



# Introducción a JavaScript, a sus tipos y valores

# JavaScript



## ◆ JavaScript

- Diseñado por Netscape en 1995 para ejecutar en un Navegador
  - ◆ Hoy se ha convertido en el **lenguaje del Web y Internet**

## ◆ Norma ECMA (European Computer Manufacturers Association)

- Versión soportada en navegadores actuales:
  - ◆ **ES5: ECMAScript v5**, Dic. 2009, (JavaScript 1.5)
- Navegadores antiguos soportan
  - ◆ ES3: ECMAScript v3, Dic. 1999, (JavaScript 1.3)



- ◆ Guía: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- ◆ Referencia: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- ◆ Libro: “*JavaScript Pocket Reference*”, D. Flanagan, O’Reilly 2012, 3rd Ed.

# Tipos, objetos y valores

## ◆ Tipos de JavaScript

### ■ **number**

- ◆ Literales de números: **32**, **1000**, **3.8**



### ■ **boolean**

- ◆ Los literales son los valores **true** y **false**



### ■ **string**

- ◆ Los literales de string son caracteres delimitados entre comillas o apóstrofes
  - **"Hola, que tal"**, **'Hola, que tal'**,
- ◆ Internacionalización con Unicode: **'Γεια σου, ίσως'**, **'嗨, 你好吗'**



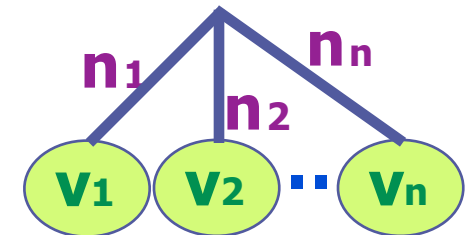
### ■ **undefined**

- ◆ **undefined**: representa **indefinido**

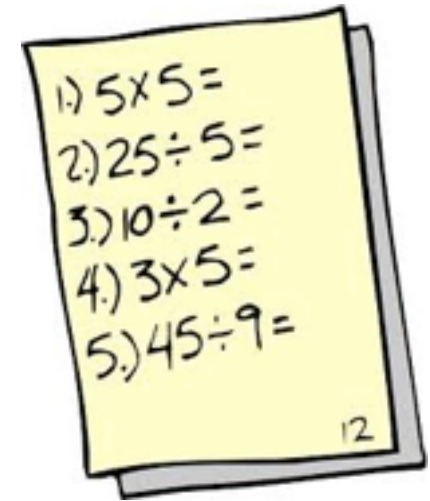
UNDEFINED

## ◆ **Objetos**: agregaciones estructuradas de valores

- Se agrupan en **clases**: **Object**, **Array**, **Date**, ...
  - ◆ Objeto **null**: valor especial que representa objeto nulo



# Operadores y expresiones



- ◆ JavaScript incluye **operadores** de tipos y objetos
  - Los **operadores** permiten formar **expresiones**
    - ◆ Componiendo **valores** con los operadores
      - Que Javascript evalua hasta obtener un resultado
- ◆ Por ejemplo, con las operaciones aritmeticas +, -, \*, /
  - podemos formar expresiones numéricas
    - ◆ Expresiones con sintaxis erronea abortan la ejecución del programa

<b>13 + 7</b>	<b>=&gt;</b>	<b>20</b>	<b>// Suma de números</b>
<b>13 - 7</b>	<b>=&gt;</b>	<b>6</b>	<b>// Resta de números</b>
<b>(8*2 - 4)/3</b>	<b>=&gt;</b>	<b>4</b>	<b>// Expresión con paréntesis</b>
<b>8 / * 3</b>	<b>=&gt;</b>	<b>Error_de_ejecución</b>	<b>// Ejecución se interrumpe</b>
<b>8 3</b>	<b>=&gt;</b>	<b>Error_de_ejecución</b>	<b>// Ejecución se interrumpe</b>

# Sobrecarga de operadores

- ◆ Algunos operadores tienen varias semánticas diferentes
- ◆ Por ejemplo, el operador **+** tiene 3 semánticas diferentes
  - **Suma de enteros** (operador binario)
  - **Signo de un número** (operador unitario)
  - **Concatenación de strings** (operador binario)



**13 + 7                    =>   20                    // Suma de números**

**+13                        =>   13                    // Signo de un número**

**"Hola " + "Pepe"   =>   "Hola Pepe"   // Concatenación de strings**



# Conversión de tipos en expresiones

- ◆ JavaScript realiza conversión automática de tipos
  - cuando hay ambigüedad en una expresión
    - ◆ utiliza las prioridades para resolver la ambigüedad
- ◆ La expresión **"13" + 7** es ambigua
  - porque combina un **string** con un **number**
    - ◆ JavaScript asigna más prioridad al **operador +** de strings, convirtiendo **7** a string
- ◆ La expresión **+"13"** también necesita conversión automática de tipos
  - El **operador +** solo está definido para **number**
    - ◆ JavaScript debe convertir el **string "13"** a **number** antes de aplicar operador **+**

<b>13 + 7</b>	<b>=&gt;</b>	<b>20</b>
<b>"13" + "7"</b>	<b>=&gt;</b>	<b>"137"</b>
<b>"13" + 7</b>	<b>=&gt;</b>	<b>"137"</b>
<b>+"13" + 7</b>	<b>=&gt;</b>	<b>20</b>

# Operador typeof

- ◆ El operador **typeof** permite conocer el tipo de un valor
  - Devuelve un string con el nombre del tipo
    - ◆ "number", "string", "boolean", "undefined", "object" y "function"

**typeof 7**

=> "number"



**typeof "hola"**

=> "string"



**typeof true**

=> "boolean"

**FALSE**  
**true**

**typeof undefined**

=> "undefined"

**UNDEFINED**

**typeof null**

=> "object"

**typeof new Date()**

=> "object"



**typeof new Function()**

=> "function"

**f(x)**

Los operadores están ordenados con prioridad descendente. Mas altos más prioridad.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators>

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator\\_Precedence](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence)

<b>.</b>	<b>Acceso a propiedad o invocar método; índice a array</b>
<b>new</b>	<b>Crear objeto con constructor de clase</b>
<b>()</b>	<b>Invocación de función/método o agrupar expresión</b>
<b>++ --</b>	<b>Pre o post auto-incremento; pre o post auto-decremento</b>
<b>! ~</b>	<b>Negación lógica (NOT); complemento de bits</b>
<b>+ -</b>	<b>Operador unitario, números. signo positivo; signo negativo</b>
<b>delete</b>	<b>Borrar propiedad de un objeto</b>
<b>typeof void</b>	<b>Devolver tipo; valor indefinido</b>
<b>* / %</b>	<b>Números. Multiplicación; división; modulo (o resto)</b>
<b>+</b>	<b>Concatenación de string</b>
<b>+ -</b>	<b>Números. Suma; resta</b>
<b>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</b>	<b>Desplazamientos de bit</b>
<b>&lt; &lt;= &gt; &gt;=</b>	<b>Menor; menor o igual; mayor; mayor o igual</b>
<b>instanceof in</b>	<b>¿objeto pertenece a clase?; ¿propiedad pertenece a objeto?</b>
<b>== != === !==</b>	<b>Igualdad; desigualdad; identidad; no identidad</b>
<b>&amp;</b>	<b>Operacion y (AND) de bits</b>
<b>^</b>	<b>Operacion ó exclusivo (XOR) de bits</b>
<b> </b>	<b>Operacion ó (OR) de bits</b>
<b>&amp;&amp;</b>	<b>Operación lógica y (AND)</b>
<b>  </b>	<b>Operación lógica o (OR)</b>
<b>?:</b>	<b>Asignación condicional</b>
<b>=</b>	<b>Asignación de valor</b>
<b>OP=</b>	<b>Asig. con operación: += -= *= /= %= &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;= &amp;= ^=  =</b>
<b>,</b>	<b>Evaluación múltiple</b>

## Operadores JavaScript

**+"3" + 7 => 10**

“+” unitario tiene mas prioridad y se evalúa antes que “+” binario



Los operadores están ordenados con prioridad descendente. Mas altos más prioridad.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators>

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator\\_Precedence](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence)

<b>.</b>	<b>Acceso a propiedad o invocar método; índice a array</b>
<b>new</b>	<b>Crear objeto con constructor de clase</b>
<b>()</b>	<b>Invocación de función/método o agrupar expresión</b>
<b>++ --</b>	<b>Pre o post auto-incremento; pre o post auto-decremento</b>
<b>! ~</b>	<b>Negación lógica (NOT); complemento de bits</b>
<b>+ -</b>	<b>Operador unitario, números. signo positivo; signo negativo</b>
<b>delete</b>	<b>Borrar propiedad de un objeto</b>
<b>typeof void</b>	<b>Devolver tipo; valor indefinido</b>
<b>* / %</b>	<b>Números. Multiplicación; división; modulo (o resto)</b>
<b>+</b>	<b>Concatenación de string</b>
<b>+ -</b>	<b>Números. Suma; resta</b>
<b>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</b>	<b>Desplazamientos de bit</b>
<b>&lt; &lt;= &gt; &gt;=</b>	<b>Menor; menor o igual; mayor; mayor o igual</b>
<b>instanceof in</b>	<b>¿objeto pertenece a clase?; ¿propiedad pertenece a objeto?</b>
<b>== != === !==</b>	<b>Igualdad; desigualdad; identidad; no identidad</b>
<b>&amp;</b>	<b>Operacion y (AND) de bits</b>
<b>^</b>	<b>Operacion ó exclusivo (XOR) de bits</b>
<b> </b>	<b>Operacion ó (OR) de bits</b>
<b>&amp;&amp;</b>	<b>Operación lógica y (AND)</b>
<b>  </b>	<b>Operación lógica o (OR)</b>
<b>?:</b>	<b>Asignación condicional</b>
<b>=</b>	<b>Asignación de valor</b>
<b>OP=</b>	<b>Asig. con operación: += -= *= /= %= &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;= &amp;= ^=  =</b>
<b>,</b>	<b>Evaluación múltiple</b>

## Operadores JavaScript

$$8 * 2 - 4 \Rightarrow 12$$

“\*” tiene mas prioridad y se evalúa antes que “-”. Es equivalente a:

$$(8 * 2) - 4 \Rightarrow 12$$

Los operadores están ordenados con prioridad descendente. Mas altos más prioridad.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators>

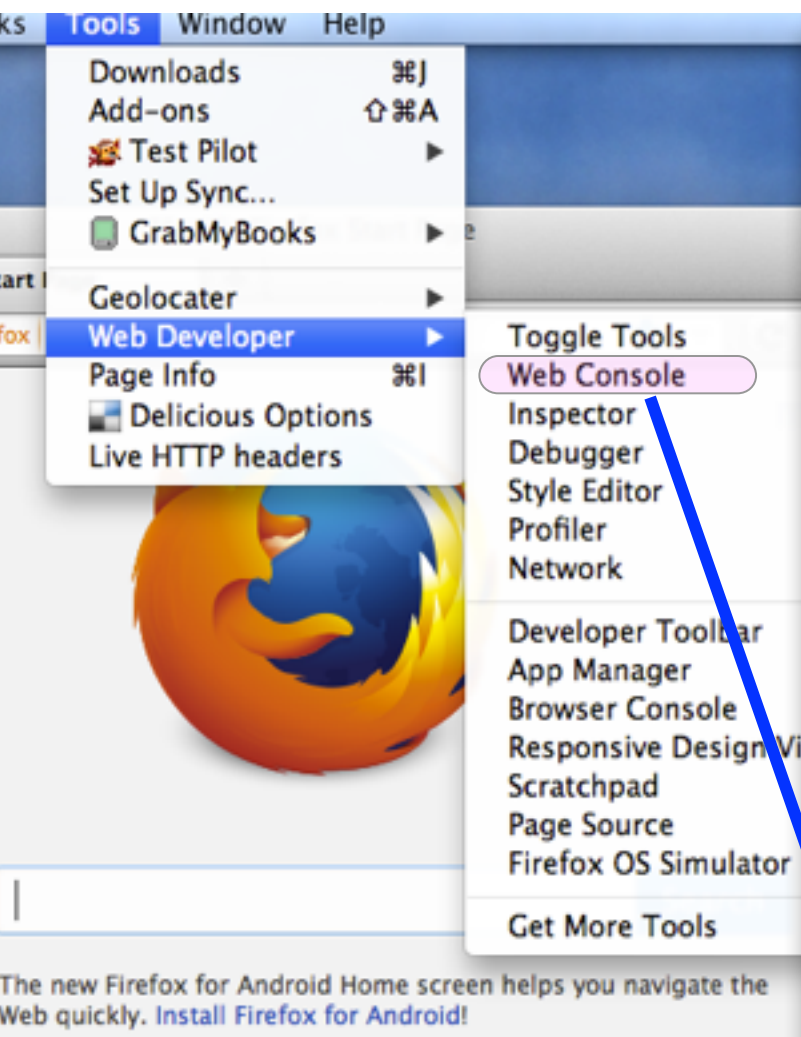
[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator\\_Precedence](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence)

<b>.</b>	<b>Acceso a propiedad o invocar método; índice a array</b>
<b>new</b>	<b>Crear objeto con constructor de clase</b>
<b>()</b>	<b>Invocación de función/método o agrupar expresión</b>
<b>++ --</b>	<b>Pre o post auto-incremento; pre o post auto-decremento</b>
<b>! ~</b>	<b>Negación lógica (NOT); complemento de bits</b>
<b>+ -</b>	<b>Operador unitario, números. signo positivo; signo negativo</b>
<b>delete</b>	<b>Borrar propiedad de un objeto</b>
<b>typeof void</b>	<b>Devolver tipo; valor indefinido</b>
<b>* / %</b>	<b>Números. Multiplicación; división; modulo (o resto)</b>
<b>+</b>	<b>Concatenación de string</b>
<b>+ -</b>	<b>Números. Suma; resta</b>
<b>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</b>	<b>Desplazamientos de bit</b>
<b>&lt; &lt;= &gt; &gt;=</b>	<b>Menor; menor o igual; mayor; mayor o igual</b>
<b>instanceof in</b>	<b>¿objeto pertenece a clase?; ¿propiedad pertenece a objeto?</b>
<b>== != === !==</b>	<b>Igualdad; desigualdad; identidad; no identidad</b>
<b>&amp;</b>	<b>Operacion y (AND) de bits</b>
<b>^</b>	<b>Operacion ó exclusivo (XOR) de bits</b>
<b> </b>	<b>Operacion ó (OR) de bits</b>
<b>&amp;&amp;</b>	<b>Operación lógica y (AND)</b>
<b>  </b>	<b>Operación lógica o (OR)</b>
<b>?:</b>	<b>Asignación condicional</b>
<b>=</b>	<b>Asignación de valor</b>
<b>OP=</b>	<b>Asig. con operación: += -= *= /= %= &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;= &amp;= ^=  =</b>
<b>,</b>	<b>Evaluación múltiple</b>

## Operadores JavaScript

$$8*(2 - 4) \Rightarrow -16$$

El paréntesis tiene más prioridad y obliga a evaluar primero “-” y luego “\*”



Number - JavaScript | MDN x Miriada X: Desarrollo en ... x

https://www.miriadax.net/web/html5mooc/edicion?p\_p\_id=e

Bancos Varios Cursos Tools Rails HTML5

✎ X ↕ ↕

- Modulo 1: Introducción a Internet, la nube, la arquitectura de la Web, HTML5 y CSS
- ✎ X ↕ ↕
- Modulo 2: Introducción a JavaScript y a las aplicaciones Web en HTML5, así como la publicación en la nube
- ✎ X ↕ ↕

**Añadir**

Transparencias y ejemplos del modulo

✎ X ↕ ↕

Tema 2.1: Tipos y valores de JavaScript

✎ X ↕ ↕

typeof ("10"+23)

- 10
- 23
- 33
- "23"
- "1023"
- "1033"
- "number"
- "string"
- "boolean"

typeof (10+23)

Net CSS JS Security Logging Clear Filter

```
< typeof("10"+23)
  > "string"
  >> typeof("10"+23)
```

La consola nos va mostrando el resultado de ejecutar las sentencias JavaScript

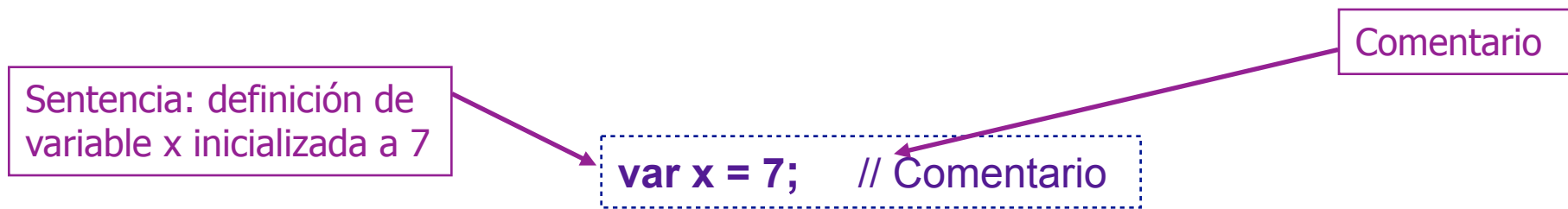
Aquí se introduce la sentencia



# Programa, sentencia, variable y comentario

# Programa, sentencias y comentarios

- ◆ Un programa es una secuencia de sentencias
  - que se ejecutan en el orden en que han sido definidas
- ◆ Las sentencias realizan tareas al ejecutarse en un ordenador
  - Son los elementos activos de un programa
- ◆ Los comentarios solo tienen valor informativo
  - para entender o recordar como funciona el programa



# Comentarios

- ◆ Los comentarios son mensajes informativos
  - Deben ser claros, concisos y explicar todo lo importante del programa
    - ◆ Incluso el propio autor los necesita con el tiempo para recordar detalles del programa
- ◆ En JavaScript hay 2 tipos de comentarios
  - Monolínea: Delimitados por // y **final de línea**
  - Multilínea: Delimitados por /\* y \*/
    - ◆ **OJO!** Los comentarios multi-línea tienen problemas con las expresiones regulares

/\* Ejemplo de comentario multilínea que ocupa 2 líneas  
-> al tener ambigüedades, se recomienda utilizarlos con cuidado \*/

```
var x = 1; // Comentario monolínea que acaba al final de esta línea
```

# Variables y estado del programa

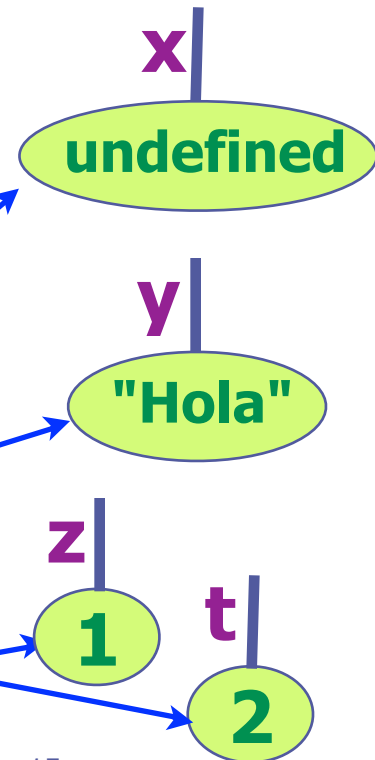
## ◆ Las variables se crean con la sentencia de definición de variables

- Comienza con la palabra reservada **var**
  - ◆ Seguida de la variable, a la que se puede asignar un valor inicial
- Se pueden crear varias variables en una sentencia
  - ◆ separando las definiciones por comas

## ◆ Estado del programa

- conjunto de valores contenido en todas sus variables

```
var x;           // crea la variable x y asigna undefined  
var y = "Hola"; // crea y, asignandole el valor "Hola"  
var z = 1, t = 2; // crea x e y, asignandoles 1 y 2 respectivamente
```



# Sentencia de asignación de variables

- ◆ Una variable es un contenedor de valores
  - La sentencia de asignación introduce un nuevo valor en la variable
    - ◆ Modificando, por tanto, el estado del programa
- ◆ Las variables de JavaScript son **no tipadas**
  - pueden contener **valores de cualquier tipo** de JavaScript

```
var x = 5; // Crea la variable x y le asigna el valor inicial 5
```

```
x = "Hola"; // Asigna el string (texto) "hola" a la variable x
```

```
x = new Date(); // Asigna objeto Date a la variable x
```

x |  
5

x |  
"Hola"

x |  
Mon Sep 02 2013 09:16:47  
GMT+0200 (CEST)



## STATEMENT SINTAXIS

<b>block</b>	<b>{ statements };</b>
<b>break</b>	<b>break [label];</b>
<b>case</b>	<b>case expression:</b>
<b>continue</b>	<b>continue [label];</b>
<b>debugger</b>	<b>debugger:</b>
<b>default</b>	<b>default:</b>
<b>do/while</b>	<b>do statement</b> <b>while(expression);</b>
<b>empty</b>	<b>;</b>
<b>expression</b>	<b>expression;</b>
<b>for</b>	<b>for(init; test; incr)</b> <b>statement</b>
<b>for/in</b>	<b>for (var in object)</b> <b>statement</b>
<b>function</b>	<b>function name([param[,...]])</b> <b>{ body }</b>
<b>if/else</b>	<b>if (expr) statement1</b> <b>[else statement2]</b>
<b>label</b>	<b>label: statement</b>
<b>return</b>	<b>return [expression];</b>
<b>switch</b>	<b>switch (expression)</b> <b>{ statements }</b>
<b>throw</b>	<b>throw expression;</b>
<b>try</b>	<b>try {statements}</b> <b>[catch { statements }]</b> <b>[finally { statements }]</b>
<b>strict</b>	<b>"use strict";</b>
<b>var</b>	<b>var name [ = expr ] [ ,... ];</b>
<b>while</b>	<b>while (expression) statement</b>
<b>with</b>	<b>with (object) statement</b>

## DESCRIPCIÓN DE LA SENTENCIA JAVASCRIPT

**Agrupar un bloque de sentencias como 1 sentencia**  
**Salir del bucle o switch o sentencia etiquetada**  
**Etiquetar sentencia dentro de sentencia switch**  
**Salto a sig. iteración de bucle actual/etiquetado**  
**Punto de parada (breakpoint) del depurador**  
**Etiquetar setencia default en sentencia switch**  
**Alternativa al bucle while con condición al final**

**Sentencia vacía, no hace nada**

**Evaluar expresión (incluye asignación a variable)**

**Bucle sencillo. "init": inicialización;**

**"test": condición; "incr": acciones final bucle**

**Enumerar las propiedades del objeto "object"**

**Declarar una función llamada "name"**

**Ejecutar statement1 o statement2**

**Etiquetar sentencia con nombre "label"**

**Devolver un valor desde una función**

**Multiopción con etiquetas "case" o "default"**

**Lanzar una excepción**

**Gestionar excepciones**

**Activar restricciones strict a script o función**

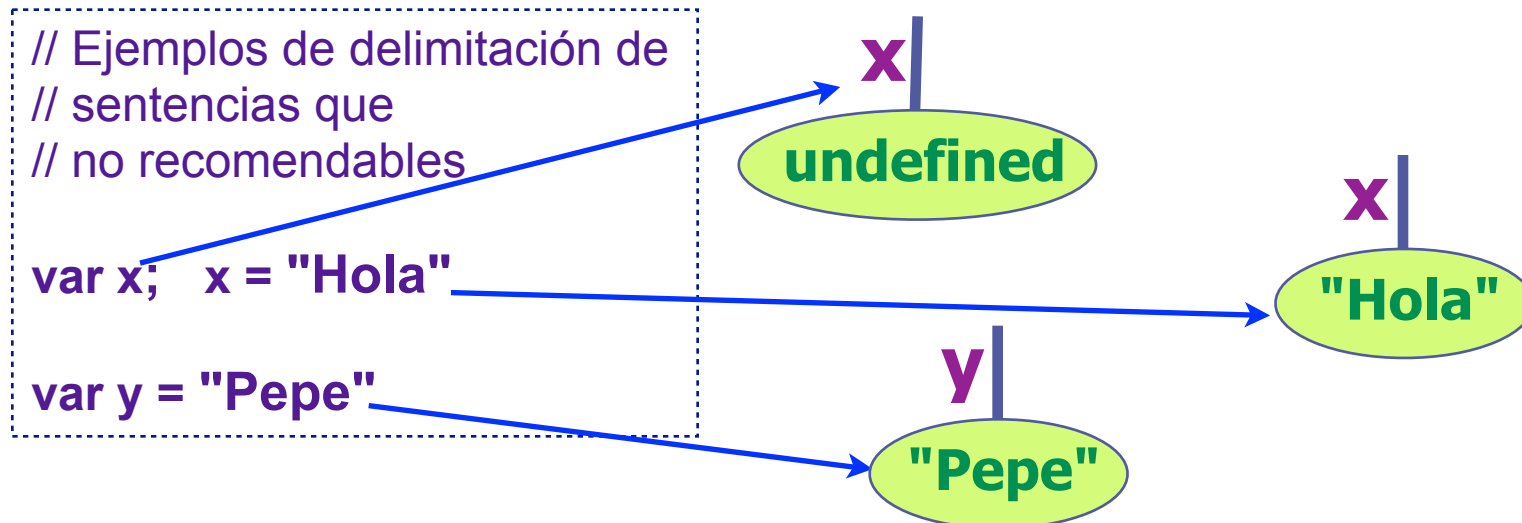
**Declarar e inicializar una o mas variables**

**Bucle básico con condición al principio**

**Extender cadena de ámbito (no recomendado)**

# Delimitación de sentencias

- ◆ “;” delimita el final de una sentencia
- ◆ El final de sentencia también puede delimitarse con nueva línea
  - Pero hay ambigüedades y no se recomienda hacerlo
- ◆ **Recomendación:** cada sentencia en **una línea terminada con “;”**
  - > es mas legible y seguro



# Nombres de variables

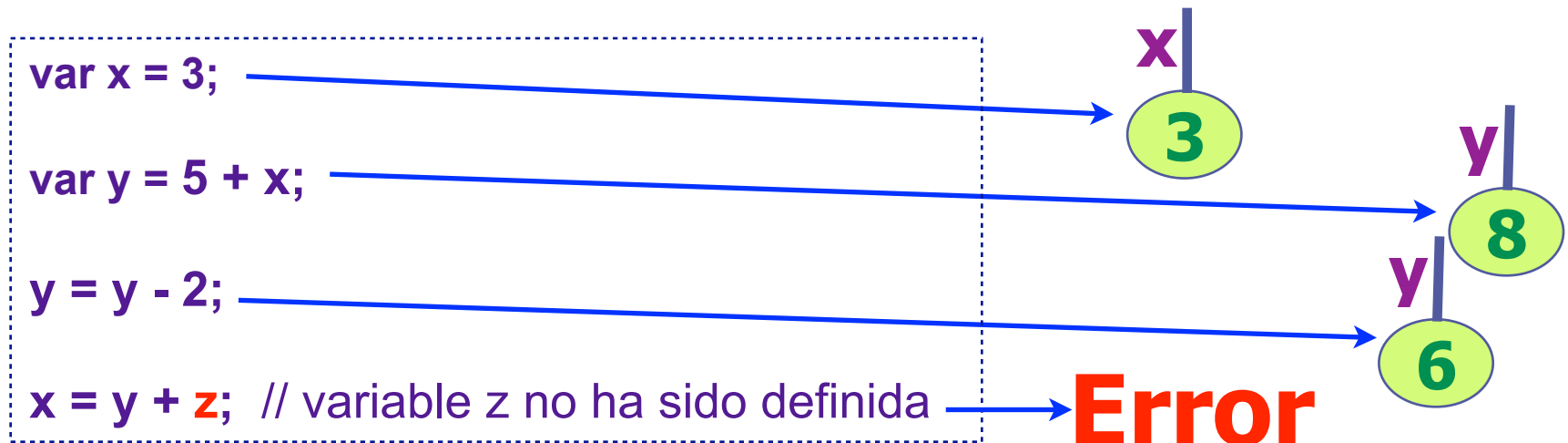
- ◆ El **nombre** (o identificador) de una variable debe comenzar por:
  - **letra, \_ o \$**
    - ◆ El nombre pueden contener además **números**
  - Nombres **bien contruidos**: **x, ya\_vás, \$A1, \$, \_43dias**
  - Nombres **mal contruidos**: **1A, 123, %3, v=7, a?b, ..**
    - ◆ Nombre incorrecto: da error\_de\_ejecución e interrumpe el programa
- ◆ Un nombre de variable
  - **no** debe ser una **palabra reservada** de JavaScript
- ◆ Las variables son sensibles a **mayúsculas**
  - **mi\_var** y **Mi\_var** son variables distintas



# Expresiones con variables

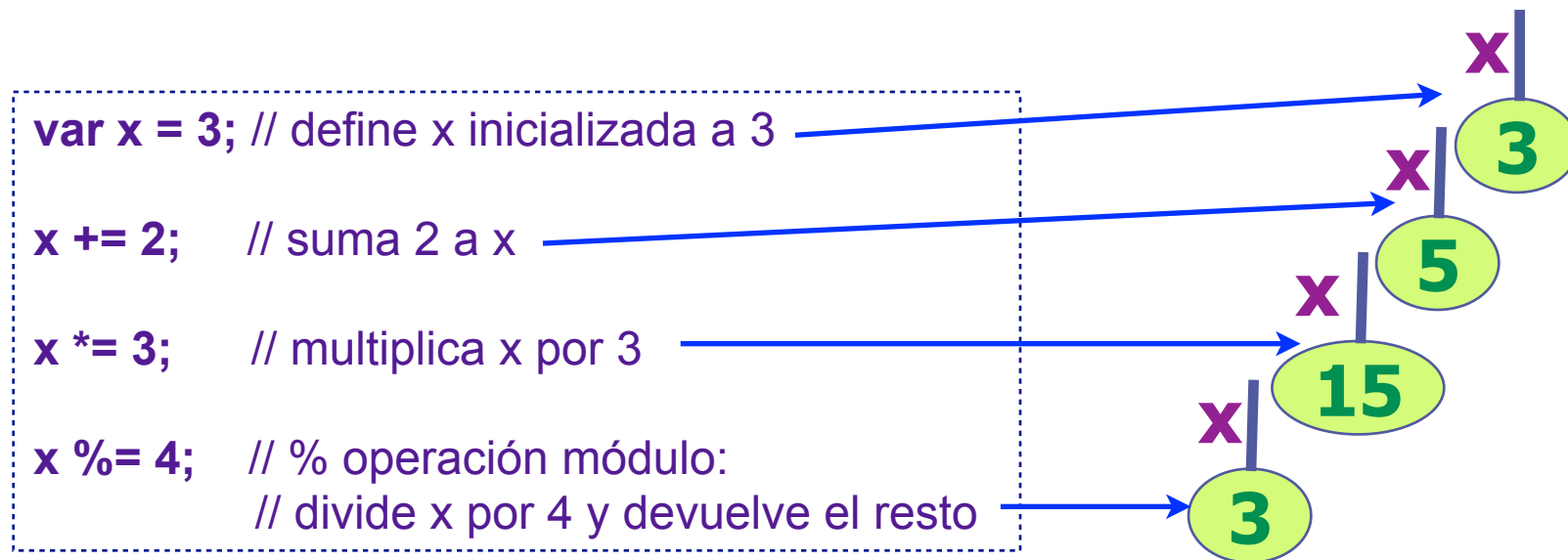
# Expresiones con variables

- ◆ Una variable representa el valor que contiene
  - Puede ser usada en expresiones como cualquier otro valor
- ◆ Una variable puede utilizarse en la expresión que se asigna a ella misma
  - La parte derecha usa el valor anterior a la ejecución de la sentencia
    - ◆ En `y = y - 2;` la variable `y` tiene el valor 8, por lo que se asigna a `y` un **6** (8-2)
- ◆ Usar una **variable no definida** en una expresión
  - provoca un **error** y la ejecución del **programa se interrumpe**



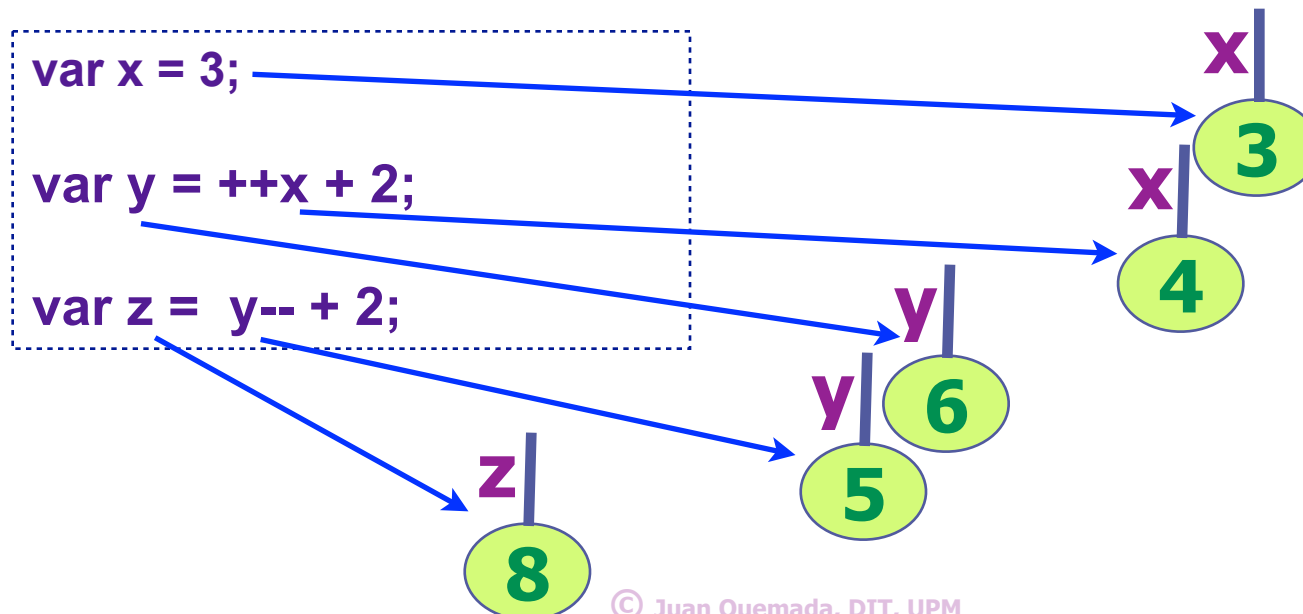
# Asignación con operación

- ◆ Es muy común modificar el valor de una variable
  - sumando, restando, .... algún valor
    - ◆ Por ejemplo,  $x = x + 7$ ;  $y = y - 3$ ;  $z = z * 8$ ; .....
- ◆ JavaScript tiene operadores especiales para estas operaciones
  - $+=$ ,  $-=$ ,  $*=$ ,  $/=$ ,  $\%=$ , .....( aplica a operadores lógicos, desplazamiento, ..)
    - ◆  $x += 7$ ; será lo mismo que  $x = x + 7$ ;



# Pre y post incremento o decremento

- ◆ JavaScript posee los operadores ++ y -- de auto-incremento/decremento
  - auto-incremento (++) suma 1 a la variable a la que se aplica
  - auto-decremento (--) resta 1 a la variable a la que se aplica
- ◆ ++ y -- se aplican a la izquierda o derecha de una variable en una expresión
  - modificando el valor antes o después de evaluar la expresión
    - ◆ Al producir efectos laterales y programas críticos, no se recomienda un uso limitado





# Scripts y entrada/salida



# Entrada/Salida y ejecución de expresiones

- ◆ La sentencia de **ejecución de expresiones** puede evaluar expresiones
  - como por ejemplo, **3+2;** o **alert("Texto");**
    - ◆ sin asignar el resultado, puede ser: **x = 3+2;** o **3+2;**
- ◆ Estas sentencias se utilizan habitualmente para comunicar con el exterior
  - p.e. **alert("Texto");** muestra una ventana desplegable al usuario
- ◆ Una expresión sin efecto lateral, solo genera un valor
  - Si ese valor no se guarda en una variable
    - ◆ La instrucción no tiene ningún efecto en el programa, solo consume recursos

```
alert("Texto");           // expresiones útiles, envían mensaje al exterior
document.write("Texto");
```

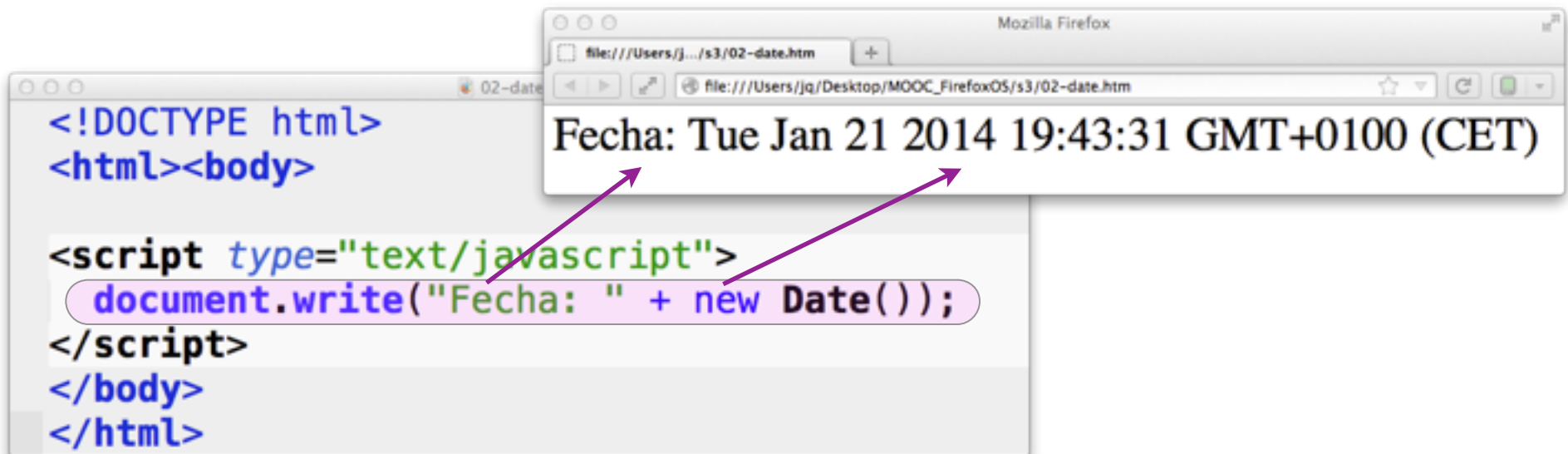
```
var x = 3;    // definición e inicialización de variable
```

```
x*5 - 2;     // es una expresión correcta, pero inútil al no guardar el resultado
```

```
x = x*5 + 2; // asignación, es una expresión útil porque guarda el resultado
```

# Ejemplo I: Script con fecha

- ◆ Script: programa JavaScript insertado en una página HTML
  - Delimitado con las marcas **<script>** con atributo “type=text/javascript”
    - ◆ Se ejecuta al cargar la página HTML en el navegador
- ◆ **document.write("Texto")** inserta **"Texto"** en la página Web
  - En lugar del bloque script, al cargar la página y ejecutar el script
    - ◆ **document.writeln("Texto")** inserta además de **Texto**, **nueva línea** al final



# Ejemplo II: Script con expresión

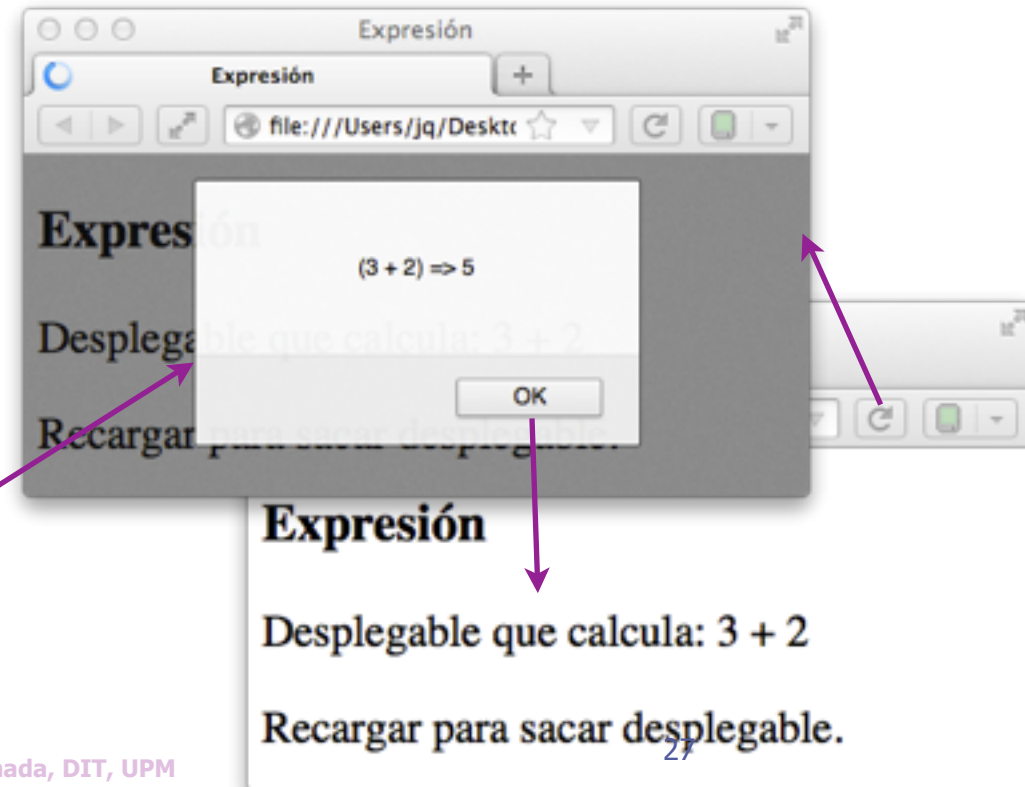
- ◆ Script: programa JavaScript insertado en una página HTML
  - Delimitado con la marca `<script>` con atributo "type=text/javascript"
    - ◆ Se ejecuta al cargar la página HTML en el navegador
- ◆ `alert("Texto")` muestra "Texto" en un desplegable
  - se utiliza para alertar al usuario sobre algún evento o resultado

```
<!DOCTYPE html>
<html><head><title>Expresión</title>
  <meta charset="UTF-8"></head>

<body>
<h3>Expresión</h3>

Desplegable que calcula: 3 + 2
<p>
Recargar para sacar desplegable.

<script type="text/javascript">
alert("(3 + 2) => " + (3+2));
</script>
</body>
</html>
```



Expresión

(3 + 2) => 5

OK

Expresión

Desplegable que calcula: 3 + 2

Recargar para sacar desplegable.

# Funciones alert(), confirm() y prompt()

## ◆ Interacción sencilla basada en “pop-ups”

### ◆ alert(msj):

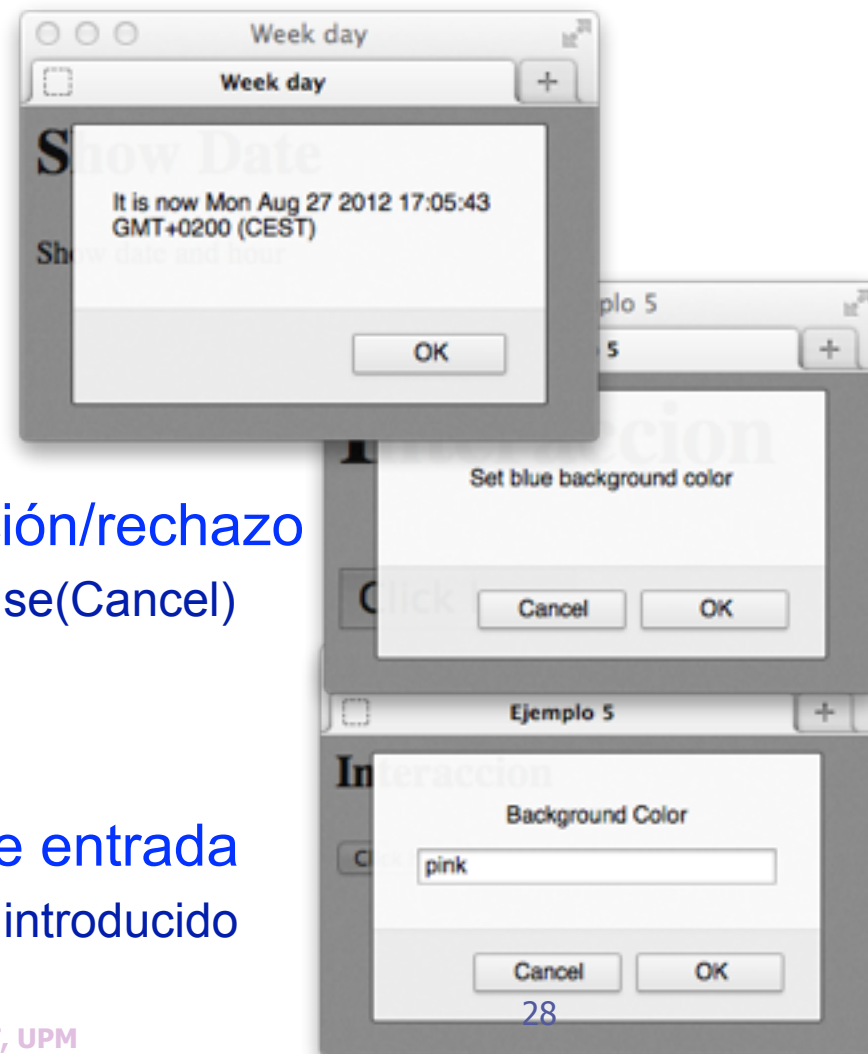
- presenta un mensaje al usuario
  - ◆ Retorna al pulsar OK

### ◆ confirm(msj):

- Presenta mensaje y pide confirmación/rechazo
  - ◆ Retorna al pulsar, devuelve true(Ok)/false(Cancel)

### ◆ prompt(msj):

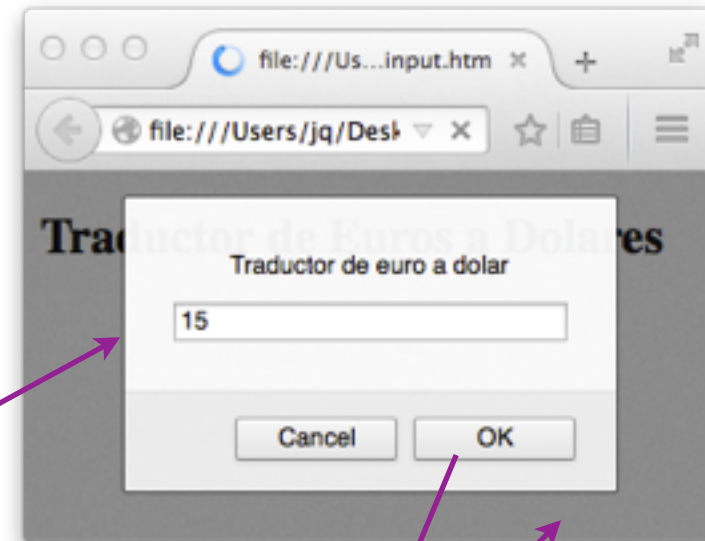
- Presenta mensaje y pide un dato de entrada
  - ◆ Al pulsar OK, retorna y devuelve string introducido
    - Si se pulsa Cancel devuelve “null”





## Ejemplo III: Script conversor

- ◆ El script pide el número de Euros a convertir a Dólares
  - con el desplegable de la función **prompt()**
- ◆ Cuando el usuario lo introduce
  - calcula el equivalente en dólares ( \* 1,34)
    - ◆ y lo presenta con **document.write(....)**
- ◆ Recargando la página
  - se vuelve a ejecutar el programa



```
<!DOCTYPE html>
<html><body>
<h2>Traductor de Euros a Dolares</h2>
<script type="text/javascript">
  var x = prompt("Traductor de euro a dolar");
  document.write(x + " Euros son "
    + x*1.34 + " Dolares.");
</script>
</body>
</html>
```