Universidad de las Fuerzas Armadas ESPE

Integrantes: Danny Rodriguez
Pablo Tamayo
NRC: 10047

1 Test de JavaScript.

1.2 Variables y operaciones

1.2.1 Responde las siguientes preguntas en la sección de comentarios:

• ¿Qué es una variable y para qué sirve?

Es aquella que toma un nombre para ser utilizada para lo que necesitemos por ejemplo la variable nombre que puede ser usada para almacenar nombres de personas, animales, cosas, etc.

• ¿Cuál es la diferencia entre declarar e inicializar una variable?

La diferencia es que al declarar una variable se le da existencia a esta mientras que inicializar es darle un valor.

• ¿Cuál es la diferencia entre sumar números y concatenar strings?

Su diferencia es que el sumar números se trata de operaciones aritméticas es decir hace uso de valores numéricos, mientras que al concatenar strings es unir cadenas de texto con el fin de obtener un mensaje.

• ¿Cuál operador me permite sumar o concatenar?

El operador es: +

1.2.2 Determina el nombre y tipo de dato para almacenar en variables la siguiente información:

• Nombre: string
• Apellido: string
• Nombre de usuario en ESPE:string
• Edad:number
• Correo electrónico: string
• Mayor de edad: boolean
• Dinero ahorrado: number
• Deudas: number

1.2.3 Traduce a código JavaScript las variables del ejemplo anterior y deja tu código en los comentarios.

• Nombre: let nombre= "Pedro";

```
console.log(typeof nombre);
```
• Apellido:
```
let apellido= "Pascal";
console.log(typeof apellido);
```
• Nombre de usuario en ESPE:
```
let usr= "ppascal'";
console.log(typeof usr);
```
• Edad:
```
let edad= 25;
console.log(typeof edad);
```
• Correo electrónico:
```
let correo="ppascal@espe.edu.ec";
console.log(typeof correo);
```
• Mayor de edad:
```
let edad= 25;
let mayedad;
if(edad>=18){
    console.log(mayedad=true);
    console.log(typeof mayedad);
}else{
    console.log(mayedad=false);
    console.log(typeof mayedad);
}
```
• Dinero ahorrado:
```
let dinAho= 2500;
console.log(typeof dinAho);
```
• Deudas:
```
let deudas= 150;
console.log(typeof deudas);
```

1.2.4 Calcula e imprime las siguientes variables a partir de las variables del ejemplo anterior:

• Nombre completo (nombre y apellido)
```
let nombreCompleto="Karina Perez";
console.log(nombreCompleto);
```

• Dinero real (dinero ahorrado menos deudas)
```
let dineroReal=dinAho-deudas;
console.log(dineroReal);
```

1.3 Funciones

1.3.1 Responde las siguientes preguntas en la sección de comentarios:

• ¿Qué es una función?

Una función es aquella que se compone de un bloque de código permitiéndonos reutilizarla en donde creamos conveniente.

• ¿Cuándo me sirve usar una función en mi código?

Nos sirve hacerlas uso cuando tenemos código que vamos a necesitar varias veces es decir reutilizar, como ejemplo podemos mencionar operaciones matemáticas como la suma, la resta, multiplicación y división, etc.

• ¿Cuál es la diferencia entre parámetros y argumentos de una función?

La diferencia de los parámetros es que son variables que son definidas al declarar la función mientras que los argumentos son los valores que se le pasan a la función es decir datos.

1.3.2 Convierte el siguiente código en una función, pero, cambiando cuando sea necesario las variables constantes por parámetros y argumentos en una función:

```
const name = "Diego Medardo";
const lastname = "Saavedra García";
const completeName = name + lastname;
const nickname = "statick";

console.log("Mi nombre es "
+ completeName
+ ", pero prefiero que me digas "
+ nickname + ".");
```

Código

```
function datos(name,lastname,nickname){
   console.log("Mi nombre es "+name +" "+lastname+", "+ "pero prefiero que digas
"+nickname);
}

datos("Diego","Toapanta","Dieguinl");
```

1.4 Condicionales

1.4.1 Responde las siguientes preguntas en la sección de comentarios:

• ¿Qué es un condicional?

Es una estructura que nos permite controlar la ejecución de código si se cumple la condición que se especifique en esta condición.

• ¿Qué tipos de condicionales existen en JavaScript y cuáles son sus diferencias?

- **if:** Ejecuta la parte de código si la condición se cumple
- **else:** Permite ejecutar otras acciones del código si la condición no se cumple
- **else if:** Permite que se ejecute el código cuando se cumple una condición y otras instrucciones cuando no se cumple esa condición
- **? (operador ternario):** Es una forma reducida de escribir un condicional. Evalúa una condición y devuelve un resultado en función de esa condición
- **switch:**Se usa cuando se tiene varias opciones y se desea realizar diferentes acciones según el valor de una expresión en particular.

• ¿Puedo combinar funciones y condicionales?

Si se puede combinar funciones y condicionales para crear una lógica más compleja y controlar el flujo de ejecución de tu programa de manera flexible. Una de las opciones mediante las cuales se puede combinar son: llamar a una función dentro de un condicional y definir una función condicional

1.4.2 Replica el comportamiento del siguiente código que usa la sentencia switch utilizando if, else y else if:

```
const tipoDeSuscripcion = "Basic";

switch (tipoDeSuscripcion) {
    case "Free":
        console.log("Solo puedes tomar los cursos gratis");
        break;
    case "Basic":
        console.log("Puedes tomar casi todos los cursos de ESPE durante un mes");
        break;
    case "Expert":
        console.log("Puedes tomar casi todos los cursos de ESPE durante un año");
        break;
    case "ExpertPlus":
        console.log("Tú y alguien más pueden tomar TODOS los cursos
        de ESPE durante un año");
        break;
}
```

```
const tipoDeSuscripcion = "Basic";

if(tipoDeSuscripcion=="Free"){
  console.log("Solo puedes tomar los cursos gratis");
}else if(tipoDeSuscripcion=="Basic"){
  console.log("Puedes tomar casi todos los cursos de ESPE durante un mes");
}else if(tipoDeSuscripcion=="Expert"){
  console.log("Puedes tomar casi todos los cursos de ESPE durante un año");
}else if(tipoDeSuscripcion=="ExpertPlus"){
  console.log("Tú y alguien más pueden tomar TODOS los cursos de ESPE durante un
año");
}else{
  console.log("Ese tipo de subscripcion no existe");
}
```

1.4.3 Replica el comportamiento de tu condicional anterior con if, else y else if, pero ahora solo con if (sin else ni else if).

```
const tipoDeSuscripcion = "Expert";
```

```
if(tipoDeSuscripcion=="Free"){
  console.log("Solo puedes tomar los cursos gratis");
}if(tipoDeSuscripcion=="Basic"){
  console.log("Puedes tomar casi todos los cursos de ESPE durante un mes");
}if(tipoDeSuscripcion=="Expert"){
  console.log("Puedes tomar casi todos los cursos de ESPE durante un año");
}if(tipoDeSuscripcion=="ExpertPlus"){
  console.log("Tú y alguien más pueden tomar TODOS los cursos de ESPE durante un
año");
}
```

Bonus: si ya eres una experta o experto en el lenguaje, te desafío a comentar cómo replicar este comportamiento con arrays y un solo condicional.

```
const tipoDeSuscripcion = "Basic";

const opSus = [
  {
    tipo: "Free",
    mensaje: "Solo puedes tomar los cursos gratis"
  },
  {
    tipo: "Basic",
    mensaje: "Puedes tomar casi todos los cursos de ESPE durante un mes"
  },
  {
    tipo: "Expert",
    mensaje: "Puedes tomar casi todos los cursos de ESPE durante un año"
  },
  {
    tipo: "ExpertPlus",
    mensaje: "Tú y alguien más pueden tomar TODOS los cursos de ESPE durante un año"
  }
];

const susp = opSus.find(opcion => opcion.tipo === tipoDeSuscripcion);

if (susp) {
  console.log(susp.mensaje);
} else {
  console.log("Tipo de suscripción no válido");
}
```

1.5 Ciclos

1.5.1 Responde las siguientes preguntas en la sección de comentarios:

• ¿Qué es un ciclo?
Es una estructura de control que permite repetir un bloque de código varias veces hasta que se cumpla una condición específica. Su principal uso es cuando se necesita realizar una tarea repetitiva y esto nos permite realizarla sin tener que escribir el mismo código una y otra vez.
• ¿Qué tipos de ciclos existen en JavaScript?

- **for:** se repite hasta que una condición especificada se evalúe como false
- **While:** ejecuta sus instrucciones siempre que una condición especificada se evalúe como true
- **Do … while:** se repite hasta que una condición especificada se evalúe como falsa

• ¿Qué es un ciclo infinito y por qué es un problema?

Es un ciclo que no tiene fin y no llega a una condición de salida y es un problema porque puede causar que nuestro equipo colapse, el programa nunca termine.

• ¿Puedo mezclar ciclos y condicionales?

Si se puede mezclar

1.5.2 Replica el comportamiento de los siguientes ciclos for utilizando ciclos while:

```
for (let i = 0; i < 5; i++) {
     console.log("El valor de i es: " + i);
}

//Replicando con ciclo while
let i=0;
while(i<5){
   console.log("El valor de i es: "+i);
   i++;
}

for (let i = 10; i >= 2; i--) {
     console.log("El valor de i es: " + i);
}

//Replicando con ciclo while
let j=10;
while(j>=2){
   console.log("El valor de j es: "+j);
   j--;
}
```

1.5.3 Escribe un código en JavaScript que le pregunte a los usuarios cuánto es 2 + 2. Si responden bien, mostramos un mensaje de felicitaciones, pero si responden mal, volvemos a empezar.

Pista: puedes usar la función prompt de JavaScript.

```
let ingreso;
do{
   ingreso=prompt("Ingrese un numero:")
   console.log(ingreso);
}while(ingreso!=4)
```

1.6 Listas
1.6.1 Responde las siguientes preguntas en la sección de comentarios:

• ¿Qué es un array?
Es una estructura de datos que se utiliza para almacenar y organizar varios elementos en una sola variable. Son objetos especiales que tienen propiedades y métodos específicos para trabajar con los elementos que contienen. Los elementos dentro de un array se almacenan en posiciones numeradas llamadas índices. El primer elemento se encuentra en el índice 0, el segundo en el índice 1 y así sucesivamente.

• ¿Qué es un objeto?
Es una entidad que agrupa datos (propiedades) y funcionalidad (métodos) relacionados en una estructura organizada. Se puede pensar en un objeto como una representación de un concepto o entidad del mundo real que tiene características y acciones asociadas.

• ¿Cuándo es mejor usar objetos o arrays?
Algunos casos de uso de objetos puede ser cuando:

- Se requiere almacenar datos con una estructura clave-valor, donde cada propiedad tiene un nombre descriptivo y único que actúa como una clave.
- Los datos tienen propiedades adicionales o métodos asociados que necesitas utilizar o acceder.
- Quieres agrupar y organizar datos relacionados de manera lógica.

Algunos casos de uso de arrays puede ser cuando:

- Si se necesita almacenar y acceder a una colección ordenada de elementos.
- No se necesitan claves o nombres específicos para los elementos y solo te importa el orden en que se almacenan.
- Si se quiere realizar operaciones comunes en los elementos del array, como agregar, eliminar o buscar elementos.

• ¿Puedo mezclar arrays con objetos o incluso objetos con arrays?
Si se puede mezclar

1.6.2 Crea una función que pueda recibir cualquier array como parámetro e imprima su primer elemento.

```
function mostrarPrimerElemento(array) {
   if (array.length > 0) {
     let primerElemento = array[0];
     console.log("El primer elemento del arreglo es:", primerElemento);
   }
 }

 var array = [90, 100, 110, 120, 130];
 mostrarPrimerElemento(array);
```

1.6.3 Crea una función que pueda recibir cualquier array como parámetro e imprima todos sus elementos uno por uno (no se vale imprimir el array completo).

```
function mostrarElementos(array) {
          array.forEach(function(elemento) {
            console.log(elemento);
          });
 }
let array = [11.5, 400, 33, 9, 105];
mostrarElementos(array);
```

1.6.4 Crea una función que pueda recibir cualquier objeto como parámetro e imprima todos sus elementos uno por uno (no se vale imprimir el objeto completo).

```
function mostrarElementosObjeto(objeto) {
          for (var clave in objeto) {
            if (objeto.hasOwnProperty(clave)) {
                    console.log(clave + ": " + objeto[clave]);
            }
          }
 }

var miObjeto = { nombre: "Mishell", edad: 26, ciudad: "Guayaquil" };
mostrarElementosObjeto(miObjeto);
```