

CURSO DESARROLLO WEB FRONT-END

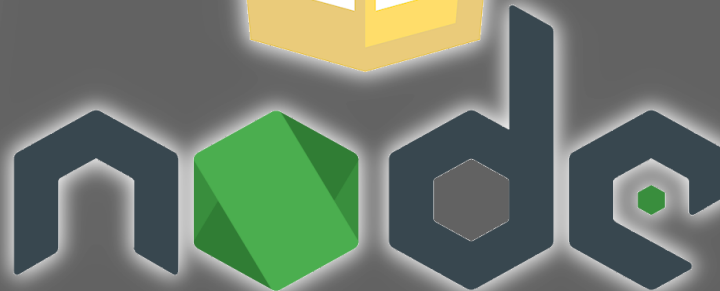
HTML



CSS



JS



Germán Caballero Rodríguez <germanux@gmail.com>

Generation
SPAIN

Cámaras
Fundación INCYDE



Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Programación básica con

The word "JavaScript" is written in a large, light blue, sans-serif font, slanted slightly to the right. It is set against a dark blue background with a complex, glowing circuit board pattern. A bright light source on the left creates a lens flare effect across the text.

JavaScript

INDICE

1. Variables
2. Tipos primitivos
3. Tipos referencia
4. Operadores
5. Funciones
6. Sentencias de control

Variables

- Se declaran mediante la palabra reservada **var**.
- En los identificadores no se deben emplear \$ ni _
- No hay obligación de inicializar las variables.
- Ni de declararla, si no se declaran (se indica var), se crea una variable global.
- Las variables pueden apuntar a cualquier tipo de objeto.

Tipos primitivos

- Los tipos de datos primitivos existentes son
 - undefined
 - null
 - boolean
 - numerico
 - string

Tipos primitivos

- Tipo **undefined**

- Variables no definidas o variables no declaradas.

```
var variableDeclarada;  
  
console.log(typeof variableNoDeclarada);  
console.log(typeof variableDeclarada);
```

- Los dos casos dan el mismo resultado **undefined**

Tipos primitivos

- Tipo **null**

- Variable similar a **undefined**, **javascript** no es capaz de diferenciarlas.

```
Var objetoQueTodaviaNoExiste = null;
```

- Representa la referencia a un objeto que todavía no existe.

Tipos primitivos

- Tipo **boolean**
 - Almacenan **true** o **false**, que son palabras reservadas, ni literales ni números.
 - La conversión de un **número** a **boolean** produce el siguiente cambio
 - 0 : false
 - cualquier otro : true

Tipos primitivos

- Tipo **numérico**
 - Almacena los siguientes tipos de valores
 - **entero** (var entero = 10)
 - **decimal** (var decimal = 3.26)
 - **octal** (var octal = 033)
 - **hexadecimal** (var hexadecimal = 0xd3)
 - **infinity** : divisiones por cero
 - **nan** : operaciones con variables no numéricas

Tipos primitivos

- Existe un método para comprobar si a lo que apunta una variable es numérico.
 - **isNaN(variable_u_operacion_a_comprobar)**, que devuelve **true** si **NO** es numérico.

Tipos primitivos

- Tipo **string**
 - Cadenas de texto rodeadas por comillas simples o dobles.
 - El primer carácter se almacena en la posición cero.
 - No permite la inserción de ciertos caracteres como el salto de línea, de forma directa, tiene que ser con secuencias de escape.

Tipos primitivos

- Tipo **string**
 - Se definen los caracteres especiales como:
 - `\n` salto de línea
 - `\t` tabulador
 - `\'` comillas simples
 - `\''` comillas dobles
 - `\\` barra

Tipos referencia

- Semejante a clases en programación orientada a objetos.
- Por un lado existe el objeto en memoria, y por otro la/las variable/s que lo hacen referencia.
- Creados a partir de la palabra reservada **new**.
- Los paréntesis no son obligatorios sino se especifican parámetros (pero si son muy recomendables!!)

Tipos referencia

- Tipo **Object**
 - Base para los objetos
 - Puede encapsular tipos primitivos

```
var valor1 = new object(72);
```

```
var valor2 = new object(true);
```

```
var valor3 = new object("esto es un texto");
```

Tipos referencia

- Tipo **boolean**

- Se pueden crear primitivos u objetos.
- Nunca se utilizan objetos booleanos, ya que cualquier objeto, sea booleano o no, se considera **true**, sea cual sea su valor interno, lo que puede llegar a confundir el uso de objetos **boolean**.

```
if(primitivoatrue && objetoafalse){  
    console.log("Esta linea no deberia ejecutarse, pero se  
ejecuta");  
}
```

- El resultado es **true**, aunque debería ser **false**.

Tipos referencia

- Tipo **number**
 - Con el método **valueOf()** obtenemos el primitivo

```
miobjetonumerico.valueOf();
```

- Con el método **toFixed**, se ajustan los decimales.

```
var numerocondecimales = new number(2.3425234);  
numerocondecimales.toFixed(); //2  
numerocondecimales.toFixed(4); //2.3425
```


Tipos referencia

- Tipo **number**
 - Cuando se redondea un decimal que acabe en 5, se puede incrementar o decrementar aleatoriamente el redondeo.

Tipos referencia

- Tipo **string**
 - Ofrece gran cantidad de métodos y utilidades, como
 - **.length** : numero de caracteres
 - **.concat()** : concatenación de cadenas (equivalente al operador +)
 - **.toupper()**: paso a mayúsculas
 - **.tolower()**: paso a minúsculas
 - **.charAt(posicion)**: devuelve el carácter en esa posición

Tipos referencia

- Tipo **string**
 - **.indexof(letra)**: primera posición del carácter indicado, o -1 sin coincidencias
 - **.lastindexof(letra)**: análogo a la anterior
 - **.substring(posicion)**: trozo de texto desde la posición indicada hasta el final. En caso de que se indique posición negativa, devuelve el texto completo.

Tipos referencia

- Tipo **string**
 - **.substring(inicio,fin)** : devuelve el texto entre la posición de inicio y la inmediatamente anterior a la del final. En caso de invertir los valores, javascript reordena los valores y usa el menor como primero y el mayor como segundo
 - **.split(separador)**: transforma un texto en un array usando como separador el texto indicado. Si no se indica nada, se separan por letras.

Tipos referencia

- **Tipo array**

- Constructor con número de elementos

```
var miarray = new array(5);
```

- Los elementos del array pueden ser de distintos tipos a la vez.

- Se puede inicializar en la declaración del **array**

```
var miarray = new array("esto es un ejemplo",true,7,5.33);
```

- O a posteriori.

```
var miArray2 = new Array();  
miArray2[0]="Esto es un ejemplo";  
miArray2[1]=true;
```

Tipos referencia

- Tipo **array**
 - Los elementos del **array** comienzan siempre en la posición cero.
 - El tamaño del **array** aumenta de forma dinámica.
 - El tipo **array** ofrece una serie de métodos y atributos
 - **.length**: número de elementos de un array
 - **.concat()**: concatenación de varios arrays
 - **.join(separador)**: contraria a split, transforma el array en una cadena de texto con el separador asignado.

Tipos referencia

- Tipo **array**
 - **.pop()** elimina el último elemento y lo devuelve
 - **.push(item)** introduce un elemento al final del array
 - **.shift()** elimina el primer elemento y lo devuelve
 - **.unshift()** añade un elemento al principio del array.
se pueden agregar tantos elementos como se quiera de una vez.
 - **.reverse()** invierte el orden de los elementos

Tipos referencia

- Tipo **date**

- Representación y manipulación de fechas.
- Es el número de milisegundos desde 1970
- Constructor con milisegundos

```
var mifecha = new date(0); /1970
```

- Otros constructores mas intuitivos

```
var mifecha = new date(año,mes,dia);  
var mifecha = new date(año,mes,dia,hora,minuto,segundo);
```

- **Los meses se cuentan desde cero!!!!!!**

- 0 -> enero, 11 -> diciembre

Tipos referencia

- Tipo **date**

- Algunos de los métodos disponibles

- **getTime()**-> fecha en ms desde 1 de enero de 1970
 - **getMonth()**-> número de mes desde 0 a 11
 - **getFullYear()** ->año en 4 cifras
 - **getYear()**->año en 2 cifras
 - **getDate()**->día del mes
 - **getDay()** -> día de la semana de 0(Domingo) a 6(Sábado)

Tipos referencia

- Tipo **date**
 - Algunos de los métodos disponibles
 - **getHours()**->horas
 - **getMinutes()**->minutos
 - **getSeconds()**->segundos
 - **getMilliseconds()**->milisegundos
 - Posee también sus correspondientes **setters** para modificar los datos.

Operadores

- **instanceof**: identifica de que tipo es el objeto referencia
- **=**: Operador de asignación, asigna valores a variables
- **++** : incremento (puede ser pre o post)
- **--** : decremento (puede ser pre o post)
- **!** : negación.
- **&&** : AND lógico.
- **||** : OR lógico.

Operadores

- **Matemáticos**

- suma (+)
- resta (-)
- multiplicación (*)
- división (/)
- módulo (%) (el resto de la división)

Operadores

- **Relacionales**

- mayor que(>)
- menor que(<)
- mayor o igual que (>=)
- menor o igual que (<=)
- igual (==). Realiza conversiones de tipo de forma transparente.
- distinto (!=)

Operadores

- **Relacionales**

- Pueden usarse para **string**. Se compara letra a letra.
- operador **==**: Comparar dos elementos sin hacer conversiones de tipos. En caso de comparar 7 texto y 7 numérico, **==** devuelve **true**, sin embargo **===** devuelve **false**.

Funciones

- No necesitan devolver un valor, ni indicar que no lo hacen. En caso de no devolver nada explícitamente, se devuelve **undefined**.
- Se puede invocar una función sin pasar todos los parámetros pedidos, o pasando mas.
 - Cuando se usan menos, se completan con **undefined**.
 - Cuando se usan más, se ignoran.
 - En realidad siempre están disponibles dentro de una variable **arguments**

Funciones

- Se pueden definir funciones internas dentro de otras funciones.

```
function operaciones(a,b){  
    function suma(x,y){  
        return x+y;  
    }  
    return suma(a,a) + suma(b,b);  
}
```

- Se pueden definir funciones sin nombre (anónimas), para aquellas que solo se necesitan una vez.

Funciones

- Todas las funciones, tienen una variable **arguments**, que ofrece información de los parámetros que le llegan a la función.

```
function suma(a,b){  
    var result = 0;  
    console.log("Numero de parametros: " + arguments.length);  
    console.log("Numero de parametros esperados: " +  
arguments.callee.length);  
    if(arguments.length>2){  
        for(i = 0; i<arguments.length;i++){  
            result += arguments[i];  
        }  
    }else{  
        result = a+b;  
    }  
    return result;  
}
```

Sentencias de control

- Las sentencias de control de flujo, permiten al programador, controlar que partes del código se ejecutan antes estados distintos de las variables de la aplicación.
- Existen dos tipos de sentencias:
 - **Condicionales.**
 - **Iterativas.**

Sentencias de control

- Condicionales
 - Sentencia **if-else**

```
if(<expresión-booleana>) {  
    //Código a ejecutar cuando se cumpla la <expresión-booleana>  
  
} else if (<expresion-booleana-2>) {  
    //Codigo a ejecutar si no se cumple la <expresion-booleana>,  
    //pero si la <expresion-booleana-2>  
} else {  
  
    //Codigo a ejecutar cuando no se cumpla ninguna de las  
    //expresiones  
  
}
```

Sentencias de control

- Condicionales
 - Sentencia **switch**

```
switch(variable) {  
    case valor_1:  
        //Codigo a ejecutar si variable igual a valor_1           break;  
    case valor_2:  
        //Codigo a ejecutar si variable igual a valor_2  
        break;  
    case valor_n:  
        //Codigo a ejecutar si variable igual a valor_n  
        break;  
    default:  
        //Codigo si variable es distinto a los valores planteados  
        break;  
}
```

Sentencias de control

- Iterativas

- Sentencia **for**

```
for(inicializacion; condicion; actualizacion) {  
    //Sentencias a ejecutar de forma repetida  
}
```

- **inicialización**: zona de asignación de valores iniciales de las variables que controlan la repetición.
 - **condición**: es el elemento que decide si continua o se detiene la iteración.
 - **actualización**: zona de asignación del nuevo valor de las variables que controlan la repetición.

Sentencias de control

- Iterativas
 - Sentencia **for** en formato **foreach**

```
for(variable in array) {  
    //Sentencias a ejecutar de forma repetida  
}
```

- **variable:** variable a emplear dentro del bucle, que contendrá en cada iteración, un elemento del array
- **array:** colección de datos en una estructura de tipo array.

Sentencias de control

- Iterativas
 - Sentencia **for**
 - La sentencia **break** aplicado a un **for**, permite finalizar las iteraciones.
 - La sentencia **continue** aplicado a un **for**, permite saltar directamente a la siguiente iteración.

Sentencias de control

- Iterativas
 - Sentencia **while**

```
while(<condicion-booleana>) {  
    //Sentencias a ejecutar de forma repetida mientras se  
    //cumpla la <condicion-booleana>  
}
```


Sentencias de control

- Iterativas
 - Sentencia **do-while**

```
do {  
    //Sentencias a ejecutar al menos una vez, y que se  
    //repetiran mientras se cumpla la <condicion-booleana>  
} while(<condicion-booleana>);
```