NodeJS

Node.js — опенсорсная кроссплатформенная среда выполнения для JavaScript, которая работает на сервере

Основные моменты

- 2009 Ryan Dahl

Основные моменты

- 2009 Ryan Dahl
- JavaScript

Основные моменты

- 2009 Ryan Dahl
- JavaScript
- Движок V8 от Google

Основные моменты

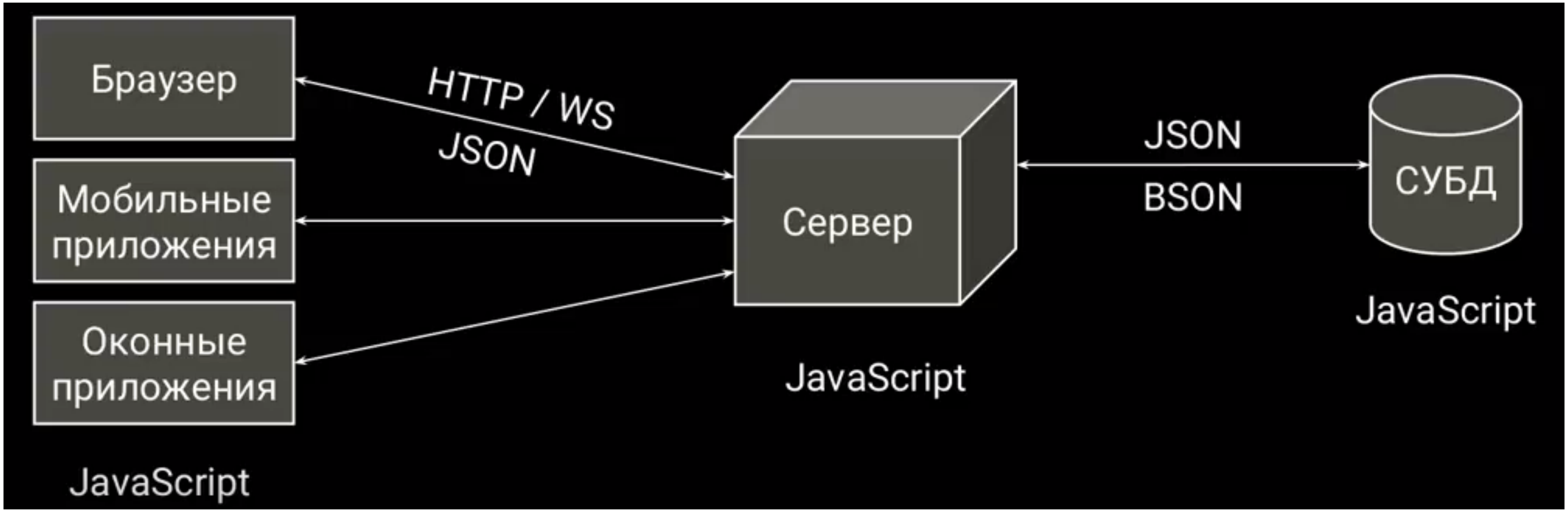
- 2009 Ryan Dahl
- JavaScript
- Движок V8 от Google
- Событийный асинхронный I/O

Основные моменты

- 2009 Ryan Dahl
- JavaScript
- Движок V8 от Google
- Событийный асинхронный I/O
- libUV – ядро

Основная идея

Один язык, один формат данных, одна парадигма, одна архитектура

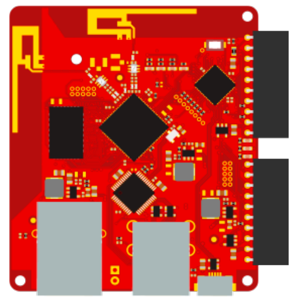


Не только Web

React Native

Electron

Tessel



Espruino

Hello world на сервере

server.js

```javascript
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

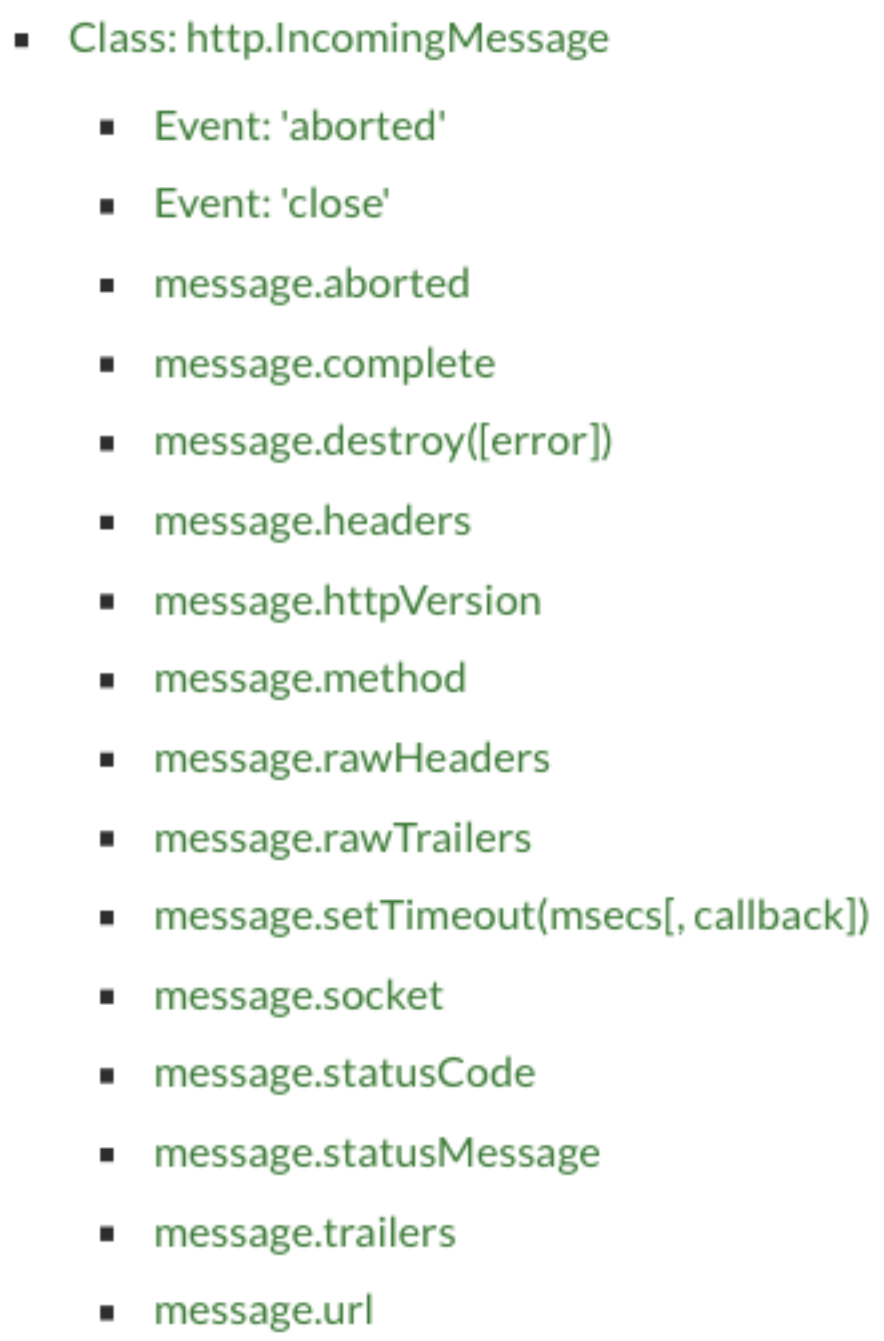
server.js

```
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 server.js
```

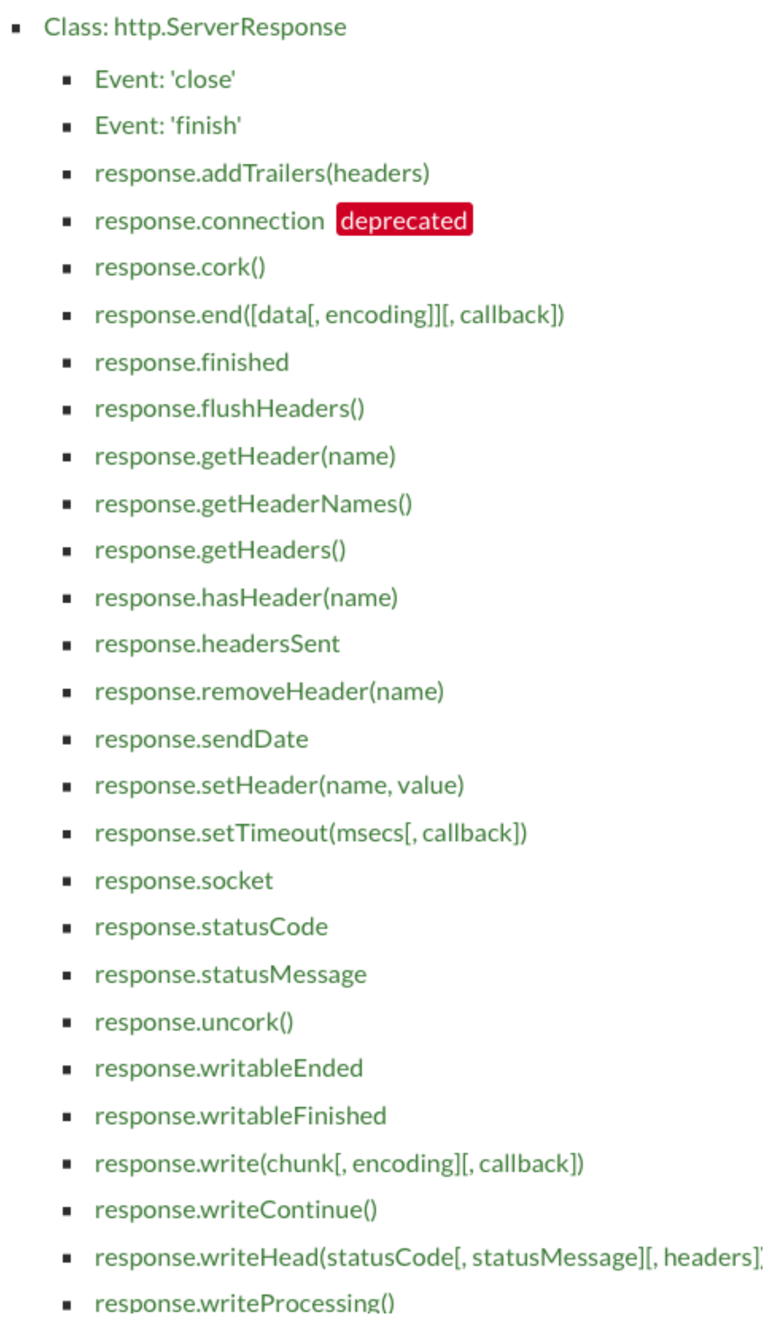
```
http://127.0.0.1:3000
```

request


- Class: http.IncomingMessage
  - Event: 'aborted'
  - Event: 'close'
  - message.aborted
  - message.complete
  - message.destroy([error])
  - message.headers
  - message.httpVersion
  - message.method
  - message.rawHeaders
  - message.rawTrailers
  - message.setTimeout(msecs[, callback])
  - message.socket
  - message.statusCode
  - message.statusMessage
  - message.trailers
  - message.url

[http.IncomingMessage]

response


- Class: http.ServerResponse
  - Event: 'close'
  - Event: 'finish'
  - response.addTrailers(headers)
  - response.connection deprecated
  - response.cork()
  - response.end([data[, encoding]][, callback])
  - response.finished
  - response.flushHeaders()
  - response.getHeader(name)
  - response.getHeaderNames()
  - response.getHeaders()
  - response.hasHeader(name)
  - response.headersSent
  - response.removeHeader(name)
  - response.sendDate
  - response.setHeader(name, value)
  - response.setTimeout(msecs[, callback])
  - response.socket
  - response.statusCode
  - response.statusMessage
  - response.uncork()
  - response.writableEnded
  - response.writableFinished
  - response.write(chunk[, encoding][, callback])
  - response.writeContinue()
  - response.writeHead(statusCode[, statusMessage][, headers]
  - response.writeProcessing()

[http.ServerResponse]

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

server.js

```
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 server.js
```

```
http://127.0.0.1:3000
```

Фреймворки

Express

Koa

Next

Socket.io

Micro

Express

Файловая структура

```
app.js

/bin
    www

package.json

/node_modules
    [...]

/public
    /images
    /javascripts
    /stylesheets
        style.css

/routes
    index.js
    users.js

/views
    error.pug
    index.pug
    layout.pug
```

/bin/www

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
...
```

app.js [1]

```
var express = require('express');
var app = express();
...
module.exports = app;
```

app.js [2]

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
```

```
var index = require('./routes/index');
var users = require('./routes/users');
```

app.js [3]

```
var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

app.js [4]

```
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
```

```
app.use('/', index);
app.use('/users', users);
```

app.js [5]

```
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err
: {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});


 module.exports = app;
```

Routes

/routes/users.js

```
var express = require('express');
var router = express.Router();

/* GET users listing. */
router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

module.exports = router;
```

Параметры routes

http://localhost:3000/users/34/books/8989

```
app.get('/users/:userId/books/:bookId', function (req, res) {
  // доступ к userId через: req.params.userId
  // доступ к bookId через: req.params.bookId
  res.send(req.params);
})
```

```
get(), post(), put(), delete(), options(), patch(), ...
```

Подробнее про Routes в Express

Views

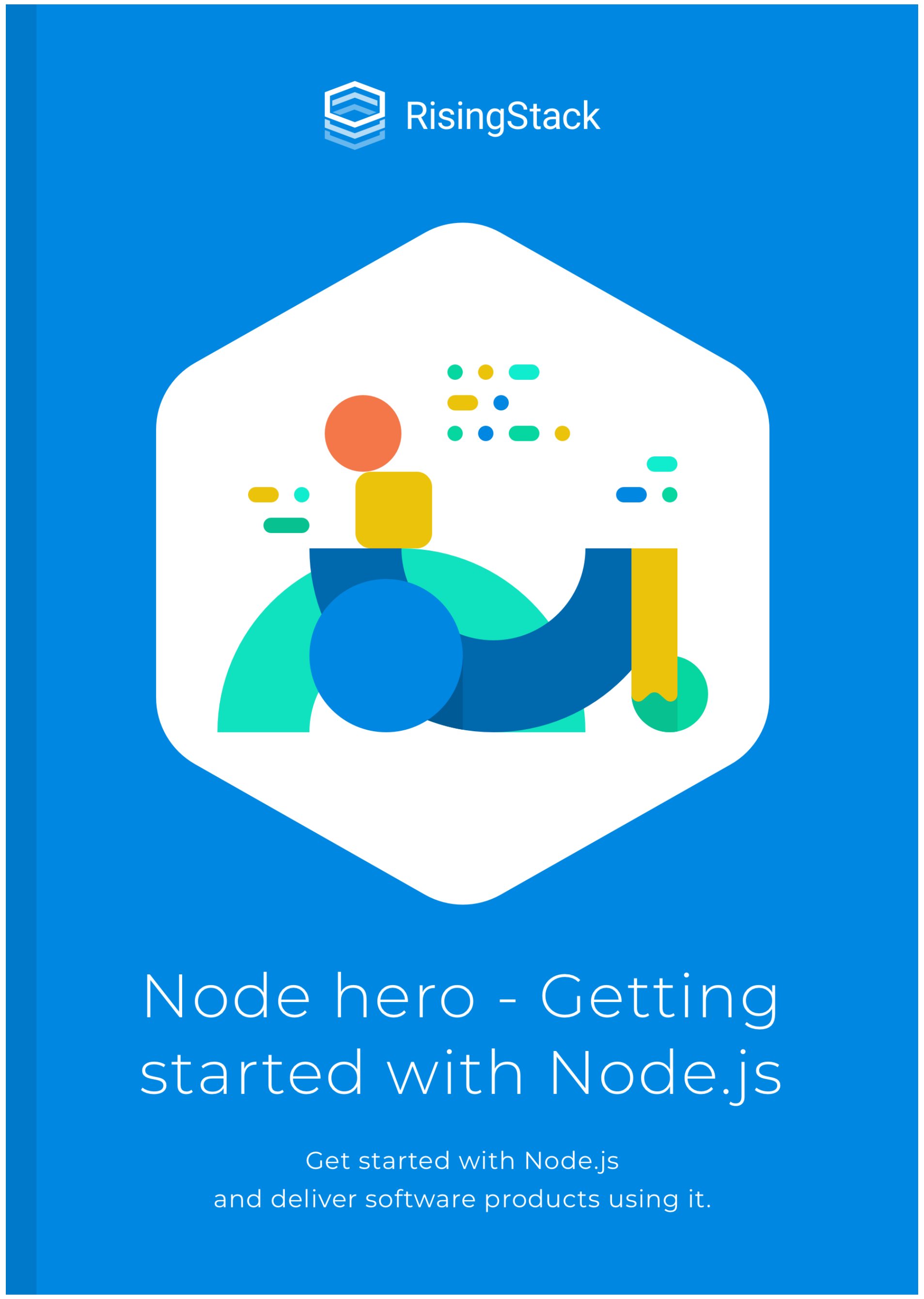
/routes/index.js

```
/* GET home page. */
router.get('/', function(req, res) {
  res.render('index', { title: 'Express' });
});
```

/views/index.pug

```
extends layout

block content
  h1= title
  p Welcome to #{title}
```

Node Hero



Node hero - Getting started with Node.js

Get started with Node.js
and deliver software products using it.

[Оригинал](#)

[На русском языке](#)

Event Loop

EN RU

+ Цикл событий, стек вызовов, таймеры

Цикл статей по NodeJS

Спасибо!