* You must create a gray paper that explains the HTML, CSS, and JS that you did. It should outline at least 30 different points, showing the code and explaining what it does.
* And the gray paper should include at least 5 salient points from the code (including what challenges you encountered and how you solved the problems).

HTML

1

<div class="glow">

DRONE SHOP

</div>

This is the div class used to create the glow effect

2

<div class="ripple">

<button>CLICK ME</button>

</div>

This is the div class used to create the ripple effect

3

<div class="button-container">

<a href="Home.html" class="button" style="text-decoration:none">

<p>

<svg xmlns="http://www.w3.org/2000/svg" width="100" height="100" fill="white" class="bi bi-airplane-engines-fill" viewBox="0 0 16 16">

<path d="M8 0c-.787 0-1.292.592-1.572 1.151A4.347 4.347 0 0 0 6 3v3.691l-2 1V7.5a.5.5 0 0 0-.5-.5h-1a.5.5 0 0 0-.5.5v1.191l-1.17.585A1.5 1.5 0 0 0 0 10.618V12a.5.5 0 0 0 .582.493l1.631-.272.313.937a.5.5 0 0 0 .948 0l.405-1.214 2.21-.369.375 2.253-1.318 1.318A.5.5 0 0 0 5.5 16h5a.5.5 0 0 0 .354-.854l-1.318-1.318.375-2.253 2.21.369.405 1.214a.5.5 0 0 0 .948 0l.313-.937 1.63.272A.5.5 0 0 0 16 12v-1.382a1.5 1.5 0 0 0-.83-1.342L14 8.691V7.5a.5.5 0 0 0-.5-.5h-1a.5.5 0 0 0-.5.5v.191l-2-1V3c0-.568-.14-1.271-.428-1.849C9.292.591 8.787 0 8 0Z"/>

</svg>

</p>

<div class="ripple">

<button>CLICK ME</button>

</div>

</a>

</div>

This HTML code block generates a button element with an SVG aircraft icon and a "CLICK ME" text button that has a ripple effect when pressed. The code is broken down as follows:

The div element with the class button container is the top container element.

There is an element inside the container that surrounds the button element, which includes the aircraft icon and the text button.

Because the element's href property is set to Home.html, clicking on the button will send the viewer to the Home.html website.

A p element with an SVG aircraft symbol is located next to the button. The image is created with the Bootstrap icon library, which is referenced in the code via the class property bi bi-airplane-engines-fill.

Following the p element is another div element with the class ripple. When the icon is pressed, this part generates the ripple effect.

A button element with the words "CLICK ME" is contained within the wave div.

4

<div class="gallery-container" style="display:flex; justify-content:center;">

<div class="gallery" style="width:85%; height:85%;">

<img src="images/DJI\_0180.JPG">

<img src="images/DJI\_0245.JPG">

<img src="images/DJI\_0246.JPG">

<img src="images/DJI\_0257.JPG">

<img src="images/DJI\_0263.JPG">

<img src="images/DJI\_0276.JPG">

<img src="images/DJI\_0284.JPG">

<img src="images/DJI\_0335.JPG">

<img src="images/DJI\_0463.JPG">

</div>

</div>

The outermost div with the class gallery-container is a container element that uses the justify-content: center style attribute to center the gallery and the display itself is set to flex which creates a flex container.

Another div element with the class gallery is located within the container element. This inner div contains the pictures and has a width and height of 85% to change the size of the slideshow.

Each img element within the collection div refers to an image file, and the src property defines the URL or file location of the image.

5

<div class="footer" style="font-size: 20px; color: white;">

<p>Copyright © 2023 <br> Rakib Rary, Shailesh Ghale, Djiedjom Gbonkou, Hang Lin</p>

</div>

Footer element along with group members.

6

<p style="color: white; font-size: 28px; text-align: center;">

Welcome valued customer! we have lots to offer in our online drone store.

<br>

check out our gallery above of shots caught by some of our drones.

<br>

You can purchase these drones in the shop tab above.

</p>

This paragraph <p> provides information about the website itself and products that are offered.

7

<div class="toggle-container">

<label class="toggle-button">

<input type="checkbox" id="good-toggle">

<span class="slider round" style="color:#fff; font: size 21px;"><br>Good</span>

</label>

<label class="toggle-button">

<input type="checkbox" id="cheap-toggle">

<span class="slider round" style="color:#fff; font: size 21px;"><br>Cheap</span>

</label>

<label class="toggle-button">

<input type="checkbox" id="fast-toggle">

<span class="slider round" style="color:#fff; font: size 21px;"><br>Fast</span>

</label>

</div>

This creates a set of three toggle buttons. The toggle-container class is applied to the outer div element, which serves as a container for the toggle icons. There are three toggle icons inside the container, each with a label tag that wraps around an input element with the type property set to checkbox. The label element is used to link the checkbox to its associated text title.

Each toggle button's input tag has a distinct id attribute (good-toggle, cheap-toggle, and fast-toggle) that can be used to identify it in JavaScript or CSS.

8

<img src="images/exspensive.webp" id="my-image" alt="about\_image is missing" width="500" height="300" >

This creates an image element with the source attribute set to "images/exspensive.webp", the ID attribute set to "my-image", the alt attribute set to "about\_image is missing", and the width and height attributes set to 500 and 300 respectively.

9

div class="card-container">

<form action="submit-form.php" method="post">

<div class="form-row">

<label for="name">Name</label>

<input type="text" id="name" name="name" required>

</div>

<div class="form-row">

<label for="email">Email</label>

<input type="email" id="email" name="email" required>

</div>

<div class="form-row">

<label for="phone">Phone</label>

<input type="tel" id="phone" name="phone" required>

</div>

<div class="form-row">

<label for="drone\_serial">Drone Serial #</label>

<input type="text" id="drone\_serial" name="drone\_serial" required>

</div>

<div class="form-row">

<label for="issue">Issue</label>

<textarea id="issue" name="issue" rows="4" cols="30" required></textarea>

</div>

<div class="form-row submit-button">

<input type="submit" value="Submit">

</div>

</form>

</div>

This creates a form with several input fields for users to submit information. When the user selects the "Submit" icon on the form, the action element specifies the URL where the form data should be transmitted, and the method attribute specifies the HTTP method to use. (POST in this case).

The form contains a number of div components with the class attribute set to "form-row," each with a label and an entry box. The label element explains the reason of the input field, and the for attribute associates the label with the id property of the input field for accessibility purposes.

The type attribute specifies the form of the entry areas, such as "text" for the name and drone serial number, "email" for the email, and "tel" for the phone. Each entry field receives the required attribute, which requires users to complete out the field before completing the form.

The "Issue" entry box on the form also has a textarea element, which enables users to submit numerous lines of text.

Finally, a submit icon with the type attribute set to "submit" has been added to the form. When the user selects this button, the form data is sent to the URL indicated in the action attribute via HTTP using the method defined in the method attribute. (POST in this case).

10

<div class="slidecontainer">

<input type="range" min="1" max="100" value="50">

</div>

The input element with the type attribute set to "range" is used to make a slider control. The class property is assigned to "slidecontainer," which allows CSS styles to be applied to this element.

The min attribute indicates the slider's minimum value, while the max attribute gives the highest value. The lowest value is set to 1 and the maximum value is set to 100 in this instance.

The value attribute determines the slider's starting number. In this instance, it is set to 50, indicating that the slider will be positioned midway between the minimum and maximum values at first.

The number of the slider changes automatically as the user adjusts it. This value can be retrieved with JavaScript and utilized for website activities such as updating the display of other components depending on the slider's value.

CSS

11

.glow {

font-size: 80px;

color: #fff;

text-align: center;

animation: glow 1s ease-in-out infinite alternate;

}

@keyframes glow {

from { text-shadow: 0 0 10px #fff, 0 0 20px #fff, 0 0 30px #14626c, 0 0 40px #14626c, 0 0 50px #14626c, 0 0 60px #14626c, 0 0 70px #14626c;

}

to { text-shadow: 0 0 20px #fff, 0 0 30px #24727c, 0 0 40px #24727c, 0 0 50px #24727c, 0 0 60px #24727c, 0 0 70px #24727c, 0 0 80px #24727c;

}

}

This adds a glowing effect to the class “glow”. The animation is generated with the "@keyframes" rule, which specifies the animation's behavior as time passes. The animation is named "glow," and it lasts one second, has an easing function of "ease-in-out," and continues indefinitely in an alternating manner.

The animation's "from" state makes the original text-shadow white and progressively raises the size and color of the shadow, producing a glow effect. The animation's "to" stage increases the size and color of the shadow while shifting the color to a deeper tone of blue.

This produces an aesthetically appealing effect that can be used to highlight essential text or headings on a website.

12@font-face {

font-family: 'Bitwise';

font-style: normal;

font-weight: 400;

src: local('Bitwise'), url('https://fonts.cdnfonts.com/s/6817/bitwise.woff') format('woff');

}

CSS rule for creating the 'Bitwise' unique font family. Normal style with a weight of 400 is selected. The font file is imported from an external source via the url() method, which refers to a WOFF (Web Open Font Format) web font file. If the font file is not accessible, the browser will fall back to a locally loaded variant of the font family with the same name (“Bitwise”). Using the font-family property, this rule can then be used in other CSS rules to apply the "Bitwise" font to particular HTML components.

13

li a:hover{

background-color: grey ;

box-shadow: 0 5px 15px grey;

}

This defines a hover effect for the anchor <a> tag within a list entry, <li>. When the user hovers over the anchor tag, the backdrop color of the tag transforms into gray and a gray box shadow effect emerges, giving the link an elevated appearance.

14.footer {

position:fixed;

bottom:0;

left:0;

width:100%;

height: 90px;

background-color: black;

}

This defines properties for the footer element.

15.ripple {

position: relative;

overflow: hidden;

}

.ripple button {

position: relative;

overflow: hidden;

background-color: grey;

color: #fff;

border: none;

padding: 16px 32px;

font-size: 24px;

cursor: pointer;

}

.ripple button:after {

content: "";

position: absolute;

background-color: rgba(255, 255, 255, 0.3);

border-radius: 50%;

width: 50px;

height: 50px;

transform: scale(0);

transition: all 0.5s ease-out;

top: 50%;

left: 50%;

margin-left: -25px;

margin-top: -25px;

pointer-events: none;

}

.ripple button:focus:after,

.ripple button:active:after {

transform: scale(2);

opacity: 0;

}

When a button is pressed, this generates a ripple effect. The ripple class contains the button element.

The button element itself has a dark backdrop, white lettering, no frame, and padding of 16px on the top and bottom and 32px on the left and right. It also has a 24px letter height and a cursor pointer to show that it can be clicked.

When the button is pressed, the: after pseudo-element is used to make a circular element. It has a white backdrop, an opacity of 0.3, and a border-radius of 50% to make it spherical. Its original size is set to 0, making it invisible until the button is pressed.

When the icon is pressed, the following occurs: The focus:after and active:after elements are used to set the opacity to 0 and raise the size of the pseudo-element to 2. This produces a ripple effect that radiates from the button's center and goes out as it expands.

To keep the ripple effect contained within the button element, the .ripple class is used to make a container for it and change its overflow value to hidden.

16.button-container {

width: auto;

background: black;

overflow-y: auto;

}

.button-container > a {

width: 300px;

height: 200px;

display: inline-block;

background: black;

margin: 15px;

text-align:20px 20px;

}

.button{

background-color:black;

border: 5px solid white;

color:silver;

font-size: 24px;

text-align: center;

display:inline-block;

padding:32px 32px;

}

This defines properties for button containers and buttons. The container has a dark backdrop and can be scrolled vertically if necessary. (overflow-y: auto). The buttons are presented inline-block with a black backdrop and a 5px white border and have a set width of 300px and height of 200px. The text inside the icons is centered, 24px in height, and styled in silver. To implement this styling, the class "button" is given to the icons.

17.slider {

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: #ccc;

border-radius: 34px;

cursor: pointer;

transition: .4s;

}

.slider:before {

position: absolute;

content: "";

height: 26px;

width: 26px;

left: 4px;

bottom: 4px;

background-color: white;

border-radius: 50%;

transition: .4s;

}

input:checked + .slider {

background-color: #2196F3;

}

input:focus + .slider {

box-shadow: 0 0 1px #2196F3;

}

input:checked + .slider:before {

transform: translateX(115px);

}

.round {

border-radius: 34px;

}

The toggle switch's general look is defined by the .slider class. It is completely placed and occupies all of the length and height of its parent element. The backdrop shade is fixed to a light gray with a border-radius of 34px. To show that it is clickable, the cursor is assigned to pointer. When the slider is selected or engaged, a transition effect is also specified.

When the slider is toggled, the :before pseudo-element is used to make the circular icon that moves. It has a height and width of 26px and is positioned at the bottom left of the scroll. It has a white backdrop and a 50% edge radius. When the slider is selected or active, a transition effect is also specified.

When the slider is toggled or focused, the input:checked and input:focus filters are used to add styles to it. The input:checked +.slider selection changes the slider's background color to blue. The input:focus +.slider selection gives the slider a box shadow.

The.round class is used to give the slider and circle icon a rounded look by applying a border-radius of 34px to them.

18img {

display: block;

margin: auto;

}

#my-image {

filter: blur(20px);

opacity: 1;

transition: filter 1s, opacity 1s;

}

The first piece of code aligns any img element by changing its display and margin properties to block and auto, respectively.

The second piece of code uses the filter property with the blur function to apply a blur effect to a picture with the ID of my-image. It also changes the image's opacity to 1 and applies a 1-second transition effect to the filter and opacity attributes.

This code helps create a visual effect in which a picture progressively becomes sharper and less blurry as a user hovers over it.

19.form-row {

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

margin-bottom: 10px;

}

.form-row label {

flex-basis: 30%;

font-weight: bold;

}

.form-row input,

.form-row textarea {

flex: 1;

padding: 5px;

border-radius: 5px;

border: none;

border-bottom: 2px solid black;

}

.form-row input:focus,

.form-row textarea:focus {

outline: none;

border-bottom: 2px solid black;

}

.form-row - This class styles a container that holds a label and an input element or a textarea element.

display: flex; - This property sets the display of the container as a flex container, allowing you to use flexbox properties to style the contents of the row.

flex-direction: row; - This property sets the direction of the flex container as a row, meaning that the elements will be displayed horizontally.

justify-content: space-between; - This property aligns the elements in the row along the main axis, which is the horizontal axis in this case. space-between value will distribute the space between the items in the row, with the first item flush left, the last item flush right, and any remaining space evenly distributed between the items.

align-items: center; - This property aligns the elements in the row along the cross axis, which is the vertical axis in this case. center value aligns the items in the center of the container.

margin-bottom: 10px; - This property sets a margin at the bottom of each row to create some space between the rows.

.form-row label - This class styles the label element that appears in each row.

flex-basis: 30%; - This property sets the initial size of the label element to take up 30% of the available space in the row.

font-weight: bold; - This property sets the font weight of the label text to bold.

.form-row input and .form-row textarea - These classes style the input and textarea elements that appear in each row.

flex: 1; - This property sets the input or textarea element to take up the remaining available space in the row.

padding: 5px; - This property adds some padding to the input or textarea element.

border-radius: 5px; - This property adds rounded corners to the input or textarea element.

border: none; - This property removes any border from the input or textarea element.

border-bottom: 2px solid black; - This property adds a solid black border to the bottom of the input or textarea element.

.form-row input:focus and .form-row textarea:focus - These classes style the input and textarea elements when they are in focus (when the user clicks on them).

outline: none; - This property removes the default focus outline from the input or textarea element.

border-bottom: 2px solid black; - This property adds a solid black border to the bottom of the input or textarea element when it is in focus.

20slidecontainer {

display: flex;

justify-content: center;

align-items: center;

width: 100%;

margin-top: 50px;

}

.slider {

-webkit-appearance: none;

width: 90%;

height: 25px;

background: #d3d3d3;

outline: none;

opacity: 0.7;

-webkit-transition: .2s;

transition: opacity .2s;

}

.slider:hover {

opacity: 1;

}

.slider::-webkit-slider-thumb {

-webkit-appearance: none;

appearance: none;

width: 25px;

height: 25px;

background: #04AA6D;

cursor: pointer;

}

.slider::-moz-range-thumb {

width: 25px;

height: 25px;

background: #04AA6D;

cursor: pointer;

}

The slidecontainer class specifies a sliding container element. Because of the display: flex attribute, its sibling components align horizontally. The justify-content: center and align-items: center attributes center the sibling components in the container horizontally and vertically.

The .slider class defines the appearance of the slider element. The -webkit-appearance property is used to remove the default styling of the slider in Safari and Chrome. The width and height properties set the dimensions of the slider, and the background property sets its background color. The outline property sets the outline style of the slider, and the opacity property sets its transparency.

The transition property is used to add a transition effect when the opacity property of the slider changes. The :hover pseudo-class is used to increase the opacity of the slider when the mouse is over it.

The ::-webkit-slider-thumb and ::-moz-range-thumb pseudo-elements define the appearance of the slider's thumb (the part of the slider that the user can drag). These properties set the dimensions and background color of the thumb and define its cursor type when hovered over.

JS

21

const buttons = document.querySelectorAll('.ripple button');

buttons.forEach(button => {

button.addEventListener('click', function(e) {

const x = e.clientX - e.target.offsetLeft;

const y = e.clientY - e.target.offsetTop;

const circle = document.createElement('span');

circle.classList.add('ripple-circle');

circle.style.left = `${x}px`;

circle.style.top = `${y}px`;

this.appendChild(circle);

setTimeout(() => {

circle.remove();

}, 500);

});

});

This adds a ripple effect to buttons when clicked. It first selects all the button elements with the class ripple using the document.querySelectorAll method and then adds a click event listener to each of them using the forEach method.

When a button is clicked, it calculates the coordinates of the click event relative to the top-left corner of the button using the clientX and clientY properties of the event object and subtracts the offsetLeft and offsetTop properties of the button to account for any padding or borders.

Then, it creates a new span element with the class ripple-circle and sets its left and top CSS properties to the calculated coordinates. It appends this new element to the clicked button using the appendChild method.

Finally, it sets a timeout of 500 milliseconds using the setTimeout method to remove the span element from the button after the ripple effect has finished.

22

$(document).ready(function(){

$('.gallery').slick({

autoplay: true,

autoplaySpeed: 2000,

dots: true,

infinite: true,

speed: 500,

slidesToShow: 1,

slidesToScroll: 1

});

});

This is jQuery code for initializing a slideshow using the Slick carousel plugin. The $(document).ready() function is used to make sure the DOM is fully loaded before the code is executed.

The $('.gallery') selector targets the HTML element with the class gallery, which is assumed to be a container for the slideshow.

The .slick() method is called on the selected element, and several options are passed as an object to configure the slideshow. These options include:

autoplay: A boolean value to enable or disable the slideshow autoplay.

autoplaySpeed: The time interval between each slide transition, in milliseconds.

dots: A boolean value to enable or disable the navigation dots.

infinite: A boolean value to enable or disable infinite looping of the slideshow.

speed: The animation speed of the slide transition in milliseconds.

slidesToShow: The number of slides to display at once.

slidesToScroll: The number of slides to scroll at once.

The Slick Carousel plugin enhances the behavior and appearance of the slideshow, adding features like responsive design, touch-swipe support, and more.

23

window.addEventListener('load', function() {

var img = document.getElementById('my-image');

img.addEventListener('load', function() {

img.style.filter = 'none';

img.style.opacity = '1';

});

});

This JS code listens for the load event of the window and then finds an image with the ID my-image. When the load event of the image is triggered, it sets the CSS filter property to none and the opacity property to 1, which removes the blur and makes the image fully visible. This is useful for cases where an image is initially blurred or has reduced opacity, but then should be fully visible once it's fully loaded.

24canvas.addEventListener('mousedown', (e) => {

isPressed = true

x = e.offsetX

y = e.offsetY

})

document.addEventListener('mouseup', (e) => {

isPressed = false

x = undefined

y = undefined

})

canvas.addEventListener('mousemove', (e) => {

if(isPressed) {

const x2 = e.offsetX

const y2 = e.offsetY

drawCircle(x2, y2)

drawLine(x, y, x2, y2)

x = x2

y = y2

}

})

The addEventListener() method is used to attach event handlers to the canvas element for mouse events: mousedown, mouseup, and mousemove.

When the mousedown event is triggered (i.e. when the user clicks on the canvas), the isPressed variable is set to true to indicate that the mouse button is being held down, and the x and y variables are set to the current mouse coordinates using the offsetX and offsetY properties of the event object.

When the mouseup event is triggered (i.e. when the user releases the mouse button), the isPressed variable is set to false to indicate that the mouse button is no longer being held down, and the x and y variables are reset to undefined.

When the mousemove event is triggered (i.e. when the user moves the mouse), the code inside the if(isPressed) block is executed if the isPressed variable is true (i.e., if the mouse button is being held down). This code draws a circle at the current mouse coordinates using the drawCircle() function (which is not shown in this code snippet) and draws a line between the previous mouse coordinates and the current mouse coordinates using the drawLine() function (which is also not shown in this code snippet).

Finally, the x and y variables are updated to the current mouse coordinates, so that the next time the mousemove event is triggered, the line will be drawn from the new coordinates to the next set of coordinates.

25

function drawCircle(x, y) {

ctx.beginPath();

ctx.arc(x, y, size, 0, Math.PI \* 2)

ctx.fillStyle = color

ctx.fill()

}

function drawLine(x1, y1, x2, y2) {

ctx.beginPath()

ctx.moveTo(x1, y1)

ctx.lineTo(x2, y2)

ctx.strokeStyle = color

ctx.lineWidth = size \* 2

ctx.stroke()

}

function updateSizeOnScreen() {

sizeEL.innerText = size}

These are the three functions used in the drawing application:

drawCircle(x, y) draws a circle at the specified coordinates (x, y) on the canvas. It uses the ctx.beginPath() method to begin drawing a new path, ctx.arc(x, y, size, 0, Math.PI \* 2) to draw the circle with the given radius size, ctx.fillStyle = color to set the fill color to the current color, and ctx.fill() to fill the circle with the current fill color.

drawLine(x1, y1, x2, y2) draws a line from the point (x1, y1) to the point (x2, y2) on the canvas. It uses the ctx.beginPath() method to begin drawing a new path, ctx.moveTo(x1, y1) to move the current point to the starting point of the line, ctx.lineTo(x2, y2) to draw the line to the ending point, ctx.strokeStyle = color to set the stroke color to the current color, ctx.lineWidth = size \* 2 to set the line width to twice the current size, and ctx.stroke() to stroke the line with the current stroke color and line width.

updateSizeOnScreen() updates the text displayed on the screen to show the current brush size. It sets the innerText property of the sizeEL element to the value of the size variable.

5 SALIENT POINTS

26

27

28

29

30