

Yi Li 1904719

by Yi Li

FILE	17851_YI_LI_YI_LI_1904719_242847_599851821.DOCX (4.27M)		
TIME SUBMITTED	17-APR-2020 08:10PM (UTC+0100)	WORD COUNT	3304
SUBMISSION ID	124370197	CHARACTER COUNT	17728

0Course: Web Development

Instructor: Mike Dacey

Student: Yi Li

Student ID: 1904719

Contents

<i>Server-side code and MVC.....</i>	<i>2</i>
Server-side code	2
Advantages of server-side code	2
MVC	3
The MVC definition	3
MVC Advantages	3
<i>JavaScript, client-side and server-side</i>	<i>4</i>
Scripting language	4
What is JavaScript.....	5
Client-side and Server-side	6
<i>node.js.....</i>	<i>8</i>
Ten frameworks of Node.js	9
The reason why we should use node.js	10
Advantages and disadvantages of using node	11
<i>AJAX</i>	<i>11</i>

Server-side code and MVC

Server-side code

Web Server, as the name suggests, is a server that provides Web services. For example, when we visit <http://google.com>, we actually use the services provided by Google's servers.

Generally speaking, there are two explanations for server: the first kind of server means hardware, which can also be called host, but more often, server means software program, this kind of program is mainly used to provide certain services, such as mail service, FTP Services, database services, web services, etc. As a developer, when referring to server, it generally refers to the latter, that is, a software program that can run 24 hours, additionally, on a host, you can run several programs at the same time. Server-side code, as the name implies, is the code that acts on the server side. It enables the server to run normally and operates the server so that the server can complete tasks such as POST, GET, PUT, and DELETE.

In this design, the overall interaction figure of the server-side and the client-side is as follows.

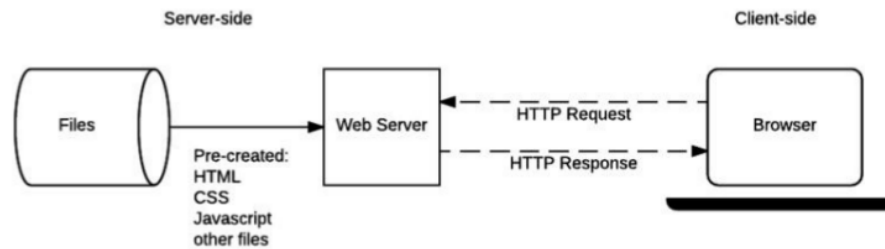


Fig 1

It is not difficult to see from the figure that the browser and the web server communicate through HTTP Request and HTTP Response. In this design, I chose the "Postman" recommended by the instructor for command operation, which is very convenient. As for the server-side code, it is stored in the local root directory as follows:

`'/Users/ MyWebServer/ NodeWebServer/ serverside/ts/server.ts'`.

'server.ts' is created and modified using JQuery templates, thus maintaining a unified standard for writing.

Advantages of server-side code

In this design, the server relies on these codes to operate and exchange data normally. The Typescript file of 'server.ts' has specific functions in the following aspects:

(1) Efficiently store and transfer information

Server-side programming allows us to store information in a database and dynamically construct and return HTML and other types of files.

(2) Customized user experience

The server can store and use information about the client to provide a convenient and customized user experience.

(3) Controlled access content

Server-side programming allows sites to restrict access to authorized users and only provide information that users are allowed to view.

MVC

The MVC definition

MVC mode is a software architecture mode in software engineering. In this mode, the software system is divided into three basic parts, model, view, and controller.

After reading some literatures, I make a picture of the concept of MVC as follows:

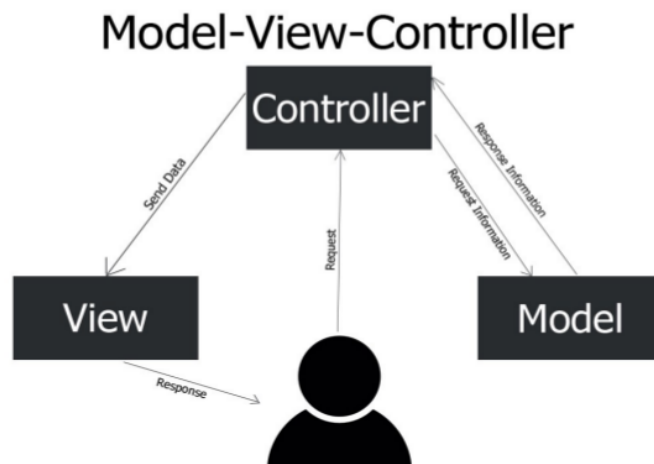


Fig 2

MVC Advantages

There are advantages and disadvantages to using this design pattern, but overall, for large-scale Web development, the advantages outweigh the disadvantages. The advantages of using the MVC pattern are as follows:

(1) Low coupling.

Multiple views can be created and used simultaneously for a model at runtime. The change-propagation mechanism can ensure that all related views get model data changes in time, so that all related views and controllers synchronize their behaviors.

(2) High applicability.

The pluggability of the view and the controller allows the view and controller objects to be replaced, and can be dynamically opened or closed according to the needs, and even replaced during operation.

(3) Model portability.

Because the model is independent of the view, a model can be independently ported to the new platform to work. All that needs to be done is to make new modifications to the views and controllers on the new platform.

(4) Potential frame structure.

You can build an application framework based on this model, not just in the design of the design interface.

MVC also has some shortcomings, as follows:

(1) Increase the complexity of system structure and implementation. For simple interfaces, strictly following MVC, and separating models, views, and controllers will increase the complexity of the structure, and may generate too many update operations, reducing operating efficiency.  2

(2) Too close connection between view and controller. Views and controllers are separated from each other, but they are closely related components. Views have no controllers, and their applications are limited, and vice versa, which prevents their independent reuse.

(3) Inefficient access to model data by views. Depending on the operation interface of the model, the view may require multiple calls to obtain sufficient display data. Unnecessary and frequent access to unchanged data will also impair operational performance. 

(4) Currently, generally advanced interface tools or constructors do not support modes. The cost of retrofitting these tools to meet MVC needs and establishing separate components is very high, which makes MVC difficult to use.  4

JavaScript, client-side and server-side

Scripting language

(1) Script is also a language, which is also composed of program code. There are many similarities between scripting languages and programming languages. Scripting languages functions are similar to programming languages, and they also involve variables. The biggest difference between them is that the syntax and rules of programming languages are stricter and more complicated.

(2) The program is independent of the data you want to process: the program has its own directory and can process different data in different locations.

The script is attached to the data it processes: the script is thrown in the directory where the data is located, and the script will directly mention specific files and specific directories in the data, and it will not run without this environment. In addition, the script has another feature, that is, it has only two links: writing and running.  5

(3) The program usually has three links: write, build, and run. And the executable file generated by the build link is not located in the place where the data to be processed is located. It is difficult to publish with the information. The build process will also generate a lot of intermediate files, which will stain the directory where the data is located. In short, the characteristics of the script are: write on the spot, run on the spot.  6

What is JavaScript

After reading a lot of information, I have a more comprehensive understanding of JavaScript and summarized the following points:

- (1) When it was created, the purpose was functional, but the company wanted him to make it more like Java. Of course, but in the end, this language is actually more like C. It is a C-style functional scripting language, except that there is a Java in the name.
- (2) It took developers only ten days to complete the language, and then the Internet spent more than ten years developing object-oriented, namespace, AJAX, Pattern, etc., and there are various book specifications that introduce how to use the language.
- (3) In the first few years, this language has been used as a scripting language. With the development of the Internet, people's demand for dynamic content has increased, so AJAX technology was invented.
- (4) Then a company called Google made an email client with this scripting language, and people had to start to reexamine this "scripting" language.
- (5) Immediately, Google found that HTML and JavaScript were the only platforms it could control, so it invested more in it.
- (6) Google tried its best to finally improve the performance of this scripting language to the extreme, that is, the V8 engine.
- (7) JavaScript compatible language JScript is one of the scripting languages supported by WSH (Windows Script Host), so it can be used for ASP development, but in fact

strictly speaking, JScript is used for ASP (this language syntax is compatible with JavaScript)

Client-side and Server-side

In the local folder, my clientside folder and serverside folder are stored in this location:

‘Users / WebServer / NodeWebServer / MyWebServer / NodeWebServer’

When comparing and contrasting client-side and server-sidescripting, we mainly study ‘server.ts’ in the server folder and ‘productmanagement.ts’ in the ‘clientside’ folder.

In ‘server.ts’, we can see that there are some dependency packages, as shown below.

```
1 import {
2   WebServer,
3   UploadDetails
4 } from './webserver';
5
6 import {
7   Router
8 } from './router';
9
10 import {
11   KeyValuePairs
12 } from './keyvaluepairs';
13
14 import * as process from 'process';
15
16
17 import {
18   ServerResponse
19 } from 'http';
20
21 import {
22   inspect
23 } from 'util';
```

Fig 3

It is the support of these support packages that supports the operating foundation of the entire server.

In the next class ‘WebPageProcess’, four basic web page functions are introduced, namely GET, POST, PUT, and DELETE, and the corresponding functions are read, add, edit, and delete. Let's take an example for the functions of the GET part.

```

44 router.AddRoute('/products/add', 'POST',
45 (url: string, response: ServerResponse,
46 data: KeyValuePairs):boolean => {
47
48     let productIndex: number;
49     let result: { done: boolean, productid: number };
50
51     response.writeHead(200, { 'content-type': 'application/json' });
52
53     this._products.push(
54     {
55         name: data.name,
56         description: data.description,
57         price: data.price
58     });
59     productIndex = this._products.length - 1; // Added to end of array
60
61     result = { done: true, productid: productIndex };
62     response.write(JSON.stringify(result));
63     return true;
64 });

```

Fig 4

The 'router.addRoute' function configures the basic method of using this function. For example, this part is to create a new product, so use 'products / add', string in 'url', response: 'ServerResponse' determines the input format and The response mode of the server.

'Data: KeyValuePairs): boolean =>' contains some callback methods. The following 'this._products.push' method explains the specific steps and format for adding products.

In the next three parts of the code, the basic framework is the same as this, so I won't go into details.

The 'productmanagement.ts' in the clientside folder mainly lists the functions of several buttons, such as 'LoadAllProductClick', 'AddProductClick', 'LoadProductClick', 'EditProductClick', 'DeleteProductClick'.

Compared with the function format in 'server.ts', the function writing format here is more clear.

```

22 private LoadAllProductClick = (e: JQuery.Event): boolean =>
23 {
24     $.ajax({
25         cache: false,
26         type: "GET",
27         url: "http://localhost:1339/products/read",
28         data: '{}',
29         contentType: "application/json",
30         dataType: "json",
31         success: this.ProductReadAll,
32         error: this.AjaxError
33     });
34
35     e.preventDefault();
36     return false;
37 };

```

Fig 5

The results are as follows:

Name

Description

Price null

Name product 1

Description This is an iPhone

Price 4999

Name product 2

Description This is an iPad

Price 3999

Name product 3

Description This is an iWatch

Price 2999

Read All Products

Delete Product 0

Name: product 3

Description: This is an iWatch

Price: 2999

Add Product

Product added at index 3

```
NodeWebServer -- node - npm TMPDIR=/var/folders/2c/czfsp84/0bv_lwv_4n02rmbh0000gn/T/...
Route body arg price=3999
Search route /products==/products/add method get==post
Search route /products/add==/products/add method post==post
Found route /products/add
Args are: name: product 2; description: This is an iPad; price: 3999;
Incoming Req: /products/add
handling post: application/json
JSON received = name: product 3; description: This is an iWatch; price: 2999;
Route body arg name=product 3
Route body arg description=This is an iWatch
Route body arg price=2999
Search route /products==/products/add method get==post
Search route /products/add==/products/add method post==post
Found route /products/add
Args are: name: product 3; description: This is an iWatch; price: 2999;
Incoming Req: /products/read?{}&_=1587141183472
Route Url arg {}=
Route Url arg _=1587141183472
Search route /products==/products/read method get==get
Search route /products/add==/products/read method post==get
No match...
Search route /products/read==/products/read method get==get
Found route /products/read
Args are: {}; ; _: 1587141183472;
```

Fig 6

With the support of back-end server-side code, they are not only buttons with shapes, but also have corresponding functionality.

node.js

Node.js is a development platform that allows JavaScript to run on the server. For programmers who only use JavaScript on the client, server-side JavaScript may be a new concept, but the concept itself is not out of reach, so why not use the programming language used on the client on the server? Therefore, Node.js, which uses JavaScript for programming on the server side, is naturally led out.

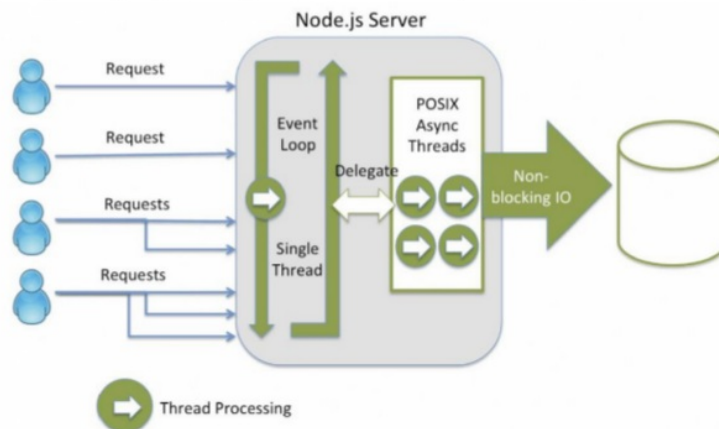


Fig 7

Ten frameworks of Node.js

The Node.js system contains many different structures, such as MVC, full stack, REST API, and generator. These structures not only improve the development efficiency of Web applications, but also optimize the development process. After reading some literature, I sorted out the ten major frameworks of Node.js mainstream.

(1) Node.js development framework Sails.js

Sails.js is like a Rails framework on the Node.js platform. This is a reliable and scalable development framework, a service-oriented architecture that provides a collection of data-driven APIs. It is very convenient for developing multi-player games, chat applications and real-time panel references, and can also be used to develop enterprise-level Node.js applications.

Sails.js is built on Node.js, Connect, Express and Socket.io.

(2) Node.js server-side framework Hapi.js

Hapi.js is a rich framework for building Node.js-based applications and services, allowing developers to focus on portable and reusable application logic rather than building architecture. Built-in input verification, caching, authentication, and other commonly used functions for Web application development.

(3) Node.js high-performance packaging Express.js

Express is a simple and flexible Node.js Web application framework, providing a series of powerful features to help you create a variety of Web applications. Express does not make a second abstraction of the existing features of Node.js, we just extend the functions required by Web applications on top of it. Rich HTTP tools and middleware from the Connect framework are available on demand, creating a robust and friendly API becomes quick and easy.

(4) Node.js web framework Kraken.js

Kraken is built on express to realize the awareness of environment variables, dynamic configuration, advanced middleware and event notification of application life cycle.

(5) Web application construction platform Meteor

Meteor is a new set of technologies for building high-quality Web applications, providing many ready-made packages that can be run directly in a browser or cloud platform.

(6) Full-stack JavaScript development framework Mean.js

A full-stack JavaScript development architecture, its name comes from: MongoDB, Express, Angular, Node.

(7) Node.js web framework Koa.js

Koa is the next generation Web framework for Node.js. Designed by the Express team. Designed to provide a smaller, more expressive, and more reliable basis for developing Web applications and APIs.

Koa can get rid of callbacks through generators and greatly improve error handling. Koa core does not bind any middleware, but provides an elegant set of methods for writing server applications quickly and happily.

(8) Node.js CMS and Web application platform KeystoneJS

KeystoneJS, a Node.js CMS and Web application platform built on Express and MongoDB.

(9) Node.js framework component flatiron.js

flatiron is a framework component of Node.js and browser, and a highly adaptable framework for building modern web applications. Flatiron provides framework components with richer configurations than Rails-like components, allowing developers to add the functional components they want.

(10) API Framework LoopBack based on Node.js

LoopBack is an open source API framework based on Node.js, which allows Node.js applications to be easily connected with various devices through the API.

The reason why we should use node.js

From a traditional point of view, when we use a programming language such as Java or PHP for server development, each request generates a new thread. We assume that a server has a total of 5GB of memory, and each thread occupies 1MB of memory. In theory, the maximum number of concurrent connections for this server is 5000 users. If the web application we develop welcomes more users (greater than 5000), if we want to run the web application normally, then we must increase the number of servers. In this way, it will increase our operating costs, and when a certain When the data that a user needs to access is stored on different servers, the data needs to be retrieved from different servers, which increases the technical cost and the chance of errors.

Next, I will give an example based on actual experience.

For example, we are using our own account to browse Facebook. At this time, we are using the client. The content displayed on the screen is the data sent by the server to the client in the past period of time. At this time, the server is not idle, it is still sending data like the client, but the data is not received by the client. When we have browsed the latest content and want to see the latest news, we click refresh, and the random client will send a received command to the server. Then in the past period, the server sent the data to the client, At this moment, it will be received by the client and displayed on the screen. This is non-blocking and event-driven.

Advantages and disadvantages of using node

The biggest advantage of Node.js is that with the help of JavaScript's natural event-driven mechanism and V8 high-performance engine, writing high-performance Web services is a breeze.

Secondly, the JavaScript language itself is a perfect functional language. In front-end development, developers often write more casually, making people feel that JavaScript is a "toy language." However, in the Node environment, through the modular JavaScript code, plus functional programming, and without having to consider browser compatibility issues, directly using the latest ECMAScript 6 standard can fully meet the needs of the project.

At the same time, using Node.js also has some unavoidable disadvantages, as follows:

- (1) For applications with heavy CPU usage and light IO usage-such as video coding and artificial intelligence, the advantages of Node.js cannot be brought into play.
- (2) Simple web applications-The characteristics of such applications are low traffic and simple physical architecture. Node.js cannot provide a powerful framework like Ruby's Rails or Python's Django.
- (3) NoSQL + Node.js-If you are just for the pursuit of fashion, and you have not deeply understood these two technologies, don't risk building business systems on these two beautiful terms, it is recommended to use MySQL Traditional databases like that.

AJAX

After using AJAX for a while, I have some insights on this technology.

The full name of ajax is Asynchronous JavaScript + XML.

The so-called asynchronous is simply explained here: when sending a request to the server, we don't have to wait for the result, but can do other things at the same time. When the result is available, we can deal with it. (Of course, this explanation may not be correct in other contexts)

This is very important. If this is not the case, we will click the button and the page will die there, and other data requests will not go down. This seems more annoying than waiting for a refresh.

(Although components that provide asynchronous communication functions are asynchronous by default, they also provide synchronous options. If you are curious to change that option to false, your page will die there.)

XML is just a data format, which is not important in this matter. We theoretically do not need this format when updating a line of text, but if we update a lot of content, then the formatted data can make us go in an orderly way. Implement updates.

Most people now use JSON instead of XML, because the former is more concise and it is said that the current parsing speed is also faster.

To put it bluntly, it is a webpage that is useless with AJAX. You click a button to refresh the page.

After using AJAX, you click, and then a line of words on the page changes, and the page itself does not need to be refreshed.

Summary: As long as it is JS calling asynchronous communication components and using formatted data to update the content or operation process on the web page, then the method we use can be regarded as AJAX.

Below is a simple landing page I made using JQuery and AJAX.
HTML:

```
Users > ey > WebServer > NodeWebServer > MyWebServer > NodeWebServer > clientside > html > hello.html > ...
1  <form id="form1">
2      <div class="user">
3          
4          <input type="text" name="user_username" id="user"/>
5      </div>
6      <div class="pwd">
7          
8          <input type="password" name="user_password" id="pwd"/>
9          <i class="by"></i>
10     </div>
11     <input type="button" value="Login" id="sub"/>
12 </form>
13
```

JQuery:

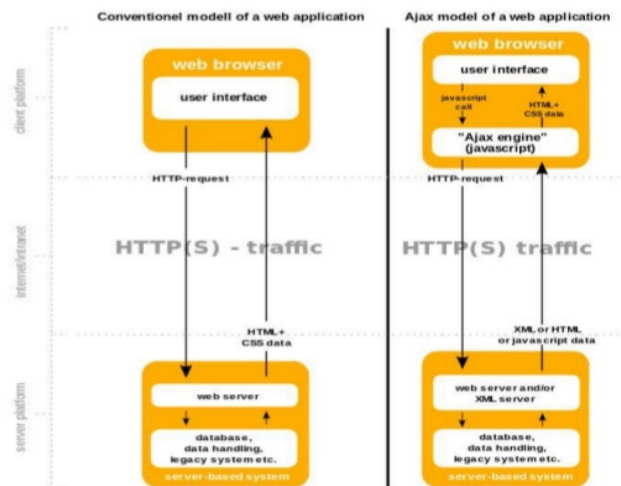
```
Users > ey > WebServer > NodeWebServer > MyWebServer > NodeWebServer > clientside > html > hello.js > on("click") callback > url
1  $("#sub").on("click",function(){
2      var formParam = $("#form1").serialize();
3      $.ajax({
4          async:false,
5          url:"hello.html",
6          type:"post",
7          data:formParam,
8          dataType:"json",
9          success: function(data){
10             alert("1");
11         }
12     });
13 });
14
```

The following is the renderings.



When should we use Ajax

The difference between Ajax and traditional Server Web architecture.



Ajax is essentially a browser-side technology. As can be seen from the figure, the biggest difference of the traditional Server Web architecture is to move a lot of business logic from the service area to the browser.

The advantage of this is that when the data changes, you only need to re-render the relevant HTML, rather than loading the entire page.

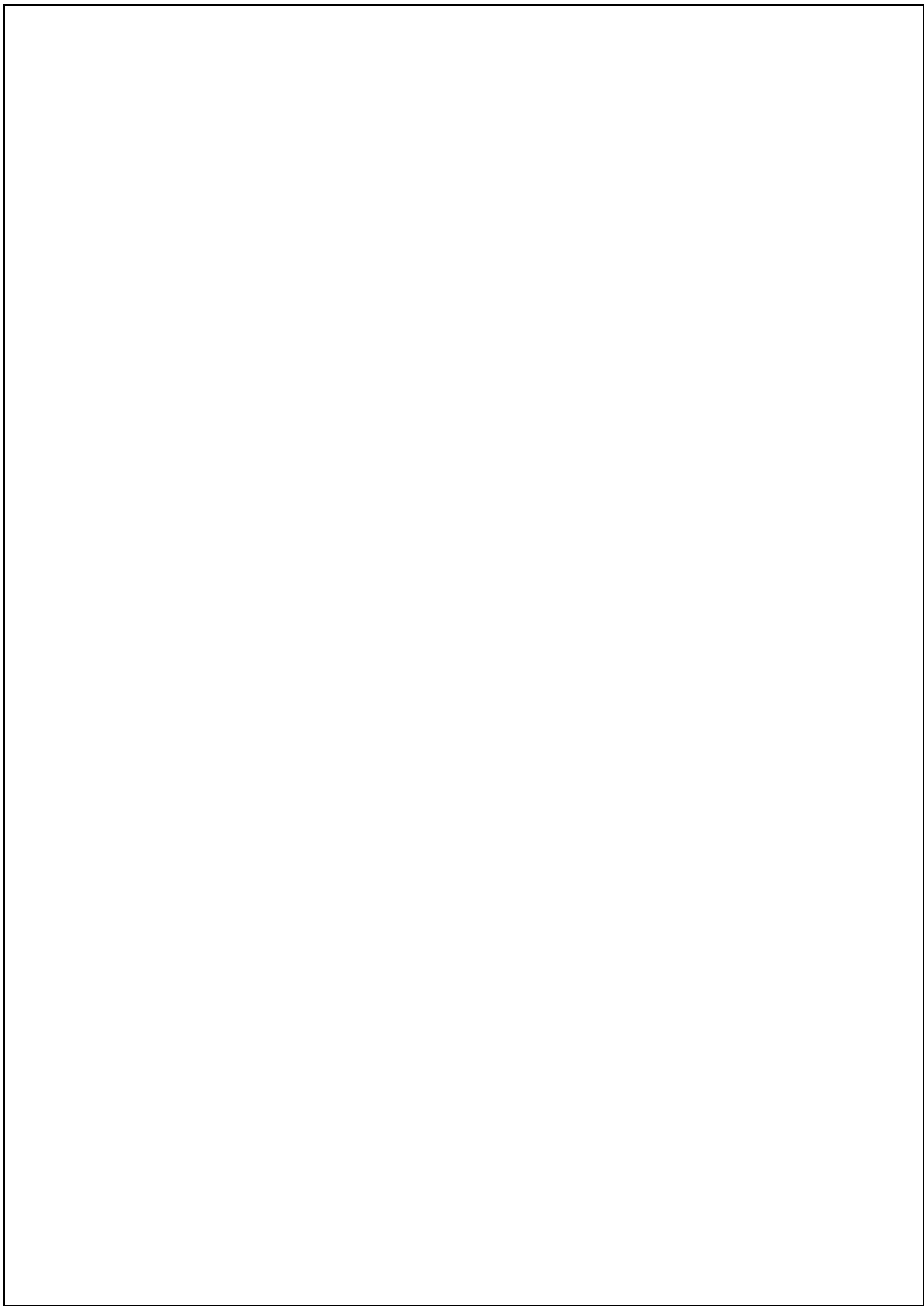
The disadvantage is that a project that uses a lot of Ajax is not so much a B / S architecture as a C / S architecture. Naturally, it also has the characteristics of a C / S architecture. In a browser environment, some features will become defects. For example, Ajax requires that business logic must be loaded before data, and the browser must load related js files before it can start loading data, so the first page preparation time will be later than the Web page. However, there are many solutions

to this problem. For example, the Application cache can save files in the browser to avoid repeatedly loading the same script and resource files.

So the conclusion is: whether you should use Ajax depends on the business process and technology stack.

Reference

- [1] Yash Gupta,Himanshu Dewan,Alka Leekha. Real-time monitoring using AJAX and WebSockets[J]. Journal of Statistics and Management Systems,2020,23(1).
- [2] General dynamics and the AJAX IFV[J]. Asia-Pacific Defence Reporter (2002),2019,45(3).
- [3] Hyun-Jun Park,Kyounghee Lee. A Web Server Implementation Providing Tensorflow-based Artificial Intelligence Platform[J]. INTERNATIONAL CONFERENCE ON FUTURE INFORMATION & COMMUNICATION ENGINEERING,2019,11(1).
- [4] Traian Turc. AJAX Technology for Internet of Things[J]. Procedia Manufacturing,2019,32.
- [5] Shashank Gupta, B.B. Gupta. RAJIVE: restricting the abuse of JavaScript injection vulnerabilities on cloud data centre by sensing the violation in expected workflow of web applications[J]. Int. J. of Innovative Computing and Applications,2018,9(1).
- [6] Fahmida Y Rashid,Fahmida Y Rashid. Node.js alert: Google engineer finds flaw in NPM scripts[J]. InfoWorld.com,2016.
- [7] Krill, Paul. Nodal: Build micro Web servers on Node.js[J]. InfoWorld.com,2016.
- [8] Shu Man Zhao,Xiao Ling Xia,Jia Jin Le. A Real-Time Web Application Solution Based on Node.js and WebSocket[J]. Advanced Materials Research,2013,2708.
- [9] Krill, Paul. The rise of Node.js: JavaScript graduates to the server[J]. InfoWorld.com,2012.
- [10] Anonymous. ComponentOne Unveils HTML5 and jQuery Tools for ASP.Net MVC[J]. Wireless News,2011.
- [11] Wayner, Peter. JavaScript conquers the server[J]. InfoWorld.com,2011.



FINAL GRADE

73/100

GENERAL COMMENTS

Instructor

Server-side/MVC (25) : 19

Client/Server Scripting (25): 16

Node.js (25): 24

AJAX (25): 14

Feedback:

=====

MVC discussion needed more explanation of certain points raised.

General discussion of scripting is fine but you should have found your own examples rather than reuse mine.

Nice discussion of node.js.
The discussion of AJAX was limited and the code provided was very short. I would have liked more working code.

PAGE 1

PAGE 2



Comment 1

Technically, this is not true. Web servers server HTML pages, although they can also provide REST web services.

PAGE 3

PAGE 4



Comment 2

you need to explain this



Comment 3

...again you need to elaborate on this point. Generally, the controller decides what needs to be displayed and feeds that data into the view.



Comment 4

What modes?

PAGE 5



Comment 5

Is it always true that the script is in the same directory as the data? It is fair to say all scripting language run within an execution environment that may restrict access to resources



Comment 6

Scripts still have to be interpreted, but your general point is correct

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15
