|  |
| --- |
| JAVA SCRIPT |
| ADA |

**Declaración de variables y operador de asignación**

[**#**](https://frontend.adaitw.org/docs/js/js01#declaraci%C3%B3n-de-variables)**Declaración de variables**

* Al programar necesitamos almacenar valores en la memoria de la computadora para poder interactuar con ellos
* Para poder identificar estos valores le asignamos un nombre descriptivo
* Es posible que a lo largo de un programa pisemos un valor guardado en una posición de memoria, es decir que este valor puede ser **variable**
* Podemos decir que una **variable** es algo que nos permite almacenar un valor en memoria de la computadora y que por medio de un nombre descriptivo podemos acceder e interactuar con él.
* En JavaScript existen las palabras reservadas **var**, **let** y **const** que nos permiten declarar una **variable**

*let* variable;

* Si en nuestro programa vamos a necesitar el nombre y edad del usuario podemos declarar las siguientes variables:

*let* nombre;

*let* edad;

* Hasta ES6 sólo podíamos utilizar **var** para declarar nuestras variables.
* Hoy se privilegia usar **let** o **const**.
* Los nombres de las variables tienen que empezar con una letra
* Para evitar problemas usemos nombres descriptivos que reflejen el valor que queremos asignar a cada variable
* En programación existen distintos tipos de forma de escribir los nombres de las variables, en EcmaScript es común utilizar una forma que se denomina camelCase donde la primer letra de la primer palabra va en minúscula y luego cada palabra comienza con mayúscula
* La primer letra del nombre de las variables va a ser siempre en minúscula
* Evitamos utilizar acentos o simbolos en los nombres de las variables

*let* nombreDeMiVariable;

* En el ejemplo anterior vemos que declaramos una variable con el nombre de **nombre de mi variable** dado que los nombres de las variables no aceptan espacios utilizamos la denominación camel case para escribirlo: **nombreDeMiVariable**

[**#**](https://frontend.adaitw.org/docs/js/js01#operador-de-asignaci%C3%B3n)**Operador de asignación**

* Una vez declarado el nombre de una variable podemos asignarle un valor. Esto se da ya que tenemos asignado un espacio en memoria donde podemos guardar un valor
* Por medio del operador de asignación **=** podemos asignar un valor a una variable

*let* nombre;

nombre = 'Ada';

* Podemos declarar todas las variables que necesitemos utilizar

*let* nombre;

*let* edad;

nombre = 'Ada';

edad = 20;

* También podemos declarar todas las variables en una línea y luego asignarle valores

*let* nombre, edad;

nombre = 'Ada';

edad = 20;

* También podemos declarar una variable y asignar un valor en la misma sentencia

*let* nombre = 'Ada';

*let* edad = 20;

* Cuando usamos **const** sólo podemos declarar variable y asignar valor en la misma sentencia, ya que const no permite cambios de valor.

*const* nombre = 'Ada';

*const* edad = 20;

* Utilizando console.log() podemos mostrar el valor de una variable en la consola

*let* nombre = 'Ada';

*let* edad = 20;

console.log(nombre);

console.log(edad);

**Ejecutamos el programa utilizando la consola del navegador**

* Vemos como salida los valores Ada y 20
* console.log() acepta varios valores separados por como para mostrar más de un valor en consola
* Podemos mostrar algo más significativo si escribimos lo mismo de la siguiente manera

*let* nombre = 'Ada';

*let* edad = 20;

console.log('nombre: ', nombre);

console.log('edad: ', edad);

**Ejecutamos el programa utilizando la consola del navegador**

* Al ejecutar este programa vemos una salida que nos explica mejor que es cada valor ya que vemos nombre: Ada y edad: 20
* Esta es una forma fácil de ver los valores de las variables y saber de donde viene

[**#**](https://frontend.adaitw.org/docs/js/js01#tipos-base)**Tipos base**

* Como vimos en los ejemplos anteriores existen distintos tipos de datos para representar distintos valores

*let* nombre = 'Ada';

*let* edad = 20;

console.log('nombre: ', nombre);

console.log('edad: ', edad);

* En este ejemplo vemos que para el nombre estamos utilizando un valor entre '' y para la edad estamos utilizando un número
* ECMAScript tiene los siguientes tipos base:
  + **string:** Los **string** también son conocidos como cadena de caracteres y no son más que un texto
  + **number:** El tipo de dato **number** son números y nos permiten hacer operaciones matemáticas
  + **boolean:** Este tipo **boolean** acepta valores del tipo **true** o **false**, es decir que podemos utilizarlo cuando necesitamos un valor **verdadero (true)** o **falso (false)**
  + **undefined:** Define que un valor es indefinido
  + **null:** Define que un valor es nulo, parece que es similar a undefined pero ya vamos a ver algunas diferencias
* A la hora de definir y asignar valores en nuestros programas vamos a tener que definir de que tipo de dato van a ser para saber que tipo de operaciones podemos hacer con ellos
* Existen más tipos en ECMAScript y los vamos a ir viendo a lo largo del curso pero por ahora podemos arrancar con estos

[**#**](https://frontend.adaitw.org/docs/js/js01#string)**String**

* Los string representan un valor de texto, lo podemos utilizar para guardar valores como nombre, apellido, dirección, etc.
* Los valores del tipo string se escriben entre comillas dobles "" o simples ''
* Si bien es lo mismo utilizar cualquier tipo de comillas por una cuestión de convensión vamos a utilizar comillas simple ('') a lo largo del curso

*let* nombre = 'Grace';

*let* apellido = "Hopper";

console.log(nombre);

console.log(apellido);

* En este ejemplo declaramos dos variables (nombre y apellido) y le asignamos dos **valores del tipo string ('Grace', 'Hopper')**
* Podemos utilizar este tipo de dato para un mensaje

*let* mensaje = '¡Bienvenidos a Javascript!';

console.log(mensaje);

* No hace falta asignar los tipos de datos a una variable, podemos utilizarlos como literales de la siguiente forma

console.log('¡Bienvenidos a Javascript!');

* En este ejemplo utilizamos un **string o cadena de caracteres** en console.log() directamente

[**#**](https://frontend.adaitw.org/docs/js/js01#concatenar-textos)**Concatenar textos**

* Utilizando el operador + podemos unir dos o más **strings**

*let* nombre = 'Sheryl';

*let* espacio = ' ';

*let* apellido = 'Sandberg';

console.log(nombre + espacio + apellido);

* Vemos como resultado de este ejemplo Sheryl Sandberd
* En este ejemplo vemos como podemos concatener o unir 3 variables del tipo string
* Podemos escribir este mismo ejemplo de la siguiente forma sin utilizar una variable para el espacio

*let* nombre = 'Sheryl';

*let* apellido = 'Sandberg';

console.log(nombre + ' ' + apellido);

* También vemos como resultado de este ejemplo Sheryl Sandberd
* Podemos ver en este ejemplo como utilizamos un valor literal para el espacio sin asignarlo a una variable

[**#**](https://frontend.adaitw.org/docs/js/js01#%C2%BFcomillas-dobles-o-simples)**¿Comillas dobles o simples?**

* Como sabemos podemos utilizar comillas dobles o simples para definir un tipo de dato **string**
* En algunos casos vamos a necesitar utilizar estos símbolos como parte del valor

*let* texto = 'Este texto tiene "comillas dobles"';

*let* otroTexto = "Este texto tiene 'comillas simples'";

console.log(texto);

console.log(otroTexto);

* Si necesitamos utilizar un tipo de comillas como parte de nuestro **string** podemos establecer el valor utilizando el otro tipo de comillas:
  + Si necesito utilizar comilla simple como contenido podemos definir el string utilizando comillas dobles
  + En caso de utilizar comilla doble como contenido definimos el string utilizando comillas simples

[**#**](https://frontend.adaitw.org/docs/js/js01#n%C3%BAmeros)**Números**

* Otro de los tipos de datos que tenemos en ECMAScript es **numbers** y nos permite utilizar números
* Este tipo de datos no va entre comillas ni dobles ni simples
* Para saber más sobre este tipo de datos pueden entrar en el [sitio de MDN](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Number)

*let* edad = 38;

*let* volumen = 50;

console.log(edad);

console.log(volumen);

* Un error común que podemos hacer cuando arrancamos a programar o aprender ECMAScript es cofundirnos al encontrar algún código similar al siguiente ejemplo

*let* edad = 38;

*let* volumen = "50";

En este caso tenemos dos variables que contienen un número. A la variable **edad** le asignamos un tipo de dato **number** y a la variable **volumen** le estamos asignando un tipo de dato **string** por más que el contenido sea un número. Es importante entender que son distintos tipos de datos por lo cual vamos a poder utilizarlos de diferente forma, por ejemplo a los **numbers** los vamos a poder utilizar para operaciones matemáticas y a los **string** no.

* Más adelante vamos a utilizar operadores aritméticos para hacer operaciones matemáticas con este tipo de datos (suma, resta, multiplicación)

[**#**](https://frontend.adaitw.org/docs/js/js01#boolean)**Boolean**

* Los tipos de datos boolean aceptan un valor del tipo true (verdadero) o false (falso)

*let* prendido = true;

*let* puedeVotar = false;

*let* casado = false;

console.log(prendido);

console.log(puedeVotar);

console.log(casado);

[**#**](https://frontend.adaitw.org/docs/js/js01#undefined)**Undefined**

* Al declarar una variable en ECMAScript y no asignarle ningún valor por defecto se le asigna el valor **undefined**
* Este valor significa que algo no esta definido
* Podemos asignarlo como valor de la siguiente manera:

*let* indefinido = *undefined*;

console.log(indefinido);

* Es común tener que validar si un valor es **undefined** y lo vamos a ver más adelante

[**#**](https://frontend.adaitw.org/docs/js/js01#null)**Null**

* Existe un valor llamado **null** para establecer que algo es nulo
* Podemos asignarlo como valor de la siguiente manera:
* También es común tener que validar si un valor es **null**

*let* nulo = *null*;

console.log(nulo);

* Si bien **null** y **undefined** parecen similares se utilizan de distinta manera

[**#**](https://frontend.adaitw.org/docs/js/js01#typeof)**typeof**

* Por medio del operador **typeof** podemos saber de que tipo es una variable

*let* nombre = 'Radia';

*let* edad = 30;

*let* casado = false;

*let* indefinido = *undefined*;

*let* nulo = *null*;

console.log(*typeof* nombre); *// string*

console.log(*typeof* edad); *// number*

console.log(*typeof* casado); *// boolean*

console.log(*typeof* indefinido); *// undefined*

console.log(*typeof* nulo); *// object*

* En este ejemplo vemos que cada tipo de dato retorna un tipo distinto
* En el caso de **null** retorna object en lugar de null como uno espera
* Object es otro tipo de dato de ECMAScript y lo vamos a ver más adelante

# Operadores aritméticos

* En ECMAScript existen operadores que nos van a permitir hacer operaciones ariméticas como pueden ser la suma, resta, multiplicación y división entre otros

### [#](https://frontend.adaitw.org/docs/js/js02#suma)Suma

* Usando el operador + podemos sumar dos números o dos tipos de dato **number**

2 + 2

* En este ejemplo estamos sumando dos valores de number literales

*let* miEdad = 20;

*let* edadDeMiHermano = 15;

console.log(miEdad + edadDeMiHermano);

* Podemos sumar el valor que tienen asignado 2 o más variables

*let* miEdad = 20;

*let* edadDeMiHermano = 15;

*let* resultado = miEdad + edadDeMiHermano;

console.log(resultado);

* También podemos utilizar una variable para guardar el resultado de sumar los valores de otras variables

### [#](https://frontend.adaitw.org/docs/js/js02#resta)Resta

* Usando el operador - podemos restar dos números o dos tipos de dato **number**

2 - 2; *// Obtenemos 0 como resultado*

*let* miEdad = 20;

*let* edadDeMiHermano = 15;

*// Mostramos en consola el resultado de restar los valores de estas dos variables*

console.log(miEdad - edadDeMiHermano);

*// Utilizamos una variable para guardar el resultado de la operación*

*let* resultado = miEdad - edadDeMiHermano;

console.log(miEdad - edadDeMiHermano);

* Podemos usar más de un operador:

10 + 2 - 2;

*let* miEdad = 20;

*let* edadDeMiHermano = 15;

*// Podemos utilizar distintos operadores como también valores literales*

console.log(miEdad - edadDeMiHermano + 10);

*let* resultado = miEdad - edadDeMiHermano + 10;

console.log('El resultado es: ' + resultado);

### [#](https://frontend.adaitw.org/docs/js/js02#multiplicaci%C3%B3n)Multiplicación

* Usando el operador \* podemos multiplicar dos números o dos tipos de dato **number**

2 \* 2; *// Obtenemos 4 como resultado*

*let* numero1 = 10;

*let* numero2 = 5;

console.log(numero1 \* numero2);

*let* resultado = numero1 \* numero2;

console.log(resultado)

* Podemos utilizar paréntesis para establecer que operación queremos que se resuelva primero, esto tiene que ver con la precendecia de operadores. Es como volver al colegio pero ahora tiene sentido!

2 + 2 \* 4; *// 10*

(2 + 2) \* 4; *// 16*

* En este caso vemos que al utilizar paréntesis estabamos definiendo como queremos que se hagan las operaciones
* En el segundo ejemplo vemos que primero se va a resolver la suma de 2 + 2 y luego se va a multiplicar el resultado
* Este concepto también funciona con variables como es de esperar

*let* dos = 2;

*let* cuatro = 4;

console.log(dos + dos \* cuatro); *// 10*

console.log( (dos + dos) \* cuatro ); *// 16*

### [#](https://frontend.adaitw.org/docs/js/js02#divisi%C3%B3n)División

* Usando el operador / podemos dividir dos números o dos tipos de dato **number**

20 / 2; *// 10*

*let* numero1 = 20;

*let* numero2 = 2;

console.log(numero1 / numero2); *// 10*

*let* resultado = numero1 / numero2;

console.log(resultado); *// 10*

* Dado que este código es una representación de una operación matemática tenemos que tener cuidado a la hora de dividir por 0
* ECMAScript obtenemos un valor del tipo **Infinity** al intentar dividir por 0
* Para saber más sobre este valor podes visitar el [sitio de MDN](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Infinity)

### [#](https://frontend.adaitw.org/docs/js/js02#m%C3%B3dulo-o-resto)Módulo o resto

* Usando el operador % podemos obtener el resto de dividir dos números o dos tipos de dato **number**

20 % 2; *// 0*

*let* numero1 = 20;

*let* numero2 = 2;

console.log(numero1 % numero2); *// 0*

*let* resultado = numero1 % numero2;

console.log(resultado); *// 0*

* Este operador nos es útil por ejemplo si queremos saber si un número es par o no.

### [#](https://frontend.adaitw.org/docs/js/js02#incremento-y-decremento)Incremento y decremento

* Por medio de distintos operadores podemos hacer operaciones de una forma más simple a nivel código

#### [#](https://frontend.adaitw.org/docs/js/js02#incrementar-en-1)Incrementar en 1

* Utilizando el operador ++ podemos incrementar un valor en 1

**Ejemeplo:**

*let* numero = 0;

numero++;

console.log(numero); *// 1*

* También podemos establecer que primero queremos incrementar la variable para luego utilizarla cambiando el lugar del operador

*let* numero = 0;

++numero;

console.log(numero); *// 1*

* En este caso parece ser lo mismo pero se pueden dar situaciones donde no lo sea

#### [#](https://frontend.adaitw.org/docs/js/js02#restar-un-n%C3%BAmero)Restar un número

* Utilizando el operador -- podemos reducir un valor en 1
* Al igual que en el incremento el operador puede ir delante o después del valor según el resultado esperado

*let* numero = 10;

--numero;

console.log(numero); *// 9*

numero--;

console.log(numero); *// 8*

#### [#](https://frontend.adaitw.org/docs/js/js02#hacer-una-operaci%C3%B3n-sobre-un-mismo-valor)Hacer una operación sobre un mismo valor

* Al definir una variable podemos asignarle un valor como ya vimos
* Podemos reutilizar esa variable para asignarle otro valor
* También podemos utilizar la variable para usar el valor y luego asignarlo de nuevo a la misma variable
* Vamos a ver un ejemplo:

*let* numero = 1;

numero = numero + 1

* Como vimos podemos usar el operador ++ para conseguir el mismo resultado

*let* numero = 1;

numero++;

* Es decir que en este caso se incrementa y asigna el valor de la variable numero
* Este operador es súper útil pero sólo nos permite operar con la suma y con un valor de 1
* Existen distintos operadores que nos permiten hacer operaciones sobre un valor y asignar el resultado a la misma variable escribiendo menos código
* Los operadores son:
  + += para la suma
  + -= para la resta
  + \*= para la multiplicación
  + /= para la división
* Este concepto se entiende mejor desde código

*let* numero = 1;

numero +=

console.log(numero); *// 2*

*let* numero = 1;

numero = numero + 10;

console.log(numero); *// 11*

* También puedo hacer esta operación de la siguiente forma

*let* numero = 1;

numero += 10;

console.log(numero); *// 11*

* Vemos que podemos sumar el valor 10 al valor que tiene la variable numero y asignar el resultado a la misma variable utilizando sólo el operador +=
* Podemos hacer esto con el resto de los operadores

*let* numero = 10;

numero -= 2;

console.log(numero); *// 8*

*let* numero = 10;

numero \*= 2;

console.log(numero); *// 20*

*let* numero = 20;

numero /= 2;

console.log(numero); *// 10*

* El concepto es siempre el mismo y lo que cambia es la operación realizada

# [#](https://frontend.adaitw.org/docs/js/js02#operadores-de-comparaci%C3%B3n-simple-y-estricta)Operadores de comparación simple y estricta

### [#](https://frontend.adaitw.org/docs/js/js02#comparaci%C3%B3n-simple)Comparación Simple

* Podemos comparar dos valores utilizando el operador == y obtener un valor **boolean** como resultado.
* Este tipo de comparación se conoce como comparación simple ya que sólo compara si un valor es igual a otro
* Al comparar 2 valores de distintos tipos podemos obtener que son el mismo valor sin importar que sean diferente tipo (ejemplo: comparar un string con un número)
* Si los valores son iguales obtenemos **true**
* En caso de que los valores sean distintos obtenemos el valor **false**

*let* numero1 = 20;

*let* numero2 = 20;

*let* numero3 = 10;

console.log(numero1 == numero2); *//true*

console.log(numero1 == numero3); *//false*

* Comparamos sólo valores:

console.log(10 == '10'); *//true*

* Ya que podemos comparar dos valores y saber si son iguales también podemos saber si son distintos utilizando el operador !=

*let* numero1 = 20;

*let* numero2 = 20;

*let* numero3 = 10;

console.log(numero1 != numero2); *// false*

console.log(numero1 != numero3); *// true*

* Otra forma de comparar valores es saber si un valor es más grande que otro
* Utilizamos el operador > para saber si el valor de la izquierda del operador es más grande que el valor de la derecha

*let* numero1 = 20;

*let* numero2 = 10;

console.log(numero1 > numero2); *// true*

* También podemos saber si un valor es más chico que otro utilizando el operador <

*let* numero1 = 20;

*let* numero2 = 10;

console.log(numero2 < numero1); *// true*

* En algunos casos necesitamos saber si un valor es más grande o igual que otro
* Es decir que esta condición se va a transformar en verdadera en caso de que el valor de la izquierda sea más grande o el mismo valor que el de la derecha

*let* numero1 = 20;

*let* numero2 = 10;

*let* numero3 = 20;

console.log(numero1 >= numero2); *// true*

console.log(numero1 >= numero3); *// true*

* Podemos hacer lo mismo para saber si es menor

*let* numero1 = 20;

*let* numero2 = 10;

*let* numero3 = 10;

console.log(numero2 <= numero1); *// true*

console.log(numero2 <= numero3); *// true*

### [#](https://frontend.adaitw.org/docs/js/js02#comparaci%C3%B3n-estr%C3%ADcta)Comparación Estrícta

* La comparación estricta no solo compara el valor sino también el tipo de dato
* Utilizamos el operador === para comparar si son el mismo tipo de dato y valor
* Utilizamos el operador !== para comparar si son el distintos tipo de dato y valor

console.log(10 === '10'); *// false*

console.log(10 !== '10'); *// true*

console.log(10 !== '10'); *// true*

* Los dos últimos casos da **true** ya que no importa el valor que tengan ambos valores son distintos tipo de dato

# [#](https://frontend.adaitw.org/docs/js/js02#operadores-l%C3%B3gicos)Operadores lógicos

* Existen operadores lógicos que nos permiten anidar condicionales
* Puedo saber si dos conciones son **true** utilizando el operador && conocido como **and** o en español como **Y**
* Por ejemplo si queremos saber si el la edad del usuario es mayor de 18 años y si el password es el esperado lo podemos hacer de la siguiente manera
* Para que este operador retorne **true** ambas condiciones deben ser verdaderas

*let* edad = 20;

*let* password = 'js1234';

*let* resultado = edad >= 18 && password ==='js1234';

console.log('resultado: ', resultado); *// true*

* En este ejemplo obtenemos un valor de true ya que ambas condiciones (edad >= 18 y password === 'js1234') son verdaderas
* Existe el operador || conocido como **or** u **O** en español que nos permite preguntar si al menos una de 2 condiciones es verdadera.
* Si la primer condición es verdadera ya no evalúa la segunda ya que al menos una de las dos condiciones es verdadera
* Caso de que la primer condición no sea verdadera va a comprobar si la segunda lo es
* Si ninguna de las dos condiciones es verdadera retorna falso
* Este operador retorna **true** si al menos una de las condiciones es verdadera

*let* edad = 20;

*let* password = 'js12345';

*let* resultado = edad >= 18 || password ==='js1234';

console.log('resultado: ', resultado); *// true*

* En este caso la condición es verdadera ya que la edad del usuario es mayor a 18 y no importa si el password es igual o no

*let* edad = 10;

*let* password = 'js1234';

*let* resultado = edad >= 18 || password ==='js1234';

console.log('resultado: ', resultado); *// true*

* En este caso el resultado es **true** ya que el usuario no es mayor de 18 pero el password es correcto

*let* edad = 10;

*let* password = 'js12345';

*let* resultado = edad >= 18 || password ==='js1234';

console.log('resultado: ', resultado); *// false*

* En este caso la condición es **false** ya que ambas condiciones son falsas

# [#](https://frontend.adaitw.org/docs/js/js02#negaci%C3%B3n)Negación

* Por medio del operador ! podemos negar una condición
* Si tenemos un valor **true** negado obtenemos un valor **false**
* Si tenemos un valor **false** negado obtenemos un valor **true**

console.log(!true); *// false*

console.log(!false); *// true*

* Por ejemplo podemos utilizar la negación en el siguiente caso:

*let* edad = 21;

*let* resultado = edad < 18;

console.log('El usuario es mayor de edad?: ', !resultado); *// la condición es false pero al negarla pasa a ser verdadera*