Grupo 1 — Fundamentos y sintaxis (10 prácticas)

1) Eco inteligente (deriva de G1#1)

Enunciado: Pide un nombre y un número `veces`. Muestra `"Hola, <nombre>"` tantas veces. **Resultado esperado**: En consola se imprimen `veces` líneas. **Ayuda parcial**:

```
const nombre = prompt("Tu nombre:");
const veces = Number(prompt("¿Cuántas veces?:"));

for (let i = 0; i < /* TODO: límite */; i++) {
  // TODO: imprimir saludo usando template literal
}
```

2) Ticket de compra (deriva de G1#3 y #4)

Enunciado: Con `precio`, `cantidad` e `IVA=0.21`, calcula **subtotal**, **IVA** y **total** con 2 decimales. **Resultado esperado**: Tres líneas con los importes. **Ayuda parcial**:

```
const precio = Number(prompt("Precio unitario:"));
const cantidad = Number(prompt("Cantidad:"));
const IVA = 0.21;

const subtotal = /* TODO: precio * cantidad */;
const ivaImporte = /* TODO: subtotal * IVA */;
const total = /* TODO: subtotal + ivaImporte */;

console.log(`Subtotal: ${subtotal.toFixed(2)}€`);
console.log(`IVA: ${ivaImporte.toFixed(2)}€`);
console.log(`Total: ${total.toFixed(2)}€`);
```

3) Redondeo de notas (deriva de G1#5)

Enunciado: Pide una nota decimal y muéstrala redondeada a 1 decimal. Si la entrada no es número, avisa. **Resultado esperado**: `"Nota: 7.3"` o `"Entrada no válida"`. **Ayuda parcial**:

```
const n = Number(prompt("Nota (0-10):"));

if (Number.isNaN(n)) {
  console.log("Entrada no válida");
} else {
  // TODO: usar toFixed(1)
  console.log(/* TODO */);
}
```

4) Normalizador de DNI (deriva de G1#9)

Enunciado: A partir de `"  12345678a  "` obtén `"12345678A"` (sin espacios, letra en mayúscula). **Resultado esperado**: `12345678A`. **Ayuda parcial**:

```
const dni = prompt("DNI con letra:");
const limpio = dni.trim()./* TODO: quitar espacios internos */.toUpperCase();
console.log(limpio);
```

Pista: puedes usar `replaceAll(" ", "")` o una `regex` para quitar espacios.

5) Validador de contraseña simple (deriva de G1#6)

Enunciado: Pide una contraseña y valida: longitud ≥ 8 y contiene al menos un dígito. **Resultado esperado**: `"Fuerte"` o mensaje con requisitos faltantes. **Ayuda parcial**:

```
const pass = prompt("Contraseña:") ?? "";
const larga = pass.length >= 8;
const tieneNumero = /_____/.test(pass); // TODO: completa el patrón

if (larga && tieneNumero) {
  console.log("Fuerte");
} else {
  console.log("Falta:", [
    larga ? null : "8+ caracteres",
    tieneNumero ? null : "un número",
  ].filter(Boolean).join(", "));
}
```

6) Conversor de tiempo (deriva de G1#7)

Enunciado: Pide días y horas, y calcula los **segundos totales**. **Resultado esperado**: Para 1 día y 1 hora: `90000`. **Ayuda parcial**:

```
const dias = Number(prompt("Días:"));
const horas = Number(prompt("Horas:"));

const segundos = /* TODO: (dias * 24 + horas) * 3600 */;
console.log(segundos);
```

7) Intercambio de variables (deriva de G1#2)

Enunciado: Intercambia el contenido de `a` y `b` **sin variable temporal**. **Resultado esperado**: Antes: `a=5, b=8` → Después: `a=8, b=5`. **Ayuda parcial**:

```
let a = 5;
let b = 8;

// TODO: usa destructuring
[ /* TODO */ ] = [ /* TODO */ ];

console.log(a, b);
```

8) Plantilla de email (deriva de G1#4 y #9)

Enunciado: Con nombre y correo, genera: `"Hola, <Nombre>. Te escribimos a <email-normalizado>"`. **Resultado esperado**: Texto con email en minúsculas y sin espacios. **Ayuda parcial**:

```
const nombre = prompt("Nombre:")?.trim();
const email = prompt("Email:") ?? "";
const normalizado = email.trim().toLowerCase();

// TODO: usa template literal
console.log(/* TODO */);
```

9) Simulación de dado (deriva de G1#8)

Enunciado: Lanza un dado 10 veces, guarda los resultados y muestra la **media**. **Resultado esperado**: Array de 10 números y `Media: X.Y`. **Ayuda parcial**:

```
const tiradas = [];
for (let i = 0; i < 10; i++) {
  const d = Math.floor(Math.random() * 6) + 1;
  tiradas.push(d);
}

const suma = tiradas.reduce((acc, n) => acc + n, 0);
const media = /* TODO: suma / longitud */;
console.log(tiradas, `Media: ${media.toFixed(2)}`);
```

10) Parseo entero seguro (deriva de G1#5)

Enunciado: Implementa `parseEnteroSeguro(str)` que devuelva `null` si no hay entero válido. **Resultado esperado**: `parseEnteroSeguro("42") -> 42`, `parseEnteroSeguro("abc") -> null`. **Ayuda parcial**:

```
function parseEnteroSeguro(str) {
  const n = parseInt(str, 10);
  if (/* TODO: caso no numérico */) {
    return null;
  }
```

```
  return n;
}

console.log(parseEnteroSeguro("42"));
console.log(parseEnteroSeguro("abc"));
```

Grupo 2 — Control de flujo y bucles (10 prácticas)

1) Clasificador de temperatura (deriva de G2#1)

Enunciado: Dada una `t` (°C): `<0` **Frío**, `0–25` **Templado**, `>25` **Calor**. **Resultado esperado**: Mensaje según rango. **Ayuda parcial**:

```
const t = Number(prompt("Temperatura °C:"));

if (/* TODO: t < 0 */) {
  console.log("Frío");
} else if (/* TODO: 0 <= t && t <= 25 */) {
  console.log("Templado");
} else {
  console.log("Calor");
}
```

2) FizzBuzz personalizable (deriva de G2#4)

Enunciado: Pide `limite`, `a` y `b`. Imprime 1..`limite` reemplazando múltiplos de `a` por `Fizz`, de `b` por `Buzz` y de ambos por `FizzBuzz`. **Resultado esperado**: Secuencia correcta. **Ayuda parcial**:

```
const limite = Number(prompt("Límite:"));
const a = Number(prompt("Divisor A:"));
const b = Number(prompt("Divisor B:"));

for (let i = 1; i <= limite; i++) {
  const fizz = /* TODO: condición múltiplo de a */;
  const buzz = /* TODO: condición múltiplo de b */;

  if (/* TODO: ambos */) {
    console.log("FizzBuzz");
  } else if (fizz) {
    console.log("Fizz");
  } else if (buzz) {
    console.log("Buzz");
  } else {
    console.log(i);
  }
}
```

3) Máximo de tres (deriva de G2#2)

Enunciado: Pide tres números y muestra el **mayor** (si hay empate, indica "Empate"). **Resultado esperado**: Mayor o "Empate". **Ayuda parcial**:

```
const a = Number(prompt("A:"));
const b = Number(prompt("B:"));
const c = Number(prompt("C:"));

const max = Math.max(a, b, c);
const repeticiones = [a, b, c].filter((n) => n === max).length;

if (/* TODO: hay más de uno igual al max */) {
  console.log("Empate");
} else {
  console.log("Mayor:", max);
}
```

4) Menú cajero (deriva de G2#8)

Enunciado: Simula un cajero con opciones: 1) Ingresar, 2) Retirar, 3) Saldo, 0) Salir. Controla que el saldo no sea negativo. **Resultado esperado**: Flujo de operaciones hasta salir. **Ayuda parcial**:

```
let saldo = 0;
let op;

do {
  op = prompt("1 Ingresar / 2 Retirar / 3 Saldo / 0 Salir");

  if (op === "1") {
    const x = Number(prompt("Cantidad a ingresar:"));
    // TODO: sumar a saldo si es número válido
  } else if (op === "2") {
    const x = Number(prompt("Cantidad a retirar:"));
    // TODO: comprobar saldo suficiente antes de restar
  } else if (op === "3") {
    console.log(`Saldo: ${saldo}`);
  }
} while (op !== "0");
```

5) Suma hasta superar N (deriva de G2#5)

Enunciado: Suma números naturales comenzando en 1 hasta que la **suma** supere `N`. Muestra el último sumando y la suma final. **Resultado esperado**: Para `N=10` → `último=4`, `suma=10`. **Ayuda parcial**:

```
const N = Number(prompt("N:"));
let suma = 0;
```

```
let i = 0;

while (/* TODO: condición para seguir sumando */) {
  i += 1;
  suma += i;
}

console.log({ ultimo: i, suma });
```

6) Contar vocales y consonantes (deriva de G2#6)

Enunciado: Dado un texto, cuenta cuántas **vocales** y **consonantes** (solo letras) tiene. **Resultado esperado**: Objeto `{vocales: X, consonantes: Y}`. **Ayuda parcial**:

```
const t = (prompt("Texto:") ?? "").toLowerCase();
let v = 0;
let c = 0;

for (const ch of t) {
  if (!/[a-záéíóúñ]/.test(ch)) continue; // ignora no letras
  if (/* TODO: es vocal */) {
    v++;
  } else {
    c++;
  }
}

console.log({ vocales: v, consonantes: c });
```

7) Tabla de multiplicar formateada (deriva de G2#4)

Enunciado: Pide `n` y genera `"n x 1 = ..."` hasta `10`, cada resultado en una línea. **Resultado esperado**: 10 líneas con la tabla. **Ayuda parcial**:

```
const n = Number(prompt("Número:"));

for (let i = 1; i <= 10; i++) {
  // TODO: imprimir `n x i = resultado`
}
```

8) ¿Es primo? (deriva de G2#7)

Enunciado: Determina si un número `n` es primo. Optimiza recorriendo hasta `√n`. **Resultado esperado**: `true/false`. **Ayuda parcial**:

```
const n = Number(prompt("n:"));
```

```
if (n < 2) {
  console.log(false);
} else {
  let primo = true;
  for (let d = 2; d <= /* TODO: límite */; d++) {
    if (n % d === 0) {
      primo = false;
      break;
    }
  }
  console.log(primo);
}
```

9) Media de notas hasta cancelar (deriva de G2#9)

Enunciado: Ve pidiendo notas (0–10) hasta que el usuario **cancele**. Muestra la media (ignora entradas inválidas). **Resultado esperado**: `Media: X.Y` o mensaje si no hay datos. **Ayuda parcial**:

```
let suma = 0;
let count = 0;

while (true) {
  const s = prompt("Nota (cancelar para terminar)");
  if (s === null) break;
  const n = Number(s);
  if (!Number.isNaN(n) && n >= 0 && n <= 10) {
    // TODO: acumular y contar
  }
}

if (count === 0) {
  console.log("Sin datos válidos");
} else {
  console.log("Media:", /* TODO: suma / count */);
}
```

10) Triángulo de # (deriva de G2#4)

Enunciado: Dado `h`, dibuja un triángulo de altura `h` con `#`:

```
#
##
###
...
```

Resultado esperado: `h` líneas crecientes. **Ayuda parcial**:

```
const h = Number(prompt("Altura:"));
let linea = "";

for (let i = 1; i <= h; i++) {
  linea = /* TODO: añade un '#' */;
  console.log(linea);
}
```
