Universidad Galileo
Gerardo Francisco Gutiérrez Valenzuela
Introducción a la programación

Tarea 7 – Ejercicios Call/Apply

Byron Fernando Cardona Sánchez
Carné: 24011342
18 de Junio del 2024

Tarea 07 CallApply

Problema 1

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

const persona = {
    nombre: 'Fernando'
};

const saludo = saludar.call(persona, persona.nombre);

console.log(saludo);
```

1. **Definición de la función saludar:**
    - Esta es una función llamada saludar que toma un parámetro nombre.
    - La función devuelve una cadena de texto que incluye el valor del parámetro nombre, utilizando plantillas de cadena para insertar el valor dentro del saludo.
2. **Creación del objeto persona:**
    - Se crea un objeto llamado persona que tiene una propiedad nombre con el valor 'Fernando'.
3. **Llamada a la función saludar con el contexto del objeto persona:**
    - Se utiliza el método call() para invocar la función saludar.
    - call() permite llamar a una función con un valor específico para this y argumentos adicionales.
    - En este código, persona es el valor de this que se pasa a call().
    - El segundo argumento es el valor que se pasa como nombre a la función saludar, en este caso persona.nombre que es Fernando.
4. **Mostrar el saludo:**
    - Se imprime en la consola el valor de la variable saludo, que contiene el resultado de la llamada a la función saludar.
    - La salida será la cadena Hola, Fernando!, porque persona.nombre es Fernando y esto es lo que se pasa a la función saludar.

Problema 2

```
const auto = {
    marca: 'Mazda',
    mostrarMarca: function() {
        return `La marca del auto es ${this.marca}.`;
    }
};

const moto = {
    marca: 'BMW'
};

const marcaMoto = auto.mostrarMarca.call(moto);

console.log(marcaMoto);
```

1. **Creación del objeto auto con una propiedad marca y un método mostrarMarca:**
    - Se define un objeto llamado auto con una propiedad marca que tiene el valor 'Mazda'.
    - El objeto también tiene un método mostrarMarca que devuelve una cadena de texto que incluye la marca del auto. Utiliza this.marca para acceder a la propiedad marca del objeto en el contexto en el que se llama la función.
2. **Creación del objeto moto con una propiedad marca:**
    - Se define un objeto llamado moto con una propiedad marca que tiene el valor 'BMW'.
3. **Llamada al método mostrarMarca del objeto auto con el contexto del objeto moto:**
    - Se utiliza el método call() para invocar el método mostrarMarca del objeto auto, pero con el contexto de this del objeto moto.
    - Esto significa que dentro de la función mostrarMarca, this se referirá a moto, por lo que this.marca será 'BMW'.
4. **Mostrar el resultado:**
    - Se imprime en la consola el valor de la variable marcaMoto, que contiene el resultado de la llamada a mostrarMarca con el contexto de moto.
    - La salida será La marca de la moto es BMW. porque el método mostrarMarca accede a this.marca, que en el contexto de moto es 'BMW'.

Problema 3

```
const persona1 = {
    nombre: 'Josue'
};

const persona2 = {
    nombre: 'Yohana'
};

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

const saludo = saludar.call(persona2);

console.log(saludo);
```

1. **Creación de los objetos persona1 y persona2 con una propiedad nombre:**
    - Se definen dos objetos: persona1 con la propiedad nombre que tiene el valor 'Josue' y persona2 con la propiedad nombre que tiene el valor 'Yohana'.
2. **Definición de la función saludar:**
    - Se define una función llamada saludar que devuelve un mensaje de saludo utilizando this.nombre.
    - this se referirá al objeto en el contexto del cual se llama la función.
3. **Invocación de la función saludar con el contexto de persona2:**
    - Se utiliza el método call() para invocar la función saludar con persona2 como el contexto.
    - Esto significa que dentro de la función saludar, this se referirá a persona2, por lo que this.nombre será 'Yohana'.
4. **Mostrar el saludo:**
    - Se imprime en la consola el valor de la variable saludo, que contiene el resultado de la llamada a la función saludar con el contexto de persona2.
    - La salida será Hola, mi nombre es Yohana! porque this.nombre en el contexto de persona2 es 'Yohana'.

Problema 4

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

const cuadrado = {
    lado: 6
};

const areaCuadrado = rectangulo.area.call({ ancho: cuadrado.lado, alto: cuadrad
o.lado });


console.log(areaCuadrado);
```

1. **Creación del objeto rectangulo con propiedades ancho y alto, y un método area:**
    - Se define un objeto llamado rectangulo con las propiedades ancho y alto, ambas inicializadas en 0.
    - El objeto también tiene un método area que calcula el área multiplicando this.ancho por this.alto.
2. **Creación del objeto cuadrado con una propiedad lado:**
    - Se define un objeto llamado cuadrado con una propiedad lado que tiene el valor 6.
3. **Ajuste de las propiedades ancho y alto del objeto rectangulo con el contexto del objeto cuadrado:**
    - Se utiliza el método call() para invocar el método area del objeto rectangulo.
    - Se crea un nuevo objeto con las propiedades ancho y alto igual a cuadrado.lado y se pasa como el contexto (this) al método area.
    - De esta manera, dentro del método area, this.ancho y this.alto serán ambos 6, simulando un cuadrado.
4. **Mostrar el resultado:**
    - Se imprime en la consola el valor de areaCuadrado, que contiene el resultado de la llamada al método area con el contexto ajustado.
    - La salida será 36 porque el área de un cuadrado con lado 6 es 6 * 6 = 36.

Problema 5

```
let persona1 = {
  nombre: 'Juan'
};

let persona2 = {
  nombre: 'Olivia'
};

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

let mensajePersona2 = presentar.apply(persona2);

console.log(mensajePersona2);
```

1. **Objetos persona1 y persona2**:
   - Creamos dos objetos simples (persona1 y persona2) con una propiedad nombrecada uno.
2. **Función presentar**:
   - Definimos una función llamada presentar que utiliza this.nombre para acceder al nombre del objeto pasado como contexto.
   - Dentro de la función, this se refiere al objeto que será pasado como contexto cuando usemos apply().
3. **Uso de apply**:
   - Llamamos a presentar.apply(persona2), lo que significa que estamos ejecutando la función presentar pero utilizando persona2 como el contexto (this).
   - Esto hace que dentro de la función presentar, this.nombre se refiera al nombre almacenado en persona2, que es "Olivia".
4. **Resultado**:
   - El valor devuelto por presentar.apply(persona2) es almacenado en mensajePersona2, que en este caso será "Hola, soy Olivia.".
5. **Impresión del resultado**:
   - Finalmente, imprimimos mensajePersona2 utilizando console.log() para mostrar el mensaje de presentación generado.

Problema 6

```
let libro = {
  titulo: 'Crimen y Castigo',
  autor: 'Raskólnikov'
};

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

let arrayCapitulos = [
  'Capítulo 1: Inicio en San Petersburgo',
   'Capítulo 2: Evitando Contacto Social'
];

agregarCapitulos.apply(libro, [arrayCapitulos]);

console.log(libro);
```

1. **Objeto libro**:
   - Creamos un objeto libro con las propiedades titulo y autor. En este código, el título es " Crimen y Castigo" y el autor es " Raskólnikov".
2. **Función agregarCapitulos**:
   - Definimos una función llamada agregarCapitulos que toma un parámetro capitulos y asigna ese array como una nueva propiedad capitulos al objeto que llama la función (this).
3. **Array de capítulos**:
   - Creamos un array arrayCapitulos que contiene varios strings representando los nombres de los capítulos del libro.
4. **Uso de apply**:
   - Llamamos a agregarCapitulos.apply(libro, [arrayCapitulos]). Esto significa que estamos ejecutando la función agregarCapitulos, pero usando libro como el contexto (this). El segundo argumento de apply es un array que contiene los argumentos que se pasarán a la función agregarCapitulos, en este caso, el array de capítulos arrayCapitulos.
5. **Resultado**:
   - Después de llamar agregarCapitulos.apply(libro, [arrayCapitulos]), el objeto libro ahora tiene una nueva propiedad capitulos que contiene el array de capítulos especificado.
6. **Impresión del resultado**:
   - Finalmente, imprimimos el objeto libro completo con console.log(libro), lo que mostrará todas sus propiedades, incluyendo los capítulos recién agregados.

Problema 7

```
let cuentaBancaria = {
  titular: 'Byron Cardona',
  saldo: 1000
};

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

let arrayMonto = [500];


actualizarSaldo.apply(cuentaBancaria, arrayMonto);

console.log(`El saldo actual de la cuenta de ${cuentaBancaria.titular} es:
${cuentaBancaria.saldo}`);
```

1. **Objeto cuentaBancaria**:
   - Creamos un objeto cuentaBancaria con las propiedades titular y saldo inicializado en 1000.
2. **Función actualizarSaldo**:
   - Definimos una función llamada actualizarSaldo que toma un parámetro monto y suma ese monto al saldo actual de la cuenta (this.saldo += monto).
3. **Array arrayMonto**:
   - Creamos un array arrayMonto que contiene el monto que queremos sumar al saldo de la cuenta. En este código, [500] indica que queremos sumar 500 al saldo actual.
4. **Uso de apply**:
   - Llamamos a actualizarSaldo.apply(cuentaBancaria, arrayMonto). Esto significa que estamos ejecutando la función actualizarSaldo, pero usando cuentaBancaria como el contexto (this). El segundo argumento de apply es un array que contiene los argumentos que se pasarán a la función actualizarSaldo, en este caso, el array arrayMonto.
5. **Resultado**:
   - Después de llamar actualizarSaldo.apply(cuentaBancaria, arrayMonto), el saldo en cuentaBancaria se incrementará en 500.
6. **Impresión del resultado**:
   - Finalmente, imprimimos el saldo actualizado de la cuenta utilizando console.log(). Esto mostrará el mensaje "El saldo actual de la cuenta de Byron Cardona es: 1500", ya que hemos sumado 500 al saldo inicial de 1000.