Introducción a la Programación
Técnico en Desarrollo Full Stack

Tarea 07 - **Resuelta**

Estimado/a participante,

En esta tarea, deberás resolver una serie de problemas utilizando el lenguaje de programación JavaScript. Para completarla, sigue los siguientes pasos:

1. Crea un repositorio en GitHub donde subirás tu trabajo. Asegúrate de que el repositorio sea público para que podamos revisarlo.
2. Dentro del repositorio, crea un archivo para cada problema que se presenta a continuación. Cada archivo debe tener extensión `.js` y debe contener la solución al problema correspondiente.
3. Además de los archivos `.js`, crea un archivo PDF que contenga todas las soluciones a los problemas. Este archivo debe estar bien estructurado y presentado de manera clara.
4. Una vez que hayas completado todos los pasos, comparte el enlace al repositorio de GitHub en la casilla correspondiente del GES para que podamos revisar tu trabajo.
5. ¡Buena suerte!

Uso de .call( )

Problema I (15 puntos)

Cambiar el contexto de `this`

Escribe una función llamada **`saludar`** que tome un parámetro **`nombre`** y devuelva un saludo personalizado. Luego, crea un objeto **`persona`** con una propiedad **`nombre`**. Utiliza el método **`call()`** para llamar a la función **`saludar`** con el contexto del objeto **`persona`**. Si lo deseas apoyate en la siguiente estructura:

```
// Función saludar
function xxxxx(xxxxxx) {
return `Hola, ${xxxxx}!`;
}

// Objeto persona
const xxxxx = {
xxxxxx: 'Juan'
};

// Llamar a la función saludar con el contexto de persona usando call()
const saludo = xxxxxx.call(persona, persona.nombre);

// Imprimir el saludo console.log(saludo); // Debe
imprimir: Hola, Juan!
```

```
// Función saludar
function saludar(nombre) {
    return `Hola, ${nombre}!`;
  }

  // Objeto persona
  const persona = {
    nombre: 'Juan'
  };
```

```
// Llamar a la función saludar con el contexto de persona usando call()
const saludo = saludar.call(persona, persona.nombre);

// Imprimir el saludo
console.log(saludo); // Debe imprimir: Hola, Juan!
```

Problema II (15 puntos)

Cambiar el contexto de `this` en un método de objeto

Crea un objeto `auto` con una propiedad `marca` y un método `mostrarMarca` que devuelva un mensaje con la marca del auto. Luego, crea otro objeto `moto` con una propiedad `marca`. Utiliza el método `call()` para llamar al método `mostrarMarca` del objeto `auto` con el contexto del objeto `moto`.

```
// Objeto auto
const xxxx = {
    xxxxx: 'Toyota',
    xxxxxxxxxxxx: function() {
        return `La marca del auto es ${this.marca}.`;
    }
};

// Objeto moto
const xxxx = {
    marca: 'Honda'
};

// Llamar al método mostrarMarca del objeto auto con el contexto de
//moto usando call() const mensaje =
auto.xxxxxxxxxxxx.xxxx(moto);

// Imprimir el mensaje console.log(mensaje); // Debe imprimir: La
marca de la moto es Honda.
```

```
// Problema II
// Objeto auto
const auto = {
    marca: 'Toyota',
    mostrarMarca: function() {
        return `La marca del auto es ${this.marca}.`;
    }
};

// Objeto moto
const moto = {
    marca: 'Honda'
};

// Llamar al método mostrarMarca del objeto auto con el contexto de moto
usando call()
const mensaje = auto.mostrarMarca.call(moto);

// Imprimir el mensaje
console.log(mensaje); // Debe imprimir: La marca de la moto es Honda.
```

Problema III (10 Puntos)

Crea dos objetos, `persona1` y `persona2`, cada uno con una propiedad `nombre`. Define una función `saludar` que devuelva un mensaje de saludo utilizando la propiedad `nombre`. Usa `call()` para invocar la función `saludar` con el contexto de `persona2`.

```
// Objeto persona1
const xxxxxxxx = {
xxxxxx: 'Juan'
};

// Objeto persona2
const xxxxxxxx= {
   xxxxxx: 'María'
};

// Función saludar
function xxxxxxx() {
```

```
    return `Hola, mi nombre es ${this.nombre}.`;
}

// Usar call() para invocar saludar con el contexto de persona2
const saludo = saludar.xxxx(persona2);

// Imprimir el saludo console.log(saludo); // Debe imprimir: Hola,
mi nombre es María.
```

```
// Problema III
// Objeto persona1
const persona1 = {
    nombre: 'Juan'
};

// Objeto persona2
const persona2 = {
    nombre: 'María'
};

// Función saludar
function saludar() {
    return `Hola, mi nombre es ${this.nombre}.`;
}

// Usar call() para invocar saludar con el contexto de persona2
const saludoPersona2 = saludar.call(persona2);

// Imprimir el saludo
console.log(saludoPersona2); // Debe imprimir: Hola, mi nombre es María.
```

Problema IV (10 Puntos)

Crea un objeto `rectangulo` con propiedades `ancho` y `alto`, y un método `area` que calcule el área del rectángulo. Luego, crea un objeto `cuadrado` con una propiedad `lado`. Usa `call()` para invocar el método `area` del objeto `rectangulo` con el contexto del objeto `cuadrado`.

```
// Objeto rectangulo const
xxxxxxxxxx = { xxxxx: 0, xxxx: 0,
```

```
xxxx: function() { return
this.ancho * this.alto;
    }
};

// Objeto cuadrado
const xxxxxxxx = {
    xxxx: 5
};

// Usar call() para invocar area con el contexto de cuadrado const
areaCuadrado = rectangulo.area.call({ ancho: cuadrado.lado, alto:
cuadrado.lado });

// Imprimir el área del cuadrado
console.log(areaCuadrado); // Debe imprimir: 25
```

```
// Problema IV
// Objeto rectangulo
const rectangulo = {
    ancho: 0,
    alto: 0,
    area: function() {
        return this.ancho * this.alto;
    }
};

// Objeto cuadrado
const cuadrado = {
    lado: 5
};

// Usar call() para invocar area con el contexto de cuadrado
const areaCuadrado = rectangulo.area.call({ ancho: cuadrado.lado, alto:
cuadrado.lado });

// Imprimir el área del cuadrado
console.log(areaCuadrado); // Debe imprimir: 25
```

Uso de .apply( )

Problema V (15 puntos)

Crea dos objetos, `persona1` y `persona2`, cada uno con una propiedad `nombre`. Define una función `presentar` que devuelva un mensaje de presentación utilizando la propiedad `nombre`. Usa `apply()` para invocar la función `presentar` con el contexto de `persona2`.

```
// Objeto persona1
const xxxxxxxx = {
nombre: 'Carlos'
};

// Objeto persona2
const xxxxxxxx = {
    nombre: 'Ana'
};

// Función presentar function
xxxxxxxxx() { return `Hola, soy
${this.nombre}.`;
}

// Usar apply() para invocar presentar con el contexto de persona2
const presentacion = presentar.apply(xxxxxxxx);

// Imprimir la presentación console.log(presentacion); //
Debe imprimir: Hola, soy Ana.
```

```
// Problema V
// Objeto persona1
const persona1V = {
    nombre: 'Carlos'
};

// Objeto persona2
const persona2V = {
    nombre: 'Ana'
```

```
};

// Función presentar
function presentar() {
    return `Hola, soy ${this.nombre}.`;
}

// Usar apply() para invocar presentar con el contexto de persona2
const presentacion = presentar.apply(persona2V);

// Imprimir la presentación
console.log(presentacion); // Debe imprimir: Hola, soy Ana.
```

Problema VI (15 puntos) Expansión de Objeto con Array

Crea un objeto `libro` con propiedades `titulo` y `autor`. Define una función `agregarCapitulos` que tome un array de capítulos y los agregue como una nueva propiedad `capitulos` al objeto `libro`. Usa `apply()` para invocar la función `agregarCapitulos` con el contexto del objeto `libro` y un array de capítulos.

```
// Objeto libro
const xxxxx = {
   xxxxxx: 'El Quijote',
    xxxxx: 'Miguel de Cervantes'
};

// Función agregarCapitulos function
xxxxxxxxxxxxxxxx(capitulos) {
this.capitulos = capitulos;
}

// Array de capítulos const capitulos = ['Capítulo 1: En un lugar de la
Mancha', 'Capítulo 2: De los molinos de viento'];

// Usar apply() para invocar agregarCapitulos con el contexto de libro y el
array de capítulos agregarCapitulos.apply(libro, [xxxxxxxxx]);

// Imprimir el objeto libro con los capítulos agregados
console.log(libro); // Debe imprimir:
```

```
// {
//     titulo: 'El Quijote',
//     autor: 'Miguel de Cervantes',
// capitulos: ['Capítulo 1: En un lugar de la Mancha', 'Capítulo 2: De los molinos de viento']
// }
```

```
// Objeto libro
const libro = {
    titulo: 'El Quijote',
    autor: 'Miguel de Cervantes'
  };

  // Función agregarCapitulos
  function agregarCapitulos(capitulos) {
    this.capitulos = capitulos;
  }

  // Array de capítulos
  const capitulos = ['Capítulo 1: En un lugar de la Mancha', 'Capítulo 2: De los molinos de viento'];

  // Usar apply() para invocar agregarCapitulos con el contexto de libro y el array de capítulos
  agregarCapitulos.apply(libro, [capitulos]);

  // Imprimir el objeto libro con los capítulos agregados
  console.log(libro);
  // Debe imprimir:
  // {
  //   titulo: 'El Quijote',
  //   autor: 'Miguel de Cervantes',
  //   capitulos: ['Capítulo 1: En un lugar de la Mancha', 'Capítulo 2: De los molinos de viento']
  // }
```

Problema VII (20 puntos) Actualización de Saldo en Cuenta Bancaria

Crea un objeto `cuentaBancaria` con propiedades `titular` y `saldo`. Define una función `actualizarSaldo` que tome un monto y lo sume al saldo actual de la cuenta. Usa `apply()` para invocar la función

`actualizarSaldo` con el contexto del objeto `cuentaBancaria` y un array que contenga el monto a agregar.

```
// Objeto cuentaBancaria
const xxxxxxxxxxxxxx = {
    xxxxxxx: 'Juan Pérez',
    xxxxx: 1000
};

// Función actualizarSaldo function
xxxxxxxxxxxxxxx(monto) {
    this.saldo += monto;
}

// Monto a agregar
const monto = 500;

// Usar apply() para invocar actualizarSaldo con el contexto de cuentaBancaria y el monto a agregar
actualizarSaldo.xxxxx(cuentaBancaria, [xxxxx]);

// Imprimir el objeto cuentaBancaria con el saldo actualizado
console.log(cuentaBancaria); // Debe imprimir:
// {
//     titular: 'Juan Pérez',
//     saldo: 1500
// }
```

```
// Objeto cuentaBancaria
const cuentaBancaria = {
    titular: 'Juan Pérez',
    saldo: 1000
  };

  // Función actualizarSaldo
  function actualizarSaldo(monto) {
    this.saldo += monto;
  }
```

```
  // Monto a agregar
  const monto = 500;

  // Usar apply() para invocar actualizarSaldo con el contexto de
cuentaBancaria y el monto a agregar
  actualizarSaldo.apply(cuentaBancaria, [monto]);

  // Imprimir el objeto cuentaBancaria con el saldo actualizado
  console.log(cuentaBancaria);
  // Debe imprimir:
  // {
  //   titular: 'Juan Pérez',
  //   saldo: 1500
  // }
```