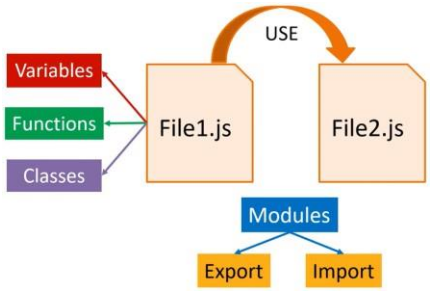
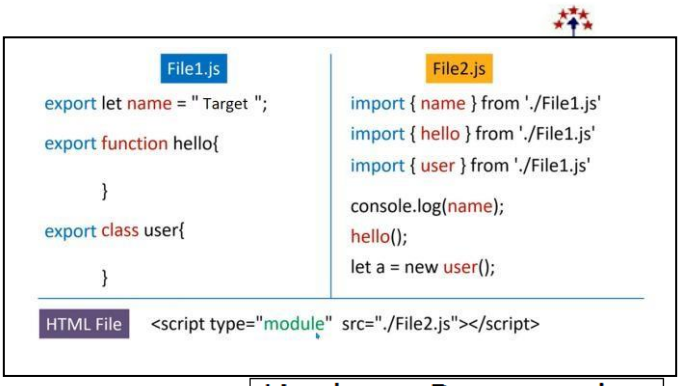
MODULE

In JavaScript, modules are a way to organize code into separate files or components. ***With modules, you can package up related functions, variables, and classes together and give them a name, so you can easily reuse them in different parts of your program.***

Prior to ECMAScript 6 (ES6), JavaScript did not have native support for modules, and developers relied on various patterns like the revealing module pattern or the CommonJS format to achieve modularity. However, with the introduction of ES6 in 2015, JavaScript gained native support for modules using the ***import and export keywords.***





ES6 Modules: This is the modern standard for JavaScript modules, defined in ECMAScript 2015 (ES6) specification. ***ES6 modules use import and export statements to define dependencies*** and expose functionality between different modules.

e.g.1

File1.js

```
export let message = "ES6 Modules";
```

file2.js

```
import {message} from "./file1.js";
//console.log(message);
document.body.innerHTML=message;
```

index.html

```
<script type="module" src="./file2.js"></script>
```

e.g.2

File1.js

```
export let message = "ES6 Modules";
export function user(){
console.log("Hello Everyone");|
}
```

file2.js

```
import {message, user} from "./file1.js";
console.log(message);
user();
```

index.html

```
<script type="module" src="./file2.js"></script>
```

MAKING THE ALIAS NAME

To make an alias while importing a module in JavaScript, specifically with ES6 modules, you can use the as keyword followed by the alias name.

e.g.3

File1.js

```
export let message = "ES6 Modules";
export function user(){
console.log("Hello Everyone");|
}
```

file2.js

```
import {message as msg, user as fun} from "./file1.js";
console.log(msg);
fun();
```

index.html

```
<script type="module" src="./file2.js"></script>
```

IMPORTING ALL ELEMENTS OF MODULE

To import all elements from a module in JavaScript, you can use the * as syntax. This imports all exported elements from the module and assigns them to a single variable.

e.g.4

File1.js

```
export let message = "ES6 Modules";
export function user(){
console.log("Hello Everyone");|
}
```

file2.js

```
import * as target from "./file1.js";
console.log(target.message);
target.user();
```

index.html

```
<script type="module" src="./file2.js"></script>
```


EXPORT DEFAULT AND IMPORT DEFAULT ELEMENT OF MODULE

Export Default

With export default, you can export a single value (function, object, class, etc.) as the default export from a module. This is particularly useful when you have only one main thing to export from a module.

e.g.4

File1.js

```
const add = (a, b) => a + b;
const subtract = (a, b) => a - b;

export default {
  add,
  subtract
};
```

file2.js

```
import math from './file1.js';
console.log(math.add(5, 2)); // Outputs: 7
console.log(math.subtract(5, 2)); // Outputs: 3
```

index.html

```
<script type="module" src="./file2.js"></script>
```

EXPORTING ELEMENT OF MODULE SPECIFICALLY

e.g.4

File1.js

```
let message = "ES6 Modules";
function user(){
console.log("Hello Everyone")
}
export {user}
```

file2.js

```
import {user} from "./file1.js";
user();
```

index.html

```
<script type="module" src="./file2.js"></script>
```

EXPORT V/S EXPORT DEFAULT

In JavaScript modules, export and export default are used to expose functionality from a module to be used in other parts of a program.

export

- With export, you can export one or more named values from a module.
- You can export variables, functions, classes, or any other JavaScript entity.
- You can have multiple named exports in a single module.
- Named exports are imported using the same name they were exported with, and they must be imported using curly braces {}.

e.g.

math.js

```
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
```

```
import {add , subtract } from "./ math.js";
console.log(msg);
```

```
fun();
```

export default

- With export default, you can export a single value as the default export from a module.
- You can export a variable, function, class, or any other JavaScript entity as the default export.
- A module can have only one default export.
- The default export can be imported using any name in the importing module, and it doesn't require curly braces {}.

e.g.

math.js

```
const add = (a, b) => a + b;
const subtract = (a, b) => a - b;
export default {
  add,
  subtract
};
```

```
import math from './math.js';
```

Akhilesh Kumar Gupta
(Technical Training specialist)
TechCarrel LLP