Problema 1

Función `saludar`:

La función `saludar` toma un parámetro `nombre` y devuelve un mensaje de saludo personalizado concatenando el texto "Hola, " con el valor del parámetro `nombre`.

Objeto `persona`:

El objeto `persona` se define con una propiedad `nombre` cuyo valor es "Juan".

Llamada a la función `saludar` con `call()`:

La línea `const saludo = saludar.call(persona, persona.nombre);` utiliza el método `call()` para invocar la función `saludar` con el contexto del objeto `persona`.

- `persona`: Se pasa el objeto `persona` como primer argumento para establecer el contexto de ejecución de la función. Esto significa que la palabra clave `this` dentro de la función `saludar` se referirá al objeto `persona` cuando se llame, utilizando su propiedad `nombre`.
- `persona.nombre`: Se pasa el valor de la propiedad `nombre` del objeto `persona` ("Juan") como segundo argumento para proporcionar el nombre que se usará en el mensaje de saludo.

Impresión del saludo:

La línea `console.log(saludo);` imprime el valor de la variable `saludo` en la consola. La variable `saludo` contiene el mensaje de saludo personalizado generado por la función `saludar` con el nombre de la persona ("Juan").

Resultado esperado:

Al ejecutar este código, se imprimirá el siguiente mensaje en la consola:

Hola, Juan!

Problema 2

Objetos `auto` y `moto`:

- Se definen dos objetos: `auto` y `moto`.
    - El objeto `auto` tiene una propiedad `marca` con el valor "Toyota" y un método `mostrarMarca` que devuelve un mensaje indicando la marca del auto utilizando la palabra clave `this`.
    - El objeto `moto` tiene una propiedad `marca` con el valor "Honda".

Llamada al método `mostrarMarca` con `call()`:

- La línea `const mensaje = auto.mostrarMarca.call(moto);` utiliza el método `call()` para invocar el método `mostrarMarca` del objeto `auto` pero con el contexto del objeto `moto`.
    - `moto`: Se pasa el objeto `moto` como primer argumento para establecer el contexto de ejecución del método `mostrarMarca`. Esto significa que la palabra clave `this` dentro del método `mostrarMarca` se referirá al objeto `moto` cuando se llame, utilizando su propiedad `marca`.
    - No se pasan argumentos adicionales al método `mostrarMarca` ya que no los requiere.

Impresión del mensaje:

- La línea `console.log(mensaje);` imprime el valor de la variable `mensaje` en la consola. La variable `mensaje` contiene el mensaje de la marca generado por el método `mostrarMarca` utilizando la marca de la moto ("Honda").

Resultado esperado:

Al ejecutar este código, se imprimirá el siguiente mensaje en la consola:

La marca de la moto es Honda.

Problema 3

Objetos `persona1` y `persona2`:

- Se definen dos objetos: `persona1` y `persona2`.
    - El objeto `persona1` tiene una propiedad `nombre` con el valor "Carlos".
    - El objeto `persona2` tiene una propiedad `nombre` con el valor "Ana".

Función `presentar`:

- La función `presentar` no tiene argumentos y devuelve un mensaje de saludo personalizado concatenando el texto "Hola, soy " con el valor de la propiedad `nombre` del objeto en el que se invoca.

Llamada a la función `presentar` con `apply()`:

- La línea `const presentacion = presentar.apply(persona2);` utiliza el método `apply()` para invocar la función `presentar` con el contexto del objeto `persona2`.
    - `persona2`: Se pasa el objeto `persona2` como primer argumento para establecer el contexto de ejecución de la función `presentar`. Esto significa que la palabra clave `this` dentro de la función `presentar` se referirá al objeto `persona2` cuando se llame, utilizando su propiedad `nombre`.
    - No se pasan argumentos adicionales a la función `presentar` ya que no los requiere.

Impresión de la presentación:

- La línea `console.log(presentacion);` imprime el valor de la variable `presentacion` en la consola. La variable `presentacion` contiene el mensaje de saludo generado por la función `presentar` utilizando el nombre de la persona ("Ana").

Resultado esperado:

Al ejecutar este código, se imprimirá el siguiente mensaje en la consola:

Hola, soy Ana.

Problema 4

Objetos `rectángulo` y `cuadrado`:

- Se definen dos objetos: `rectángulo` y `cuadrado`.
    - El objeto `rectángulo` tiene propiedades `ancho` y `alto` inicializadas en 0, y un método `area()` que calcula y devuelve el área del rectángulo multiplicando el `ancho` por el `alto`.
    - El objeto `cuadrado` tiene una propiedad `lado` con el valor 5.

Llamada al método `area` con `call()`:

- La línea `const areaCuadrado = rectangulo.area.call({ ancho: cuadrado.lado, alto: cuadrado.lado });` utiliza el método `call()` para invocar el método `area` del objeto `rectángulo` con un contexto personalizado.
    - `{ ancho: cuadrado.lado, alto: cuadrado.lado }`: Se pasa un objeto anónimo como primer argumento para establecer el contexto de ejecución del método `area`. Este objeto tiene propiedades `ancho` y `alto` con valores tomados del `lado` del objeto `cuadrado`. Esto significa que la palabra clave `this` dentro del método `area` se referirá a este objeto anónimo cuando se llame, utilizando sus propiedades `ancho` y `alto`.
    - No se pasan argumentos adicionales al método `area` ya que no los requiere.

Impresión del área del cuadrado:

- La línea `console.log(areaCuadrado);` imprime el valor de la variable `areaCuadrado` en la consola. La variable `areaCuadrado` contiene el área calculada por el método `area` utilizando las dimensiones del cuadrado (lado 5), que es 25.

Resultado esperado:

Al ejecutar este código, se imprimirá el siguiente mensaje en la consola:

25

Problema 5

Objetos `persona1` y `persona2`:

- Se definen dos objetos: `persona1` y `persona2`.
    - El objeto `persona1` tiene una propiedad `nombre` con el valor "Carlos".
    - El objeto `persona2` tiene una propiedad `nombre` con el valor "Ana".

Función `presentar`:

- La función `presentar` no tiene argumentos y devuelve un mensaje de saludo personalizado concatenando el texto "Hola, soy " con el valor de la propiedad `nombre` del objeto en el que se invoca.

Llamada a la función `presentar` con `apply()`:

- La línea `const presentacion = presentar.apply(persona2);` utiliza el método `apply()` para invocar la función `presentar` con el contexto del objeto `persona2`.
    - `persona2`: Se pasa el objeto `persona2` como primer argumento para establecer el contexto de ejecución de la función `presentar`. Esto significa que la palabra clave `this` dentro de la función `presentar` se referirá al objeto `persona2` cuando se llame, utilizando su propiedad `nombre`.
    - No se pasan argumentos adicionales a la función `presentar` ya que no los requiere.

Impresión de la presentación:

- La línea `console.log(presentacion);` imprime el valor de la variable `presentacion` en la consola. La variable `presentacion` contiene el mensaje de saludo generado por la función `presentar` utilizando el nombre de la persona ("Ana").

Resultado esperado:

Al ejecutar este código, se imprimirá el siguiente mensaje en la consola:

Hola, soy Ana.

Problema 6

Objeto `libro`:

- Se define un objeto llamado `libro` con dos propiedades:
  - `titulo`: con el valor "El Quijote".
  - `autor`: con el valor "Miguel de Cervantes".

Función `agregarCapitulos`:

- Se define una función llamada `agregarCapitulos` que toma un parámetro `capitulos`.
- La función `agregarCapitulos` asigna el valor del parámetro `capitulos` a la propiedad `capitulos` del objeto en el que se invoca la función.

Array de capítulos:

- Se define un array llamado `capitulos` que contiene dos elementos, los títulos de los dos primeros capítulos del libro.

Llamada a la función `agregarCapitulos` con `apply()`:

- La línea `agregarCapitulos.apply(libro, [capitulos]);` utiliza el método `apply()` para invocar la función `agregarCapitulos` con el contexto del objeto `libro`.
  - `libro`: Se pasa el objeto `libro` como primer argumento para establecer el contexto de ejecución de la función `agregarCapitulos`. Esto significa que la palabra clave `this` dentro de la función `agregarCapitulos` se referirá al objeto `libro` cuando se llame, utilizando su propiedad `capitulos`.
  - `[capitulos]`: Se pasa el array `capitulos` como segundo argumento para proporcionar el valor que se asignará a la propiedad `capitulos` del objeto `libro`.

Impresión del objeto libro con los capítulos agregados:

- La línea `console.log(libro);` imprime el objeto `libro` en la consola. El objeto `libro` ahora tiene una propiedad `capitulos` que contiene el array de títulos de capítulos.

Problema 7

Objetos y función:

- `cuentaBancaria`: Se define un objeto llamado `cuentaBancaria` con dos propiedades:
    - `titular`: Con el valor "Juan Pérez", que indica el nombre del titular de la cuenta.
    - `saldo`: Con el valor 1000, que representa el saldo inicial de la cuenta en alguna unidad monetaria (no especificada en el código).
- `actualizarSaldo`: Se define una función llamada `actualizarSaldo` que toma un parámetro `monto`. Esta función actualiza el saldo de la cuenta bancaria sumando el valor del parámetro `monto` al saldo actual (`this.saldo`).

Actualizando el saldo:

- `monto`: Se define una variable `monto` con el valor 500, que representa la cantidad que se quiere añadir al saldo de la cuenta.
- `actualizarSaldo.apply(cuentaBancaria, [monto])`: Esta línea utiliza el método `apply()` para invocar la función `actualizarSaldo`.
    - `cuentaBancaria`: Se pasa el objeto `cuentaBancaria` como primer argumento para establecer el contexto de ejecución de la función `actualizarSaldo`. Esto significa que la palabra clave `this` dentro de la función se referirá a `cuentaBancaria`.
    - `[monto]`: Se pasa el array `[monto]` como segundo argumento. En este caso, el array solo tiene un elemento que es el valor a añadir al saldo. La función `actualizarSaldo` solo necesita un argumento, por lo que solo se incluye uno en el array.

Imprimiendo el resultado:

- `console.log(cuentaBancaria);`: Esta línea imprime el objeto `cuentaBancaria` en la consola. Después de ejecutar `actualizarSaldo`, el saldo de la cuenta se habrá actualizado sumando el valor de `monto`.