Comp1021

Coursework 2 – Written Report

Jacob Holland (sc15j3h/200933619)

Note: This coursework uses the "Materialize" framework as an alternative to the Bootstrap framework. This decision was authorized by Sam Wilson.

Design Plans

Due to the limitations of the first coursework, I was very limited in terms of what I could technically achieve. The lack of JavaScript made it difficult to do things when an event was triggered, which caused limited functionality with some of my elements.

I was relatively content with the overall design of my first website, so I will be trying to keep true to my layout as much as possible. As my design was inspired by Google's "Material Design" I thought it would be more appropriate to use this as my framework to help preserve my vision.

With the availability of JavaScript, the first thing I will make functional is my form on my contact page. I will first put in form validation, to prevent the user from submitting a form without the required contact information, which in a professional setting, would make responding to the query a lot easier. The submit button will also work when submitting a form as well, but as PHP is not allowed/I don't know how to use it yet I will create a notification that will pop up with an onclick method. In a real world scenario, a PHP script would be used to send the form to a specific e-mail address.

I will also include a function that acquires the user's geolocation and provides navigation directions to a specific location using Google Maps. I will have it launch the google maps website instead of providing an iframe in the site, as I believe it is more functionally useful to have access to all of Google Maps' options. I will include a video to a random YouTube video in an iframe on my homepage, like my previous coursework.

Another thing I will modify will be the portfolio pages, specifically the showcase. I would rather have the screenshots take centre stage, with the descriptions being available either via a hover or a click. Using Materialize, elements like this should already exist and be easy to implement. Changing the layout to a format like this benefits the overall aesthetic, and lets the content take centre stage, promoting user interaction if they wish to know more about the content in question.

Finally, I will make improvements to the navigation bar. Having access to the Materialize framework will make it easier to implement a dropdown menu without as many formatting issues as the previous iteration as JQuery will now be used to initialise it instead of li:hover. In addition to this, it will allow me to implement a secondary navigation bar that will be used instead of the normal one on smaller-screened devices, which increases the usability of the website and allows a larger demographic of consumers to visit the website, as a good majority of users browser the internet on mobile devices.

Evaluation/Justification

Upon completion of my website, I made some changes to the design and functionality to some of it.

The first noticeable aesthetic change would be to the navigation bar. Instead of having the logo change colour on each page, the entire navbar changes colour. This allows for a greater amount of contrast between the content and the navigation, and generally looks more appealing. The drop box is now activated by a click rather than a hover, which makes it easier to use on something such as a touchscreen. Hover elements don't always typically revert back to their original state on a touch interface, causing problems with layout and interaction.

Most pages have also received a 'splash-screen' type of appearance upon launch. A full screen image with the main header will first greet the user upon arrival. This is mainly a design decision, rather than a functionality one. As not every user will have the initiative to scroll down past the image, I have included a button that scrolls down to the appropriate content. This could have been easily done using ids in HTML and linking a button to that ID, but the motion was jerky and sudden. Using JavaScript, I have implemented a function that smoothly scrolls down to the content, creating a more pleasurable experience to the user.

On the contact page, I did implement what I initially set out to do. The submit/send button allows a message to appear when a form has been successfully been filled out. However, I resorted to using form validation using HTML5, as it was simpler to implement and made it easier to impose character limits in text boxes. However I have also implemented a character counter in the textarea box of the contact form. This imposes a limit on the user on how much they can type, but it would be beneficial to the mail server as it helps regulates the overall size of the message, allowing more to be stored (if it was used for actual business use). Given the opportunity, I would like to make a PHP script that would send the contents of the form to a test e-mail address.

The use of the Materialize framework allows an easier modification of the layout of the site. Materialize adopts a 12 column layout, with the ability to push elements a certain amount of columns across. This framework allows the programmer to choose how many columns a specific element takes up on each screen size. For example, an information element may only take up 6 columns on a large display, but it may take up 12 on a smaller device. This allows complete control of the layout, allowing for easier implementation of a responsive design whilst maintaining aesthetic appeal.

As proposed, I modified the design of the portfolio pages. The content is now presented as cards that reveal more information when interacted with. This decision has allowed more content to be displayed in the same area, whilst not compromising on content delivery.

Other Additions I have included is an iframe on the homepage that allows content from an external site to be displayed in my website. I have also implemented a rollover event on my image on my about page. These features don't really enhance my website. They are more there for entertainment purposes and proof of concept. I have also included meta tags, which would be beneficial for the website, as its keywords would allow it to be indexed more efficiently with a search engine (such as Google or Bing). There is also a print style in the CSS, which allows for easy printing of the 'important' content without using too much paper or ink.

Test Cases

What I'm Testing	How I'm Testing	Predicted	Result
Cross-Browser compatibility	Open instances of my website in Chrome, Firefox, Edge and IE	The website should display in a similar/identical manner in all browsers.	Works as expected in Edge, Firefox and Chrome. IE does not display/respond correctly
Form Validation	Leave all input fields blank and press "submit", then fill every field in one by one, clicking "submit" each time	The form should stop you from executing the submit script and prompt you to fill in a field	As expected.
Responsive Layout	Turn on "device mode" in chrome and resize the screen for a typical desktop, iPad and mobile device	The website should move/resize its elements accordingly and display an alternative menu on small to medium screen sizes	As expected
Printing	Pull up a print preview for an A4 piece of paper	The website should display the content of the website, whilst excluding the navbar and the footer, as well as the parallax image.	As expected in Google Chrome. Firefox will only show blank pages. Reason uncertain…
Geolocation	Press the "Get directions" to me and allow geolocation	Google maps should open with directions to the postcode LS6 2QF	As Expected.
Smooth scrolling	Click on the "Go to content"	The webpage should smoothly scroll down to the "content" id specified in its HTML file.	As expected.

Architecture of the Web

The web is comprised of an interconnection of documents on available on the internet. The web is essentially a group of individual web pages that form together to make a website. These web pages are generally linked together to allow the user to navigate all areas of the site that they can access. Although these pages are generally of the HTTP file format, any document type can be linked.

Nowadays, the web uses what is known as a "client-server" model. The client is typically the application running on the user's terminal, which sends requests for a page (and its corresponding elements) to be downloaded to their machine from the server. A Web server's sole purpose is to provide web pages requested by the client. In this model, there are also

automated servers, none as bots. These request web pages to perform data mining or indexing (so they can be used in search engines). Bots will not appear available to the user.

Description of HTTP

HTTP is an application layer protocol for sharing and editing web documents. It is a request-response protocol: The client will always initialize the request for a resource from the server and the server will respond by either providing the source, along with a response code. If the file is unavailable. The server must send a response code back describing why it is unable to service the request. The HTTP will define the means of how a client can request a resource. These actions are communicated via HTTP verbs. Below are some of the request type's verbs:

GET – Retrieve what information (entity) is identified by the request URI (Uniform Resource Identifier – a string of characters used to identify the specific resource on the server). If the URI is a data producing process, then the data shall be returned in the form of an entity

The semantics of the GET method can change to a "conditional GET" if the request has an "if-modified-since" flag. A conditional GET will request that the entity only be transferred under the circumstances described the conditional header fields. This is intended to reduce unnecessary network usage by allowing cached entities to be refreshed without requiring multiple access requests, or transferring already existing data client-side.

POST – Used to request that origin server to accept the entity enclosed in the request as a new subordinate of the resource. This is used for annotating existing resources, submitting forms, posting messages to a forum and appending a database.

The function performed is dictated by the server and is usually dependent on the request URI. The action performed by POST might not result in a resource that can be identified via URI. In this case, 200 (OK) or 204 (No content) is the appropriate response status that the server can provide the client. If a new resource has been created. 201 (Created) should be the servers response of choice, along with an entity that describes the status of the request with a reference to the new resource & a location header.

POST responses are not cacheable, unless cache control or "expires" header fields are included in the response. Alternative, a 303 response can direct the client to retrieve a cacheable resource.

PUT – The put method requests that the enclosed entity should be stored under the supplied request. If the request URI points to an existing resource, the entity in question should be considered as a modified version of the original entity.

If it does not point to an existing resource, but the URI is capable of being defined as a new resource from the client, the server can create the resource within that URI. The origin server must respond with a 201 (CREATED response) if this occurs.

If an existing resource is modified, the server should respond with codes 200 (OK) or 204 (No content), in order to indicated 'successful' completion of the clients request and close the connection.

HTML & CSS

when creating a web page, the use of HTML and CSS to create a web page provide degrees of function during its write-up. HTML is typically the foundation of any web page. It is used to describe the structure of a web page, i.e. – What each element is typically defined as, what media to include, and where certain events should be triggers (by using CSS or JQuery). HTML is also how a web page becomes a website, as the language dictates how documents are linked/related to each other in order to provide the complete experience. You can style a HTML document or even import scripts without the use of JS or CSS, but this created unneeded complexity in the code, and doesn't particularly allow code re-use,

While HTML dictates how a page is structured, CSS will dictate how that content is presented. With CSS, you can easily change how the colours, fonts and images are displayed on screen. Using CSS, you can also change how the content is laid out on a page. And have it change in response to a change in state, such as window/screen size without harming the functionality of the web site. Having one CSS file promotes code re-use, and allows for easier modification of the website as a whole once it's been correctly structured by HTML, changing code in a CSS file is a lot more efficient than modifying each style tag individually per web page, as a change in setting can be viewed site-wide almost instantly. While CSS is useful, it is dependent on HTML and its structure (including its semantics/divisions). Without it, the CSS would not be able to modify or display anything.