JavaScript Part 3
Modules

Samuel Cho, Ph.D.

NKU ASE/CS

1. Require/Exports
2. Import/Export

Require/Exports

Module

- The concept of a module was introduced by Node.js when JavaScript supported only a single file script.
- Node.js module has the ‘module.exports‘ variable that has the information about the module.

module.exports

- A module can have functions.
- We can select the functions that are exported (line 6).

Code #arithmetic.js

```
1 const add = (a, b) => a + b; // We don't export this
2 const mul = (a, b) => {
3     ...
4 };
5
6 module.exports.mul = mul;
```

- The 'console.log(module)' command prints out the properties of the module.
- The 'module.exports' is a dictionary that maps from the exported function name to its body.

Command > console.log(module)

```
{
  ...
  exports: { mul: [Function: mul] },
  ...
}
```

require

- To use the module's functions, we use 'require' (line 1) to access the exported dictionary.
- The dictionary reference (arith) is used to call the method (line 2).

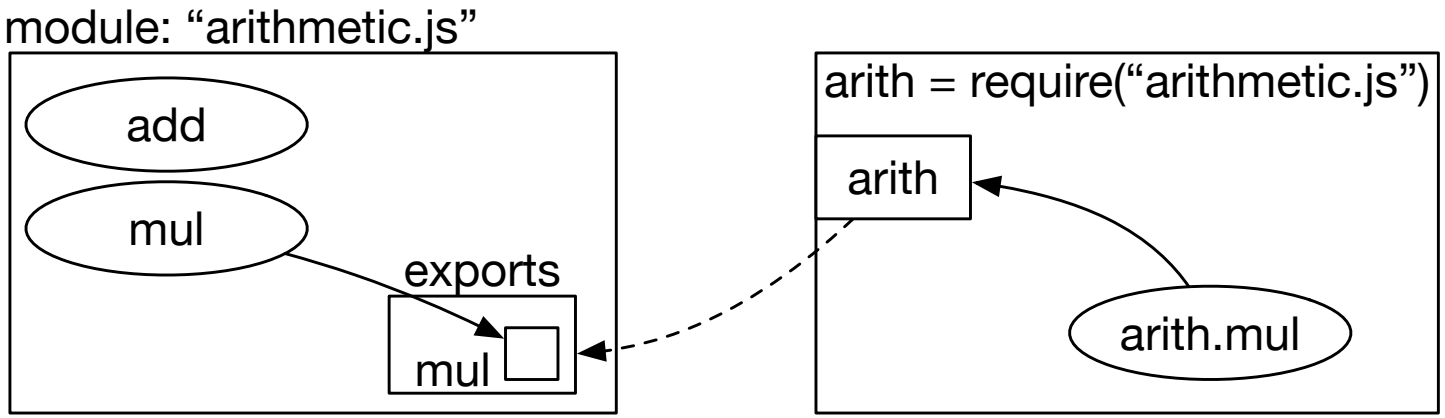
Code #a.js

```
1 const arith = require('./
    ↪ arithmetic');
2 console.log(arith.mul(3, 7));
```

Command > node a.js

```
1 > node run.js
2 21
```

- This diagram shows the relationship between the module and its user.
- Notice that we can use both ‘arithmetic’ or ‘arithemtic.js’ for the module name.



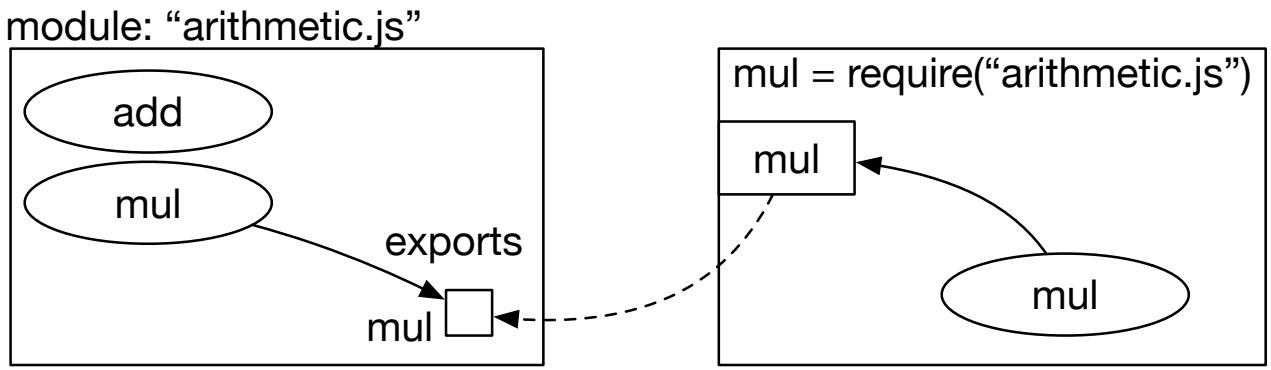
Single Function Export

- When we export only one function, we can make the value of 'module.exports' as the function name, not a dictionary.

Code #

```
1 // In arithmetic2.js
2 module.exports = mul;
3
4 // In run2.js
5 const mul = require('./arithmetic2.js');
6 console.log(mul(3, 7));
```

- This diagram shows the relationship between the module and its user when a single function is exported.
- Notice that the 'module.exports' is not a dictionary anymore.



Multiple Functions Export

- We can export multiple functions.

Code #arithemtic3.js

```
1 const add = (x, y) => x + y
2 const sub = (x, y) => x - y
3
4 module.exports.add = add
5 module.exports.sub = sub
```

- Or, we can simply export multiple functions.

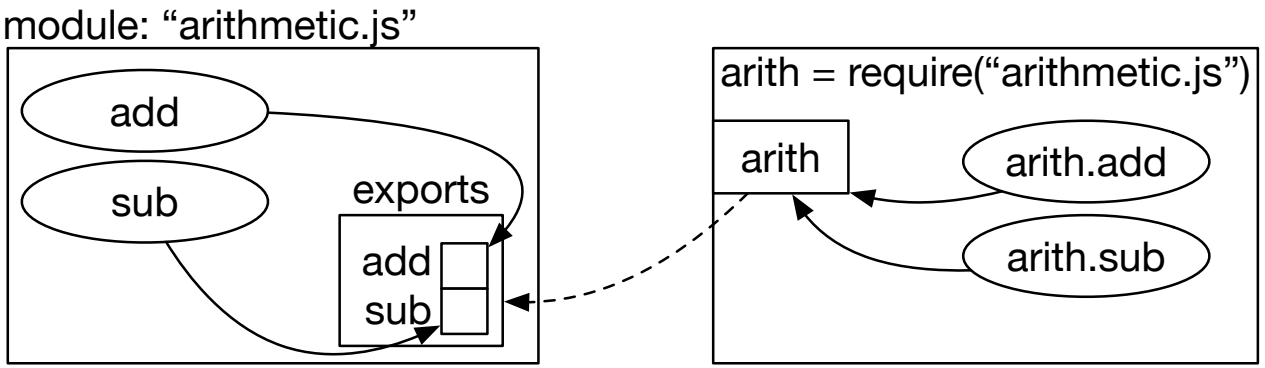
Code #

```
1 module.exports = {add, sub}
```

- To use the exported functions, we can use 'require' to access the functions.

Code #c.js

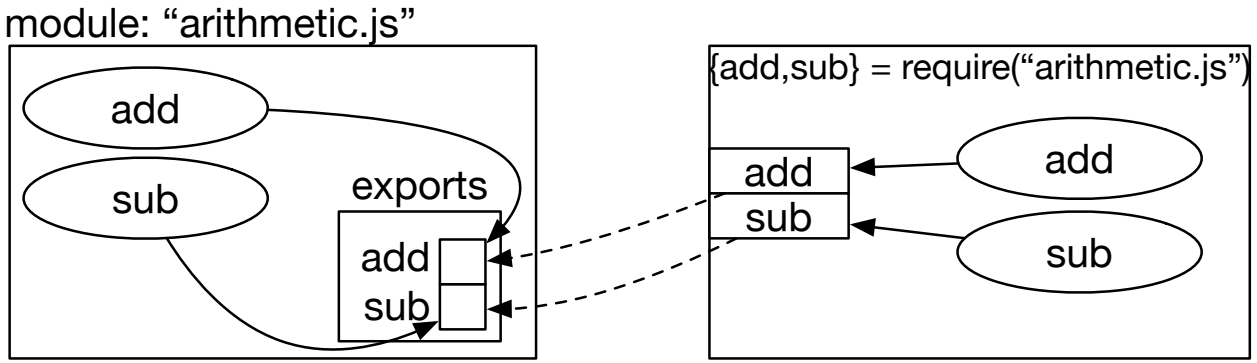
```
1 const arith = require('./arithmetic3.js');
2 console.log(arith.add(10, 20))
3 console.log(arith.sub(10, 20))
```



- When we use a block { ... } with 'require,' we can use add/sub to reference the module's add/sub functions.

Code #

```
1 const {add, sub} = require('./arithmatic4');
2 console.log(add(10, 20))
3 console.log(sub(10, 20))
```



Import/Export

JavaScript's Module

- In ES6, JavaScript supports the module system.
- It uses a different syntax than that of Node.js.

package.json

- Also, we should notify Node.js that we are using a JavaScript module, not Node.js using the package.json.
- Without the package.json with this data, Node.js cannot use JavaScript modules.

Code #package.json

```
1 {
2     "type": "module"
3 }
```

Export

- In this module system, we can specify what functions are exported by prepending 'export.'

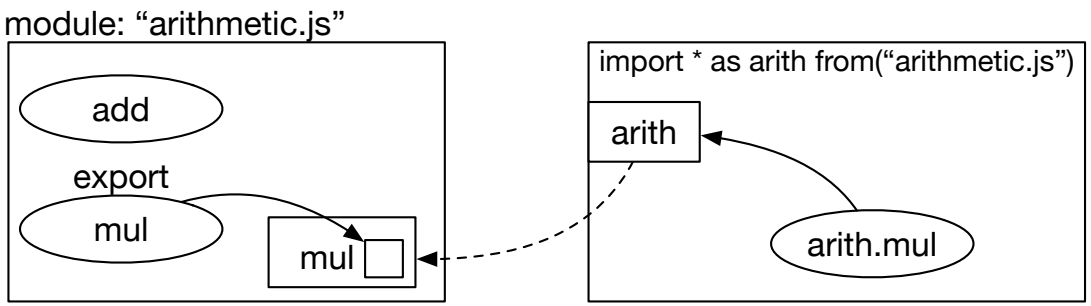
Code #

```
1 const add = (a, b) => { // not exporting the function
2     return a + b;
3 };
4
5 export const mul = (a, b) => { // export the function
6     ...
7     return result
8 }
```

- To access the functions, we use the 'import * as name from' syntax.
- We use the 'name' (arith in this example) to access the export dictionary.

Code #

```
1 import * as arith from './arithmetic.js';
2
3 const result = arith.mul(3, 7);
4 console.log(result);
```



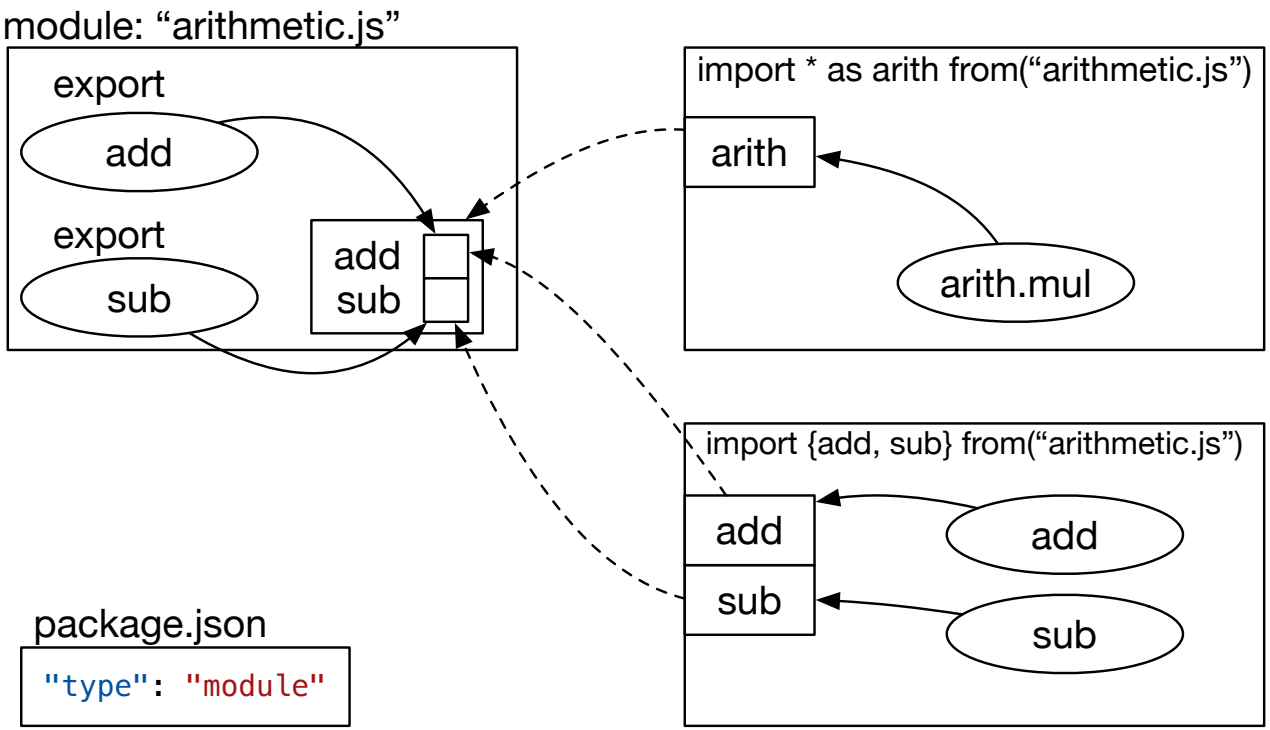
Multiple Functions Export

- We can export multiple functions.
- We can access them using a dictionary reference arith (lines 5 – 6) or name access (lines 8 – 9).

Code #arithemtic3.js

```
1 // arithmetic.js
2 export const add = (a, b) => a + b;
3 export const sub = (a, b) => a - b;
4 // run.js
5 import * as arith from './arithmetic2.js';
6 const sumResult = arith.add(3, 7);
7
8 import { add, sub } from './arithmetic2.js';
9 const sumResult = add(3, 7);
```

- This diagram shows the relationship when we use export multiple functions.



Single Function Export

- When we want to export only one function, we can use 'default export' (line 2).
- Then, we can directly access the function (lines 4 – 5).

Code #

```
1 const mul = (a, b) => { ... }
2 export default mul;
3
4 import mul from './arithmetic.js';
5 const result = mul(3, 7);
```

- This diagram shows the relationship when we use ‘export default.’