Javascript Avanzado II

Metodos de listas/vectores/arrays/etc

.flat()

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

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

.join()

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

console.log(numbers.join(", ")); // "4, 5, 2, 1, 1, 7"
```

.find()

```
function findFunction(item) {
  return item === 4;
}

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

console.log(numbers.find(findFunction)); // 4
```

.find()

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

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

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

.find()

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

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

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.

.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 "mappeo", y retorna los elementos resultantes de la funcion de "mappeo".

.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 "mappeo", y retorna los elementos resultantes de la funcion de "mappeo".

No puede ser utilizada para filtrar.

.sort()

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

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

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

.sort()

```
function sortFunction(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 funcion de "ordenamiento", y retorna los elementos ordenados segun esa funcion.

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 insercion.

.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 reduccion, y retorna el resultado de reducir el array usando la funcion.

.reduce()

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

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

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

.reduce()

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

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

console.log(numbers.reduce(reduceFunction, 1)); // 20
```

Intermision

Destructuracion Objetos

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

const { nombre } = user;

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

Destructuracion Objetos

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

const { nombre } = user;

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

Destructuracion Objetos

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

const { nombre, altura } = user;

console.log(nombre); // "Ismael"
console.log(altura); // 1.75
```

Destructuracion Objetos

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

const { nombre, altura } = user;

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

Destructuracion Objetos

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

```
// OPCION A
const { nombre, altura } = user;
```

```
// OPCION B
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 variable

```
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
```

Destructuracion Objetos

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

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

```
// ??
```

Intermision

Destructuracion 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"
```

Destructuracion

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

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

Destructuracion

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

Destructuracion

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

Destructuracion Listas

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

```
// OPCION A
const [user_0, user_1] = users;
```

```
// OPCION B
const user_0 = users[0];
const user_1 = users[1];
```

Destructuracion Listas

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

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

```
// OPCION A
const [_, user_1] = users;
```

```
// OPCION B
const user_1 = users[1];
```

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"
```

Asignacion Avanzada

Asignacion Avanzada - Objetos

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

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

Asignacion Avanzada - Brevedad

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

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

Asignacion Avanzada - Renombrar

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

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

Asignacion Avanzada - Spread Op.

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

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

Asignacion Avanzada - Spread Op.

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Multi Spread

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

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

Asignacion Avanzada - Listas

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

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

Asignacion Avanzada - Listas

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

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

Asignacion Avanzada - Listas

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

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

Asignacion Avanzada - Listas

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

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

Asignacion Avanzada - Spread Op

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

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

Asignacion Avanzada - Spread Op

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Sobreescritura

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

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

Asignacion Avanzada - Sobreescritura

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

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