**University of Wolverhampton**

**School of Engineering, Computational and Mathematical Sciences**

**5CS020 Human-Computer Interaction**

**Workshop 4 – HTML User Interfaces**

In this workshop you will be experimenting with a number of user interfaces based on single web page apps.

**Part 1**

**Stopwatch**

HTML:

* The first part of the code defines the structure of the stopwatch app using HTML (HyperText Markup Language).
* The **<head>** section contains the styles for the stopwatch using CSS (Cascading Style Sheets).
* The **<body>** section contains the actual stopwatch, which is displayed in the browser window.
* The stopwatch consists of a time display element (**<p id="time">00:00:00</p>**), and three buttons (**<button>**) with the id's "startBtn", "stopBtn", and "resetBtn".

CSS:

* The styles in the **<head>** section define the look of the stopwatch.
* The **#stopwatch** selector sets the font size and text alignment for the stopwatch.
* The **button** selector sets the padding, font size, and margin for the buttons.

JavaScript:

* The last part of the code contains the JavaScript (JS) code that makes the stopwatch work.
* First, the code selects the elements from the HTML using **document.getElementById** and assigns them to variables (e.g., **const timeDisplay = document.getElementById("time");**).
* The **start** function uses **setInterval** to increment the **time** variable by one every second (1000 milliseconds). The **time** variable is used to calculate the elapsed hours, minutes, and seconds, which are displayed in the **timeDisplay** element.
* The **stop** function uses **clearInterval** to stop the timer.
* The **reset** function stops the timer and resets the **time** variable back to 0, and also resets the time display back to "00:00:00".
* Finally, the code adds event listeners to the buttons using **addEventListener**, which trigger the **start**, **stop**, and **reset** functions when the buttons are clicked.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**#stopwatch {**

**font-size: 30px;**

**text-align: center;**

**margin-top: 50px;**

**}**

**button {**

**padding: 10px 20px;**

**font-size: 20px;**

**margin: 20px;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="stopwatch">**

**<p id="time">00:00:00.000</p>**

**<button id="startBtn">Start</button>**

**<button id="stopBtn">Stop</button>**

**<button id="resetBtn">Reset</button>**

**</div>**

**<script>**

**const stopwatch = document.getElementById("stopwatch");**

**const timeDisplay = document.getElementById("time");**

**const startBtn = document.getElementById("startBtn");**

**const stopBtn = document.getElementById("stopBtn");**

**const resetBtn = document.getElementById("resetBtn");**

**let interval;**

**let startTime;**

**let elapsedTime = 0;**

**function start() {**

**startTime = Date.now();**

**interval = setInterval(() => {**

**let now = Date.now();**

**elapsedTime = now - startTime;**

**let hours = Math.floor((elapsedTime % (1000 \* 60 \* 60 \* 24)) /**

**(1000 \* 60 \* 60));**

**let minutes = Math.floor((elapsedTime % (1000 \* 60 \* 60)) /**

**(1000 \* 60));**

**let seconds = Math.floor((elapsedTime % (1000 \* 60)) /**

**1000);**

**let milliseconds = elapsedTime % 1000;**

**timeDisplay.innerHTML =**

**(hours < 10 ? "0" + hours : hours) +**

**":" +**

**(minutes < 10 ? "0" + minutes : minutes) +**

**":" +**

**(seconds < 10 ? "0" + seconds : seconds) +**

**"." +**

**(milliseconds < 10 ? "00" + milliseconds : milliseconds < 100 ? "0" +**

**milliseconds : milliseconds);**

**}, 10);**

**startBtn.setAttribute("disabled", "disabled");**

**}**

**function stop() {**

**clearInterval(interval);**

**startBtn.removeAttribute("disabled");**

**}**

**function reset() {**

**stop();**

**elapsedTime = 0;**

**timeDisplay.innerHTML = "00:00:00.000";**

**}**

**startBtn.addEventListener("click", start);**

**stopBtn.addEventListener("click", stop);**

**resetBtn.addEventListener("click", reset);**

**</script>**

**</body>**

**</html>**

**Part 2**

**Timer**

HTML:

* The first part of the code defines the structure of the countdown timer app using HTML (HyperText Markup Language).
* The **<head>** section contains the styles for the countdown timer using CSS (Cascading Style Sheets).
* The **<body>** section contains the actual countdown timer, which is displayed in the browser window.
* The countdown timer consists of a time display element (**<p id="time">00:00:00</p>**), an input field (**<input type="text" id="inputTime">**) for entering the time in seconds, and three buttons (**<button>**) with the id's "startBtn", "stopBtn", and "resetBtn".

CSS:

* The styles in the **<head>** section define the look of the countdown timer.
* The **#countdown** selector sets the font size and text alignment for the countdown timer.
* The **button** selector sets the padding, font size, and margin for the buttons.

JavaScript:

* The last part of the code contains the JavaScript (JS) code that makes the countdown timer work.
* First, the code selects the elements from the HTML using **document.getElementById** and assigns them to variables (e.g., **const timeDisplay = document.getElementById("time");**).
* The **start** function uses **parseInt** to convert the input time from a string to an integer in seconds, then converts the input time to milliseconds. The function then uses **setInterval** to decrement the **time** variable by 1000 milliseconds (1 second) every second. The **time** variable is used to calculate the remaining hours, minutes, and seconds, which are displayed in the **timeDisplay** element. If the **time** reaches 0, the **interval** is cleared using **clearInterval**, and an alert is displayed saying "Time's up!".
* The **stop** function uses **clearInterval** to stop the timer.
* The **reset** function stops the timer, resets the time display back to "00:00:00", and clears the input field.
* Finally, the code adds event listeners to the buttons using **addEventListener**, which trigger the **start**, **stop**, and **reset** functions when the buttons are clicked.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**#countdown {**

**font-size: 30px;**

**text-align: center;**

**margin-top: 50px;**

**}**

**button {**

**padding: 10px 20px;**

**font-size: 20px;**

**margin: 20px;**

**}**

**</style>**

**</head>**

**<body>**

**<div id="countdown">**

**<p id="time">00:00:00</p>**

**<input type="text" id="inputTime">**

**<button id="startBtn">Start</button>**

**<button id="stopBtn">Stop</button>**

**<button id="resetBtn">Reset</button>**

**</div>**

**<script>**

**const countdown = document.getElementById("countdown");**

**const timeDisplay = document.getElementById("time");**

**const inputTime = document.getElementById("inputTime");**

**const startBtn = document.getElementById("startBtn");**

**const stopBtn = document.getElementById("stopBtn");**

**const resetBtn = document.getElementById("resetBtn");**

**let interval;**

**let time;**

**function start() {**

**time = parseInt(inputTime.value) \* 1000;**

**interval = setInterval(() => {**

**time -= 1000;**

**let minutes = Math.floor(time / 60 / 1000);**

**let seconds = Math.floor(time / 1000) % 60;**

**let hours = Math.floor(time / 3600 / 1000);**

**timeDisplay.innerHTML =**

**(hours < 10 ? "0" + hours : hours) +**

**":" +**

**(minutes < 10 ? "0" + minutes : minutes) +**

**":" +**

**(seconds < 10 ? "0" + seconds : seconds);**

**if (time <= 0) {**

**clearInterval(interval);**

**alert("Time's up!");**

**}**

**}, 1000);**

**startBtn.setAttribute("disabled", "disabled");**

**}**

**function stop() {**

**clearInterval(interval);**

**startBtn.removeAttribute("disabled");**

**}**

**function reset() {**

**stop();**

**timeDisplay.innerHTML = "00:00:00";**

**inputTime.value = "";**

**}**

**startBtn.addEventListener("click", start);**

**stopBtn.addEventListener("click", stop);**

**resetBtn.addEventListener("click", reset);**

**</script>**

**</body>**

**</html>**

**Part 3**

**Alarm Clock**

HTML:

* The first part of the code defines the structure of the alarm clock app using HTML (HyperText Markup Language).
* The **<head>** section contains the styles for the alarm clock using CSS (Cascading Style Sheets).
* The **<body>** section contains the actual alarm clock, which is displayed in the browser window. The **onload** attribute is added to the **<body>** element, which triggers the **startClock** function when the page loads.
* The alarm clock consists of a time display element (**<p id="time">00:00:00</p>**), an input field (**<input type="text" id="inputTime">**) for entering the alarm time, and two buttons (**<button>**) with the id's "setAlarmBtn" and "clearAlarmBtn".

CSS:

* The styles in the **<head>** section define the look of the alarm clock.
* The **#alarmClock** selector sets the font size and text alignment for the alarm clock.
* The **button** selector sets the padding, font size, and margin for the buttons.

JavaScript:

* The last part of the code contains the JavaScript (JS) code that makes the alarm clock work.
* First, the code selects the elements from the HTML using **document.getElementById** and assigns them to variables (e.g., **const timeDisplay = document.getElementById("time");**).
* The **startClock** function uses **setInterval** to get the current time every second and display it in the **timeDisplay** element. If the current time matches the alarm time, an alert is displayed saying "Wake up! Alarm is ringing.".
* The **setAlarm** function sets the alarm time using the value from the input field and disables the "Set Alarm" button.
* The **clearAlarm** function uses **clearInterval** to stop the clock, enables the "Set Alarm" button, resets the time display, and clears the input field.
* Finally, the code adds event listeners to the buttons using **addEventListener**, which trigger the **setAlarm** and **clearAlarm** functions when the buttons are clicked.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**#alarmClock {**

**font-size: 30px;**

**text-align: center;**

**margin-top: 50px;**

**}**

**button {**

**padding: 10px 20px;**

**font-size: 20px;**

**margin: 20px;**

**}**

**</style>**

**</head>**

**<body onload="startClock()">**

**<div id="alarmClock">**

**<p id="time">00:00:00</p>**

**<input type="text" id="inputTime">**

**<button id="setAlarmBtn">Set Alarm</button>**

**<button id="clearAlarmBtn">Clear Alarm</button>**

**</div>**

**<script>**

**const alarmClock = document.getElementById("alarmClock");**

**const timeDisplay = document.getElementById("time");**

**const inputTime = document.getElementById("inputTime");**

**const setAlarmBtn = document.getElementById("setAlarmBtn");**

**const clearAlarmBtn = document.getElementById("clearAlarmBtn");**

**let interval;**

**let alarmTime;**

**function startClock() {**

**interval = setInterval(() => {**

**let now = new Date();**

**let hours = now.getHours();**

**let minutes = now.getMinutes();**

**let seconds = now.getSeconds();**

**timeDisplay.innerHTML =**

**(hours < 10 ? "0" + hours : hours) +**

**":" +**

**(minutes < 10 ? "0" + minutes : minutes) +**

**":" +**

**(seconds < 10 ? "0" + seconds : seconds);**

**if (timeDisplay.innerHTML === alarmTime) {**

**alert("Wake up! Alarm is ringing.");**

**}**

**}, 1000);**

**}**

**function setAlarm() {**

**alarmTime = inputTime.value;**

**setAlarmBtn.setAttribute("disabled", "disabled");**

**}**

**function clearAlarm() {**

**clearInterval(interval);**

**setAlarmBtn.removeAttribute("disabled");**

**timeDisplay.innerHTML = "00:00:00";**

**inputTime.value = "";**

**}**

**setAlarmBtn.addEventListener("click", setAlarm);**

**clearAlarmBtn.addEventListener("click", clearAlarm);**

**</script>**

**</body>**

**</html>**

**Part 4**

**Hiding and displaying sections**

HTML:

* The first part of the code defines the structure of the example using HTML (HyperText Markup Language).
* The **<head>** section contains the styles for the example using CSS (Cascading Style Sheets).
* The **<body>** section contains the actual example, which is displayed in the browser window. The **onload** attribute is added to the **<body>** element, which triggers the **showSection** function and passes the **section1** argument when the page loads.
* The example consists of 3 buttons (**<button id="btn1">**, **<button id="btn2">**, and **<button id="btn3">**) and a container DIV (**<div class="container">**) that contains 3 sections (**<div id="section1" class="section">**, **<div id="section2" class="section">**, and **<div id="section3" class="section">**).

CSS:

* The styles in the **<head>** section define the look of the example.
* The **.container** selector sets the display, justification, alignment, and height of the container DIV.
* The **.section** selector sets the width, height, background color, border, text alignment, and font size of the sections.
* The **button** selector sets the padding, margin, and font size of the buttons.

JavaScript:

* The last part of the code contains the JavaScript (JS) code that makes the example work.
* First, the code selects the elements from the HTML using **document.getElementById** and assigns them to variables (e.g., **const btn1 = document.getElementById("btn1");**).
* The **showSection** function sets the **display** property of all sections to **none** and then sets the **display** property of the selected section to **block**.
* Finally, the code adds event listeners to the buttons using **addEventListener**, which trigger the **showSection** function and pass the selected section as an argument when the buttons are clicked.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**.container {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 100vh;**

**}**

**.section {**

**width: 200px;**

**height: 200px;**

**background-color: lightgray;**

**border: 2px solid gray;**

**text-align: center;**

**font-size: 20px;**

**}**

**button {**

**padding: 10px 20px;**

**margin: 20px;**

**font-size: 20px;**

**}**

**</style>**

**</head>**

**<body onload=showSection(section1)>**

**<div>**

**<button id="btn1">Show Section 1</button>**

**<button id="btn2">Show Section 2</button>**

**<button id="btn3">Show Section 3</button>**

**</div>**

**<div class="container">**

**<div id="section1" class="section">Section 1</div>**

**<div id="section2" class="section">Section 2</div>**

**<div id="section3" class="section">Section 3</div>**

**</div>**

**<script>**

**const btn1 = document.getElementById("btn1");**

**const btn2 = document.getElementById("btn2");**

**const btn3 = document.getElementById("btn3");**

**const section1 = document.getElementById("section1");**

**const section2 = document.getElementById("section2");**

**const section3 = document.getElementById("section3");**

**function showSection(section) {**

**section1.style.display = "none";**

**section2.style.display = "none";**

**section3.style.display = "none";**

**section.style.display = "block";**

**}**

**btn1.addEventListener("click", () => {**

**showSection(section1);**

**});**

**btn2.addEventListener("click", () => {**

**showSection(section2);**

**});**

**btn3.addEventListener("click", () => {**

**showSection(section3);**

**});**

**</script>**

**</body>**

**</html>**

**Part 5 – Challenge task**

**Building a combined stopwatch, timer and alarm clock in one HTML app**

Based on what you have learned so far, build a combined stopwatch, timer and alarm clock in one HTML file.

When you are successful, separate the HTML, CSS and JavaScript into 3 separate files.

**Part 6**

**Clock with retro-style numbers**

The following is an example of HTML and CSS code to display retro style white numbers on black squares with rounded corners for a running clock:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<style>**

**.container {**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**height: 100vh;**

**}**

**.digit {**

**width: 50px;**

**height: 50px;**

**background-color: black;**

**color: white;**

**border-radius: 10px;**

**display: flex;**

**justify-content: center;**

**align-items: center;**

**font-size: 20px;**

**font-weight: bold;**

**font-family: "Arial Black", Arial, sans-serif;**

**margin: 10px;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="container">**

**<div class="digit" id="hours1"></div>**

**<div class="digit" id="hours2"></div>**

**<div class="digit">:</div>**

**<div class="digit" id="minutes1"></div>**

**<div class="digit" id="minutes2"></div>**

**<div class="digit">:</div>**

**<div class="digit" id="seconds1"></div>**

**<div class="digit" id="seconds2"></div>**

**</div>**

**<script>**

**function updateTime() {**

**const now = new Date();**

**const hours = now.getHours().toString().padStart(2, "0");**

**const minutes = now.getMinutes().toString().padStart(2, "0");**

**const seconds = now.getSeconds().toString().padStart(2, "0");**

**document.getElementById("hours1").innerHTML = hours[0];**

**document.getElementById("hours2").innerHTML = hours[1];**

**document.getElementById("minutes1").innerHTML = minutes[0];**

**document.getElementById("minutes2").innerHTML = minutes[1];**

**document.getElementById("seconds1").innerHTML = seconds[0];**

**document.getElementById("seconds2").innerHTML = seconds[1];**

**}**

**setInterval(updateTime, 1000);**

**</script>**

**</body>**

**</html>**

In this example, the **.container** class sets the display, justification, alignment, height, and wrap of the container DIV.

The .digit class sets the width, height, background color, color, border radius, display, justification, alignment, font size, font weight, font family, and margin of each digit square.

The clock is displayed using <div class="digit"> elements, and each digit is updated separately using document.getElementById and innerHTML.

The JavaScript code uses the **setInterval** function to update the clock every second. The **updateTime** function gets the current time using the Date object, and updates the contents of the <div> elements using **document.getElementById** and **innerHTML** with the corresponding digits of the hours, minutes, and seconds.

This is the end of Workshop 4.

**Part 7 – Challenge task**

**Modifying the combined stopwatch, timer and alarm clockwith the retro-style numbers**

Based on what you have learned so far, change the combined stopwatch, timer and alarm clock into the retro-style numbers.