# README.md:

* Es obligatorio documentar el código realizado en cada uno de los ejercicios. Su no inclusión penaliza con 0,5 puntos en cada ejercicio no aplicado.
* Se tendrá en cuenta para la puntuación final de cada ejercicio no sólo la validez de este sino también la forma de implementar la solución, limpieza de código y gestión de errores.
* La forma de calificar cada pregunta es la siguiente:  
  10% puntos por legibilidad del código (tabulaciones, declaración de  
  variables etc).
* 20% puntos por bug no encontrados.
* -0,5 puntos por la ausencia de comentarios.
* 70% puntos por la funcionalidad, teniendo en cuenta la forma de  
  resolver el problema.
* Diríjase al aula virtual a la sección Examen-Primer-Trimestre, para descargar el material.
* Realizar los siguientes ejercicios en JavaScript. Para ello, en primer lugar, se debe editar el nombre del directorio descargado con sus datos personales "apellidos\_nombre\_DWEC\_Examen", donde se alojarán los ejercicios del examen, en sus respectivas carpetas.
* Debe tener en cuenta usar let para variables de bloque, var para variables globales y const para constantes.
* Para el examen NO se puede utilizar ningún Framework de JavaScript, ni Inteligencia Artificial. Solo está permitido JavaScript, las presentaciones de clase, la documentación oficial de Mozilla y demás herramientas que autorice el profesor si es el caso.
* No está permitido el uso de lenguajes de programación que no sea JavaScript.
* Para cada ejercicio se debe descargar el archivo zip que contiene todos los directorios.
* Cada directorio será un proyecto y corresponderá a una pregunta del examen.
* Se debe crear un repositorio Git que sea la raíz de todos los demás directorios, es decir, sobre "apellidos\_nombre\_DWEC\_Examen”.
* Cambiar de nombre la rama máster a main y crear la rama examen desde donde se continuará.
* El contenido de esta rama será la que se entregue posteriormente en archivo comprimido(solamente Código Fuente sin la carpeta .git).
* Se debe el repositorio a GitHub, con todas sus ramas y se harán únicamente dos commit en la rama examen, uno inicial que diga “Inicio examen” y el final cuando se termine el código, deberá tener el comentario “Examen finalizado”.
* Finalmente se creará un tag llamado v0.1-examen sobre la rama examen.
* Si el código fuente del tag no coincide con el del archivo zip que se subirá a educamos se pondrá una calificación de uno.
* Escribir el enlace al repositorio al inicio del documento como se indica en la cabecera.
* La calificación acumulará un total de 10 puntos.

# R.A. 1. Selecciona las arquitecturas y tecnologías de programación sobre clientes Web, identificando y analizando las capacidades y características de cada una.

Criterios de evaluación:

a) Se han caracterizado y diferenciado los modelos de ejecución de código en el servidor y en el cliente Web.

b) Se han identificado las capacidades y mecanismos de ejecución de código de los navegadores Web.

c) Se han identificado y caracterizado los principales lenguajes relacionados con la programación de clientes Web.

d) Se han reconocido las particularidades de la programación de guiones y sus ventajas y desventajas sobre la programación tradicional.

e) Se han verificado los mecanismos de integración de los lenguajes de marcas con los lenguajes de programación de clientes Web.

f) Se han reconocido y evaluado las herramientas de programación sobre clientes Web.

Ejercicio 1: Puntos 0.5

* Este criterio será evaluado con el cuestionario que se encuentra en el aula virtual, con el empleo de Git, GitHub, PRÁCTICA\_UT1, el uso del entorno de ejecución Node.js y el editor VS code.
* Ver en la sección del aula virtual denominada Examen-Primer-Trimestre para la resolución de este

# R.A. 2. Escribe sentencias simples, aplicando la sintaxis del lenguaje y verificando su ejecución sobre navegadores Web.

Criterios de evaluación:

a) Se ha seleccionado un lenguaje de programación de clientes Web en función de sus posibilidades.

b) Se han utilizado los distintos tipos de variables y operadores disponibles en el lenguaje.

c) Se han identificado los ámbitos de utilización de las variables.

d) Se han reconocido y comprobado las peculiaridades del lenguaje respecto a las conversiones entre distintos tipos de datos.

e) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.

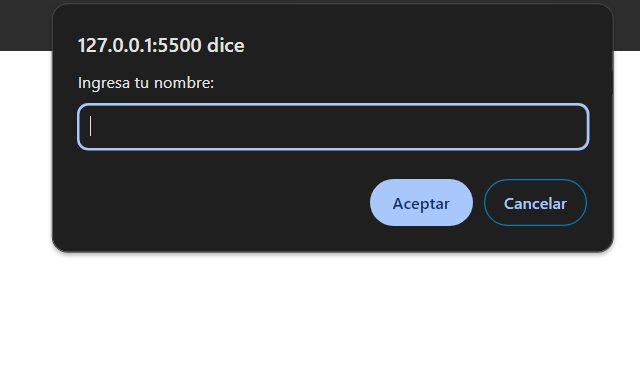
f) Se han utilizado bucles y se ha verificado su funcionamiento.

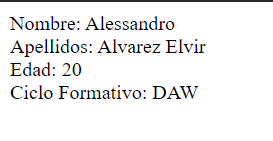
g) Se han añadido comentarios al código.

h) Se han utilizado herramientas y entornos para facilitar la programación, prueba y depuración del código.

Ejercicio 2: Puntos 0.5

En el Directorio datos-estudiante deberá crear un script que pregunte su nombre, apellidos, su edad y el ciclo formativo que cursa. Los mostrará en la página web. Cada dato debe ser preguntado en una ventana emergente distinta y mostrarlo en una línea distinta, en el campo del ciclo deberá mostrar el texto por defecto “DAW”.

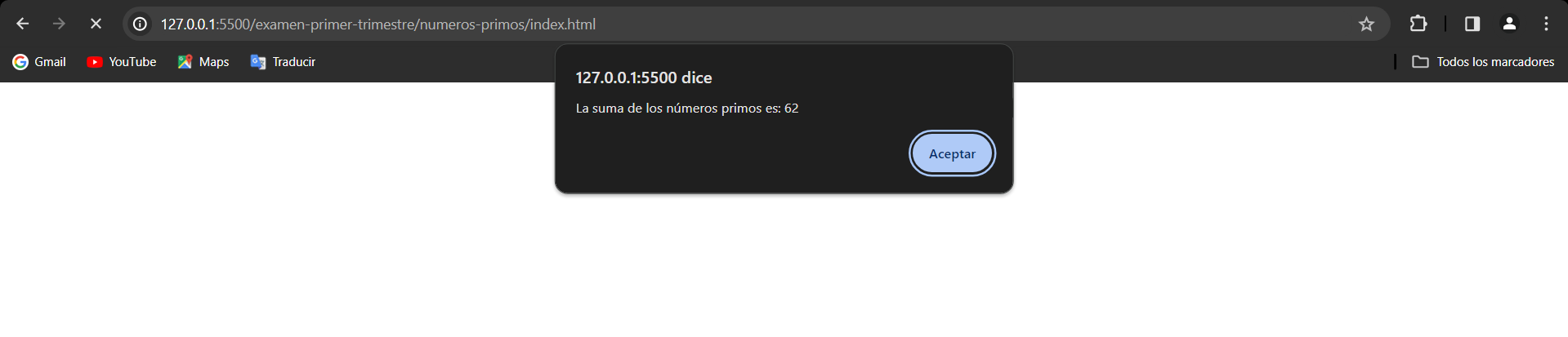




Ejercicio 3: Puntos 1.0

En el directorio números-primos deberá programar embebido el código directamente en el HTML el siguiente script:

Crea un script con una función llamada calcularSumaPrimos que pida al usuario 15 números y al hacer click muestre en pantalla únicamente la suma de los que son números primos. Tenga en cuenta que al solicitar información por el prompt se debe dar formato de número antes de programar.



# R.A. 3. Escribe código, identificando y aplicando las funcionalidades aportadas por los objetos predefinidos del lenguaje.

Criterios de evaluación:

a) Se han identificado los objetos predefinidos del lenguaje.

b) Se han analizado los objetos referentes a las ventanas del navegador y los documentos web que contienen.

c) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para cambiar el aspecto del navegador y el documento que contiene.

d) Se han generado textos y etiquetas como resultado de la ejecución de código en el navegador.

e) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para interactuar con el usuario.

f) Se han utilizado las características propias del lenguaje en documentos compuestos por varias ventanas y marcos.

h) Se ha depurado y documentado el código.

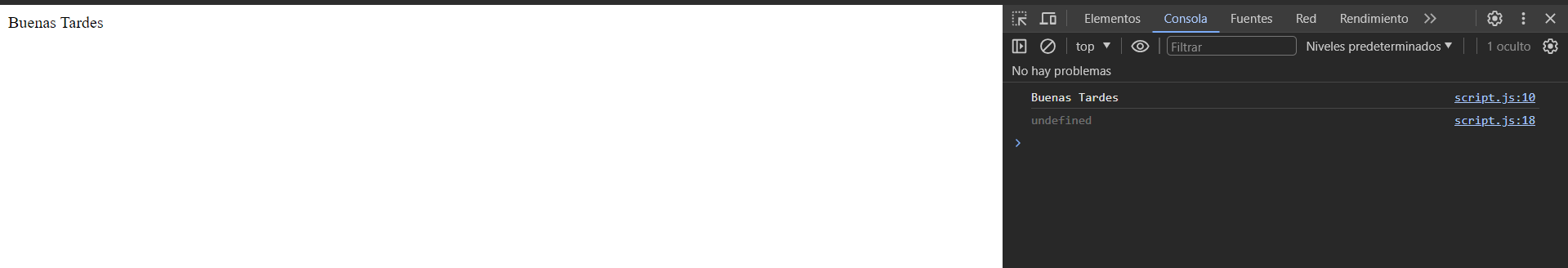
Ejercicio 4: Puntos 0.5

En el Directorio if-op-logicos tiene la estructura básica de un HTML, deberá crear un fichero llamado script.js, enlazarlo al archivo HTML y crear el siguiente programa:

Escribirá una función llamada saludarSegunHora(), en ella deberá instanciar un objeto de la clase Date() que es un objeto predefinido de JavaScript, este objeto tiene un método llamado getHours() que devuelve la hora actual en formato numérico.

Deberá programar una serie de sentencias condicionales que saluden al usuario según sea la hora: 5 – 12 “buenos días”; 12 – 18 “buenas tardes” y lo demás “buenas noches”.

El saludo se debe escribir sobre el documento HTML utilizando JavaScript dentro de etiquetas de párrafo.



# R.A. 4. Programa código para clientes Web analizando y utilizando estructuras definidas por el usuario.

Criterios de evaluación:

a) Se han clasificado y utilizado las funciones predefinidas del lenguaje.

b) Se han creado y utilizado funciones definidas por el usuario.

c) Se han reconocido las características del lenguaje relativas a la creación y uso de arrays.

d) Se han creado y utilizado arrays.

e) Se han reconocido las características de orientación a objetos del lenguaje.

f) Se ha creado código para definir la estructura de objetos.

g) Se han creado métodos y propiedades.

h) Se ha creado código que haga uso de objetos definidos por el usuario.

i) Se ha depurado y documentado el código.

Ejercicio 5: Puntos 1.0

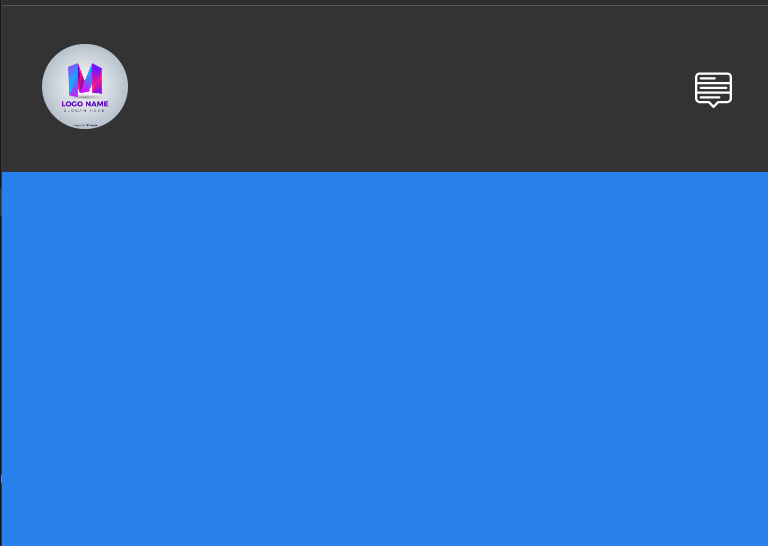
En el Directorio menu-hamburguesa tiene la estructura típica de una página web que al disminuir el tamaño de su pantalla se hace responsive y oculta el header para mostrar un menú hamburguesa alternativo.

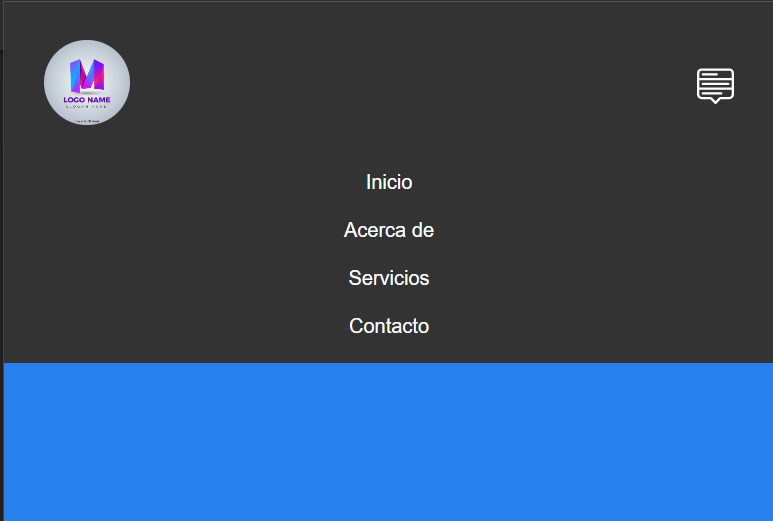
Debe programar la funcionalidad en el archivo hamburguesa.js que realice lo siguiente:

Observe que el elemento imagen del icono de la hamburguesa tiene asignado en el ***media querie*** la clase hamburger, sobre este elemento se debe programar un evento (implementando la función callback en formato flecha) que al darle clic aplique la clase nav-links.active empleando el método toggle. Esto permitirá cada vez que se ejecute cambiar de estado la visibilidad del elemento HTML, es decir, si está visible pasa a oculto y si se encuentra oculto pasa a visible.

De esta manera se desplegará el menú alternativo en pantallas menores o iguales a 768 px.

El código debe tener además de lo anterior, una parte comentada donde se implemente la solución pero modificando la función callback como anónima y en notación declarativa con el identificador toggleNav()





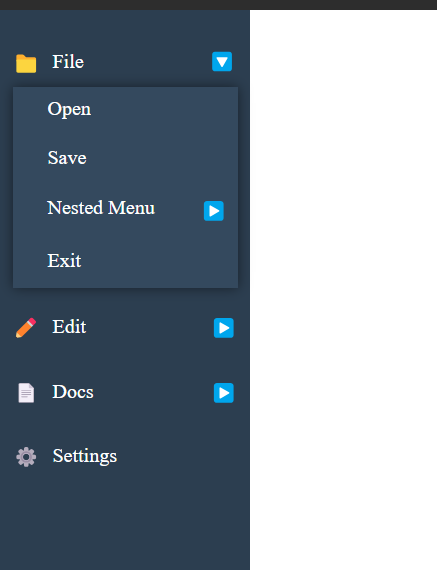
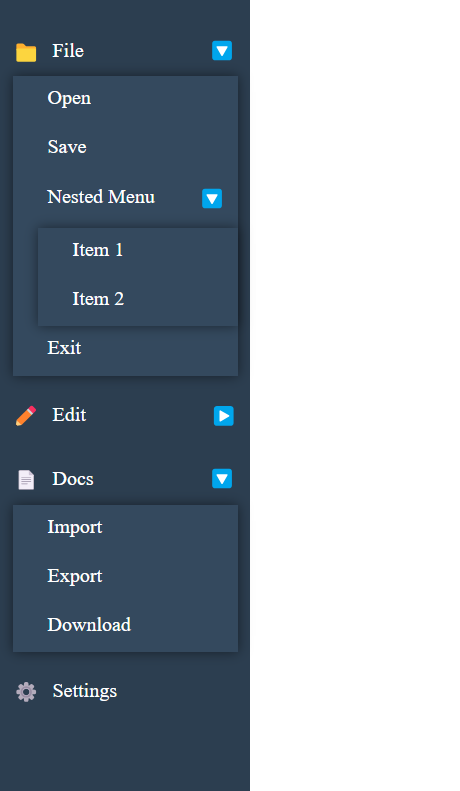
Ejercicio 6: Puntos 1.0

En el directorio menu-lateral-arbol en los archivos HTML y CSS proporcionados se puede observar que hay una serie de iconos denominados con la clase arrow, estos elementos permitirán desplegar unos submenús que tienen asignada la clase .sub-menu y se encuentran ocultos por medio del estilo display: none.

Se pide que por medio de un recorrido forEach se aplique a todas las flechas un evento de clic que a todos los submenús del código le aplique la clase ‘show’ para desplegarlos.

Recomendación: tenga en cuenta que en el contexto de este código para acceder a los submenús desde las flechas se debe ir primero al elemento padre y posteriormente al elemento hermano, de esta manera el objeto seleccionado será relativo a la flecha y siempre será un submenú.

Finalmente aplique la clase rotate-arrow a las flechas con el método toggle para hacer el efecto de giro.



Ejercicio 7: Puntos 2.0

En el Directorio class-Aulas realizar lo siguiente:

Del siguiente diagrama de clases:

Diagrama

Descripción generada automáticamente

Escriba el código necesario para modelar cada una de las clases en un único fichero que llamará Aulas.js, el método presentarMateriales() debe estar acorde al contexto de cada clase y sus atributos (es suficiente con mostrarlos por consola con la estructura ${` … `}).

Sobrescriba el método presentarMateriales() en cada una de las clases.

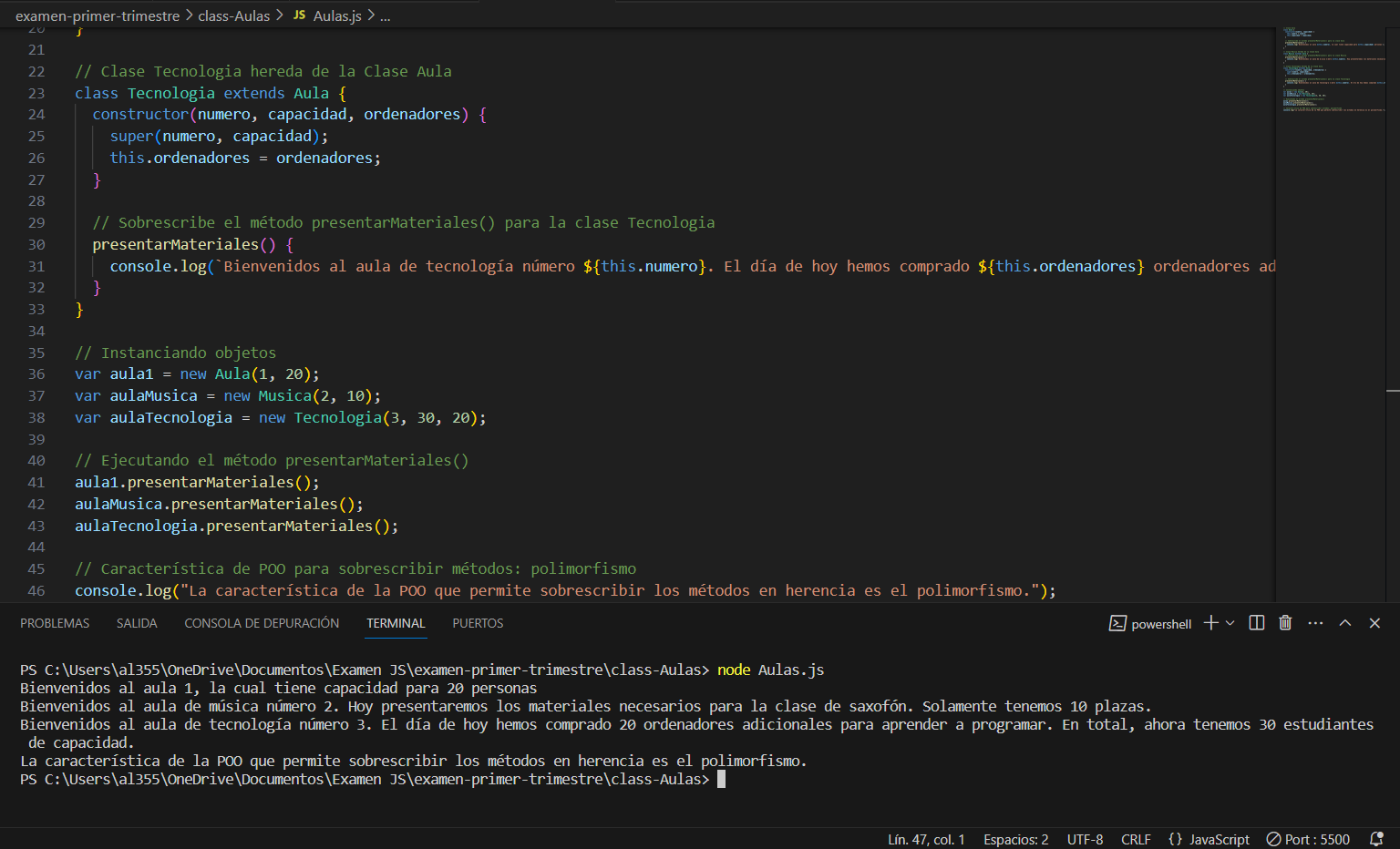
Para la clase Aula se espera que retorne: “Bienvenidos al aula 1, la cual tiene capacidad para 20 personas”.

Para la clase Musica: “Bienvenidos al aula de música número 2. Hoy presentaremos los materiales necesarios para la clase de saxofón, solamente tenemos 10 plazas”.

Para la clase Tecnologia: “Bienvenidos al aula de tecnología número 3. El día de hoy hemos comprador 20 ordenadores adicionales para aprender a programar, en total ahora tenemos 30 estudiantes de capacidad”.

Utilice Node.js para ejecutar el programa instanciando un objeto para cada clase, haga una captura de pantalla con el comando, péguela aquí e inclúyala al repositorio.

¿Cómo se llama la característica de la POO que permite sobrescribir los métodos en herencia? Para responder a esta pregunta haga un console.log al final del código.



Ejercicio 8: Puntos 2.0 (nota: Los valores numéricos que asigna al saldo no tienen por qué ser iguales a las salidas similares que se proponen, se da libertad para escoger la cantidad)

En el directorio prototipo-function-account escriba en un fichero llamado account-prototype.js una función prototipo llamada CuentaBancaria(nombre, dni, saldo), los atributos se deben inicializar con el constructor y el saldo inicial al menos tendrá la alternativa de valer cero o el valor dado por parámetro.

Además deberá tener un atributo que no se incluye en el constructor llamado: numeroCuenta, para inicializarlo utilice el siguiente algoritmo: "ES" + Math.floor(Math.random() \* 99999999999999999999).toString().padStart(20, '0');

Este algoritmo creará un número de cuenta aleatorio por cada instancia con el formato de cuenta español.

El prototipo tendrá cuatro métodos:

* this.mostrarDatos = function() { … }

Dentro del cuerpo del método se debe programar: visualizar por consola los valores de los atributos concatenados usando la estructura ${ … } y separados por un salto de línea.

Se espera una salida similar a:

Nombre: Juan Pérez

DNI: 12345678A

Número de Cuenta: ES29817694284975090000

Saldo: 1000€

* this.realizarIngreso = function(monto) { … }

Dentro del cuerpo se debe programar: una asignación de adición (+=) que simule un depósito en la cuenta con un valor pasado por parámetro denominado monto, que será sumado al saldo. Tenga en cuenta que el monto debe ser un número positivo.

Se debe sacar un mensaje por consola del estilo:

Ingreso realizado. Nuevo saldo: 1200€

* this.realizarRetiro = function(monto) { … }

Dentro del cuerpo se debe programar una asignación de resta (-=) para simular un retiro y un condicional que impida retirar dinero si el saldo es menor a la cantidad que se pretende extraer. Tenga en cuenta que el monto debe ser un número positivo.

Se debe sacar un mensaje por consola del estilo:

Retiro realizado. Nuevo saldo: 1150€

* this.realizarTransferencia = function(cuentaDestino, monto)

Dentro del cuerpo se debe programar un algoritmo que realice una transferencia por el valor del parámetro monto al objeto cuenta que también se pasa por parámetro.

Recomendaciones: El monto debe ser positivo y menor que el saldo, debe utilizar el método anteriormente definido realizarIngreso(monto) para aplicarlo sobre el objeto cuenta al que se le desea hacer la transferencia.

Se debe sacar un mensaje por consola del estilo:

Transferencia realizada. Nuevo saldo: 1050€

Instancie dos objetos para realizar pruebas, las salidas que se esperan deben ser similares a:

Nombre: Juan Pérez

DNI: 12345678A

Número de Cuenta: ES96437901235214400000

Saldo: 1000€

--------------------------------------------------------------

Nombre: Ana Gómez

DNI: 87654321B

Número de Cuenta: ES34618150159602370000

Saldo: 500€

Ingreso realizado. Nuevo saldo: 600€

Transferencia realizada. Nuevo saldo: 900€

--------------------------------------------------------------

Nombre: Juan Pérez

DNI: 12345678A

Número de Cuenta: ES96437901235214400000

Saldo: 900€

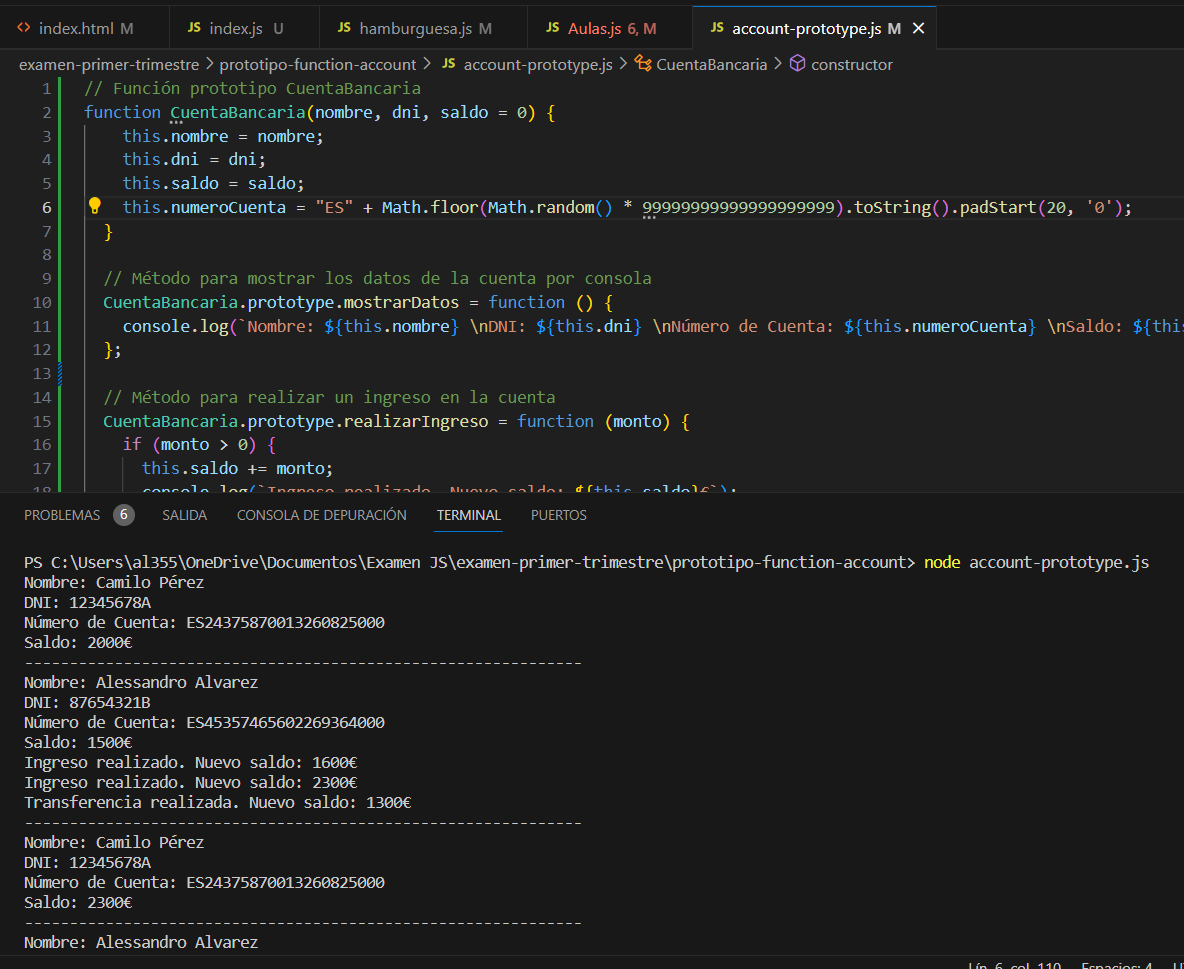
--------------------------------------------------------------

Nombre: Ana Gómez

DNI: 87654321B

Número de Cuenta: ES34618150159602370000

Saldo: 600€



Ejercicio 9: Puntos 1.5

Del directorio llamado poo-jason-api, ejecute el index.html en un navegador, en el campo de nombre de usuario, emplee su cuenta de GitHub para traer sus datos, los cuales podrá visualizarlos en la consola del navegador en formato JASON.

Creará un fichero con el nombre DatosGitHub.js

Utilizará los datos y formará una cadena JASON con lo siguiente:

Login, id, email y public\_repos.

Esa información le servirá como parámetro de entrada para el constructor de una clase que denominará DatosGitHub, la información será asignada como atributos de la clase, recuerde que debe transformar la cadena a objeto JavaScript usando el método parse.

Defina un único método para la clase llamado datosRecibidos() el cual mostrará por consola la información de cada atributo en una cadena formateada y concatenada usando la estructura ${` … `}.

Instancie el objeto con el nombre: datosGitHubInstancia con la cadena JASON que recolectó.

Se espera la salida de tipo:

El Usuario de GitHub \_\_\_\_\_\_ tiene un id\_\_\_\_\_\_\_\_ cuyo correo es \_\_\_\_\_\_\_\_\_\_\_ y tiene \_\_\_\_\_\_ repositorios públicos.

Si al invocar el método datosRecibidos() algún dato venía null desde GitHub adiciónelo al objeto y ejecute de nuevo el método.

Utilice Node.js para ejecutar el programa, haga una captura de pantalla con el comando, péguela aquí e inclúyala al repositorio.

