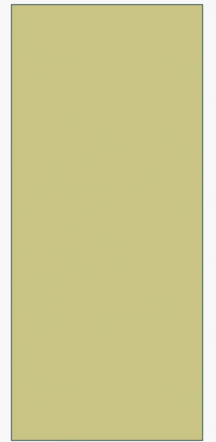


EXPRESSJS

FUNDAMENTALS



WHAT IS EXPRESS JS

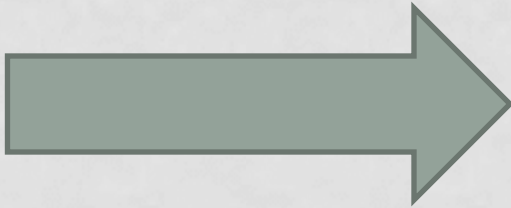
A web server that is
built for Node
to make client/server applications

INSTALLATION

You do not need
to install it

like you would with Apache,
Nginx or IIS (Microsoft)

DECLARE IN PACKAGE.JSON



```
{ } package.json x
1  {
2    "name": "express",
3    "version": "1.0.0",
4    "description": "express as dependency",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "dependencies": {
10     "express": "4.16.2"
11   },
12   "keywords": [
13     "express"
14   ],
15   "author": "Andrew Sheehan (asheehan@bu.edu)",
16   "license": "ISC"
17 }
```

REGARDING PACKAGE.JSON



Never check-in the
`node_modules` folder to your
VCS (Version Control System)

GIT, SVN

REVISITING HELLO WORLD

```
1  const express = require('express' 4.16.3 )
2  const app = express()
3  const port = 3000
4
5  app.get('/', (req, res) => res.send('Hello World!'))
6
7  app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

REQUIRE() THEN INSTANTIATE

```
/* index.js */
```

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

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

THE ROOT DIRECTORY

The root directory

is where you ran **npm init** from

ROUTING

Routing is how an application responds to a client request, **which is a URI** (or path) and a specific **HTTP request method**.
(like GET, POST or DELETE)

ROUTING METHODS

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

BASIC ROUTING STRUCTURE

```
app.METHOD(PATH, HANDLER)
```

- app is an instance of express.
- METHOD is an [HTTP request method](#), in lowercase.
- PATH is a path on the server.
- HANDLER is the function executed when the route is matched.

SIMPLE ROUTE (GET APP ROOT)

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

SIMPLE ROOT (POST TO APP ROOT)

```
app.post('/', function (req, res) {  
    res.send('Got a POST request')  
})
```

SPECIAL ROUTING

APP.ALL()

```
app.all('/secret', function (req, res, next) {  
  console.log('Accessing the secret section ...')  
  next() // pass control to the next handler  
})
```


ROUTING BASED ON PATTERNS

```
app.get('/ab?cd', function (req, res) {  
  res.send('ab?cd')  
})
```

Matches 'abd', 'abc' & 'abcd'

ROUTING: PATTERN MATCHING

Sometimes called 'handler functions'



```
app.get(/a/, (req, res) => {  
  res.send("route /a/");  
});
```


ROUTING: PATTERN MATCHING

```
app.get(/.*fly$/, function (req, res) {  
  res.send('/.*fly$/')  
})
```

Matches butterfly and
savefly, **but not flyDelete**

ROUTING: PATTERN MATCHING

querystrings are not
considered part of
the route path



ROUTING: PATTERN MATCHING



Route
parameters
must consist
of `[A-Za-z0-9]`

ROUTING: PARAMETERS

```
app.get('/books/:isbn/info',  
  (req, res) => {  
    //do something..  
  });
```

REQUEST.PARAMS



```
Route path: /users/:userId/books/:bookId
```

```
Request URL: http://localhost:3000/users/34/books/8989
```

```
req.params: { "userId": "34", "bookId": "8989" }
```

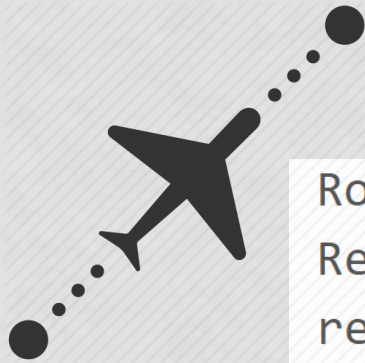
REQUEST.PARAMS



“-” & “.”

interpreted literally

REQUEST.PARAMS



Route path: `/flights/:from-:to`

Request URL: `http://localhost:3000/flights/LAX-SFO`

req.params: `{ "from": "LAX", "to": "SFO" }`

ROUTING: MIDDLEWARE


You can use multiple
callbacks (functions)
that is middleware
(security, logic tokenizing,
etc..)

ROUTING: ROUTE RESPONSES

You must
respond
client will
hang if not.




ROUTING: HANDERS: NEXT()

```
app.get('/example/b', function (req, res, next) {  
  console.log('the response will be sent by the next function ...')  
  next()   
}, function (req, res) {  
  res.send('Hello from B!')  
})
```

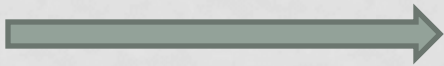
next() will invoke the next
callback in the chain (if any...)

ROUTING: HANDLERS (ARRAY)

```
var cb0 = function (req, res, next) {  
  console.log('CB0')  
var cb1 = function (req, res, next) {  
  console.log('CB1')  
var cb2 = function (req, res) {  
  res.send('Hello from C!')  
}  
  
app.get('/example/c', [cb0, cb1, cb2])
```



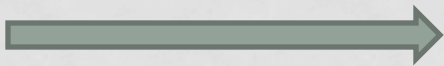
ROUTING: APP.ROUTE()



```
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')  
  })
```

ROUTING MODULES

Defining routing
modules

Modularize your paths
in separate files.

ROUTING MODULES

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

// Car brands page

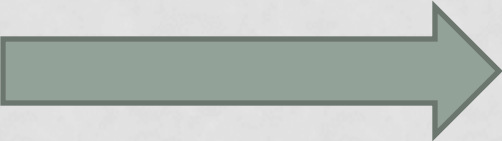
router.get('/brands', function(req, res) {
    res.send('Audi, BMW, Mercedes')
})

// Car models page

router.get('/models', function(req, res) {
    res.send('Audi Q7, BMW X5, Mercedes GL')
})

module.exports = router
```

ROUTING MODULES



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

app.use('/cars', require('./cars'))

app.listen(3000, function() {
  console.log('Listening on port 3000...')
})
```

GOOD USE CASE.

Access of your api or services
probably needs authentication...

```
router.all('/api/*', requireAuthentication);
```