



# Assessment para Desarrollador Junior

---



## Parte 1 – Evaluación escrita (40 minutos)

### 1. Suma más cercana (2 puntos)

Escribe una función en JavaScript que reciba un arreglo de enteros no ordenados y un número objetivo X.

La función debe retornar todos los pares de números cuya suma sea la más cercana posible a X **sin superarlo**.

**Ejemplo:**

```
javascript
const números = [3, 12, 5, 1, 20, 18];
const x = 23;
// Resultado esperado: [[5, 18], [3, 20]]
```

---

### 2. Detección de anagramas con limpieza de caracteres (2 puntos)

Escribe una función que determine si dos cadenas de texto son anagramas, ignorando espacios, signos y diferencias entre mayúsculas y minúsculas.

**Ejemplo:**

"Elvis vive" y "Es evil Levi" → `true`

---

### 3. Inversión de arreglo sin alterar símbolos especiales (2 puntos)

Escribe una función que reciba un arreglo de caracteres y devuelva el mismo arreglo invertido, **sin mover los caracteres especiales**.

**Ejemplo:**

Entrada: `["h", "t", "@", "#", "u", "f"]`

Resultado esperado: `["f", "u", "@", "#", "t", "h"]`

---

#### 4. Análisis de texto desde HTML (2 puntos)

Dado un campo de texto HTML, escribe una función JavaScript que, al hacer clic en un botón, muestre:

- Total de palabras
  - Primera palabra
  - Última palabra
  - Oración invertida
- 

#### 5. Validación lógica de condiciones múltiples (2 puntos)

Completa el siguiente fragmento de código para que devuelva "Acceso permitido" solo si:

- El usuario es mayor de 21 años
- Ha verificado su correo
- No está bloqueado

javascript

```
function validarAcceso(usuario) {  
    if (_____) {  
        return "Acceso permitido";  
    } else {  
        return "Acceso denegado";  
    }  
}
```

---

---



## Parte 2 – Proyecto para desarrollar en casa (hasta 24 horas)




### Objetivo:

Evaluar habilidades prácticas, dominio básico de HTML/JS, validación de formularios, manipulación del DOM, lógica implementada y buenas prácticas en el uso de GitHub.



### Instrucciones generales:

- Crea un repositorio en GitHub llamado:  
`evaluacion-dev-junior`
- Incluye los siguientes archivos:
  - `index.html`
  - `scripts.js`
  - `README.md` (con instrucciones de uso)
- Envía el enlace del repositorio al correo electrónico:  
 [felipe.ruiz@qsystems.com.co](mailto:felipe.ruiz@qsystems.com.co)



---

### Ejercicio A: Formulario inteligente de registro

Construye un formulario HTML con campos para:

- Nombre
- Correo electrónico
- Contraseña

### Requisitos:

1. Validar contraseña (mínimo 8 caracteres, al menos 1 mayúscula y 1 número)
2. Validar que el correo tenga un formato válido
3. Mostrar en pantalla los datos (sin la contraseña) y la contraseña encriptada (usando Cifrado César, rotación 3)



## Ejercicio B: Analizador de texto

Desde una caja de texto con botón:

- Mostrar número de palabras
  - Mostrar la palabra más larga y la más corta
  - Listar palabras únicas ordenadas alfabéticamente
- 



## Ejercicio C (opcional +1 punto): Temporizador interactivo

- Permitir ingresar un número de segundos
  - Iniciar una cuenta regresiva visible
  - Al finalizar, mostrar el mensaje:  
👉 “¡Tiempo finalizado!”
- 



## Ejercicio D (opcional +2 puntos): Buscador inteligente de coincidencias

### Objetivo:

Crear una miniaplicación de búsqueda interactiva en una lista simulada de usuarios.

### Requisitos:

1. Simula 20 usuarios con los campos:
  - `nombreCompleto`, `correo`, `rol` (admin, user, guest), `estado` (activo/inactivo)
2. Crea un campo de búsqueda + selector de rol.
  - Filtra resultados que coincidan parcialmente en nombre o correo
  - Filtra por tipo de rol
3. Muestra los resultados en una tabla:

- Sombrear filas de usuarios inactivos
- Mostrar conteo de coincidencias

**Bonus (+0.5):** Agregar ordenamiento ascendente/descendente por nombre o correo

---

### **Ejercicio E (opcional +2 puntos): Editor de tareas con historial**

#### **Objetivo:**

Desarrollar un gestor de tareas que permita revertir cambios de edición.

#### **Requisitos:**

1. Permitir agregar tareas (título + descripción)
2. Permitir editar las tareas
  - Antes de cada edición, guardar la versión anterior
  - Mostrar el historial de cambios por tarea
3. Hacer posible revertir una tarea a cualquier versión anterior
4. Implementar interfaz clara con JS puro

**Bonus (+0.5):** Permitir exportar/importar historial como archivo JSON

### **Criterios de evaluación:**

Criterio	Peso
Correctitud y lógica	40%
Claridad y buenas prácticas	20%
HTML y JS correctos	15%
Validaciones funcionales	15%
Organización del repositorio	10%