give me an overview of ES6 import and export didderent functionalities with example

In ES6 (ECMAScript 2015), **import** and **export** are used for **modular JavaScript**, allowing code to be split into reusable pieces across multiple files. This modular structure makes your code easier to maintain and test.

Here's an overview of the functionalities with examples:

---

1. Named Export and Import

You can export multiple functions, variables, or objects from a file and import only the ones you need.

Exporting:

```javascript
// mathUtils.js
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;
export const PI = 3.14159;
```

Importing:

```javascript
// main.js
import { add, PI } from './mathUtils.js';

console.log(add(5, 3)); // 8
console.log(PI);        // 3.14159
```

> **Note: You must use the same names while importing.**

---

2. Default Export and Import

A file can have one **default export**, which can be imported without curly braces and renamed on import.

Exporting:

```javascript
// greeting.js
const greet = (name) => `Hello, ${name}!`;
export default greet;
```

Importing:

```javascript
// main.js
import greet from './greeting.js';

console.log(greet('Nahian')); // Hello, Nahian!
```

> **Default exports are useful when a module exports a single primary functionality.**

---

3. Combining Named and Default Exports

You can mix **default** and **named exports** in the same file.

Exporting:

```javascript
// shapes.js
export const square = (x) => x * x;
export const circleArea = (radius) => Math.PI * radius * radius;

const shapesInfo = "Basic shape utilities";
export default shapesInfo;
```

Importing:

```javascript
// main.js
import shapesInfo, { square, circleArea } from './shapes.js';

console.log(shapesInfo);        // Basic shape utilities
console.log(square(4));         // 16
console.log(circleArea(3));     // 28.274333882308138
```

---

4. Renaming Exports and Imports

You can rename exports and imports using the `as` keyword.

Exporting:

```javascript
// data.js
export const fetchData = () => { /* fetching logic */ };
export const saveData = () => { /* saving logic */ };
```

Importing:

```javascript
// main.js
import { fetchData as getData, saveData as storeData } from './data.js';

getData();
storeData();
```

---

5. Import All (`* as`)

You can import all named exports from a module as an object.

Exporting:

```javascript
// utilities.js
export const logInfo = (message) => console.log(`Info: ${message}`);
export const logError = (error) => console.error(`Error: ${error}`);
```

Importing:

```javascript
// main.js
import * as utils from './utilities.js';

utils.logInfo('Everything is running smoothly.');
utils.logError('Something went wrong.');
```

---

6. Re-exporting (Export from Another Module)

You can re-export modules without explicitly importing them into the current file.

Re-exporting:

```javascript
// allMath.js
export { add, subtract } from './mathUtils.js';
export { square } from './shapes.js';
```

Importing from re-exported module:

```javascript
// main.js
import { add, square } from './allMath.js';

console.log(add(2, 3)); // 5
console.log(square(3)); // 9
```

---

7. Dynamic Imports (Lazy Loading)

Dynamic `import()` lets you load modules on demand, improving performance by splitting the code into chunks.

Using Dynamic Import:

```javascript
// main.js
const loadModule = async () => {
  const module = await import('./mathUtils.js');
  console.log(module.add(5, 6)); // 11
};

loadModule();
```

> **Useful for code splitting and lazy loading in modern applications.**

Summary Table:

Functionality	Export Syntax	Import Syntax
Named Export	`export const name = value;`	`import { name } from './file';`
Default Export	`export default value;`	`import name from './file';`
Renaming Imports/Exports	`export { name as alias };`	`import { name as alias } ...;`
Import All	-	`import * as obj from './file';`
Re-exporting	`export { name } from './file';`	-
Dynamic Import	-	`import('./file.js')`

These features together make ES6 modules extremely flexible for modern web development.