PROGRAMACIÓN WEB EN ENTORNO CLIENTE UD-1

JAVASCRIPT. INTRODUCCIÓN. VARIABLES

ÍNDICE

[INTRODUCCIÓN 3](#_Toc146812565)

[JAVASCRIPT 4](#_Toc146812566)

[INTEGRACIÓN DE SCRIPTS EN EL DOCUMENTO HTML 4](#_Toc146812567)

[INSTRUCCIONES DE SALIDA 6](#_Toc146812568)

[COMENTARIOS 7](#_Toc146812569)

[VARIABLES Y ÁMBITO (SCOPE) 7](#_Toc146812570)

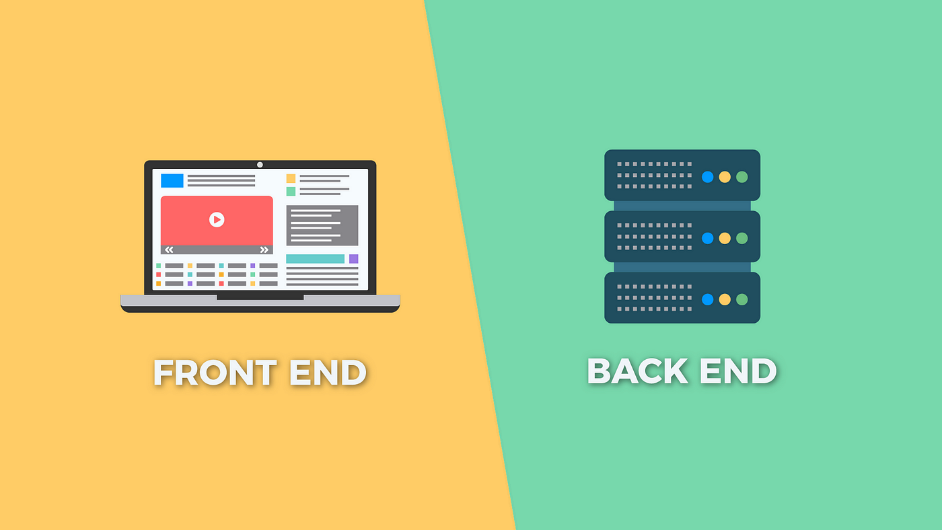
[OBJETOS 10](#_Toc146812571)

[typeof 11](#_Toc146812572)

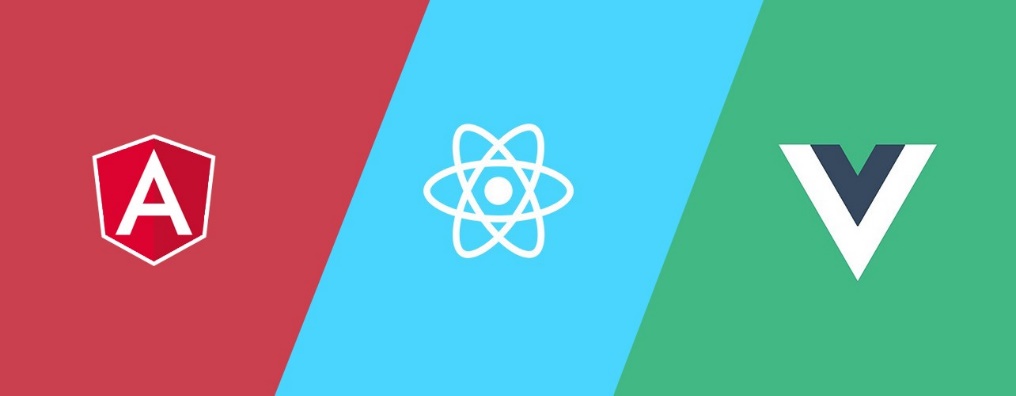
[DOM 11](#_Toc146812573)

# INTRODUCCIÓN

En el desarrollo web podemos observar dos claros campos bien diferenciados, el campo del front-end y el campo del back-end.



En el caso del primer campo hablamos de todo aquello que es visible e interactivo con el usuario, por lo que se va a enfocar principalmente en el navegador web. En el front-end contamos con tecnologías como HTML5, CSS3 o JavaScript, pero también con frameworks como Angular, React y VueJS así como de lenguaje legacy (código que actualmente se encuentra obsoleto y que ya no se continua) como por ejemplo jQuery.



Por su parte el back-end es la parte del desarrollo web que se va a encargar de la comunicación con el servidor y de todo lo que pasa por detrás y de lo que el usuario no es consciente. Las tecnologías utilizadas en este ámbito son PHP y node.JS.

La fusión de ambos campos da lugar a lo que se conoce como full-stack que es aquel desarrollador que se puede encargar tanto del front como del back-end, es por esto por lo que JS ya no es sólo un lenguaje de front sino también de back a través de node.

# JAVASCRIPT

Logotipo

Descripción generada automáticamenteJavaScript es un lenguaje de programación desde el lado del cliente que cuenta con las siguientes características principales:

* Es **débilmente tipado**, es decir, cuando declaramos variables en JS estas no necesitan ser tipadas en su declaración. Al contrario de lo que pasa con lenguajes como Java, en JS podemos tener una variable que al inicio del script es utilizada como un Number pasar más tarde a ser utilizada como un String.
* Es un lenguaje **interpretado**, es decir, no necesita de una compilación previa, sino que el navegador va a ir ejecutando sentencia a sentencia.
* Es un lenguaje **orientado a objetos** **y basado en prototipos.** JS permite la creación de objetos nuevos, pero también cuenta con otros objetos ya incluidos en su codificación previa como el objeto window. Así mismo permite la creación de objetos desde prototipos, es decir, clonar un objeto desde otro sin necesidad instanciarlos.
* Es **imperativo y estructurado**.

# INTEGRACIÓN DE SCRIPTS EN EL DOCUMENTO HTML

Para integrar un script de JS en nuestro documento HTML tenemos varias opciones, en todas ellas deberemos usar la etiqueta <script>.

* En el <head> del documento HTML. En este caso vamos a meter nuestro script dentro de la cabecera del documento.

Texto

Descripción generada automáticamente

* En el <body> del documento HTML. Al igual que el anterior, pero en el cuerpo del documento HTML.

Texto

Descripción generada automáticamente

Aunque en los dos ejemplos anteriores se separa en <head> y <body> también se puede hacer parte del script en el head y otra parte en el body, por ejemplo, podemos declarar la función en la cabecera y llamarla en el cuerpo:

Texto

Descripción generada automáticamente

* En un archivo .js externo. Esta es la mejor manera ya que al igual que pasa con las hojas de estilo CSS, el separar el documento HTML del documento JS, facilita la modularización y el mantenimiento del código.

En este caso debemos utilizar la etiqueta <script> con el atributo src y en el valor colocaremos la ruta donde se encuentra nuestro archivo .js.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# INSTRUCCIONES DE SALIDA

En todos los ejemplos anteriores va a suceder siempre lo mismo, una vez abramos nuestra web se cargará el script de JS y se irá interpretando línea a línea, en este caso tenemos una llamada a una función llamada holaMundo() que produce la ejecución de una instrucción de salida a través del método alert() del objeto window, el cual produce la aparición de una ventana emergente en nuestro navegador con el mensaje “Hola mundo”.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Otro ejemplo de instrucción de salida sería el console.log(). Por ejemplo:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

En este caso en lugar de una ventana emergente lo que va a suceder en el navegador es que se va a escribir el mensaje “Hola mundo” en su consola, la cual es accesible desde la tecla F12.

Texto

Descripción generada automáticamente

# COMENTARIOS

Al igual que ocurre con otros lenguajes de programación, se puede comentar nuestro código con teclas muy similares, podemos realizar comentarios de una línea con la doble barra (//) y comentarios multilínea con el punto-asterisco (/\*).

Texto

Descripción generada automáticamente

# VARIABLES Y ÁMBITO (SCOPE)

De forma general a la hora de nombrar variables en JS debemos tener en cuenta ciertas reglas:

* Deben ser **identificadores únicos**. No debe haber dos variables con el mismo nombre.
* Es recomendable que sus nombres sean descriptivos.
* Deben empezar por una letra, $, \_. Después de estos caracteres puede ir un número.
* Son *case sensitive*, es decir la variable nombre es diferente a la variable Nombre.
* No pueden utilizarse palabras reservadas para nombres de variables.
* Es posible declarar múltiples variables en una única sentencia:



* Se puede asignar una operación a una variable:



En JavaScript contamos con 2 tipos de variables y 1 tipo de constantes.

Texto

Descripción generada automáticamente con confianza media

* **var**. Su uso ya no está aconsejado. Tiene un scope global, es decir, se puede ver desde todo nuestro programa js, lo cual puede dar lugar a problemas a la hora de depurar nuestro código.
* **let**. Tiene un scope de bloque, es decir, sólo son visibles dentro del bloque en el que son declaradas. Entendemos como bloque toda estructura de control que comienza y termina en llaves ({}), por ejemplo, en el siguiente ejemplo, el primer let será visible en todo el programa mientras que el segundo let sólo será visible dentro del if.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

En este caso en la consola veremos el siguiente resultado:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Como se puede ver la consola nos informa de que la variable mayorEdad no está definida en el scope fuera del if, si en lugar de declarlo como let, lo hubiéramos declarado como var, tendríamos este resultado:

Pantalla de un video juego

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La variable mayorEdad es visible tanto en el if como fuera del mismo. Esto en programas más largos puede ser un problema para depurar el código ya que no se sabe en qué punto puede haber fallado una variable.

Hay que tener en cuenta que en el caso de las funciones esto no aplica, una variable de tipo var sólo sería visible dentro de la función y no fuera de ella, aunque dentro de la función tendría un alcance completo siendo visible durante toda su ejecución. Lo mismo pasaría con let.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza mediaUna captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamenteUna captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Aunque en el error vemos que nos informa que la variable edad no está definida en el scope, realmente tampoco lo está mayorEdad, la consola no nos informa de ellos porque una vez que se produce un error en la interpretación del script, se paraliza su ejecución.

* **const.** Es la forma que tenemos de declarar las constantes en JavaScript, al igual que ocurre con let, sólo tiene alcance de bloque.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza mediaPantalla de un video juego

Descripción generada automáticamente

# OBJETOS

Los objetos en JS se declaran entre llaves de apertura y cierre:

Texto

Descripción generada automáticamente

Como se puede ver los atributos son parejas clave:valor separadas por comas, los métodos por su parte se declaran como si se tratara de funciones.

Si queremos hacer una llamada al objeto deberemos hacerlo a través del nombre del objeto, eso nos permitirá acceder a sus atributos y métodos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Por ejemplo:

Pantalla azul con letras blancas

Descripción generada automáticamente con confianza media

Que produce este resultado en el navegador:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

# typeof

El operador typeof devuelve una cadena que indica el tipo del operando sin evaluarlo.

Por ejemplo, teniendo las siguientes variables:

Texto

Descripción generada automáticamente

Vamos a realizar las siguientes impresiones por consola:

Texto

Descripción generada automáticamente

Y conseguimos el resultado:

Texto

Descripción generada automáticamente

Esto nos va a ser útil para conocer el tipo de una variable y, de esa manera, actuar de una u otra manera, por ejemplo, a través de condicionales:



Captura de pantalla de un celular

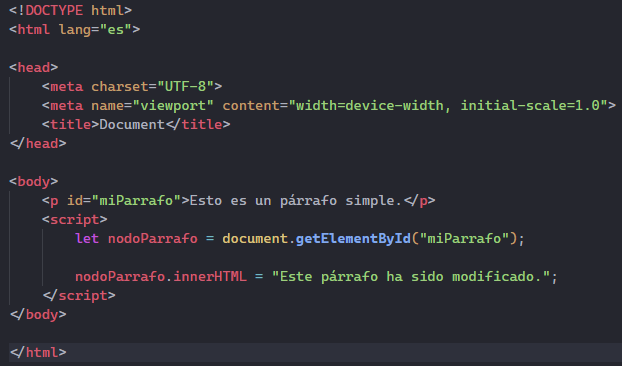
Descripción generada automáticamente

# DOM

El DOM, acrónimo de *Document Object Model,* es una API definida para representar e interactuar con cualquier documento HTML o XML. El DOM es un modelo de documento que se carga en el navegador web y que representa el documento como un árbol de nodos, en donde cada nodo representa una parte del documento (puede tratarse de un elemento, una cadena de texto o un comentario).

Con JS vamos a ser capaces de afectar directamente al DOM.

Dado un HTML vamos a crear un script que en primer lugar capture un elemento p con el ID miParrafo y, a continuación, cambie su contenido.



Para ello hemos tenido que hacer uso en primer lugar del método getElementById() del objeto document y, a continuación, a través de innerHTML cambiar su contenido.

Hay que tener en cuenta que el script de forma normal se va a cargar de abajo arriba, en este caso el elemento p se cargará antes del script, pero si invertimos el orden:

Texto

Descripción generada automáticamente

Obtendremos el siguiente resultado:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Esto se debe a que el script ha intentado capturar el elemento antes de que se creara. Para evitar esto podemos hacer uso de la palabra reservada defer que hace que el DOM se cargue en primer lugar y, a continuación, ejecute el script. Pero hay que tener en cuenta que el atributo defer sólo se aplica si el script está en un archivo externo.

Interfaz de usuario gráfica, Texto, Sitio web

Descripción generada automáticamente