DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

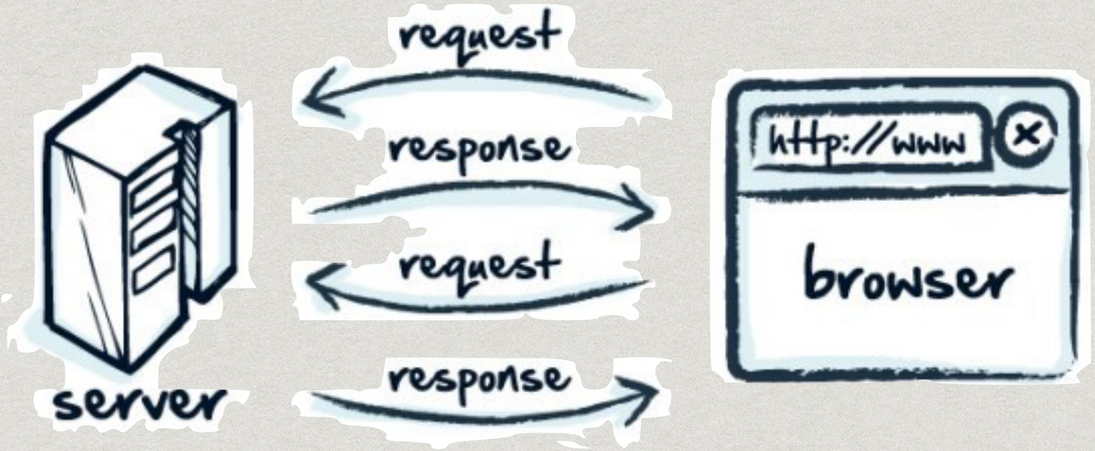
WEB DEVELOPMENT WITH NODE.JS AND EXPRESS

COMP7980 - WEB AND MOBILE PROGRAMMING

CHAPTER 3 - 2022 - HKBU

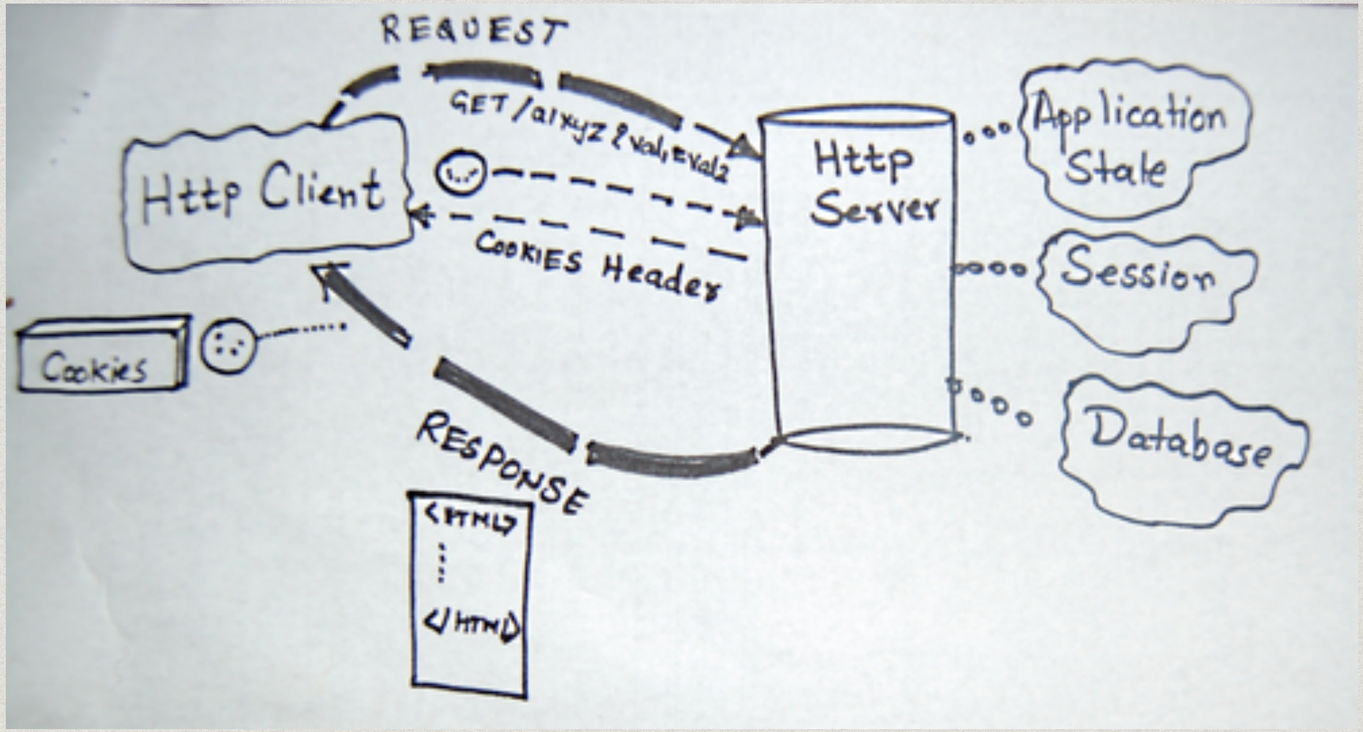
MARTIN & KENNY

Agenda



- **HTTP** Protocol
  - Client and Server model
- **Node.js** - JavaScript runtime environment
- Web Application Development with **Express**
- **No-SQL** Database

HyperText Transfer Protocol (HTTP)



- A **request-response protocol** that enables the transfer of hypertext documents and other files between a **client and server**.

HTTP Protocol

- HTTP defines **a set of request methods** to indicate the desired action to be performed for a given resource.
  - GET— requests a representation of the specified resource. Requests using GET should only **retrieve data**.
  - POST — is used to submit an entity to the specified resource, often **causing a change in state or side effects on the server**.
  - PUT, DELETE, etc… to be covered later.

URL

Example of URL: http://localhost:5500/lab02/index.html

- URL comprises of
  - Protocol (the browser's default is usually https://)
  - **IP address or domain**
  - **Port number**
  - Subdirectory path from the 'document root'
  - Name of the resources

IP Addresses

- Every machine on the Internet has to have **a unique IP address** so that communications can be routed to the correct computer.
- IPv4: **32-bit binary number**
  - e.g. 123.45.67.89
  - Each number ranges from 0 to 255.

Domain Names & Local Machine

- **Domain name**
  - Domain names are **mapped to IP addresses**.
  - Easier to remember **hkbu.edu.hk -> 158.182.0.81**
- To reference the **local machine**, we can use
  - **127.0.0.1** (IP address) or **localhost** (domain name)

Port

Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
22	TCP	SSH
25	TCP	SMTP
53	UDP, TCP	DNS
80	TCP	HTTP (WWW)
110	TCP	POP3
443	TCP	SSL

- Different **Internet services** will use different **ports**
  - Web server usually uses **port 80**.
  - **Ports 8000 and 8080** are common ports for software providing **http services** that is not a core HTTP server.

Response Status

- **HTTP response status codes** indicate whether a specific HTTP request has been successfully completed. **Responses are grouped in five classes:**
  - Informational responses (100–199),
  - **Successful responses (200–299)**,
  - Redirects (300–399),
  - Client errors (400–499),
  - and Server errors (500–599).

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

HTML Form Submission

```html
<form action="https://www.httpbin.org/post" method="POST">
    <input name="email" type="email">

    <input name="numTickets" type="number" min=1 max=4>

    <button type="submit">Submit</button>
</form>
```

- Form is used to pass data to a server.
- The **submit button** will trigger the submission.
- action specifies **where** the form data will be submitted to.
- method specifies the **HTTP request method** for sending form data.

Name Attribute

```
<input name="numTickets" type="number">
```

- The name attribute specifies the name of an <input> element.
- The name attribute could be used to reference elements in client-side JavaScript, or to **reference form data in form submission**.
  - Note: Only form elements with a name attribute will be included in form submission.

Checkbox

☐ Option 1

```
<input name="box" type="checkbox" value="dummy">
```

- When this checkbox is clicked, this form element name, **together with its value**, is submitted to the server.

```
box = "dummy"
```

- If the box is NOT clicked, this form element **won't be submitted**.

☑

```
<input name="box" type="checkbox" value="dummy" checked>
```

- To check the box, put the checked attribute in the opening tag.

Form Submission - GET or POST?

- When a form is submitted via GET, the **form-data** are **encoded in the URL** in what is called a **query string**.

http://server/path/endpoint?input1=value1&input2=value2&...

- Query string can be **bookmarked**.
- **Not suitable for sending password** as it can be seen on the URL.
- Data in a query string is **limited**.

Form Submission - GET or POST?

- When a form is submitted via POST, the **form-data** are submitted via the **request body**.
  - Can handle arbitrarily large amount of data.
- Rule of Thumb:
  - Form contains **password**: Use POST
  - Form **modifies the server-side database** : Use POST
  - Form would not modify the server-side database: Use GET

NODE.JS & MORE JAVASCRIPT

Node.js

- Node.js is built on top of **Google's V8 engine**, which in turns compiles **JavaScript** and allows it to be executed on **server-side.**

In the old days, the partnership was simple. JavaScript handled little details on the browser, while PHP managed all the server-side tasks that existed between port 80 and MySQL. It was a happy union that continues to support many of the crucial parts of the Internet. Between WordPress, Drupal, and Facebook, people can hardly go a minute on the Web without running into PHP.

But then some clever kid discovered he could get JavaScript running on the server. Suddenly, there was no need to use PHP to build the next generation of server stacks. One language was all it took to build Node.js and the frameworks running on the client. "JavaScript everywhere" became the mantra for some.

http://www.infoworld.com/article/2866712/php/php-vs-node-js-an-epic-battle-for-developer-mind-share.html

Node.js Server-side Scripting

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

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello World\n');
}).listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

- Node.js supports **server-side scripting** to produce **dynamic web page**.

JavaScript

- **Loosely typed**
  - The same variable could be **re-assigned to values of different types**.
- Variable naming rules
  - Names are **case sensitive**.
  - Names begin with **a letter** or the **underscore** character.

JavaScript Types

- To parse a string into a number, we can use the parseInt() function.

```
numVariable = parseInt(stringValue);
```

- If the input string doesn't start with a number, NaN **(Not-A-Number)** will be returned.
- We can check for this value with the isNaN() function.

```
isNaN(numVariable)
```

Falsy Values

https://www.sitepoint.com/javascript-truthy-falsy/

- A value is either "**truthy**" or "**falsy**".
- A variable **without a value assigned**, is of type **undefined**.

```
> var undefinedVar;
undefined
> typeof undefinedVar
'undefined'
```

```
> if (undefinedVar)
  foo = "It's truthy";
  else foo = "It's falsy"
"It's falsy"
```

Falsy Value	Type
false	Boolean
0	Number
NaN	Number
'' or "" (empty string)	String
null	Object
undefined	Undefined

Equality Comparisons

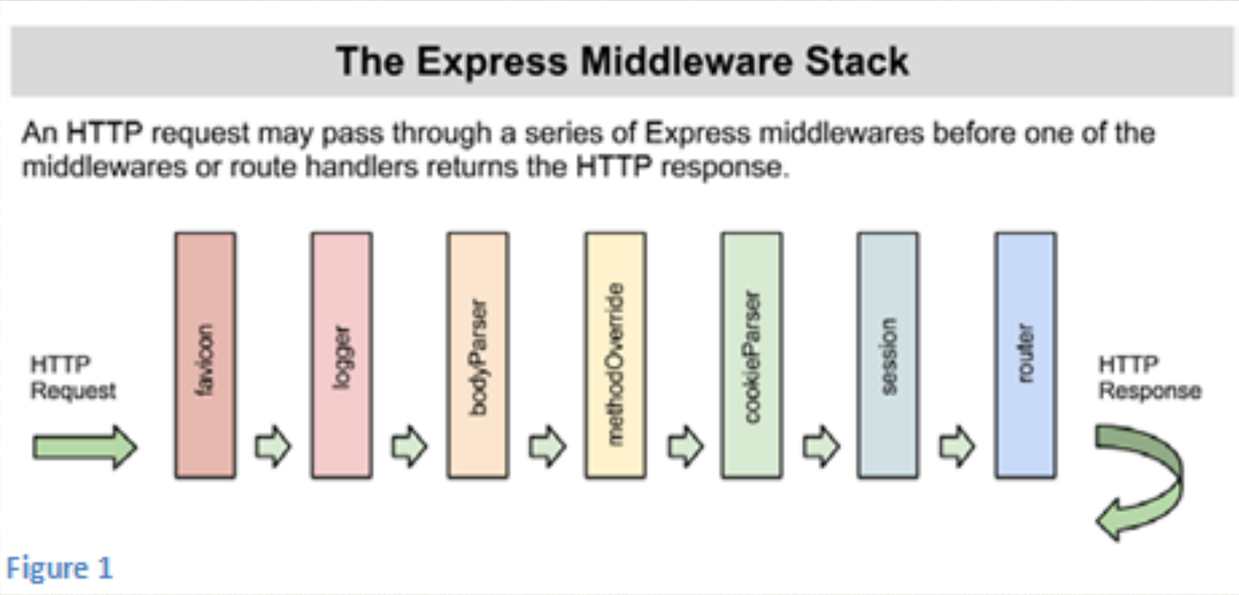
`==`	`true`	`false`	`0`	`''`
`true`	true	false	false	false
`false`	false	true	true	true
`0`	false	true	true	true
`''`	false	true	true	true

`===`	`true`	`false`	`0`	`''`
`true`	true	false	false	false
`false`	false	true	false	false
`0`	false	false	true	false
`''`	false	false	false	true

HANDLING FORM WITH EXPRESS

Express

- A **web application framework**, with several layers
- Bottom layer:
  - **Node**'s 'http' server
- **Middleware stack**:
  - Body parser, cookie parser, session … all added to the **request** (req) **object**.



Express

- **Routings**
  - Map **requests** to specific **route handlers**
  - May access **database**
- **Template Engine**
  - **Combine data and template** to produce final HTML.

```
router.post('/form', function (req, res) {

    var response = {
        header: req.headers,
        body: req.body
    };

    res.json(response);
});
```

```
<tr>
    <td><%= booking.name %></td>
    <td><%= booking.numTickets %></td>
    <td><%= booking.email %></td>
</tr>
```

The Request Object

- Under Node.js/Express, the req object represents the HTTP request and has properties:

req.params	**Parameters** (e.g., /:id)
req.query	**Query string**, form data submitted via GET
req.body	**Request body**, form data submitted via POST

Web Template Engine

- A web template engine work at the server-side, it combines **dynamic elements** with **a static template** to produce HTML.

```
<tr>
    <td><%= booking.name %></td>
    <td><%= booking.numTickets %></td>
    <td><%= booking.email %></td>
</tr>
```

```
<tr>
    <td>Tony Stark</td>
    <td>2</td>
    <td>tony@stark.com</td>
</tr>
```

Response to client

- Examples include **Embedded JavaScript** (ejs), Jade, Handlebar and Mustache.

Web Template Engine

```
<table>
    <% for (var booking of bookings) { %>

        <tr>
            <td><%= booking.name %></td>
            <td><%= booking.numTickets %></td>
        </tr>

    <% } %>
</table>
```

<% %>	‘Scriptlet’ tag, for **control-flow**, no output
<%= %>	**Outputs** the value into the template (**HTML escaped**)
<%- %>	**Outputs** the **unescaped** value into the template

HTML Escape

Unescaped	Escaped	Meaning
<	<	Less than
>	>	Greater than
"	"	Quotation mark
&	&	Ampersand
(a space)		Non-breaking space

- Some characters are **reserved in HTML** and must be **escaped into** the corresponding HTML entities.
- Reference http://www.w3schools.com/tags/ref_charactersets.asp

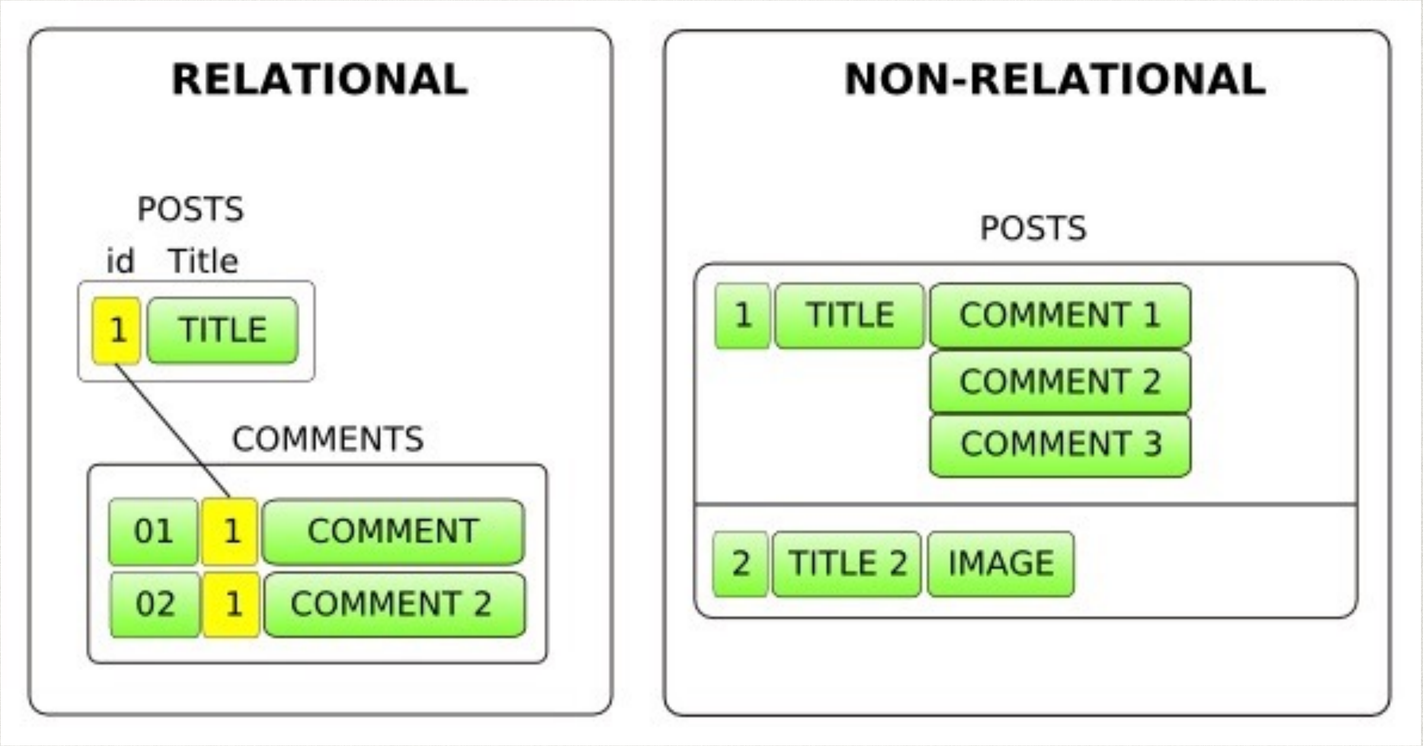
DATABASE

Database

- **A collection of information** that is organized so that it can easily be accessed, managed, and updated.
- Some popular database engines:
  - **Relational**:
    - PostgreSQL, Oracle DBMS, MySQL, Microsoft SQL Server, etc.
  - **Document-based NoSQL**:
    - MongoDB, CosmosDB, etc…

Relational vs. NoSQL

- Relational database has **fixed schema**.
- NoSQL database is **schema-free**.
  - It can represent in **a single entity** that would normally **require several tables to properly represent in a relational database**.



Collection and Document

Relational	NoSQL
Database	Database
Table	**Collection**
Row	**Document**
Column	Field

MongoDB APIs

- MongoDB supports database operations, with corresponding **function calls**.

```
db.users.insertOne(          ← collection
   {
      name: "sue",           ← field: value
      age: 26,               ← field: value   } document
      status: "pending"      ← field: value
   }
)
```

```
db.users.find(                          ← collection
   { age: { $gt: 18 } },                ← query criteria
   { name: 1, address: 1 }              ← projection
).limit(5)                              ← cursor modifier
```