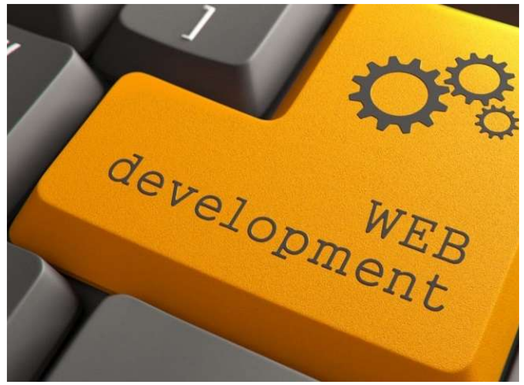
24.07.24 15:20

CSCI 165
Introduction to the Internet and the World Wide Web
Lecture 8: Backend Programming II



Jetic Gū

Overview

- Focus: Web Development
- Architecture: Internet
- Core Ideas:
    1. Protocols
    2. NodeJS cont’

Protocols

HTTP

Summary

P1
Protocols

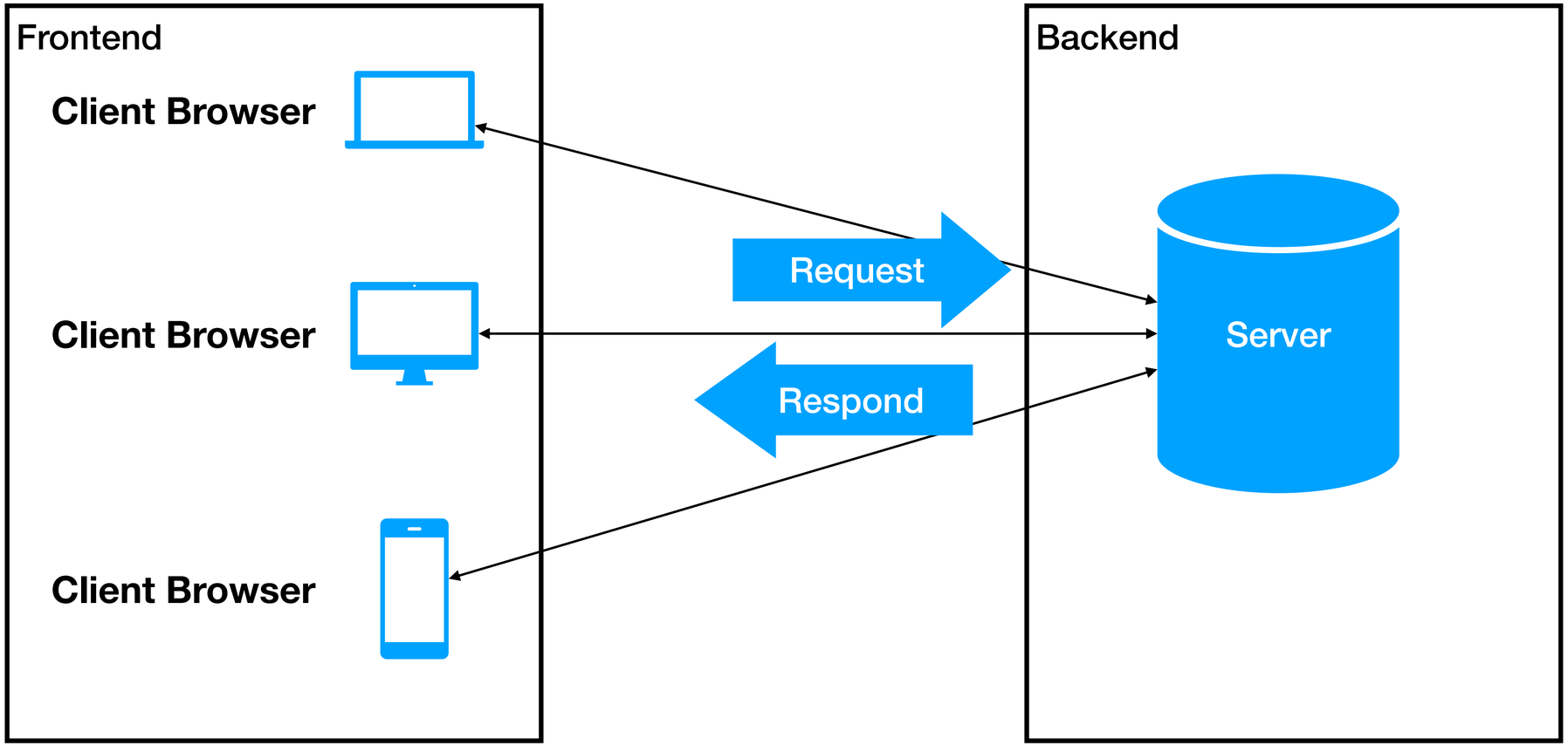
What we've done so far

- Frontend
  - HTML + CSS + Javascript
  - Executed on the browser
- Backend
  - NodeJS

Review

P1
Protocols

What we've done so far



Review

How do computers talk to each other?

- Through **protocols**
  - Established rules determine how data is transmitted between different devices in the same network.
  - Protocols are like **languages**, for two computers to talk to each other, they **must speak the same language**
  - Computer Networks have a lot of Protocols

Concept

How do computers talk to each other?

- Low-Level protocol: TCP/IP
- High-Level protocols:
  - HTTP: Hypertext (HTML) transfer protocol, default port 80
  - HTTPS: secure HTTP protocol, default port 443
  - FTP: file transfer protocol 21

Concept

So...

- I have my HTML file on my desktop, I double clicked to open it
  - This is using `file` protocol, which accesses files on your hard drive
    Other computers on the network cannot do that.
  - NodeJS: creates an HTTP/HTTPS server, provide access to your HTML for all computers in the network

NodeJS

Continued

Summary

NodeJS Code

- Serving a static HTML page

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS Code

- create variable `http` as a new instance of NodeJS `http` type
  - This is your webserver's protocol module

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS Code

- create variable `server` as a new instance of NodeJS `http server` type
  - This is your webserver instance

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS Code

- Depending on the request, you can write different things
- Content-Type can be `'text/plain'`, `'text/html'`, `'text/css'`, etc.

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS Code

- Transmitting the body for the same type

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS Code

- Run the server on specified port

```
var http = require("http");
var port = 8080;


var server = http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end('<h1>Hello World!</h1>');
})

server.listen(port, function() {
    console.log('Server unning at http://localhost:8080');
})
```

Demo

NodeJS File Serving

What if you have index.html to serve?

Summary

P3
NodeJS

Serve a File

- Assume you already have an html file ready to be served
- You will need to read from the file, and save its content as a variable
- Then, when an HTTP request comes in, serve the variable’s content

Concept

Read From a File

- Assume the file to be served is `index.html`
- You will need to use `fs` library (File System)

```
var fs = require('fs');
```

- Use the `readFile` method of `fs`

```
fs.readFile('index.html', function(error, content) {
    response.writeHead(200, { 'Content-Type': 'text/html' });
    response.end(content, 'utf-8');
});
```

Concept

P3
NodeJS

Read From a File

- Use the readFile method of fs

```
fs.readFile('index.html', function(error, content) {
    response.writeHead(200, { 'Content-Type': 'text/html' });
    response.end(content, 'utf-8');
});
```

- Within `fs.readFile` method, two arguments are required
    - `filename`: `'index.html'`, string type, location of the file (relative)
    - `function (error, content)`: function, `error` contain error codes, `content` contain the file's content in plain text after being read.

Concept