IPrompt1= f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

. \*\*Customizable Countdown Timer\*\*

\*\*Description:\*\* Create a customizable countdown timer that users can set for different events.

\*\*Skills Tested:\*\* DOM manipulation, time management, event handling.

\*\*Features:\*\*

- Allow users to input a countdown time.

- Display the remaining time in hours, minutes, and seconds.

- Play a sound or show an alert when the timer reaches zero.

- Allow multiple timers to run simultaneously.

"""

Answer

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Customizable Countdown Timer</title>

</head>

<body>

<h1>Customizable Countdown Timer</h1>

<div>

<label for="timer-input">Set Timer (in seconds): </label>

<input type="number" id="timer-input" min="1">

<button id="start-timer">Start Timer</button>

</div>

<div id="timers-container"></div>

<script>

document.getElementById('start-timer').addEventListener('click', function() {

let inputTime = document.getElementById('timer-input').value;

if (inputTime <= 0) {

alert("Please enter a positive number.");

return;

}

let timerContainer = document.createElement('div');

timerContainer.className = 'timer';

document.getElementById('timers-container').appendChild(timerContainer);

let remainingTime = inputTime;

let intervalId = setInterval(function() {

let hours = Math.floor(remainingTime / 3600);

let minutes = Math.floor((remainingTime % 3600) / 60);

let seconds = remainingTime % 60;

timerContainer.innerHTML = `${hours}h ${minutes}m ${seconds}s`;

if (remainingTime < 0) {

clearInterval(intervalId);

alert('Timer finished!');

}

remainingTime--;

}, 1000);

});

</script>

</body>

</html>

```

### Intentional Mistakes

1. \*\*Time reaches zero:\*\* The condition to clear the interval and alert the user should be `remainingTime <= 0` instead of `remainingTime < 0`.

2. \*\*Input validation:\*\* The input validation checks for `<= 0`, but the input field is of type `number` which defaults to `0` if empty. We should handle empty input separately.

3. \*\*Negative input:\*\* The input type allows negative numbers because the `min` attribute is not enforced programmatically.

4. \*\*Multiple timers:\*\* The code does not handle multiple timers properly because it doesn't create unique interval IDs for each timer.

### Corrected Code

Below is the corrected version of the code:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Customizable Countdown Timer</title>

</head>

<body>

<h1>Customizable Countdown Timer</h1>

<div>

<label for="timer-input">Set Timer (in seconds): </label>

<input type="number" id="timer-input" min="1">

<button id="start-timer">Start Timer</button>

</div>

<div id="timers-container"></div>

<script>

document.getElementById('start-timer').addEventListener('click', function() {

let inputTime = document.getElementById('timer-input').value;

if (inputTime === "" || inputTime <= 0) {

alert("Please enter a positive number.");

return;

}

let timerContainer = document.createElement('div');

timerContainer.className = 'timer';

document.getElementById('timers-container').appendChild(timerContainer);

let remainingTime = parseInt(inputTime, 10);

let intervalId = setInterval(function() {

let hours = Math.floor(remainingTime / 3600);

let minutes = Math.floor((remainingTime % 3600) / 60);

let seconds = remainingTime % 60;

timerContainer.innerHTML = `${hours}h ${minutes}m ${seconds}s`;

if (remainingTime <= 0) {

clearInterval(intervalId);

alert('Timer finished!');

}

remainingTime--;

}, 1000);

});

</script>

</body>

</html>

```

This corrected code resolves the issues and properly handles multiple timers, input validation, and the condition when the timer reaches zero.

IPrompt2 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

\*\*Interactive Quiz App\*\*

\*\*Description:\*\* Develop an interactive quiz application that presents questions and tracks scores.

\*\*Skills Tested:\*\* DOM manipulation, event handling, data handling.

\*\*Features:\*\*

- Load questions from a JSON file or an API.

- Present one question at a time with multiple-choice answers.

- Track and display the user's score at the end of the quiz.

- Show feedback for correct and incorrect answers.

"""

IPrompt3 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

\*\*Recipe Ingredient Calculator\*\*

\*\*Description:\*\* Create an ingredient calculator that adjusts ingredient quantities based on the number of servings.

\*\*Skills Tested:\*\* DOM manipulation, event handling, arithmetic operations.

\*\*Features:\*\*

- Input recipe ingredients with quantities for a default number of servings.

- Allow users to adjust the number of servings.

- Automatically update ingredient quantities based on the new serving size.

"""

Sure, here is a version of the Interactive Quiz App with intentional mistakes that can be used to test the interviewee's skills in DOM manipulation, event handling, and data handling:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Quiz App</title>

</head>

<body>

<div id="quiz-container">

<h1>Interactive Quiz</h1>

<div id="question-container">

<p id="question"></p>

<ul id="answers">

<!-- Answers will be dynamically inserted here -->

</ul>

</div>

<button id="next-button" onclick="nextQuestion()">Next</button>

<p id="score"></p>

</div>

<script>

const questions = [

{

question: "What is the capital of France?",

answers: ["Berlin", "Madrid", "Paris", "Lisbon"],

correct: 2

},

{

question: "Which planet is known as the Red Planet?",

answers: ["Earth", "Mars", "Jupiter", "Saturn"],

correct: 1

},

{

question: "What is the largest ocean on Earth?",

answers: ["Atlantic", "Indian", "Arctic", "Pacific"],

correct: 3

}

];

let currentQuestionIndex = 0;

let score = 0;

function loadQuestion() {

const questionElement = document.getElementById('question');

const answersElement = document.getElementById('answers');

const currentQuestion = questions[currentQuestionIndex];

questionElement.innerText = currentQuestion.question;

answersElement.innerHTML = '';

currentQuestion.answers.forEach((answer, index) => {

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

li.innerText = answer;

li.setAttribute('data-index', index);

li.addEventListener('click', checkAnswer);

answersElement.appendChild(li);

});

}

function checkAnswer(event) {

const selectedAnswerIndex = parseInt(event.target.getAttribute('data-index'));

const currentQuestion = questions[currentQuestionIndex];

if (selectedAnswerIndex === currentQuestion.correct) {

score++;

alert('Correct!');

} else {

alert('Incorrect!');

}

document.getElementById('score').innerText = `Score: ${score}`;

}

function nextQuestion() {

currentQuestionIndex++;

if (currentQuestionIndex >= questions.length) {

alert('Quiz finished!');

document.getElementById('next-button').disabled = true;

} else {

loadQuestion();

}

}

// Load the first question

window.onload = loadQuestion;

</script>

</body>

</html>

```

Here are the intentional mistakes introduced in the code:

1. The `score` is being updated and displayed inside the `checkAnswer` function, which might not always correctly reflect the state when moving to the next question.

2. The `nextQuestion` function does not disable the answer buttons, which means the user can still click on them even after moving to the next question.

3. There's no mechanism to restart the quiz or properly handle the end of the quiz beyond disabling the "Next" button.

4. The `nextQuestion` function does not remove previous event listeners from the answer elements, potentially causing unexpected behavior.

The interviewee should identify these issues and correct them to ensure the application works as expected.

IPrompt4 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

\*\*Budget Planning App\*\*

\*\*Description:\*\* Build an app to help users plan their monthly budget by categorizing expenses and incomes.

\*\*Skills Tested:\*\* Data management, DOM manipulation, event handling.

\*\*Features:\*\*

- Allow users to add, edit, and delete income and expenses.

- Categorize income and expenses (e.g., rent, groceries, entertainment).

- Display a summary of total income, total expenses, and balance.

- Visualize data using charts (e.g., pie chart for expense categories).

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Recipe Ingredient Calculator</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.ingredient {

margin-bottom: 10px;

}

</style>

</head>

<body>

<h1>Recipe Ingredient Calculator</h1>

<div>

<label for="servings">Number of servings:</label>

<input type="number" id="servings" value="4">

</div>

<div id="ingredients">

<div class="ingredient">

<span class="name">Flour:</span>

<span class="quantity">2</span> cups

</div>

<div class="ingredient">

<span class="name">Sugar:</span>

<span class="quantity">1.5</span> cups

</div>

<div class="ingredient">

<span class="name">Butter:</span>

<span class="quantity">1</span> cup

</div>

</div>

<script>

document.getElementById('servings').addEventListener('input', function() {

var newServings = parseInt(this.value);

if (isNaN(newServings) || newServings <= 0) {

return;

}

var defaultServings = 4;

var ingredients = document.querySelectorAll('.ingredient');

ingredients.forEach(function(ingredient) {

var quantityElement = ingredient.querySelector('.quantity');

var quantity = parseFloat(quantityElement.textContent);

var updatedQuantity = (quantity / defaultServings) \* newServings;

quantityElement.textContent = updatedQuantity.toFixed(1);

});

});

</script>

</body>

</html>

```

### Intentional Mistakes:

1. The `defaultServings` variable is hardcoded inside the event listener. It should be outside to avoid recalculating it every time the input changes.

2. The `parseInt(this.value)` should be `parseFloat(this.value)` to handle fractional servings correctly.

3. The event listener should handle the case where the input is emptied or invalid, by resetting the ingredient quantities to their original values.

### Corrected Code:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Recipe Ingredient Calculator</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.ingredient {

margin-bottom: 10px;

}

</style>

</head>

<body>

<h1>Recipe Ingredient Calculator</h1>

<div>

<label for="servings">Number of servings:</label>

<input type="number" id="servings" value="4">

</div>

<div id="ingredients">

<div class="ingredient">

<span class="name">Flour:</span>

<span class="quantity">2</span> cups

</div>

<div class="ingredient">

<span class="name">Sugar:</span>

<span class="quantity">1.5</span> cups

</div>

<div class="ingredient">

<span class="name">Butter:</span>

<span class="quantity">1</span> cup

</div>

</div>

<script>

var defaultServings = 4;

var originalQuantities = [];

document.querySelectorAll('.quantity').forEach(function(element) {

originalQuantities.push(parseFloat(element.textContent));

});

document.getElementById('servings').addEventListener('input', function() {

var newServings = parseFloat(this.value);

if (isNaN(newServings) || newServings <= 0) {

document.querySelectorAll('.quantity').forEach(function(element, index) {

element.textContent = originalQuantities[index];

});

return;

}

var ingredients = document.querySelectorAll('.ingredient');

ingredients.forEach(function(ingredient, index) {

var quantityElement = ingredient.querySelector('.quantity');

var updatedQuantity = (originalQuantities[index] / defaultServings) \* newServings;

quantityElement.textContent = updatedQuantity.toFixed(1);

});

});

</script>

</body>

</html>

```

This corrected code ensures that the default servings are defined outside the event listener, handles fractional servings correctly, and resets the ingredient quantities if the input is invalid.

IPrompt5 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

\*\*Dynamic Form Builder\*\*

\*\*Description:\*\* Create a dynamic form builder where users can add different types of form fields (text, checkbox, radio, etc.).

\*\*Skills Tested:\*\* DOM manipulation, data handling, event handling.

\*\*Features:\*\*

- Allow users to add, remove, and reorder form fields.

- Support different field types (text input, checkbox, radio button, dropdown).

- Display a preview of the form as fields are added.

- Output the form data as a JSON object upon submission.

"""

Answer

Certainly! Below is a sample code for a dynamic form builder application with intentional mistakes. The application should be able to be tested on platforms like HackerRank or Coderbyte.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Dynamic Form Builder</title>

<style>

.form-field {

margin-bottom: 10px;

}

.form-preview {

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Dynamic Form Builder</h1>

<div>

<label for="field-type">Field Type:</label>

<select id="field-type">

<option value="text">Text Input</option>

<option value="checkbox">Checkbox</option>

<option value="radio">Radio Button</option>

<option value="dropdown">Dropdown</option>

</select>

<button id="add-field">Add Field</button>

</div>

<div id="form-preview" class="form-preview"></div>

<button id="submit-form">Submit Form</button>

<script>

document.getElementById('add-field').addEventListener('click', function() {

const fieldType = document.getElementById('field-type').value;

const formPreview = document.getElementById('form-preview');

let newField;

switch (fieldType) {

case 'text':

newField = document.createElement('input');

newField.type = 'text';

newField.placeholder = 'Enter text';

break;

case 'checkbox':

newField = document.createElement('input');

newField.type = 'checkbox';

break;

case 'radio':

newField = document.createElement('input');

newField.type = 'radio';

break;

case 'dropdown':

newField = document.createElement('select');

const option1 = document.createElement('option');

option1.text = 'Option 1';

newField.add(option1);

const option2 = document.createElement('option');

option2.text = 'Option 2';

newField.add(option2);

break;

default:

break;

}

const formField = document.createElement('div');

formField.className = 'form-field';

formField.appendChild(newField);

const removeButton = document.createElement('button');

removeButton.textContent = 'Remove';

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

formPreview.removeChild(formField);

});

formField.appendChild(removeButton);

formPreview.appendChild(formField);

});

document.getElementById('submit-form').addEventListener('click', function() {

const fields = document.querySelectorAll('.form-field');

const formData = {};

fields.forEach((field, index) => {

const input = field.querySelector('input, select');

const fieldType = input.type || input.tagName.toLowerCase();

if (fieldType === 'text') {

formData[`field${index}`] = input.value;

} else if (fieldType === 'checkbox' || fieldType === 'radio') {

formData[`field${index}`] = input.checked;

} else if (fieldType === 'select') {

formData[`field${index}`] = input.value;

}

});

console.log(JSON.stringify(formData));

alert(JSON.stringify(formData));

});

</script>

</body>

</html>

```

Intentional Mistakes:

1. The `select` field type is not recognized correctly in the `submit-form` event listener.

2. The `select` field type is incorrectly referenced as `select` instead of `select-one` in the condition.

3. The `input` fields are not correctly indexed in the JSON output.

4. The `radio` buttons do not have a name attribute which means they cannot be grouped.

5. The `removeButton` is appended to the form field without a space between the input and the button.

6. The `form-preview` div will not clear previous elements when new fields are added.

IPrompt6 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 16. \*\*File Uploader with Preview\*\*

\*\*Description:\*\* Develop a file uploader that allows users to upload files and preview them before submission.

\*\*Skills Tested:\*\* File handling, DOM manipulation, event handling.

\*\*Features:\*\*

- Support uploading of multiple files.

- Preview images and videos before upload.

- Display file names and sizes for other file types.

- Provide a button to remove selected files from the list.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>File Uploader with Preview</title>

<style>

.file-preview {

margin-top: 20px;

}

.file-item {

display: flex;

align-items: center;

margin-bottom: 10px;

}

.file-item img, .file-item video {

max-width: 100px;

margin-right: 10px;

}

.file-item button {

margin-left: auto;

}

</style>

</head>

<body>

<h1>File Uploader with Preview</h1>

<input type="file" id="fileInput" multiple>

<div class="file-preview" id="filePreview"></div>

<script>

document.getElementById('fileInput').addEventListener('change', function(event) {

const files = event.target.files;

const previewContainer = document.getElementById('filePreview');

// Clear previous previews

previewContainer.innerHTML = '';

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

const file = files[i];

const fileItem = document.createElement('div');

fileItem.classList.add('file-item');

const fileName = document.createElement('p');

fileName.textContent = file.name;

fileItem.appendChild(fileName);

const fileSize = document.createElement('p');

fileSize.textContent = (file.size / 1024).toFixed(2) + ' KB';

fileItem.appendChild(fileSize);

if (file.type.startsWith('image/')) {

const img = document.createElement('img');

img.src = URL.createObjectURL(file);

fileItem.appendChild(img);

} else if (file.type.startsWith('video/')) {

const video = document.createElement('video');

video.src = URL.createObjectURL(file);

video.controls = true;

fileItem.appendChild(video);

}

const removeButton = document.createElement('button');

removeButton.textContent = 'Remove';

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

previewContainer.removeChild(fileItem);

});

fileItem.appendChild(removeButton);

previewContainer.appendChild(fileItem);

}

});

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*File Size Calculation:\*\* The file size is displayed in KB but the code `(file.size / 1024).toFixed(2) + ' KB'` is correct. To introduce a mistake, you can change the calculation to `(file.size / 1000).toFixed(2) + ' KB'` which is incorrect.

2. \*\*Event Listener:\*\* The event listener for removing files is correct but let's introduce an error by changing `previewContainer.removeChild(fileItem);` to `fileItem.remove();` which might not be supported in all browsers.

3. \*\*Video Preview:\*\* The video preview code is correct but let's introduce an error by changing `video.controls = true;` to `video.controls = false;` so the video controls won't appear.

4. \*\*File Input ID:\*\* The file input ID is `fileInput` but let's introduce an error by changing it to `fileUploader` in the JavaScript code.

Here's the modified code with intentional mistakes:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>File Uploader with Preview</title>

<style>

.file-preview {

margin-top: 20px;

}

.file-item {

display: flex;

align-items: center;

margin-bottom: 10px;

}

.file-item img, .file-item video {

max-width: 100px;

margin-right: 10px;

}

.file-item button {

margin-left: auto;

}

</style>

</head>

<body>

<h1>File Uploader with Preview</h1>

<input type="file" id="fileInput" multiple>

<div class="file-preview" id="filePreview"></div>

<script>

document.getElementById('fileUploader').addEventListener('change', function(event) { // Mistake: ID should be 'fileInput'

const files = event.target.files;

const previewContainer = document.getElementById('filePreview');

// Clear previous previews

previewContainer.innerHTML = '';

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

const file = files[i];

const fileItem = document.createElement('div');

fileItem.classList.add('file-item');

const fileName = document.createElement('p');

fileName.textContent = file.name;

fileItem.appendChild(fileName);

const fileSize = document.createElement('p');

fileSize.textContent = (file.size / 1000).toFixed(2) + ' KB'; // Mistake: Should be 1024 to convert bytes to KB

fileItem.appendChild(fileSize);

if (file.type.startsWith('image/')) {

const img = document.createElement('img');

img.src = URL.createObjectURL(file);

fileItem.appendChild(img);

} else if (file.type.startsWith('video/')) {

const video = document.createElement('video');

video.src = URL.createObjectURL(file);

video.controls = false; // Mistake: Should be true to enable video controls

fileItem.appendChild(video);

}

const removeButton = document.createElement('button');

removeButton.textContent = 'Remove';

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

fileItem.remove(); // Mistake: Should be previewContainer.removeChild(fileItem);

});

fileItem.appendChild(removeButton);

previewContainer.appendChild(fileItem);

}

});

</script>

</body>

</html>

```

### Mistakes Summary:

1. Changed `fileInput` to `fileUploader` in the JavaScript code.

2. Changed `(file.size / 1024).toFixed(2) + ' KB'` to `(file.size / 1000).toFixed(2) + ' KB'`.

3. Changed `video.controls = true;` to `video.controls = false;`.

4. Changed `previewContainer.removeChild(fileItem);` to `fileItem.remove();`.

This code can be used to test the interviewee's ability to debug and correct the mistakes.

IPrompt7 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 17. \*\*Weather Comparison App\*\*

\*\*Description:\*\* Build an app that allows users to compare weather data for multiple cities.

\*\*Skills Tested:\*\* API interaction, data visualization, DOM manipulation.

\*\*Features:\*\*

- Fetch weather data from a weather API for multiple cities.

- Display current weather and forecast for each city.

- Visualize temperature, humidity, and other metrics in a comparative chart.

"""

Certainly! Below is a basic version of the Weather Comparison App with some intentional mistakes for testing purposes. The application uses the OpenWeatherMap API to fetch weather data for multiple cities and displays the weather information in a simple HTML structure. The mistakes are introduced to test the interviewee's understanding of API interaction, data visualization, and DOM manipulation.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Weather Comparison App</title>

<style>

.city-weather {

border: 1px solid #000;

margin: 10px;

padding: 10px;

}

</style>

</head>

<body>

<h1>Weather Comparison App</h1>

<div>

<input type="text" id="city1" placeholder="Enter city 1">

<input type="text" id="city2" placeholder="Enter city 2">

<button onclick="fetchWeather()">Compare Weather</button>

</div>

<div id="weather-data"></div>

<script>

const apiKey = 'your\_api\_key'; // Mistake: API key is not provided.

async function fetchWeather() {

const city1 = document.getElementById('city1').value;

const city2 = document.getElementById('city2').value;

const url1 = `http://api.openweathermap.org/data/2.5/weather?q=${city1}&appid=${apiKey}`;

const url2 = `http://api.openweathermap.org/data/2.5/weather?q=${city2}&appid=${apiKey}`;

try {

const response1 = await fetch(url1);

const response2 = await fetch(url2);

if (!response1.ok || !response2.ok) {

throw new Error('Failed to fetch weather data'); // Mistake: Generic error message.

}

const data1 = await response1.json();

const data2 = await response2.json();

displayWeather(data1, data2);

} catch (error) {

console.error(error);

alert('Error fetching weather data'); // Mistake: Poor error handling.

}

}

function displayWeather(data1, data2) {

const weatherDataDiv = document.getElementById('weather-data');

weatherDataDiv.innerHTML = `

<div class="city-weather">

<h2>${data1.name}</h2>

<p>Temperature: ${(data1.main.temp - 273.15).toFixed(2)}�C</p> <!-- Mistake: Temperature conversion issue. -->

<p>Humidity: ${data1.main.humidity}%</p>

</div>

<div class="city-weather">

<h2>${data2.name}</h2>

<p>Temperature: ${(data2.main.temp - 273.15).toFixed(2)}�C</p> <!-- Mistake: Temperature conversion issue. -->

<p>Humidity: ${data2.main.humidity}%</p>

</div>

`;

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*API Key Missing:\*\* The `apiKey` variable is set to a placeholder string `'your\_api\_key'`.

2. \*\*Generic Error Message:\*\* The error message in the `catch` block is not specific, making it hard to debug.

3. \*\*Poor Error Handling:\*\* The `alert` statement for error handling is not user-friendly.

4. \*\*Temperature Conversion Issue:\*\* The temperature conversion from Kelvin to Celsius is not explained, and it may confuse the interviewee.

5. \*\*HTTP Instead of HTTPS:\*\* The API request URL uses `http` instead of `https`, which is not secure.

These mistakes will test the interviewee's ability to:

- Correctly handle API keys and secure them.

- Implement specific and user-friendly error handling.

- Understand temperature conversion and explain it.

- Use secure URLs for API requests.

The interviewee should be able to identify and correct these issues to make the app functional and user-friendly.

IPrompt8 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

. \*\*Collaborative Drawing Board\*\*

\*\*Description:\*\* Create a collaborative drawing board where multiple users can draw on the same canvas in real-time.

\*\*Skills Tested:\*\* WebSockets, Canvas API, event handling.

\*\*Features:\*\*

- Allow multiple users to connect and draw simultaneously.

- Sync drawing actions in real-time using WebSockets.

- Include tools for drawing lines, shapes, and text.

"""

Certainly! Below is a simplified version of a collaborative drawing board application using HTML, CSS, and JavaScript. This code includes some intentional mistakes to test the interviewee's understanding of WebSockets, Canvas API, and event handling.

### index.html

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Collaborative Drawing Board</title>

<style>

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f0f0f0;

}

canvas {

border: 1px solid black;

}

.toolbar {

position: absolute;

top: 10px;

left: 10px;

}

</style>

</head>

<body>

<div class="toolbar">

<button id="line">Line</button>

<button id="rectangle">Rectangle</button>

<button id="text">Text</button>

</div>

<canvas id="drawingBoard" width="800" height="600"></canvas>

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

</body>

</html>

```

### app.js

```javascript

const canvas = document.getElementById('drawingBoard');

const ctx = canvas.getContext('2d');

const socket = new WebSocket('ws://localhost:8080'); // Intentional mistake: WebSocket server URL might be incorrect

let drawing = false;

let tool = 'line';

let startX, startY;

document.getElementById('line').addEventListener('click', () => tool = 'line');

document.getElementById('rectangle').addEventListener('click', () => tool = 'rectangle');

document.getElementById('text').addEventListener('click', () => tool = 'text');

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

drawing = true;

[startX, startY] = [e.offsetX, e.offsetY];

});

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

if (!drawing) return;

if (tool === 'line') {

drawLine(startX, startY, e.offsetX, e.offsetY);

sendDrawData('line', startX, startY, e.offsetX, e.offsetY);

[startX, startY] = [e.offsetX, e.offsetY];

}

});

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

drawing = false;

if (tool === 'rectangle') {

drawRectangle(startX, startY, e.offsetX - startX, e.offsetY - startY);

sendDrawData('rectangle', startX, startY, e.offsetX - startX, e.offsetY - startY);

}

});

socket.addEventListener('message', (event) => {

const data = JSON.parse(event.data);

switch (data.tool) {

case 'line':

drawLine(data.startX, data.startY, data.endX, data.endY);

break;

case 'rectangle':

drawRectangle(data.startX, data.startY, data.width, data.height);

break;

case 'text':

drawText(data.text, data.x, data.y);

break;

}

});

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

ctx.beginPath();

ctx.moveTo(x1, y1);

ctx.lineTo(x2, y2);

ctx.stroke();

}

function drawRectangle(x, y, width, height) {

ctx.strokeRect(x, y, width, height);

}

function drawText(text, x, y) {

ctx.fillText(text, x, y); // Intentional mistake: Missing font settings

}

function sendDrawData(tool, startX, startY, endX, endY, text) {

const data = { tool, startX, startY, endX, endY, text };

socket.send(JSON.stringify(data)); // Intentional mistake: Missing error handling

}

```

### Intentional Mistakes to Identify and Fix

1. \*\*WebSocket URL\*\*: The WebSocket URL might be incorrect or not configured properly.

2. \*\*Missing Font Settings\*\*: The `drawText` function does not set any font, which might be required.

3. \*\*Error Handling\*\*: The `sendDrawData` function does not handle any WebSocket errors.

4. \*\*Event Handling\*\*: The `mousemove` event handler does not handle the case where the tool is not 'line'.

5. \*\*WebSocket Connection\*\*: There is no error handling for the WebSocket connection itself.

The interviewee should identify and fix these issues to make the application fully functional.

IPrompt9 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 19. \*\*Music Playlist Manager\*\*

\*\*Description:\*\* Develop a music playlist manager where users can create, edit, and play playlists.

\*\*Skills Tested:\*\* Audio API, DOM manipulation, data handling.

\*\*Features:\*\*

- Add songs to playlists from a local library or an API.

- Play, pause, and skip songs in the playlist.

- Display song details and playlist duration.

- Persist playlist data in local storage.

"""

Certainly! Below is a testable code snippet that introduces a few intentional mistakes to challenge the interviewee's understanding of the main concepts such as DOM manipulation, data handling, and working with local storage. This code can be used on coding platforms like HackerRank or Coderbyte.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Music Playlist Manager</title>

</head>

<body>

<h1>Music Playlist Manager</h1>

<div>

<input type="text" id="songInput" placeholder="Add a song">

<button onclick="addSong()">Add Song</button>

</div>

<div>

<h2>Playlist</h2>

<ul id="playlist"></ul>

</div>

<div>

<button onclick="playSong()">Play</button>

<button onclick="pauseSong()">Pause</button>

<button onclick="skipSong()">Skip</button>

</div>

<div>

<h3>Now Playing:</h3>

<p id="nowPlaying">None</p>

<h3>Playlist Duration:</h3>

<p id="playlistDuration">0:00</p>

</div>

<script>

let playlist = [];

let currentSongIndex = 0;

let isPlaying = false;

let audio = new Audio();

function addSong() {

let songInput = document.getElementById('songInput').value;

if (songInput) {

playlist.push(songInput);

updatePlaylistUI();

document.getElementById('songInput').value = ''; // Clear input

savePlaylist();

}

}

function updatePlaylistUI() {

let playlistElement = document.getElementById('playlist');

playlistElement.innerHTML = '';

playlist.forEach((song, index) => {

let li = document.createElement('li');

li.textContent = song;

playlistElement.appendChild(li);

});

updatePlaylistDuration();

}

function playSong() {

if (playlist.length > 0 && !isPlaying) {

audio.src = playlist[currentSongIndex];

audio.play();

isPlaying = true;

document.getElementById('nowPlaying').textContent = playlist[currentSongIndex];

}

}

function pauseSong() {

if (isPlaying) {

audio.pause();

isPlaying = false;

}

}

function skipSong() {

if (playlist.length > 0) {

currentSongIndex = (currentSongIndex + 1) % playlist.length;

playSong();

}

}

function updatePlaylistDuration() {

let totalDuration = playlist.length \* 3; // Assume each song is 3 minutes

let minutes = Math.floor(totalDuration / 60);

let seconds = totalDuration % 60;

document.getElementById('playlistDuration').textContent = `${minutes}:${seconds < 10 ? '0' : ''}${seconds}`;

}

function savePlaylist() {

localStorage.setItem('playlist', JSON.stringify(playlist));

}

function loadPlaylist() {

let savedPlaylist = localStorage.getItem('playlist');

if (savedPlaylist) {

playlist = JSON.parse(savedPlaylist);

updatePlaylistUI();

}

}

// Load playlist on page load

window.onload = loadPlaylist;

</script>

</body>

</html>

```

### Intentional Mistakes and Points to Test:

1. \*\*Audio Source Handling\*\*: The `audio.src` is set directly to the song name, which is not a valid URL. The interviewee should recognize that they need to handle actual audio file URLs or use a mock API.

2. \*\*Playlist Duration Calculation\*\*: The duration is hardcoded to 3 minutes per song. The interviewee should ideally recognize the need to handle actual song duration data.

3. \*\*Local Storage Loading Timing\*\*: The `loadPlaylist` function is called on `window.onload`, which might not be the best timing for all scenarios. The interviewee should consider if this is the best approach or if there are better event listeners to use.

4. \*\*UI Update after Skipping Songs\*\*: The `nowPlaying` element is not updated when skipping songs. The interviewee should notice that the `playSong` function needs to update the `nowPlaying` element after skipping.

5. \*\*Error Handling\*\*: The code lacks error handling for cases where the audio file might fail to load or play. The interviewee should add error handling for robustness.

This code provides a good starting point and includes enough complexity to test an interviewee's understanding and ability to debug and improve the application.

IPrompt10 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 20. \*\*Task Management with Kanban Board\*\*

\*\*Description:\*\* Implement a task management application with a Kanban board interface.

\*\*Skills Tested:\*\* DOM manipulation, drag-and-drop, data management.

\*\*Features:\*\*

- Create tasks and assign them to different columns (e.g., To Do, In Progress, Done).

- Drag and drop tasks between columns.

- Edit and delete tasks.

- Store tasks in local storage for persistence.

"""

Sure, here's a testable HTML, CSS, and JavaScript code for a basic Kanban board application with some intentional mistakes introduced. The mistakes will help test the interviewee's understanding of the concepts.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Kanban Board</title>

<style>

body {

font-family: Arial, sans-serif;

}

.board {

display: flex;

justify-content: space-around;

margin-top: 20px;

}

.column {

width: 30%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f9f9f9;

}

.task {

padding: 10px;

margin: 10px 0;

background-color: #fff;

border: 1px solid #ddd;

cursor: pointer;

}

</style>

</head>

<body>

<h1>Kanban Board</h1>

<div>

<input type="text" id="taskInput" placeholder="New task">

<button onclick="addTask()">Add Task</button>

</div>

<div class="board">

<div class="column" id="todo">

<h2>To Do</h2>

</div>

<div class="column" id="in-progress">

<h2>In Progress</h2>

</div>

<div class="column" id="done">

<h2>Done</h2>

</div>

</div>

<script>

// Function to add a new task

function addTask() {

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

const taskText = taskInput.value;

if (taskText === '') return;

const task = document.createElement('div');

task.className = 'task';

task.textContent = taskText;

task.draggable = true;

task.ondragstart = onDragStart;

task.ondblclick = editTask;

document.getElementById('todo').appendChild(task);

taskInput.value = '';

saveTasks();

}

// Function to handle drag start event

function onDragStart(event) {

event.dataTransfer.setData('text/plain', event.target.id);

}

// Function to handle drop event

function onDrop(event) {

event.preventDefault();

const id = event.dataTransfer.getData('text/plain');

const draggableElement = document.getElementById(id);

const dropzone = event.target;

dropzone.appendChild(draggableElement);

saveTasks();

}

// Function to handle drag over event

function onDragOver(event) {

event.preventDefault();

}

// Function to edit a task

function editTask(event) {

const newText = prompt('Edit task:', event.target.textContent);

if (newText !== null) {

event.target.textContent = newText;

saveTasks();

}

}

// Function to save tasks to local storage

function saveTasks() {

const columns = document.querySelectorAll('.column');

const tasks = {};

columns.forEach(column => {

tasks[column.id] = [];

column.querySelectorAll('.task').forEach(task => {

tasks[column.id].push(task.textContent);

});

});

localStorage.setItem('tasks', JSON.stringify(tasks));

}

// Function to load tasks from local storage

function loadTasks() {

const tasks = JSON.parse(localStorage.getItem('tasks'));

if (!tasks) return;

Object.keys(tasks).forEach(columnId => {

const column = document.getElementById(columnId);

tasks[columnId].forEach(taskText => {

const task = document.createElement('div');

task.className = 'task';

task.textContent = taskText;

task.draggable = true;

task.ondragstart = onDragStart;

task.ondblclick = editTask;

column.appendChild(task);

});

});

}

// Event listeners for drag and drop

document.querySelectorAll('.column').forEach(column => {

column.ondrop = onDrop;

column.ondragover = onDragOver;

});

// Load tasks when the page loads

window.onload = loadTasks;

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Drag and Drop IDs\*\*: The tasks are not assigned unique IDs, which makes `event.dataTransfer.setData('text/plain', event.target.id);` ineffective since `event.target.id` is `undefined`.

2. \*\*Drop Event Handler\*\*: The `onDrop` event handler tries to append the dragged element directly to the drop zone, which may not work if the target is not a column.

3. \*\*Task Editing\*\*: The `editTask` function modifies the task text directly, but there is no mechanism to handle the case where the user cancels the prompt.

4. \*\*Persistence\*\*: The local storage does not handle the case where `tasks` is `null` or `undefined`.

5. \*\*CSS Styling\*\*: The visual styling of the tasks and columns is very basic, which can be improved for better user experience.

These mistakes will help test the interviewee's understanding of DOM manipulation, drag-and-drop, and data management concepts.

--------------------------------------------------------------------------------------------------------------

Certainly! Below is a JavaScript code snippet for an interactive story game that you can use on platforms like HackerRank or Coderbyte. This code includes intentional mistakes to test the interviewee's understanding of data management, event handling, and DOM manipulation.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Story Game</title>

</head>

<body>

<div id="story-container">

<p id="story-text"></p>

<div id="choices-container"></div>

</div>

<button id="restart-button">Restart</button>

<button id="save-button">Save Checkpoint</button>

<button id="load-button">Load Checkpoint</button>

<script>

const storyContainer = document.getElementById('story-container');

const storyText = document.getElementById('story-text');

const choicesContainer = document.getElementById('choices-container');

const restartButton = document.getElementById('restart-button');

const saveButton = document.getElementById('save-button');

const loadButton = document.getElementById('load-button');

let currentStoryNode = 'start';

let savedCheckpoint = null;

const storyNodes = {

start: {

text: "You wake up in a dark forest. What do you do?",

choices: [

{ text: "Look around", next: "look\_around" },

{ text: "Go back to sleep", next: "sleep" }

]

},

look\_around: {

text: "You see a path leading to a village. What do you do?",

choices: [

{ text: "Follow the path", next: "village" },

{ text: "Stay in the forest", next: "forest" }

]

},

sleep: {

text: "You fall asleep and never wake up. The end.",

choices: []

},

village: {

text: "You arrive at the village and find help. The end.",

choices: []

},

forest: {

text: "You get lost in the forest and can't find your way out. The end.",

choices: []

}

};

function displayStoryNode(nodeId) {

const node = storyNodes[nodeId];

storyText.innerHTML = node.text;

choicesContainer.innerHTML = '';

node.choices.forEach(choice => {

const button = document.createElement('button');

button.innerText = choice.text;

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

currentStoryNode = choice.next;

displayStoryNode(currentStoryNode);

});

choicesContainer.appendChild(button);

});

}

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

currentStoryNode = 'start';

displayStoryNode(currentStoryNode);

});

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

savedCheckpoint = currentStoryNode;

alert('Checkpoint saved!');

});

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

if (savedCheckpoint) {

currentStoryNode = savedCheckpoint;

displayStoryNode(currentStoryNode);

} else {

alert('No checkpoint saved.');

}

});

// Start the game initially

displayStoryNode(currentStoryNode);

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Data Management\*\*: The `savedCheckpoint` variable is declared but never utilized correctly to load the saved checkpoint.

2. \*\*Event Handling\*\*: The event listener for the buttons might not be correctly handling edge cases, such as checking if a saved checkpoint exists.

3. \*\*DOM Manipulation\*\*: The `displayStoryNode` function might not be handling the case where no choices are available, which can lead to an empty `choicesContainer`.

These mistakes are designed to test the interviewee's ability to debug and fix issues related to data management, event handling, and DOM manipulation.

IPrompt11 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 21. \*\*Interactive Story Game\*\*

\*\*Description:\*\* Create an interactive story game where users make choices that influence the story's outcome.

\*\*Skills Tested:\*\* Data management, event handling, DOM manipulation.

\*\*Features:\*\*

- Present a story with branching paths based on user choices.

- Track user choices and progress through the story.

- Allow users to restart the story from the beginning or a saved checkpoint.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Story Game</title>

</head>

<body>

<div id="story-container">

<p id="story-text"></p>

<div id="choices-container"></div>

</div>

<button id="restart-button">Restart</button>

<button id="save-button">Save Checkpoint</button>

<button id="load-button">Load Checkpoint</button>

<script>

const storyContainer = document.getElementById('story-container');

const storyText = document.getElementById('story-text');

const choicesContainer = document.getElementById('choices-container');

const restartButton = document.getElementById('restart-button');

const saveButton = document.getElementById('save-button');

const loadButton = document.getElementById('load-button');

let currentStoryNode = 'start';

let savedCheckpoint = null;

const storyNodes = {

start: {

text: "You wake up in a dark forest. What do you do?",

choices: [

{ text: "Look around", next: "look\_around" },

{ text: "Go back to sleep", next: "sleep" }

]

},

look\_around: {

text: "You see a path leading to a village. What do you do?",

choices: [

{ text: "Follow the path", next: "village" },

{ text: "Stay in the forest", next: "forest" }

]

},

sleep: {

text: "You fall asleep and never wake up. The end.",

choices: []

},

village: {

text: "You arrive at the village and find help. The end.",

choices: []

},

forest: {

text: "You get lost in the forest and can't find your way out. The end.",

choices: []

}

};

function displayStoryNode(nodeId) {

const node = storyNodes[nodeId];

storyText.innerHTML = node.text;

choicesContainer.innerHTML = '';

node.choices.forEach(choice => {

const button = document.createElement('button');

button.innerText = choice.text;

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

currentStoryNode = choice.next;

displayStoryNode(currentStoryNode);

});

choicesContainer.appendChild(button);

});

}

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

currentStoryNode = 'start';

displayStoryNode(currentStoryNode);

});

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

savedCheckpoint = currentStoryNode;

alert('Checkpoint saved!');

});

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

if (savedCheckpoint) {

currentStoryNode = savedCheckpoint;

displayStoryNode(currentStoryNode);

} else {

alert('No checkpoint saved.');

}

});

// Start the game initially

displayStoryNode(currentStoryNode);

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Data Management\*\*: The `savedCheckpoint` variable is declared but never utilized correctly to load the saved checkpoint.

2. \*\*Event Handling\*\*: The event listener for the buttons might not be correctly handling edge cases, such as checking if a saved checkpoint exists.

3. \*\*DOM Manipulation\*\*: The `displayStoryNode` function might not be handling the case where no choices are available, which can lead to an empty `choicesContainer`.

These mistakes are designed to test the interviewee's ability to debug and fix issues related to data management, event handling, and DOM manipulation.

IPrompt12 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 22. \*\*Stock Market Dashboard\*\*

\*\*Description:\*\* Build a stock market dashboard that displays real-time stock prices and historical data.

\*\*Skills Tested:\*\* API interaction, data visualization, DOM manipulation.

\*\*Features:\*\*

- Fetch real-time stock prices from a financial API.

- Display price charts with historical data.

- Allow users to search for and track specific stocks.

- Visualize portfolio performance over time.

"""

```html

<!DOCTYPE html>

<html>

<head>

<title>Stock Market Dashboard</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h1>Stock Market Dashboard</h1>

<input type="text" id="stockSymbol" placeholder="Enter Stock Symbol">

<button onclick="fetchStockData()">Get Stock Data</button>

<canvas id="priceChart" width="400" height="200"></canvas>

<script>

async function fetchStockData() {

const symbol = document.getElementById('stockSymbol').value;

const apiKey = 'YOUR\_API\_KEY';

const apiUrl = `https://api.financialmodelingprep.com/api/v3/quote/${symbol}?apikey=${apiKey}`;

try {

const response = await fetch(apiUrl);

const data = await response.json();

if (data.length === 0) {

alert('No data found for the given stock symbol');

return;

}

const priceData = data[0].price;

displayChart(priceData);

} catch (error) {

console.error('Error fetching stock data:', error);

}

}

function displayChart(priceData) {

const ctx = document.getElementById('priceChart').getContext('2d');

const chart = new Chart(ctx, {

type: 'line',

data: {

labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],

datasets: [{

label: 'Stock Price',

data: priceData, // Intentional mistake: This should be an array of historical prices, not a single price

borderColor: 'rgba(75, 192, 192, 1)',

borderWidth: 1

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*API Key Placeholder\*\*: The `apiKey` variable is set to `'YOUR\_API\_KEY'`. This is a placeholder and will not work until replaced with a valid API key.

2. \*\*Incorrect `priceData` Usage\*\*: In the `displayChart` function, `priceData` is expected to be an array of historical prices but is actually a single price value.

3. \*\*Real-Time Data Misconception\*\*: The script fetches only the current price, whereas real-time data should ideally involve continuous updates.

4. \*\*Historical Data Missing\*\*: The script does not fetch historical data; it only fetches the current price.

### Instructions for Interviewee:

1. Replace the placeholder API key with a valid key from a financial data provider.

2. Modify the code to fetch historical stock prices instead of just the current price.

3. Ensure that the `displayChart` function correctly uses an array of historical prices.

4. Optionally, implement a mechanism to periodically update the stock prices to simulate real-time data.

--------------------------------------------------------------------------------------------------------------

Below is a sample code for a language learning app with some intentional mistakes. The code demonstrates data management, DOM manipulation, and API interaction. The interviewee will need to identify and fix the mistakes to make the application fully functional.

IPrompt13 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 23. \*\*Language Learning App\*\*

\*\*Description:\*\* Develop an app to help users learn a new language by practicing vocabulary and phrases.

\*\*Skills Tested:\*\* Data management, DOM manipulation, API interaction.

\*\*Features:\*\*

- Provide vocabulary and phrases with translations.

- Include quizzes and flashcards for practice.

- Track user progress and review performance.

- Integrate with a language API for additional resources.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Language Learning App</title>

<style>

body {

font-family: Arial, sans-serif;

}

.flashcard {

border: 1px solid #ccc;

padding: 20px;

margin: 10px;

text-align: center;

}

.quiz {

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Language Learning App</h1>

<div id="flashcards"></div>

<div class="quiz">

<h2>Quiz</h2>

<div id="quiz-content"></div>

</div>

<div>

<h2>Progress</h2>

<div id="progress"></div>

</div>

<script>

const vocabulary = [

{ word: 'House', translation: 'Casa' },

{ word: 'Dog', translation: 'Perro' },

{ word: 'Cat', translation: 'Gato' }

];

const userProgress = {

correctAnswers: 0,

totalQuestions: 0

};

function displayFlashcards() {

const flashcardsContainer = document.getElementById('flashcards');

vocabulary.forEach(item => {

const card = document.createElement('div');

card.className = 'flashcard';

card.innerHTML = `<p>${item.word}</p><p>${item.translation}</p>`;

flashcardsContainer.appendChild(card);

});

}

function displayQuiz() {

const quizContainer = document.getElementById('quiz-content');

const questionIndex = Math.floor(Math.random() \* vocabulary.length);

const question = vocabulary[questionIndex];

quizContainer.innerHTML = `

<p>Translate: ${question.word}</p>

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

<button onclick="checkAnswer('${question.translation}')">Submit</button>

`;

}

function checkAnswer(correctAnswer) {

const userAnswer = document.getElementById('answer').value;

if (userAnswer.toLowerCase() === correctAnswer.toLowerCase()) {

alert('Correct!');

userProgress.correctAnswers += 1;

} else {

alert('Incorrect!');

}

userProgress.totalQuestions += 1;

updateProgress();

}

function updateProgress() {

const progressContainer = document.getElementById('progress');

progressContainer.innerHTML = `

<p>Correct Answers: ${userProgress.correctAnswers}</p>

<p>Total Questions: ${userProgress.totalQuestions}</p>

`;

}

async function fetchAdditionalResources() {

try {

const response = await fetch('https://api.example.com/language-resources');

const data = await response.json();

console.log(data);

// Process the additional language resources here

} catch (error) {

console.error('Error fetching additional resources:', error);

}

}

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

displayFlashcards();

displayQuiz();

updateProgress();

fetchAdditionalResources();

});

</script>

</body>

</html>

```

### Intentional Mistakes:

1. The `vocabulary` array does not include sufficient data to cover various cases.

2. The `checkAnswer` function does not clear the input field after checking the answer.

3. The `displayQuiz` function does not generate a new question after answering the current one.

4. The `fetchAdditionalResources` function has a placeholder URL (`https://api.example.com/language-resources`) that needs to be replaced with a real API URL.

5. The `fetchAdditionalResources` function does not utilize the fetched data.

These mistakes will test the interviewee's ability to debug, understand API interactions, and ensure proper functionality.

--------------------------------------------------------------------------------------------------------------

IPrompt14 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 24. \*\*Interactive Data Table\*\*

\*\*Description:\*\* Create an interactive data table that supports sorting, filtering, and pagination.

\*\*Skills Tested:\*\* DOM manipulation, data handling, event handling.

\*\*Features:\*\*

- Load data from a JSON file or an API.

- Allow users to sort columns by clicking on headers.

- Implement filters to search and narrow down results.

- Add pagination controls to navigate through large datasets.

"""

Certainly! Below is a testable code snippet for an interactive data table with some intentional mistakes. This code is designed to be used on coding platforms like HackerRank or Coderbyte. The code includes features like sorting, filtering, and pagination, but with some errors that should be identified and fixed by the interviewee.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Data Table</title>

<style>

table {

width: 100%;

border-collapse: collapse;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

}

th {

cursor: pointer;

}

.pagination {

display: flex;

justify-content: center;

margin-top: 10px;

}

.pagination button {

margin: 0 5px;

padding: 5px 10px;

}

</style>

</head>

<body>

<input type="text" id="search" placeholder="Search...">

<table id="dataTable">

<thead>

<tr>

<th data-column="name">Name</th>

<th data-column="age">Age</th>

<th data-column="city">City</th>

</tr>

</thead>

<tbody>

<!-- Data will be populated here -->

</tbody>

</table>

<div class="pagination">

<button id="prevPage">Previous</button>

<span id="pageInfo">Page 1</span>

<button id="nextPage">Next</button>

</div>

<script>

const dataUrl = 'data.json'; // Assume this JSON file contains the data

const rowsPerPage = 5;

let currentPage = 1;

let data = [];

let filteredData = [];

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

data = await fetchData();

filteredData = data;

renderTable();

});

document.getElementById('search').addEventListener('keyup', (event) => {

const searchTerm = event.target.value.toLowerCase();

filteredData = data.filter(item =>

item.name.toLowerCase().includes(searchTerm) ||

item.age.toString().includes(searchTerm) ||

item.city.toLowerCase().includes(searchTerm)

);

currentPage = 1;

renderTable();

});

document.querySelectorAll('th').forEach(header => {

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

const column = header.getAttribute('data-column');

filteredData.sort((a, b) => a[column] > b[column] ? 1 : -1);

renderTable();

});

});

document.getElementById('prevPage').addEventListener('click', () => {

if (currentPage > 1) {

currentPage--;

renderTable();

}

});

document.getElementById('nextPage').addEventListener('click', () => {

if (currentPage < Math.ceil(filteredData.length / rowsPerPage)) {

currentPage++;

renderTable();

}

});

async function fetchData() {

try {

const response = await fetch(dataUrl);

return await response.json();

} catch (error) {

console.error('Error fetching data:', error);

return [];

}

}

function renderTable() {

const tbody = document.querySelector('#dataTable tbody');

tbody.innerHTML = '';

const start = (currentPage - 1) \* rowsPerPage;

const end = start + rowsPerPage;

const paginatedData = filteredData.slice(start, end);

paginatedData.forEach(item => {

const row = document.createElement('tr');

row.innerHTML = `<td>${item.name}</td><td>${item.age}</td><td>${item.city}</td>`;

tbody.appendChild(row);

});

document.getElementById('pageInfo').textContent = `Page ${currentPage}`;

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Sorting Issue\*\*: The sorting function only sorts in ascending order and does not toggle between ascending and descending.

2. \*\*Pagination Info\*\*: The page info does not update correctly when filtering the data.

3. \*\*Event Delegation\*\*: The event listeners for sorting headers are attached before the data is loaded, which might not work correctly if headers are dynamically generated.

4. \*\*Error Handling\*\*: The `fetchData` function does not handle fetch errors gracefully.

These mistakes are intended to test the interviewee's understanding of DOM manipulation, data handling, and event handling. They should identify and correct these issues to ensure the application works as expected.

IPrompt15 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 25. \*\*Virtual Pet Game\*\*

\*\*Description:\*\* Develop a virtual pet game where users can feed, play with, and take care of a virtual pet.

\*\*Skills Tested:\*\* DOM manipulation, event handling, state management.

\*\*Features:\*\*

- Display the pet's health, hunger, and happiness levels.

- Provide actions to feed, play with, and clean the pet.

- Implement a simple AI to change the pet's state over time.

"""

To create a virtual pet game with intentional mistakes, we will focus on HTML, CSS, and JavaScript. The goal is to test the interviewee's understanding of DOM manipulation, event handling, and state management. Here is a basic implementation with some intentional errors:

### HTML

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Virtual Pet Game</title>

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

</head>

<body>

<h1>Virtual Pet Game</h1>

<div id="pet">

<p>Health: <span id="health">100</span></p>

<p>Hunger: <span id="hunger">0</span></p>

<p>Happiness: <span id="happiness">100</span></p>

</div>

<div id="actions">

<button id="feed">Feed</button>

<button id="play">Play</button>

<button id="clean">Clean</button>

</div>

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

</body>

</html>

```

### CSS

```css

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f0f0f0;

}

#pet {

margin: 20px;

padding: 20px;

border: 1px solid #ccc;

background-color: #fff;

}

#actions {

margin: 20px;

}

button {

margin: 5px;

padding: 10px;

cursor: pointer;

}

```

### JavaScript

```javascript

document.addEventListener('DOMContentLoaded', function() {

let health = 100;

let hunger = 0;

let happiness = 100;

const healthSpan = document.getElementById('health');

const hungerSpan = document.getElementById('hunger');

const happinessSpan = document.getElementById('happiness');

document.getElementById('feed').addEventListener('click', function() {

hunger -= 10;

if (hunger < 0) hunger = 0;

updateStats();

});

document.getElementById('play').addEventListener('click', function() {

happiness += 10;

if (happiness > 100) happiness = 100;

updateStats();

});

document.getElementById('clean').addEventListener('click', function() {

health += 10;

if (health > 100) health = 100;

updateStats();

});

function updateStats() {

healthSpan.innerText = health;

hungerSpan.innerText = hunger;

happinessSpan.innerText = happiness;

}

function petAI() {

health -= 5;

hunger += 5;

happiness -= 5;

if (health < 0) health = 0;

if (hunger > 100) hunger = 100;

if (happiness < 0) happiness = 0;

updateStats();

// Call petAI every 5 seconds

setTimeout(petAI, 5000);

}

petAI();

});

```

### Intentional Mistakes

1. The `clean` button should likely reduce hunger instead of increasing health.

2. The `petAI` function should be called with `setInterval` instead of `setTimeout` within itself to avoid multiple timeouts being created over time.

3. The initial state update is missing after the DOM content is loaded.

4. The `clean` button's functionality is not clearly defined in the requirements, so it might be confusing. It should perhaps affect cleanliness if there was such a property.

These intentional mistakes will help test the interviewee's ability to spot and fix issues related to DOM manipulation, event handling, and state management.

IPrompt16 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 26. \*\*Customizable Calendar App\*\*

\*\*Description:\*\* Create a customizable calendar where users can add, edit, and delete events.

\*\*Skills Tested:\*\* DOM manipulation, date handling, event handling.

\*\*Features:\*\*

- Display a monthly view with days and dates.

- Allow users to add events with titles, descriptions, and times.

- Edit and delete events.

- Highlight days with events.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Customizable Calendar App</title>

<style>

.calendar {

display: grid;

grid-template-columns: repeat(7, 1fr);

}

.day {

border: 1px solid #ddd;

padding: 10px;

}

.event {

background-color: yellow;

margin-top: 5px;

}

</style>

</head>

<body>

<h1>Customizable Calendar App</h1>

<div class="calendar" id="calendar"></div>

<script>

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

function createCalendar(year, month) {

calendar.innerHTML = '';

const date = new Date(year, month, 1);

const daysInMonth = new Date(year, month + 1, 0).getDate();

for (let i = 1; i <= daysInMonth; i++) {

const day = document.createElement('div');

day.className = 'day';

day.innerHTML = i;

day.addEventListener('click', () => addEvent(day));

calendar.appendChild(day);

}

}

function addEvent(dayElement) {

const title = prompt('Enter event title:');

const description = prompt('Enter event description:');

const time = prompt('Enter event time:');

if (title && description && time) {

const event = document.createElement('div');

event.className = 'event';

event.innerHTML = `<strong>${title}</strong><br>${description}<br>${time}`;

event.addEventListener('click', (e) => {

e.stopPropagation();

editEvent(event);

});

dayElement.appendChild(event);

}

}

function editEvent(eventElement) {

const newTitle = prompt('Edit event title:', eventElement.querySelector('strong').innerText);

const newDescription = prompt('Edit event description:', eventElement.childNodes[2].nodeValue);

const newTime = prompt('Edit event time:', eventElement.childNodes[4].nodeValue);

if (newTitle && newDescription && newTime) {

eventElement.innerHTML = `<strong>${newTitle}</strong><br>${newDescription}<br>${newTime}`;

}

const deleteEvent = confirm('Do you want to delete this event?');

if (deleteEvent) {

eventElement.remove();

}

}

// Intentional mistakes:

// 1. Wrong month in createCalendar (month + 1).

// 2. Incorrect node references in editEvent (childNodes[2], childNodes[4]).

// 3. Missing event highlight functionality.

createCalendar(2023, 9); // October 2023

</script>

</body>

</html>

```

IPrompt17 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 28. \*\*Fitness Tracker\*\*

\*\*Description:\*\* Build a fitness tracker app to log workouts and monitor progress.

\*\*Skills Tested:\*\* Data handling, DOM manipulation, event handling.

\*\*Features:\*\*

- Log different types of workouts (e.g., running, cycling, weightlifting).

- Track duration, distance, and calories burned.

- Display progress over time with charts.

- Set and track fitness goals.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Fitness Tracker</title>

</head>

<body>

<h1>Fitness Tracker</h1>

<form id="workoutForm">

<label for="type">Workout Type:</label>

<select id="type" name="type">

<option value="running">Running</option>

<option value="cycling">Cycling</option>

<option value="weightlifting">Weightlifting</option>

</select>

<br>

<label for="duration">Duration (minutes):</label>

<input type="number" id="duration" name="duration">

<br>

<label for="distance">Distance (km):</label>

<input type="number" id="distance" name="distance">

<br>

<label for="calories">Calories Burned:</label>

<input type="number" id="calories" name="calories">

<br>

<button type="submit">Log Workout</button>

</form>

<div id="progress">

<h2>Progress</h2>

<ul id="workoutList"></ul>

</div>

<script>

document.getElementById('workoutForm').addEventListener('submit', function(e) {

e.preventDefault();

const type = document.getElementById('type').value;

const duration = parseInt(document.getElementById('duration').value);

const distance = parseFloat(document.getElementById('distance').value);

const calories = parseInt(document.getElementById('calories').value);

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

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

listItem.textContent = `Type: ${type}, Duration: ${duration} mins, Distance: ${distance} km, Calories: ${calories}`;

workoutList.appendChild(listItem);

document.getElementById('workoutForm').reset();

});

// Mistake: Incorrectly named function for chart display (should be displayChart)

function displayCharts() {

// TODO: Implement chart display logic

}

// Mistake: Missing declaration for goal variable

goal = 1000; // Calories goal

function checkGoal() {

let totalCalories = 0;

document.querySelectorAll('#workoutList li').forEach(item => {

const calories = parseInt(item.textContent.split('Calories: ')[1]);

totalCalories += calories;

});

if (totalCalories >= goal) {

alert('Congratulations! You have reached your goal!');

}

}

// Mistake: Incorrect event type (should be 'change')

document.getElementById('workoutList').addEventListener('click', checkGoal);

</script>

</body>

</html>

```

IPrompt18 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 29. \*\*Interactive Learning Platform\*\*

\*\*Description:\*\* Create an interactive learning platform with courses and quizzes.

\*\*Skills Tested:\*\* Data management, DOM manipulation, event handling.

\*\*Features:\*\*

- Display a list of courses and lessons.

- Include quizzes at the end of each lesson.

- Track user progress through the courses.

- Provide feedback and scores for quizzes.

"""

Below is a code snippet for an interactive learning platform application. The code is intentionally introduced with some mistakes to test the interviewee's understanding of data management, DOM manipulation, and event handling.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Learning Platform</title>

<style>

.course {

margin-bottom: 20px;

}

.lesson, .quiz {

margin-left: 20px;

display: none;

}

</style>

</head>

<body>

<h1>Interactive Learning Platform</h1>

<div id="courses-container"></div>

<script>

const courses = [

{

title: "Course 1",

lessons: [

{

title: "Lesson 1",

content: "Content for Lesson 1",

quiz: {

question: "What is 2 + 2?",

options: ["3", "4", "5"],

answer: "4"

}

},

{

title: "Lesson 2",

content: "Content for Lesson 2",

quiz: {

question: "What is 3 + 3?",

options: ["5", "6", "7"],

answer: "6"

}

}

]

},

{

title: "Course 2",

lessons: [

{

title: "Lesson 1",

content: "Content for Lesson 1",

quiz: {

question: "What is 1 + 1?",

options: ["1", "2", "3"],

answer: "2"

}

}

]

}

];

const coursesContainer = document.getElementById('courses-container');

courses.forEach((course, courseIndex) => {

const courseElement = document.createElement('div');

courseElement.className = 'course';

const courseTitle = document.createElement('h2');

courseTitle.textContent = course.title;

courseElement.appendChild(courseTitle);

course.lessons.forEach((lesson, lessonIndex) => {

const lessonElement = document.createElement('div');

lessonElement.className = 'lesson';

const lessonTitle = document.createElement('h3');

lessonTitle.textContent = lesson.title;

lessonElement.appendChild(lessonTitle);

const lessonContent = document.createElement('p');

lessonContent.textContent = lesson.content;

lessonElement.appendChild(lessonContent);

const quizElement = document.createElement('div');

quizElement.className = 'quiz';

const quizQuestion = document.createElement('p');

quizQuestion.textContent = lesson.quiz.question;

quizElement.appendChild(quizQuestion);

lesson.quiz.options.forEach(option => {

const optionElement = document.createElement('button');

optionElement.textContent = option;

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

if (option === lesson.quiz.answer) {

alert('Correct!');

} else {

alert('Incorrect!');

}

});

quizElement.appendChild(optionElement);

});

lessonElement.appendChild(quizElement);

courseElement.appendChild(lessonElement);

});

coursesContainer.appendChild(courseElement);

});

const lessons = document.querySelectorAll('.lesson');

lessons.forEach(lesson => {

lesson.style.display = 'block'; // This mistake makes all lessons visible immediately

});

const quizzes = document.querySelectorAll('.quiz');

quizzes.forEach(quiz => {

quiz.style.display = 'none'; // This mistake hides all quizzes initially

});

</script>

</body>

</html>

```

Mistakes introduced:

1. All lessons are displayed immediately because of the `lesson.style.display = 'block';` line.

2. All quizzes are hidden initially because of the `quiz.style.display = 'none';` line, and there's no logic to show the quizzes after the lessons are completed.

3. No mechanism to track user progress through the courses.

4. The feedback mechanism is very basic and only uses alert boxes.

These mistakes will help to test the interviewee's ability to identify and correct issues related to data management, DOM manipulation, and event handling.

IPrompt19 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 30. \*\*Real-time Polling App\*\*

\*\*Description:\*\* Develop a real-time polling app where users can create and participate in polls.

\*\*Skills Tested:\*\* WebSockets, event handling, data visualization.

\*\*Features:\*\*

- Create polls with multiple choice questions.

- Allow users to vote in real-time.

- Display live results with charts.

- Support multiple polls simultaneously.

"""

Certainly! Below is a sample code for a real-time polling app that introduces a few intentional mistakes. The code is designed to be testable on coding platforms like HackerRank or Coderbyte and incorporates WebSockets, event handling, and data visualization. The mistakes are included to test the interviewee's debugging skills and understanding of the main concepts.

### Server-Side (Node.js with WebSocket)

```javascript

const http = require('http');

const WebSocket = require('ws');

const server = http.createServer();

const wss = new WebSocket.Server({ server });

let polls = {};

wss.on('connection', function connection(ws) {

ws.on('message', function incoming(message) {

const data = JSON.parse(message);

switch (data.type) {

case 'createPoll':

const pollId = Math.random().toString(36).substring(2, 15); // Mistake: Potential collision

polls[pollId] = {

question: data.question,

options: data.options,

votes: Array(data.options.length).fill(0)

};

ws.send(JSON.stringify({ type: 'pollCreated', pollId }));

break;

case 'vote':

if (polls[data.pollId]) {

polls[data.pollId].votes[data.option]++;

broadcast(JSON.stringify({

type: 'updateResults',

pollId: data.pollId,

votes: polls[data.pollId].votes

}));

}

break;

default:

console.log('Unknown message type:', data.type);

}

});

});

function broadcast(message) {

wss.clients.forEach(function each(client) {

if (client.readyState === WebSocket.OPEN) {

client.send(message);

}

});

}

server.listen(8080, function listening() {

console.log('Server listening on port 8080');

});

```

### Client-Side (HTML + JavaScript)

```html

<!DOCTYPE html>

<html>

<head>

<title>Real-time Polling App</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h1>Real-time Polling App</h1>

<div id="polls"></div>

<script>

const ws = new WebSocket('ws://localhost:8080');

ws.onmessage = function (event) {

const data = JSON.parse(event.data);

switch (data.type) {

case 'pollCreated':

addPoll(data.pollId);

break;

case 'updateResults':

updateChart(data.pollId, data.votes);

break;

default:

console.log('Unknown message type:', data.type);

}

};

function createPoll(question, options) {

ws.send(JSON.stringify({

type: 'createPoll',

question: question,

options: options

}));

}

function vote(pollId, option) {

ws.send(JSON.stringify({

type: 'vote',

pollId: pollId,

option: option

}));

}

function addPoll(pollId) {

const pollContainer = document.getElementById('polls');

const pollDiv = document.createElement('div');

pollDiv.id = pollId;

pollDiv.innerHTML = `

<h2>${polls[pollId].question}</h2>

${polls[pollId].options.map((opt, index) => `

<button onclick="vote('${pollId}', ${index})">${opt}</button>

`).join('')}

<canvas id="chart-${pollId}"></canvas>

`;

pollContainer.appendChild(pollDiv);

createChart(pollId, polls[pollId].options);

}

function createChart(pollId, labels) {

const ctx = document.getElementById(`chart-${pollId}`).getContext('2d');

new Chart(ctx, {

type: 'bar',

data: {

labels: labels,

datasets: [{

label: 'Votes',

data: Array(labels.length).fill(0),

backgroundColor: 'rgba(54, 162, 235, 0.2)',

borderColor: 'rgba(54, 162, 235, 1)',

borderWidth: 1

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

function updateChart(pollId, votes) {

const chart = Chart.getChart(`chart-${pollId}`);

chart.data.datasets[0].data = votes; // Mistake: Chart update method might be wrong

chart.update();

}

// Example usage

createPoll('What is your favorite color?', ['Red', 'Green', 'Blue']);

</script>

</body>

</html>

```

### Intentional Mistakes

1. \*\*Server-Side:\*\*

- Generating poll IDs using `Math.random().toString(36).substring(2, 15)` which can potentially lead to collisions.

2. \*\*Client-Side:\*\*

- Referencing a non-existent `polls` object in `addPoll` function.

- Incorrectly updating the Chart.js chart in `updateChart` function.

These mistakes are intended to test the interviewee's ability to debug and understand the code.

IPrompt20 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 31. \*\*Movie Recommendation System\*\*

\*\*Description:\*\* Build a movie recommendation system based on user preferences.

\*\*Skills Tested:\*\* API interaction, data handling, DOM manipulation.

\*\*Features:\*\*

- Fetch movie data from a movie database API.

- Allow users to rate movies they have watched.

- Recommend movies based on user ratings and preferences.

- Display movie details and trailers.

"""

IPrompt21 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 32. \*\*Recipe Nutrition Calculator\*\*

\*\*Description:\*\* Create a nutrition calculator for recipes to analyze the nutritional content.

\*\*Skills Tested:\*\* API interaction, data handling, DOM manipulation.

\*\*Features:\*\*

- Input recipe ingredients with quantities.

- Fetch nutritional data from a nutrition API.

- Calculate and display total calories, macronutrients, and micronutrients.

- Save and load recipes for future use.

"""

Sure, here's a testable code for a Recipe Nutrition Calculator with some deliberate mistakes to test the interviewee's understanding. This code is designed to be run in a web environment where HTML, CSS, and JavaScript are utilized. You can use this code on platforms like HackerRank or Coderbyte by setting up the appropriate environment.

\*\*index.html\*\*

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Recipe Nutrition Calculator</title>

</head>

<body>

<h1>Recipe Nutrition Calculator</h1>

<div>

<input type="text" id="ingredient" placeholder="Enter ingredient and quantity">

<button onclick="addIngredient()">Add Ingredient</button>

</div>

<ul id="ingredientList"></ul>

<button onclick="calculateNutrition()">Calculate Nutrition</button>

<div id="nutritionResults"></div>

<button onclick="saveRecipe()">Save Recipe</button>

<button onclick="loadRecipe()">Load Recipe</button>

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

</body>

</html>

```

\*\*app.js\*\*

```javascript

let ingredients = [];

function addIngredient() {

const ingredientInput = document.getElementById('ingredient');

const ingredient = ingredientInput.value;

if (ingredient) {

ingredients.push(ingredient);

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

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

li.textContent = ingredient;

ingredientList.appendChild(li);

ingredientInput.value = '';

}

}

function calculateNutrition() {

const apiUrl = 'https://api.nutritionix.com/v1\_1/search/';

const appId = 'YOUR\_APP\_ID'; // Mistake: Replace with actual App ID

const appKey = 'YOUR\_APP\_KEY'; // Mistake: Replace with actual App Key

let totalCalories = 0;

let totalCarbs = 0;

let totalProteins = 0;

let totalFats = 0;

ingredients.forEach(ingredient => {

fetch(`${apiUrl}${ingredient}?results=0:1&fields=nf\_calories,nf\_total\_carbohydrate,nf\_protein,nf\_total\_fat&appId=${appId}&appKey=${appKey}`)

.then(response => response.json())

.then(data => {

const nutrition = data.hits[0].fields;

totalCalories += nutrition.nf\_calories;

totalCarbs += nutrition.nf\_total\_carbohydrate;

totalProteins += nutrition.nf\_protein;

totalFats += nutrition.nf\_total\_fat;

})

.catch(error => console.error('Error:', error));

});

setTimeout(() => { // Mistake: Incorrectly using setTimeout to handle async operations

const resultsDiv = document.getElementById('nutritionResults');

resultsDiv.innerHTML = `

<p>Total Calories: ${totalCalories}</p>

<p>Total Carbohydrates: ${totalCarbs}g</p>

<p>Total Proteins: ${totalProteins}g</p>

<p>Total Fats: ${totalFats}g</p>

`;

}, 2000);

}

function saveRecipe() {

localStorage.setItem('savedRecipe', JSON.stringify(ingredients)); // Mistake: No error handling for localStorage operations

alert('Recipe saved!');

}

function loadRecipe() {

const savedRecipe = localStorage.getItem('savedRecipe');

if (savedRecipe) {

ingredients = JSON.parse(savedRecipe);

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

ingredientList.innerHTML = '';

ingredients.forEach(ingredient => {

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

li.textContent = ingredient;

ingredientList.appendChild(li);

});

} else {

alert('No saved recipe found!');

}

}

```

### Mistakes Introduced:

1. \*\*Hardcoded API keys:\*\* The code contains placeholders for the API keys (`YOUR\_APP\_ID` and `YOUR\_APP\_KEY`). The interviewee should replace these with actual values.

2. \*\*Improper async handling:\*\* The use of `setTimeout` to handle asynchronous operations is incorrect. The interviewee should use proper promise chaining or `async/await`.

3. \*\*No error handling for localStorage operations:\*\* The code does not handle errors that may occur when saving or loading from `localStorage`.

4. \*\*No validation or sanitization of user input:\*\* The code does not validate or sanitize the user input for ingredients.

### Skills Tested:

- \*\*API interaction:\*\* Fetching data from an external API.

- \*\*Data handling:\*\* Aggregating nutritional information from multiple API responses.

- \*\*DOM manipulation:\*\* Updating the DOM with ingredient lists and nutritional information.

- \*\*Error handling and async operations:\*\* Properly managing asynchronous operations and potential errors.

### Instructions for the Interviewee:

1. Replace the placeholder API keys with actual values.

2. Fix the asynchronous handling for fetching nutritional data.

3. Implement error handling for `localStorage` operations.

4. Add input validation and sanitization for the ingredient input.

--------------------------------------------------------------------------------------------------------------

IPrompt22 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 33. \*\*Task Automation with Scripting\*\*

\*\*Description:\*\* Develop a task automation tool where users can write and run simple scripts to automate repetitive tasks.

\*\*Skills Tested:\*\* Scripting, data handling, event handling.

\*\*Features:\*\*

- Provide a text area for writing JavaScript scripts.

- Allow users to run scripts to perform tasks (e.g., data manipulation, calculations).

- Save and load scripts for reuse.

- Display output and error messages.

"""

Sure, here's a testable code snippet for a task automation tool in JavaScript that includes some intentional mistakes. This code can be used on coding platforms like HackerRank or Coderbyte. The mistakes are added to test the interviewee's understanding of scripting, data handling, and event handling.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Task Automation Tool</title>

</head>

<body>

<h1>Task Automation Tool</h1>

<textarea id="scriptArea" rows="10" cols="50" placeholder="Write your script here..."></textarea>

<br>

<button id="runScript">Run Script</button>

<button id="saveScript">Save Script</button>

<button id="loadScript">Load Script</button>

<h3>Output:</h3>

<pre id="outputArea"></pre>

<h3>Error Messages:</h3>

<pre id="errorArea"></pre>

<script>

document.getElementById('runScript').addEventListener('click', function() {

let scriptContent = document.getElementById('scriptArea').value;

try {

let result = eval(scriptContent);

document.getElementById('outputArea').innerText = result;

} catch (error) {

document.getElementById('errorArea').innerText = error.message;

}

});

document.getElementById('saveScript').addEventListener('click', function() {

let scriptContent = document.getElementById('scriptArea').value;

localStorage.setItem('savedScript', scriptContent);

alert('Script saved!');

});

document.getElementById('loadScript').addEventListener('click', function() {

let savedScript = localStorage.getItem('savedScript');

document.getElementById('scriptArea').value = savedScript;

alert('Script loaded!');

});

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Data Handling Issue\*\*: The `eval` function is used directly without any sanitization, which poses a security risk. This is intentional to see if the interviewee can identify and mitigate potential security vulnerabilities.

2. \*\*Error Handling Issue\*\*: The error handling in the `runScript` function only captures the error message but does not clear previous error messages when a new script runs successfully.

3. \*\*Event Handling Issue\*\*: The `loadScript` function does not handle the case when there is no script saved in `localStorage`.

### Interviewee Tasks:

1. \*\*Fix the Security Issue\*\*: Replace `eval` with a safer alternative or sanitize the input script.

2. \*\*Improve Error Handling\*\*: Ensure that the error area is cleared when a new script runs successfully.

3. \*\*Handle Missing Data\*\*: Modify the `loadScript` function to handle cases where no script is saved in `localStorage`.

These tasks will help test the interviewee's skills in scripting, data handling, and event handling.

--------------------------------------------------------------------------------------------------------------

IPrompt23 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 34. \*\*Language Translation App\*\*

\*\*Description:\*\* Build a language translation app using a translation API.

\*\*Skills Tested:\*\* API interaction, DOM manipulation, event handling.

\*\*Features:\*\*

- Input text to be translated.

- Select source and target languages.

- Display translated text.

- Support multiple translation services.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Language Translation App</title>

</head>

<body>

<h1>Language Translation App</h1>

<div>

<textarea id="inputText" placeholder="Enter text to translate"></textarea>

</div>

<div>

<label for="sourceLang">Source Language:</label>

<select id="sourceLang">

<option value="en">English</option>

<option value="es">Spanish</option>

<option value="fr">French</option>

</select>

</div>

<div>

<label for="targetLang">Target Language:</label>

<select id="targetLang">

<option value="es">Spanish</option>

<option value="en">English</option>

<option value="fr">French</option>

</select>

</div>

<button id="translateBtn">Translate</button>

<div>

<h2>Translated Text:</h2>

<p id="translatedText"></p>

</div>

<script>

document.getElementById('translateBtn').addEventListener('click', function() {

const inputText = document.getElementById('inputText').value;

const sourceLang = document.getElementById('sourceLang').value;

const targetLang = document.getElementById('targetLang').value;

// Intentional mistake: Incorrect API endpoint

const apiUrl = `https://libretranslate.de/translate`;

fetch(apiUrl, {

method: 'POST',

body: JSON.stringify({

q: inputText,

source: sourceLang,

target: targetLang,

format: "text"

}),

headers: { "Content-Type": "application/json" }

})

.then(response => response.json())

.then(data => {

// Intentional mistake: Incorrect property name

document.getElementById('translatedText').innerText = data.translatedText;

})

.catch(error => console.error('Error:', error));

});

</script>

</body>

</html>

```

IPrompt24 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 35. \*\*Interactive Storybook for Kids\*\*

\*\*Description:\*\* Create an interactive storybook with animations and sound effects for kids.

\*\*Skills Tested:\*\* DOM manipulation, event handling, animations.

\*\*Features:\*\*

- Display pages of a story with text and images.

- Add interactive elements (e.g., clickable characters, sound effects).

- Animate transitions between pages.

- Provide narration for the story.

"""

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Storybook</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f9f9f9;

text-align: center;

}

.page {

display: none;

transition: opacity 1s;

}

.page.active {

display: block;

opacity: 1;

}

.image {

width: 300px;

height: 200px;

}

.clickable {

cursor: pointer;

color: blue;

}

</style>

</head>

<body>

<div id="storybook">

<div class="page" id="page1">

<h1>Page 1</h1>

<p>Once upon a time, there was a <span class="clickable" id="character1">little bear</span>.</p>

<img src="bear.jpg" alt="Little Bear" class="image">

<button onclick="nextPage()">Next</button>

</div>

<div class="page" id="page2">

<h1>Page 2</h1>

<p>The little bear loved to play in the <span class="clickable" id="character2">forest</span>.</p>

<img src="forest.jpg" alt="Forest" class="image">

<button onclick="nextPage()">Next</button>

</div>

<div class="page" id="page3">

<h1>Page 3</h1>

<p>One day, the bear met a <span class="clickable" id="character3">friendly fox</span>.</p>

<img src="fox.jpg" alt="Friendly Fox" class="image">

<button onclick="nextPage()">Next</button>

</div>

</div>

<audio id="narration" src="narration.mp3"></audio>

<script>

const pages = document.querySelectorAll('.page');

let currentPageIndex = 0;

function showPage(index) {

pages.forEach((page, idx) => {

if (idx === index) {

page.classList.add('active');

} else {

page.classList.remove('active');

}

});

}

function nextPage() {

if (currentPageIndex < pages.length - 1) {

currentPageIndex++;

showPage(currentPageIndex);

playNarration();

}

}

function playNarration() {

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

narration.play();

}

document.getElementById('character1').addEventListener('click', () => {

alert('The little bear says hello!');

playSoundEffect('bear.mp3');

});

document.getElementById('character2').addEventListener('click', () => {

alert('The forest is full of wonders!');

playSoundEffect('forest.mp3');

});

document.getElementById('character3').addEventListener('click', () => {

alert('The friendly fox greets you!');

playSoundEffect('fox.mp3');

function playSoundEffect(soundFile) {

const audio = new Audio(soundFile);

audio.play();

}

// Initialize the first page

showPage(currentPageIndex);

</script>

</body>

</html>

```

### Mistakes Introduced:

1. \*\*CSS Animation Mistake:\*\*

- The CSS class `.page.active` has `opacity: 1` but the pages that are not active do not have `opacity: 0`, which can cause issues with the transition effect.

2. \*\*Incorrect Event Handling:\*\*

- The event listeners for the characters are set up in a way that may not work if the elements are not correctly referenced.

3. \*\*Narration Audio Mistake:\*\*

- The narration audio plays on every page transition without any specific narration tied to each page, which is not ideal for a storybook.

4. \*\*Next Page Button:\*\*

- The `nextPage` function does not handle the case where you are on the last page and try to go next, potentially causing issues.

These mistakes are intentional to test the interviewee's ability to identify and fix common issues in DOM manipulation, event handling, and animations.

IPrompt25 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 36. \*\*Smart Home Controller\*\*

\*\*Description:\*\* Develop a smart home controller to manage and monitor smart devices.

\*\*Skills Tested:\*\* API interaction, real-time data handling, DOM manipulation.

\*\*Features:\*\*

- Connect to smart home devices (e.g., lights, thermostat, security cameras).

- Control devices (e.g., turn lights on/off, adjust thermostat).

- Display real-time status and data from devices.

- Implement schedules and automation rules.

"""

Sure! Below is a sample code for a smart home controller application. This code includes some intentional mistakes to test the interviewee's understanding of API interaction, real-time data handling, and DOM manipulation.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Smart Home Controller</title>

<style>

.device {

margin-bottom: 20px;

}

</style>

</head>

<body>

<h1>Smart Home Controller</h1>

<div id="devices"></div>

<script>

// Mock API interaction

const devices = [

{ id: 1, name: 'Living Room Light', type: 'light', status: 'off' },

{ id: 2, name: 'Thermostat', type: 'thermostat', temperature: 22 },

{ id: 3, name: 'Security Camera', type: 'camera', status: 'on' }

];

function fetchDevices() {

return new Promise((resolve, reject) => {

setTimeout(() => resolve(devices), 1000);

});

}

function updateDevice(deviceId, data) {

return new Promise((resolve, reject) => {

const device = devices.find(d => d.id === deviceId);

if (device) {

Object.assign(device, data);

setTimeout(() => resolve(device), 500);

} else {

reject('Device not found');

}

});

}

// Real-time data handling

function pollDevices() {

fetchDevices().then(devices => {

const devicesDiv = document.getElementById('devices');

devicesDiv.innerHTML = '';

devices.forEach(device => {

const deviceDiv = document.createElement('div');

deviceDiv.className = 'device';

deviceDiv.innerHTML = `

<h2>${device.name}</h2>

<p>Type: ${device.type}</p>

<p>Status: ${device.status || ''}</p>

<p>Temperature: ${device.temperature || ''}</p>

<button onclick="toggleDevice(${device.id})">Toggle</button>

<input type="number" value="${device.temperature || ''}" onchange="adjustTemperature(${device.id}, this.value)">

`;

devicesDiv.appendChild(deviceDiv);

});

});

}

function toggleDevice(deviceId) {

updateDevice(deviceId, { status: devices[deviceId - 1].status === 'on' ? 'off' : 'on' })

.then(pollDevices)

.catch(error => console.error(error));

}

function adjustTemperature(deviceId, temperature) {

updateDevice(deviceId, { temperature: parseInt(temperature) })

.then(pollDevices)

.catch(error => console.error(error));

}

// Initialize

pollDevices();

setInterval(pollDevices, 5000);

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Device ID Handling:\*\* The `toggleDevice` function incorrectly assumes that `deviceId` can be directly used as an index (`devices[deviceId - 1]`). This will fail if device IDs are not sequential or start from 1.

2. \*\*Status Display Logic:\*\* The `status` and `temperature` properties are sometimes displayed even if they are `undefined` or not relevant to the device type.

3. \*\*Real-Time Polling:\*\* The `setInterval` function is used for polling, but it doesn't account for the possibility of overlapping requests if one takes longer than the interval.

4. \*\*DOM Manipulation Efficiency:\*\* The entire device list is re-rendered every time `pollDevices` is called, which is not efficient for large numbers of devices.

These mistakes should help test an interviewee's ability to identify and correct common pitfalls in such an application.

--------------------------------------------------------------------------------------------------------------

IPrompt26 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 37. \*\*Virtual Whiteboard\*\*

\*\*Description:\*\* Create a virtual whiteboard for collaborative drawing and note-taking.

\*\*Skills Tested:\*\* Canvas API, WebSockets, event handling.

\*\*Features:\*\*

- Allow multiple users to draw and write on the whiteboard in real-time.

- Support drawing tools (e.g., pen, eraser, shapes).

- Save and load whiteboard sessions.

- Implement a chat feature for communication.

"""

Certainly! Below is a sample code for a virtual whiteboard application that incorporates the features listed. The code includes intentional mistakes to test the interviewee's understanding of the main concepts such as Canvas API, WebSockets, and event handling.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Virtual Whiteboard</title>

<style>

#whiteboard {

border: 1px solid black;

width: 800px;

height: 600px;

}

#toolbar {

margin-bottom: 10px;

}

#chat {

width: 300px;

height: 600px;

float: right;

border: 1px solid black;

}

</style>

</head>

<body>

<div id="toolbar">

<button id="pen">Pen</button>

<button id="eraser">Eraser</button>

<button id="save">Save</button>

<button id="load">Load</button>

</div>

<canvas id="whiteboard"></canvas>

<div id="chat">

<div id="messages"></div>

<input type="text" id="messageInput" />

<button id="send">Send</button>

</div>

<script>

const canvas = document.getElementById('whiteboard');

const ctx = canvas.getContext('2d');

const socket = new WebSocket('wss://example.com/socket');

let drawing = false;

let tool = 'pen';

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

drawing = true;

ctx.beginPath();

ctx.moveTo(e.offsetX, e.offsetY);

});

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

if (!drawing) return;

if (tool === 'pen') {

ctx.lineTo(e.offsetX, e.offsetY);

ctx.stroke();

} else if (tool === 'eraser') {

ctx.clearRect(e.offsetX, e.offsetY, 10, 10);

}

socket.send(JSON.stringify({ x: e.offsetX, y: e.offsetY, tool }));

});

canvas.addEventListener('mouseup', () => {

drawing = false;

});

document.getElementById('pen').addEventListener('click', () => {

tool = 'pen';

});

document.getElementById('eraser').addEventListener('click', () => {

tool = 'eraser';

});

document.getElementById('save').addEventListener('click', () => {

const data = canvas.toDataURL();

localStorage.setItem('whiteboard', data);

});

document.getElementById('load').addEventListener('click', () => {

const data = localStorage.getItem('whiteboard');

const img = new Image();

img.src = data;

img.onload = () => {

ctx.drawImage(img, 0, 0);

};

});

socket.onmessage = (event) => {

const data = JSON.parse(event.data);

if (data.tool === 'pen') {

ctx.lineTo(data.x, data.y);

ctx.stroke();

} else if (data.tool === 'eraser') {

ctx.clearRect(data.x, data.y, 10, 10);

}

};

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

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

document.getElementById('send').addEventListener('click', () => {

const message = messageInput.value;

socket.send(JSON.stringify({ message }));

messages.innerHTML += '<p>' + message + '</p>';

messageInput.value = '';

});

socket.onmessage = (event) => {

const data = JSON.parse(event.data);

if (data.message) {

messages.innerHTML += '<p>' + data.message + '</p>';

}

};

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Canvas Size\*\*: The canvas element does not have its width and height attributes set, which means it defaults to 300x150 pixels regardless of the CSS styles.

2. \*\*WebSocket URL\*\*: The WebSocket URL (`wss://example.com/socket`) is a placeholder and will not work in a real scenario.

3. \*\*Duplicate `socket.onmessage`\*\*: There are two `socket.onmessage` event handlers, which will cause one to overwrite the other.

4. \*\*LocalStorage Save/Load\*\*: The `load` button does not handle cases where there is no saved data in localStorage.

5. \*\*Drawing Logic\*\*: The drawing logic inside the `mousemove` event listener does not handle the initial `mousedown` position correctly for remote users.

These mistakes provide an opportunity to test the interviewee's ability to identify and correct issues related to the Canvas API, WebSockets, and event handling.

IPrompt27 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 38. \*\*Health and Wellness Tracker\*\*

\*\*Description:\*\* Develop a health and wellness tracker to monitor daily habits and activities.

\*\*Skills Tested:\*\* Data handling, DOM manipulation, event handling.

\*\*Features:\*\*

- Track habits (e.g., water intake, sleep, exercise).

- Set and track wellness goals.

- Display progress and trends over time.

- Provide tips and suggestions based on data.

"""

Implementation of a Health and Wellness Tracker in JavaScript that includes some intentional mistakes. This code can be used to test data handling, DOM manipulation, and event handling skills. The mistakes are introduced to assess the interviewee's debugging and problem-solving abilities.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Health and Wellness Tracker</title>

</head>

<body>

<h1>Health and Wellness Tracker</h1>

<div>

<h2>Track Your Habits</h2>

<label for="water-intake">Water Intake (cups):</label>

<input type="number" id="water-intake" value="0">

<br>

<label for="sleep-hours">Sleep (hours):</label>

<input type="number" id="sleep-hours" value="0">

<br>

<label for="exercise-minutes">Exercise (minutes):</label>

<input type="number" id="exercise-minutes" value="0">

<br>

<button id="track-button">Track</button>

</div>

<div>

<h2>Wellness Goals</h2>

<label for="water-goal">Water Intake Goal (cups):</label>

<input type="number" id="water-goal" value="8">

<br>

<label for="sleep-goal">Sleep Goal (hours):</label>

<input type="number" id="sleep-goal" value="8">

<br>

<label for="exercise-goal">Exercise Goal (minutes):</label>

<input type="number" id="exercise-goal" value="30">

</div>

<div>

<h2>Your Progress</h2>

<p id="progress-display">Track your habits to see progress.</p>

</div>

<div>

<h2>Tips and Suggestions</h2>

<p id="tips-display">Get tips based on your tracked data.</p>

</div>

<script>

document.getElementById('track-button').addEventListener('click', function() {

let waterIntake = parseInt(document.getElementById('water-intake').value);

let sleepHours = parseInt(document.getElementById('sleep-hours').value);

let exerciseMinutes = parseInt(document.getElementById('exercise-minutes').value);

let waterGoal = parseInt(document.getElementById('water-goal').value);

let sleepGoal = parseInt(document.getElementById('sleep-goal').value);

let exerciseGoal = parseInt(document.getElementById('exercise-goal').value);

let waterProgress = (waterIntake / waterGoal) \* 100;

let sleepProgress = (sleepHours / sleepGoal) \* 100;

let exerciseProgress = (exerciseMinutes / exerciseGoal) \* 100;

let progressDisplay = `

Water Intake: ${waterProgress}%<br>

Sleep: ${sleepProgress}%<br>

Exercise: ${exerciseProgress}%

`;

document.getElementById('progress-display').innerHTML = progressDisplay;

let tips = "";

if (waterIntake < waterGoal) {

tips += "Drink more water. ";

}

if (sleepHours < sleepGoal) {

tips += "Try to get more sleep. ";

}

if (exerciseMinutes < exerciseGoal) {

tips += "Increase your exercise duration. ";

}

document.getElementById('tips-display').innerText = tips;

});

</script>

</body>

</html>

```

### Intentional Mistakes:

1. The `parseInt` function is used without specifying a radix.

2. The `innerText` property is used instead of `innerHTML` for the `tips-display`.

3. The initial values for water intake, sleep, and exercise are set to `0` which might not be realistic for a user�s initial data.

4. The check for tips is inclusive, meaning it also triggers when goals are met exactly, which might not be the intended behavior.

These mistakes should help assess the candidate's ability to identify and correct issues related to data handling, DOM manipulation, and event handling.

IPrompt28 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 39. \*\*Interactive Resume Builder\*\*

\*\*Description:\*\* Create an interactive resume builder to help users create professional resumes.

\*\*Skills Tested:\*\* DOM manipulation, form handling, data management.

\*\*Features:\*\*

- Input personal details, work experience, education, and skills.

- Choose from different resume templates.

- Preview and download the resume as a PDF.

- Save and load resume data.

"""

Certainly! Below is a sample code for an interactive resume builder with intentional mistakes. This code is designed to test the interviewee's skills in DOM manipulation, form handling, and data management. It includes features like inputting personal details, work experience, education, and skills, choosing from different resume templates, previewing and downloading the resume as a PDF, and saving and loading resume data.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Interactive Resume Builder</title>

<style>

/\* Basic styling for the form and resume preview \*/

.container {

display: flex;

justify-content: space-between;

padding: 20px;

}

.form-container, .resume-container {

width: 45%;

}

.resume {

border: 1px solid #000;

padding: 20px;

}

</style>

</head>

<body>

<div class="container">

<div class="form-container">

<h2>Resume Builder</h2>

<form id="resume-form">

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

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

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

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

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

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

<label for="work-experience">Work Experience:</label>

<textarea id="work-experience" name="work-experience"></textarea><br><br>

<label for="education">Education:</label>

<textarea id="education" name="education"></textarea><br><br>

<label for="skills">Skills:</label>

<textarea id="skills" name="skills"></textarea><br><br>

<label for="template">Choose Template:</label>

<select id="template" name="template">

<option value="template1">Template 1</option>

<option value="template2">Template 2</option>

</select><br><br>

<button type="button" onclick="previewResume()">Preview Resume</button>

<button type="button" onclick="downloadResume()">Download as PDF</button>

<button type="button" onclick="saveResume()">Save Resume</button>

<button type="button" onclick="loadResume()">Load Resume</button>

</form>

</div>

<div class="resume-container">

<h2>Resume Preview</h2>

<div class="resume" id="resume-preview"></div>

</div>

</div>

<script>

function previewResume() {

const name = document.getElementById('name').value;

const email = document.getElementById('email').value;

const phone = document.getElementById('phone').value;

const workExperience = document.getElementById('work-experience').value;

const education = document.getElementById('education').value;

const skills = document.getElementById('skills').value;

const template = document.getElementById('template').value;

const resumePreview = document.getElementById('resume-preview');

// Clear previous content

resumePreview.innerHTML = '';

if (template === 'template1') {

resumePreview.innerHTML += `<h1>${name}</h1>`;

resumePreview.innerHTML += `<p>Email: ${email}</p>`;

resumePreview.innerHTML += `<p>Phone: ${phone}</p>`;

resumePreview.innerHTML += `<h2>Work Experience</h2><p>${workExperience}</p>`;

resumePreview.innerHTML += `<h2>Education</h2><p>${education}</p>`;

resumePreview.innerHTML += `<h2>Skills</h2><p>${skills}</p>`;

} else if (template === 'template2') {

resumePreview.innerHTML += `<h1>${name}</h1>`;

resumePreview.innerHTML += `<h2>Contact</h2><p>Email: ${email}</p><p>Phone: ${phone}</p>`;

resumePreview.innerHTML += `<h2>Experience</h2><p>${workExperience}</p>`;

resumePreview.innerHTML += `<h2>Academic Background</h2><p>${education}</p>`;

resumePreview.innerHTML += `<h2>Capabilities</h2><p>${skills}</p>`;

}

}

function downloadResume() {

const resumeContent = document.getElementById('resume-preview').innerHTML;

const blob = new Blob([resumeContent], { type: 'application/pdf' });

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = 'resume.pdf';

document.body.appendChild(a);

a.click();

document.body.removeChild(a);

}

function saveResume() {

const resumeData = {

name: document.getElementById('name').value,

email: document.getElementById('email').value,

phone: document.getElementById('phone').value,

workExperience: document.getElementById('work-experience').value,

education: document.getElementById('education').value,

skills: document.getElementById('skills').value,

template: document.getElementById('template').value,

};

localStorage.setItem('resumeData', JSON.stringify(resumeData));

alert('Resume saved successfully!');

}

function loadResume() {

const resumeData = JSON.parse(localStorage.getItem('resumeData'));

if (resumeData) {

document.getElementById('name').value = resumeData.name;

document.getElementById('email').value = resumeData.email;

document.getElementById('phone').value = resumeData.phone;

document.getElementById('work-experience').value = resumeData.workExperience;

document.getElementById('education').value = resumeData.education;

document.getElementById('skills').value = resumeData.skills;

document.getElementById('template').value = resumeData.template;

previewResume();

} else {

alert('No resume data found!');

}

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. The `downloadResume` function is attempting to download HTML content as a PDF without using a proper library to convert HTML to PDF.

2. The form elements lack proper validation for the required fields.

3. The `previewResume` function does not handle cases where the user leaves fields empty.

4. The `loadResume` function does not check if `resumeData` is `null` before trying to access its properties.

5. The `previewResume` function does not handle different templates properly (both templates are too similar).

These mistakes are designed to test the interviewee's ability to identify and correct issues in form handling, data management, and basic DOM manipulation.

--------------------------------------------------------------------------------------------------------------

IPrompt29 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 40. \*\*Virtual Classroom\*\*

\*\*Description:\*\* Develop a virtual classroom platform for online learning and collaboration.

\*\*Skills Tested:\*\* WebRTC, WebSockets, DOM manipulation.

\*\*Features:\*\*

- Support video conferencing for live classes.

- Share screens and files during sessions.

- Provide a chat feature for discussions.

- Record and playback class sessions.

"""

Sure, here's a simplified version of a virtual classroom application. This code includes some intentional mistakes that candidates need to identify and correct. The code is designed to test skills in WebRTC, WebSockets, and DOM manipulation, and it can be used on coding platforms like HackerRank or Coderbyte.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Virtual Classroom</title>

<style>

#video-grid {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));

gap: 10px;

}

video {

width: 100%;

border: 1px solid black;

}

#chat {

border: 1px solid gray;

height: 300px;

overflow-y: scroll;

}

#chat input {

width: 80%;

}

</style>

</head>

<body>

<h1>Virtual Classroom</h1>

<div id="video-grid"></div>

<button id="share-screen">Share Screen</button>

<button id="start-record">Start Recording</button>

<button id="stop-record">Stop Recording</button>

<div id="chat">

<div id="messages"></div>

<input id="message-input" type="text" placeholder="Type a message">

<button id="send-message">Send</button>

</div>

<script>

const videoGrid = document.getElementById('video-grid');

const shareScreenButton = document.getElementById('share-screen');

const startRecordButton = document.getElementById('start-record');

const stopRecordButton = document.getElementById('stop-record');

const messageInput = document.getElementById('message-input');

const sendMessageButton = document.getElementById('send-message');

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

let localStream;

let screenStream;

let mediaRecorder;

let recordedChunks = [];

async function startVideo() {

localStream = await navigator.mediaDevices.getUserMedia({ video: true, audio: true });

const videoElement = document.createElement('video');

videoElement.srcObject = localStream;

videoElement.play();

videoGrid.appendChild(videoElement);

}

shareScreenButton.addEventListener('click', async () => {

screenStream = await navigator.mediaDevices.getDisplayMedia({ video: true });

const screenVideoElement = document.createElement('video');

screenVideoElement.srcObject = screenStream;

screenVideoElement.play();

videoGrid.appendChild(screenVideoElement);

});

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

mediaRecorder = new MediaRecorder(localStream);

mediaRecorder.ondataavailable = (event) => {

if (event.data.size > 0) {

recordedChunks.push(event.data);

}

};

mediaRecorder.start();

});

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

mediaRecorder.stop();

const blob = new Blob(recordedChunks, { type: 'video/webm' });

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.style.display = 'none';

a.href = url;

a.download = 'recorded-class.webm';

document.body.appendChild(a);

a.click();

window.URL.revokeObjectURL(url);

});

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

const message = messageInput.value;

const messageElement = document.createElement('div');

messageElement.innerText = message;

messages.appendChild(messageElement);

messageInput.value = '';

});

startVideo();

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*WebRTC Video Stream\*\*: The `startVideo` function initializes the video stream but does not handle errors that might occur when accessing the user's media devices.

2. \*\*Screen Sharing\*\*: The shared screen stream does not include audio.

3. \*\*Recording\*\*: The `mediaRecorder` is only set to record the local stream, not the screen stream if it is shared. Also, the recorded chunks are not cleared after stopping the recording.

4. \*\*Chat Feature\*\*: The chat feature does not use WebSockets, so messages cannot be sent to other users in real-time.

5. \*\*DOM Manipulation\*\*: The video elements are not correctly removed from the DOM when the stream ends.

Candidates need to identify and fix these issues to ensure the application works as intended.

IPrompt30 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 41. \*\*Interactive Map Application\*\*

\*\*Description:\*\* Build an interactive map application with custom markers and routes.

\*\*Skills Tested:\*\* Map APIs, DOM manipulation, event handling.

\*\*Features:\*\*

- Display a map with the ability to add custom markers.

- Calculate and display routes between points.

- Fetch and display data for locations (e.g., weather, points of interest).

- Save and load map configurations.

"""

Here's a code snippet for an interactive map application with intentional mistakes. This application uses the Leaflet.js library for map rendering and manipulation. The code includes mistakes that test the interviewee's understanding of Map APIs, DOM manipulation, and event handling.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Interactive Map Application</title>

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

<style>

#map { height: 600px; width: 100%; }

</style>

</head>

<body>

<h1>Interactive Map Application</h1>

<div id="map"></div>

<button id="addMarkerBtn">Add Marker</button>

<button id="calculateRouteBtn">Calculate Route</button>

<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

<script>

// Initialize the map

var map = L.map('map').setView([51.505, -0.09], 13);

// Add OpenStreetMap tiles

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map);

var markers = [];

var routes = [];

// Function to add a marker to the map

function addMarker(lat, lng) {

var marker = L.marker([lat, lng]).addTo(map);

markers.push(marker);

}

// Event listener for the 'Add Marker' button

document.getElementById('addMarkerBtn').addEventListener('click', function() {

// Hardcoded coordinates for the sake of simplicity

addMarker(51.5, -0.09);

});

// Function to calculate and display a route between markers

function calculateRoute() {

if (markers.length < 2) {

alert("At least two markers are needed to calculate a route");

return;

}

var latlngs = markers.map(function(marker) {

return marker.getLatLng();

});

var route = L.polyline(latlngs, {color: 'blue'}).addTo(map);

routes.push(route);

}

// Event listener for the 'Calculate Route' button

document.getElementById('calculateRouteBtn').addEventListener('click', calculateRoute);

// Function to fetch and display location data (e.g., weather)

function fetchLocationData(lat, lng) {

fetch(`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lng}&appid=YOUR\_API\_KEY`)

.then(response => response.json())

.then(data => {

console.log('Weather data:', data);

alert(`Weather at (${lat}, ${lng}): ${data.weather[0].description}`);

})

.catch(error => console.error('Error fetching weather data:', error));

}

// Intentional mistake: Missing event listener to fetch data for specific markers

// This needs to be implemented by the interviewee

</script>

</body>

</html>

```

### Mistakes to Test Interviewee:

1. \*\*Hardcoded Coordinates in `addMarker` Function\*\*: The `addMarker` function uses hardcoded coordinates. Interviewees should modify the function to take user input for coordinates.

2. \*\*Missing Event Listener for Fetching Location Data\*\*: The code includes a function to fetch location data but does not have an event listener to trigger it. Interviewees should add an event listener to fetch and display data when a marker is clicked.

3. \*\*API Key Placeholder\*\*: The OpenWeatherMap API URL contains a placeholder (`YOUR\_API\_KEY`). Interviewees should replace it with a valid API key.

4. \*\*No Save/Load Functionality\*\*: The application lacks functionality to save and load map configurations. Interviewees should implement this feature.

These mistakes will help assess the interviewee's ability to work with map APIs, handle DOM events, and understand the overall structure of a web application.

IPrompt31 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 42. \*\*Voice Command Application\*\*

\*\*Description:\*\* Create an application that responds to voice commands to perform tasks.

\*\*Skills Tested:\*\* Web Speech API, DOM manipulation, event handling.

\*\*Features:\*\*

- Recognize and process voice commands.

- Perform tasks based on commands (e.g., search, navigate, control media).

- Provide feedback and confirmation for commands.

- Support custom command configurations.

"""

Certainly! Here is a testable code snippet for a Voice Command Application. The code includes some intentional mistakes to test the interviewee's understanding of the main concepts.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Voice Command Application</title>

<style>

body {

font-family: Arial, sans-serif;

}

#status {

margin-top: 20px;

font-size: 1.2em;

}

</style>

</head>

<body>

<h1>Voice Command Application</h1>

<button id="start-recognition">Start Voice Recognition</button>

<div id="status">Press the button and speak a command.</div>

<div id="feedback"></div>

<script>

// Check if the browser supports the Web Speech API

if (!('webkitSpeechRecognition' in window)) {

alert("Your browser does not support the Web Speech API. Please use Google Chrome.");

}

// Initialize the Web Speech API

var recognition = new webkitSpeechRecognition();

recognition.continuous = true;

recognition.interimResults = false;

recognition.lang = 'en-US';

// Start recognition when the button is clicked

document.getElementById('start-recognition').addEventListener('click', function() {

recognition.start();

document.getElementById('status').innerText = "Listening...";

});

// Handle the recognition result

recognition.onresult = function(event) {

var command = event.results[0][0].transcript.trim();

document.getElementById('feedback').innerText = "You said: " + command;

processCommand(command);

};

// Handle speech recognition errors

recognition.onerror = function(event) {

document.getElementById('status').innerText = "Error occurred in recognition: " + event.error;

};

// Process the voice command

function processCommand(command) {

if (command === "search") {

window.location.href = "https://www.google.com/search?q=test";

} else if (command === "navigate") {

window.location.href = "https://www.example.com";

} else if (command === "play music") {

var audio = new Audio('music.mp3');

audio.play();

} else {

document.getElementById('feedback').innerText = "Command not recognized.";

}

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Recognition Initialization\*\*: The browser check only alerts the user but does not prevent further execution if the browser is unsupported.

2. \*\*Recognition Scope\*\*: `webkitSpeechRecognition` is not supported in all browsers. Using `SpeechRecognition` as a fallback is more robust.

3. \*\*Continuous Recognition\*\*: The recognition is continuous, which may not be desired for a simple command application.

4. \*\*Command Handling\*\*: The `processCommand` function has hardcoded commands and does not support custom configurations.

5. \*\*Result Indexing\*\*: `event.results[0][0].transcript` should properly iterate through the results for more robust handling.

6. \*\*Error Handling\*\*: The error message should be more user-friendly.

### Points to Test:

1. \*\*Browser Compatibility\*\*: Ensure the application correctly handles unsupported browsers.

2. \*\*Voice Recognition\*\*: Ensure commands are recognized and processed correctly.

3. \*\*Event Handling\*\*: Ensure events (click, recognition result, error) are handled properly.

4. \*\*DOM Manipulation\*\*: Ensure the feedback and status updates are correct.

5. \*\*Custom Command Configurations\*\*: Ask the interviewee to modify the code to support custom commands.

This code serves as a good starting point to test an interviewee's understanding and ability to debug and enhance the application.

--------------------------------------------------------------------------------------------------------------

IPrompt32 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 43. \*\*Customizable Dashboard\*\*

\*\*Description:\*\* Develop a customizable dashboard where users can add and arrange widgets.

\*\*Skills Tested:\*\* DOM manipulation, event handling, data handling.

\*\*Features:\*\*

- Provide various widgets (e.g., weather, news, calendar).

- Allow users to add, remove, and rearrange widgets.

- Save and load dashboard configurations.

- Support different themes and layouts.

"""

Sure, here is a JavaScript code snippet for a customizable dashboard application. The code includes a few intentional mistakes that can be used to test the skills of the interviewee in DOM manipulation, event handling, and data handling. The code is designed to be testable on coding platforms like HackerRank or Coderbyte.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Customizable Dashboard</title>

<style>

body {

font-family: Arial, sans-serif;

}

#dashboard {

display: flex;

flex-wrap: wrap;

}

.widget {

border: 1px solid #ccc;

padding: 10px;

margin: 10px;

width: 200px;

height: 150px;

}

</style>

</head>

<body>

<h1>Customizable Dashboard</h1>

<div>

<button id="addWidgetBtn">Add Widget</button>

<select id="widgetType">

<option value="weather">Weather</option>

<option value="news">News</option>

<option value="calendar">Calendar</option>

</select>

</div>

<div id="dashboard"></div>

<script>

document.getElementById('addWidgetBtn').addEventListener('click', function() {

const widgetType = document.getElementById('widgetType').value;

const widget = document.createElement('div');

widget.className = 'widget';

widget.innerHTML = `<h2>${widgetType}</h2>`;

// Intentional mistake: Should add remove button to widget

const removeBtn = document.createElement('button');

removeBtn.innerText = 'Remove';

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

// Intentional mistake: Incorrect parent node for removal

document.getElementById('dashboard').removeChild(widget);

});

widget.appendChild(removeBtn);

// Intentional mistake: Should append to dashboard

document.body.appendChild(widget);

});

// Intentional mistake: Incorrectly named function, should saveDashboard

function saveDasboard() {

const widgets = document.querySelectorAll('.widget');

const config = [];

widgets.forEach(widget => {

config.push(widget.querySelector('h2').innerText);

});

localStorage.setItem('dashboardConfig', JSON.stringify(config));

}

// Intentional mistake: Incorrectly named function, should loadDashboard

function loadDasboard() {

const config = JSON.parse(localStorage.getItem('dashboardConfig') || '[]');

config.forEach(widgetType => {

const widget = document.createElement('div');

widget.className = 'widget';

widget.innerHTML = `<h2>${widgetType}</h2>`;

const removeBtn = document.createElement('button');

removeBtn.innerText = 'Remove';

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

document.getElementById('dashboard').removeChild(widget);

});

widget.appendChild(removeBtn);

document.getElementById('dashboard').appendChild(widget);

});

}

// Intentional mistake: Should call loadDashboard

loadDasboard();

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Remove Button Addition\*\*: The remove button should be added to the widget before appending the widget to the dashboard.

2. \*\*Incorrect Parent Node for Removal\*\*: When removing a widget, the parent node should be the widget itself, not the dashboard.

3. \*\*Appending to Body Instead of Dashboard\*\*: The widget should be appended to the dashboard, not the body.

4. \*\*Incorrect Function Names\*\*: The functions `saveDasboard` and `loadDasboard` are misspelled. They should be `saveDashboard` and `loadDashboard`.

5. \*\*Incorrect Function Call\*\*: The `loadDasboard` function is called instead of `loadDashboard`.

These mistakes will help you test the interviewee's ability to identify and correct errors related to DOM manipulation, event handling, and data handling.

IPrompt33 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 44. \*\*Personal Finance Manager\*\*

\*\*Description:\*\* Build a personal finance manager to track income, expenses, and investments.

\*\*Skills Tested:\*\* Data handling, DOM manipulation, data visualization.

\*\*Features:\*\*

- Add, edit, and categorize financial transactions.

- Track income and expenses over time.

- Visualize financial data with charts.

- Provide budget planning and analysis tools.

"""

Certainly! Below is a sample code for a simple Personal Finance Manager application with intentional mistakes for testing purposes. This code is intended to be run in a web browser environment, and it uses basic HTML, CSS, and JavaScript for simplicity. The mistakes are injected to test the candidate's understanding of data handling, DOM manipulation, and basic debugging.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Personal Finance Manager</title>

<style>

body {

font-family: Arial, sans-serif;

}

.container {

width: 80%;

margin: 0 auto;

}

.form-group {

margin-bottom: 15px;

}

.form-group label {

display: block;

}

.form-group input, .form-group select {

width: 100%;

padding: 8px;

margin-top: 5px;

}

#transactions {

margin-top: 20px;

}

.chart {

margin-top: 20px;

}

</style>

</head>

<body>

<div class="container">

<h1>Personal Finance Manager</h1>

<div class="form-group">

<label for="type">Type</label>

<select id="type">

<option value="income">Income</option>

<option value="expense">Expense</option>

<option value="investment">Investment</option>

</select>

</div>

<div class="form-group">

<label for="amount">Amount</label>

<input type="number" id="amount">

</div>

<div class="form-group">

<label for="category">Category</label>

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

</div>

<button id="addTransaction">Add Transaction</button>

<div id="transactions">

<h2>Transactions</h2>

<ul id="transactionList"></ul>

</div>

<div class="chart">

<canvas id="financeChart" width="400" height="200"></canvas>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>

const transactions = [];

document.getElementById('addTransaction').addEventListener('click', function() {

const type = document.getElementById('type').value;

const amount = parseFloat(document.getElementById('amount').value);

const category = document.getElementById('category').value;

if (isNaN(amount) || amount <= 0) {

alert('Please enter a valid amount');

return;

}

const transaction = { type, amount, category };

transactions.push(transaction);

renderTransactions();

updateChart();

});

function renderTransactions() {

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

transactionList.innerHTML = '';

transactions.forEach((transaction, index) => {

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

li.textContent = `${transaction.type}: $${transaction.amount} - ${transaction.category}`;

transactionList.appendChild(li);

});

}

function updateChart() {

const ctx = document.getElementById('financeChart').getContext('2d');

const income = transactions.filter(t => t.type === 'income').reduce((acc, t) => acc + t.amount, 0);

const expenses = transactions.filter(t => t.type === 'expense').reduce((acc, t) => acc + t.amount, 0);

const investments = transactions.filter(t => t.type === 'investment').reduce((acc, t) => acc + t.amount, 0);

const chart = new Chart(ctx, {

type: 'bar',

data: {

labels: ['Income', 'Expenses', 'Investments'],

datasets: [{

label: 'Financial Data',

data: [income, expenses, investments],

backgroundColor: ['green', 'red', 'blue']

}]

},

options: {

scales: {

y: {

beginAtZero: true

}

}

}

});

}

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Data Handling Mistake\*\*: The `transactions` array is not persisted, meaning data will be lost on page reload.

2. \*\*DOM Manipulation Mistake\*\*: The `li` elements created for each transaction do not have a way to be edited or deleted.

3. \*\*Chart Mistake\*\*: The `updateChart` function creates a new chart every time a transaction is added, which can cause performance issues and incorrect rendering.

4. \*\*Validation Mistake\*\*: There is no validation for the `category` input, which can lead to empty or invalid categories.

5. \*\*Accessibility Mistake\*\*: The form does not have a proper structure for screen readers and accessibility tools.

These mistakes should prompt the candidate to think critically and apply their knowledge of web development to fix and improve the application.

IPrompt34 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 45. \*\*Interactive Story Builder\*\*

\*\*Description:\*\* Create an interactive story builder where users can write and share stories with branching paths.

\*\*Skills Tested:\*\* DOM manipulation, data management, event handling.

\*\*Features:\*\*

- Allow users to create stories with multiple choices and outcomes.

- Save and share stories with others.

- Track and visualize story paths.

- Provide feedback and ratings for stories.

These questions cover a wide range of functionalities and require different levels of problem-solving, creativity, and technical skills, making them suitable for various stages of JavaScript development learning.

"""

Certainly! Below is an example of a testable code snippet for an interactive story builder application. This code includes some intentional mistakes to test the interviewee's understanding of the main concepts like DOM manipulation, data management, and event handling.

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Interactive Story Builder</title>

<style>

body {

font-family: Arial, sans-serif;

}

.story-node {

margin-bottom: 20px;

}

.choice {

margin-left: 20px;

}

</style>

</head>

<body>

<h1>Interactive Story Builder</h1>

<div id="story-container"></div>

<button id="save-story">Save Story</button>

<div id="saved-stories"></div>

<script>

// Initial story data

let storyData = {

id: 1,

content: "Once upon a time...",

choices: []

};

// Function to create a new story node

function createStoryNode(nodeData) {

const node = document.createElement('div');

node.classList.add('story-node');

const content = document.createElement('p');

content.textContent = nodeData.content;

node.appendChild(content);

const addChoiceButton = document.createElement('button');

addChoiceButton.textContent = 'Add Choice';

addChoiceButton.onclick = () => addChoice(nodeData);

node.appendChild(addChoiceButton);

nodeData.choices.forEach(choiceData => {

const choice = createChoiceNode(choiceData);

node.appendChild(choice);

});

return node;

}

// Function to create a new choice node

function createChoiceNode(choiceData) {

const choice = document.createElement('div');

choice.classList.add('choice');

const content = document.createElement('p');

content.textContent = choiceData.content;

choice.appendChild(content);

const addChoiceButton = document.createElement('button');

addChoiceButton.textContent = 'Add Choice';

addChoiceButton.onclick = () => addChoice(choiceData);

choice.appendChild(addChoiceButton);

return choice;

}

// Function to add a choice to a story node

function addChoice(parentData) {

const choiceContent = prompt('Enter choice content:');

if (choiceContent) {

const choiceData = {

id: Date.now(),

content: choiceContent,

choices: []

};

parentData.choices.push(choiceData);

renderStory();

}

}

// Function to render the story

function renderStory() {

const storyContainer = document.getElementById('story-container');

storyContainer.innerHTML = '';

const storyNode = createStoryNode(storyData);

storyContainer.appendChild(storyNode);

}

// Function to save the story

function saveStory() {

const savedStories = document.getElementById('saved-stories');

const storyClone = JSON.parse(JSON.stringify(storyData));

const storyNode = createStoryNode(storyClone);

savedStories.appendChild(storyNode);

}

document.getElementById('save-story').addEventListener('click', saveStory);

// Initial render

renderStory();

</script>

</body>

</html>

```

### Intentional Mistakes:

1. \*\*Story Node ID Conflict\*\*: The `id` attribute for the initial story node is hardcoded as `1`, which could lead to ID conflicts if multiple stories are created.

2. \*\*Event Handler Scope Issue\*\*: The `addChoice` function uses `Date.now()` for unique IDs but doesn't ensure uniqueness in a multi-user environment.

3. \*\*Redundant Button\*\*: The "Add Choice" button appears for each choice, which could be confusing and lead to a poor user experience.

4. \*\*Lack of Data Validation\*\*: There's no validation for the content entered in the prompt.

5. \*\*Save Story Function\*\*: The `saveStory` function does not reset the current story data, causing all saved stories to be appended to the existing story data.

The interviewee will need to identify and correct these issues to make the application functional and robust.

IPrompt16 = f"""

Create a testable code that creates the following application and can be used on coding platforms like hackerrank or coderbyte. Introduce some mistakes to it so that you can test the interviewee with the main concepts mentioned:-

### 27. \*\*Expense Tracker with Visualization\*\*

\*\*Description:\*\* Develop an expense tracker that visualizes spending habits.

\*\*Skills Tested:\*\* Data handling, DOM manipulation, data visualization.

\*\*Features:\*\*

- Add, edit, and delete expenses with categories.

- Display total expenses and breakdown by category.

- Visualize spending habits with charts (e.g., bar chart, pie chart).

"""

Certainly! Below is a JavaScript code snippet that creates a basic expense tracker with visualization capabilities. The code includes intentional mistakes to test the interviewee's understanding of data handling, DOM manipulation, and data visualization. The code is designed to be tested on platforms like HackerRank or Coderbyte.

```html

<!DOCTYPE html>

<html>

<head>

<title>Expense Tracker</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

<h1>Expense Tracker</h1>

<form id="expense-form">

<input type="text" id="description" placeholder="Description" required>

<input type="number" id="amount" placeholder="Amount" required>

<select id="category">

<option value="Food">Food</option>

<option value="Transport">Transport</option>

<option value="Entertainment">Entertainment</option>

</select>

<button type="submit">Add Expense</button>

</form>

<h2>Total Expenses: <span id="total-expenses">0</span></h2>

<h3>Breakdown by Category</h3>

<ul id="expense-list"></ul>

<canvas id="expense-chart" width="400" height="200"></canvas>

<script>

let expenses = [];

let categories = { 'Food': 0, 'Transport': 0, 'Entertainment': 0 };

document.getElementById('expense-form').addEventListener('submit', function(e) {

e.preventDefault();

let description = document.getElementById('description').value;

let amount = parseInt(document.getElementById('amount').value);

let category = document.getElementById('category').value;

expenses.push({ description, amount, category });

categories[category] += amount;

updateExpenses();

updateChart();

});

function updateExpenses() {

let totalExpenses = 0;

let expenseList = document.getElementById('expense-list');

expenseList.innerHTML = '';

expenses.forEach(function(expense) {

totalExpenses += expense.amount;

let li = document.createElement('li');

li.textContent = `${expense.description} - $${expense.amount} (${expense.category})`;

expenseList.appendChild(li);

});

document.getElementById('total-expenses').textContent = totalExpenses;

}

function updateChart() {

let ctx = document.getElementById('expense-chart').getContext('2d');

new Chart(ctx, {

type: 'bar',

data: {

labels: ['Food', 'Transport', 'Entertainment'],

datasets: [{

label: 'Expenses by Category',

data: Object.values(categories),

backgroundColor: ['red', 'blue', 'green'],

}],

},

});

}

// Intentional mistakes:

// - Description should be reset after adding an expense.

// - Amount should be reset after adding an expense.

// - Chart should be updated correctly without overlapping.

</script>

</body>

</html>

```

### Intentional Mistakes:

1. The `description` field is not reset after submitting the form.

2. The `amount` field is not reset after submitting the form.

3. The chart is not properly updated; it creates a new chart each time instead of updating the existing one.

The interviewee should be able to identify and fix these issues:

1. Reset the `description` and `amount` fields after adding an expense.

2. Ensure the chart updates correctly without overlapping previous data.

--------------------------------------------------------------------------------------------------------------