**HTML, CSS, JAVASCRIPT QUESTIONS- CODES**

1. The Calculator

* Write a function called squareNumber that will take one argument (a number), square that number, and return the result. It should also log a string like "The result of squaring the number 3 is 9."
* Write a function called halfNumber that will take one argument (a number), divide it by 2, and return the result. It should also log a string like "Half of 5 is 2.5.".
* Write a function called percentOf that will take two numbers, figure out what percent the first number represents of the second number, and return the result. It should also log a string like "2 is 50% of 4."
* Write a function called areaOfCircle that will take one argument (the radius), calculate the area based on that, and return the result. It should also log a string like "The area for a circle with radius 2 is 12.566370614359172."
  + Bonus: Round the result so there are only two digits after the decimal.
* Write a function that will take one argument (a number) and perform the following operations, using the functions you wrote earlier1:
  + Take half of the number and store the result.
  + Square the result of #1 and store that result.
  + Calculate the area of a circle with the result of #2 as the radius.
  + Calculate what percentage that area is of the squared result (#3).

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>JavaScript Calculator</title>

  <link rel="stylesheet" href="styles.css"> </head>

<body>

  <h1>JavaScript Calculator</h1>

  <input type="number" id="inputNumber" placeholder="Enter a number">

  <button onclick="complexCalculation()">Calculate</button>

  <div id="output"></div>

  <script src="script.js"></script> </body>

</html>

**script.js**

function squareNumber(number) {

    const result = number \* number;

    console.log(`The result of squaring the number ${number} is ${result}.`);

    return result;

  }

  function halfNumber(number) {

    const result = number / 2;

    console.log(`Half of ${number} is ${result.toFixed(2)}.`);

    return result;

  }

  function percentOf(num1, num2) {

    const result = (num1 / num2) \* 100;

    console.log(`${num1} is ${result.toFixed(2)}% of ${num2}.`);

    return result;

  }

  function areaOfCircle(radius) {

    const area = Math.PI \* radius \* radius;

    console.log(`The area for a circle with radius ${radius} is ${area.toFixed(2)}.`);

    return area;

  }

  function complexCalculation() {

    const inputNumber = document.getElementById("inputNumber").value;

    if (isNaN(inputNumber)) {

      alert("Please enter a valid number!");

      return;

    }

    const number = parseFloat(inputNumber); // Convert input to number

    const half = halfNumber(number);

    const squaredHalf = squareNumber(half);

    const circleArea = areaOfCircle(squaredHalf);

    const percentOfSquared = percentOf(circleArea, squaredHalf);

    const outputText = `

      ----- Complex Calculation Results -----<br>

      Half of ${number} is ${half.toFixed(2)}.<br>

      Squaring ${half.toFixed(2)} gives ${squaredHalf}.<br>

      The area of the circle with radius ${squaredHalf.toFixed(2)} is ${circleArea.toFixed(2)}.<br>

      ${circleArea.toFixed(2)} is ${percentOfSquared.toFixed(2)}% of ${squaredHalf}.

    `;

    document.getElementById("output").innerHTML = outputText;

  }

**style.css**

body {

    font-family: sans-serif;

    text-align: center;

    padding: 100px;

  }

  h1 {

    margin-top: 20px;

  }

  input {

    padding: 10px;

    margin: 10px;

    border: 1px solid #ccc;

  }

  button {

    padding: 10px 20px;

    background-color: #4CAF50;

    color: white;

    border: none;

    cursor: pointer;

  }

  #output {

    margin-top: 20px;

    padding: 10px;

    border: 1px solid #ccc;

  }

2. MixUp

* Create a function called mixUp. It should take in two strings, and return the concatenation of the two strings (separated by a space) slicing out and swapping the first 2 characters of each. You can assume that the strings are at least 2 characters long. For example:
* mixUp('mix', pod'): 'pox mid'
* mixUp('dog', 'dinner'): 'dig donner'

index.html

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>MixUp Function</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>MixUp Function</h1>

  <input type="text" id="inputStr1" placeholder="Enter first string">

  <input type="text" id="inputStr2" placeholder="Enter second string">

  <button onclick="mixUpStrings()">Mix Up!</button>

  <div id="output"></div>

  <script src="script.js"></script>

</body>

</html>

style.css

body {

    font-family: sans-serif;

    text-align: center;

  }

  h1 {

    margin-top: 20px;

  }

  input {

    padding: 10px;

    margin: 10px;

    border: 1px solid #ccc;

  }

  button {

    padding: 10px 20px;

    background-color: #4CAF50;

    color: white;

    border: none;

    cursor: pointer;

  }

  #output {

    margin-top: 20px;

    padding: 10px;

    border: 1px solid #ccc;

  }

script.js

function mixUp(str1, str2) {

    if (str1.length < 2 || str2.length < 2) {

      return "Invalid input: Both strings must be at least 2 characters long.";

    }

    const first2Str1 = str1.slice(0, 2);

    const remainingStr1 = str1.slice(2);

    const first2Str2 = str2.slice(0, 2);

    const remainingStr2 = str2.slice(2);

    return first2Str2 + remainingStr1 + " " + first2Str1 + remainingStr2;

  }

  function mixUpStrings() {

    const inputStr1 = document.getElementById("inputStr1").value;

    const inputStr2 = document.getElementById("inputStr2").value;

    const mixedUpString = mixUp(inputStr1, inputStr2);

    document.getElementById("output").textContent = mixedUpString;

  }

### 3. FixStart

* Create a function called fixStart. It should take a single argument, a string, and return a version where all occurences of its first character have been replaced with '\*', except for the first character itself. You can assume that the string is at least one character long. For example:
* fixStart('babble'): 'ba\*\*le'

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Fix Start Function</title>

    <link rel="stylesheet" href="style.css"> </head>

<body>

    <h1>Fix Start Function</h1>

    <p>Enter a string:</p>

    <input type="text" id="inputString">

    <button id="fixStartButton">Fix String</button>

    <p id="output"></p>

    <script src="script.js"></script>  </body>

</html>

**Script.js**

function fixStart(str) {

    // Handle empty string or strings with only one character

    if (str.length <= 1) {

      return str;

    }

    const firstChar = str[0];

    const restOfString = str.slice(1);

    const replacedString = restOfString.replace(new RegExp(firstChar, 'g'), '\*');

    return firstChar + replacedString;

  }

  function getInputValue() {

    const inputElement = document.getElementById("inputString");

    return inputElement.value;

  }

  function setOutputValue(output) {

    const outputElement = document.getElementById("output");

    outputElement.textContent = output;

  }

  // Add event listener after the DOM is loaded (ensures button exists)

  window.onload = function() {

    const fixStartButton = document.getElementById('fixStartButton');

    fixStartButton.addEventListener('click', function() {

      const userInput = getInputValue();

      const fixedString = fixStart(userInput);

      setOutputValue(fixedString);

    });

  };

### 4. Verbing

* Create a function called verbing. It should take a single argument, a string. If its length is at least 3, it should add 'ing' to its end, unless it already ends in 'ing', in which case it should add 'ly' instead. If the string length is less than 3, it should leave it unchanged. For example:
* verbing('swim'): 'swimming'
* verbing('swimming'): 'swimmingly'
* verbing('go'): 'go'

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Verbing Function</title>

    <link rel="stylesheet" href="style.css"> </head>

<body>

    <h1>Verbing Function</h1>

    <p>Enter a string:</p>

    <input type="text" id="inputString">

    <button id="verbingButton">Verbing</button>

    <p id="output"></p>

    <script src="script.js"></script>  </body>

</html>

**Script.js**

function verbing(str) {

    // Handle empty string or strings less than 3 characters

    if (str.length < 3) {

      return str;

    }

    const ending = str.slice(-3); // Get the last 3 characters

    if (ending === "ing") {

      return str + "ly";

    } else {

      return str + "ing";

    }

  }

  function getInputValue() {

    const inputElement = document.getElementById("inputString");

    return inputElement.value;

  }

  function setOutputValue(output) {

    const outputElement = document.getElementById("output");

    outputElement.textContent = output;

  }

  // Add event listener after the DOM is loaded (ensures button exists)

  window.onload = function() {

    const verbingButton = document.getElementById('verbingButton');

    verbingButton.addEventListener('click', function() {

      const userInput = getInputValue();

      const verbedString = verbing(userInput);

      setOutputValue(verbedString);

    });

  };

### 5. Not Bad

* Create a function called notBad that takes a single argument, a string.
* It should find the first appearance of the substring 'not' and 'bad'.
* If the 'bad' follows the 'not', then it should replace the whole 'not'...'bad' substring with 'good' and return the result.
* If it doesn't find 'not' and 'bad' in the right sequence (or at all), just return the original sentence.

For example:

* notBad('This dinner is not that bad!'): 'This dinner is good!'
* notBad('This movie is not so bad!'): 'This movie is good!'
* notBad('This dinner is bad!'): 'This dinner is bad!'

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Not Bad Function</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>Not Bad Function</h1>

    <p>Enter a sentence:</p>

    <input type="text" id="inputSentence">

    <button id="fixSentenceButton">Fix Sentence</button>

    <p id="output"></p>

    <script src="script.js"></script>

</body>

</html>

**Script.js**

function notBad(sentence) {

    // Find the index of 'not' and 'bad' (considering case sensitivity)

    const notIndex = sentence.toLowerCase().indexOf('not');

    const badIndex = sentence.toLowerCase().indexOf('bad');

    // Check if 'not' and 'bad' are found and in the correct order

    if (notIndex !== -1 && badIndex !== -1 && badIndex > notIndex) {

      return sentence.slice(0, notIndex) + 'good' + sentence.slice(badIndex + 3);

    } else {

      return sentence;

    }

  }

  function getInputValue() {

    const inputElement = document.getElementById("inputSentence");

    return inputElement.value;

  }

  function setOutputValue(output) {

    const outputElement = document.getElementById("output");

    outputElement.textContent = output;

  }

  // Ensure script execution after window.onload

  window.onload = function() {

    const fixSentenceButton = document.getElementById('fixSentenceButton');

    // Error handling (optional): Check for button existence

    if (fixSentenceButton) {

      fixSentenceButton.addEventListener('click', function() {

        const userInput = getInputValue();

        const fixedSentence = notBad(userInput);

        setOutputValue(fixedSentence);

      });

    } else {

      console.error("Button element not found (fixSentenceButton)");

    }

  };

**------------------------------------------------------------------------------------------------------------------------**

### 6. The Pluralizer

* Write a function named pluralize that:
  + takes 2 arguments, a noun and a number.
  + returns the number and pluralized form, like "5 cats" or "1 dog".
* Call that function for a few different scores and log the result to make sure it works.
* Bonus: Make it handle a few collective nouns like "sheep" and "geese".

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Pluralizer</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>Pluralizer</h1>

  <p>Enter a number and a noun:</p>

  <input type="number" id="numberInput" placeholder="Number">

  <input type="text" id="nounInput" placeholder="Noun">

  <button id="pluralizeButton">Pluralize</button>

  <p id="output"></p>

  <script src="script.js"></script>

</body>

</html>

**Script.js**

const pluralize = (number, noun) => {

    // Handle irregular plurals and collective nouns (consider expanding this as needed)

    const irregularPlurals = {

      sheep: 'sheep',

      goose: 'geese',

      mouse: 'mice',

      person: 'people',

      child: 'children',

      deer: 'deer',

      species: 'species',

      series: 'series',

      fish: 'fish',

      information: 'information',

      equipment: 'equipment',

    };

    if (irregularPlurals[noun.toLowerCase()]) {

      return `${number} ${irregularPlurals[noun.toLowerCase()]}`;

    }

    // Handle singular and regular plurals

    const pluralForm = number === 1 ? noun : noun.endsWith('s') || noun.endsWith('x') ? noun : noun + 's';

    return `${number} ${pluralForm}`;

  };

  const getInputValue = (id) => document.getElementById(id).value;

  const setOutputValue = (value) => {

    const outputElement = document.getElementById('output');

    outputElement.textContent = value;

  };

  document.addEventListener('DOMContentLoaded', () => {

    const pluralizeButton = document.getElementById('pluralizeButton');

    pluralizeButton.addEventListener('click', () => {

      const number = parseInt(getInputValue('numberInput'));

      const noun = getInputValue('nounInput');

      if (isNaN(number)) {

        setOutputValue('Please enter a valid number.');

        return;

      }

      const pluralizedSentence = pluralize(number, noun);

      setOutputValue(pluralizedSentence);

    });

  });

### The Array

* Create an array to hold your top choices (colors, presidents, whatever).
* For each choice, log to the screen a string like: "My #1 choice is blue."
* **Bonus:** Change it to log "My 1st choice, "My 2nd choice", "My 3rd choice", picking the right suffix for the number based on what it is.

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Top Choices</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>My Top Choices</h1>

  <ul id="choicesList"></ul>

  <input type="text" id="choiceInput" placeholder="Enter your choice">

  <button id="addButton">Add Choice</button>

  <script src="script.js"></script>

</body>

</html>

**Script.js**

function getOrdinalSuffix(number) {

    const ones = number % 10;

    const tens = Math.floor(number / 10) % 10;

    if (tens === 1) {

      return "th";

    } else if (ones === 1) {

      return "st";

    } else if (ones === 2) {

      return "nd";

    } else if (ones === 3) {

      return "rd";

    } else {

      return "th";

    }

  }

  const choicesList = document.getElementById('choicesList');

  const choiceInput = document.getElementById('choiceInput');

  const addButton = document.getElementById('addButton');

  let choices = []; // Array to store user-entered choices

  addButton.addEventListener('click', () => {

    const newChoice = choiceInput.value.trim(); // Trim whitespace

    if (!newChoice) { // Check for empty input

      alert('Please enter a choice.');

      return;

    }

    choices.push(newChoice); // Add choice to the array

    choiceInput.value = ''; // Clear the input field

    updateChoicesList();

  });

  function updateChoicesList() {

    choicesList.innerHTML = ''; // Clear previous entries

    for (let i = 0; i < choices.length; i++) {

      const ordinal = i + 1;

      const suffix = getOrdinalSuffix(ordinal);

      const listItem = document.createElement('li');

      listItem.textContent = `My ${ordinal}${suffix} choice is ${choices[i]}.`;

      choicesList.appendChild(listItem);

    }

  }

**style.css**

body {

    font-family: sans-serif;

    margin: 20px;

    text-align: center; /\* Center elements horizontally \*/

  }

  ul {

    list-style: none;

    margin: 0 auto; /\* Center the list horizontally \*/

    width: 300px; /\* Set fixed width for the list \*/

    border: 1px solid #ddd; /\* Simple border \*/

    padding: 10px; /\* Padding inside the list \*/

  }

  button {

    padding: 5px; /\* Consistent padding for input and button \*/

    border: 1px solid #ccc; /\* Simple border \*/

    border-radius: 3px; /\* Rounded corners \*/

    margin-bottom: 10px; /\* Spacing below input and button \*/

  }

  button {

    background-color: #4CAF50; /\* Green background for button \*/

    color: white; /\* White text for button \*/

  }

### Simple word guessing game

You'll create a simple word guessing game where the user gets infinite tries to guess the word (like Hangman without the hangman, or like Wheel of Fortune without the wheel and fortune).

* Create two global arrays: one to hold the letters of the word (e.g. 'F', 'O', 'X'), and one to hold the current guessed letters (e.g. it would start with '\_', '\_', '\_' and end with 'F', 'O', 'X').
* Write a function called guessLetter that will:
  + Take one argument, the guessed letter.
  + Iterate through the word letters and see if the guessed letter is in there.
  + If the guessed letter matches a word letter, changed the guessed letters array to reflect that.
  + When it's done iterating, it should log the current guessed letters ('F\_\_')
  + and congratulate the user if they found a new letter.
  + It should also figure out if there are any more letters that need to be guessed,
  + and if not, it should congratulate the user for winning the game.
* Pretend you don't know the word, and call guessLetter multiple times with various letters to check that your program works.
* **Bonus:** Make it more like Wheel of Fortune:
  + Start with a reward amount of $0
  + Every time a letter is guessed, generate a random amount and reward the user if they found a letter (multiplying the reward if multiple letters found), otherwise subtract from their reward.
  + When they guess the word, log their final reward amount.
* **Bonus:** Make it like Hangman:
  + Keep track of all the guessed letters (right and wrong) and only let the user guess a letter once. If they guess a letter twice, do nothing.
  + Keep track of the state of the hangman as a number (starting at 0), and subtract or add to that number every time they make a wrong guess.
  + Once the number reaches 6 (a reasonable number of body parts for a hangman), inform the user that they lost and show a hangman on the log.

**Index.html**<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Word Guessing Game</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <h1>Word Guessing Game</h1>

  <div id="game-container">

    <p id="hidden-word"></p>

    <p id="reward">Reward: $<span id="reward-amount">0</span></p>

    <input type="text" id="guess-input" placeholder="Guess a letter">

    <button id="guess-button">Guess</button>

  </div>

  <script src="script.js"></script>

</body>

</html>

**Script.js**

// Global variables

let word = ['F', 'O', 'X']; // Replace with your word

let guessedLetters = word.map(() => '\_'); // Hidden word placeholders

let reward = 0;

let wrongGuesses = new Set(); // Keeps track of used letters

// Function to guess a letter

function guessLetter(letter) {

  letter = letter.toUpperCase(); // Ensure uppercase for case-insensitive matching

  let found = false;

  // Check if letter is already guessed

  if (wrongGuesses.has(letter) || guessedLetters.includes(letter)) return;

  // Check for matching letters

  for (let i = 0; i < word.length; i++) {

    if (word[i] === letter) {

      guessedLetters[i] = letter;

      found = true;

    }

  }

  // Update game state based on guess

  if (found) {

    reward += Math.floor(Math.random() \* 100); // Random reward for each letter

    console.log(`Great guess! You found a letter. Current word: ${guessedLetters.join('')}`);

    // Check for win condition

    if (guessedLetters.join('') === word.join('')) {

      console.log(`Congratulations! You guessed the word and won with a reward of $${reward}`);

      guessButton.disabled = true; // Disable guess button on win

      return; // Exit function early on win

    }

  } else {

    wrongGuesses.add(letter);

    console.log(`Oops! '${letter}' is not in the word.`);

  }

}

// DOM elements

const hiddenWordElement = document.getElementById('hidden-word');

const rewardAmountElement = document.getElementById('reward-amount');

const guessInput = document.getElementById('guess-input');

const guessButton = document.getElementById('guess-button');

// Start game with hidden word

hiddenWordElement.textContent = guessedLetters.join('');

// Function to handle guess button click

guessButton.addEventListener('click', () => {

  const guess = guessInput.value.trim();

  if (guess) {

    guessLetter(guess);

    guessInput.value = ''; // Clear input field after guess

    hiddenWordElement.textContent = guessedLetters.join('');

    rewardAmountElement.textContent = reward;

  }

});

**Style.css**

body {

    font-family: sans-serif;

    margin: 0;

    padding: 20px;

  }

  h1 {

    text-align: center;

  }

  #game-container {

    display: flex;

    flex-direction: column;

    align-items: center;

    gap: 10px;

  }

  #hidden-word {

    font-size: 24px;

  }

  #reward {

    font-weight: bold;

  }

  #guess-input {

    padding: 5px;

    border: 1px solid #ccc;

  }

  #guess-button {

    padding: 10px 20px;

    background-color: #4CAF50;

    color: white;

    border: none;

    cursor: pointer;

  }

  #guess-button:hover {

    background-color: #3e8e41;

  }