COM3504/6504 The Intelligent Web

Lecture 2:
Server-side development and Node.JS

University of Sheffield | Department of Computer Science

Learning Objectives

- During this unit you will learn
  - What is server-side programming
  - Basics of client-server architecture Static vs dynamic websites
  - Node.JS

    Serving static pages with Node.Js

    Serving dynamic pages with Node.Js
  - Routes
  - Views and Ejs

SERVER SIDE PROGRAMMING

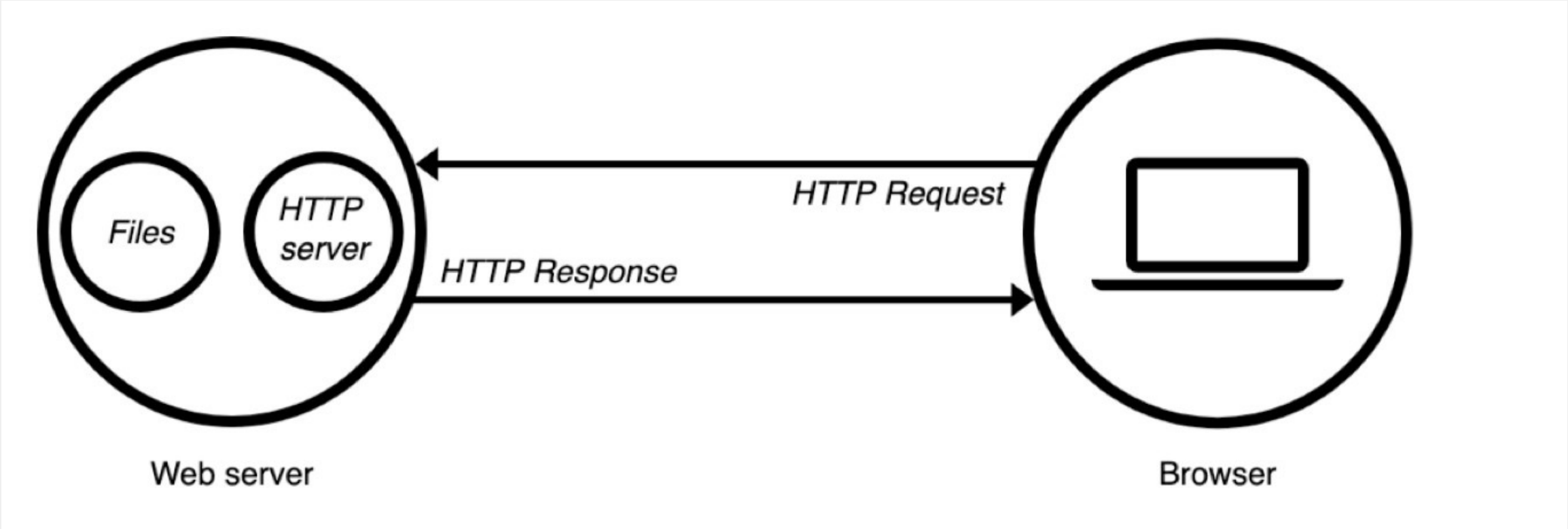
University of Sheffield | Department of Computer Science

What is it?

- In previous module you might have seen:
  - CLIENT-SIDE PROGRAMMING

    HTML, CSS, JavaScript
- Server-side programming
  - The activity happens on a server
  - Server-side code has full access to the server operating system
    - E.G. your browser sends a request to a web server which proceeds to search for the requested file in its own storage space

A basic client server architecture



Advantages of server-side programming

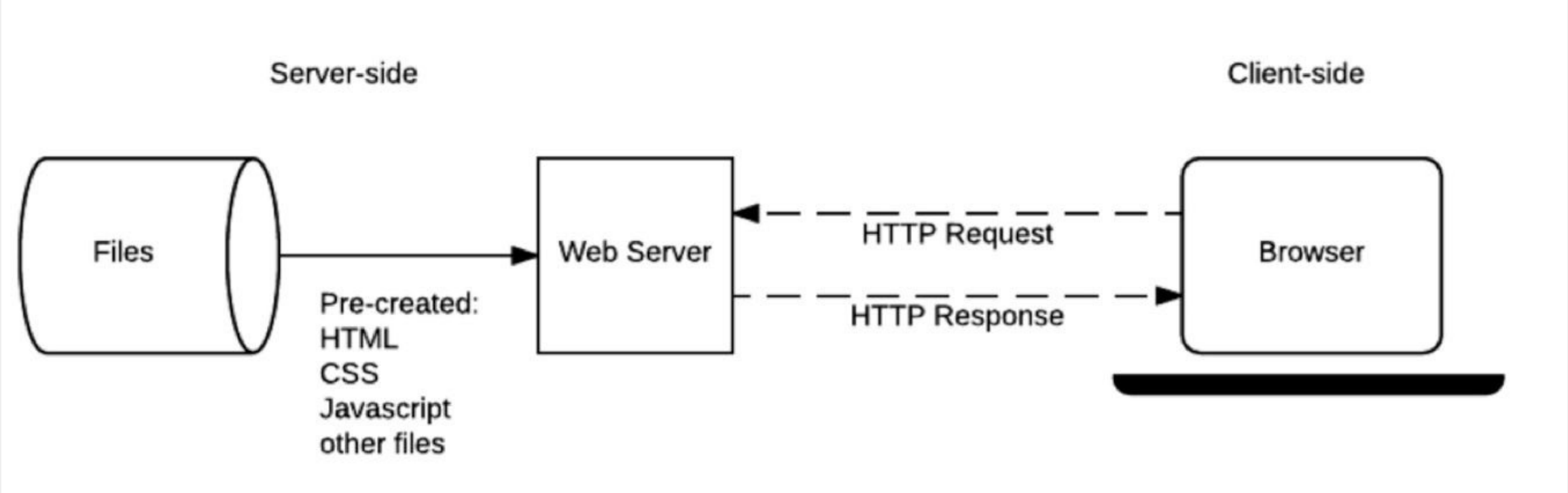
- Creation of information tailored for individual users
- Efficient storage and delivery of data
  - Store the information in a database and dynamically construct and return HTML and other types of files
- Control access to content
- Restrict access only to authorised users
- Store session information
  - And tailor content accordingly
- Can send notifications

STATIC VS DYNAMIC WEBSITES

Static web sites

- You are looking for a specific web page
- Your browser sends a request to a web server
- The web server searches for the requested file in its own storage space.
- On finding the file, the server reads it, processes it as needed, and sends it to the browser.

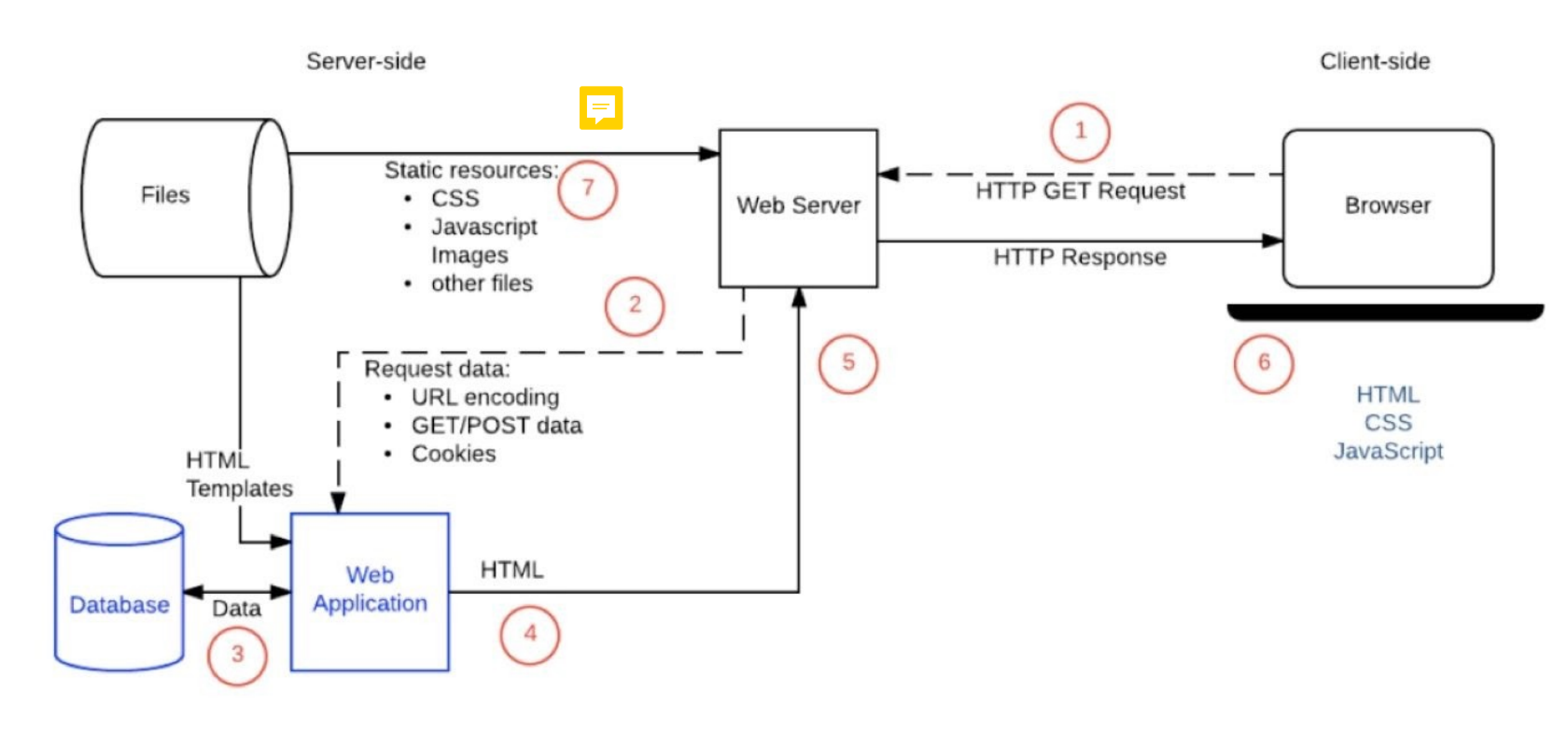
Static web site architecture



Dynamic websites

- Web pages do not exist!
- Data is saved in databases
- The response (web page) is generated *dynamically* only when needed
  - by inserting data from a database into placeholders in HTML templates
  - And styling it with CSS

Dynamic web site architecture



University of Sheffield | Department of Computer Science

Full stack web development

COM1008

Back-end	Front-end - Interface	Database
functionality		
• PHP • Rails • ASP • **Node.JS**	• **HTML** • **JavaScript** • **CSS** • Kotlin • Swift	• MySQL • SQLServer • **MongoDB**

SERVER SIDE FRAMEWORKS

University of Sheffield | Department of Computer Science

What are they?

- Server-side frameworks are software development containers
- Make writing code to handle server-side operations easier
- They have pre-defined methods and pre-written code

Main functionalities

- Handle requests and responses
- Map URLs for different resources/pages to specific handler functions
- Route requests to the appropriate handler in the code
- Abstract and simplify database access
- Provide templating system to render the data
  - e.g. specify an HTML structure with placeholders
  - Dynamically insert data

NODE.JS

What is Node.Js?

- Node.js is an open source server framework
  - Allows to run Javascript on a server
- It can help you building fast, scalable network applications
  - It is different from other server-side languages that you might have seen (e.g. PhP)
- single-threaded, non-blocking, **asynchronously** programming

Why is Node.Js different?

Traditional server-side

1. The browser sends a request to the server
2. The server checks for data in file system and reads the data.
3. The server returns the content to the client.
4. The server is ready to handle the next request.

Need to handle one request before to be ready for the next one

Node.Js

1. The browser sends a request to the server
2. The server is Ready to handle the next request.
3. When the file system has opened and read the data, the server returns the content to the client.

Node.Js is non blocking

- Node.js is event based:
- JavaScript is an event-based language, so anything that happens on the server triggers a non-blocking event
  - Each new connection fires an event; data being received from an upload form fires a data-received event;
  - requesting data from the database fires an event.
- This increases efficiency and multi-user support.

Advantages of Node.Js

- Performance and scalability.
  - Node is fast and can scale up easily
- Node is perfect for offering a RESTful API
  - A web service which takes a few input parameters and passes a little data back
- Simple data manipulation without a huge amount of computation.
- It is JavaScript

Disadvantages of Node.Js

- You don’t want to use Node.js for CPU-intensive operations
- Using it for heavy computation will annul nearly all of its advantages

What can Node.Js do?

- Generate dynamic page content
- Create, open, read, write, delete, and close files on the server
- Collect form data
- Add, delete, modify data in your database

Basic Node.js rules

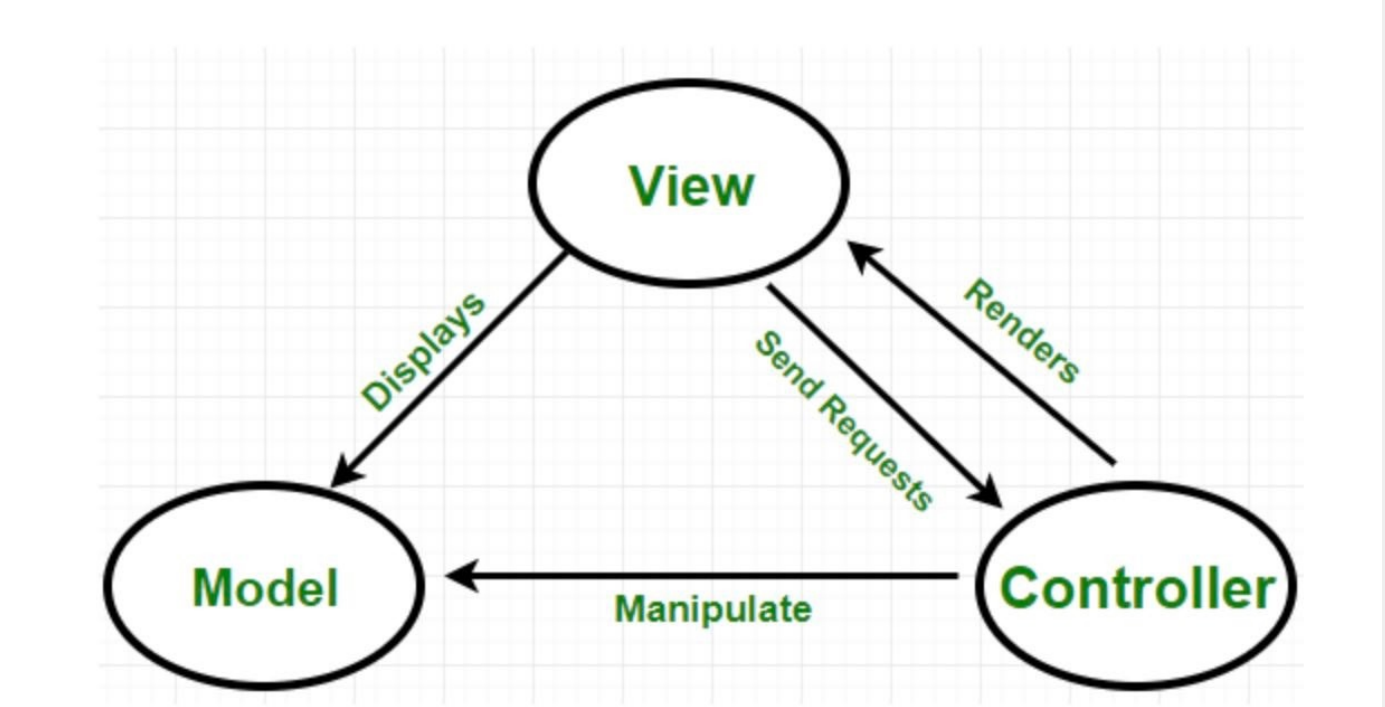
- A Node.js file contains tasks that will be executed on certain events

  e.g. someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension .js

NODE.JS CLIENT-SERVER ARCHITECTURE

University of Sheffield | Department of Computer Science

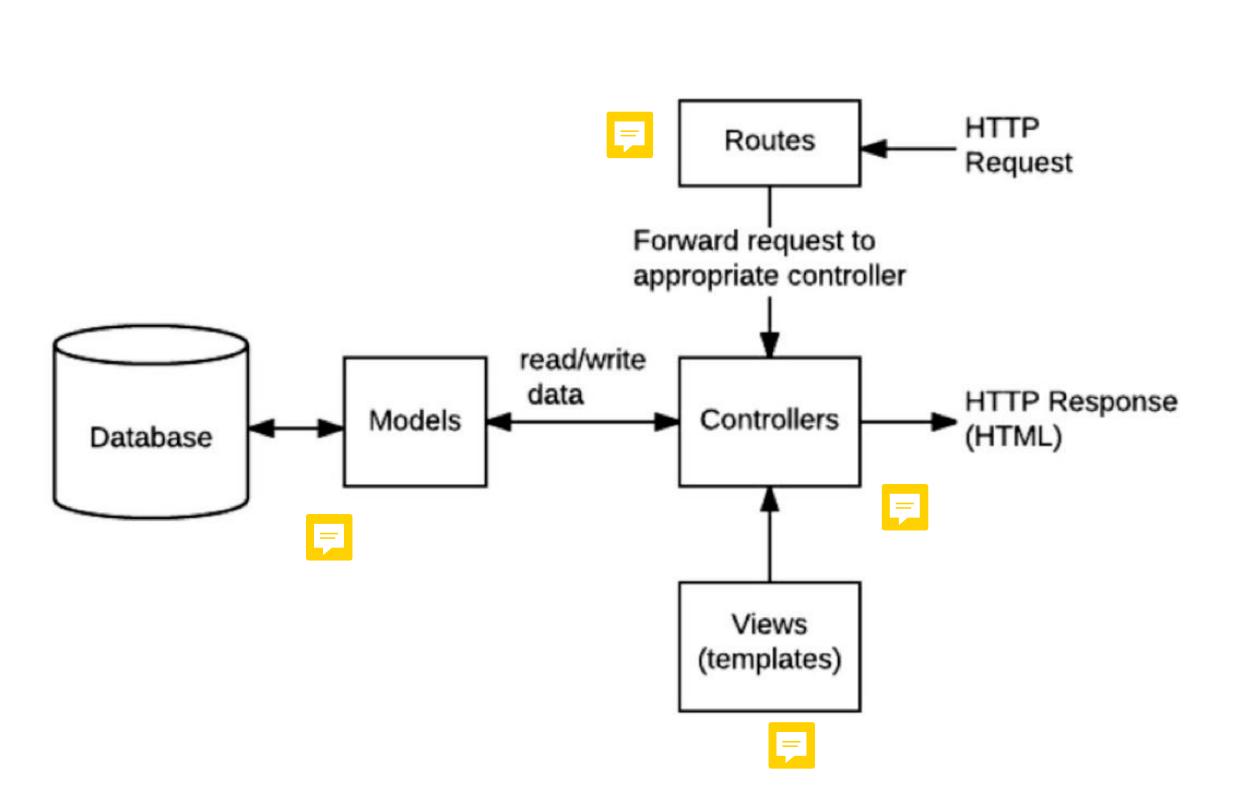
MVC Architecture



MVC architecture

Architecture diagram

More than one for each



NODE.JS MAIN COMPONENTS

University of Sheffield | Department of Computer Science

Routes

- A route is the endpoint the user wants to reach
  - i.e. bbc.co.uk/music is a route
- defines the way in which the client requests are handled by the application endpoints Forward request to appropriate handler
- In node there are two main ~~ways~~ to handle routing
  - Native node.JS
  - Express
    - We will use Express

Controllers

- Controllers **control** the requests
- Generate appropriate responses
  - which are fed to the viewer.
  - the controller renders the appropriate view with the model data as a response.
- In node.js crontollers are javascript code that carries out the operations

Views

- Visualisation of data that gets sent back to a user
  - Typically templates that are filled by the controller with data retrieved from the DB according to the model
- There are several engines that can be used in Node.Js to render the views
  - We will look at ejs

Models

- The model represents the structure of data, the format, how data is handled
- There are various ways to build models depending on the Db used
  - We will look at MongoDB and Mongoose

NODE.JS MODULES

University of Sheffield | Department of Computer Science

https://nodejs.org/api/modules.html

What are modules

- In Node.js modules are a set of pre-defined functions to include in an application
  - Like JQuery or other Javascript libraries
- Node.js has a set of built-in modules which you can use without any further installation
- You simply use the function “require()” at the beginning of your node file
  - var http = require('http');
- Or you can create your own modules

List of pre-build modules

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.

List of pre-build modules - cont

net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols
tty	Provides classes used by a text terminal
url	To parse URL strings
util	To access utility functions
v8	To access information about V8 (the JavaScript engine)
vm	To compile JavaScript code in a virtual machine

EXPRESS

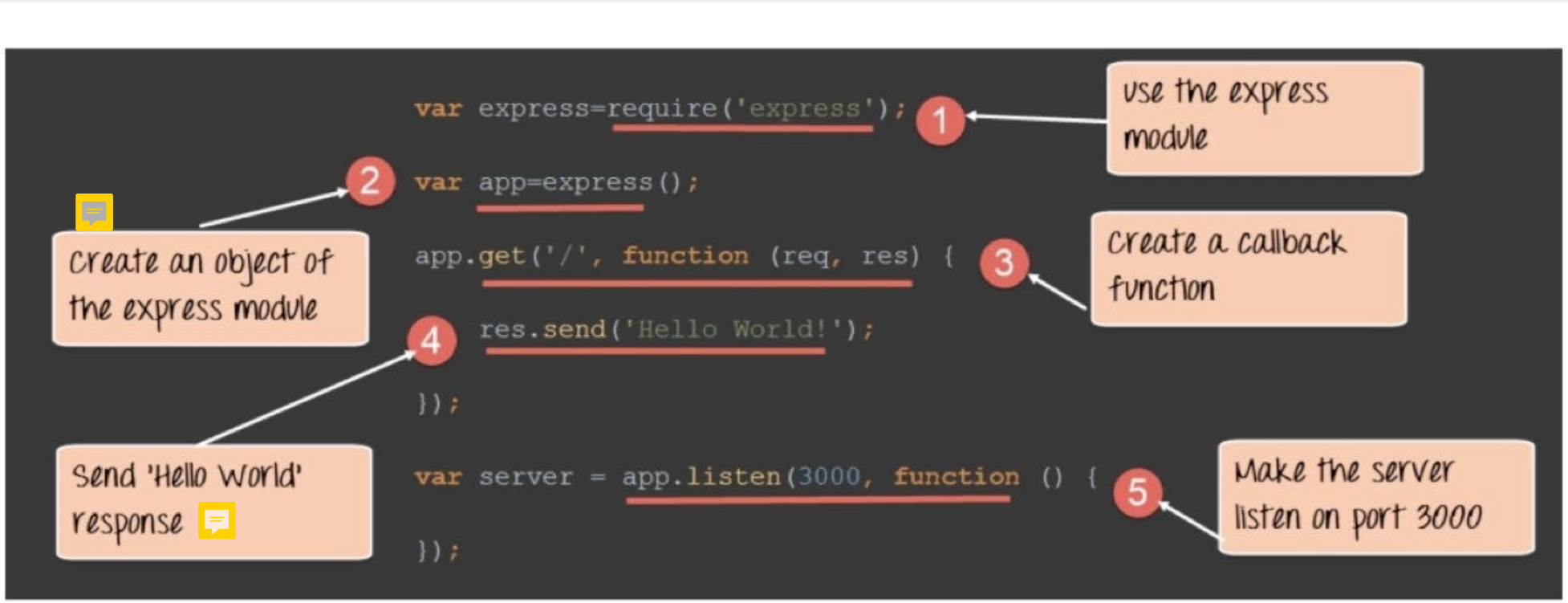
What is Express

- A minimal and flexible node.js web application framework with a set of features for web applications
- Especially useful for routing

Main functionalities of Express

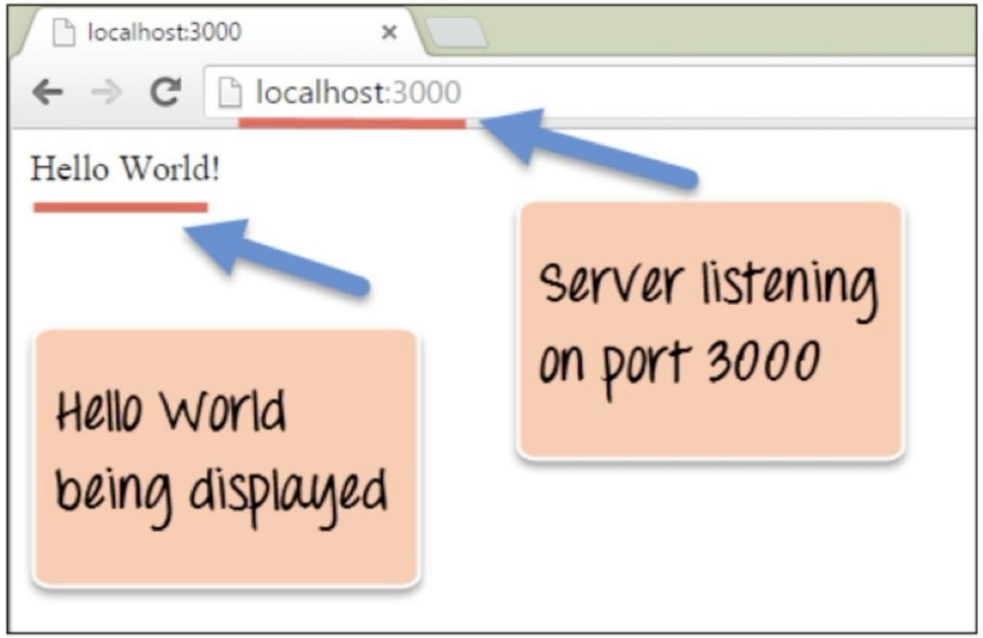
- Easy routing
  - handlers for requests with different HTTP verbs at different URL paths
- Integration with view rendering engines
  - generate responses by inserting data into templates.
- Set once common web application settings
  - Add additional request processing "middleware" at any point

Example of Express and Node.js - code



Example of Express and Node.js - output

Output:



Questions