BIT607

Assignment 2

Tayla Brown

Report

1. Cross site scripting (XSS) is a security issue a website designer should be aware of. This is because cross site scripting is used to steal user data, steal cookies or redirect users to a different webpage. Regarding preventing XSS from happening to my website when I was developing it, I considered when the php code is added to the html and CSS code is to use the function htmlspecialchars() whenever I want to output something to the browser that came from the user input like reserving a table. The htmlspecialchars() function converts some predefined characters to HTML entities which will prevent XSS from happening to my website.

Another security issue a website designer should be aware of is SQL injection. This is where a user enters data into a SQL statement allowing bypass authentication, steal information, modify database records. To prevent SQL injections, you can use parameterized queries which is also known as prepared statements. Parameterized queries work by the sqlQuery is sent as a query at the database knows exactly what the query will do, and only then will it insert the username and password merely as values. This means they cannot affect the query because the database already knows what the quey is supposed to do. (Parker, 2011)

1. HTTP caching and content delivery networks (CDN) can further improve responsiveness by the CDN letting users connect to a geographically closer data center which reduces distance in turn less travel time means a faster service for the user. CDN also improves responsiveness faster server response as it prevents server overload because it is across multiple servers across the country. CDN also can be proactive or passive which if it is proactive then the content is replicated to servers at different places globally. Passive replicates content by using cached to a server after it has been used. (Xiao, 2008)

HTTP caching is used to store copies of a given resource and sends it back when it is requested. This further improves responsiveness as it reduces unnecessary network request. HTTP caching also reduces latency and network traffic which lessens the time needed to display a representation of a resource which in turn makes the website become for responsive. (HTTP caching, 2020)

1. A responsive layout like a fluid grid system allows the system to rearrange automatically depending on the size of the screen without the use or need for media query selectors. This is one of the main principles of responsive design as I have used this principle throughout my application by defining a maximum layout size for my application and then dividing my application content into columns so that I am able to keep the layout of my application clean and easier to handle. An example of this in my applications is in my style.css I have set a specific width throughout my containers in every html throughout my website so that when the user uses a mobile device rather than a pc my content sizes down automatically so its able to be read and still can be easily used for the user.

Another responsive design principle is fluid images. Images are not naturally fluid; they will stay the same size as they are on a pc to a mobile device which creates a problem as the user is not able to see the image correctly. The way I handled this in my application was by using a simple fluid image. This meant for most of my images I sized my images in relative units, rather than absolute pixel dimensions. The solution I did for my applications images was to set the max-width of the image at 100%. Which made the image responsive and would size up or down depending on the size of the device the user was using.

Another one of the main responsive design principles is media queries. Media queries are used to alter the layout of the site when a certain design size is met. The idea behind media queries is that your able to specify when the screen content should be rearranged depending on what size the screen is. I used media queries a lot throughout my application. An example of this was in my navbar throughout my entire website if I didn’t use a media query the user wouldn’t be able to view the full navbar in a mobile screen because the content in my navbar wouldn’t be able to fir across the entire screen so I decided to use the media query to create a vertical navbar when the screen is at a max-width of 600px which works great because now when the screen is sized below 600px a “hamburger” which is three stacked lines appear which the user then can click and a vertical navbar will appear and works the same way as what the original horizontal navbar did.

1. The first extension I installed is called Prettier. Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account and wrapping code necessary. I installed prettier onto visual code and tried it out on my own code. I also changed the settings so that every time I would save my work prettier would then update my work and format it to how they think it should be.

Below are four screenshots of my code, two are before prettier and two are after prettier is installed. As you can see in my html code not a lot has changed at all. The only noticeable change I can see is that the length of the paragraph has been adjusted. In my CSS code there is also no big changes except now having spaces before the curly bracket. I did test out if I had made a mistake in my code by not having a semi-column where I needed it as soon as I clicked saved a semi-column appeared where it needed to be.

Overall, the Prettier extension I think could be useful if you are messy when writing your code and are more focused on writing out as much code as you can rather than looking over the code and making sure it is readable to other users. It is also useful for if you accidently forget a semi column somewhere in you code Prettier will see it and add a semi-column in for you, which I think would be very handy to have when working on big projects.

Before Prettier:

Text

Description automatically generated

After Prettier:

Text

Description automatically generated

Before Prettier (CSS):

Text

Description automatically generated

After Prettier (CSS):

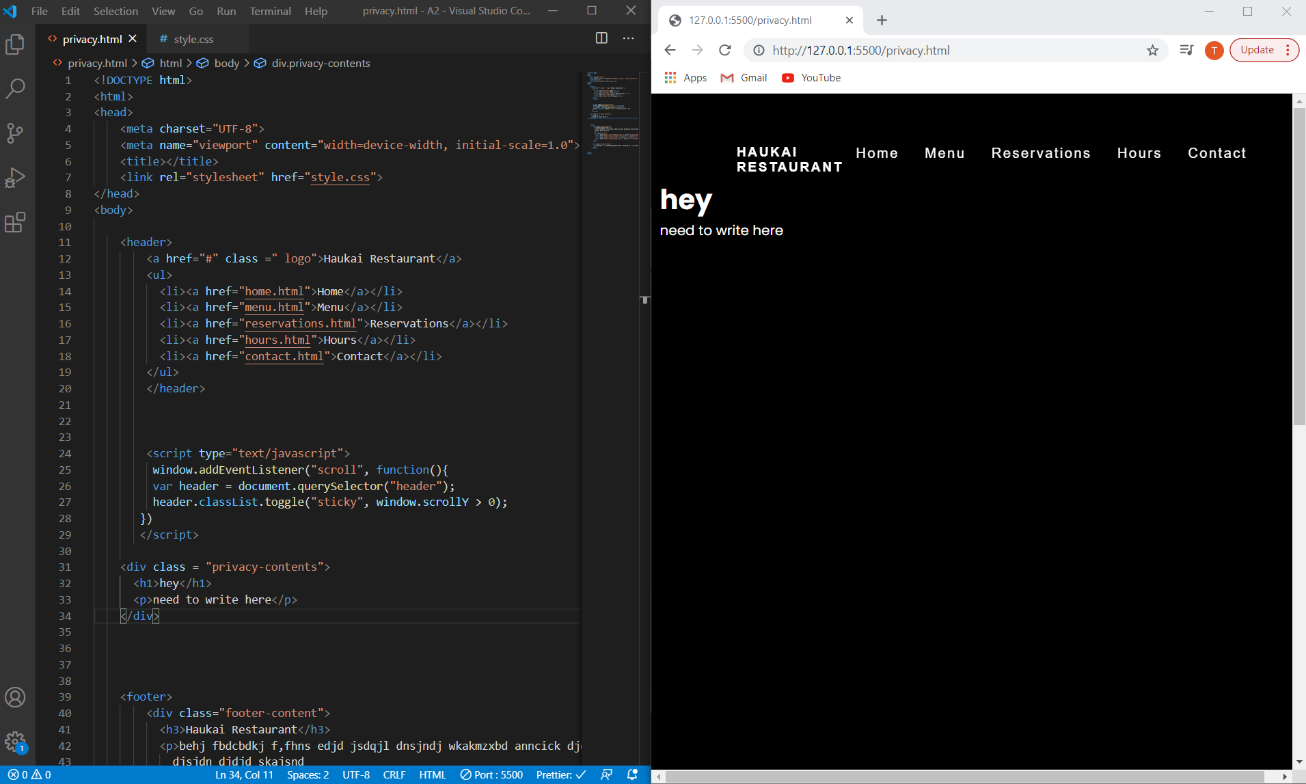
Text

Description automatically generated

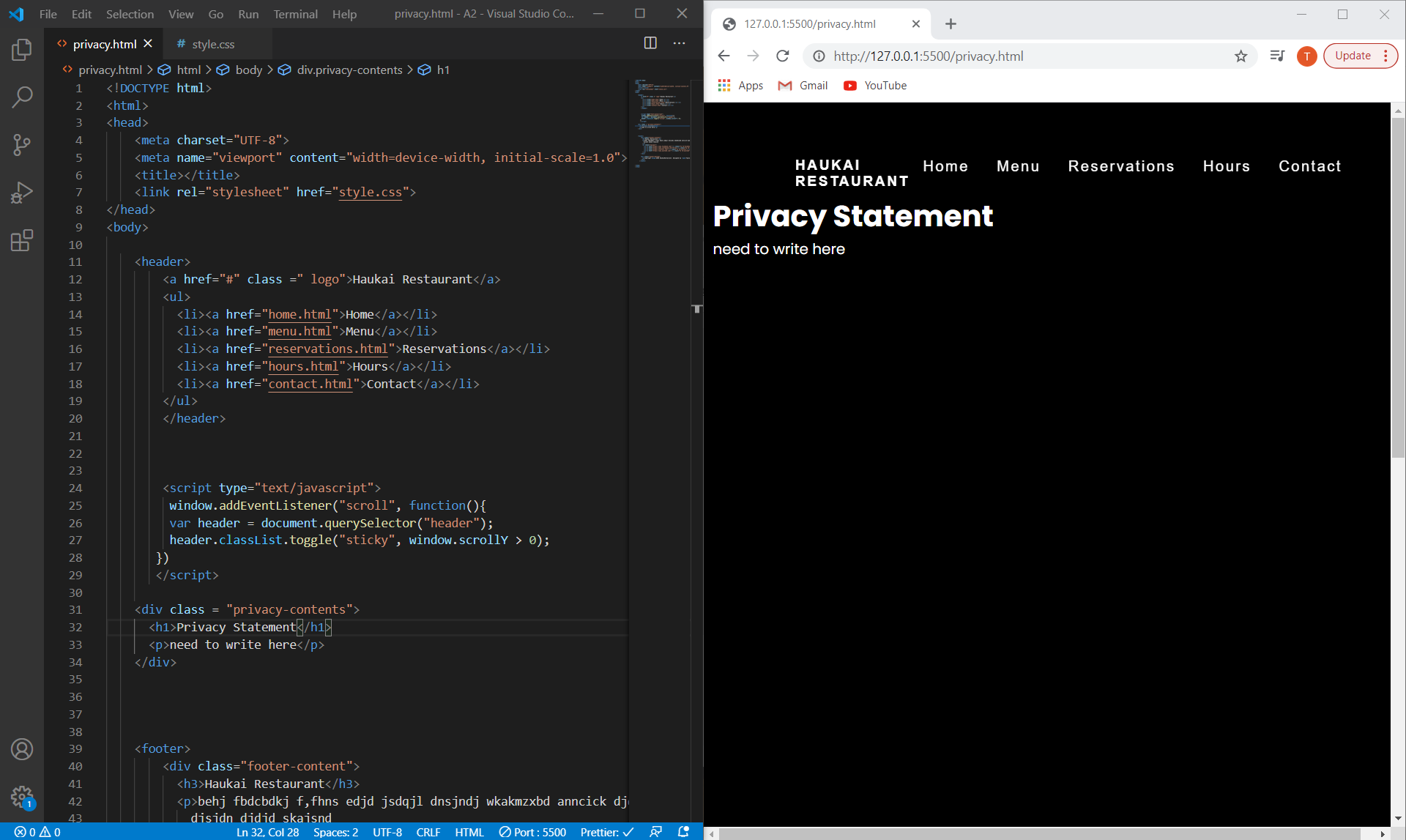
The next extension I installed is called Live Server. Live Server launches a local development server with a live reload feature for static and dynamic pages. I installed live server and tried it out myself on my own code. All I had to do was open the html code with live server and whenever I made a change to my code and wanted to preview it I saved the html code and automatically the live server would update the html page and I would be able to view the changes straight away without having to reload the page. Below I added the screenshot of how it looks when updating your code.

Overall, I think this extension is very useful. For me personally I would use it for all my projects because it saves a lot of time when you do not have to be constantly opening a browser and click the refresh button. All you need to do with the live server is hit ctrl+S to save the html code and then it automatically refreshes the browser so you can see the changed code straight away. I think it will also be very useful to use with CSS as you are constantly reloading the html page to see what size font and borders fit the best on your website.

Before:



After:



1. Using chromes development tool, I looked at every page within my website to find the networks performance and speed. Below I have put together a table for each page which shows the networks performance when it is using online, slow 3G and fast 3G.

**Home Page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 12 | 12 | 12 |
| **Transferred** | 966 kB | 966 kB | 966 kB |
| **DOMContentLoaded** | 20ms | 21ms | 20ms |
| **Loaded** | 39ms | 31ms | 29ms |
| **Finish** | 39ms | 31ms | 29ms |

Table

Description automatically generated

The home page finished loading the fastest using a fast 3G connection only taking 29ms to do so. Looking at the performance I noticed that an image took the longest to finish which was the insiderest.jpg which took 12ms to finish, but I have no concerns to fix this as it is still fast.

**Menu page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 19 | 12 | 12 |
| **Transferred** | 4.4mb | 4.4mb | 4.4mb |
| **DOMContentLoaded** | 25ms | 28ms | 20ms |
| **Loaded** | 46ms | 58ms | 45ms |
| **Finish** | 45ms | 56ms | 43ms |

Graphical user interface, application, table, Excel

Description automatically generated

The menu page took 43ms to finish and 45ms to load 19 requests on a fast 3G network. Looking at the overall network performance for the menu page it is slower than the home page because I can see that since the menu page has a lot more images and requests it took 14ms longer to finish. I can see that there is three images that look double the amount of time to load so I uploaded the same image again which seemed to make in faster.

**Reservation page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 10 | 10 | 10 |
| **Transferred** | 97.1kB | 97.1kB | 97.1kB |
| **DOMContentLoaded** | 18ms | 24ms | 18ms |
| **Loaded** | 31ms | 35ms | 29ms |
| **Finish** | 26ms | 30ms | 23ms |

Graphical user interface, application, table, Excel

Description automatically generated

The reservation page only had 10 request and took 23ms to finish and 29ms to load on a fast 3G network. Looking at the overall speed of all the request I have no concerns or problems with any as the speed for every request is great.

**Hours page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 12 | 12 | 12 |
| **Transferred** | 20.1kB | 20.1kB | 20.1kB |
| **DOMContentLoaded** | 21ms | 23ms | 24ms |
| **Loaded** | 35ms | 44ms | 50ms |
| **Finish** | 1.41s | 6.25s | 1.94s |

Graphical user interface, table

Description automatically generated

The hours page had 13 request which took 1.94s to finish and 50ms to load using a good 3G network connection. Looking at the overall speed of all the request I can see that auth is taking the longest amount of time which is under the type xhr. I am unsure of what this means and how to fix this problem to improve the overall performance and speed of this page.

**Contact page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 24 | 24 | 24 |
| **Transferred** | 276kB | 276kB | 276kB |
| **DOMContentLoaded** | 21ms | 23ms | 29ms |
| **Loaded** | 473ms | 603ms | 561ms |
| **Finish** | 5.6s | 8.31s | 6.24s |

Graphical user interface, application, table

Description automatically generated

The contact page has the most request out of my entire website with 25 request which took 561ms to load and 6.24s to finish. Looking at all the request I can see why this page has taken the longest to finish due to the map on the contact page taking nearly 4s to finish.

**Privacy page:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Online** | **Slow 3G** | **Fast 3G** |
| **Requests** | 8 | 8 | 8 |
| **Transferred** | 23.6kB | 23.6kB | 23.6kB |
| **DOMContentLoaded** | 19ms | 22ms | 18ms |
| **Loaded** | 256ms | 30ms | 27ms |
| **Finish** | 225ms | 24ms | 22ms |

Graphical user interface, application, table

Description automatically generated

The privacy page is the smallest page out of the website with only 8 request and only 27ms to load and 22ms to finish. Looking at all the request nothing seems to be taking a long time to finish so overall nothing to change to further improve the performance of this webpage.

**References:**

Parker, J., 2011. *What Is Parameterized Query?*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/4712037/what-is-parameterized-query#:~:text=A%20parameterized%20query%20(also%20known,of%20preventing%20SQL%20injection%20attacks.> [Accessed 26 December 2020].

Xiao, X., 2008. *Content Delivery Network - An Overview | Sciencedirect Topics*. [online] Sciencedirect.com. Available at: <https://www.sciencedirect.com/topics/computer-science/content-delivery-network#:~:text=CDN,places%20before%20it%20is%20requested.> [Accessed 27 December 2020].

MDN Web Docs. 2020. *HTTP Caching*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching \> [Accessed 27 December 2020].