DR INŻ. PIOTR SIEWNIAK

Moduły

WPROWADZENIE

Moduł (*module*) w `node.js` odpowiada bibliotece (*library*) w `JS`. Moduły w `node.js` można podzielić na trzy zasadnicze grupy:

1. moduły podstawowe (*core modules*);
2. moduły lokalne (*local modules*);
3. moduły zewnętrzne innych firm (*third-party external modules*).

Moduły podstawowe

Moduły podstawowe są modułami wbudowanymi (*built-in modules*). Nie ma potrzeby specjalnej instalacji modułów wbudowanych. Po uruchomieniu `node` moduły wbudowane są dostępne dla programisty w sposób automatyczny, ponieważ są one dołączane do platformy `node` w czasie jej instalacji – stanowią jej integralne komponenty.

Jednakże, aby wykorzystać wybrany moduł wbudowany w aplikacji, należy go wcześniej do niej dołączyć (tj. zaimportować jego zasoby publiczne) za pomocą funkcji standardowej `require()`.

Funkcja `require()` zwraca obiekt lub funkcję, lub właściwość itp. – w zależności od tego, jaki moduł jest dołączany do aplikacji (czyli co stanowi jego zwracany zasób publiczny). Zasób publiczny modułu to zasób udostępniany światu zewnętrznemu, dostępny na zewnątrz tego modułu.

Np. w instrukcji

```
const http = require('http');
```

funkcja `require()` zwraca obiekt, który zostaje podstawiony do zmiennej `http`.

Przykład aplikacji, w której wykorzystano moduł wbudowany

```
// Import (dołączenie) modułu wbudowanego http:
var http = require('http');

// Utworzenie serwera webowego nasłuchującego na porcie 8080:
http.createServer(function(req, res) { // req - żądanie (request), res - odpowiedź (response)

    // Wysłanie nagłówka odpowiedzi dla żądania:
    res.writeHead(200, {'Content-Type': 'text/html'});
```

```
    /* UWAGA
        Metoda writeHead() pozwala na wysłanie nagłówka odpowiedzi dla żądania.
        200 to kod stanu HTTP (HTTP status code) oznaczający operację zakończoną sukcesem.
        {'Content-Type': 'text/html'} jest obiektem reprezentującym nagłówki odpowiedzi.
     */
    // Wysłanie odpowiedzi do klienta:
    res.write('<html lang="pl">' +
        '<head>' +
        '<meta charset="UTF-8"></head>' +
        '<body>' +
        '<h1>Odpowiedź serwera na żądanie ...</h1>' +
        '</body>' +
        '</html>');

    // Wyświetlenie komunikatu na localhost:8080:
    res.end('Nagłówek odpowiedzi i sama odpowiedź zostały w komplecie przesłane do klienta ...');
}).listen(8080);
/* UWAGA
  Metoda end() pozwala na przesłanie odpowiedzi do klienta oraz sygnalizuje serwerowi,
  że nagłówek odpowiedzi i zawartość odpowiedzi zostały przesłane w komplecie.
/*

// Wyświetlenie informacji w konsoli:
console.log('Serwer webowy NODE pracuje i nasłuchuje na porcie 8080 ...');
```

Przykłady modułów wbudowanych

`http`	Tworzenie serwera HTTP
`fs`	Przetwarzanie plików
`assert`	Funkcje pomocne w procesie testowania
`path`	Praca ze ścieżkami plików
`process`	Sterowanie procesami w Node
`os`	Informacje o systemie operacyjnym
`querystring`	Przetwarzanie ciągów zapytań (*query strings*)
`url`	Praca z adresami URL

Moduły lokalne

Moduły lokalne danej aplikacji są lokalne względem tejże aplikacji. Mogą one być definiowane samodzielnie przez programistę. Można również dołączać do aplikacji moduły, które zostały zdefiniowane wcześniej – przez innych programistów lub zespołu programistów.

Moduły lokalne są zapisywane w plikach. Zwykle jest tak, że dany moduł odpowiada jednemu, określonemu plikowi. Pliki modułów można separować w danej aplikacji od innych modułów (plików) w odrębnych, dedykowanych folderach.

Moduły napisane samodzielnie przez programistę/programistów można grupować w pakiety i publikować jako ogólnodostępne. Można to zrealizować za pomocą menadżera `npm`.

Przykład modułu lokalnego zdefiniowanego przez programistę

```
// Definicja funkcji polePr() połączona z jej eksportem:
module.exports.polePr = function(a, b) {
        return a * b;
}
// Definicja funkcji obwodPr() połączona z jej eksportem:
module.exports.obwodPr = (a, b) => {
        return 2 * a + 2 * b;
}
/* UWAGA
   Wyeksportowane zasoby modułu stanowią jego zasoby publiczne, udostępnione światu zewnętrznemu.
   Z kolei zasoby prywatne modułu to wszystkie stałe, zmienne, funkcje, obiekty itp., które nie
   zostały wyeksportowane. Ich zasięg jest lokalny. Zasoby lokalne modułu nie są dostępne
   na zewnątrz tego modułu, tj. w jego otoczeniu.
*/
```

Analogicznie jak moduły wbudowane, zasoby publiczne modułów lokalnych są dołączane do aplikacji za pomocą funkcji standardowej `require()`.

Moduły zewnętrzne (pakiety)

Moduły zewnętrzne – pakiety to moduły o określonej funkcjonalności, które zostały napisane i opublikowane przez inne firmy, instytucje, niezależnych programistów itp. Sztandarowym przykładem modułu zewnętrznego jest `express`. Inne przykłady pakietów `node` to `mongoose`, `angular`, `react`.

Aby można było wykorzystać w aplikacji zasoby publiczne danego modułu zewnętrznego należy go wcześniej pobrać i zainstalować za pomocą menadżera pakietów `npm`.

Instalacja pakietu może być prowadzona globalnie lub lokalnie.

Instalacja globalna pakietu, np. pakietu `express`

```
npm install express -g
```
lub
```
npm install express --global
```

spowoduje zapisanie pakietu w folderze `node_modules`, jako podkatalogu katalogu użytkownika systemowego, np. `c:\users\piotr\node_modules`.

Instalacja lokalna pakietu dotyczy danego projektu. Np.

```
npm install express
```

spowoduje instalację lokalną pakietu `express` w folderze `node_modules`, jako podkatalogu foldera aplikacji.

Inne operacje wykonywane za pośrednictwem menadżera `npm` to:

- aktualizowanie pakietu, np.

  ```
  npm update express
  ```

- odinstalowanie pakietu, np.

  ```
  npm uninstall express.
  ```

Użycie opcji `--save` przy odinstalowywaniu pakietu, czyli np.

```
npm uninstall express --save
```

spowoduje automatyczną aktualizację pliku `package.json`.