Javascript Avanzado II

✏️ 2025-01 — ⏱️ 45 min.

Some y Every

Some y Every

```
function muyPeque(item, index, array) {
  return item < 3;
}

const numeros = [0, 1, 2, 3, 4, 5];

console.log(numeros.some(muyPeque)); // ?
console.log(numeros.every(muyPeque)); // ?
```

Some y Every

```
function muyPeque(item, index, array) {
  return item < 3;
}

const numeros = [1, 2, 3, 4, 5];

console.log(numeros.some(muyPeque)); // true
console.log(numeros.every(muyPeque)); // false
```

.find()

```
// función de busqueda que retorna true para el elemento que queremos
function findFunction(item) {
  return item === 4;
}

const numbers = [4, 5, 2, 1, 7];

// se le pasa la función de busqueda a .find
console.log(numbers.find(findFunction)); // 4
```

.find()

```
// función de busqueda que retorna true para el elemento que queremos
function findFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

// se le pasa la función de busqueda a .find
console.log(numbers.find(findFunction)); // ??
```

.find()

```
// función de busqueda que retorna true para el elemento que queremos
function findFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

// se le pasa la función de busqueda a .find
console.log(numbers.find(findFunction)); // 4
```

.find()

Le pasas una funcion de busqueda, y retorna el primer elemento para el que la funcion retorne true.

.findIndex()

```
function findFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.findIndex(findFunction)); // ??
```

.findIndex()

```
function findFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.findIndex(findFunction)); // 0
```

.findIndex()

Le pasas una funcion de busqueda, y retorna el **indice** del primer elemento para el que la funcion retorne true.

Pregunta Ejemplo #1

Para retornar si en una lista de estudiantes se encuentra un estudiante con una cedula dada, que función se utilizaría?

Some	Every
Find	FindIndex

Pregunta Ejemplo #1

Para retornar si en una lista de estudiantes se encuentra un estudiante con una cedula dada, que función se utilizaría?

Some	~~Every~~
~~Find~~	~~FindIndex~~

Pregunta Ejemplo #2

Para retornar la información del estudiante de una lista de estudiantes con una cedula dada, que función se utilizaría?

Some	Every
Find	FindIndex

Pregunta Ejemplo #2

Para retornar la información del estudiante de una lista de estudiantes con una cedula dada, que función se utilizaría?

~~Some~~	~~Every~~
Find	~~FindIndex~~

.filter()

```
function filterFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.filter(filterFunction)); // ??
```

.filter()

```
function filterFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.filter(filterFunction)); // [4, 2]
```

.filter()

Le pasas una funcion de filtro, y retorna todos los elemento para los que la funcion retorne `true`.

.map()

```
function mapFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.map(mapFunction)); // ??
```

.map()

```
function mapFunction(item) {
  return !(item % 2);
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.map(mapFunction)); // [true, false, true, false, false, false]
```

.map()

Le pasas una funcion de mutación y retorna los elementos tras ser mutados con esa función.

.map()

```
function mapFunction(item) {
  if (item % 2 === 0) {
    return item;
  }
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.map(mapFunction)); // ??
```

.map()

```
function mapFunction(item) {
  if (item % 2 === 0) {
    return item;
  }
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.map(mapFunction)); // [4, undefined, 2, undefined, undefined]
```

.map()

Le pasas una funcion de mutación y retorna los elementos tras ser mutados con esa función.

Map no filtra.

.sort()

```
1  const numbers = [41, 5, 2, 19, 7];
2
3  console.log(numbers.sort()); // ??
```

.sort()

```
1  const numbers = [41, 5, 2, 19, 7];
2
3  console.log(numbers.sort()); // [19, 2, 41, 5, 7]
```

.sort()

```
function sortAscFunction(item1, item2) {
  if (item1 < item2) {
    return 1;
  } else if (item2 > item1) {
    return -1;
  } else {
    return 0;
  }
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.sort(sortFunction)); // ??
```

.sort()

```
function sortAscFunction(item1, item2) {
  if (item1 < item2) {
    return 1;
  } else if (item2 > item1) {
    return -1;
  } else {
    return 0;
  }
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.sort(sortFunction)); // [1, 2, 4, 5, 7]
```

.sort()

```
function sortAscFunction(item1, item2) {
  return item1 - item2;
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.sort(sortFunction)); // ??
```

.sort()

```
function sortAscFunction(item1, item2) {
  return item1 - item2;
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.sort(sortFunction)); // [1, 2, 4, 5, 7]
```

.sort()

Le pasas una función de "ordenamiento", y retorna los elementos ordenados segun esa función.

La funcion de "ordenamiento" debe retornar numero negativo, positivo, o cero, segun el orden relativo de los elementos a comparar.

No es que le vayas a pasar burbuja, o inserción.

Si no pasas una función, los ordena como strings.

.reduce()

```
function reduceFunction(acumulado, actual) {
  return acumulado + actual;
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.reduce(reduceFunction)); // ??
```

.reduce()

```
function reduceFunction(acumulado, actual) {
  return acumulado + actual;
}

const numbers = [4, 5, 2, 1, 7];

console.log(numbers.reduce(reduceFunction)); // 19
```

.reduce()

Le pasas una funcion de reducción, y retorna el resultado de reducir el array usando la función.

.reduce()

Le pasas una funcion de reducción, y retorna el resultado de reducir el array usando la función.

El primer valor del acumulado es el primer elemento del array.

Alternativamente, se le puede pasar un valor inicial como segundo parametro del reduce.

.reduce()

```
function reduceFunction(acumulado, actual) {
  return acumulado + actual;
}

const numbers = [4, 5, 2, 1, 7];

// 1 es el valor inicial
console.log(numbers.reduce(reduceFunction, 1)); // 20
```

Intermisión

Destructuración Objetos

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre } = user;

console.log(nombre); // ??
console.log(user.nombre); // ??
```

Destructuración Objetos

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre } = user;

console.log(nombre); // "Ismael"
console.log(user.nombre); // "Ismael"
```

Destructuración Objetos

```
1   const user = {
2     nombre: "Ismael",
3     altura: 1.75,
4     puntaje: 3,
5   };
6
7   const { nombre, altura } = user;
8
9   console.log(nombre); // "Ismael"
10  console.log(altura); // 1.75
```

Destructuración Objetos

```
const user = {
  nombre: "Ismael",
  // altura: 1.75,
  puntaje: 3,
};

const { nombre, altura } = user;

console.log(nombre); // "Ismael"
console.log(altura); // undefined
```

Destructuración Objetos

```
// OPCIÓN A
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre, altura } = user;
```

```
// OPCIÓN B
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const nombre = user.nombre;
const altura = user.altura;
```

Valor Predeterminado

```
const user = {
  nombre: "Ismael",
  // altura: 1.75,
  puntaje: 3,
};

const { nombre, altura = 1.7 } = user;

console.log(nombre); // "Ismael"
console.log(altura); // 1.7
```

Renombrar variable

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre: name } = user;

console.log(nombre); // undefined
console.log(name); // "Ismael"
```

Renombrar + Predeterminado

```
const user = {
  // nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre: name = "Persona" } = user;

console.log(nombre); // undefined
console.log(name); // "Persona"
```

Spread Operator

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre, ...otrosDatos } = user;

console.log(Object.keys(otrosDatos)); // ??
```

Spread Operator

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre, ...otrosDatos } = user;

console.log(Object.keys(otrosDatos)); // ["altura", "puntaje"]
```

Spread Operator

```
const user = {
  nombre: "Ismael",
  altura: 1.75,
  puntaje: 3,
};

const { nombre, ...otrosDatos } = user;

console.log(otrosDatos.altura); // 1.75
console.log(user.altura); // 1.75
```

Intermisión

Destructuración Listas

```
1  const users = ["Ismael", "Rafael", "Otroel"];
2
3  console.log(users[1]); // ??
4
5  const [user_0, user_1] = users;
6
7  console.log(user_1); // ??
8  console.log(users[1]); // ??
```

Destructuración Listas

```
const users = ["Ismael", "Rafael", "Otroel"];

console.log(users[1]); // "Rafael"

const [user_0, user_1] = users;

console.log(user_1); // "Rafael"
console.log(users[1]); // "Rafael"
```

Destructuración

Para destructurar un objeto, se usan { **llaves** }.

Para destructurar un array, se usan [**corchetes**].

Destructuración

Esto sigue la noción que un objeto se crea con { **llaves** }, mientras que un array se crea con [**corchetes**].

Destructuración

```
const users = ["Ismael", "Rafael", "Otroel"]; // array
const user = { nombre: "Ismael", altura: 1.75 }; // objeto
```

Destructuración Listas

```
// OPCION A
const users = ["Ismael", "Rafael", "Otroel"];

const [user_0, user_1] = users;
```

```
// OPCION B
const users = ["Ismael", "Rafael", "Otroel"];

const user_0 = users[0];
const user_1 = users[1];
```

Destructuracion Listas

Y si quiero el elemento #1, pero no el #0?

```
// OPCION B
const users = ["Ismael", "Rafael", "Otroel"];

const user_1 = users[1];
```

```
// OPCION A
const users = ["Ismael", "Rafael", "Otroel"];

const [_, user_1] = users;
```

Valor Predeterminado

```
const users = ["Ismael"];

const [user_0, user_1 = "Usuario 2"] = users;

console.log(user_0); // "Ismael"
console.log(user_1); // "Usuario 2"
```

Spread Operator

```
const users = ["Ismael", "Rafael", "Otroel"];

const [user_0, ...otrosUsuarios] = users;

console.log(otrosUsuarios.length); // ??
console.log(otrosUsuarios[0]); // ??
```

Spread Operator

```
const users = ["Ismael", "Rafael", "Otroel"];

const [user_0, ...otrosUsuarios] = users;

console.log(otrosUsuarios.length); // 2
console.log(otrosUsuarios[0]); // "Rafael"
```

Asignación Avanzada

Asignación Avanzada - Objetos

```
const nombre = "Ismael";
const altura = 1.75;

const user1 = {
  nombre: nombre,
  altura: altura,
};
```

Asignación Avanzada - Brevedad

```
const nombre = "Ismael";
const altura = 1.75;

const user1 = {
  nombre,
  altura,
};
```

Asignación Avanzada - Renombrar

```
const nombre = "Ismael";
const altura = 1.75;

const user1 = {
  primerNombre: nombre,
  altura,
};
```

Asignación Avanzada - Spread Op.

```
const userDefault = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};

const user1 = {
  ...userDefault,
};
```

Asignación Avanzada - Spread Op.

```
const userDefault = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};

const user1 = {
  userDefault,
};
// user1.userDefault.nombre
```

Asignación Avanzada - Sobreescritura

```
const userPredeterminado = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};

const user1 = {
  ...userPredeterminado,
  nombre: "Rafael",
};
```

Asignación Avanzada - Sobreescritura

```
const userPredeterminado = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};

const user1 = {
  nombre: "Rafael",
  ...userPredeterminado,
};
```

Asignación Avanzada - Sobreescritura

```
const userPredeterminado = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};
const nombre = "Rafael";

const user1 = {
  ...userPredeterminado,
  nombre: nombre,
};
```

Asignación Avanzada - Sobreescritura

```
const userPredeterminado = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};
const nombre = "Rafael";

const user1 = {
  ...userPredeterminado,
  nombre,
};
```

Asignación Avanzada - Multi Spread

```
const userPredeterminado = {
  nombre: "Usuario",
  altura: 1.75,
  puntaje: 0,
};

const adminPredeterminado = {
  nombre: "Admin",
};

const nombre = "Rafael";

const user1 = {
  ...userPredeterminado,
  ...adminPredeterminado,
  nombre,
};
```

Asignación Avanzada - Listas

```
const user_1 = "Ismael";
const user_2 = "Rafael";
const users = [user_1, user_2];

console.log(users[0]); // ??
```

Asignación Avanzada - Listas

```
const user_1 = "Ismael";
const user_2 = "Rafael";
const users = [user_1, user_2];

console.log(users[0]); // "Ismael"
```

Asignación Avanzada - Listas

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [user_1, user_2];

console.log(users[0]); // ??
```

Asignación Avanzada - Listas

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [user_1, user_2];

console.log(users[0]); // ["Ismael", "Cael"]
```

Asignación Avanzada - Spread Op.

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [...user_1, user_2];

console.log(users[0]); // ??
```

Asignación Avanzada - Spread Op.

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [...user_1, user_2];

console.log(users[0]); // "Ismael"
```

Asignación Avanzada - Sobreescritura

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [user_2, ...user_1];

console.log(users[0]); // ??
```

Asignación Avanzada - Sobreescritura

```
const user_1 = ["Ismael", "Cael"];
const user_2 = "Rafael";
const users = [user_2, ...user_1];

console.log(users[0]); // ["Rafael"]
```

Asignación Avanzada - Sobreescritura

```
const user_1 = ["Ismael", "Cael"];
const user_2 = ["Rafael"];
const users = [...user_2, ...user_1];

console.log(users[0]); // "Rafael"
```

Son expertos en JS!