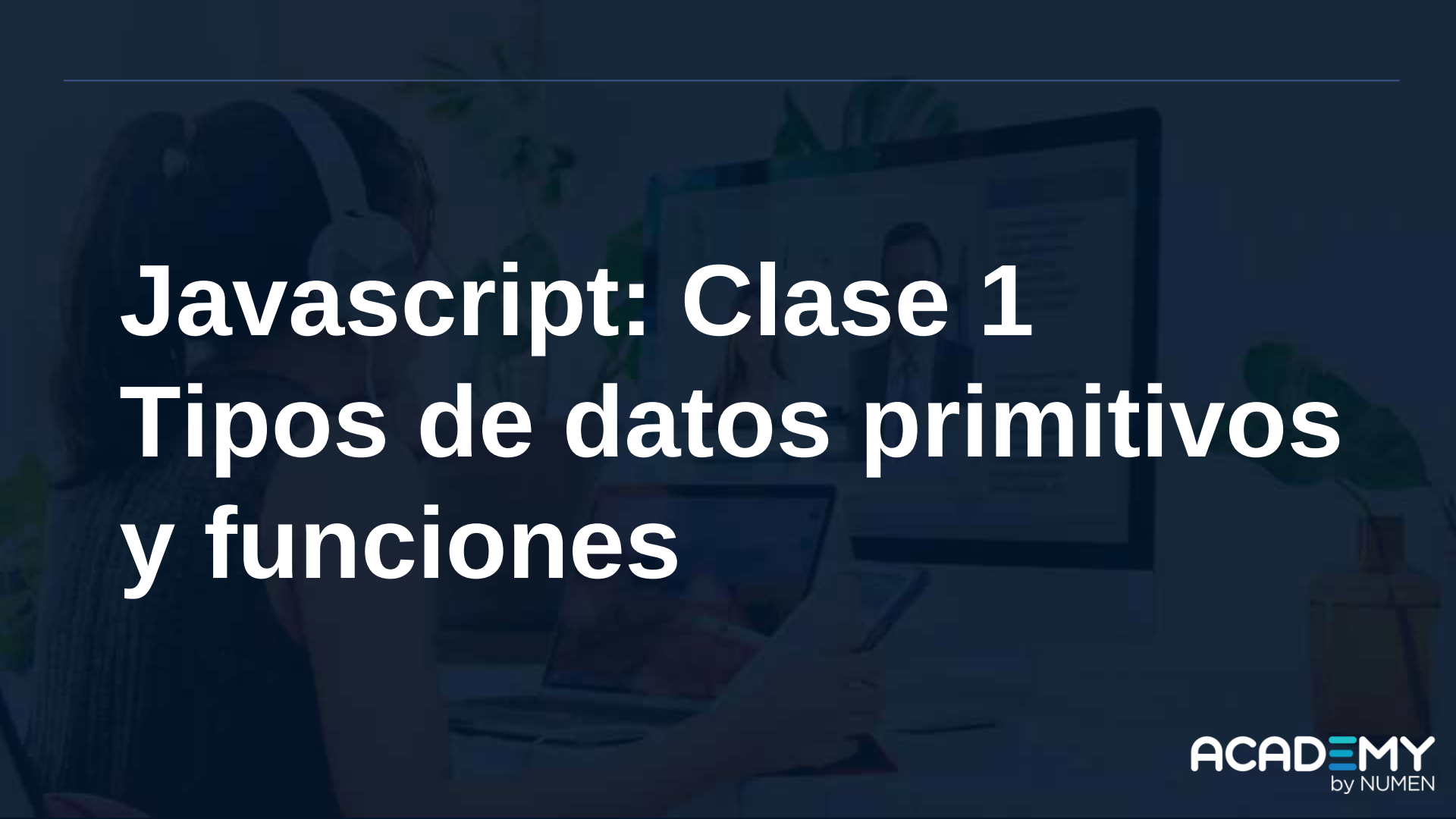
ACADEMY by NUMEN

FULL STACK

# Comenzamos en unos minutos


ACADEMY
by NUMEN

# Javascript: Clase 1
# Tipos de datos primitivos y funciones



ACADEMY by NUMEN

# Preparación

Vamos a instalar algunos programas para poder iniciar con Javascript

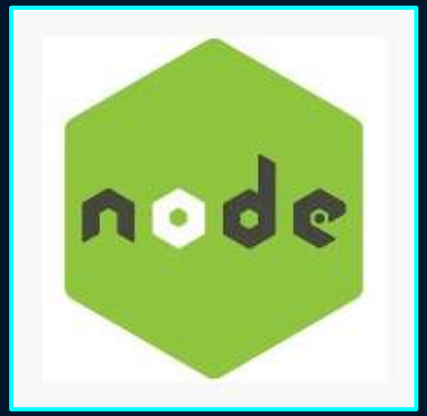
- Editor de Texto
- Github
- Git (Gitbash)
- Node Js

# GitHub

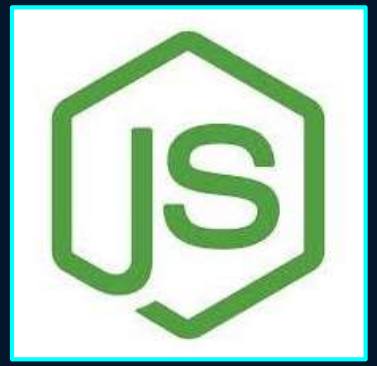
Es una red para almacenar tus repositorios, sería un repositorio de repositorios. Es uno de los tantos disponibles en internet, y el más popular. GitHub NO es lo mismo que Git, aunque funcionen muy bien juntos. Github es un lugar donde podés compartir tu código o encontrar otros proyectos. También actúa como portfolio para cualquier código en el que hayas trabajado.


# Node

Node. js es un entorno JavaScript que nos permite ejecutar en el servidor, de manera asíncrona, con una arquitectura orientada a eventos y basado en el motor V8 de Google. Es una plataforma que avanza muy rápidamente y cada vez está más presente en el mercado.







# Git / GitBash

**Git Bash** es una aplicación para entornos de Microsoft Windows que ofrece una capa de emulación para una experiencia de líneas de comandos de **Git**. Bash es el acrónimo en inglés de Bourne Again Shell. Una shell es una aplicación de terminal que se utiliza como interfaz con un sistema operativo mediante comandos escritos.

# Isomorfismo

Javascript es el único lenguaje que puede hacer isomorfismo. ¿Qué significa esto? Que puede trabajar tanto en Frontend como en Backend y base de datos. Es decir, es la única tecnología con la que puedes hacer toda una aplicación de principio a fin. Todos los otros lenguajes necesitan transformarse a Javascript para poder aplicarse del lado del cliente.

1. Frontend (Javascript)
2. Backend (Node.js)
3. Persistencia de datos (MongoDB, Couch DB, Firebase, etc)

# ¿Qué puedes hacer con Javascript?

- Diseño y desarrollo web.
- Hacer Videojuegos.
- Experiencias 3D, Realidad Aumentada, Realidad Virtual.
- Controlar Hardware (Drones, robots, electrodomésticos).
- Aplicaciones híbridas y móviles. (Ionic, React Native, etc).
- Machine Learning (aprendizaje automático).
- etc.

# Características de Javascript

- Lenguaje de Alto Nivel
    - Interpretado
    - Dinámico
    - Débilmente tipado
    - Multiparadigma
    - Sensible a MAYÚSCULAS y minúsculas
    - No necesitas los puntos y comas al final de cada línea.

# Tipos de datos

**Primitivos**: Se accede **directamente** al valor.

- Undefined
- Boolean (true, false)
- Number
- String (cadena de textos)
- null
- NaN

**Compuestos**: Se accede a la **referencia** del valor.

- Object = {}
- Function() {}
- Array = []
- Class {}

# Estructura de código

Los identificadores deben comenzar con:

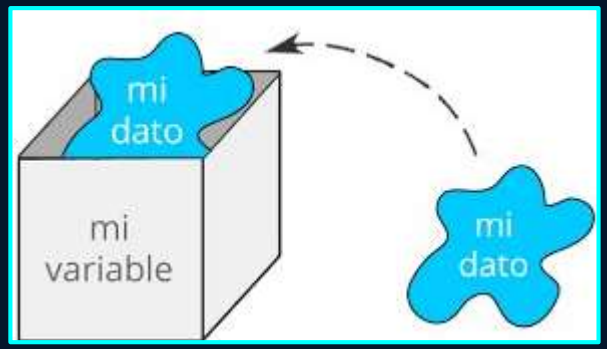
- Una letra
- Un signo de dólar ($)
- Un guión bajo (_)
- Nunca con números o caracteres especiales (*, #, %, etc).

# Palabras Reservadas

```
A: abstract
B: boolean, break, byte
C: case, catch, char, class, const, continue
D: debugger, default, delete, do, double
E: else, enum, export, extends
F: false, final, finally, float, for, function
G: goto
I: if, implements, import, in, instanceof, int, interface
L: let, long
N: native, new, null
P: package, private, protected, public
R: return
S: short, static, super, switch, synchronized
T: this, throw, throws, transient, true, try, typeof
V: var, volatile, void
W: while, with
```

# Variables

- Una variable es una forma de almacenar el valor de algo para usar más tarde.
- Una variable es un recurso de memoria del ordenador reservado para alojar una información.
- Las variables tienen un continente que puede o no almacenar un contenido. En caso de no tenerlo javascript guarda un espacio de memoria con el valor *undefined*



# Estructura de una variable

# Tipos de datos primitivos: Cadena de texto

El **String** se utiliza para representar y manipular una secuencia de caracteres.

Las cadenas son útiles para almacenar datos que se pueden representar en forma de texto.

Por detrás, un string es una colección de caracteres con una posición determinada. Esto nos permitirá interactuar con esas posiciones en futuras clases.

Las cadenas de texto pueden recibir propiedades. Una de ellas es la propiedad **length** que nos permite contar la cantidad de caracteres que componen a una cadena.

```
var cadena = "Academia Numen"

console.log(cadena)

var cadena2 = new String("Numen")

console.log(cadena2)

console.log(cadena.length)
```

# Concatenación

**Concatenar** es una elegante palabra de la programación que significa: "unir". Para unir cadenas en JavaScript el símbolo de más (+), el mismo operador que usamos para sumar números, pero en este contexto hace algo diferente.

Una concatenación se vería de esta manera:

```
var nombre = "Sherlock",
  apellido = "Holmes";

var saludo = "Hola mi nombre es " + nombre + " " + apellido + "."
```

# Tipos de datos primitivos: Números

**`Number`** es un objeto primitivo envolvente que permite representar y manipular valores numéricos cómo 37 o -9.25.

Al igual que con las cadenas, los números son también en el fondo, colecciones de números individuales con posición determinada.

En este caso, podemos notar que además interactuar con propiedades que nos brindan información (como el length en el caso de las cadenas), también es posible interactuar con métodos son, a diferencia de las propiedades, son acciones que modifican los datos.

```
var numero = 2021,
  numeroNegativo = -32,
  numeroConDecimales = 3.1416;

console.log(numero)
console.log(numeroNegativo)
console.log(numeroConDecimales)

var numero2 = new Number("Numen")

console.log(numero2)

console.log(numero.toString())
```

# Operadores aritméticos

| Operador | Descripción | Valor de y | Operación | Valor de x |
|---|---|---|---|---|
| + | Suma | 5 | x = y + 2 | 7 |
| - | Resta | 5 | x = y - 2 | 3 |
| * | Multiplicación | 5 | x = y * 2 | 10 |
| / | División | 5 | x = y / 2 | 2.5 |
| % | Resto división | 5 | x = y % 2 | 1 |
| -- | Decrementar | Dado y=5 | | |
| | | 4 | y-- | |
| | | 4 | x = --y | 4 |
| | | 4 | x = y-- | 5 |
| ++ | Incrementar | Dado y=5 | | |
| | | 6 | y++ | |
| | | 6 | x = ++y | 6 |
| | | 6 | x = y++ | 5 |

# Datos Primitivos: Booleanos

Un **boolean** es un dato lógico que solo puede tener los valores true o false.

Este tipo de dato tiene muchas utilidades. Entre ellas nos permite cerciorarnos del resultado de una comparación, como por ejemplo si un número es igual a otro, o si es mayor o menor a otro, etc.

También nos permite programar en falso cuando algo está apagado de modo cambie a verdadero cuando está encendido y muchas cosas más.

Hay ciertos datos que tienden a verdadero (truthy= y otros que tienden a falso (falsy)..

```
var verdadero = true;
var falso = false;

var v = new Boolean(true)
var f = new Boolean(false)

console.log(verdadero)
console.log(falso)
console.log(v)
console.log(f)

console.log(new Boolean(""))
console.log(new Boolean("Hola mundo"))
console.log(new Boolean(0))
console.log(new Boolean(1))
```

# Operadores de comparación

En la última lección presentamos nuestros operadores de comparación, (> >= < <= === !==). Estos operadores funcionan como lo harían en una clase de matemáticas, mayor que, menor que, etc. Utilizamos estos operadores para evaluar dos expresiones. A medida que la computadora ejecuta el código, el operador devolverá un verdadero (si la declaración es verdadera) o un falso.

```
var mayorQue = 1 > 2;
var menorQue = 2 < 3;

var mayorOIgualQue = 10 >= 10;
var menorOIgualQue = 10 <= 10;

console.log(mayorQue);
console.log(menorQue);
console.log(mayorOIgualQue);
console.log(menorOIgualQue);
```

# Operadores de comparación

El "triple igual" (===) no debe confundirse con un solo signo igual (que indica asignar un valor a una variable). El triple igual comparará todo sobre los dos elementos, incluido el tipo, y devolverá si son exactamente iguales o no:

```
// igual que
var dobleIgual = 1 == 1;
var tripleIgual = 1 === 1;

console.log(dobleIgual);
console.log(tripleIgual);
```

```
// Distinto que
var distinto = 1 != 1;
var distinto2 = 1 !== 1;
```

# Tipos de Datos Primitivos: Not a Number

La propiedad global NaN es un valor que representa Not-A-Number (no es un número). Es un tipo de dato que se arrojará cada vez que se realice una operación involucré operadores numéricos pero que como resultado no pueda dar un nùmero.
También puede utilizarse intencionalmente para corroborar que el valor que se ingrese sea un número, pero esto lo último lo veremos más adelante.

```
var noEsUnNumero = "Hola" * 3.7;

console.log(noEsUnNumero);
```

# Tipos de datos primitivos: Nulo

El valor `null` es un literal de Javascript que representa intencionalmente un valor nulo o "vacío".

```
var noSeAsignoUnValor = null;

console.log(noSeAsignoUnValor);
```

# Tipos de datos primitivos: Indefinido

El valor undefined representa un valor que aún no se ha definido.
Una variable a la que no se le ha asignado valor, o no se ha declarado en absoluto (no se declara, no existe) son de tipo undefined. Un método o sentencia también devuelve undefined si la variable que se está evaluando no tiene asignado un valor. Una función devuelve undefined si no se ha devuelto un valor.

```
var noSeAsignoValor;

console.log(noSeAsignoValor);

console.log(nombre);
```

# Undefined y null

Non-zero value

null

0

undefined

# Tipos de datos compuestos: Funciones

Una **función** es un bloque de código, autocontenido, que se puede definir una vez y ejecutar en cualquier momento. Opcionalmente una función puede aceptar **parámetros** y devolver un **valor**.
Las funciones en JS son objetos, un tipo especial de objetos:
Se dice que las funciones son ciudadanos de primera clase porque pueden asignarse a un valor, y pueden pasarse como argumentos y usarse como un valor de retorno.

```
// Declaración de función
function miPrimeraFuncion() {
    console.log("1");
    console.log("2");
    console.log("3");
}

// Llamado a la función
miPrimeraFuncion();
```

```
// Declaración de función con parámetros
function suma(a, b) {
    return a + b;
}

// Llamado a la función
suma();
```

# Tipos de funciones

## Funciones **declaradas**:

Una función declarada puede invocarse en cualquier parte de nuestro código, incluso antes de que la función sea declarada. Una función declarada tiene la siguiente pinta:

```
function funcionDeclarada() {
    console.log("Hola Mundo")
}
```

## Funciones **expresadas**:

Una función expresada es un tipo de función que se le ha asignado como valor a una variable y que no puede invocarse antes de su definición.

Una función expresada sería algo así:

```
var funcionExpresada = function () {
    console.log("Adios Mundo")
}
```

ACADEMY
by NUMEN