**Trabajo de Investigación**

**¿Cuáles son las diferencias entre ECMAScript 5 y ECMAScript 6?**

Estas son algunas diferencias que ES6 tiene con ES5:

* Función Flecha (arrow)

//*ES5*  
function(){  
 ...  
}//*ES6*  
() => {  
 ...  
}

* Parámetros

Cuando las funciones no tienen ningún parámetro, es obligatorio poner los paréntesis.

Cuando se tiene sólo un parámetro, no es obligatorio poner los paréntesis:

value => {  
 ...  
}

Cuando se tienen más de un parámetro (o ninguno), sí es obligatorio poner los paréntesis:

(value1, value2) => {  
 ...  
}

* Si la función retorna o devuelve un resultado (un valor), tienen las siguientes reglas:

Si la función sólo cuenta con una línea de código, no es necesario poner las llaves:

//*ES5*array.filter(function (value){  
 return value > 5;  
});//*ES6*array.filter(value => value > 5);

* Si lo que necesita retornar es un objeto, sería de la siguiente forma:

//*ES5*array.map(function (value, index){  
 return {  
 index: index,  
 value: value  
 }  
});//*ES6*array.map((value, index) => **(**{ index: index, value: value }**)**);

* Asignación “clave:valor” en un objeto

En los objetos, no siempre es necesario poner la clave:valor**.** Cuando en las llaves, se pone sólo la variable, el objeto tendrá de clave el nombre de la variable. Es decir:

*let value = "hola mundo";//ES5  
let exampleObject = { value: value };*console.log(JSON.stringify(*exampleObject*)); *//*{"value":"hola mundo"}*//ES6  
let exampleObject = { value };*console.log(JSON.stringify(*exampleObject*)); *//*{"**value**":"hola mundo"}  
*//En el ejemplo anterior, es válido tener esto:  
array.map((value, index) => ({ index, value }));*

* Valores por defecto en las funciones

Los parámetros de una función, pueden llevar valores por defecto así:

(value1 = [], value2) => {  
 ...  
}

En el ejemplo, se muestra que el parámetro value1 es igual []. ¿Qué significa eso?, si ese parámetro es recibido como indefinido (*undefined*), tomará el valor [], de lo contrario, su valor será el enviado cuando se ejecuta dicha función.

* Concatenación de cadenas

Cuando se concatenen cadenas, normalmente se hace algo parecido a esto:

//*ES5*  
let name1 = "hello";  
let name2 = "world";  
let concatName = name1 + " " + name2;  
console.log(concatName);  
//*hello world*

Pero, con ES6 ya es más fácil la concatenación:

//*ES6*  
name1 = "hello";  
let name2 = "world";  
let concatName = **`${name1}** **${name2}`**;  
console.log(concatName);  
//*hello world*

Para que se concatene el valor de la variable (y que no sea una palabra más dentro de la cadena) se debe de poner dentro de ${} (al interior de las llaves, como la sustitución por plantilla en Python).

* Número de parámetros indeterminados

Cuando necesitamos enviar un número no definido de parámetros una función, una de las primeras cosas que se vienen a la mente es un Array y está bien. Pero otra forma de entregarle un Array de valores a una función, es enviarlo uno a uno como valores independientes. En ES5 pasaría lo siguiente:

function total(value1, value2){  
 console.log(JSON.stringify(arguments));  
}  
total("h", "e", "l", "l", "o");  
//*{ 0: "h", 1: "e", 2: "l", 3: "l", 4: "o" }*

¿Qué se puede hacer para obtener los otros parámetros?; una idea muy trillada, sería pensar en un método que convierta el objeto en un Array, crear otro Array eliminan las clave del objeto, y tener sólo los valores (así, value1 y value2 no los necesitaríamos porque y están en el Array). Pero, hay otra forma con ES6, y es, poniendo … (tres puntos) antecedida de la variable (parámetro) así:

function total(**...values**){  
 console.log(values);  
}  
total("h", "e", "l", "l", "o");  
//*["h", "e", "l", "l", "o", 1]*

Así, le estamos entregando a la función un Array con todos los valores, enviados. También se pueden enviar parámetros fuera del Array, como valores relevantes dentro de la función.

function total(value1, value2, **...values**){  
 console.log(value1, value2, values);  
}  
total("h", "e", "l", "l", "o");  
//*"h", "e", ["l", "l", "o"]*

Los … (tres puntos), también sirven para concatenar Arrays así:

let exampleArray = [1, 2, 3];//*ES5*  
console.log(exampleArray.concat([4, 5, 6]));  
//*[1, 2, 3, 4, 5, 6]*//*ES6*console.log([**...exampleArray**, 4, 5, 6]);//*[1, 2, 3, 4, 5, 6]*

* Llamar a funciones y/o variables desde otro archivo Javascript

Para mantener los archivos Javascript (.js) organizados y fáciles de entender, se debe de crear varios archivos, cada uno con una responsabilidad diferente. Entonces, supongamos que tenemos el siguiente archivo:

exponential.js

function exponential2(value){  
 return value ^ 2;  
}function exponential3(value){  
 return value ^ 3;  
}function exponential4(value){  
 return value ^ 4;  
}

Para, poder exportar todas las funciones de este archivo y poderlas usar en otro, se hace antecediendo export así:

//*exportar una variable* **export** const PI = 3.14159265359;//*exportar función*  
**export** function exponential2(value){  
 return value ^ 2;  
}**export** function exponential3(value){  
 return value ^ 3;  
}//*exportar función y/o variable por defecto*  
**export default** random(){  
 return Math.random();  
}

Para usar las funciones del archivo exponential.js, se hace de diferentes formas:

* Se importan todas las funciones y/o variables así:

//*from "ruta del archivo .js"*  
//*todas las funciones y/o variables*  
//*del archivo, referenciadas por*  
//*la variable* ***exponentials***import **\* as exponentials** from "exponential";

* Se importan sólo las funciones y/o variables en específico así:

//*from "ruta del archivo .js"*  
//*Aquí, se obtiene una función y*  
//*una variable, más no todo el contenido*  
//*del archivo*  
import { **exponential2, PI** } from "exponential";

* Se importa sólo una función y/o variable en específico así:

//*from "ruta del archivo .js"*  
//*Para esto, la función y/o variables*  
//*debe estar antecedida por* ***export default*** *...*   
import **random** from "exponential";

* Importación de JS

Supóngase que instaló [lodash](https://lodash.com/) en su proyecto. La forma de usarlo tanto con ES5 y ES6 es la siguiente:

//*ES5*var \_ = require("lodash");//*ES6*import \_ from "lodash";