Desarrollo Web en Entorno Cliente

FUNCIONES–

Funciones Básicas:

Las funciones son bloques de código que realizan tareas específicas.

Se definen con la palabra clave function.

Declaración de función:

```
function multipli( ) {
return 3*4;
}
multipli( );
```

Expresión de función:

```
const multipli = function( ) {
return 3*4;
}
multipli( );
```

Pueden tener parámetros que actúan como variables locales dentro de la función.

Pueden devolver un valor usando return.

```
function suma (a, b) {
return a + b;
}
suma (2,5)
```

*** Funciones y métodos**

- console.log(parseInt(prompt("edad"))) **/ Función: independiente**
- console.log(prompt("edad").toString()) **/ Método : asociada a un objeto**

Funciones Anónimas:

Las funciones anónimas son funciones sin nombre.

Se pueden asignar a variables o utilizar como argumentos de otras funciones.

```
const miFuncion = function(x) {
return x * 2;
};
```

```
console.log( function ( ) {
return "Estoy haciendo algo";
}( ));
```

Funciones con parámetros

```
const sumar = function( a,b) {
return a+b;
}
sumar (2, 4 );
```

Funciones con parámetros por defecto

```
const saludar = function( nombre="Desconocido", apellidos) {
console.log();
}
saludar ( );
```

Funciones Flecha (Arrow Functions):

Proporcionan una sintaxis más concisa para definir funciones.

```
const fn1= ( ) =>{ }
const fn2 = param =>param;
const fn3 = (param1,param2) => { ....}
```

Funciones de Orden Superior:

Pueden aceptar funciones como argumentos o devolver funciones como resultados.

Ejemplos **map, filter, reduce**, que operan en arrays.

Funciones Constructoras:

Utilizadas para crear objetos a partir de un prototipo.

Se definen con la palabra clave function y se llaman con new.

```
functionPersona(nombre, edad) {
this.nombre = nombre;
this.edad = edad;
}
```

Funciones Callback:

Se pasan como argumentos a otras funciones.

```
functionmostrarResultado(resultado) {
console.log(`El resultado es: ${resultado}`);
}
realizarOperacion(5, 3, mostrarResultado);
```

Se llaman de vuelta cuando se completa una tarea, como una operación asíncrona.

Clausuras (Closures):

Son funciones que "recuerdan" su entorno léxico.

Pueden acceder a variables fuera de su alcance.

Métodos de Función:

bind(): Cambia el valor de this en una función.

call(): Invoca una función con un contexto específico.

apply(): Invoca una función con argumentos en un arreglo.

Funciones Asíncronas:

Se utilizan **async** y **await** para manejar operaciones asíncronas de manera más legible.

Las promesas son comunes en operaciones asíncronas.