Консольні додатки NodeJS

Для створення найпростішого консольного датку необхідно створити файл з скриптом на JS і запустити його на виконання з використанням команди node

|  |  |
| --- | --- |
|  |  |
| node шлях\_до\_файлу\_з\_скриптом | ---Файл з скриптом---    --- Запуск файлу на виконання ---- |

Задання параметрів під час запуску скрипта

|  |  |
| --- | --- |
|  |  |
| При виклику скрипта з парамтрами  node шлях\_до\_крипта **параметр1** **параметр2**, …  Значеня у скрипт передаються у вигляді масиву  ["шлях до node", "шлях до скрипта", "параметр 1", "параметр 2", …]  Самі ж параметри можемо одержати скопіювавши елементи масиву починаючи з індекса 2  process.argv.slice(2) |  |

**Виведення даних**

Для виведення даних можна використати метод

console.log(arg1, arg2, …)

та інші (console.error(), console.warn()), для виведення у відповідні потоки виводу

**Введення даних**

var readline = require('readline');var rl = readline.createInterface({

input: process.stdin, // ввод из стандартного потокаoutput: process.stdout // вывод в стандартный поток

});

Обробка введених рядків

|  |  |
| --- | --- |
|  |  |
| rl.on('line', function (введений рядок )  {  обробка рядка  }); |  |

Получение . ответа на вопрос (аналогично prompt в браузере):.  
rl.question('What is your favorite food?', function(answer) { console.log('Oh, so your favorite food is ' + answer);});Додатові фунції для роботи з readline:

Пауза (блокування вводу)

rl.pause() Розблокування вводу:.  
rl.resume() Закінчення роботи з інтерфейсом readline:.  
rl.close()

Модулі JS

Модуль – це окрема програмна одиниця (окремий файл), яка може бути використана (підключена) при реаліазції інших модулів з використанням директиви require. У NodeJS є ряд типів модулів (js, NODE)

<https://nodejs.org/api/modules.html>

<https://metanit.com/web/nodejs/2.2.php>

<http://imnotgenius.com/6-priyomy-raboty-s-modulyami/#objmodule>

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| Створення  (потрібно лише створити файл з розширенням js) |  |  |
| Підключення інших модулів  (необхідно використати require) | require(‘ шлях до іншого модуля’) |  |
| Опис локальних змінних  (змінні, які описані у модулі є локальними) |  |  |
| Повертаємо об’єкти (змінні, функції, класи) з модуля  (для цього використаємо exports)  (this, module.exports) | Опис програмного об’єкта  exports . експортований\_об’єкт = об’єкт з модуля  exports . назва\_об’єкта при експорті = об’єкт з модуля  -------------  var змінна = require (‘ шлях\_до\_модуля ’)  const змінна = require (‘ шлях\_до\_модуля ’) | exports.User = User  ------ модуль, що використовує  var user = require(‘.user’)  var ivan = new user.User(“Іван”) |
| Повернення об’єктів, як глобальних  (не рекомендовано) | Опис програмного об’єкта  global . експортований\_об’єкт = об’єкт з модуля  -------------  var змінна = require (‘шлях\_до\_модуля’) | **global**.User = User  ------ модуль, що використовує  var user = require(‘.user’)  var ivan = new User(“Іван”) |
| Модуль, як директорія з файлом index.js  (у цій же папці можна розташовувати додаткові ресурси, які використовуються тільки цим модулем) | 1)Створюємо папку, з назвою модуля  2)В цій папці створюємо файл **index.js** |  |
| Використання  module.exports |  |  |
| Використання спеціального каталогу  module\_exports  при підключенні модуля, який роміщено у папці module\_exports  рівнем вище від місця підключення можна вказувати лише назву модуля (повний шлях вказувати не потрібно) |  |  |
| Передача даних у модуль (модуль-фабрика) | У модулі експортуємо функцію, через параметри якої модуль і одержує дані ззовні. Для їх збереження використовуємо замикання (функція повертає іншу функцію, для якої параметри зовнішньої є глобальними і доступні через замикання)  module.exports=**function** (параметри) {  **return function** (. . . ) {  використання параметрів  } }  ----------------------------------------------------------  При імпорті модуля-функції передаємо значення у модуль  require(“модуль”) ( параметри ) |  |

Інший сучасний стиль підключення модулів

import

<https://learn.javascript.ru/modules>

Властивості об’єкта module. Об’єкт модуль містить інформацію про модуль, в середині якого до змінної module звертаються

|  |
| --- |
| Module {  **id:** 'C:\\start\\user\\index.js',                               // 1  **exports**: { User: [Function: User] },                           // 2  **parent:**                                                        /\* 3     Module {                                                      \*       id: '.',                                                    \*       exports: {},                                                \*       parent: null,                                               \*       filename: 'C:\\start\\server.js',                           \*       loaded: false,                                              \*       children: [ [Circular] ],                                   \*       paths: [ 'C:\\start\\node\_modules', 'C:\\node\_modules' ] }, \*/    filename: 'C:\\start\\user\\index.js',                         // 4  **loaded**: false,                                                 // 5    children:                                                      /\* 6     [ Module {                                                    \*         id: 'C:\\start\\user\\ru.json',                           \*         exports: [Object],                                        \*         parent: [Circular],                                       \*         filename: 'C:\\start\\user\\ru.json',                     \*         loaded: true,                                             \*         children: [],                                             \*         paths: [Object] } ],                                      \*/    paths:                                                         /\*7     [ 'C:\\start\\user\\node\_modules',                            \*       'C:\\start\\node\_modules',                                  \*       'C:\\node\_modules' ] }                                      \*/ |

Итак:

1. «id» как правило содержит полный путь к файлу. Если операционная система поддерживает символические ссылки и файловая система их поддерживает конечно же, то все символические ссылки будут здесь отображены. Если вы являетесь пользователем Windows и не знаете, что такое символические ссылки, то ничего страшного, для понимания это не критично. «id»  используется внутри Node.JS. Как правило мы сами его использовать не будем.
2. «exports» мы уже знаем. Это экспорты того, что выдается наружу.
3. «parent» Это ссылка на родительский модуль, то есть на тот модуль который required данный.
4. if
5. «loaded» Загрузился ли модуль. На момент, когда мы выводим в консоль модуль, этот модуль еще до конца не обработан, файл то не выполнен до конца, поэтому «loaded: false».
6. «children» Это соответственно те модули, которые данный модуль подключил, через «require()». В данном случае тут только один модуль, наш «ru.json», который, обратите внимание, «loaded: true».
7. «path:» Это тоже внутренняя переменная как и «id» и мы ее не будем использовать, я скажу о ней несколько слов, позже, когда мы будем разбирать порядок поиска модулей с учетом путей.

Из этого обширного списка нам важны два свойства, которые на практике используются наиболее часто. Это свойство «parent» и «exports», по этому о них мы в дальнейшем поговорим более подробно.

Визначення того, чи запущено модуль на виконання безпосередньо (а не підключено до іншого модуля)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Для цього використовуємо властивість  module.parent  яка містить посилання на модуль, який підключив розглядуваний модуль (якщо такий є) | ... текст модуля ...  if(module.parent){      ... модуль підключено іншим ...  ... експортуємо дані ...  }else{  ... модуль запущено безпосередньо ...  } | var user = require('./user');    function run(){  var vasya = new user.User("Вася");  var petya = new user.User("Петя");    vasya.hello(petya);  }  //----------------------  if(module.parent){      exports.run = run;  }else{      run();  } |

Модулі JSON

JSON – формат представлення даних.

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
| Загальна форма формату JSON | {  “назва\_властивості1”: “значення1”,  “назва\_властивості2”: “значення2”,  . . . . . . .  “назва\_властивостіN”: “значенняN”,  }  (назва властивості обов’язково у дужках) |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Кешування модулів**

Якщо модуль один раз було завантажено, то при повторному підключенні використовуються використовується його копія з кеша.

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  | Загальна форма | Приклад |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |