SERVER-SIDE WEB PROGRAMMING 06016418

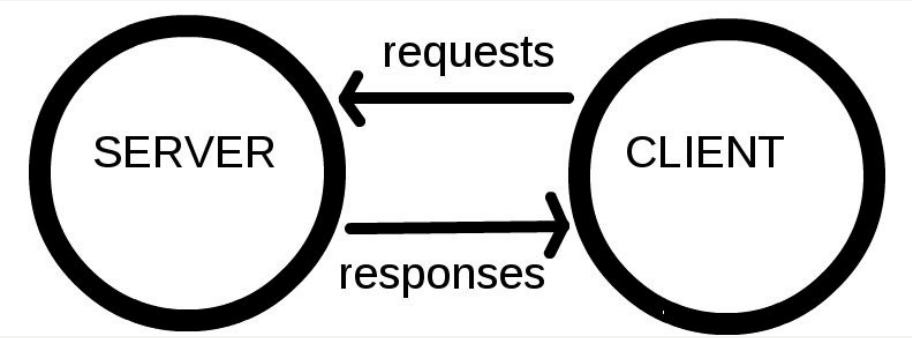
BY DR BUNDIT THANASOPON

HOW THE WEB WORKS

Src: https://developer.Mozilla.Org/en-us/docs/learn/getting_started_with_the_web/how_the_web_works

CLIENTS AND SERVERS

- Computers connected to the web are called **clients** and **servers**. A simplified diagram of how they interact might look like this:



- **Clients** are the typical web user's internet-connected devices (client agent: Web Browsers).

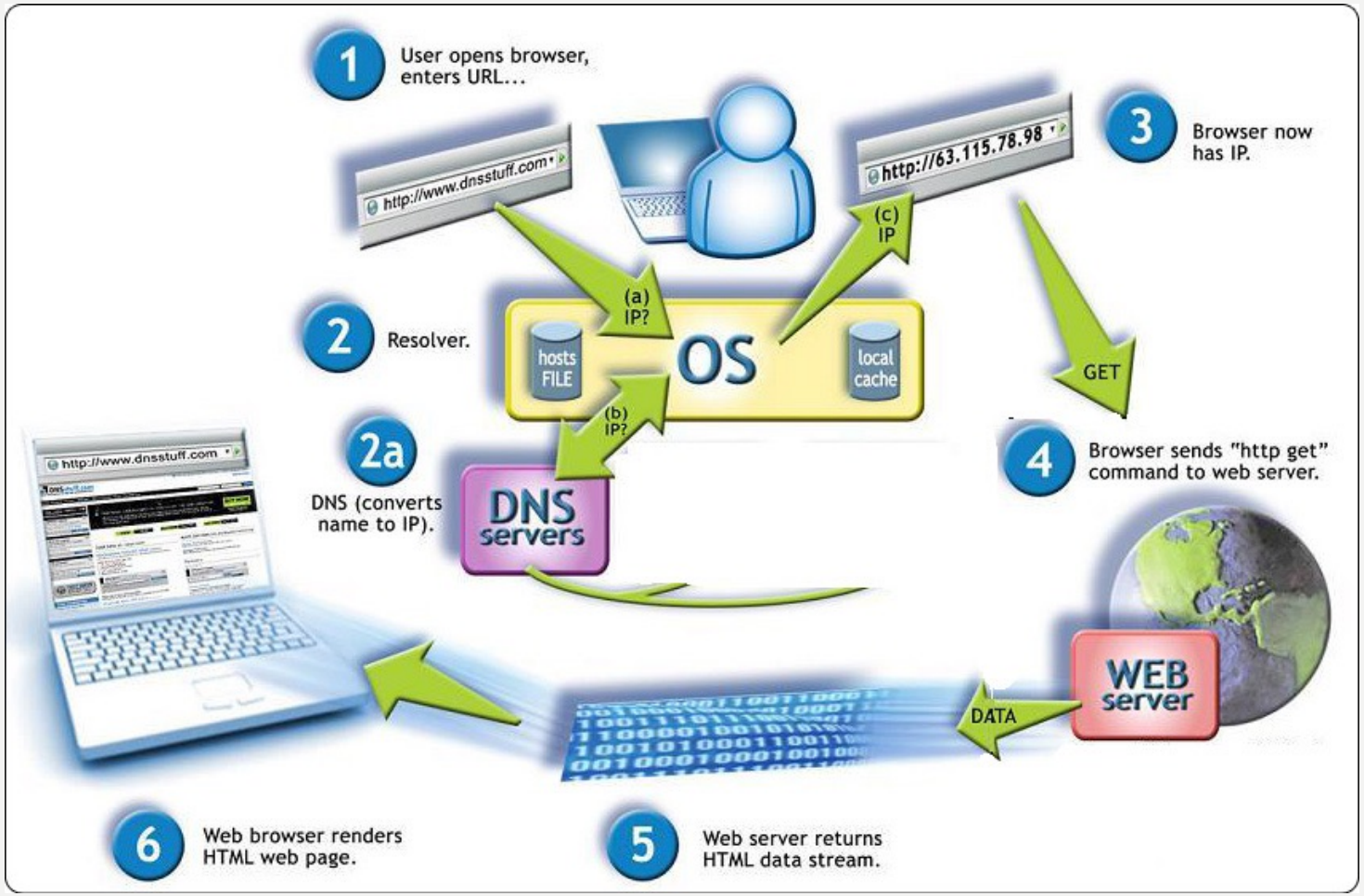
- **Servers** are computers that store webpages, sites, or apps (Web Servers).

OTHER IMPORTANT COMPONENTS

- **Your internet connection**: Allows you to send and receive data on the web.
- **TCP/IP**: Transmission Control Protocol and Internet Protocol are communication protocols that define how data should travel across the web.
- **DNS**: Domain Name Servers are like an address book for websites.
- **HTTP:** Hypertext Transfer Protocol is an application protocol that defines a language for clients and servers to speak to each other.
- **Component files**: A website is made up of many different files.
  - **Code files -** HTML, CSS, and JavaScript
  - **Assets -** images, music, video, Word documents, and PDFs

OVERVIEW OF DNS

DOMAIN NAME SERVERS - DNS

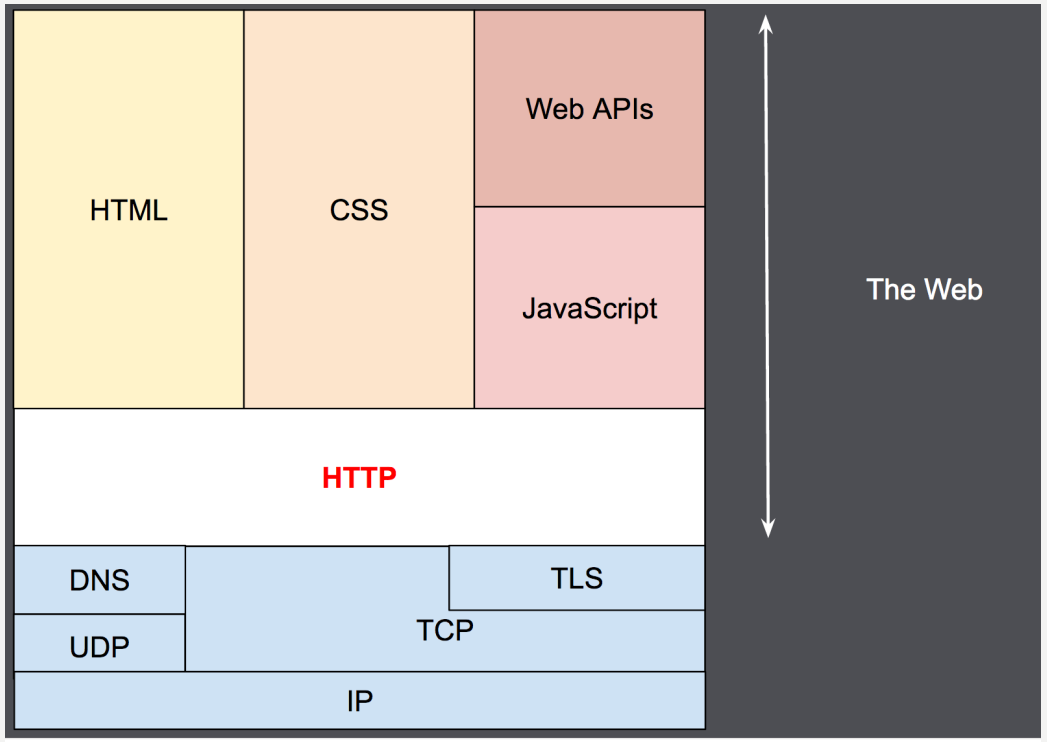


Src: http://www.gargasz.info/how-internet-works-dns/

OVERVIEW OF HTTP

HTTP

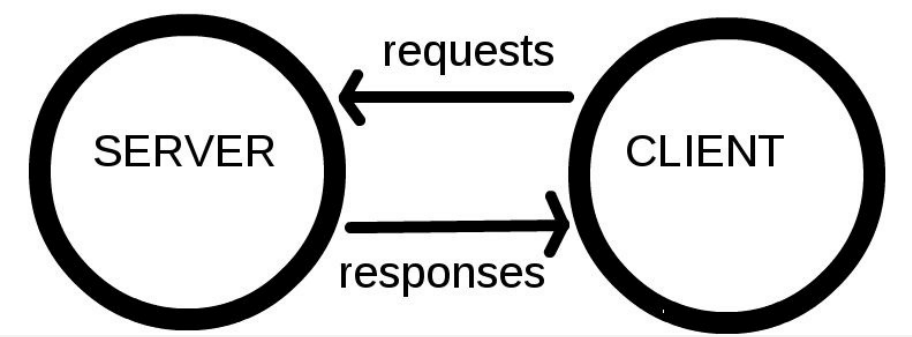
- **HTTP** is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and a client-server protocol.
- It is an application layer protocol that is sent over TCP, or over a TLS-encrypted TCP connection



COMPONENTS OF HTTP-BASED SYSTEMS

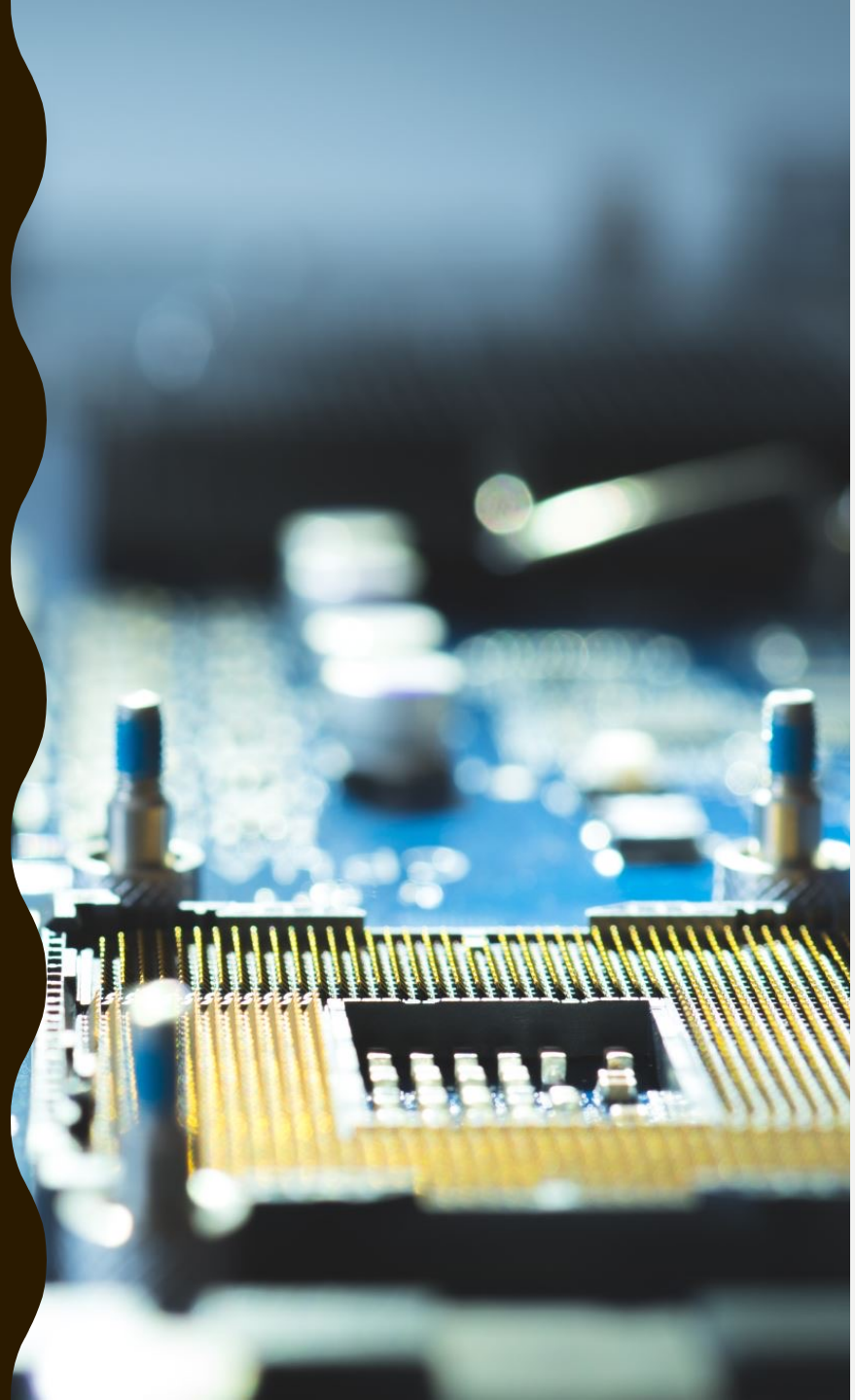
- HTTP is a client-server protocol: **requests** are sent by one entity, the **user-agent** -> a Web browser

- Each individual request is sent to a **server**, which will handle it and provide an answer, called the ***response***.



CLIENT: THE USER-AGENT

- The ***user-agent*** is any tool that acts on the behalf of the user.
- The browser is **always** the entity initiating the **HTTP request**. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages, i.e., web sockets).
- To present a Web page, the browser sends an original **request** to fetch the HTML document from the page.
  - It then parses this file, fetching additional requests corresponding to execution scripts, layout information (CSS) to display, and sub-resources contained within the page (usually images and videos).
  - The Web browser then mixes these resources to present to the user a complete document, the Web page.



THE WEB SERVER

- On the opposite side of the communication channel, is the server which *serves* the document as requested by the client.

- A server presents only as a single machine virtually: this is because it may actually be a collection of servers:
  - Load balancers
  - Database servers
  - E-commerce servers

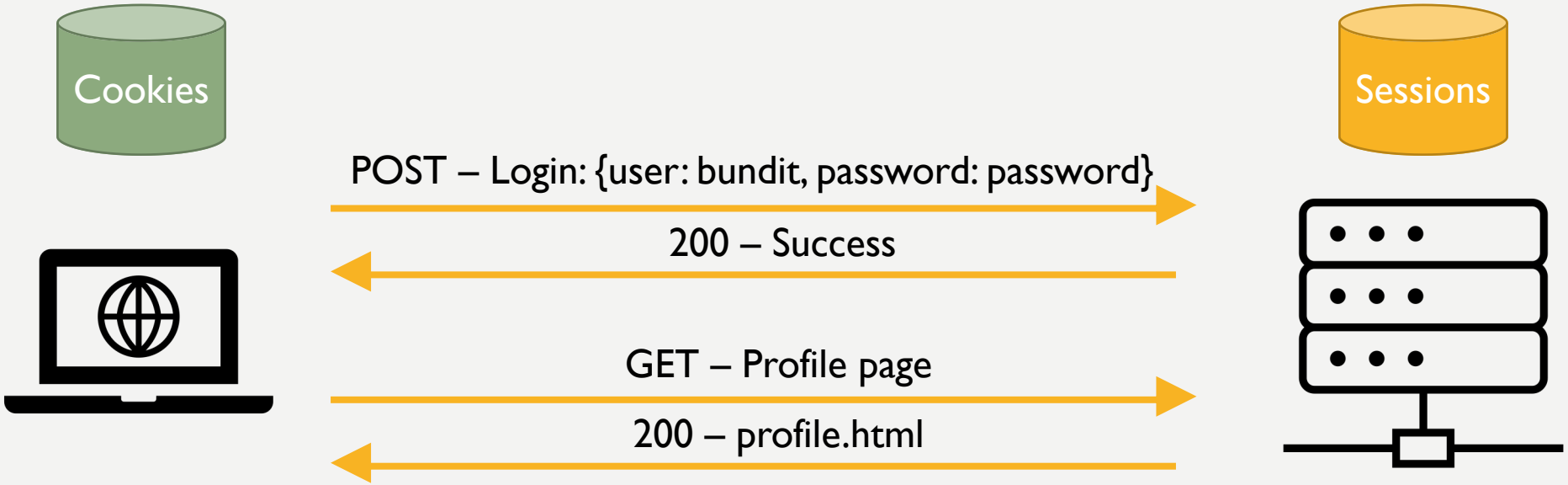
BASIC ASPECTS OF HTTP

- **HTTP is simple** - HTTP messages can be read and understood by humans, providing easier developer testing, and reduced complexity for new-comers.

- **HTTP is stateless, but not sessionless** - HTTP is stateless: there is no link between two requests being successively carried out on the same connection. If so:
  - How shopping basket on e-commerce websites work?
  - How websites maintain logged-in states?

- **HTTP Cookies/Sessions** allow session creation on each HTTP request to share the same context, or the same state.

HTTP IS STATELESS, BUT NOT SESSIONLESS



If HTTP is stateless, how does the server return the correct profile page?

HTTP FLOW

Let's check out
https://www.it.kmitl.ac.th/th/

- When the client wants to communicate with a server, either being the final server or an intermediate proxy, it performs the following steps:
  - Open a TCP connection
  - Send an HTTP request

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

  - Read the response sent by the server

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html
```

  - Close or reuse the connection for further requests.

SERVER-SIDE RENDERING VS CLIENT-SIDE RENDERING

SOLUTELABS

SSR

https://www.it.kmitl.ac.th

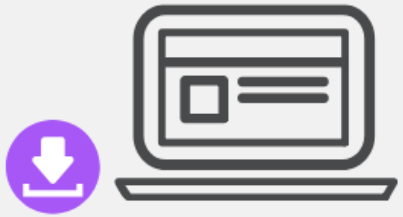
1.

User Requests a Website

2.

Server creates "Ready to Render" HTML files

3.



The Browser can quickly render the HTML but the site isn't interactive

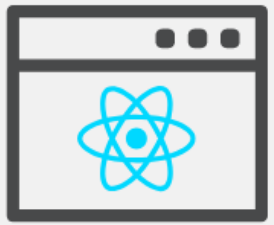
4.

The Browser downloads the Javascript

5.

The user can view content and the interactions can be recorded

6.



The Browser Executes the JS Framework

7.

The recorded interactions can be executed and the page is now interactive

SOLUTELABS

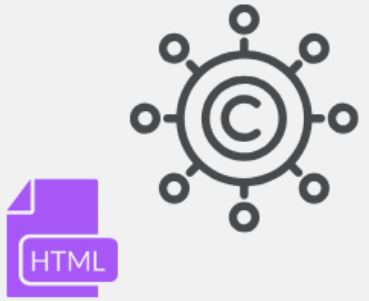
CSR

https://carbon.it.kmitl.ac.th

1.

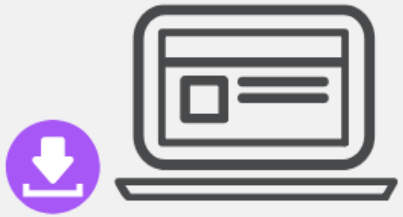
User Requests a Website

2.



a CDN can quickly serve HTML files with links to JS

3.

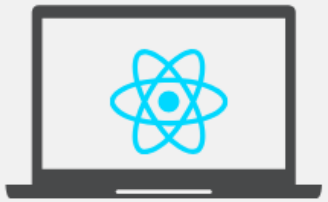


Browser downloads the HTML and then the JS, meanwhile the site isn't visible to the user

4.

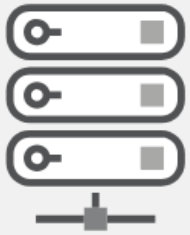
The Browser downloads the Javascripts

5.



The JS is then executed, APIs are called for data, & user sees placeholders

6.



The Server Responds with the data asked by the API

7.

The data from the APIs fill the placeholders and the page is now interactive

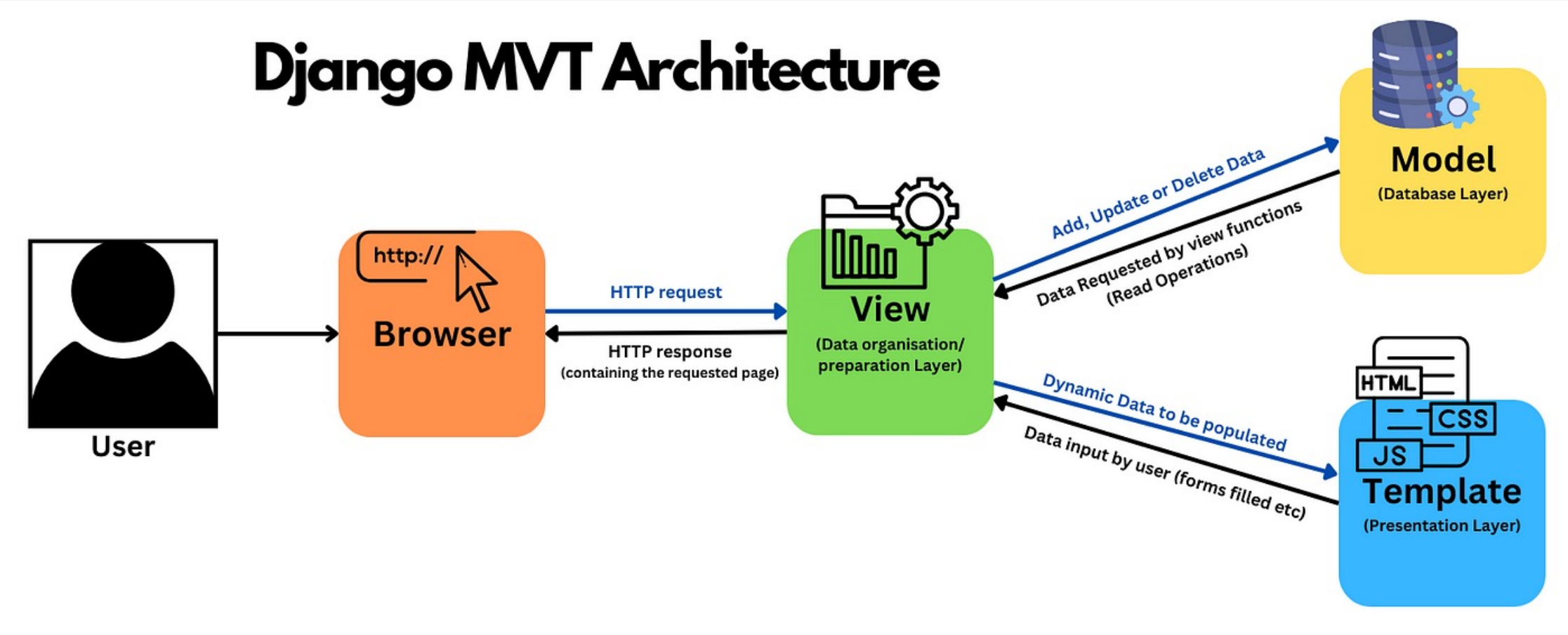
BASICS OF DJANGO: MODEL-VIEW-TEMPLATE (MVT) ARCHITECTURE

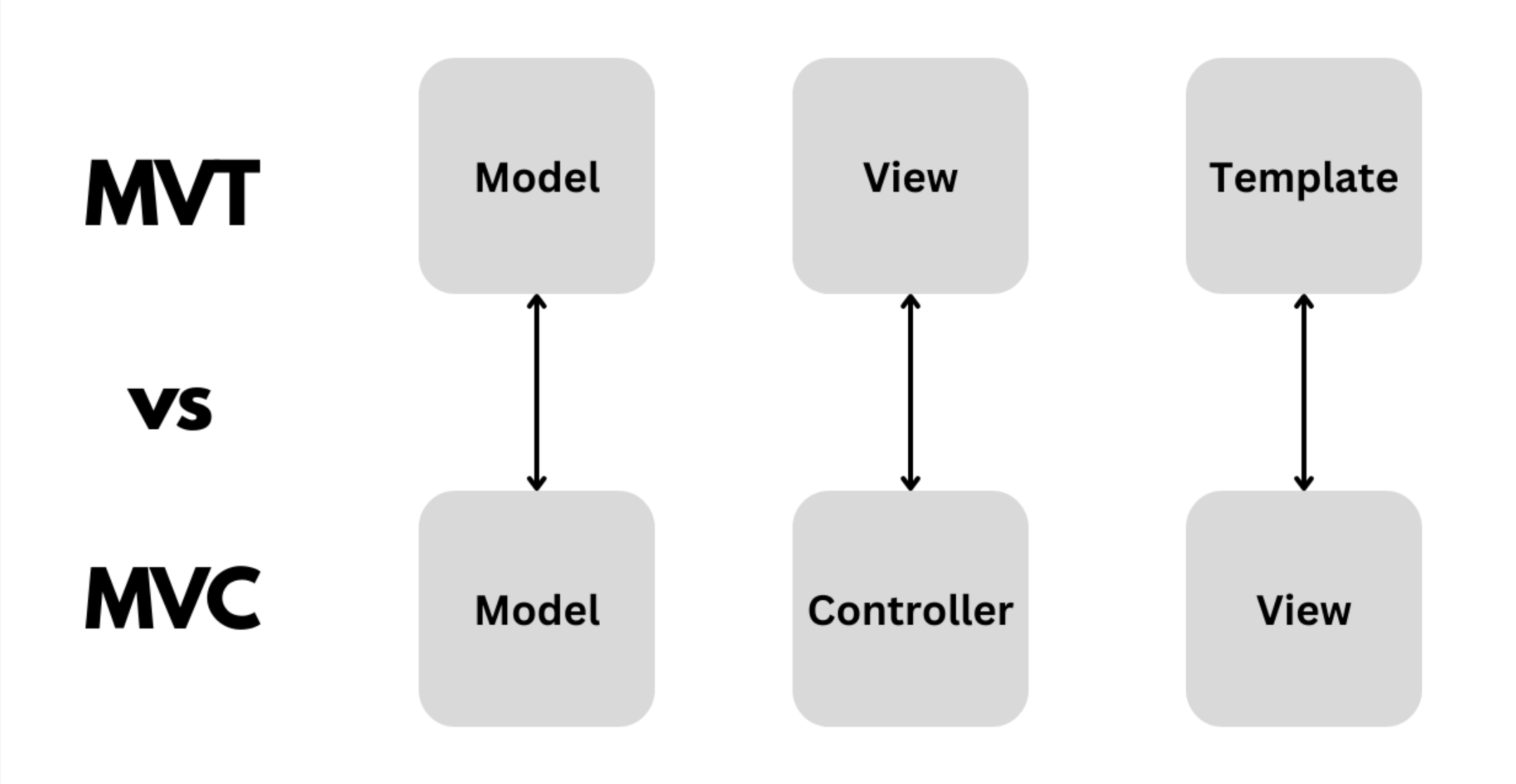
WHAT IS DJANGO?

- **Django** is ‘a high-level Python web framework that encourages rapid development and clean, pragmatic design.
- The framework itself was released as an open-source project in the summer of 2005 and has since become one of the most popular Python web frameworks available today.

MODEL-VIEW-TEMPLATE (MVT) ARCHITECTURE

- **Model-View-Template** architecture, otherwise known as MVT, is a software design pattern within Django that utilizes:
  - Models to handle data logic and structure of your database
  - Views to handle the applications logic and functionality
  - Templates to handle the layout and structure of the user facing application


Django MVT Architecture
User
http://
Browser
HTTP request
HTTP response
(containing the requested page)
View
(Data organisation/
preparation Layer)
Add, Update or Delete Data
Data Requested by view functions
(Read Operations)
Model
(Database Layer)
Dynamic Data to be populated
Data input by user (forms filled etc)
HTML
CSS
JS
Template
(Presentation Layer)


MVT
vs
MVC
Model
View
Template
Model
Controller
View

FRONT-END VS. BACK-END

FRONT-END VS. BACK-END

- When we discuss the "**frontend**" of the web, what we're really talking about is the part of the web that you can see and interact with.
- The frontend usually consists of two parts: the **web design** and **front end web development**.
  - Web designers – those how can work strictly Photoshop and Fireworks
  - Front-end web developers – those who code using **HTML, CSS, JavaScript, jQuery, Vue, React**, etc.

- The **backend** usually consists of three parts: a server, an application, and a database.
- We call a person that builds all of this technology to work together a *backend developer*.
  - Backend technologies usually consist of languages like **PHP, Ruby, Python**, etc.

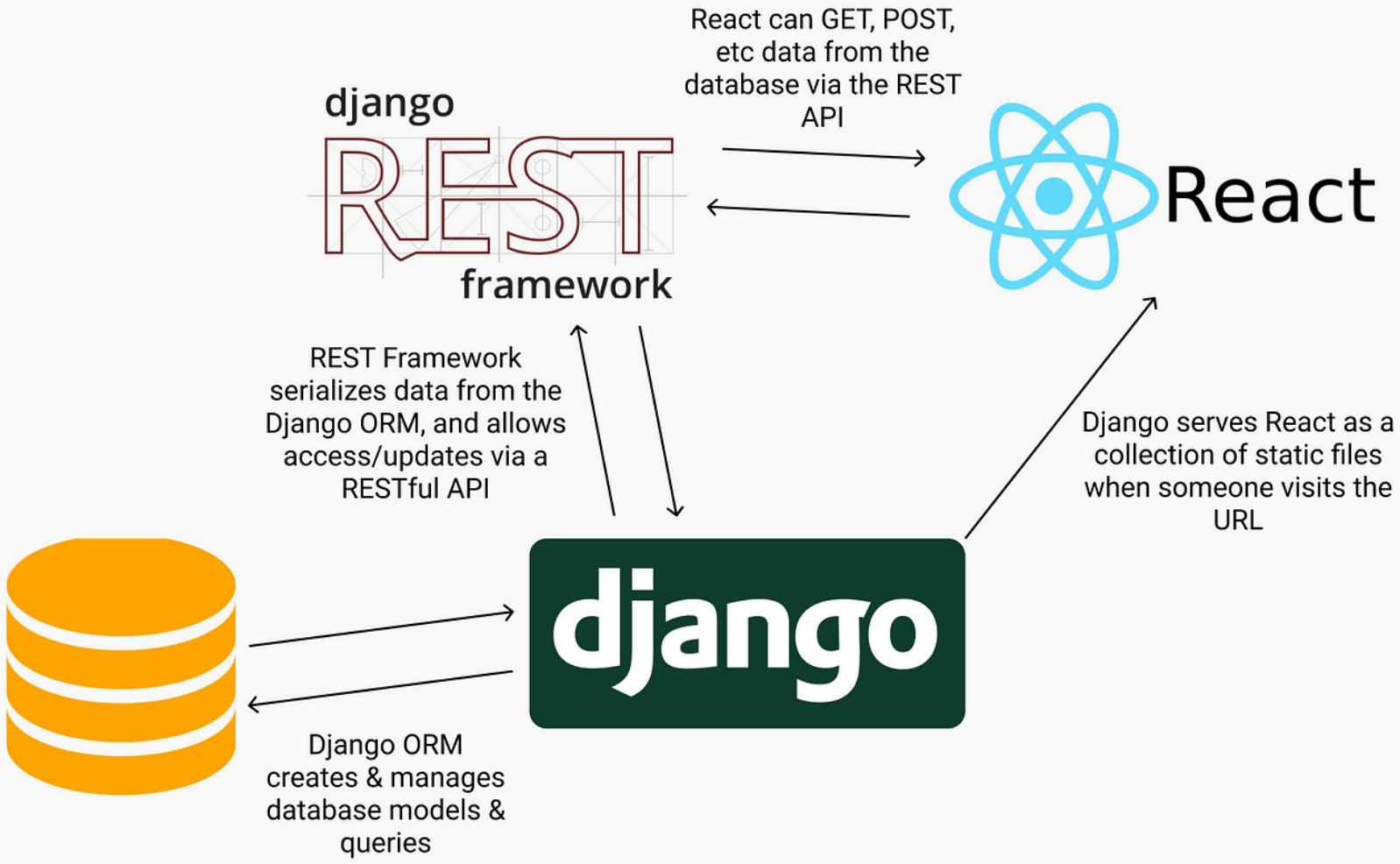
SHOULD YOU BE A BACK-END, FRONT-END OR FULL-STACK DEVELOPER?

- **Key front-end development skills**
  - Have some artistic vision to present the data, UX and UI
  - Mastering HTML, CSS, some CSS pre-processor like SAS, and some (mainstream) JavaScript frameworks such as Angular, React or Vue
  - Have an understanding of event-based interaction, security, and performance.

- **Key back-end development skills**
  - Backend developers work implementing the business logic
  - Have knowledge of backend frameworks, software architecture, design patterns, databases, APIs
  - Be able to manage abstract concepts and complex logic
  - Have a deep understanding of servers and databases (SQL or no SQL), API layer
  - Mastering program languages such as Java, python, PHP, C#, go and scala

COURSE SYLLABUS

- Django installation
- Models
- Making queries (Django ORM)
- Django admin
- Views
- Templates
- Working with forms
- Authentication & authorization
- Django REST framework
- Testing in Django


React can GET, POST, etc data from the database via the REST API
django
REST
framework
React
REST Framework serializes data from the Django ORM, and allows access/updates via a RESTful API
Django serves React as a collection of static files when someone visits the URL
django
Django ORM creates & manages database models & queries

EVALUATION

	WEEK	SCORE
Labs + Quiz	2-14	30
Mid-term exam	8.5	20
Final project	15	30
Final exam	16	20