# Web Application Development

Node.js - Basics

# Assignment3

**Wednesday 2/21**

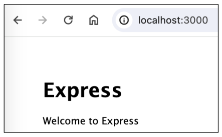
## Back-end Frameworks

- JavaScript: Node.js, Express
- C#: ASP.NET
- Java: Play, Spark
- Python: Django, Flask
- Ruby: Ruby on Rails, Sinatra
- PHP: Laravel, Cake
- etc

## Node.js

- Installations
  – www.seikyung.com/README.html
- How does it print "Welcome to Express"?

**app.js**
**routes/index.js**
**public/index.html**
**public/stylesheets/style.css**



**Express**

Welcome to Express

`package.json`
Configures project name, scripts, and dependencies.
`app.js`
The entry point into this project.
`bin/www`
A wrapper script for app.js
`public`
Where to place client-side code.
routes
Where to place server-side code.
`views`
no templates for now

```
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
├── views (no views for now)
```

- `Node` was developed as a JavaScript runtime environment built on Chrome's V8 JavaScript engine. `Node` made it possible to start using JavaScript on the server side to build applications.
- `npm` (Node's default package management system) gives you access to thousands of reusable Node packages built by developers all over the world.
  – Alternatively, you can use **`yarn`** developed by Facebook.
- `Node` has an event-driven architecture capable of asynchronous, non-blocking I/O, which eliminates the waiting approach to serving requests.
  – This allows you to build scalable and lightweight real-time web applications that can efficiently handle many requests.

- Express
  - Express is a simple server-side web framework for building web applications with Node. It complements Node with a layer of rudimentary web application features that provide HTTP utility methods and middleware functionality
- package.json
  - The package.json file will contain meta-information about the application, as well as list the module dependencies.

7

7

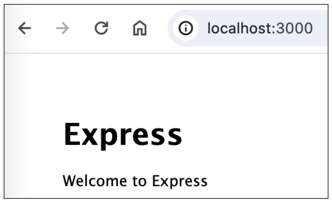
## When to use Node.js?

- Node's goal is to provide an easy way to build scalable network programs.
- Node.js is good for creating streaming based real-time services, web chat applications, browser games, etc.

8

8

- Did you follow Node.js installation instruction from **https://www.seikyung.com/README.html** ?
- Did you successfully run server and saw "Welcome to Express" message?



localhost:3000

**Express**

Welcome to Express

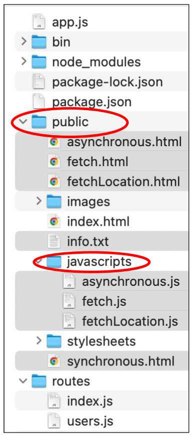
9

9

- Download Node0-Basics.zip from BB
- Did you put all files appropriate folder?

**public/asynchronous.html**
**public/fetch.html**
**public/fetchLocation.html**
**public/info.txt**
**public/synchronous.html**

**public/javascripts/asynchronous.js**
**public/javascripts/fetch.js**
**public/javascripts/fetchLocation.js**



```
app.js
> bin
> node_modules
package-lock.json
package.json
v public
    asynchronous.html
    fetch.html
    fetchLocation.html
  > images
    index.html
    info.txt
  v javascripts
      asynchronous.js
      fetch.js
      fetchLocation.js
  > stylesheets
    synchronous.html
v routes
    index.js
    users.js
```

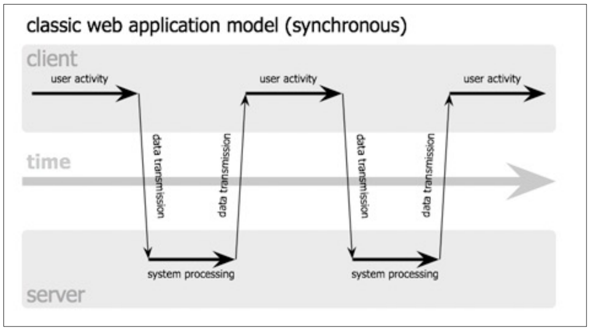
10

## AJAX

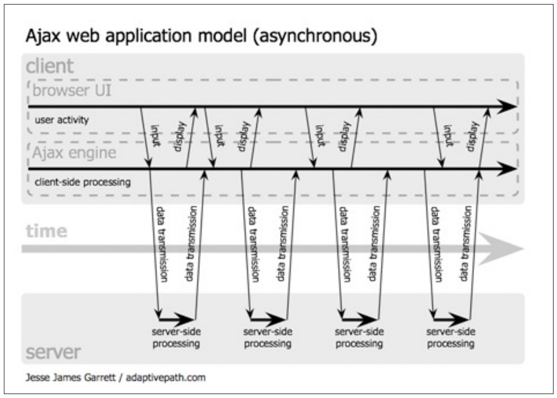
- Asynchronous JavaScript and XML (AJAX) is a term used to describe a paradigm that allows a web browser to send messages back to the server without interrupting the flow of what's being shown in the browser.
- With AJAX, you can
  - update a web page without reloading the page
  - request data from a server – after the page has loaded
  - receive data from a server – after the page has loaded
  - send data to a server – in the background
- Example
  - Google suggest, maps etc

**https://www.w3schools.com/js/js_ajax_intro.asp**

11



12



Jesse James Garrett / adaptivepath.com

13

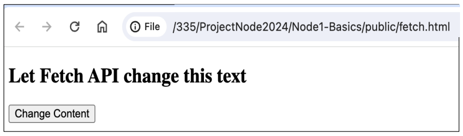
## fetch()

public/fetch.html
public/info.txt
public/javascripts/fetch.js

- fetch()
  - Provides a JavaScript interface for accessing and manipulating HTTP requests and responses. fetch() method provides an easy way to fetch resources asynchronously across the network.
  - It is XMLHttpRequest alternative
- POST requests:
  - Can post an unlimited amount of data
  - Do not generate viewable URLs for each action
  - Possible to transmit files
- GET requests:
  - Typically, not used for forms (limitation on data size)

**https://www.w3schools.com/tags/ref_httpmethods.asp**

14

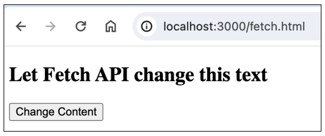


File /335/ProjectNode2024/Node1-Basics/public/fetch.html

**Let Fetch API change this text**

Change Content

✗

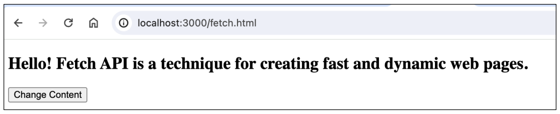


localhost:3000/fetch.html

**Let Fetch API change this text**

Change Content

✓



localhost:3000/fetch.html

**Hello! Fetch API is a technique for creating fast and dynamic web pages.**

Change Content

15

## JSON

public/fetchLocation.html
public/javascripts/fetchLocation.js

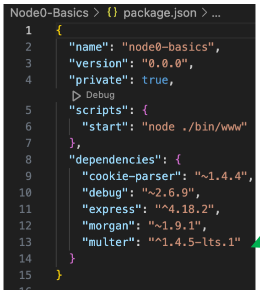
- **J**ava**S**cript **O**bject **N**otation
  - When receiving and sending data from/to server, **it must be string**.
  - JSON is just for exchanging data between a browser and a server
  - Convert JavaScript object into JSON and send JSON to the server
  - JSON syntax is derived (similar) from JavaScript Object syntax.
- JSON.stringify()
  - Convert a JavaScript object into a string with **JSON.stringify()**
- JSON.parse()
  - Parse the data with **JSON.parse()**, and the data becomes a JavaScript Object

**https://www.w3schools.com/js/js_json_intro.asp**

16

```
Node0-Basics % npm install multer --save
```

Installing **“multer”** package for file upload
**-- save** will update **package.json**



Node0-Basics > {} package.json > ...

```
{
  "name": "node0-basics",
  "version": "0.0.0",
  "private": true,
  ▷ Debug
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "cookie-parser": "~1.4.4",
    "debug": "~2.6.9",
    "express": "^4.18.2",
    "morgan": "~1.9.1",
    "multer": "^1.4.5-lts.1"
  }
}
```

17

17

**routes/index.js**

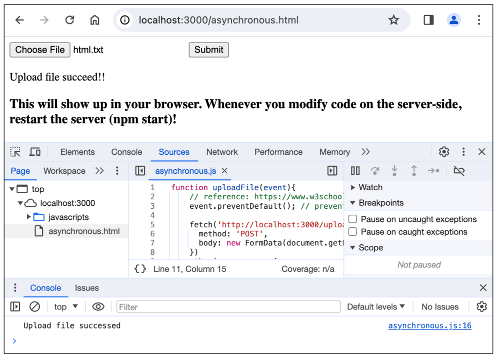
```
// Do NOT replace, add this code right before module.exports = router;

// require means include package
var multer  = require('multer');

// will create a folder name "uploadDir"
var upload = multer({ dest: 'uploadDir' });

// HTTP POST request from http://localhost:3000/upload is received
// "file_up" is the input field name from the HTML form
// selected file will be uploaded into the destination folder
// upload object is from multer package

router.post('/upload', upload.single('file_up'), function(req, res) {

  var message =  "This will show up in your browser. "

  res.send(message);

});
```

public/synchronous.html
public/asynchronous.html
public/javascripts/asynchronous.js

18

18

- Did you install multer package for file upload?
- Did you modify routes/index.js?



localhost:3000/asynchronous.html

Choose File html.txt Submit

Upload file succeed!!

**This will show up in your browser. Whenever you modify code on the server-side, restart the server (npm start)!**

```
function uploadFile(event){
    // reference: https://www.w3schoo
    event.preventDefault(); // preven

    fetch('http://localhost:3000/uploa
      method: 'POST',
      body: new FormData(document.getl
    })
```

```
Upload file successed                                   asynchronous.js:16
>
```

19

- Did you test all example files?
- Do you understand all codes?
- Next Class – React

20