WYKŁAD 8
Node.js

dr inż. Jacek Paluszak

Node.js

Wstęp

Pod koniec 2009 roku na konferencji JavaScript w Berlinie Ryan Dahl zaprezentował technologię nazwaną **Node.JS** (http://nodejs.org/).

Nowa technologia przewidywała wykonywanie kodu JavaScript na serwerze. Pomysł ten pobudził wyobraźnię publiczności, która przyjęła go owacją na stojąco ☺

„Jeśli wszystko pójdzie dobrze, będziemy mogli pisać aplikacje sieciowe w tylko jednym języku.”

Node.js

Wstęp

Node.js jest frameworkiem, dzięki któremu programiści mogą projektować aplikacje sieciowe o bardzo wysokiej wydajności, w porównaniu z innymi popularnymi rozwiązaniami.

Node.js

Wstęp

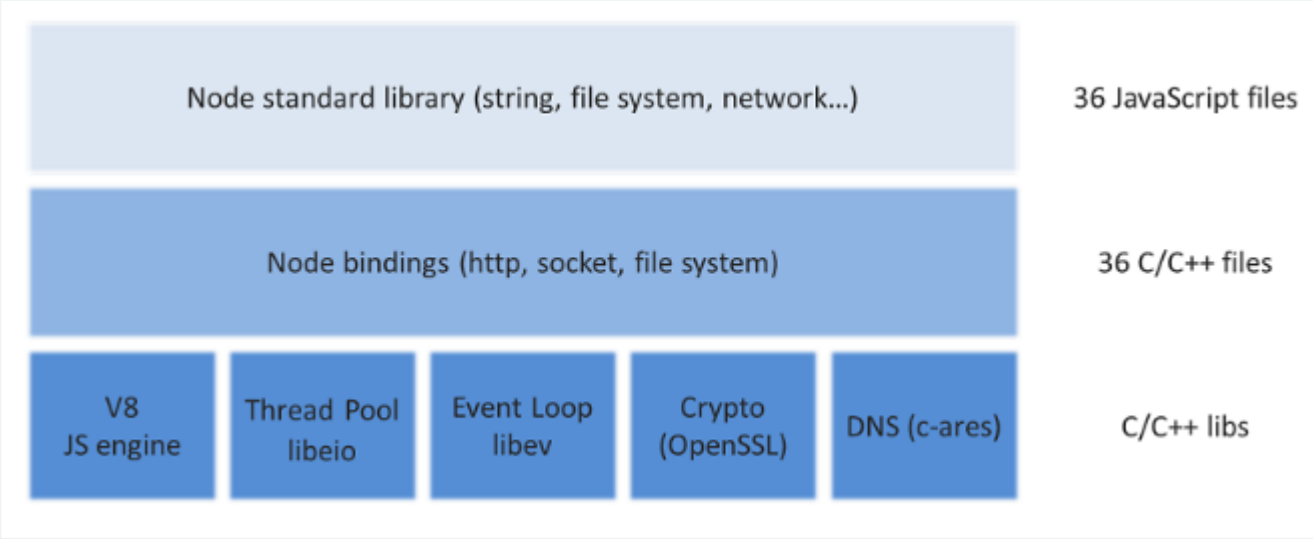
Walmart ogłosił, że dzięki przejściu na platformę **Node.js** był w stanie podwoić liczbę obsługiwanych żądań i skrócić o 35% (200 ms) czas udzielania odpowiedzi. W 2014 przeszedł całkowicie na Node.JS.

Używają Node.js:

- Netflix
- Trello
- PayPal
- LinkedIn
- Uber
- Medium
- Groupon
- Ebay
- NASA
- Home Made
- Yahoo

Node.js

Wstęp - platforma



Node.js

Wstęp - przykład

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

const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n'); }
);

server.listen(port, hostname, () => {
   console.log(`Server running at http://${hostname}:${port}/`);
});
```

Node.js

Wstęp - przykład

```
var fs = require('fs');
fs.readFile('./sample.txt', 'utf8', function (err,data) {
  if (err) {
    return console.log(err);
  }
  console.log(data);
});
```

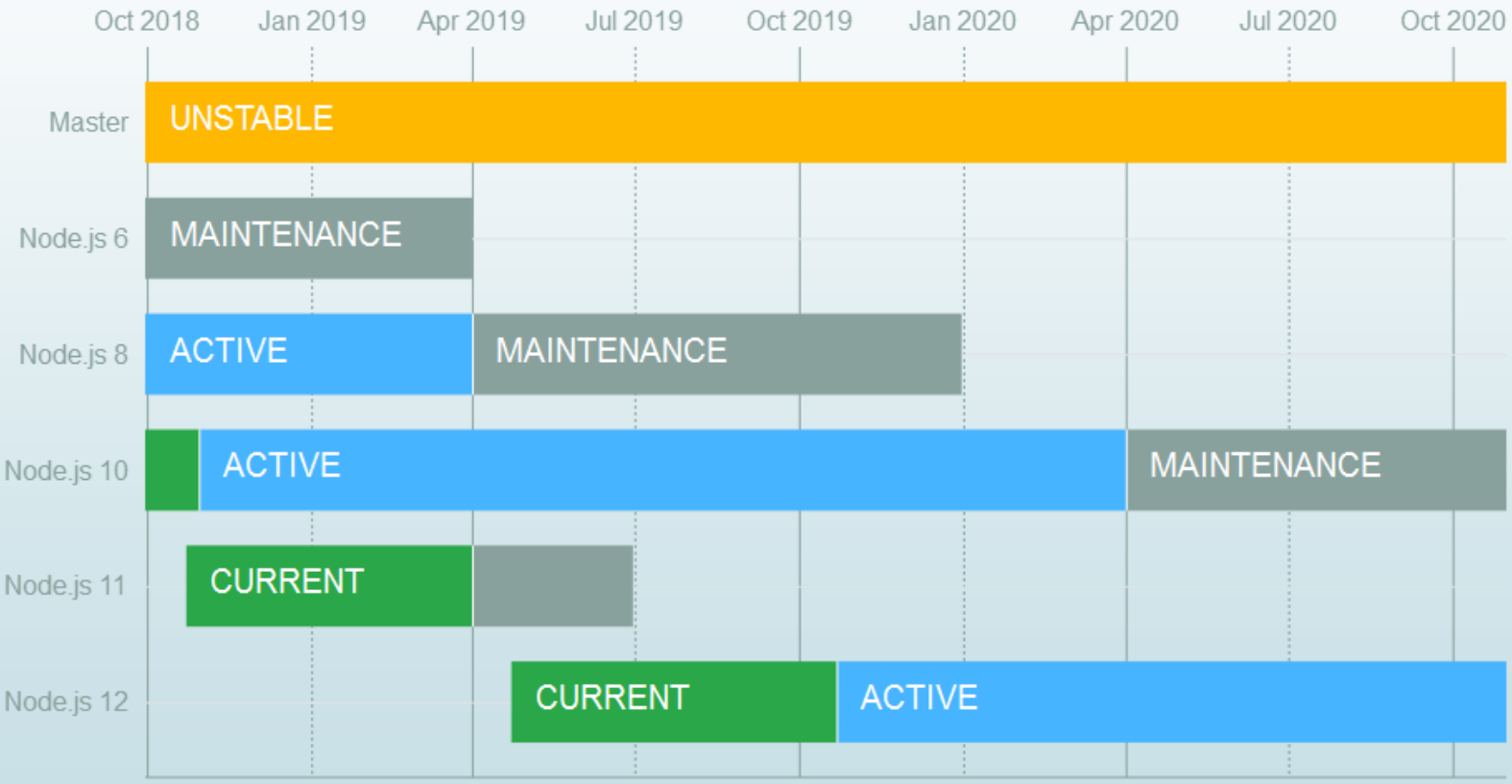
Node.js

Aktualne wersje

Release	Status	Codename	Initial Release	Active LTS Start	Maintenance LTS Start	End-of-life
6.x	**Maintenance LTS**	Boron	2016-04-26	2016-10-18	2018-04-30	April 2019
8.x	**Active LTS**	Carbon	2017-05-30	2017-10-31	April 2019	December 2019
10.x	**Active LTS**	Dubnium	2018-04-24	2018-10-30	April 2020	April 2021
11.x	**Current Release**		2018-10-23			June 2019
12.x	**Pending**		2019-04-23	October 2019	April 2021	April 2022

Node.js

Aktualne wersje



Node.js

Wstęp

Node.JS zawdzięcza swoją niewiarygodną szybkość i wydajność technice zwanej pętlą zdarzeń (ang. event loop) i **silnikowi V8**, na którym jest oparty.

Ten ostatni to interpreter i wirtualna maszyna **JavaScript**, stworzone przez **Google** w trakcie pracy nad przyspieszaniem przeglądarki **Chrome**.

Node.js

Wstęp

Nie ma już konieczności pisania skryptów, które uruchamiane są przez instalowany osobno serwer WWW, jak w tradycyjnym modelu LAMP:

- Linux (system operacyjny)
- Apache (serwer WWW)
- MySQL (serwer bazy danych)
- Perl, PHP, ew. Python (język skryptowy)

Node.JS zmienia zasady gry.

Node.js

Wstęp

Ogólna idea „współbieżności” w oprogramowaniu polega na tym, aby umożliwić jednoczesne wykonywanie wielu zadań.

Trudność, jaką sprawia opracowanie platformy zapewniającej bezpieczeństwo wątków powoduje, że języki takie jak **Ruby, Python i PHP nie obsługują** wątków pozwalających na uzyskanie **prawdziwej współbieżności** i niewymagające uruchamiania niestandardowych plików binarnych.

Powstały więc specjalne biblioteki wspierające współbieżność.

Node.js

Wstęp

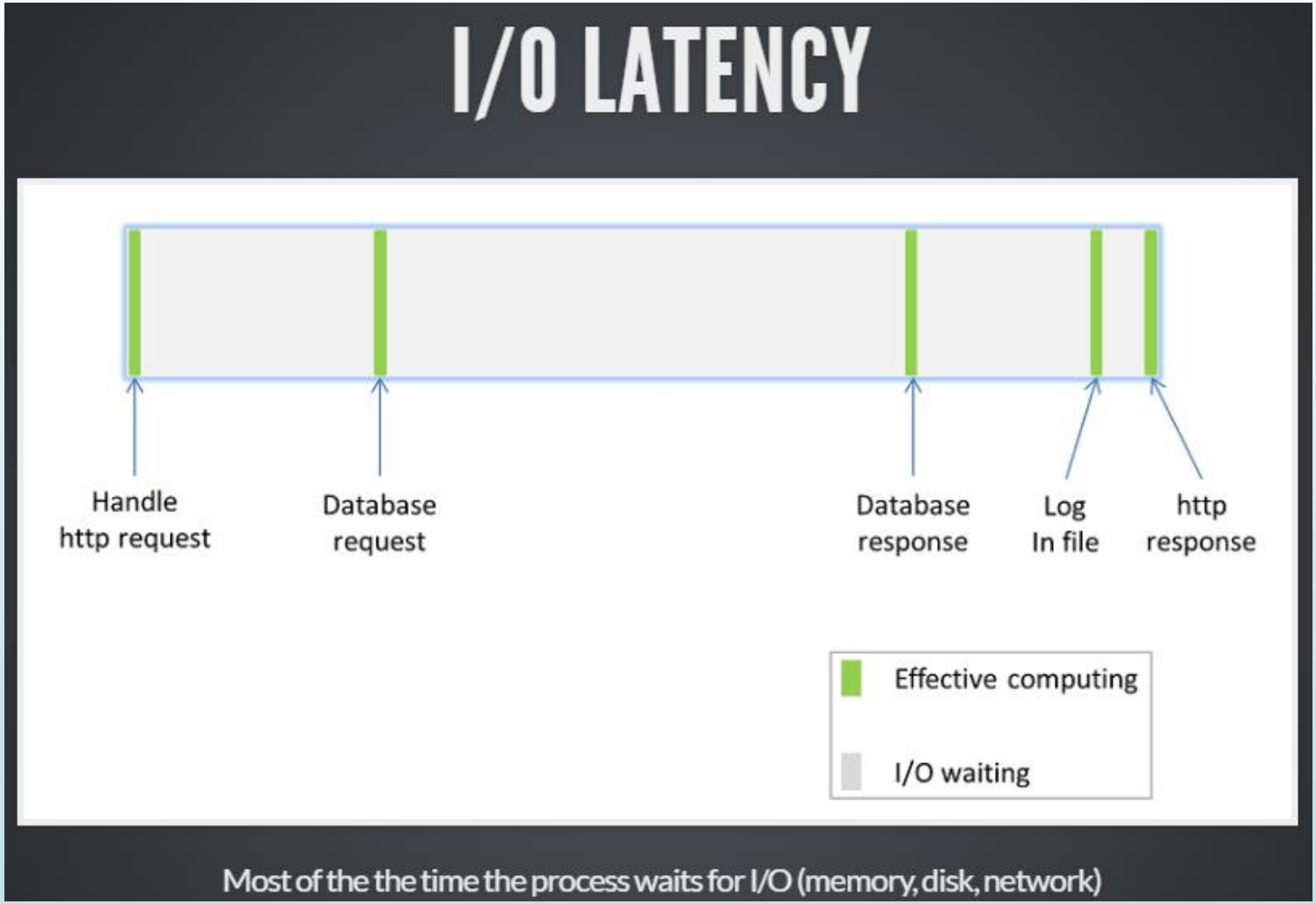
Język JavaScript również jest **jednowątkowy**, ale oferuje możliwość jednoczesnego wykonywania wielu ścieżek kodu dzięki użyciu **zdarzeń – „event loop”**.

Twórca **Node.js** (Ryan Dahl) przyjął za cel opracowanie platformy pozwalającej na obsługę bardzo dużej liczby operacji wejścia-wyjścia, aby umożliwić budowę aplikacji, w której żądania HTTP mogą pozostawać aktywne przez dłuższy czas.

Osiągnięcie tego celu z wykorzystaniem języków skryptowych jest zupełnie niemożliwe, ponieważ cały **proces będzie zablokowany** podczas przybycia żądania sieciowego.

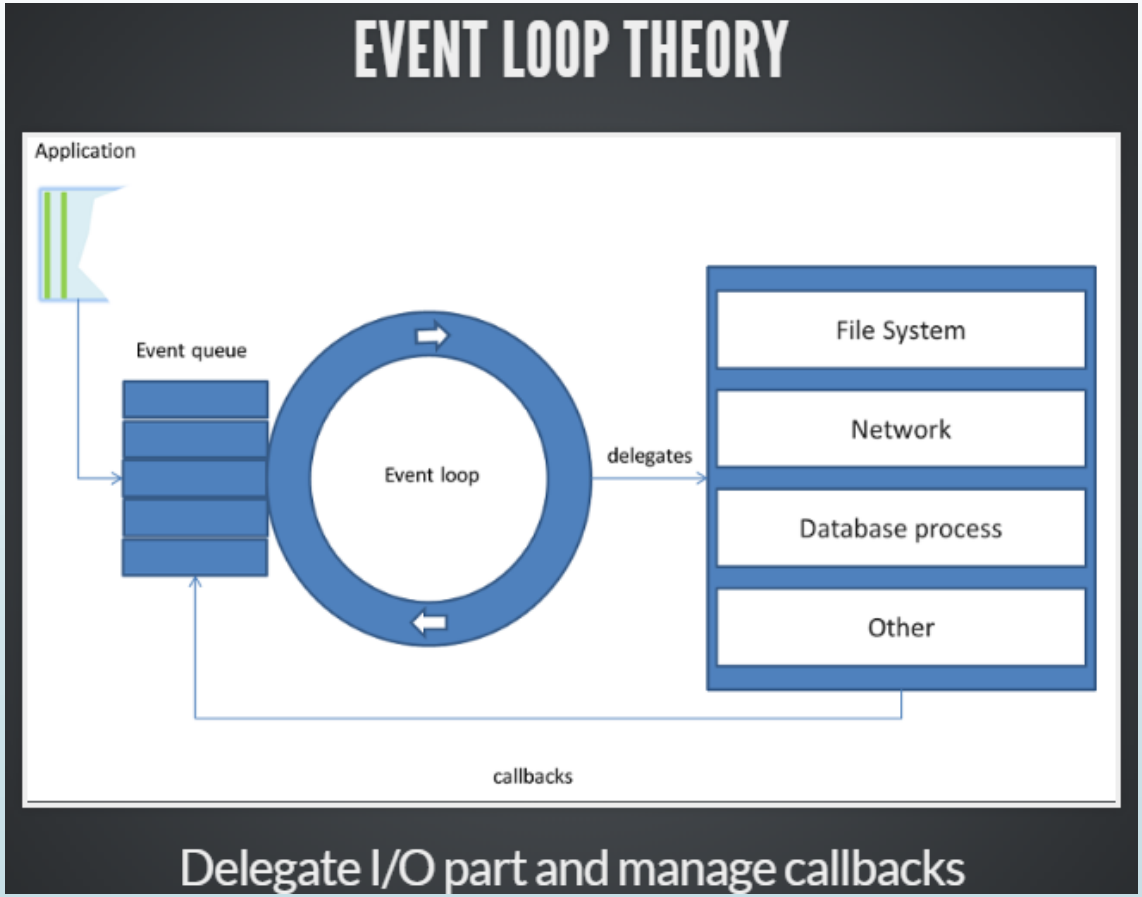
Node.js

czas oczekiwania – standardowe podejście



Node.js

„event loop” - architektura oparta na zdarzeniach



Node.js

„event loop” - architektura oparta na zdarzeniach

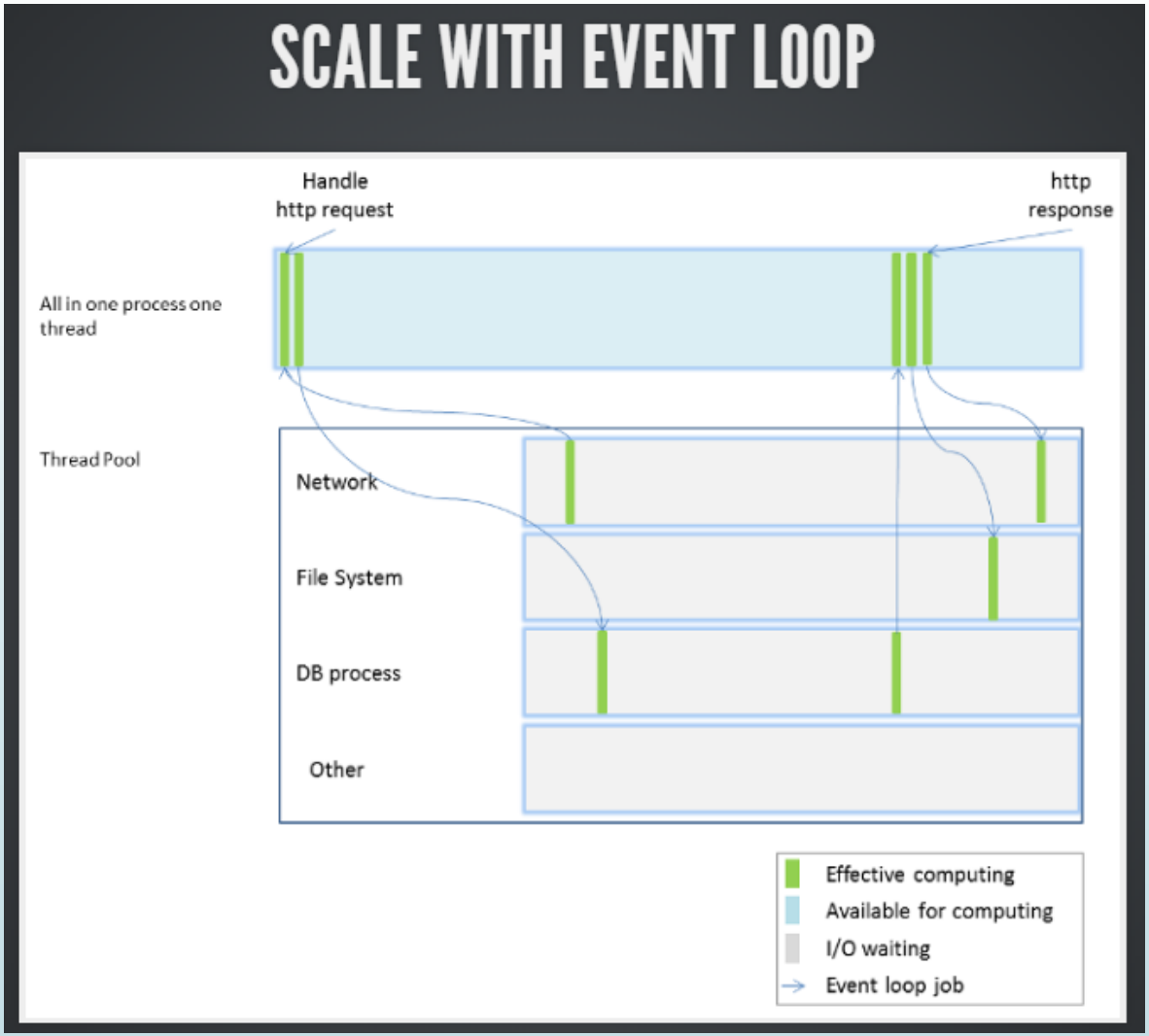
1. Node rejestruje zdarzenia i uruchamia nieskończoną pętlę, w której odpytuje jądro, aby uzyskać informację, czy zdarzenia są gotowe do przetworzenia.

2. Jeśli tak jest, uruchamia odpowiadające zdarzeniu wywołanie zwrotne, po czym pętla przechodzi dalej.

3. W przypadku braku oczekujących zdarzeń Node kontynuuje cykl do momentu pojawienia się nowych zdarzeń.

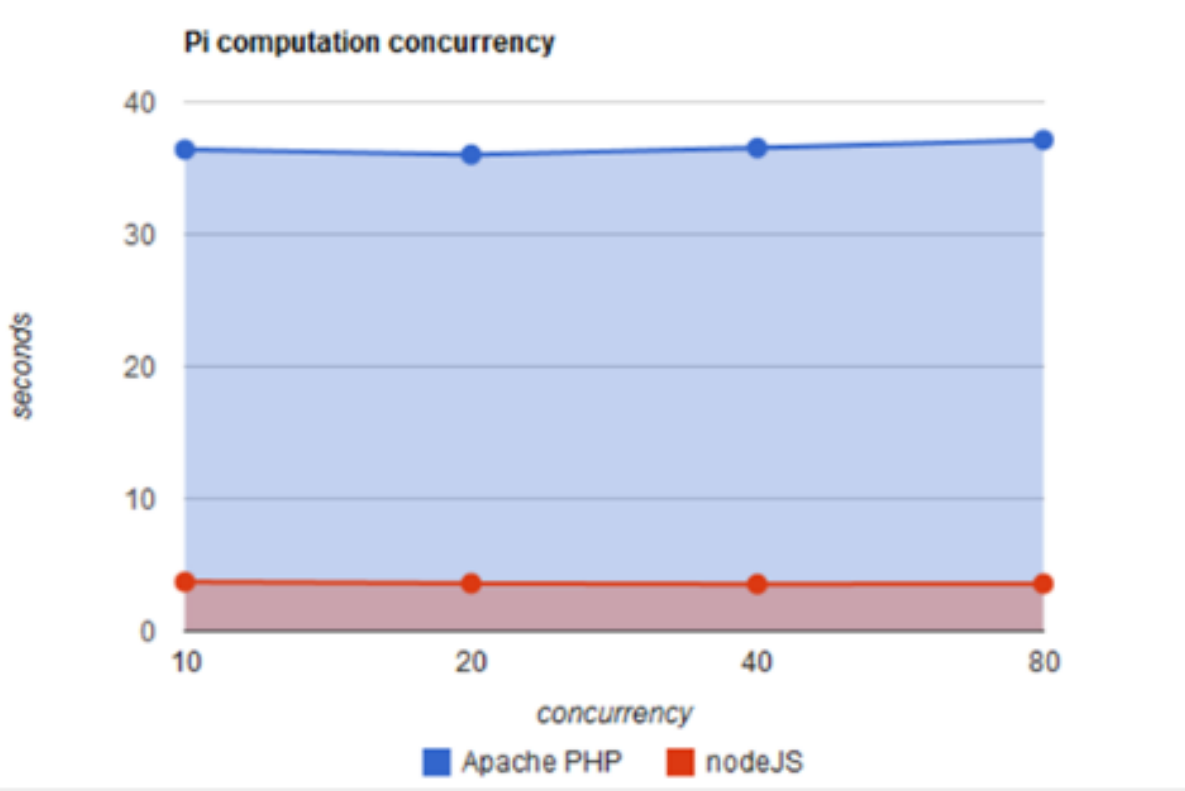
Node.js

„event loop” - architektura oparta na zdarzeniach



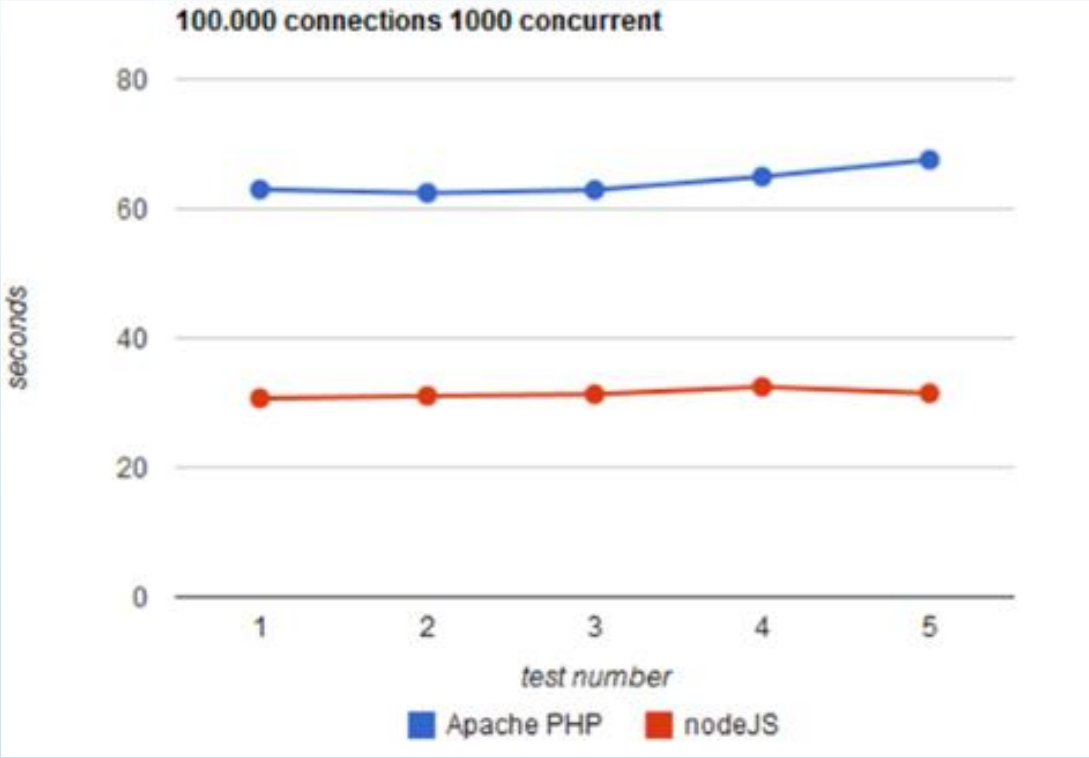
Node.js

współbieżne obliczanie liczby Pi



Node.js

prędkość przetwarzania połączeń



Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

Node.js wprowadza do JavaScript zaawansowany mechanizm:

współbieżność stanu dzielonego
(ang. shared-state concurrency)

Mechanizm ten nie funkcjonuje w tradycyjnych modelach tworzenia aplikacji sieciowych.

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

W Node.js **musisz uważać** na to, w jaki sposób Twoje **wywołania zwrotne** (ang. callbacks) modyfikują zmienne ze swojego otoczenia (stan), znajdujące się w danej chwili w pamięci.

Musisz zatem zwrócić szczególną uwagę na to, **jak obsługujesz błędy**, które mogą potencjalnie w nieprzewidziany sposób zmienić ten stan i sprawić, że proces nie będzie się nadawał do użytku.

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

JS

```
var books = [
    'Metamorfoza',
    'Zbrodnia i kara'
];

function serveBooks () {
    // wyświetlam kod HTML klientowi
    var html = '<b>' + books.join('</b><br><b>') + '</b>';
    // Jestem paskudny i zaraz zmienię stan!
    books = [];
    return html;
}
```

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

PHP

```
$books = array(
    'Metamorfoza',
    'Zbrodnia i kara'
);

function serveBooks () {
    $html = '<b>'.join($books, '</b><br><b>').'</b>';
    $books = array();
    return $html;
}
```

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

W obydwu funkcjach serveBooks zerujemy tablicę books.

Użytkownik wysyła dwa kolejne żądania /books do serwera Node i dwa kolejne żądania do serwera PHP.

Co się wtedy stanie:

- **Node obsłuży pierwsze żądanie i zwróci książki.**
  Drugie żądanie nie zwróci książek.

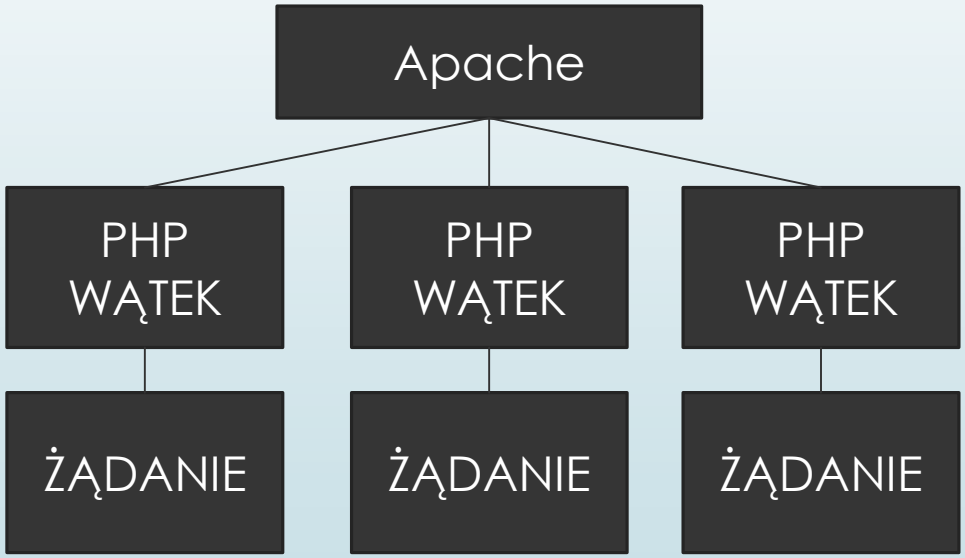
- **PHP zwróci książki w obu przypadkach.**

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

W **PHP** przy kolejnym uruchomieniu interpretera zmienna $books jest ponownie wypełniana wartościami.



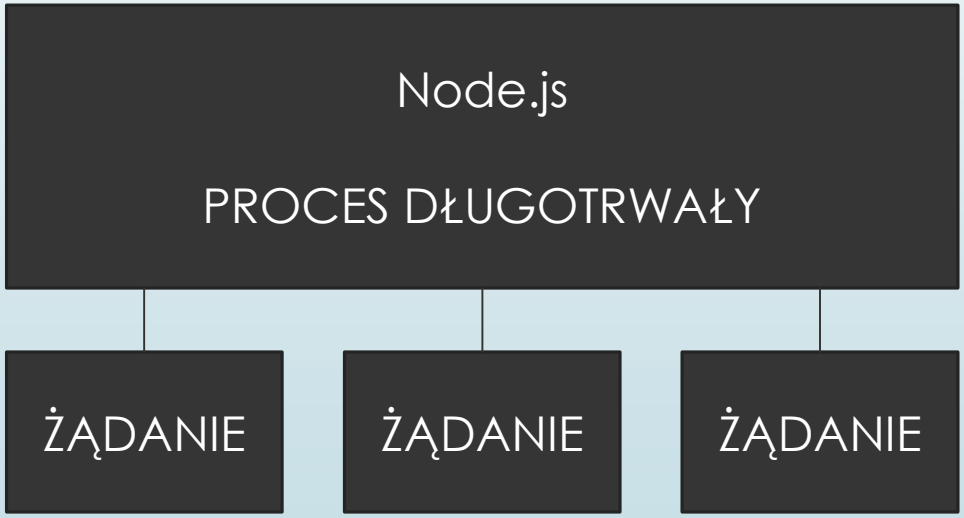
Apache tworzy jeden **wątek dla każdego żądania** rozpoczynający się każdorazowo świeżym stanem.

Node.js

„Z wielką mocą wiąże się wielka odpowiedzialność.”

Stan Lee

w Node wywoływana jest ponownie funkcja serveBooks, a zmienna pozostaje niezmieniona.



Node.js używa **pojedynczego wątku** wykonawczego.

DZIĘKUJĘ ZA UWAGĘ