express js

Fast, Unopinionated, Minimalist Web Framework for Node.js

Node HTTP

Node's built in module 'http' allow you run basic http server

```javascript
//contents of index.js
const http = require('http')
const port = 3000

const requestHandler = (request, response) => {
  console.log(request.url)
  response.end('Hello Node.js Server!')
}

const server = http.createServer(requestHandler)

server.listen(port, (err) => {
  if (err) {
    return console.log('something bad happened', err)
  }

  console.log(`server is listening on ${port}`)
})
```

The http module is very low-level - creating a complex web application using the snippet above is very time-consuming.

This is the reason why we need expressjs

express js

- A routing & sugar layer on top of Node HTTP Server
- Declarative routing
- Basic middleware pattern

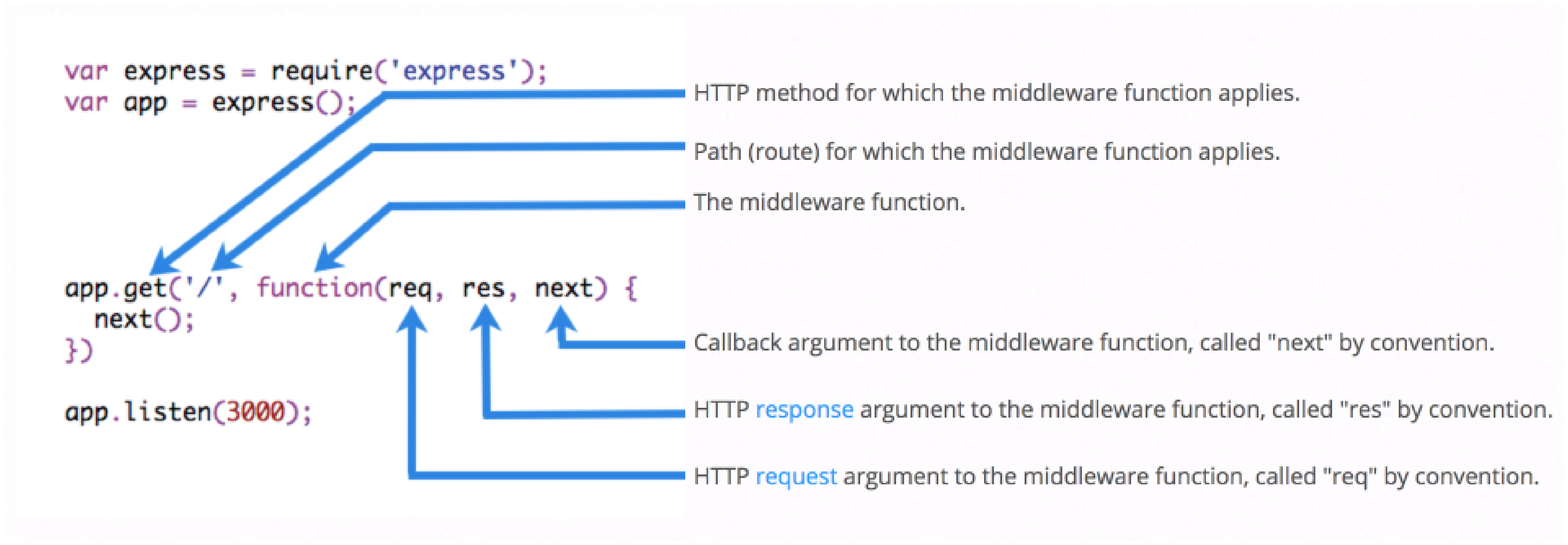
Routing

Definition of application end points and how they respond to client requests

Basic Route

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

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

app.listen(3000)
```

Basic Route



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

app.get('/', function(req, res, next) {
  next();
})

app.listen(3000);
```

HTTP method for which the middleware function applies.
Path (route) for which the middleware function applies.
The middleware function.
Callback argument to the middleware function, called "next" by convention.
HTTP response argument to the middleware function, called "res" by convention.
HTTP request argument to the middleware function, called "req" by convention.

Route Definition

app.METHOD(PATH, HANDLER)

- app is an instance of express
- METHOD is am HTTP request method in lowercase
- PATH is a path on server
- HANDLER is the function executed when route is matched

Sample Route Defintions

```
app.get("/", function(req, res) {

    //--
});


app.post("/", function(req, res) {

    //--
});

app.put("/", function(req, res) {

    //--
});

app.delete("/", function(req, res) {

    //--
});
```

Sample Route Defintions

```
app.get("/", function(req, res) {

    //--
});


app.post("/", function(req, res) {

    //--
});

app.put("/", function(req, res) {

    //--
});

app.delete("/", function(req, res) {

    //--
});
```

<= GET http://localhost:3000/

<= POST http://localhost:3000/

<= PUT http://localhost:3000/

<= DELETE http://localhost:3000/

Supported Route Methods

- get
- post
- put
- delete
- options
- head
- trace
- copy

more..

special routing method - all

```
app.all("/secret", function(req,res,next){

    //...
    //...

    //pass control to next handler
    next();
});
```

Route Path

In combination with request method, Route path define the end points at which requests can be made

strings, string patterns and regular expressions are allowed in paths

```
'/'

'/abcd'

'/ab?cd'  //match acd, abcd

'/ab*cd'  //match abcd, abbcd, abbbcd and so on..
```

Route Parameters

Named URL segments are used to capture values specified at position of URL

Captured values are available in req.params object

```
app.get('/users/:userid/books/:bookid', function(req, res, next)

    var userid = req.params.userid;
    var bookid = req.params.bookid;

    // ---

});
```

Route Handlers

Handler is a callback function to handle the request. You can provide multiple callback functions for single request.

```
app.get('/', function(req, res, next) {

    // ---
    next();

},

function(req,res,next){
    // --
}

);
```

Route Handlers

Array of Callback functions can be supplied

```
var cb1 = function(req,res,nex) { //-- next(); }

var cb2 = function(req,res,nex) { //-- next(); }

var cb3 = function(req,res,nex) { //-- next(); }


app.get('/', [cb1,cb2,cb3]);


//User signup
app.get('/users', [saveUser,sendWelcomeLetter,addUserToMailchimp
```

Combination of individual and array of functions allowed

Response Methods

Response method send response to client and end the request-response cycle.Otherwise, client request will left hanging

```
app.get('/', function(req,res,next) {

    res.send('Send this to client and end');

});
```

Combination of individual and array of functions allowed

Response Methods

Response method send response to client and end the request-response cycle.Otherwise, client request will left hanging

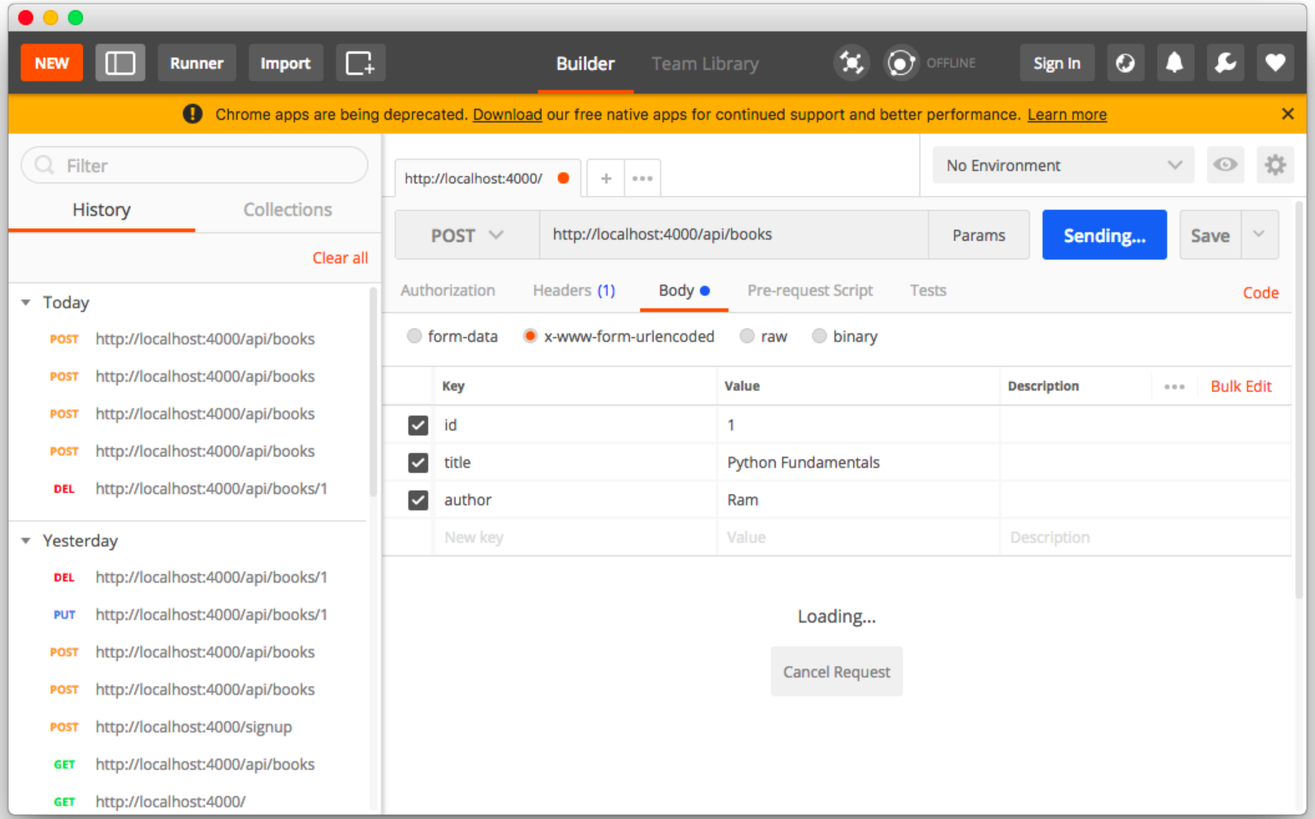
```
app.get('/', function(req,res,next) {

    res.send('Send this to client and end');

});
```

Method	Description
res.download()	Prompt a file to be downloaded.
res.end()	End the response process.
res.json()	Send a JSON response.
res.jsonp()	Send a JSON response with JSONP support.
res.redirect()	Redirect a request.
res.render()	Render a view template.
res.send()	Send a response of various types.
res.sendFile()	Send a file as an octet stream.
res.sendStatus()	Set the response status code and send its string representation as the response body.

Node API Testing

Chrome Postman

Install Postman chrome plugin from Chrome Webstore



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

Today

- POST http://localhost:4000/api/books
- POST http://localhost:4000/api/books
- POST http://localhost:4000/api/books
- POST http://localhost:4000/api/books
- DEL http://localhost:4000/api/books/1

Yesterday

- DEL http://localhost:4000/api/books/1
- PUT http://localhost:4000/api/books/1
- POST http://localhost:4000/api/books
- POST http://localhost:4000/api/books
- POST http://localhost:4000/signup
- GET http://localhost:4000/api/books
- GET http://localhost:4000/

http://localhost:4000/

No Environment

POST http://localhost:4000/api/books

form-data [✓ x-www-form-urlencoded] raw binary

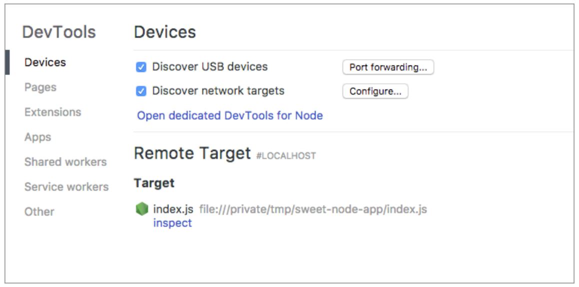
	Key	Value	Description
✓	id	1	
✓	title	Python Fundamentals	
✓	author	Ram	
	New key	Value	Description

Loading...

Node Debugging

Steps to debug:

- **Run node with the --inspect flag:**
  - $ node --inspect index.js
- **Open about:inspect in Chrome**
  - It'll redirect you to chrome://inspect quickly and you'll see something like:



DevTools — Devices

Devices | Pages | Extensions | Apps | Shared workers | Service workers | Other

- [✓] Discover USB devices — Port forwarding...
- [✓] Discover network targets — Configure...

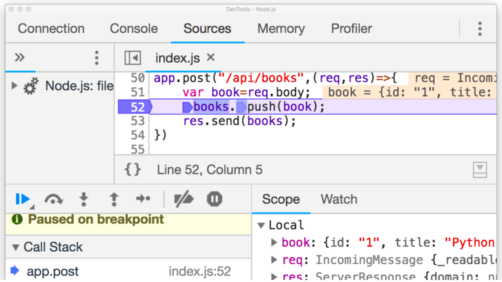
Open dedicated DevTools for Node

Remote Target #LOCALHOST

Target

index.js file:///private/tmp/sweet-node-app/index.js
inspect

- **Click the *Open dedicated DevTools for Node* link**
  - You'll get a popup window for debugging your node session.



DevTools - Node.js

Connection | Console | Sources | Memory | Profiler

Node.js: file — index.js

```
app.post("/api/books",(req,res)=>{   req = Incomi
    var book=req.body;   book = {id: "1", title:
    books.push(book);
    res.send(books);
})
```

Line 52, Column 5

Paused on breakpoint

Call Stack
app.post index.js:52

Scope | Watch

Local
- book: {id: "1", title: "Python
- req: IncomingMessage {_readabl
- res: ServerResponse {domain: n

Node Debugging

For more information on debugging,
Watch this video

app.route()

You can create chainable route handlers for a route path by using app.route().

```
app.get('/book', function (req, res) {
    res.send('Get a random book')
  });

app.post('/book', function (req, res) {
    res.send('Add a book')
  });

app.put('/book', function (req, res) {
    res.send('Update the book')
  })
```

app.route()

You can create chainable route handlers for a route path by using app.route().

```
app.route('/book')
  .get(function (req, res) {
    res.send('Get a random book')
  })
  .post(function (req, res) {
    res.send('Add a book')
  })
  .put(function (req, res) {
    res.send('Update the book')
  })
```

express.Router()

Use the **express.Router** class to create modular, mountable route handlers. A **Router** instance is a complete middleware and routing system

```
//books.js

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


// define the home page route
router.get('/', function (req, res) {
  res.send('Books')
})
// define the about route
router.get('/:id', function (req, res) {
  res.send('Single book')
})

module.exports = router;
```

```
//users.js

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


// define the home page route
router.get('/', function (req, res) {
  res.send('Users')
})
// define the about route
router.get('/:id', function (req, res) {
  res.send('Single user')
})

module.exports = router;
```

```
//Server.js

var books = require('./books')
var users = require('./users')

// ...

app.use('/books', books)
```

API Security