RICE®

Web Development

COMP 431 / COMP 531

Lecture 16: Back End

Instructor: Mack Joyner

Department of Computer Science, Rice University

mjoyner@rice.edu

http://www.clear.rice.edu/comp431

Part II – Back End Development

Homework Assignment 5
(Front-End App)
Now Due Sunday 11/29

- HW5 Front-End App
  - Unit testing, "resource" wraps around "fetch"
  - Example:
    - https://www.clear.rice.edu/comp431/sample/profileActions.ts
    - https://www.clear.rice.edu/comp431/sample/app.component.spec.ts

What is Plagiarism?

- the practice of taking someone else's work or ideas and passing them off as one's own.

Angular Unit Test Example

profileActions.ts

```
const url = 'https://webdev-dummy.herokuapp.com'

const resource = (method, endpoint, payload) => {
    const options: RequestInit = {
        method,
        credentials: 'include',
        headers: {
            'Content-Type': 'application/json'
        },
        body: ''
    }
    if (payload) {
      options.body = JSON.stringify(payload);
    }

    return fetch(`${url}/${endpoint}`, options)
        .then(r => {
            if (r.status === 200) {
                if (r.headers.get('Content-Type').indexOf('json') > 0) {
                    return r.json();
                } else {
                    return r.text();
                }
            } else {
                // useful for debugging, but remove in production
                console.error(`${method} ${endpoint} ${r.statusText}`);
                throw new Error(r.statusText);
            }
        })
}
```

resource in unit testing requirements

Must define body here

Testing http error

*.component.spec.ts

Create DOM with ids used by logout function

```
35      const createDOM = (username, password, message) => {
36          const add = (tag, id, value) => {
37              const el = document.createElement(tag);
38              el.id = id;
39              el.value = value;
40              el.style = { display: 'inline' };
41              document.body.appendChild(el);
42              return el;
43          };
44          add('input', 'username', username);
45          add('input', 'password', password);
46          const d = add('div', 'message', message);
47          d.innerHTML = message;
48          return d;
49      };
```

```
70  it(`should log the user out`, async(() => {
71    const div = createDOM('user', 'pass', 'hello');
72    expect(div.innerHTML).toEqual('hello');
73
74    mock(`${url}/logout`, {
75          method: 'PUT',
76          headers: { 'Content-Type': 'application/json' }
77        });
78        logout()
79            .then(_ => {
80                expect(div.innerHTML).toEqual('You have logged out');
81            });
82    }));
```

Mock the logout

Angular Unit Test Example

```
50    const logout = () => {
51        const box = document.querySelector("#message");
52        return resource('PUT', 'logout','')
53            .then(r => box.innerHTML = "You have logged out")
54            .catch(r => box.innerHTML = `"${r.message}" when logging out`);
55    };
56
```

Calls resource wrapper for fetch

```
68    export { url, login, logout };
```

Angular Component Unit Test

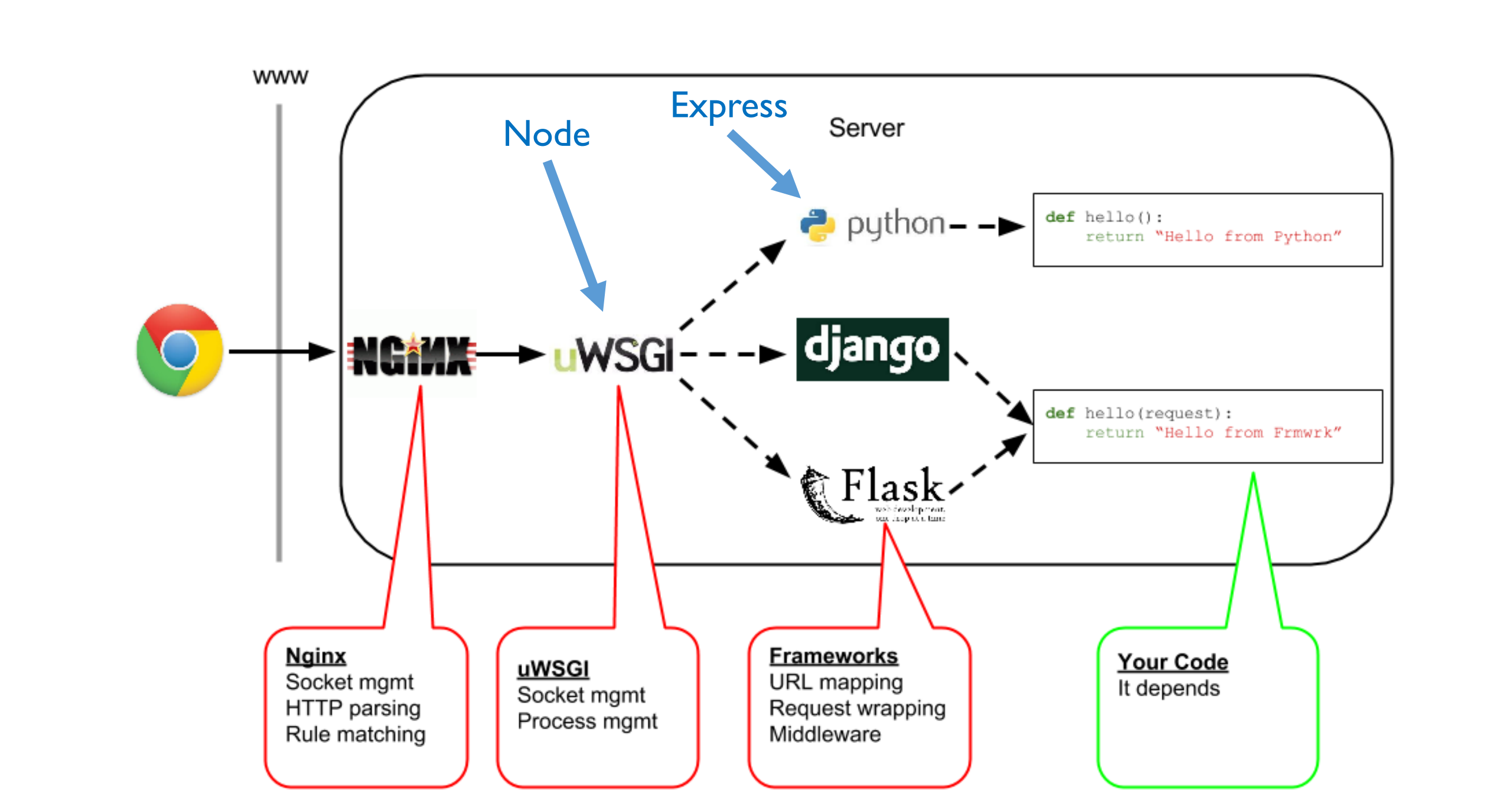
```
57    it(`should have as title 'app'`, async(() => {
58      const fixture = TestBed.createComponent(AppComponent);
59      const app = fixture.debugElement.componentInstance;
60      expect(app.title).toEqual('app');
61    }));
```

Unit Mock Testing

- Can test mocks using mocha and chai or with karma and jasmine
  - Jasmine syntax is similar to mocha and chai (requires a test runner)
- Make sure npm installs are done
  - npm install node-fetch
  - npm install https://www.clear.rice.edu/comp431/sample/mock-fetch.tgz
  - npm install mockery (fix mockery module: m = require('module') error)

Languages, Platforms, Frameworks

- Lots of choices
- Each have advantages and disadvantages
- Consider the long term solution
- Consider a short term solution

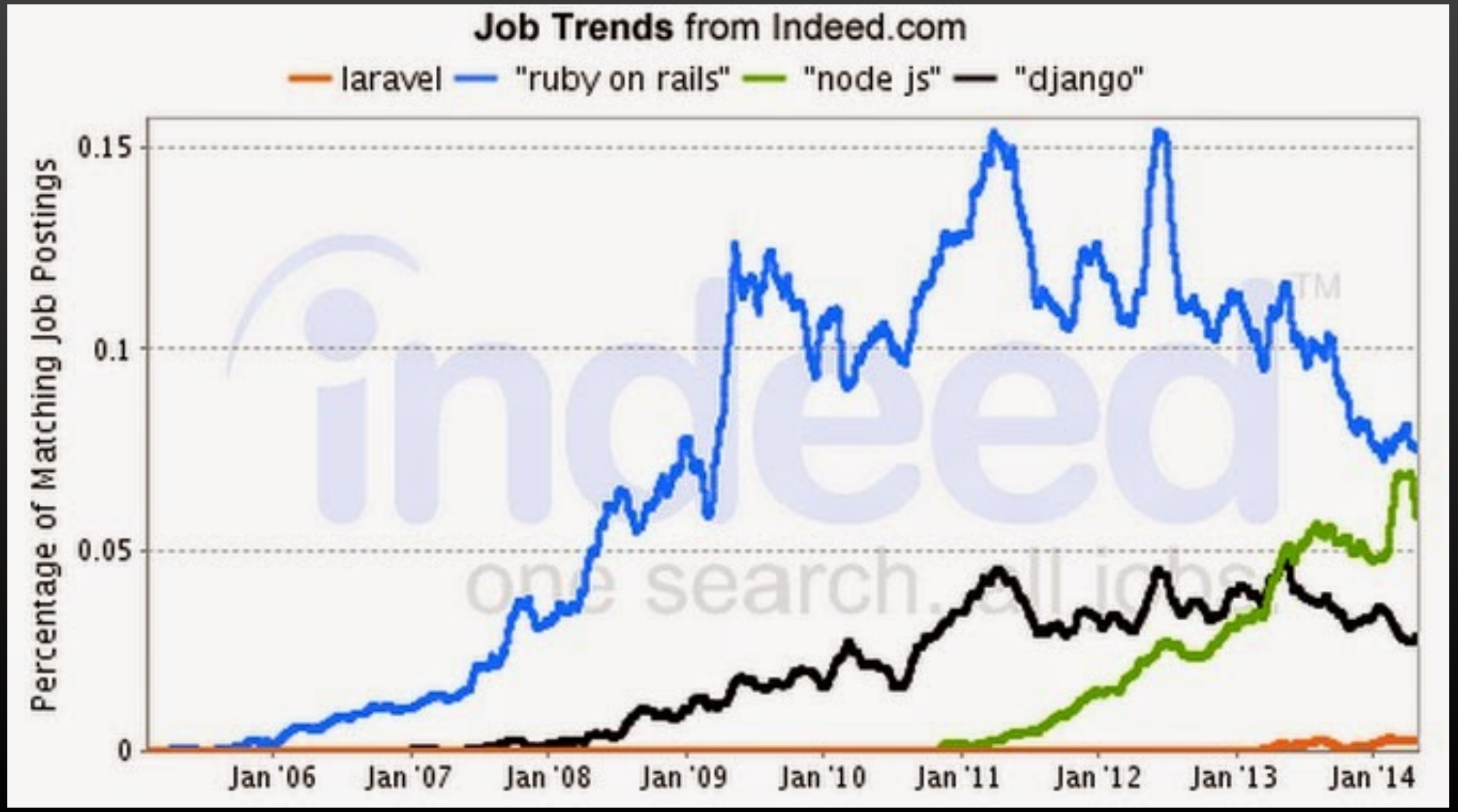


http://www.redsuncube.com/2015/08/host-your-django-website-in-ubuntu.html

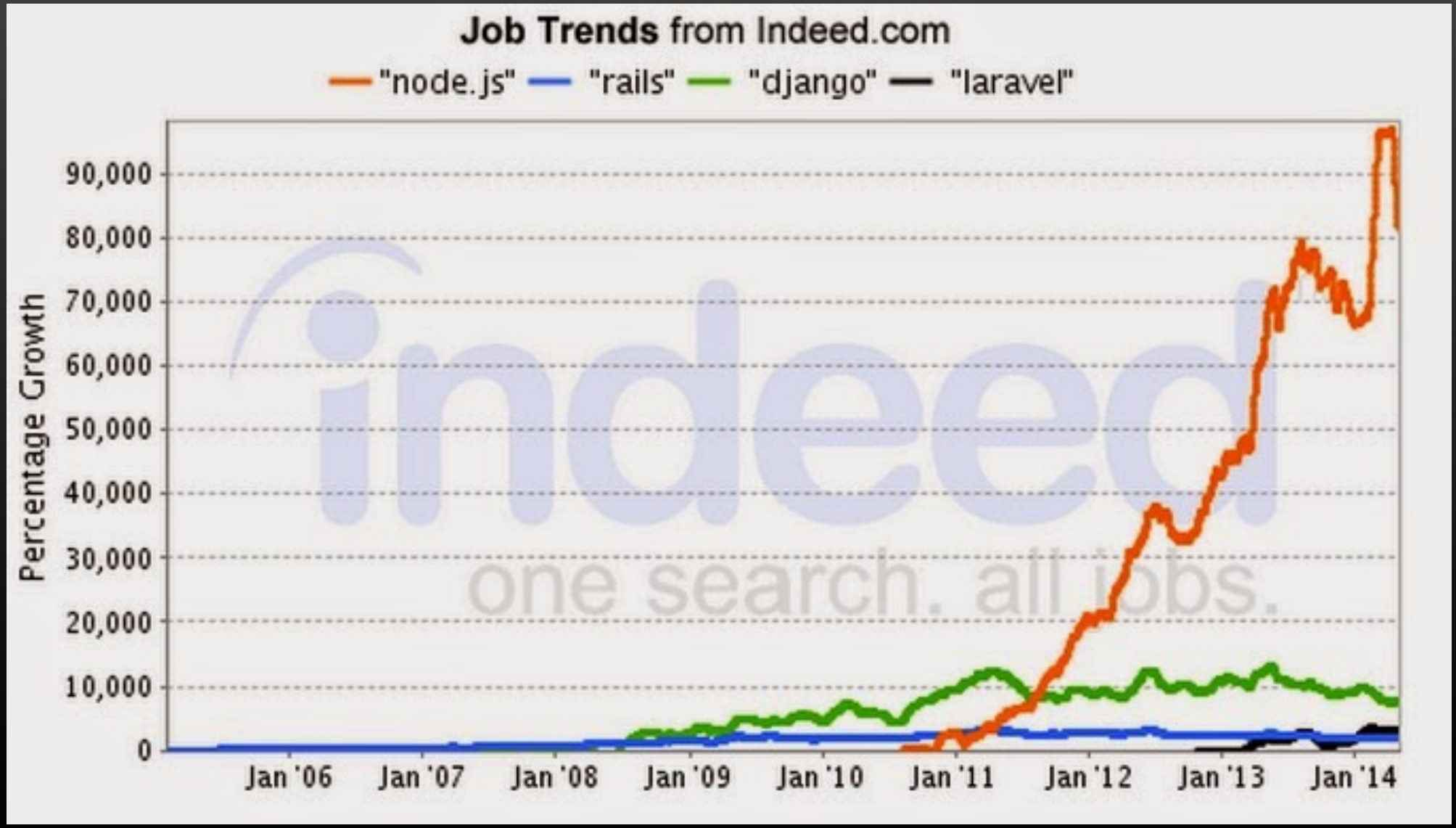
Middleware

- Middleware is *anything* you put in between the server/gateway and the final application/framework
- Middleware is compliant,
  - they accept a request and pass it along
  - they accept a response and pass it along
- As middleware, they can modify the request or response, e.g.,
  - check for authentication
  - add or strip headers
  - format or transform content

What's Hot


Job Trends from Indeed.com
laravel
"ruby on rails"
"node js"
"django"
Percentage of Matching Job Postings
0.15
0.1
0.05
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
indeed
one search. all jobs.

What's Hot


Job Trends from Indeed.com
"node.js"
"rails"
"django"
"laravel"
Percentage Growth
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
Jan '06
Jan '07
Jan '08
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
indeed
one search. all jobs.

What is a MEAN stack?
Mongo-Express-Angular-Node

- Express adds to Node a number of helpful libraries
- Minimalist philosophy
- Middleware is key

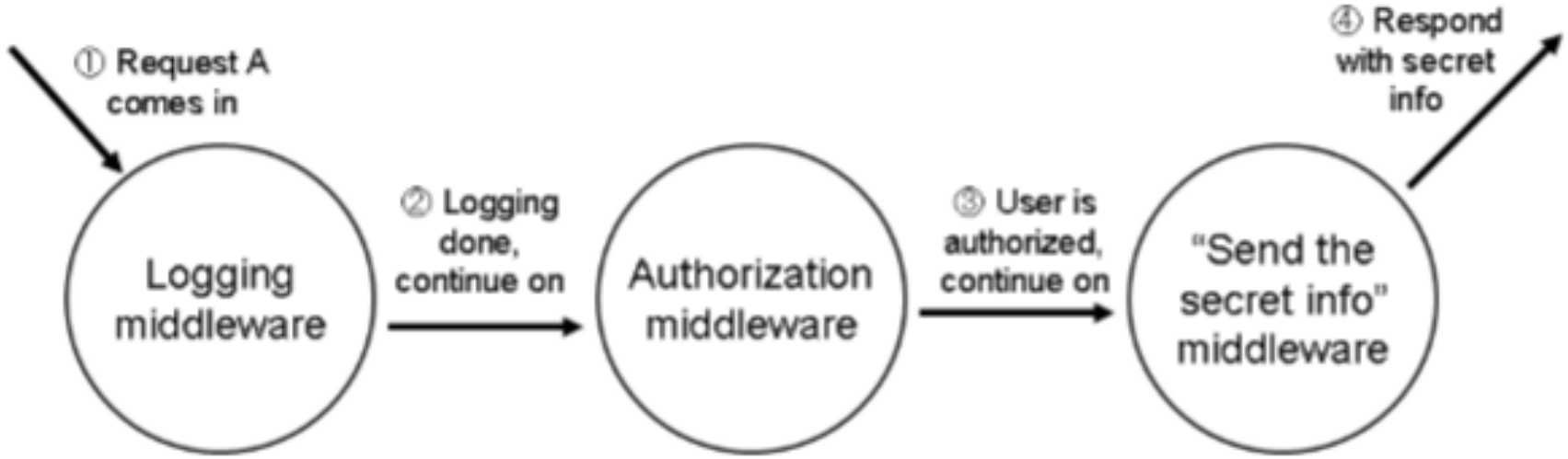
```
> mkdir backend; cd backend
> npm init -y
> npm install express --save
```

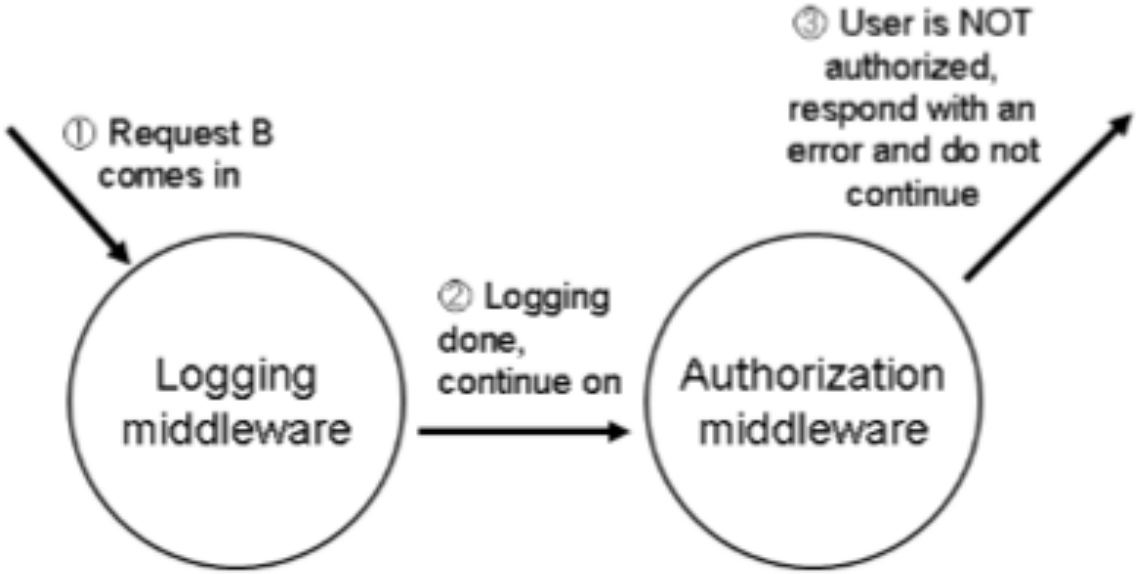
Express

Fast, unopinionated, minimalist web framework for Node.js

Middleware

```
app.put('/logout', isLoggedIn, logout)
function isLoggedIn(req, res, next)
```



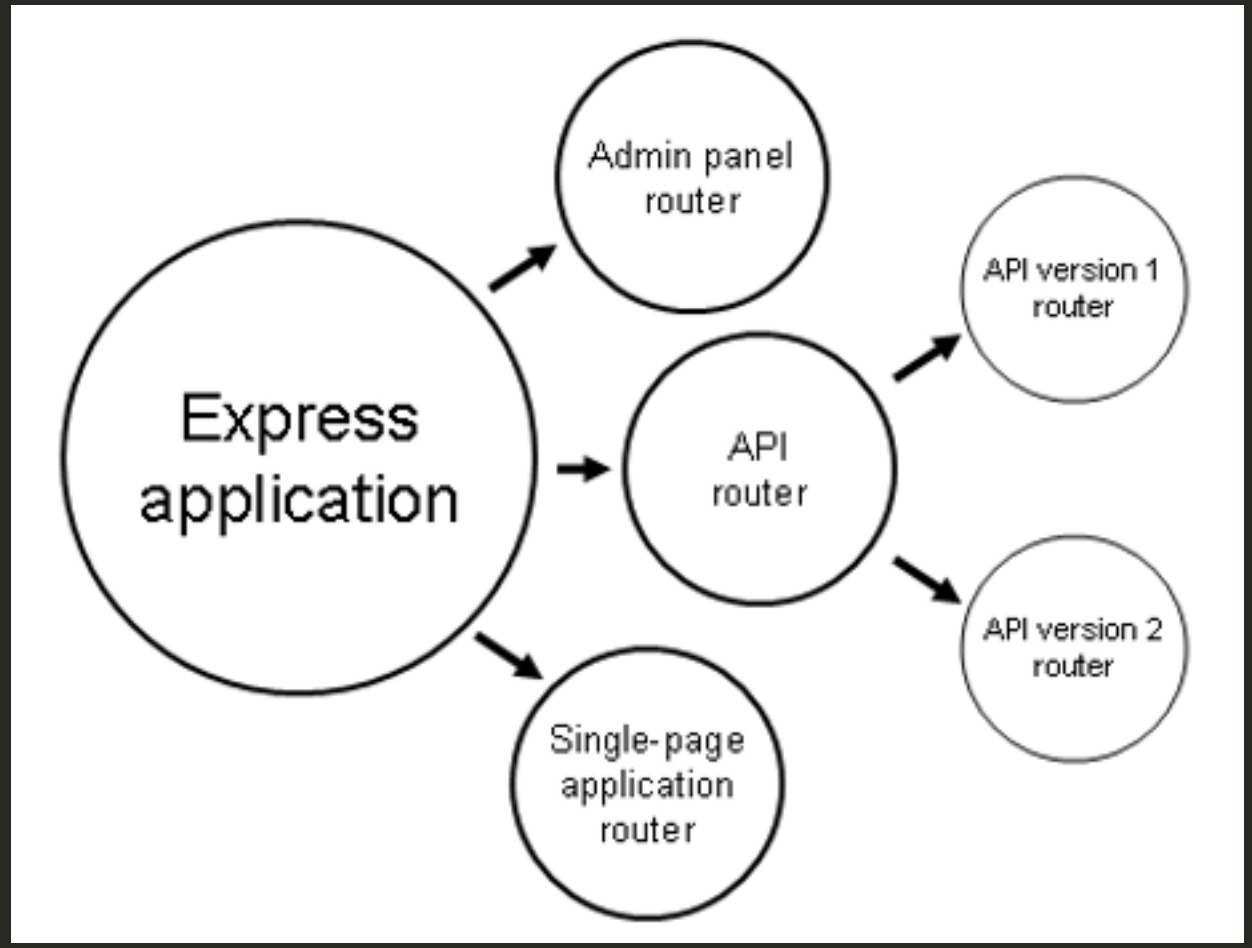


Redirection

"Send the secret info" middleware

Routing with Express

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

var app = express()

app.get('/', getIndex)
app.post('/', postIndex)

function getIndex(req, res) {
    res.send('hello world!')
}

function postIndex(req, res) {
    res.send('You POSTed to the homepage')
}

const port = process.env.PORT || 3000
const server = app.listen(port, () => {
    const addr = server.address()
    console.log(`Server listening at http://${addr.address}:${addr.port}`)
})
```



Install some Middleware

```
> npm install body-parser --save
```

```
2  const express = require('express')
3
4
5  const addArticle = (req, res) => {
6       console.log('Payload received', req.body)
7       res.send(req.body)
8  }
9
10 const hello = (req, res) => res.send({ hello: 'world' })
11
12 const app = express()
13
14 app.post('/article', addArticle)
15 app.get('/', hello)
16
17 const port = process.env.PORT || 3000
```

```
Server listening at http://:::3000
Payload received undefined
```

```
> curl -H 'Content-Type: application/json' \
      -d '{"Hello": "World" }' \
      http://localhost:8080/post
```

Accepting JSON Payloads

```
> npm install body-parser --save
```

```
2  const express = require('express')
3  const bodyParser = require('body-parser')
4
5  const addArticle = (req, res) => {
6      console.log('Payload received', req.body)
7      res.send(req.body)
8  }
9
10 const hello = (req, res) => res.send({ hello: 'world' })
11
12 const app = express()
13 app.use(bodyParser.json())
14 app.post('/article', addArticle)
15 app.get('/', hello)
16
17 const port = process.env.PORT || 3000
```

```
Server listening at http://:::3000
Payload received { text: 'This is my
```

```
> curl -H 'Content-Type: application/json' \
      -d '{"Hello": "World" }' \
      http://localhost:8080/post
{"Hello":"World"}
```