Client-side and Server-side Validation

Introduction

Securing a website through data validation presents a formidable challenge for web developers in a landscape rife with malicious actors. These adversaries are relentless in their pursuit, probing and attempting to bypass the validation defenses meticulously erected by developers. Their intentions, often nefarious, range from seeking unauthorized access to a treasure trove of sensitive and proprietary data to potentially overloading the server, triggering catastrophic crashes.

In this evolving digital battleground, safeguarding your server is paramount. The key lies in harnessing the combined power of client-side and server-side validation. Together, they form a strong defense that shields your web application from assaults.

Client-Side Validation

Client-side validation is done on the user's browser immediately after the data has been entered. This method ensures that the user gets immediate feedback, with no need to wait on the server to return the validated information. The user immediately knows if they have made a mistake, and the information entered can be kept intact so that the user only needs to correct their mistake before attempting to send the data again.

JavaScript is the most common language used for client-side validation. JavaScript has a rich variety of libraries and frameworks available that can be used to simplify the validation process and make the user's experience faster and easier. In a world where many users see filling out a form as a nuisance to be skipped, making the experience as quick and painless as possible is a top priority for a web developer. When success on a webpage is often measured in analytic conversions, dropped web forms can be a disaster, especially if that webform is a means to income for the website's organization.

There are other means to client-side validation, including HTML5 built-in form validation where the developer can include attributes of "required", "min", "max", "pattern" as well as others. The example below shows how simple HTML5 validation can be, with the "required" attribute being placed at the end of the standard input field.

```
<label for="username">Username:</label>
<input type="text" id="username" name="username" required>
```

JavaScript continues to maintain its status as the preferred language for client-side validation in web development, even with the availability of alternative options. There are several compelling reasons underpinning its widespread adoption:

To begin with, JavaScript enjoys native support across web browsers, ensuring reliable compatibility. This is especially important for older browsers that may lack HTML5 support or only offer partial compatibility.

Furthermore, JavaScript empowers developers with the ability to implement highly customizable validation logic. This level of control allows for the creation of intricate and tailored validation rules to meet specific project requirements.

Additionally, JavaScript enables dynamic validation that can adapt to user interactions. For instance, it facilitates the validation of one input field based on the content of another, such as confirming an email address in a separate field.

Moreover, JavaScript provides developers with the means to enhance the user experience by offering real-time feedback as users interact with a website. This real-time interaction significantly improves the overall usability and responsiveness of web applications.

In addition to these advantages, separation of concerns should be high on a list of reasons to stick with JavaScript. If the website already has JavaScript files, especially if some JavaScript is already being used in validation, then keeping the client-side validation cohesively in one language is desirable.

In addition to providing immediate feedback, client-side validation ensures that the server only receives valid data, meaning that server load is reduced. By performing validation in the user's browser, invalid data never leaves the user's computer. If the website's server has high traffic or if you are on an inexpensive server purchased and run by a budget conscious company, the server load might be a legitimate concern and constantly bombarding that server with invalid data that it must reject could lead to increased latency. Latency could then lead to frustrated users submitting the same information multiple times while they wait for a response from the server.

While client-side validation can prevent invalid data from reaching the server, it should never be thought of as a security measure. JavaScript can easily be bypassed by simply turning off JavaScript functionality in the browser. With just a few short steps in the browser's dev tools, a user can stop JavaScript from running on their computer and disable all the JavaScript validation written into a webpage. HTML5 is slightly harder to bypass, but it is possible by using a command line tool such as curl. Using tools such as curl makes it possible to send data to a website's server without even opening the website.

Additionally, any code that is written for the website can be directly edited by a user through the browser's dev tools. A user could completely rewrite all the JavaScript code if they wanted. The changes would only affect the website on their own computer, as they are only editing the code that their browser downloaded onto their computer, but it would change any client-side validation that was erased or rewritten. For these reasons, client-side validation is about user experience through immediate feedback and the convenience of reduced server load, not security.

Server-side Validation

Server-side validation is a crucial component of any website, providing more security than client-side validation because, as the name suggests, all validation is done on the server. When a user interacts with a web application, they send data to the server. The server receives the data, and it needs to be validated before it is processed or stored. Because it is validated on the server, a user does not have direct access to any of the code.

There is no one language that dominates the field of server-side validation. According to the Stack Overflow Developer Survey 2023, the most popular language used for server-side validation is JavaScript. The language has matured in recent years, and it has become more popular as a back-end language. However, JavaScript must use a runtime environment such as Node.js to be used as a backend language and complex maintenance is often listed as a hindrance in using it in this role.

Other languages that can be used for server-side validation include, but are not limited to, PHP, Ruby, and Python. These languages all have well-developed frameworks and communities that can help you get started quickly. In a Node.js application, you can use libraries such as Express.js to handle the validation. Django is a Python web framework that has built-in form handling and validation. Below is a simple example of a Django form created in forms.py where Django receives a POST request and instantiates a form with the submitted data. The validation is performed by calling "form.is_valid()". The user is then given an error message for failed validation or success message if the validation passes.

```
from django.shortcuts import render
from django.http import HttpResponse, HttpResponseRedirect
from .forms import RegistrationForm

def register(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            # Process registration if validation passes
            return HttpResponse('Registration successful.')
    else:
        form = RegistrationForm()

    return render(request, 'registration/register.html', {'form': form})
```

It is important to note that, despite the emergence of other languages being used server-side, PHP is still used to empower close to seventy percent of websites in the world and remains highly relevant. PHP offers an easy to learn, flexible and scalable language and should not be dismissed as an option.

Advantages and Disadvantages

Server-side validation offers a significant advantage in terms of ensuring consistency throughout the data processing pipeline. It guarantees that all incoming data is treated uniformly, irrespective of the user's choice of computer or web browser. While modern browsers strive to adhere to the web standards established by the World Wide Web Consortium, occasional discrepancies may still arise. Server-side validation eliminates these concerns related to how browsers interact with JavaScript.

Furthermore, even if a user attempts to bypass client-side validation, such as through command-line tools, they encounter the same server-side validation procedures as those using web browsers. To stress again, the uniformity in validation extends beyond the browser; whether the data source is a mobile application, an API request, or any other means, server-side validation ensures consistent checks and standards are applied uniformly across all sources.

It is for this reason that a developer can look to the server-side validation for the security that is lacking on the client-side. Even if the developer included hidden or encrypted fields, the server is able to validate this data whereas it is not available for the client to validate. Examples of hidden or encrypted fields would be for a password, credit card number, social security number, medical records, and even proprietary business data. These items cannot be validated on the client because any information available for client-side validation can be seen by anyone that has access to that computer. Sensitive data can be validated on the server while taking appropriate precautions and implementing security measures to ensure it remains inaccessible to unauthorized parties.

The drawbacks to server-side validation is the exact opposite of the server-side benefits. Submitting data to the server can take more time, especially if there is high latency due to the user’s internet connection or traffic between them and the server. This can be exacerbated if the server is busy, and many users are submitting data that needs to be validated by the server. With increased server load, the latency could be caused by the server itself as it strains to keep up with the increased load. Even when everything is going perfectly, users are entering valid data, there are no malicious actors and there is no latency between connections, the strain of a busy session can still cause problems on a server. Many global corporations have found their servers becoming nonresponsive on the opening of a popular new feature or the sale of a highly sought-after product.

Most of the fixes to these problems will be put on the shoulders of the server administrator, but the developer can help by making sure his validation process is as efficient as possible. Choosing appropriate data structures, optimizing your iterations, using data validation libraries for common validation tasks, and implementing a routine that checks for common errors and exits as soon as validation failure is encountered are some of the commonly suggested ways to make your code more efficient that can be easily researched and implemented.

Conclusion

Both client-side and server-side validation have advantages and disadvantages. Comparing them, it is easy to see that many of the features of each are in direct opposition to the other. Client-side is immediate while server-side can take time depending on latency. Client-side offers no security while server-side is considered the last gatekeeper for incoming data. These contrasts should not be considered as a reason to choose between the two but should be seen as each side complimenting the other. Using both client-side and server-side is essential for developing a secure, user-friendly web application. When planning a website that will need to validate data, a multifaceted approach that uses both client-side and server-side validation is critical. By layering both together, you create a redundancy, a synergy, that will establish a robust and reliable system to safeguard your web application.

In the landscape of web development, nothing stays the same and securing a website against constant threats is paramount. Client-side validation is your immediate feedback and server-side is your gatekeeper against database attacks and using both can help to improve the overall quality of the data that is collected and stored by a web application. When data is validated on both the client side and the server side, it is more likely to be accurate and complete. This can lead to better decision-making and improved business outcomes. Whether you write your own code or use a library, clean, efficient methods are the key. By combining the advantages of both, you can develop secure and user-friendly web applications.