JavaScript

```
/* Variable */

const millisecToMinute = 60000;
const averageWordLength = 4.5;
const contingencyData = {
    content: "The wound is the place where the light enters you.",
    author: "Rumi",
};

var event = "relay";
var eventActive = false;
var type = "quote";
var target = "";
var targetSource = "";
var targetAnswer = "";
var targetIndex = 0;
var time = {
    initial: 0,
    end: 0,
}


/* Dom Object */

var container = document.getElementById("container");
var containerStyle = window.getComputedStyle(container);

var introScreen = document.getElementById("introScreen");
var eventScreen = document.getElementById("eventScreen");

var targetText = document.getElementById("targetText");
var targetSourceText = document.getElementById("targetSourceText");
var targetAnswerText = document.getElementById("targetAnswerText");
var userInput = document.getElementById("userInput");

var startButton = document.getElementById("startButton");
var continueButton = document.getElementById("continueButton");
var exitEventButton = document.getElementById("exitButton");

var eventButton = {
    relay: document.getElementById("relayButton"),
    precision: document.getElementById("precisionButton"),
}
var typeButton = {
    quote: document.getElementById("quoteButton"),
    trivia: document.getElementById("triviaButton"),
    paragraph: document.getElementById("paragraphButton"),
    word: document.getElementById("wordButton"),
    character: document.getElementById("characterButton"),
}

var statText = {
    current: {
        wpm: document.getElementById("currentStat_wpm"),
        time: document.getElementById("currentStat_time"),
        backspace: document.getElementById("currentStat_backspace"),
        error: document.getElementById("currentStat_error"),
    },
    best: {
        wpm: document.getElementById("bestStat_wpm"),
        time: document.getElementById("bestStat_time"),
        backspace: document.getElementById("bestStat_backspace"),
        error: document.getElementById("bestStat_error"),
    }
}
```

```
/* Setup Stat */

var stat = {};

for (x in {current: 0, best: 0}) {
    stat[x] = {};
    for (y in eventButton) {
        stat[x][y] = {};
        for (z in typeButton) {
            stat[x][y][z] = {wpm: 0, time: 0, backspace: 0, error: 0};
        }
    }
}
for (y in eventButton) {
    for (z in typeButton) {
        for (thing in {wpm: 0, time: 0, backspace: 0, error: 0}) {
            if (window.sessionStorage["typeOMeter_stat_" + y + "_" + z + "_" + thing]) {
                stat.best[y][z][thing] = window.sessionStorage["typeOMeter_stat_" + y + "_" + z + "_" + thing];
            }
            else {
                window.sessionStorage["typeOMeter_stat_" + y + "_" + z + "_" + thing] = 0;
            }
        }
    }
}


/* Load Function Call */

centerContainer();
setEvent("relay");
setType("quote");
//window.sessionStorage.clear();
//setupEvent();
console.log("averageWordLength = " + averageWordLength);


/* Event Function */

async function setupEvent() {
    if (type == "quote") {
        var data = await getQuote();
    }
    else if (type == "trivia") {
        var data = await getTrivia();
    }
    else if (type == "paragraph") {
        var data = await getParagraph();
    }
    else if (type == "word") {
        var data = await getWord();
    }
    else if (type == "character") {
        var data = getCharacter();
    }

    setTarget(data);
    beginEvent();
}
```

Algorithm

```
function setTarget(data) {
    target = data.content;
    targetIndex = 0;
    targetSource = data.author;
    targetAnswer = "";
    if (data.answer) {
        targetAnswer = data.answer;
    }

    if (event == "relay") {
        targetText.innerHTML = target;
        targetSourceText.innerHTML = targetSource;
        targetAnswerText.innerHTML = targetAnswer;
    }
    else if (event == "precision") {
        targetText.innerHTML = "[<span class='targetEmphasis'>" + target[targetIndex] + "</span>]" + target.substr(targetIndex +
1);
        targetSourceText.innerHTML = targetSource;
        targetAnswerText.innerHTML = targetAnswer;
    }
}

function beginEvent() {
    eventActive = true;

    introScreen.style.display = "none";
    eventScreen.style.display = "block";
    continueButton.innerHTML = "Enter";

    updateBestStat();

    clearInput();
    userInput.focus();

    time.initial = new Date();
}

function concludeEvent() {
    if (eventActive) {
        eventActive = false;
        continueButton.innerHTML = "Next";

        updateStat();
        updateBestStat();

        clearInput();
    }
}


/* Generate Target Function */

// API source: https://github.com/lukePeavey/quotable
async function getQuote() {
    // Fetch a random quote from the Quotable API
    const response = await fetch("https://api.quotable.io/random");
    const data = await response.json();
    if (response.ok) {
        return data;
    }
    else {
        return contingencyData;
    }
}
```

```
// API source: https://github.com/sottenad/jService
async function getTrivia() {
    // Fetch a random quote from the Quotable API
    const response = await fetch("http://jservice.io/api/random");
    const data = await response.json();
    if (response.ok) {
        return {content: capitalize(data[0].category.title, true) + ": " + data[0].question + ".", author: "Jeopardy", answer: "Answer: "
+ data[0].answer};
    }
    else {
        return contingencyData;
    }
}

// API source: https://github.com/petenelson/wp-any-ipsum
async function getParagraph() {
    // Fetch a random quote from the Quotable API
    const response = await fetch("https://baconipsum.com/api/?type=meat-and-filler&paras=1&start-with-lorem=1");
    const data = await response.json();
    if (response.ok) {
        return {content: data[0].replace(/  /g, " "), author: "Bacon Ipsum"};
    }
    else {
        return contingencyData;
    }
}

// API source: https://github.com/RazorSh4rk/random-word-api
async function getWord() {
    // Fetch a random quote from the Random Word API
    var randomLength = Math.floor(Math.random() * 10) + 5;
    const response = await fetch("https://random-word-api.herokuapp.com/word?swear=0&number=" + randomLength);
    const data = await response.json();
    if (response.ok) {
        return {content: capitalize(data.join(" ") + ".", false), author: "English Dictionary"};
    }
    else {
        return contingencyData;
    }
}

function getCharacter() {
    var string = "";
    var wordLength = Math.floor(Math.random() * 10) + 5;
    for (var x=0; x<wordLength; x++) {
        var characterLength = Math.floor(Math.random()*10) + 1;
        for (var y=0; y<characterLength; y++) {
            // normal ASCII: 33-126
            var character = String.fromCharCode(Math.floor(Math.random()*93) + 34);
            if (character == "<" || character == ">" || character == "&") {
                y -= 1;
            }
            else {
                string += character;
            }
        }
        if (x < wordLength - 1) {
            string += " ";
        }
    }
    return {content: string, author: "Lady Luck"};
}
```

```
/* Button Function */

function setEvent(chosenEvent) {
    event = chosenEvent;
    for (x in eventButton) {
        eventButton[x].classList.remove("selected");
    }
    eventButton[event].classList.add("selected");

    if (event == "relay") {
        statText.current.backspace.style.display = "block";
        statText.best.backspace.style.display = "block";
    }
    else if (event == "precision") {
        statText.current.backspace.style.display = "none";
        statText.best.backspace.style.display = "none";
    }

    statText.current.wpm.innerHTML = "wpm (word/min): " + stat.current[event][type].wpm;
    statText.current.time.innerHTML = "time (min): " + stat.current[event][type].time;
    statText.current.backspace.innerHTML = "backspace (count): " + stat.current[event][type].backspace;
    statText.current.error.innerHTML = "error (count): " + stat.current[event][type].error;

    statText.best.wpm.innerHTML = "wpm (word/min): " + stat.best[event][type].wpm;
    statText.best.time.innerHTML = "time (min): " + stat.best[event][type].time;
    statText.best.backspace.innerHTML = "backspace (count): " + stat.best[event][type].backspace;
    statText.best.error.innerHTML = "error (count): " + stat.best[event][type].error;
}

function setType(chosenType) {
    type = chosenType;
    for (x in typeButton) {
        typeButton[x].classList.remove("selected");
    }
    typeButton[type].classList.add("selected");

    statText.current.wpm.innerHTML = "wpm (word/min): " + stat.current[event][type].wpm;
    statText.current.time.innerHTML = "time (min): " + stat.current[event][type].time;
    statText.current.backspace.innerHTML = "backspace (count): " + stat.current[event][type].backspace;
    statText.current.error.innerHTML = "error (count): " + stat.current[event][type].error;

    statText.best.wpm.innerHTML = "wpm (word/min): " + stat.best[event][type].wpm;
    statText.best.time.innerHTML = "time (min): " + stat.best[event][type].time;
    statText.best.backspace.innerHTML = "backspace (count): " + stat.best[event][type].backspace;
    statText.best.error.innerHTML = "error (count): " + stat.best[event][type].error;
}

function exitEvent() {
    eventActive = false;
    concludeEvent();

    introScreen.style.display = "block";
    eventScreen.style.display = "none";
}
```

```
/* User Input Function */

function checkUserInput(key) {
    var input = userInput.value;
    var keyCode = key.which || key.keyCode;
    // Enter: 13
    // Backspace: 8

    if (keyCode == 13 && !eventActive) {
        key.preventDefault();
        setupEvent();
    }
    else if (keyCode == 13 && event == "relay") {
        key.preventDefault();
        if (input == target) {
            concludeEvent();
        }
        else {
            stat.current[event][type].error += 1;
        }
    }
    if (keyCode == 8) {
        stat.current[event][type].backspace += 1;
    }
    if (eventActive) {
        updateStat();
    }
}

function checkUserInput_precision() {
    var input = userInput.value;
    if (eventActive && event == "precision") {
        if (input == "") {}
        else if (input == target[targetIndex]) {
            targetIndex += 1;
            if (targetIndex >= target.length) {
                targetText.innerHTML = target;
                concludeEvent();
            }
            else if (target[targetIndex] == " "){
                targetText.innerHTML = "<span class='targetIgnore'>" + target.substring(0, targetIndex)
                    + "<wbr></span><span class='targetEmphasis'>[ ]</span>" + target.substr(targetIndex + 1);
            }
            else {
                targetText.innerHTML = "<span class='targetIgnore'>" + target.substring(0, targetIndex)
                    + "</span>[<span class='targetEmphasis'>" + target[targetIndex]
                    + "</span>]" + target.substr(targetIndex + 1);
            }
        }
        else {
            stat.current[event][type].error += 1;
        }
        clearInput();
        if (eventActive) {
            updateStat();
        }
    }
}
```

```
function resizeInputField(element, minimize) {
    if (minimize) {
            element.style.height = "0px";
    }
    else {
        element.style.height = "auto";
        //console.log(element.scrollHeight) [used to determine value when row = 1]
        if (element.scrollHeight < 40) {
            element.style.height = "0px";
        }

        var containerHeight = Number(containerStyle.getPropertyValue("height").replace("px", ""));
        element.style.height = Math.min((element.scrollHeight), containerHeight/3) + "px";
    }
}

function penalizePaste(key) {
    key.preventDefault();
    eventActive = false;
    concludeEvent();
    setTarget({content: "I will not copy and paste. I will type my answer. I will play fairly.", author: "Fair Play Police"});
    beginEvent();
}
```

Abstraction

```
/* Stat Function */

function updateStat() {
    time.end = new Date();
    stat.current[event][type].time = floor_nDecimal((time.end - time.initial) / millisecToMinute, 2);
    if (event == "relay") {
        stat.current[event][type].wpm = floor_nDecimal(userInput.value.length / (averageWordLength *
Math.max(stat.current[event][type].time, 0.01)), 2);
    }
    else if (event == "precision") {
        stat.current[event][type].wpm = floor_nDecimal(targetIndex / (averageWordLength * Math.max(stat.current[event]
[type].time, 0.01)), 2);
    }

    statText.current.wpm.innerHTML = "wpm (word/min): " + stat.current[event][type].wpm;
    statText.current.time.innerHTML = "time (min): " + stat.current[event][type].time;
    statText.current.backspace.innerHTML = "backspace (count): " + stat.current[event][type].backspace;
    statText.current.error.innerHTML = "error (count): " + stat.current[event][type].error;
}

function updateBestStat() {
    var currentStat = stat.current[event][type];
    var currentBest = stat.best[event][type];
    if (event == "relay") {
        var newBest = betterStat(currentStat, currentBest, ["wpm", "error", "backspace", "time"]);
    }
    else if (event == "precision") {
        var newBest = betterStat(currentStat, currentBest, ["wpm", "error", "time"]);
    }
    for (thing in currentBest) {
        currentBest[thing] = newBest[thing];
        window.sessionStorage["typeOMeter_stat_" + event + "_" + type + "_" + thing] = newBest[thing];
    }

    statText.best.wpm.innerHTML = "wpm (word/min): " + stat.best[event][type].wpm;
    statText.best.time.innerHTML = "time (min): " + stat.best[event][type].time;
    statText.best.backspace.innerHTML = "backspace (count): " + stat.best[event][type].backspace;
    statText.best.error.innerHTML = "error (count): " + stat.best[event][type].error;
}
```

```
function betterStat(stat1, stat2, criteriaArray) {
    var currentCriteria = criteriaArray[0];

    // [wpm, time]: bigger is better
    // [error, backspace]: smaller is better

    var reverseModifer = 1; // reverse logic
    if (["wpm", "time"].includes(currentCriteria)) {
        reverseModifier = -1;
    }
    var stat1_value = stat1[currentCriteria] * reverseModifier;
    var stat2_value = stat2[currentCriteria] * reverseModifier;

    if (stat1_value < stat2_value) {
        return stat1;
    }
    else if (stat1_value > stat2_value) {
        return stat2;
    }
    else if (criteriaArray.length == 1) {
        return stat1;
    }
    else {
        criteriaArray.shift();
        return betterStat(stat1, stat2, criteriaArray);
    }
}


/* Center Container Function */

function centerContainer() {
    // Get container's dimension
    var containerWidth = Number(containerStyle.getPropertyValue("width").replace("px", ""));
    var containerHeight = Number(containerStyle.getPropertyValue("height").replace("px", ""));
    // Calculate offset (minimum of 0)
    var offsetX = Math.max(0, (window.innerWidth - containerWidth)/2);
    var offsetY = Math.max(0, (window.innerHeight - containerHeight)/2);
    // Apply offset to container
    container.setAttribute("style", "transform: translate(" + offsetX + "px, " + offsetY + "px);");
}


// Helper Function

function clearInput() {
    userInput.value = "";
}

function floor_nDecimal(num, n) {
    return Math.floor(num * Math.pow(10, n)) / Math.pow(10, n);
}

function capitalize(string, all) {
    if (all) {
        var wordArray = string.split(" ");
        var length = wordArray.length;
        for (x=0; x<length; x++) {
            wordArray[x] = capitalize(wordArray[x], false);
        }
        return wordArray.join(" ");
    }
    else {
        return string[0].toUpperCase() + string.substr(1);
    }
}
```

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Type-o-meter</title>
    <link href="style.css" rel="stylesheet">
    <link rel="shortcut icon" href="assets/favicon/lightbulb.png" type="image/x-icon">
  </head>
  <body onresize="centerContainer();">
    <div id="container">
      <div id="introScreen">
        <h1>Type-o-meter</h1>
        <br>
        <h2>Select Event:</h2>
        <center>
          <button id="relayButton" class="selected" onclick="setEvent('relay');">Relay</button>
          <button id="precisionButton" onclick="setEvent('precision');">Precision</button>
          <button id="startButton" onclick="setupEvent();">Start</button>
        </center>
        <br>
        <h2>Select Type:</h2>
        <center>
          <button id="quoteButton" class="selected" onclick="setType('quote');">Quote</button>
          <button id="triviaButton" onclick="setType('trivia');">Trivia</button>
          <button id="paragraphButton" onclick="setType('paragraph');">Paragraph</button>
          <button id="wordButton" onclick="setType('word');">Word</button>
          <button id="characterButton" onclick="setType('character');">Character</button>
        </center>
      </div>
      <div id="eventScreen" style="display: none;">
        <h1>Type-o-meter</h1>
        <div id="quoteSpace">
          <p id="targetText">...</p>
          <p id="targetSourceText">...</p>
        </div>
        <br>
        <div id="inputSpace">
          <textarea type="text" id="userInput" onkeypress="checkUserInput(event);" oninput="checkUserInput_precision();
resizeInputField(this, false);" onfocus="resizeInputField(this, false);" onblur="resizeInputField(this, true);"
onpaste="penalizePaste(event);" placeholder="Type quote here."></textarea>
          <button id="continueButton" onclick="checkUserInput({keyCode: 13, preventDefault: function()
{}});">Enter</button>
        </div>
        <p id="targetAnswerText">...</p>
      </div>
      <div id="statScreen">
        <h1>Stats:</h1>
        <h2>Current:</h2>
        <p id="currentStat_wpm">wpm (word/min): 0</p>
        <p id="currentStat_time">time (min): 0</p>
        <p id="currentStat_backspace">backspace (count): 0</p>
        <p id="currentStat_error">error (count): 0</p>
        <br>
        <h2>Best:</h2>
        <p id="bestStat_wpm">wpm (word/min): 0</p>
        <p id="bestStat_time">time (min): 0</p>
        <p id="bestStat_backspace">backspace (count): 0</p>
        <p id="bestStat_error">error (count): 0</p>
        <br>
        <button id="exitButton" onclick="exitEvent();">Exit</button>
      </div>
    </div>

    <script src="main.js"></script>
  </body>
</html>
```

CSS

```css
/* Font */

@font-face {
    font-family: "chakraPetch";
    src: url("assets/fonts/chakraPetch/font.ttf") format("truetype");
}


/* Variable */

:root {
  --container-width: 1200px;
  --container-height: 640px;
    --screen-padding: 15px;

    --button-width: 120px;
    --button-border: 1.5px;
    --input-padding: 3px;
}


/* Main Body */

html, body {
    background-color: black;
    margin: 0;
    padding: 0;

    color: white;
    font-family: chakraPetch;
}

#container {
    width: var(--container-width);
    height: var(--container-height);
    border: 2px solid white;

  display: flex;
  flex-wrap: wrap;
}

#introScreen, #eventScreen {
    flex: 4;
    padding: var(--screen-padding);
}

#statScreen {
    flex: 1;
    border-left: 2px solid white;
    padding: var(--screen-padding);
}


/* Text */

h1 {
    text-align: center;
}


/* Input Field */

#inputSpace {
    position: relative;
}
```

```css
textarea {
    width: 300px;
    margin: auto;
    padding: 0px 3px;

    font-family: chakraPetch;
    text-overflow: ellipsis;

    opacity: 0.85;
    transition: width 0.7s, opacity 0.7s;

    /* for auto-resize feature */
    resize: none;
    overflow: auto;
    min-height: 20px;
}

textarea:hover {
    opacity: 1;
}

textarea:focus {
    width: calc(100% - var(--button-width) - var(--screen-padding) - 2*var(--input-padding));
    opacity: 1;
    outline: none;
}


/* Button */

button {
    width: var(--button-width);
    height: 24px;
    background-color: black;
    border: var(--button-border) solid white;
    margin: 0px;

    color: white;
    font-family: chakraPetch;

    transition: all 0.5s ease 0s;
    cursor: pointer;
}

button:hover {
    color: gold;
    border: var(--button-border) solid white;
}

button:focus {
    outline: calc(2*var(--button-border)) double gold;
}

.selected {
    color: gold;
    border: var(--button-border) solid gold;
}

#startButton {
    margin-left: 50px;
}

#startButton:hover {
    color: springgreen;
}
```

```css
#startButton:focus {
    color: springgreen;
    outline: calc(var(--button-border)) double springgreen;
    margin-left: 0px;
}

#continueButton {
    position: absolute;
    top: 0px;
    margin-left: 10px;
}

#exitButton:hover {
    color: red;
}

#exitButton:focus {
    color: red;
    outline: calc(var(--button-border)) double red;
}


/* Quote Space */

#quoteSpace {
    width: 80%;
    margin: auto;
}

#targetText {
    font-size: 20px;
    text-indent: 30px;
}

#targetText::before {
    position: relative;
    display: inline;
    height: 0;
    top: 20px;
    left: -5px;

    content: open-quote;
    font-family: sans-serif;
    font-size: 64px;
    line-height: 0;
}

#targetText::after {
    position: relative;
    display: inline;
    height: 0;
    top: 25px;
    left: 5px;

    content: close-quote;
    font-family: sans-serif;
    font-size: 64px;
    line-height: 0;
}

.targetIgnore {
    color: darkgrey;
}

.targetEmphasis {
    color: gold;
}
```

```css
#targetSourceText {
    margin-top: -20px;
    text-align: right;
}

#targetSourceText::before {
    content: "- ";
}

#targetAnswerText {
    position: absolute;
    bottom: calc(var(--screen-padding)/2);
    left: calc(var(--screen-padding)/-2);
    padding-top: 10px;

    color: red;
    background-color: black;
    writing-mode: vertical-rl;
}
```

.