NodeJS

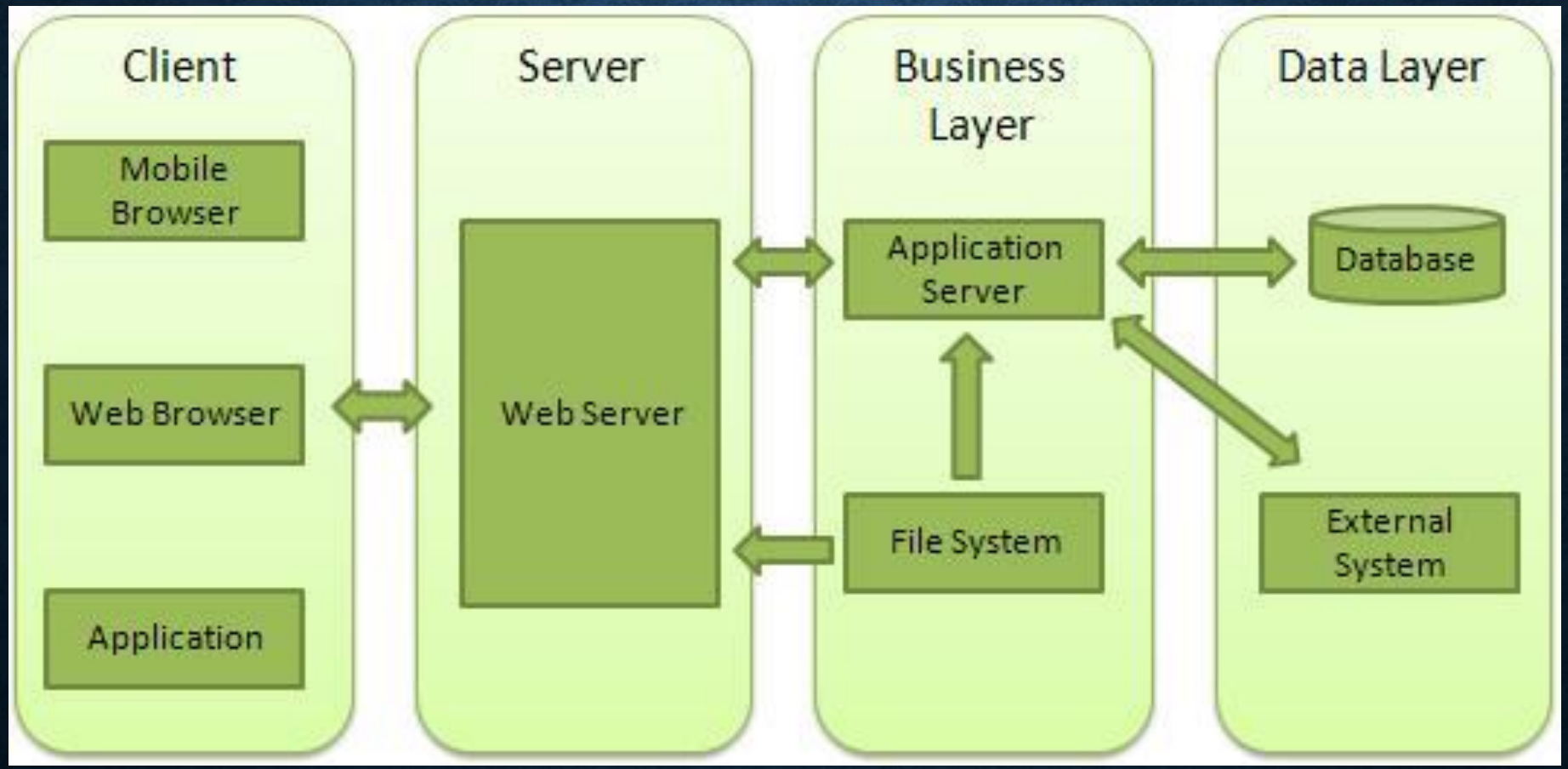
Webserver and Frameworks

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

NODEJS – WEB APP ARCHITECTURE

- Client Layer
- Server Layer
- Business Layer
- Data Layer

NODEJS – WEB APP ARCHITECTURE



NODEJS – CREATE A SERVER

- Require the “http” module.
- Use the method “createServer” and a port to listen on.

```
var http = require('http’);
var fs = require('fs’);
var url = require('url’);
// Create a server
http.createServer( function (request, response) {
// webserver logic
}).listen(8081);
```

NODEJS – CREATE A SERVER

- Now add some processing and logic
- Every webpage/request needs an HTTP status code, a server needs to respond with this code so that the client/recipient knows the status of the request.

- writeHead function takes the first parameter as the status code and a second parameter as a JSON array for the MIME types.

- Example:
  - response.writeHead(404, {'Content-Type': 'text/html'});

NODEJS – CREATE A SERVER

- To write any data the “write” function is used from the response object.
- Example:
  - response.write(“Error: Could not find the file.”);

- To send the response you use the “end” function.
- Example:
  - response.end();

NODEJS – CREATE A SERVER

- Putting it all together

```
http.createServer(function (request, response) {
    // Send the HTTP header
    // HTTP Status: 200 : OK
    // Content Type: text/plain
    response.writeHead(200, {'Content-Type': 'text/plain'});

    // Send the response body as "Hello World"
    response.end('Hello World\n');
}).listen(8081);
```

NODEJS – CREATE A CLIENT

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

// Options to be used by request
var options = {
   host: 'localhost',
   port: '8081',
   path: '/index.htm'
};

// Callback function is used to deal with response
var callback = function(response){
   // Continuously update stream with data
   var body = '';
   response.on('data', function(data) {
      body += data;
   });

   response.on('end', function() {
      // Data received completely.
      console.log(body);
   });
}
// Make a request to the server
var req = http.request(options, callback);
req.end();
```

NODEJS – CREATE WEB SOCKET

```
const WebSocket = require('ws');

// create a WebSocket
const ws = new WebSocket('wss://echo.websocket.org/', {
  origin: 'https://websocket.org' // For some CORS
});

// once a connection has been established
ws.on('open', function open() {
  console.log('connected');
  ws.send(Date.now());
});

// Always good to close a connection if its no longer needed
ws.on('close', function close() {
  console.log('disconnected');
});

// Triggered each time a change/message comes through the socket
ws.on('message', function incoming(data) {
  console.log(`Roundtrip time: ${Date.now() - data} ms`);
  ws.send(Date.now());

});
```

NODEJS – EXPRESS FRAMEWORK

- Middleware for HTTP requests
- Routing table
- Dynamic rendering and templates

NODEJS – EXPRESS FRAMEWORK

- In order for Express to work you need to install it alongside some other modules.
  - npm install express --save
  - npm install body-parser --save
  - npm install cookie-parser --save
  - npm install multer --save

body-parser – This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
cookie-parser – Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
multer – This is a node.js middleware for handling multipart/form-data.

NODEJS – EXPRESS FRAMEWORK

- Express is a module, so in order to use it you need to require it and initialize and create an object:
  - var express = require('express');
  - var app = express();
- You will use the variable app for any express related functions/logic

NODEJS – EXPRESS FRAMEWORK

- Creating an Express Server:

```
var server = app.listen(8081, function () {
    var host = server.address().address
    var port = server.address().port
    console.log("Example app listening at http://%s:%s", host, port)
});
```

NODEJS – EXPRESS FRAMEWORK

- Example of a HTTP Request and Response in Express:

```
app.get('/', function (req, res) {
    console.log("Got a GET request for the homepage");
    res.send('Hello GET');
})
```

NODEJS – EXPRESS FRAMEWORK

- Extracting information from a GET request:

```
app.get('/process_get', function (req, res) {
   // Prepare output in JSON format
   response = {
      first_name:req.query.first_name,
      last_name:req.query.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
});
```

NODEJS – EXPRESS FRAMEWORK

- Extracting information from a POST request:

```
var bodyParser = require('body-parser');
// Create application/x-www-form-urlencoded parser
var urlencodedParser = bodyParser.urlencoded({ extended: false })

app.post('/process_post', urlencodedParser, function (req, res) {
   // Prepare output in JSON format
   response = {
      first_name:req.body.first_name,
      last_name:req.body.last_name
   };
   console.log(response);
   res.end(JSON.stringify(response));
});
```

NODEJS – EXPRESS STATIC FILES

- What happens if you want to serve JS, CSS, Images in Express?
- Express comes with this feature where you specify the folder for the static content that you want served.
- The "**use**" function of express is used to load other supported modules/libraries into express.
  - Example of serving static files:
    - app.use(express.static('public')); // where "public" is a folder with content inside

NODEJS – EXPRESS STATIC FILES

- Example of serving a file:

```
app.get('/index.htm', function (req, res) {
   res.sendFile( __dirname + "/" + "index.htm" );
});
```

NODEJS – EXPRESS COOKIE MANAGEMENT

- For this you would need to require "cookie-parser"

```
var cookieParser = require('cookie-parser');
var app = express();
app.use(cookieParser());
```


node
SERVER


node
JS
express