11. using modules (no npm)

Wednesday, June 7, 2023 3:43 PM

Creating our own custom modules for other script.js to call:

mkdir ./modules
cd ./modules
touch replaceTemplate.js

** This kind of Node.js templating is outdated in 2024 **
index.js:
Cut the highlighted part => Paste to ./modules/replaceTemplate.js

```
// Server
// Arrow function to replace {%%} with product props in data.json
const replaceTemplate = (temp, product) => {
    // Avoid direct mutation of original template by storing temp in a
new var
    let output = temp.replace(/{%PRODUCTNAME%}/g,
product.productName);
    // this function mutates all {%%} in template-xx.html with each
obj in json
    // const cardsHtml = dataObj.map(element =>
replaceTemplate(tempCard, element)).join('');
    // Start mutating now
    // Replacing all {%%} with
    output = output.replace(/{%IMAGE%}/g, product.image);
    output = output.replace(/{%PRICE%}/g, product.price);
    output = output.replace(/{%FROM%}/g, product.from);
    output = output.replace(/{%NUTRIENTS%}/g, product.nutrients);
    output = output.replace(/{%QUANTITY%}/g, product.quantity);
    output = output.replace(/{%DESCRIPTION%}/g, product.description);
    output = output.replace(/{%ID%}/g, product.id);
    if (!product.organic) output = output.replace(/{%NOT_ORGANIC%}/g,
'not-organic')
    return output; // output final Html after replacing
}
// Top-level code only executes once
// Can only use Sync for top-level code
const tempOverview = fs.readFileSync(`${__dirname}/templates/template-
```

```
    return output; // output final html after replacing
}
// Top-level code only executes once
// Can only use Sync for top-level code
const tempOverview = fs.readFileSync(`
${__dirname}/templates/template-overview.html`, 'utf-8');
const tempCard = fs.readFileSync(`
${__dirname}/templates/template-card.html`, 'utf-8');
const tempProduct = fs.readFileSync(`
${__dirname}/templates/template-product.html`, 'utf-8');
const data = fs.readFileSync(`${__dirname}/dev-data/data.json`,
'utf-8');
const dataObj = JSON.parse(data);
```

<u>replaceTemplate.js</u>:

// Export an Arrow function as a module

```
// Using module.exports = (arugment1, argument2) => {...} to
export
// an Arrow Function, without an ES6 package.json --> "type":
"module",
module.exports = (temp, product) => {
    // Avoid direct mutation of original template by storing
temp in a new var
    let output = temp.replace(/{%PRODUCTNAME%}/g,
product.productName);
    // this function mutates all {%%} in template-xx.html with
each obj in json
    // const cardsHtml = dataObj.map(element =>
replaceTemplate(tempCard, element)).join('');
    // Start mutating now
    // Replacing all {%%} with
    output = output.replace(/{%IMAGE%}/g, product.image);
    output = output.replace(/{%PRICE%}/g, product.price);
    output = output.replace(/{%FROM%}/g, product.from);
    output = output.replace(/{%NUTRIENTS%}/g,
product.nutrients);
    output = output.replace(/{%QUANTITY%}/g, product.quantity);
    output = output.replace(/{%DESCRIPTION%}/g,
product.description);
    output = output.replace(/{%ID%}/g, product.id);
    if (!product.organic) output = output.replace(/{%
NOT_ORGANIC%}/g, 'not-organic')
    return output; // output final Html after replacing
}
```

index.js:
```
// Read-Eval-Print-Loop (REPL)
// const repl = require('repl');
const fs = require('fs');
// Networking capabilities => Building a HTTP server
const http = require('http');
// URL module
const url = require('url');

// import our custom module
const replaceTemplate = require('./modules/replaceTemplate');
```