Fundamentos de programación

0.7 Funciones

0.7.3 Parámetros por valor y por referencia

Funciones Puras
Una función pura es una función que:

- Toma uno o más valores de entrada.
- Devuelve un resultado sin modificar el estado del sistema.
- No produce efectos colaterales.

Ejemplo de Función Pura:

```
function sumar(a, b) {
    return a + b;
}
```

En el ejemplo anterior, la función **sumar** no altera los valores de **a** y **b**, solo devuelve su suma.
Funciones Impuras
Una función impura es una función que:

- Puede modificar el estado del sistema.
- Tiene efectos colaterales, lo que puede complicar el proceso de depuración.

Ejemplo de Función Impura:

```
function ordenarArray(array) {
    return array.sort();
}
```

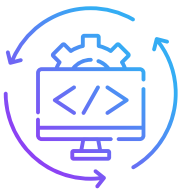
Esta función modifica el array original, alterando su orden.
Diferencias Principales

1. **Estado del Sistema:**
   - Las funciones puras no lo modifican.
   - Las funciones impuras sí lo modifican.
2. **Efectos Colaterales:**
   - Las funciones puras no tienen efectos colaterales.
   - Las funciones impuras tienen efectos colaterales.

Ejemplo Detallado
Función Impura:

```
let array = [3, 1, 2];
function ordenarArray(array) {
    array.sort();
    return array;
}
console.log(ordenarArray(array)); // [1, 2, 3]
console.log(array); // [1, 2, 3]
```

La función **ordenarArray** modifica el array original.
Transformación a Función Pura:



0.7.3 Parámetros por valor y por referencia

```
let array = [3, 1, 2];
function ordenarArrayPuro(array) {
    let otroArray = [...array];
    otroArray.sort();
    return otroArray;
}
let arrayOrdenado = ordenarArrayPuro(array);
console.log(array); // [3, 1, 2]
console.log(arrayOrdenado); // [1, 2, 3]
```

La función **ordenarArrayPuro** crea una copia del array original y ordena la copia, manteniendo el array original sin cambios.

Importancia de las Funciones Puras

- **Depuración:** Las funciones puras facilitan la identificación de errores porque no tienen efectos colaterales.
- **Reusabilidad:** Son más predecibles y reutilizables.
- **Mantenimiento:** Mejoran la legibilidad y el mantenimiento del código.

Nombres Significativos en Funciones

Usar nombres significativos es crucial:

- Facilita la comprensión del código.
- Ayuda a recordar la funcionalidad después de un tiempo.
- Mejora la colaboración en equipo.

Ejemplo:

```
function ordenarArrayDeEnteros(array) {
    // código
}
```

Un nombre descriptivo como **ordenarArrayDeEnteros** es preferible a un nombre genérico como **ordenar**.