****

**Preguntas sobre JavaScript**

**Descripción**

**Punto 1.**

**¿Qué es un *closure*? Proporcione un ejemplo.**

JavaScript tiene la propiedad de anidar funciones dentro de las funciones, de recibir funciones como parámetro. En este caso, cuando la función interna (d) contiene una referencia a una variable declarada en el scope de una función superior, esta se convierte en una closure.

Un ejemplo de este es el siguiente, la función incrementByAFactor es una función que tiene una variable factor dentro, y adicionalmente hay una función dentro que hace uso de esta variable:

function incrementByAFactor() {

  let factor = 2;

  return function () {

    factor = factor\*2;

    return factor;

  };

}

El código complete está en el link de github referenciado en la parte de abajo.

**Punto 2.**

**¿Qué es *hoisting*? Proporcione un ejemplo.**

El hoisting permite que siempre que se define una función, el intérprete de JavaScript eleve la función al inicio de la cola de ejecución, lo que permite que se pueda invocar una función antes de su declaración, sin que esto genere errores.

Un ejemplo es la siguiente funcion saludar, que esta declarada después de que se llama:

saludar();

saludar("Hola antes de la funcion")

function saludar(saludo) {

  return saludo;

}

El código complete está en el link de github referenciado en la parte de abajo.

**Punto 3.**

**Explique los contextos en los cuales se usa el objeto *this.* Proporcione un ejemplo en código para cada contexto.**

A continuación, se presentan los diferentes casos en donde se puede usar el objeto this:

* Si this está siendo invocado dentro de un método, entonces hace referencia al objeto que contiene el método donde se invoca
* const person = {
* place: "Colombia",
* sayMyCountry: function () {
* console.log(`Hi, I am from ${this.place}`);
* },
* };
* person.sayMyCountry();
* invocación directa: cuando se pone solito, en el caso de los navegadores el objeto será window, en la consola será el objeto global
* function myCoutry() {
* console.log("I am from ", this);
* }
* myCoutry();
* Invocación de método: Ocurre cuando invocamos al método de un objeto; this corresponderá entonces a ese objeto
* const person = {
* country: "Colombia",
* sayMyCountry: function () {
* console.log(`I am from ${this.country}`);
* },
* friend: {
* country: "Poland",
* sayMyCountry: function () {
* console.log(`I am from ${this.country}`);
* },
* },
* };
* person.sayMyCountry();
* person.friend.sayMyCountry();
* Invocación indirecta: cuando quiero invocar un método sobre otro que tenga la propiedad que quiero llamar, por ejemplo, nombre, esto se hará a través de la propiedad call, es decir de forma indirecta.
* const person = {
* place: "Colombia",
* sayMyCountry: function () {
* console.log(`Hi, I am from ${this.place}`);
* },
* };
* const visitor = { place: "Poland" };
* person.sayMyCountry.call(visitor);
* También se puede utilizar para instanciar objetos, por ejemplo, asignándole cierta propiedad al objeto con this
* class RobocolMember {
* constructor(name) {
* this.name = name;
* }
* }
* const student = new RobocolMember("Laura");
* También se pueden utilizar para las arrow functions
* const person = {
* name: "Juan",
* place: "Colombia",
* sayHello: function () {
* const sayMyCountry = () => {
* console.log(`I am from ${this.place} `);
* };
* console.log(`I am ${this.name}`);
* sayMyCountry();
* },
* };
* person.sayHello();

**Punto 4.**

El máximo común divisor (*mcd*) de dos números *a*, *b* es:

*a*, si *a* y *b* son iguales, o

el mcd de *c* y *d*, donde *c* es el menor entre *a* y *b*, y *d* es la diferencia entre el mayor y el menor.

Implemente una función *arrow*, de una sola línea, recursiva, que calcule el mcd de dos números dados.

const mcd = (a, b) => a == b ? a : mcd(Math.min(a, b), Math.max(a, b) - Math.min(a, b));

console.log("mcd arrow function", mcd(50,10));

**Entregable**

Subir como respuesta a la actividad de BS un archivo que contenga las respuestas a los puntos anteriores.