goto

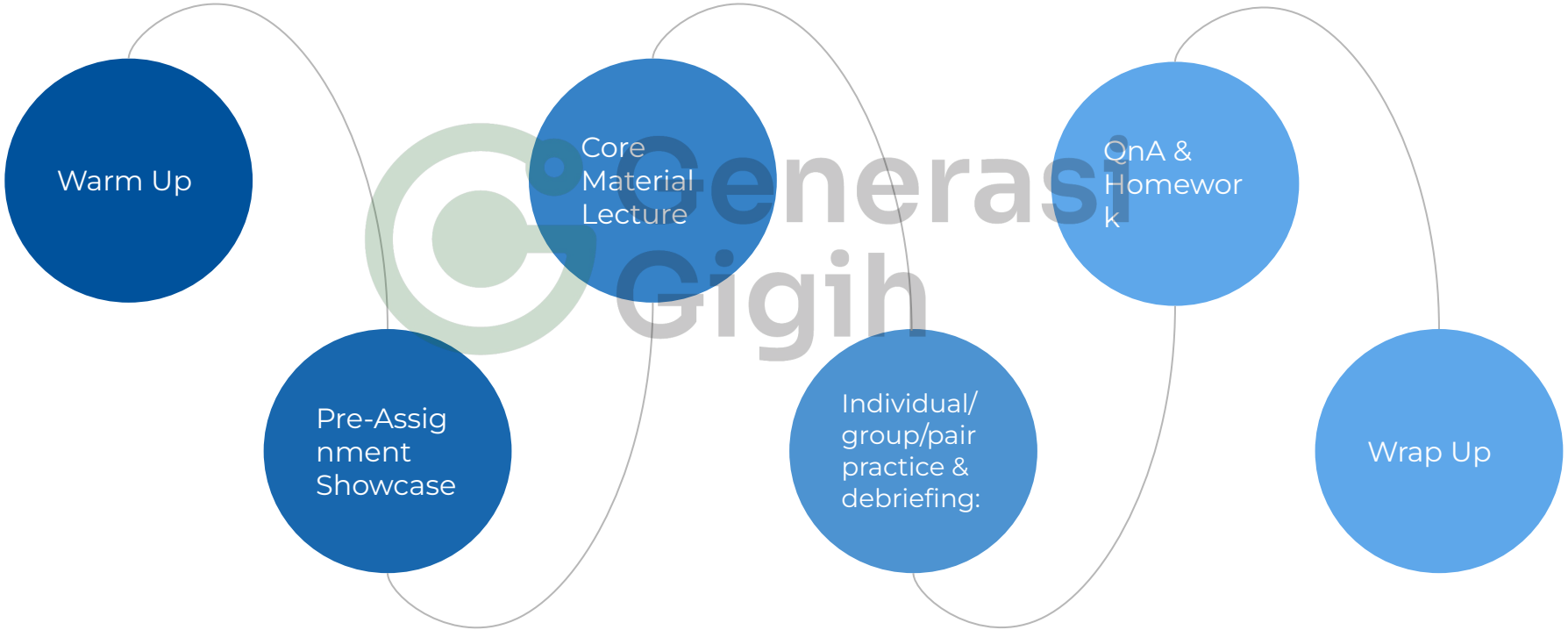
goto impact foundation

Generasi Gigih

Full Stack Engineer

Module 2.1: Introduction to Server Side Development with Node JS

Our Agenda


Warm Up
Pre-Assignment Showcase
Core Material Lecture
Individual/ group/pair practice & debriefing:
QnA & Homework
Wrap Up

Generasi Gigih

Let’s Warm Up!

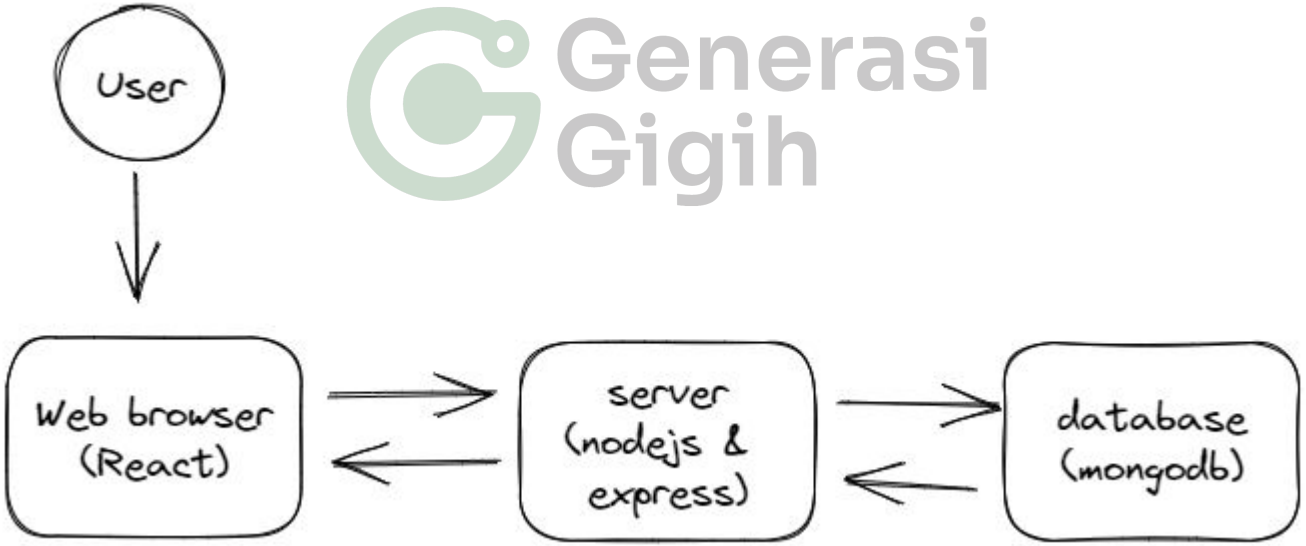
Let’s Discuss

[Session outline]

Generasi Gigih

Let’s Talk About The Materials

Review: Full Stack Engineering



Introduction to Web

Web App Evolution

1. **Static Web App**

   Pre-built HTML, CSS and JS, every user will get the same content.

2. **Dynamic Web App**

   HTML, CSS, and JS can be generated by server app.
   Server can generate personalized content, and typically rely on database to manage user's data. Enable web with user generated content (social media, ecommerce, etc)

3. **Single Page Application + API**

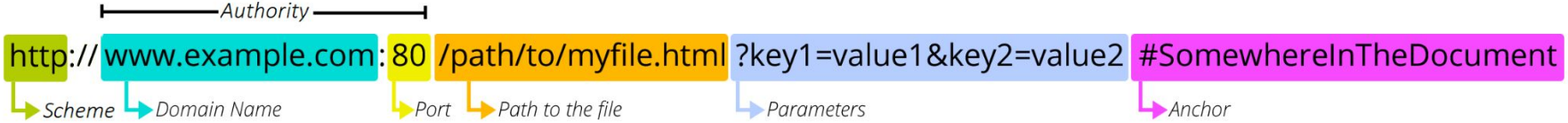
- HTML, CSS and JS are typically pre-built, but the content is generated dynamically using data provided by server app.
- Server app response to client in Data format (i.e. JSON) instead of HTML page.
- Enhance user experience, but require client-side processing with Javascript.

Client-Server Communication

HTTP (HyperText Transfer Protocol) is a protocol designed to standardize the communication between client and server

HTTP is built on top of TCP, a connection oriented protocol that provide reliable data transmission over the internet protocol.

URL (Universal Resource Locator)

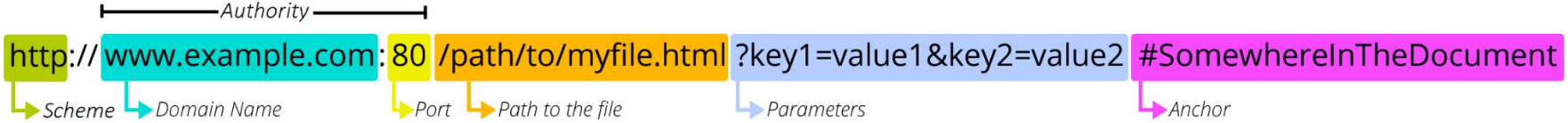


Source: developer.mozilla.org

In the context of Web:

1. **Scheme:** HTTP or HTTPS for encrypted connection
2. **Authority:**
   a. **Domain Name or IP address**: using domain name requires DNS that translate the name to IP address
   b. **Port (optional)**: 80 is the default for HTTP and 443 for HTTPS.
3. **Path:** Location of file or routes
4. **Parameters (Optional)**
5. **Anchor (Optional)**

TCP (Transmission Control Protocol)



TCP is a transport layer protocol of HTTP, it provides reliable data transfer meaning the data received by the receiver will be the exact same as the data transferred by sender.

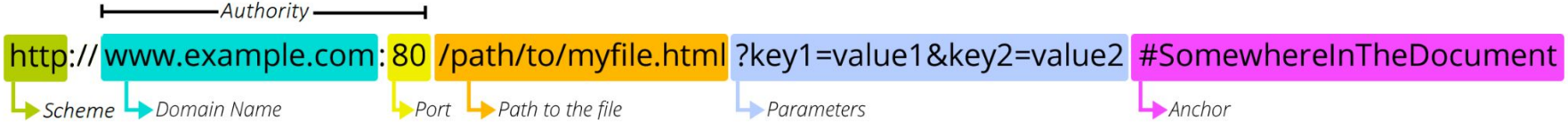
Domain Name and Port are required to initiate TCP Connection

HTTP (HyperText Transfer Protocol)

HTTP is the application level protocol that used in web as a communication protocol between client and server.

HTTP use Request-Response model, the client send an HTTP request and the server receive, process and response to the client with an HTTP response.

HTTP Request Anatomy



Is equal to

```
> GET /path/to/myfile.html?key1=value1&key2=value2 HTTP/1.1
> Host: example.com
> User-Agent: curl/7.84.0
> Accept: */*
```

- Method (Mandatory): Define the desired action
- Path (Mandatory): Define the resource location within the server, can be file or abstract name
- Version (Mandatory): HTTP/1.1 is the most common used version
- Headers: Host header is mandatory as per HTTP/1.1 spec, other headers is optional.

Writing HTTP Request

We're using info.cern.ch as an example as it's the first web created

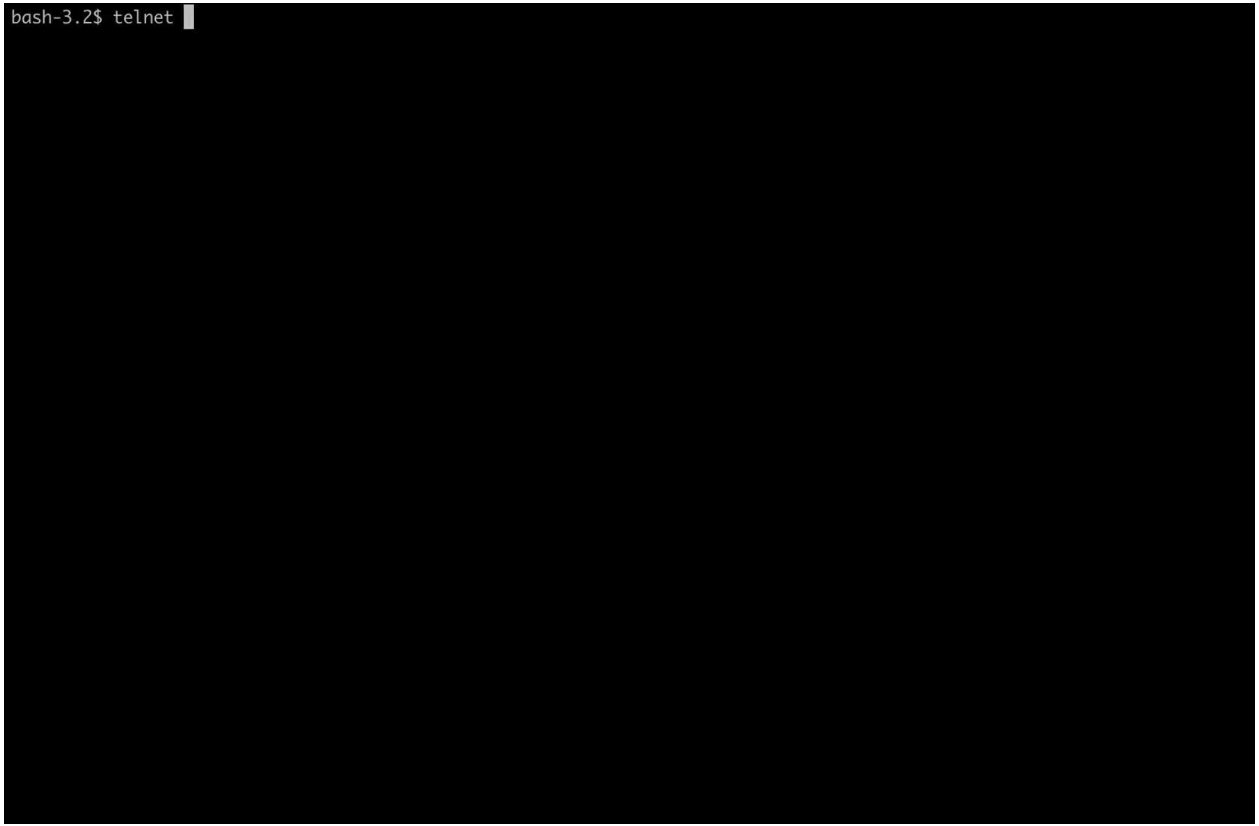
```
~ telnet info.cern.ch 80
Trying 188.184.21.108...
Connected to webafs706.cern.ch.
Escape character is '^]'.
```


With this command we established the connection with cern server port 80 (HTTP server), now to write HTTP Request with the correct format

```
GET / HTTP/1.1
Host: info.cern.ch


HTTP/1.1 200 OK
Date: Thu, 30 Mar 2023 08:23:17 GMT
Server: Apache
…
```

Writing HTTP Request - Demo



Writing HTTP Request - cURL

cURL is the command used for creating HTTP call in terminal

```
~ curl -v http://info.cern.ch/
*   Trying 188.184.21.108:80...
* Connected to info.cern.ch (188.184.21.108) port 80 (#0)
> GET / HTTP/1.1
> Host: info.cern.ch
> User-Agent: curl/7.84.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Accept-Ranges: bytes
< Content-Length: 646
< Connection: close
< Content-Type: text/html

<html><head></head><body><header>
<title>http://info.cern.ch</title>
</header>
…
```

This command equals to opening http://info.cern.ch/ using browser.

Creating Simple Web App with Node JS - 1

helloworld.js

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


const server = http.createServer((req, res) => {
 res.writeHead(200, {'Content-Type': 'text/plain'});
 res.end('Hello, World!');
});
server.listen(3000, () => {
 console.log('Server running on port 3000');
});
```

Terminal 1 - Run helloworld server

```
> node helloworld.js
Server running on port 3000
# Do not close the program
```

Creating Simple Web App with Node JS - 2

Terminal 2 - Get data from helloworld.js

```
~ curl -v localhost:3000
*   Trying 127.0.0.1:3000...
* Connected to localhost (127.0.0.1) port 3000 (#0)
> GET / HTTP/1.1
> Host: localhost:3000
> User-Agent: curl/7.84.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Date: Thu, 30 Mar 2023 10:03:03 GMT
< Connection: keep-alive
< Keep-Alive: timeout=5
< Transfer-Encoding: chunked
<
* Connection #0 to host localhost left intact
Hello, World!%
```

Alternatively try to access with browser: http://localhost:3000

Generasi Gigih

Congratulations!
You have created your first Web App

Generasi Gigih

Now, Let’s add more page to the Web App

Handling multiple path - 1

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

const server = http.createServer((req, res) => {
 res.setHeader('Content-Type', 'text/plain');
 if (req.url === '/') {
   res.statusCode = 200;
   res.end('This is the index page');
 }
 else if (req.url === '/about') {
   res.statusCode = 200;
   res.end('This is the about page');
 }
 else {
   res.statusCode = 404;
   res.end('Page not found');
 }
});
server.listen(3000, () => {
 console.log('Server running on port 3000');
});
```

Handling multiple path - 2

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

const server = http.createServer((req, res) => {
 res.setHeader('Content-Type', 'text/plain');
 if (req.url === '/') {
   res.statusCode = 200;
   res.end('This is the index page');
 }
 else if (req.url === '/about') {
   res.statusCode = 200;
   res.end('This is the about page');
 }
 else {
   res.statusCode = 404;
   res.end('Page not found');
 }
});
server.listen(3000, () => {
 console.log('Server running on port 3000');
});
```

Default NodeJS

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
 res.send('This is the index page');
});

app.get('/about', (req, res) => {
 res.send('This is the about page');
});

app.use((req, res) => {
 res.status(404).send('Page not found');
});

app.listen(3000, () => {
 console.log('Server running on port 3000');
});
```

With Express

Introducing: Express

Express is a minimalistic Node.js framework for building web applications

Why Express?

- Minimalistic: High performance, flexible
- Declarative Routing: Clean and more manageable
- Community Support: Large community support

This class will use Express on top of Node.js going onward.

Hands on: Creating your first Express Project

1. Create a new folder called HelloWorld
2. Open the folder in VS Code
3. `Create file index.js`
4. Open terminal in VS Code
5. Run: `npm init`
6. Run: `npm install express`
7. Copy code from: https://expressjs.com/en/starter/hello-world.html
8. Open http://localhost:3000 from your browser

Generasi Gigih

Q&A!

Generasi Gigih

Finally, Let’s Wrap Up!