# C20060_Andrik_Laboratorio2B

## ¿Cuáles son los data types que soporta javascript?

JS soporta estos 8 tipos básicos de datos.

- Number
- String
- Boolean
- Null
- Undefined
- BigInt
- Symbol
- Object

## ¿Cómo se puede crear un objeto en javascript? De un ejemplo

Primero declaramos es objeto y le ponemos atributos.

```
let Clase = {
    nombre: "PI",
    estudiantes: 20,
    universidad: "ucr"
};
Depués la podemos instanciar,

console.log(Clase.nombre);
console.log(Clase.estudiantes);
console.log(Clase.universidad);
```

## ¿Cuáles son los alcances (scope) de las variables en javascript?

Están el **Ambito Global:**
Son variables que se declaran fuera de cualquier función y pueden llegar a cualquier parte del código

**Ambito de Función/local:**
Estas van dentro de una función y solo tienen alcance dentro de la misma función.

**Ambito de Bloque:**
Se toma por ejemplo las variables `let` y `const` que tienen alcance por bloque, es decir, solo son accesibles dentro de las {} donde fueron declaradas.

**Ambito de Cadena:**
Este se refiere más a la forma de buscar una variable a través de diferentes scopes de manera jerarquica.

## ¿Cuál es la diferencia entre undefined y null?

**Undefined** : Es un valor que no se ha asignado a una variable declarada, lo pone JS por defecto.
**null**: Es un valor vacío que se asigna explicitamente con el objetivo de tener la variable inicializada.

## ¿Qué es el DOM?

Es una interfaz de programación que es compatible con HTML Y JS. Define sus estructura y la forma en la los lenguajes pueden acceder y modificar los mismos documentos.
Es también el que le da "nocion" a JS de la existencia de la instancia de las páginas web y XML, ya que el DOM presente varias nuevas formas que se encargan de la representación del mismo, tanto como las formas de guardarlo.

Osea, conecta páginas web con lenguajes de programación.

## Usando Javascript se puede acceder a diferentes elementos del DOM. ¿Qué

**hacen, que retorna y para qué funcionan las funciones getElement y querySelector? Cree un ejemplo**

Los getElement actuan de manera similar a un getter de toda la vida, se declara y agarra los elementos ya sea con el ID específico o con la etiqueta seleccionada.

Ejemplo

```
<div class="myClass">elm 1</div>
<div class="myClass">elm 2</div>
<div class="myClass">elm 3</div>
<script>
```

```
  let elementos = document.getElementsByClassName("miClase");
  console.log(elementos.length);  // Aquí imprime: 3 que es el numero de
elementos que exiten en myClass
</script>
```

El `querySelector(selector)` es un poco similar pero este selecciona el **primer** elemento que coincide con el selector CSS proporcionado ya sea `id`, una clase, un tipo de etiqueta.

Ejemplo,

```
<div class="myClass">Div 1</div>
<div class="myClass">Div 2</div>
<script>
  let primerDiv = document.querySelector(".myClass");
  console.log(primerDiv.textContent);  // Este imprime: "Div 1" es decir el
primer div de myClass`
</script>
```

# Investigue cómo se pueden crear nuevos elementos en el DOM usando

### Javascript. De un ejemplo

Se pueden crear elementos directamente en el dom utilizando **`createElement()`** y **`appendChild()`**. El primero crea un nuevo nodo elemento y el segundo a un nodo como hijo.

Ejemplo. Donde se accede a un HTML para crear un nuevo elemento de tipo li. Usado en el lab.

```
function agregar() {
    var lista = document.getElementById("lista");

    var elementoActual = lista.getElementsByTagName("li").length;
    var newElemento = document.createElement("li");

    newElemento.textContent = "Elemento" + (elementoActual + 1);
    lista.appendChild(newElemento);
}
```

# ¿Cuál es el propósito del operador this?

Hace referencia al **contexto de ejecución** o **objeto** en el que una función o método está siendo ejecutado. El valor que toma `this` depende de **cómo** y **dónde** se llama la función. Es decir, usualmente se usa para redeclarar el mismo atributo de objeto en la función pero depende del contexto

Ejemplo.

```
const persona = {
    nombre: "Andrik",
    saludar: function() {
        console.log("que tal, " + this.nombre);  // 'this' es persona
    }
};

persona.saludar();  // "que tal, Andrik"`
```

## ¿Qué es un promise en Javascript?De un ejemplo

Es una representación de la salida ya sea buena o a mala de una operación asíncrona.
Tiene los estados:

- *pending*: Inicial
- *fulfilled*: Completado
- *rejected*: rechazado.

Un ejemplo simple de una.

```
const promesa = new Promise((resolve, reject) => {
 //Se simula un retraso de 2 segundos
  setTimeout(() => {
    const exito = true;

    if (exito) {
      resolve("¡La operación fue exitosa!");
    } else {
      reject("Hubo un error en la operación.");
    }
  }, 2000);
});

// maneja la promesa con .then() y .catch()`
promesa
  .then((resultado) => {
    console.log(resultado);  // para exito
```

```
  })
  .catch((error) => {
    console.log(error);  // rechazada
  });
```

Sirve muy bien para gestionar los errores.

## ¿Qué es Fetch en Javascript? De un ejemplo

Es una API que se puede usar para enviar una solicitud a cualquier URL y tratar de recibir una respuesta.

```
fetch('<URL>', {})
```

Esto es implicitamente una promise, así que se debe manejar con then y catch acorde a la respuesta.

## ¿Qué es Asynch/Await en Javascript ? De un ejemplo

Sirve para trabajar con promesas de manera más cómoda.
**Asynch** es usado cuando una función siempre devolverá una promesa.

**Await** lo que hace es que se declara dentro de una función Asynch y espera a que esta ya sea resuelta para continuar.

Ejemplo.

```
async function f() {

  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("¡Hecho!"), 1000)
  });

  let result = await promise; // espera hasta que la promesa se resuelva (*)

  alert(result); // "¡Hecho!"
}

f();
```

## ¿Qué es un Callback? De un ejemplo

Es una función que se pasa a otra como argumento para poder completar una tarea.

```
function saludar(nombre) {
  console.log(`Hola, ${nombre}`);
}
function procesarUsuario(usuario, callback) {
  console.log("Procesando user");
  callback(usuario);  //  llama al callback
}
procesarUsuario("Rolo", saludar);
```

Es útil ya que permite ejecutar funciones aún cuando la tarea ya se ha completado.

## ¿Qué es Clousure?

Es cuando se crea una función dentro del scope de otra, por ejemplo:

```
function init() {
  var name = "Mozilla"; // name is a local variable created by init
  function displayName() {
    // displayName() is the inner function, that forms a closure
    console.log(name); // use variable declared in the parent function
  }
  displayName();
}
init();
```

Esto es útil ya que la función interna puede acceder a parametros de la función externa. También se da encapsulación y se pueden crear variables que no pueden ser accedidas desde afuera.

## ¿Cómo se puede crear un cookie usando Javascript?

Una cookie puede crearse así:

```
document.cookie = "usuario=Andrik; expires=Thu, 31 Dec 2025 12:00:00 UTC;
path=/";
```

- `usuario=Andrik` : El nombre de la cookie es "usuario" y su valor es "Andrik".
- `expires=Thu, 31 Dec 2025 12:00:00 UTC` : Expirará el 31 de diciembre de 2025 a las 12:00 PM.
- `path=/` : La cookie estará disponible para toda la aplicación.

Sirven básicamente para recordar información del usuario.

## ¿Cuál es la diferencia entre var, let y const?

**Var:** Tiene alcance global, se puede redeclarar y mutable. Es una opción general pero si es abusada puede llegar a generar confusión.
**let:** Alcance por bloque y no se puede reeclarar en el mismo ambito. Sirve para cuando se quiera una variable cuyo valor pueda cambiar,
**Const:** Esta tiene un scope de bloque, y no se puede redeclarar, es mejor usarlo cuando es un tipo fijo (que no se vaya a reasignar) o no cambiará o mutará mucho.

## Referencias.


https://brainstation.io/learn/javascript/what-is-javascript
https://www.w3schools.com/js//js_scope.asp
https://www.freecodecamp.org/news/scope-in-javascript-global-vs-local-vs-block-scope/
https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction
https://www-w3schools-com.translate.goog/jS/js_cookies.asp?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise
https://www.freecodecamp.org/espanol/news/javascript-fetch-api-para-principiantes/#como
https://es.javascript.info/async-await
https://developer.mozilla.org/es/docs/Glossary/Callback_function