Examen JS/NodeJS desarrolla.software  
Primavera 2021

* Defina los tipos de datos **primitivos** de JavaScript. Declare una variable para cada uno de ellos.

1. String
2. Number
3. Boolean
4. Null
5. Undefined

* Defina los tipos de datos **complejos** de JavaScript. Declare una variable para cada uno de ellos.

1. function
2. object

* Defina una función llamada **MostrarMensaje**. Dicha función toma de parámetro 3 valores. Un código del tipo String, un número y un texto.  
  La idea de esta función es enviar los 3 valores en el siguiente orden: **(code, number, description)** y mostrar un mensaje de la siguiente manera en la consola, pero usando los parámetros en lugar de estos 3 valores de ejemplo:

***user-not-found: 404***  
***The requested user could not be found***

**function MostrarMensaje (code,number,description){**

**var mostrarMensaje = code+number+description**

**return mostrarMensaje**

**}**

**var mostrarMensaje = MostrarMensaje("user-not-found: ","404 ","The requested user could not be found");**

**console.log(mostrarMensaje);**

**var code = "user-not-found: "**

**var number = "404 "**

**var description = "The requested user could not be found"**

**function MostrarMensaje (){**

**var mensaje = code+number+description**

**console.log(mensaje)**

**}**

**MostrarMensaje()**

* Escriba el bloque de código necesario para mostrar los siguientes resultados:
  + Se tienen tres datos: nombreUsuario, tipoUsuario y operación.  
    Dados los 3 valores, comprobar lo siguiente:
    - Si el tipo de usuario es **‘admin’**, mostrar en consola **“Permisos otorgados para: nombreUsuario”** para cualquier tipo de operación.
    - Si el tipo de usuario es **‘colaborador’**, mostrar en consola **“Permisos otorgados”** solo si el tipo de operación es **“Lectura”, “Edición” y “Mover”**. Si el tipo de operación es **“Creación”** ó **“Eliminar”**, mostrar el mensaje **“Permisos denegados para: nombreUsuario”**.
    - Si el tipo de usuario es **‘regular’** y la operación es **“Lectura”**, mostrar **“Permisos otorgados para: nombreUsuario”**, si no, mostrar **“Permisos denegados para: nombreUsuario”**.

var Usuarios = {

nombreUsuario: "Juan Andres",

tipoUsuario: "regular",

operación: "Mover"

};

if (Usuarios.tipoUsuario === "admin"){

console.log("Permisos otorgados para: "+Usuarios.nombreUsuario);

}

else if (Usuarios.tipoUsuario === "colaborador"&&(Usuarios.operación === "Lectura" ||Usuarios.operación === "Edición"||Usuarios.operación === "Mover")){

console.log("Permisos otorgados");

}

else if (Usuarios.tipoUsuario === "colaborador"&&(Usuarios.operación === "Creación" ||Usuarios.operación ==="Eliminar")){

console.log("Permisos denegados para: " +Usuarios.nombreUsuario);

}

else if (Usuarios.tipoUsuario === "regular"&&Usuarios.operación === "Lectura"){

console.log("Permisos otorgados para: " +Usuarios.nombreUsuario);

}

else if (Usuarios.tipoUsuario === "regular"&&(Usuarios.operación === "Edición" ||Usuarios.operación ==="Mover" ||Usuarios.operación ==="Creación" ||Usuarios.operación ==="Eliminar")){

console.log("Permisos denegados para: " +Usuarios.nombreUsuario);

}

* Dada una lista de números llamada **numbers**, recorrerla y mostrar en consola el número y si es par o impar. Ejemplo de salida en consola:

*Número:* ***6*** *es* ***par***  
*Número:* ***7*** *es* ***impar***

var numbers = [0,1,2,3,4,5,6,7,8,9]

for (let i = 0; i < numbers.length; i++) {

const element = numbers[i];

if(i % 2 === 0) {

console.log("Numero: "+i+" es par")

}

else {

console.log("Numero: "+i+" es impar")

}

}

* **Declare un objeto** que represente la información de una transacción bancaria.  
  Una transacción bancaria se describe por tener el **número de cuenta que paga**, el **número de cuenta de destino**, la cantidad de **dinero transferido**, el nombre del **titular de la cuenta que paga**, el nombre del **titular de la cuenta destino** y la **fecha/hora de operación**.

var transacciónBancaria = {

númeroCuentaPaga: 12345,

númeroCuentaDestino: 54321,

dineroTransferido: 5000,

titularCuentaPaga: "Andres Ontiveros",

titularCuentaDestino: "Clara Ontiveros",

fechaOperacion: "15/04/2021",

horaOperación: "5pm"

}

* Cree una **lista** **que represente compañías** de venta de dispositivos móviles. Mínimo cada compañía debe tener la siguiente información: **nombre de la compañía, país de origen, modelos principales** que vende (Mínimo 2 datos en este campo. Tip: es un array).

var compañias = [{

nombreCompañia: "Motorola",

paisOrigen: "Estados Unidos",

modelosPrincipales: [

"g30",

"g9",

"One Fusion +",

"edge +"

]

},

{

nombreCompañia: "Samsung",

paisOrigen: "Corea del Sur",

modelosPrincipales: [

"Galaxy S21 Ultra 5G",

"Galaxy S21",

"Galaxy S20 FE",

"Galaxy Z Fold2"

]

},

{

nombreCompañia: "Xiaomi",

paisOrigen: "China",

modelosPrincipales: [

"Xiaomi Mi 10T Pro",

"Xiaomi Mi Note 10 Lite",

"Xiaomi Redmi Note 9 Pro",

"Xiaomi Redmi Note 9"

]

}

]

* Dada la siguiente lista: **[ ‘A’, ‘B’, ‘C’ ]**, añada después de **‘C’** la letra **‘D’**. Lo debe añadir de las dos posibles formas de hacerlo: **Por índice** y **con la función** que permite agregar elementos al final de una lista.

var Lista = ["A","B","C"];

Lista.push("D");

console.log(Lista)

var Lista = ["A","B","C"];

Lista[3] = "D"

console.log(Lista)

* De la **lista de compañía del ejercicio 7**, tome el **último elemento** y muestre en la consola su información. **NO DEBE USAR EL ÍNDICE EXACTO**. Utilice lógica para determinar siempre el último valor de la lista. Por ejemplo, si en este momento hay 4 compañías, el último es 3, pero **NO debe poner companies[3]**, busque una forma diferente de obtener el índice 3.   
  Esta información debe venir detallada. Ejemplo:

***Nombre de la compañía***  
***País: país de origen***  
***Modelos: Modelo A, Modelo B, Modelo C, etc.***

var Last = (compañias.length) - 1;

console.log("Nombre de la compañia: "+compañias[Last].nombreCompañia);

console.log("País: "+compañias[Last].paisOrigen);

console.log("Modelos: "+compañias[Last].modelosPrincipales);

* Cree una función llamada **GenerarObjeto** que reciba de parámetro 3 datos: **nombre, apellidos y edad**.  
  La función debe **retornar un objeto** con la siguiente estructura:

***{***

***name: ----,  
 lastName: ----,  
 age: ----,***

***}***

function generarObjeto() {

var objeto = {

nombre: "Andres",

apellidos: "Ontiveros Chairez",

edad: "27 años"

}

return(objeto)

}

console.log(generarObjeto());

* Cree un **servidor** básico de **express**, con un **único endpoint** llamado **‘/hello’**. Dicho endpoint será el método **HTTP – GET** y regresará un mensaje diciendo **“Hola!”**

var express = require('express');

const app = express();

app.use(express.static('../'));

app.get("/hello",(req,res) =>{

res.send("Hola");

});

console.log('Ejecutando en el puerto: 661');

console.log('Verificar las peticiones en el endpoint: <http://localhost:661>');

app.listen(661);

* Escriba el bloque de código necesario para hacer que el archivo **utils.js** **exporte un módulo** de **NodeJS**. Dicho módulo debe exportar lo siguiente:
  + Una función llamada **“checkYear”** que recibirá de parámetro un **número** que represente un año y retornará true solo si el año es mayor o igual a 1995, y false si es menor a 1995.
* Escriba el código necesario para **crear una cookie** en el navegador del cliente que se llama **SPGG** con el valor **“desarrolla.software”**.
* Defina la diferencia entre: **query, params y body** en las peticiones **HTTP de express**.  
  Ponga un ejemplo sencillo de cada uno (O sea, un endpoint y qué datos se le pasan a esa llamada, **NO CÓDIGO JS**, solo la llamada como si se hiciera desde PostMan)
* Escriba la diferencia de los operadores de **“igualdad”** y de **“identidad”** en JS junto con un ejemplo. Hacer lo mismo con los operadores de **“no igual”** y **“no idéntico”**.

Igualdad **=**

**Ej. 5 == “5”**

**Identidad**

**Ej. 5 === 5**

**No igual**

**Ej. 5 !== “5”**

**No identico**

**Ej. 5 !=== 5**

* Escriba el código necesario para seleccionar un posible caso de un valor.  
  El valor que se pasa es un código de número y tiene los posibles valores:
  + Si el código es 15, mostrar “Error de escritura”.
  + Si el código es 22, mostrar “Archivo no encontrado”.
  + Si el código es 320, mostrar “No se puede acceder a la unidad”.
  + Si el código es 7890, mostrar “El archivo es demasiado grande para la unidad”.
  + Si el código es 871, mostrar “El archivo está corrupto”.
  + Si el código es 12, mostrar “El archivo está bloqueado”.

switch (Number) {

case 15:

console.log("Error de escritura")

break;

case 22:

console.log("Archivo no encontrado")

break;

case 320:

console.log("No se puede acceder a la unidad")

break;

case 7890:

console.log("El archivo es demasiado grande para la unidad")

break;

case 871:

console.log("mostrar “El archivo está corrupto")

break;

case 12:

console.log("El archivo está bloqueado")

break;

}

* Escriba el código para un **ciclo** que vaya **sumando de 2 en 2** a la variable **“contador”**. Este ciclo termina cuando el **contador es mayor o igual a 500**.
* Escriba cómo buscar un dato en el **esquema** de **Mongoose**: **Company**.  
  Debe buscar un solo elemento en la colección **companies**, donde el **identificador** de la compañía sea **“Softtek Servicios Corporativos S.A. de C.V”**. (Tip: Esto es muy similar a lo de User.findOne…).
* Escriba qué significa **CRUD** y su equivalente de cada sigla en **métodos HTTP**.
* Create
* Read
* Update
* Delete