Javascript

Primeros Pasos

Acerca de mí

Cristian Buitrago

Software Engineer

- **Software Engineer** who loves working with Javascript and System Design .
- Anime Lover.
- **Education Enthusiast**.

Github @buty619

Código de conducta

Sé respetuoso, no hay malas preguntas o ideas.

Sé cordial y paciente.

Sé cuidadoso con tus palabras.

Puntos importantes

Identifícate en Zoom utilizando tu nombre y apellido.

Mantén tu micrófono apagado durante el transcurso de la sesión.

Utiliza el chat para hacer tus preguntas durante la sección de Q&A.

Procura enfocar tus preguntas al tema presentado.

Apaga tu cámara en caso de tener problemas con tu conexión.

Contenido del curso

1

Funciones

¿Qué es una función?
¿Cómo se declara una función?
¿Para qué usamos las funciones?
Arrow function

2

Iteradores

Bucle For
Bucle while y do while
Bucle Infinito

3

Arreglos

Propiedad length
Método forEach
Método find
Método filter
Método map
Método sort
Método slice

4

Objetos literales

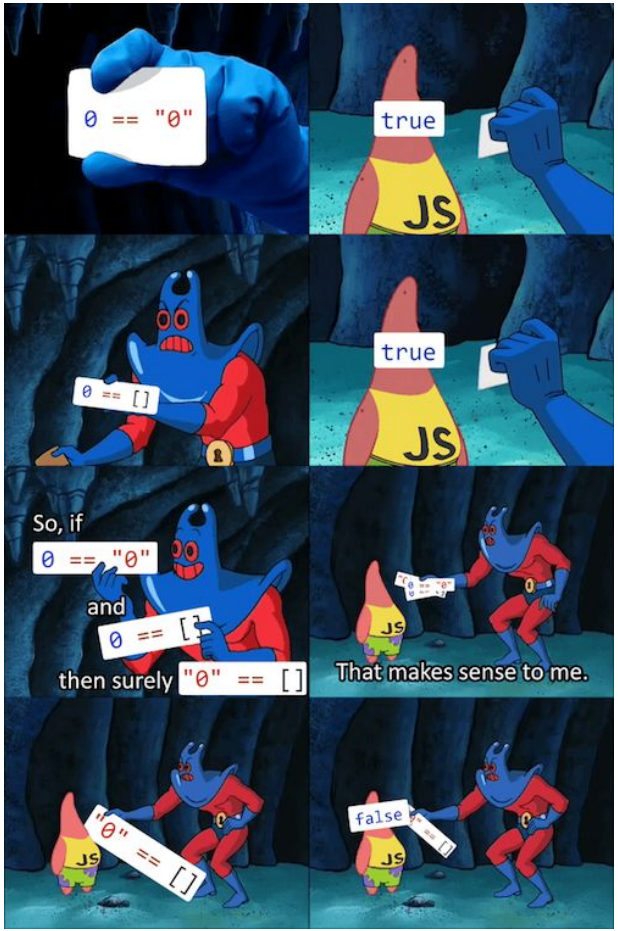
Método entries
Método keys
Método values
Método create

Objetivos de la sesión

Al final de la sesión serás capaz de:

- ➔ Identificar los tipos de funciones que existen
- ➔ Declarar una función
- ➔ Conocer las diferencias entre una función regular y una función flecha

Javascript



Funciones - ¿Qué es una función?

Una función es un bloque de código que diseñamos con el fin de ejecutar una tarea determinada

¿Cómo se declara una función?

```
function operacion(parametro1, parametro2) {
 // código a ejecutar
}
```

¿Para qué usamos las funciones?

La utilidad de las funciones en JavaScript es la de reutilizar bloques de código

para llamar una función usaremos el nombre de la función seguido de paréntesis, dentro de los paréntesis si es el caso, indicaremos los argumentos que evaluara la función separados por comas

```
operacion(2, 4);
```

Funciones - Scope

```
// el código NO accederá a la variable suma de la función operación
function operacion(parametro1,parametro2){
  let suma = parametro1 + parametro2;
  // el código SI accederá a la variable suma
}
// el código NO accederá a la variable suma de la función operación
```

Funciones - Return

Cuando las funciones alcanzan la palabra clave return detienen su ejecución,

```
function operacion(parametro1, parametro2) {
 let suma = parametro1 + parametro2;
 return suma;
}


let resultado = operacion(2, 4);
```

Hands on

Hands On

1. En esta actividad, en Visual Studio Code, crea una función llamada operacionDivision que reciba dos parámetros, nombrados parametro1 y parametro2, en el bloque de código de la función crea una variable llamada div asignándole el valor de la operación parametro2 / parametro1.
2. Ahora añade la palabra return div, para finalizar, por fuera de la función declara una variable llamada resultado y asigna como resultado a la llamada de la función operacionDivision(4/2), imprime en consola la variable resultado.

Funciones - Arrow function

Las funciones Flecha son una alternativa compacta para escribir una función.

```
function suma(x, y) {
 return x + y;
}
console.log(suma(3, 4));
```

```
const suma = (x, y) => {
 return x + y;
};
console.log(suma(3, 4));
```

Funciones - function vs Arrow function

Las arrow functions fueron creadas para simplificar el scope de las funciones.

```
const persona = {
    nombre: 'Agustin',
    imprimirNombre: function(){
      console.log(this.name)
    }
}
console.log(persona.imprimirNombre()) // OUTPUT : 'Agustin'
```

Funciones - function vs Arrow function

```
const persona = {
    nombre: 'Agustin',
    imprimirNombre: function() {
        console.log(this.nombre)
    },
    imprimirNombreFlecha: () => {
        console.log(this.nombre)
    }
}
persona.imprimirNombreFlecha(); // OUTPUT : undefined
```

Funciones - Early Return

El early return es la manera de escribir funciones donde el resultado positivo de la condición se devuelve al final de la función

```
const esPar = (parametro1) => {
  if(parametro1 % 2 === 0) return true;
  if(isNaN(parametro1 % 2)) return 'el parámetro dado no es un número';
  return false
}
console.log(esPar(2)) // true
console.log(esPar('hola')) // el parámetro dado no es un número
console.log(esPar(3)) // false
```

Hands on

Hands On

1. En visual Studio Code, declara una variable comparación y asigna a esta variable una función flecha con dos parámetros, el bloque de código , deberá evaluar mediante condicionales si el valor de los parámetros es igual o diferente, en cuyo caso deberás devolver el resultado correspondiente.
2. Ahora escribe console.log(comparación(2,10));, ¿qué resultado obtuviste?, realiza más pruebas e intenta comparar palabras.

Q&A

Objetivos de la sesión

Al final de la sesión serás capaz de:

- ➔ Entender el concepto de bucles en javascript
- ➔ Generar bucles con el método for
- ➔ Generar bucles con el método while

¿Qué se debe hacer sí necesitamos que un determinado bloque de código se repita una cierta cantidad de veces?

Iteradores - For, While & Do While

```
for (inicializador; condición de salida; iterador){
// código a ejecutar
}
```

```
While (condición) {
// código a ejecutar
iterador
}
```

```
do{
// código a ejecutar
iterador
} while (condición)
```

Iteradores - Bucle infinito

Es muy importante tener claro qué en cualquier bucle debemos estar seguros de que la condición que declaremos se cumpla, de lo contrario habremos creado un bucle infinito que bloqueara el código e impedirá que puedas seguir ejecutando el flujo de codigo.

Hands on

Hands On

- Para aplicar lo aprendido, deberás tomar el bucle while y modificarlo para crear uno do while, ejecutalo en Visual Studio Code compara los resultados, por último, elimina el iterador y vuelve a ejecutar el código.

```
let i=0
while(i<10){
 const texto += 'el número es ' + i;
 console.log(texto);
 i++
}
```

- Ahora genera el mismo bucle utilizando la sentencia for

Hands On

- ¡Ayuda a corregir todos los errores en la función incrementItems! ¡Está destinado a agregar 1 a cada elemento en el arreglo!

```
function incrementItems(arr) {
 for (let i = 0; i < array.length; i++) {
   arr[i] === arr[i] + 1;
 }
 return array;
}
```

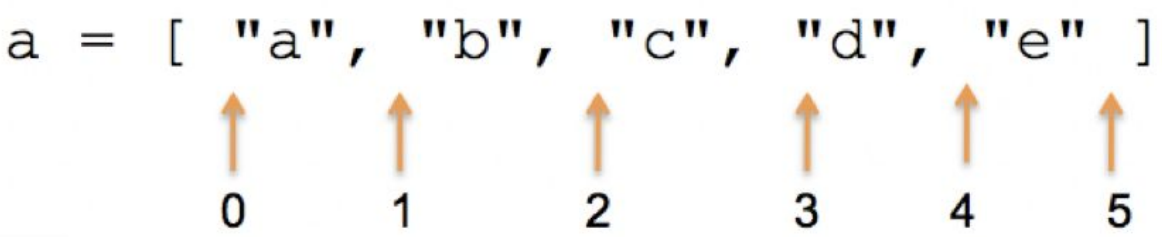
Q&A

Objetivos de la sesión

Al final de la sesión serás capaz de:

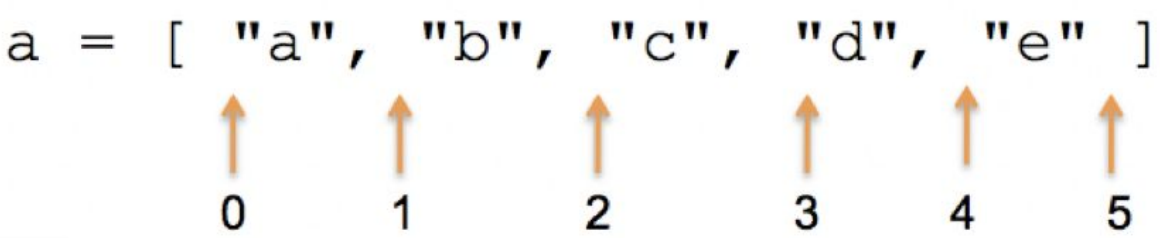
- ➔ Entender el concepto del tipo de dato Array
- ➔ Entender los métodos más relevantes de los Array
- ➔ Manipular un Array (CRUD)

Arreglos




```
console.log(a[5]);
```

Arreglos - Propiedad length




```
console.log(a.length); // imprimirá el número 5
```

Arreglos - Método forEach

```
//función flecha
array.forEach((elemento, index, array) => expresión())
//función tradicional
array.forEach(function(element, index, array){expresion})


const lista1 = ['cristian', 'yina', 'andrea'];
lista1.forEach((elemento, index) => {
    console.log(index + ' ' + elemento)
});
```

Arreglos - Método find

```
//función flecha
array.find((elemento, index, array) => expresión);
//función tradicional
array.find(function (element, index, array) {
 expresion;
});

const lista1 = [1, 2, 3, 4, 5];
console.log(lista1.find((elemento) => elemento > 1));
// imprimirá el número 2
```

Arreglos - Método filter

```
//función flecha
filter((elemento, index, array) => expresión);
//función tradicional
filter(function (element, index, array) {
 expresion;
});

const lista1 = [1, 2, 3, 4, 5];
console.log(lista1.filter((elemento) => elemento > 1));
// imprimirá el array [2,3,4,5]
```

Arreglos - Método Map

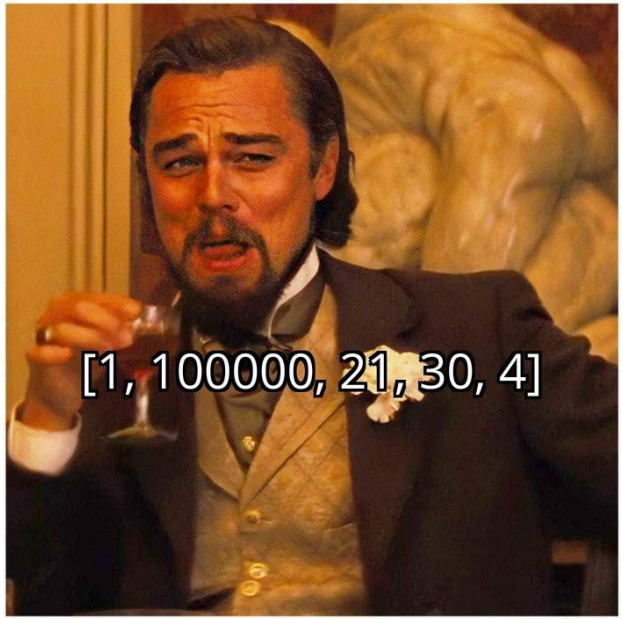
```
//función flecha
map((elemento, index, array) => expresión);
//función tradicional
map(function (element, index, array) {
 expresion;
});


let listaNumeros = [5, 6, 7, 8];
let listaDoble = listaNumeros.map((elemento) => elemento * 2);
// listaDoble corresponderá a [10, 12, 14 , 16];
// listaNumeros continuará con su valor original
```

Arreglos - Método Sort

People learning JavaScript:
"I'll use array.sort() to
sort this list of numbers!"

JavaScript:



Arreglos - Método Sort

En el caso que se inyecte una función al método .sort lo elementos del array serán ordenados dependiendo del valor de retorno de la función siendo val1 y val2 los elementos comparados dentro de la función de comparación:

- Si comparando val1 y val2 es menor que 0, val1 tomará un índice menor que val2. Es decir, val1 viene primero que val2.
- Si comparando val1 y val2 es mayor que 0, val1 tomará un índice mayor que val2. Es decir, val1 viene después de val2.
- Si comparando val1 y val2 es igual que 0 no se intercambian posiciones.

```
const listaNumeros = [8, 0, 3, 1, 9];
let listaOrdenada = listaNumeros.sort((a, b) => {
 if (a > b) {
   return 1;
 }
 return -1;
});
console.log(listaOrdenada); // [0, 1, 3, 8, 9]
```

Arreglos - Método Slice

```
const list1 = [28, 45, 12, 32, 98, 72];
const listSlice = list1.slice(2);
console.log(listSlice); // [12, 32, 98, 72];


const list1 = [28, 45, 12, 32, 98, 72];
const listSlice = list1.slice(1, 4);
console.log(listSlice); // [45, 12, 32];
```

Hands on

Hands On

1. Declararemos una variable list1 a la cual le asignaremos la lista de números 28, 45 y 19, ahora declaremos otra variable list2 a la cual le asignaremos una lista de nombres, Juan, Carlos, Cristian, lo primero que haremos será comparar si las dos listas tienen la misma longitud e imprimiremos ese resultado en consola.
2. Ahora, necesitamos tomar cada elemento de list1 e imprimir ese elemento en consola usaremos foreach.
3. Otro ejercicio será, encontrar en list2 el primer elemento que su longitud sea mayor a 7.
4. Ahora declararemos una variable nuevaLista donde guardaremos el resultado de filtrar los elementos de list2 donde su letra inicial sea C e imprimiremos en consola nuevaLista.

Q&A

Objetivos de la sesión

Al final de la sesión serás capaz de:

- ➔ Entender el concepto del tipo de dato Objeto
- ➔ Entender los métodos más relevantes de los Objeto
- ➔ Manipular un Objeto (CRUD)

Objetos

Objetos

Objetos

El tipo de dato objeto o object permite almacenar múltiples variables y su respectivo valor, el nombre correcto de esas propiedades son clave y valor

```
const vehículo = {
    modelo: 2022,
    marca: "chevrolet",
    color: "rojo"
};
console.log(vehiculo.color); // imprimirá rojo.
console.log(vehiculo[`color`]);// imprimirá rojo.
```

Objeto - Método Entries

```
const vehículo = {
    modelo: 2022,
    marca: "chevrolet",
    color: "rojo"
};
let objetolista = Object.entries(vehículo);
console.log(objetolista)// [Array(2), Array(2), Array(2)]
console.log(objetolista[0]) //  ['modelo', 2022]
console.log(objetolista[1]) //  ['marca', 'chevrolet']
console.log(objetolista[2]) //  ['color', 'rojo']
```

Objeto - Método Keys

```
const vehículo = {
    modelo: 2022,
    marca: "chevrolet",
    color: "rojo"
};
let objetoKeys = Object.keys(vehiculo);
//objetoKeys = [modelo, marca, color]
```

Objeto - Método Values

```
const vehículo = {
    modelo: 2022,
    marca: "chevrolet",
    color: "rojo"
};
let objetoValues = Object.values(vehiculo);
// objetoValues = [2022, chevrolet, rojo]
```

Objeto - Método Create

```
const vehículo = {
 modelo: 2022,
 marca: "chevrolet",
 color: "rojo",
};
const camion = Object.create(vehículo);
//Aquí se determinarán las características
heredadas del objeto prototipo vehículo.
camion.modelo = 2015;
camion.marca = "renault";
camion.color = "blanco";
//Aquí se creará una propiedad del objeto camion
camion.capacidadCarga = "3 toneladas";
```

Hands on

Comprobar si la propiedad existe en el objeto

Escriba una función que tome un objeto (a) y un string (b) como argumento. Devuelva true si el objeto tiene una propiedad con la clave 'b'. Devuelva false de lo contrario.

Test Case	Expected
myFunction({a:1,b:2,c:3},'b')	`true`
myFunction({x:'a',y:'b',z:'c'},'a')	`false`
myFunction({x:'a',y:'b',z:undefined},'z')	`false`

Hands On

Creación de objetos Javascript

Escribe una función que tome dos arreglos (a y b) como argumentos.

Crear un objeto que tenga propiedades con claves 'a' y valores correspondientes 'b'. Devolver el objeto.

Test Case	Expected
myFunction(['a','b','c'],[1,2,3])	{a:1,b:2,c:3}
myFunction(['a','b','c'],[1,() => {}, {name: 'khriztian'}])	{a:1,b:() => {}, c:{name: 'khriztian'}}
myFunction(['name','hobbies','isAdmin'],['khriztian',['music', 'tv series'], false])	{name:'khriztian', hobbies:['music', 'tv series'], isAdmin:false}

Sumar valores de objeto

Escribir una función que tome un objeto como argumento. Devuelve la suma de todos los valores de las propiedades del objeto.

Test Case	Expected
myFunction({a:1,b:2,c:3})	6
myFunction({j:9,i:2,x:3,z:4})	18
myFunction({w:15,x:22,y:13})	50

Sumar valores de objeto

Escribir una función que tome un objeto como argumento. Devuelve la suma de todos los valores de las propiedades del objeto.

Test Case	Expected
myFunction({a:1,b:2,c:3})	6
myFunction({j:9,i:2,x:3,z:4})	18
myFunction({w:15,x:22,y:13})	50

Secuencia de fibonacci

Escriba una función llamada fibonacci que reciba un número y devuelva el número n de la secuencia de fibonacci.

Test Case	Expected
fibonacci(0)	**1**
fibonacci(1)	**1**
fibonacci(7)	**1 1 2 3 5 8 13**

Referencias

- https://www.youtube.com/watch?v=pCt672Dq05M
- https://developer.mozilla.org/es/docs/Web/JavaScript/Inheritance_and_the_prototype_chain
- https://github.com/getify/You-Dont-Know-JS/blob/2nd-ed/scope-closures/README.md
- https://www.youtube.com/watch?v=5yPf74sCu2k
- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://www.youtube.com/watch?v=VX-AyFo4yHE
- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Object

Queremos conocer tu opinión

Gracias