

# 1904733

*by* Mimi Zhang

---

|                |                                                           |                 |       |
|----------------|-----------------------------------------------------------|-----------------|-------|
| FILE           | 17847_MIMI_ZHANG_1904733_242847_1621586120.DOCX (122.27K) |                 |       |
| TIME SUBMITTED | 16-APR-2020 04:06PM (UTC+0100)                            | WORD COUNT      | 4220  |
| SUBMISSION ID  | 124267779                                                 | CHARACTER COUNT | 21643 |

# Web Development

Major: MSc Software Engineering ( Web Development)

Module code: ACCA7004

Student: Mimi Zhang

Email: [1904733@student.uwtsd.ac.uk](mailto:1904733@student.uwtsd.ac.uk)

Tutor: Mike Dacey

## Contents

|                                                                 |    |
|-----------------------------------------------------------------|----|
| Contents.....                                                   | 2  |
| 1. Part 1 .....                                                 | 3  |
| 1.1 Server-side code.....                                       | 3  |
| 1.2 Model View Controller (MVC) design pattern .....            | 4  |
| 1.2.1 definition.....                                           | 4  |
| 1.2.2 benefits&drawbacks .....                                  | 5  |
| 2. Part 2 .....                                                 | 5  |
| 2.1 client-side and sever-side scripting.....                   | 5  |
| 2.1.1 Database connectivity.....                                | 7  |
| 2.1.2 The required run-time environment(e.g. the browser) ..... | 7  |
| 2.2 main feature.....                                           | 7  |
| 3. Part 3 .....                                                 | 9  |
| 3.1 Definition of node.js.....                                  | 9  |
| 3.2 Why should we use node.js?.....                             | 9  |
| 3.3 The benefits and drawbacks of using it?.....                | 10 |
| 3.4 How does node.js work? .....                                | 10 |
| 4. Part 4 .....                                                 | 13 |
| 4.1 The importance of AJAX.....                                 | 13 |
| 4.2How to use AJAX? .....                                       | 13 |
| 5. Reference List.....                                          | 14 |

## 1. Part 1

### 1.1 Server-side code

In the world of computers, the party that provides the service is called the server, and the party that receives the service is called the client. The server is for the client, and the content of the service is to provide resources to the client and save the client data. It is the most important way to realize the game's specialization, and it is also the most direct technology that can be expressed through the game. For example, if you want to modify the parameters of an NPC, it will be reflected in the game immediately after reloading [1].

Simply put, the server is for the client, and the content of the service is to provide resources to the client, save the client data, etc. The client can be any computer, as long as it is connected to the server, and gets After the server-side authorization, you can use the server-side services. It can be understood that google's website is the server-side, and the computer we use now is the client-side. We can use its services.

For example, the game server, the game server is for the game client. The content of the service includes providing login for the client, saving game player information, and providing player online games, which is also different from the stand-alone game. The login of the game client requires Server authorization, for example: the game server provides a login to the player, and the login is the necessary tool to connect to the server. The login can also be understood as the authorization file of the server [2].

Generally, when we visit a large website, it is us, that is, the client sends a request, and then the other party's server segment responds. The result is that the page path is returned to us, and we see the page according to the path. , The server is the background, and the client is the foreground. The server is invisible to users, and the client is visible to users.

The server-side is usually a high-level server-level PC, so that multi-client access will not cause delays or even data overflow. The server is a remote server, and the running result is generated by the server; the client is the netizen's computer.

Another example:

For example, for online games, the interface you play is the client, and the server is the server.

You browse the web on the Internet Your machine is the client, and the server provides the web page.

Services on the Internet generally require you to install the corresponding client program to enjoy the service [3].

For example, to browse the Internet, you need to install a browser such as IE. Online games also need to install the corresponding login program.

Client-side: The request object exists in the web and sends the request to the server for processing.

Server-side: As the name implies, it is a service. The request sent by the client is handed over to the server for processing, and it exists as a response object. After the server is processed, it is fed back to the client.

## 1.2 Model View Controller (MVC) design pattern

### 1.2.1 definition

Model refers to the part of the application that is used to process the application data logic. Generally speaking, the model object is responsible for packaging and implementing specific functions of the application. Can realize the business logic in the system, usually can be realized with JavaBean. Model data and business rules.

View refers to the part of the application processing data display. Used for interaction with users. Used to present the content of the model to the user. The user can request the model to be updated through the view. The view obtains the data to be displayed from the model, and then displays it to the user in its own way, which is equivalent to providing a page for human-computer interaction with the user. For example, the user clicks OK after completing the information filling in the login registration interface, thereby sending this request to the controller [4].

Controller refers to the part of the application that handles user interaction. Usually the controller is responsible for coordinating the model and the view, **reading data from the view**, controlling user input, and sending data to the model, choosing which model to call for processing business according to the user request, and which view ultimately responds to the user. Controller is the bridge between Model and View. Used to control the flow of the application and process the requests issued by the view. When the controller receives the user's request, it will map the user's data with the model, that is, call the model to implement the function requested by the user. The controller then selects the view to respond to and presents the updated data of the model to the user.

When it comes to coding principles, Model View Controller (MVC) is the most basic. The core idea is that MVC describes the relationship between application data, user interface and "brain". By design, all three components of MVC remain independent.

Examples of MVC architecture patterns:

Case 1: Switch cassette game console, Control is the host, generally I just buy a host, as long as he is not bad, he can let me play this kind of game all the time. View is a TV and a game controller. The TV can work independently. Whether he is inputting a TV signal, a DVD player signal or a game console signal, he only cares about the display, and he determines what the effect we see is, if I want a larger or colorful display. I only need to buy a corresponding TV set. The handle can also be changed. The joystick is still vibrating. Model is a game cassette. He decides what game I play, whether it is Contra or Super Mary, and the game console and TV manufacturers never know what kind of games may be run on it. There may be game codes and storage units in the cassette, all designed according to the needs of the game [5].

Case 2: The model in MVC is where the application stores data. For the address book in the phone, the data is all contacts. The model knows all the data of everyone in the address book, but it has no way to present the data to the user. Because the address book model is separate from the view and controller, as a developer, you can access user address book contact information in your own applications and use it for different purposes. You can redesign the way contact information is presented to users, or redesign the way your application interacts with data. Similarly, when designing your own application, the design that separates the model from the view and controller makes it easier and faster to upgrade, fix bugs, and reuse code.

The view in MVC is usually the user interface: that is, what the user sees, including the options needed to use the application. In some cases, the view can talk directly to the model, but usually the

controller interacts with the model, and the controller determines what a behavior should do. The address book view is the list of contacts you see on the device screen. When you select a contact in the view, this behavior is sent to the controller. The controller then obtains the necessary contact information from the model, and then passes this information back to the view, and the view is finally displayed to the user.

The controller is the brain of the application. The controller usually determines the navigation style and workflow of the application. The model stores data, the view presents data, and the controller determines how to store and store what kind of data, and how to present and present what kind of data [6].

### 1.2.2 benefits&drawbacks

#### Advantages:

1. Separation of view and data model: using different views to display the same data; separating visible and invisible components, can independently test the model. Because the separation of visual components reduces external dependencies and facilitates testing. (The database is also an external component)
2. Separation of view and controller: Controller is a component that represents logic, not a business logic component. MVC can be used as a presentation mode or as a construction mode, meaning that the Controller can also be business logic. Separate logic and specific display, can independently test the logic.
3. The responsibilities of these three parts of the MVC model are very clear and separate from each other, so each part can be changed independently without affecting the other parts, thereby greatly improving the flexibility and reusability of the application.

#### Disadvantages:

1. Strictly follow MVC, and separate the model, view and controller, which will 1increase the complexity of the structure, and may generate too many update operations, reducing operating efficiency.
2. The MVC architecture is circular. Information and behavior flow freely between components, but the establishment of each component requires high requirements to ensure the independence of these core tasks.

## 2. Part 2

### 2.1 client-side and sever-side scripting

Script is an executable file written in a specific format using a specific descriptive language, also known as a macro or batch file. Their characteristic is that it runs on the server and returns to the client for display after parsing through the server, so they are used for display after operation on the server [7].

#### Client-side:

The client script is completely interpreted and executed in the client browser, mainly to control the user's interaction with the browser, it does not do any processing on the server side, including

1. `<script>` Mark the code for alignment;
2. The code written in a separate js file (such as test.js in `<script id = "s" src = "test.js"> </ script>`).

**Service-side:**

It is mainly used to generate webpage code. The server-side script in asp.net and the code in the background .cs file have the same status, that is, the background code is transferred to the foreground in the form of server-side script. include

1. `<scriptrunat = "server">` mark the code in the pair (only methods and attributes can be placed, not directly add statements and expressions);
2. Code in `<%%>`.

There are few client-side scripts. The most common one is JAVASCRIPT. He cannot directly operate the server. He increases the user experience on the client and interacts with the server through AJAX.

The scripting language that runs on the server side is the server-side script. The script that runs on the client segment is the client segment script. The essential difference is that the running location is different.

### 2.1.1 Database connectivity

The server-side script runs on the server. You can never see the contents of the server. All you get is the HTML code sent back by the script interpreter. Instead of directly performing database operations, he implemented simple logic on the client.

The purpose of the client-side script is obvious, not only to avoid repeated redundant links, but also to ease the pressure on the server (no need to query the database multiple times).

### 2.1.2 The required run-time environment(e.g. the browser)

#### 1. Client-side script

Supported languages: JavaScript JScript VBScript

Operating environment: client browser

#### 2. Server-side script

Supported languages: PHP, JSP, ASP, ASP.NET

Operating environment: Web server

## 2.2 main feature

Client script (mainly used to improve user experience):

1. Provide user interaction.
2. Verify the data.
3. Dynamically modify the content.
4. Realize different page dynamic effects.

Server script (mainly used to ensure data integrity and data security):

1. Dynamically edit, change or add any content to the Web page.
2. Respond to user requests or data submitted by HTML forms.
3. Access the data or database and return the results to the browser.
4. Improve web page security [8].

Code examples and results are shown below

```

1 <html xmlns="client-side">
2 <head>
3   <title>jstest</title>
4   <script src="./part2.js">
5   </script>
6   <script type="text/javascript">
7     alert("This is client-side scripting.");
8     function a(){
9       alert("client-defined function.");
10    }
11  </script>
12 </head>
13 <body>
14   <form method="POST" action = "eg.asp">
15     Module: <input type="text" name="modulename"/>
16     Tutor: <input type="text" name="tutorname"/>
17     <input type="submit" value="Submit" name="submit"/>
18   </form>
19 </body>
20 </html>

```

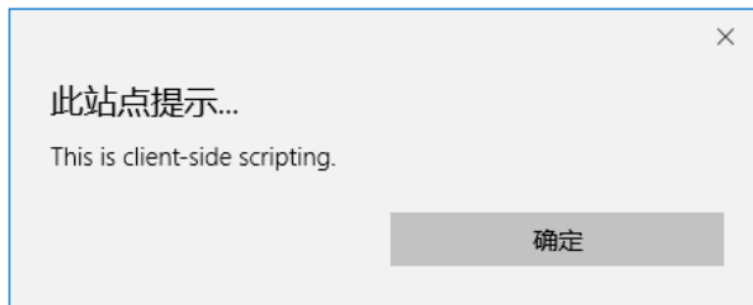
Picture 1: client-side

```

1 <html xmlns="server-side">
2 <head>
3   <title>jstest</title>
4
5   </script>
6   <%
7     Response.Write ("This is server-side scripting");
8   %>
9   <script type="text/javascript" runat="server">
10     public void ServerSide()
11     {
12       Response.Write("This is server-side scripting 1.");
13     }
14   </script>
15   <%Response.Write("This is server-side scripting 2.");
16 </head>
17 </html>

```

Picture 2: server-side code



Picture 3: result 1 for the code

Module:  Tutor:

Picture 4:result 2 for the code

### 3. Part 3

#### 3.1 Definition of node.js

It can be said that two points contributed to the birth of Node.js. First of all, Ryan found that the characteristics of the JS language itself are event-driven and non-blocking I / O, which is a perfect match with his thinking. The second point is that Chrome's JS engine, that is, the V8 engine is open source, and its performance is particularly good. So Ryan developed Node.js based on V8. Note that Node.js seems to be a JS library, but it is not. Node.js is developed by C ++. You can see it on the official website <http://nodejs.org> [9].

Node.js is a Javascript runtime environment based on Chrome V8 engine. So Node.js is not a library, it is a running environment, or a JS language interpreter.

The initial positioning of Node.js was to improve Ryan's own daily work efficiency, which is used to write server code, but then did not expect that Node.js shines in the front-end field..

#### 3.2 Why should we use node.js?

The birth of Node.js brings a big surprise to people. Traditionally, web developers write front-end in JS, but they must use another language when writing server-side code, similar to Ruby / Java / PHP. But after the emergence of Node.js, JS took all the time. If you go online to search for Node.js information, many use Node.js to write server code.

However, Node.js has also become an essential infrastructure for front-end Web development today. Note that the JS code of the Web front end is still running in the browser, so when it is running, or in the production environment, it does not depend on Node.js. However, after the birth of Node.js, the front-end broke out, and the development environment of front-end frameworks like React / Vuejs became very powerful and responsible. Node.js is the basis for the operation of these development environments [10].

In summary, the reasons for choosing Nodejs can be summarized as the following 4 points:

Dynamic language: The development efficiency is very high, and the ability to build complex systems, such as `q1.io`.

Performance and I / O load: Nodejs solves the IO-intensive problem very well, through asynchronous IO.

Connection memory overhead: Each Node.js process can support more than 120,000 active connections, and each connection consumes approximately 2K of memory.

Operational: A Nodejs monitoring system for the memory stack is implemented.

### 3.3 The benefits and drawbacks of using it?

#### Benefits:

1. Unified front and back end, database.
2. Node.js did not re-develop the runtime environment, but chose the fastest browser kernel V8 as the execution engine to ensure the performance and stability of Node.js.
3. The Node.js community is growing. Not only is the number of packages increasing rapidly, but the quality of packages is also significantly better than other languages. Many bright star packages are simple and smart, designed for developers' usage habits.

#### Drawbacks:

1. Some packages are not mature enough, and debugging is very difficult. Without stack trace, if something goes wrong.
2. It is easy to write bad code. The execution flow of callback is sometimes not very intuitive, and it needs to be reviewed and refactored regularly to avoid.
3. Memory control, compared with Javascript and Java, do not have an advantage in the definition of complex data types. Because the object-oriented of Javascript is based on JSON, and Java directly uses the memory structure. So, through the process of JSON serialization and deserialization to control memory, Javascript has lost.
4. For large memory applications, due to the memory design limitations of the V8 engine, the maximum heap in a 32-bit environment is 1G, and the maximum heap in a 64-bit environment is also less than 2G. If you want to read 10G data at once, for Node.js It cannot be achieved.
5. Static server, although the advantages of Nodejs are in IO-intensive applications, there is still a big gap from Nginx in handling static resources.
6. Applications that do not require asynchronous: such as system management, self-scripting, etc., or Python is more convenient, asynchronous call of Node.js may bring some trouble to programming [11].

### 3.4 How does node.js work?

"Single thread, asynchronous I / O, event-driven", the application request process can be divided into two parts: CPU operation and I / O read and write, CPU calculation speed is usually much higher

than disk read and write speed, which leads to The operation has been completed, but I have to wait for the completion of the disk I / O task before continuing to the next business.

Therefore, I / O is the bottleneck of the application. In I / O-intensive services, it is assumed that the request requires 100ms to complete, of which 99ms is on I / O. If we need to optimize the application so that it can process more requests at the same time, we will use multi-threading and open 100 or 1000 threads at the same time to improve our request processing. Of course, this is also a considerable solution.

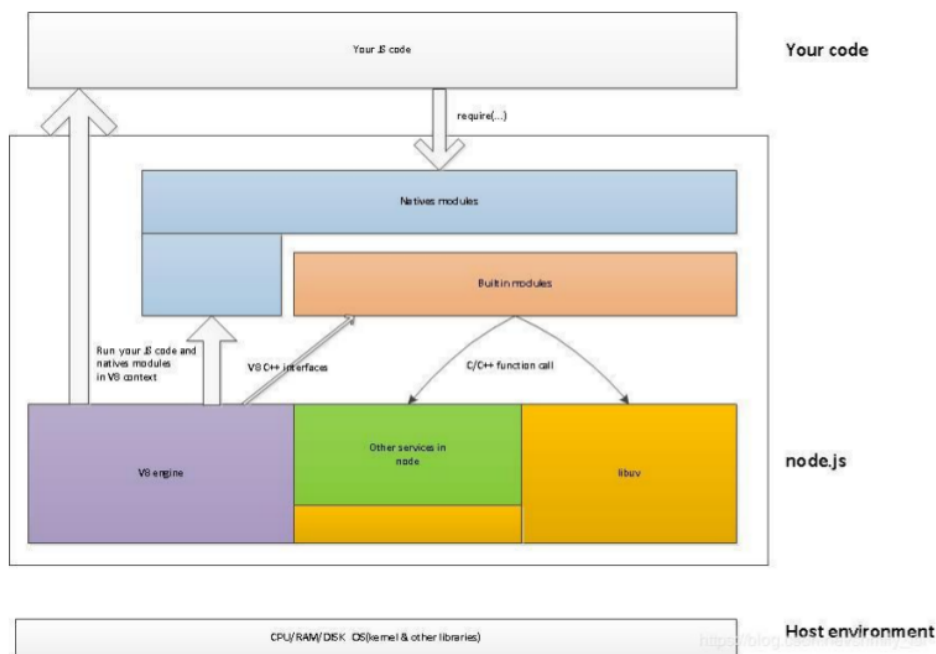
However, since a CPU core can only do one thing at a time, the operating system can only divide the CPU into time slices so that threads can use CPU resources more evenly. However, when the kernel switches threads, the operating system also needs to switch the thread's online text. When there are too many threads, time will be consumed in context switching. Therefore, in large concurrency, the multi-threaded structure is still unable to achieve strong scalability [12].

The single thread of Node.js is not really single thread, just open a single thread for business processing (cpu operation), and open other threads to deal specifically with I / O. When an instruction arrives at the main thread, the main thread will directly pass this event to the I / O thread after discovering that there is I / O. Instead of waiting for the end of the I / O, it will process the following business, but after getting a status Go down immediately, this is "single thread", "asynchronous I / O".

After the I / O operation is finished? There will be a callback event after I / O processing of Node.js, this event will be placed in an event processing queue, when the process starts, node will create a loop similar to While (true), each time it polls Will go to see if there is an event that needs to be processed, if there is a callback function associated with the event that needs to be processed, if there is, then add the next poll, and if not, exit the process. This is called "event-driven". This also explains what is "event-driven" from the perspective of Node.

In node.js, events mainly come from network requests, file I / O, etc. The observers are classified according to different events. There are file I / O observers and network I / O observers. Event-driven is a typical producer / consumer model. When the request reaches the observer, the event loop consumes from the observer, and the main thread can focus on the business non-stop and no longer have to wait for I / O .

From the perspective of module dependency:



Picture 5: Node.js component

As shown in the figure above: your code is the editing code, node.js core, and Host environment is the host environment (providing various services, such as file management, multi-threading, multi-process, IO etc)

Nodejs components: v8 engine, libuv, builtin modules, native modules and other auxiliary services.

v8 engine: There are two main functions:

1. The function of the virtual machine, execute js code (own code, third-party code and native modules code).
2. Provide C ++ function interface, provide v8 initialization for nodejs, create context, scope, etc.

libuv: It is an event-driven asynchronous IO model library. Our js code sends a request and is finally completed by libuv, and the callback function we set is triggered in libuv.

builtin modules: It is written by C ++ code into various modules, including the basic functions of crypto, zlib, file stream etc. (V8 provides a function interface, libuv provides an asynchronous IO model library, and some nodejs functions to provide services for builtin modules).

native modules: it is written in js and provides a library called by our application. At the same time, these modules rely on builtin modules to obtain corresponding service support

To sum up briefly: If you think of nodejs as a black box, the interface exposed to developers is native modules. When we initiate a request, the request goes from top to bottom, traverses the native modules, and passes the request to v8 through builtin modules And other auxiliary services, the request ends, then trace back to the top, and finally call our callback function.

## 4. Part 4

### 4.1 The importance of AJAX

Among the above technologies, except for the XMLHttpRequest object, all other technologies are based on the web standard and have been widely used. Although XMLHttpRequest has not been adopted by the W3C, it is already a de facto standard. Almost all major browsers support it [13].

Benefits of AJAX:

1. Usability. Developers and designers are beginning to realize that not only are large user experiences successful in the market, but also how such experiences affect user costs. Google Maps based on AJAX technology is more successful than the traditional choice of MapQuest, proving the success of products that provide a better user experience. People already know that they need an excellent user interface and have the willingness to invest in the interface. The prerequisite is that users can quickly obtain information regardless of whether the data is an intranet application or a wide area network service. The biggest point is that the page does not refresh, and communicates with the server within the page, giving the user a very good experience.
2. Benefit conventional WEB application. These are the advantages that have prompted businesses and users to adopt WEB applications since the 1990s. AJAX technology can not only benefit WEB applications, but also benefit end users.
3. For AJAX, the release of Mozilla 1.0, the foundation of the FIREFOX browser and supporting the XMLHttpRequest object is a turning point. This technology that allows asynchronous data exchange has been supported by Internet Explorer for many years. This support and the large number of FIREFOX browsers have been adopted to truly make it possible for people to understand rich Internet applications across browsers.
4. Cross-browser and cross-platform compatibility. This is one of the most important reasons why AJAX applications have become so popular. Many developers realized the possibility of the popularity of AJAX technology many years ago, but it has not been popular because of browser manufacturers.
5. Promote the upgrading of skills, tools and technologies. This means that the development community that develops WEB applications needs to accelerate the upgrade of their user interfaces to AJAX, but does not require a large-scale upgrade and rewrite their WEB applications. Since the 1990s, those systems that have invested heavily in developing browser-based application methods strongly hope to increase the user experience on the basis of existing applications.
6. Good integration with technologies such as Flex and Flash. Most development communities no longer support the hot discussion of FlashvsAJAX. Both technologies have their own advantages and disadvantages on different occasions, but they have a lot of opportunities to integrate and work together.
7. AJAX is based on the uncertainty of server technology. The independence of the browser is the same as AJAX technology.

### 4.2 How to use AJAX?

See code documentation for details.

## 5. Reference List

- [1]G. Manduchi,A. Luchetta,G. Moro,A. Rigoni,C. Taliercio. Web-based streamed waveform display using MDSplus events and Node.js[J]. Fusion Engineering and Design,2020,157.
- [2]Yash Gupta,Himanshu Dewan,Alka Leekha. Real-time monitoring using AJAX and WebSockets[J]. Journal of Statistics and Management Systems,2020,23(1).
- [3] The Boundaries of Hatred: Odysseus' Pity in Sophocles' Ajax[J]. Nova tellus,2019,37(1).
- [4]Eric Bronson. The Myth of Resilience: Ajax Undressed[J]. Mosaic: an interdisciplinary critical journal,2019,52(4).
- [5]Konstantinos Tserpes,Maria Pateraki,Iraklis Varlamis. Strand: scalable trilateration with Node.js[J]. Journal of Cloud Computing,2019,8(1).
- [6]Anonymous. Ajax Rescue Tools' EXTRICATION TOMAHAWK[J]. Fire Engineering,2019,172(11).
- [7].General dynamics and the AJAX IFV[J]. Asia-Pacific Defence Reporter (2002),2019,45(3).
- [8]Kulawiak Marcin. Client-side versus server-side geographic data processing performance comparison: Data and code.[J]. Data in brief,2019,26.
- [9]. Notice Pursuant to the National Cooperative Research and Production Act of 1993--Openjs Foundation (Formerly Node.js Foundation)[J]. The Federal Register / FIND,2019,84(71).
- [10] MariaDB Corporation; MariaDB and Node.js Connection Strengthens, Adding Features and Faster Performance[J]. Computers, Networks & Communications,2018.
- [11]Serpette P. Bernard,Serrano Manuel. An interpreter for server-side hop[J]. ACM SIGPLAN Notices,2011,47(2).

[12]Engin Kirda,Nenad Jovanovic,Christopher Kruegel,Giovanni Vigna. Client-side cross-site scripting protection[J]. Computers & Security,2009,28(7).

[13]Jan de Leeuw. Server-side Statistics Scripting in PHP[J]. Journal of Statistical Software,1997,2(1).

FINAL GRADE

88/100

GENERAL COMMENTS

**Instructor**

Server-side/MVC (25) : 23

Client/Server Scripting (25): 20

Node.js (25): 23

AJAX (25): 22

Feedback:

=====

The MVC discussion was a little misleading in parts but generally demonstrated a understanding of the main benefits of the approach.

Scripting didn't really talk enough about the run-time environment e.g. JavaScript engines and the DOM.

Nice explanation of node.js, one of the main disadvantages is that you need to avoid computationally expensive operations which will delay the server or use worker threads.

Benefits of AJAX didn't really highlight how or why developers use it e.g. single page applications. Code was quite fun, but you should separate HTML and JS rather than mix them in the same file.



### **Comment 1**

Not sure what this means



### **Comment 2**

I would not say that it reads the input values from the user through a view. This goes through the controller.



### **Comment 3**

You need to define these terms



### **Comment 4**

You need to elaborate on this, what does this mean?