Programación Imperativa

Módulo 3. JS Intermedio

Sesión 10: Strings y arrays: Trabajando con colecciones

Agosto 30 de 2021

Métodos de strings

Los strings en JS son objetos, y por tanto cuentan con métodos y propiedades.

length, indexOf(), slice()

Ej miString = "Me gusta JS"

miString.length // Retorna 11
miString.indexOf("gusta") // Retorna 3. Si la cadena no se encuentra, retorna -1.
miString.slice(3, 8) // Retorna "gusta". El segundo parámetro es opcional.

trim, split, replace

Ej: miString = " me gusta JS "

miString.trim() // Devuelve "me gusta JS". (Elimina los espacios en blanco antes y después de los caracteres alfanuméricos).

Ej: miNuevoString = "Me gusta JS"
miNuevoString.split(" ") // Devuelve un array: ["Me", "gusta", "JS"]

miNuevoString.replace("gusta", "encanta") // Retorna "Me encanta JS". El método replace retorna una nueva cadena, pero no altera la cadena original.

Introducción arrays

Los arrays permiten agrupar diferentes tipos de datos.

let miArray1 = ['Erika', 28, true];

Un elementos de un array puede a su ver ser un array:

let miArray2 = ['Juana', 23, false];
let miArray3 = [miArray1, miArray2];

miArray3[1][0] // Retorna 'Juana'
miArray2.length // Retorna 3

Métodos de arrays

push, pop, shift, unshift

unArray = ['azul', 'rojo', 'verde'];

unArray.pop() // Elimina y retorna el último elemento del array.
(unArray = ['azul', 'rojo']).

unArray.shift() // Elimina y retorna el primer elemento del array.
(unArray = ['rojo']).

unArray.unshift('negro', 'gris') // Agrega los elementos al inicio del array y retorna la nueva longitud del array.
(unArray = ['negro', 'gris', 'rojo']).

unArray.push('naranja', 'rosa') // Agrega los elementos al final del array y retorna la nueva longitud del array.
(unArray = ['negro', 'gris', 'rojo', 'naranja', 'rosa']).

indexOf, lastIndexOf, join, includes

miArray.indexOf("texto"); // Retorna el índice en el que se encuentra por primera vez el string "texto" dentro de miArray. Si la cadena no se encuentra, retorna -1. Funciona con strings, números.

miArray.lastIndexOf("texto"); // Similar al anterior, pero se muestra el índice de la última ocurrencia.

Ejemplo:
let unArray = ["viernes", "lunes", "martes", "viernes"];

unArray.indexOf("martes"); // Retorna 2

unArray.lastIndexOf("viernes"); // Retorna 3

unArray.join(); // Retorna "viernes,lunes,martes,viernes". (Un array que encadena los elementos del array).

unArray.join(" - "); // Retorna "viernes - lunes – martes – viernes".

UnArray.includes(“lunes”); // Retorna true. (Similar a indexOf, pero retorna booleano).

Sesión 11. Ciclos: repetir… repetir… repetir

Septiembre 1 de 2021.

For loop

```
for (let i = 1; i <= 10; i++) {
	console.log(7 * i);  // Imprime la tabla del 7
}
```

While

```
let i = 1;
while (i <= 10) {
	console.log(7 * i);  // Imprime la tabla del 7
	i++;
}
```

do while

```
let i = 1;
do {
	console.log(7 * i); // El do-while asegura que el código se ejecute por lo menos una vez.
	i++;
} while (i <= 10);
```

Sesión 13. Literalmente Objetos y Viernes 13 (JSON)

Septiembre 6 de 2021

Objetos literales

Un objeto es una estructura de datos que puede contener propiedades y métodos.

Ejemplo:

```
let miPerro = {
        nombre: 'Perla',
        edad: 11,
        amigos: ['Niki'],
        presentarse: function() {
                return "Me llamo " + this.nombre + " y tengo " + this.edad + " años."
}
```

Sintaxis

```
let nombreObjeto = {
        propiedad1: valor1,
        propiedad2: valor2,
        metodo1: function() {
                // Algunas sentencias
        }
}
```

Constructor

```
function Perro(nombre, edad, amigos) {
    this.nombre = nombre,
    this.edad = edad,
    this.amigos = amigos
    this.presentarse = function() {
        return "Me llamo " + this.nombre + " y tengo " + this.edad + " años."
    }
}
```

Instanciar un objeto

```
let miPerro = new Perro('Perla', 11, ['Niki']);
```

JSON: JavaScript Object Notacion
Es un formato de texto utilizado para el intercambio de datos entre distintos sistemas.

parse:
Convierte un string en objeto (o array).

stringify:
Convierte un objeto (o array) en string.

```
let miPerro = {
        nombre: 'Perla',
        edad: 11,
        amigos: ['Niki']
}
```

let stringMiPerro = JSON.stringify(miPerro); // Devuelve un string en formato JSON

let objetoMiPerro = JSON.parse(stringMiPerro); // Devuelve el objeto original

Objeto Literal	JSON
Admite comillas simples y dobles.	Solo se pueden usar comillas dobles.
Las claves del objeto van sin comillas.	Las claves van entre comillas.
Podemos escribir métodos sin problemas.	No admite métodos, solo propiedades y valores.
Se recomienda poner una coma en la última propiedad.	No podemos poner una coma en el último elemento.

Diferencias entre Objetos literales de JavaScript y JSON

Sincrónico

https://www.codewars.com/

Sesión 14.

Sincrónico

```
listarDepartamentos: function(arrayDeptos = this.departamentos) {
        //
        // Si no se pasa un argumento, toma 'this.departamentos' por defecto
}
```

Prettier: estiliza el código. Extensión de vsc.

Sesión 16. Modul-ando

Septiembre 13 de 2021

Módulo
Es un bloque de código reutilizable.

- Módulos nativos. Vienen instalados por defecto.
- Módulos externos. Se instalan usando npm.
- Módulos creados.

Módulos nativos de Node.js:
https://www.nodejs.org/api

Módulos creados:

Para exportar:
module.exports = objetoCompartido;
//Escribir esta línea al final del archivo

Para importar:
const objetoImportado = require('./objetoCompartido/index');
//Requiere devuelve un objeto literal; se requiere una variable para almacenarlo.

Notas:
- Cuando se trata de un archivo .js no se requiere especificar la extensión.
- El módulo que se exporta se suele nombrar index.js y se guarda en una carpeta que tiene el mismo nombre del objeto que se exporta.

Lectura y escritura de archivos

writeFileSync()

const fs = require('fs');
fs.writeFileSync('texto.txt', 'Hola Mundo!');
// Crea el archivo texto.txt y escribe en él el texto 'Hola Mundo!'. Si el archivo ya existe, lo sobreescribe.

appendFileSync()
// Agrega contenido al final de un archivo existente. Si el archivo no existe, lo crea.

```
let usuario = {
        nombre: 'Mauricio Pineda',
        cc: 71264076
```

```
}
let usuarioJson = JSON.stringify(usuario);
//El objeto se debe convertir a Json para poder imprimirlo en el archivo.
fs.appendFileSync(usuarioJson);
```

readFileSync()

```
//Lee el contenido de un archivo existente. Se requiere decodificar el archivo, ya que lo que se recibe es un buffer

let contenido = fs.readFileSync('archivo.txt', { encoding: 'utf-8' });
```

Sincrónico

Exportar un módulo con dos funciones:

```
module.export = {saludar, suma}
```

En el archivo que llama al módulo:

```
const jsonHelper = require('./jsonHelper');

let saludo =jsonHelper.saludar();
```

export default

Una forma de exportar funciones de ECMAScript 6 (revisar si funciona con Node.js).

```
export default { saludar, suma };
```

Otra opción (revisar):

```
export default function miFunction() {
	// Código de la función.
}
```

Leer y escribir archivos

Para leer un Json y retornar el objeto:

```
const lector = function (nombreArchivo) {
	return JSON.parse(fs.readFileSync(`${__dirname}/${nombreArchivo}.json`, "utf8"))
```

Para escribir un archivo Json

```
const escritor = function(nombreArchivo, datos) {
        fs.writeFileSync(`${__dirname}/${nombreArchivo}.json`,
        JSON.stringify(datos));
}
```