



Zaawansowane Techniki WWW (HTML, CSS i JavaScript)

Dr inż. Marcin Zieliński

Środa 15:30 - 17:00 sala: A-1-04

WYKŁAD 9

Wykład dla kierunku: **Informatyka Stosowana II rok**

Rok akademicki: **2015/2016 - semestr zimowy**

Przypomnienie z poprzedniego wykładu

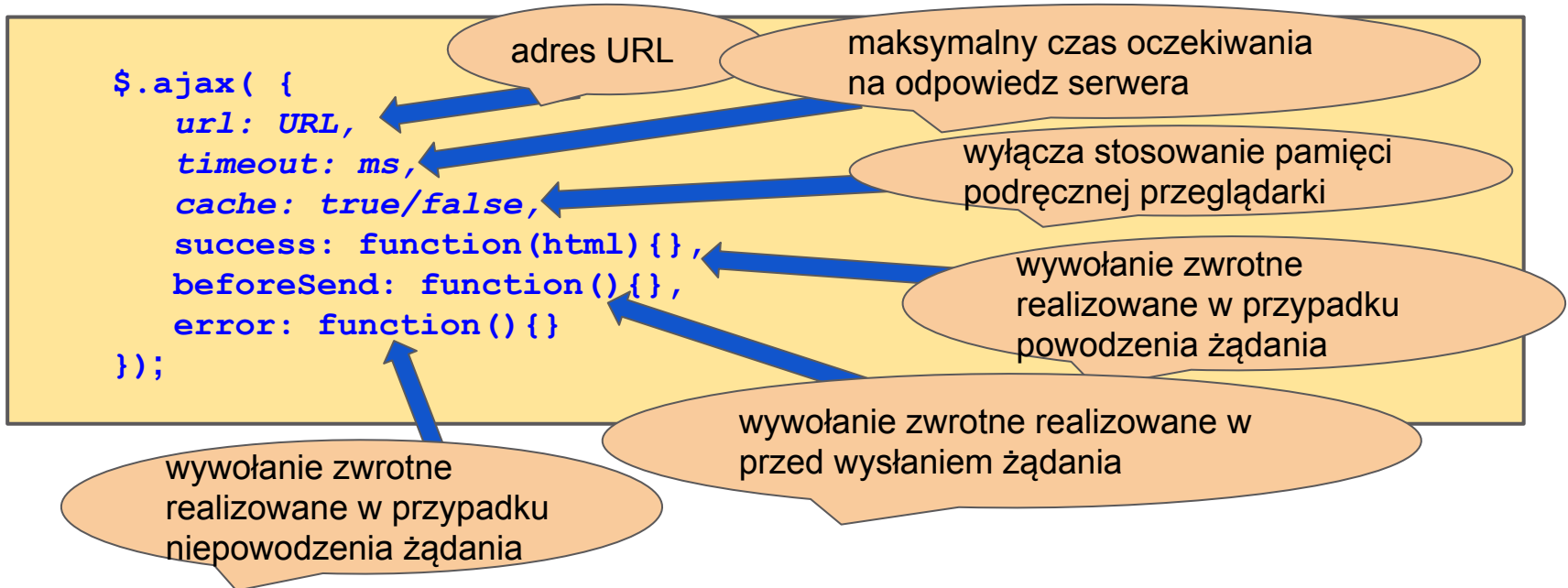
Wykorzystanie biblioteki jQuery do realizacji asynchronicznej komunikacji klient-serwer

Wprowadzenie do środowiska Node.js

AJAX i jQuery



Użycie metody “ajax()” jest nieco trudniejsze:



AJAX i jQuery



Przykład 2:

```
$(function() {  
    $.ajax({  
        url: "localhost/api/getDataJson",  
        dataType: "json",  
        success: function(json) {  
            $("#tresc").html(JSON.stringify(json));  
        }  
    });  
});  
<body>  
    <div id="tresc"></div>  
</body>
```

określamy typ danych
przychodzących

Pobieramy asynchronicznie dane ze adresu URL i w razie sukcesu zakończenia żądania wyświetlamy treść (json) w elemencie #tresc.

AJAX i jQuery



Przykład 3:

```
$(function(){  
  $.ajax({  
    url: 'localhost/api/getData',  
    type: 'GET',  
    data: 'Username=jquery4u',  
    success: function(data) {  
      $('#results').html(data);  
    },  
    error: function(e) {  
      console.log(e.message);  
    }  
  });  
});
```

określamy metody wysyłania
danych

przesyłane dane
w metodzie GET


AJAX i jQuery



Przykład 4:

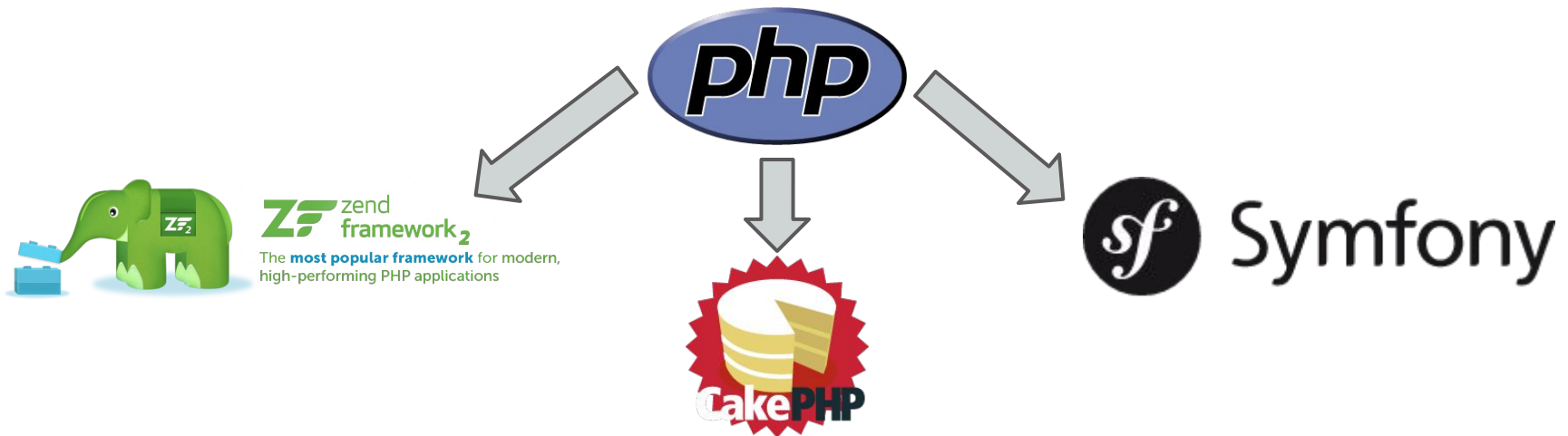
```
$(function(){  
  $.ajax({  
    method: 'POST',  
    url: 'localhost/api/sendData',  
    data: { name: "Jan", fname: "Kowalski" },  
    success: function() {  
      alert( 'Dane zostały przesłane ' );  
    }  
  });  
});
```

określamy metody wysyłania
danych



Popularne dostępne rozwiązania

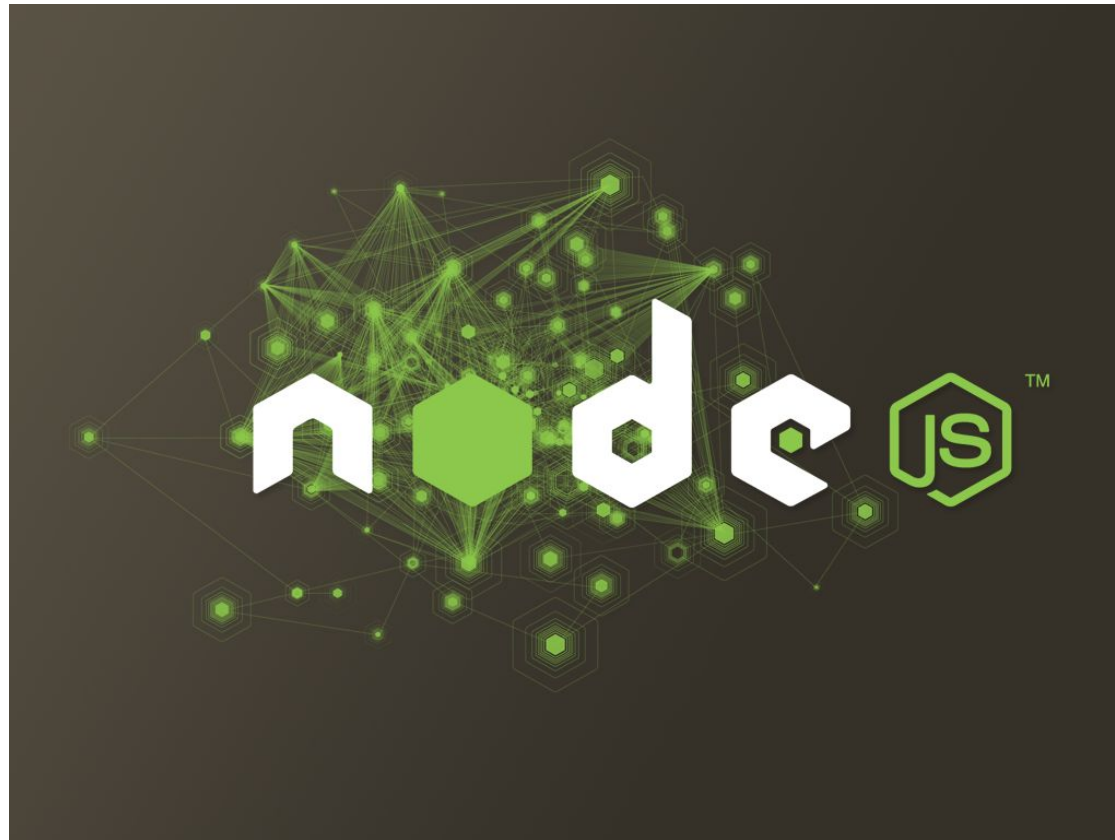
Najpopularniejsze środowiska programistyczne:



oraz systemy CMS (Content Manager System):



Node.js



<http://nodejs.org/>

Krótką historia Node.js

Powstanie Node.js było zainspirowane przez funkcjonalności “push” jakie oferuje np. poczta GMAIL...

Co to jest “push” ?



Usługi push działają w oparciu o przekazane przez klienta wcześniej informacje, na podstawie których serwer dostarcza klientowi nowych danych w zależności od tego czy są one dostępne.

Przykład:

Przykładem usługi “push” jest synchroniczny chat.

JavaScript po stronie serwera

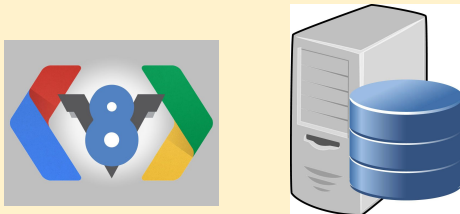
**Musimy odwrócić sposób myślenia jeśli chodzi
wykonywanie JavaScriptu.**



JavaScript po stronie serwera

SERVER SIDE

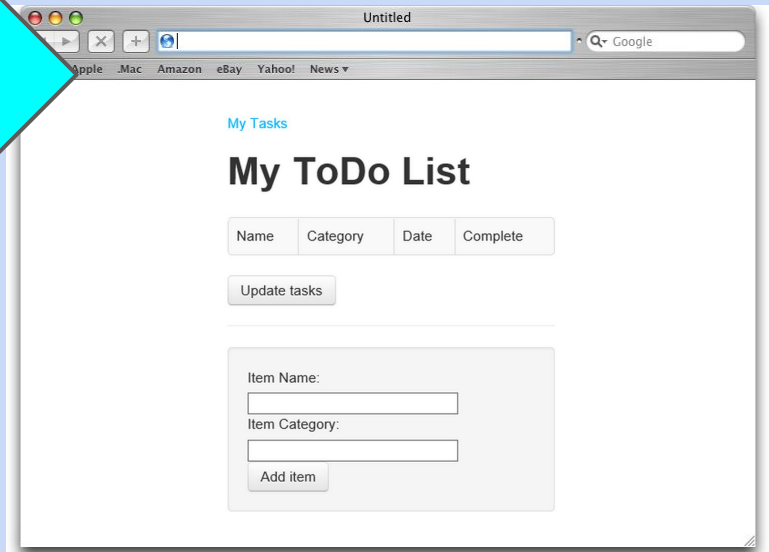
```
1 var edge = require('edge');
2
3 var hello = edge.func(function () { /*
4   async (input) =>
5     {
6       return ".NET welcomes " + input.ToString();
7     }
8   */});
9
10 hello('Node.js', function (error, result) {
11   if (error) throw error;
12   console.log(result);
13 });
```



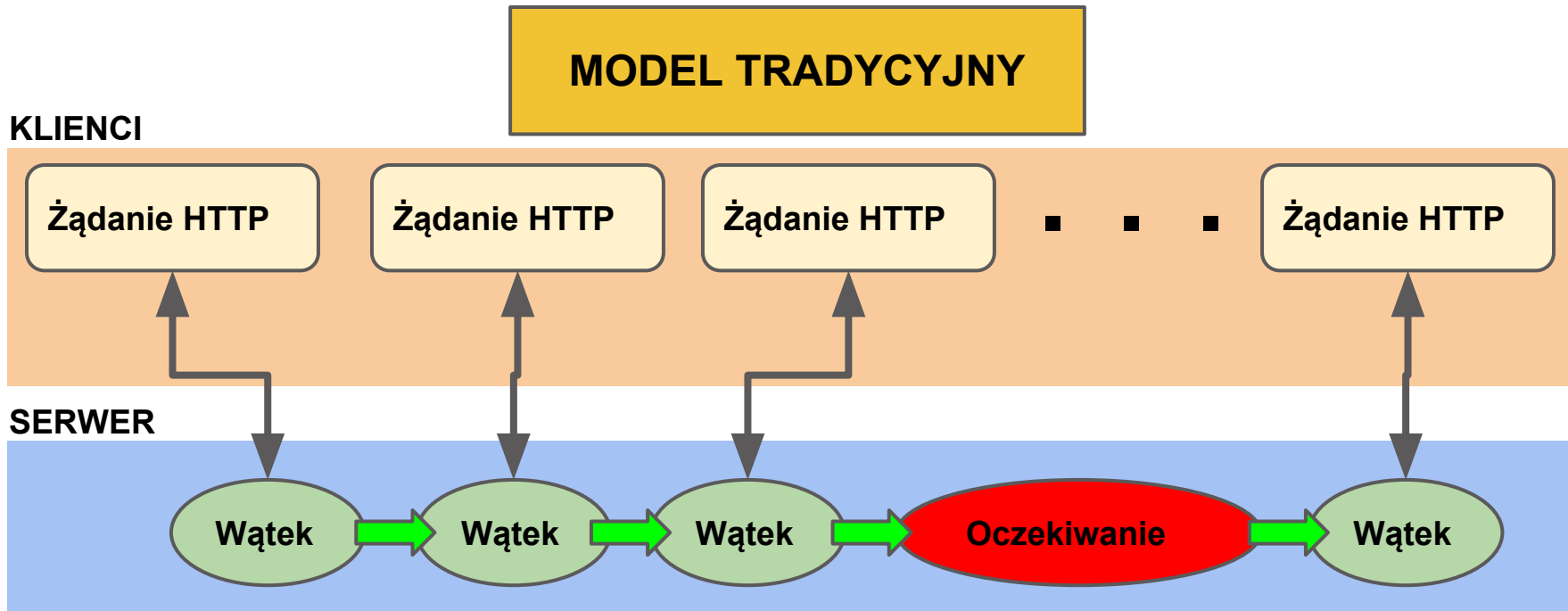
CLIENT SIDE

```
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html" />
<title>CSS3</title>
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="container"> <!-- Main Container -->
<div id="menu"> <!-- Menu here -->
<div class="article"><h2>CSS3 Sample</h2></div>
<div class="article"><h2>CSS3 & HTML5 are so good!</h2></div>
</div>
</body>
</html>
```

HTTP



Obsługa żądań



Przykład:

Dla systemu z 8GB pamięci RAM przydzielającego 2MB pamięci na wątek możemy obsłużyć maksymalnie w tym samym czasie **4000 żądań** (w rzeczywistości jest to mniej ponieważ zużywamy jeszcze pamięć na inne operacje).

Obsługa żądań

MODEL ZDARZENIOWY

KLIENCI

Żądanie HTTP

Żądanie HTTP

Żądanie HTTP

■ ■ ■

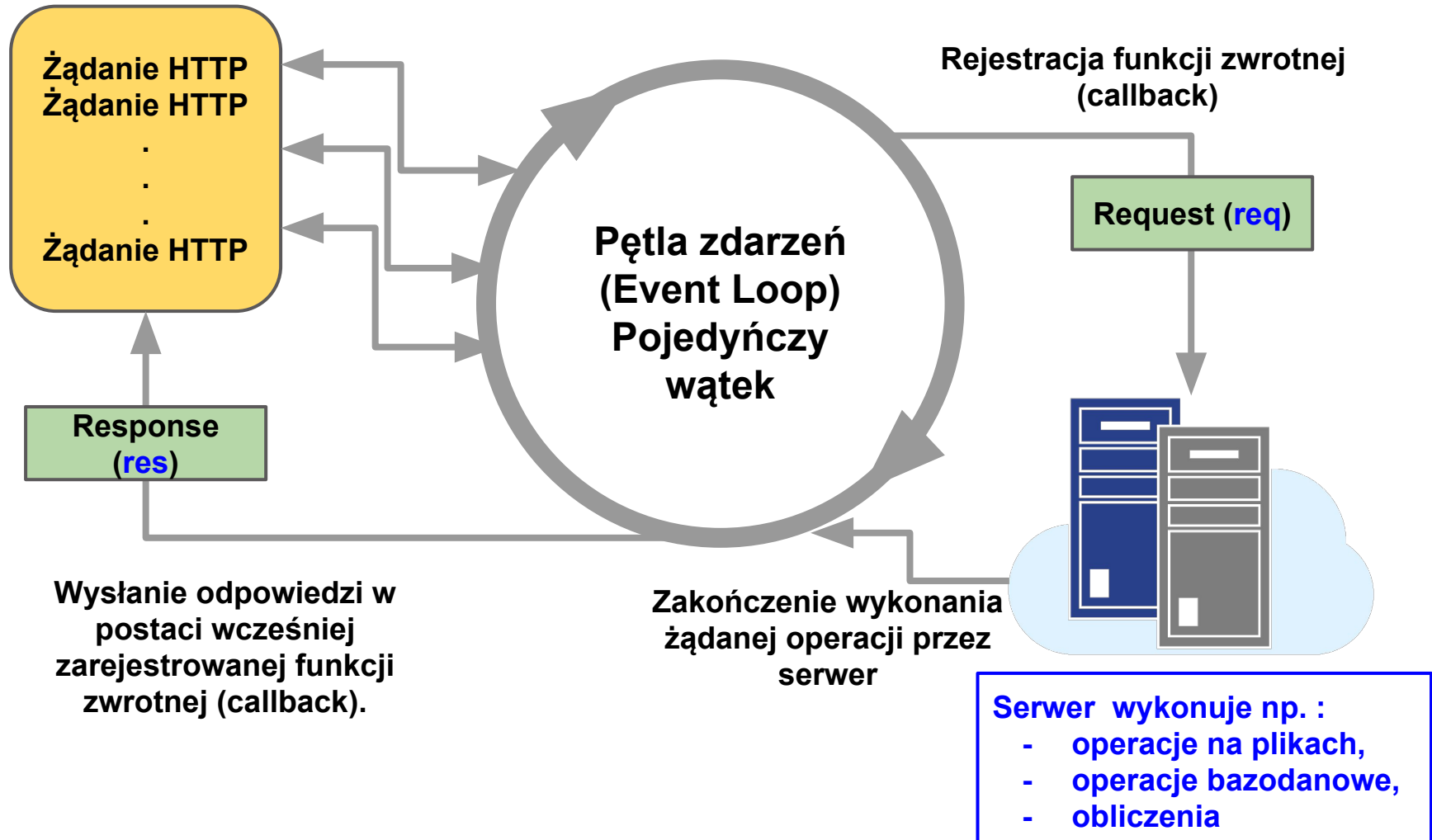
Żądanie HTTP

SERWER

Wątek

W modelu zdarzeniowym Node.js wykorzystuje tylko jeden wątek do obsługi wielu żądań, oraz “pętłę zdarzeń” co powoduje że aplikacja taka jest bardzo wydajna i skalowalna. W praktyce przy żądaniach które nie wymagają złożonych operacji obliczeniowych można obsłużyć nawet do **1 miliona żądań** jednocześnie.

Pętla zdarzeń (Event loop)



Zalety wykorzystania Node.js

1. **Wbudowana asynchroniczność.**
 2. **Sterowany zdarzeniami.**
 3. **“non-blocking” I/O (wykorzystanie jednego wątku).**
 4. **Praca w czasie rzeczywistym.**
 5. **Skalowalność.**
 6. **Duża biblioteka gotowych modułów (NPM).**
 7. **Język JavaScript.**
-

Liderzy wykorzystujący Node.js



Przykład prostego kodu (z strony nodejs.org)

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('Hello World\n');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Prosty serwer przyjmujący żądania http

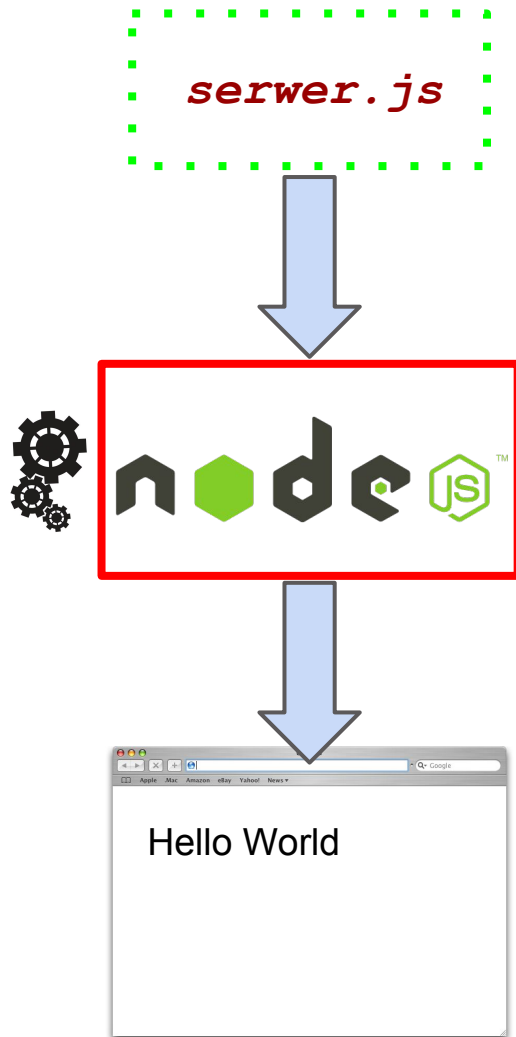
Przykład prostego kodu (z strony nodejs.org)

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('Hello World\n');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Prosty serwer przyjmujący żądania http

Założmy że powyższy kod jest zapisany w pliku “serwer.js”

Przykład prostego kodu (z strony nodejs.org)



Jeśli node.js został zainstalowany globalnie na komputerze, jego uruchomienie polega na wywołaniu z konsoli polecenia:

```
> node  serwer.js
```

```
MacBook-Pro-Marcin-2:~ marcin$ node serwer.js  
Server running at http://127.0.0.1:1337/
```

Przykład prostego kodu (z strony nodejs.org)

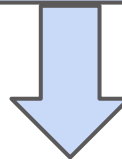
```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('Hello World\n');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Po uruchomieniu tej “aplikacji” powstał serwer http oczekujący na żądania na porcie 1337.

Przykład prostego kodu (z strony nodejs.org)

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('Hello World\n');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Po uruchomieniu tej “aplikacji” powstał serwer http oczekujący na żądania na porcie 1337.



Po otrzymaniu żądania http serwer zwraca odpowiedź OK przesyłając do wyświetlenia zwykły tekst: “Hello World”, który jest widoczny w przeglądarce.

Przykład prostego kodu (z strony nodejs.org)

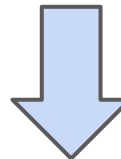
```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('<html><head></head><body>Test</body></html>');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

We wcześniejszym przykładzie serwer zwracał w odpowiedzi zwykły tekst, natomiast teraz zwraca dokument hipertekstowy ze statyczną stroną internetową.

Przykład prostego kodu (z strony nodejs.org)

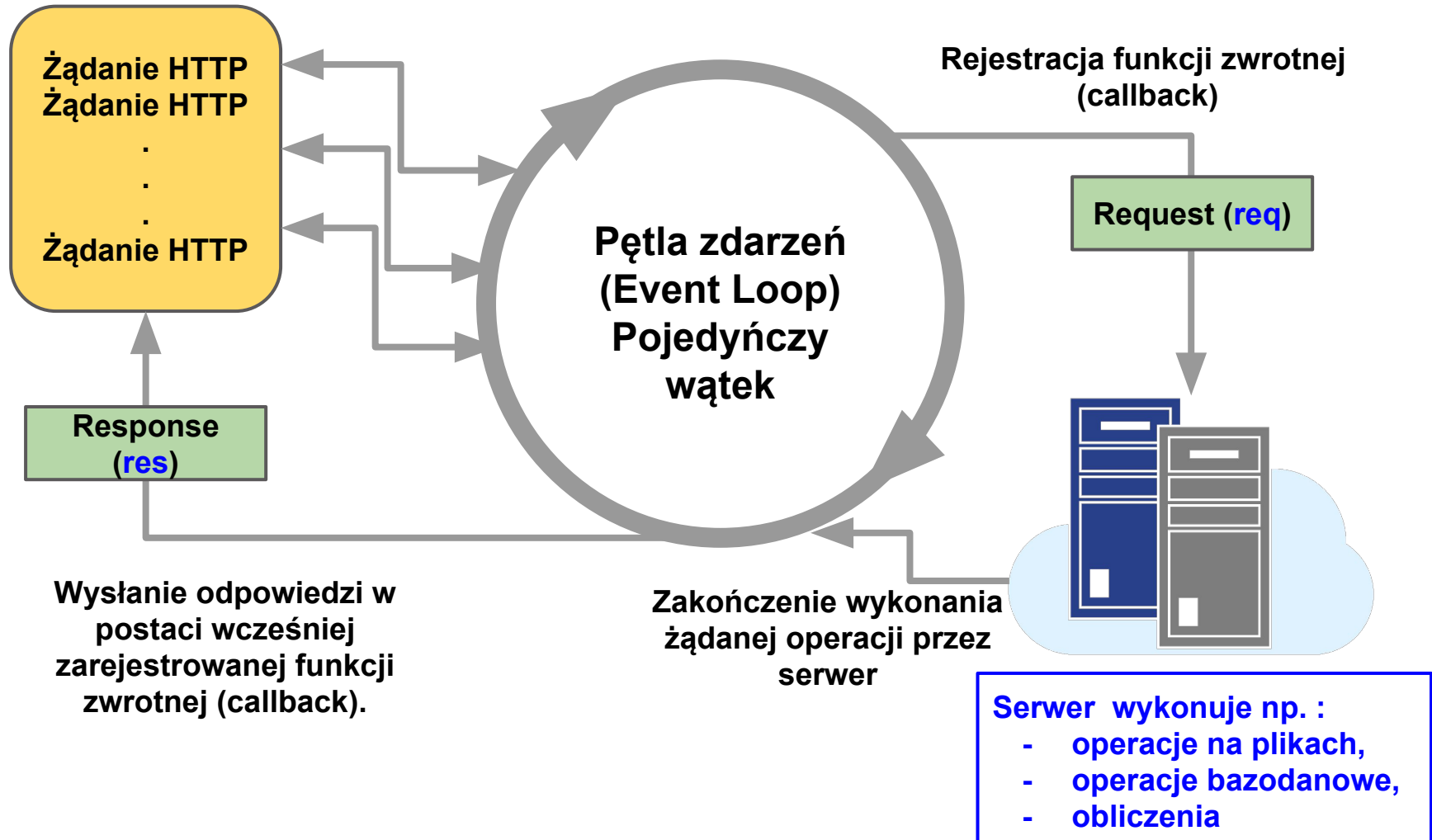
```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.end('<html><head></head><body>Test</body></html>');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

We wcześniejszym przykładzie serwer zwracał w odpowiedzi zwykły tekst, natomiast teraz zwraca dokument hipertekstowy ze statyczną stroną internetową.



Stworzyliśmy serwer obsługujący żądania HTTP !!!

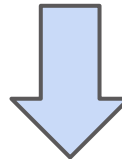
Pętla zdarzeń (Event loop)



Moduły i obiekty

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('hello world');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

We wcześniejszym przykładzie serwer zwracał w odpowiedzi zwykły tekst, natomiast teraz zwraca dokument hipertekstowy ze statyczną stroną internetową.



Stworzyliśmy serwer obsługujący żądania HTTP !!!



Moduły i obiekty

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('hello world');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Wróćmy do naszego “prostego” przykładu który spełniał rolę serwera http...

Moduły i obiekty

```
var http = require('http');  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/plain'});  
  res.end('hello world');  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Wróćmy do naszego “prostego” przykładu który spełniał rolę serwera http...

W języku JavaScript a tym samym w node.js wszystko jest “obiektem” i dlatego możemy zawsze obejrzeć “podejrzeć” każdy obiekt. Można to zrobić korzystając ze specjalnej biblioteki “utils”, którą można pobrać z repozytorium NPM:

```
> npm -g install utils
```

Moduły i obiekty

```
var http = require('http');  
var utils = require('utils');  
http.createServer(function (req, res) {  
    res.writeHead(200, {'Content-Type': 'text/plain'});  
    res.end(utils.inspect(req.headers));  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Dodajemy moduł za pomocą metody "require".

Moduły i obiekty

```
var http = require('http');  
var utils = require('utils');  
http.createServer(function (req, res) {  
    res.writeHead(200, {'Content-Type': 'text/plain'});  
    res.end(utils.inspect(req.headers));  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Dodajemy moduł za pomocą metody “require”.

Korzystamy z metody “inspect()” z modułu “utils” która zwraca postać obiektu w formie tekstu, który możemy wypisać.

Moduły i obiekty

```
var http = require('http');  
var utils = require('utils');  
http.createServer(function (req, res) {  
    res.writeHead(200, {'Content-Type': 'text/plain'});  
    res.end(utils.inspect(req.headers));  
}).listen(1337, '127.0.0.1');  
console.log('Server running at http://127.0.0.1:1337/');
```

Dodajemy moduł za pomocą metody “require”.

Korzystamy z metody “inspect()” z modułu “utils” która zwraca postać obiektu w formie tekstu, który możemy wypisać.

```
{ host: 'localhost:1337',  
  connection: 'keep-alive',  
  'cache-control': 'max-age=0',  
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',  
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36',  
  'accept-encoding': 'gzip, deflate, sdch',  
  'accept-language': 'pl-PL,pl;q=0.8,en-US;q=0.6,en;q=0.4' }
```

Obiekt nagłówek żądania http

Moduły i obiekty

```
var module = require('module_name');
```

Nazwa obiektu odnoszącego
się do modułu

Nazwa modułu

Dzięki modułom można tworzyć zestawy metod (biblioteki), mogą być wykorzystane wielokrotnie w różnych aplikacjach.

Moduły i obiekty

```
var module = require('module_name');
```

Nazwa obiektu odnoszącego się do modułu

Nazwa modułu

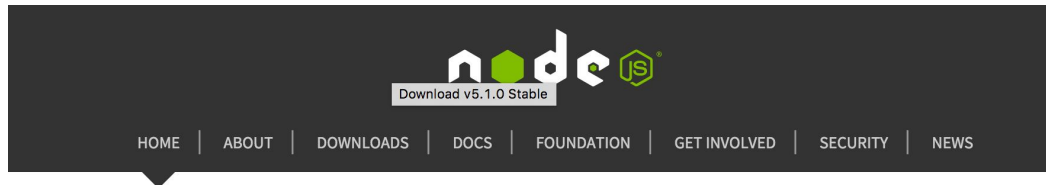
Dzięki modułom można tworzyć zestawy metod (biblioteki), mogą być wykorzystane wielokrotnie w różnych aplikacjach.

Powstały w ten sposób obiekt jest niejako wskaźnikiem na dany moduł przez który możemy wywoływać z moduły dostępne metody i własności:

```
module.zrobCos();
```


Instalacja

<http://nodejs.org/>



Node.js® is a JavaScript runtime built on **Chrome's V8 JavaScript engine**. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

Download for OS X (x64)



Other Downloads: LTS / Stable | Changelog: LTS / Stable | API Docs: LTS / Stable

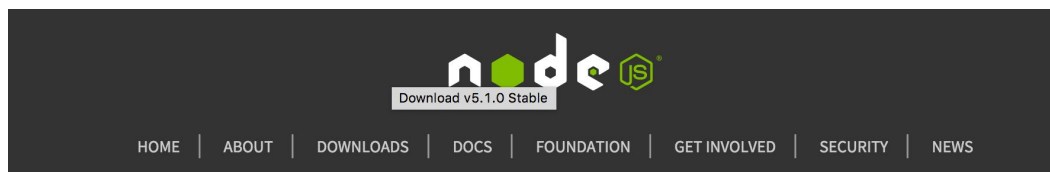
Pobieramy pakiet w postaci archiwum tar który instalujemy:

node-v5.1.0.tar.gz



Instalacja

<http://nodejs.org/>



Node.js® is a JavaScript runtime built on **Chrome's V8 JavaScript engine**. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

Download for OS X (x64)

v4.2.2 LTS

Mature and Dependable

v5.1.0 Stable

Latest Features

Other Downloads: [LTS / Stable](#) | [Changelog: LTS / Stable](#) | [API Docs: LTS / Stable](#)

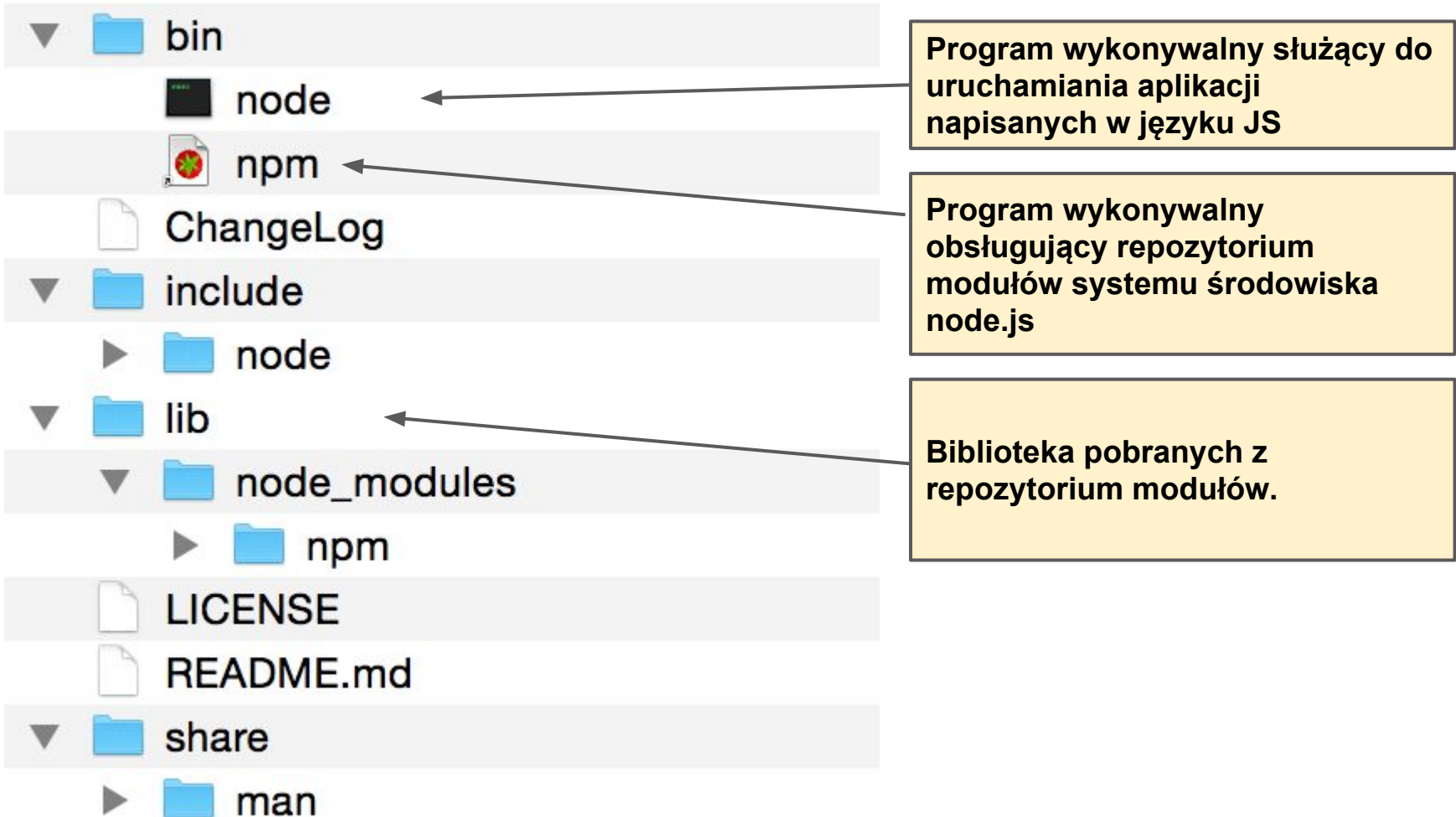
Pobieramy pakiet w postaci archiwum tar który instalujemy:

node-v5.1.0.tar.gz



Alternatywnie mamy dostęp do wersji na różne systemy operacyjne oraz wersje skompilowane w postaci plików wykonywalnych.

Instalacja dla Linux



Instalacja dla Linux

Przechodzimy do kartoteki bin gdzie wykonujemy polecenie:

```
MacBook-Pro-Marcin-2:bin marcin$ ./node -v  
v0.10.33
```

Program odpowiada
numerem wersji

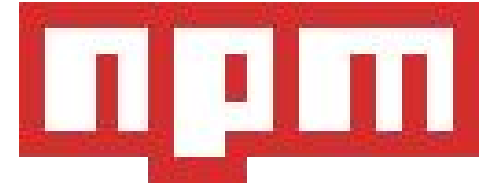
Zaleca się zainstalowanie globalnej wersji node.js dostępnej dla wszystkich użytkowników oraz możliwe do uruchomienie a każdego miejsca w strukturze kartotek.

Uruchomienie skryptu polega na wywołaniu programu “node” z parametrem określającym nazwę skryptu (gdy node.js jest zainstalowany lokalnie):

```
> ./node  serwer.js
```

Repozytorium NPM

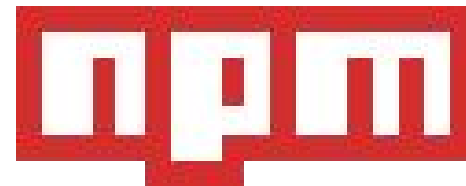
Integralną częścią środowiska NODE.JS, jest bogate repozytorium modułów (bibliotek), dzięki któremu mamy dostęp do wielu gotowych funkcji.



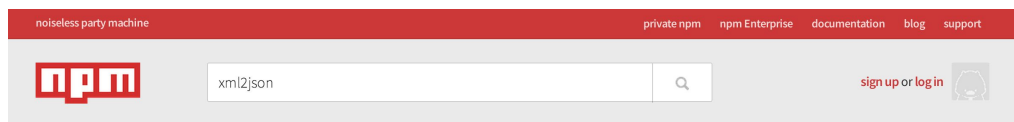
<http://npmjs.org/>

Repozytorium NPM

Integralną częścią środowiska NODE.JS, jest bogate repozytorium modułów (bibliotek), dzięki któremu mamy dostęp do wielu gotowych funkcji.



<http://npmjs.org/>



npm is the package manager for **javascript**.



112 811
total packages



32 164 109
downloads in the last day



174 569 610
downloads in the last week



675 621 989
downloads in the last month

packages people 'npm install' a lot



browserify

browser-side require() the node way
6.2.0 published 2 months ago by substack

express

express

Fast, unopinionated, minimalist web framework
4.10.1 published 2 months ago by dougwilson



pm2

Modern CLI process manager for Node apps with a builtin...
0.11.1 published 2 months ago by tknew



grunt-cli

The grunt command line interface.
0.1.13 published a year ago by tkellen



npm

A package manager for node
2.1.6 published 2 months ago by othym23

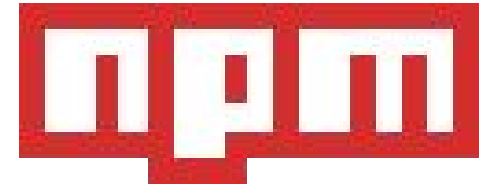


karma

Spectacular Test Runner for JavaScript.
0.12.24 published 3 months ago by vojtaJina

Repozytorium NPM

Integralną częścią środowiska NODE.JS, jest bogate repozytorium modułów (bibliotek), dzięki któremu mamy dostęp do wielu gotowych funkcji.



<http://npmjs.org/>

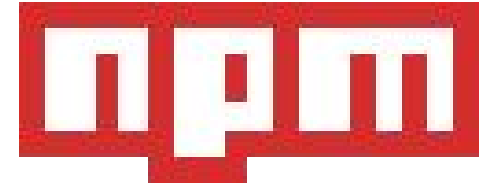
Na stronie internetowej projektu można przeglądać i wyszukiwać pakiety, jednak ich instalacja odbywa się w poziomym wierszu poleceń:

```
| > ./npm install nazwa-pakietu |
```

Pakiety instalują się w kartotece [lib/node_modules](#).

Repozytorium NPM

Integralną częścią środowiska NODE.JS, jest bogate repozytorium modułów (bibliotek), dzięki któremu mamy dostęp do wielu gotowych funkcji.



<http://npmjs.org/>

Na stronie internetowej projektu można przeglądać i wyszukiwać pakiety, jednak ich instalacja odbywa się w poziomym wierszu poleceń:

```
| > ./npm install nazwa-pakietu |
```

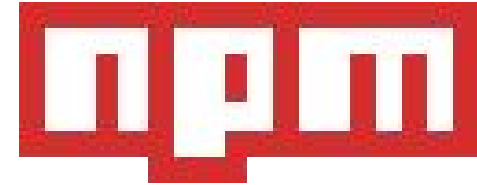
Pakiety instalują się w kartotece `lib/node_modules`.

Do globalnej instalacji pakietów NPM należy posiadać prawa administratora i wydać polecenie:

```
| > ./npm -g install nazwa-pakietu |
```


Repozytorium NPM

```
> ./npm install -g nazwa-pakietu
```



```
MacBook-Pro-Marcin-2:bin marcin$ sudo npm -g install xml2json
npm http GET https://registry.npmjs.org/xml2json
npm http 304 https://registry.npmjs.org/xml2json
npm http GET https://registry.npmjs.org/node-expat
npm http 200 https://registry.npmjs.org/node-expat
npm http GET https://registry.npmjs.org/node-expat/-/node-expat-2.3.3.tgz
npm http 200 https://registry.npmjs.org/node-expat/-/node-expat-2.3.3.tgz
npm http GET https://registry.npmjs.org/bindings
```

Model-View-Controller

Model-View-Controller (MVC) [*Model-Widok-Kontroler*] - jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:

Model-View-Controller

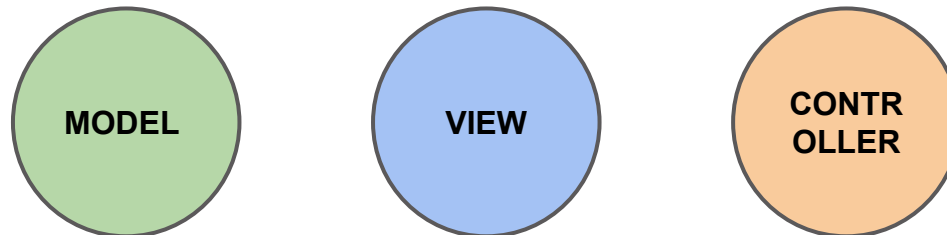
Model-View-Controller (MVC) [*Model-Widok-Kontroler*] - jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:

- Model (dane / logika)
 - Widok (prezentacja danych)
 - Kontroler (interakcja z użytkownikiem + sterowanie aplikacją)
-

Model-View-Controller

Model-View-Controller (MVC) [*Model-Widok-Kontroler*] - jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:

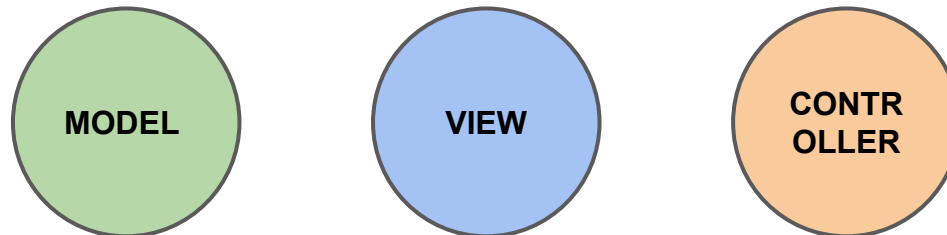
- Model (dane / logika)
- Widok (prezentacja danych)
- Kontroler (interakcja z użytkownikiem + sterowanie aplikacją)



Model-View-Controller

Model-View-Controller (MVC) [*Model-Widok-Kontroler*] - jest to wzorzec projektowy (podejście które jest bazą w oparciu o którą tworzymy aplikację), dzielący projektowaną aplikację na trzy warstwy:

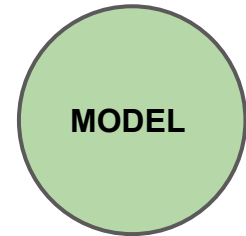
- Model (dane / logika)
- Widok (prezentacja danych)
- Kontroler (interakcja z użytkownikiem + sterowanie aplikacją)



Można go zaimplementować bez użycia bibliotek czy specjalistycznych platform programistycznych, stosując jasne reguły podziału na konkretne komponenty w kodzie źródłowym. W ten sposób każdy komponent aplikacji można niezależnie od siebie rozwijać, implementować i testować.

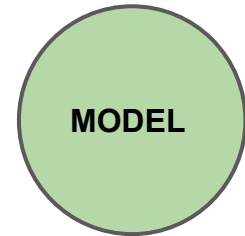
Model-View-Controller

Model - jest odpowiedzialny za przechowywanie wszystkich obiektów danych wykorzystywanych w aplikacji. Model odwozuje logikę danych i fizyczność przetwarzanych danych. Model “nie wie” nic o widokach i kontrolerach. W modelach nie może być zapisany żaden szablon widoku warstwy prezentacji danych ponieważ jest to sprzeczne z wzorcem MVC.



Model-View-Controller

Model - jest odpowiedzialny za przechowywanie wszystkich obiektów danych wykorzystywanych w aplikacji. Model odwozuje logikę danych i fizyczność przetwarzanych danych. Model “nie wie” nic o widokach i kontrolerach. W modelach nie może być zapisany żaden szablon widoku warstwy prezentacji danych ponieważ jest to sprzeczne z wzorcem MVC.



Przykład:

Model użytkownika (USER) może przechowywać listę użytkowników, ich atrybuty oraz wyznacza logikę powiązania tych danych z innymi modelami danych.

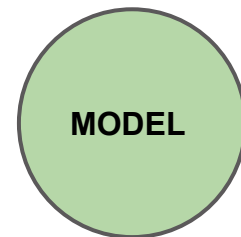
Kiedy kontroler pobiera dane z modelu tworzy na jego podstawie nową “instancję” tego modelu.

Tworzone modele powinny być zorientowane obiektowo !!!

```
var user = Users["Jan"] ;  
UsunUzytkownika(user) ;
```

Model-View-Controller

Model - jest odpowiedzialny za przechowywanie wszystkich obiektów danych wykorzystywanych w aplikacji. Model odwozuje logikę danych i fizyczność przetwarzanych danych. Model “nie wie” nic o widokach i kontrolerach. W modelach nie może być zapisany żaden szablon widoku warstwy prezentacji danych ponieważ jest to sprzeczne z wzorcem MVC.



Przykład:

Model użytkownika (USER) może przechowywać listę użytkowników, ich atrybuty oraz wyznacza logikę powiązania tych danych z innymi modelami danych.

Kiedy kontroler pobiera dane z modelu tworzy na jego podstawie nową “instancję” tego modelu.

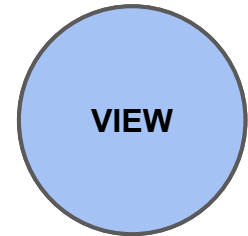
Tworzone modele powinny być zorientowane obiektowo !!!

```
var user = Users["Jan"];  
UsunUzytkownika(user);
```

```
var user = Users.znajdz("Jan");  
user.usun();
```

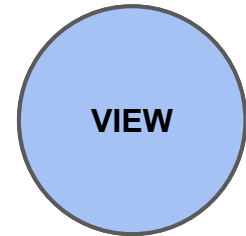

Model-View-Controller

Widok - odpowiada za prezentację danych pobranych z modelu dla użytkownika. Widoki z reguły nie mają żadnej logiki poza kilkoma prostymi instrukcjami warunkowymi. Zgodnie z regułami widoki powinny być całkowicie oddzielone od innych części aplikacji. Logika prezentacji powinna być ujęta w funkcjach “helperach” czyli specjalnych narzędzi do obsługi widoków.



Model-View-Controller

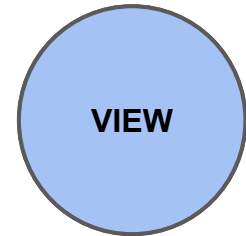
Widok - odpowiada za prezentację danych pobranych z modelu dla użytkownika. Widoki z reguły nie mają żadnej logiki poza kilkoma prostymi instrukcjami warunkowymi. Zgodnie z regułami widoki powinny być całkowicie oddzielone od innych części aplikacji. Logika prezentacji powinna być ujęta w funkcjach “helperach” czyli specjalnych narzędzi do obsługi widoków.



```
<div>
  <script>
    function ZmienDate() {};
  </script>
</div>
```

Model-View-Controller

Widok - odpowiada za prezentację danych pobranych z modelu dla użytkownika. Widoki z reguły nie mają żadnej logiki poza kilkoma prostymi instrukcjami warunkowymi. Zgodnie z regułami widoki powinny być całkowicie oddzielone od innych części aplikacji. Logika prezentacji powinna być ujęta w funkcjach “helperach” czyli specjalnych narzędzi do obsługi widoków.



```
<div>
  <script>
    function ZmienDate() {};
  </script>
</div>
```

```
// w osobnym malym skrypcie
var helper = {};
helper.ZmienDate = function() {};

// w widoku
<div>
  ${helper.ZmienDate()}
</div>
```

Model-View-Controller

Kontroler - spełnia rolę pośrednika pomiędzy warstwą modelu danych, a widokami na których te dane mają zostać zaprezentowane. Przyjmują zdarzenia i dane przychodzące od użytkowników, przetwarzają je z zastosowaniem modeli i kierują do odpowiednich widoków. W trakcie ładowania strony kontroler dodaje do widoków procesy nasłuchiwanie zdarzeń.



CONTR
OLLER

Model-View-Controller

Kontroler - spełnia rolę pośrednika pomiędzy warstwą modelu danych, a widokami na których te dane mają zostać zaprezentowane. Przyjmują zdarzenia i dane przychodzące od użytkowników, przetwarzają je z zastosowaniem modeli i kierują do odpowiednich widoków. W trakcie ładowania strony kontroler dodaje do widoków procesy nasłuchiwanie zdarzeń.



CONTR
OLLER

```
app.get('/login', function(req, res) {  
    res.render('users/login', {data:data} );  
});
```

Model-View-Controller

Kontroler - spełnia rolę pośrednika pomiędzy warstwą modelu danych, a widokami na których te dane mają zostać zaprezentowane. Przyjmują zdarzenia i dane przychodzące od użytkowników, przetwarzają je z zastosowaniem modeli i kierują do odpowiednich widoków. W trakcie ładowania strony kontroler dodaje do widoków procesy nasłuchiwanie zdarzeń.

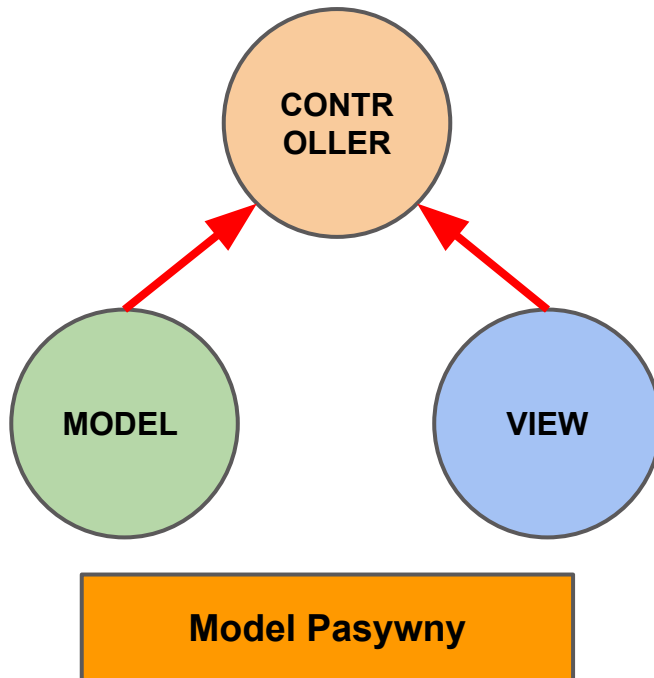


CONTR
OLLER

```
app.get('/login', function(req, res) {  
    res.render('users/login', {data:data} );  
});
```

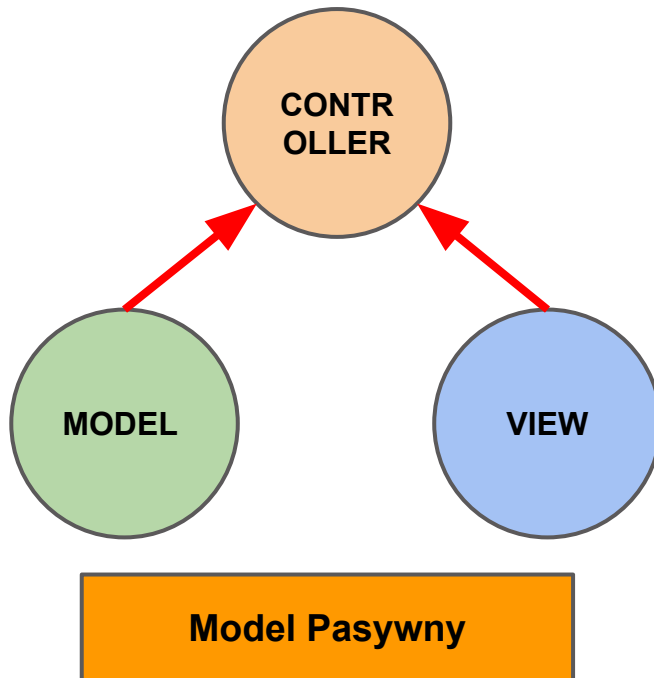
Do implementacji kontrolerów nie są potrzebne żadne specjalne biblioteki ani platformy programistyczne, jednak wygodniej jest z nich korzystać.

Model-View-Controller

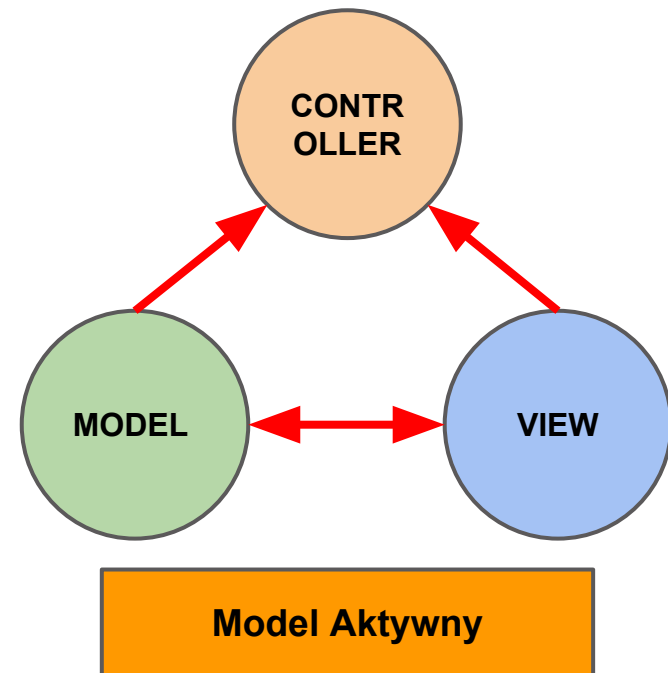


W modelu “pasywny” nie ma wymiany danych pomiędzy modelem a widokiem z pominięciem kontrolera, wszystkie akcje są wywoływane i sterowane przez kontroler.

Model-View-Controller



W modelu “pasywny” nie ma wymiany danych pomiędzy modelem a widokiem z pominięciem kontrolera, wszystkie akcje są wywoływane i sterowane przez kontroler.

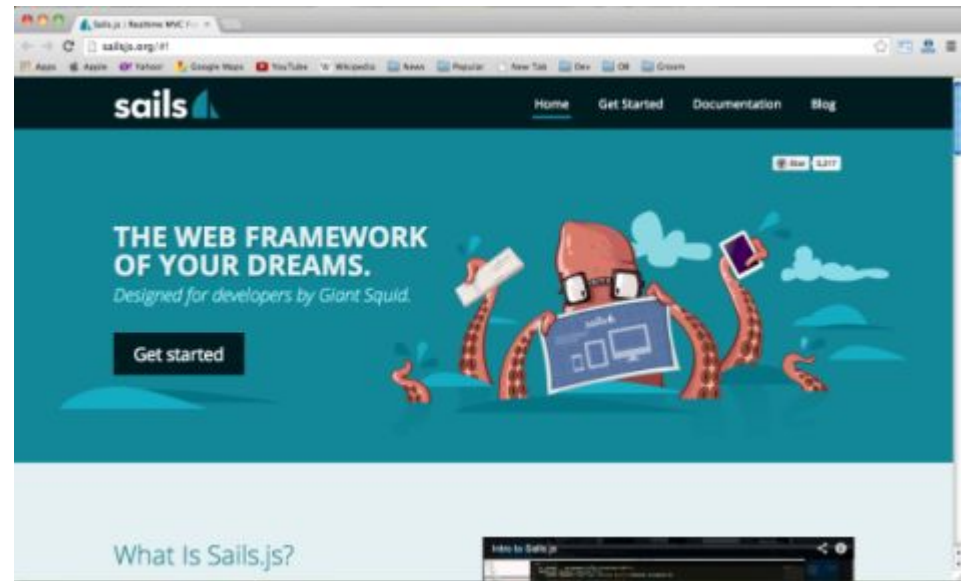


W model aktywnym model może bezpośrednio przekazywać dane do widoku z pominięciem kontrolera.

Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

<http://sailsjs.org/>



Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

<http://www.partialjs.com>



Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

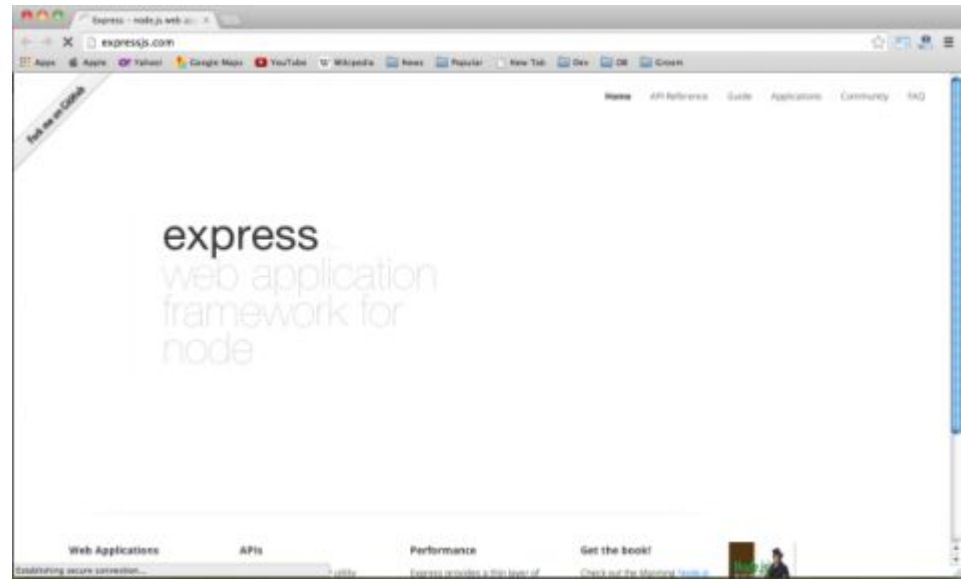
<https://www.totaljs.com/>



Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

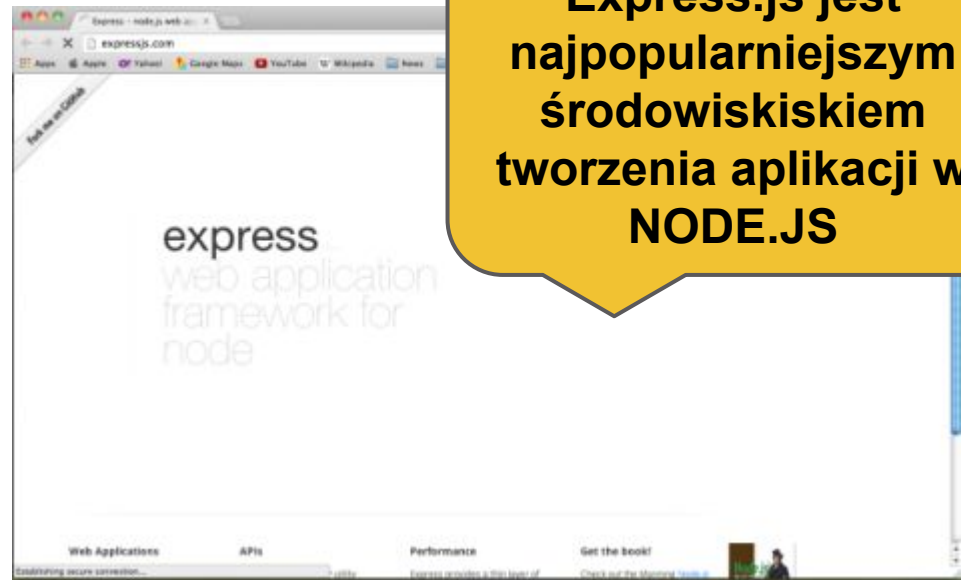
<http://expressjs.com/>



Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

<http://expressjs.com/>

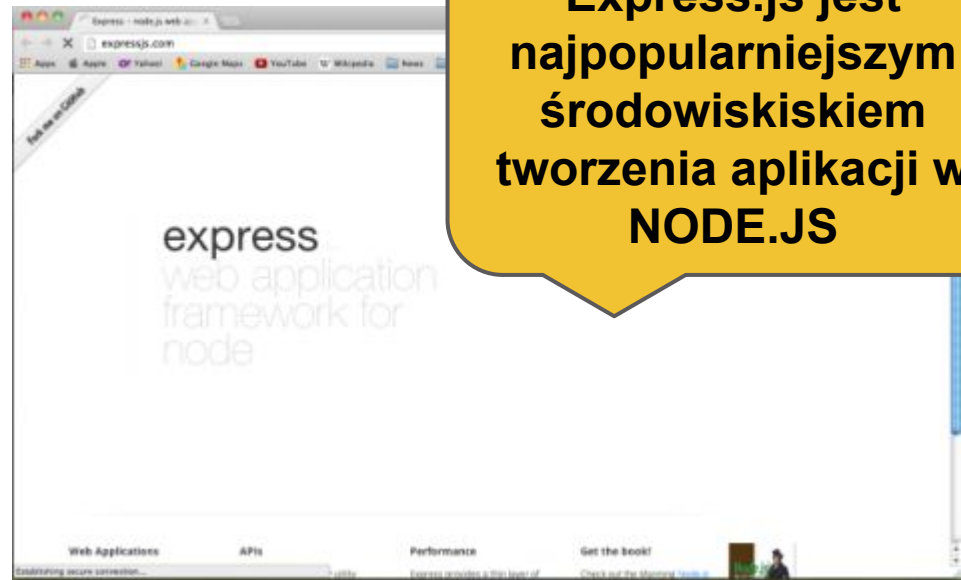


**Express.js jest
najpopularniejszym
środowiskiskiem
tworzenia aplikacji w
NODE.JS**

Node.js + MVC

Istnieje kilka środowisk ułatwiających tworzenie aplikacji według wzorca projektowego MVC w systemie NODE.JS:

<http://expressjs.com/>



**Express.js jest
najpopularniejszym
środowiskiskiem
tworzenia aplikacji w
NODE.JS**

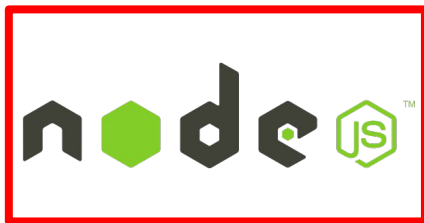
To środowisko będziemy poznawać na zajęciach ...

Express JS

Express.js jest środowiskiem które pozwala na tworzenie aplikacji internetowych w formie jednostronicowych oraz wielostronicowych witryn dostosowanych do wyświetlania na urządzeniach mobilnych oraz normalnych komputerach.

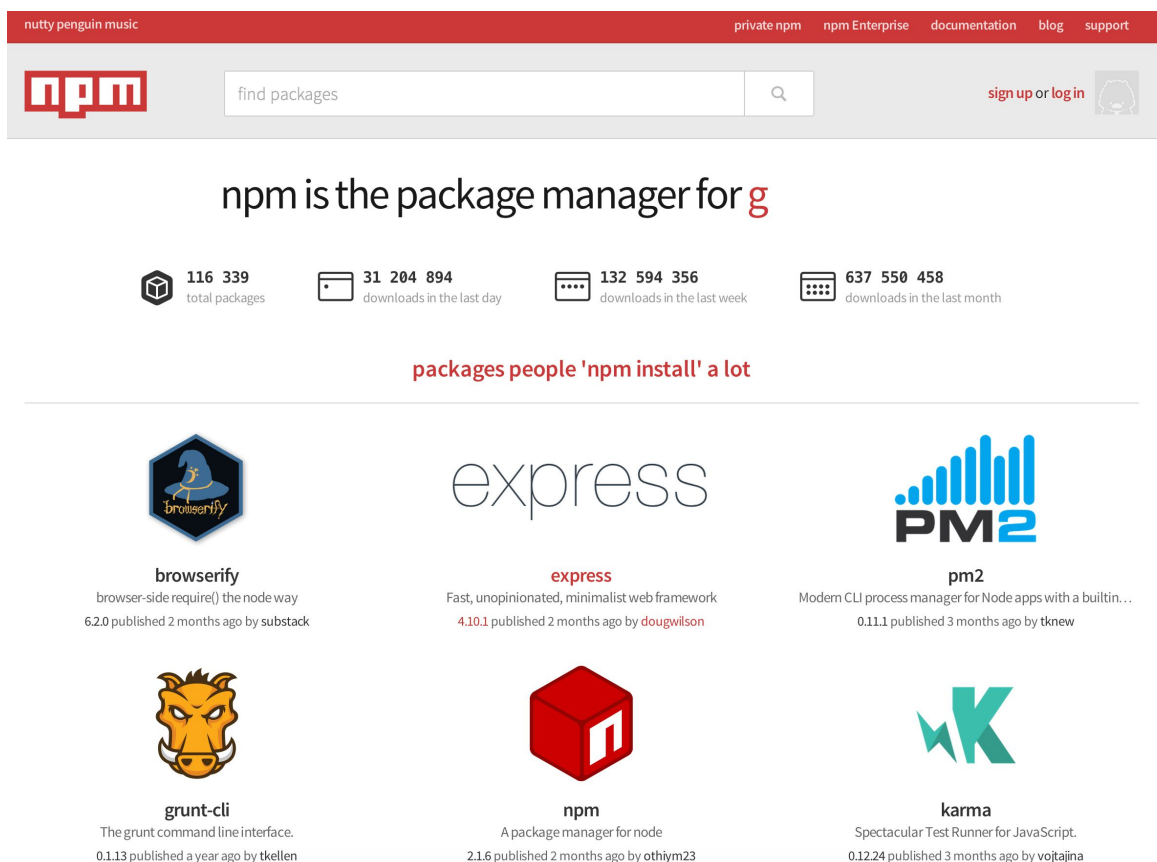
Express JS

Express.js jest środowiskiem które pozwala na tworzenie aplikacji internetowych w formie jednostronicowych oraz wielostronowych witryn dostosowanych do wyświetlania na urządzeniach mobilnych oraz normalnych komputerach.



Express JS - instalacja

Instalacja Express.js odbywa się za pomocą menadżera pakietów NPM.

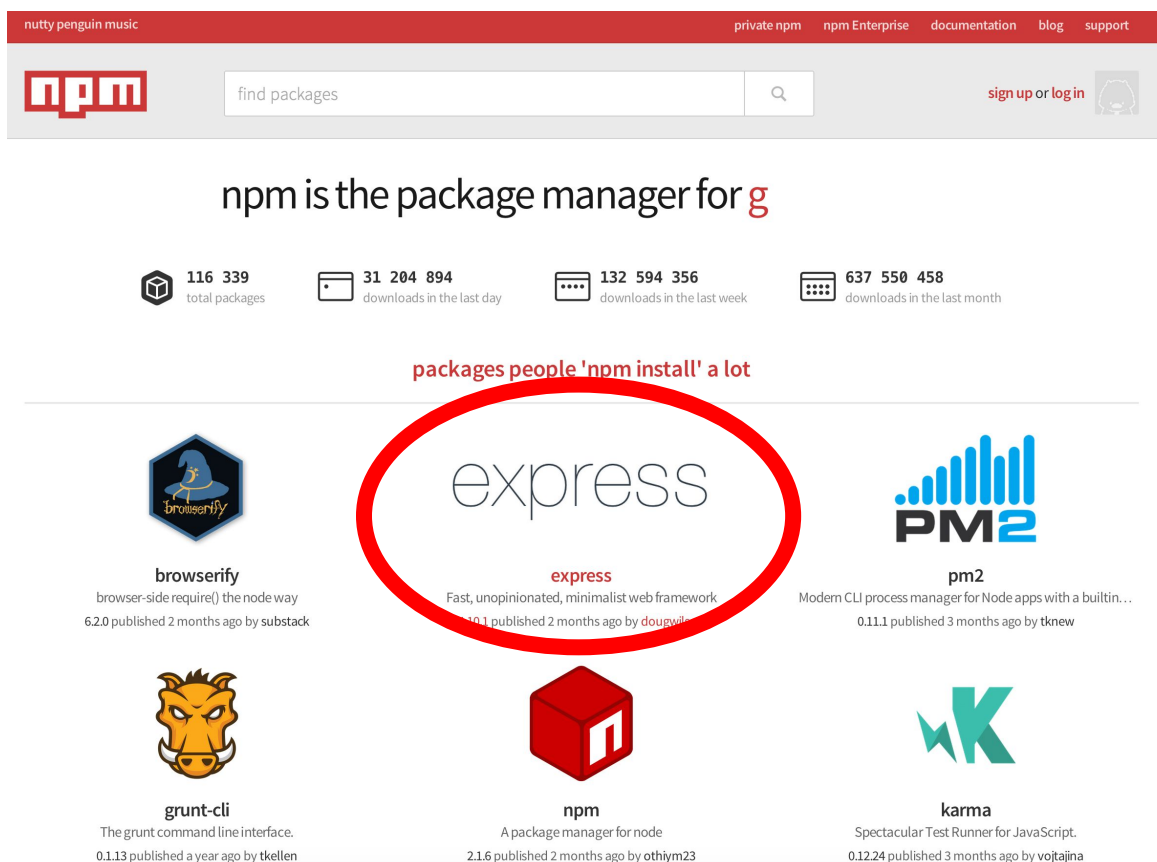


The screenshot shows the npm website interface. At the top, there's a navigation bar with links: "nutty penguin music", "private npm", "npm Enterprise", "documentation", "blog", and "support". Below this is a search bar with the text "find packages" and a magnifying glass icon. To the right of the search bar are links for "sign up or log in" and a user profile icon. The main content area features the text "npm is the package manager for g". Below this, there are four statistics: "116 339 total packages", "31 204 894 downloads in the last day", "132 594 356 downloads in the last week", and "637 550 458 downloads in the last month". A red banner below the statistics says "packages people 'npm install' a lot". Below the banner, there are six featured packages arranged in a 2x3 grid:

Package Name	Description	Version	Published	Author
browserify	browser-side require() the node way	6.2.0	2 months ago	substack
express	Fast, unopinionated, minimalist web framework	4.10.1	2 months ago	dougwilson
pm2	Modern CLI process manager for Node apps with a builtin...	0.11.1	3 months ago	tknew
grunt-cli	The grunt command line interface.	0.1.13	published a year ago	tkellen
npm	A package manager for node	2.1.6	published 2 months ago	othiym23
karma	Spectacular Test Runner for JavaScript.	0.12.24	published 3 months ago	vojtajina

Express JS - instalacja

Instalacja Express.js odbywa się za pomocą menadżera pakietów NPM.






The screenshot shows the npm website interface. At the top, there's a navigation bar with links like 'private npm', 'npm Enterprise', 'documentation', 'blog', and 'support'. Below this is a search bar with the text 'find packages' and a search icon. The main content area features the text 'npm is the package manager for g' followed by statistics: 116 339 total packages, 31 204 894 downloads in the last day, 132 594 356 downloads in the last week, and 637 550 458 downloads in the last month. A section titled 'packages people 'npm install' a lot' displays a grid of popular packages. The 'express' package is highlighted with a red circle. Other visible packages include browserify, pm2, grunt-cli, npm, and karma.

Package Name	Description	Version	Published	Author
browserify	browser-side require() the node way	6.2.0	2 months ago	substack
express	Fast, unopinionated, minimalist web framework	4.19.1	2 months ago	dougville
pm2	Modern CLI process manager for Node apps with a builtin...	0.11.1	3 months ago	tknew
grunt-cli	The grunt command line interface.	0.1.13	1 year ago	tkellen
npm	A package manager for node	2.1.6	2 months ago	othiym23
karma	Spectacular Test Runner for JavaScript.	0.12.24	3 months ago	vojtaJina

Express JS - instalacja

Instalacja Express.js odbywa się za pomocą menadżera pakietów NPM.



[sign up](#) or [log in](#)

★ **express**

Fast, unopinionated, minimalist web framework

npm

v4.10.7

downloads

2M/month

build

passing

coverage

100%

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

 npm install express

Package Info

4.10.7 published 3 days ago by [dougwilson](#)

[github.com/strongloop/express](#)

[expressjs.com](#)

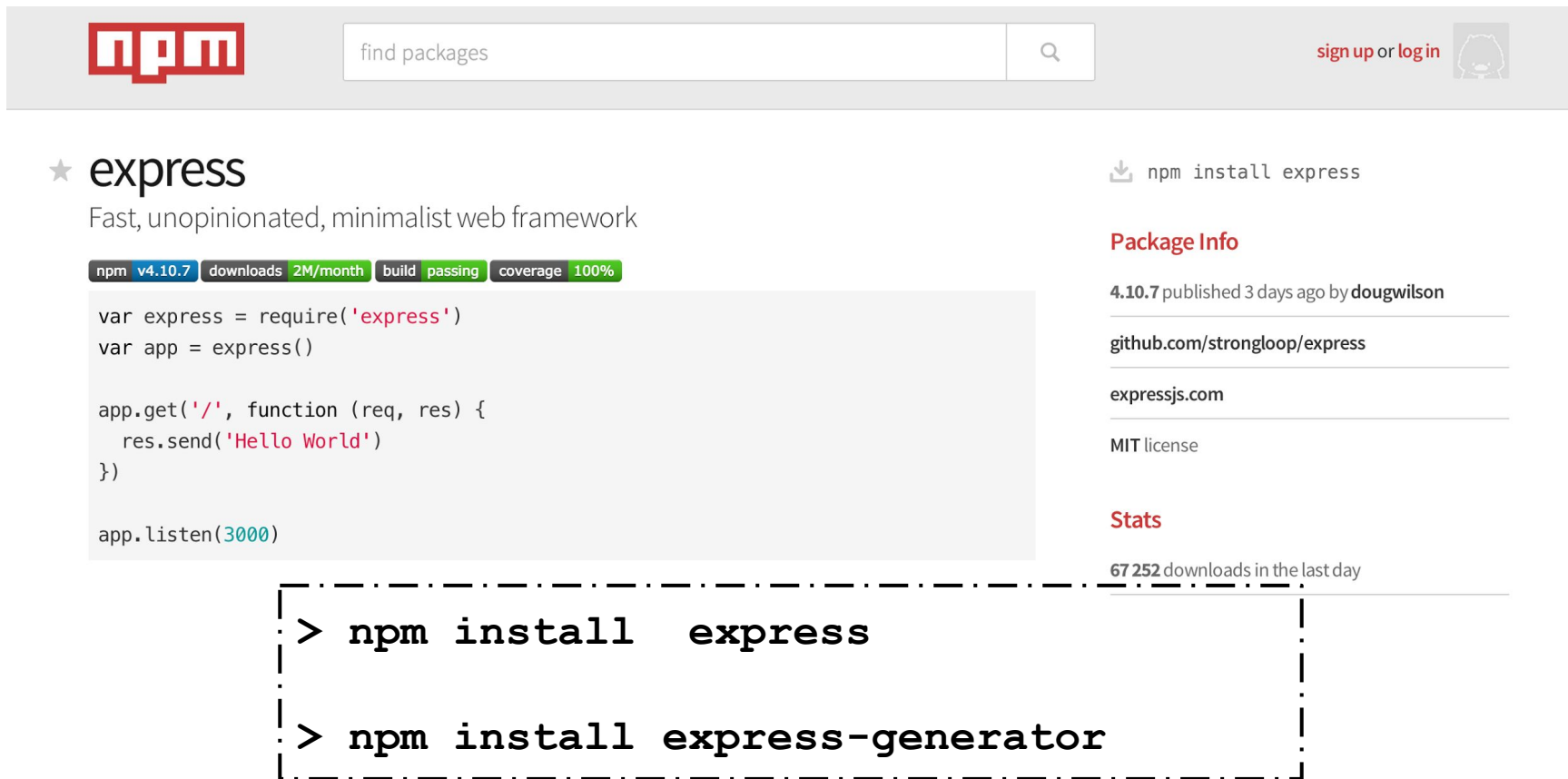
MIT license

Stats

67 252 downloads in the last day

Express JS - instalacja

Instalacja Express.js odbywa się za pomocą menadżera pakietów NPM.



The screenshot shows the npm website interface. At the top is the npm logo and a search bar with the text "find packages". To the right of the search bar are links for "sign up or log in" and a user profile icon. Below the search bar, the "express" package is featured with a star icon. The package description reads "Fast, unopinionated, minimalist web framework". Below the description are several status badges: "npm v4.10.7", "downloads 2M/month", "build passing", and "coverage 100%". A code block displays a sample Express.js application setup. To the right of the code block, there is a section for "Package Info" which includes the version "4.10.7" published 3 days ago by "dougwilson", the GitHub repository "github.com/strongloop/express", the website "expressjs.com", and the "MIT license". Below the package info is a "Stats" section showing "67 252 downloads in the last day". At the bottom of the screenshot, a dashed box contains two terminal commands: `> npm install express` and `> npm install express-generator`.

npm

find packages

sign up or log in

★ express

Fast, unopinionated, minimalist web framework

npm v4.10.7 downloads 2M/month build passing coverage 100%

```
var express = require('express')
var app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

npm install express

Package Info

4.10.7 published 3 days ago by dougwilson

github.com/strongloop/express

expressjs.com

MIT license

Stats

67 252 downloads in the last day

```
> npm install express
> npm install express-generator
```

Express JS - instalacja

```
> npm install express  
  
> npm install express-generator
```

Pierwsze polecenie instaluje środowisko Express.js w ramach posiadanej dystrybucji NODE.JS, natomiast drugim poleceniem instalujemy aplikację pomocniczą służącą do generowania standardowych i działających “wzorców” aplikacji (projektów “0”).

KONIEC WYKŁADU 9
