Express cheatsheet

A quick reference cheatsheet for Express, a flexible and streamlined web framework for Node.js

Getting Started

Hello World

"Create project, add `package.json` configuration

```
$ mkdir myapp # create directory
$ cd myapp    # enter the directory
$ npm init -y # Initialize a configura
```

Install dependencies

```
$ npm install express
```

Entry file `index.js` add code:

```
const express = require('express')
const app = express()
const port = 3000
app.get('/', (req, res) => {
  res. send('Hello World!')
})
app. listen(port, () => {
  console.log(`Listening port on ${por
})
```

Run the application using the following command

```
$ node index.js
```

express -h

```
Usage: express [options] [dir]
Options:
  -h, --help output usage information
      --version output version number
  -e, --ejs add ejs engine support
      --hbs add hbs engine support
      --pug add pug engine support
  -H, --hogan add hogan.js engine support
      --no-view No view engine generated
  -v, --view <engine> add view <engine> su
  -c, --css <engine> add stylesheet <engir
      --git add .gitignore
  -f, --force force non-empty directories
```

Create a `myapp` project

```
$ express --view=pug myapp
# run the application
$ DEBUG=myapp:*npm start
```

express()

`express.json()`	#
`express.raw()`	#
`express.Router()`	#
`express.static()`	#
`express.text()`	#
`express.urlencoded()`	#

Router

`router.all()`	#
`router.METHOD()`	#
`router.param()`	#
`router.route()`	#
`router.use()`	#

Application

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

console.dir(app.locals.title)
//=> 'My App'
console.dir(app.locals.email)
//=> 'me@myapp.com'
```

Attribute	
`app.locals`	Local variables in the application #
`app.mountpath`	Path pattern for mounting sub-apps #

Events	
`mount`	The child application is mounted on the parent application, and the event is triggered on the child application #

Method	
`app.all()`	#
`app.delete()`	#
`app.disable()`	#
`app.disabled()`	#
`app.enable()`	#
`app.enabled()`	#
`app.engine()`	#
`app.get(name)`	#
`app.get(path, callback)`	#
`app.listen()`	#
`app.METHOD()`	#
`app.param()`	#

Request

Attribute	
`req.app`	#
`req.baseUrl`	#
`req.body`	#
`req.cookies`	#
`req.fresh`	#
`req.hostname`	#
`req.ip`	#
`req.ips`	#
`req.method`	#
`req.originalUrl`	#
`req.params`	#
`req.path`	#
`req.protocol`	#
`req.query`	#
`req.route`	#
`req.secure`	#
`req.signedCookies`	#
`req.stale`	#
`req.subdomains`	#
`req.xhr`	#

Method	
`req.accepts()`	#
`req.acceptsCharsets()`	#

Response

```
app.get('/', function (req, res) {
  console.dir(res.headersSent) //false
  res.send('OK')
console.dir(res.headersSent) //true
})
```

Attribute	
`res.app`	#
`res.headersSent`	#
`res.locals`	#

Method	
`res.append()`	#
`res.attachment()`	#
`res.cookie()`	#
`res.clearCookie()`	#
`res.download()`	Prompt for files to download #
`res.end()`	end the response process #
`res.format()`	#
`res.get()`	#
`res.json()`	Send JSON response #
`res.jsonp()`	Send a response with JSONP support #
`res.links()`	#
`res.location()`	#
`res.redirect()`	Redirect request #
`res.render()`	render view template #

app.path()	#
app.post()	#
app.put()	#
app.render()	#
app.route()	#
app.set()	#
app.use()	#

req.acceptsEncodings()	#
req.acceptsLanguages()	#
req.get()	Get HTTP request header fields #
req.is()	#
req.param()	#
req.range()	#

res.send()	Send various types of responses #
res.sendFile()	Send a file as an octet stream #
res.sendStatus()	#
res.set()	#
res.status()	#
res.type()	#
res.vary()	#

Example

Router

Called for any request passed to this router

```
router. use(function (req, res, next) {
  //.. some logic here .. like any other m
  next()
})
```

will handle any request ending in `/events`

```
//depends on where the router "use()"
router. get('/events', (req, res, next) =>
  //..
})
```

Response

The `res` object represents the HTTP response sent by the `Express` application when it receives an HTTP request

```
app.get('/user/:id', (req, res) => {
  res.send('user' + req.params.id)
})
```

Request

A `req` object represents an `HTTP` request and has properties for the request query string, parameters, body, HTTP headers, etc.

```
app.get('/user/:id', (req, res) => {
  res.send('user' + req.params.id)
})
```

res. end()

```
res. end()
res.status(404).end()
```

End the response process. This method actually comes from the Node core, specifically the `response.end()` method of `http.ServerResponse`

res.json([body])

```
res.json(null)
res.json({ user: 'tobi' })
res.status(500).json({ error: 'message' })
```

app.all

```
app.all('/secret', function (req, res, nex
  console.log('access secret section...')
  next() // Pass control to the next handl
})
```

app.delete

```
app.delete('/', function (req, res) {
  res.send('DELETE request to homepage')
})
```

app.disable(name)

```
app.disable('trust proxy')
app.get('trust proxy')
// => false
```

app.disabled(name)

```
app.disabled('trust proxy')
// => true

app.enable('trust proxy')
app.disabled('trust proxy')
// => false
```

app.engine(ext, callback)

```
var engines = require('consolidate')

app.engine('haml', engines.haml)
app.engine('html', engines.hogan)
```

app.listen([port[, host[, backlog]]][, callback])

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

var app = express()
app.listen(3000)
```

Routing

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

//Respond to "hello world" when making a G
app.get('/', (req, res) => {
  res.send('hello world')
})


// GET method routing
app.get('/', (req, res) => {
  res.send('GET request to the homepage')
})

// POST method routing
app.post('/', (req, res) => {
  res.send('POST request to the homepage')
})
```

Middleware

```
function logOriginalUrl (req, res, next) {
  console.log('ReqURL:', req.originalUrl)
  next()
}

function logMethod (req, res, next) {
  console.log('Request Type:', req.method)
```

Using templates

```
app.set('view engine', 'pug')
```

Create a Pug template file named `index.pug` in the `views` directory with the following content

```
html
  the head
```

```
  next()
}

const log = [logOriginalUrl, logMethod]

app.get('/user/:id', log,
  (req, res, next)=>{
    res.send('User Info')
  }
)
```

```
    title= title
  the body
    h1=message
```

Create a route to render the `index.pug` file. If the view engine property is not set, the extension of the view file must be specified

```
app.get('/', (req, res) => {
  res. render('index', {
    title: 'Hey', message: 'Hello there!'
  })
})
```

Related Cheatsheet

- **ES6** Cheatsheet Quick Reference
- **JSON** Cheatsheet Quick Reference
- **Kubernetes** Cheatsheet Quick Reference
- **TOML** Cheatsheet Quick Reference

Recent Cheatsheet

- **Kotlin** Cheatsheet Quick Reference
- **Express** Cheatsheet Quick Reference
- **ChatGPT** Cheatsheet Quick Reference
- **Google Search** Cheatshee Quick Reference

QuickRef.ME

Share quick reference and cheat sheet for developers.

中文版 #Notes