JavaScript応用実習

06.モジュールシステム

株式会社ジードライブ

今回学ぶこと

- モジュールシステム
  - エクスポート
  - インポート
- モジュールバンドラ

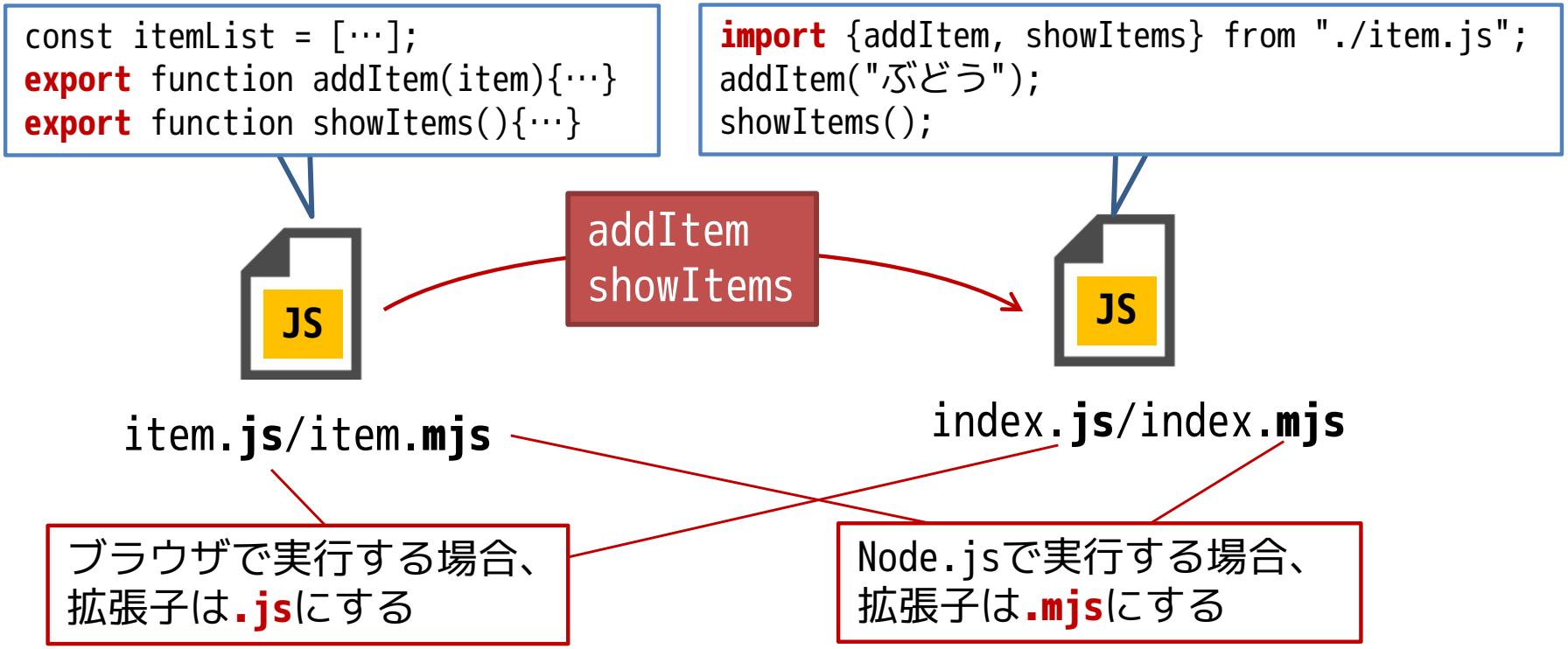
モジュールシステムとは

- あるファイルの中で定義した関数やクラスなどを、別のファイルで読み込み、利用するためのしくみ
  - モジュール： プログラムの部品となる関数やクラスなど
- JavaScriptでは、おもに２つのモジュールシステム(CommonJSとES Modules)が使用されている
  - CommonJSは、サーバーサイドJavaScriptのために開発されたシステムで、ブラウザ用JavaScriptでは使用できない
  - 後発のES Modulesは、サーバーサイドとブラウザの両方に対応している

本講座ではES Modulesについて学習する

export / import

- ES Modulesでは、外部で使用するためのモジュールに対して**export**というキーワードを付与する
  - exportされたモジュールは**import**して利用することができる
  - 2種類のエクスポート(名前付きとデフォルト)方式が存在する



名前付きエクスポート

- 外部で使用するためのモジュールの前にexportというキーワードを付ける

エクスポートされない

エクスポートされる

item.js/item.mjs

```
const itemList = ["りんご", "みかん", "バナナ"];

export function addItem(item) {
  itemList.push(item);
}

export function showItems() {
  itemList.forEach((item) => console.log(item));
}
```

名前付きエクスポート

- エクスポートするものをまとめて列挙することも可能

エクスポートされない

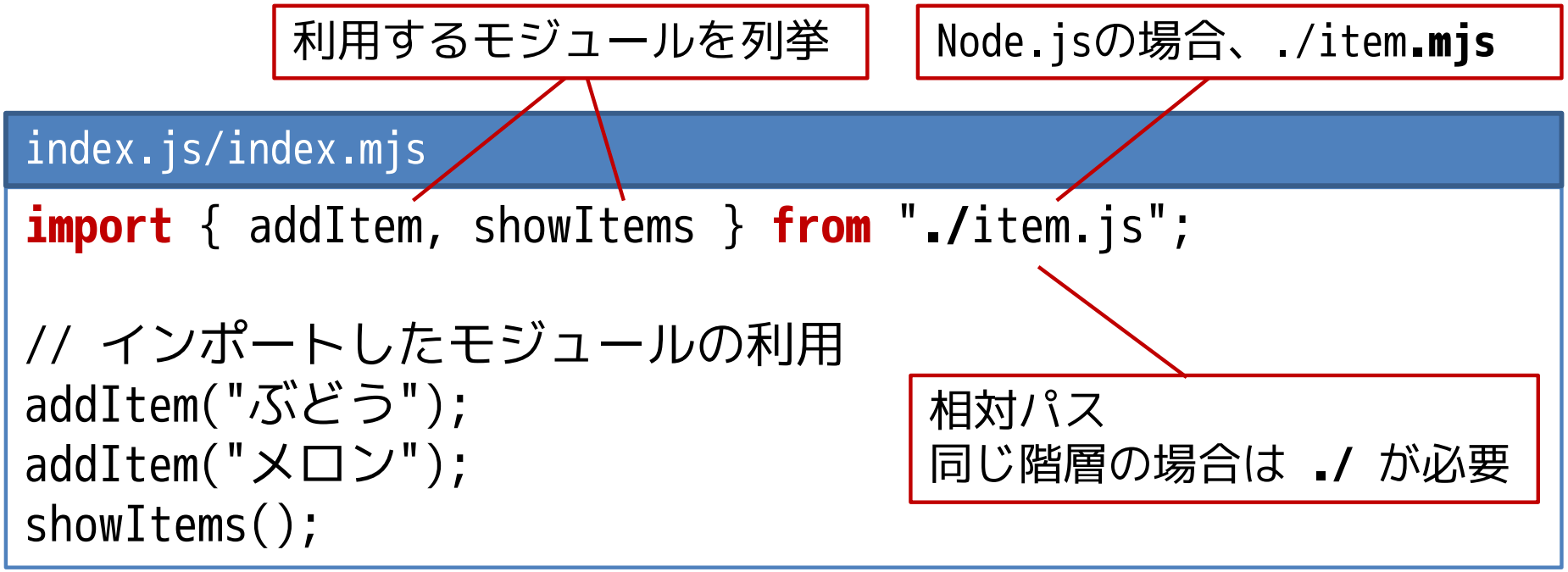
エクスポートされる

item.js/item.mjs

```
const itemList = ["りんご", "みかん", "バナナ"];

function addItem(item) {
  itemList.push(item);
}

function showItems() {
  itemList.forEach((item) => console.log(item));
}

export { addItem, showItems };
```

インポート

- モジュールを利用する側では、import～fromという構文を使い、利用したいモジュールとファイルパスを記述する



scriptタグの記述

- ES Modulesを利用するJavaScriptをscriptタグで読み込む場合は、type属性の値をmoduleに設定する

```HTML
<script src="js/index.js" type="modlue"></script>
```

ES Modulesを利用するJavaScriptファイル

asキーワード

- エクスポート、またはインポートするにあたって、asキーワードでモジュール名を変更することができる
  - 長いモジュール名を短くしたり、モジュール名の重複を防ぐことができる

エクスポート時に名前を変更する場合

```
export { addItem as add, showItems as show };
```

インポート時に名前を変更する場合

```
import { addItem as add, showItems as show } from "./item.js";
```

デフォルトエクスポート

- defaultキーワードを付けてエクスポートされたモジュールは、インポート時に { } を付けない
  - defaultキーワードを付けてエクスポートできるモジュールは１ファイルにつき１つだけ
  - インポート時に自由に命名できる

エクスポート側

```
export { addItem };
export default showItems;
```

インポート側

```
import show, { addItem } from "./item.js";
```

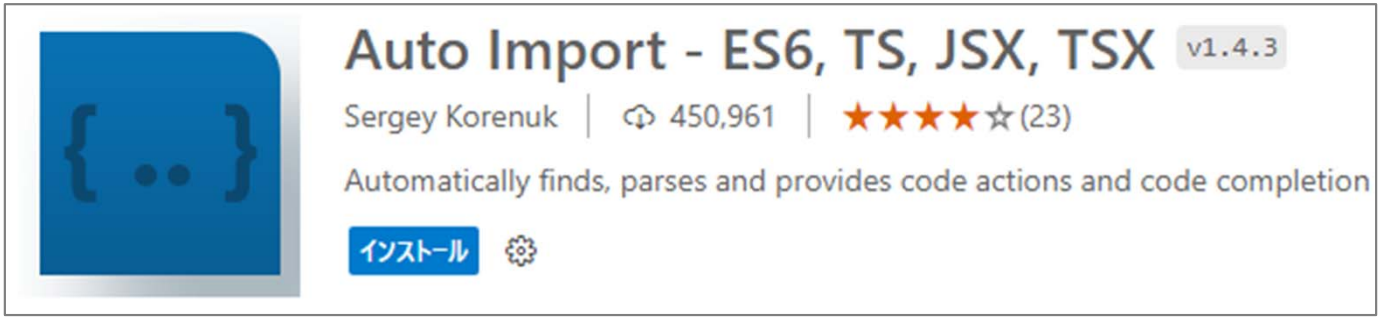
{}を付けない。任意の名前を設定できる

モジュールの特徴・ルール

- モジュール内のコードは常にStrictモードで実行される
- export文が使えるのはモジュール内のトップレベルスコープ(最上位階層)のみ
  - 関数の中などではexport文は使えない
- letやvarで宣言された変数をインポートした場合でも、インポートした側ではその変数への再代入はできない(constと同じ扱い)

拡張機能: インポートの入力補助

- VS Codeに以下のような拡張機能を追加しておくと、モジュールのインポートに対し、入力補助を得ることができる

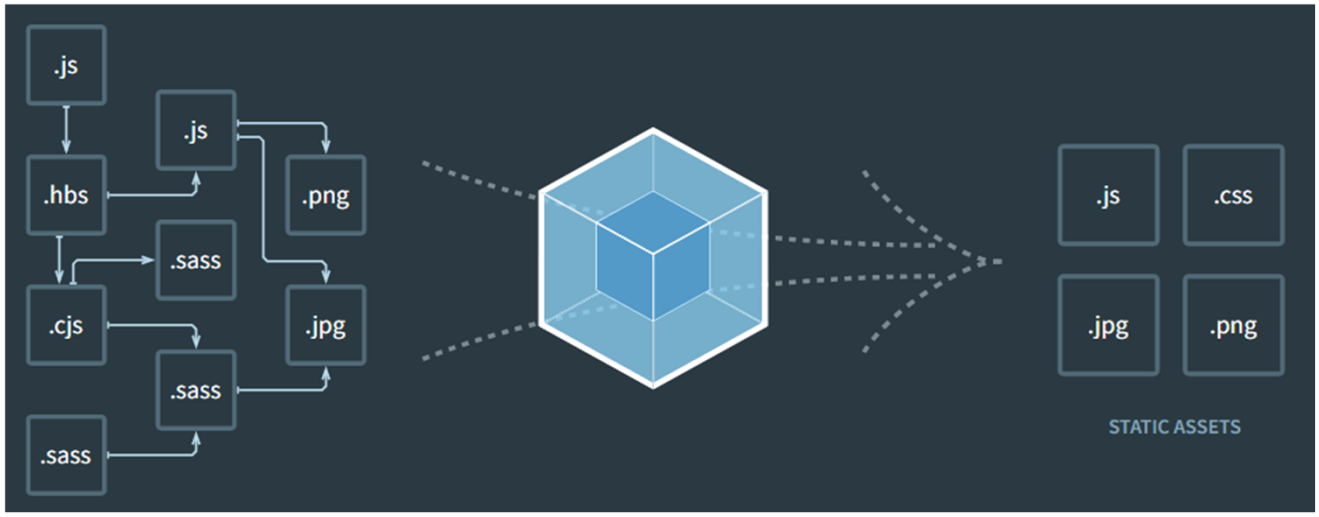


練習

- 練習06-1

Webpackとは

- Webpackはモジュールバンドラと呼ばれるツールで、複数のモジュールを１つにまとめる(バンドルする)ことができる
  - 複数のJavaScriptファイルをまとめることができ、HTMLから読み込む際のリクエストを削減することができる
  - npmで管理しているパッケージもバンドルできる
  - ローダーを利用することで、TypeScriptやCSSなどJavaScript以外のファイルもまとめることができる



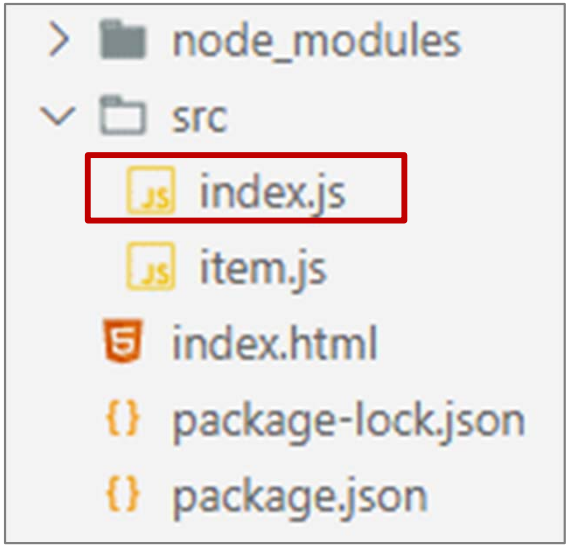
Webpackの利用手順

① WebpackとWebpack CLIをインストールする

```
npm install --save-dev webpack webpack-cli
```

② srcフォルダ内にindex.jsを配置する

– デフォルトでフォルダ構成やファイル名が決まっているが、設定ファイルでカスタマイズ可能



Webpackの利用手順

③ インポート時は拡張子が不要になる

index.js

```
import { addItem, showItems } from "./item";

...
```

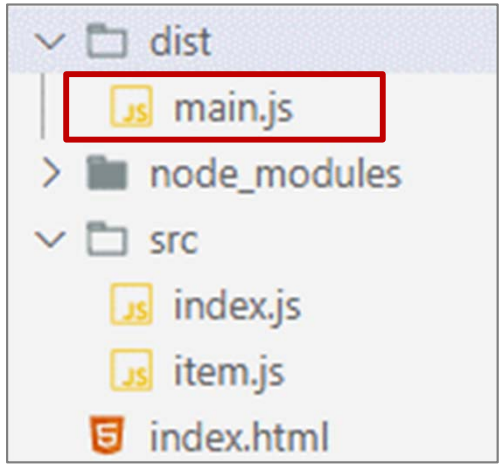
インポート時は、拡張子不要

④ 以下のコマンドを実行することでバンドルファイルが生成される

- distフォルダにmain.jsが生成される

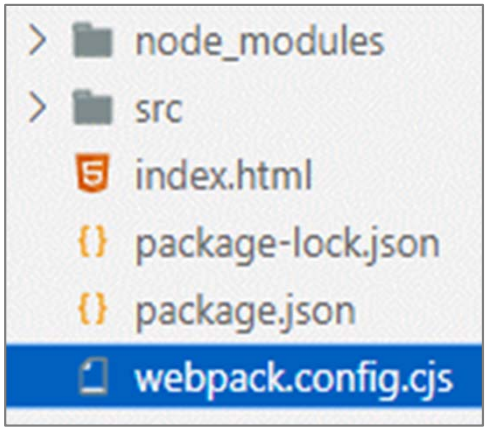
```
npx webpack
```

```
npx webpack --watch （監視オプションも使用可能）
```



Webpackの設定

- Webpackは、webpack.config.cjsというファイルで設定を行うことができる
  - プロジェクトのルート階層に配置する



webpack.config.cjsの記述例

```
module.exports = {
  // エントリーポイント
  entry: "./src/app.js",
  // 出力設定
  output: {
    path: __dirname + "/dist/js",
    filename: "bundle.js",
  },
};
```

初期値：index.js

初期値：distフォルダ

初期値：main.js

練習

- 練習06-2