Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

Ejemplos de sentencia cíclicas e iterativas
WHILE, DO-WHILE, FOR

Todos y cada uno de los ejemplos abordan conceptos básicos y sencillos del manejo de ciclos e iteraciones, te propongo intentar en los casos que se pueda cambiar las que son while por for y viceversa (*recuerda no en todos los casos es posible*). Más adelante veremos otras estructuras de datos que nos permitirán el uso de for-each, for-of y for-in.

1. Generar recibos numerados

Uno de los ejemplos más sencillos es la construcción de un contador incremental para la generación de recibos consecutivos, para esto vamos a mostrar por consola la salida del algoritmo, y vamos a contar desde el 1 hasta el 50 los números generados con el formato “Recibo N° #” donde # es el número generado con el ciclo.

2. Generar tablas de multiplicar

Así cómo escribíamos las tablas de multiplicar en nuestros cuadernos cuando éramos pequeños, aquí abordaremos el uso de un ciclo repetitivo para construir una tabla de multiplicar, para ello solicitamos al usuario un número entero y construiremos dicha solución del 1 al 10.

3. Lista de pendientes

Permitir al usuario ingresar una frase que represente una tarea a realizar, el algoritmo nos debe permitir cargar tareas siempre que el usuario no ingrese una cadena de texto vacía. Durante el proceso de carga debemos mostrar por consola la tarea cargada y el orden en el que fue creada. Al final del proceso mostrar en un alert la cantidad de tareas proporcionadas.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

4. Cuenta regresiva de un temporizador

Simular un temporizador con una cuenta regresiva, imprimir por consola una cuenta regresiva desde 60 a 0.

5. Control de inventario

Simular el proceso de control de inventario en una tienda de venta de productos, para este proceso vamos a permitir al usuario ingresar el número de unidades de un producto vendido siempre y cuando queden existencias.

6. Calcular promedios

Suponer que recorremos la lista de calificaciones de un estudiante y debemos calcular el promedio de calificaciones de este. Para hacerlo vamos a crear un algoritmo que solicite N números (*N es un número entero que ingresamos por la interfaz*) y calcularemos el promedio con los valores ingresados. Tener en cuenta que no sabemos cuántas calificaciones va a ingresar el usuario, definir cuál es la condición de corte.

7. Calcular gastos

Imaginar que estamos haciendo las cuentas de fin de mes para evaluar nuestros ingresos y egresos, para este caso vamos a realizar un ciclo repetitivo donde ingresamos por ejemplo, 10 números, positivos o negativos, y el algoritmo debe calcular el resultado final de sumar o restar estos valores y decirnos cuánto es el resultado final.

8. Cálculo de intereses bancarios por año

Estás trabajando en el algoritmo que calcula los intereses sobre una cuenta, tú trabajo es iterar durante N años para calcular cómo crece el saldo con un interés fijo. Dónde el saldo es un número decimal, N es un número entero y el interés es un número decimal, proporcionados por el usuario.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

9. Simular pagos recurrentes

El gerente de una agencia de préstamos necesita un sistema para generar las cuotas para sus afiliados, para ello necesitaremos conocer el monto total del préstamo, el porcentaje de ganancia sobre el préstamo, y mediante este calcularemos cada uno de los pagos mensuales del préstamo por N meses. Dónde el valor del préstamo es un número decimal, el porcentaje es un decimal y N es el número de meses que escogió el afiliado. Para resolverlo considerar el sistema de interés compuesto.

10. Asistente de citas médicas

Suponer que trabajamos para el mantenimiento de un sistema de consultorios médicos, y una de las nuevas características que quieren añadir es la posibilidad de sugerir horarios de citas tomando en cuenta un horario de inicio, un horario de finalización y un intervalo. Para este caso vamos a suponer una atención de 08:00hs a 16:00hs y un intervalo de 20 minutos por consulta.

11. Simular consumo sobre débitos

Plantear un algoritmo para el control sobre el consumo sobre una tarjeta de débito, donde vamos a permitir al usuario ingresar un consumo siempre que cuente con saldo disponible para consumirlo.

12. Control de carga de combustible

Plantear la solución para un algoritmo que permita el control sobre la carga de combustible de un surtidor en una estación de servicio. Este planteo puede hacerse tanto en unidades cargadas o en monto de la compra, entonces el planteo es el siguiente, solicitar al operador que accione el gatillo del dispensador siempre que no haya llegado al límite de lo solicitado (*ya sea unidades en centímetros cúbicos o monto de moneda*), además para completar la tarea primero debemos solicitar al

usuario este primer dato que nos ofrece el límite de la carga (*de nuevo ya sea unidades en centímetros cúbicos o monto de moneda*).

13. Intentos de acceso permitidos

Suponer que trabajas sobre el módulo de autenticación de usuarios sobre un sistema y para el control de accesos se pide un número limitado de intentos de acceso por credenciales incorrectas. Para ello vamos a crear un algoritmo que simule esta operación y que cuente el número de intentos fallidos y nos impida el acceso definitivamente una vez alcanzado este límite (*por ejemplo 3 intentos*).

14. Simulación de facturación en una tienda

Plantéate los procesos que ocurren en la caja de un supermercado mientras realizas una compra, suponer que llegas con tú carrito de compras con los productos escogidos. Para plantear la solución, crearás un algoritmo en el cual vamos a cargar el nombre de un producto y el precio de este siempre y cuando el nombre del producto no sea una cadena vacía. Durante el proceso se deberá imprimir por consola el nombre del producto ingresado y el precio, al final debemos conocer el total de la compra.

15. Cálculo de deuda acumulada

Suponer que eres uno de los devs de un equipo dentro de un banco y estás a cargo de desarrollar el algoritmo que calcule y re-estructure la deuda sobre la deuda de una tarjeta de crédito. Para simular el cálculo de intereses acumulados mes a mes hasta que se pague vamos a necesitar conocer el saldo inicial y el porcentaje de incremento mediante el sistema compuesto (*se calcula sobre el último ya calculado y actualizado sobre el saldo*) si la deuda no se salda por completo.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

16. Validador de condiciones sobre una contraseña

El sub-sistema de control de acceso y registro sobre usuarios de un sistema requiere probar el algoritmo para validación de contraseña, para pasar la prueba el usuario debe seguir ingresando la contraseña siempre y cuando no tenga entre 6 y 13 caracteres.

17. Calcular potencias

Simula el funcionamiento de una calculadora donde puedes calcular la potencia de un número. Tener en cuenta que la fórmula para una potencia es la siguiente x^b donde *x* y *b* son números enteros. El algoritmo es la multiplicación de *x* por sí mismo tantas veces como lo indica *b*.

18. Temperaturas

En un centro de control meteorológico a final de mes se confeccionan informes que deben ser entregados, aquí mostrar el promedio de temperaturas durante el mes, la temperatura más elevada y la más baja (*y los días que ocurrió*). Para resolver esto, el operador deberá cargar las temperaturas una a una durante los últimos 30 días, y mientras se ingresan los datos, se debe llevar el cálculo de la suma total, identificar la temperatura más baja y la más alta (*y el día*), al final del proceso imprimir el promedio, la temperatura más baja y más alta y los días en los que ocurrió cada una de estas.

19. Juego de adivinanza

Simular un juego de adivinanzas donde el usuario recibirá un input en pantalla para indicar un número entero entre el 1 y el 100, si no acierta contra el generado pseudo-aleatoriamente por el algoritmo, se le debe pedir nuevamente y contar un intento como fallido, al final del proceso una vez adivine el número o alcance el límite de 10 intentos, se debe informar si adivinó o no el número generado originalmente y cuantos intentos fallidos posee si los tuviera.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

20. Descargar un archivo

Debemos monitorear el avance de carga de un archivo durante la subida a un servidor. Para ello vamos a simular el proceso ingresando el valor de avance en la descarga continuamente hasta llegar al total de bytes transmitidos, a medida que avanzamos debemos indicar el porcentaje de avance.

21. Jugamos un poco

Suponer que eres miembro de un equipo que desarrolla y mantiene un juego popular, en este juego se puede restaurar la salud (*vida*) de los personajes que han peleado en batalla. Para lograr esto posees una botella de posición que cura únicamente una cantidad específica de esta “salud”, digamos 128 puntos. Se propone desarrollar un algoritmo que permita al usuario ordenar restaurar la vida completa de este personaje o bien hasta donde se pueda con la cantidad de estas botellas de las que disponga.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

Resultados propuestos

1. Generar recibos numerados

```
for (let i = 1; i < 51; i++) {
    console.log("Recibo N° " + i.toString());
}
```

Explicación: el algoritmo inicia con un ciclo que define una variable de paso denominada como **i** con el valor inicial a **1**. Luego comienza a iterar hasta alcanzar el límite de 50 vueltas que está definido por **i < 5** dando saltos de 1 en 1 por **i++**. Durante cada vuelta que da al ciclo se imprime un mensaje en consola con el número del recibo concatenado.

2. Generar tablas de multiplicar

```
const numero = parseInt(prompt("Ingresa un número del 1 al 10 para construir su tabla de multiplicar."));

if (numero == NaN || !(numero >= 1 && numero <= 10)) {
    alert("Debes ingresar un número entero entre 1 y 10");
} else {
    // multiplicando --> i
    // multiplicador --> numero
    // producto      --> numero * i
    for (let i = 1; i <= 10; i++) {
        console.log(numero.toString() + " * " + i.toString() + " = " + (numero * i).toString());
    }
}
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

Explicación: primero antes de comenzar con la construcción de la tabla vamos a validar los datos proporcionados; en este caso el número; para ello verificamos que la función **parseInt()** haya logrado convertir el **número** y que este no sea un **NaN** (*Not a Number*), luego que se encuentre entre **1** y **10**. A continuación si pasa la validación, se procede con el ciclo, usando el contador de paso **i** como multiplicando, vamos a cambiar en cada ciclo el valor por el cual multiplicaremos en la fórmula final $f(x) => multiplicador \times multiplicando = producto$. Usaremos la función .toString() (*también puedes usar String(número) en su lugar*) en los casos donde tomemos los números y los querramos concatenar a nuestra cadena de resultado por consola.

3. Lista de pendientes

```
let tarea = prompt("Ingresa la tarea que quieres agregar a la lista");

let contador = 0;
while (tarea !== "") {
    contador++;
    console.log("Tarea " + String(contador) + ": " + tarea);
    // esta expresión es equivalente a
    // contador = contador + 1;
    // es muy importante volver a pedir el input del usuario sino caeremos en un
bucle infinito
    tarea = prompt("Ingresa la tarea que quieres agregar a la lista");
}

alert("Tareas pendientes en la lista " + String(contador));
```

Explicación: el usuario comienza describiendo la primera **tarea**, si esta no es una cadena vacía ingresamos al ciclo, incrementamos el **contador** en 1 e imprimimos en

la consola, luego volvemos a pedir al usuario la siguiente tarea, si no es una cadena vacía, se repite la operación de incrementar en 1 el contador, mostrar la tarea por la consola y volver a pedir un input al operador. Al final del proceso, cuando la tarea indicada es una cadena vacía, se corta el ciclo y se muestra con un **alert** por pantalla la cantidad de tareas escritas.

4. Cuenta regresiva de un temporizador

```
console.log("--> Inicio del contador <--");

for (let countdown = 60; countdown >= 0; countdown--) {
    console.log("Contador en " + countdown.toString());
}

console.log("--> Fin del contador <--");
```

Explicación: añadí mensajes en consola para indicar cuando inicia y cuando finaliza el proceso. Entre estos dos mensajes aparecerá la secuencia descendente de 60 a 0. Notar que, definimos la estructura distinta a como veníamos viendo, la variable de paso denominada **countdown** se inicia en 60, la condición de límite de pasos es **countdown >= 0** (*lo opuesto a lo que hicimos al principio*) y el paso en vez de incrementar (**++**), lo decrementamos (**--**) para hacer bajar **countdown** de 60 a 0.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

5. Control de inventario

```
let existencias = 100;

let cantidad = parseInt(prompt("Cantidad de productos vendidos"));

while (cantidad < existencias) {
    existencias = existencias - cantidad;
    console.log("Existencias reducida en " + cantidad.toString() + " unidades.");
    cantidad = parseInt(prompt("Cantidad de productos vendidos"));
}

alert("La existencias quedó con " + existencias.toString() + " unidades.");
```

Explicación: aquí suponemos que tenemos 100 unidades en existencia (*stock*), comenzamos el proceso preguntándole al operador la cantidad de existencias vendidas (*la cantidad de existencias que se deben restar de las existentes*), luego comenzamos con el ciclo solo si la cantidad ingresada es menor a las existencias actuales (*en la primera vuelta son 100*), si entramos primero vamos a restar de las unidades existentes la cantidad de unidades vendidas y actualizamos las actuales (*existencias = existencias - cantidad*), mostramos en consola con un mensaje su operación efectuada y volvemos a solicitar al usuario nuevamente una cantidad vendida y se vuelve a comprobar la condición inicial del ciclo pero esta vez con el la cantidad de unidades existentes actualizadas con el valor de la operación anterior. Al final mostraremos en cuántas unidades quedó al cortar con el ciclo.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

6. Calcular promedios

```
let calificacion = parseFloat(prompt("Calificación del estudiante"));
let sumatoria = 0;
let cantidad = 0;

while (calificacion > 0) {
    sumatoria = sumatoria + calificacion;
    cantidad++;
    calificacion = parseFloat(prompt("Calificación del estudiante"));
}
if (Boolean(cantidad)) {
    alert("El promedio de la/s " + cantidad.toString() + " calificacion/es del
estudiante es de " + (sumatoria / cantidad).toFixed(2));
} else {
    alert("No hay calificaciones con que calcular el promedio.");
}
```

Explicación: teniendo en cuenta que una calificación no podría ser cero o negativo, usaremos esto como condición de corte, por ello comenzamos el algoritmo solicitando al usuario un número decimal y si este es superior a cero, vamos a sumarlo a la sumatoria de calificaciones (*sumatoria = sumatoria + calificacion*), incrementamos el número de notas ingresadas en 1 (*cantidad++*) y volvemos a pedir al usuario una calificación más y volvemos a comprobar la condición de corte. Al final del proceso evaluamos si se han ingresado efectivamente calificaciones (*Boolean(cantidad)*), si esto es verdadero, vamos a mostrar un mensaje con la cantidad de calificaciones y el promedio calculado, de lo contrario informaremos lo opuesto.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

NOTA: la expresión *Boolean(cantidad)* se entiende como la conversión explícita de un número en un valor boolean, para los números valores superiores a cero se interpretan como true y aquellos iguales y menores a cero son interpretados como false.

7. Calcular gastos

```
let total = 0;


for (let index = 0; index < 10; index++) {
    const monto = parseFloat(prompt("Ingrese el monto del ingreso o egreso
número " + (index + 1).toString()));
    total = total + monto;
}
alert("El monto resultante del balance es $ " + total.toFixed(2));
```

Explicación: claro que el escenario debe ser más dinámico y no es necesario que el flujo sea estrictamente 10 valores únicamente, sino más bien dependerá de los ingresos y egresos de una persona cualquiera. Mientras el flujo esté en ejecución se irán solicitando los montos (*estos pueden ser positivos o negativos*) y se van acumulando en *total*, al final se muestra el resultado de hacer todas las sumatorias y restas.

8. Cálculo de intereses bancarios por año

```
const balance = parseFloat(prompt("Saldo de la cuenta"));
const years = parseInt(prompt("Cantidad de años"));
const interestRate = parseFloat(prompt("Porcentaje del interés expresado entre 0
y 100"));
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
const months = years * 12;

let total = balance;
const percent = interestRate / 100;

for (let index = 0; index < months; index++) {
    const increment = total * percent;
    total = total + increment;
}
alert("El saldo de la cuenta luego de " + years.toString() + " año/s es de $ " +
total.toFixed(2) + " con un interés de " + interestRate.toFixed(2) + "%");
```

Explicación: inicialmente necesitamos los datos de entrada, verdad? el saldo de la cuenta, la cantidad de años y el porcentaje de interés, estos se almacenan en *balance*, *years* y *interestRate*, ahora calcularemos los meses que equivalen a los años ingresados, multiplicando *years* por 12 y almacenando en *months*. Esto lo hacemos porque los intereses se acumulan mes a mes y no año a año. Luego transformamos el porcentaje entero en un valor decimal equivalente a 100/porcentaje, esto se hace para poder calcular el total de un valor respecto del porcentaje ingresado, luego acumulamos este valor de incremento al saldo total y repetimos la operación en el ciclo siguiente. Al final informamos los datos ingresados junto con el resultado del cálculo.

NOTA: para entender un poquito más lo de los porcentajes, por ejemplo el 50% de $ 800 es $ 400, esto se logra haciendo el siguiente cálculo $ 800 * (50/100), lo que resulta de la operación entre paréntesis es 0,5 lo que es equivalente al 50%, si nosotros multiplicamos $ 800 + 50 no daría como resultado $ 400, sino $ 40.000 y al final de cuentas deberíamos de dividir este resultado entre 100. Dicho de otra

manera, si vamos a usar el formato del porcentaje de 0 a 100 y no de 0 a 1, siempre vamos a necesitar dividir por 100.

9. Simular pagos recurrentes

```
const balance = parseFloat(prompt("Monto total del préstamo"));
const interestRate = parseFloat(prompt("Porcentaje del interés expresado entre 0 y 100"));
const months = parseInt(prompt("Cantidad de meses"));

let initialFeeValue = balance / months;
const percent = interestRate / 100;

let total = 0;
for (let month = 1; month <= months; month++) {
    const increment = initialFeeValue * percent;
    initialFeeValue = initialFeeValue + increment;
    total = total + initialFeeValue;
    console.log("Valor de la cuota N° " + month.toString() + " es de $ " + initialFeeValue.toFixed(2));
}
alert("Al final de el/los " + months.toString() + " mes/es usted habrá pagado $ " + total.toFixed(2));
```

Explicación: muy similar al ejercicio anterior, aquí vamos a pedir al usuario el monto del préstamo, el valor de interés y la cantidad de meses. Avanzamos calculando el valor de la cuota base inicial (*initialFeeValue = balance / months*), determinamos el porcentaje (*ya que está expresado en un rango entre 0 y 100*) e inicializamos el total a pagar en 0. Ahora comenzamos con el ciclo de cálculos repetitivos, por cada mes vamos a calcular el incremento y esto se lo vamos a

sumar a la cuota base inicial, actualizando en cada iteración del ciclo su valor para la próxima vuelta. Sumamos al total a pagar esta cuota y mostramos en pantalla el valor de la cuota calculada con el número de mes (cuota) al que corresponde. Luego al finalizar el proceso se muestra el valor total cobrado por la agencia una vez pasados los meses escogidos.

10. Asistente de citas médicas

```
const inicioEnHoras = prompt("Hora de inicio de intervalo expresado en formato ##:##. Dónde # es un número entero entre el 0 y el 9, por ejemplo para decir 08:00 escribiremos 08:00");
const finEnHoras = prompt("Hora de finalización de intervalo expresado en formato ##:##. Dónde # es un número entero entre el 0 y el 9, por ejemplo para decir 16:00 escribiremos 16:00");
const intervaloEnMinutos = parseInt(prompt("Intervalo expresado en minutos de 0 a 60"));

// Convertiremos los intervalos en minutos
const [horasInicio, minutosInicio] = inicioEnHoras.split(":").map(Number);
const inicioEnMinutos = horasInicio * 60 + minutosInicio;

const [horasFin, minutosFin] = finEnHoras.split(":").map(Number);
const finEnMinutos = horasFin * 60 + minutosFin;

// Generamos la lista propuesta de horarios
for (let tiempo = inicioEnMinutos; tiempo < finEnMinutos; tiempo += intervaloEnMinutos) {
    const horas = Math.floor(tiempo / 60);
    const minutos = tiempo % 60;
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
    console.log("--> " + horas.toString().padStart(2, '0') + ":" +
minutos.toString().padStart(2, '0'));
}
```

Explicación:

NOTA 1: El método **split** en JavaScript se utiliza para dividir una cadena (string) en un array de subcadenas, basándose en un separador especificado. Es útil para descomponer una cadena en partes más pequeñas, como palabras, números o caracteres.

NOTA 2: El método **map** en JavaScript es una función de los arrays que crea un nuevo array aplicando una función a cada elemento del array original. Es útil para transformar datos de forma sencilla y declarativa.

NOTA 3: El método **padStart** en JavaScript se utiliza para completar una cadena (*string*) con un carácter o conjunto de caracteres al principio, hasta que alcance una longitud deseada. Esto es útil para formatear cadenas, como agregar ceros a la izquierda de números o asegurar que un texto tenga una longitud específica.

11. Simular consumo sobre débitos

```
let saldo = parseFloat(prompt("Indique el saldo inicial de la cuenta"));

let monto = parseFloat(prompt("Ingrese el consumo requerido"));

while ((monto <= saldo) && (monto > 0) && (saldo > 0)) {
    saldo = saldo - monto;
    monto = parseFloat(prompt("Ingrese el consumo requerido"));
```

```
}


alert("La cuenta ha quedado con el saldo de $ " + saldo.toFixed(2));
```

Explicación: para conseguir simular el proceso, vamos a solicitar por prompt lo que nos retornaría la consulta por servicio, el saldo de la cuenta. Comenzamos entonces por el primer monto a debitar y luego vamos a hacer un ciclo repetitivo siempre que el monto solicitado pueda ser cubierto por el saldo de la cuenta, dicho de otra manera, el monto es menor al saldo. Por otro lado, incluimos dos condiciones más para considerar que no se trabajan con valores negativos (monto > 0 y saldo > 0). Al finalizar el proceso, se informa el saldo con el cual quedamos en la cuenta.

12. Control de carga de combustible

```
const litros = parseFloat(prompt("Indique la cantidad de litros a cargar."));
let centrimetrosCubicos = litros * 1000;
let volumen = parseFloat(prompt("Ingrese el volumen en cc a cargar."));
debugger
while ((volumen <= centrimetrosCubicos) && (volumen > 0) &&
(centrimetrosCubicos > 0)) {
    centrimetrosCubicos = centrimetrosCubicos - volumen;
    volumen = parseFloat(prompt("Ingrese el volumen en cc a cargar."));
}


alert("Cargaste un total de " + ((litros * 1000) - centrimetrosCubicos).toFixed(2) +
"cc");
```

Explicación: el planteo de este ejercicio es muy similar al anterior item, la diferencia es que manejamos otras unidades. Vamos a indicar la cantidad de litros que

queremos ingresar al tanque. Al final del proceso vamos a calcular el volumen original, vamos a restar el volumen que no se cargó y con esto sabremos cuánto es el volumen real que ingresamos a nuestro tanque de combustible.

13. Intentos de acceso permitidos

```
const RESTORED = "ILoveYou&2001";
const attemptsLimit = 3;
let intent = 1;
let password = prompt("¿Cuál es su contraseña?");

// Procesamos la contraseña
while ((password !== RESTORED) && (intent < attemptsLimit)) {
    console.error("Contraseña incorrecta, le restan " + (attemptsLimit - intent).toString() + " intentos.");
    intent++;
    password = prompt("¿Cuál es su contraseña?");
}

// Evaluamos si damos ingreso al sistema
if (password === RESTORED && intent < attemptsLimit) {
    alert("Bienvenido al sistema, ya puedes operar.");
} else {
    alert("Bloqueamos tú cuenta por alcanzar el límite de intentos, contacta con tú administrador.");
}
```

Explicación: simulamos que tenemos la contraseña del usuario recuperada de los servicios del sistema en la constante RESTORED. Comenzamos pidiendo al usuario su contraseña y evaluamos si esta es o no idéntica a la recuperada de la

almacenada originalmente, si no lo es, entonces ingresamos al bloque de código del ciclo y volvemos a pedir la contraseña nuevamente.

También vamos a evaluar si alcanzó el límite de intentos, en caso de haberlo alcanzado, no importa si la contraseña no es la almacenada, el ciclo se corta. Durante el proceso de pedir nuevamente la contraseña, se restan el número de intentos de los que lleva actualmente y se deja un registro de log en la consola para simular la consola del servidor.

Al final del proceso comprobamos si al final de cuentas la contraseña es validada o no, si es así le damos ingreso y sino le informamos que la cuenta fue bloqueada.

14. Simulación de facturación en una tienda

```
let total = 0;
let nombre = prompt("Nombre del producto");
debugger
while (nombre.trim() !== "") {
    let precioUnitario = parseFloat(prompt("Precio unitario para " + nombre.trim()));
    while (precioUnitario <= 0) {
        precioUnitario = parseFloat(prompt("Ingrese nuevamente el precio unitario
para " + nombre.trim() + ", el valor no puede ser un número negativo"));
    }
    total = total + precioUnitario;
    nombre = prompt("Nombre del próximo producto");
}

alert("Costo total de la suma de sus productos $ " + total.toFixed(2));
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

Explicación: Comenzamos solicitando el nombre del producto, si este no es vacío, continuamos con el proceso y solicitamos dentro del ciclo el precio unitario, como control adicional impedimos que el operador ingresa precios con número igual a cero o negativos, una vez que se pase por este proceso de validación, vamos a sumar al total el precio unitario proporcionado, luego volvemos a pedir el nombre del producto. Cuando el ciclo termine se va a informar el monto total de la suma de los precios unitarios cargados.

15. Cálculo de deuda acumulada

```
let saldo = parseFloat(prompt("Saldo de la cuenta"));
const interestRate = parseFloat(prompt("Porcentaje del interés por mes,
expresado entre 0 y 100"));
const percent = interestRate / 100;
let pago = parseFloat(prompt("¿Cuánto quiere abonar de la deuda?"));

while (pago > 0 && pago < saldo && saldo > 0) {
    saldo = saldo - pago;
    const incremento = saldo * percent;
    saldo = saldo + incremento;
    console.log("Se ha añadido un incremento de $ " + incremento.toString() + " a
su cuenta, quedando en un saldo de $ " + saldo.toFixed(2));
    pago = parseFloat(prompt("¿Cuánto quiere abonar de la deuda?"));
}

alert("El saldo final de la cuenta es de $ " + saldo.toFixed(2));
```

Explicación: simulamos con los dos primeros prompts la recuperación del saldo y el porcentaje de interés de servicios del sistema, luego avanzamos solicitando al operador el monto de pago del saldo de la cuenta. Para evitar valores negativos o

iguales a cero, validamos el monto del pago y el valor del saldo, si estas validaciones son efectivas, primero vamos a actualizar el saldo de la cuenta restándole el pago efectuado, luego al saldo que le queda le añadimos el incremento calculado y al final volvemos a pedir el próximo pago. Al final del proceso, informamos el saldo con el cual quedará la cuenta.

16. Validador de condiciones sobre una contraseña

```
const MIN_LENGTH = 6;
const MAX_LENGHT = 13;
let password = prompt("Ingrese la contraseña. Esta debe tener entre [" +
MIN_LENGTH.toString() + "-" + MAX_LENGHT.toString() + "] caracteres.");
debugger
while (password.length < MIN_LENGTH || password.length > MAX_LENGHT) {
    password = prompt("La contraseña ingresada no cumple con el criterio, vuelva
a intentarlo. Recuerde que debe tener entre [" + MIN_LENGTH.toString() + "-" +
MAX_LENGHT.toString() + "] caracteres.");
}

alert("Contraseña validada satisfactoriamente.");
```

Explicación: para comenzar arrancamos definiendo los límites propuestos para la cantidad de caracteres, lo hacemos en variables constantes para poder usarlas en los mensajes. Solicitamos al usuario su contraseña, si no cumple con las condiciones solicitadas, ingresamos al cuerpo del bucle y volvemos a pedirla, ahora con un mensaje diferente. Al final, el algoritmo deja de ejecutarse cuando el usuario ingresa una contraseña cuya cantidad de caracteres se encuentre dentro de los límites establecidos y se informa con un mensaje en pantalla.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

17. Calcular potencias

```
const base = parseInt(prompt("Ingresa la base."));
const exponente = parseInt(prompt("Ingresa el exponente."));
let resultado = 0;

// Si el exponente es 0, la potencia de cualquier número es 1
if (exponente === 0) {
    resultado = 1
} else {
    const esNegativo = exponente < 0;
    // convertimos el valor a su equivalente absoluto para determinar el número de iteraciones.
    const parsedExp = Math.abs(exponente);
    // Calculamos la potencia con un ciclo for
    resultado = base;
    // aquí hacemos -1 al exponente porque la primera iteración la hicimos al darle a resultado el valor inicial de la base
    for (let i = 0; i < (parsedExp - 1); i++) {
        resultado = resultado * base;
        // si operación equivalente es resultado*= base;
    }

    // Si el exponente es negativo, trabajamos con el inverso
    if (esNegativo) {
        resultado = 1 / resultado;
    }
}
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
alert("La base " + base.toString() + " elevado a " + exponente.toString() + " es igual a: " + resultado.toString());
```

Explicación: comenzamos el algoritmo solicitando al usuario la base y el exponente, luego comprobamos el exponente, si es 0, devolvemos 1, sino, iniciamos el proceso, evaluamos si el exponente es negativo, luego lo convertimos a su valor absoluto (*ya que necesitamos el número de iteraciones, no el signo del valor*). Antes de comenzar el ciclo, vamos a asignarle al resultado el valor inicial de la base, y al ciclo en total le restamos una iteración por esta inicialización que acabamos de efectuar. Al final del proceso si el exponente originalmente era negativo expresamos la solución con su inversa y si no mostramos directamente el resultado.

18. Temperaturas

```
const CANTIDAD_DIAS = 30;

let temperatura = parseFloat(prompt("Ingrese temperatura para el día 1"));
console.log("Temperatura ingresada para el día 1: " + temperatura.toFixed(2));

// aquí tomamos los valores iniciales (primera iteración)
let suma = temperatura;
let temp_mas_alta = temperatura;
let dia_temp_alta = 1;
let temp_mas_baja = temperatura;
let dia_temp_baja = 1;

for (let dia = 2; dia <= CANTIDAD_DIAS; dia++) {
    temperatura = parseFloat(prompt("Ingrese temperatura para el día " + dia.toString()));
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
    console.log("Temperatura ingresada para el día " + dia.toString() + ": " +
temperatura.toFixed(2));
    suma = suma + temperatura;
    if (temperatura > temp_mas_alta) {
        temp_mas_alta = temperatura;
        dia_temp_alta = dia;
    } else if (temperatura < temp_mas_baja) {
        temp_mas_baja = temperatura;
        dia_temp_baja = dia;
    }
}

const promedio = suma / CANTIDAD_DIAS;
alert("INFORME:\nPromedio de temperaturas: " + promedio.toFixed(2) +
"\nTemperatura más baja: " + temp_mas_baja.toFixed(2) + " el día " +
dia_temp_baja + "\nTemperatura más alta: " + temp_mas_alta.toFixed(2) + " el día
" + dia_temp_alta);
```

Explicación: arrancamos definiendo la cantidad de días en una constante. Luego le solicitamos al usuario la temperatura del primer día, usamos este como valor inicial para las temperaturas más alta y baja, respectivamente, incluyendo sus días y el valor inicial de la suma de temperaturas. Luego avanzaremos con el ciclo repetitivo, esta vez, a partir del día 2, ya que el primero ya fue contabilizado al inicializar las mencionadas. Solicitamos al usuario la temperatura del día correspondiente, la sumamos a la variable que sumariza todas para luego calcular el promedio, evaluamos si el valor ingresado es la más alta o la más baja y avanzamos. Al finalizar el ciclo, calculamos el promedio y lo incluimos en el informe en pantalla.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

NOTA: para conseguir que en el informe se hagan saltos de línea (*ver cada ítem uno debajo del otro*), usamos “\n”.

19. Juego de adivinanza

```
const MIN_LIMIT = 0;
const MAX_LIMIT = 100;
const ATTEMPTS_LIMIT = 10;
const GENERATED = Math.floor(Math.random() * MAX_LIMIT) + 1;// generar un número pseudo-aleatorio
let attempts = 0;
let value;

do {
    value = parseInt(prompt("Dime un número entre el " + MIN_LIMIT.toString() + " y el " + MAX_LIMIT.toString() + "."));

    // Validar la entrada
    while (isNaN(value) || value < MIN_LIMIT || value > MAX_LIMIT) {
        alert("Por favor, ingresa un número válido entre " + MIN_LIMIT.toString() + " y " + MAX_LIMIT.toString() + ".");
        value = parseInt(prompt("Dime un número entre el " + MIN_LIMIT.toString() + " y el " + MAX_LIMIT.toString() + "."));
    }
    if (value !== GENERATED && value < GENERATED) {
        alert("Por debajo! Intenta con un número más alto.");
    } else if (value !== GENERATED && value > GENERATED) {
        alert("Por arriba! Intenta con un número más bajo.");
    }
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
    attempts++;
} while ((value !== GENERATED) && (attempts < ATTEMPTS_LIMIT));

if ((value === GENERATED) && (attempts < ATTEMPTS_LIMIT)) {
    alert("¡Felicidades! Adivinaste el número con " + attempts.toString() + "
intentos.");
} else {
    alert("¡Lo siento! Superaste el límite de " + attempts.toString() + " intentos
permitidos sin adivinarlo.\nEl número era " + GENERATED.toString());
}
```

Explicación: las variables constantes en mayúsculas son los parámetros establecidos para el algoritmo, GENETARED es el número generado con la librería Math (*puedes investigar qué herramientas tiene para ofrecerte*). Comenzamos solicitando el primer número, lo convertimos en un entero y validamos que sea un número, que no esté fuera de los límites establecidos.

Como una forma de ayudar a nuestro usuario, evaluamos si el número ingresado está por debajo o por encima del generado, para darle una sugerencia de en qué dirección ir.

Luego incrementamos el número de intentos, evaluamos la condición del do-while para ver si lo repetimos o vamos al final del algoritmo. Al final evaluamos si adivino la palabra y le dejamos un mensaje en la condición que corresponda.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

20. Descargar un archivo

```
let BYTES_TOTALES = parseInt(prompt("Tamaño total del archivo medido en bytes."));

// Validación del tamaño total del archivo
while (isNaN(BYTES_TOTALES) || BYTES_TOTALES <= 0) {
    alert("El tamaño del archivo debe ser un número entero superior a 0 (cero).");
    BYTES_TOTALES = parseInt(prompt("Tamaño total del archivo medido en bytes."));
}

let totalTransmitido = 0;
debugger
while (totalTransmitido < BYTES_TOTALES) {
    let recibidos = parseInt(prompt("Indica la cantidad de bytes recibidos"));

    // Validar el recibidos ingresado
    while (isNaN(recibidos) || recibidos < 0) {
        alert("Los bytes recibidos deben ser un número entero superior a 0 (cero).");
        recibidos = parseInt(prompt("Indica la cantidad de bytes recibidos"));
    }

    totalTransmitido = totalTransmitido + recibidos;

    // En caso de ingresar un valor superior, igualarlos
    if (totalTransmitido > BYTES_TOTALES) {
        totalTransmitido = BYTES_TOTALES;
    }
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
    const PORCENTAJE = (totalTransmitido / BYTES_TOTALES) * 100;

    // Mostrar el estado de recibidos
    alert("Progreso lleva " + PORCENTAJE.toFixed(2) + "%  del progreso.");
}

alert("Archivo descargado satisfactoriamente.");
```

Explicación: primero vamos a solicitar para simular, el total de bytes que tiene el archivo en total, para saber hasta cuánto debemos avanzar. Ingresamos al ciclo, solicitamos el total de progreso en bytes, y luego validamos si el número ingresado es válido. A continuación sumamos los bytes recibidos al total acumulado, y luego evaluamos si llegamos al límite de descarga o no. Al final dentro del bucle calculamos el porcentaje y lo informamos al usuario. Al finalizar el algoritmo vamos a avisar que la descarga ha finalizado.

21. Jugamos un poco

```
const MEDICINE_BOTTLE_UNIT = 30;
const HEALING_CAPACITY = 128;

let totalHealth = parseInt(prompt("Salud total del personaje"));

// Validación del total de salud
while (isNaN(totalHealth) || totalHealth <= 0) {
    alert("La salud del personaje debe ser un número entero superior a 0 (cero).");
    totalHealth = parseInt(prompt("Ingresa nuevamente la salud total del
personaje"));
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
}

let damage = parseInt(prompt("Daño recibido"));

// Validación del total de daño
while (isNaN(damage) || damage <= 0 || damage >= totalHealth) {
    alert("El daño recibido por el personaje debe ser un número entero superior a 0 (cero) e inferior a " + totalHealth.toString() + ".");
    damage = parseInt(prompt("Ingresa nuevamente el daño recibido"));
}

let health = totalHealth - damage;
const NEEDED_BOTTLES = Math.ceil(damage / HEALING_CAPACITY);// redondeamos hacia arriba

let limit = NEEDED_BOTTLES;
if (NEEDED_BOTTLES > MEDICINE_BOTTLE_UNIT) {
    limit = MEDICINE_BOTTLE_UNIT;
}

for (let dose = 0; dose < limit; dose++) {
    // Validamos que no pasemos del límite total de la slaud del personaje al aplicar la medicina
    if ((health + HEALING_CAPACITY) > totalHealth) {
        health = totalHealth;
    } else {
        health = health + HEALING_CAPACITY;
    }
```

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

```
    console.log("Restaurando " + health.toString() + " de " + totalHealth.toString());
    // Evaluamos que el daño registrado no sea un número negativo
    if ((damage - HEALING_CAPACITY) < 0) {
        damage = 0
    } else {
        damage = damage - HEALING_CAPACITY;
    }
}

alert("Salud del personaje restaurada al 100% con " + limit.toString() + " botella/s
de medicina.");
```

Explicación: comenzamos el algoritmo definiendo la capacidad de curación de cada botella de posición (*HEALING_CAPACITY*) y la cantidad de unidades de esta que disponemos (*MEDICINE_BOTTLE_UNIT*), Luego para poder hacer cálculos solicitamos la capacidad de salud total del personaje y la de daño recibido, ambos valores se validan para que sean números, no sean negativos y no superen límites, luego calculamos el nivel de salud actual (*health*), a continuación determinamos el número de botellas de medicina necesarios para cubrir el 100% de la salud faltante (*NEEDED_BOTTLES*), evaluamos si contamos con el número necesario de botellas o no y si no es así decimos que el límite con el que contamos son las unidades que disponemos.

Una vez hechos todos los cálculos necesarios, comenzamos con el proceso de curación, consistirá en un ciclo de aplicaciones de medicina una botella a la vez. Para ello conocemos el número total de dosis a aplicar, y en cada aplicación vamos a restaurar la salud total (*evitando dar más salud de la que puede tener*) y disminuir el daño (*evitando disminuir el daño más allá de 0*). Al final informamos el resultado del proceso.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

APARTADO 1: Aplicación de Break y Continue

Break: en cada uno de estos ejemplos, el uso de break permite detener el ciclo tan pronto como se encuentra lo que se busca, ahorrando tiempo y recursos de ejecución.

Continue: en cada ejemplo, continue se usa para omitir elementos no deseados y pasar al siguiente elemento en el ciclo, sin interrumpir completamente la iteración.

1. Búsqueda en una lista de nombres

Imagina que tienes una lista de nombres y necesitas encontrar un nombre específico. Una vez encontrado, no necesitas seguir buscando.

2. Detectar el primer número impar en una lista de números

Supongamos que estás revisando una lista para encontrar el primer número impar. No es necesario seguir revisando después de encontrarlo.

3. Ignorar números negativos en un cálculo

Estás sumando los números de una lista, pero quieres ignorar los números negativos.

4. Verificar si un ingrediente está disponible en la despensa

Si tienes una lista de ingredientes que necesitas, y buscas si un ingrediente específico está en tu despensa

5. Saltar palabras cortas al procesar un texto

Procesas una lista de palabras, pero deseas ignorar aquellas que tengan menos de 4 letras.

Para mis estudiantes del curso de JavaScript de Coder House. El presente material NO es responsabilidad de Coder House y NO tengo su consentimiento expreso para difundirlo como material propio del programa al que se inscribieron, por ello es que es un aporte de mi para ustedes, como material complementario con la intención de ofrecerles propuestas que considero les podrá ayudar a explorar los temas.

PUC / ASC KRUJOSKI, Saúl A.

6.Encontrar una parada específica en un trayecto de autobús

Supón que tienes una lista de paradas de autobús y estás buscando si una parada específica está incluida en el trayecto.

7. Comprobar si un archivo en una lista es del tipo correcto

Estás buscando en una lista de nombres de archivos el primero que sea del tipo .pdf.

8. Saltar un ingrediente no disponible al cocinar

Revisas una lista de ingredientes para preparar un platillo, pero si un ingrediente no está disponible, simplemente lo ignoras.

9. Filtrar correos no válidos de una lista

Estás procesando una lista de correos electrónicos, pero quieres saltar aquellos que no contengan un @.

10. Saltar días lluviosos al planear actividades al aire libre

Revisas un calendario y solo quieres mostrar los días soleados para actividades al aire libre.

11. Generar lista de estudiantes

Creas una lista de estudiantes con aquellos que poseen una calificación igual o superior a 8.