node JS®

Installation

```
sudo apt install nodejs npm   #ubuntu
brew install node   #macos

sudo npm install −global n   # sudo npm i -g n

sudo n stable
```

other variants
https://nodejs.org/en/download

console

- log(), info()
- error(), warn()
- time()
- timeEnd()

```
console.time('100-elements');
for (let i = 0; i < 100; i++) {
    //code
}
console.timeEnd('100-elements');
```

process

- `console.log(process);`
    - `console.log(process.version);`
    - `console.log(process.platform);`
    - `console.log(process.title);`
    - `console.log(process.execPath);`
    - `console.log(process.arch);`
    - `console.log(process.memoryUsage());`
    - `console.log(process.env);`

Path

```
const path = require("path");
```

-
  ```
  const filename = path.basename(_filename);
  console.log(filename); //server.js
  ```

-
  ```
  path.resolve( ) // converts relative address to absolute
  console.log(path.resolve("./server"));
  ```

-
  ```
  const extention = path.extname(_filename);
  console.log(extention); //.js
  ```

-
  ```
  path.isAbsolute( )
  console.log(path.isAbsolute(_filename)); //true
  console.log(path.isAbsolute("./server")); //false
  ```

Path

- `path.join( );`

```
const file = "index.html";
const filePath = path.join(_dirname, file);
console.log(filePath); // /home/tigran/apps/node/index.html
```

- `path.parse( );`

```
const pathParse = path.parse(filePath);
console.log(pathParse);
// {
//   root: '/',
//   dir: '/home/tigran/apps/node',
//   base: 'index.html',
//   ext: '.html',
//   name: 'index'
// }
```

Fs

```
const fs  = require('fs'); // to use the callback and sync APIs:
const fsp  = require('fs/promises'); // To use the promise-based APIs:

const data = fs.readFileSync('./file.txt', 'utf8'); // read file data

fs.writeFileSync('./file.txt', 'text here'); // write file data

fs.renameSync('./file.txt', './newFile.txt'); // rename or move file

fs.copyFileSync('./file.txt', './newFile.txt'); // copy file

fs.unlinkSync('./file.txt'); // delete file

fs.rmdirSync(_dirname, { recursive: true }); // delete folder

fs.rmSync('./file.txt', { recursive: true }); // delete file or directory

const data = fs.readdirSync(_dirname); // get directory files list
```

Fs

- `fs.stat( )`

```
const stats = fs.stat("./file.txt");
```

- `stats.isFile() //checks, the object is considered a file.`
- `stats.isDirectory() //checks, the object is considered a directory.`

- `stats.size`
- `stats.atime // access time`
- `stats.mtime // modified time`
- `stats.ctime // change time`
- `stats.birthtime // birthtime`

Server

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


const port = 8080;

const hostname = "127.0.0.1";


const server = http.createServer((request, response) => {

  response.statusCode = 200;

  response.setHeader("Content-Type", "text/plain");

  response.end("Hello from Node.js");

});


server.listen(port, hostname, () => {

  console.log(Server running on port ${port}...);

});
```

Static Server

```
...
const mime = require("mime-types");


const server = http.createServer((req, res) => {
  let { pathname } = url.parse(req.url);
  if (pathname === '/')  pathname = 'index.html';
  const filePath = path.join(__dirname, 'data', pathname);
  if (fs.existsSync(filePath)) {
    const data = fs.readFileSync(filePath)
    res.setHeader('Content-Type', mime.lookup(filePath));
    res.end(data);
  } else {
    res.statusCode = 404;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Not Found')
  }
})
...
```

node

JS®

+

Express

Express

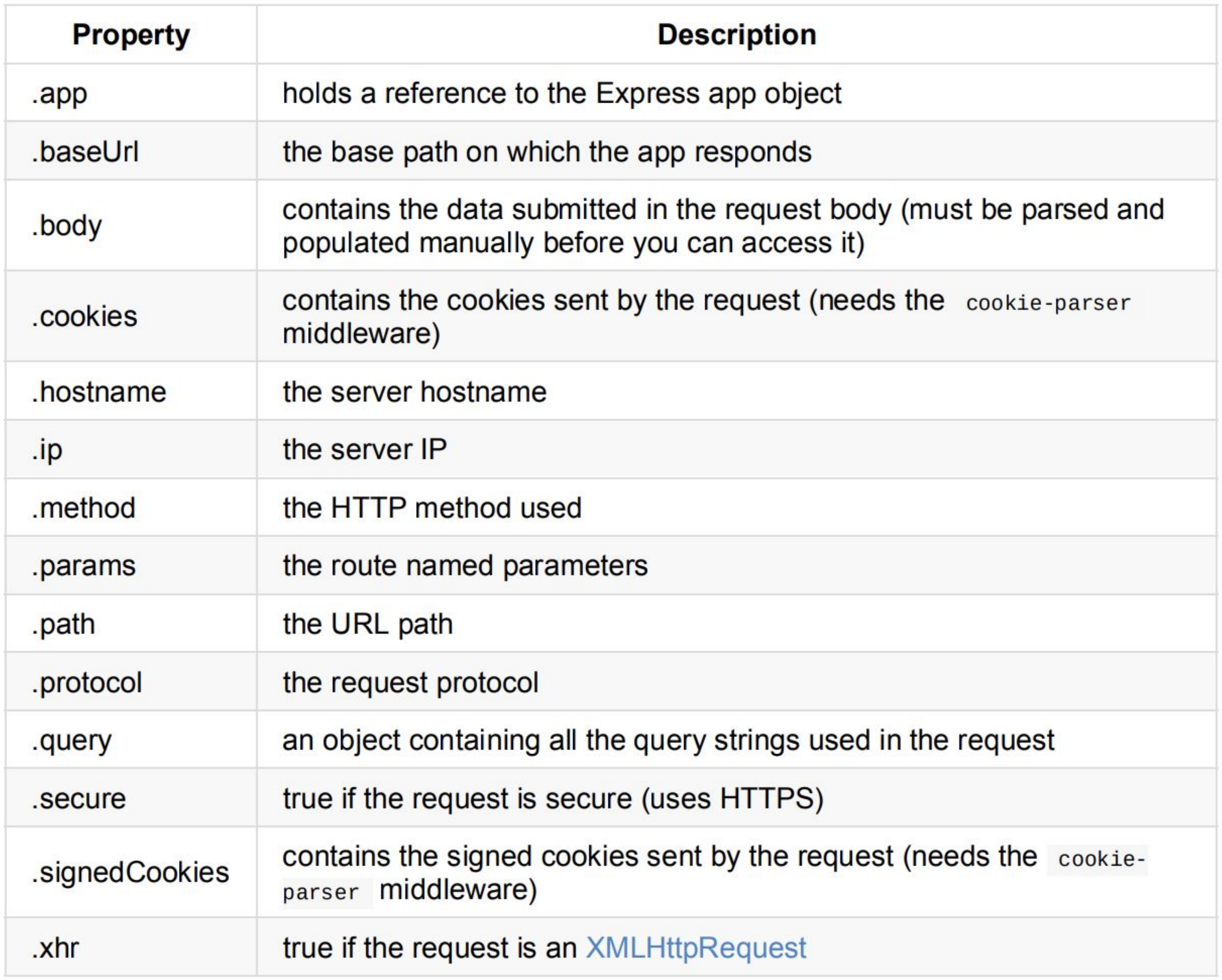
```
const express = require('express');
const app = express();

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

app.post('/', (req, res) => { /* */ });
app.put('/', (req, res) => { /* */ });
app.delete('/', (req, res) => { /* */ });
app.patch('/', (req, res) => { /* */ });

app.listen(3000, () => {
  console.log('Server ready')
})
```

Express Request parameters



Property	Description
.app	holds a reference to the Express app object
.baseUrl	the base path on which the app responds
.body	contains the data submitted in the request body (must be parsed and populated manually before you can access it)
.cookies	contains the cookies sent by the request (needs the `cookie-parser` middleware)
.hostname	the server hostname
.ip	the server IP
.method	the HTTP method used
.params	the route named parameters
.path	the URL path
.protocol	the request protocol
.query	an object containing all the query strings used in the request
.secure	true if the request is secure (uses HTTPS)
.signedCookies	contains the signed cookies sent by the request (needs the `cookie-parser` middleware)
.xhr	true if the request is an XMLHttpRequest

Express retrieve the POST data

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

app.use(express.json()); //Content-Type:application/json
app.use(express.urlencoded()); //Content-Type:application/x-www-form-urlencoded


app.post('/form', (req, res) => {
  const { name, email } = req.body;
});
```

```
. . .

app.use(express.urlencoded({
  extended: true,
  limit: '50mb'
}));

. . .
```

Express Serving static files

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

app.use(express.static('public'));

...

app.listen(3000, () => console.log('Server ready'))
```

Express HTTP response status

```
res.status(404).end();
res.status(404).send(Not found');
res.sendStatus(404);

res.sendStatus(200);
res.status(200).send('OK');

res.sendStatus(401);
res.status(401).send('Unauthorized');

res.sendStatus(403);
res.status(403).send('Forbidden');

res.sendStatus(500);
res.status(500).send('Internal Server Error');
```

Express response

```
res.send('Hello World!');  // Content Type: text/plain


res.json({ text: 'Hello World!' id: 1 }); // Content Type: application/json


res.sendFile(fileName, options, (err) => {
  // done
});

res.download(fileName, options, (err) => {
  // done
});

res.redirect('/go-there'); // status code === 302
res.redirect(301, '/go-there');  // status code === 301
res.redirect('../go-there');
res.redirect('..');
res.redirect('back');
```

Express headers

```
res.set('Access-Control-Allow-Origin', '*');

//set multiple
res.set({
   'Access-Control-Allow-Origin': '*',
   'Access-Control-Allow-Credentials': true
})

. . .
res.set('Content-Type', 'text/html');
res.type('.html'); // => 'text/html'
res.type('html'); // => 'text/html'
res.type('json'); // => 'application/json'
res.type('application/json'); // => 'application/json'
res.type('png'); // => image/png:
. . .
```

Express routing

```
app.get('/', (req, res) => {
  res.send('root')
});

app.get('/users', (req, res) => {
  res.send('users')
});

app.get('/users/:userId/books/:bookId', (req, res) => {
  res.send(req.params)
});

// This route path will match /abe and /abcde.

app.get('/ab(cd)?e', (req, res) => {

  res.send('ab(cd)?e')

});

//This route path will match butterfly and dragonfly, but not butterflyman, dragonflyman, and so on.

app.get(/.*fly$/, (req, res) => {

  res.send('/.*fly$/')

});
```

Express routing

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

router.get('/book/:id', (req, res) => {  res.send('Get a book') });
router.post('/book', (req, res) => { res.send('Add a book') });
router.put('/book/:id', (req, res) => { res.send('Update the book') });
router.delete('/book/:id', (req, res) => { res.send('Update the book') });

// or

router.route('/book')
  .get((req, res) => { res.send('Get a book') })
  .post((req, res) => { res.send('Get a book') })
  .put((req, res) => { res.send('Update the book') })
  .delete((req, res) => { res.send('Delete the book') });

module.exports = router;
```

Cookies

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

const cookieParser = require('cookie-parser');


app.use(cookieParser());


app.get('/', (req, res) => {
  res.cookie('username', 'Flavio');
  res.cookie('username', 'Flavio', {

    domain: '.flaviocopes.com',

    path: '/administrator',

    secure: true,

    expires: new Date(Date.now() + 900000),

    httpOnly: true

  });

  res.clearCookie('username')

});
```

Session

```
const express = require('express');
const session = require('express-session');
const app = express();

app.use(session({
  'secret': '343ji43j4n3jn4jk3n'
}))

app.get('/', (req, res, next) => {
  req.session.name = 'Flavio';
  console.log(req.session.name)
}

app.get('/route2, (req, res, next) => {
  console.log(req.session.name)
}
```

It can store session data in

- memory, not meant for production
- a database like MySQL or Mongo
- a memory cache like Redis or Memcached

https://github.com/expressjs/session

Custom authorization middleware

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

function authorization(req, res, next) {
  try {
    const { authorization = '' } = req.headers;
    const { email } = jwt.verify(authorization.replace('Bearer ', ''), '{SECRET}');
    req.email = email;
    next();
  } catch (e) {
    e.code = 401;
    next(e);
   }
}

app.use(authorization)

app.get('/', (req, res, next) => {
  console.log(req.email)
}
```

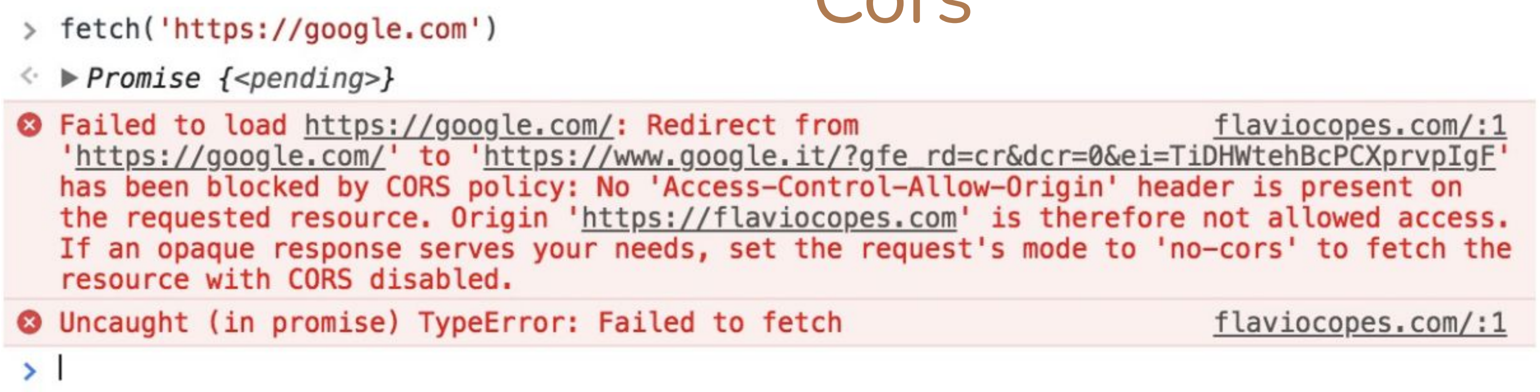
Custom error handler

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

app.get('/', (req, res, next) => {
  try {
    res.unknownFunction();
  } catch (e) {
    next(e);
  }
}

app.use((err, req, res, next) => {
  res.status(err.code || 500);
  res.json({
    status: 'error',
    message: err.message,
    errors: err.errors,
    stack: process.env.NODE_ENV !== 'production' ? err.stack: undefined
  });
})
```

Cors



```
> fetch('https://google.com')
< ▶ Promise {<pending>}
⊗ Failed to load https://google.com/: Redirect from                         flaviocopes.com/:1
  'https://google.com/' to 'https://www.google.it/?gfe_rd=cr&dcr=0&ei=TiDHWtehBcPCXprvpIgF'
  has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on
  the requested resource. Origin 'https://flaviocopes.com' is therefore not allowed access.
  If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the
  resource with CORS disabled.
⊗ Uncaught (in promise) TypeError: Failed to fetch                            flaviocopes.com/:1
> |
```

```
const WHITELIST = ['http://google.com'];

function header(req, res, next) {

  const { origin = '*' } = req.headers;

  if(WHITELIST.includes(origin)){

    res.setHeader('Access-Control-Allow-Methods', 'GET,HEAD,PUT,PATCH,POST,DELETE');

    res.setHeader('Access-Control-Allow-Origin', origin);

    res.setHeader('Access-Control-Allow-Headers', 'Origin,Accept,Content-Type,Authorization');

  }

}
```

Mysql

```
const connection = mysql.createConnection({
  host: 'localhost',
  port: 2206,
  user: 'root',
  password: 'root',
  database: 'test'
});


const db = connection.promise();

const [rows, fields] = await db.execute(

  'SELECT * FROM `table` WHERE `name` = ? AND `age` > ?', ['Morty', 14]

);
```