ALURA LATAM

ESTUDIO DE PROGRAMACION DE LO QUE VOY PROGRESANDO

ALURA LATAM

NODE.JS- JavaScript

JavaScript tipos, variables y funciones

24/07/2024

CONTENIDO

# Preparando el ambiente

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84237/next)

JavaScript es conocido como el “lenguaje de la web”. Aunque fue desarrollado en el front-end, hoy en día usamos JavaScript cada vez más en el llamado back-end, o server side (lado del servidor) de las aplicaciones web. Este curso se enfoca en el uso de JavaScript para el backend.

Así, necesitaremos instalar dos herramientas: NodeJS, necesaria para ejecutar código JavaScript sin necesidad del navegador, y Visual Studio Code o VSC, el editor que usamos para escribir código.

## NodoJS

Si ya tomó un curso anterior de JavaScript de back-end, es posible que NodeJS ya esté instalado. Si no estás seguro, puedes consultar con los siguientes pasos:

1) Abra una ventana de terminal en su computadora. Esto puede hacerse de la siguiente manera:

**Windows:** puede acceder a la búsqueda desde el menú de inicio y buscar Símbolo del sistema, o acceder a ella a través de Menú Inicio > Sistema de Windows > Símbolo del sistema.

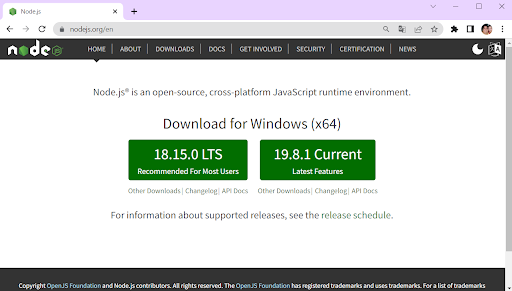
**MacOs:** Terminal está disponible en el menú Aplicaciones, dentro de la carpeta Utilidades.

**Linux (Ubuntu):** La Terminal está disponible en el menú Programas/Aplicaciones. Si no, puede estar dentro de la carpeta Utilidades.

2) En la terminal, escriba node --version o node -v; si devuelve un número de versión, como v16.15.0, NodeJS ya está instalado.

Si necesita instalarlo, siga las instrucciones para cada sistema operativo que se encuentran en la página de inicio de NodeJS. Este curso fue desarrollado utilizando la versión 16.15.0.

El sitio web de NodeJS ofrece dos opciones de descarga, LTS y "actual". Puede elegir la versión LTS (Soporte a largo plazo) y hacer clic en el botón correspondiente para descargar e instalar normalmente como cualquier otro programa.



## Visual Studio Code

Visual Studio Code es el editor que elegimos usar para escribir nuestro código durante este curso. Los enlaces para descargar e instalar según su sistema operativo se encuentran en la página de inicio de VSC.

Hay varios otros editores como Atom, Sublime y Notepad++. Si ya tienes uno de estos instalado o prefieres otro, ¡no hay problema! Solo tenga en cuenta que algunas funciones y herramientas varían de un editor a otro y es posible que no estén disponibles o funcionen de manera diferente.

¡Ya podemos empezar!

# Para saber más: más sobre números

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84239/next)

Los programas funcionan gestionando valores, como el número 3.14 o el texto Juliana y Leonardo. Los tipos de valores que se pueden representar y manipular en un lenguaje de programación se conocen como tipos, y una de las características más fundamentales de un lenguaje de programación es el conjunto de tipos que soporta.

**Variables**

Cuando un programa necesita mantener un valor para uso futuro, asigna el valor a una variable o "almacena" los datos en una variable. Las variables tienen nombres y nos permiten usar esos nombres en nuestros programas para referirnos a valores. La forma en que funcionan las variables es otra característica fundamental de cualquier lenguaje de programación.

Los tipos de datos de JavaScript se pueden dividir en dos categorías: tipos primitivos y tipos de objetos o estructurados. Los tipos primitivos de JavaScript incluyen números, palabras o texto (conocidos como cadenas) y valores booleanos (conocidos como booleanos).

**Tipos numéricos**

Como vimos en clase, podemos almacenar números de diferentes maneras:

**const** edadLeonardo = 39;

**const** pi = 3.14;COPIA EL CÓDIGO

Consejo: En Javascript puedes obtener exacto de pi usando la propiedad Math.PI

El punto flotante puede tener un punto decimal; utilizan la sintaxis tradicional para números reales. Un valor real se representa como la parte entera del número, seguida de un punto decimal y la parte fraccionaria del número.

Los puntos flotantes también se pueden representar mediante notación exponencial: un número real seguido de la letra e (o E), seguido de un signo opcional más (+) o menos (-) y un exponente entero. Esta notación representa el número real multiplicado por 10 elevado a la potencia del exponente.

La división por cero no es un error de JavaScript: simplemente devuelve "Infinito". Sin embargo, hay una excepción: cero dividido por cero no tiene un valor definido y el resultado de esta operación es el valor especial no numérico NaN.

**var** a = 10

**var** b = 0

console.**log**(a/b) // InfinityCOPIA EL CÓDIGO

**var** a = 0

**var** b = 0

console.**log**(a/b) // NaN

# Para saber más: codificación de strings

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84240/next)

Acabamos de ver que usamos el tipo de cadena cada vez que queremos trabajar con datos de texto. Pero si nos detenemos a pensarlo, varios idiomas utilizan distintos caracteres, como acentos e ideogramas. ¿Cómo manejan esto los lenguajes de programación? ¿Y los emojis? ¿Alguna vez visitó un sitio web y notó que los caracteres de los textos no parecían correctos, que en lugar de algunos de ellos aparecían signos de interrogación, cuadrados o guiones?

Todo esto tiene que ver con la codificación de caracteres. En las últimas décadas se han desarrollado varios conjuntos de caracteres especiales, cada uno con sus propios códigos, para que las personas que escriben y leen en idiomas distintos al inglés puedan utilizar computadoras con sus propios idiomas. Y, ¿cómo funciona?

Para que la computadora pueda descifrar un carácter especial, es necesario utilizar un sistema específico que básicamente tiene un código para cada carácter, y que la computadora puede acceder a él para hacer la conversión, una idea similar a la de la criptografía.

Se han desarrollado varios conjuntos de caracteres, desde los específicos del idioma como occidental, latinoamericano, japonés, etc., hasta ASCII (American Standard Code for Information Interchange). y a partir de 2007 se adoptó el formato Unicode. El estándar UTF (Unicode Transformation Format) se utiliza como estándar en la web hasta el día de hoy.

Unicode tiene códigos específicos para "cifrar" y "descifrar" caracteres de más de 150 idiomas antiguos y modernos, así como varios otros conjuntos de caracteres, como símbolos matemáticos e incluso emojis. Wikipedia tiene una lista extensa de todas las tablas con códigos y caracteres Unicode, como las siguientes:

Descripción oficial **del** personaje UTF-16

$ U+0024 SIGNO DE DÓLAR

A U+0041 LETRA A MAYÚSCULA LATINA

✅ U+2705 MARCA DE VERIFICACIÓN

ぁ U+3041 HIRAGANA LETRA A PEQUEÑACOPIA EL CÓDIGO

Podemos probar la transformación/conversión del código Unicode en caracteres usando console.log(). Haz la prueba:

**const** simboloPeso = '\u0024'

**const** aMayusculo = '\u0041'

**const** check = '\u2705'

**const** hiragana = '\u3041'

console.log(simboloPeso)

console.log(aMayusculo)

console.log(check)

console.log(hiragana)COPIA EL CÓDIGO

Los caracteres \u al comienzo del código son caracteres de escape que usamos para señalar a JavaScript que estamos hablando de códigos Unicode y no de las cadenas de texto habituales.

JavaScript usa UTF-16 por defecto. El número 16 se relaciona con los espacios de bits ocupados por cada carácter, 16 en este caso. No profundizaremos en la relación entre los tipos de datos y el espacio de memoria ocupado por cada tipo, puedes investigar más sobre el tema, así como sobre los caracteres de escape, pero por ahora es genial ver cómo funciona Unicode en la práctica.

Las bases de datos pueden aceptar otros tipos de codificación de caracteres, lo que tiene sentido teniendo en cuenta que UTF-16 utiliza una cantidad de espacio de memoria relativamente grande para guardar cada carácter. 16 bits suena pequeño, ¡pero a veces los bancos necesitan guardar grandes cantidades de datos! Sin embargo, con las tecnologías de almacenamiento y tráfico de datos que tenemos hoy, esto ya no es una preocupación tan grande, excepto en casos muy específicos. Ya no es muy común utilizar una codificación diferente a UTF en las bases de datos incluso en el caso de grandes volúmenes de datos, pero siempre dependerá del caso.

Detalles y documentación más precisos sobre Unicode en la página de la [Fundación Unicode](https://home.unicode.org/).

<https://home.unicode.org/>

# Para saber más: trabajando con strings

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84241/next)

JavaScript trae en su biblioteca base varios métodos que utilizamos para manipular cadenas de texto: cambiar de mayúsculas a minúsculas, contar cuántas letras hay en una palabra, eliminar espacios, unir dos cadenas, etc.

Pensemos en algunos ejemplos prácticos para hacer este tipo de cambio. Por ejemplo, para estandarizar una comparación entre cadenas:

**const** ciudad = "buenos aires";

**const** input = "Buenos Aires";

console.log(cidade === input); // falseCOPIA EL CÓDIGO

Nosotros, como personas, pudimos percibir el valor de la ciudad y las variables de entrada como de la misma ciudad, Buenos Aires. Pero para JavaScript, ambos datos son solo cadenas de caracteres, y la comparación fallará, porque como hemos visto, JavaScript distingue entre mayúsculas y minúsculas, tanto en los valores de los datos como en el código que escribimos.

Una de las formas de lidiar con esto es establecer por defecto todas las entradas en el formato de texto que se comparará antes de realizar la comparación. En este caso, pasando todos los caracteres a minúsculas.

**const** ciudad = "buenos aires";

**const** input = "Buenos Aires";

**const** inputMinusculo = input.toLowerCase();

console.log(ciudad === inputMinusculo); // trueCOPIA EL CÓDIGO

Arriba, vemos uno de los métodos de cadena nativos de JavaScript en acción, toLowerCase() que convierte todos los caracteres de la cadena informada (en este caso, input) a letras minúsculas (si son números, no se convierte nada). Puedes consultar más sobre este método en la [MDN](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase).

Otro ejemplo: cualquier entrada de texto que requiera una cantidad mínima de caracteres, como una contraseña o un nombre. La propiedad de longitud se puede utilizar para averiguar cuántos caracteres contiene una cadena:

**const** clave = "miClave123"

console.log(clave.length) // 10 caracteresCOPIA EL CÓDIGO

La propiedad de length se usa mucho en el desarrollo web diario. Puedes saber más sobre ella [aquí](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String/length).

¿Observaste que la longitud no tiene paréntesis al final de la palabra? Hay una diferencia entre los métodos y las propiedades que no cubriremos durante este curso, ¡pero dejaremos el consejo aquí en caso de que tengas curiosidad! ;)

Puedes consultar la lista completa de [métodos de cadenas de MDN](https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String#m%C3%A9todos) (hay varios), con la descripción de cada uno, y practicar con los ejemplos.

# Cargando datos iniciales

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84263/next)

En JavaScript, como en la mayoría de los lenguajes de programación, es posible almacenar datos en la memoria de la computadora a través de variables.

Estos datos pueden ser palabras, números o valores lógicos como “true” o “false” (llamados tipos booleanos), como se ilustra en la siguiente tabla:

| **nombre de variable** | **valor** |
| --- | --- |
| edad | 2 |
| nombre | "Lara" |
| apellido | "Ayumi" |
| graduado | false |
| diezAnosDespues | "12" |

Sabiendo esto, analice las siguientes afirmaciones y seleccione la verdadera.:

Principio del formulario

* Alternativa correta



Al ejecutar el código const nombreCompleto = nombre + apellido, no se mostraría ningún error y el valor almacenado en la variable sería "LaraAyumi".

Usamos el signo más (+) para unir 2 palabras. Para acercar esta salida al mundo real, podemos agregar una cadena vacía entre las variables, como se muestra en el siguiente código const nombreCompleto = nombre + " " +apellido.

# Para saber más: patrones de nombres de JavaScript

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84242/next)

Un detalle muy importante, del que a veces no nos damos cuenta cuando empezamos a programar, es que cada lenguaje tiene sus propios estándares. Sirven no solo para escribir código que funcione, sino también para crear nombres de variables, estructurar un programa y mucho más.

Lo primero que debemos tener en cuenta es que JavaScript distingue entre mayúsculas y minúsculas, es decir, distingue entre mayúsculas y minúsculas. Esto significa que todo lo que escribimos, ya sean instrucciones específicas del idioma (como console.log) o al nombrar una variable, debe hacerse con el mismo patrón, que incluye las mayúsculas.

Para ilustrar, JavaScript trata los cuatro ejemplos a continuación como variables diferentes y no arrojará ningún error si ejecuta el programa:

**const** miVar = 1;

**const** MiVar = "texto";

**const** mivar = "3";

**const** MIVAR = 2;

console.log(miVar, MiVar, mivar, MIVAR);COPIA EL CÓDIGO

Podemos ver que, en un programa muy grande, la posibilidad de problemas es grande. Entonces, ¿cómo sabemos la forma correcta de nombrarlo? Ahí es donde entra lo que llamamos convenciones, para estandarizar estos aspectos del código.

Hay varias convenciones de nomenclatura, y cada lenguaje de programación tiene una. Estos son algunos de ellos:

**camelCase:** Comienza con una letra minúscula y la primera letra de cada palabra se escribe en mayúscula.

* **Por ejemplo:** miVariable o claveUsuario. Esta es la convención utilizada por JavaScript para variables y funciones.

**snake\_case:** los espacios se reemplazan por el carácter \_ (guion bajo), con todas las palabras en minúsculas.

* **Por ejemplo:** mi\_variable o clave\_usuario.

**kebab-case:** Similar al anterior, pero con espacios reemplazados por guiones.

* Por ejemplo: mi-variable o clave-usuario. Esta convención no se puede utilizar en JavaScript para variables y funciones.

**PascalCase:** Similar a CamelCase, pero en este caso todas las palabras comienzan con una letra mayúscula.

* Por ejemplo: MiVariable o ClaveUsuario.

**Importante:** Nunca use espacios o caracteres especiales, ni comience nombres de variables con números.

Cuando hablamos de convención, hablamos de buenas prácticas y estandarización. Si usa alguno de los patrones anteriores para nombrar variables con JavaScript (con la excepción del patrón kebab-case), su código seguirá funcionando, pero seguir las convenciones es parte del desarrollo de un código legible y bien escrito.

Este es un tema amplio con muchos detalles, y es parte de nuestro trabajo diario como desarrolladores para garantizar que se sigan las llamadas guías de estilo definidas para un producto de código.

Puede aprender los detalles poco a poco a medida que estudia y ver cómo se aplican a los códigos que ve a su alrededor.

Final del formulario

# Para saber más: tipos null y undefined

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84243/next)

En esta clase hablamos de tres tipos primitivos: número, cadena y booleano. Pero hay dos tipos más que no hemos cubierto en profundidad: null e undefined.

null es un tipo especial, ya que se puede traducir como “ausencia de valor” y se puede asignar como el valor de una variable:

**let** input = null;

**if** (input === null) {

console.**log**('No hay información);

} **else** {

console.**log**(input);

}COPIA EL CÓDIGO

Si es así, ¿cuál sería la diferencia entre los dos casos a continuación?

**let** input = **null**;

**let** input2;

console.**log**(input); // null

console.**log**(input2); // undefinedCOPIA EL CÓDIGO

Aquí es donde entra el tipo undefined. Este tipo también representa “ausencia de valor”, pero de otra forma: normalmente, mientras nulo es un valor asignado a una variable que existe y ha sido inicializada, undefined se refiere al valor de una variable que no ha sido inicializada (es decir, no se le ha asignado ningún valor).

undefined es también el valor devuelto por una función que no tiene cláusula de retorno. Veremos más sobre las funciones y volveremos más adelante en el curso.

Es importante tener en cuenta que aunque ambos tipos se utilizan para indicar la ausencia de un valor, los operadores de comparación de JavaScript pueden o no diferenciarlos:

console.**log**(**null** == **undefined**); // true

console.**log**(**null** === **undefined**); // falseCOPIA EL CÓDIGO

En la vida cotidiana es común considerar undefined como una ausencia de valor “inesperada” (provocada por un bug o error en el código) y null como un tipo de dato que también significa ausencia de valor, pero no inesperadamente. Por ejemplo, un campo en una tabla de base de datos que no tiene datos o información solicitada que no es obligatoria y no ha sido completada por el usuario puede tener un valor null.

# Haga lo que hicimos: comportamiento de los tipos

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84258/next)

En esta lección, comenzamos a aprender sobre tipos, un concepto inicial y uno de los más importantes en cualquier lenguaje de programación. Estudiamos los tipos primitivos número, cadena y booleano con mayor detalle, pero tenemos que practicar para interiorizar.

VER OPINIÓN DEL INSTRUCTOR

### Opinión del instructor

En el caso de los números, es posible guardar en una variable no sólo el número en sí, como en const num = 5; sino también el resultado de una operación (o más):

**const** suma = 10 + 10;

**const** multiplicacion = 10 \* 10;

**const** operacion = (suma + multiplicacion) + 10;COPIA EL CÓDIGO

Tenga en cuenta que, en el código anterior, usamos paréntesis ( ) para seguir la regla matemática de precedencia en las operaciones.

¿Puedes pensar en más casos con operaciones matemáticas básicas para probar?

# Lo que aprendimos

[**PRÓXIMA ACTIVIDAD**](https://app.aluracursos.com/course/javascript-tipos-variables-funciones/task/84253/next)

Lo que aprendimos en esta aula:

* Que la computadora solo puede almacenar valores en algunos tipos de variables, como texto y números booleanos.
* Cómo declarar una variable numérica, de tipo número, y asociarle un valor, que puede ser un valor entero o de coma flotante.
* Cómo declarar una variable de texto, conocida como cadena, y poner una frase, número, puntuación o una combinación de todos estos dentro de ella.
* Cómo declarar una variable booleana y establecer su valor en verdadero (verdadero) o falso (falso).
* Dónde aparecen más los booleanos, como resultado de comparaciones con === u otros operadores de comparación como >=.