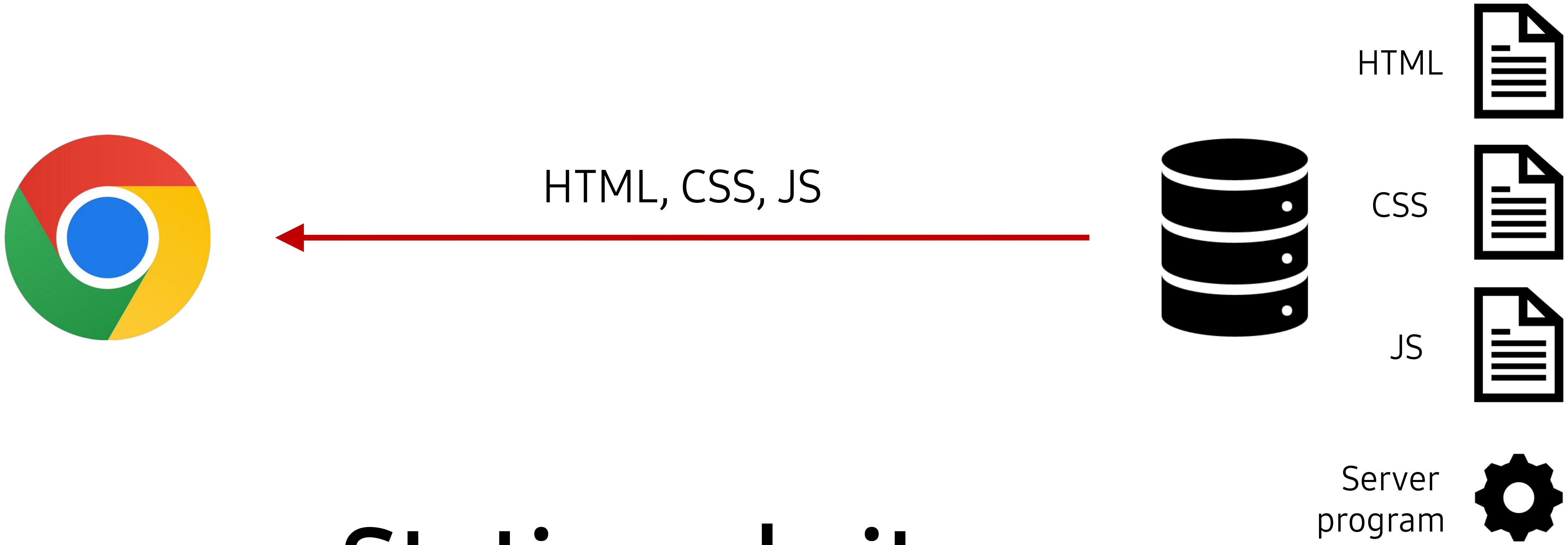
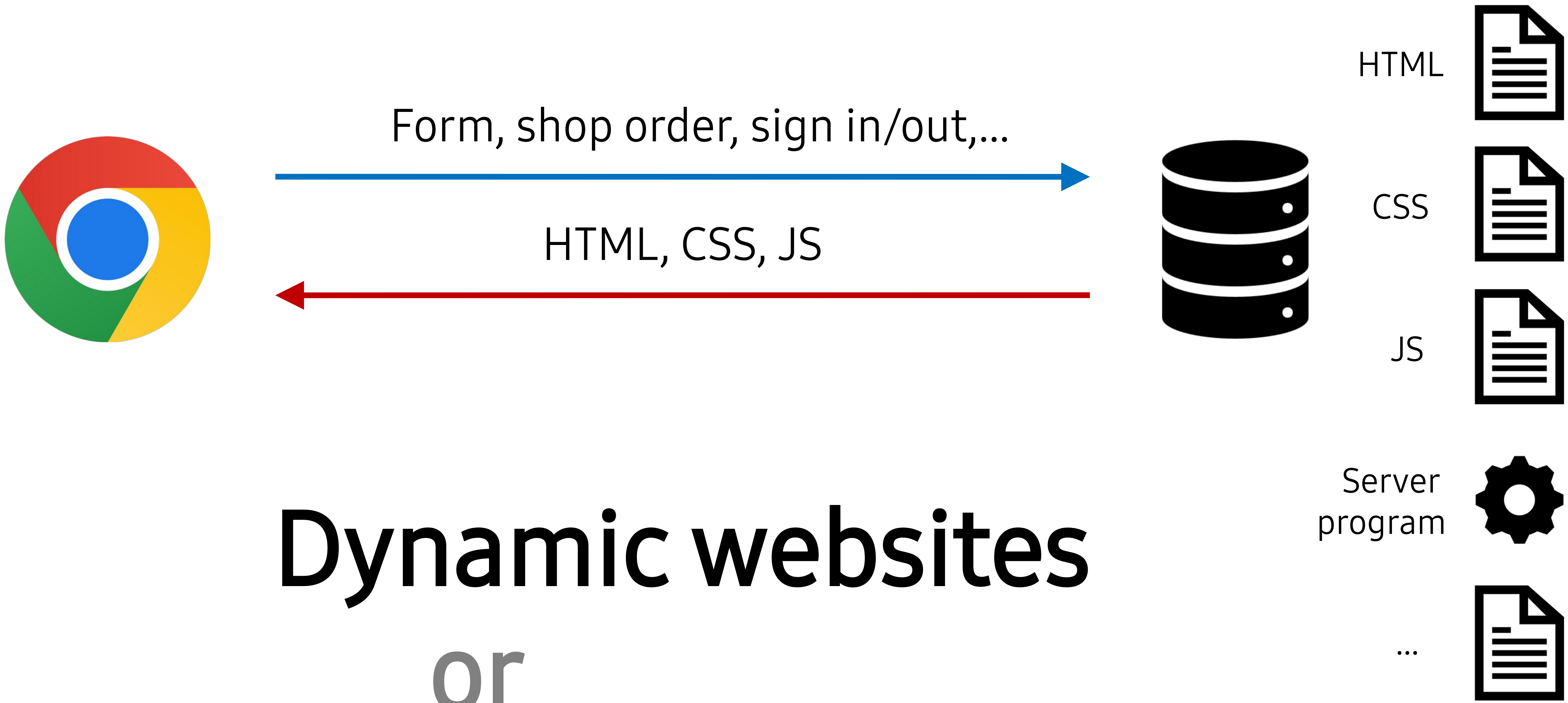
eview



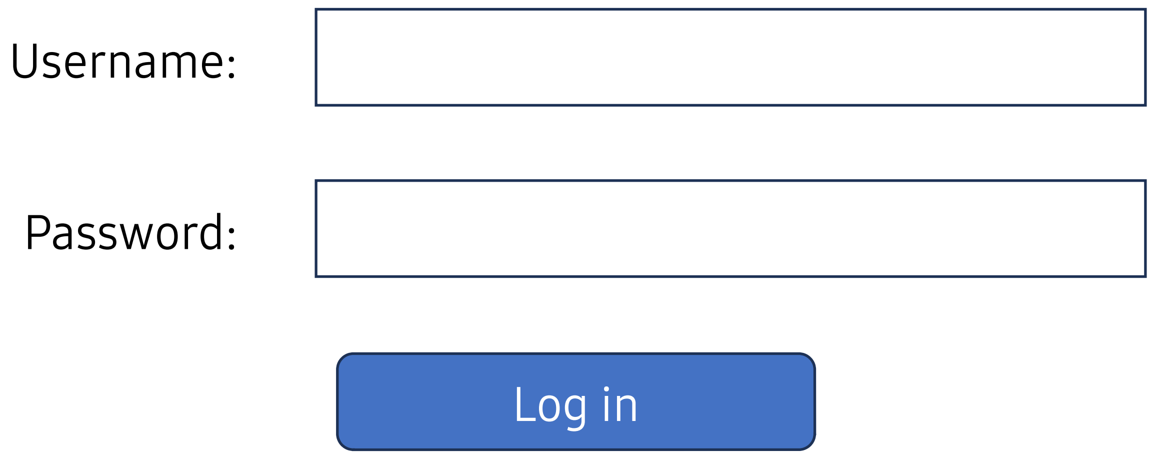
Static websites



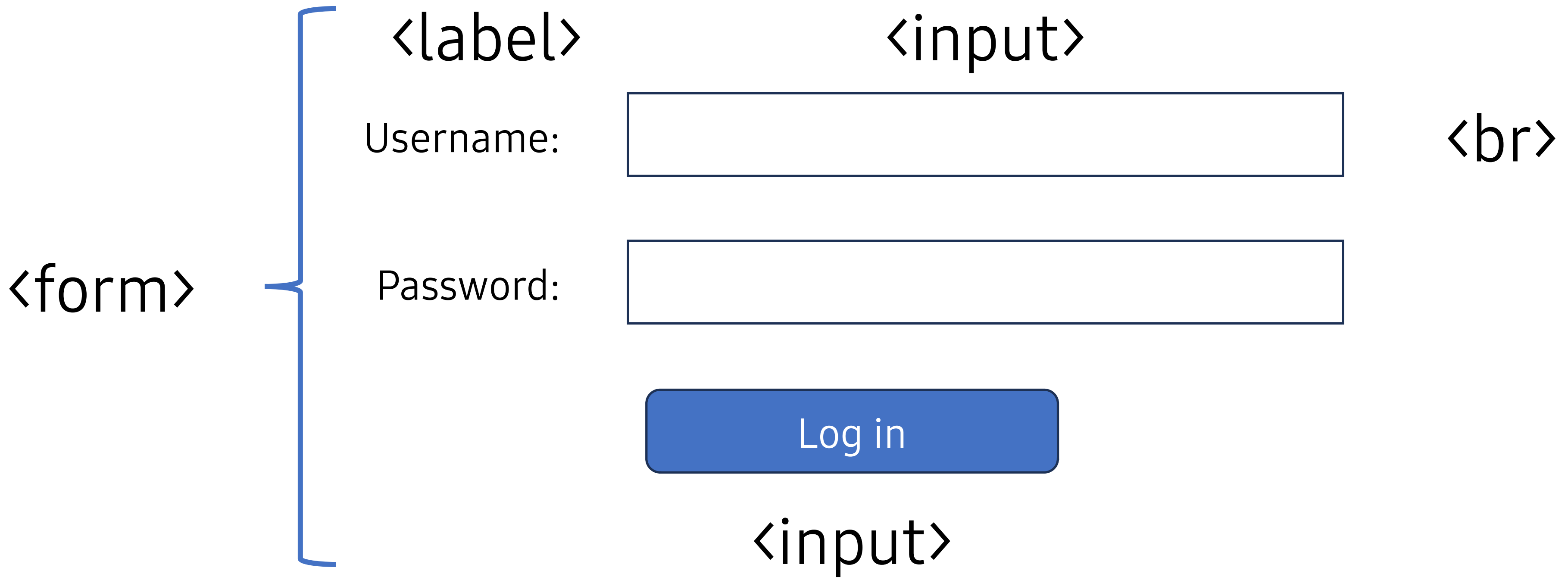
Dynamic websites

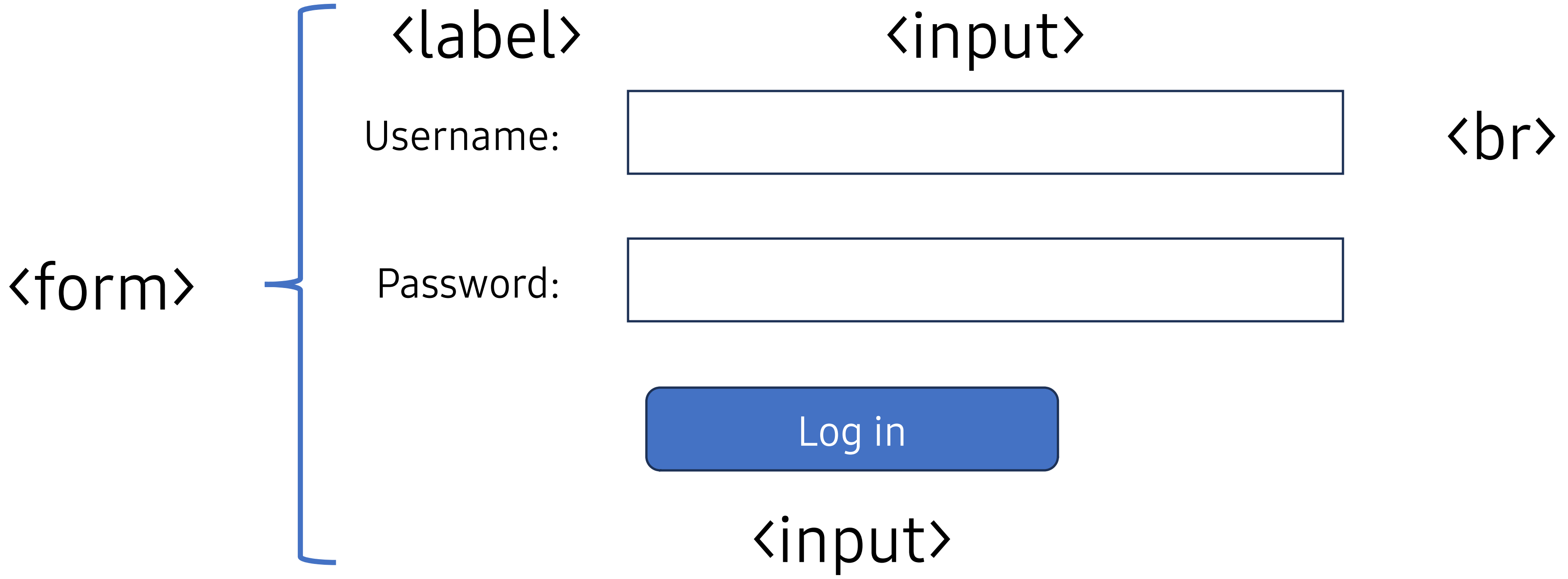
or

HTML Form

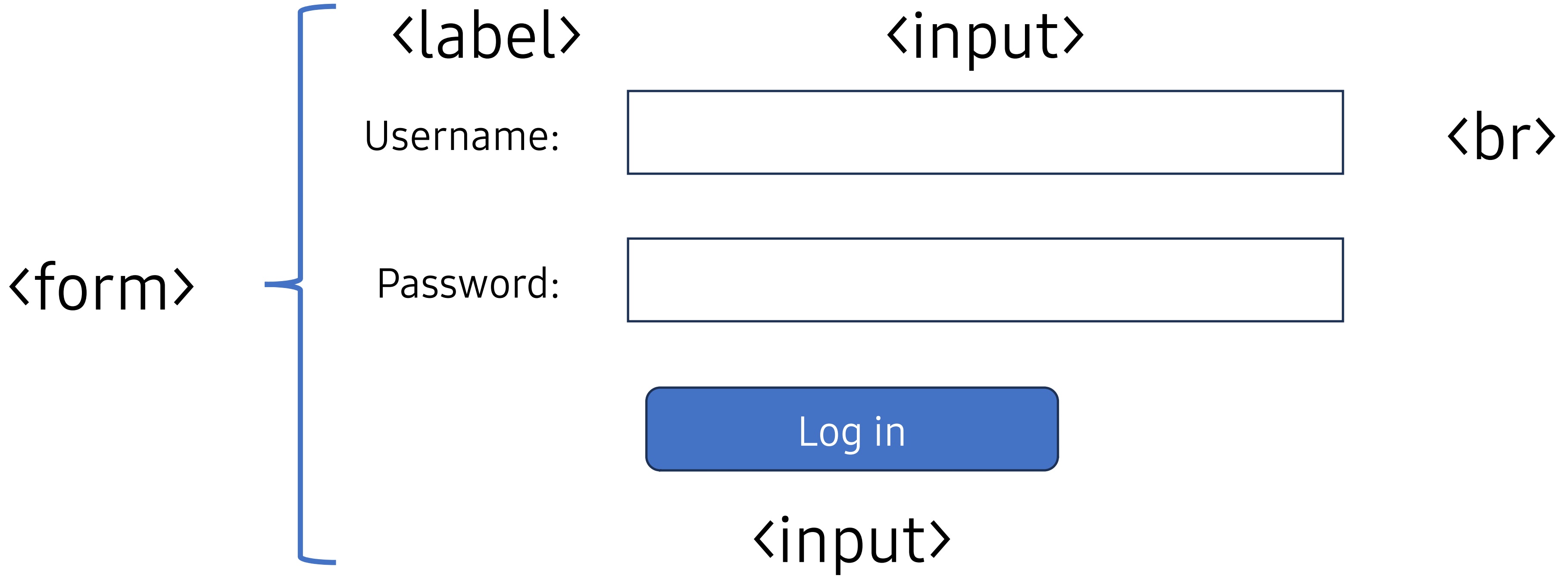


HTML Form

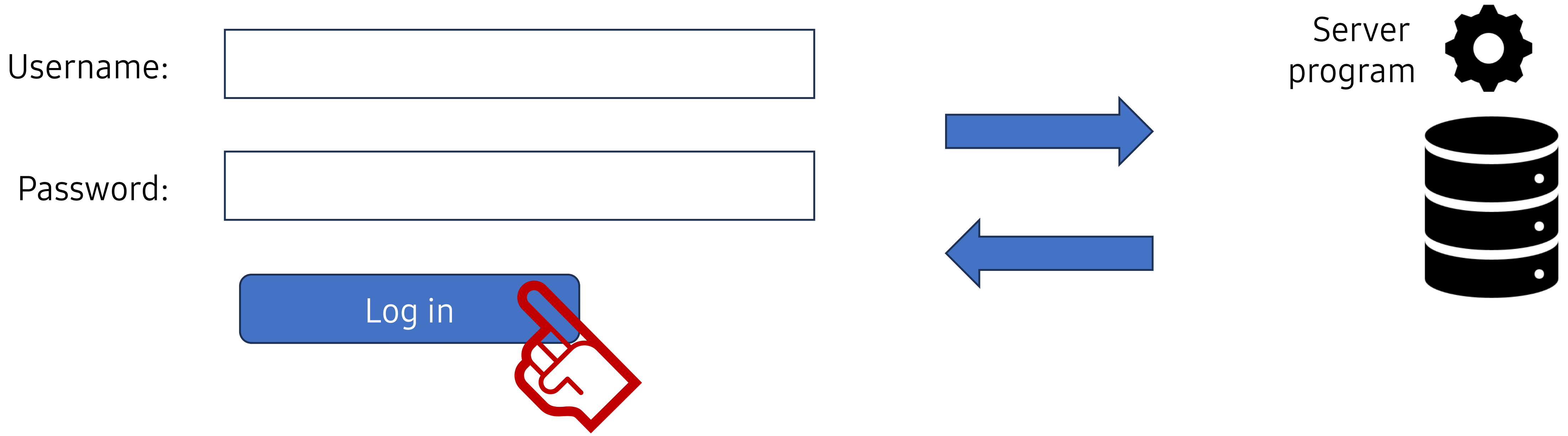




```
<form action="/login" method="post">
        <label>Username: </label>
        <input type="text" name="username"><br>
        <label>Password: </label>
         <input type="password" name="pw"><br>
        <input type="submit" value="Log in">
</form>
```



```
<form action="/login" method="post">
      <label>Username: </label>
      <input type="text" name="username"><br>
      <label>Password: </label>
       <input type="password" name="pw"><br>
      <input type="submit" value="Log in">
</form>
```


Username:
Password:
Log in
Server
program

Open port

Process reqs/resps

Security

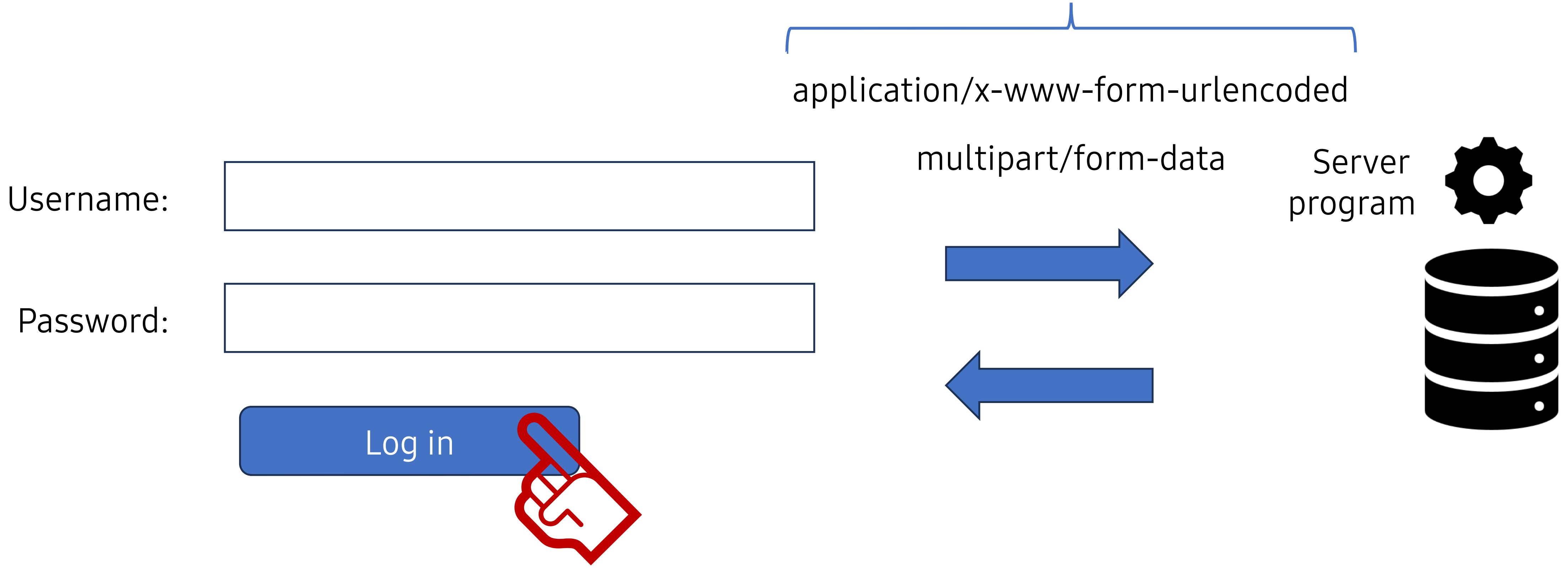
Server program

Support developing process

Standardize

New technologies implemented

Multipurpose Internet Mail Extensions (MIME)



What to focus

- Action URL (endpoint)
- Method (GET/POST/DELETE/PUT)
  - HEAD/OPTIONS/PATCH
- Content
- MIME

RESTful API

RESTful API

#1 Uniform interface: Req/resp must have resource identified.

#2 Statelessness: Each request is independent.

#3 Layered system: If A, B and C are 3 layers where C is the server, a request can be successfully processed from both hidden A and B.

#4 Cacheability: Some contents can be cached.

#5 Code on demand: Server can command the UI when necessary.

https://aws.amazon.com/what-is/restful-api/

Summary

- If there’s a request, there’s a response.
- Request must strictly identify one or some resources.
- Method priority: GET, POST, DELETE, PUT,...
- Processing result included. Ex: code & message.
- For multimedia, take of of the cache.

```
{
   “code”: “SUCCESS”,
   “message”: “”,
   ...
}
```

Java

Java

- First appeared in 1995.
- Simple, robust.
- Architecture independent, portable.
- High performance.
- Interpreted, threaded, dynamic.

Day 3
Java Web Programming

Lecturer: Msc. Minh Tan Le

Step-by-step

I. Java SE & Jakarta EE
II. Java Servlet
III. Software Design Patterns
IV. JSP

I. Java Standard Edition (Java SE)

- java.lang
- java.io
- streams
- java.net
- java.beans
- java.awt
- java.sql

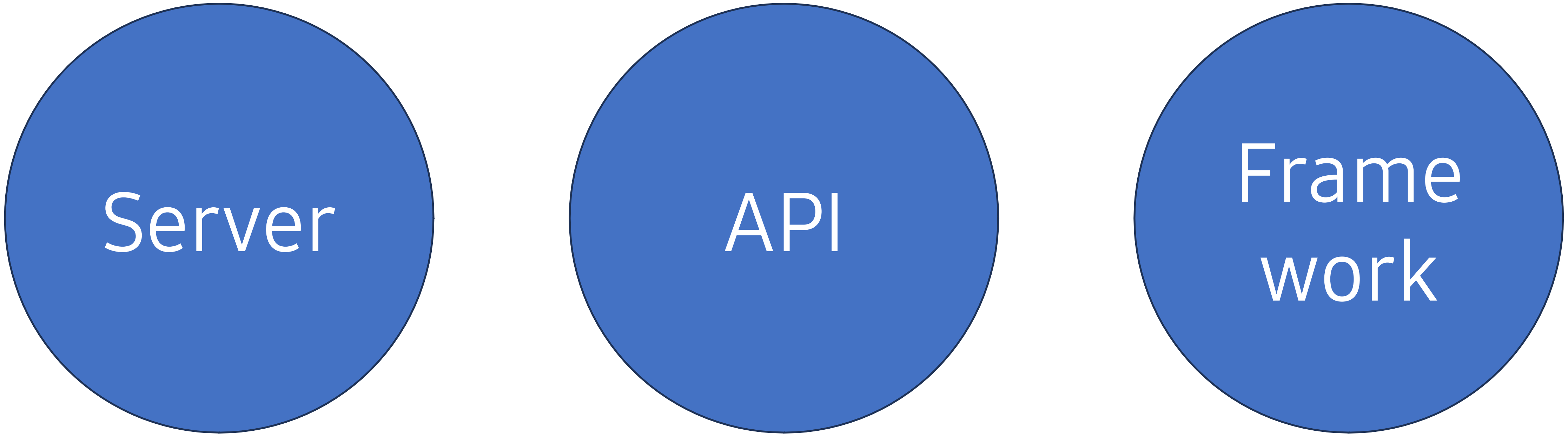
```java
public class Dog {
    String name;
    int age;

    public Dog(String name, int age) {
        this.name = name;
        this.age = age;
    }
}
```

Jakarta EE (Java EE)

- In pure Java, devs might not follow all standards of products.
- 1999, JEE released as a set of technologies for Java.
  - Mainly for web app.

Technologies

- Web
  - Servlet
  - Java Server Page
  - Faces
- Web services
  - Java API
- Extensions
  - Bean


Server
API
Frame work

Server

API

Frame
work

Servlet

Spring

Tomcat

Demo: A servlet “Hello world” web

- Step 1. Create a dynamic web project
- Step 2. Add library (jar)
- Step 3. Write servlet codes
- Step 4. Run Tomcat
- Step 5. Try with form submission

Useful APIs

API	Purpose	How to get it?
javax.servlet.http.HttpServletRequest	Contains request information	*“Do…” method argument.*
javax.servlet.http.HttpServletResponse	Contains response information	*“Do…” method argument.*
javax.servlet.ServletContext	Get the context.	ServletContext ctx = getServletContext();
javax.servlet.RequestDispatcher	Get the request dispatcher.	ctx.getRequestDispatcher("/myurl");
javax.servlet.GenericServlet.getServletConfig	Get application config.	

Architect that all hate

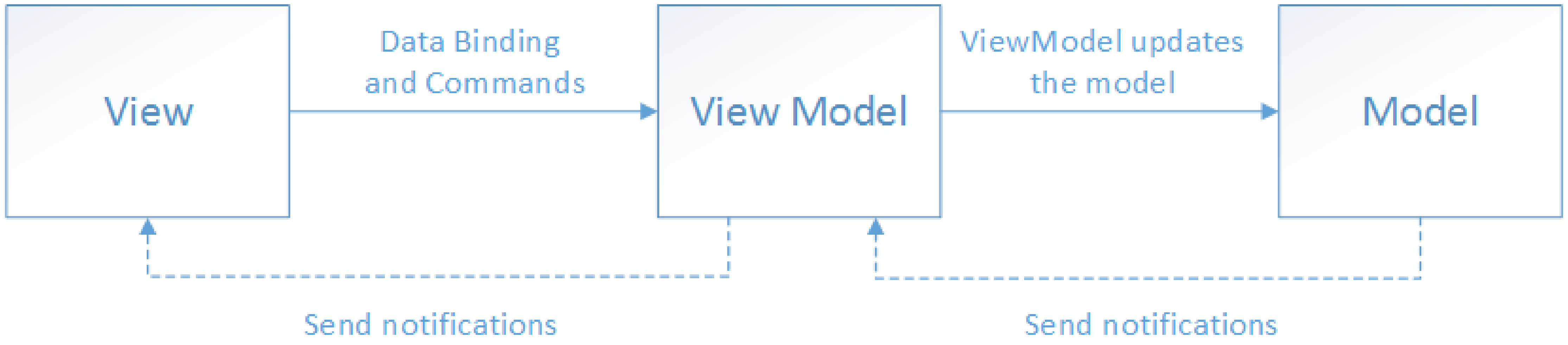
- Put HTML, CSS, JS into strings & return.
- Problems:
  - Multi languages in one source file 👩‍💻
  - Not efficient with big projects 💥

Web forms

- Everything C#
- WYSIWYG
- Very similar to .NET desktop app (WinForms):
  - Forms
  - Controls

Model – View – ViewModel
MVVM

MVVM



Defining UI

Binding data

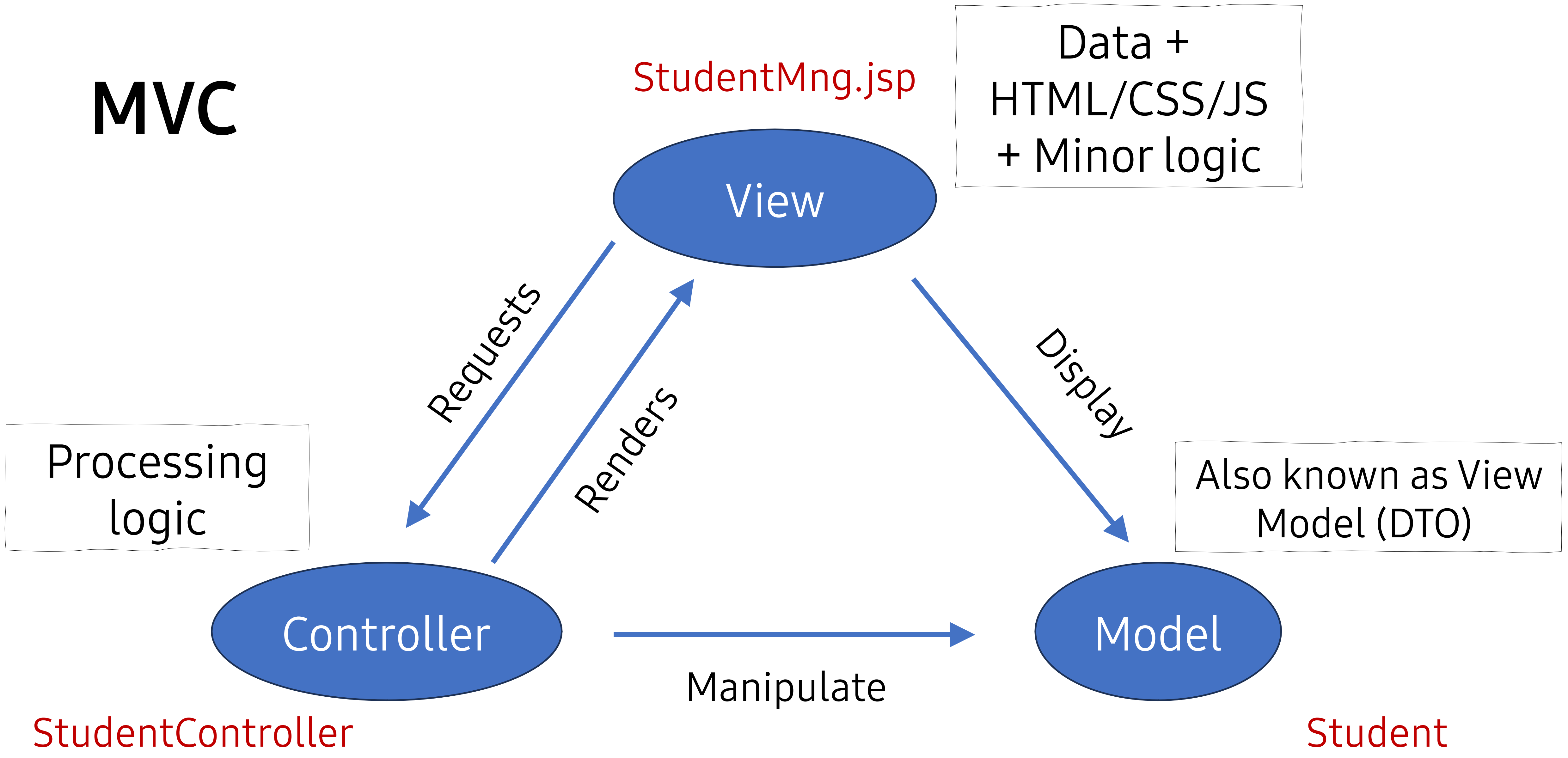
Providing data

MVVM

- What’s so great?
  - Separated front-end & back-end codes
  - Separated UI logic (no JS knowledge required)
- What’s not so great?
  - Heavy (may need event loops)
  - Not common
  - Not language consistent (according to MS)

Web forms
MVC

MVC


StudentMng.jsp
Data + HTML/CSS/JS + Minor logic
View
Requests
Renders
Display
Processing logic
Also known as View Model (DTO)
Controller
Manipulate
Model
StudentController
Student

Our servlet cannot separate view

Solution: JSP (Java Server Pages)

No	Element	Description	Example
1	Front-end codes	Codes that run on client browser	
2	Expressions	Expression that return printable value.	<%= result %> ${...}
3	Scriplets	Code fragments, multi-line supported. Injected variables: Request, Response, Session, Out	<% script %>
4	Directives	A JSP description. <%@ page/include/taglib attr="..." %>	<%@ page import="java.util.Date" %>
5	Declarations	Defining functions & variables.	<%! ... %>

Demo

Step 1: Create a JSP file under webapp.

Step 2: Try using MVC pattern by letting servlet file call JSP to handle.

Step 3: Send some data from Controller to View.