📚 WEB班ワークショップ説明資料案3（穴埋め問題で能動学習版）

🎯 今日のゴール

穴埋め問題を解きながら、タスク管理アプリを段階的に完成させて、CSS装飾とJavaScript動作の実装力を身につける

🧩 穴埋め問題形式について

- ⚡ **ただの写経ではありません！**
- 🧠 **自分で考えて空欄を埋める**ことで理解を深めます
- 🤝 **ペアで話し合い**ながら最適解を見つけます
- 💡 **なぜその答えなのか**を説明できるようになります

🚀 ワークショップの流れ

Step 0: 完成形を体験しよう（10分）

まずは今日作るアプリを実際に触ってみましょう！

やること:

1. 完成形のタスク管理アプリを操作
2. どんな要素・機能があるか観察
3. 「これを作るには何が必要？」を考える

Step 1: HTML穴埋めチャレンジ（30分）

🧩 問題1-1: 基本構造穴埋め（10分）

以下のHTMLの空欄を埋めてください：

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>学習タスク管理</title>
  <link rel="stylesheet" href="_______.css">
</head>
<body>
  <!-- ヘッダー部分 -->
  <_______ class="_______">
    <h1>📚 今日の学習タスク</h1>
    <p>目標: CSS装飾とJavaScript動きをマスターしよう！</p>
  </_______>

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

選択肢:

- div, main, header, style, index, main-header, main-footer

▶ 💡 解答と解説

🧩 問題1-2: 進捗バー構造穴埋め（10分）

```
<!-- 進捗表示 -->
<_______ class="progress-section">
  <h2>📈 学習進捗</h2>
  <div class="progress-container">
    <div class="_______-bar">
      <div class="_______-fill" id="_______Fill"></div>
    </div>
    <p class="_______-text" id="_______Text">0 / 6 完了 (0%)</p>
  </div>
</_______>
```

🧠 考える問題:

1. 進捗表示全体を囲むタグは？（セマンティックHTMLを意識）
2. 進捗バーの外側divのclass名は？
3. 進捗バーの内側divのclass名は？
4. JavaScriptで操作するためのid名は？

ヒント:

- 外側は「枠」、内側は「実際の進捗」

- id名は JavaScript で使いやすい名前

▶ 💡 解答と解説

🧩 問題1-3: タスクリスト穴埋め（10分）

```
<!-- タスクリスト -->
<section class="tasks-section">
  <h2>✅ タスクリスト</h2>

  <div class="_______-item">
    <input type="_______" id="task1" class="_______-checkbox">
    <_______ for="_______">HTML基本構造を作成</label>
  </div>

  <!-- 🎯 チャレンジ: 残り5つのタスクを自分で書いてみよう！ -->

</section>

<!-- 完了メッセージ -->
<div class="completion-message" id="_______Message">
  <h2>🎉 おめでとうございます！</h2>
  <p>全てのタスクが完了しました！</p>
</div>
```

🧠 自分で考える問題:

1. タスク1つ分のclass名は？
2. チェックボックスのtype属性は？
3. チェックボックスとテキストを関連付けるタグは？
4. 完了メッセージのid名は？

🎯 課題: 残り5つのタスクアイテムを自分で作ってみてください！ （内容: "CSSで基本レイアウト", "グラデーション背景を追加", "ホバーエフェクト実装", "JavaScriptで進捗バー作成", "完了エフェクト追加"）

▶ 💡 解答例

Step 2: CSS穴埋めチャレンジ（50分）

🧩 問題2-1: 基本設定穴埋め（10分）

```
/* 基本設定 */
_______ {
```

```
  margin: _______;
  padding: _______;
  box-sizing: _______-box;
}

body {
  font-family: "Hiragino Sans", "Yu Gothic", sans-serif;
  line-height: _____;
  color: #___;
  min-height: _____vh;
  background: _______; /* グレー系の色 */
  padding: _______px;   /* 全体の余白 */
}
```

🤝 ペアで話し合い:

1. 全要素を選択するセレクタは？
2. 要素の外側余白をリセットする値は？
3. `box-sizing` の値は？
4. `line-height` の適切な値は？
5. `color` の値は？（#から始まる）
6. `min-height` の値は？
7. 背景色は何色が良い？（色コードで）
8. 全体の余白は何px？

ヒント:

- リセットCSS の基本
- グレー系: `#f3f4f6`, `#e5e7eb`, `#gray` など

▶ 💡 解答と解説

🧩 問題2-2: カードレイアウト穴埋め（15分）

```
/* カード型レイアウト */
.main-header, _______ {
  background: _______;
  padding: _______rem;
  border-radius: _______px;
  text-align: _______;
  margin-bottom: _______rem;
  box-shadow: _______ _______ _______px rgba(0,0,0,_______);
  border-left: _______px solid #_______;  /* 左端のアクセント線 */
}

/* 進捗バー */
.progress-bar {
```

```
  width: _______;
  height: _______px;
  background: #e5e7eb;
  border-radius: _______px;
  overflow: _______;
}

.progress-fill {
  height: _______;
  background: _______; /* 緑系の色 */
  width: _______%;     /* 初期値 */
  transition: width _______s ease;
}
```

🧠 **考える問題**：

1. headerと同じスタイルを適用するセレクタは？
2. カードの背景色は？
3. ヘッダーの `text-align` の値は？
4. `box-shadow` の構文は `水平 垂直 ぼかし 色` ですが、適切な値は？
5. セクションの左端アクセント線は何px？何色？
6. 進捗バーの初期幅は何%？
7. アニメーション時間は何秒が自然？

💡 **実験してみよう**：値を変更して効果を確認！

▶ 💡 解答例

🧩 問題2-3：ヘッダーと見出しスタイル穴埋め（10分）

```
/* ヘッダー内のスタイル */
.main-header h1 {
  color: #_______;  /* メインカラー */
  font-size: _______rem;
  margin-bottom: _______rem;
}

.main-header p {
  color: #____;     /* 薄いグレー */
  font-size: _______rem;
}

/* セクション見出し */
section h2 {
  color: #_______;  /* メインカラー */
  margin-bottom: _______rem;
  font-size: _______rem;
}
```

```
/* 進捗コンテナ */
.progress-container {
  text-align: _______;
}

.progress-text {
  font-weight: _____;
  font-size: _______rem;
  color: #_______;  /* メインカラー */
}
```

🎯 タイポグラフィチャレンジ：

1. メインカラー（#4f46e5）を適切な場所に配置
2. 文字サイズの階層を考える（h1 > h2 > p）
3. 余白の統一感を出す
4. フォントウェイトの効果的な使用

▶ 💡 解答例

🧩 問題2-4：タスクリストスタイルとホバーエフェクト穴埋め（10分）

```
/* タスクアイテム */
.task-item {
  display: _______;
  align-items: _______;
  padding: 1rem;
  background: #f8fafc;
  border-radius: 8px;
  border: 2px solid #e5e7eb;
  cursor: _______;
  transition: _______ _______s ease;
}

/* 🎯 チャレンジ：ホバーエフェクトを自分で作ろう！ */
.task-item:hover {
  background: _______; /* 少し違う色に */
  border-color: _______; /* アクセントカラーに */
  transform: translateY(_______px); /* 上に移動 */
  box-shadow: _______ _______ _______px rgba(79, 70, 229, 0.2);
}
```

🎯 ホバーエフェクト完全自作チャレンジ：

1. `display` と `align-items` でどんなレイアウト？
2. マウスカーソルの形は？
3. ホバー時の背景色は？（少し変化させる）
4. 何px上に移動させる？

5. 影の効果は？

🤝 **ペアで実験**：値を変更して一番良い効果を見つけよう！

▶ 💡 解答例

🧩 問題2-5：タスクリスト詳細スタイル穴埋め（10分）

```
/* チェックボックス */
.task-checkbox {
  width: ____px;
  height: ____px;
  margin-right: ____rem;
  cursor: ______;
}

/* ラベルスタイル */
.task-item label {
  flex: _;           /* 残りスペースを使用 */
  cursor: ______;
  font-size: ____rem;
  color: #___;      /* 通常の文字色 */
}

/* 完了状態のスタイル */
.task-item.completed {
  background: #_______;  /* 薄い緑 */
  border-color: #_______;  /* 緑 */
}

.task-item.completed label {
  text-decoration: _______-_____;  /* 取り消し線 */
  color: #_______;  /* グレー */
}
```

🧠 **考える問題**：

1. チェックボックスのサイズは何px？
2. flexプロパティで残りスペースを使うには？
3. 完了時の背景色・ボーダー色は？
4. 取り消し線のCSSプロパティは？

▶ 💡 解答例

🧩 問題2-6：完了メッセージスタイル穴埋め（10分）

```
/* 完了メッセージ（重要！） */
.completion-message {
  position: _______;     /* 画面に固定 */
```

```
  top: ___%;            /* 縦中央 */
  left: ___%;           /* 横中央 */
  transform: translate(-50%, -50%);  /* 完全中央配置 */
  background: #_______;  /* 緑 */
  color: _____;         /* 白 */
  padding: ____rem;
  border-radius: ____px;
  text-align: _______;
  display: ____;        /* 初期状態は非表示 */
  box-shadow: _ _px ____px rgba(0, 0, 0, 0.2);
  z-index: ____;        /* 最前面に表示 */
}

.completion-message h2 {
  color: _____;         /* 白 */
  margin-bottom: ____rem;
}
```

🎯 モーダル実装チャレンジ：

1. `position` で画面固定にするには？
2. 完全中央配置の仕組みは？
3. 初期状態で非表示にするには？
4. 最前面に表示するプロパティは？

▶ 💡 解答例

🧩 問題2-7：レスポンシブ対応穴埋め（5分）

```
/* レスポンシブ対応 */
@media (max-width: ____px) {
  body {
    padding: ____px;  /* モバイルでは狭く */
  }

  .main-header,
  section {
    padding: ____rem;  /* 内側余白も調整 */
  }

  .main-header h1 {
    font-size: ____rem;  /* 文字サイズ縮小 */
  }

  .task-item {
    padding: ____rem;   /* タスクアイテムも調整 */
  }
}
```

🧠 **考える問題**：

1. モバイル判定のブレイクポイントは？
2. 各要素の余白をどう調整する？

▶ 💡 解答例

Step 3: JavaScript穴埋めチャレンジ（25分）

🧩 問題3-1: 要素取得穴埋め（5分）

```
document.addEventListener("_______", function () {
  // 要素を取得
  const taskCheckboxes = document._______(".______-checkbox");
  const progressFill = document._______("______Fill");
  const progressText = document._______("______Text");
  const completionMessage = document._______("______Message");
});
```

🧠 **考える問題**：

1. DOMが読み込まれた時のイベント名は？
2. 複数要素を取得するメソッドは？
3. ID で1つの要素を取得するメソッドは？
4. class名とid名の正確な名前は？

▶ 💡 解答

🧩 問題3-2: イベント処理ロジック穴埋め（10分）

```
// 各チェックボックスにイベントを追加
taskCheckboxes.forEach(function(checkbox) {
  checkbox.addEventListener("change", function() {
    handleTaskChange(this);
  });
});
function handleTaskChange(checkbox) () {
  const taskItem = checkbox._______(".task-item");

  if (checkbox._______) {
    taskItem.classList._______("completed");
  } else {
    taskItem.classList._______("completed");
  }
```

```
  updateProgress(); // 進捗更新関数を呼び出し
}
```

🎯 ロジック理解チャレンジ:

1. 配列の各要素に処理を実行するメソッドは？
2. チェック状態変化を検知するイベントは？
3. 親要素を取得するメソッドは？
4. チェック状態を確認するプロパティは？
5. クラスを追加/削除するメソッドは？

▶ 💡 解答

🧩 問題3-3: 進捗計算完全自作チャレンジ（10分）

```
function updateProgress() {
  // 🎯 以下を自分で実装してみよう！

  // 1. 全タスク数を取得（ヒント: taskCheckboxes.length）
  const totalTasks = _______;

  // 2. 完了したタスク数を取得（ヒント: :checked セレクタ）
  const completedTasks = document.querySelectorAll("_______").length;

  // 3. パーセンテージを計算（四捨五入）
  const percentage = Math.round((_______ / _______) * 100);

  // 4. 進捗バーの幅を更新
  progressFill.style._______ = percentage + "%";

  // 5. 進捗テキストを更新
  progressText._______ = `${_______} / ${_______} 完了 (${_______}%)`;

  // 6. 全完了時の判定
  if (_______ === _______) {
    completionMessage.style.display = "_______";
  } else {
    completionMessage.style.display = "_______";
  }
}
```

🧠 完全自作問題: この関数を自分で完成させてください！

考えるポイント:

1. 全タスク数の取得方法
2. 完了タスクの条件（:checked）
3. パーセンテージ計算式

4. DOM要素のプロパティ更新
5. 条件分岐の判定

🤝 **ペア作業**: 一緒に考えて実装しよう！

▶ 💡 解答例

🎯 問題4-1: 機能拡張チャレンジ（10分）

基本機能が完成したら、以下の機能を追加してみよう！

🎯 **チャレンジミッション**（難易度別）:

初級: 完了メッセージをクリックで閉じる機能

```
completionMessage.addEventListener("_______", function() {
  // ここに処理を書こう
});
```

中級: タスク完了時に背景色を変える機能

```
/* 問題2-5で学習済み！ */
.task-item.completed {
  background: #ecfdf5; /* 薄い緑 */
  border-color: #10b981; /* 緑 */
}

.task-item.completed label {
  text-decoration: line-through; /* 取り消し線 */
  color: #6b7280; /* グレー */
}
```

上級: 進捗バーの色をパーセンテージで変える機能

```
// ヒント: パーセンテージに応じて色を変更
if (percentage < 50) {
  progressFill.style.background = "_______"; // 赤系
} else if (percentage < 100) {
  progressFill.style.background = "_______"; // 黄系
} else {
  progressFill.style.background = "_______"; // 緑系
}
```

🤝 **ペアで挑戦**：どのレベルまでできるか試してみよう！

🎉 穴埋め完了！おめでとうございます！

🧠 学習効果チェック

🤝 **ペアで話し合い**：

1. 一番難しかった穴埋めは？その理由は？
2. 一番「なるほど！」と思った部分は？
3. 写経と穴埋め、どちらが理解しやすかった？

🎯 できるようになったこと

- ✅ HTMLの構造を**理由と共に**設計
- ✅ CSSプロパティの**意味を理解**して記述
- ✅ JavaScriptロジックを**自分で構築**
- ✅ バグを**発見・修正**する能力
- ✅ 機能を**拡張**する思考力

🆘 穴埋めサポートシステム

段階別ヒント提供

レベル1：方向性ヒント

- 「○○について考えてみてください」
- 「完成形と比べて何が必要ですか？」

レベル2：選択肢ヒント

- 「A, B, C のどれが適切でしょうか？」
- 「○○という方法はどうでしょう？」

レベル3: 直接ヒント

- 「この部分は〇〇を使います」
- 「構文は『〇〇: △△;』です」

穴埋め特有のサポート

考える時間の確保

- ⏰ **制限時間**: 各問題に適切な時間設定
- 🤝 **ペア相談**: わからない時は相談OK
- 💡 **ヒント要求**: 困ったら手を挙げる

答え合わせプロセス

1. **自分の答えを発表**
2. **理由を説明**
3. **正解と比較**
4. **なぜその答えなのか理解**

よくある間違いパターン

❌ **「書けばいいと思った」** ✅ **「この理由でこの答えにした」**

❌ **「なんとなく選んだ」** ✅ **「〇〇の効果を狙ってこれを選んだ」**

穴埋め効果の検証

写経との違い:

- 📝 写経: コードを見て書く → 手の記憶
- 🧩 穴埋め: 理由を考えて書く → 頭の理解

能動性の向上:

- 🤔 「なぜこの値？」を常に考える
- 💡 「他の選択肢はダメ？」を検討する
- 🎯 「どんな効果を狙う？」を明確化

Think, Fill, Code! 🧩 穴埋めで理解を深めて、本当の実装力を身につけよう！