

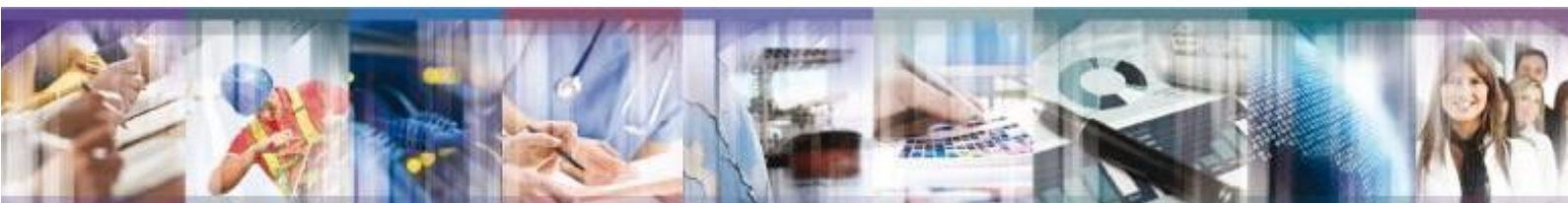


UTN.BA FACULTAD
REGIONAL
BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

**Centro de
e-Learning**

JavaScript

Programando JavaScript



www.sceu.frba.utn.edu.ar/e-learning

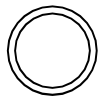
Variables y Operadores



Presentación

Cuando hablamos de JavaScript, nos referimos a este lenguaje muy utilizado en el ámbito del desarrollo que nos provee de un amplio abanico de posibilidades en cuanto a ejecución de acciones y comportamientos del lado del cliente, lo que nos permite que el navegador actúe de cierta manera que programamos de una manera mas rápida y dinámica.

Este lenguaje es muy utilizado hoy en día y actúa también com base para muchos otros. Por ende, aprovechando que ya aprendimos en la unidad pasada como enlazarlo para que pueda ser leído, es importante ahora que aprendamos como funciona, como hacemos para indicarle las acciones a realizar y alguna de las herramientas que dispone este lenguaje.



Temario:

Temario:

1. Variables
2. Funciones
3. Alcance de las variables
4. Adquiriendo algunas buenas prácticas.
5. Operadores
6. En resumen
7. Bibliografía utilizada

1. Variables

En JavaScript, definimos variables a elementos que nos sirven para guardar un valor en la computadora. Es un espacio en la memoria que le podemos asignar un nombre y que nos va a servir para poder guardar valores e ir accediendo a ellos en el momento que lo necesitemos.

Pero... ¿qué es un valor en JavaScript? Un valor es el componente más simple de JavaScript que puede ser desde un número o una letra, hasta una función y mucho más, hay muchos tipos de valores (ya en la unidad anterior vimos algunos como Strings, Numbers y Booleans), pero no nos profundizaremos ahora en ello ya que vamos a ir viéndolos unidad a unidad.

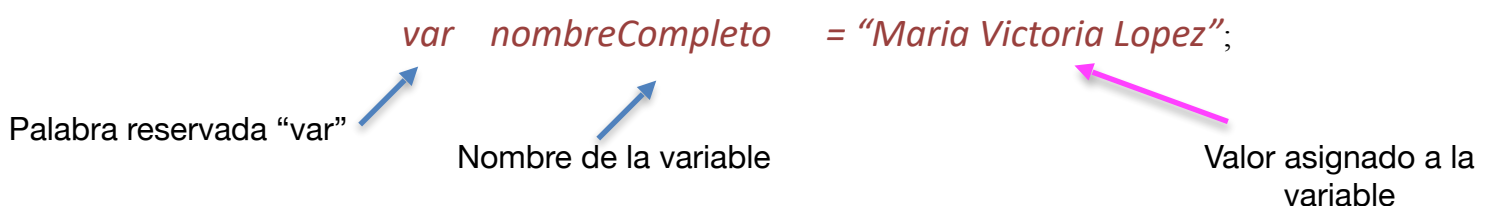
Entonces, teniendo en cuenta la definición de variable, podríamos pensarlo como un buzón donde vamos a poner cosas como una sentencia, y después le damos a la variable una dirección (en este caso un nombre) que puede usar para ir a buscar esa sentencia más tarde.

Estos nombres no deben contener espacios y pueden utilizar tanto letras mayúsculas como minúsculas; y al igual que en un buzón, nosotros podemos cambiar el contenido de esa variable, darle otros valores a lo largo del programa en caso que necesitemos.

Sintaxis

Para poder decirle a JavaScript que lo que estamos definiendo es una variable, utilizamos la palabra reservada **var** seguido por el nombre que le vamos a dar a esa variable, luego un signo igual (=) seguido por el valor asignado a esa variable; de esta forma le estamos diciendo al navegador que guarde en esa variable todos los valores que se encuentren del lado derecho del igual, de esta forma cada vez que en el código escribamos ese nombre definido, vamos a estar invocando al valor o valores guardados en dichas variables.

var nombreCompleto = "Maria Victoria Lopez";



Palabra reservada "var"

Nombre de la variable

Valor asignado a la variable

Viendo este ejemplo podemos notar que cada vez que llamemos a nombreCompleto este me va a estar trayendo el string “Maria Victoria Lopez”; en caso que en vez de definir ese valor hubiera colocado el valor 13435223, al invocar a dicha variable me hubiera traído el número que le definimos.

Otro ejemplos serian:

```
var edad = 35 ;  
var verdadero = true ;  
var dato_inicial = “ “;
```

1. Alcance de las variables

Ahora que ya logramos poder entender qué es una función y como armarla o utilizarla, podemos profundizar un poco sobre el alcance de las variables, o como se dice en el ámbito del desarrollo **scope**.

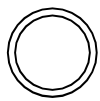
El alcance de una variable no es más que el ámbito en donde están disponibles las variables, es decir que nosotros podemos hacer que las variables puedan utilizarse en cualquier lugar de mi código JavaScript o que solamente exista en algunos lugares. De esta forma podemos limitar el espacio de memoria que vamos a estar utilizando, es decir que tal vez, necesitemos utilizar un valor para una acción en particular únicamente y no nos interese seguir guardándose luego, por lo que si limitamos esa variable a solamente existir para esa acción únicamente, dejamos más espacio de memoria libre para variables que tal vez necesitemos usar en varias acciones.

Este tipo de variables que solamente “existen” en algunas acciones en particulares, se dice que son **variables locales**.

Estas **variables locales** se definen únicamente dentro de las funciones que las necesitan, por lo que solamente van a existir cuando se ejecute dicha función.

En cambio, aquellas variables que necesitamos utilizar en varias acciones distintas ,por lo que necesitamos que existan más allá de cualquier función, se llaman **variables globales** y se definen al inicio del archivo JavaScript por fuera de cualquier función.

Esto permite que yo pueda invocarlas dentro de cualquier función y accionar según necesite, mientras que en las variables locales, solamente voy a poder invocarlas dentro de la misma función en la que se encuentran definidas.



2. Adquiriendo algunas buenas prácticas

Con todo este nuevo conocimiento adquirido en la unidad hasta ahora, podemos aprovechar para marcar algunas buenas prácticas sobre las variables en JavaScript:

- Al momento de nombrar las variables estas solamente pueden comenzar con una letra(la cual puede ser mayúscula o minúscula) , el símbolo pesos (\$) o un guión bajo (_).
- No podemos utilizar como nombre de variables alguna de las palabras reservadas ni tampoco , en caso que utilizar un nombre que se compone de varias palabras, debe contener espacios.
- Tener en cuenta al momento de nombrar e invocar variables el uso de mayúsculas y minúsculas, ya que JavaScript es case-sensitive(es decir que es susceptible a mayúsculas y minúsculas, por lo que no va a tomar como misma variable a una que tiene el mismo nombre en mayúscula que en minúscula).
- En JavaScript se suele agregar comentarios para ayudar al orden del código. Para ello contamos con dos maneras de agregar comentarios:

➡ `//` : Que nos permite agregar comentarios de una línea.

Ejemplo:

`// Este es un comentario en línea`

➡ `/**/` : Que nos permite agregar comentarios de más de una línea. Para utilizar este último lo que debemos hacer es encerrar el comentario entre estos signos.

Ejemplo:

`/* Este es un comentario que me permite`

`hacer un comentario de varias líneas. */`

- Al finalizar cada instrucción, se recomienda colocar un punto y coma (;) y generar un salto de línea para que el código quede legible, prolijo y asegurarnos de no producir errores en la sintaxis.

Ejemplo:

```
var precio = 1200 ;
```

```
var apellido = "Perez";
```

```
console.log(precio);
```

5. Operadores

Para poder realizar alguna de las acciones, los programas que desarrollamos necesitan de operadores.

Estos nos van a permitir tomar valores y darle sentido a las instrucciones e incluso poder tanto obtener nuevos valores como nueva información sobre los valores obtenidos.

Teniendo en cuenta que existen una amplia variedad de operadores, en esta sección vamos a ocuparnos de los tres grupos principales de operadores:

Operadores Aritméticos:

Son aquellos que ya conocemos que nos permiten usar números y realizar operaciones con ellos.

Estos utilizan los siguientes símbolos :

+ (suma)

- (resta)

* (multiplicación)

/ (división)

Son aquellos que producen una respuesta afirmativa(true) o negativa(false) a una pregunta de comparación. Es utilizada para comparar dos valores.

Estos operadores son:

Igual a (==)	Solo compara la igualdad del valor en sí de cada uno.
Igual a (===)	Compara igualdad entre el valor y tipo de valor.
Distinto a (!=)	Solo compara la desigualdad del valor en sí.
Distinto a (!==)	Compara la desigualdad del valor y tipo.
Mayor a (>)	Compara cuál de los dos valores es mayor
Menor a (<)	Compara cuál de los dos valores es menor
Mayor o igual a (>=)	Compara si alguno de los valores es mayor o igual que el otro
Menor o igual a (<=)	Compara si alguno de los valores es menor o igual que el otro

Operadores Lógicos:

Son aquellos que toman dos operandos y los evalúan, dando como resultado una respuesta afirmativa (true),o negativa (false); al igual que los operadores de comparación.

Ellos son:

AND (&&) :

Nos permite preguntarnos si dos cosas son simultáneamente verdaderas (es decir que son **true**).

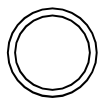
OR (||) :

Nos permite preguntarnos si al menos alguna de las dos cosas son verdaderas (es decir que son **true**)..

NOT(!):

Nos permite preguntarnos si algo no es lo que esperábamos que fuera(es decir si alguna de ellas es **false**) .

Tanto los operadores Lógicos como los de Comparación nos pueden servir para generar preguntas y tomar acciones según su respuesta, es decir utilizarlo como condiciones.



Esto nos va a permitir tomar distintos caminos según dicha respuesta y empezar a generar un comportamiento en el navegador dependiendo de una circunstancia en particular.

Por ahora , en una primera instancia vamos a armar estas condiciones con la siguiente sintaxis para que JavaScript lo pueda entender.

Esta consta de la comparación utilizando el operador , seguido de un signo de pregunta donde lo que coloquemos luego de ella se referirá a la acción a tomar en caso que la devolución de la operación de verdadero ,luego se colocan dos puntos (:) y seguido a esto, la acción que se tomara en caso que la operación dé como resultado falso.

comparación con los operadores ? acción si el operador devuelve *true* : acción si devuelve *false*

Ejemplo

```
var numero1 = 10 ;  
var numero2 = 20 ;  
  
numero1 > 100 ? console.log("es mayor a 100") : console.log("es menor a 100");
```

6.En resumen

En esta unidad pudimos aprender no solamente que son los valores sino que contamos con la posibilidad de guardar dichos valores en variables, que son y cómo poder determinar su alcance ; también vimos cómo podemos empezar a crear nuestras primeras funciones y algunas de las funciones creadas por el mismo lenguaje JavaScript.

Dentro de estas funciones vimos como poder empezar a utilizar operadores para darles nuestros primeros comandos a los programas.

Por último , hablamos sobre cómo podemos aplicar buenas prácticas en el lenguaje para ya desde un principio armar un código prolijo y claro.

7.Bibliografía utilizada y sugerida

<https://jsparagatos.com/>

<https://developer.mozilla.org/es/docs/Glossary/Variable>

<https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/var>

https://www.w3schools.com/js/js_variables.asp

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators#comparacion

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators#aritmeticos

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Expressions_and_Operators#logico