WEB SYSTEM BACKEND

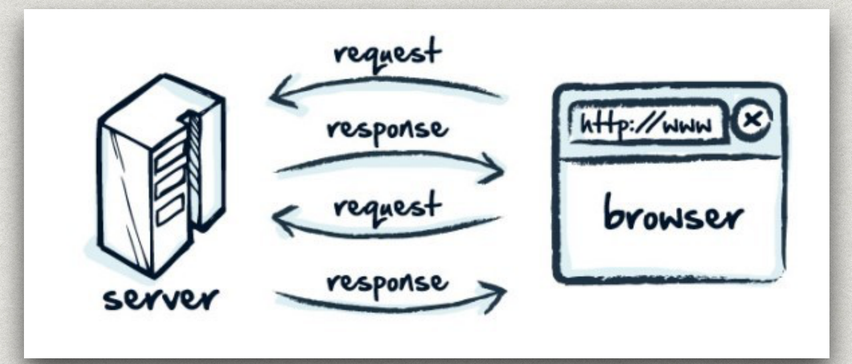
ITEC2016 - DATA-DRIVEN VISUALIZATION FOR THE WEB

CHAPTER 8 - 2018 - HKBU

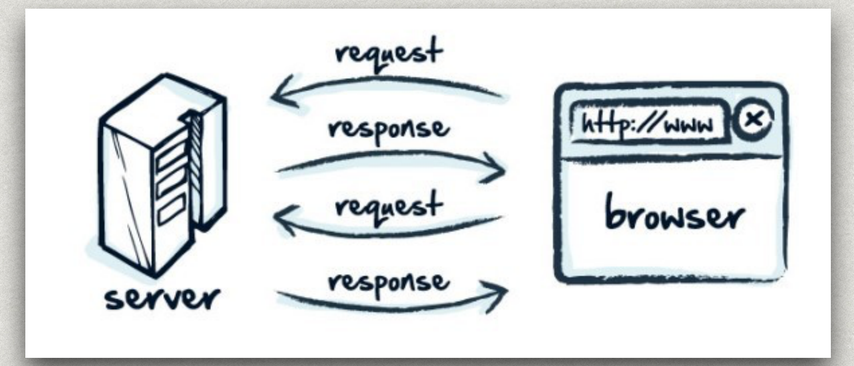
DR. MARTIN CHOY

Agenda

- Basics of the Internet, **client and server model**
- **HTTP** Protocol
- More on HTML - **HTML Form**
- **Node.js** - JavaScript runtime environment
- Web Application Development with **Express.js**



IP Addresses



- How to specify **which server to connect**?
- 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

- The **Domain Name System (DNS)** enables a domain name to be converted into a valid IP address.
- **Easier to remember** and the names usually reflect the identity of the owner.
- The DNS consists of a **number of dedicated servers** that maintain naming information for different zones.
- The mapping between domain name and an IP address **can change**, so the same name can **migrate** between different host systems.

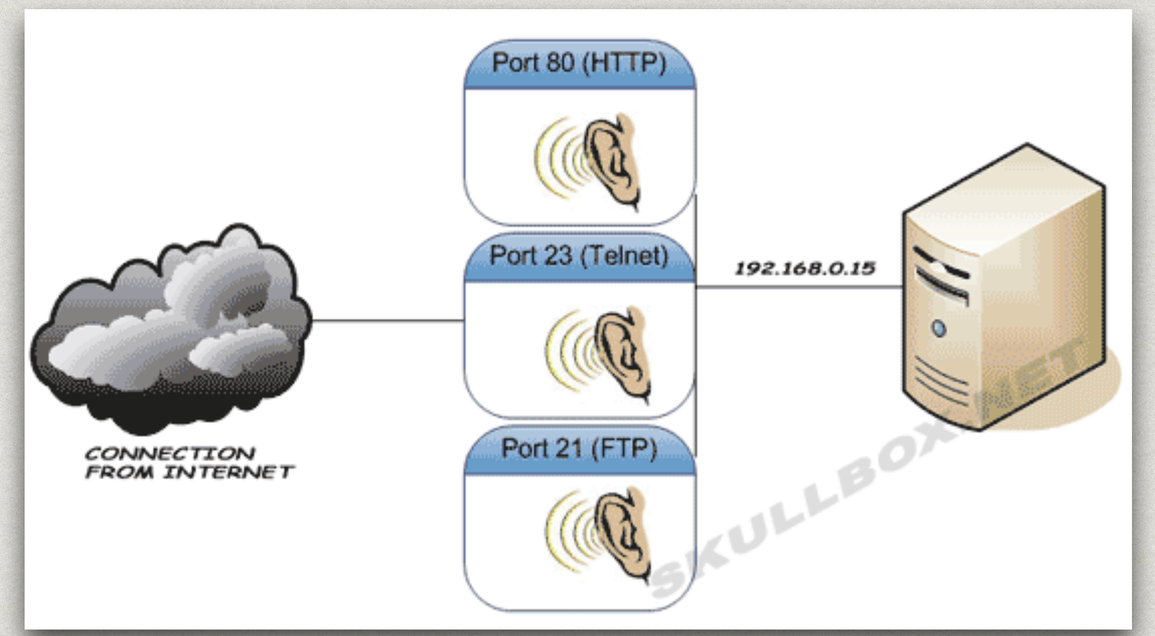
Demo

- **Check the current IP**
  - ifconfig
- Check the **DNS server** and **IP of a domain name**
  - nslookup
  - type www.tvb.com

Local Machine

- To reference the **local machine**, we can use
  - Domain name (**localhost**) or
  - IP address (**127.0.0.1**)

Port



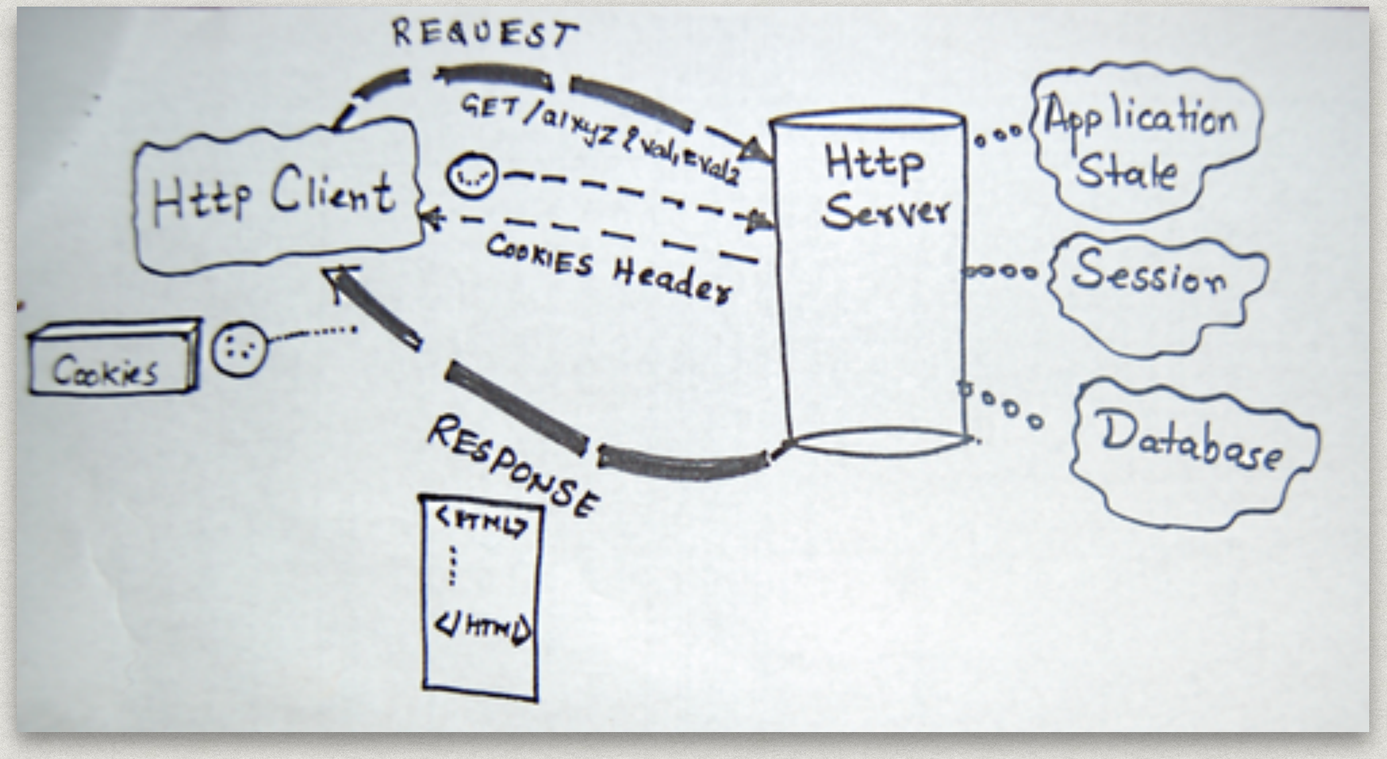
- Different **Internet services** will use different **ports**
  - Web server usually uses **port 80**.
  - **Ports 8000 and 8080** are common ports for software providing **http services** that is not a core http server.

URL

Example of URL: http://www.comp.hkbu.edu.hk:80/RCUC/project.html

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

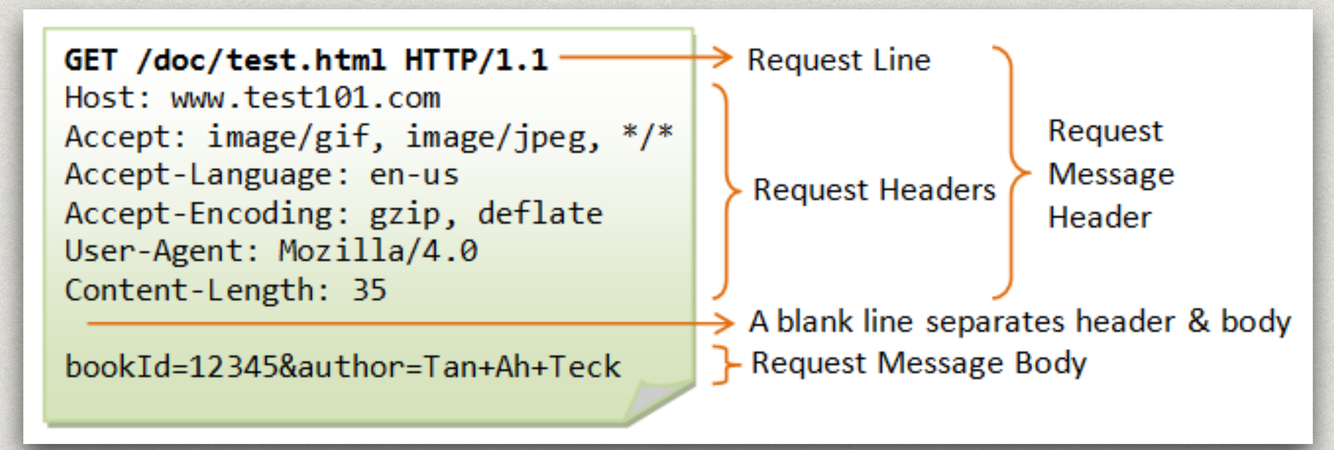
HyperText Transfer Protocol (HTTP)

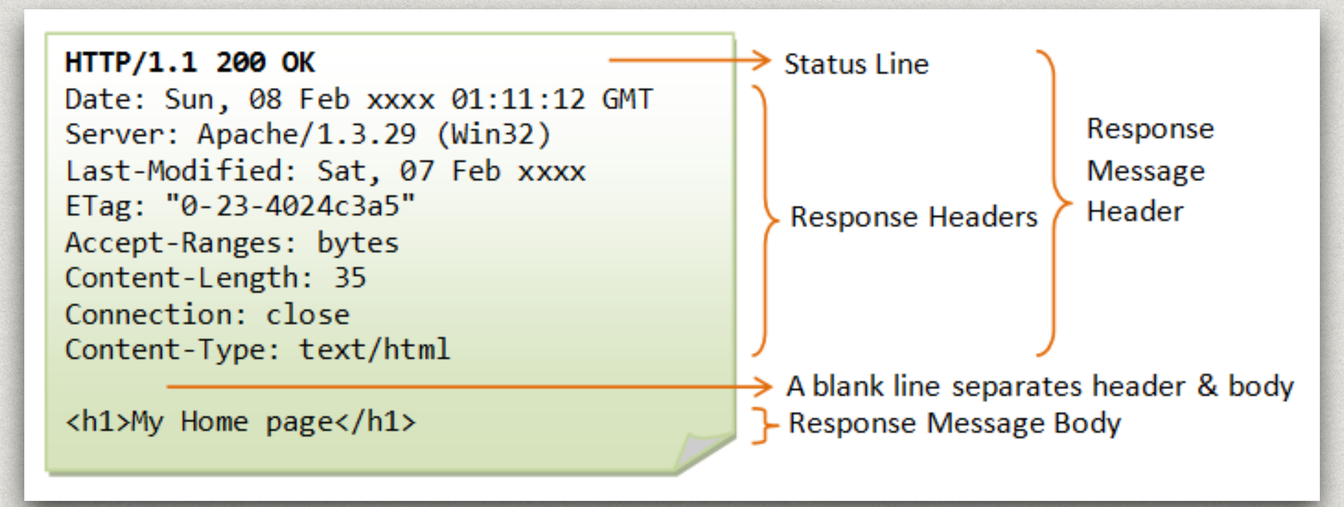


- A **request-response protocol** that enables the transfer of hypertext documents and other files between a **client and server**.

Request and Response

- **Request methods**:
  - GET, POST and more…
- Response:
  - **Status line** of a succeeded request: HTTP/1.1 200 OK





Demo

- **Telnet** to a web server on **port 80**:
  - telnet [IP or name of the web server] 80
- Send the following contents
  - GET /index.html HTTP/1.1
  - Host: [IP or name of the web server]
- Observe the response.

```
>telnet www.comp.hkbu.edu.hk 80
Trying 158.182.8.1...
Connected to www.comp.hkbu.edu.hk.
Escape character is '^]'.
GET /v1/ HTTP/1.1
host: comp.hkbu.edu.hk

HTTP/1.1 200 OK
Date: Mon, 17 Sep 2018 03:25:20 GMT
Server: Apache
Set-Cookie: PHPSESSID=597ed9658f895f995f
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-
Pragma: no-cache
Transfer-Encoding: chunked
Content-Type: text/html
```

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.

HTML Form

- Form is used to pass data to a server.
- The **submit button** will trigger the submission.

```
<form action="/person/create" method="POST">
    <input name="pName" type="text">

    <input name="pAge" type="number" min=0 max=120>

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

- **action** specifies **where** to send the form-data when a form is submitted.
- **method** specifies the **HTTP request method** for sending form-data.

HTML Form Elements

- **<Input>** with different types
  - text, number, url, date, checkbox

button	date	email	image	password	reset	tel	url
checkbox	datetime	file	month	radio	search	text	week
color	datetime-local	hidden	number	range	submit	time	

- **<Select>** to give a combo box for selection.

Name Attribute

```
<input name="pName" type="text">
```

- 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 after a form is submitted**.
  - Note: **Only form elements with a name attribute** will have their values passed when submitting a form.

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

WEB APPLICATION DEVELOPMENT WITH NODE.JS & EXPRESS.JS

PHP vs. Node.js

- Node.js allows **JavaScript** 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

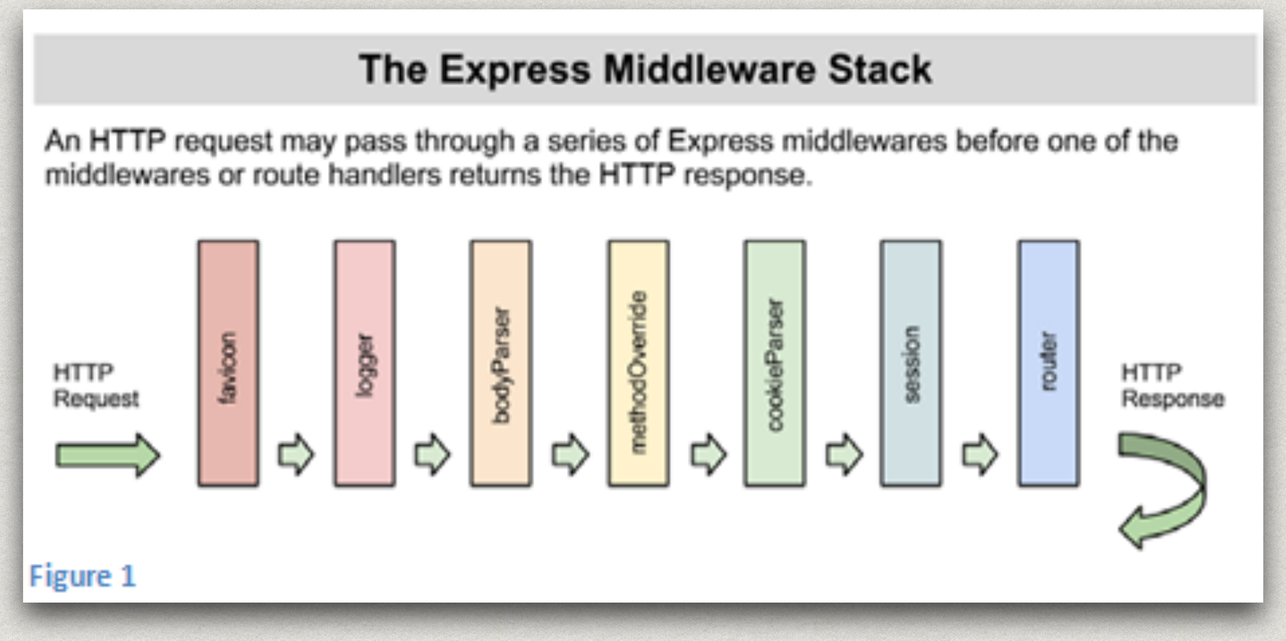
Node.js

```
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/');
```

- A **JavaScript** runtime environment
  - for **server-side scripting** to produce **dynamic web page**.
- Over 700k **reusable packages/modules**
  - could be installed via **npm**, the **pre-installed package manager**

Express.js

- 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.js

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

    res.json(req.body);

});
```

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

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

Form Handling in Express.js

- For both **GET or POST** methods, Express.js sets up **a constant** for each **form element** with the same name as the element.
  - **req.params** contains the values in the **parameter list (e.g., /:id)**.
  - **req.query** contains values from the **query strings** (submitted via GET)
  - **req.body** contains values from the **request body** (submitted via POST)

Web Template Engine

- Combines **dynamic elements** and **a static template** to produce HTML,

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

  - While the **dynamic elements** to be defined based on the **parameters of the web request**.

- Examples include: Jade, HandleBars, EJS…

Web Template Engine

- **Embedded JavaScript** (ejs) is a web template engine format.
  - Help to separate

    business logic `<%  %>` ,

    dynamic data `<%=  %>`

    with HTML, making the **code more readable**.

```
<% bookings.forEach( function(booking) { %>

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

<% }); %>
```

JSON

- JSON (**JavaScript Object Notation**) is a lightweight **data-interchange format**.
- JSON is built on two structures:
  - A collection of **name/value pairs**.
  - An **ordered list** of values.

```
[
  {
    "name": "Martin Choy",
    "age": 23,
    "id": 7,
  },
  {
    "name": "Kenny Cheng",
    "age": 22,
    "id": 8,
  }
]
```

JSON

- **Values** could be
  - primitives
  - object
  - array
  - array of objects…

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```