

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

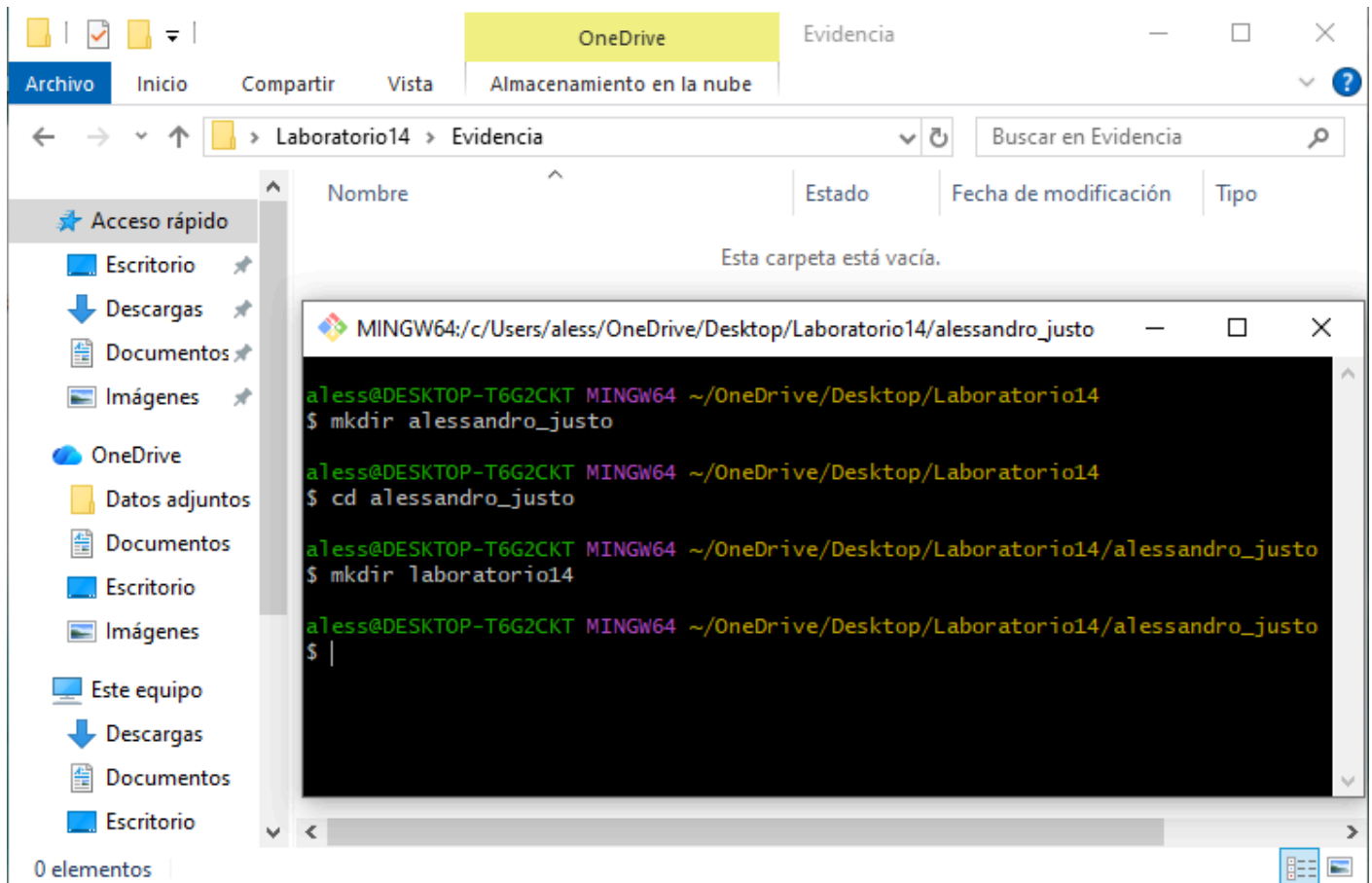
1. **Crear un directorio que tenga su nombre y un subdirectorio laboratorio14. Recomendación: usar minúsculas, sin espacios, sin tildes ni “ñ” y con guiones medios o bajos**

// 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 > ...

```
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Laboratorio 14</title>
7  </head>
8  <body>
9      <main>
10         <h1>Laboratorio 14</h1>
11
12         <section id="ej3">
13             <h2>3. Redondeo de precios</h2>
14             <button id="run-ej3">Ejecutar</button>
15             <pre id="out-ej3"></pre>
16             <script src="js/ejercicio03.js"></script>
17         </section>
18
19         <section id="ej4">
20             <h2>4. Número aleatorio en un rango</h2>
21             <button id="run-ej4">Ejecutar</button>
22             <pre id="out-ej4"></pre>
23             <script src="js/ejercicio04.js"></script>
24         </section>
25
26         <section id="ej5">
27             <h2>5. Lanzamiento de dados</h2>
28             <button id="run-ej5">Ejecutar</button>
29             <pre id="out-ej5"></pre>
30             <script src="js/ejercicio05.js"></script>
31         </section>
32
33         <section id="ej6">
34             <h2>6. Potencias y raíces</h2>
35             <button id="run-ej6">Ejecutar</button>
36             <pre id="out-ej6"></pre>
37             <script src="js/ejercicio06.js"></script>
38         </section>
39
40         <section id="ej7">
41             <h2>7. Grados ↔ Radianes</h2>
42             <button id="run-ej7">Ejecutar</button>
43             <pre id="out-ej7"></pre>
44             <script src="js/ejercicio07.js"></script>
45         </section>
```

46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81

```
<section id="ej8">
  <h2>8. Contraseña numérica de 6 dígitos</h2>
  <button id="run-ej8">Ejecutar</button>
  <pre id="out-ej8"></pre>
  <script src="js/ejercicio08.js"></script>
</section>

<section id="ej9">
  <h2>9. Distancia entre dos puntos</h2>
  <button id="run-ej9">Ejecutar</button>
  <pre id="out-ej9"></pre>
  <script src="js/ejercicio09.js"></script>
</section>

<section id="ej10">
  <h2>10. Normalización de calificaciones</h2>
  <button id="run-ej10">Ejecutar</button>
  <pre id="out-ej10"></pre>
  <script src="js/ejercicio10.js"></script>
</section>

<section id="ej11">
  <h2>11. Clase Producto (encapsulación)</h2>
  <button id="run-ej11">Ejecutar</button>
  <pre id="out-ej11"></pre>
  <script src="js/ejercicio11.js"></script>
</section>

<section id="ej12">
  <h2>12. Producto con formato de precio</h2>
  <button id="run-ej12">Ejecutar</button>
  <pre id="out-ej12"></pre>
  <script src="js/ejercicio12.js"></script>
</section>
```

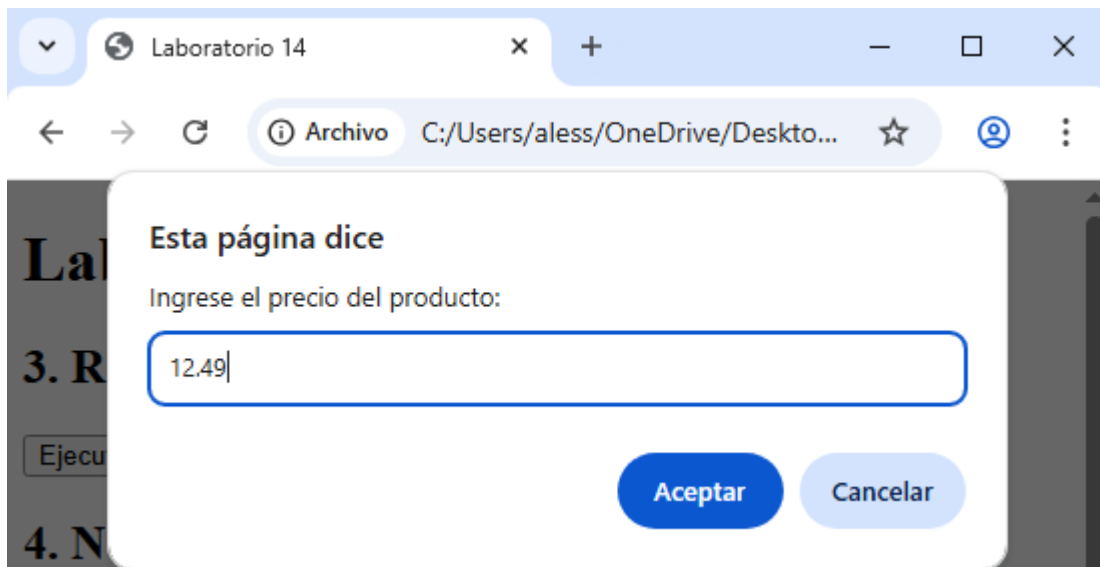
```

82     <section id="ej13">
83         <h2>13. Figura - Cuadrado - Triángulo</h2>
84         <button id="run-ej13">Ejecutar</button>
85         <pre id="out-ej13"></pre>
86         <script src="js/ejercicio13.js"></script>
87     </section>
88
89     <section id="ej14">
90         <h2>14. Usuario - Cliente - Administrador</h2>
91         <button id="run-ej14">Ejecutar</button>
92         <pre id="out-ej14"></pre>
93         <script src="js/ejercicio14.js"></script>
94     </section>
95
96     <section id="ej15">
97         <h2>15. Polimorfismo lista de usuarios</h2>
98         <button id="run-ej15">Ejecutar</button>
99         <pre id="out-ej15"></pre>
100        <script src="js/ejercicio15.js"></script>
101    </section>
102
103    <section id="ej16">
104        <h2>16. Empleado - Programador - Senior</h2>
105        <button id="run-ej16">Ejecutar</button>
106        <pre id="out-ej16"></pre>
107        <script src="js/ejercicio16.js"></script>
108    </section>
109
110    <section id="ej17">
111        <h2>17. Cuenta - Ahorro - Corriente</h2>
112        <button id="run-ej17">Ejecutar</button>
113        <pre id="out-ej17"></pre>
114        <script src="js/ejercicio17.js"></script>
115    </section>
116
117    <section id="ej18">
118        <h2>18. Carrito de compras (composición)</h2>
119        <button id="run-ej18">Ejecutar</button>
120        <pre id="out-ej18"></pre>
121        <script src="js/ejercicio18.js"></script>
122    </section>
123
124    <section id="ej19">
125        <h2>19. Notificaciones (polimorfismo)</h2>
126        <button id="run-ej19">Ejecutar</button>
127        <pre id="out-ej19"></pre>
128        <script src="js/ejercicio19.js"></script>
129    </section>
130
131    </main>
132 </body>
133 </html>

```

js > JS ejercicio03.js > ...

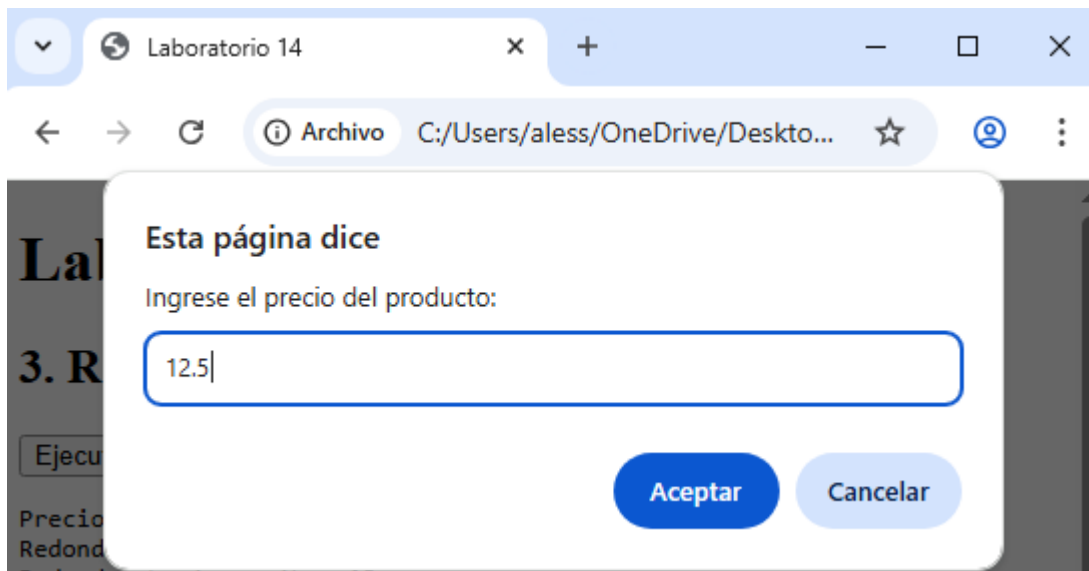
```
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej3');
3   const out = document.getElementById('out-ej3');
4
5   btn.addEventListener('click', () => {
6     let precio = parseFloat(prompt("Ingrese el precio del producto: "));
7
8     let abajo = Math.floor(precio);
9     let arriba = Math.ceil(precio);
10    let normal = Math.round(precio);
11
12    out.textContent =
13      `Precio original: ${precio}
14      Redondeo hacia abajo: ${abajo}
15      Redondeo hacia arriba: ${arriba}
16      Redondeo normal: ${normal}`;
17  });
18 });
--
```



3. Redondeo de precios

Ejecutar

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



3. Redondeo de precios

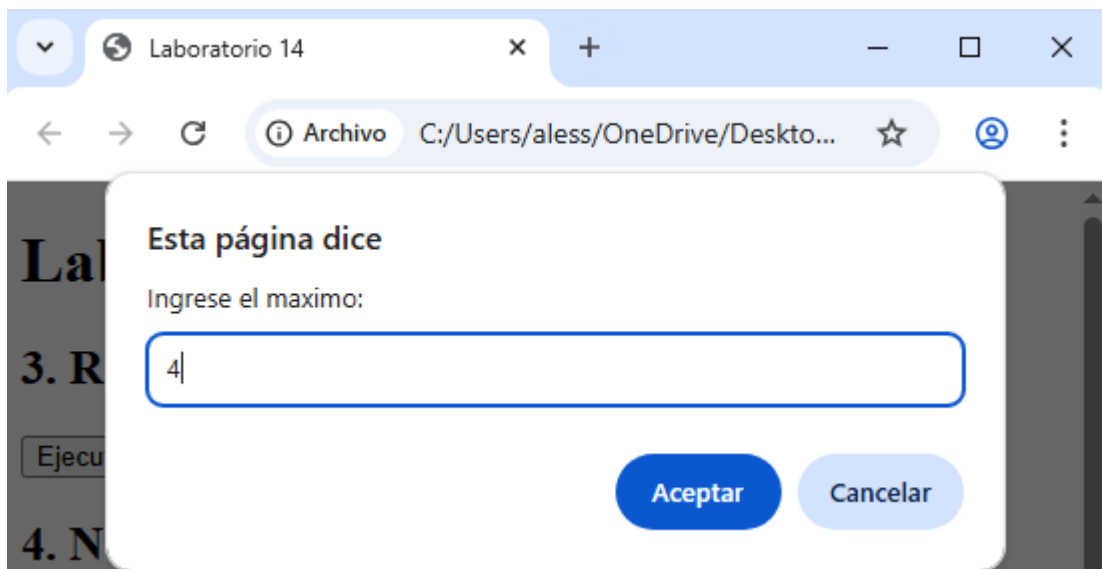
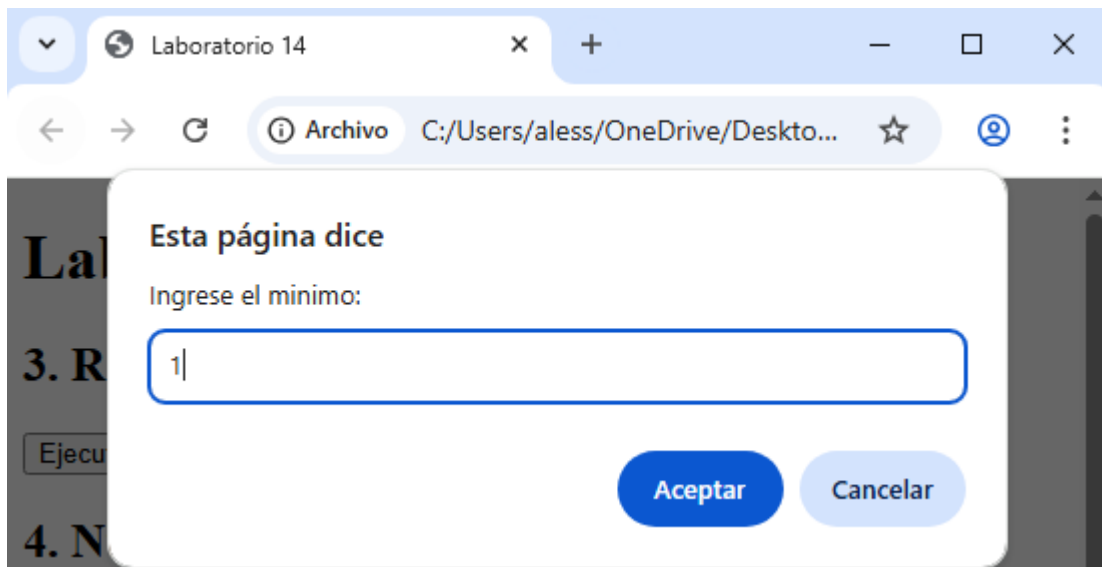
Ejecutar

Precio original: 12.5
Redondeo hacia abajo: 12
Redondeo hacia arriba: 13
Redondeo normal: 13

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  JS ejercicio  JS ejercicio04.js > ...
1  document.addEventListener('DOMContentLoaded', () => {
2      const btn = document.getElementById('run-ej4');
3      const out = document.getElementById('out-ej4');
4
5      function numeroAleatorio(min, max) {
6          return Math.floor(Math.random() * (max - min + 1)) + min;
7      }
8
9      btn.addEventListener('click', () => {
10         let min = parseInt(prompt("Ingrese el minimo: "));
11         let max = parseInt(prompt("Ingrese el maximo: "));
12
13         let resultado = numeroAleatorio(min, max);
14
15         out.textContent = `Número aleatorio generado: ${resultado}`;
16     });
17 });
```

4. Número aleatorio en un rango

Ejecutar

Número aleatorio generado: 3

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

JS ejercicio

JS ejercicio04.js

js > JS ejercicio05.js > ...

```
1 document.addEventListener('DOMContentLoaded', () => {
2     const btn = document.getElementById('run-ej5');
3     const out = document.getElementById('out-ej5');
4
5     function numeroAleatorio(min, max) {
6         return Math.floor(Math.random() * (max - min + 1)) + min;
7     }
8
9     btn.addEventListener('click', () => {
10         let dado1 = numeroAleatorio(1, 6);
11         let dado2 = numeroAleatorio(1, 6);
12         let suma = dado1 + dado2;
13
14         out.textContent = `Dado 1: ${dado1}\nDado 2: ${dado2}\nSuma: ${suma}`;
15     });
16 });
```

5. Lanzamiento de dados

Ejecutar

Dado 1: 6
Dado 2: 2
Suma: 8

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

JS ejercicio

JS ejercicio03.js

js > JS ejercicio06.js > ...

```
1 document.addEventListener('DOMContentLoaded', () => {
2     const btn = document.getElementById('run-ej6');
3     const out = document.getElementById('out-ej6');
4
5     btn.addEventListener('click', () => {
6         let num = parseFloat(prompt("Ingrese un número:"));
7
8         let cuadrado = Math.pow(num, 2);
9         let cubo = Math.pow(num, 3);
10        let raiz = Math.sqrt(num);
11
12        out.textContent =
13        `Número: ${num}
14        Cuadrado: ${cuadrado}
15        Cubo: ${cubo}
16        Raíz cuadrada: ${raiz}`;
17    });
18 });
```

Laboratorio 14

Archivo C:/Users/aless/OneDrive/Deskto...

Esta página dice

Ingrese un número:

120

Aceptar Cancelar

6. Potencias y raíces

Ejecutar

Número: 120
Cuadrado: 14400
Cubo: 1728000
Raíz cuadrada: 10.954451150103322

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 $\text{radianes} = \text{grados} * (\pi / 180)$.

Crea una función `radianesAGrados(radianes)` que convierta ángulos de radianes a grados

Tip: usa la fórmula $\text{grados} = \text{radianes} * (180 / \pi)$

// Laboratorio Nro 14 - Ejercicio7

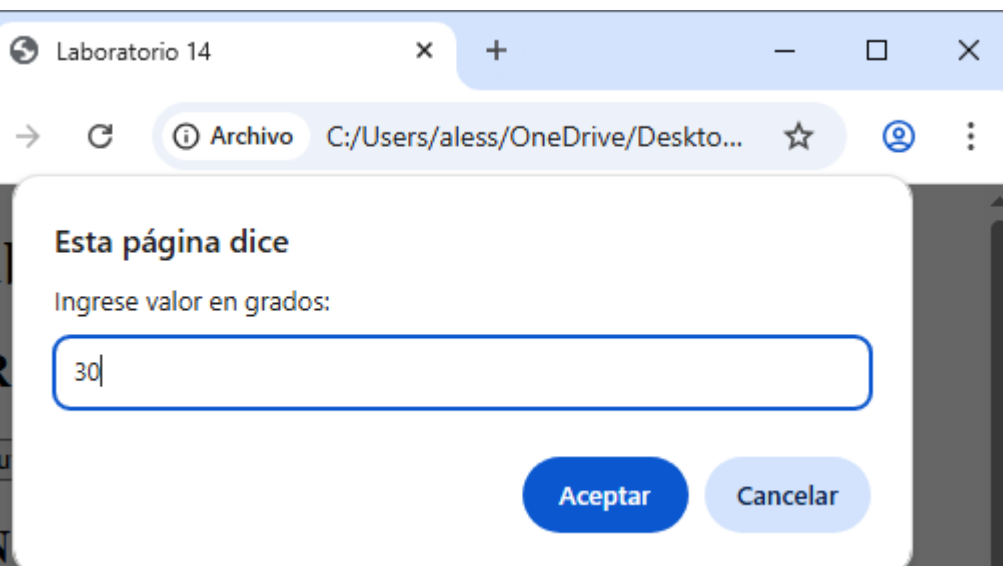
// Autor: Alessandro Josue Justo Vilca

// Colaboró : sin colaboradores

// Tiempo : 16min

js > JS ejercicio07.js > ...

```
1 document.addEventListener('DOMContentLoaded', () => {
2     const btn = document.getElementById('run-ej7');
3     const out = document.getElementById('out-ej7');
4
5     function gradosARadianes(grados) {
6         return grados * (Math.PI / 180);
7     }
8
9     function radianesAGrados(radianes) {
10        return radianes * (180 / Math.PI);
11    }
12
13    btn.addEventListener('click', () => {
14        let grados = parseFloat(prompt("Ingrese valor en grados:"));
15        let rad = gradosARadianes(grados);
16        let sinVal = Math.sin(rad);
17        let cosVal = Math.cos(rad);
18
19        out.textContent =
20        `Grados: ${grados}
21        Radianes: ${rad}
22        Seno: ${sinVal}
23        Coseno: ${cosVal}
24
25        Conversión inversa:
26        Radianes → Grados: ${radianesAGrados(rad)}`;
27    });
28 });
```



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  JS ejercicio8.js  JS ejercicio08.js  JS ejercicio07.js
js > JS ejercicio08.js > ...
1  document.addEventListener('DOMContentLoaded', () => {
2      const btn = document.getElementById('run-ej8');
3      const out = document.getElementById('out-ej8');
4
5      function generarPasswordNumerica() {
6          let pass = '';
7          for (let i = 0; i < 6; i++) {
8              pass += Math.floor(Math.random() * 10);
9          }
10         return pass;
11     }
12
13     btn.addEventListener('click', () => {
14         const password = generarPasswordNumerica();
15         out.textContent = `Contraseña numérica (6 dígitos): ${password}`;
16     });
17 });
```

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

js > JS ejercicio09.js > ...

```
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej9');
3   const out = document.getElementById('out-ej9');
4
5   function distanciaEntrePuntos(x1, y1, x2, y2) {
6     const dx = x2 - x1;
7     const dy = y2 - y1;
8     return Math.sqrt(dx * dx + dy * dy);
9   }
10
11   function distanciaAlOrigen(x, y) {
12     return Math.sqrt(x * x + y * y);
13   }
14
15   btn.addEventListener('click', () => {
16     const x1 = parseFloat(prompt('Ingrese x1 (ej: 3):', '3'));
17     const y1 = parseFloat(prompt('Ingrese y1 (ej: 4):', '4'));
18     const x2 = parseFloat(prompt('Ingrese x2 (ej: 7):', '7'));
19     const y2 = parseFloat(prompt('Ingrese y2 (ej: 1):', '1'));
20
21     const d = distanciaEntrePuntos(x1, y1, x2, y2);
22     const d1 = distanciaAlOrigen(x1, y1);
23     const d2 = distanciaAlOrigen(x2, y2);
24
25     out.textContent =
26 `Distancia entre (${x1}, ${y1}) y (${x2}, ${y2}) = ${d}
27 Distancia punto1 → origen = ${d1}
28 Distancia punto2 → origen = ${d2}
29 Suma distancias al origen = ${ (d1 + d2) }`;
30   });
31 });
```

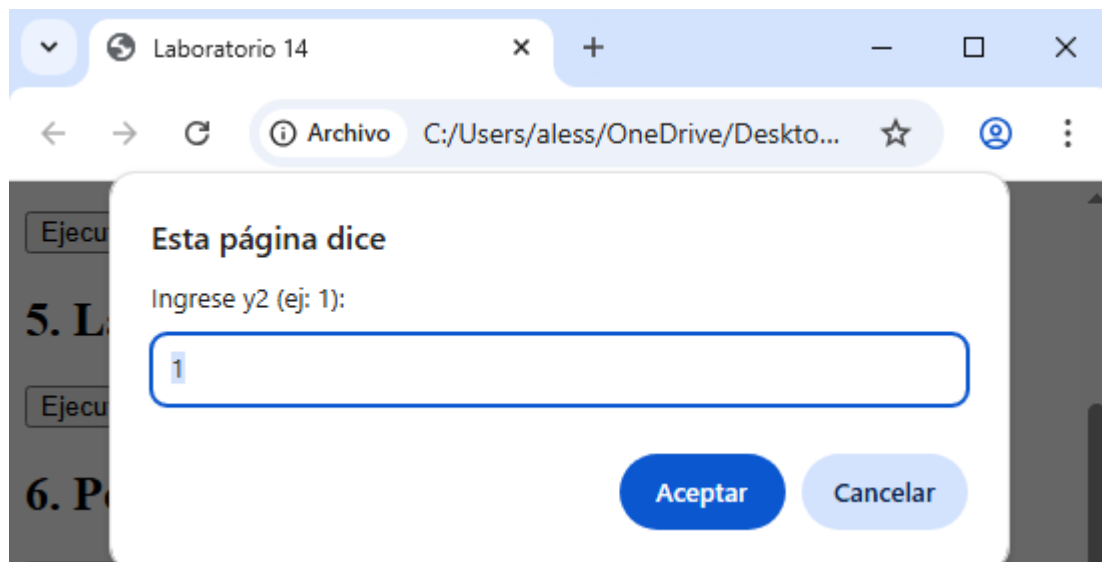
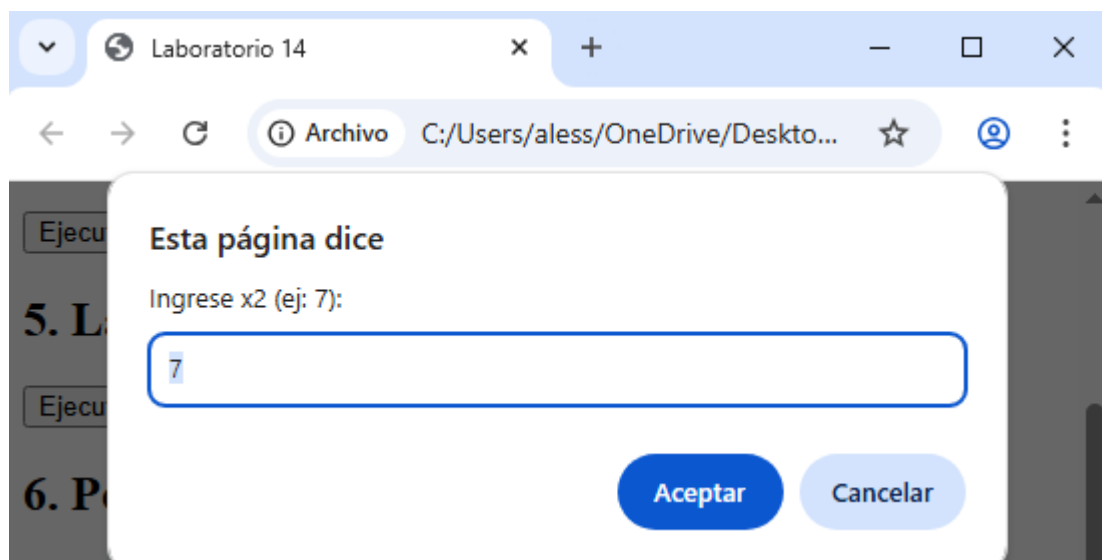
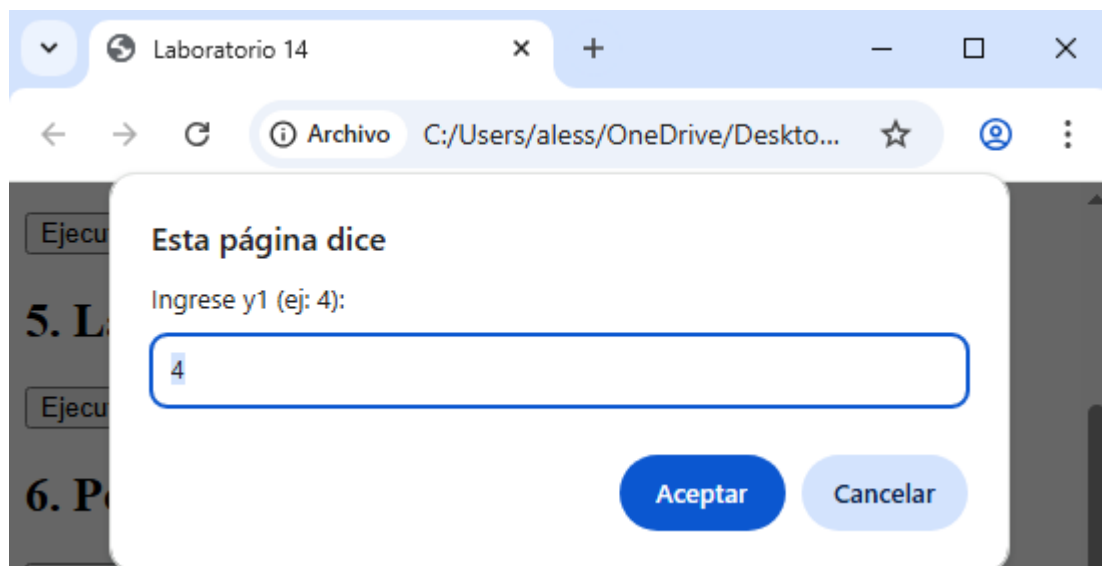
Esta página dice

Ingrese x1 (ej: 3):

3|

Aceptar

Cancelar



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)`.

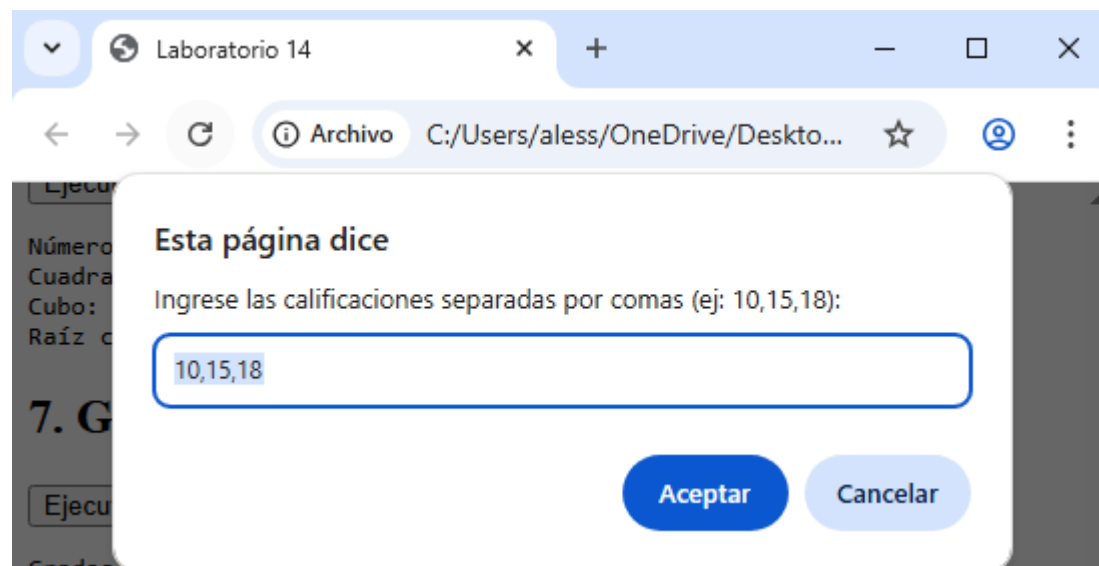
// Laboratorio Nro 14 - Ejercicio10

// Autor: Alessandro Josue Justo Vilca

// Colaboró : sin colaboradores

// Tiempo : 17min

```
<> index.html JS ejercic JS ejercicio10.js X JS ejercicio09.js JS ejercicio08.js JS ejer {}   
js > JS ejercicio10.js > ...  
1 document.addEventListener('DOMContentLoaded', () => {  
2   const btn = document.getElementById('run-ej10');  
3   const out = document.getElementById('out-ej10');  
4  
5   function normalizarCalificaciones(arr) {  
6     if (!arr.length) return [];  
7     const max = Math.max(...arr);  
8     if (max === 0) return arr.map(() => 0);  
9     return arr.map(n => n / max);  
10  }  
11  
12  btn.addEventListener('click', () => {  
13    const entrada = prompt('Ingrese las calificaciones separadas por comas (ej: 10,15,18):', '10,15,18');  
14    const arr = entrada.split(',').map(s => parseFloat(s.trim())).filter(n => !isNaN(n));  
15    const normalizadas = normalizarCalificaciones(arr);  
16    out.textContent =  
17    `Calificaciones: [${arr.join(', ')}]  
18    Máximo: ${Math.max(...arr)}  
19    Normalizadas (0-1): [${normalizadas.map(n => n.toFixed(4)).join(', ')}]`;  
20  });  
21  });
```



10. Normalización de calificaciones

Ejecutar

Calificaciones: [10, 15, 18]

Máximo: 18

Normalizadas (0-1): [0.5556, 0.8333, 1.0000]

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 JS ejercicio11.js JS ejercicio10.js JS ejercicio09.js JS ejer
js > JS ejercicio11.js > ...
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej11');
3   const out = document.getElementById('out-ej11');
4   class Producto {
5     #nombre;
6     #precio;
7     #stock;
8     constructor(nombre, precio, stock) {
9       this.#nombre = nombre;
10      this.setPrecio(precio);
11      this.setStock(stock);
12    }
13    setPrecio(v) {
14      const n = Number(v);
15      if (!isNaN(n) && n > 0) this.#precio = n;
16      else throw new Error('Precio debe ser número > 0');
17    }
18    setStock(v) {
19      const n = Number(v);
20      if (!isNaN(n) && n >= 0) this.#stock = n;
21      else throw new Error('Stock debe ser número >= 0');
22    }
23    vender(cantidad) {
24      const c = Number(cantidad);
25      if (isNaN(c) || c <= 0) throw new Error('Cantidad inválida');
26      if (c > this.#stock) return false;
27      this.#stock -= c;
28      return true;
29    }
30    info() {
31      return `Producto: ${this.#nombre}\nPrecio(raw): ${this.#precio}\nStock: ${this.#stock}`;
32    }
33  }
34  btn.addEventListener('click', () => {
35    try {
36      const p = new Producto('Arroz', 12.5, 20);
37      const antes = p.info();
38      const ok = p.vender(5);
39      const despues = p.info();
40      out.textContent = `${antes}\n\nvender(5) -> ${ok}\n\n${despues}`;
41    } catch (e) {
42      out.textContent = `Error: ${e.message}`;
43    }
44  });
45  });
```

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 JS ejercic JS ejercicio12.js JS ejercicio11.js JS ejer
js > 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

```
js > index.html JS ejercicio13.js JS ejercicio12.js JS ejercicio11.js
js > JS ejercicio13.js > document.addEventListener("DOMContentLoaded") callback > Triangulo
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej13');
3   const out = document.getElementById('out-ej13');
4
5   class Figura {
6     constructor() {}
7     area() { throw new Error('area() no implementado'); }
8     perimetro() { throw new Error('perimetro() no implementado'); }
9   }
10
11   class Cuadrado extends Figura {
12     constructor(lado) { super(); this.lado = lado; }
13     area() { return this.lado * this.lado; }
14     perimetro() { return 4 * this.lado; }
15   }
16
17   class Triangulo extends Figura {
18
19     constructor(base, altura, ladoA, ladoB, ladoC) {
20       super();
21       this.base = base;
22       this.altura = altura;
23       this.ladoA = ladoA; this.ladoB = ladoB; this.ladoC = ladoC;
24     }
25     area() { return (this.base * this.altura) / 2; }
26     perimetro() { return this.ladoA + this.ladoB + this.ladoC; }
27   }
28
29   btn.addEventListener('click', () => {
30     const c = new Cuadrado(5);
31     const t = new Triangulo(6, 4, 3, 4, 5);
32     out.textContent =
33     `Cuadrado (lado=5) -> Área: ${c.area()} | Perímetro: ${c.perimetro()}`
34     `Triángulo (base=6, altura=4, lados=3,4,5) -> Área: ${t.area()} | Perímetro: ${t.perimetro()}`;
35   });
36 });
```

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  JS ejercicio14.js  JS ejercicio14.js x  JS ejercicio13.js  JS ejercicio12.js  JS ejer
js > JS ejercicio14.js > ...
1  document.addEventListener('DOMContentLoaded', () => {
2      const btn = document.getElementById('run-ej14');
3      const out = document.getElementById('out-ej14');
4
5      class Usuario {
6          constructor(nombre, email) { this.nombre = nombre; this.email = email; }
7          mostrarInfo() { return `Nombre: ${this.nombre} | Email: ${this.email}`; }
8      }
9
10     class Cliente extends Usuario {
11         constructor(nombre, email, nivelFidelidad) {
12             super(nombre, email);
13             this.nivelFidelidad = Math.max(1, Math.min(5, Number(nivelFidelidad || 1)));
14         }
15         mostrarInfo() {
16             return `Cliente -> ${super.mostrarInfo()} | Nivel fidelidad: ${this.nivelFidelidad}`;
17         }
18     }
19
20     class Administrador extends Usuario {
21         constructor(nombre, email, permisos = []) {
22             super(nombre, email);
23             this.permisos = permisos;
24         }
25         mostrarInfo() {
26             return `Administrador -> ${super.mostrarInfo()} | Permisos: ${this.permisos.join(', ')}`;
27         }
28     }
29
30     btn.addEventListener('click', () => {
31         const c = new Cliente('Ana', 'ana@mail.com', 4);
32         const a = new Administrador('Luis', 'luis@mail.com', ['crear', 'editar', 'eliminar']);
33         out.textContent = `${c.mostrarInfo()}\n${a.mostrarInfo()}`;
34     });
35 }
```

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 JS ejercicio15.js JS ejercicio15.js X JS ejercicio14.js JS ejercicio13.js JS ejer {} [
js > JS ejercicio15.js > ...
1  document.addEventListener('DOMContentLoaded', () => {
2      const btn = document.getElementById('run-ej15');
3      const out = document.getElementById('out-ej15');
4
5      class Usuario {
6          constructor(nombre, email) { this.nombre = nombre; this.email = email; }
7          mostrarInfo() { return `${this.nombre} <${this.email}>`; }
8      }
9      class Cliente extends Usuario {
10         constructor(nombre, email, nivel) { super(nombre, email); this.nivel = nivel; }
11         mostrarInfo() { return `Cliente: ${this.nombre} | Nivel: ${this.nivel}`; }
12     }
13     class Administrador extends Usuario {
14         constructor(nombre, email, permisos) { super(nombre, email); this.permisos = permisos; }
15         mostrarInfo() { return `Administrador: ${this.nombre} | Permisos: ${this.permisos.join(', ')}'; }
16     }
17
18     btn.addEventListener('click', () => {
19         const usuarios = [
20             new Cliente('Mario', 'mario@mail', 2),
21             new Administrador('Laura', 'laura@mail', ['crear', 'eliminar']),
22             new Cliente('Pedro', 'pedro@mail', 5)
23         ];
24
25         const lines = usuarios.map(u => u.mostrarInfo());
26         out.textContent = lines.join('\n');
27     });
28 };
```

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 sobrescribe 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 JS ejercicio16.js JS ejercicio15.js JS ejercicio14.js JS ejer {}
js > 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  });
```

16. Empleado - Programador - Senior

Ejecutar

Programador: Carlos -> Sueldo calculado: 2200
ProgramadorSenior: Marta -> Sueldo calculado: 3750

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 JS ejercicio17.js JS ejercicio16.js JS ejercicio15.js JS ejer {}
js > JS ejercicio17.js > document.addEventListener('DOMContentLoaded') callback > btn.addEventListener('click') callback
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej17');
3   const out = document.getElementById('out-ej17');
4
5   class Cuenta {
6     #saldo;
7     constructor(saldoInicial = 0) { this.#saldo = Number(saldoInicial); }
8     depositar(monto) {
9       if (Number(monto) <= 0) throw new Error('Monto debe ser > 0');
10      this.#saldo += Number(monto);
11    }
12    retirar(monto) { throw new Error('retirar() debe implementarse en la subclase'); }
13    getSaldo() { return this.#saldo; }
14    transferir(destino, monto) {
15      if (!(destino instanceof Cuenta)) throw new Error('Destino inválido');
16      this.retirar(monto);
17      destino.depositar(monto);
18    }
19  }
20
21  class CuentaAhorro extends Cuenta {
22    retirar(monto) {
23      monto = Number(monto);
24      if (monto <= 0) throw new Error('Monto inválido');
25      if (monto > this.getSaldo()) throw new Error('Saldo insuficiente en CuentaAhorro');
26      // usar depositar con negativo para mantener encapsulación
27      this.depositar(-monto);
28    }
29  }
30
31  class CuentaCorriente extends Cuenta {
32    constructor(saldoInicial = 0, limiteSobregiro = 500) {
33      super(saldoInicial);
34      this.limite = Number(limiteSobregiro);
35    }
36    retirar(monto) {
37      monto = Number(monto);
38      if (monto <= 0) throw new Error('Monto inválido');
39      if (this.getSaldo() - monto < -this.limite) throw new Error('Excede límite de sobregiro');
40      this.depositar(-monto);
41    }
42  }
43
44  btn.addEventListener('click', () => {
45    try {
46      const ahorro = new CuentaAhorro(1000);
47      const corriente = new CuentaCorriente(200, 300);
48      ahorro.transferir(corriente, 300); // 1000-300, corriente 200+300
49      corriente.retirar(450); // permite sobregiro hasta -100 (200+300-450 = 50)
50      out.textContent =
51      `CuentaAhorro saldo: ${ahorro.getSaldo()}
52      CuentaCorriente saldo: ${corriente.getSaldo()}`;
53    } catch (e) {
54      out.textContent = `Error: ${e.message}`;
55    }
56  });
57 });

```

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 JS ejercici JS ejercicio18.js X JS ejercicio17.js JS ejercicio16.js ●
js > JS ejercicio18.js > ...
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej18');
3   const out = document.getElementById('out-ej18');
4
5   class Producto {
6     constructor(nombre, precio) { this.nombre = nombre; this.precio = Number(precio); }
7   }
8
9   class Carrito {
10    constructor() { this.productos = []; }
11    agregarProducto(producto) { this.productos.push(producto); }
12    calcularTotal() { return this.productos.reduce((s, p) => s + p.precio, 0); }
13    mostrarResumen() {
14      const lineas = this.productos.map(p => `${p.nombre} - ${p.precio.toFixed(2)} `);
15      lineas.push(`Total: ${this.calcularTotal().toFixed(2)} `);
16      return lineas.join('\n');
17    }
18  }
19
20  btn.addEventListener('click', () => {
21    const carrito = new Carrito();
22    carrito.agregarProducto(new Producto('Pan', 2.5));
23    carrito.agregarProducto(new Producto('Leche', 4.2));
24    carrito.agregarProducto(new Producto('Huevos', 6.0));
25    out.textContent = carrito.mostrarResumen();
26  });
27 });
```

18. Carrito de compras (composición)

Ejecutar

Pan - \$2.50
Leche - \$4.20
Huevos - \$6.00
Total: \$12.70

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 JS ejercici JS ejercicio19.js X JS ejercicio18.js JS ejercicio17.js JS e
js > JS ejercicio19.js > ...
1 document.addEventListener('DOMContentLoaded', () => {
2   const btn = document.getElementById('run-ej19');
3   const out = document.getElementById('out-ej19');
4
5   class Notificacion {
6     enviar() { return 'Notificación base'; }
7   }
8   class Email extends Notificacion {
9     constructor(destino, asunto) { super(); this.destino = destino; this.asunto = asunto; }
10    enviar() { return `Email -> To: ${this.destino} | Asunto: ${this.asunto}`; }
11  }
12  class SMS extends Notificacion {
13    constructor(destino, mensaje) { super(); this.destino = destino; this.mensaje = mensaje; }
14    enviar() { return `SMS -> To: ${this.destino} | Msg: ${this.mensaje}`; }
15  }
16  class Push extends Notificacion {
17    constructor(usuario, mensaje) { super(); this.usuario = usuario; this.mensaje = mensaje; }
18    enviar() { return `Push -> Usuario: ${this.usuario} | Msg: ${this.mensaje}`; }
19  }
20
21  function procesarNotificaciones(lista) {
22    return lista.map(n => n.enviar());
23  }
24
25  btn.addEventListener('click', () => {
26    const lista = [
27      new Email('a@mail.com', 'Bienvenido'),
28      new SMS('+51999999999', 'Codigo: 1234'),
29      new Push('usuario_1', 'Tu pedido ha llegado')
30    ];
31    out.textContent = procesarNotificaciones(lista).join('\n');
32  });
33 };
```

19. Notificaciones (polimorfismo)

Ejecutar

Email -> To: a@mail.com | Asunto: Bienvenido
SMS -> To: +51999999999 | Msg: Codigo: 1234
Push -> Usuario: usuario_1 | Msg: Tu pedido ha llegado

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

GitHub repository view for **laboratorio14-IDWeb** by **Ajustov**. The repository is public and has 1 branch (master) and 0 tags. The commit history shows a single commit: "Primer commit - laboratorio14 - ejercicio20" by Ajustov, dated 989886d, now. The file list includes `js` and `index.html`, both from the same commit. The README section prompts to "Add a README". A terminal window shows the following commands and output:

```
MINGW64/c/Users/aless/OneDrive/Desktop/Laboratorio14/alessandro_justo...
create mode 100644 js/ejercicio14.js
create mode 100644 js/ejercicio15.js
create mode 100644 js/ejercicio16.js
create mode 100644 js/ejercicio17.js
create mode 100644 js/ejercicio18.js
create mode 100644 js/ejercicio19.js

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$ git push -u origin master
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 8 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 8.32 KiB | 774.00 KiB/s, done.
Total 21 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Ajustov/laboratorio14-IDWeb.git
 * [new branch] master -> master
branch 'master' set up to track 'origin/master'.

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$
```

GitHub repository view for **laboratorio14-IDWeb** by **Ajustov**. The repository is public and has 1 branch (master) and 0 tags. The commit history shows two commits: "Segundo commit - laboratorio14 - ejercicio20" by Ajustov, dated 539f704, now, and "Primer commit - laboratorio14 - ejercicio20" by Ajustov, dated 989886d, now. The file list includes `js`, `LABORATORIO 14 - JAVASCRIPT_ ORIENTACI...`, and `index.html`. The README section prompts to "Add a README". A terminal window shows the following commands and output:

```
MINGW64/c/Users/aless/OneDrive/Desktop/Laboratorio14/alessandro_justo...
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 8 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 8.32 KiB | 774.00 KiB/s, done.
Total 21 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/Ajustov/laboratorio14-IDWeb.git
 * [new branch] master -> master
branch 'master' set up to track 'origin/master'.

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$ git add .

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$ git commit -m "Segundo commit - laboratorio14 - ejercicio20"
[master 539f704] Segundo commit - laboratorio14 - ejercicio20
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "LABORATORIO 14 \342\200\223 JAVASCRIPT_ ORIENTACI\303\223N A OBJETO5.pdf"

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$ git push
Enumerating objects: 4, done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.23 KiB | 243.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ajustov/laboratorio14-IDWeb.git
989886d..539f704 master -> master

aless@DESKTOP-T6G2CKT MINGW64 ~/OneDrive/Desktop/Laboratorio14/alessandro_justo/
laboratorio14 (master)
$
```

Build and test a Node.js project with npm.

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