Node.js and NPM

Jogesh K. Muppala

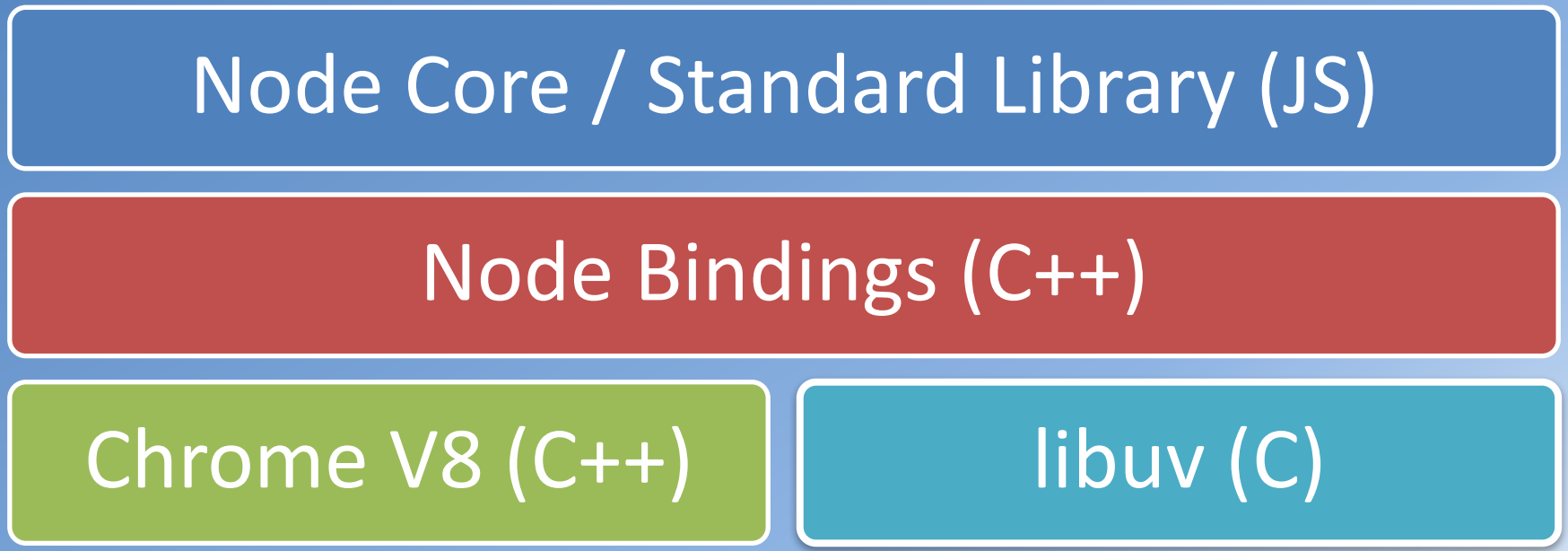
THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

What is Node.js?

- JavaScript runtime built on Chrome V8 JavaScript Engine
- Uses an event-driven, non-blocking I/O model
  - Makes it lightweight and efficient

Node Architecture



Node.js Use Cases

- Utilities written in JavaScript for web development:
  - Bower, Grunt, Gulp, Yeoman etc.
- Server-side Development
  - Web server, Business logic, Database access

Node Package Manager

- Node package manager (NPM): manages ecosystem of node modules / packages
- A package contains:
  - JS files
  - package.json (manifest)

Node Modules

Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

JavaScript Modules

- JavaScript does not define a standard library
- CommonJS API fills this gap by defining APIs for common application needs
  - It defines a module format
  - Node follows the CommonJS module specification

Node Modules

- Each file in Node is its own module
- The *module* variable gives access to the current module definition in a file
- The *module.exports* variable determines the export from the current module
- The *require* function is used to import a module

Node Modules

- File-based Modules
- Core Modules
  - Part of core Node
  - Examples: path, fs, os, util, . . .
- External Node modules
  - Third-party modules
  - Installed using NPM
  - node_modules folder in your Node application

Using Node Modules

- Include them using require function
- File-based modules:
  - require('./module_name')
  - Specify the relative path to the file
- Core and External modules:
  - require('module_name')
  - Looks for external modules in:
    - ./node_modules, ../node_modules, ../../node_modules, . . .
    - Up the folder hierarchy until the module is found

Node Modules: Callbacks and Error Handling

Jogesh K. Muppala

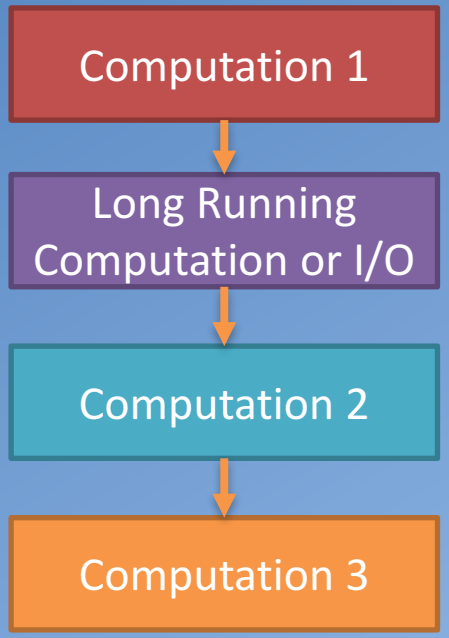
THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

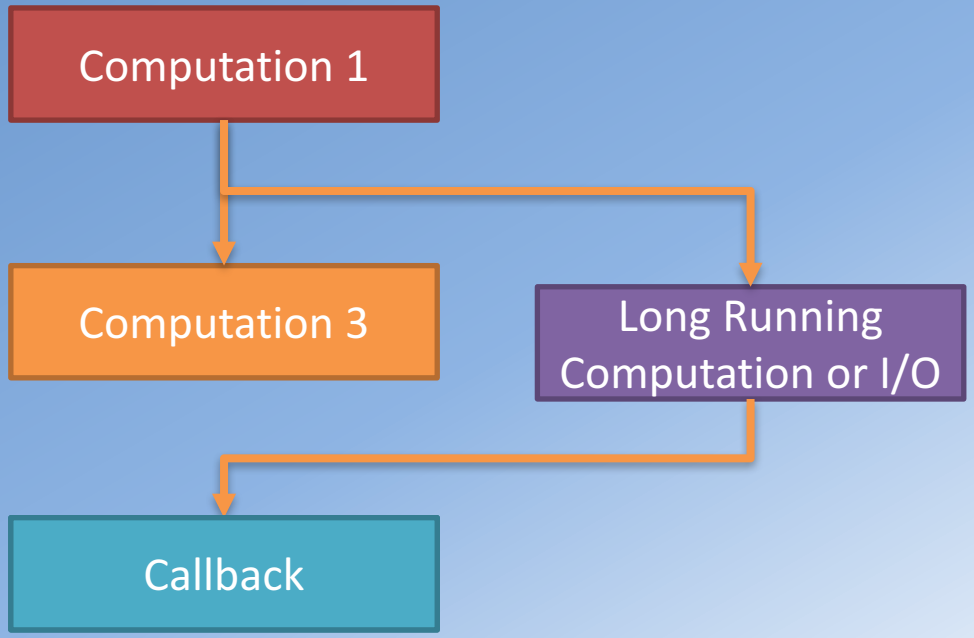
Two Salient Features of JavaScript

- First-class functions: A function can be treated the same way as any other variable
- Closures:
  - A function defined inside another function has access to all the variables declared in the outer function (outer scope)
  - The inner function will continue to have access to the variables from the outer scope even after the outer function has returned

Asynchronous Programming

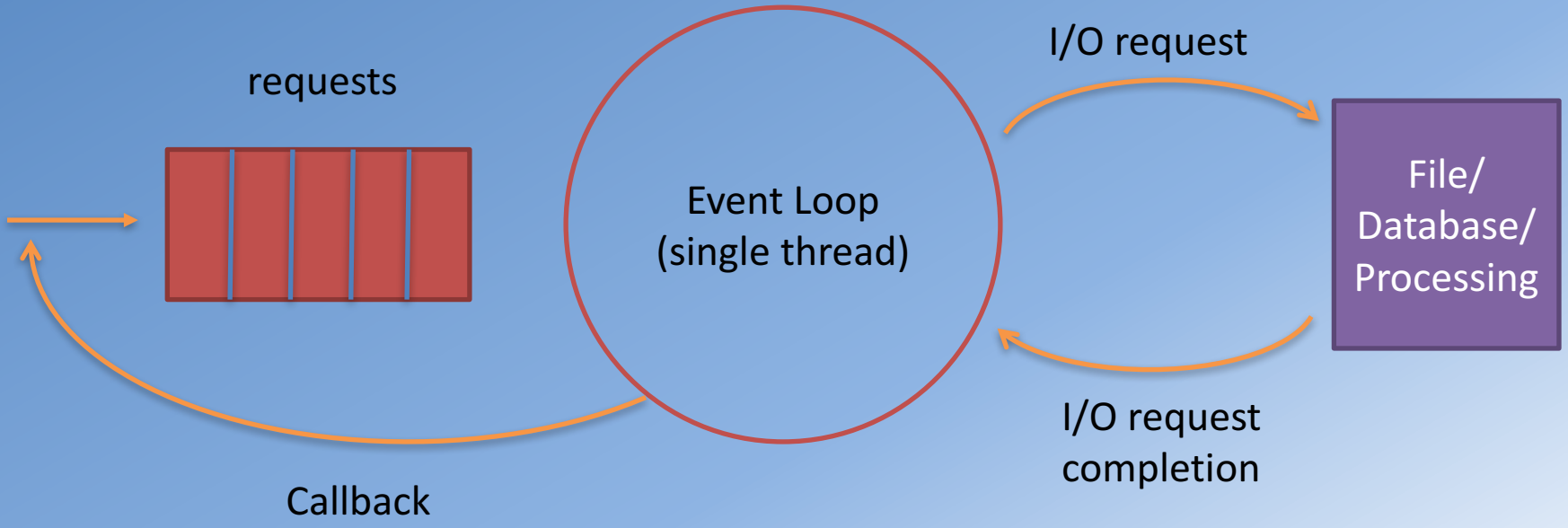


Synchronous Programming

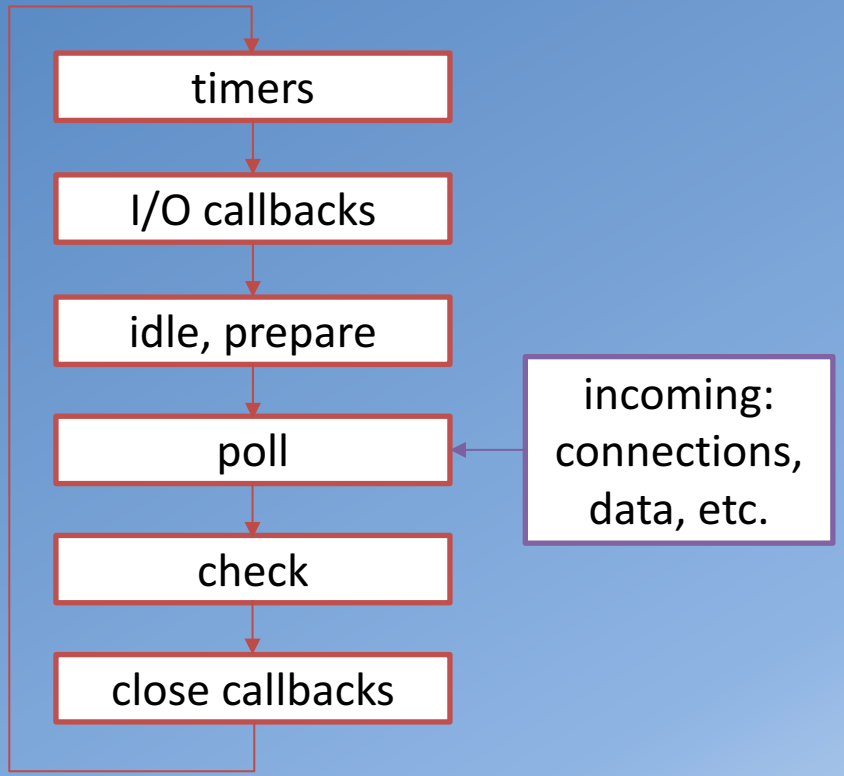


Asynchronous Programming

Node, Async I/O and Callbacks



Event Loop



- **timers**: this phase executes callbacks scheduled by setTimeout() and setInterval().
- **I/O callbacks**: executes almost all callbacks with the exception of close callbacks, the ones scheduled by timers, and setImmediate().
- **idle, prepare**: only used internally.
- **poll**: retrieve new I/O events; node will block here when appropriate.
- **check**: setImmediate() callbacks are invoked here.
- **close callbacks**: e.g. socket.on('close', ...).

Networking Essentials

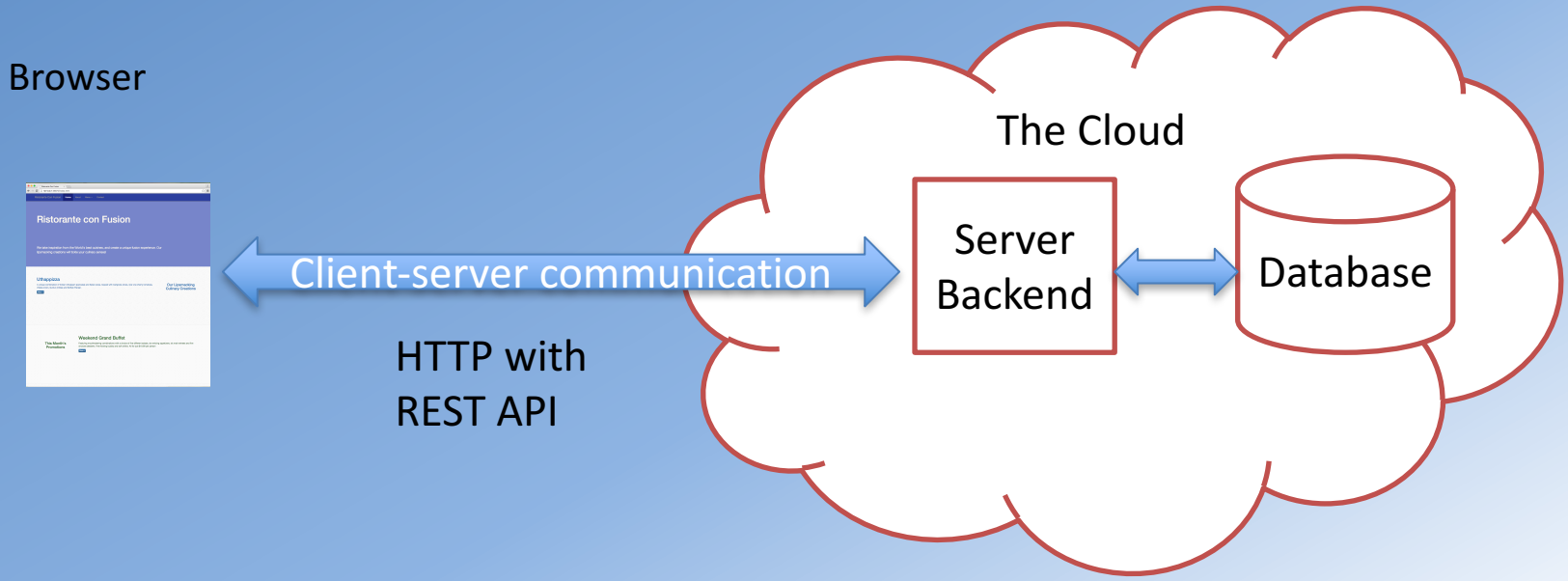
Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Client and Server

- Web applications are not stand-alone
- Many of them have a “Cloud” backend



The Networking Alphabet Soup

Client-Server Communication

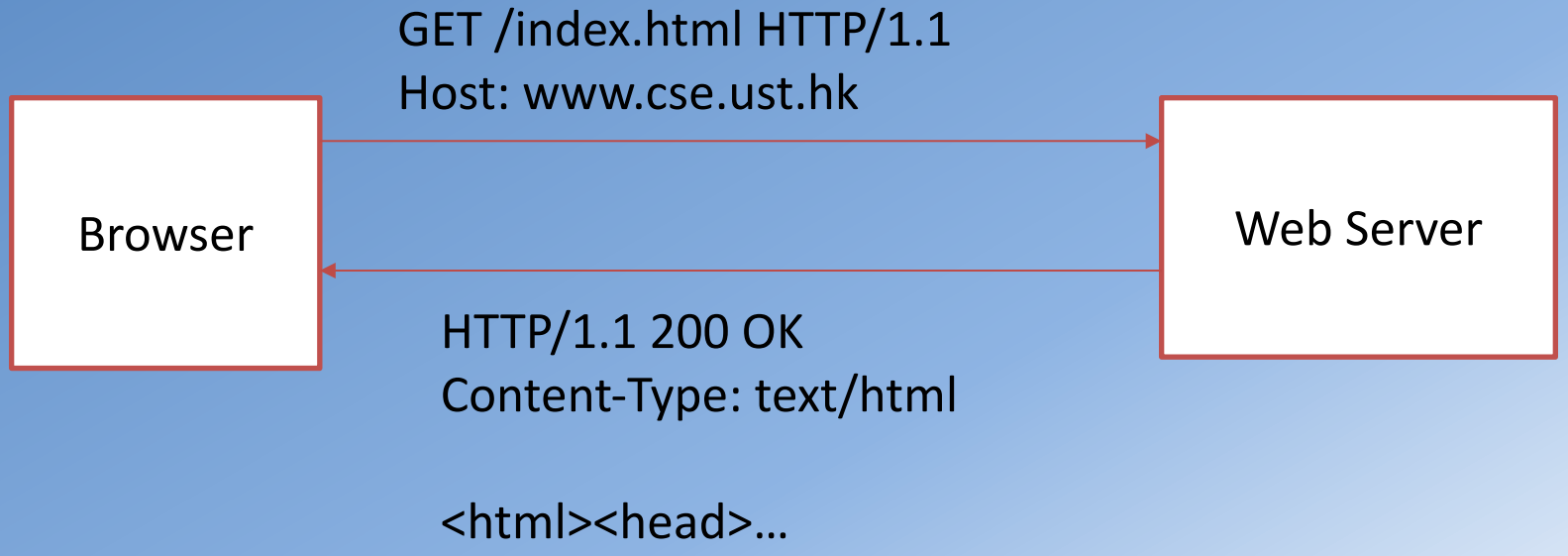
- Network operations cause unexpected delays
- You need to write applications recognizing the asynchronous nature of communication
  - Data is not instantaneously available

Hypertext Transfer Protocol (HTTP)

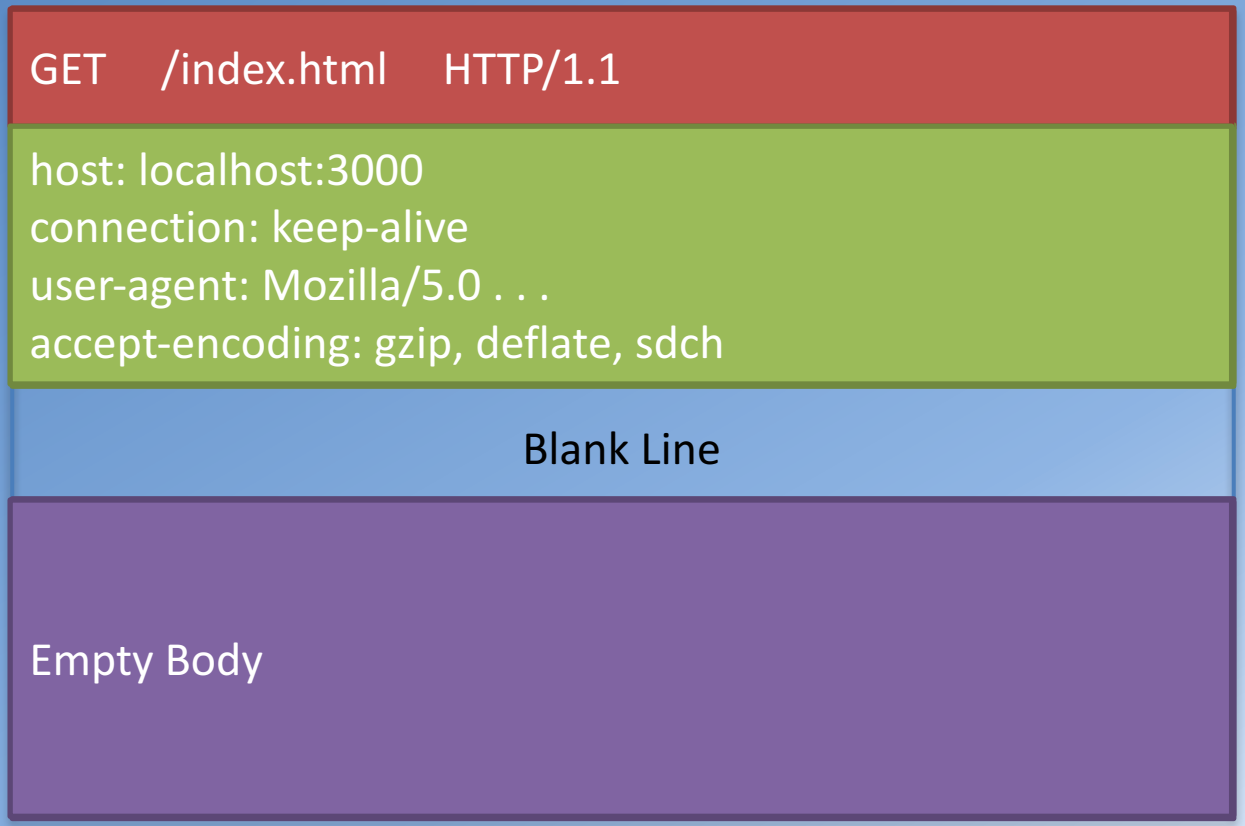
- A client-server communications protocol
- Allows retrieving inter-linked text documents (hypertext)
  - World Wide Web.

- HTTP Verbs
  - HEAD
  - GET
  - POST
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT

Hypertext Transfer Protocol (HTTP)



HTTP Request Message



HTTP Response Message

HTTP/1.1 200 OK

Connection: keep-alive
Content-Type: text/html
Date: Sun, 21 Feb 2016 06:01:43 GMT
Transfer-Encoding: chunked

Blank Line

<html><title>This is index.html</title><body><h1>Index.html</h1> This is the contents of this file </body></html>

HTTP Response Codes (Main ones)

Code	Meaning
200	OK
201	Created
301	Moved Permanently
304	Not Modified
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
422	Unprocessable Entry
500	Internal Server Error
505	HTTP Version Not Supported

HTTP Response

- Server may send back data in a specific format:
  - eXtensible Markup Language (XML)
  - Javascript Object Notation (JSON)

Javascript Object Notation (JSON)

- http://www.json.org
- Lightweight data interchange format
- Language independent *
- *Self-describing* and easy to understand

Javascript Object Notation (JSON)

- Data structured as:
  - A collection of name/value pairs
  - Ordered list of values

- Example

```
{ "promotions": [
    {
      "id": 0,
      "name": "Weekend Grand Buffet",
      "image": "images/buffet.png",
      "label": "New",
      "price": "19.99",
      "description": "Featuring mouthwatering combinations . . . "
    }
  ]
}
```

Node and the HTTP Module

Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Node HTTP Module

- Core networking module supporting a high-performance foundation for a HTTP stack
- Using the module:

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

- Creating a server:

  ```
  const server = http.createServer(function(req, res){ . . . });
  ```

- Starting the server:

  ```
  server.listen(port, . . . );
  ```

Node HTTP Module

- Incoming request message information available through the first parameter “req”
  - req.headers, req.body, . . .
- Response message is constructed on the second parameter “res”
  - res.setHeader("Content-Type", "text/html");
  - res.statusCode = 200;
  - res.write(’Hello World!');
  - res.end('<html><body><h1>Hello World</h1></body></html>');

Node path Module

- Using path Module:

```
const path = require('path');
```

- Some example path methods:

```
path.resolve('./public'+fileUrl);
path.extname(filePath);
```

Node fs Module

- Use fs module in your application

  const fs = require('fs');

- Some example fs methods:

  fs.exists(filePath, function(exists) { . . . });

  fs.createReadStream(filePath).pipe(res);

Introduction to Express

Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

What is Express

- Express: Fast, unopinionated, minimalist web framework for Node.js (from expressjs.com)
- Web application framework that provides a robust set of features
- Many third-party *middleware* to extend functionality
- Installing Express:
  - In your project folder do: npm install express --save

Express Middleware

- Middleware provide a lot of plug-in functionality that can be used within your Express application
- Example: *morgan* for logging

```
var morgan = require('morgan');
app.use(morgan('dev'));
```

- Serving static web resources:

```
app.use(express.static(__dirname + '/public'));
```

  - Note: __filename and __dirname give you the full path to the file and directory for the current module

A Brief Tour of a Node Module

- Examine package.json file
- Semantic Versioning
  - <Major Version>.<Minor Version>.<Patch>
  - npm install can specify the acceptable package version:
    - Exact: npm install express@4.0.0
    - Patch acceptable: npm install express@”~4.0.0”
    - Minor version acceptable: npm install express@”^4.0.0”

Brief
Representational State Transfer (REST)

Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Web Services

- A system designed to support interoperability of systems connected over a network
  - Service oriented architecture (SOA)
  - A standardized way of integrating web-based applications using open standards operating over the Internet
- Two common approaches used in practice:
  - SOAP (Simple Object Access Protocol) based services
    - Uses WSDL (Web Services Description Language)
    - XML based
  - REST (Representational State Transfer)
    - Use Web standards
    - Exchange of data using either XML or JSON
    - Simpler compared to SOAP, WSDL etc.

Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web.
- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification.
- A collection of network architecture principles which outline how resources are defined and addressed

Representational State Transfer (REST)

- Four basic design principles:
  - Use HTTP methods explicitly
  - Be stateless
  - Expose directory structure-like URIs
  - Transfer using XML, JavaScript Object Notation (JSON), or both

REST and HTTP

- The motivation for REST was to capture the characteristics of the Web that made the Web successful
  - URI (Uniform Resource Indicator) Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET
  - HTTP PUT, HTTP DELETE
  - Preserve Idempotence

REST Concepts

Nouns (Resources)
unconstrained
i.e., http://www.conFusion.food/dishes/123

Verbs
constrained
i.e., GET, PUT, POST, DELETE

Representations
constrained
i.e., XML, JSON

Resources

- The key abstraction of information in REST is a resource.
- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Hong Kong"), a collection of other resources, a non-virtual object (e.g. a person), and so on
- Represented with a global identifier (URI in HTTP)
  - http://www.conFusion.food/dishes/123

Naming Resources

- REST uses URI to identify resources
  - http://www.conFusion.food/dishes/
  - http://www.conFusion.food/dishes/123
  - http://www.conFusion.food/promotions/
  - http://www.conFusion.food/leadership/
  - http://www.conFusion.food/leadership/456
- As you traverse the path from more generic to more specific, you are navigating the data
- Directory structure to identify resources

Verbs

- Represent the actions to be performed on resources
  - Corresponding to the CRUD operations
- HTTP GET ←→ READ
- HTTP POST ←→ CREATE
- HTTP PUT ←→ UPDATE
- HTTP DELETE ←→ DELETE

HTTP GET

- Used by clients to request for information
- Issuing a GET request transfers the data from the server to the client in some representation (XML, JSON)
  - GET http://www.conFusion.food/dishes/
    - Retrieve all dishes
  - GET http://www.conFusion.food/dishes/452
    - Retrieve information about the specific dish

HTTP PUT, HTTP POST, HTTP DELETE

- HTTP POST creates a resource
  - POST http://www.conFusion.food/feedback/
    - Content: {first name, last name, email, comment etc.}
    - Creates a new feedback with given properties
- HTTP PUT updates a resource
  - PUT http://www.conFusion.food/dishes/123
    - Content: {name, image, description, comments …}
    - Updates the information about the dish, e.g., comments
- HTTP DELETE removes the resource identified by the URI
  - DELETE http://www.conFusion.food/dishes/456
    - Delete the specified dish

Representations

- How data is represented or returned to the client for presentation
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data
  - Client can request the data in a specific format if supported

Stateless Server

- Server side should not track the client state:
  - Every request is a new request from the client
- Client side should track its own state:
  - E.g., using cookies, client side database
  - Every request must include sufficient information for server to serve up the requested information
  - Client-side MVC setup

REST?

- The REST is not history!

Express Router

Jogesh K. Muppala

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Express Application Routes

- We examined REST in the previous lecture
- Identify an end point with a URI
- Apply the verb on the URI
- Express supports this through app.all, app.get, app.post, app.put, app.delete methods

Express Application Routes

- Application Routes:

```
app.all('/dishes', function(req,res,next) { . . . });
app.get('/dishes', function(req,res,next) { . . . });
app.post('/dishes', function(req,res,next) { . . . });
app.put('/dishes', function(req,res,next) { . . . });
app.delete('/dishes', function(req,res,next) { . . . });
```

Routes with Parameters

- Example:

```
app.get('/dishes/:dishId', (req,res,next) => {
        res.end('Will send details of the dish: ’
                + req.params.dishId +' to you!')
});
```

Body Parser

- Middleware to parse the body of the message
- Using Body Parser:

```
var bodyParser = require('body-parser');
app.use(bodyParser.json()); // parse the JSON in the body
```

- Parses the body of the message and populates the *req.body* property

Express Router

- Express Router creates a mini-Express application:

```
var dishRouter = express.Router();
dishRouter.use(bodyParser.json());

dishRouter.route('/')
    .all(. . .);
    .get(. . .);
    . . .
```