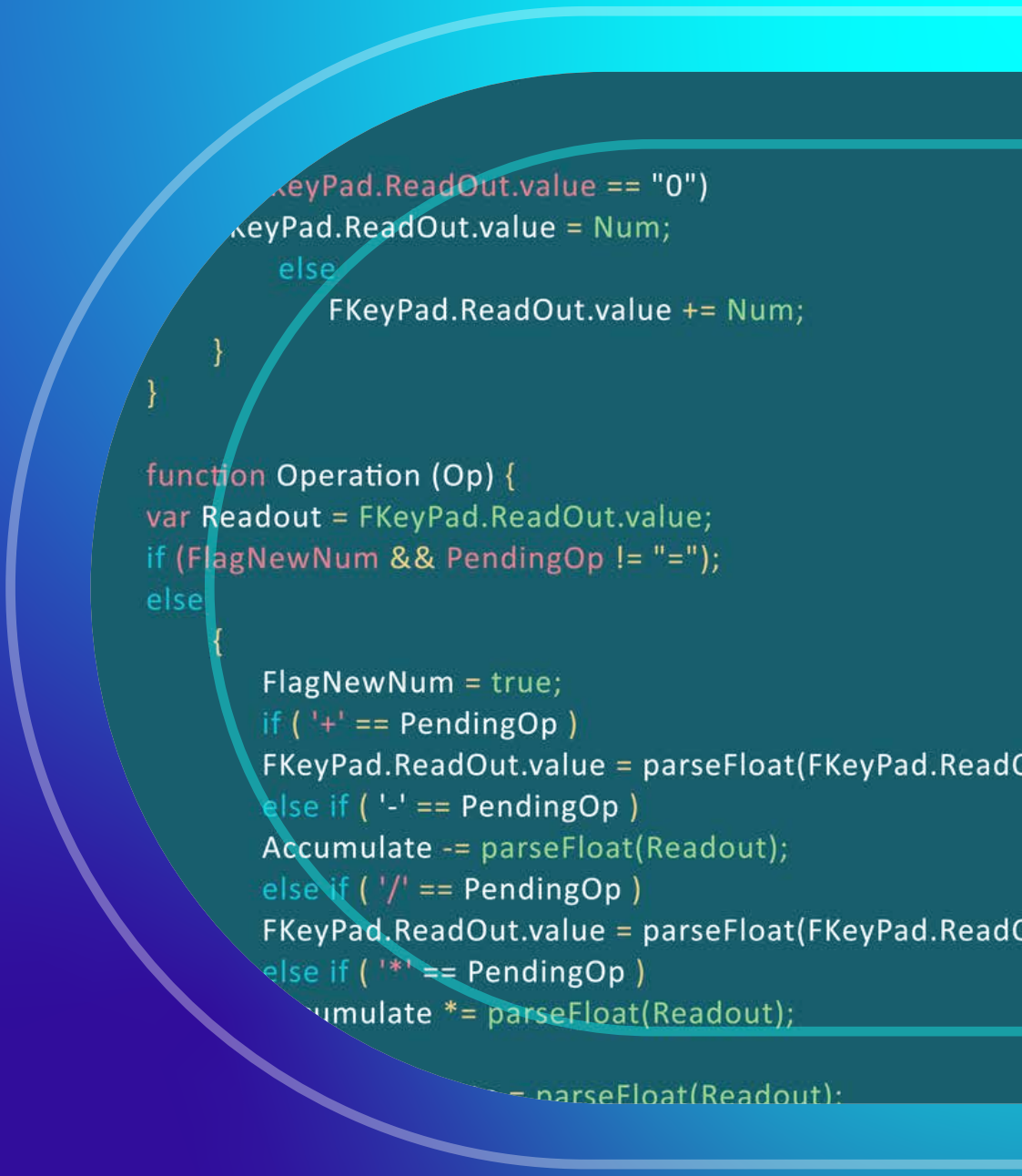


Programación con Java Script I

Proyecto Final

Desarrollo del proyecto

Nombre del proyecto: Aplicación Presupuesto

Objetivo

Poner en práctica algunos de los elementos más utilizados en el desarrollo con JavaScript aprendidos en el curso Programación con JavaScript I.

Descripción

La aplicación realiza el cálculo del presupuesto del usuario a partir de los ingresos y egresos manifestados por el usuario.

Instrucciones detalladas

Con la intención de comenzar a crear tu portafolio y de que practiques los conceptos adquiridos, desarrolla una aplicación web frontend que pueda calcular tu presupuesto. La aplicación tendrá como ejes principales los siguientes elementos de JavaScript:

- Uso de clases
- Creación de funciones
- Programación funcional
- Manejo de eventos y manipulación del DOM

En el curso encontrarás cuatro avances, uno por semana, para que construyas tu proyecto final que deberás entregar en la semana 5. Aunque estos avances no son evaluables, se sugiere que los realices para mayor organización.

Avance 1

Consideraciones Proyecto semana 1

Duración: 2 horas

(Colocar instrucciones y recomendaciones del Avance 1 para el proyecto final incluye temas 1 y 2)

Avance del proyecto semanal.

1. Toma la base del proyecto que se incluye en la sección Archivos adjuntos y ábrelo en Visual Studio Code.
2. Recrea el contenido que se encuentra en la imagen **maqueta.jpg** en el archivo **index.html** del proyecto con las siguientes especificaciones:
    a. Crea la estructura de un archivo html
    b. Agrega la hoja de estilos **"estilos.css"** en el html
    c. Crea el cabecero con las siguientes instrucciones:

Cabecero:

- Crea un **div** en el que utilices la clase cabecero del **css provisto**.
- Dentro de este, crea un **div** con la clase **presupuesto**.
- Dentro de este, crea un **div** con la clase **presupuesto_titulo** y coloca la cadena "Presupuesto Disponible".
- A continuación, crea un **div** con la clase presupuesto_valor, si gustas, ponle +2,000.00.
- Crea otro div con las clases **presupuesto_ingreso** y **limpiarEstilos** y, dentro de este **div**, crea dos **div**:
    - En el primero, coloca la clase **presupuesto_ingreso--texto** y coloca la palabra "Ingresos".
    - En el siguiente **div**, coloca la clase **derecha** y crea nuevamente otros 2 div dentro de este:
        - En el primer **div**, coloca la clase **presupuesto_ingreso—valor** y colócale una cadena, por ejemplo, +4,000.00.
        - En el segundo **div**, colócale las clases **presupuesto_ingreso--porcentaje** y colócale solamente un espacio.
            - Crea un **div** con las clases **presupuesto_egreso** y **limpiarEstilos** y, dentro de este **div**, crea dos **div**:
                - En el primero, coloca la clase **presupuesto_egreso--texto** y coloca la palabra "Egresos" en el cuerpo del div.
                - En el siguiente **div**, coloca la clase **derecha** y **limpiarEstilos**. Dentro de este, crea nuevamente otros 2 **div**:
                    - En el primero, coloca la clase **presupuesto_egreso--valor**. Colócale un valor cualquiera, por ejemplo, –1,900.00.
                    - En el segundo, colócale la clase **presupuesto_egreso—porcentaje**. En el cuerpo del div, colócale 45 %, solo para que se pueda visualizar el resultado.

Avance 1

Form

- Crea un formulario y asígnale el id **forma**.
- Crea un **div** con la clase **agregar**.
- Dentro de este, crea un **div** con la clase **agregar_contenedor**.
- Dentro de este, crea un selector, 3 input y un botón con las siguientes características:
  - Al selector, agrégale la clase **agregar_tipo** y el id **tipo**.
  - Dentro de la clase, agrega dos opciones: la primera con el valor **ingreso** y ponla seleccionada por defecto. En el cuerpo de la opción, colócale el signo +.
  - La segunda opción tendrá el valor **egreso** y colócale el signo - en el cuerpo.
  - El primer input deberá ser de tipo texto y tener la clase **agregar_descripcion**. Como la descripción, colócale "Agregar Descripción" y el id **descripcion**.
  - El segundo input será de tipo numérico y con clase **agregar_valor**, colócale el valor por defecto **valor** y **step any**.
  - El botón tendrá la clase **agregar_btn**. Además, dentro del botón, crea un ícono con la etiqueta **ion-icon** con el nombre **checkmark-circle-outline**.

Para que se pueda visualizar el ícono, agrega hasta el final del cuerpo (antes de la etiqueta </body>) una etiqueta script del tipo module y el source.
https://unpkg.com/ionicons@5.4.0/dist/ionicons/ionicons.esm.js.

Contenedor

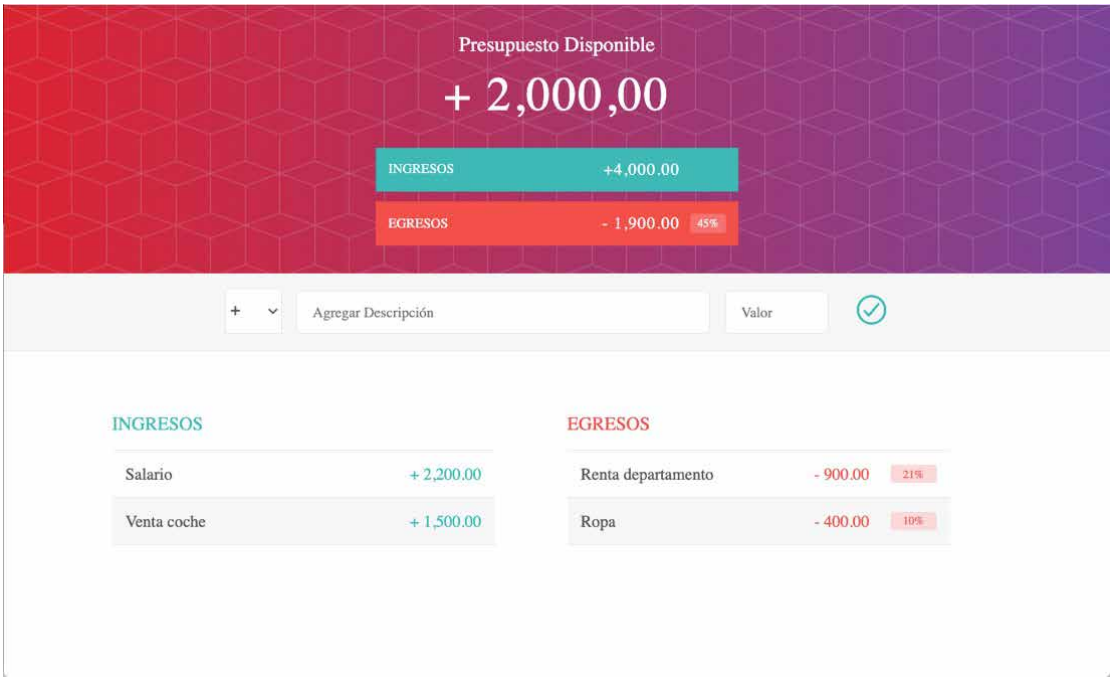
- Crea un div con las clases contenedor y limpiarEstilos.
- Dentro de este, crea dos contenedores principales, uno con la clase ingreso y otro con la clase egreso.
- Dentro del div ingreso, crea lo siguiente:
  - H2 clase ingreso_titulo y en el cuerpo la palabra Ingresos.
  - Un div con id lista-ingresos:
  - Dentro un div con las clases elemento y limpiarEstilos
  - Agrega dos divs:
    - El primero tendrá la clase elemento_descripcion y el contenido dirá "salario".
    - El segundo tendrá las clases derecha y limpiarEstilos y tendrá dos div dentro del cuerpo:
      - El primero tendrá la clase elemento_valor y tendrá un valor, por ejemplo +2,200.00.
      - El segundo tendrá la clase elemento_eliminar y en el cuerpo tendrá un botón:
      - El botón tendrá la clase elemento_eliminar_btn
      - Dentro de este, crea un ícono ion-icon con el nombre close-circle-outline.

Avance 1

- Crea otro **div** con las clases **elemento** y **limpiarEstilos**.
- Agrega dos **divs**:
  - El primero tendrá la clase **elemento_descripcion** y el contenido dirá “Venta Coche”.
  - El segundo tendrá las clases derecha y limpiarEstilos y tendrá dos div dentro del cuerpo:
    - El primero tendrá la clase **elemento_valor** y tendrá un valor, por ejemplo, +1,500.00.
    - El segundo tendrá la clase **elemento_eliminar** y en el cuerpo tendrá un botón:
    - El botón tendrá la clase **elemento_eliminar_btn.**
    - Dentro del botón, crea un ícono ion-icon con el nombre close-circle-outline
- Dentro del **div** con la clase egreso, crea lo siguiente:
  - H2 clase **egreso_titulo** y en el cuerpo la palabra “Egresos”.
  - Un div con id **lista-egresos:**
    - Dentro un **div** con las clases elemento y limpiarEstilos.
    - Agrega dos **div:**
      - El primero tendrá la clase elemento_descripcion y el contenido dirá, por ejemplo, “Renta departamento”.
      - El segundo tendrá las clases derecha y limpiarEstilos y tendrá tres div dentro del cuerpo:
        - El primero tendrá la clase **elemento_valor** y tendrá un valor, por ejemplo, -900.00.
        - El segundo tendrá la clase **elemento_porcentaje** y ponle en el cuerpo, por ejemplo, 21 %.
        - El tercero tendrá la clase **elemento_eliminar** y en el cuerpo tendrá un botón:
        - El botón tendrá la clase **elemento_eliminar_btn**.
        - Dentro del botón, crea un ícono ion-icon con el nombre **close-circle-outline**.
    - Crea otro **div** con las clases elemento y limpiarEstilos.
    - Agrega dos **divs**:
      - El primero tendrá la clase **elemento_descripcion** y el contenido dirá “Ropa”.
      - El segundo tendrá las clases **derecha y limpiarEstilos** y tendrá tres div dentro del cuerpo:
        - El primero tendrá la clase **elemento_valor** y tendrá un valor, por ejemplo, -400.00.
        - El segundo tendrá la clase **elemento_porcentaje** y ponle en el cuerpo, por ejemplo, 9 %.
        - El tercero tendrá la clase **elemento_eliminar** y en el cuerpo tendrá un botón:
        - El botón tendrá la clase **elemento_eliminar_btn.**
        - Dentro del botón, crea un ícono ion-icon con el nombre **close-circle-outline.**

Avance 1

Con esto habrás terminado el maquetado de tu página web y tu página deberá lucir más o menos así:



Aunque aún no luce exactamente como debería quedar, esas pequeñas diferencias las quitaremos con el código JavaScript.

Consideraciones

Maestro:

- En esta semana es importante validar que el aprendedor sepa utilizar HTML para poder crear las secciones solicitadas. Se sugiere poner especial atención en el uso de las etiquetas div y de los id en estas etiquetas para poder tener acceso a los elementos desde JavaScript.

  Haz notar al aprendedor la importancia del manejo de HTML y CSS para JavaScript.
  Se sugiere que se fomente el uso de GIT y que el aprendedor pueda manipularlo.

Alumno:

- Asegúrate de tener instalado el plugin de Live server para que puedas visualizar los cambios realizados en el proyecto.
  Una vez que tengas todos tus cambios realizados, recuerda subirlos a tu repositorio GIT.

Avance 2

Consideraciones Proyecto semana 2

Duración: 2 horas

(Colocar instrucciones y recomendaciones del Avance 2 para el proyecto final, incluye hasta el tema 4)

Avance del proyecto semanal.

En este avance se van a crear las funciones que van a cargar dinámicamente los datos en el html. Dichas funciones son las que realizarán los cálculos de ingresos y egresos y las que van a dar formato a los números.

- Crea la función **cargarCabecero** con las siguientes características:
  - Es una función flecha.
  - Crea una variable llamada **presupuesto** y asígnale el resultado de la resta de la función totalIngresos() menos el contenido de la función **totalEgresos()**.
  - Crea una variable **porcentajeEgreso** y asígnale el resultado de la división de la función **totalEgresos()** entre el resultado de la función **totalIngresos().**
- Como puedes ver, se requieren un par de funciones **totalIngresos y totalEgresos**, para ello, realiza lo siguiente:
  - Crea la función **totalIngresos** con estas características:
    - Defínela como función flecha.
    - Declara la variable **totalIngresos** e inicialízala en 0.
    - Itera el arreglo **ingresos** en un ciclo for of. Recuerda que es necesario declarar la variable **ingreso** para poder iterar.
    - Dentro del ciclo, a la variable **totalIngreso**, suma cada uno de los valores de los elementos del arreglo **ingresos**.
    - Cuando termine de iterar, regresa la variable **totalIngreso**.
  - Crea la función **totalEgresos** con estas características:
    - Defínela como una función flecha.
    - Dentro del cuerpo de la función, declara la variable **totalEgreso** e inicialízala en 0.
    - Con un ciclo for, itera para cada elemento egreso del arreglo egresos y súmalo a la variable **totalEgreso.**
    - Cuando termine de iterar, retorna la variable **totalEgreso.**
  - Para validar el funcionamiento de cargar cabecera, crea dos arreglos:

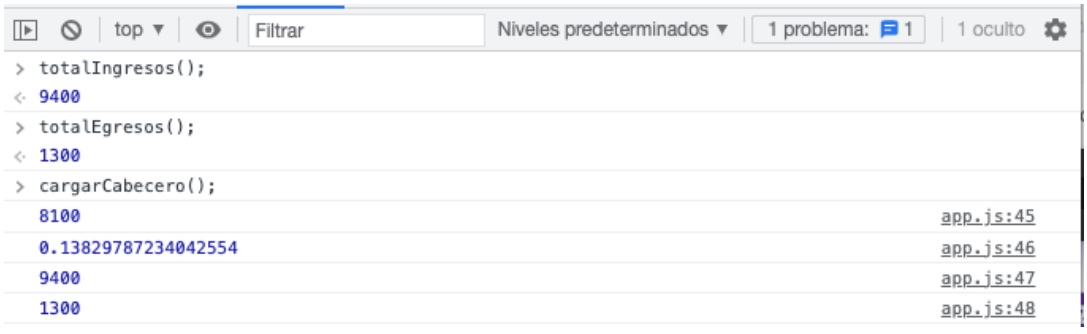
```
let egresos = {
Renta: 900,
Ropa: 400
};

let ingresos = {
Quincena: 9000,
Venta: 400
};
```

o Coloca las siguientes líneas en la función cargarCabecero:

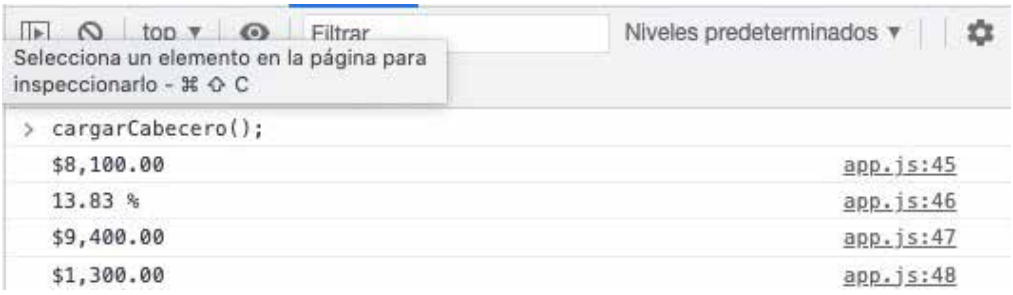
```
console.log(presupuesto);
console.log(porcentajeEgreso);
console.log(totalIngresos());
console.log(totalEgresos());
```

- El resultado de la prueba debería ser el siguiente:



Avance 2

- Ahora, es necesario darle **formato a la moneda** y al porcentaje. Para ello, crea dos funciones:
  - Crea la función **formatoMoneda** con las siguientes características:
    - Créala como función flecha.
    - Se debe recibir el valor que se requiere formatear.
    - Utiliza la función **toLocaleString**, especificando el idioma, y luego un arreglo indicando el estilo, que en este caso sería moneda, el tipo de moneda, en el que se sugiere pongas el valor MXN y mínimo de dígitos decimales igual a 2.
  - Crea la función **formatoPorcentaje** con las siguientes características:
    - Créala como función flecha.
    - Se debe recibir el valor que se requiere formatear.
    - Utiliza la función **toLocaleString**, especificando el idioma que se desea utilizar. Posteriormente, especifica un objeto en el que indiques el estilo, para este caso será **percent** y el mínimo de dígitos decimales igual a 2.
  - Aplica los formatos a las salidas de la consola y el resultado debería ser el siguiente:



Consideraciones

Maestro:

- Posiblemente a algunos de los aprendedores les cueste trabajo utilizar las funciones flecha, por lo tanto, refuerza este tema.

Alumno:

- Asegúrate de saber utilizar las funciones flecha, ya que son las más fáciles de utilizar en los proyectos.

Avance 3

Consideraciones Proyecto semana 3

Duración: 2 horas

(Colocar instrucciones y recomendaciones del Avance 3 para el proyecto final, incluye hasta el tema 6))

Avance del proyecto semanal.

En este avance vas a crear las clases de la aplicación.

- En el archivo llamado Dato.js crea una clase padre para el manejo de datos que se llame Dato
- Agrega un constructor al cual le vas a pasar dos atributos: descripción y valor
- Dentro del cuerpo del constructor define los atributos _descripción y _valor y asígnale el valor de los parámetros que se están recibiendo
- Agrega los métodos ***get*** y ***set*** para el atributo descripción:
  - En el **get** retorna este valor del atributo **descripcion**.
  - En el **set**, haz que se reciba el parámetro **descripción** y modifica el atributo de **descripción** con el valor que se está recibiendo.
- Crea los métodos set y get para el atributo valor de la misma manera que se realizó para el atributo **descripción** con sus respectivos valores.
- Recuerda que es importante encapsular los datos

Define ahora las clases hijas:

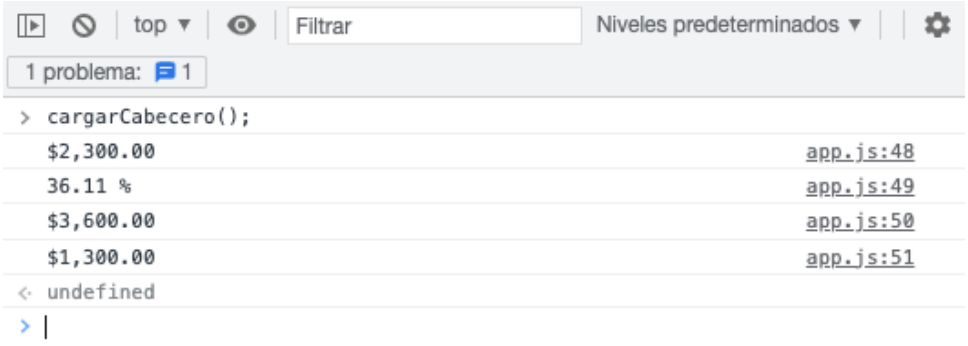
- Utilizando el archivo **Ingreso.js**, define la clase **Ingreso:**
  - Define la clase ingreso, que extiende de la clase padre **Dato**.
  - Define la variable estática contadorIngresos e iguálala a 0.
  - Crea el constructor de la clase en el que recibas los parámetros **descripción y valor.**
  - En el cuerpo del constructor, invoca al constructor de la clase padre, la cual debe recibir los parámetros **descripción y valor.**
  - Define el atributo _id y para asignarle un valor, utiliza la variable estética, para hacerlo, realiza un preincremento a la variable estática de la clase **Ingresos**.
  - Crea el método **get id**, el cual va a regresar el valor de this._id, no agregues el método set porque este valor no deberá ser modificado.

Avance 3

- Utilizando el archivo **Egreso.js**, crea la clase Egreso, que es hija de la clase Dato:
  - Define una variable estática contadorEgresos inicializada en 0.
    - Crea el constructor que recibe los valores de **descripcion y valor.**
    - Inicializa el objeto de la clase **padre,** el cual recibe el valor de descripcion y valor.
    - Define el atributo **_id** y, para asignarle un valor, utiliza la variable estática. Para hacerlo, realiza un preincremento a la variable estática de la clase Egresos para asegurar que inicie en 1.
    - Crea el método **get id**, el cual va a regresar el valor de this._id, no agregues el método set, pues este valor no deberá ser modificado.

  - Crea los arreglos que van a manejar los ingresos y los egresos de la aplicación (elimina los arreglos que se habían creado para poder realizar la prueba de las funciones). Para ello, define la variable ingresos y egresos fuera de toda función (al principio del archivo de ser posible) en el archivo app.js de la siguiente manera:
    - El objeto **ingresos** es un arreglo que instancia a la clase Ingreso y recibe como parámetros, a manera de ejemplo, los valores ('Salario', 20000) y ('Venta auto', 50000).
    - La constante **egresos** es un arreglo que crea dos objetos del tipo Egreso y recibe como parámetros, a manera de ejemplo, los valores ('Renta', 4000) y ('Ropa', 800).

- Modifica la clase **totalIngresos** y recorre el arreglo **ingresos** con un ciclo for of. Asigna a la variable **totalIngresos** la suma del valor del elemento **ingreso.**
- Modifica la clase totalEgresos y recorre el arreglo egresos con un ciclo for of. Asigna a la variable **totalEgreso** la suma del valor del elemento **ingreso.**
- Prueba nuevamente y valida que la salida de la instrucción **cargarCabecero()** sea la siguiente:



Avance 3

Consideraciones

Maestro:

- Conforme a la planeación de tu clase, de ser posible, refuerza el concepto de clases y resalta la diferencia entre las funciones y las clases.

Alumno:

- Asegúrate de comprender la diferencia entre clase y función y sigue los pequeños tips de nomenclatura indicados en la literatura. El test en la consola te puede ayudar para ver los resultados.

Avance 4

Consideraciones Proyecto semana 4

Duración: 2 horas

Comencemos ahora a presentar de manera dinámica los datos generados. Para hacerlo, realiza lo siguiente:

Avance del proyecto semanal.

Comencemos ahora a presentar de manera dinámica los datos generados. Para hacerlo, realiza lo siguiente:

Cabecero:

- En el archivo **index.html,** agrega
  - El id **presupuesto al div** con clase **presupuesto_valor.**
  - El id **ingresos al div** con clase **presupuesto_ingreso--valor.**
  - El id **egresos al div** con la clase **presupuesto_egreso--valor.**
  - El id **porcentaje al div** con la clase **presupuesto_egres--porcentaje.**

- En el archivo **app.js,** modifica la función **cargarCabecero** para que, en lugar de imprimir en consola los valores generados, puedas recuperar por id los siguientes elementos:
  - **Presupuesto:** en la propiedad **innerHTML**, modifica su valor, asignando el contenido de la variable presupuesto pasada por la función **formatoMoneda**.
  - **Porcentaje:** en la propiedad innerHTML, modifica su valor, asignando el contenido de la variable **porcentajeEgreso** pasada por la función formatoPorcentaje.
  - **Ingresos:** en la propiedad **innerHTML**, modifica su valor, asignando el valor resultante de la invocación de la función totalIngresos pasada por la función formatoMoneda.
  - **Egresos:** en la propiedad **innerHTML**, modifica su valor, asignando el valor resultante de la invocación de la función **totalEgresos** contenido de la variable **presupuesto** pasada por la función **formatoMoneda.**

- En este momento, al recargar el documento, debería sustituir el contenido del elemento que tiene el id **presupuesto, porcentaje, ingresos y egresos** del contenedor cabecero, sin embargo, no lo hace. Para que lo haga, crea la función **cargarApp** en la que vas a mandar llamar a la función **cargarCabecero().**
- En el documento html, indícale en el body que, cuando cargue (con el evento **onload**), debe mandar llamar a la función **cargarApp().**
- Prueba tu aplicación, ahora se deben mostrar en pantalla los valores calculados en lugar de los valores asignados estáticamente.

Avance 4

Ingresos:

• Para cargar los ingresos dinámicamente, es necesario recorrer el arreglo ingresos. Para ello, crea la siguiente función:

- Nómbrala **cargarIngresos.**
- Debe ser una función tipo flecha.
- Declara la variable vacía **ingresosHTML.**
- Recorre el arreglo **ingresos con** un ciclo for of.
- En el cuerpo del ciclo, concatena el resultado de una función llamada **crearIngresoHTML** y pásale como parámetro cada uno de los elementos del arreglo **ingreso.**
- Una vez que termine el ciclo, recupera el elemento **lista-ingresos** a través de su id o asígnale el contenido de la variable **ingresosHTML.**

• Como se desea que cada elemento generado por **crearIngresoHTML** cree dinámicamente el contenido del **div lista-ingresos**, debes crear todo el contenido html dentro de la función. Para lograrlo, haz lo siguiente:

- Declara la función flecha crearIngresoHTML.
- Pásale como parámetro ingreso.
- Declara la variable **ingresoHTML** y asígnale, por medio de template string, el contenido del div lista-ingresos con los siguientes cambios:
  - En lugar de escribir una cadena en el **div elemento-descripcion**, toma el contenido de **ingreso.descripcion.**
  - En el contenido del **div elemento-valor**, asígnale el valor del elemento ingreso pasado por la función formatoMoneda.
  - En el ícono **close-circle-outline**, asígnale el evento onclick e iguálalo a la función elimiarIngreso y pásale como parámetro el id del elemento ingreso.

- Retorna la cadena ingresoHTML.
- Manda llamar esta función **cargarIngresos()** desde la función **cargarApp().**
- Modifica tu arreglo **ingresos** (cambia valores y agrega un nuevo elemento al arreglo) y valida que se estén cargando correctamente los elementos en tu aplicación.

Avance 4

Egresos:

• Para cargar los egresos dinámicamente, es necesario recorrer el arreglo egresos. Para ello, crea la siguiente función:

- Nómbrala **cargarEgresos.**
- Debe ser una función tipo flecha.
- Declara la variable vacía **egresosHTML**.
- Recorre el arreglo ingresos con un ciclo for of.
- En el cuerpo del ciclo, concatena el resultado de una función llamada **crearEgresoHTML** y pásale como parámetro cada uno de los elementos del arreglo egreso.
- Una vez que termine el ciclo, recupera el elemento **lista-egresos** a través de su id o asígnale el contenido de la variable **egresosHTML**.

• Como se desea que cada elemento generado por **crearEgresoHTML** cree dinámicamente el contenido del **div lista-ingresos**, debes crear todo el contenido html dentro de la función. Para lograrlo, haz lo siguiente:

- Declara la función flecha **crearEgresoHTML**.
- Pásale como parámetro **egreso**.
- Declara la variable **egresoHTML** y asígnale, por medio de template string, el contenido del div lista-egresos con los siguientes cambios:
    - En lugar de escribir una cadena en el **div elemento-descripcion**, toma el contenido de egreso.descripcion.
    - En el contenido del **div elemento-valor**, asígnale el valor del elemento egreso pasado por la función **formatoMoneda**.
    - En el ícono **close-circle-outline**, asígnale el evento onclick e iguálalo a la función **eliminarEgreso** y pásale como parámetro el id del elemento **egreso.**

- Retorna la cadena **egresoHTML**.
- Manda llamar a la función **cargarEgresos()** desde la función **cargarApp()**.
- Modifica tu arreglo egresos (cambia valores y agrega un nuevo elemento al arreglo) y valida que se estén cargando correctamente los elementos en tu **aplicación**.

Avance 4

- Para programar la función eliminarEgreso llamada desde el evento onclick, realiza lo siguiente:
    - Define la función flecha.
    - Recibe el parámetro id.
    - En el cuerpo de la función, crea una variable llamada indiceEliminar y asígnale el método findIndex del arreglo egresos. En este método, obtén el índice del elemento que se desea eliminar, a través de una función flecha que pasa como parámetro. Lo que hace esa función es que cuando el id pasado como argumento coincide con el id del elemento egreso evaluado, regresa el id del elemento.
    - Utilizando el método splice del arreglo egresos, indícale que elimine el elemento contenido en la variable indiceEliminar y, como segundo parámetro, indícale que solo se desea eliminar 1 elemento.
    - Recarga el cabecero llamando a la función cargarCabecero() para que se realicen los cálculos con los elementos que queden en el arreglo egresos.
    - Recarga el arreglo de los egresos con la función cargarEgresos() para que se dibujen solo los elementos restantes del arreglo egresos.

Una vez que ya estamos mostrando los datos dinámicamente en el cabecero, en la lista de ingresos y en la lista de los egresos, es momento de darle funcionalidad al formulario.

- En el archivo html:
    - Agrega el evento onclick al botón con la clase agregar-btn.
    - Manda llamar a la función **agregarDato()**.
    - Agrega el evento **onsubmit**, asignándole los valores “return false” al formulario con id **forma.**
- En el archivo app.js:
    - Declara la función flecha **agregarDato**.
    - En la variable forma, recupera el formulario con id **forma**.
    - Recupera el tipo de **select** que seleccionó el usuario, asignando a la variable tipo el contenido del **select** con id tipo del formulario **forma**.
    - Recupera la **descripción** del formulario forma y asígnalo a la variable descripción.
    - Recupera el valor del formulario forma y asígnalo a la variable valor.
    - Valida que el valor de descripcion y de valor no estén vacíos. Si la condición se cumple:
        - Valida si el valor de tipo es igual a ingreso:
        - Agrega el valor de la **descripcion** y el **valor** al arreglo **ingresos** a través del método push e invocando a un nuevo objeto del tipo **Ingreso.**
        - Invoca a la función **cargarCabecero y CargarIngreso.**

Ahora tu aplicación esta lista para que puedas probar toda su funcionalidad.

Avance 4

Consideraciones

Maestro:

- En esta parte del proyecto se utilizan eventos y manejo del DOM, por tanto, asegúrate de que el aprendedor comprenda ambos conceptos y la diferencia entre ellos. Invítalo a que lea la bibliografía adicional.

Alumno:

- Asegúrate de poder manipular el DOM y hacer el backtesting en la consola antes de presentarlo en el HTML.

a) Importante: En la sesión sincrónica del tema 8 se deberá establecer que el aprendedor debe tener listo su proyecto para entregarlo en la semana 5. El proyecto se entregará por los medios indicados por el instructor.

Criterios de evaluación

Rúbrica de evaluación de proyecto

	Nivel de desempeño			
Criterios de evaluación	**Altamente competente 100% - 86%**	**Competente 85% - 70%**	**Aún sin desarrollar la competencia 69% - 0%**	**%**
1. Desarrolla las secciones de la página web solicitadas.	**10 - 8** Desarrolla correctamente el contenido html, incluyendo bootstrap y css, conforme a la solicitud.	**7 - 5** Desarrolla el contenido html, incluyendo solamente bootstrap o el css, conforme a la solicitud.	**4 - 0** No desarrolla el contenido html solicitado.	10
2. Crea las funciones que van a cargar dinámicamente los datos en el html (las que realizarán los cálculos de ingresos y egresos).	**15 - 14** Desarrolla completamente las funciones y los objetos necesarios para cargar dinámicamente los datos en el html.	**13 - 11** Desarrolla parcialmente las funciones y los objetos necesarios para cargar dinámicamente los datos en el html.	**10 - 0** No desarrolla las funciones ni los objetos necesarios para cargar dinámicamente los datos en el html.	15
3. Crea las funciones que dan formato a los números.	**15 - 14** Desarrolla satisfactoriamente las funciones y los objetos necesarios para dar formato a los datos generados.	**13 - 11** Desarrolla la mayoría de las funciones y los objetos necesarios para dar formato a los datos generados.	**10 - 0** No desarrolla todas las funciones ni los objetos necesarios para dar formato a los datos generados.	15

Criterios de evaluación

	Nivel de desempeño			
Criterios de evaluación	**Altamente competente 100% - 86%**	**Competente 85% - 70%**	**Aún sin desarrollar la competencia 69% - 0%**	**%**
4. Crea las clases de la aplicación.	**20 - 18** Desarrolla satisfactoriamente las clases necesarias para el funcionamiento de la aplicación.	**17 - 15** Desarrolla parcialmente las clases necesarias para el funcionamiento de la aplicación.	**14 - 0** No desarrolla las clases necesarias para el funcionamiento de la aplicación.	20
5. Presenta de manera dinámica los datos generados.	**20** Presenta correctamente los datos generados dinámicamente manipulando DOM.	**17 - 15** Presenta parcialmente los datos generados dinámicamente manipulando DOM.	**14 - 0** No presenta los datos generados dinámicamente manipulando DOM.	20
6. Sube a GitHub el código de su aplicación.	**10 - 8** Realiza correctamente el versionamiento de su proyecto.	**7 - 5** Realiza parcialmente el versionamiento de su proyecto.	**4 - 0** No versiona su proyecto en GitHub.	10
7. Publica su aplicación realizada en Netlify.	**10 - 8** Realiza correctamente el versionamiento de su proyecto.	**7 - 5** Realiza parcialmente el versionamiento de su proyecto.	**4 - 0** No versiona su proyecto en GitHub.	10
			Total	100%