JavaScript:
Booleanos y sentencia if/else

Booleanos: true y false

El **tipo booleano** solo posee los valores: **true** y **false**.

Un booleano se obtiene como resultado de:

- **comparación de orden de números**
  - menor: **<** menor_o_igual: **<=**
  - mayor: **>** mayor_o_igual: **>=**
- **comparación de identidad de valores**
  - identidad: **===** no_identidad: **!==**

Se combinan con **operadores lógicos:**

negación: **!**

```
!true  => false
!false => true
```

op. y: **&&**

```
true  && true  => true
true  && false => false
false && true  => false
false && false => false
```

op. o: **||**

```
true  || true  => true
true  || false => true
false || true  => true
false || false => false
```

Booleanos: true y false

El **tipo booleano** solo posee los valores: **true** y **false**.

Un booleano se obtiene como resultado de:

- **comparación de orden de números**
  menor: **<** menor_o_igual: **<=**
  mayor: **>** mayor_o_igual: **>=**
- **comparación de identidad de valores**
  identidad: **===** no_identidad: **!==**

Se combinan con **operadores lógicos:**

negación: **!**
!true => false
!false => true

op. y: **&&**
true && true => true
true && false => false
false && true => false
false && false => false

op. o: **||**
true || true => true
true || false => true
false || true => true
false || false => false

10-bool.htm

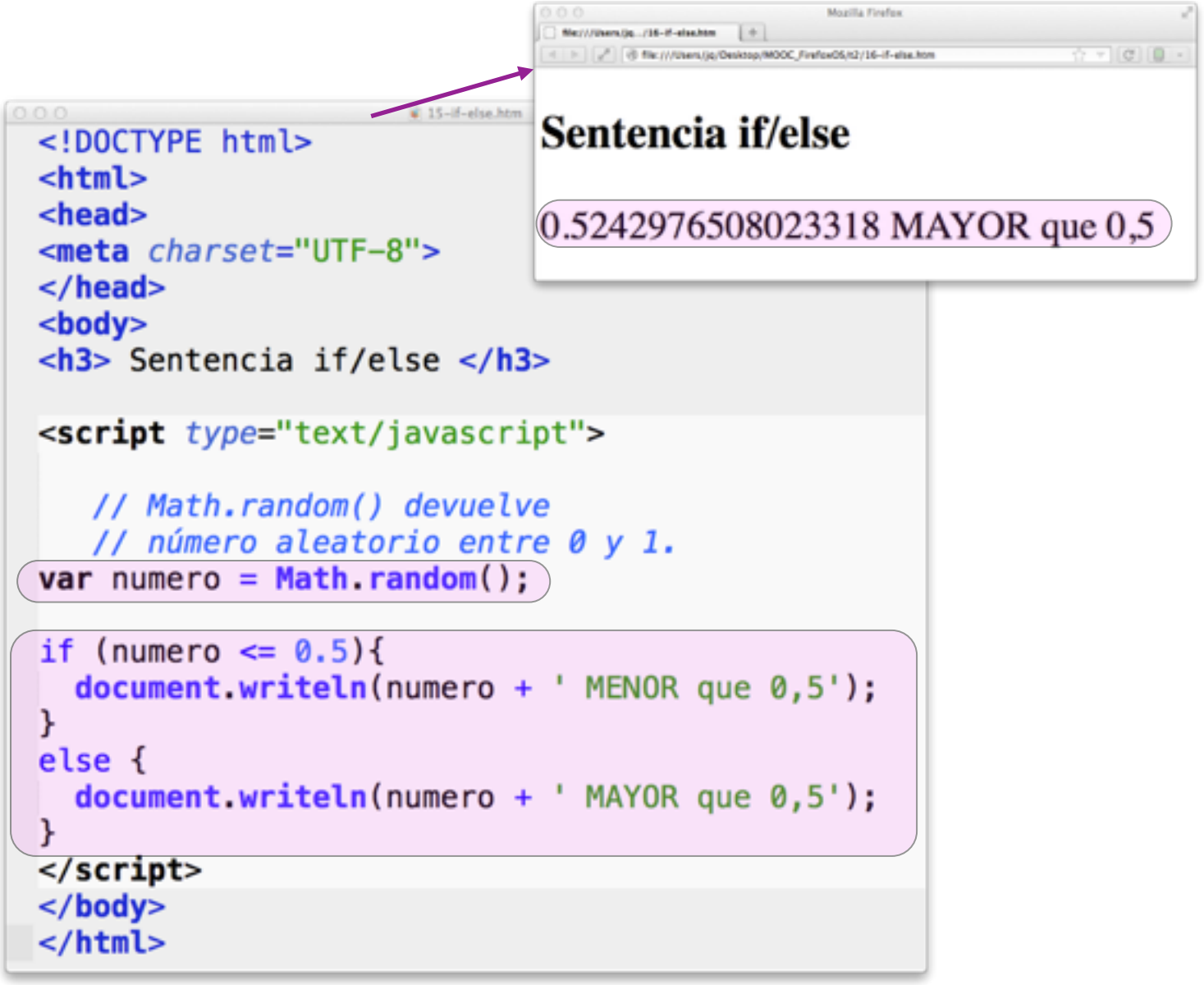
```
1  <!DOCTYPE html><html>
2  <head><meta charset="UTF-8"></head>
3  <body>
4  <h3> Booleanos </h3>
5
6  <script type="text/javascript">
7
8  function entre_3_y_8 (x) {
9      return (3 <= x) && (x <= 8);
10 };
11
12 document.write("entre_3_y_8(4) => " + entre_3_y_8(4));
13 document.write("<p>");
14 document.write("entre_3_y_8(2) => " + entre_3_y_8(2));
15 </script>
16 </body>
17 </html>
```

Booleanos

entre_3_y_8(4) => true

entre_3_y_8(2) => false

Sentencia if/else



Sentencia if/else


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<h3> Sentencia if/else </h3>

<script type="text/javascript">

   // Math.random() devuelve
   // número aleatorio entre 0 y 1.
var numero = Math.random();

if (numero <= 0.5){
  document.writeln(numero + ' MENOR que 0,5');
}
else {
  document.writeln(numero + ' MAYOR que 0,5');
}
</script>
</body>
</html>
```

Crea la variable **numero** y le asigna un valor aleatorio entre 1 y 1.

La sentencia **if/else** comienza con la palabra reservada **if**

El **primer bloque** de sentencias va después de la condición, delimitado entre llaves: **{}**

La **condición** va entre paréntesis. Según sea **true** o **false**, se ejecuta el primer o el segundo bloque.

La **condición** se evalúa a **true** o a **false** en función del **estado del progr.** (valores de las variables).

El **segundo bloque** de sentencias va precedido por la palabra reservada **else** y delimitado entre llaves: **{}**

Sentencia if/else


```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<h3> Sentencia if </h3>

<script type="text/javascript">

   // Math.random() devuelve
   // número aleatorio entre 0 y 1.
var numero = Math.random();
var str = ' MAYOR que 0,5';

if (numero <= 0.5){
  str = ' MENOR que 0,5';
}

document.writeln(numero + str);
</script>
</body>
</html>
```

Sentencia if/else

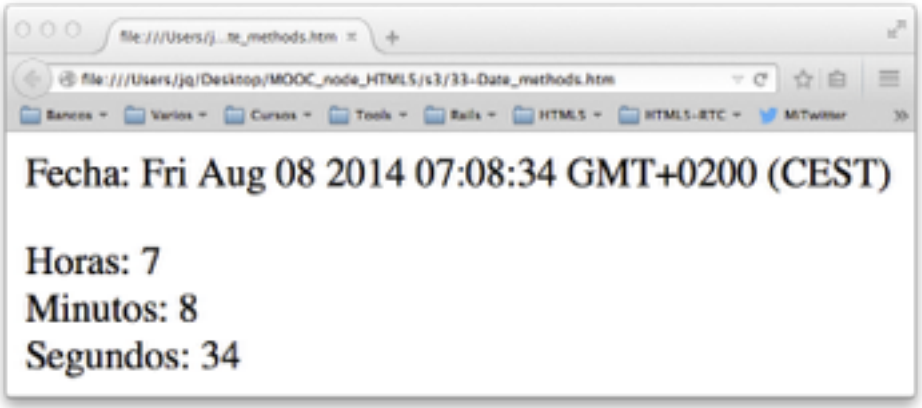
Este programa es equivalente al anterior, pero con diferente estructura. No utiliza la parte **else** (opcional), pero añade la variable **str = ' MAYOR que 1,5'**

La **sentencia if** tiene ahora solo la primera parte. Esta cambia el contenido asignado a la variable por **str = ' MENOR que 1,5'**

El mensaje enviado a consola se genera concatenando **numero** y **str**.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<h3> Sentencia if </h3>

<script type="text/javascript">

   // Math.random() devuelve
   // número aleatorio entre 0 y 1.
var numero = Math.random();
var str = ' MAYOR que 0,5';

if (numero <= 0.5){
  str = ' MENOR que 0,5';
}

document.writeln(numero + str);
</script>
</body>
</html>
```

La clase Date: fechas y horas



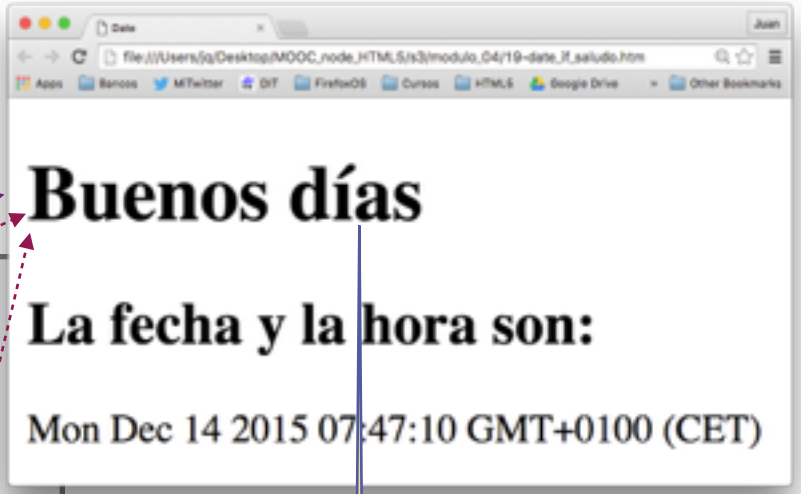
new Date() crea un objeto con la fecha y la hora en el momento de crearlo.

Sus métodos **toString(), getDay(), ..** permiten obtener fecha, día, ..

```
<!DOCTYPE html>
<html><body>
<script>
var fecha = new Date();

document.write(
  "Fecha: "+ fecha.toString() + "<p>"
+ "Horas: "+ fecha.getHours() + "<br>"
+ "Minutos: "+ fecha.getMinutes() + "<br>"
+ "Segundos: "+ fecha.getSeconds() + "<br>")
</script>
</body></html>
```

- **Clase Date**
  - **new Date()** crea objetos pertenecientes a esta clase
    - **new Date()** devuelve un objeto con fecha y hora del reloj en el momento de crear el objeto
- Sobre los objetos de esta clase se pueden invocar los métodos definidos para ella
  - **getDay()**, **getHours()**, **getMinutes()**, **getFullYear()**, …..
    - Invocar un método que no está definido en un objeto da un error_de_ejecución
- Más información sobre sus métodos y propiedades en:
  - https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Date

if/else-if encadenado



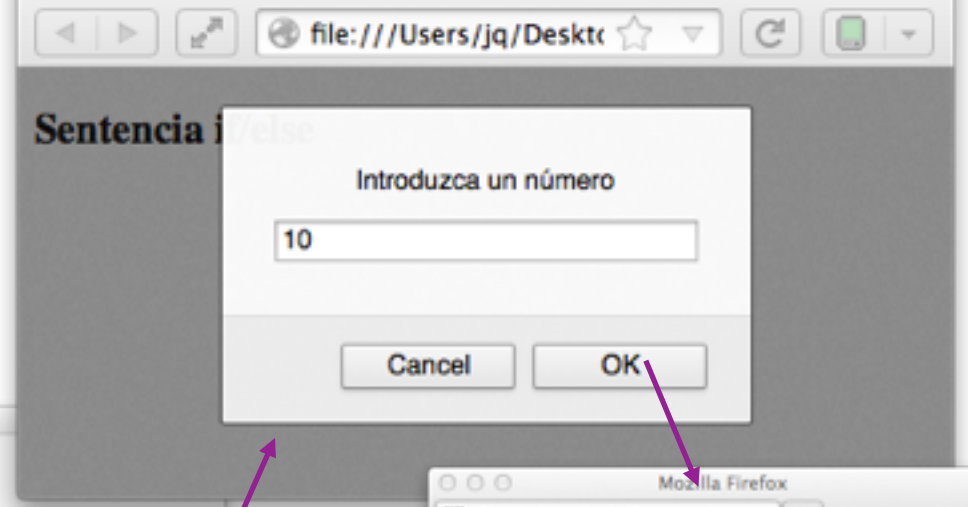

```
<!DOCTYPE html><html>
<head>
  <title>Date</title>
  <meta charset="UTF-8">
</head>

<body>
  <h1 id="h1"></h1>
  <h2 id="h2">La fecha y la hora son:</h2>

  <div id="fecha"><div>

<script type="text/javascript">
  var fecha = new Date();
  var msj;

  if      (fecha.getHours() < 7)  { msj = "Buenas noches";}
  else if (fecha.getHours() < 12) { msj = "Buenos días";}
  else if (fecha.getHours() < 21) { msj = "Buenas tardes";}
  else                            { msj = "Buenas noches";}

  document.getElementById("h1").innerHTML    = msj;
  document.getElementById("fecha").innerHTML = fecha;
</script>

</body>
</html>
```

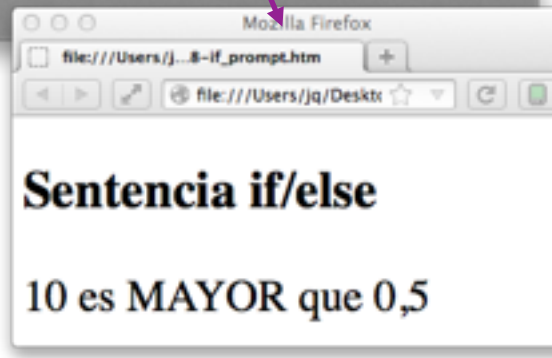
En esta variante añadimos al ejemplo fecha y hora un título de nivel 1 (h1) vacío, en el que se inserta un saludo (Buenos días/tardes/noches) en función de la hora del día.

Las **sentencias if/else** pueden encadenarse así, para comprobar múltiples condiciones en cascada (de las cuales solo se ejecutará una), tal y como se hace para seleccionar el saludo.

Final del tema

Ejemplo con prompt()





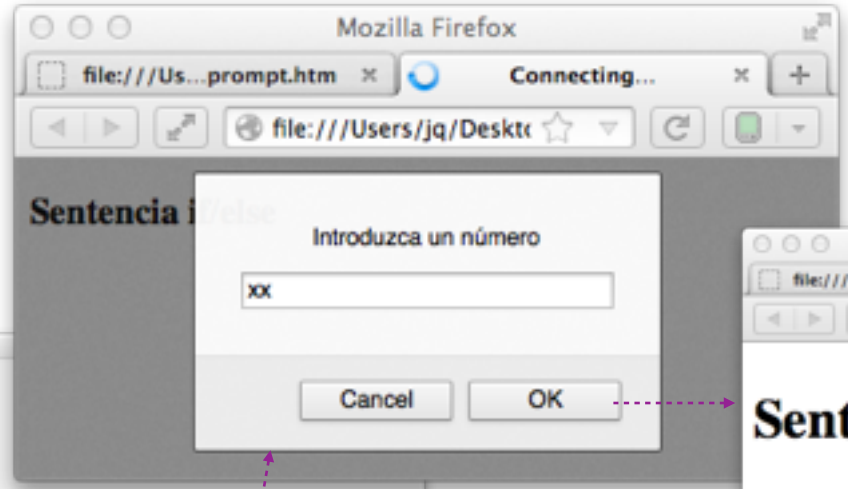
24-if_prompt.htm

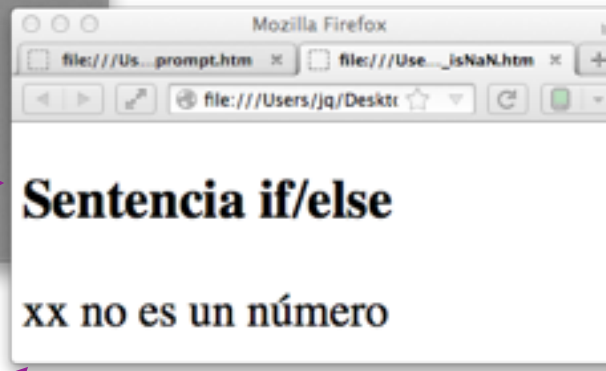
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<h3> Sentencia if/else </h3>

<script type="text/javascript">

   // Prompt pide un dato con un desplegable
var numero = prompt("Introduzca un número");

if (numero <= 0.5){
  document.writeln(numero + ' es MENOR que 0,5');
}
else {
  document.writeln(numero + ' es MAYOR que 0,5');
}
</script>
</body>
</html>
```

En este ejemplo el número lo teclea el usuario en el cajetín de la ventana que despliega la **función prompt(<msj>)**.

prompt() es una función de JavaScript en desuso, pero muy sencilla de utilizar.

Sentencia if/else-if






```
<!DOCTYPE html>
<html><head>
<meta charset="UTF-8">
</head><body>
<h3> Sentencia if/else </h3>

<script type="text/javascript">

   // Prompt pide un dato con un desplegable
var numero = prompt("Introduzca un número");

   // isNaN(..) determina si es un número
if (isNaN(numero)) {
  document.write(numero + ' no es un número');
}
else if (numero <= 0.5) {
  document.write(numero + ' es MENOR que 0,5');
}
else
  document.write(numero + ' es MAYOR que 0,5');
}
</script>
</body>
</html>
```

isNaN(numero) determina (devuelve true) si numero es un literal de número incorrecto, indicandolo con un mensaje.

Las **sentencias if/else** pueden encadenarse así, para comprobar múltiples condiciones en cascada (de las cuales solo se ejecutará una), tal y como se hace en el ejemplo de la función sonido.

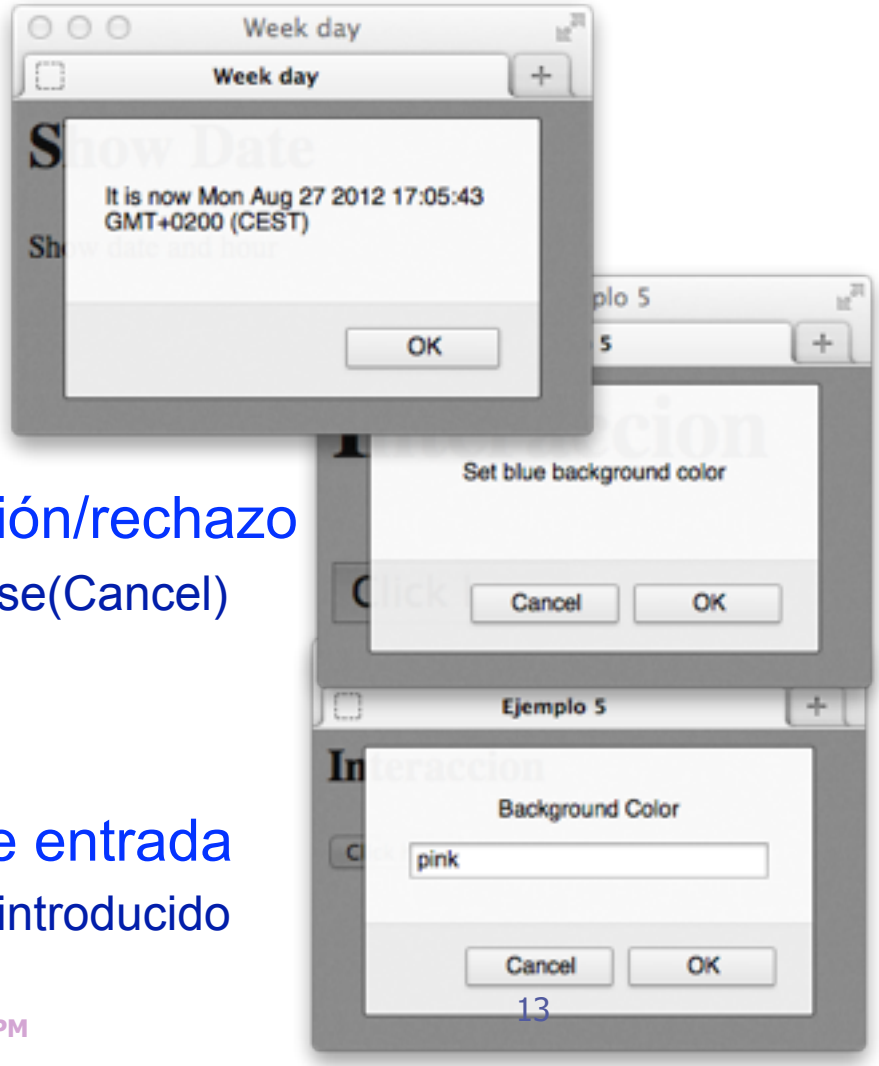
Funciones alert(), confirm() y prompt()

- 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"



JavaScript: el tipo string

El tipo string

- Los literales de string se utilizan para representar textos escritos
  - Puede representar lenguas diferentes porque utiliza el código UNICODE
    - Lenguas y símbolos soportados en UNICODE: http://www.unicode.org/charts/
- Literal de string: textos delimitados por **comillas** o **apóstrofes**
  - **"hola, que tal", 'hola, que tal', 'Γεια σου, ίσως' o '嗨，你好吗'**
    - en la línea anterior se representa "hola, que tal" en castellano, griego y chino
  - String vacío: "" o ''
  - **"texto 'entrecomillado' "**
    - comillas y apóstrofes se pueden anidar: **'entrecomillado'** forma parte del texto
- Operador de concatenación de strings: **+**
  - **"Hola" + " " + "Pepe"** => **"Hola Pepe"**

Alfabeto, código y codificación

- **Juego de caracteres** o **alfabeto**

  - **Conjunto de símbolos** normalizados para representar una lengua

- **Código de caracteres**
  - Conjunto de **puntos de código** dados a los símbolos de un alfabeto, p.e.
    - **ASCII:** alfabeto inglés codificado en 7 bits (128 caracteres y 95 imprimibles)
    - **ISO-8859-1, 2, .., 15:** Alfabetos de Europa occidental codificados en 8 bits
    - **UNICODE:** código internacionalizado que contiene casi todos los alfabetos
      - Posee 17 planos codificados en 2 bytes cada uno (1er Plano: BMP)

Char	Code
@	80
A	81
B	82
C	83

- **Codificación**
  - **representación binaria** de un código de caracteres

    - **ASCII e ISO Latin-x**: el valor del **punto del código** coincide con la representación binaria
    - **UNICODE UTF-8:** Codificación binaria en 1, 2, 3 o 4 bytes eficiente con lenguas latinas
    - **UNICODE UTF-16:** Codificación del plano BMP en 2 bytes y de otros planos en 4 bytes
    - **UNICODE UTF-32:** Codificación de todos los planos en 4 bytes

Ejemplos de teclados

Entrada y Salida de Caracteres

- JavaScript utiliza el plano básico multilingüe (BMP) de UNICODE
  - Caracteres codificados en 2 octetos (16 bits): **65536 combinaciones**
- El **texto se introduce** en el ordenador **tecleando en el teclado**
  - Los **teclados** suelen incluir solo las teclas del **alfabeto(s) de un país**
    - Tecleando solo es posible introducir un **conjunto muy pequeño de símbolos**
  - Caracteres **no incluidos en el teclado** pueden añadirse por **"corta y pega**"
    - por ejemplo, de strings traducidos con Google translate: https://translate.google.es
  - O también pueden añadirse con los **códigos escapados**
    - Permiten introducir caracteres no existentes en el teclado con códigos especiales
- **Pantalla**: es gráfica y **puede mostrar cualquier carácter**

Códigos escapados

- Definen caracteres UNICODE
  - Comienzan con barra inversa: **\…**
    - La definición solo incluye **caracteres ASCII**
      - Se incluyen en strings como otros caracteres
- Hay tres tipos
  - Caracteres de control ASCII
  - Caracteres UNICODE
  - Caracteres ISO-8859-1

CARACTERES DE CONTROL ASCII

NUL (nulo):	**\0, \x00, \u0000**
Backspace:	**\b, \x08, \u0008**
Horizontal tab:	**\t, \x09, \u0009**
Newline:	**\n, \x0A, \u000A**
Vertical tab:	**\t, \x0B, \u000B**
Form feed:	**\f, \x0C, \u000C**
Carriage return:	**\r, \x0D, \u000D**
Comillas (dobles):	**\", \x22, \u0022**
Apóstrofe :	**\', \x27, \u0027**
Backslash:	**\\, \x5C, \u005C**

- **Caracteres UNICODE o ISO-8859-1** se definen con **punto de código**, así**:**
  - UNICODE: **\uHHHH** donde **HHHH** es el punto del código (4 dígitos hex), p.e. **\u005C**
  - ISO-8859-1: **\xHH** donde **HH** es el punto del código (2 dígitos hex), p.e. **\x5C**
- Algunos ejemplos
  - Las "Comillas dentro de **\"comillas\""** deben ir escapadas dentro del string.
  - En "Dos **\n** lineas" el retorno de línea (**\n**) separa las dos líneas.
  - En "Dos **\u000A** lineas" las líneas se separan con el código UNICODE **\u000A** equivalente a **\n.**

Ejemplos de codificación

El código ASCII solo permite codificar símbolos del alfabeto inglés, por lo que los símbolos ñ y ó no se pueden representar.

Codificación	Texto	Codificación hexadecimal
ASCII: codificación que soporta solo la lengua inglesa.	hola! cañón	68 6f 6c 61 21 0a 63 61 xx xx 6e
ISO 8859 1 (Latin 1): codificación utilizada en páginas HTML no internacionalizadas.	hola! cañón	68 6f 6c 61 21 0a 63 61 f1 f3 6e
UTF-8: codificación de UNICODE utilizada hoy en páginas HTML.	hola! cañón	68 6f 6c 61 21 0a 63 61 c3 b1 c3 b3 6e
UTF-16BE: codificación del plano BMP de UNICODE utilizada en programas JavaScript	hola! cañón	00 68 00 6f 00 6c 00 61 00 21 00 0a 00 63 00 61 00 f1 00 f3 00 6e

hola!
cañon

se codifica en formato \xHH como: \x68\x6f\x6c\x61\x21 \x0a\x63\x61\xf1\xf3\x6e

UTF-8 es una codificación de longitud variable, que permite representar todos los planos de UNICODE. Es el más eficiente para representar lenguas latinas y se utiliza mucho en páginas Web internacionalizadas.

JavaScript soporta los símbolos del plano UNICODE-BMP, que corresponden a la codificación UTF-16BE (Big Endian).

hola!
cañon

se codifica en formato \uHHHH como: \u0068\u006f\u006c \u0061\u0021\u000a\u0063\u0061\u00f1\u00f3\u006e

Caracteres ASCII (Basic Latin) en UNICODE BMP

© UNICODE: http://www.unicode.org/charts/PDF/U0000.pdf

	000	001	002	003	004	005	006	007
0	NUL 0000	DLE 0010	SP 0020	0 0030	@ 0040	P 0050	` 0060	p 0070
1	SOH 0001	DC1 0011	! 0021	1 0031	A 0041	Q 0051	a 0061	q 0071
2	STX 0002	DC2 0012	" 0022	2 0032	B 0042	R 0052	b 0062	r 0072
3	ETX 0003	DC3 0013	# 0023	3 0033	C 0043	S 0053	c 0063	s 0073
4	EOT 0004	DC4 0014	$ 0024	4 0034	D 0044	T 0054	d 0064	t 0074
5	ENQ 0005	NAK 0015	% 0025	5 0035	E 0045	U 0055	e 0065	u 0075
6	ACK 0006	SYN 0016	& 0026	6 0036	F 0046	V 0056	f 0066	v 0076
7	BEL 0007	ETB 0017	' 0027	7 0037	G 0047	W 0057	g 0067	w 0077
8	BS 0008	CAN 0018	(0028	8 0038	H 0048	X 0058	h 0068	x 0078
9	HT 0009	EM 0019	) 0029	9 0039	I 0049	Y 0059	i 0069	y 0079
A	LF 000A	SUB 001A	* 002A	: 003A	J 004A	Z 005A	j 006A	z 007A
B	VT 000B	ESC 001B	+ 002B	; 003B	K 004B	[005B	k 006B	{ 007B
C	FF 000C	FS 001C	, 002C	< 003C	L 004C	\ 005C	l 006C	| 007C
D	CR 000D	GS 001D	- 002D	= 003D	M 004D	] 005D	m 006D	} 007D
E	SO 000E	RS 001E	. 002E	> 003E	N 004E	^ 005E	n 006E	~ 007E
F	SI 000F	US 001F	/ 002F	? 003F	O 004F	_ 005F	o 006F	DEL 007F

Extensión ISO Latin1 en UNICODE BMP

¥ será "\xA5" o "\u00A5"

© UNICODE: http://www.unicode.org/charts/PDF/U0080.pdf

	008	009	00A	00B	00C	00D	00E	00F
0	XXX 0080	DCS 0090	NB SP 00A0	° 00B0	À 00C0	Đ 00D0	à 00E0	ð 00F0
1	XXX 0081	PU1 0091	¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
2	BPH 0082	PU2 0092	¢ 00A2	2 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2
3	NBH 0083	STS 0093	£ 00A3	3 00B3	Ã 00C3	Ó 00D3	ã 00E3	ó 00F3
4	IND 0084	CCH 0094	¤ 00A4	´ 00B4	Ä 00C4	Ô 00D4	ä 00E4	ô 00F4
5	NEL 0085	MW 0095	¥ 00A5	µ 00B5	Å 00C5	Õ 00D5	å 00E5	õ 00F5
6	SSA 0086	SPA 0096	¦ 00A6	¶ 00B6	Æ 00C6	Ö 00D6	æ 00E6	ö 00F6
7	ESA 0087	EPA 0097	§ 00A7	· 00B7	Ç 00C7	× 00D7	ç 00E7	÷ 00F7
8	HTS 0088	SOS 0098	¨ 00A8	¸ 00B8	È 00C8	Ø 00D8	è 00E8	ø 00F8
9	HTJ 0089	XXX 0099	© 00A9	1 00B9	É 00C9	Ù 00D9	é 00E9	ù 00F9
A	VTS 008A	SCI 009A	ª 00AA	º 00BA	Ê 00CA	Ú 00DA	ê 00EA	ú 00FA
B	PLD 008B	CSI 009B	« 00AB	» 00BB	Ë 00CB	Û 00DB	ë 00EB	û 00FB
C	PLU 008C	ST 009C	¬ 00AC	¼ 00BC	Ì 00CC	Ü 00DC	ì 00EC	ü 00FC
D	RI 008D	OSC 009D	SHY 00AD	½ 00BD	Í 00CD	Ý 00DD	í 00ED	ý 00FD
E	SS2 008E	PM 009E	® 00AE	¾ 00BE	Î 00CE	Þ 00DE	î 00EE	þ 00FE
F	SS3 008F	APC 009F	¯ 00AF	¿ 00BF	Ï 00CF	ß 00DF	ï 00EF	ÿ 00FF

Radicales Kangxi de UNICODE BMP

© UNICODE: http://www.unicode.org/charts/PDF/U2F00.pdf

鬲 será "\u2FC0"

	2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD
0	一 2F00	凵 2F10	士 2F20	己 2F30	支 2F40	比 2F50	瓜 2F60	示 2F70	聿 2F80	衣 2F90	辰 2FA0	革 2FB0	鬲 2FC0	鼻 2FD0
1	丨 2F01	刀 2F11	夕 2F21	巾 2F31	攴 2F41	毛 2F51	瓦 2F61	内 2F71	肉 2F81	襾 2F91	辵 2FA1	韋 2FB1	鬼 2FC1	齊 2FD1
2	丶 2F02	力 2F12	夂 2F22	干 2F32	文 2F42	氏 2F52	甘 2F62	禾 2F72	臣 2F82	見 2F92	邑 2FA2	韭 2FB2	魚 2FC2	齒 2FD2
3	丿 2F03	勹 2F13	夕 2F23	幺 2F33	斗 2F43	气 2F53	生 2F63	穴 2F73	自 2F83	角 2F93	酉 2FA3	音 2FB3	鳥 2FC3	龍 2FD3
4	乙 2F04	匕 2F14	大 2F24	广 2F34	斤 2F44	水 2F54	用 2F64	立 2F74	至 2F84	言 2F94	釆 2FA4	頁 2FB4	鹵 2FC4	龜 2FD4
5	亅 2F05	匚 2F15	女 2F25	廴 2F35	方 2F45	火 2F55	田 2F65	竹 2F75	臼 2F85	谷 2F95	里 2FA5	風 2FB5	鹿 2FC5	龠 2FD5

Métodos de String

- Algunos métodos y elementos útiles de String
  - Más info: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String
- Un string es un array de caracteres
  - un índice entre **0 y número_de_caracteres-1** referencia cada carácter
- Acceso como array: **'ciudad'[2] => 'u'**
- Propiedad: **'ciudad'.length => 6**
  - La propiedad **length** contiene el número de caracteres del string
- Método: **'ciudad'.substring(2,5) => 'uda'**
  - devuelve el substring comprendido entre ambos índices
- Método: **'ciudad'.charCodeAt(2) => 117**
  - devuelve el número (decimal) con código UNICODE del tercer carácter
- Método: **String.fromCharCode(117) => 'u'**
  - devuelve el string con el carácter correspondiente al código numérico (decimal)

Ejemplos de strings

La Ñ existe en el código ISO-8859-1 y su código numérico en hexadecimal es d1, por lo que se puede incluir en un string tecleando (1ª) o como \xd1 (2ª).

La á existe también en el código ISO-8859-1 y la introducimos tecleando el acento y luego la a (1ª) o con el código numérico en hexadecimal \xe1 (2ª).

La barra inclinada (backslash) debe escaparse (\\) para que se visualice.

EL Euro no existe en ISO-8859-1 porque este código se creo antes de existir el Euro. Unicode se actualizó al aparecer el Euro añadiendo el símbolo € con el código \u20ac.

EL Yen Japonés si existe en ISO-8859-1: código hexadecimal \xa5.

Un string es un array (matriz) de caracteres, numerados de 1 a n-1 (último-1). **"Ciudad"[1]** indexa el segundo carácter del string, el primero será "**Ciudad"[0]**.

Al invocar el método **charCodeAt(1)** con el operador "**.**" sobre el string "**Ciudad**" nos devuelve el valor numérico decimal del **punto del código** del 2º carácter ("i").

String.fromCharCode(105) realiza la operación inversa, devuelve un string con el carácter cuyo valor (punto del código) se pasa como parámetro.

"**Ciudad".substring(3,5)** devuelve la subcadena entre 3 y 5: "**da**"

"**Ciudad".substring(3,5).length** devuelve la longitud de la subcadena devuelta ("da"), que tiene 2 caracteres.

JavaScript: el tipo number

Tipo number

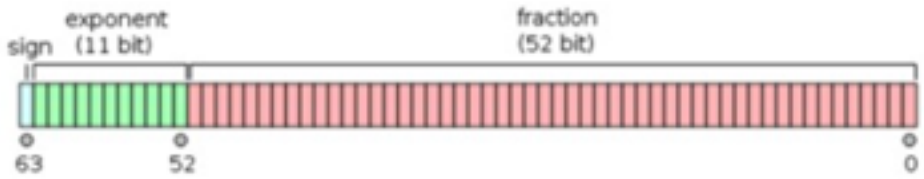
- Los números del tipo number
  - se representan con **literales de números**
- JavaScript 3 y 5 (ES3 y ES5)
  - tiene 3 formatos de literales
    - Decimales, hexadecimales y coma flotante
- **Decimales**
  - Enteros: 1, **32,** ….
  - Enteros con decimales: 1**.2**, **32.1,** ….
- **Hexadecimales**
  - Solo enteros: **0xFF**, **0X10,** …..
- **Coma flotante** (decimal)
  - Coma flotante: **3.2e1** ($3{,}2 \times 10^{1}$)
    - sintaxis: **<mantisa>e<exponente>**

```
10 + 4      => 14     // sumar
10 - 4      => 6      // restar
10 * 4      => 40     // multiplicar
10 / 4      => 2.5    // dividir
10 % 4      => 2      // operación resto

0xA  + 4    => 14  //  A es 10 en base 16
0x10 + 4    => 20  // 10 es 16 en base 16

3e2 + 1     => 301   //  3x10^2
3e-2 + 1    => 1,03  //  3x10^-2
```

Infinity y NaN

- El tipo number posee **2 valores especiales**
  - **NaN**
  - **Infinity**
- **NaN** (Not a Number)
  - representa un **valor no numérico**
    - números complejos
    - strings no convertibles a números
- **Infinity** representa
  - El **infinito matemático**
  - **Desbordamiento**
- El tipo number utiliza el formato
  - IEEE 754 coma flotante doble precisión (64 bits)
    - Reales máximo y mínimo: **~1,797x10^308** y **5x10^-324**
    - Entero máximo: **9007199254740992**
  - Cualquier literal se convierte a este formato

```
+'xx'    =>  NaN        // no representable

+10/0    =>  Infinity   // Infinito matemático
-10/0    => -Infinity   // Infinito matemático

          // Números demasiado grandes
 5e500   =>  Infinity   //desborda
-5e500   => -Infinity   //desborda

          // Número demasiado pequeño
5e-500   => 0           // redondeo

// los decimales dan error de redondeo

0.1 + 1.2    => 1,3000000000004
```



Modulo Math

- El Modulo Math contiene
  - constantes y funciones matemáticas
- **Constantes**
  - Números: **E**, **PI**, **SQRT2**, …
- **Funciones**
  - sin(x), cos(x), tan(x), asin(x), ….
  - log(x), exp(x), pow(x, y), sqrt(x), ….
  - abs(x), ceil(x), floor(x), round(x), ….
  - min(x,y,z,..), max (x,y,z,..), …
  - random()

```
Math.PI  => 3.141592653589793
Math.E   => 2.718281828459045

        // numero aleatorio entre 0 y 1
Math.random()    => 0.7890234

Math.pow(3,2)     => 9  // 3 al cuadrado
Math.sqrt(9)      => 3  // raíz cuadrada de 3

Math.min(2,1,9,3)   => 1  // número mínimo
Math.max(2,1,9,3)   => 9  // número máximo

Math.floor(3.2)       => 3
Math.ceil(3.2)        => 4
Math.round(3.2)       => 3

Math.sin(1)      => 0.8414709848078965
Math.asin(0.8414709848078965)    => 1
```

- Mas info:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math

Métodos de Number

- Algunos métodos útiles de Number son
  - **toFixed(n)** devuelve string equiv. a número
    - redondeando a n decimales
  - **toPrecision(n)** devuelve string equiv. a número
    - redondeando número a n dígitos
  - **toExponential(n)** devuelve string eq. a número
    - redondeando mantisa a n decimales
  - **toString([base])** convierte un número a
    - string con el número en la base indicada
      - [base] es opcional, por defecto base 10

```
(1).toFixed(2)          => "1.00"
(1).toPrecision(2)      => "1.0"

1.toFixed(2)            => Error

Math.PI.toFixed(4)      => "3.1416"
Math.E.toFixed(2)       => "2.72"

(1).toExponential(2)    => "1.00e+0"

(31).toString(2)        => "11111"
(31).toString(10)       => "31"
(31).toString(16)       => "1f"

(10.75).toString(2)     => "1010.11"
(10.75).toString(16)    => "a.c"
```

- Los métodos se invocan con el operador punto: "."
  - **OJO!** El número debe estar entre paréntesis para invocar el método, sino da error

- En este enlace se pueden ver otros métodos de Number:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number

parseInt(..) y parseFloat(..)

- **parseInt(string, [base])**: función predefinida de JS que convierte **string** a **number**
  - **string** se interpreta como un entero en la **base** indicada (por defecto base 10)

- **parseFloat(string)**: función predefinida de JS que convierte **string** a **number**
  - **string** se interpreta coma un número en coma flotante

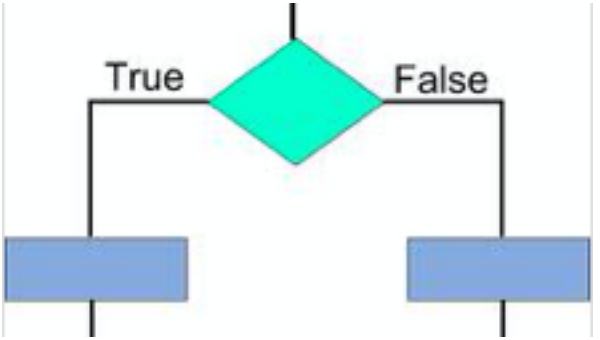
- **ParseInt(..)** o **parseFloat(..)**: realizan una conversión similar a la conversión automática
  - **OJO!** Convierten a número si un prefijo no vacío del string representa un número

```
parseInt('60')          => 60
parseInt('60.45')       => 60
parseInt('0060')        => 60
parseInt('xx')          => NaN

parseFloat("1e2")       => 100
parseFloat("1.3e2")     => 130
parseFloat("xx1e2")     => NaN
```

```
parseInt("1010")     => 1010
parseInt("1010",2)   => 10

parseInt("12",8)     => 10
parseInt("10",10)    => 10
parseInt("a",16)     => 10

parseInt("01xx")      => 1
parseInt("01+4")      => 1
```

JavaScript: Booleanos

Tipo booleano

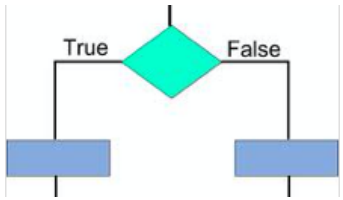


- El **tipo boolean** (booleano) solo tiene 2 valores
  - **true**: verdadero
  - **false**: falso
- Los valores booleanos se utilizan para **tomar decisiones**
  - En sentencias condicionales: **If/else**, **bucles**, etc.
- La regla de conversión de otros tipos a booleano es
  - **0, -0, NaN, null, undefined, "", ''** son equivalentes a **false**
  - **resto de valores** son equivalentes a **true**

```
!4          => false
!"4"        => false
!null       => true
!0          => true

!!""        => false
!!4         => true
```

- La **negación** convierte un valor en el **valor booleano equivalente negado**
  - La doble negación convierte un valor en su equivalente booleano: **!!<v>**

Operadores de identidad e igualdad



- Identidad o igualdad estricta: **===**
  - determina si **2 valores** son **exactamente los mismos**
    - Es igualdad semántica solo en: **number**, **boolean**, **strings** y **undefined**
      - **OJO!** En objetos es identidad de referencias (punteros)
  - La identidad determina igualdad de tipo y de valor
- Desigualdad estricta: **!==**
  - negación de la igualdad estricta
- Igualdad y desigualdad débil: **==** y **!=**
  - **OJO!** No debe utilizarse
    - Realiza conversiones impredecibles
- Mas info:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Sameness

Tipos básicos: identidad

```
0 === 0          => true
0 === 0.0        => true
0 === 0.00       => true

0 === 1          => false
0 === false      => false

'2' === "2"      => true
'2' === "02"     => false

" === ""         => true
" === " "        => false
```

Operadores de comparación

- JavaScript tiene 4 operadores de comparación
  - Menor: **<**
  - Menor o igual: **<=**
  - Mayor: **>**
  - Mayor o igual: **>=**
- Utilizar comparaciones solo con números (number)
  - donde es una relación de orden bien definida

```
1.2 < 1.3    => true

1 <   1      => false
1 <= 1       => true
1 >   1      => false
1 >= 1       => true

false < true    => true

"a" < "b"       => true
"a" < "aa"      => true
```

- No se recomienda utilizar con otros tipos: **string**, **boolean**, **object**, ..
  - Las relación de orden en estos tipos existe, pero es muy poco intuitiva
    - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Comparison_Operators

Operadores y (&&) y o (||)

- Operador lógico y (and): **<v1> && <v2>**
  - devuelve <v1> o <v2> **sin modificarlos**
    - **<v1> && <v2>**
      - devuelve **<v1>** -> si **<v1>** es equivalente a **false**
      - devuelve **<v2>** -> en caso contrario

```
true  && true    => true
false && true    => false
true  && false   => false
false && false   => false


0 && true          => 0
1 && "5"           => "5"
```

- Operador lógico o (or): **<v1> || <v2>**
  - devuelve <v1> o <v2> **sin modificarlos**
    - **<v1> || <v2>**
      - devuelve **<v1>** -> si **<v1>** es equivalente a **true**
      - devuelve **<v2>** -> en caso contrario

```
true  || true    => true
false || true    => true
true  || false   => true
false || false   => false

undefined || 1       => 1
13 || 1              => 13
```

Operador de asignación condicional

- El operador de **asignación condicional**
  - devuelve un valor en función de una condición lógica (siempre entre paréntesis)
    - Es una **versión funcional** del operador **if/else**
- Sintaxis: **(condición) ? <v1> : <v2>**
  - devuelve **<v1>** -> si **condición** es equivalente a **true**
  - devuelve **<v2>** -> en caso contrario

```
(true) ? 1 : 7          => 1
(false)? 1 : 7          => 7
```

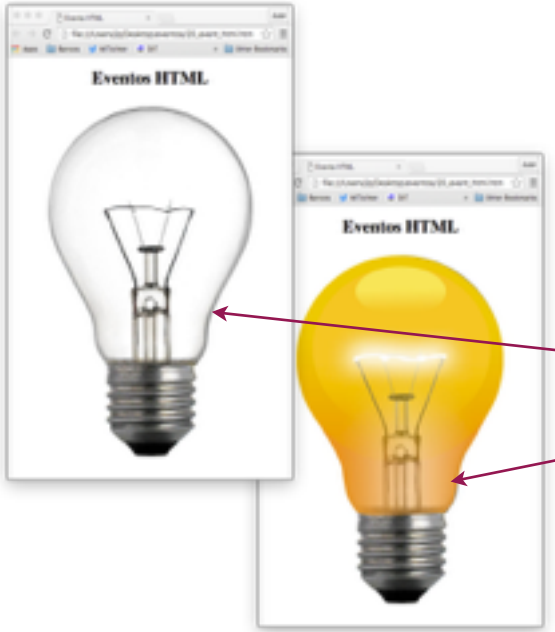
```
(7)  ? 1 : 7            => 1
("") ? 1 : 7            => 7
```

- Esta sentencia permite definir parámetros por defecto en funciones con la asignación
  - **param = (x!==undefined) ? x : <parámetro_por_defecto>;**
    - También se define a veces como "**x || <parámetro_por_defecto>**", pero aplicaría también si **x** es **"", 1, null, ..**

```
function comer (persona, comida) {
    persona  = (persona !== undefined) ? persona : 'Alguién';
    comida   = (comida  !== undefined) ? comida  : 'algo';
    return (persona + " come " + comida);
};

comer('José');      =>  'José come algo'
comer();            =>  'Alguien come algo'
```

JavaScript:
DOM, Eventos e interacción con el usuario

Eventos en HTML

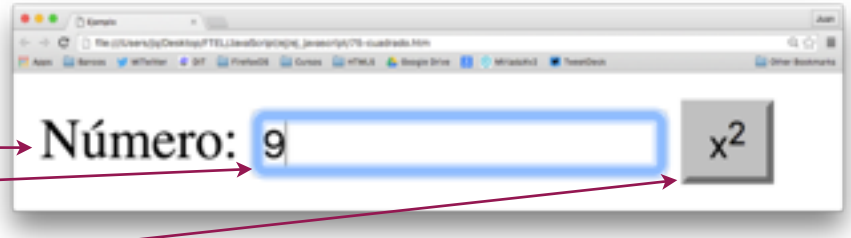



```
<!DOCTYPE html>
<html><head><title>Evento HTML</title>
<style> body{text-align:center;} </style>
</head><body>
  <h1>Eventos HTML</h1>

  <img src="lamp_off.jpg"
       ondblclick="this.src='lamp_off.jpg';"
       onclick="this.src='lamp_on.jpg';"
       />
</body>
</html>
```

- HTML permite definir **eventos** de interacción con el usuario
  - Los **eventos** se definen con **atributos con nombres especiales** de elementos HTML
    - **onclick** (hacer clic), **ondbclick** (hacer doble clic), **onload** (página cargada), …
      - http://librosweb.es/libro/javascript/capitulo_6/modelo_basico_de_eventos_2.html
- El **valor asignado** al atributo es **código JavaScript** (string) ejecutado al ocurrir el evento
  - **this** referencia el **objeto DOM** del elemento HTML asociado al evento
    - **onclick="this.src ='scare.png'"** asigna al **atributo src,** de la **imagen,** el URL al icono **scare.png**
      - **this.src** se refiere a la **propiedad** asociada al **atributo src** del objeto DOM de la **imagen**
    - **onclick="this.src ='wait.png'"** asigna al **atributo src,** de la **imagen,** el URL al icono **wait.png**
- El ejemplo asocia 2 **eventos** a la imagen (elemento **<img .... >**)
  - **onclick="this.src ='scare.png'"** muestra el icono **scare.png** al hacer **clic** en la **imagen**
  - **ondbclick="this.src ='wait.png'"** muestra el icono **clic.png** al hacer **doble clic** en la **imagen**

Calculadora

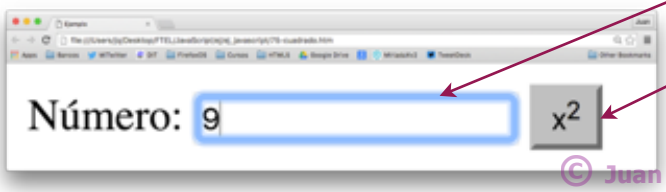
- Ejemplo de calculadora muy sencilla
  - Con los elementos básicos de interacción
    - **cajetines** para **teclear texto**
      - **<input type="text" ….. >**
    - **botones** para interactuar con el programa
      - **<button>**..nombre..**</button>**
- Comportamiento de la **calculadora**
  - Al **pulsar el botón** se calcula
    - el **cuadrado** del **número tecleado** en el cajetín
  - Al **seleccionar** el cajetín
    - se **borra** el contenido existente
- La calculadora tiene 3 elementos HTML
  - Un texto indicativo
  - El cajetín donde teclear
  - El botón con el que calcular el cuadrado



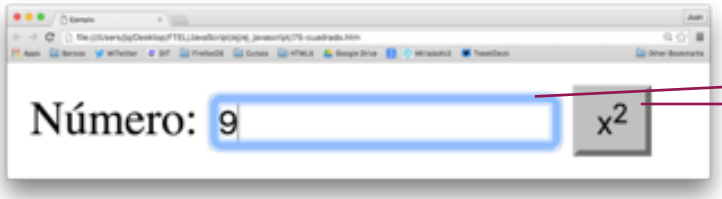
Calculadora: DOM

- La interacción con el usuario se realiza con:
  - **Botón** donde clicar: **<button onclick="cuadrado()"> x²</button>**
    - Este elemento muestra el texto HTML **"x²"** donde la marca **<sup>** define un exponente o superíndice
  - **Cajetín** de entrada de texto: **<input type="text" id="n1" onclick="vaciar()">**
    - El objeto DOM obtenido con **var num = document.getElementById("n1")** permite interacción con el cajetín
- Las **propiedades** del **objeto num** dan acceso a los atributos HTML y a otros, por ejemplo
  - **num.type** contiene **"text"**
  - **num.id** contiene **"n1"**
  - **num.value** contiene **"9"**
    - es el contenido tecleado en el cajetín
  - **num.innerHTML** contiene: **""**
    - <input> no tiene HTML interno
  - **num.outerHTML** contiene:
    - **"<input type="text" id="n1" onclick="vaciar()">"**
      - Es el HTML completo del elemento
- Modificar la página visualizada, por ej.
  - Asignar **num.value = 5**;
    - Mostrará 5 en el cajetín
  - Asignar **num.outerHTML = Hola **
    - Mostrará un párrafo con“Hola" en vez del cajetín

```
<!DOCTYPE html><html><head>
<title>Ejemplo</title><meta charset="utf-8">

<script type="text/javascript">

function vaciar () {
  document.getElementById("n1").value = "";
}

function cuadrado() {
  var num = document.getElementById("n1");
  num.value = num.value * num.value;
}
</script>
</head><body>
  Número:
  <input type="text" id="n1" onclick="vaciar()">

  <button onclick="cuadrado()">
    x<sup>2</sup>
  </button>
</body></html>
```



Atención a eventos

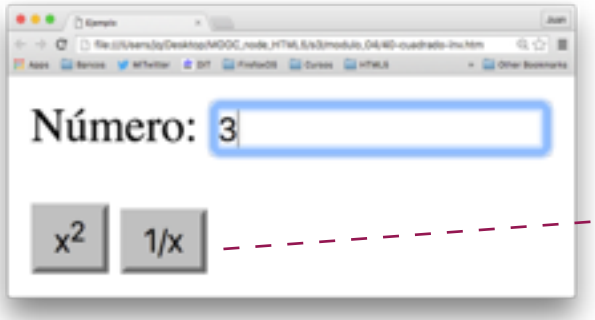



```
<!DOCTYPE html><html><head>
<title>Ejemplo</title><meta charset="utf-8">
<script type="text/javascript">

function vaciar () {
  document.getElementById("n1").value = "";
}
function cuadrado() {
  var num = document.getElementById("n1");
  num.value = num.value * num.value;
}
</script>
</head><body>
  Número:
  <input type="text" id="n1" onclick="vaciar()">

  <button onclick="cuadrado()"> x<sup>2</sup> </button>
</body></html>
```

- Los **objetos DOM** tienen **eventos asociados** atendidos por **funciones, por ej.**
  - El atributo **onClick="vaciar()"** de **<input type="text" id="n1" onClick="vaciar()">**
    - Asocia al evento **clicar en el cajetín** la función **vaciar()** que vacía el cajetín al ocurrir este
  - El atributo **onClick="cuadrado()"** de **<button onClick="cuadrado()"> ….**
    - Asocia al evento **clicar en <button..>** la función **cuadrado()** que muestra en el cajetín el cuadrado del número contenido en este antes de clicar
- Existen muchos más eventos de interacción con el usuario
  - Se puede encontrar más información sobre los más básicos en:
    - http://librosweb.es/libro/javascript/capitulo_6/modelo_basico_de_eventos_2.html

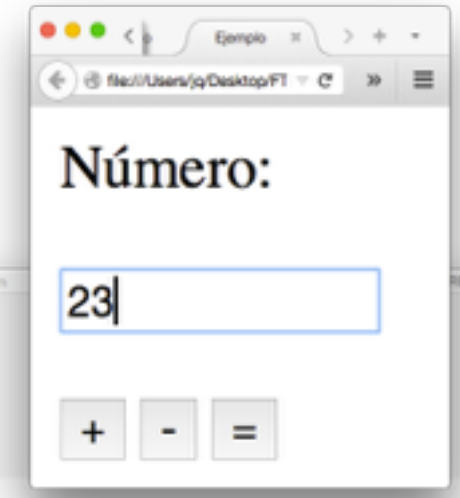
Calculadora: añadir Botón 1/x

- En este ejemplo añadimos
  - Un botón más a la calculadora
    - El dot calcula el inverso (1/x) del cajetín
- Añadir un nuevo botón es sencillo
  - Se añade un nuevo **botón** HTML
    - con el texto: **1/x**
    - con atributo : onclick="inverso()"
      - asocia inverso() a clic en él
  - Se añade la función: **inverso()**
    - Calcula el inverso del número del cajetín
  - Se añade una marca de párrafo en HTML
    - Para separar el cajetín de los botones




```
<!DOCTYPE html><html>
<head>
<title>Ejemplo</title>
<meta charset="utf-8">

<script type="text/javascript">

function vaciar () {
  document.getElementById("n1").value = "";
}

function cuadrado() {
  var num = document.getElementById("n1");
  num.value = num.value * num.value;
}

function inverso() {
  var num = document.getElementById("n1");
  num.value = 1/num.value;
}
</script>
</head>
<body>

  Número:
  <input type="text" id="n1" onclick="vaciar()">
  <p>
  <button onclick="cuadrado()">x<sup>2</sup></button>
  <button onclick="inverso()"> 1/x </button>
</body>
</html>
```

Calculadora extendida




```
<html>
<head>
<title>Ejemplo</title>
<meta charset="utf-8">

<script type="text/javascript">
  var num, acc = 0, op = "";

  function mas()   { acc = num.value;  op = "+"; }
  function menos() { acc = num.value;  op = "-"; }

  function calcular() {
    if (op === "+") {num.value = (+acc + +num.value)}
    if (op === "-") {num.value = (+acc - +num.value)}
  }

  function inic ()   {num = document.getElementById("num");}
  function vaciar () {num.value = "";}
</script>
</head>
<body onLoad="inic()">
  Número:<p>
  <input type="text" size="12" id="num" onClick="vaciar()"><p>
  <button onClick="mas()">+</button>
  <button onClick="menos()">-</button>
  <button onClick="calcular()">=</button>
</body>
</html>
```

- **Calculadora extendida**
  - Suma o resta 2 números, así:
    - Se teclea el primer número
    - Se pulsa **+** o **-**
    - Se teclea el segundo número
    - se pulsa **=**
- Variables **num**, **acc** y **op**
  - Las **variables num**, **acc** y **op** se definen al principio del script para su uso en las funciones
    - Las variables globales son visibles en las funciones
- Evento **onLoad** y **función inic()**
  - El **evento onLoad** de body indica que se ha construido ya el árbol DOM
    - Este evento invoca la **función inic()** que carga en la **variable num** el objeto DOM de acceso al cajetín, que utilizan el resto de las funciones
- Variables **acc** y **op**
  - Las **variables acc** y **op** guardan el número que está en el cajetín y la operación pulsada (**+** o **-**), cuando se pulsa el botón **+** (invoca **función mas()**) o el **-** (invoca **función menos()**).
- Botón **=**
  - El botón **=** invoca la **función calcular()**, que suma o resta el número que está en el cajetín con el número guardado en **acc**, dependiendo de la operación (**+** o **-**) que se ha guardado en la variable op

Final del tema