What is the difference between import and require?

1. **When using `import ... from ...`, the module path must be a string literal. When using `require`, the module path can be dynamic.**

For example, this works:

```
const name = "module2";
const obj = require(`./${name}`);
```

But this will result in the error `SyntaxError: Unexpected template string` when run with `node`.

```
const name = "module2";
import { func } from `./${name}`;
```

Code like `import { func } from ("./partial/path" + someVariable);` will also result in a syntax error.

Why is this? See the next point.

2. Order of execution differs. `require` will be run inline, after the code above it has executed. `import` runs before the rest of the script.

Assuming `module2.js` has `console.log("require module2");` at the top, then if we run this code:

```
console.log("require module1");const obj =
require("./module2");
console.log(`module2 = ${obj.module2}`);
```

it results in the following:

```
require module1
require module2
module2 = require module2
```

With ES modules, on the other hand...

```
console.log("require module1");import module2 from
"./module2.js";
console.log(`module2 = ${module2}`);
```

3. You can leave out a `.js` extension when importing a local module with `require`, but cannot do the same when using `import`.

This is true by default in the browser and Node.js. For example, `require("./module2")` works, but the equivalent using `import` must be written as `import module2 from "./module2.js"`. If you omit the extension in Node.js, you will get an error like: `Error [ERR_MODULE_NOT_FOUND]: Cannot find module …`

In Node.js, you can use the `--experimental-specifier-resolution=node` option to circumvent this behavior, i.e. this will allow you to omit `.js` extensions when using `import`.

Furthermore, webpack has an option that changes this behavior. Specifically, if `resolve.enforceExtension` is `true`, then extensions are required. This option is set to `false` by default, which explains why in many frameworks (like Next.js, which uses webpack behind the scenes) you can use `import` without specifying file extensions.