Introducción

JavaScript es un lenguaje de programación interpretado(El navegador lee directamente el código, sin necesidad de terceros) de alto nivel. Es el lenguaje de programación encargado de dotar de mayor interactividad y dinamismo a las páginas web.

¿Cuáles son los principales objetivos de estos ejercicios?

- Entender cómo utilizar el operador Spread/Rest
- Aprender a utilizar la desestructuración

1. Ejercicios Destructuring

- Dado el siguiente objeto:

```
const empleados = [

  { name: "Luis", email: "Luis@gmail.com" },

  { name: "Ana", email: "Ana@gmail.com" },

  { name: "Andrea", email: "Andrea@gmail.com" },

];
```

    - Extrae la empleada Ana con todos sus datos personales:

```
{"name":"Ana", "email":"Ana@gmail.com"}
```

    - Extrae el email del empleado Luis --> Luis@gmail.com

- Dado el siguiente objeto:

```
const pokemon = {

    name: "Bulbasaur",

    type: "grass",

    ability: {

        "primary": "Overgrow",

        "hidden": "Chlorophyll"

    },

    stats: {

        hp: 45,

        attack: 49,
```

```
        deffense: 59,

        speed: 45

    },

    moves: [

        "Growl", "Tackle", "Vine Whip", "Razor Leaf"

    ]

}
```

- Cambia el nombre de la propiedad “name” por “nombre”.
- Extrae el nombre del pokémon.
- Extrae el tipo de pokémon que es.
- Extrae el movimiento “Tackle”.

2. Ejercicios Spread/Rest

- Mergea el siguiente pokémon con el pokémon del ejercicio anterior:

```
const pikachu = {
    name: "Pikachu",
    type: "electric",
    ability: {
        "primary": "Static",
        "hidden": "Lightning rod"
    },
    stats: {
        hp: 35,
        attack: 55,
        deffense: 40,
        speed: 90
    },
    moves: [
        "Quick Attack", "Volt Tackle", "Iron Tail", "Thunderbolt"
    ],
}
```

- Escribe una función llamada **sumEveryOther** que pueda recibir cualquier cantidad de números y devuelva la suma de todos los demás argumentos.

```
sumEveryOther(6, 8, 2, 3, 1); //20
sumEveryOther(11, 3, 12); //26
```

- Escribe una función llamada **addOnlyNums** que pueda recibir cualquier número de argumentos (incluyendo números y strings) y retorne la suma solo de los números.

```
addOnlyNums(1, 'perro', 2, 4); //7
```

- Escribe una función llamada **countTheArgs** que pueda recibir cualquier número de argumentos y devuelva un número que indique cuántos argumentos ha recibido.

```
countTheArgs('gato', 'perro'); //2
countTheArgs('gato', 'perro', 'pollo', 'oso'); //4
```

- Escribe una función llamada **combineTwoArrays** que reciba dos array como argumentos y devuelva solo un array que combine los dos (usando Spread Operator).

3. Extras

- Dado el siguiente objeto:

```
const HIGH_TEMPERATURES = {

  yesterday: 30,

  today: 35,

  tomorrow: 32,

};
```

    - Cambiar las siguientes líneas para guardar desestructurando los valores de temperaturas en las variables maximaHoy y maximaManana.

```
const maximaHoy = HIGH_TEMPERATURES.today;

const maximaManana = HIGH_TEMPERATURES.tomorrow;

console.log(maximaHoy)

console.log(maximaManana)
```

- Escriba una función llamada **onlyUniques** que acepte cualquier número de argumentos y devuelva un array de elementos únicos, sin repetidos.

```
onlyUniques('gato', 'pollo', 'cerdo', 'cerdo');

//['gato', 'pollo', 'cerdo']

onlyUniques(1, 1, 2, 2, 3, 6, 7, 8); //[1, 2, 3, 6, 7, 8]
```

- Escriba una función llamada **combineAllArrays** que pueda recibir cualquier cantidad de arrays como argumentos y los combine todos en un solo array.

```
combineAllArrays([3, 6, 7, 8],[2, 7, 3, 1])

// [3, 6, 7, 8, 2, 7, 3, 1]

combineAllArrays([2, 7, 3, 1],[2, 7, 4, 12],[2,
44, 22, 7, 3, 1]);

// [2, 7, 3, 1, 2, 7, 4, 12, 2, 44, 22, 7, 3, 1]
```

- Escriba una función llamada **sumAndSquare** que reciba cualquier número de argumentos, los eleve al cuadrado y devuelva la suma de todos los valores cuadrados.

Entregables

- Subir el ejercicio a un repositorio en GitHub
- Dejar el enlace del repositorio adjuntado en Classroom.