JS Import of External Modules

Module systems

The **JavaScript** node supports the ES6 (ECMAScript 2015) and CommonJS module systems:

- For ES6, the node's code is considered the root module.
- Both static and dynamic import of ES6 modules is supported.
- To import CommonJS modules, use the `require` function (see Full API Description).

▼ Examples

The ES6 module system

External module `"./foo/module/module.js"` :

```
function sayHello() {
    return "Hello";
}
export { sayHello };
```

External module `"./foo/foo.js"` :

```
import { sayHello } from "./module/module.js";
function cube(x) {
    return x * x * x;
}
export { cube, sayHello };
```

Implementing modules into JS script:

```
// Static import
import { cube, sayHello } from "foo/foo.js";
console.log(sayHello());
console.log('3^3 = ', cube(3));

// Dynamic import
import("foo/foo.js").then(mod => {
     console.log(mod.sayHello());
     console.log('3^3 = ', mod.cube(3));
}).catch(e => {
     console.error(e);
});
// or
(async () => {
     try {
        const mod = await import("foo/foo.js");
        console.log(mod.sayHello());
        console.log('3^3 = ', mod.cube(3));
     } catch(e) {
        console.error(e);
     }
})();
```

The CommonJS module system

External module `"./foo/module/module.js"` :

```
function sayHello() {
    return "Hello";
}
module.exports = sayHello;
```

External module `"./foo/foo.js"` :

```
const sayHello = require("./module/module.js");
function cube(x) {
    return x * x * x;
}
exports.sayHello = sayHello;
exports.cube = cube;
```

Implementing modules into JS script:

```
const foo = require("foo/foo.js");
console.log(foo.sayHello());
console.log('3^3 = ', foo.cube(3));
```

External modules can be encoded in UTF-8 or UTF-16 Little Endian with BOM.

The CommonJS modules operate according to the specification, except for the following:

- The optional properties `require.main`, `require.paths`, and `module.uri` are not included.
- Additional optional properties `require.resolve` and `require.cache` are included (like in NodeJS).
- Module objects have properties `parent`, `loaded`, and `filename` (like in NodeJS).
- The global variable `__filename` is available within the module. It contains the absolute path to the module in the file storage.

▼ Example

External module `"child_module.js"`:

```
console.log("Hello! I am " + __filename);
exports.filename = module.filename; // returns the full path to
child_module.js
exports.parent = module.parent.id;  // returns the identifier of the
calling module
exports.loaded = module.loaded;     // returns true or false, showing
if the module was loaded
```

Using `"child_module.js"` in JS script within the node:

```
// require.resolve:
// in server editions, returns the full path to the module in the file
storage
// in desktop editions, returns the full path to the module in the
file system
let path = require.resolve("child_module.js");
console.log(path);
// Calling external module of CommonJS system
let childModule = require("child_module.js");
console.log(childModule.filename);
console.log(childModule.parent);
console.log(childModule.loaded);
// Clearing the cache of "child_module.js"
delete require.cache[path];
// and calling the external module once again,
// which results in displaying "Hello! I am ..." again.
// This won't happen without clearing the cache
require("child_module.js");
```

Only CommonJS modules can be loaded from CommonJS modules.

Import paths of external modules

External modules can be imported using relative or absolute paths.

Relative path

When using relative paths in the root module (the script in the node's code editor):

- If the package is saved: Use a path relative to the package location.
- If the package is unsaved: Use a path relative to the user directory.

Relative paths to external modules should start from the location of the importing module.

For example:

```
import { cube, foo, sayHello } from "./foo/foo.js";
```

Absolute path

On the Megaladata server

A path starting with "/" is interpreted as an absolute path within the file storage, starting with the user directory. For example:

```
import { cube, foo, sayHello } from "/user/foo/foo.js";
```

In desktop editions

The absolute path is a complete path in the file system. For example:

```
import { cube, foo, sayHello } from "C:\\Script\\foo\\foo.js";
```

Read on: JavaScript: Input Datasets