Lógica de programación

Variables y tipos de datos

Profesor: Cristian Ariel Corbalan

Email: cristian.corbalan@davinci.edu.ar

Contenidos

- Que son los **valores** y como se guardan
- Variables
  - ¿Qué son?
  - Como se crean
  - Reglas de Nomenclatura
  - Constantes
  - Buenas prácticas
- Tipos de datos

Introducción

- La computadora trabaja con datos que están almacenados en **secuencias de bits** (sistema binario) 💾.
- El sistema binario es el sistema numérico base que utiliza la computadora y representa los números utilizando **solo dos símbolos**: 0️⃣ y 1️⃣.
- Podemos decir entonces que toda computadora funciona con el **sistema numérico base 2**.
- Cada dígito en una secuencia binaria tiene un valor que depende de su posición en la secuencia, lo que se conoce como sistema posicional.
- Al aumentar la posición a la **izquierda**, el valor de **cada dígito se duplica** ➡️✖️2️⃣.
- Por ejemplo, el número decimal 15 se puede representar en binario con solamente dos dígitos: **1111**.
- En la siguiente tabla están representados los bits que conforman el número 15, con la posición de cada dígito mostrado debajo de él:

0	**0**	**0**	**0**	**1**	**1**	**1**	**1**	Número binario
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**	Valor del bit de acuerdo a su posición expresado en números
2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**	Valor del bit de acuerdo a su posición expresado en forma de potencias de 2

Valores

- Una computadora tiene una gran cantidad de bits en su almacenamiento de datos 💿.
- Para trabajar con estos bits de forma ordenada, debemos separarlos en porciones que representen "compartimentos" de información, llamadas valores.
- **Cada valor está hecho de bits y tiene un tipo que determina su rol**.
- Algunos valores pueden ser *numéricos* 🔢, otros pueden ser *cadenas de texto* 📝, otros *funciones* ⚙️ y así sucesivamente.
- En sí, **los valores son información**.
- La mayoría de las veces, en la web se necesita trabajar con información. Por ejemplos:
  - **Una tienda en línea** 🛒: la información puede incluir productos que se venden y un carrito de compras.
  - **Una aplicación de chat** 💬: la información puede incluir usuarios, mensajes y mucho más.
- Por lo que necesitamos una forma de almacenar dicha información y poder acceder a ella las veces que sea necesario 📂.

📦 Variables

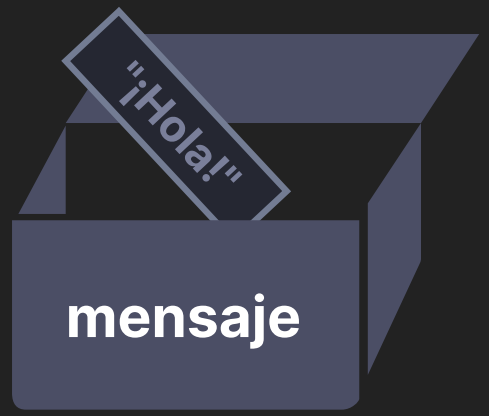
📌 ¿Qué son las variables?

- Las variables son un "almacenamiento con nombre" 🏷️ para los datos.
- Cada una de estas variables reserva un espacio de la memoria RAM para almacenar un determinado valor.
- **Este valor puede ser modificado** a lo largo de la ejecución del programa 🔄.
- Están asociadas a algún *tipo de dato*.
- A cada variable se le **debe dar un nombre diferente**, para diferenciarla de otras variables y poder identificarla de manera única.

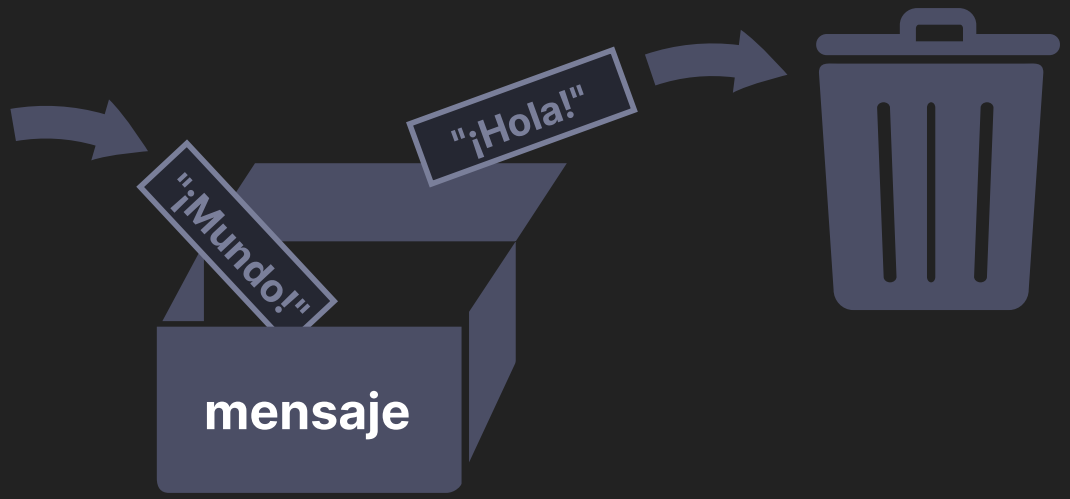
📦 Variables

Una analogía de la vida real

- Podemos comprender fácilmente el concepto de "variable" si la imaginamos como una "caja" 📦 para datos, con una etiqueta de nombre único .
- Por ejemplo, se puede imaginar la variable 'mensaje' como una caja etiquetada "mensaje" ✏️ que contiene el valor "*¡Hola!*" dentro.



- Podemos poner cualquier valor en la caja.
- También podemos cambiarlo tantas veces como queramos.
- Cuando se cambia el valor, los datos antiguos se eliminan de la variable 🗑️:



📦 Variables

Declaración

- Una variable debe de ser declarada para poder reservar un espacio en la memoria 🧠.
- En JavaScript, se declara una variable con la palabra reservada **`let`** seguida del nombre de la variable.

```
let mensaje;
```

- Es posible declarar varias variables a la vez separando sus nombres por comas.

```
let nombre, edad, mensaje;
```

- Puede parecer más corto, pero no es recomendable 🚫. Para una mejor legibilidad, es mejor tener una sola variable por línea.
- Algunas personas también definen múltiples variables en este estilo multilínea:

```
let nombre,
    edad,
    mensaje;
```

📦 Variables

`var` en lugar de `let`

- En los scripts más antiguos, también puede encontrar otra palabra clave: **`var`** en lugar de `let`:

```
1 var mensaje;
```

- La palabra reservada `var` es *casi* igual que `let`. También declara una variable, pero de una forma ligeramente diferente, "de la vieja escuela".
- Existen sutiles diferencias entre `let` y `var`, pero aún no nos importan. Las verán el cuatrimestre que viene en Programación I.

📦 Variables

Asignación e inicialización

- Para almacenar un valor en una variable utilizaremos el operador de asignación **=** (sí, el igual).

```
let mensaje;
mensaje = 'Hola'; // guardamos el string 'Hola' en la variable con el nombre mensaje
```

- La primera vez que se le asigna un valor a la variable se la inicializa (se le da el valor inicial).
- La inicialización puede suceder al declarar la variable o durante el desarrollo del programa.

Ejemplo:

```
let horasCursadas;          // Declaración

horasCursadas = 2;          // Iniciación (le asignamos el valor 2 a la variable declarada anteriormente).

let diasCursados = 1;       // Declaración e iniciación (le asignamos el valor 1).

diasCursados = 2;           // Reasignación, asignamos un nuevo valor a una variable ya inicializada.
```

📦 Variables

Constantes:

- Para declarar una variable constante (que no varía) 🔒, se utiliza **`const`** en lugar de `let`:

```
1 const fechaNacimiento = '18.04.1982';
```

- Las variables declaradas con `const` se denominan "constantes" y no pueden reasignarse 🚫.
- Un intento de hacerlo causaría un error ❗ :

```
1 const fechaNacimiento = '18.04.1982';
2
3 fechaNacimiento = '01.01.2001'; // error, no puede reasignar la constante!
```

📦 Variables

✍️ Nombre de las variables

- Nosotros mismos podemos elegir qué nombres dar a nuestras variables.
- Es aconsejable utilizar nombres significativos a su contenido 💡.
- En JavaScript el nombre de la variable debe cumplir con los siguientes requisitos:
  - 🔤 Utilizar caracteres del alfabeto inglés (a-z, sin ñ, ni tildes, ni caracteres especiales).
  - 🔢 Puede tener números, PERO NO empezar con uno.
  - Puede empezar y/o contener:
    - Guion bajo: _
    - Signo peso: $
- 🚫 No puede contener espacios en blanco, ni signos de puntuación, ni los siguientes símbolos: % + ( ) / -.

Nombre válido	Nombre no válido
segundo_cuatrimestre	~~2~~_cuatrimestre
materia1	materia~~-~~1
Comision	~~%~~Comisi~~é~~n
NombreApellidos	nombre~~&~~apellido
$cuota	~~€~~cuota

ℹ️ Case sensitive

- Se distingue entre mayúsculas y minúsculas.
- Las variables `edad`, `Edad` y `EDAD` son variables diferentes y válidas.

📦 Variables

🛑 Palabras reservadas

- Existe una lista de palabras reservadas, que no pueden utilizarse como nombres de variables porque las utiliza el propio lenguaje.
- Por ejemplo: `let`, `class`, `return`, y `function` son palabras reservadas 🖨️.
- El código siguiente da un error de sintaxis ❗ :

```
1 let let = 5;     // no puede nombrar una variable "let", ¡error!
2 let return = 5; // tampoco puede llamarse "return", ¡error!
```

Si bien no pueden ser el nombre de la variable, sí se pueden utilizar como parte del mismo 🧩. Por ejemplo: `let letrero;`

📦 Variables

Algunas recomendaciones

- El nombre de una variable debe tener un **significado claro y obvio** 🔎, que describa los datos que almacena.
- **Evite abreviaturas o nombres cortos** como `a`, `b` y `c` 🚫.
- Los nombres deben ser los más **descriptivos y concisos** posibles ✨. Ejemplos de nombres malos son `info` y `valor`. Esos nombres no dicen nada. Solo es correcto utilizarlos si el contexto del código hace excepcionalmente obvio a qué dato o valor hace referencia la variable.
- Es una buena práctica que los nombres de las variables estén escritas siempre en minúsculas 🔡.
- Si necesitamos que el nombre de la variable contenga más de una palabra, al no poder utilizar espacios, podemos separar estas de utilizando Camel case 🐫:
  - Escribir en mayúsculas la primera letra de cada palabra siguiente a la primera. Ejemplo: unaVariable.
- Todos los nombres tienen que estar en el mismo idioma 🌍.

🧠🏋️ Mini desafío

- **Crea una variable con el nombre de nuestro planeta. ¿Cómo nombrarías dicha variable?**

```
1 const nuestroPlanetaNombre = "Tierra";
```

- Nota, podríamos usar un nombre más corto, como `planeta` por ejemplo, pero podría no ser obvio a qué planeta se refiere.
- Es bueno ser más verboso. Al menos hasta que la variable noSeaDemasiadoLarga.

Intentemos otro:

- **Cree una variable para almacenar el nombre del usuario actual de un sitio web. ¿Cómo nombrarías esa variable?**

```
1 const usuarioActualNombre = "Jhon";
```

- De nuevo, podríamos acortarlo a `usuarioNombre` si sabemos con seguridad que es el usuario actual.
- Los editores modernos y el autocompletado facilitan la escritura de nombres largos de variables. No ahorren en ellos. Un nombre con 3 palabras está bien.

Veamos ahora los tipos de datos

Tipos de datos

¿Qué es un tipo de dato?

- Un valor en JavaScript es siempre de un tipo determinado. Por ejemplo, una cadena o un número.
- Es una restricción impuesta por el sistema informático que determina los valores que va a manejar la variable.
- Hay 9 tipos de datos existentes en JavaScript:
  - 🔢 Número (`number`)
  - 🧮 Número grande (`bigInt`)
  - 📝 Cadena (`string`)
  - ✅ Lógico (`boolean`)
  - 🚫 Nulo (`null`)
  - ❓ Indefinido (`undefined`)
  - 📦 Objeto (`object`)
  - 🏷️ Símbolo (`symbol`)
  - ⚙️ Función (`function`)
- Hablemos de estos con más detalle.

Tipos de datos

🔢 Número (`number`)

- Los valores del tipo `number` son valores numéricos.
- Representa tanto números enteros (integer) como decimales (float).
- Los números enteros pueden ser representados en distintos sistemas:
- **Sistema decimal** (base 10):
  - De 0 a 9.
- **Sistema octal** (base 8):
  - De 0 a 7.
  - Empiezan con 0.
- **Sistema hexadecimal** (base 16):
  - De 0 a 9 y de A a F.
  - Empiezan con 0x.
- **Sistema binario** (base 2):
  - De 0 a 1.

Tipos de datos

Número (`number`)

- Tabla comparativa de números en diferentes bases:

Base 10	Base 8	Base 16	Base 2
0	0	0	00000
1	01	0x1	00001
2	02	0x2	00010
3	03	0x3	00011
4	04	0x4	00100
5	05	0x5	00101
6	06	0x6	00110
7	07	0x7	00111
8	010	0x8	01000
9	011	0x9	01001
10	012	0xA	01010

Base 10	Base 8	Base 16	Base 2
11	013	0xB	01011
12	014	0xC	01100
13	015	0xD	01101
14	016	0XE	01110
15	017	0xF	01111
16	**020**	0x10	10000
17	**021**	0x11	10001
18	**022**	0x12	10010
19	**023**	0x13	10011
20	**024**	0x14	10100

Tipos de datos

Número (`number`)

- 📍 **Punto flotante / decimales (`float`):**
  - De 0 a 9
  - Son los números que llevan coma:
    - Se utiliza el punto (.) en lugar de coma (,) para separar la parte entera y la decimal.
- ∞ **Infinito (`infinity`):**
  - Representa el infinito matemático ∞. Es un valor especial que es **mayor que cualquier número**.
  - Se representa con el valor `Infinity` cuando el dato es muy grande.
- !? **No un número (`NaN`):**
  - Se representa con el valor `NaN` cuando el dato no es un número.
  - `NaN` **representa un error de cálculo**. Es el resultado de una operación matemática incorrecta o indefinida, por ejemplo:

```
1 "not a number" - 2; // NaN, dicha resta es errónea
```

Tipos de datos

🧱 Número grande (`bigint`)

- Se trata de una adición reciente a JavaScript
- `BigInt` es un tipo numérico especial que admite enteros de longitud arbitraria.
- En JavaScript, el tipo de dato `number` **no puede representar con seguridad** valores enteros mayores que `(253-1)` (es decir, `9007199254740991`), o menores que `-(253-1)` para los negativos.
- En realidad, `number` puede almacenar enteros más grandes, pero fuera del rango de enteros seguros `±(253-1)` habrá un error de precisión, porque no todos los dígitos caben en el almacenamiento fijo de 64 bits.
- 👉 Un `bigint` se crea añadiendo `n` al final de un número entero o llamando a la función **`BigInt`** que crea `bigints` a partir de cadenas, números, etc.

```
1 let bigint = 1234567890123456789012345678901234567890n;
2
3 let mismoBigint = BigInt("1234567890123456789012345678901234567890");
4
5 let bigintDesdeNumero = BigInt(10); // Igual que 10n
```

Tipos de datos

📝 Cadena (`string`)

- Se utilizan para representar datos textuales / cadenas de texto.
- Puede contener cero o más caracteres (letras, números y símbolos).
- Cada `string` debe ir entre comillas.
- En JavaScript, existen 3 tipos de comillas.
  - Par de comillas simples (`'Hola'`).
  - Par de comillas dobles (`"Hola"`).
  - Par de Backticks / comillas invertidas (`` `Hola` ``)
- Las comillas dobles y simples son comillas "simples". Prácticamente, no hay diferencia entre ellas.
- Las comillas invertidas nos permite generar plantillas literales. Pero es algo que veremos más adelante. 😈

Tipos de datos

✅ Lógico (`boolean`)

- Existen solamente dos valores lógicos:
  - ✅ Verdadero (`true`).
  - ⛔ Falso (`false`).
- Este tipo de dato se utiliza normalmente para almacenar valores sí/no: `true` significa "sí, correcto", y `false` significa "no, incorrecto".

```
1 let campoNombreVerificado = true; // sí, el campo nombre está verificado
2 let campoEdadVerificado = false; // no, el campo edad no está verificado
```

Tipos de datos

🚫 Nulo (`null`)

- Es un valor que representa "nada", "vacío" o "valor desconocido".
- Nos permite definir o comprobar que una variable está vacía.
- **Por ejemplo**: el siguiente código indica que la `edad` es desconocida:

```
1 let edad = null;
```

Tipos de datos

❓ Indefinido (`undefined`)

- Es un valor similar a `null`.
- El significado de `undefined` es "*valor no asignado*".
- Se obtiene cuando se declara una variable, pero no se inicializa:

```
let edad; // edad, en estos momentos guarda el valor undefined.
```

- Técnicamente, es posible asignar el valor `undefined` a una variable:

```
let edad = 100;

// cambiamos el valor a undefined
edad = undefined;
```

- … 👉 Pero **no es recomendable hacerlo**. Normalmente, se utiliza `null` para asignar un valor "vacío" o "desconocido" a una variable, mientras que `undefined` se reserva como valor inicial por defecto para las cosas no asignadas.

Tipos de datos

📦 Objeto (`object`)

- Los objetos son un tipo de dato especial.
- Todos los tipos de datos que vimos hasta ahora se denominan "**primitivos**" porque sus valores **solo pueden contener una única cosa** (Una cadena, un número o lo que sea).
- En cambio, los objetos sirven para almacenar colecciones de datos (propiedades) y entidades más complejas (métodos).
- Son un valor superimportante con el cual trabajan el cuatrimestre que viene.
- Pero por ahora pueden pensarlo simplemente como un valor que puede guardar muchos valores.

🏷️ Símbolo (`symbol`)

- Los símbolos se utiliza para crear identificadores únicos para los objetos.
- Lo menciono aquí porque también es un tipo de dato, pero hasta que no vean objetos no tiene sentido explicarlo más.

⚙️ Función (`function`)

- Son bloques de código que pueden ser invocados para ejecutar determinada tarea.
- Son reutilizables y nos permite optimizar y estructurar nuestro código.
- Otra vez, para el próximo cuatri. 😅

Creación de algoritmos

Ahora que sabemos cosas nuevas, vamos a ver como las podemos implementar en nuestros algoritmos 💡:

- Para definir variables vamos a utilizar `let` seguido del nombre de la variable y entre `()` su tipo de dato.
  - Ej.: `let edad (number integer)`
- Cuando queremos ingresar un valor ponemos **INGRESAR VALOR**.
  - Ej.: `edad = INGRESAR VALOR`
- Cuando queremos mostrar algo utilizamos **MOSTRAR**.
  - Ej.: `MOSTRAR edad`

Ejemplo: Realice el algoritmo para solicitar el nombre del usuario y mostrarlo.

```
let nombre (string)

nombre = INGRESAR VALOR

MOSTRAR nombre
```

🧠 Hora del desafío 🏋️

🧠🏋️ 🧠⏰ **10–20 min**

- Se le solicita al usuario su nombre, apellido, edad, altura y si está casado o no.
- Realice el algoritmo para informar los datos ingresados.

🎊 Fin de la clase 🎊