HTML CSS JS Examples

Creating a basic HTML document

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
</head>
<body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage.</p>
</body>
</html>
```

This is a basic HTML document that includes a title, heading, and paragraph. The <!DOCTYPE html> declaration tells the browser that this is an HTML5 document. The <html> element is the root element of the document and contains all other elements. The <head> element contains metadata about the document, such as the title, and is not displayed in the browser. The <body> element contains the visible content of the document.

Creating a stylesheet with CSS

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
    <style>
        h1 {
            color: red;
            font-size: 36px;
        }
        p {
            color: blue;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage.</p>
</body>
</html>
```

This example adds a stylesheet to the previous HTML document. The <style> element is used to define CSS rules for the document. In this example, the h1 element is styled with red text and a font size of 36 pixels, while the p element is styled with blue text and a font size of 18 pixels.

Adding an image with HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
</head>
<body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage.</p>
    <img src="myimage.jpg" alt="My Image">
</body>
</html>
```

This example adds an image to the HTML document using the <img> element. The src attribute specifies the URL of the image file, while the alt attribute provides alternative text for the image, which is displayed if the image cannot be loaded.

Creating a navigation bar with CSS

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
    <style>
        nav {
            background-color: #333;
            overflow: hidden;
        }
```

```
        nav a {
            float: left;
            color: white;
            text-align: center;
            padding: 14px 16px;
            text-decoration: none;
        }
        nav a:hover {
            background-color: #ddd;
            color: black;
        }
    </style>
</head>
<body>
    <nav>
        <a href="#">Home</a>
        <a href="#">About</a>
        <a href="#">Contact</a>
    </nav>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage.</p>
</body>
</html>
```

This example creates a navigation bar using CSS. The <nav> element is used to contain the navigation links. The background-color property sets the background color of the navigation bar, while the overflow property is used to clear any floats inside the <nav> element. The <a> element is used to define the links, and the float property is used to align the links horizontally. The padding property sets the space around the link

text, while the text-decoration property removes the underline from the links.

Creating a dropdown menu with CSS and JavaScript

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Webpage</title>
    <style>
        nav {
            background-color: #333;
            overflow: hidden;
        }
        nav a {
            float: left;
            color: white;
            text-align: center;
            padding: 14px 16px;
            text-decoration: none;
        }
        nav a:hover {
            background-color: #ddd;
            color: black;
        }
        .dropdown {
            float: left;
            overflow: hidden;
        }
```

```
        .dropdown .dropbtn {
            font-size: 16px;
            border: none;
            outline: none;
            color: white;
            padding: 14px 16px;
            background-color: inherit;
            font-family: inherit;
            margin: 0;
        }
        .dropdown-content {
            display: none;
            position: absolute;
            background-color: #f9f9f9;
            min-width: 160px;
            box-shadow: 0px 8px 16px 0px
rgba(0,0,0,0.2);
            z-index: 1;
        }
        .dropdown-content a {
            float: none;
            color: black;
            padding: 12px 16px;
            text-decoration: none;
            display: block;
            text-align: left;
        }
        .dropdown-content a:hover {
            background-color: #ddd;
        }
        .dropdown:hover .dropdown-content {
            display: block;
```

```
        }
    </style>
</head>
<body>
    <nav>
        <a href="#">Home</a>
        <div class="dropdown">
            <button class="dropbtn">Dropdown</button>
            <div class="dropdown-content">
                <a href="#">Link 1</a>
                <a href="#">Link 2</a>
                <a href="#">Link 3</a>
            </div>
        </div>
        <a href="#">About</a>
        <a href="#">Contact</a>
    </nav>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage.</p>
    <script>
        // JavaScript code to toggle the dropdown menu
        let dropdown =
document.getElementsByClassName("dropdown");
        let i;
        for (i = 0; i < dropdown.length; i++) {
            dropdown[i].addEventListener("click",
function() {
                this.classList.toggle("active");
                let dropdownContent =
this.nextElementSibling;
                if (dropdownContent.style.display ===
"block") {
```

```
                    dropdownContent.style.display =
"none";
                } else {
                    dropdownContent.style.display =
"block";
                }
            });
        }
    </script>
</body>
</html>
```

This example adds a dropdown menu to the navigation bar using CSS and JavaScript. The dropdown class is used to define the dropdown container, while the dropbtn class is used for the dropdown button. The dropdown-content class is used for the dropdown menu items. The JavaScript code adds an event listener to toggle the display of the dropdown menu when the dropdown button is clicked.

Simple HTML Page:

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Web Page</title>
</head>
<body>
    <h1>Hello World!</h1>
```

```
    <p>This is my first web page.</p>
</body>
</html>
```

This is a simple HTML page that displays a heading and a paragraph of text. The <!DOCTYPE html> declaration specifies that the document is an HTML5 document, and the <html> element is the root element of the document. The <head> element contains information about the document, such as the title of the page, and the <body> element contains the content of the page.

CSS Styling:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Example</title>
    <style>
        h1 {
            color: red;
            font-size: 24px;
        }
        p {
            color: blue;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1>Hello World!</h1>
```

```
    <p>This is my first web page.</p>
</body>
</html>
```

This is an HTML page with CSS styling applied to the heading and paragraph elements. The CSS is defined in the <style> element in the <head> section of the document. The h1 selector targets the heading element and sets its color to red and font size to 24 pixels. The p selector targets the paragraph element and sets its color to blue and font size to 16 pixels.

JavaScript Alert Box:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Example</title>
    <script>
        alert("Hello World!");
    </script>
</head>
<body>
    <h1>Hello World!</h1>
    <p>This is my first web page.</p>
</body>
</html>
```

This is an HTML page with a JavaScript alert box that displays a message when the page loads. The alert() function is called in a <script> element in the <head> section of the document. When

the page is loaded, the alert box displays the message "Hello World!".

JavaScript Event Handling:

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Example</title>
    <script>
        function showMessage() {
            alert("Button clicked!");
        }
    </script>
</head>
<body>
    <h1>Hello World!</h1>
    <button onclick="showMessage()">Click me</button>
</body>
</html>
```

This is an HTML page with a button element that triggers a JavaScript function when clicked. The showMessage() function is defined in a <script> element in the <head> section of the document. When the button is clicked, the function is called and an alert box is displayed with the message "Button clicked!".

CSS Box Model:

```
<!DOCTYPE html>
<html>
<head>
    <title>CSS Example</title>
    <style>
        div {
            border: 1px solid black;
            padding: 10px;
            margin: 10px;
            width: 200px;
            height: 200px;
        }
    </style>
</head>
<body>
    <div>This is a box</div>
</body>
</html>
```

This is an HTML page with a div element styled using CSS box model properties. The div selector sets a black border around the div element with a thickness of 1 pixel, adds 10 pixels of padding inside the border, and 10 pixels of margin outside the border.

HTML: Creating a basic web page

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Web Page</title>
```

```
</head>
<body>
    <h1>Welcome to my website</h1>
    <p>This is my first web page.</p>
</body>
</html>
```

This is a basic HTML code for creating a web page. The <!DOCTYPE html> declaration indicates that this is an HTML5 document. The <html> element encloses the entire web page content, while the <head> element contains meta information about the page, including the title which appears in the browser tab. The <body> element contains the actual content of the web page, in this case, a header (<h1>) and a paragraph (<p>).

CSS: Styling a web page

```
body {
    background-color: #f0f0f0;
    font-family: Arial, sans-serif;
}

h1 {
    color: #333;
    font-size: 2em;
    text-align: center;
}

p {
    color: #666;
```

```
    font-size: 1.2em;
}
```

This CSS code adds some styling to the previous HTML example. It sets the background color of the page to light gray (#f0f0f0) and the font family to Arial or sans-serif. It also styles the header (<h1>) to be centered, with a font size of 2em and a color of dark gray (#333), while the paragraph (<p>) has a font size of 1.2em and a color of gray (#666).

JavaScript: Displaying a message

```
alert("Hello, World!");
```

This JavaScript code displays an alert box with the message "Hello, World!". When the user clicks the OK button, the alert box will disappear.

HTML: Adding an image

```
<img src="https://www.example.com/image.jpg"
alt="Example Image">
```

This HTML code adds an image to the web page. The src attribute specifies the URL of the image file, while the alt attribute provides alternative text for the image in case it cannot be displayed.

CSS: Creating a navigation bar

```
nav {
    background-color: #333;
}

nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
}

nav li {
    float: left;
}

nav a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
```

This CSS code creates a simple navigation bar. The background color of the navigation bar is set to dark gray (#333). The ul and li elements are used to create an unordered list of links. The a elements are styled to be block-level elements with white text, centered, and with some padding.

JavaScript: Getting user input

```
let name = prompt("What is your name?");
alert("Hello, " + name + "!");
```

This JavaScript code prompts the user to enter their name and stores the result in a variable called name. It then displays an alert box with a personalized greeting that includes the user's name.

HTML Example - Creating a simple web page structure:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Webpage</title>
</head>
<body>
  <h1>Welcome to my webpage!</h1>
  <p>Here's some content for my page.</p>
</body>
</html>
```

This example demonstrates the basic structure of an HTML document. The <!DOCTYPE html> declaration tells the browser that this is an HTML5 document. The <html> element contains the entire document, while the <head> element contains information about the document, such as the page title. The

<body> element contains the visible content of the page, such as headings and paragraphs.

CSS Example - Changing font color:

```
p {
  color: red;
}
```

This example demonstrates how to change the color of text using CSS. In this case, we're targeting all <p> elements and setting their color to red. The color property is a CSS property that specifies the color of the text.

JavaScript Example - Creating an alert box:

```
alert("Hello, world!");
```

This example demonstrates how to create an alert box using JavaScript. When this code is executed, it will display an alert box with the message "Hello, world!".

HTML Example - Creating a link:

```
<a href="https://www.example.com">Visit Example.com</a>
```

This example demonstrates how to create a hyperlink using HTML. The <a> element is used to create a link, and the href attribute specifies the URL that the link should point to. The text between the opening and closing <a> tags is what the user sees as the link text.

CSS Example - Changing background color:

```
body {
  background-color: #f0f0f0;
}
```

This example demonstrates how to change the background color of a web page using CSS. In this case, we're targeting the <body> element and setting its background color to a light gray color.

JavaScript Example - Creating a variable:

```
let x = 5;
```

This example demonstrates how to create a variable using JavaScript. The let keyword is used to declare a variable, and the variable name (x) is followed by an equals sign and the variable's initial value (5 in this case).

HTML Example - Adding an image:

```
<img src="example.jpg" alt="Example image">
```

This example demonstrates how to add an image to a web page using HTML. The <img> element is used to add an image, and the src attribute specifies the URL of the image file. The alt attribute provides alternative text for the image, which is displayed if the image cannot be loaded.

CSS Example - Changing font size:

```
p {
  font-size: 18px;
}
```

This example demonstrates how to change the font size of text using CSS. In this case, we're targeting all <p> elements and setting their font size to 18px.

JavaScript Example - Using an if statement:

```
let x = 5;
if (x > 10) {
  alert("x is greater than 10");
} else {
  alert("x is less than or equal to 10");
}
```

This example demonstrates how to use an if statement in JavaScript. The code checks the value of the variable x, and if it's greater than 10, an alert box displays the message "x is greater than 10". If x is less than or equal

Example of HTML code:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
</head>
<body>
    <h1>Welcome to my website!</h1>
    <p>This is my first website. I hope you enjoy
it!</p>
</body>
</html>
```

This is a simple HTML document that displays a heading and paragraph. The <!DOCTYPE html> tag specifies the version of HTML that is being used, and the <html> tag encloses the entire document. The <head> tag contains metadata about the page, such as the title of the page. The <body> tag contains the visible content of the page, which in this case includes a heading and a paragraph.

Example of CSS code:

```
h1 {
    color: blue;
    font-size: 36px;
}
```

This is a CSS style rule that targets all <h1> elements and sets their color to blue and font size to 36 pixels. The color property sets the color of the text, while the font-size property sets the size of the font.

Example of JavaScript code:

```
let myVariable = "Hello, world!";
console.log(myVariable);
```

This is a simple JavaScript program that creates a variable called myVariable and assigns it the value "Hello, world!". The console.log() function is used to print the value of myVariable to the browser console.

Example of an HTML form:

```
<form>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br>
    <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email"><br>
    <label for="message">Message:</label>
    <textarea id="message"
name="message"></textarea><br>
    <input type="submit" value="Submit">
</form>
```

This is an HTML form that allows users to submit their name, email, and a message. The label tags provide labels for each input field, and the input and textarea tags create the fields themselves. The type attribute of the input tag specifies the type of field (text or email), and the id and name attributes are used to identify each field. The submit input type creates a button that submits the form when clicked.

Example of CSS grid layout:

```
.grid-container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 10px;
}
```

This is a CSS style rule that creates a grid layout with three columns that are each one-third of the available width, and a gap of 10 pixels between each cell. The display: grid property specifies that this element should use the grid layout.

Example of JavaScript function:

```
function addNumbers(a, b) {
    return a + b;
}

let result = addNumbers(5, 7);
console.log(result);
```

This is a JavaScript function that takes two parameters (a and b) and returns their sum. The let keyword is used to declare a variable called result, which is assigned the result of calling the addNumbers function with arguments 5 and 7. The console.log() function is used to print the value of result to the browser console.

HTML: Creating a basic webpage structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage using HTML.</p>
  </body>
</html>
```

Explanation: This code creates a basic webpage structure with a title, a heading, and a paragraph.

CSS: Styling a webpage

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      body {
        background-color: lightblue;
      }
      h1 {
        color: white;
        text-align: center;
      }
    </style>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage using HTML and CSS.</p>
  </body>
</html>
```

Explanation: This code adds CSS styles to the webpage, setting the background color of the body to light blue and the color of the heading to white, and centering the text of the heading.

JavaScript: Displaying an alert message

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <p>This is my first webpage using HTML, CSS, and
JavaScript.</p>
    <button onclick="alert('Hello World!')">Click
me</button>
  </body>
</html>
```

Explanation: This code creates a button that, when clicked, displays an alert message with the text "Hello World!"

HTML: Adding an image to a webpage

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <img src="image.jpg" alt="A beautiful image">
  </body>
```

```
</html>
```

Explanation: This code adds an image to the webpage with the source file "image.jpg" and an alternative text description "A beautiful image".

CSS: Adding a border to an element

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      img {
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <img src="image.jpg" alt="A beautiful image">
  </body>
</html>
```

Explanation: This code adds a 1 pixel black border around the image on the webpage.

JavaScript: Changing the text of an element

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <script>
      function changeText() {
        document.getElementById("text").innerHTML =
"New text!";
      }
    </script>
  </head>
  <body>
    <h1>Welcome to my website!</h1>
    <p id="text">This is the original text.</p>
    <button onclick="changeText()">Change text</button>
  </body>
</html>
```

Explanation: This code creates a button that, when clicked, changes the text of the paragraph element with the ID "text" to "New text!".

HTML/CSS: Creating a responsive navigation bar

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
```

```
/* CSS for navigation bar */
.navbar {
  overflow: hidden;
  background-color: #333;
  font-family: Arial, sans-serif;
}
.navbar a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
.navbar a:hover {
  background-color: #ddd;
  color: black;
}
.navbar a.active {
  background-color: #4CAF50;
  color: white;
}
/* CSS for responsive navigation bar */
.navbar.responsive a.icon {
  position: absolute;
  right: 0;
  top: 0;
}
.navbar.responsive a {
  float: none;
  display: block;
  text-align: left;
```

```
      }
    </style>
    <script>
      // JavaScript for responsive navigation bar
      function toggleNavbar() {
        let navbar = document.getElementById("navbar");
        if (navbar.className === "navbar") {
          navbar.className += " responsive";
        } else {
          navbar.className = "navbar";
        }
      }
    </script>
  </head>
  <body>
    <!-- HTML for navigation bar -->
    <div class="navbar" id="navbar">
      <a href="#" class="active">Home</a>
      <a href="#">About</a>
      <a href="#">Contact</a>
      <a href="javascript:void(0);" class="icon"
onclick="toggleNavbar()">
        &#9776;
      </a>
    </div>
    <h1>Welcome to my website!</h1>
    <p>This is a responsive navigation bar using HTML,
CSS, and JavaScript.</p>
  </body>
</html>
```

Explanation: This code creates a responsive navigation bar using HTML, CSS, and JavaScript. The navigation bar has links to different pages and a toggle icon for smaller screens. When the toggle icon is clicked, the navigation bar becomes responsive and can be expanded or collapsed.

CSS: Creating a card layout

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      /* CSS for card layout */
      .card {
        box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
        transition: 0.3s;
        width: 300px;
        margin: 20px;
        display: inline-block;
        border-radius: 5px;
      }
      .card:hover {
        box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);
      }
      .card img {
        width: 100%;
        border-radius: 5px 5px 0 0;
      }
      .card-container {
```

```
        padding: 2px 16px;
      }
    </style>
  </head>
  <body>
    <!-- HTML for card layout -->
    <div class="card">
      <img src="image1.jpg" alt="Image 1">
      <div class="card-container">
        <h4><b>Image 1</b></h4>
        <p>Description of Image 1.</p>
      </div>
    </div>
    <div class="card">
      <img src="image2.jpg" alt="Image 2">
      <div class="card-container">
        <h4><b>Image 2</b></h4>
        <p>Description of Image 2.</p>
      </div>
    </div>
    <div class="card">
      <img src="image3.jpg" alt="Image 3">
      <div class="card-container">
        <h4><b>Image 3</b></h4>
        <p>Description of Image 3.</p>
      </div>
    </div>
  </body>
</html>
```

Explanation: This code creates a card layout using CSS. The layout displays multiple cards with images and descriptions. The

cards have a box-shadow effect on hover and a transition effect. The cards are displayed inline and have a width of 300 pixels. The images have a border-radius of 5 pixels on the top corners, while the card container has padding.

JavaScript: Creating a countdown timer

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      /* CSS for countdown timer */
      #timer {
        font-size: 48px;
        font-weight: bold;
      }
    </style>
    <script>
      // JavaScript for countdown timer
      let countDownDate = new Date("Mar 1, 2023
00:00:00").getTime();
      let x = setInterval(function() {
        let now = new Date().getTime();
        let distance = countDownDate - now;
        let days = Math.floor(distance / (1000 * 60 *
60 * 24));
        let hours = Math.floor((distance % (1000 * 60 *
60 * 24)) / (1000 * 60 * 60));
```

```
        let minutes = Math.floor((distance % (1000 * 60
* 60)) / (1000 * 60));
        let seconds = Math.floor((distance % (1000 *
60)) / 1000);
        document.getElementById("timer").innerHTML =
days + "d " + hours + "h "
          + minutes + "m " + seconds + "s ";
        if (distance < 0) {
          clearInterval(x);
          document.getElementById("timer").innerHTML =
"EXPIRED";
        }
      }, 1000);
    </script>
  </head>
  <body>
    <!-- HTML for countdown timer -->
    <div id="timer"></div>
    <p>Countdown to March 1, 2023.</p>
  </body>
</html>
```

Explanation: This code creates a countdown timer using JavaScript. The countdown timer displays the time remaining until March 1, 2023. The timer is updated every second using the setInterval() method. The timer displays the remaining days, hours, minutes, and seconds. Once the countdown timer reaches zero, the timer displays "EXPIRED".

HTML/CSS/JavaScript: Creating a modal window

```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>My Website</title>
    <style>
      /* CSS for modal window */
      .modal {
        display: none;
        position: fixed;
        z-index: 1;
        left: 0;
        top: 0;
        width: 100%;
        height: 100%;
        overflow: auto;
        background-color: rgba(0,0,0,0.4);
      }
      .modal-content {
        background-color: #fefefe;
        margin: 15% auto;
        padding: 20px;
        border: 1px solid #888;
        width: 80%;
        max-width: 600px;
        border-radius: 5px;
      }
      .close {
        color: #aaa;
        float: right;
        font-size: 28px;
        font-weight: bold;
      }
      .close:hover,
```

```
      .close:focus {
        color: black;
        text-decoration: none;
        cursor: pointer;
      }
    </style>
    <script>
      // JavaScript for modal window
      let modal = document.getElementById("myModal");
      let btn = document.getElementById("myBtn");
      let span =
document.getElementsByClassName("close")[0];
      btn.onclick = function() {
        modal.style.display = "block";
      }
      span.onclick = function() {
        modal.style.display = "none";
      }
      window.onclick = function(event) {
        if (event.target == modal) {
          modal.style.display = "none";
        }
      }
    </script>
  </head>
  <body>
    <!-- HTML for modal window -->
    <button id="myBtn">Open Modal</button>
    <div id="myModal" class="modal">
      <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Modal Header</h2>
```

```
        <p>Modal content goes here.</p>
      </div>
    </div>
  </body>
</html>
```

Explanation: This code creates a modal window using HTML, CSS, and JavaScript. The modal window is hidden by default and is displayed when the user clicks on the "Open Modal" button. The modal window has a close button that can be used to hide the modal window. The JavaScript code handles the opening and closing of the modal window. The modal window has a fixed position and a transparent black background. The modal content is centered on the screen and has a maximum width of 600 pixels.

HTML/CSS: Creating a responsive navigation bar

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      /* CSS for modal window */
      .modal {
        display: none;
        position: fixed;
        z-index: 1;
        left: 0;
```

```
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgba(0,0,0,0.4);
  }
  .modal-content {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
    max-width: 600px;
    border-radius: 5px;
  }
  .close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
  }
  .close:hover,
  .close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
  }
</style>
<script>
  // JavaScript for modal window
  let modal = document.getElementById("myModal");
  let btn = document.getElementById("myBtn");
```

```
      let span =
document.getElementsByClassName("close")[0];
      btn.onclick = function() {
        modal.style.display = "block";
      }
      span.onclick = function() {
        modal.style.display = "none";
      }
      window.onclick = function(event) {
        if (event.target == modal) {
          modal.style.display = "none";
        }
      }
    </script>
  </head>
  <body>
    <!-- HTML for modal window -->
    <button id="myBtn">Open Modal</button>
    <div id="myModal" class="modal">
      <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Modal Header</h2>
        <p>Modal content goes here.</p>
      </div>
    </div>
  </body>
</html>
```

Explanation: This code creates a modal window using HTML, CSS, and JavaScript. The modal window is hidden by default and is displayed when the user clicks on the "Open Modal" button. The modal window has a close button that can be used to hide the

modal window. The JavaScript code handles the opening and closing of the modal window. The modal window has a fixed position and a transparent black background. The modal content is centered on the screen and has a maximum width of 600 pixels.

HTML/CSS: Creating a responsive navigation bar

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Website</title>
    <style>
      /* CSS for navigation bar */
      .topnav {
        overflow: hidden;
        background-color: #333;
      }
      .topnav a {
        float: left;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
        font-size: 17px;
      }
      .topnav a:hover {
        background-color: #ddd;
```

```
      color: black;
    }
    .topnav a.active {
      background-color: #4CAF50;
      color: white;
    }
    .icon {
      display: none;
    }
    @media screen and (max-width: 600px) {
      .topnav a:not(:first-child) {display: none;}
      .topnav a.icon {
        float: right;
        display: block;
      }
    }
    @media screen and (max-width: 600px) {
      .topnav.responsive {position: relative;}
      .topnav.responsive .icon {
        position: absolute;
        right: 0;
        top: 0;
      }
      .topnav.responsive a {
        float: none;
        display: block;
        text-align: left;
      }
    }
  </style>
</head>
<body>
```

```
    <!-- HTML for navigation bar -->
    <div class="topnav" id="myTopnav">
      <a href="#" class="active">Home</a>
      <a href="#">About</a>
      <a href="#">Contact</a>
      <a href="#" class="icon" onclick="myFunction()">
        <i class="fa fa-bars"></i>
      </a>
    </div>
    <script>
      // JavaScript for responsive navigation bar
      function myFunction() {
        let x = document.getElementById("myTopnav");
        if (x.className === "topnav") {
          x.className += " responsive";
        } else {
          x.className = "topnav";
        }
      }
    </script>
  </body>
</html>
```

Explanation: This code creates a responsive navigation bar using HTML and CSS. The navigation bar has links to different pages on the website, and it also includes a "hamburger" menu icon. When the screen size is less than 600 pixels, the links in the navigation bar are hidden, and the hamburger icon is displayed. When the user clicks on the icon, the links are displayed in a dropdown menu. The JavaScript code handles the opening and closing of the dropdown menu when the icon is clicked. The navigation bar has a fixed position at the top of the screen, and it has a dark

background color. The active link is highlighted with a different background color, and the links change color when the user hovers over them.

HTML/CSS/JavaScript: Creating a slideshow

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Slideshow</title>
    <style>
      /* CSS for slideshow */
      .slideshow {
        width: 100%;
        height: 400px;
        position: relative;
      }
      .slideshow img {
        width: 100%;
        height: 400px;
        object-fit: cover;
      }
      .slideshow .prev, .slideshow .next {
        position: absolute;
        top: 50%;
        transform: translateY(-50%);
        font-size: 30px;
        font-weight: bold;
        padding: 10px;
        background-color: rgba(255, 255, 255, 0.5);
```

```
  color: black;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
.slideshow .prev:hover, .slideshow .next:hover {
  background-color: rgba(0, 0, 0, 0.5);
  color: white;
}
.slideshow .prev {
  left: 0;
}
.slideshow .next {
  right: 0;
}
/* CSS for slideshow indicators */
.slideshow-indicators {
  text-align: center;
}
.slideshow-indicators button {
  display: inline-block;
  width: 12px;
  height: 12px;
  border-radius: 50%;
  border: none;
  margin: 0 5px;
  background-color: #bbb;
  cursor: pointer;
  transition: background-color 0.3s ease;
}
.slideshow-indicators button.active {
  background-color: #555;
}
```

```
    </style>
  </head>
  <body>
    <!-- HTML for slideshow -->
    <div class="slideshow">
      <img src="slide1.jpg" alt="Slide 1">
      <img src="slide2.jpg" alt="Slide 2">
      <img src="slide3.jpg" alt="Slide 3">
      <img src="slide4.jpg" alt="Slide 4">
      <div class="prev"
onclick="plusSlides(-1)">&#10094;</div>
      <div class="next"
onclick="plusSlides(1)">&#10095;</div>
      <div class="slideshow-indicators">
        <button class="active"
onclick="currentSlide(1)"></button>
        <button onclick="currentSlide(2)"></button>
        <button onclick="currentSlide(3)"></button>
        <button onclick="currentSlide(4)"></button>
      </div>
    </div>
    <script>
      // JavaScript for slideshow
      let slideIndex = 1;
      showSlides(slideIndex);

      function plusSlides(n) {
        showSlides(slideIndex += n);
      }

      function currentSlide(n) {
        showSlides(slideIndex = n);
```

```
      }

      function showSlides(n) {
        let i;
        let slides =
document.getElementsByClassName("slideshow")[0].getElem
entsByTagName("img");
        let dots =
document.getElementsByClassName("slideshow-indicators")
[0].getElementsByTagName("button");
        if (n > slides.length) {
          slideIndex = 1;
        }
        if (n < 1) {
          slideIndex = slides.length;
        }
        for (i = 0; i < slides.length; i++) {
          slides[i].style.display = "none";
        }
        for (i = 0; i < dots.length; i++) {
dots[i].className = dots[i].className.replace("
active", "");
}
slides[slideIndex-1].style.display = "block";
dots[slideIndex-1].className += " active";
}
</script>

  </body>
</html>
```

This example demonstrates how to create a responsive slideshow using HTML, CSS, and JavaScript. The slideshow consists of several images that automatically change after a certain period of time. It also has navigation arrows and indicators that allow the user to manually switch between images.

HTML/CSS: Creating a pricing table

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Pricing Table</title>
    <style>
      /* CSS for pricing table */
      .pricing-table {
        width: 100%;
        max-width: 800px;
        margin: 0 auto;
        display: flex;
        justify-content: center;
        align-items: center;
      }
      .pricing-table .column {
        width: 100%;
        max-width: 300px;
        margin: 0 20px;
        background-color: #fff;
        box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
        text-align: center;
        transition: box-shadow 0.3s ease;
```

```
}
.pricing-table .column:hover {
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
}
.pricing-table .header {
  background-color: #333;
  color: #fff;
  font-size: 24px;
  font-weight: bold;
  padding: 20px;
  border-top-left-radius: 5px;
  border-top-right-radius: 5px;
}
.pricing-table .price {
  font-size: 48px;
  font-weight: bold;
  margin: 20px 0;
}
.pricing-table .description {
  font-size: 16px;
  color: #888;
  margin-bottom: 20px;
}
.pricing-table .features {
  list-style: none;
  padding: 0;
  margin: 20px 0;
  border-bottom-left-radius: 5px;
  border-bottom-right-radius: 5px;
}
.pricing-table .features li {
  font-size: 16px;
```

```
        padding: 10px;
        border-bottom: 1px solid #ccc;
        transition: background-color 0.3s ease;
      }
      .pricing-table .features li:last-child {
        border-bottom: none;
      }
      .pricing-table .features li:hover {
        background-color: #f2f2f2;
      }
    </style>
  </head>
  <body>
    <!-- HTML for pricing table -->
    <div class="pricing-table">
      <div class="column">
        <div class="header">Basic</div>
        <div class="price">$9.99/month</div>
        <div class="description">Lorem ipsum dolor sit
amet, consectetur adipiscing elit.</div>
        <ul class="features">
          <li>10 GB storage</li>
          <li>50 GB bandwidth</li>
          <li>5 email accounts</li>
          <li>24/7 support</li>
        </ul>
      </div>
      <div class="column">
        <div class="header">Pro</div>
        <div class="price">$19.99/month</div>
        <div class="description">Lorem ipsum dolor sit
amet consectetur adipiscing elit.</div>
```

```
<ul class="features">
<li>50 GB storage</li>
<li>200 GB bandwidth</li>
<li>20 email accounts</li>
<li>24/7 support</li>
</ul>
</div>
<div class="column">
<div class="header">Premium</div>
<div class="price">$49.99/month</div>
<div class="description">Lorem ipsum dolor sit amet,
consectetur adipiscing elit.</div>
<ul class="features">
<li>100 GB storage</li>
<li>500 GB bandwidth</li>
<li>50 email accounts</li>
<li>24/7 support</li>
</ul>
</div>
</div>

  </body>
</html>
```

This example demonstrates how to create a simple pricing table using HTML and CSS. The pricing table consists of three columns, each with a header, price, description, and list of features. The CSS is used to style the pricing table and make it responsive.

HTML/CSS/JavaScript: Creating a simple game of "Simon"

```
<!DOCTYPE html>
<html>
  <head>
    <title>Simon Game</title>
    <style>
      /* CSS for Simon game */
      #simon-board {
        width: 100%;
        max-width: 500px;
        margin: 0 auto;
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        align-items: center;
      }
      .simon-button {
        width: 100%;
        max-width: 150px;
        height: 150px;
        margin: 10px;
        background-color: #eee;
        border: 5px solid #333;
        border-radius: 50%;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
        cursor: pointer;
        transition: transform 0.3s ease;
      }
      .simon-button:hover {
```

```
  transform: scale(1.1);
}
.simon-button.green {
  background-color: #6ab04c;
  border-color: #4e8b39;
}
.simon-button.red {
  background-color: #e74c3c;
  border-color: #c0392b;
}
.simon-button.yellow {
  background-color: #f1c40f;
  border-color: #d4ac0d;
}
.simon-button.blue {
  background-color: #3498db;
  border-color: #2980b9;
}
.simon-button.disabled {
  opacity: 0.5;
  cursor: default;
}
#simon-message {
  text-align: center;
  font-size: 24px;
  font-weight: bold;
  margin-top: 20px;
}
#simon-score {
  text-align: center;
  font-size: 16px;
  margin-top: 10px;
```

```
      }
    </style>
  </head>
  <body>
    <!-- HTML for Simon game -->
    <div id="simon-board">
      <div class="simon-button green"></div>
      <div class="simon-button red"></div>
      <div class="simon-button yellow"></div>
      <div class="simon-button blue"></div>
    </div>
    <div id="simon-message"></div>
    <div id="simon-score">Score: <span
id="score">0</span></div>

scss
Copy code
<script>
  // JavaScript for Simon game
  const buttons =
document.querySelectorAll(".simon-button");
  const message =
document.getElementById("simon-message");
  const score = document.getElementById("score");

  let pattern = [];
  let playerPattern = [];
  let level = 1;

  // Generates random pattern
  function generatePattern() {
    pattern = [];
```

```
    for (let i = 0; i < level; i++) {
      pattern.push(Math.floor(Math.random() * 4));
    }
  }

  // Highlights buttons in pattern
  function playPattern() {
    message.textContent = "Watch carefully!";
    let i = 0;
    const intervalId = setInterval(() => {
      highlightButton(buttons[pattern[i]]);
      i++;
      if (i >= pattern.length) {
        clearInterval(intervalId);
        message.textContent = "Your turn!";
      }
    }, 1000);
  }

  // Highlights button
  function highlightButton(button) {
    button.classList.add("active");
    setTimeout(() => {
      button.classList.remove("active");
    }, 500);
  }

  // Checks if player's pattern matches computer's
pattern
  function checkPattern() {
    for (let i = 0; i < playerPattern.length; i++) {
      if (playerPattern[i] !== pattern[i]) {
```

```
        return false;
      }
    }
    return true;
  }

  // Resets game
  function resetGame() {
    pattern = [];
    playerPattern = [];
    level = 1;
    score.textContent = "0";
  }

  // Main game function
  function playGame() {
    generatePattern();
    playPattern();

    buttons.forEach((button) => {
      button.addEventListener("click", () => {
        if (message.textContent === "Your turn!") {

playerPattern.push(Array.from(buttons).indexOf(button))
;
          highlightButton(button);
          if (playerPattern.length === pattern.length)
{
            if (checkPattern()) {
              level++;
              score.textContent = level - 1;
              playerPattern = [];
```

```
              playGame();
            } else {
              message.textContent = "Game over! Click
any button to start again.";
              buttons.forEach((button) => {
                button.classList.add("disabled");
              });
              setTimeout(() => {
                resetGame();
                buttons.forEach((button) => {
                  button.classList.remove("disabled");
                });
                playGame();
              }, 3000);
            }
          }
        }
      });
    });
  }

  // Start game
  playGame();
</script>
  </body>
</html>
```

This example demonstrates how to create a simple game of "Simon" using HTML, CSS, and JavaScript. The game consists of four colored buttons and a message and score display. The JavaScript is used to generate a random pattern of button presses, play back the pattern, and check the player's pattern

against the computer's pattern. The game continues until the player makes an error, at which point the game is reset and the player can try again.