JavaScript Basics

Breve repaso de JavaScript

1. Scope de una variable.
2. Funciones personalizadas: tipología, etapas, argumentos y retorno.
3. Objetos literales.

Scope de una variable en JavaScript.

Dependiendo del lugar donde se declare una variable, ésta puede ser temporal, ya sea dentro de un bloque o de una función.

En programación la zona en la que está activa una variable se llama ***ámbito*** o ***alcance(scope)***.

Clasificación de el scope de una variable en JavaScript.

- Global scope.
- Function scope.
- Block scope

Function scope.

Solo se puede acceder a ellas desde dentro de la función.

```
2  // Aquí no puedes usar la variable carName
3
4  function myFunction() {
5      let carName = "Volvo";
6      // Aquí sí puedes usar la variable carName
7    }
8
9    // Aquí no puedes usar la variale carName
10
```

Global scope.

Todos los scripts y funciones de una página web pueden acceder a ella.

```
2  let carName = "Volvo";
3  // Aquí sí puedes usar carName
4
5  function myFunction() {
6  // Aquí también puedes usar carName
7  }
8
```

Block scope.

- Antes de ES6 (2015), JavaScript solo tenía **Global Scope** y **Function Scope**.
- ES6 introdujo dos nuevas e importantes palabras reservadas de JavaScript: **let** y **const**.
- Estas dos palabras clave proporcionan **Block Scope** en JavaScript.
- No se puede acceder a las variables declaradas dentro de un bloque { } desde fuera del bloque.

Veamos un ejemplo.

Funciones personalizadas en JavaScript.

Las funciones de JavaScript se definen con la palabra reservada function.

Puede utilizar una declaración de función o una expresión de función.

```
15  function nombreFuncion(parametros) {
16      // Código que será ejecutado
17      // cada que mandemos a llamar
18      //la función.
19  }
```

Veamos algunos ejemplos.
(Incluso mandemos a imprimir la estructura de una función.)

Ejercicios.

1. Crear una función que dado el peso y la altura de una persona imprima en la ventana del navegador su IMC.

2. Hacer una función que reciba un monto y si este excede los $1000 se aplica un descuento de 7%, imprimir el monto final en la ventana del navegador.

Expresiones de función.

Una función de JavaScript también se puede definir usando una expresión .

Una expresión de función se puede almacenar en una variable:

```
21
22  var x = function(a){ return a*a};
23
```

La función anterior es en realidad una función anónima (una función sin nombre).

Veamos algunos ejemplos.

Ejercicio.

1. Crear una expresión de función que reciba una cantidad de segundos y la transforme a su equivalente en minutos con segundos. Ejemplo t(215) e imprime: Minutos 3, Segundos: 35.

Funciones autoinvocadas.

Una función se puede ser llamada en automático sin necesidad de escribir la llamada.

```
26 (function () {
27     let x = "Hola!!";  // Este mensaje sale en automatico.
28     document.write("<br> --> "+x);
29 })();
```

Probemos el ejemplo.

Ejercicio.

1. Crear una función autoinvocada que imprima un mensaje deseando buen día, con el día exacto en texto, ejemplo: “Excelente martes para ti”.

Funciones flecha.

Las funciones de flecha se introdujeron en ES6.

Las funciones de flecha nos permiten escribir una sintaxis de función más corta:

```
1
2  let potencia = (a,b)=>a**b;
3
```

Probemos algunos ejemplos.

Ejercicio.

1. Crear una función flecha que reciba el nombre de una persona e imprima un saludo.

2. Crear una función flecha que imprima la hora actual.

Objetos literales.

Esta es la forma más fácil de crear un objeto JavaScript. Usando un objeto literal, usted define y crea un objeto en una declaración. Se aceptan saltos de línea.

```
12
13  const persona = {nombre:"Reus", apellido:"Campos", edad: 30};
14
```

Hagamos un ejemplo.