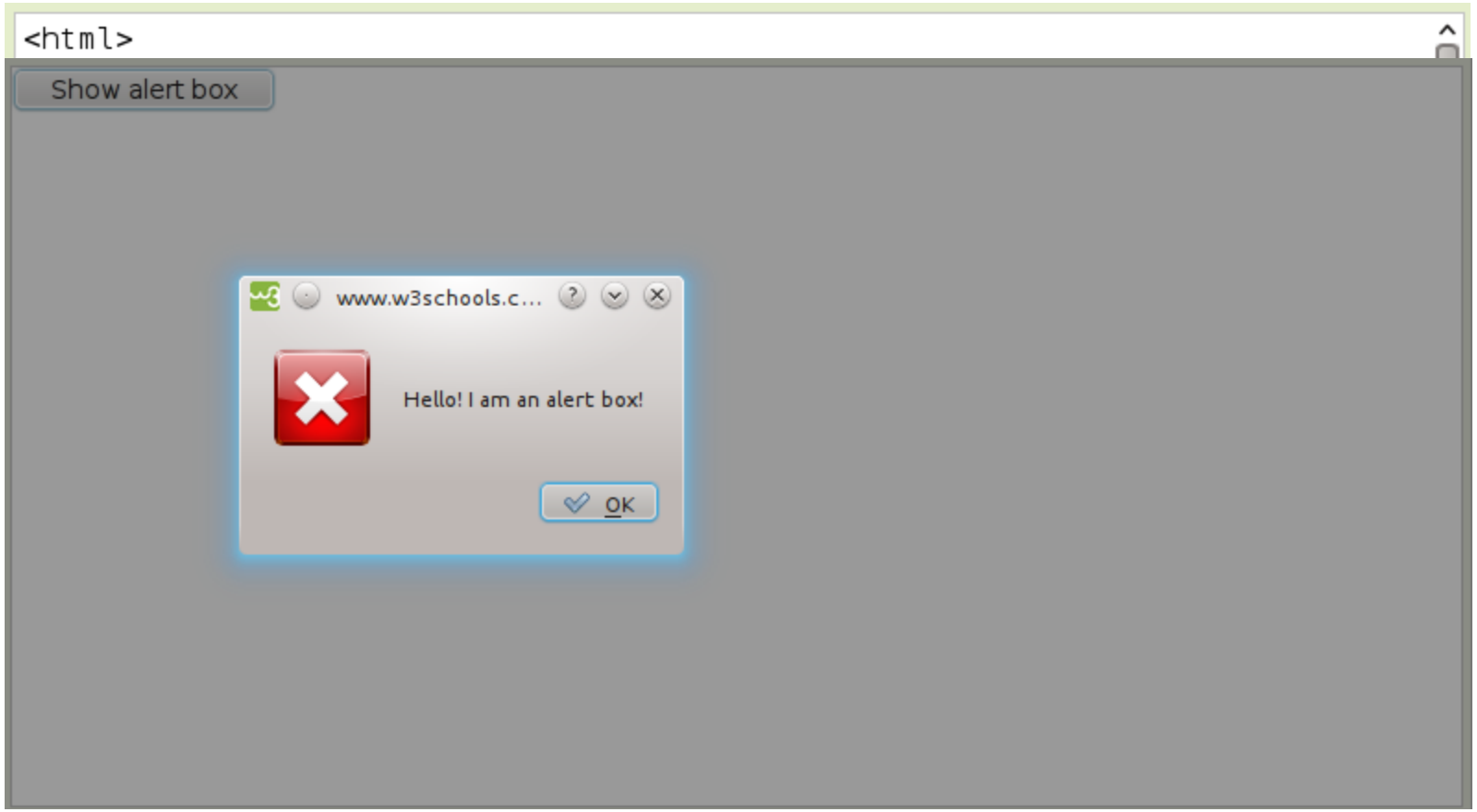
L'interfaccia di **`window`** contiene alcune funzionalità molto utili, tra cui

- ✓ **`alert(messaggio)`** - mostra il messaggio dato in un dialog box (con il solo bottone OK)
- ✓ **`confirm(messaggio)`** - mostra il messaggio dato in un dialog box con i bottoni OK e Cancel; la funzione ritorna true se l'utente preme OK, false altrimenti
- ✓ **`prompt(messaggio,default)`** - mostra il messaggio dato in un dialog box, insieme a un campo di input con valore iniziale default; se l'utente preme OK, il contenuto del campo (anche vuoto) di input viene restituito dalla funzione, altrimenti la funzione restituisce null

Alert - Esempio


```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("Hello! I am an alert box!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

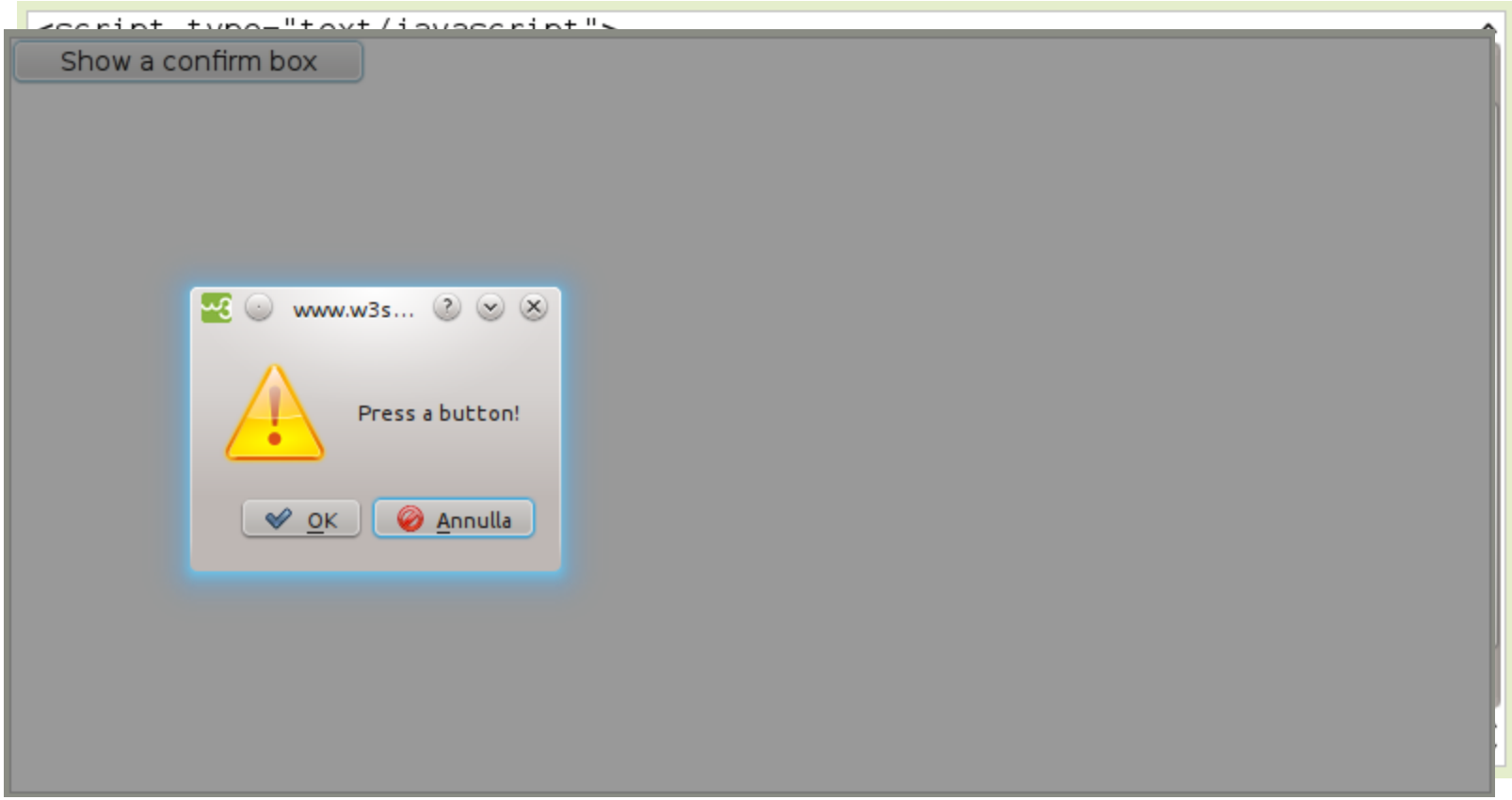
Alert - Esempio



Confirm - Esempio


```
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button!");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show a confirm box" />

</body>
</html>
```

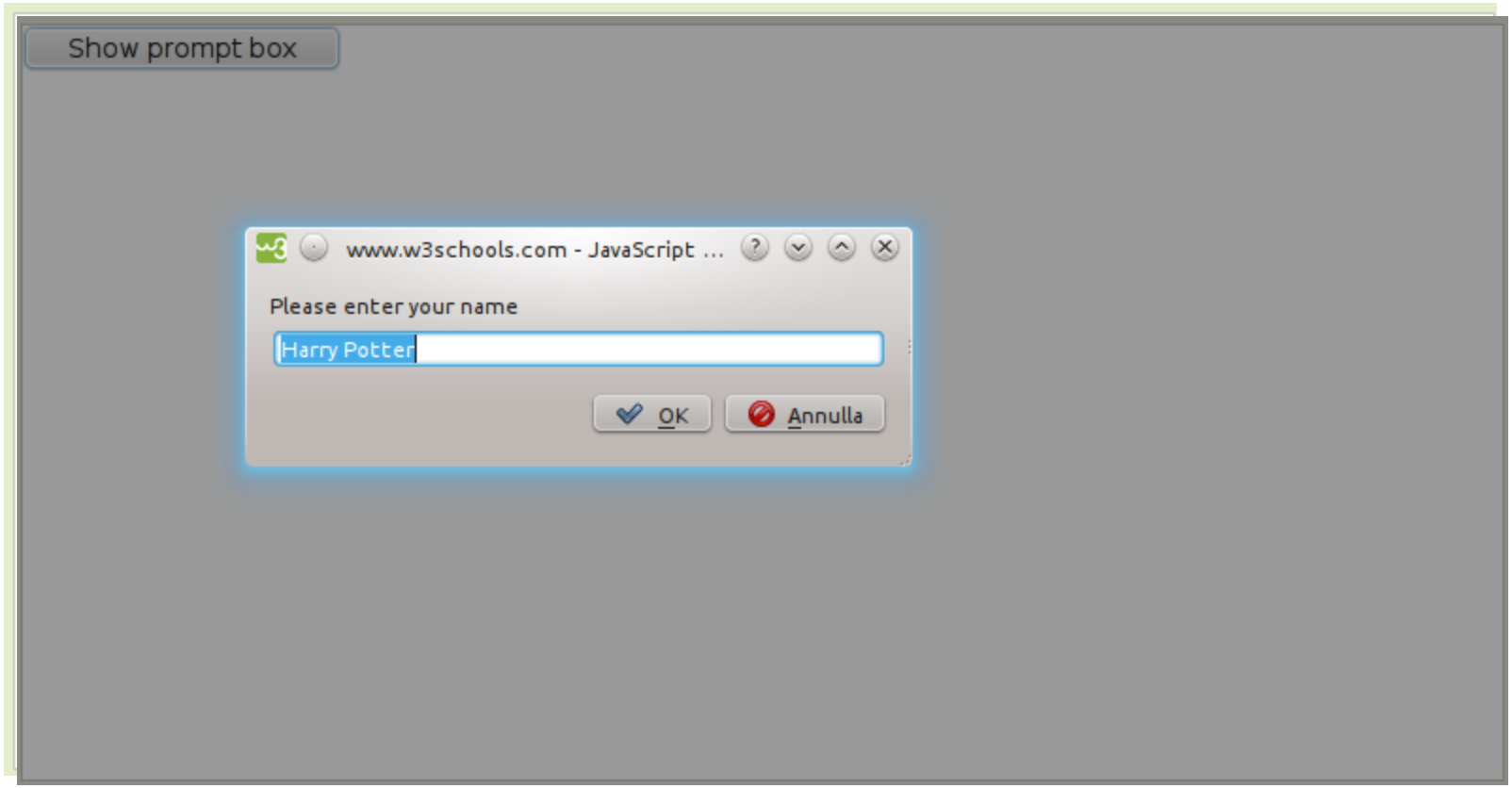
Confirm - Esempio



Prompt - Esempio


```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

Prompt - Esempio



Oggetti predefiniti – String (I)

Gli oggetti **`String`** sono usati in Javascript per contenere stringhe di caratteri

Possono essere creati implicitamente, utilizzando una costante stringa, o esplicitamente tramite il costruttore:

```
s = new String(valore)
```

I principali metodi e proprietà della classe **`String`** sono i seguenti:

`length` - restituisce la lunghezza della stringa

`charAt(posizione)` - restituisce il carattere (stringa di lunghezza uno) alla posizione data (base zero)

`charCodeAt(posizione)` - come **`charAt`**, ma restituisce il codice ASCII del carattere

`indexOf(s,offset)` - restituisce la posizione della prima occorrenza (a partire da offset, se specificato) di **`s`** nella stringa; restituisce -1 se **`s`** non è una sottostringa della stringa data (a partire dall'offset)

Oggetti predefiniti – String (I)

`lastIndexOf(s,offset)` - come `indexOf`, ma restituisce la posizione dell'ultima occorrenza

`substr(os[,l])` - restituisce la sottostringa di lunghezza `l` (default, la massima possibile) che inizia `os` caratteri dall'inizio della stringa

`substring(os,oe)` - restituisce la sottostringa che inizia `os` caratteri e termina a `oe` caratteri dall'inizio della stringa

`toLowerCase()` - ritorna la stringa convertita in minuscolo

`toUpperCase()` - ritorna la stringa convertita in maiuscolo

Stringhe – Esempio (I)


```
<html>
<body>

<script type="text/javascript">

var txt = "Hello World!";
document.write(txt.length);

</script>

</body>
</html>
```

Stringhe – Esempio (II)


```
var txt = "Hello World!";

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>");
```

Stringhe – Esempio (II)

Big: **Hello World!**

Small: Hello World!

Bold: **Hello World!**

Italic: *Hello World!*

Fixed: `Hello World!`

Strike: ~~Hello World!~~

Fontcolor: Hello World!

Fontsize: Hello World!

Subscript: Hello World!

Superscript: Hello World!

Link: Hello World!

Stringhe – Esempio (III)


```
<html>
<body>

<script type="text/javascript">
var str="Hello world!";
document.write(str.match("world") + "<br />");
document.write(str.match("World") + "<br />");
document.write(str.match("worlld") + "<br />");
document.write(str.match("world!"));
</script>

</body>
</html>
```

Stringhe – Esempio (III)


```
world
null
null
world!
```

Stringhe – Esempio (IV)


```
<html>
<body>

<script type="text/javascript">

var str="Visit Microsoft!";
document.write(str.replace("Microsoft","W3Schools"));

</script>
</body>
</html>
```

Stringhe – Esempio (IV)

Visit W3Schools!

Array

Gli *Array* o *Vettori* sono liste numerate di oggetti

```
var nomeArray = new Array(num);
```

dove

`nomeArray` è il nome dell'array che si vuole dichiarare

`num` è il numero di elementi contenuti nell'array ed è opzionale, se non è presente il numero di elementi non è definito a priori

Per accedere agli elementi di un array si utilizza l'indice **`i`** che indica la posizione dell'elemento all'interno del array stesso

```
nomeArray[i]
```

Array – Metodi principali

`length` - restituisce la dimensione dell'array

`concat(e1,e2,e3,…)` - aggiunge gli elementi dati alla fine dell'array

`join(separatore)` - converte l'array in una stringa, concatenando la versione `String` di ciascun elemento ed usando il separatore dato (default ",")

`reverse()` - inverte l'ordine dell'array

`slice(os[,l])` - restituisce il sotto-array di lunghezza `l` (default, la massima possibile) che inizia all'indice `os`

`sort([sortfun])` - ordina l'array; la funzione opzionale `sortfun` può essere usata per specificare un criterio di ordinamento non standard

Esempi di array (I)


```
var tigers = new Array(3);
```
crea un array di dimensione 3 con nome **tigers**

```
var animals = new Array();
```
crea un array di dimensione non definita con nome **animals**

```
Animals[0] = "duck";
Animals[1] = "elephant";
Animals[2] = "dog";
```
al primo elemento dell'array viene assegnato il valore "duck", al secondo "elephant" ed al terzo "dog"

```
var animals = new Array("duck","elephant", "dog");
```
crea l'array **animals** con i primi tre elementi definiti

Esempi di array (II)


```
<html>
<body>

<script type="text/javascript">

var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);

</script>

</body>
</html>
```

Esempi di array (II)


```
Jani,Tove,Cecilie,Lone
```

Esempi di array (III)


```
<html>
<body>

<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));

</script>

</body>
</html>
```

Esempi di array (III)


```
Banana,Orange,Apple,Mango
Banana+Orange+Apple+Mango
Banana and Orange and Apple and Mango
```

Esempi di array (IV)


```
<html>
<body>

<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());

</script>

</body>
</html>
```

Esempi di array (IV)


```
Apple,Banana,Mango,Orange
```

Esempi di array (V)


```
<html>
<body>

<script type="text/javascript">

function sortNumber(a, b)
{
return a - b;
}

var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));

</script>

</body>
</html>
```

Esempi di array (V)


```
1,5,10,25,40,100
```

Oggetti predefiniti – Date (I)

- ✓ L'oggetto **`Date`** permette di manipolare valori di tipo data e ora
- ✓ Dispone di diversi costruttori:

`Date()` - inizializza l'oggetto alla data/ora corrente

`Date(y,m,d,hh,mm,ss)` - inizializza l'oggetto alla data/ora dd/mm/yy hh:mm:ss

`Date(stringa)` - tenta di riconoscere la stringa come una data e inizializza l'oggetto di conseguenza

Oggetti predefiniti – Date (II)

- ✓ Gli oggetti `Date` possono essere confrontati tra loro con i normali operatori di confronto
- ✓ I metodi degli oggetti Date permettono di leggerne e scriverne tutti i membri

Ad esempio

`getYear`

`getMonth`

`setYear`

`setMonth`

`getDay`

`getDate`

`setDate`

Date – Esempio (I)


```
<html>
<body>

<script type="text/javascript">

var d=new Date();
document.write(d.getFullYear());

</script>

</body>
</html>
```

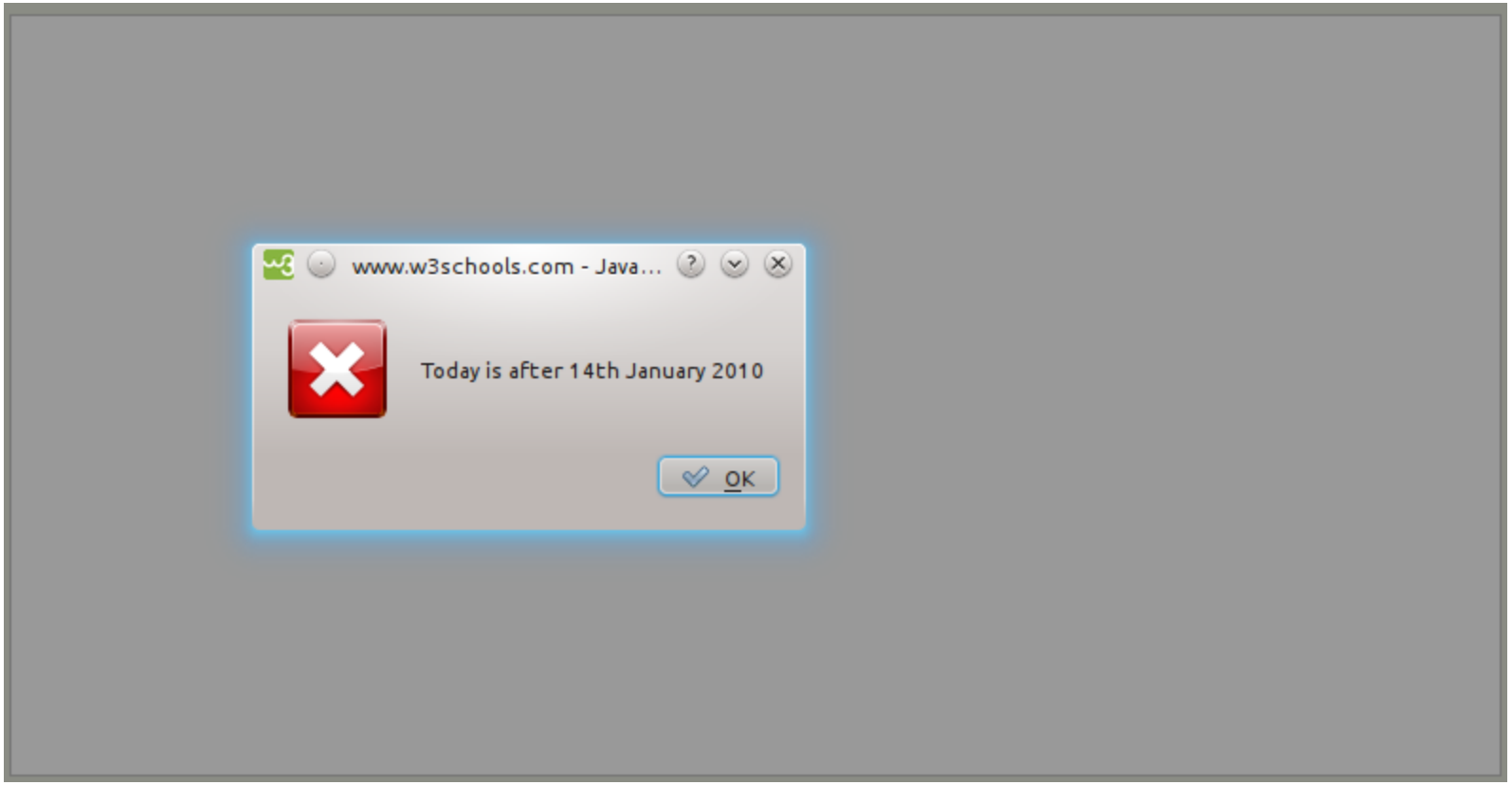
Date – Esempio (I)


```
2011
```

Date – Esempio (II)


```
<html>
<body>

<script type="text/javascript">

var myDate=new Date();
 myDate.setFullYear(2010,0,14);
var today = new Date();

 if (myDate>today)
   {
   alert("Today is before 14th January 2010");
   }
 else
   {
   alert("Today is after 14th January 2010");
   }

</script>

</body>
```

Date – Esempio (II)



Oggetto navigator

Questo oggetto contiene tutte le informazioni sul browser del client

```
<html>
<body>
<div id="example"></div>

<script type="text/javascript">

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";

document.getElementById("example").innerHTML=txt;

</script>

</body>
</html>
```

Oggetto navigator

Questo oggetto contiene tutte le informazioni sul browser del client

Browser CodeName: Mozilla

Browser Name: Netscape

Browser Version: 5.0 (compatible; Konqueror/4.6; Linux) KHTML/4.6.0 (like Gecko) Kubuntu

Cookies Enabled: true

Platform: Linux x86_64

User-agent header: Mozilla/5.0 (compatible; Konqueror/4.6; Linux) KHTML/4.6.0 (like Gecko) Kubuntu

Funzioni utili – Timer (I)

Javascript, tramite l'oggetto **`window`**, permette di eseguire azioni temporizzate; a questo scopo si usano i seguenti metodi

`setTimeout(stringa_o_funzione,millisecondi,arg1,…,argN)` - dopo il numero specificato di millisecondi, Javascript esegue il codice dato dal primo argomento, che può essere una stringa contenente del codice da valutare o il nome di una funzione da chiamare; in quest'ultimo caso, è possibile specificare opzionalmente una serie di argomenti (arg1…argN) da passare alla funzione; l'azione viene quindi eseguita una sola volta

Funzioni utili – Timer (II)

`setInterval(stringa_o_funzione,millisecondi,arg1,…,argN)` - ogni `millisecondi`, Javascript esegue il codice dato dal primo argomento, che può essere una stringa contenente del codice da valutare o il nome di una funzione da chiamare; in quest'ultimo caso, è possibile specificare opzionalmente una serie di argomenti (arg1…argN) da passare alla funzione; l'azione viene quindi eseguita periodicamente

Entrambe le funzioni possono essere chiamate più volte, e restituiscono un timer id (numerico), tramite il quale è possibile annullare la temporizzazione usando le rispettive funzioni:

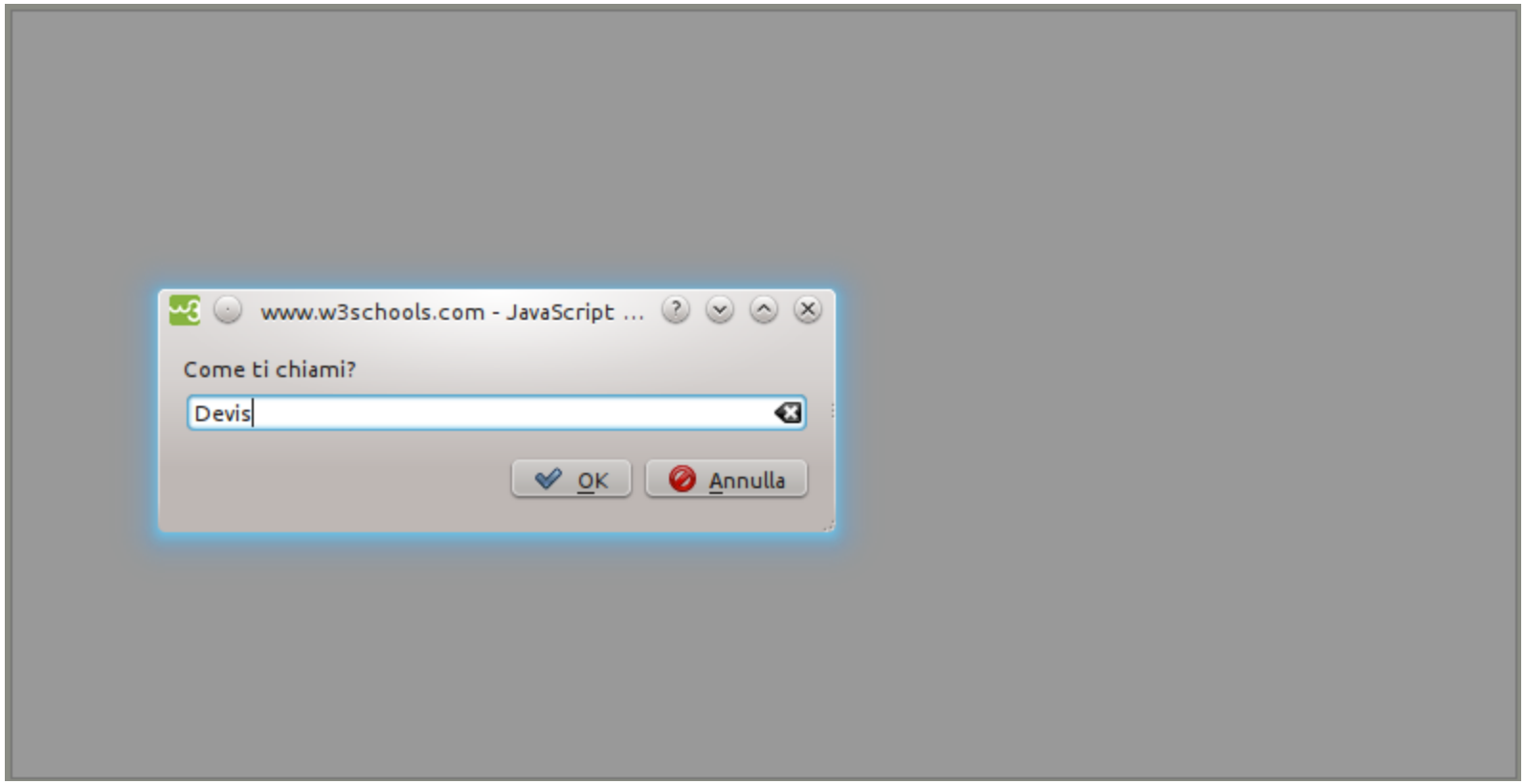
`clearTimeout(id)` per le temporizzazioni avviate con `setTimeout()`

`clearInterval(id)` per le temporizzazioni avviate con `setInterval()`

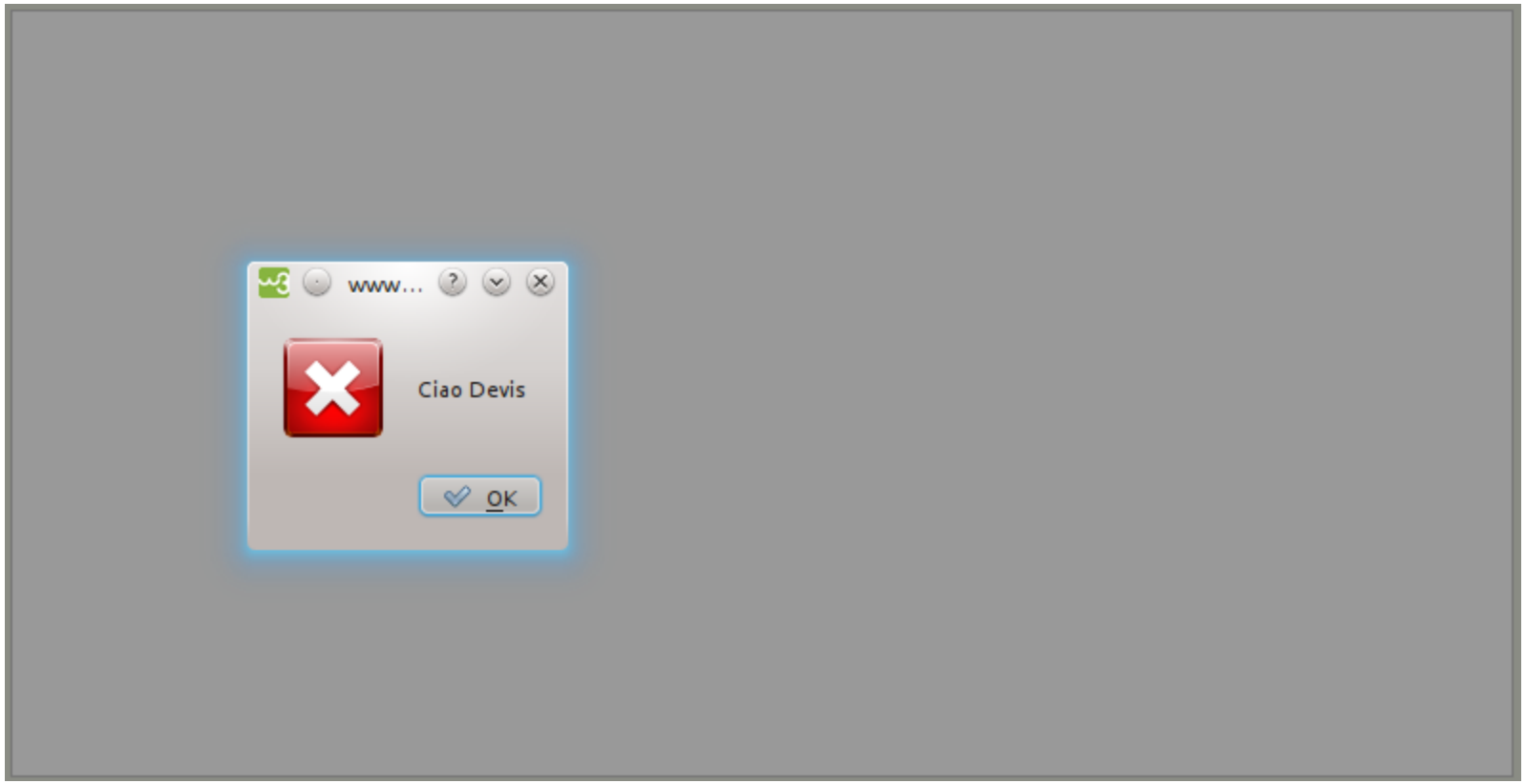
Timer – Esempio (I)


```
<html>
<head>
<script type="text/javascript">
function saluta(nome) {
alert("Ciao "+nome);
}

//Richiede il nome e saluta dopo cinque secondi
var nome = prompt("Come ti chiami?");
if (nome) setTimeout(saluta,5000,nome);
//avverte dell'ora corrente ogni minuto
setInterval("d=new Date(); alert('Ora sono le
'+d.getHours()+':'+d.getMinutes())",25000);
</script>
</head>

<body>
</body>
</html>
```

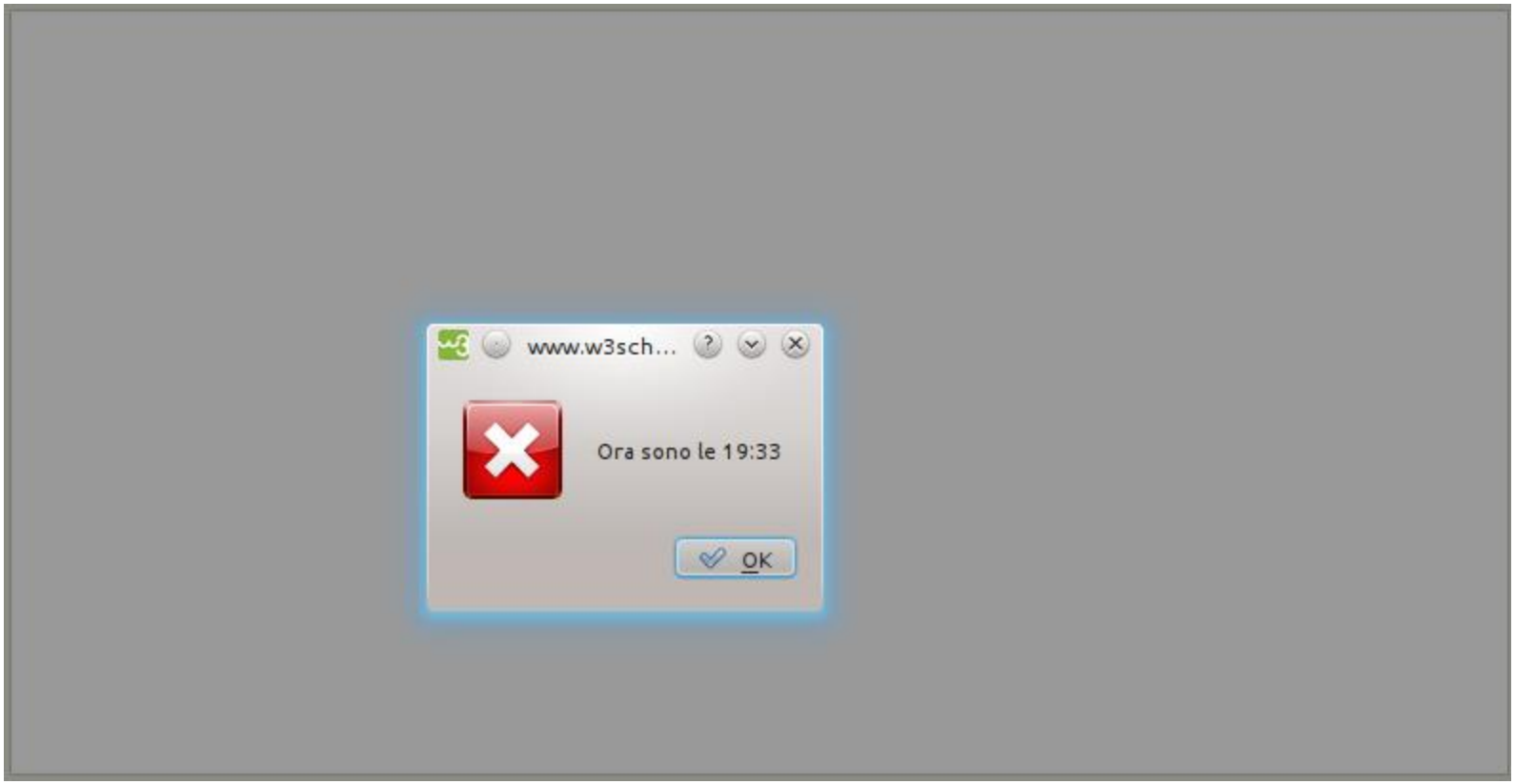
Timer – Esempio (I)



Timer – Esempio (I)



Timer – Esempio (I)



Timer – Esempio (II)


```
<html>
<head>
<script type="text/javascript">
var t;
function timeMsg()
{
t=setInterval("alertMsg()",2000);
}
function alertMsg()
{
alert("Hello");
}

function stopCount()
 {
 clearTimeout(t);
 }
</script>
</head>
<body>
<form>
<input type="button" value="Display alert box every 3 seconds"
onClick="timeMsg()" />
<input type="button" value="Please stop!" onClick="stopCount()" />
</form>
</body>

</html>
```

Timer – Esempio (III)


```
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}

function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>

<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

PROGRAMMAZIONE WEB

JAVASCRIPT - APPROFONDIMENTI

SINTASSI DI BASE, ARRAY, STRINGHE E ALTRI OGGETTI