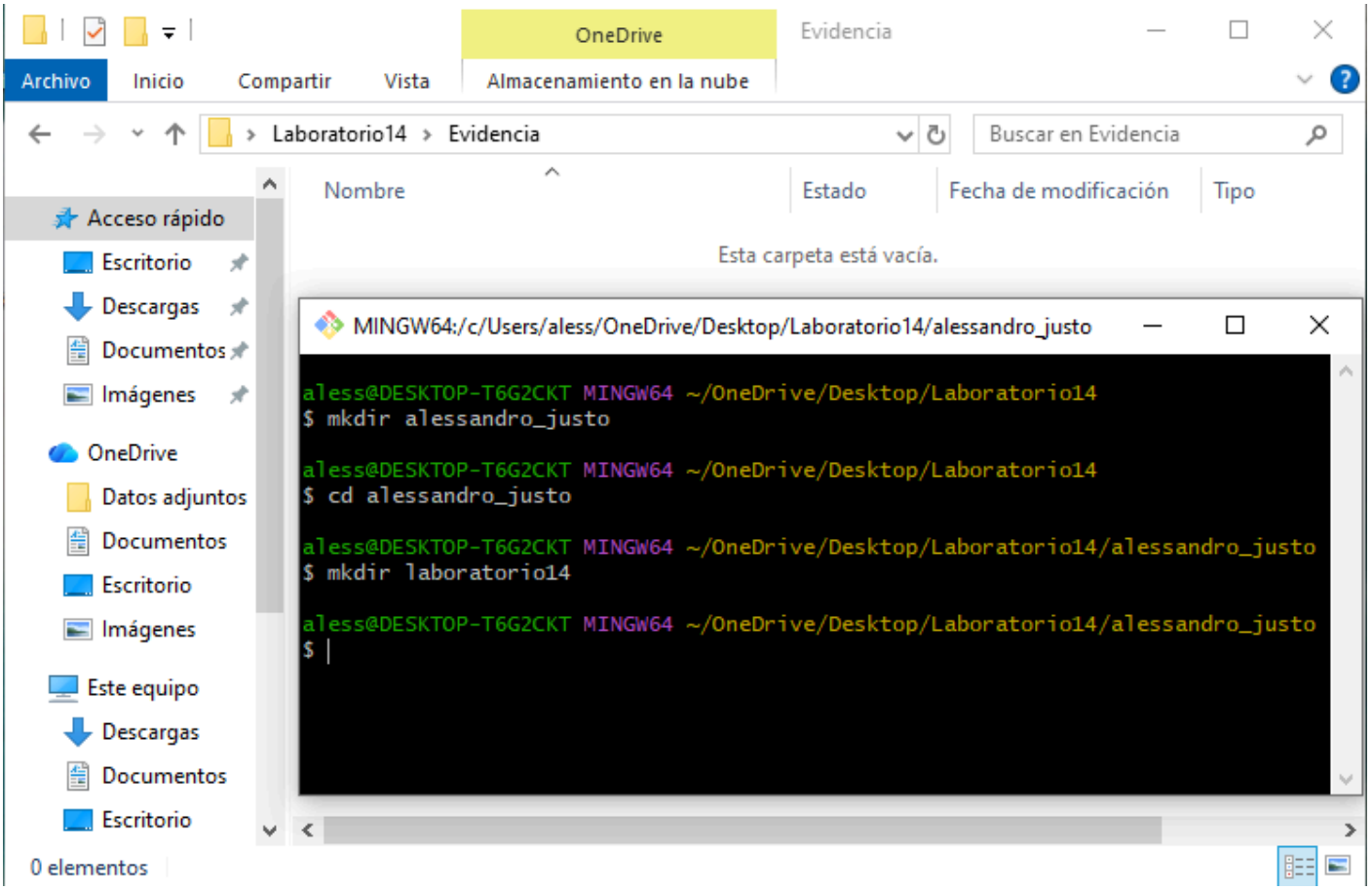
Universidad Nacional de San Agustín
Facultad de Ingeniería
Escuela Profesional de Ingeniería de Sistemas

Curso: Introducción al Desarrollo Web Laboratorio
Grupo: "D"
Docente: Carlo Jose Luis Corrales Delgado
Título de la Actividad: Laboratorio 14 – Javascript: Orientación a Objetos
Estudiante: Alessandro Josue Justo Vilca

2025

1. **Crear un directorio que tenga su nombre y un subdirectorio laboratorio14. Recomendación: usar minúsculas, sin espacios, sin tildes ni "ñ" y con guiones medios o bajos**
   // Laboratorio Nro 14 - Ejercicio1
   // Autor: Alessandro Josue Justo Vilca
   // Colaboró : sin colaboradores
   // Tiempo : 1min



2. **Utilizar los atajos de teclado o combinación de teclas para agilizar su trabajo**
   // Laboratorio Nro 14 - Ejercicio2
   // Autor: Alessandro Josue Justo Vilca
   // Colaboró : sin colaboradores
   // Tiempo : 1min
   Atajos de teclado que use y tuve en cuenta en VS Code:
   Ctrl + Ñ: abre la terminal integrada.
   Ctrl + /: comentar línea seleccionada.
   Alt + ↑: mover línea hacia arriba.
   Shift + Alt + ↓: duplicar línea.
   Atajos de teclado útiles (recomendados)
   Ctrl + S → guardar
   Ctrl + C / Ctrl + V → copiar / pegar
   Ctrl + Z / Ctrl + Y → deshacer / rehacer
   Ctrl + F → buscar texto en archivo
   Ctrl + Shift + F → buscar en todo el proyecto
   Ctrl + P → abrir archivo por nombre (VSCode)
   Ctrl + ` → abrir terminal integrado (VSCode)
   Ctrl + Shift + T → reabrir pestaña cerrada (navegador)
   Alt + Tab → cambiar de aplicación
3. **Redondeo de precios. Pide un número decimal que represente el precio de un producto y que muestre:**
   • Redondeo hacia abajo

• Redondeo hacia arriba
• Redondeo normal
Tip: prueba con el número 12.49 y 12.5
// Laboratorio Nro 14 - Ejercicio3
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 20min
HTML de ejecución de codigos:

index.html × ejercicio03.js

index.html > ...

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Laboratorio 14</title>
</head>
<body>
    <main>
        <h1>Laboratorio 14</h1>

        <section id="ej3">
            <h2>3. Redondeo de precios</h2>
            <button id="run-ej3">Ejecutar</button>
            <pre id="out-ej3"></pre>
            <script src="js/ejercicio03.js"></script>
        </section>

        <section id="ej4">
            <h2>4. Número aleatorio en un rango</h2>
            <button id="run-ej4">Ejecutar</button>
            <pre id="out-ej4"></pre>
            <script src="js/ejercicio04.js"></script>
        </section>

        <section id="ej5">
            <h2>5. Lanzamiento de dados</h2>
            <button id="run-ej5">Ejecutar</button>
            <pre id="out-ej5"></pre>
            <script src="js/ejercicio05.js"></script>
        </section>

        <section id="ej6">
            <h2>6. Potencias y raíces</h2>
            <button id="run-ej6">Ejecutar</button>
            <pre id="out-ej6"></pre>
            <script src="js/ejercicio06.js"></script>
        </section>

        <section id="ej7">
            <h2>7. Grados ↔ Radianes</h2>
            <button id="run-ej7">Ejecutar</button>
            <pre id="out-ej7"></pre>
            <script src="js/ejercicio07.js"></script>
        </section>
```

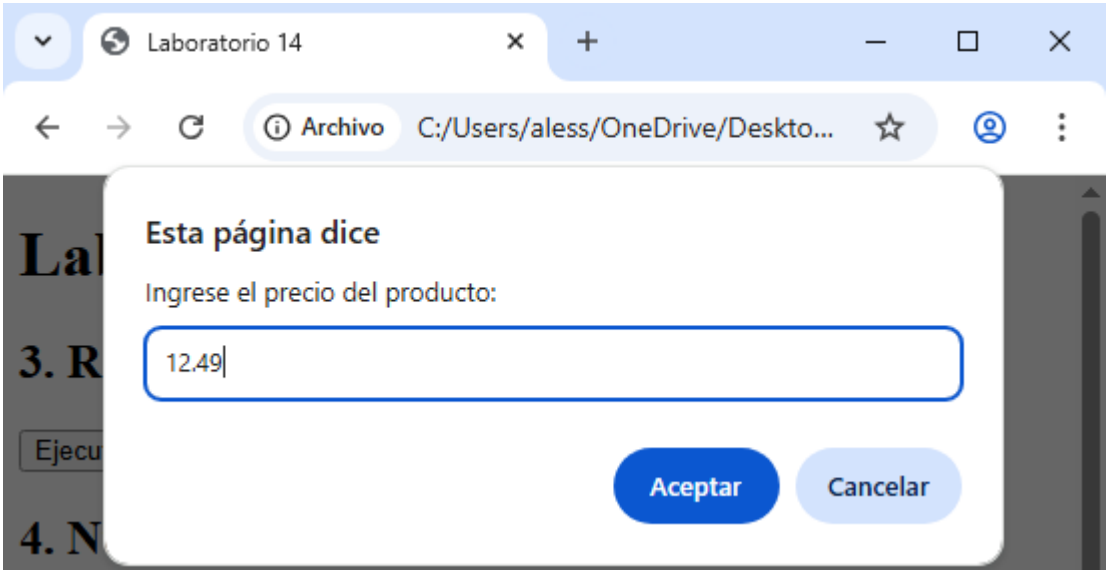
```
46
47          <section id="ej8">
48              <h2>8. Contraseña numérica de 6 dígitos</h2>
49              <button id="run-ej8">Ejecutar</button>
50              <pre id="out-ej8"></pre>
51              <script src="js/ejercicio08.js"></script>
52          </section>
53
54          <section id="ej9">
55              <h2>9. Distancia entre dos puntos</h2>
56              <button id="run-ej9">Ejecutar</button>
57              <pre id="out-ej9"></pre>
58              <script src="js/ejercicio09.js"></script>
59          </section>
60
61          <section id="ej10">
62              <h2>10. Normalización de calificaciones</h2>
63              <button id="run-ej10">Ejecutar</button>
64              <pre id="out-ej10"></pre>
65              <script src="js/ejercicio10.js"></script>
66          </section>
67
68          <section id="ej11">
69              <h2>11. Clase Producto (encapsulación)</h2>
70              <button id="run-ej11">Ejecutar</button>
71              <pre id="out-ej11"></pre>
72              <script src="js/ejercicio11.js"></script>
73          </section>
74
75          <section id="ej12">
76              <h2>12. Producto con formato de precio</h2>
77              <button id="run-ej12">Ejecutar</button>
78              <pre id="out-ej12"></pre>
79              <script src="js/ejercicio12.js"></script>
80          </section>
81
```

```
        <section id="ej13">
            <h2>13. Figura - Cuadrado - Triángulo</h2>
            <button id="run-ej13">Ejecutar</button>
            <pre id="out-ej13"></pre>
            <script src="js/ejercicio13.js"></script>
        </section>

        <section id="ej14">
            <h2>14. Usuario - Cliente - Administrador</h2>
            <button id="run-ej14">Ejecutar</button>
            <pre id="out-ej14"></pre>
            <script src="js/ejercicio14.js"></script>
        </section>

        <section id="ej15">
            <h2>15. Polimorfismo lista de usuarios</h2>
            <button id="run-ej15">Ejecutar</button>
            <pre id="out-ej15"></pre>
            <script src="js/ejercicio15.js"></script>
        </section>

        <section id="ej16">
            <h2>16. Empleado - Programador - Senior</h2>
            <button id="run-ej16">Ejecutar</button>
            <pre id="out-ej16"></pre>
            <script src="js/ejercicio16.js"></script>
        </section>

        <section id="ej17">
            <h2>17. Cuenta - Ahorro - Corriente</h2>
            <button id="run-ej17">Ejecutar</button>
            <pre id="out-ej17"></pre>
            <script src="js/ejercicio17.js"></script>
        </section>

        <section id="ej18">
            <h2>18. Carrito de compras (composición)</h2>
            <button id="run-ej18">Ejecutar</button>
            <pre id="out-ej18"></pre>
            <script src="js/ejercicio18.js"></script>
        </section>

        <section id="ej19">
            <h2>19. Notificaciones (polimorfismo)</h2>
            <button id="run-ej19">Ejecutar</button>
            <pre id="out-ej19"></pre>
            <script src="js/ejercicio19.js"></script>
        </section>

    </main>
</body>
</html>
```

index.html ejercicio03.js

js > ejercicio03.js > ...

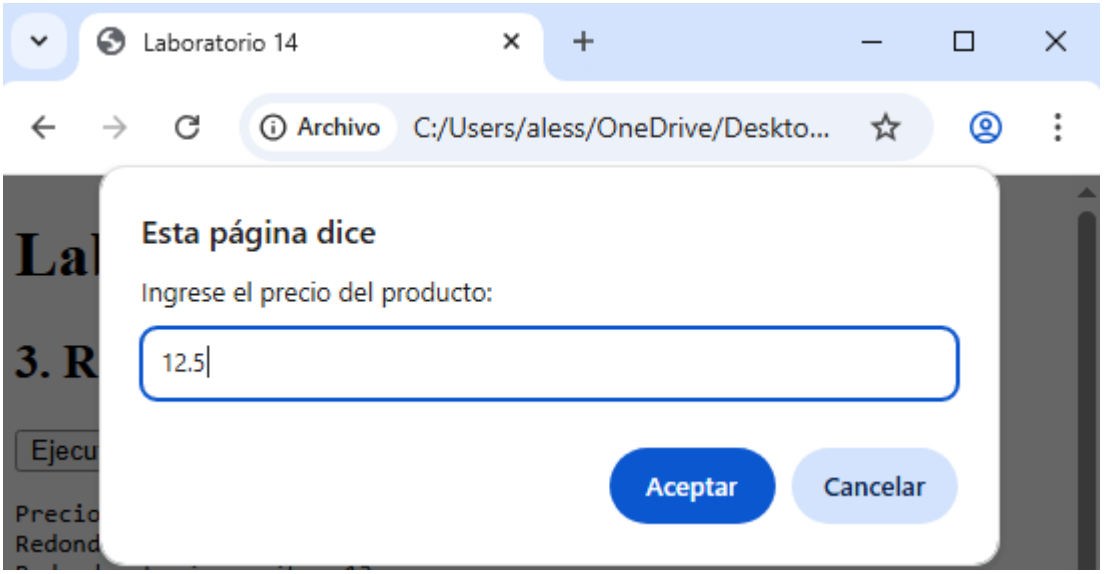
```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej3');
    const out = document.getElementById('out-ej3');

    btn.addEventListener('click', () => {
        let precio = parseFloat(prompt("Ingrese el precio del producto: "));

        let abajo = Math.floor(precio);
        let arriba = Math.ceil(precio);
        let normal = Math.round(precio);

        out.textContent =
            `Precio original: ${precio}
            Redondeo hacia abajo: ${abajo}
            Redondeo hacia arriba: ${arriba}
            Redondeo normal: ${normal}`;
    });
});
```

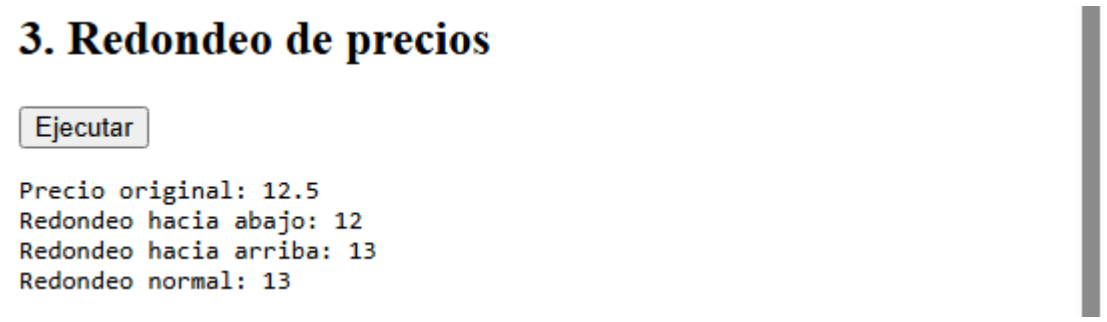


3. Redondeo de precios

Ejecutar

```
Precio original: 12.49
Redondeo hacia abajo: 12
Redondeo hacia arriba: 13
Redondeo normal: 12
```



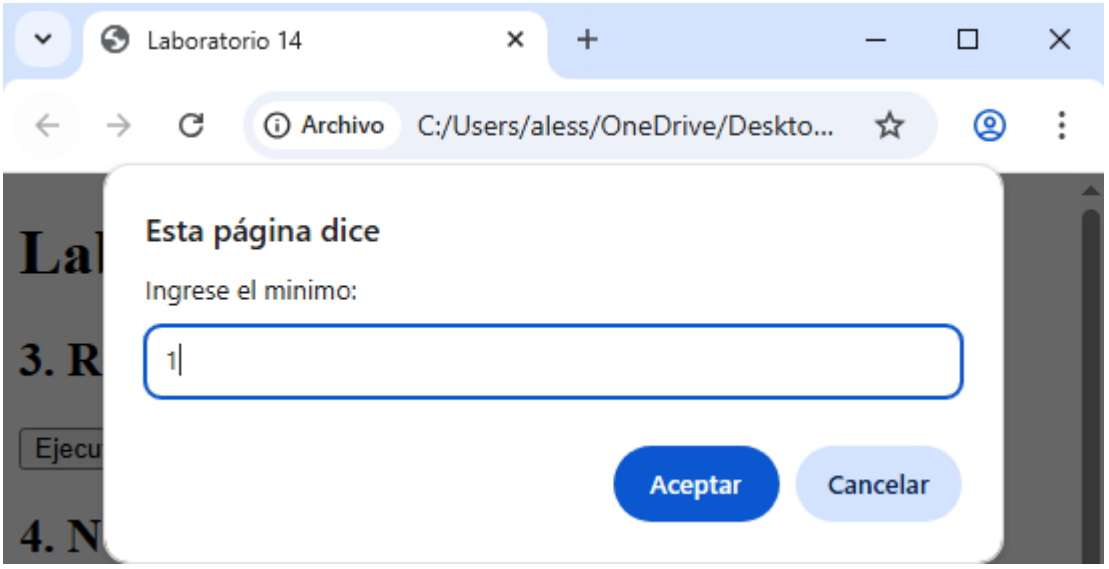


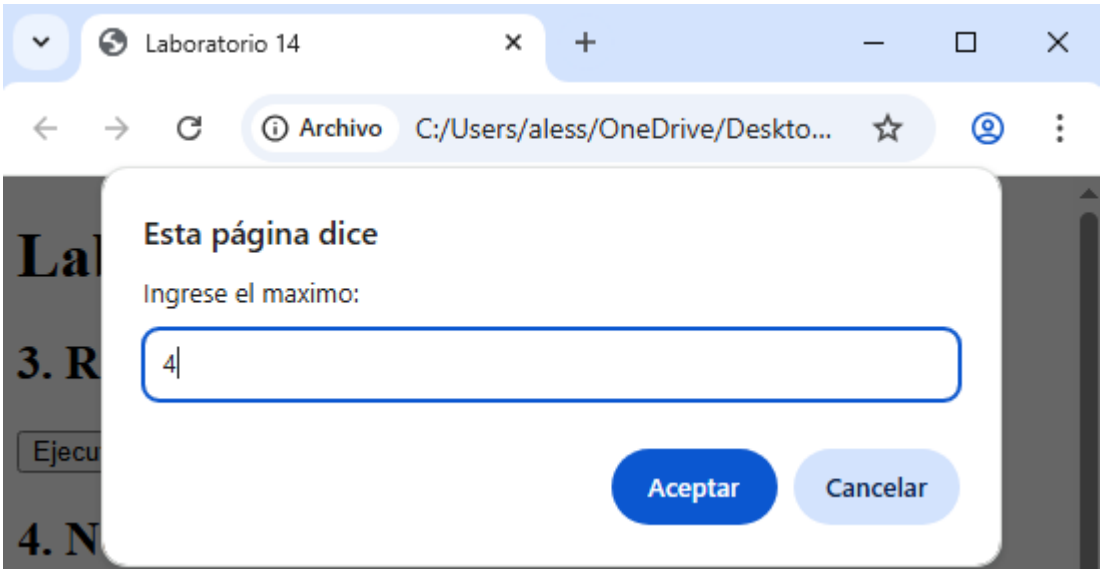
4. **Número aleatorio en un rango. Crear una función numeroAleatorio(min, max) que devuelva un número entero entre min y max (incluidos)**

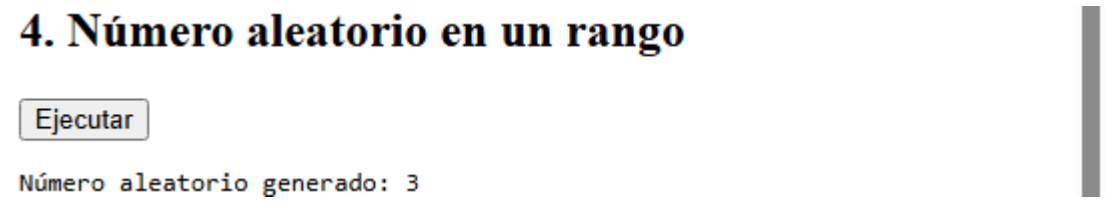
// Laboratorio Nro 14 - Ejercicio4
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 12min

index.html ejercicio04.js

```
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej4');
    const out = document.getElementById('out-ej4');

    function numeroAleatorio(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    btn.addEventListener('click', () => {
        let min = parseInt(prompt("Ingrese el minimo: "));
        let max = parseInt(prompt("Ingrese el maximo: "));

        let resultado = numeroAleatorio(min, max);

        out.textContent = `Número aleatorio generado: ${resultado}`;
    });
});
```







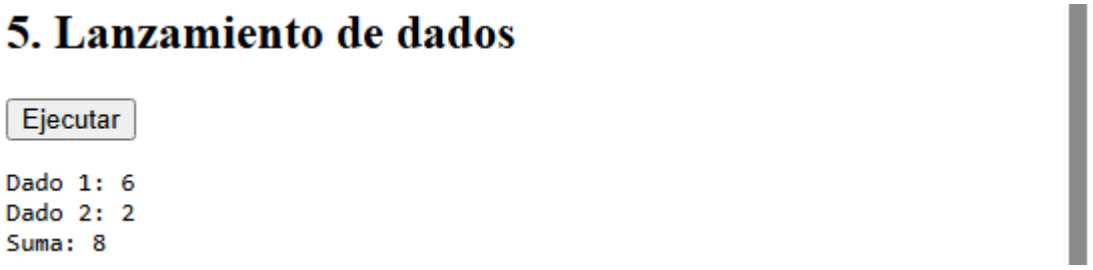
5. **Lanzamiento de dados. Simula el lanzamiento de dos dados (valores del 1 al 6) y muestra su suma. Tip: reutiliza la función del ejercicio anterior**

```
// Laboratorio Nro 14 - Ejercicio5
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 13min
```

index.html | ejercicio | ejercicio04.js

js > ejercicio05.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej5');
    const out = document.getElementById('out-ej5');

    function numeroAleatorio(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    btn.addEventListener('click', () => {
        let dado1 = numeroAleatorio(1, 6);
        let dado2 = numeroAleatorio(1, 6);
        let suma = dado1 + dado2;

        out.textContent = `Dado 1: ${dado1}\nDado 2: ${dado2}\nSuma: ${suma}`;
    });
});
```



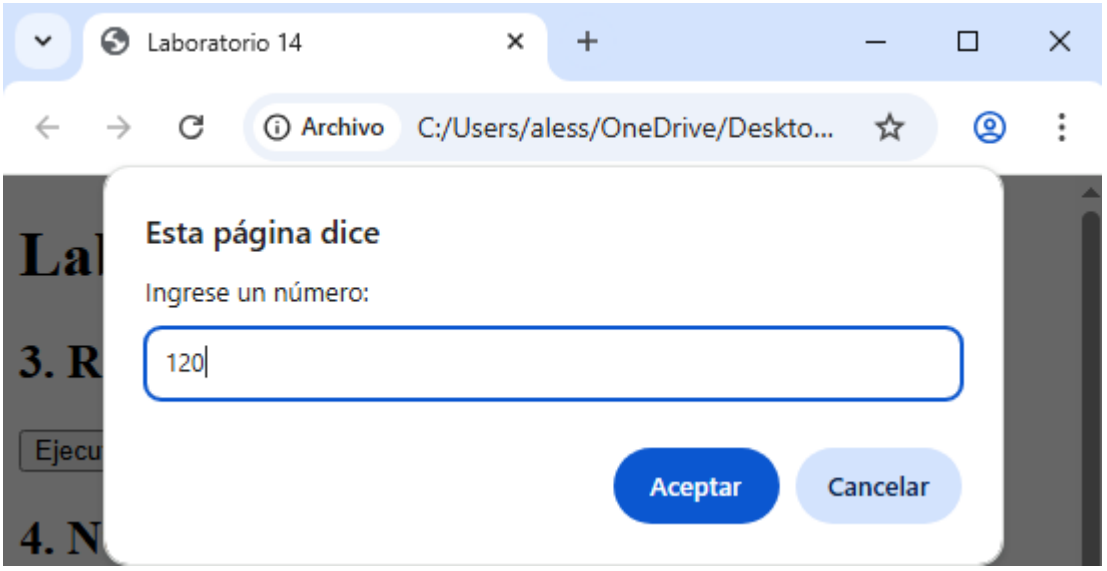
6. **Potencias y raíces. Solicita un número y muestra su cuadrado, cubo y raíz cuadrada usando Math.pow() y Math.sqrt()**

// Laboratorio Nro 14 - Ejercicio6
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 14min

index.html | ejercicio | ejercicio03.js

js > ejercicio06.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej6');
    const out = document.getElementById('out-ej6');

    btn.addEventListener('click', () => {
        let num = parseFloat(prompt("Ingrese un número:"));

        let cuadrado = Math.pow(num, 2);
        let cubo = Math.pow(num, 3);
        let raiz = Math.sqrt(num);

        out.textContent =
`Número: ${num}
Cuadrado: ${cuadrado}
Cubo: ${cubo}
Raíz cuadrada: ${raiz}`;
    });
});
```

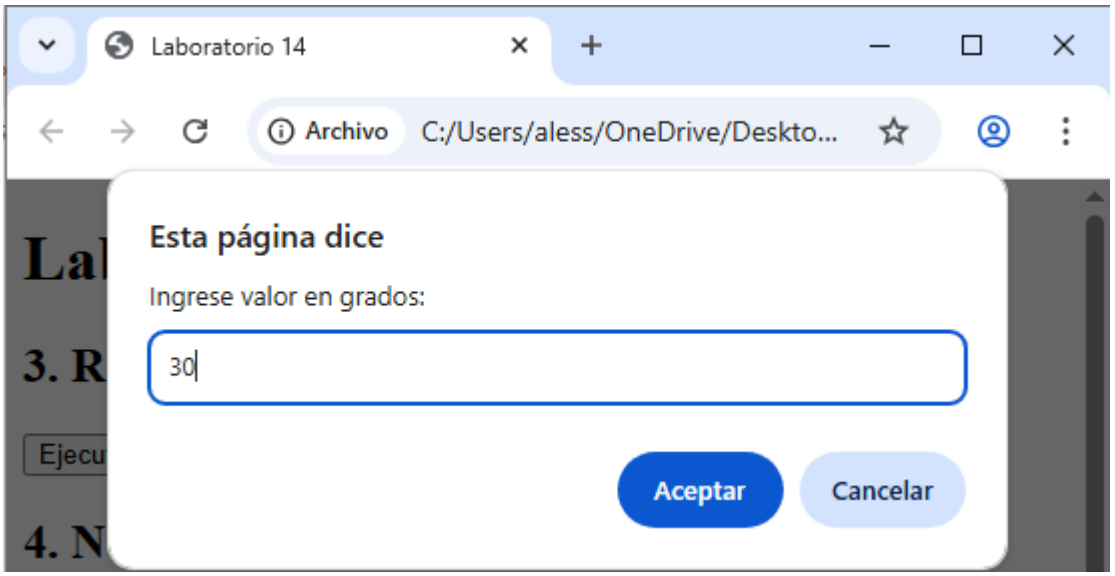


7. **Conversión de grados a radianes y de radianes a grados. Crea una función gradosARadianes(grados) que convierta ángulos de grados a radianes y muestre el seno y coseno del ángulo**
   Tip: usa la fórmula radianes = grados * (π / 180).
   Crea una función radianesAGrados(radianes) que convierta ángulos de radianes a grados
   Tip: usa la fórmula grados=radianes * (180/ π)
   // Laboratorio Nro 14 - Ejercicio7
   // Autor: Alessandro Josue Justo Vilca
   // Colaboró : sin colaboradores
   // Tiempo : 16min

index.html | ejercicio | ejercicio07.js | ejercicio

js > ejercicio07.js > ...

```
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej7');
    const out = document.getElementById('out-ej7');

    function gradosARadianes(grados) {
        return grados * (Math.PI / 180);
    }

    function radianesAGrados(radianes) {
        return radianes * (180 / Math.PI);
    }

    btn.addEventListener('click', () => {
        let grados = parseFloat(prompt("Ingrese valor en grados:"));
        let rad = gradosARadianes(grados);
        let sinVal = Math.sin(rad);
        let cosVal = Math.cos(rad);

        out.textContent =
`Grados: ${grados}
Radianes: ${rad}
Seno: ${sinVal}
Coseno: ${cosVal}

Conversión inversa:
Radianes → Grados: ${radianesAGrados(rad)}`;
    });
});
```



7. Grados ↔ Radianes

Ejecutar

```
Grados: 30
Radianes: 0.5235987755982988
Seno: 0.49999999999999994
Coseno: 0.8660254037844387

Conversión inversa:
Radianes → Grados: 29.999999999999996
```

8. **Generar contraseñas numéricas. Crear una función que genere una contraseña aleatoria de 6 dígitos (sólo números).**
   Tip: recorre un bucle 6 veces y genera un dígito del 0 al 9 en cada iteración

```
// Laboratorio Nro 14 - Ejercicio8
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 16min
```

index.html | ejercicio | ejercicio08.js | ejercicio07.js

js > ejercicio08.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej8');
    const out = document.getElementById('out-ej8');

    function generarPasswordNumerica() {
        let pass = '';
        for (let i = 0; i < 6; i++) {
            pass += Math.floor(Math.random() * 10);
        }
        return pass;
    }

    btn.addEventListener('click', () => {
        const password = generarPasswordNumerica();
        out.textContent = `Contraseña numérica (6 dígitos): ${password}`;
    });
});
```

8. Contraseña numérica de 6 dígitos

Ejecutar

```
Contraseña numérica (6 dígitos): 585122
```

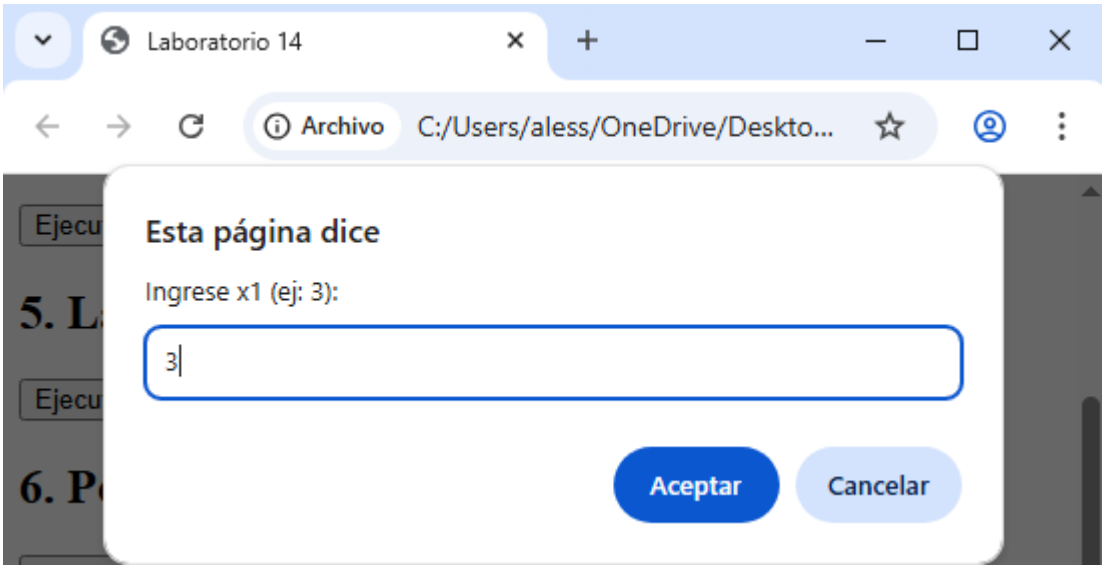
9. **Calcular distancia entre dos puntos en el plano cartesiano. Dadas las coordenadas (x1, y1) y (x2, y2), calcular la distancia entre los puntos y la suma de las distancias de cada punto al origen**

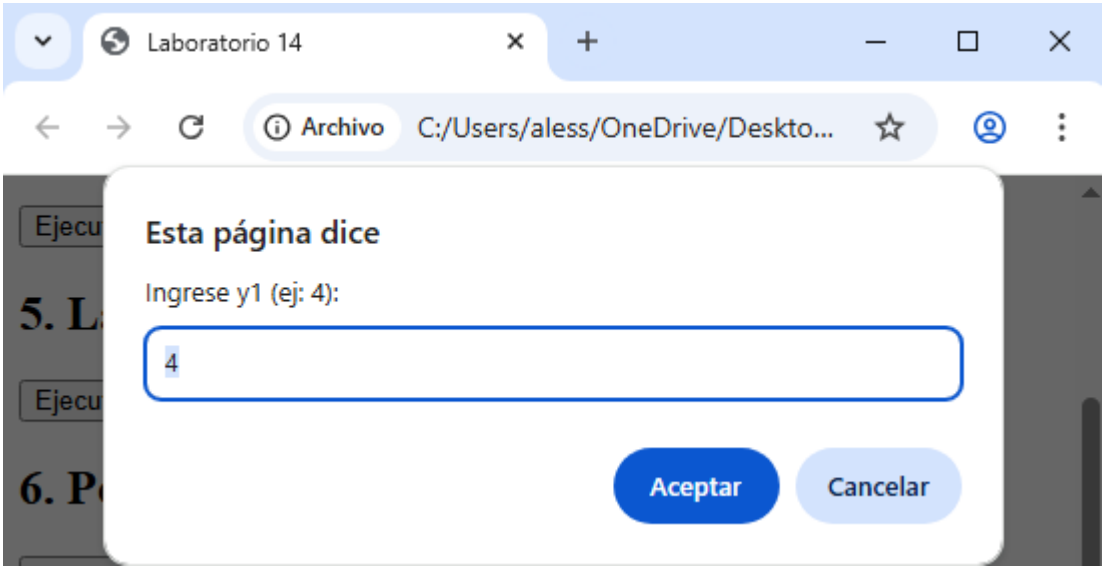
```
// Laboratorio Nro 14 - Ejercicio9
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 18min
```

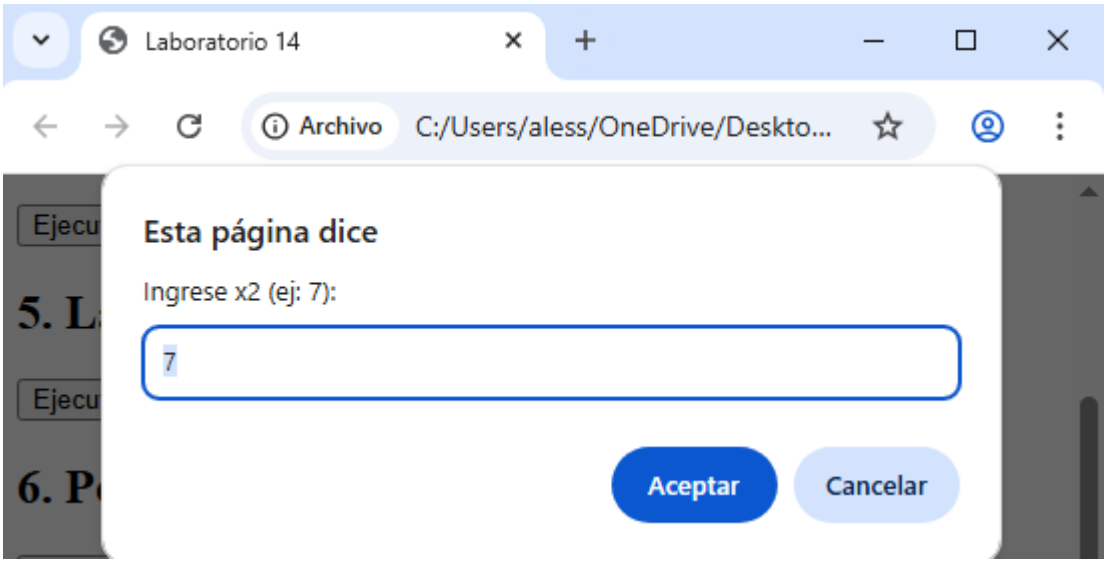
index.html | ejercicio | ejercicio09.js | ejerci

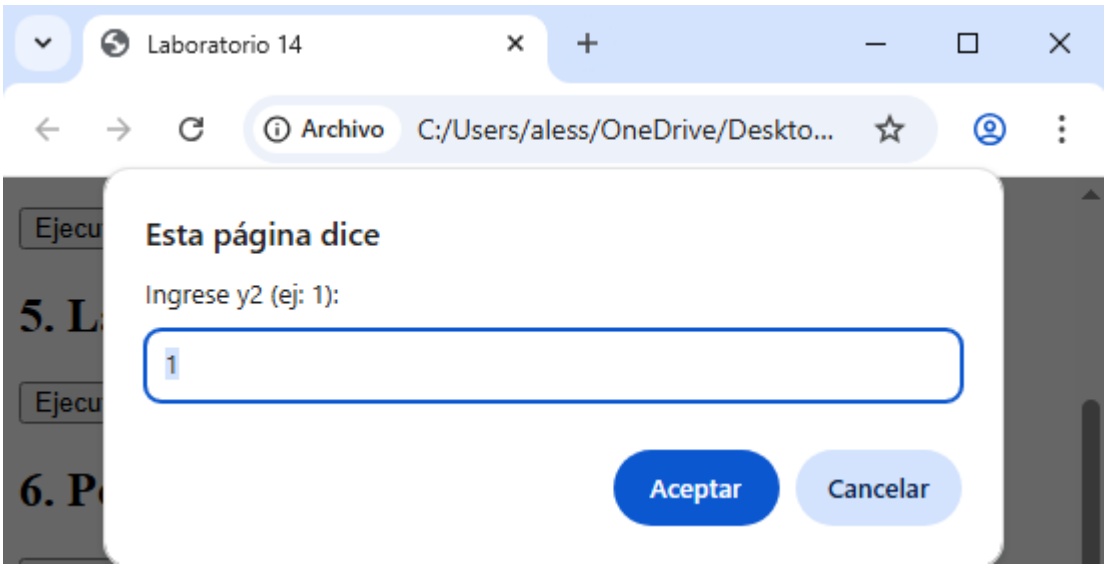
js > ejercicio09.js > ...

```
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej9');
    const out = document.getElementById('out-ej9');

    function distanciaEntrePuntos(x1, y1, x2, y2) {
        const dx = x2 - x1;
        const dy = y2 - y1;
        return Math.sqrt(dx * dx + dy * dy);
    }

    function distanciaAlOrigen(x, y) {
        return Math.sqrt(x * x + y * y);
    }

    btn.addEventListener('click', () => {
        const x1 = parseFloat(prompt('Ingrese x1 (ej: 3):', '3'));
        const y1 = parseFloat(prompt('Ingrese y1 (ej: 4):', '4'));
        const x2 = parseFloat(prompt('Ingrese x2 (ej: 7):', '7'));
        const y2 = parseFloat(prompt('Ingrese y2 (ej: 1):', '1'));

        const d = distanciaEntrePuntos(x1, y1, x2, y2);
        const d1 = distanciaAlOrigen(x1, y1);
        const d2 = distanciaAlOrigen(x2, y2);

        out.textContent =
`Distancia entre (${x1}, ${y1}) y (${x2}, ${y2}) = ${d}
Distancia punto1 → origen = ${d1}
Distancia punto2 → origen = ${d2}
Suma distancias al origen = ${ (d1 + d2) }`;
    });
});
```









9. Distancia entre dos puntos

Ejecutar

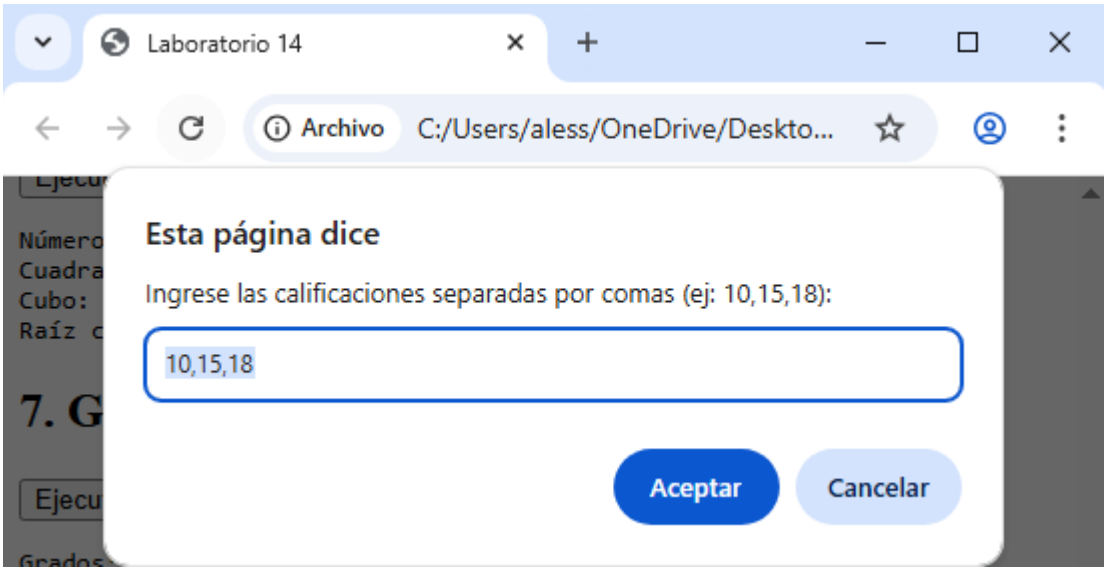
```
Distancia entre (3, 4) y (7, 1) = 5
Distancia punto1 → origen = 5
Distancia punto2 → origen = 7.0710678118654755
Suma distancias al origen = 12.071067811865476
```

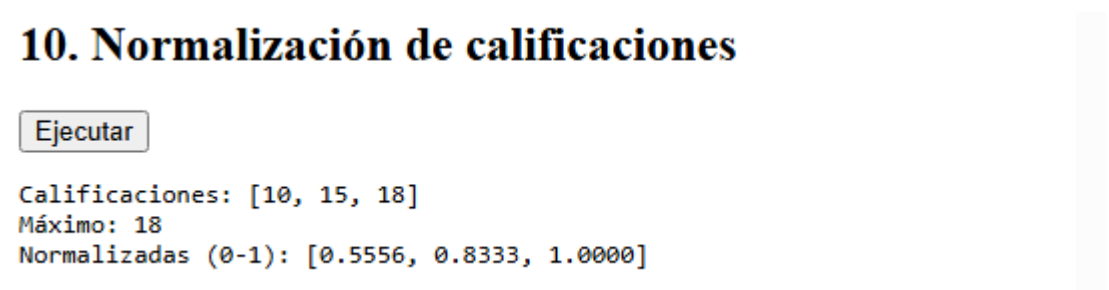
10. Normalización de calificaciones. Dado un arreglo de calificaciones, normalizar todos los valores al rango 0–1 dividiendo cada nota entre el máximo del arreglo
Tip: usa el operador de propagación: Math.max(...array).

```
// Laboratorio Nro 14 - Ejercicio10
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 17min
```

index.html | ejercicio10.js | ejercicio09.js | ejercicio08.js

```
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej10');
    const out = document.getElementById('out-ej10');

    function normalizarCalificaciones(arr) {
        if (!arr.length) return [];
        const max = Math.max(...arr);
        if (max === 0) return arr.map(() => 0);
        return arr.map(n => n / max);
    }

    btn.addEventListener('click', () => {
        const entrada = prompt('Ingrese las calificaciones separadas por comas (ej: 10,15,18):', '10,15,18');
        const arr = entrada.split(',').map(s => parseFloat(s.trim())).filter(n => !isNaN(n));
        const normalizadas = normalizarCalificaciones(arr);
        out.textContent =
`Calificaciones: [${arr.join(', ')}]
Máximo: ${Math.max(...arr)}
Normalizadas (0-1): [${normalizadas.map(n => n.toFixed(4)).join(', ')}]`;
    });
});
```





11. Control de inventario con encapsulación. Crear una clase Producto con atributos privados nombre, precio, stock.
Implementa setters que validen que el precio y el stock sean mayores a 0
Agregar un método vender(cantidad) que disminuya el stock solo si hay unidades suficientes

```
// Laboratorio Nro 14 - Ejercicio11
// Autor: Alessandro Josue Justo Vilca
```

// Colaboró : sin colaboradores
// Tiempo : 21min

index.html | ejercicio | ejercicio11.js ● | ejercicio10.js | ejercicio09.js | ejer

js > ejercicio11.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej11');
    const out = document.getElementById('out-ej11');
    class Producto {
        #nombre;
        #precio;
        #stock;
        constructor(nombre, precio, stock) {
            this.#nombre = nombre;
            this.setPrecio(precio);
            this.setStock(stock);
        }
        setPrecio(v) {
            const n = Number(v);
            if (!isNaN(n) && n > 0) this.#precio = n;
            else throw new Error('Precio debe ser número > 0');
        }
        setStock(v) {
            const n = Number(v);
            if (!isNaN(n) && n >= 0) this.#stock = n;
            else throw new Error('Stock debe ser número >= 0');
        }
        vender(cantidad) {
            const c = Number(cantidad);
            if (isNaN(c) || c <= 0) throw new Error('Cantidad inválida');
            if (c > this.#stock) return false;
            this.#stock -= c;
            return true;
        }
        info() {
            return `Producto: ${this.#nombre}\nPrecio(raw): ${this.#precio}\nStock: ${this.#stock}`;
        }
    }
    btn.addEventListener('click', () => {
        try {
            const p = new Producto('Arroz', 12.5, 20);
            const antes = p.info();
            const ok = p.vender(5);
            const despues = p.info();
            out.textContent = `${antes}\n\nvender(5) -> ${ok}\n\n${despues}`;
        } catch (e) {
            out.textContent = `Error: ${e.message}`;
        }
    });
});
```

11. Clase Producto (encapsulación)

Ejecutar

```
Producto: Arroz
Precio(raw): 12.5
Stock: 20

vender(5) -> true

Producto: Arroz
Precio(raw): 12.5
Stock: 15
```

12. Modificar Producto para que el getter precio devuelva el valor con formato de moneda ($120.00) y que el setter acepte tanto número como cadena ("120.5")
Tip: puedes usar Number() y toFixed(2)

// Laboratorio Nro 14 - Ejercicio12
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 20min

index.html | ejercicio | ejercicio12.js | ejercicio11.js | ejer

js > ejercicio12.js > ...

```
1   document.addEventListener('DOMContentLoaded', () => {
2       const btn = document.getElementById('run-ej12');
3       const out = document.getElementById('out-ej12');
4
5       class Producto {
6           #nombre;
7           #precio = 0;
8
9           constructor(nombre, precio) {
10              this.#nombre = nombre;
11              this.precio = precio;
12          }
13
14          set precio(v) {
15              const n = Number(v);
16              if (isNaN(n) || n < 0) throw new Error('Precio inválido');
17              this.#precio = n;
18          }
19
20          get precio() {
21              return `$${this.#precio.toFixed(2)}`;
22          }
23
24          info() {
25              return `Producto: ${this.#nombre}\nPrecio formateado: ${this.precio}`;
26          }
27      }
28
29      btn.addEventListener('click', () => {
30          try {
31              const p = new Producto('Cuaderno', '120.5'); // string aceptado
32              const antes = p.info();
33              p.precio = 99; // número aceptado
34              const despues = p.info();
35              out.textContent = `${antes}\n\nDespués de setPrecio(99):\n${despues}`;
36          } catch (e) {
37              out.textContent = `Error: ${e.message}`;
38          }
39      });
40  });
```

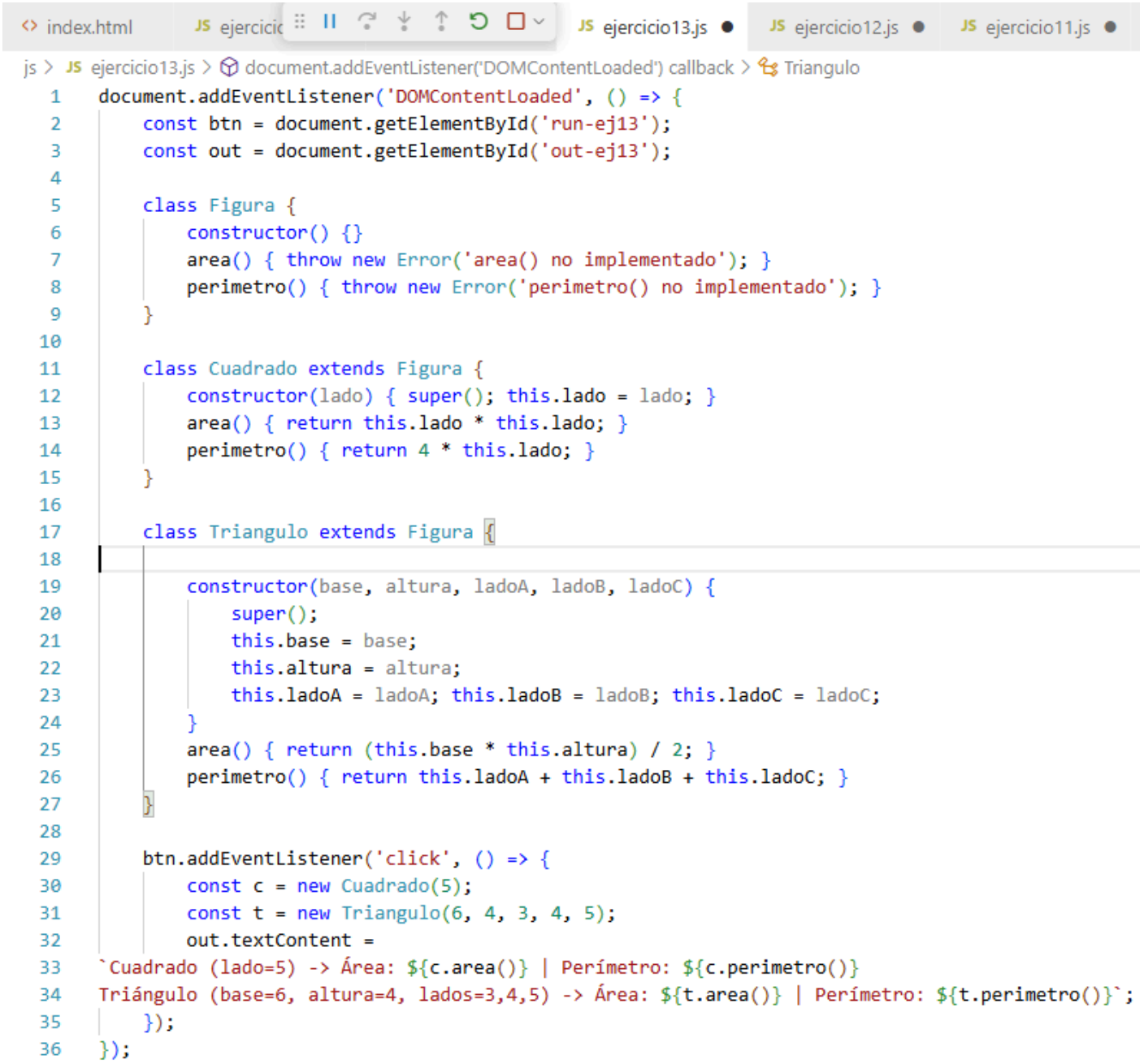
12. Producto con formato de precio

Ejecutar

```
Producto: Cuaderno
Precio formateado: $120.50

Después de setPrecio(99):
Producto: Cuaderno
Precio formateado: $99.00
```

13. Herencia. Crear una clase Figura. Debe tener un método area() y perímetro() que las subclases Cuadrado y Triangulo deben sobrescribir.
Tip: llamar a constructor de la superclase

// Laboratorio Nro 14 - Ejercicio13
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 20min




```
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej13');
    const out = document.getElementById('out-ej13');

    class Figura {
        constructor() {}
        area() { throw new Error('area() no implementado'); }
        perimetro() { throw new Error('perimetro() no implementado'); }
    }

    class Cuadrado extends Figura {
        constructor(lado) { super(); this.lado = lado; }
        area() { return this.lado * this.lado; }
        perimetro() { return 4 * this.lado; }
    }

    class Triangulo extends Figura {

        constructor(base, altura, ladoA, ladoB, ladoC) {
            super();
            this.base = base;
            this.altura = altura;
            this.ladoA = ladoA; this.ladoB = ladoB; this.ladoC = ladoC;
        }
        area() { return (this.base * this.altura) / 2; }
        perimetro() { return this.ladoA + this.ladoB + this.ladoC; }
    }

    btn.addEventListener('click', () => {
        const c = new Cuadrado(5);
        const t = new Triangulo(6, 4, 3, 4, 5);
        out.textContent =
`Cuadrado (lado=5) -> Área: ${c.area()} | Perímetro: ${c.perimetro()}
Triángulo (base=6, altura=4, lados=3,4,5) -> Área: ${t.area()} | Perímetro: ${t.perimetro()}`;
    });
});
```

13. Figura - Cuadrado - Triángulo

Ejecutar

```
Cuadrado (lado=5) -> Área: 25 | Perímetro: 20
Triángulo (base=6, altura=4, lados=3,4,5) -> Área: 12 | Perímetro: 12
```

14. Herencia. Crear una clase base Usuario con nombre y email. Que lo hereden Cliente y Administrador

• Cliente tiene un nivel de fidelidad [1–5]

• Administrador tiene permisos (crear, editar, eliminar)

Cada uno sobrescribe mostrarInfo() con diferente detalle

Tip: llama a super() para reutilizar atributos base

// Laboratorio Nro 14 - Ejercicio14
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 21min

index.html | ejercicio14.js | ejercicio13.js | ejercicio12.js

js > ejercicio14.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej14');
    const out = document.getElementById('out-ej14');

    class Usuario {
        constructor(nombre, email) { this.nombre = nombre; this.email = email; }
        mostrarInfo() { return `Nombre: ${this.nombre} | Email: ${this.email}`; }
    }

    class Cliente extends Usuario {
        constructor(nombre, email, nivelFidelidad) {
            super(nombre, email);
            this.nivelFidelidad = Math.max(1, Math.min(5, Number(nivelFidelidad || 1)));
        }
        mostrarInfo() {
            return `Cliente -> ${super.mostrarInfo()} | Nivel fidelidad: ${this.nivelFidelidad}`;
        }
    }

    class Administrador extends Usuario {
        constructor(nombre, email, permisos = []) {
            super(nombre, email);
            this.permisos = permisos;
        }
        mostrarInfo() {
            return `Administrador -> ${super.mostrarInfo()} | Permisos: ${this.permisos.join(', ')}`;
        }
    }

    btn.addEventListener('click', () => {
        const c = new Cliente('Ana', 'ana@mail.com', 4);
        const a = new Administrador('Luis', 'luis@mail.com', ['crear','editar','eliminar']);
        out.textContent = `${c.mostrarInfo()}\n${a.mostrarInfo()}`;
    });
});
```

14. Usuario - Cliente - Administrador

Ejecutar

```
Cliente -> Nombre: Ana | Email: ana@mail.com | Nivel fidelidad: 4
Administrador -> Nombre: Luis | Email: luis@mail.com | Permisos: crear, editar, eliminar
```

15. **Polimorfismo. Crear una lista de usuarios (Cliente, Administrador) y recórrela mostrando la información con mostrarInfo().**
    Tip: usa un forEach o for...of para recorrer el array

// Laboratorio Nro 14 - Ejercicio15
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 22min

index.html | ejercicio | ejercicio15.js | ejercicio14.js | ejercicio13.js | ejer

js > ejercicio15.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej15');
    const out = document.getElementById('out-ej15');

    class Usuario {
        constructor(nombre, email) { this.nombre = nombre; this.email = email; }
        mostrarInfo() { return `${this.nombre} <${this.email}>`; }
    }
    class Cliente extends Usuario {
        constructor(nombre, email, nivel) { super(nombre, email); this.nivel = nivel; }
        mostrarInfo() { return `Cliente: ${this.nombre} | Nivel: ${this.nivel}`; }
    }
    class Administrador extends Usuario {
        constructor(nombre, email, permisos) { super(nombre, email); this.permisos = permisos; }
        mostrarInfo() { return `Administrador: ${this.nombre} | Permisos: ${this.permisos.join(', ')}`; }
    }

    btn.addEventListener('click', () => {
        const usuarios = [
            new Cliente('Mario', 'mario@mail', 2),
            new Administrador('Laura', 'laura@mail', ['crear','eliminar']),
            new Cliente('Pedro', 'pedro@mail', 5)
        ];

        const lines = usuarios.map(u => u.mostrarInfo());
        out.textContent = lines.join('\n');
    });
});
```

15. Polimorfismo lista de usuarios

Ejecutar

```
Cliente: Mario | Nivel: 2
Administrador: Laura | Permisos: crear, eliminar
Cliente: Pedro | Nivel: 5
```

16. **Herencia. Crear la jerarquía Empleado - Programador - ProgramadorSenior**
    • Empleado tiene nombre y sueldoBase
    • Programador añade lenguaje y método calcularSueldo() con bono del 10%
    • ProgramadorSenior sobreescribe calcularSueldo() con un bono del 25%
    Tip: llama a super.calcularSueldo() desde la subclase

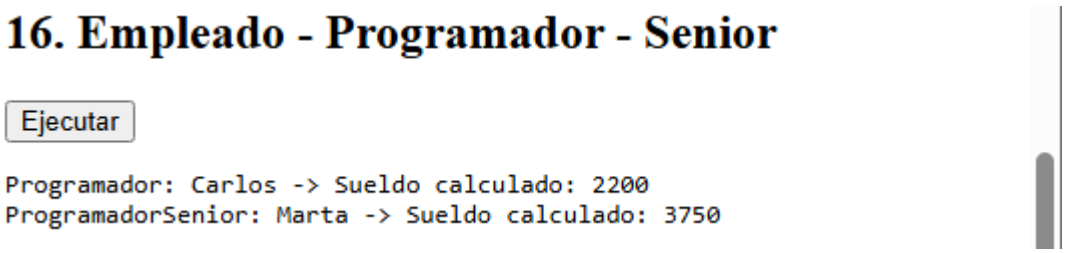
// Laboratorio Nro 14 - Ejercicio16
// Autor: Alessandro Josue Justo Vilca

// Colaboró : sin colaboradores
// Tiempo : 23min

index.html | ejercicio | ejercicio16.js | ejercicio15.js | ejercicio14.js | ejer

js > ejercicio16.js > ...

```
1  document.addEventListener('DOMContentLoaded', () => {
2      const btn = document.getElementById('run-ej16');
3      const out = document.getElementById('out-ej16');
4
5      class Empleado {
6          constructor(nombre, sueldoBase) { this.nombre = nombre; this.sueldoBase = Number(sueldoBase); }
7          calcularSueldo() { return this.sueldoBase; }
8      }
9
10     class Programador extends Empleado {
11         constructor(nombre, sueldoBase, lenguaje) { super(nombre, sueldoBase); this.lenguaje = lenguaje; }
12         calcularSueldo() { return super.calcularSueldo() * 1.10; } // bono 10%
13     }
14
15     class ProgramadorSenior extends Programador {
16         calcularSueldo() {
17             // llamamos a super.calcularSueldo() (que aplica 10%) y lo ajustamos para obtener 25% total
18             const baseCon10 = super.calcularSueldo(); // sueldoBase * 1.10
19             const sueldoBase = baseCon10 / 1.10; // recuperamos sueldoBase
20             return sueldoBase * 1.25; // aplicamos 25% total
21         }
22     }
23
24     btn.addEventListener('click', () => {
25         const p = new Programador('Carlos', 2000, 'JS');
26         const s = new ProgramadorSenior('Marta', 3000, 'Java');
27         out.textContent =
28 `Programador: ${p.nombre} -> Sueldo calculado: ${p.calcularSueldo()}
29 ProgramadorSenior: ${s.nombre} -> Sueldo calculado: ${s.calcularSueldo()}`;
30     });
31 });
```



17. Encapsulación y polimorfismo. Crear una clase Cuenta con atributo privado saldo y métodos depositar() y retirar().
Luego crea subclases CuentaAhorro y CuentaCorriente que redefinan retirar() según sus reglas (por ejemplo, permitir
sobregiro en la cuenta corriente pero con un límite) y también la transferencia entre cuentas
Tip: implementa validaciones distintas en cada clase hija.
// Laboratorio Nro 14 - Ejercicio17
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 26min

index.html | ejercicio | ejercicio17.js | ejercicio16.js | ejercicio15.js | ejer

js > ejercicio17.js > document.addEventListener('DOMContentLoaded') callback > btn.addEventListener('click') callback

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej17');
    const out = document.getElementById('out-ej17');

    class Cuenta {
        #saldo;
        constructor(saldoInicial = 0) { this.#saldo = Number(saldoInicial); }
        depositar(monto) {
            if (Number(monto) <= 0) throw new Error('Monto debe ser > 0');
            this.#saldo += Number(monto);
        }
        retirar(monto) { throw new Error('retirar() debe implementarse en la subclase'); }
        getSaldo() { return this.#saldo; }
        transferir(destino, monto) {
            if (!(destino instanceof Cuenta)) throw new Error('Destino inválido');
            this.retirar(monto);
            destino.depositar(monto);
        }
    }

    class CuentaAhorro extends Cuenta {
        retirar(monto) {
            monto = Number(monto);
            if (monto <= 0) throw new Error('Monto inválido');
            if (monto > this.getSaldo()) throw new Error('Saldo insuficiente en CuentaAhorro');
            // usar depositar con negativo para mantener encapsulación
            this.depositar(-monto);
        }
    }

    class CuentaCorriente extends Cuenta {
        constructor(saldoInicial = 0, limiteSobregiro = 500) {
            super(saldoInicial);
            this.limite = Number(limiteSobregiro);
        }
        retirar(monto) {
            monto = Number(monto);
            if (monto <= 0) throw new Error('Monto inválido');
            if (this.getSaldo() - monto < -this.limite) throw new Error('Excede límite de sobregiro');
            this.depositar(-monto);
        }
    }

    btn.addEventListener('click', () => {
        try {
            const ahorro = new CuentaAhorro(1000);
            const corriente = new CuentaCorriente(200, 300);
            ahorro.transferir(corriente, 300); // 1000-300, corriente 200+300
            corriente.retirar(450); // permite sobregiro hasta -100 (200+300-450 = 50)
            out.textContent =
`CuentaAhorro saldo: ${ahorro.getSaldo()}
CuentaCorriente saldo: ${corriente.getSaldo()}`;
        } catch (e) {
            out.textContent = `Error: ${e.message}`;
        }
    });
});
```

17. Cuenta - Ahorro - Corriente

Ejecutar

```
Error: Monto debe ser > 0
```

18. Composición. Crear una clase Carrito que contenga una lista de objetos Producto. Agrega métodos agregarProducto(), calcularTotal() y mostrarResumen()

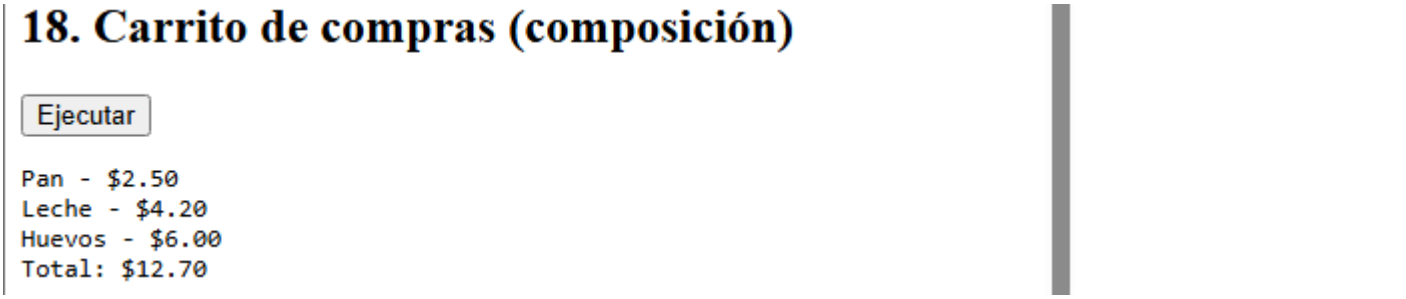
Tip: usa un array de objetos

// Laboratorio Nro 14 - Ejercicio18
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 21min

index.html | ejercicio | ejercicio18.js | ejercicio17.js | ejercicio16.js

js > ejercicio18.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej18');
    const out = document.getElementById('out-ej18');

    class Producto {
        constructor(nombre, precio) { this.nombre = nombre; this.precio = Number(precio); }
    }

    class Carrito {
        constructor() { this.productos = []; }
        agregarProducto(producto) { this.productos.push(producto); }
        calcularTotal() { return this.productos.reduce((s, p) => s + p.precio, 0); }
        mostrarResumen() {
            const lineas = this.productos.map(p => `${p.nombre} - $${p.precio.toFixed(2)}`);
            lineas.push(`Total: $${this.calcularTotal().toFixed(2)}`);
            return lineas.join('\n');
        }
    }

    btn.addEventListener('click', () => {
        const carrito = new Carrito();
        carrito.agregarProducto(new Producto('Pan', 2.5));
        carrito.agregarProducto(new Producto('Leche', 4.2));
        carrito.agregarProducto(new Producto('Huevos', 6.0));
        out.textContent = carrito.mostrarResumen();
    });
});
```



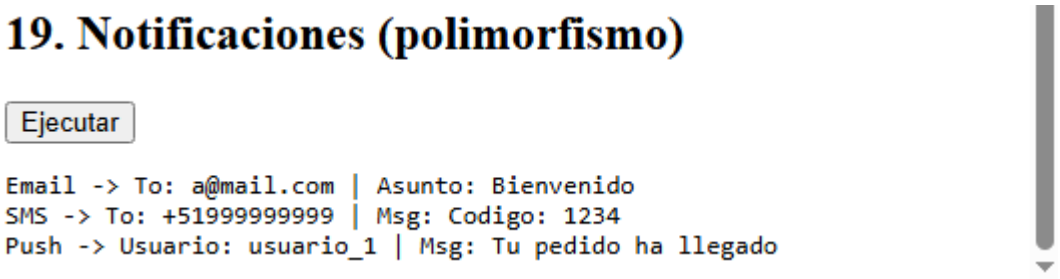
19. **Polimorfismo. Crear una clase base Notificacion con un método enviar(). Implementa subclases Email, SMS y Push que sobrescriban enviar() con mensajes distintos. Luego crea una función que reciba una lista de notificaciones y las procese dinámicamente**
    Tip: usa una estructura de datos adecuada que almacene objetos y que llame a notificacion.enviar() en un bucle

    // Laboratorio Nro 14 - Ejercicio19
    // Autor: Alessandro Josue Justo Vilca
    // Colaboró : sin colaboradores
    // Tiempo : 24min

index.html | ejercicio | ejercicio19.js | ejercicio18.js | ejercicio17.js

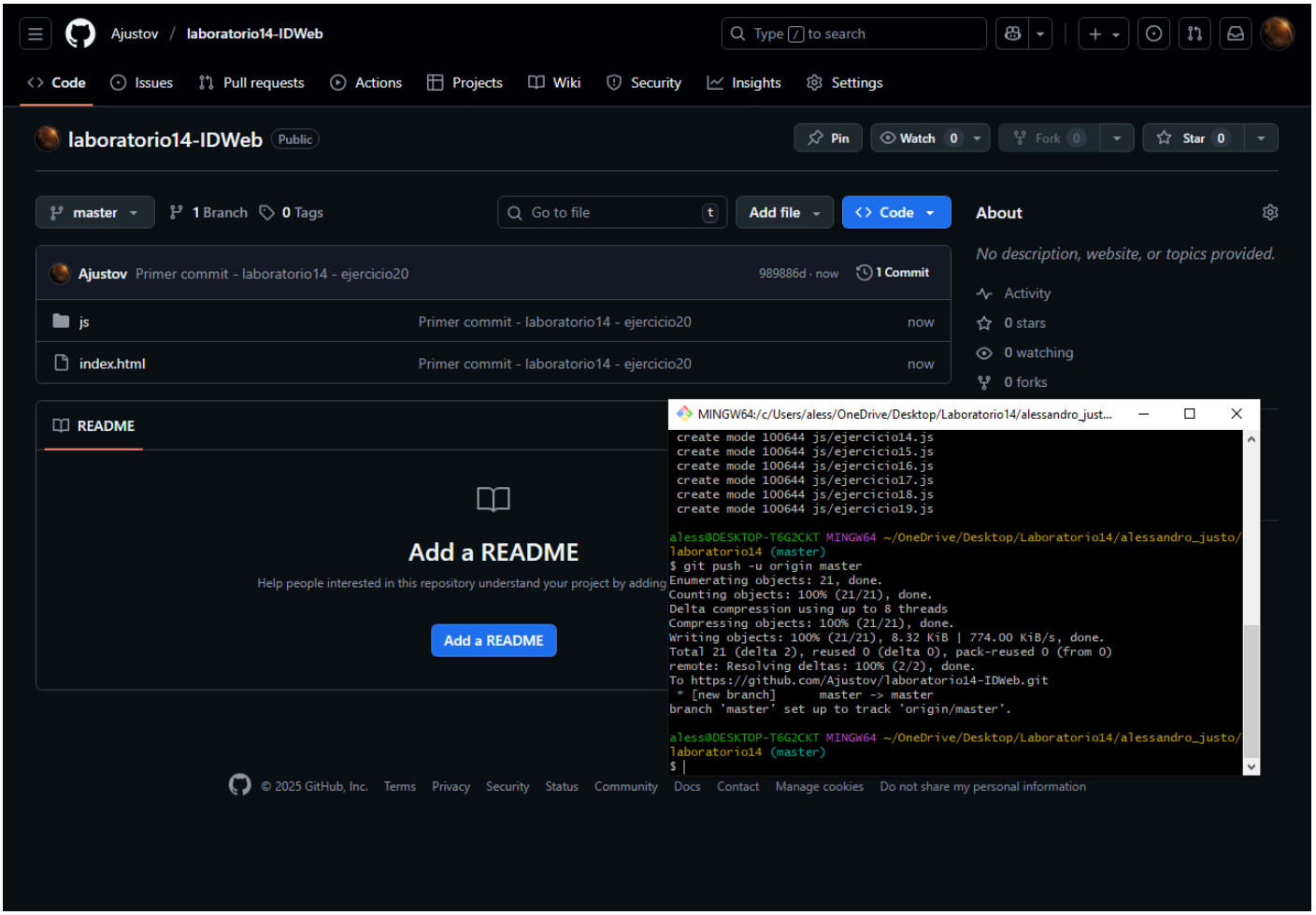
js > ejercicio19.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const btn = document.getElementById('run-ej19');
    const out = document.getElementById('out-ej19');

    class Notificacion {
        enviar() { return 'Notificación base'; }
    }
    class Email extends Notificacion {
        constructor(destino, asunto) { super(); this.destino = destino; this.asunto = asunto; }
        enviar() { return `Email -> To: ${this.destino} | Asunto: ${this.asunto}`; }
    }
    class SMS extends Notificacion {
        constructor(destino, mensaje) { super(); this.destino = destino; this.mensaje = mensaje; }
        enviar() { return `SMS -> To: ${this.destino} | Msg: ${this.mensaje}`; }
    }
    class Push extends Notificacion {
        constructor(usuario, mensaje) { super(); this.usuario = usuario; this.mensaje = mensaje; }
        enviar() { return `Push -> Usuario: ${this.usuario} | Msg: ${this.mensaje}`; }
    }

    function procesarNotificaciones(lista) {
        return lista.map(n => n.enviar());
    }

    btn.addEventListener('click', () => {
        const lista = [
            new Email('a@mail.com', 'Bienvenido'),
            new SMS('+51999999999', 'Codigo: 1234'),
            new Push('usuario_1', 'Tu pedido ha llegado')
        ];
        out.textContent = procesarNotificaciones(lista).join('\n');
    });
});
```



20. Crear un repositorio remoto en GitHub y subir tu repositorio local. Compartir URL y pdf con captura de pantalla del código de los archivos js y de la ejecución

// Laboratorio Nro 14 - Ejercicio20
// Autor: Alessandro Josue Justo Vilca
// Colaboró : sin colaboradores
// Tiempo : 3min



URL: https://github.com/Ajustov/laboratorio14-IDWeb