## Modules :-

modules are reusable pieces of code that encapsulate related functionality and can be exported from one file and imported into another. This allows for better organization, maintainability, and reusability of code in large applications. ES6 (ECMAScript 2015) introduced native support for modules.

Note:- mention type=”module” in script tag to work with the modules

```
<script type="module">
import message from "./message.js";
</script>
```

There are two types of exports: **Named Exports** and **Default Exports**.

### 1)Named export

```
const name = "Jesse";
const age = 40;

export {name, age};
```

Named import

```
import { name, age } from "./person.js";
```

### 2)Default Export

```
export default  message = () => {
const name = "Jesse";
const age = 40;
return name + ' is ' + age + 'years old.';
};
```

Default import

```
import message from "./message.js";
```

# sets and maps

## Sets in JavaScript

A Set is a collection of unique values. This means that no value can occur more than once in a set. Sets allow you to store distinct values of any type, whether primitive or object references.

**Key Characteristics of Sets:**

- **Unique Values:** No duplicates are allowed. If you try to add the same value multiple times, it will only be added once.
- **Order of Values:** Insertion order is maintained. The elements are iterated in the order of their insertion.

- **No Key-Value Pairs:** Unlike objects or maps, a set is just a collection of values (no keys).

### Common Methods for Sets:

1. **add(value):** Adds a new value to the set.
2. **delete(value):** Removes the specified value from the set.
3. **has(value):** Checks if the value exists in the set.
4. **clear():** Removes all values from the set.
5. **size:** Returns the number of values in the set.
6. **forEach():** Iterates over the values in the set.
7. **values():** Returns an iterator over the set's values.
8. **keys():** Identical to values(), as sets don't have keys.
9. **entries():** Returns an iterator of [value, value] pairs, mimicking the [key, value] behavior of maps.

**Example:**

```
let mySet = new Set();

  // Add values
  mySet.add(1);
  mySet.add(2);
  mySet.add(2);  // Won't be added, as 2 is already present
  mySet.add('hello');

  // Check if a value exists
  console.log(mySet.has(1));  // true
  console.log(mySet.size);    // 3

  // Remove a value
  mySet.delete(2);

  // Iterate over the set
  mySet.forEach(value => console.log(value));
```

# Maps in JavaScript

A Map is a collection of key-value pairs, similar to an object. However, maps have some key differences that make them more useful in specific scenarios.

**Key Characteristics of Maps:**

- Key-Value Pairs: Each element in a map is stored as a pair of keys and values.
- Any Data Type as Keys: Unlike objects, where keys are strings or symbols, in maps, any type (primitive or object) can be used as a key.
- Map Size: You can easily determine the number of entries in a map using the size property.

## Common Methods for Maps:

1. **set(key, value):** Adds a key-value pair to the map or updates an existing key.
2. **get(key):** Returns the value associated with the key.
3. **has(key):** Checks if the map contains the specified key.
4. **delete(key):** Removes the key and its associated value from the map.
5. **clear():** Removes all key-value pairs from the map.
6. **size:** Returns the number of key-value pairs in the map.
7. **forEach():** Iterates over the key-value pairs in the map.
8. **keys():** Returns an iterator over the keys of the Map.
9. **values():** Returns an iterator over the values of the Map.
10. **entries():** Returns an iterator over [key, value] pairs.

## Example:

```
let myMap = new Map();

// Add key-value pairs
myMap.set('name', 'John');
myMap.set(1, 'one');
myMap.set(true, 'boolean value');

// Get values
console.log(myMap.get('name'));  // 'John'
console.log(myMap.get(1));       // 'one'

// Check for existence of a key
console.log(myMap.has(true));    // true

// Remove a key-value pair
myMap.delete('name');

// Iterate over the map
myMap.forEach((value, key) => {
console.log(key, value);
});
```