**Indice**

1. [JavaScript: Primeros Pasos](#_JavaScript: Primeros pasos con el lenguaje)
2. [Js en la web I: DOM](#_JS en la Web: Manipulación del DOM con JavaScript)
3. [Js en la web II: Forms](#_JS en la Web: Validación de Formularios y HTML5)
4. [JavaScript: POO](#_JavaScript: Introducción a la Orientación a Objetos)
5. [JavaScript: Herencia e Interfaces](#_JavaScript: Herencia e Interfaces en Orientación a Objetos)
6. [Js en la web III: Almacenando datos en el nav](#_JS en la Web: Almacenando datos en el navegador)
7. [JS en la web IV: CRUD](#_JS en la Web: CRUD con JavaScript asíncrono)

# ****JavaScript: Primeros pasos con el lenguaje****

console.log('Mi primer programa');

console.log(2+2);

const nombreMadre = "María";

const nombrePadre = "José";

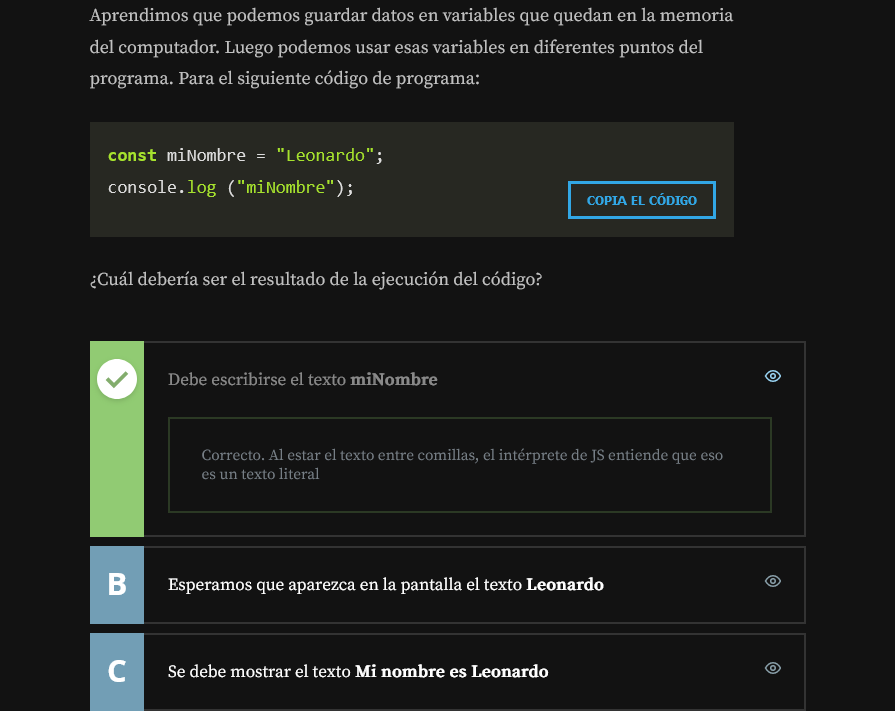
const nombreHijo = "Jesús";

const edadMadre = 60;

console.log(nombreMadre);

console.log("El nombre de mi madre es "+nombreMadre);

console.log("La edad de mi madre es "+edadMadre);





const nombreMadre = "María";

const NombreMadre = "Teresa";

console.log(nombreMadre);

console.log(NombreMadre);

const nombreDeMiMadreCuandoSoltera = "María Perez";

const nombreDeMiPais = "Venezuela";

console.log(nombreDeMiMadreCuandoSoltera);

console.log(NombreMaria);

const edadMadre = 60;

const pi = 3.1416;

const esSoltero = false;

let miEdad = 40;

Operaciones Aritmeticas:

console.log("Operaciones aritmeticas");

const variableTexto = "variableTexto";

console.log(variableTexto);

let nuevaVariableTexto = "Valor que cambia";

console.log(nuevaVariableTexto);

nuevaVariableTexto = "Por otro valor";

console.log(nuevaVariableTexto);

var variableGlobal = "Varible de todo el programa";

console.log(variableGlobal);

variableGlobal = "Otro valor global";

console.log(variableGlobal);

const unNumero = 10;

const numeroDecimal = 20.5;

let sumaNumeros = 0;

sumaNumeros = 2 + 2;

console.log(sumaNumeros);

sumaNumeros = 2 + 10 \* 5;

console.log(sumaNumeros);

sumaNumeros = (2 + 10) \* 5;

console.log(sumaNumeros);

const nombreMadre = "María";

console.log("El nombre de mi madre es:" + nombreMadre);

Conversion tipo datos:

console.log("Conversion de Tipos de datos");

console.log("10" + "2");

console.log(parseInt("10") + parseInt("2"));

console.log("10" / "2");

console.log("Maria" / "2");

Asignacion de Variables:

console.log("Asignando valores a las variables");

const miNombre = "Leonardo";

const miApellido = "Lacruz";

console.log(miNombre);

const miNombreCompleto = miNombre + " " + miApellido;

console.log(miNombreCompleto);

const textoMiNombre = `Mi nombre es: ${miNombre} ${miApellido}`;

console.log(textoMiNombre);

const textoMiNombreDobles = "Mi nombre es: ${miNombre} ${miApellido}";

console.log(textoMiNombreDobles);

const textoMiNombreSimples = "Mi nombre es: ${miNombre} ${miApellido}";

console.log(textoMiNombreSimples);

let variableTexto = "Un texto";

console.log(variableTexto);

variableTexto = 10;

console.log(variableTexto);

variableTexto = 10.5;

console.log(variableTexto);

variableTexto = false;

console.log(variableTexto);



Trabajando con Listas:

console.log(`Trabajando con Listas`);

const Bogota = "Bogotá";

const Lima = "Lima";

const Santiago = "Santiago";

const BuenosAires = "Buenos Aires";

//console.log(Bogota,Lima,Santiago,BuenosAires);

//const ciudadesDisponibles = ['Bogotá','Lima','Santiago','Buenos Aires'];

const ciudadesDisponibles = **new** *Array*();

ciudadesDisponibles.push("Bogotá");

ciudadesDisponibles.push("Caracas");

ciudadesDisponibles.push("Lima");

console.log(ciudadesDisponibles);

const ciudadesMasVendidas = [];

ciudadesMasVendidas.push("São Paulo");

ciudadesMasVendidas.push("Buenos Aires");

console.log(ciudadesMasVendidas);

console.log(ciudadesDisponibles[2]);

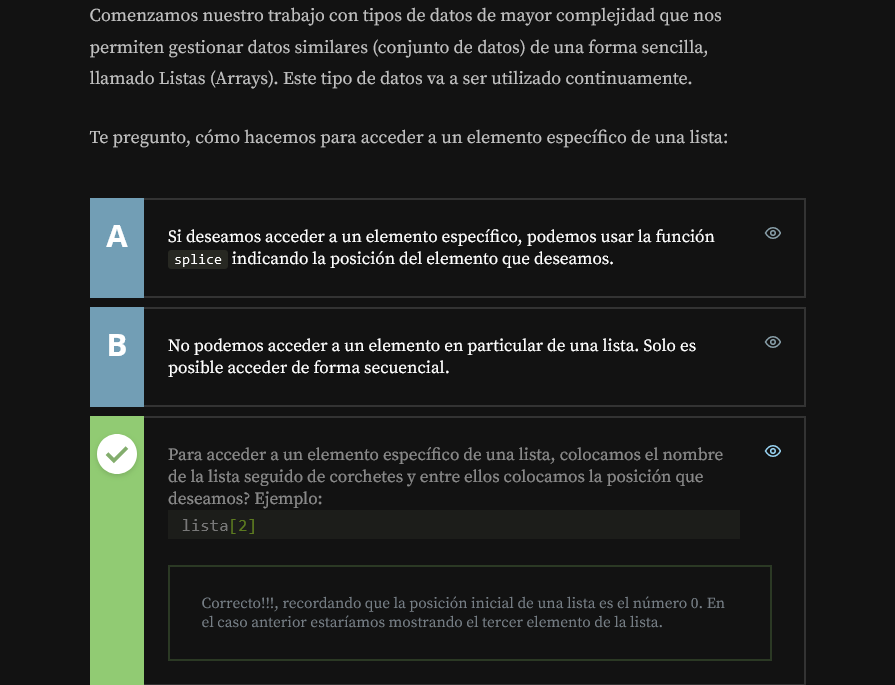
console.log(ciudadesMasVendidas[0]);

const cantidadCiudades = ciudadesDisponibles.length;

console.log(`Tenemos ${cantidadCiudades} ciudades disponibles`);

ciudadesDisponibles.splice(1);

console.log(ciudadesDisponibles);



Condicionales:

console.log(`Trabajando con condiciones`);

const ciudadesDisponibles = **new** *Array*();

ciudadesDisponibles.push("Bogotá");

ciudadesDisponibles.push("Caracas");

ciudadesDisponibles.push("Lima");

const edadComprador = 15;

const estaAcompanado = true; //Variable booleana: true o false

const tienePasaje = true;

if (edadComprador >= 18 || estaAcompanado) {

    //Aqui se ejecuta el código si la edad del comprador es mayor a 18

    //Es decir es mayor de Edad

    console.log(`Es posible vender pasajes`);

    console.log(`Ciudades disponibles para viaje`);

    console.log(ciudadesDisponibles);

} else {

    console.log(`El comprador NO es mayor de edad y NO está acompañado`);

    console.log(`No es posible ofrecer viajes`);

}

//Proceso de embarque

console.log(`Proceso de embarque: \n \n`);

//       A       Y            B     (C  O   D)

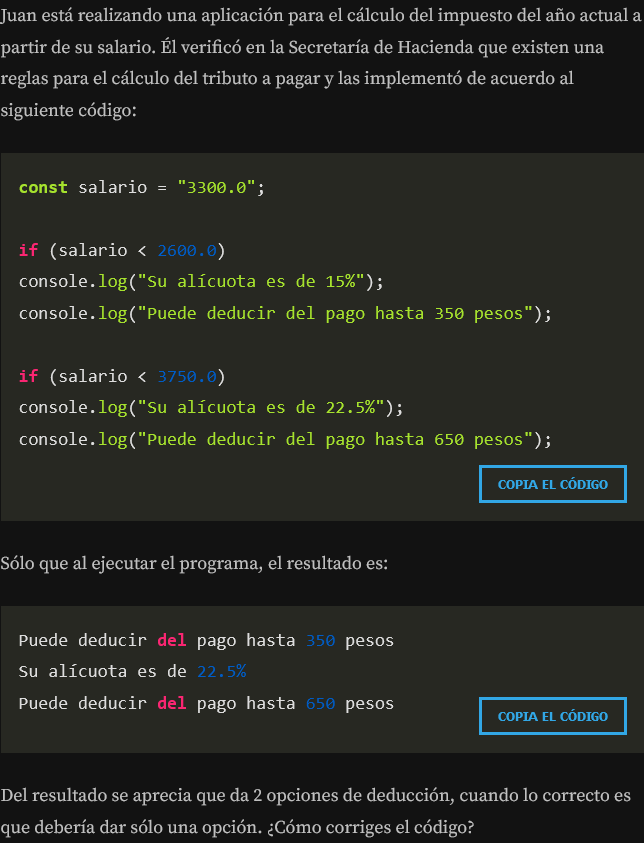
if (tienePasaje && (edadComprador >= 18 || estaAcompanado)) {

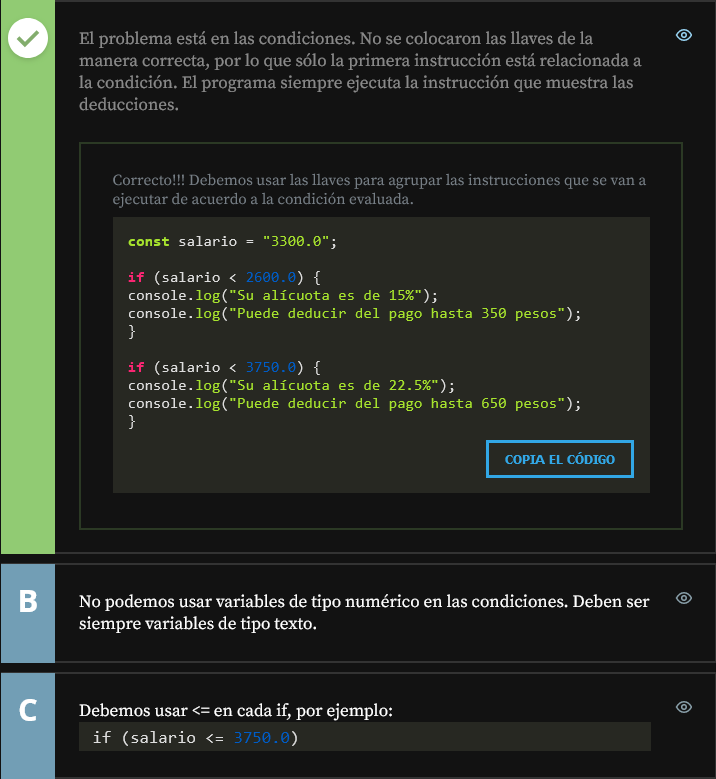
    console.log(`Comprador con pasaje. Feliz viaje`);

} else {

    console.log(`Comprador sin pasaje. No es posible hacer el embarque`);

}







Lazos: Ejecutar mas de una vez el mismo codigo

console.log(`Trabajando con condiciones`);

const ciudadesDisponibles = **new** *Array*();

ciudadesDisponibles.push("Bogotá");

ciudadesDisponibles.push("Caracas");

ciudadesDisponibles.push("Lima");

ciudadesDisponibles.push("Santiago");

const edadComprador = 15;

const estaAcompanado = true; //Variable booleana: true o false

const tienePasaje = true;

const ciudadDestino = "Quito";

let posicion = 0;

let destinoDisponible = false;

let puedeComprar = false;

console.log(ciudadesDisponibles);

puedeComprar = edadComprador >= 18 || estaAcompanado;

// while(posicion < ciudadesDisponibles.length) {

//     if (ciudadesDisponibles[posicion] == ciudadDestino) {

//         destinoDisponible = true;

//         break;

//     }

//     posicion++;

// }

//Lazo for

for (posicion = 0; posicion < ciudadesDisponibles.length; posicion++) {

    if (ciudadesDisponibles[posicion] == ciudadDestino) {

        destinoDisponible = true;

        break;

    }

}

if (puedeComprar && destinoDisponible) {

    console.log(

        `Es posible vender el pasaje. La persona puede comprar y el destino está disponible`

    );

} else if (puedeComprar) {

    console.log(

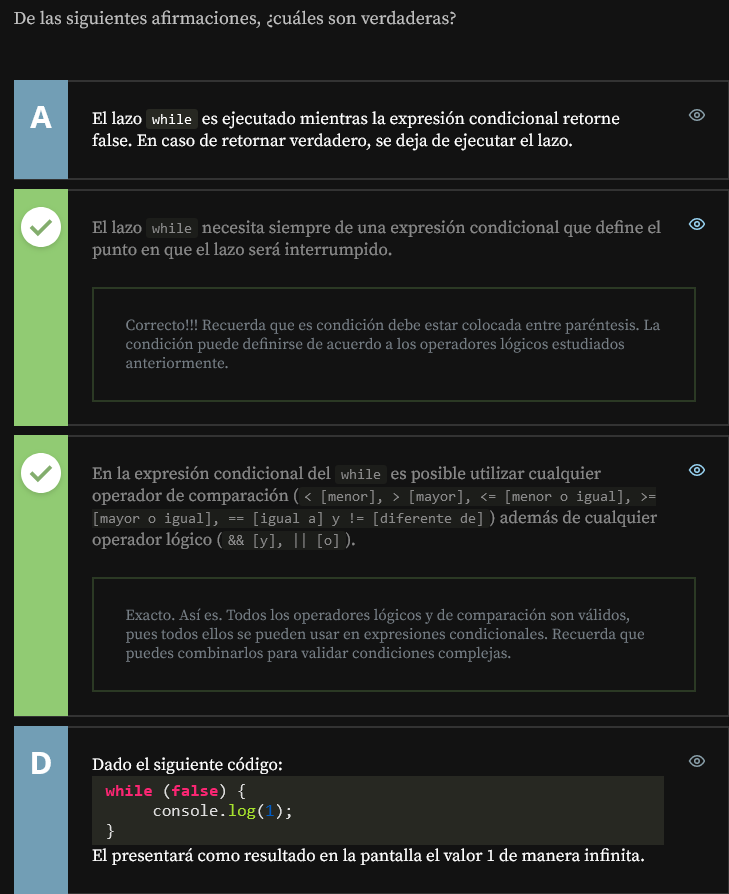
        `No es posible vender el pasaje. La persona puede comprar pero el destino no está disponible`

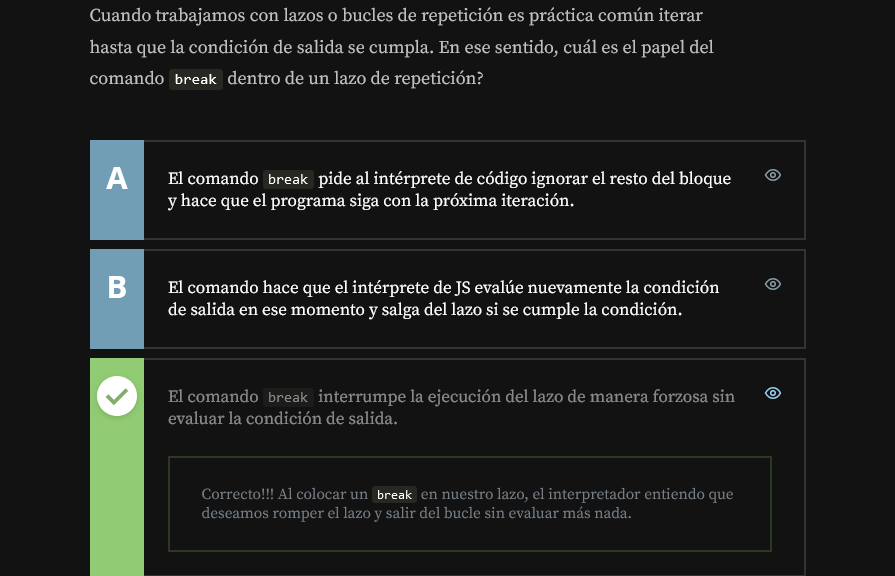
    );

} else {

    console.log(`No es posible vender el pasaje`);

}

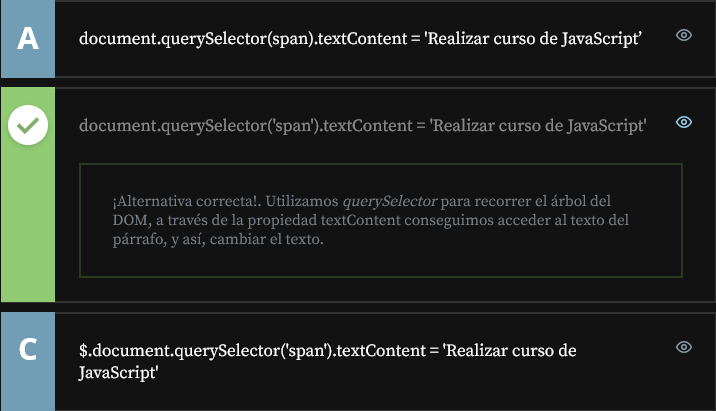


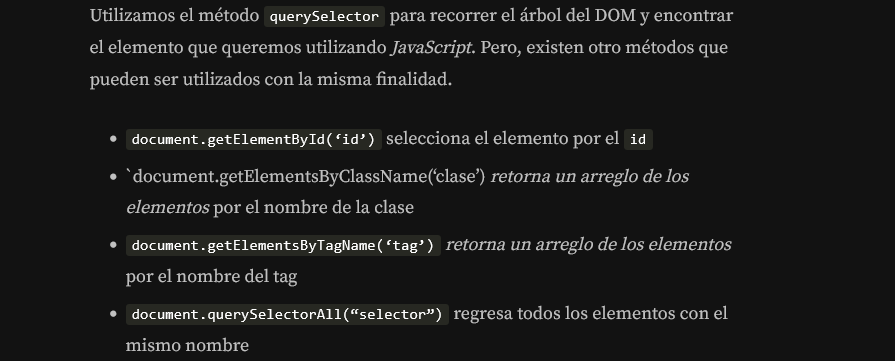


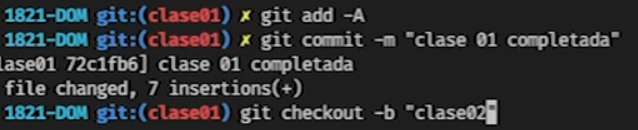
# ****JS en la Web: Manipulación del DOM con JavaScript****

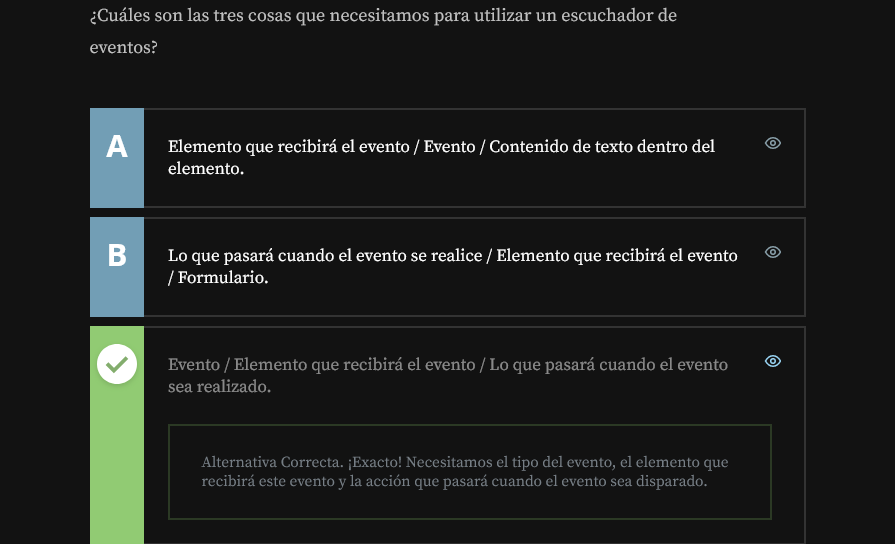
El HTML y el CSS ya estan hechos, lo que falta ahora es agregar a traves de js la funcionalidad de la pagina.











import checkComplete from "./components/checkComplete.js";

import deleteIcon from "./components/deleteIcon.js";

const btn = document.querySelector("[data-form-btn]"); // Selecciona el elemento con el atributo data-form-btn

const createTask = (*evento*) => {

*evento*.preventDefault(); // Evita que el formulario se envíe

    const input = document.querySelector("[data-form-input]"); // Selecciona el elemento con el atributo data-form-input

    const value = input.value; // Obtiene el valor del input

    const list = document.querySelector("[data-list]"); // Selecciona el elemento con el atributo data-list

    const task = document.createElement("li"); // Crea un elemento li

    task.classList.add("card"); // Agrega la clase card al elemento li

    input.value = "";

    //backticks

const taskContent = document.createElement("div"); // Crea un elemento div

    const titleTask = document.createElement("span"); // Crea un elemento span

    titleTask.classList.add("task"); // Agrega la clase task al elemento span

    titleTask.innerText = value; // Agrega el valor del input al elemento span

    taskContent.appendChild(checkComplete()); // Agrega el elemento i al elemento div

taskContent.appendChild(titleTask); // Agrega el elemento span al elemento div

// task.innerHTML = content;

    task.appendChild(taskContent); // Agrega el elemento div al elemento li

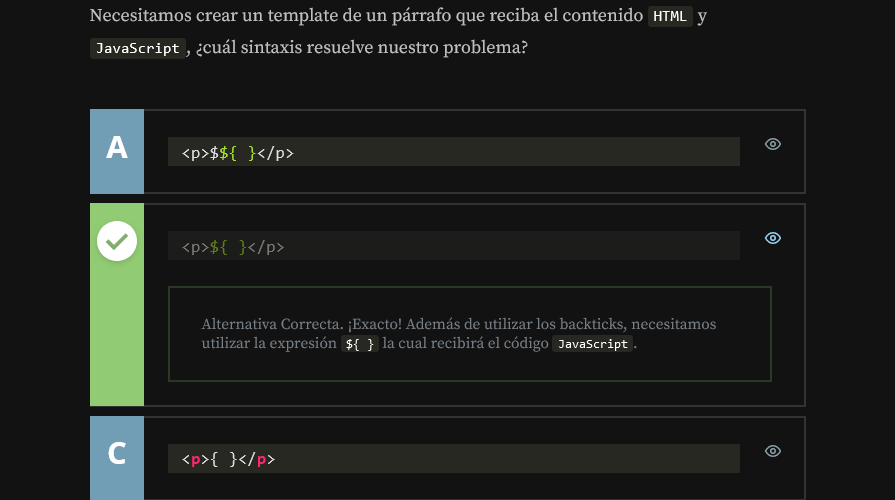
    task.appendChild(deleteIcon()); // Agrega el elemento i al elemento li

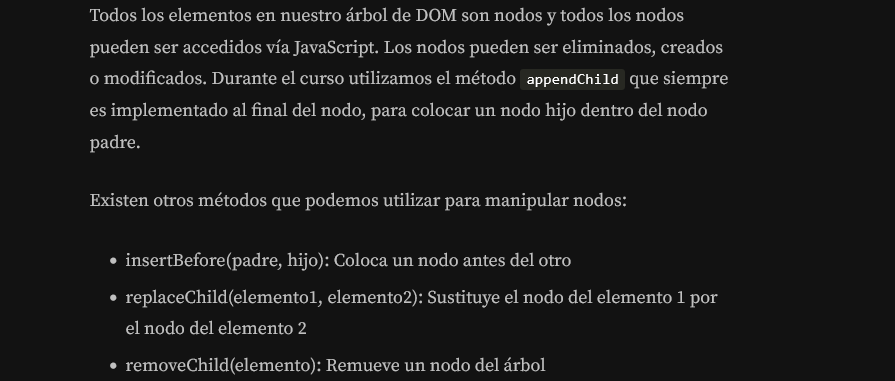
    list.appendChild(task); // Agrega el elemento li al elemento ul

};

//Arrow functions o funciones anonimas

btn.addEventListener("click", createTask); // Agrega un evento al elemento btn





const checkComplete = () => {

    const i = document.createElement("i"); // create an icon

    i.classList.add("far", "fa-check-square", "icon"); // Agrega clases a i

    i.addEventListener("click", completeTask); // Agrega un evento a i

    return i;

};

// Immediately invoked function expression IIFE

const completeTask = (*event*) => {

    const element = *event*.target; // element is the icon

    element.classList.toggle("fas"); // toggle the class fas

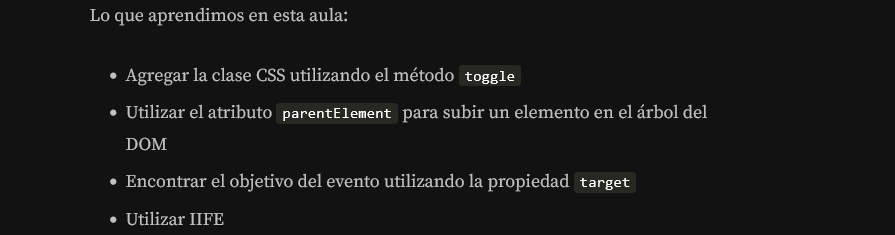
    element.classList.toggle("completeIcon"); // toggle the class completeIcon

    element.classList.toggle("far"); // toggle the class far

};

export default checkComplete; // export the function





const deleteIcon = () => {

    const i = document.createElement("i"); // create an icon

    i.classList.add("fas", "fa-trash-alt", "trashIcon", "icon"); // Agrega clases a i

    i.addEventListener("click", deleteTask); // Agrega un evento a i

    return i;

};

const deleteTask = (*event*) => {

    const parent = *event*.target.parentElement; // parent is the li

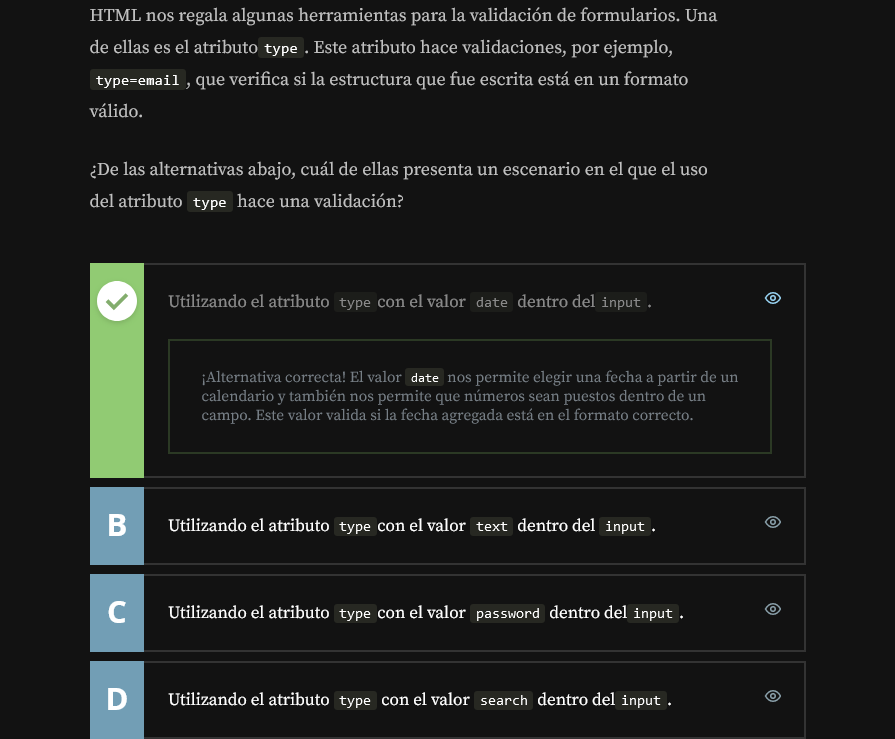
    parent.remove(); // remove the li

};

export default deleteIcon; // export the function



# ****JS en la Web: Validación de Formularios y HTML5****



Patron de referencia de password: [^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?!.\*[ !@#$%^&\*\_=+-]).{6,12}$]



Validaciones.js

const validadores = {

  nacimiento: (*input*) => validarNacimiento(*input*),

};

function validarNacimiento(*input*) {

  const fechaCliente = **new** *Date*(*input*.value);

  let mensaje = "";

  if (!mayorDeEdad(fechaCliente)) {

    mensaje = "Debes tener al menos 18 años de edad";

  }

*input*.setCustomValidity(mensaje);

}

function mayorDeEdad(*fecha*) {

  const fechaActual = **new** *Date*();

  const diferenciaFechas = **new** *Date*(

*fecha*.getUTCFullYear() + 18,

*fecha*.getUTCMonth(),

*fecha*.getUTCDate()

  );

  return diferenciaFechas <= fechaActual;

}

App.js

import { valida } from "./validaciones.js";

const inputs = document.querySelectorAll("input");

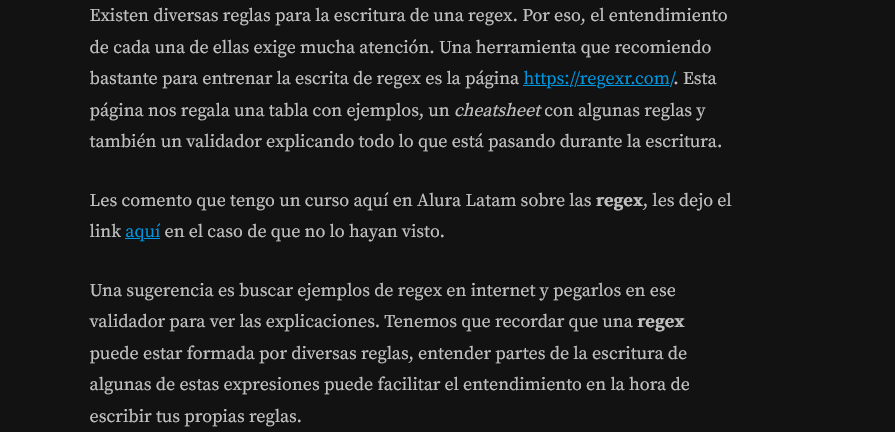
inputs.forEach((*input*) => {

*input*.addEventListener("blur", (*input*) => {

    valida(*input*.target);

  });

});



<https://regexr.com/>

Validaciones.js

export function valida(*input*) {

  const tipoDeInput = *input*.dataset.tipo;

  if (validadores[tipoDeInput]) {

    validadores[tipoDeInput](*input*);

  }

  if (*input*.validity.valid) {

*input*.parentElement.classList.remove("input-container--invalid");

*input*.parentElement.querySelector(".input-message-error").innerHTML = "";

  } else {

*input*.parentElement.classList.add("input-container--invalid");

*input*.parentElement.querySelector(".input-message-error").innerHTML =

      mostrarMensajeDeError(tipoDeInput, *input*);

  }

}

const tipoDeErrores = [

  "valueMissing",

  "typeMismatch",

  "patternMismatch",

  "customError",

];

const mensajesDeError = {

  nombre: {

    valueMissing: "El campo nombre no puede estar vacío",

  },

  email: {

    valueMissing: "El campo correo no puede estar vacío",

    typeMismatch: "El correo no es válido",

  },

  password: {

    valueMissing: "El campo contraseña no puede estar vacío",

    patternMismatch:

      "Al menos 6 caracteres, máximo 12, debe contener una letra minúscula, una letra mayúscula, un número y no puede contener caracteres especiales.",

  },

  nacimiento: {

    valueMissing: "Este campo no puede estar vacío",

    customError: "Debes tener al menos 18 años de edad",

  },

  numero: {

    valueMissing: "Este campo no puede estar vacío",

    patternMismatch: "El formato requerido es XXXXXXXXXX 10 números",

  },

  direccion: {

    valueMissing: "Este campo no puede estar vacío",

    patternMismatch: "La dirección debe contener entre 10 a 40 caracteres.",

  },

  ciudad: {

    valueMissing: "Este campo no puede estar vacío",

    patternMismatch: "La ciudad debe contener entre 10 a 40 caracteres.",

  },

  estado: {

    valueMissing: "Este campo no puede estar vacío",

    patternMismatch: "El estado debe contener entre 10 a 40 caracteres.",

  },

};

function mostrarMensajeDeError(*tipoDeInput*, *input*) {

  let mensaje = "";

  tipoDeErrores.forEach((*error*) => {

    if (*input*.validity[*error*]) {

      console.log(*tipoDeInput*, *error*);

      console.log(*input*.validity[*error*]);

      console.log(mensajesDeError[*tipoDeInput*][*error*]);

      mensaje = mensajesDeError[*tipoDeInput*][*error*];

    }

  });

  return mensaje;

}



<https://vercel.com/>

<https://pages.github.com/>

# ****JavaScript: Introducción a la Orientación a Objetos****

Index.js

import {Cliente} from './Cliente.js'

import {CuentaCorriente} from './CuentaCorriente.js';

const cliente = **new** Cliente('Leonardo','13804050','123224');

const cliente2 = **new** Cliente('María','16979808','8989');

const cuentaDeLeonardo = **new** CuentaCorriente(cliente, '1', '001');

const cuentaDeMaria = **new** CuentaCorriente(cliente2,'2','002');

let saldo = cuentaDeLeonardo.verSaldo();

saldo = cuentaDeLeonardo.depositoEnCuenta(150);

cuentaDeLeonardo.transferirParaCuenta(parametroValor,cuentaDeMaria);

const saldoMaria = cuentaDeMaria.verSaldo();

console.log('Cuenta de Maria',cuentaDeMaria);

console.log('El Saldo actual (cuentaMaria) '+saldoMaria);

const saldoLeonardo = cuentaDeLeonardo.verSaldo();

console.log('El Saldo actual (cuentaLeonardo) '+saldoLeonardo);

console.log(CuentaCorriente.cantidadCuentas);

console.log(CuentaCorriente.cantidadCuentas);

Cliente.js

export class Cliente

{

    nombreCliente;

    dniCliente;

    rutCliente;

    constructor(*nombreCliente*, *dniCliente*, *rutCliente*) {

*this*.nombreCliente = *nombreCliente*;

*this*.dniCliente = *dniCliente*;

*this*.rutCliente = *rutCliente*;

    }

}

CuentaCorriente.js

import { Cliente } from "./Cliente.js";

export class CuentaCorriente

{

    #cliente;

    numero;

    agencia;

    #saldo;

static cantidadCuentas = 0;

    set cliente(*valor*) {

        if (*valor* instanceof Cliente)

*this*.#cliente = *valor*;

}

    get cliente() {

        return *this*.#cliente;

}

    constructor(*cliente*, *numero*, *agencia*) {

*this*.cliente = *cliente*;

*this*.numero = *numero*;

*this*.agencia = *agencia*;

*this*.#saldo = 0;

        CuentaCorriente.cantidadCuentas++;

}

    depositoEnCuenta(*valor*) {

        if (*valor* > 0)

*this*.#saldo += *valor*;

        return *this*.#saldo;

}

    retirarDeCuenta(*valor*) {

        if (*valor* <= *this*.#saldo)

*this*.#saldo -= *valor*;

        return *this*.#saldo;

}

    verSaldo() {

        return *this*.#saldo;

}

    transferirParaCuenta(*valor*,*cuentaDestino*) {

*this*.retirarDeCuenta(*valor*);

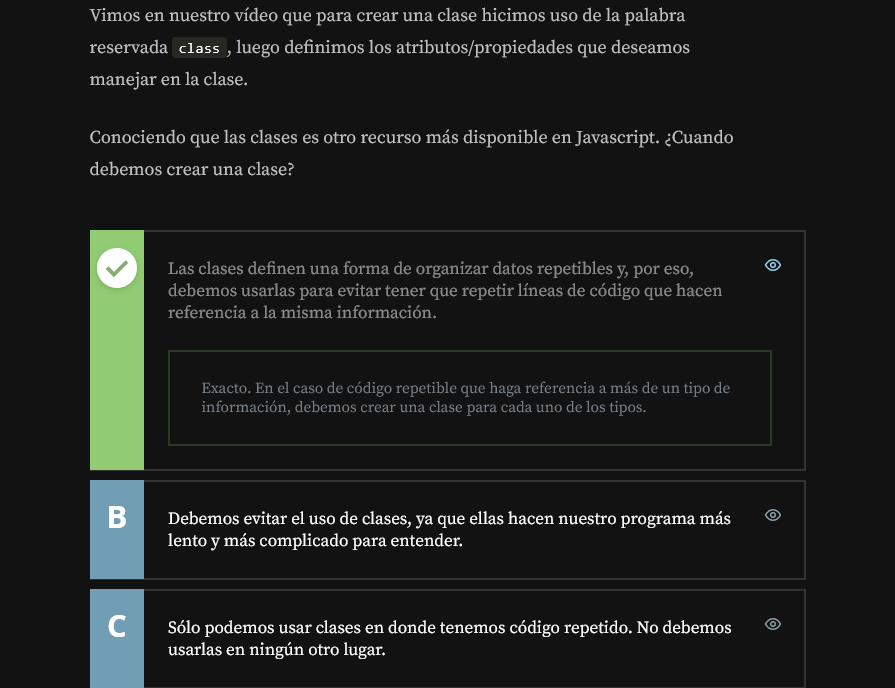
*cuentaDestino*.depositoEnCuenta(*valor*);

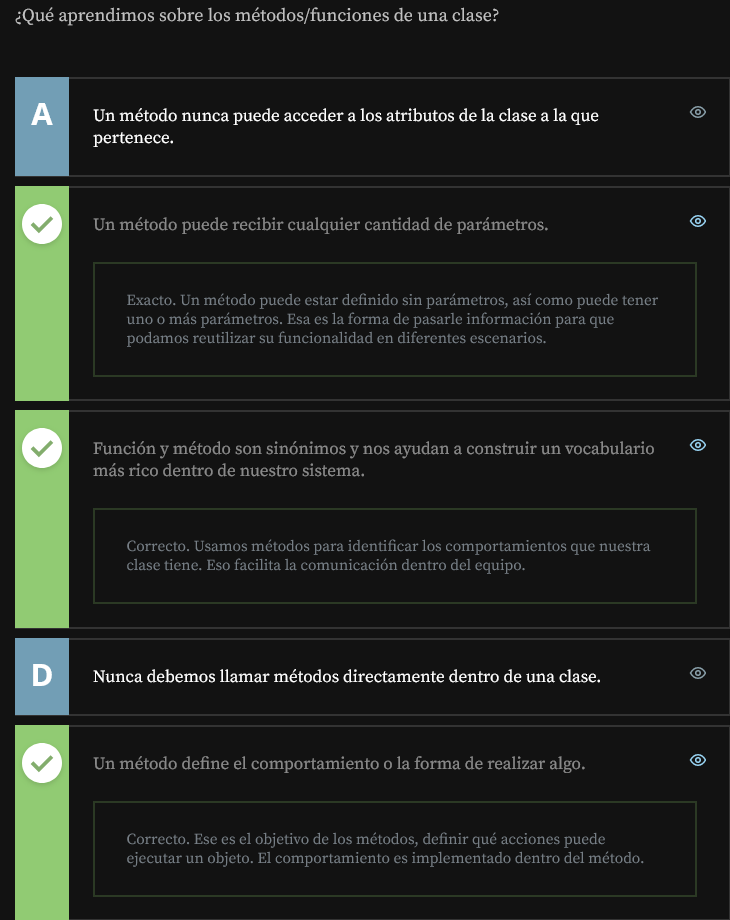
*valor* = 200;

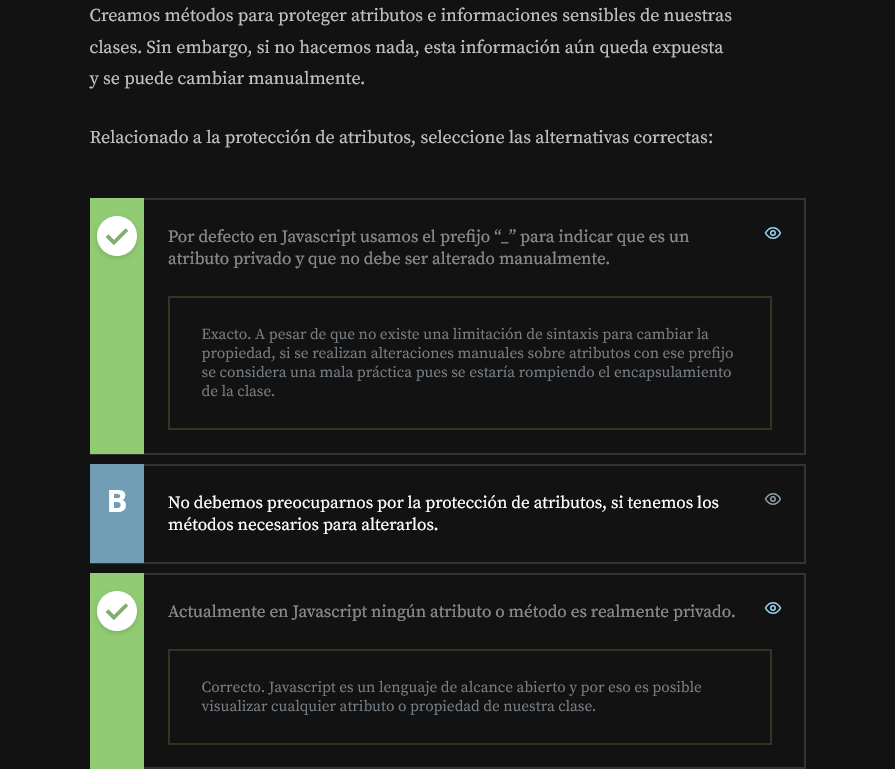
*valor* = *valor*\*1000;

    }

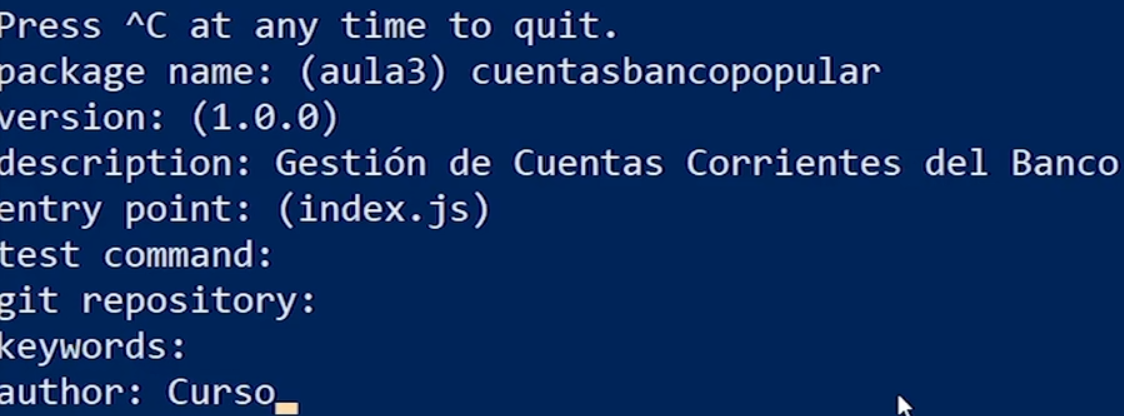
}











{

  "name": "cuentasbancopopular",

  "version": "1.0.0",

  "description": "Gestión de Cuentas Corrientes del Banco Popular",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

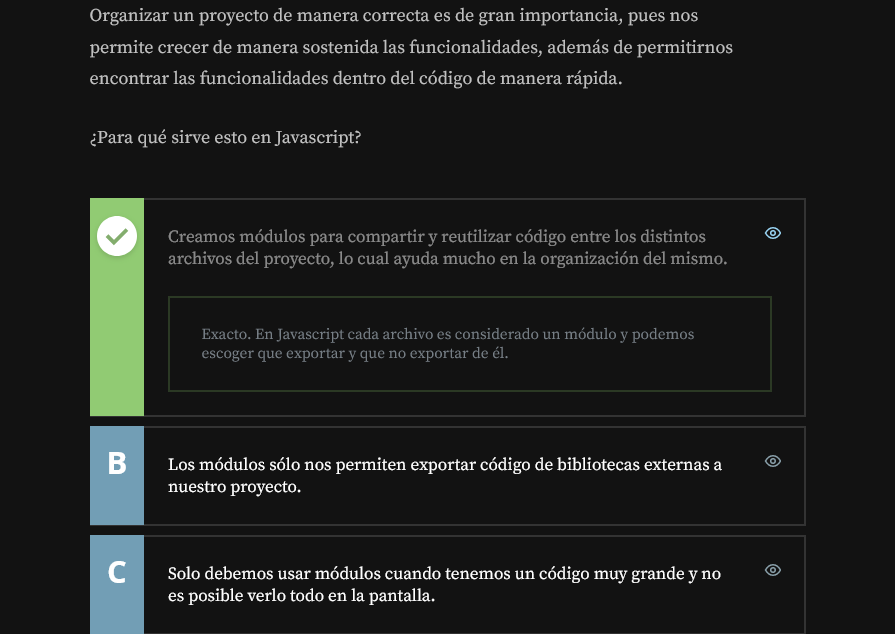
  },

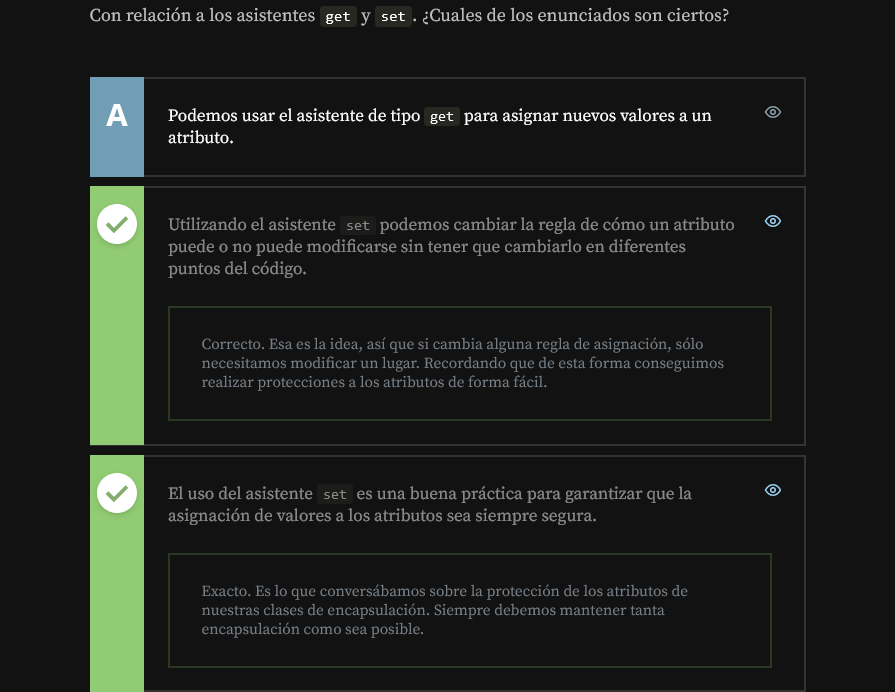
  "author": "Curso Alura",

  "license": "ISC",

  "type":"module"

}





Constructor en js:

constructor(*cliente*, *numero*, *agencia*) {

*this*.cliente = *cliente*;

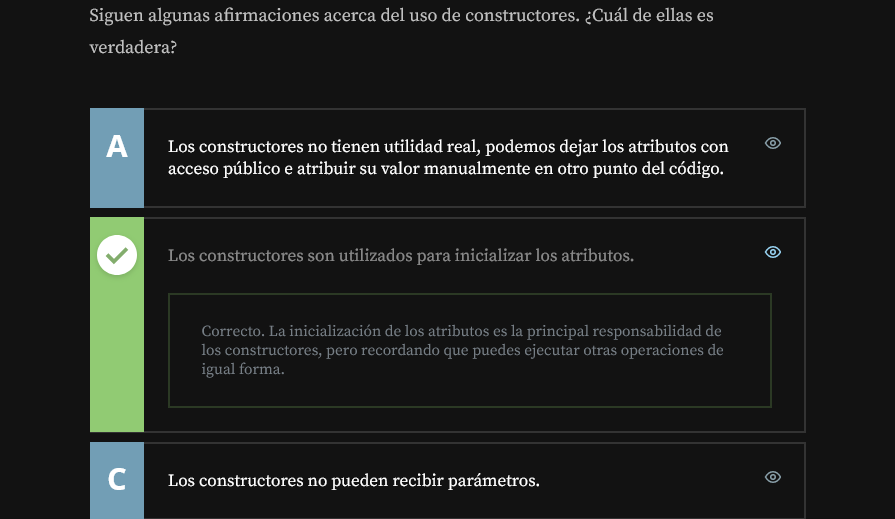
*this*.numero = *numero*;

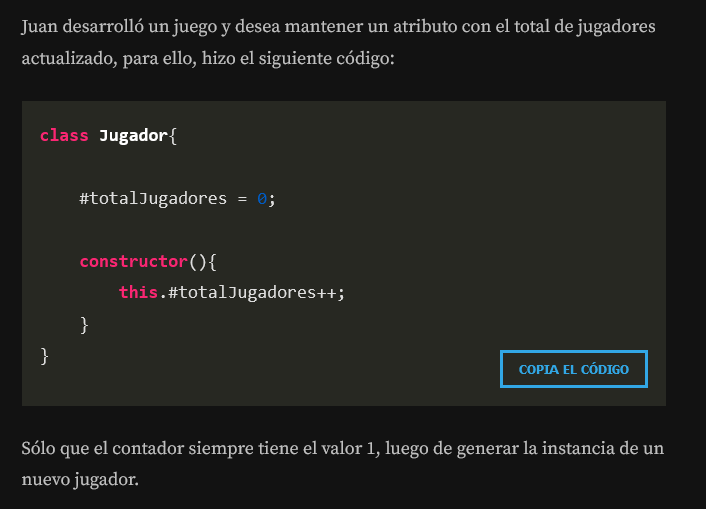
*this*.agencia = *agencia*;

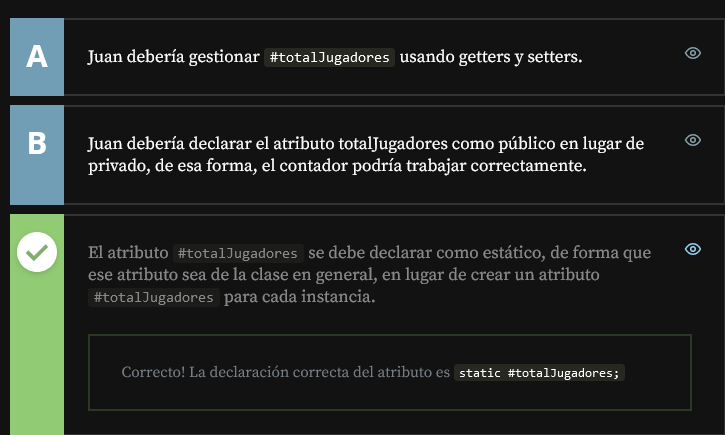
*this*.#saldo = 0;

        CuentaCorriente.cantidadCuentas++;

    }







# ****JavaScript: Herencia e Interfaces en Orientación a Objetos****

CuentaAhorro.js

import { Cuenta } from "./Cuenta.js";

export class CuentaAhorro extends Cuenta {

    constructor(*cliente*, *numero*, *agencia*, *saldo*) {

*super*(*cliente*,*numero*,*agencia*, *saldo*);

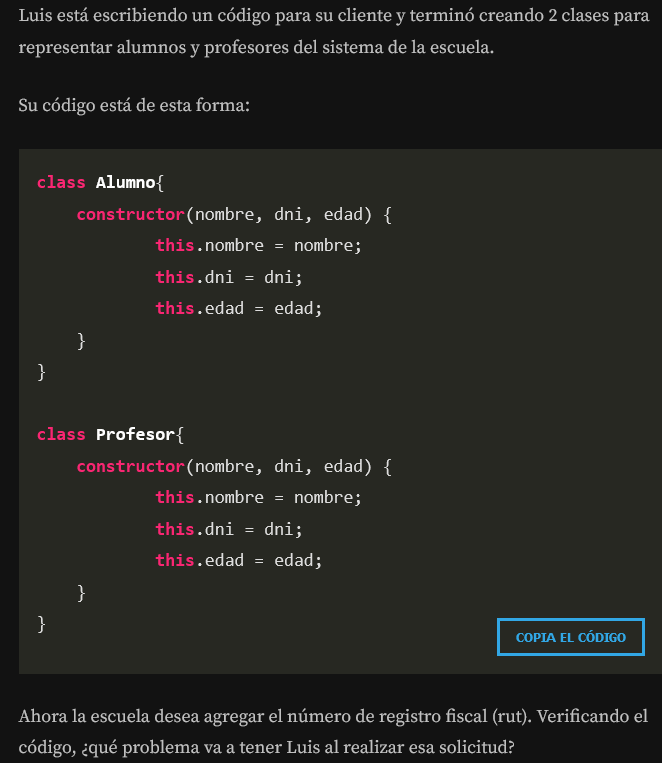
}

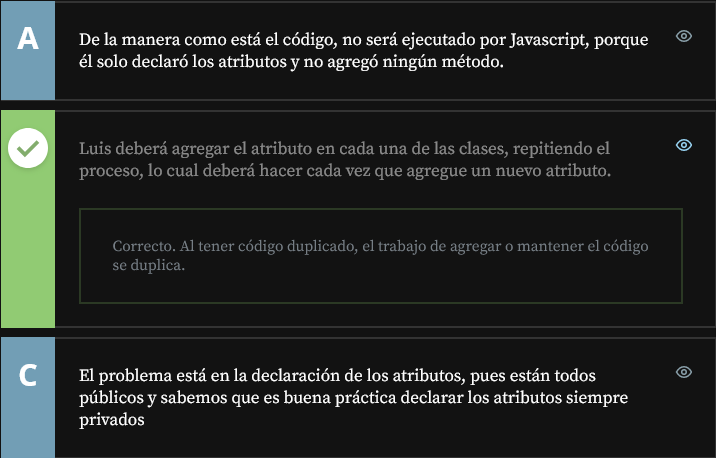
    retirarDeCuenta(*valor*) {

*super*.\_retirarDeCuenta(*valor*,2);

    }

}





CuentaCorriente.js

import { Cuenta } from "./Cuenta.js";

export class CuentaCorriente extends Cuenta {

static cantidadCuentas = 0;

    constructor(*cliente*, *numero*, *agencia*) {

*super*(*cliente*,*numero*,*agencia*,0);

        CuentaCorriente.cantidadCuentas++;

}

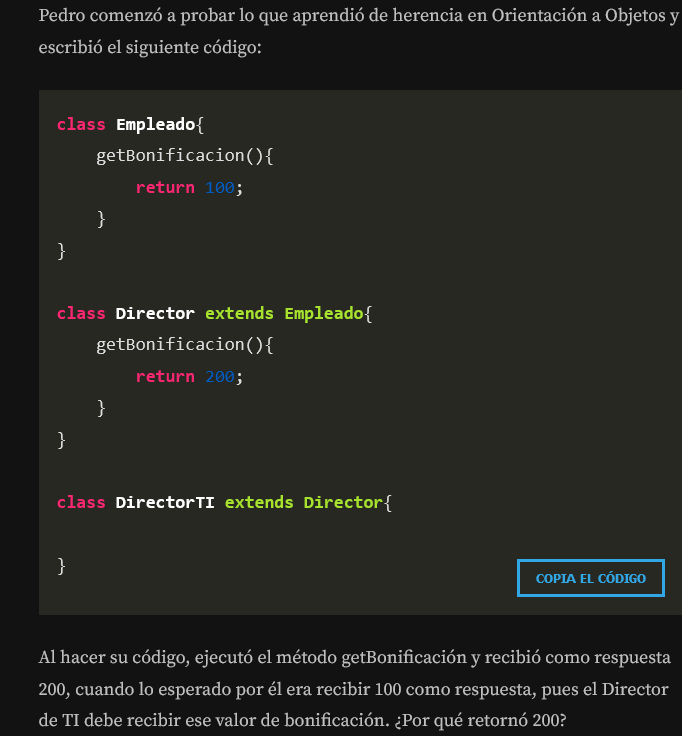
    retirarDeCuenta(*valor*) {

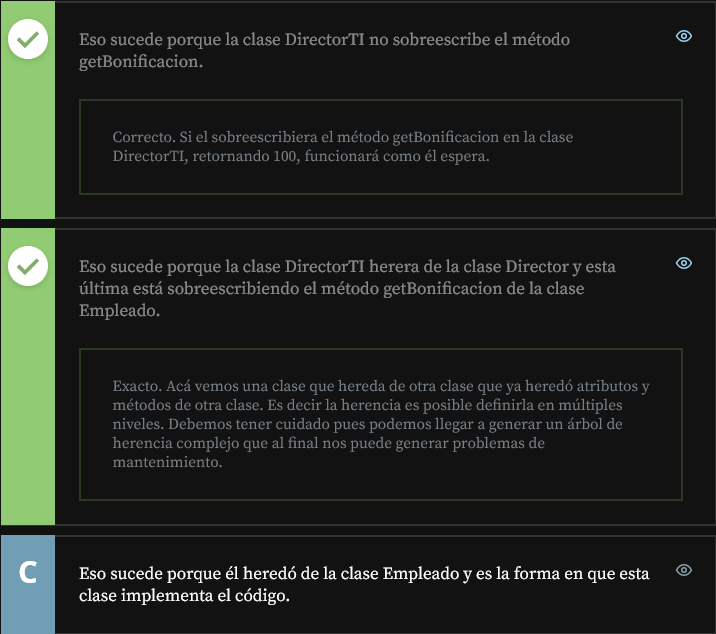
*super*.\_retirarDeCuenta(*valor*,5);

    }

}







CuentaNomina.js

import { Cuenta } from "./Cuenta.js";

export class CuentaNomina extends Cuenta {

    constructor(*cliente*, *numero*, *agencia*, *saldo*) {

*super*(*cliente*,*numero*,*agencia*, *saldo*);

    }

    retirarDeCuenta(*valor*) {

*super*.\_retirarDeCuenta(*valor*,1);

    }

}

Cuenta.js

export class Cuenta {

    #cliente;

    #saldo;

constructor(*cliente*, *numero*, *agencia*, *saldo*) {

        if (*this*.constructor == Cuenta) {

            throw **new** *Error*('No se debe instanciar objetos de la clase Cuenta');

        }

*this*.numero = *numero*;

*this*.agencia = *agencia*;

*this*.#cliente = *cliente*;

*this*.#saldo = *saldo*;

}

    set cliente(*valor*) {

        if (*valor* instanceof *Cliente*)

*this*.#cliente = *valor*;

}

    get cliente() {

        return *this*.#cliente;

}

    depositoEnCuenta(*valor*) {

        if (*valor* > 0)

*this*.#saldo += *valor*;

        return *this*.#saldo;

}

    retirarDeCuenta(*valor*) {

        //Método abstracto

        throw **new** *Error*('Debe implementar el método retirarDeCuenta en su clase');

}

    \_retirarDeCuenta(*valor*, *comision*) {

*valor* = *valor* \* (1+*comision*/100);

        if (*valor* <= *this*.#saldo)

*this*.#saldo -= *valor*;

        return *this*.#saldo;

}

    verSaldo() {

        return *this*.#saldo;

}

    transferirParaCuenta(*valor*,*cuentaDestino*) {

*this*.retirarDeCuenta(*valor*);

*cuentaDestino*.depositoEnCuenta(*valor*);

*valor* = 200;

*valor* = *valor*\*1000;

}

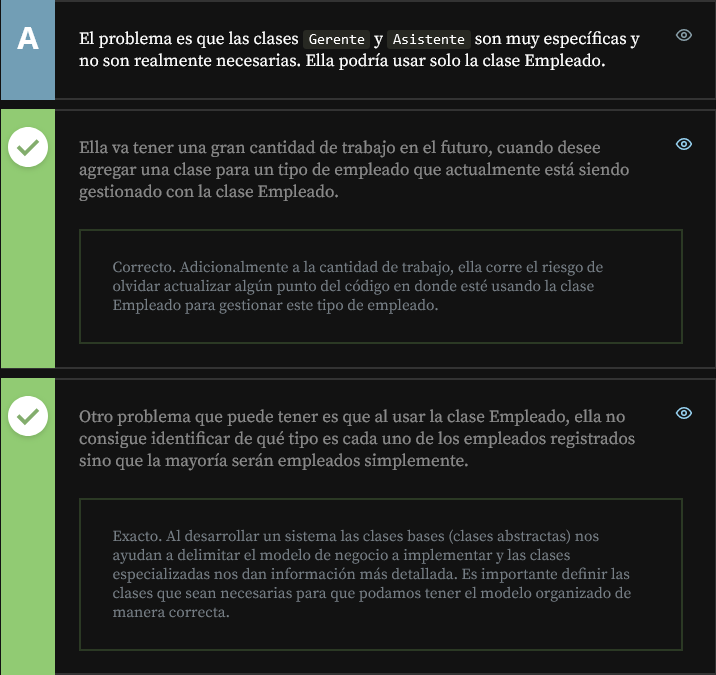
    prueba() {

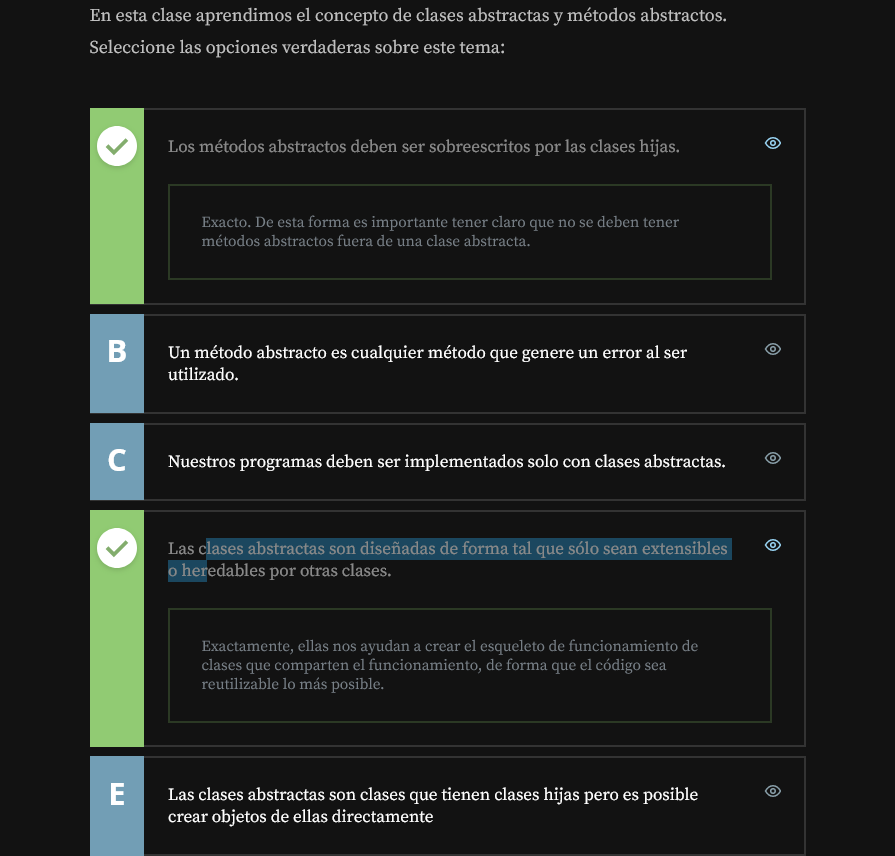
        console.log('Método padre');

    }

}







SistemaAutoenticacion.js

export class SistemaAutenticacion {

    static login(*usuario*,*clave*) {

        if ("autenticable" in *usuario*

            && *usuario*.autenticable instanceof *Function*)

            return *usuario*.autenticable(*clave*);

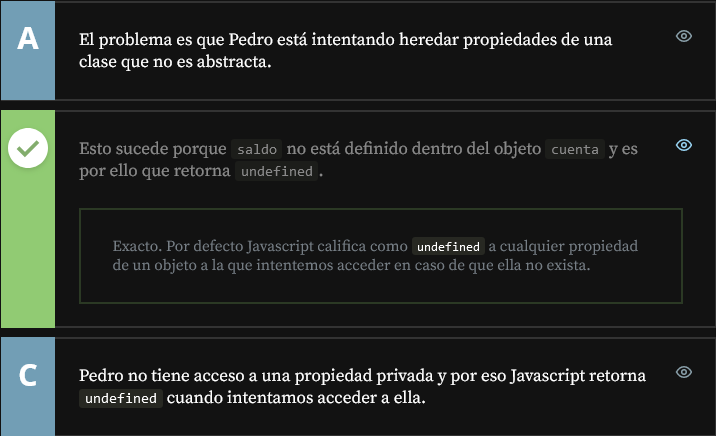
        else

            return false;

    }

}





Director.js

import {Empleado} from './Empleado.js';

export class Director extends Empleado {

    constructor(*nombre*,*dni*,*salario*) {

*super*(*nombre*,*dni*,*salario*);

}

    verBonificion() {

        const bono = 10;

        return *super*.\_verBonificion(bono);

    }

}

Empleado.js

export class Empleado {

    #nombre;

    #dni;

    #salario;

    #clave;

    constructor(*nombre*, *dni*, *salario*) {

*this*.#nombre = *nombre*;

*this*.#dni = *dni*;

*this*.#salario = *salario*;

*this*.#clave = '';

}

    asignarClave(*clave*) {

*this*.#clave = *clave*;

}

    autenticable(*clave*) {

        return *clave* == *this*.#clave;

}

    verBonificion() {

        return *this*.#salario;

}

    \_verBonificion(*bono*) {

        return *this*.#salario + *this*.#salario\**bono*/100;

    }

}

Gerente.js

import {Empleado} from './Empleado.js';

export class Gerente extends Empleado {

    constructor(*nombre*,*dni*,*salario*) {

*super*(*nombre*,*dni*,*salario*);

}

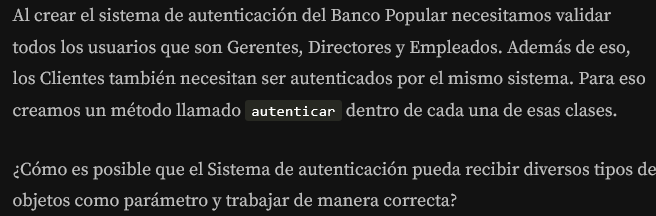
    verBonificion() {

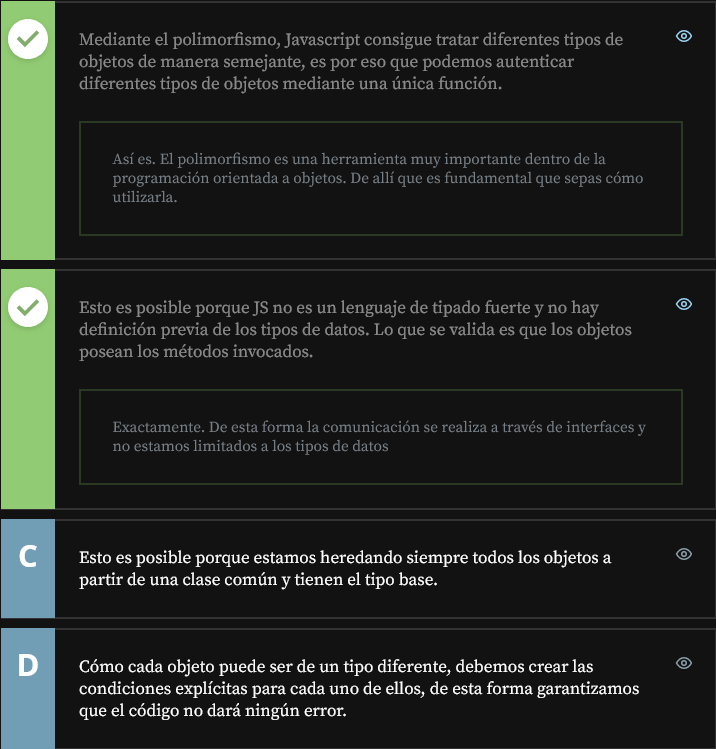
        const bono = 5;

        return *super*.\_verBonificion(bono);

    }

}





# ****JS en la Web: Almacenando datos en el navegador****

Para que muestre fecha:

<input *type*="datetime-local" *data-form-date* *placeholder*="YYYY/MM/DD"/>

Script.js

import { addTask } from './components/addTask.js';

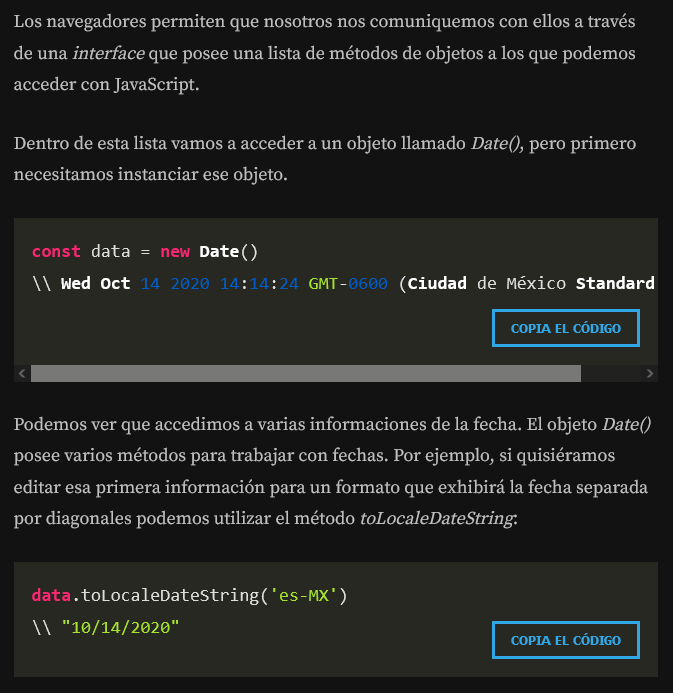
import { displayTasks } from './components/readTasks.js';

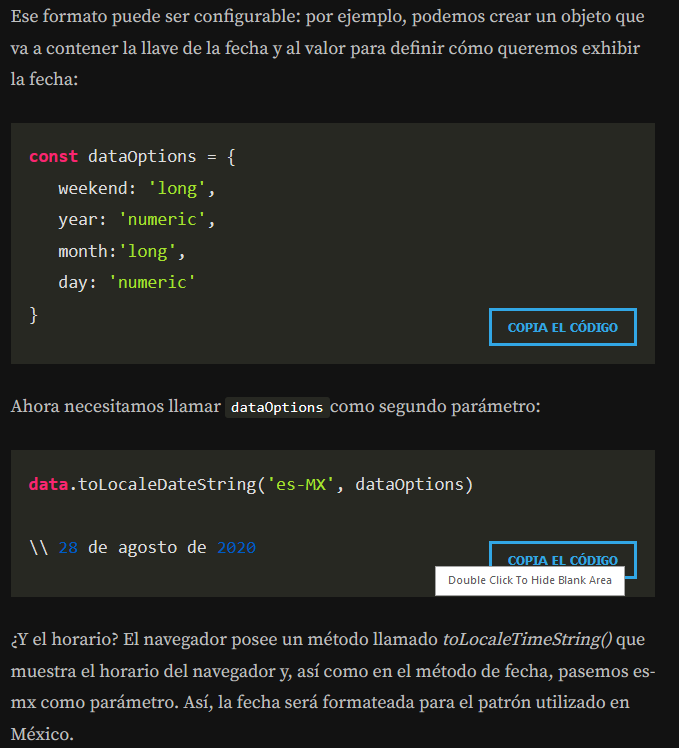
const btn = document.querySelector('[data-form-btn]');

//Arrow functions o funciones anonimas

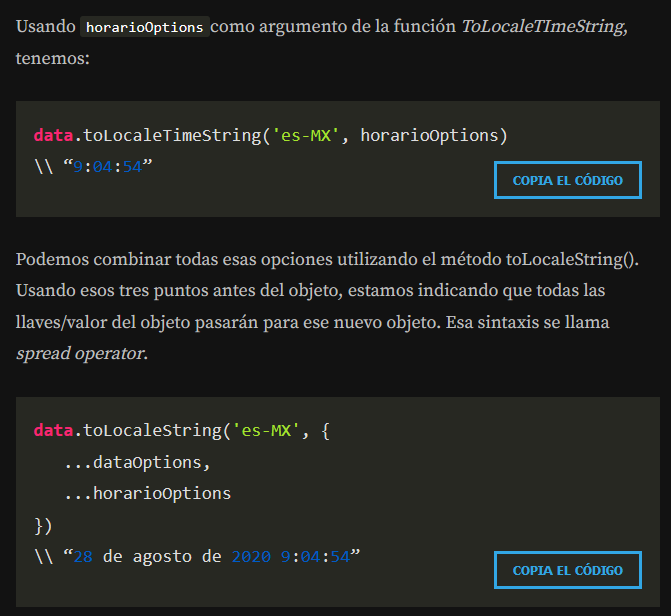
btn.addEventListener('click', addTask);

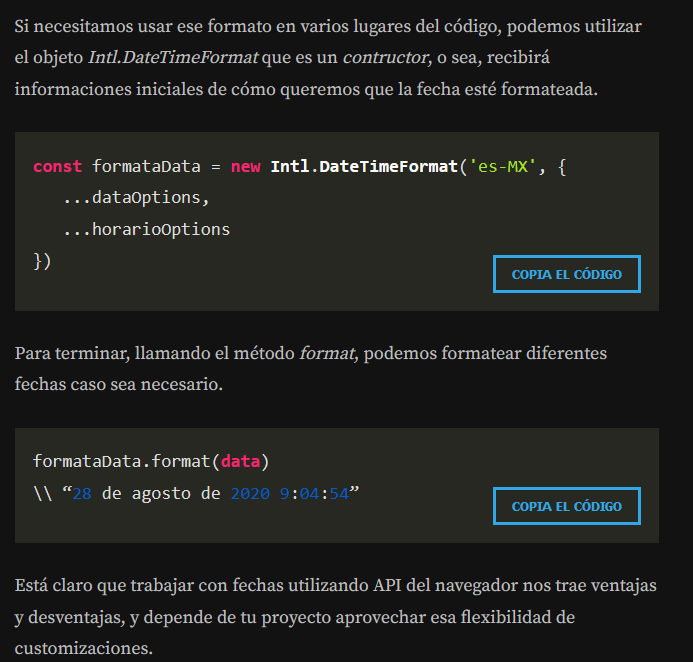
displayTasks();















DeleteIcon.js

import { displayTasks } from './readTasks.js';

const deleteIcon = (*id*) => {

  const i = document.createElement('i');

  i.classList.add('fas', 'fa-trash-alt', 'trashIcon', 'icon');

  i.addEventListener('click', () => deleteTask(*id*));

  return i;

};

const deleteTask = (*id*) => {

  const li = document.querySelector('[data-list]');

  const tasks = JSON.parse(localStorage.getItem('tasks'));

  const index = tasks.findIndex((*item*) => *item*.id === *id*);

  console.log(index);

  tasks.splice(index, 1);

  li.innerHTML = '';

  localStorage.setItem('tasks', JSON.stringify(tasks));

  displayTasks();

};

export default deleteIcon;

CheckComplete.js

const checkComplete = (*id*) => {

  const i = document.createElement('i');

  i.classList.add('far', 'fa-check-square', 'icon');

  i.addEventListener('click', (*event*) => completeTask(*event*, *id*));

  return i;

};

// Immediately invoked function expression IIFE

const completeTask = (*event*, *id*) => {

  const element = *event*.target;

  element.classList.toggle('fas');

  element.classList.toggle('completeIcon');

  element.classList.toggle('far');

  const tasks = JSON.parse(localStorage.getItem('tasks'));

  const index = tasks.findIndex((*item*) => *item*.id === *id*);

  tasks[index]['complete'] = !tasks[index]['complete'];

  localStorage.setItem('tasks', JSON.stringify(tasks));

};

export default checkComplete;

AddTask.js

import checkComplete from './checkComplete.js';

import deleteIcon from './deleteIcon.js';

import { displayTasks } from './readTasks.js';

export const addTask = (*evento*) => {

*evento*.preventDefault();

  const list = document.querySelector('[data-list]');

  const input = document.querySelector('[data-form-input]');

  const calendar = document.querySelector('[data-form-date]');

  const value = input.value;

  const date = calendar.value;

  const dateFormat = moment(date).format('DD/MM/YYYY');

  if (value === '' || date === '') {

    return;

  }

  input.value = '';

  calendar.value = '';

  const complete = false;

  const taskObj = {

    value,

    dateFormat,

    complete,

    id: uuid.v4(),

  };

  list.innerHTML = '';

  const taskList = JSON.parse(localStorage.getItem('tasks')) || [];

  taskList.push(taskObj);

  localStorage.setItem('tasks', JSON.stringify(taskList));

  displayTasks();

};

export const createTask = ({ *value*, *dateFormat*, *complete*, *id* }) => {

  const task = document.createElement('li');

  task.classList.add('card');

  const taskContent = document.createElement('div');

  const check = checkComplete(*id*);

  if (*complete*) {

    check.classList.toggle('fas');

    check.classList.toggle('completeIcon');

    check.classList.toggle('far');

  }

  const titleTask = document.createElement('span');

  titleTask.classList.add('task');

  titleTask.innerText = *value*;

  taskContent.appendChild(check);

  taskContent.appendChild(titleTask);

  task.appendChild(taskContent);

  task.appendChild(deleteIcon(*id*));

  return task;

};

readTasks.js

import { createTask } from './addTask.js';

import { uniqueDates, orderDates } from '../services/date.js';

import dateElement from './dateElement.js';

export const displayTasks = () => {

  const list = document.querySelector('[data-list]');

  const tasksList = JSON.parse(localStorage.getItem('tasks')) || [];

  const dates = uniqueDates(tasksList);

  orderDates(dates);

  dates.forEach((*date*) => {

    const dateMoment = moment(*date*, 'DD/MM/YYYY');

    list.appendChild(dateElement(*date*));

    tasksList.forEach((*task*) => {

      const taskDate = moment(*task*.dateFormat, 'DD/MM/YYYY');

      const diff = dateMoment.diff(taskDate);

      if (diff === 0) {

        list.appendChild(createTask(*task*));

      }

    });

  });

};

dataElement.js

export default (*date*) => {

  const dateElement = document.createElement('li');

  dateElement.classList.add('date');

  dateElement.innerHTML = *date*;

  return dateElement;

};

Date.js

export const uniqueDates = (*tasks*) => {

  const unique = [];

*tasks*.forEach((*task*) => {

    if (!unique.includes(*task*.dateFormat)) unique.push(*task*.dateFormat);

  });

  return unique;

};

export const orderDates = (*dates*) => {

  return *dates*.sort((*a*, *b*) => {

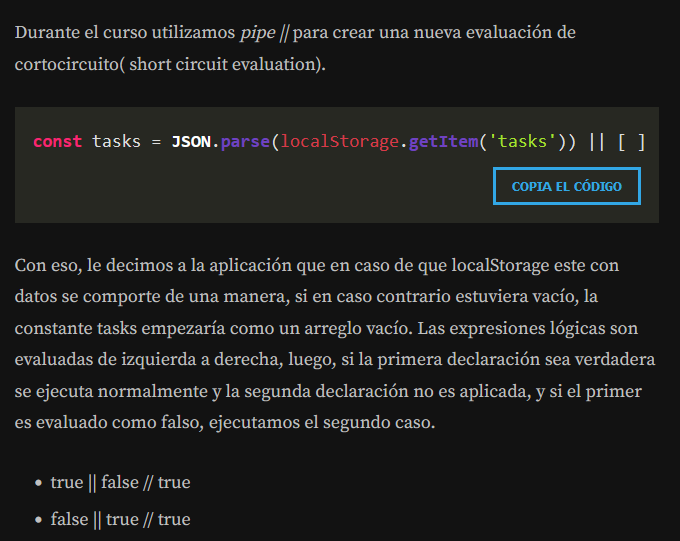
    const firstDate = moment(*a*, 'DD/MM/YYYY');

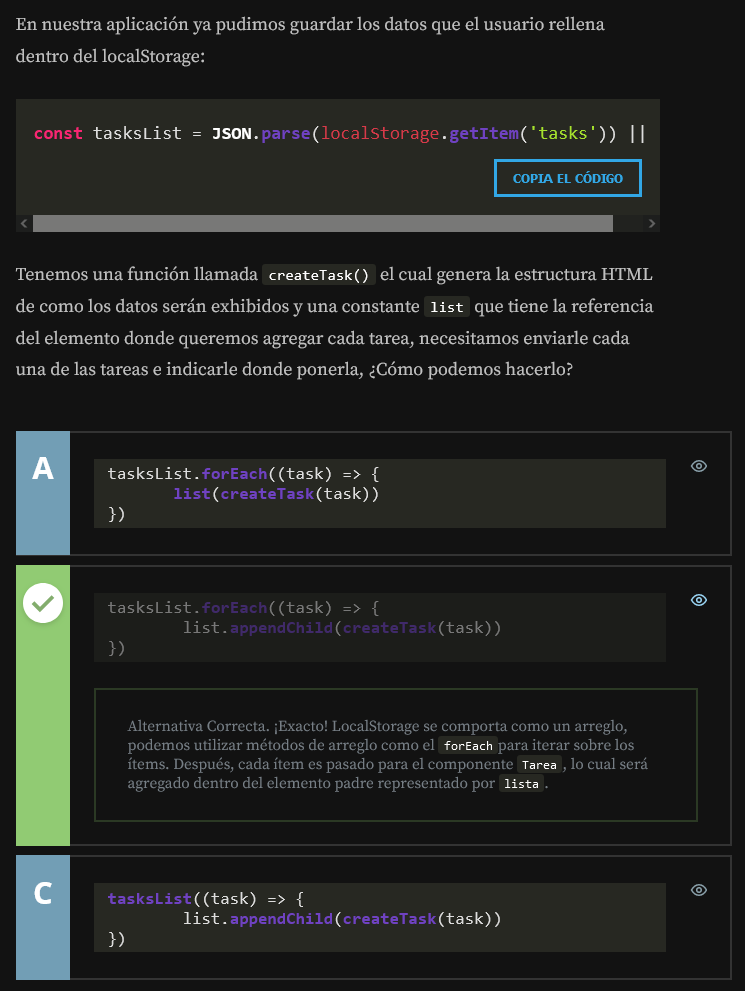
    const secondDate = moment(*b*, 'DD/MM/YYYY');

    return firstDate - secondDate;

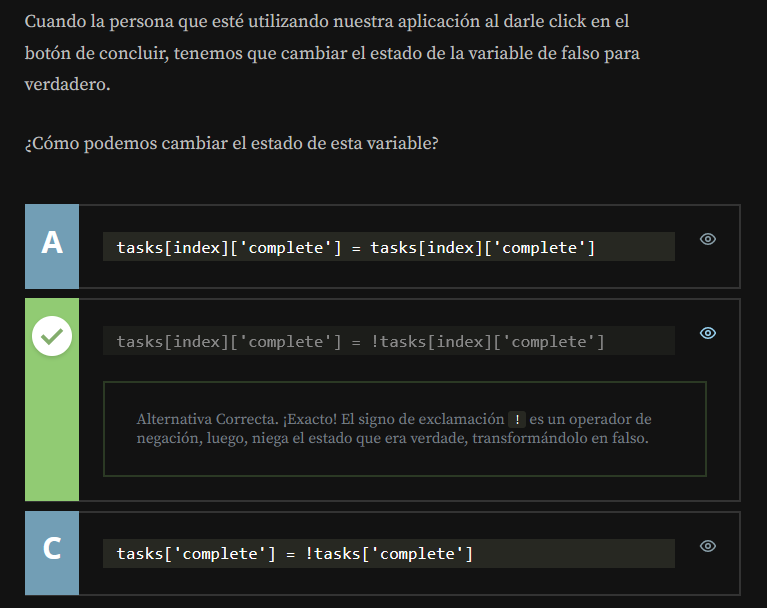
  });

};

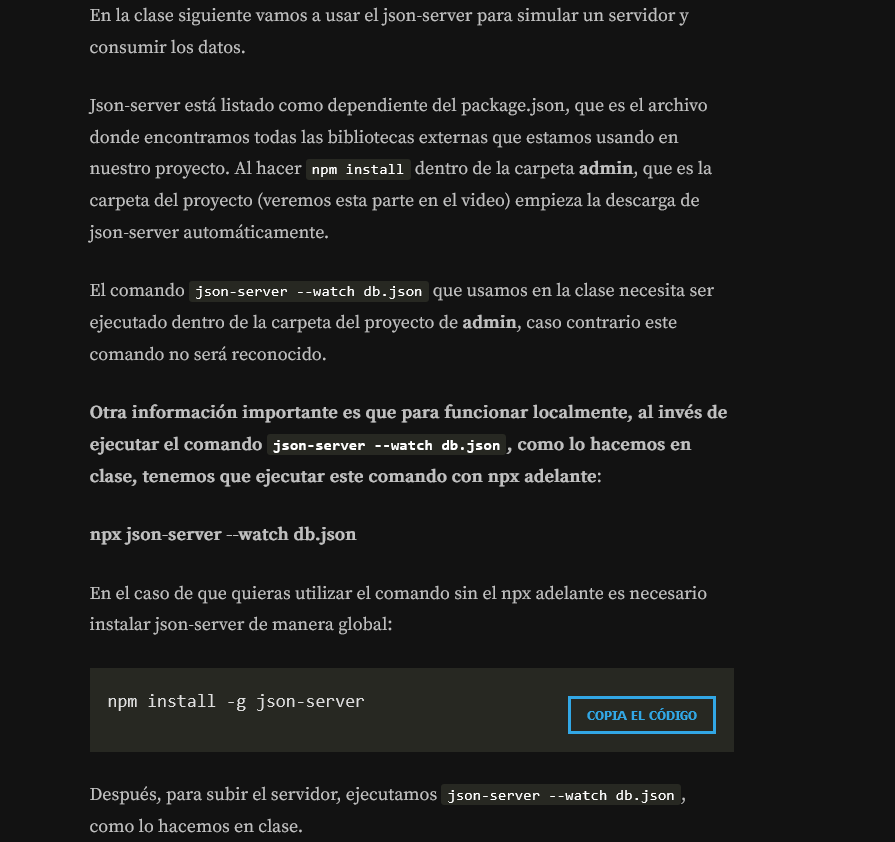








# ****JS en la Web: CRUD con JavaScript asíncrono****



Db.json

{

  "perfil": [

    {

      "nombre": "Pancho",

      "email": "pancho@alura.com",

      "id": 1

    },

    {

      "nombre": "Pedro",

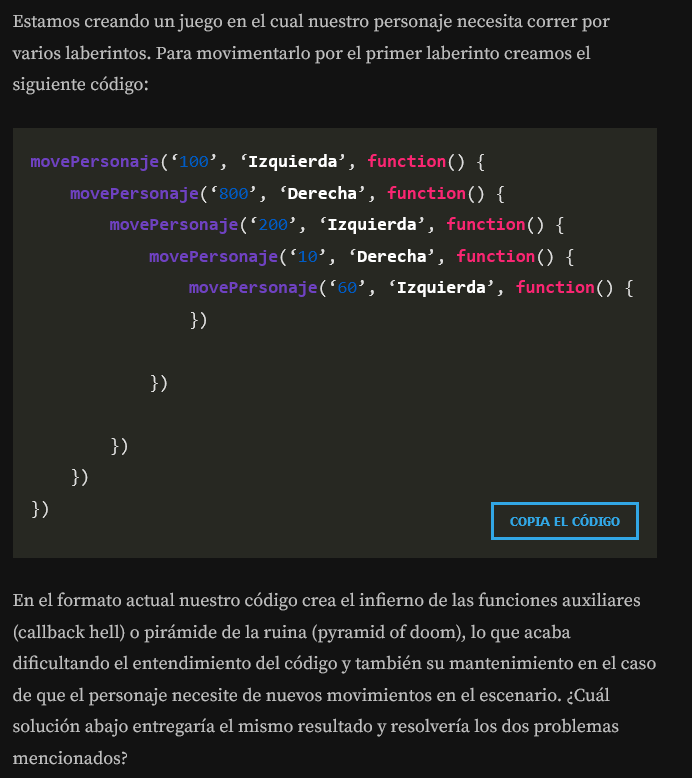
      "email": "pedro@alura.com",

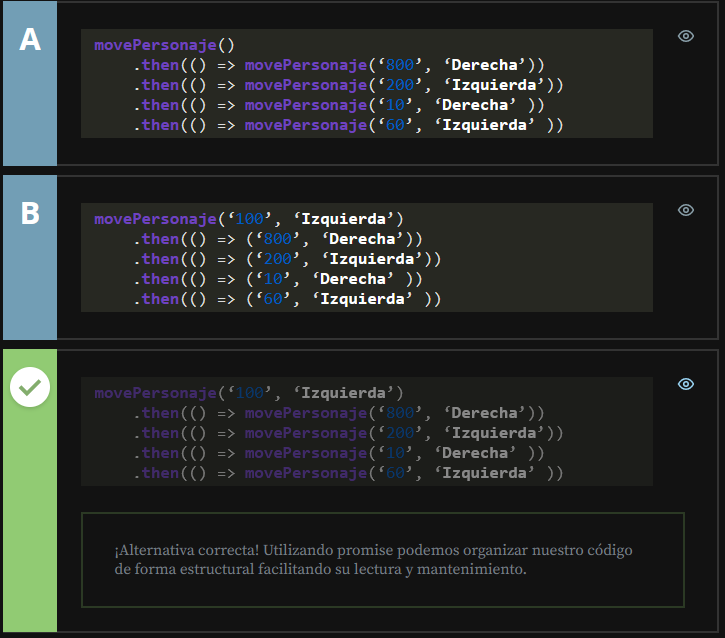
      "id": "b007442c-8aad-456b-b6a1-4fee1562eb84"

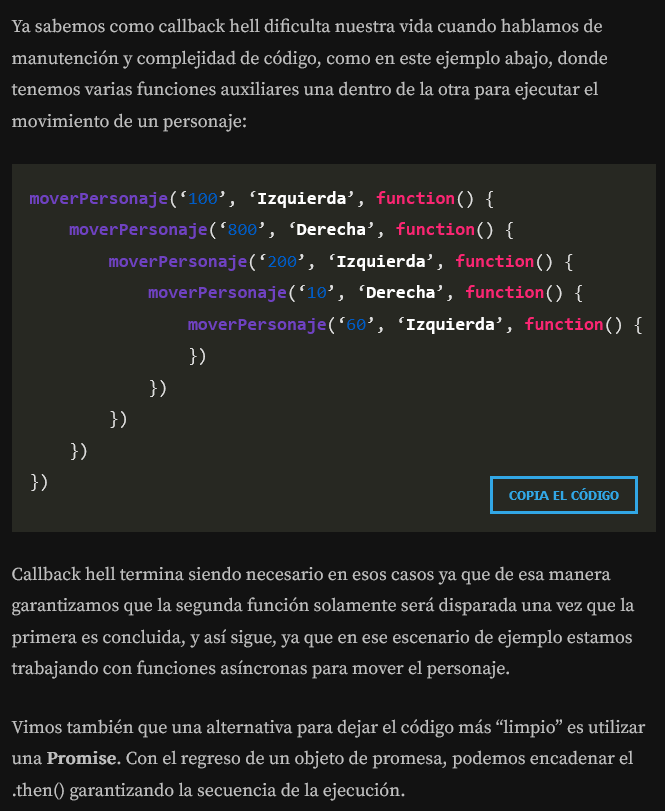
    }

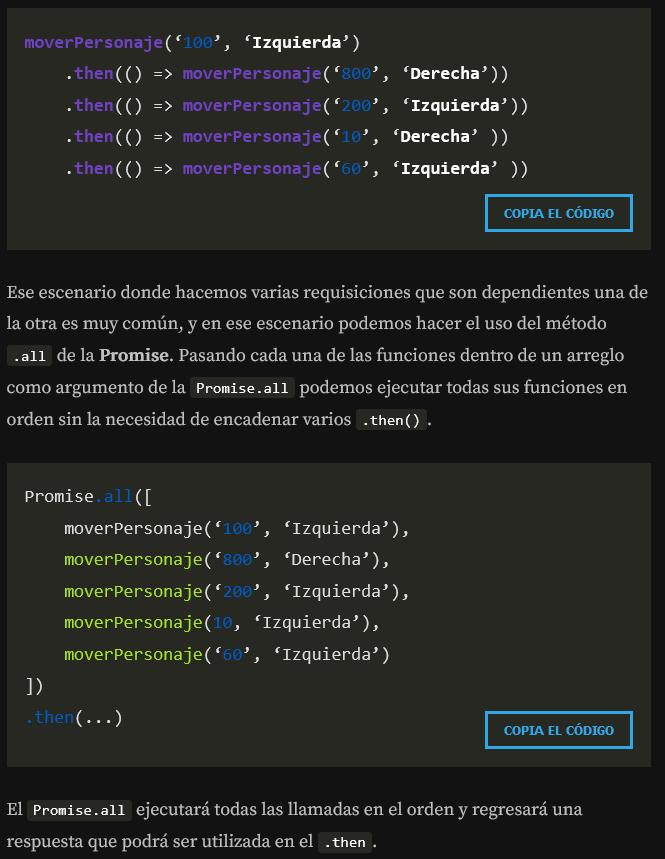
  ]

}









Client-service.js

const listaClientes = () =>

fetch("http://localhost:3000/perfil").then((*respuesta*) => *respuesta*.json()); // Esto hace que se pueda exportar a otros archivos para poder usarlo(?)

const crearCliente = (*nombre*, *email*) => {

    return fetch("http://localhost:3000/perfil", {

        // Esto es para crear un nuevo cliente

        method: "POST",

        headers: {

            "Content-Type": "application/json",

        },

        body: JSON.stringify({ nombre, email, id: uuid.v4() }), // JSON stringify convierte un objeto a un string

    });

};

[npm install -g browser-sync]

<https://www.aluracursos.com/blog/empezando-con-fetch-en-javascript?utm_source=gnarus&utm_medium=timeline>

const eliminarCliente = (*id*) => {

    return fetch(`http://localhost:3000/perfil/${*id*}`, {

        method: "DELETE",

    });

};

const detalleCliente = (*id*) => {

    return fetch(`http://localhost:3000/perfil/${*id*}`).then((*respuesta*) =>

*respuesta*.json()

    );

};

const actualizarCliente = (*nombre*, *email*, *id*) => {

    return fetch(`http://localhost:3000/perfil/${*id*}`, {

            method: "PUT",

            headers: {

                "Content-Type": "application/json",

            },

            body: JSON.stringify({ nombre, email }), // JSON stringify convierte un objeto a un string

        })

        .then((*respuesta*) => *respuesta*) // Esto es para devolver la respuesta

        .catch((*err*) => console.log(*err*)); // Esto es para que si hay un error, lo muestre en la consola

};

export const clientServices = {

    listaClientes,

    crearCliente,

    eliminarCliente,

    detalleCliente,

    actualizarCliente,

};

ClientController.js

import { clientServices } from "../service/client-service.js";

//backticks

const crearNuevaLinea = (*nombre*, *email*, *id*) => {

  const linea = document.createElement("tr");

  const contenido = `

    <td class="td" data-td>

      ${*nombre*}

    </td>

    <td>${*email*}</td>

    <td>

      <ul class="table\_\_button-control">

        <li>

          <a

            href="../screens/editar\_cliente.html?id=${*id*}"

            class="simple-button simple-button--edit"

          >

            Editar

          </a>

        </li>

        <li>

          <button class="simple-button simple-button--delete" type="button" id="${*id*}">

            Eliminar

          </button>

        </li>

      </ul>

    </td>

  `;

  linea.innerHTML = contenido;

  const btn = linea.querySelector("button");

  btn.addEventListener("click", () => {

    const id = btn.id;

    clientServices

      .eliminarCliente(id)

      .then((*respuesta*) => {

        console.log(*respuesta*);

      })

      .catch((*err*) => alert("Ocurrió un error"));

  });

  return linea;

};

const table = document.querySelector("[data-table]");

clientServices

  .listaClientes()

  .then((*data*) => {

*data*.forEach(({ *nombre*, *email*, *id* }) => {

      const nuevaLinea = crearNuevaLinea(*nombre*, *email*, *id*);

      table.appendChild(nuevaLinea);

    });

  })

  .catch((*error*) => alert("Ocurrió un error"));



<http://latentflip.com/loupe/?code=JC5vbignYnV0dG9uJywgJ2NsaWNrJywgZnVuY3Rpb24gb25DbGljaygpIHsKICAgIHNldFRpbWVvdXQoZnVuY3Rpb24gdGltZXIoKSB7CiAgICAgICAgY29uc29sZS5sb2coJ1lvdSBjbGlja2VkIHRoZSBidXR0b24hJyk7ICAgIAogICAgfSwgMjAwMCk7Cn0pOwoKY29uc29sZS5sb2coIkhpIS>

registroCliente.js

import { clientServices } from "../service/client-service.js";

const formulario = document.querySelector("[data-form]");

formulario.addEventListener("submit", (*evento*) => {

*evento*.preventDefault();

    const nombre = document.querySelector("[data-nombre]").value;

    const email = document.querySelector("[data-email]").value;

    clientServices

        .crearCliente(nombre, email)

        .then(() => {

            window.location.href = "/screens/registro\_completado.html";

        })

        .catch((*err*) => console.log(*err*));

});



<https://www.aluracursos.com/blog/diferencias-entre-get-y-post>