令和３年度　卒業論文

プログラマー向けタイピングアプリの制作

近畿大学工学部　情報学科

情報物理研究室

1810990059　和田尚大

研究成果の概要（和文）：
プログラミングを学び始めた人の問題の 1 つに，慣れない英文のタイピングによるタイピングミスから引き起こされるプログラムエラーがある．そこで，本研究では，初心者の人たちのタイピングミスを減らすべく，問題文がプログラミングで使用する英文となったタイピングアプリを開発し，様々な種類のユーザでテストを行った．本研究で作成したアプリケーションで，どのようなタイプのユーザにより効果が得られるかを分析した結果，プログラミングを経験したが，ここ最近は特にプログラミングを行っていないというような経験者という枠のユーザに特に効果が見られ，復習の為の教材としての成果が得られた．また，プログラミング初心者のタイピングミスを減らすという点も，効果が得られた．

研究成果の概要（英文）：
One of the problems for those who have just started learning programming is programming errors caused by typing mistakes caused by unfamiliar English typing. Therefore, in this research, in order to reduce typing mistakes for beginners, we developed a typing application in which the problem sentences are English sentences used in programming, and tested them with various types of users. As a result of analyzing what type of user can be effective with the application created in this research, I experienced programming, but recently I have experienced users who are not programming in particular. Especially effective was seen, and the result as a teaching material for review was obtained. It was also effective in reducing typing mistakes for programming beginners.

1 はじめに

学習指導要領の改訂により，高等学校の情報教科の導入をはじめとして，初等教育，中等教育においても本格的な情報の教育が開始された．その中で，特に近年注視されているのが「プログラミング」であり，初等教育でのプログラミングの導入に加え，ビジネスや副業としてもプログラミングは活用されており，このスキルを習得しようと考えている者も増えている．そのプログラミング学習の最初の壁として，言語の学習と共に，タイピングミスによるプログラムのエラーがある．その割合は非常に多く，情報学科出身である私の周りの人々に聞いたところ，プログラムを書いたことがあるものでタイピングミスによるプログラムエラーの経験をしたことがない人は，1 人もいなかった．タイプミスが多い理由として，慣れない英語や英文のタイピングをしなければならないというものが挙げられる．

しかし，その対策方法は特になく，既存のタイピングアプリケーションもローマ字入力のものが大半を占めており，少数ある英文字入力のタイピングアプリケーションも簡単な英文の入力に慣れる為のものであり，プログラミングに特化したものは無い．

そこで，本研究では，上記のプログラミング初心者の課題の解決の為のタイピングのアプリケーションを開発した．このアプリケーションは，プログラミング初心者のタイピングミスを減らすこと，プログラミング学習のサポートをすることを目的とし，タイピングアプリケーションの問題文を学習する言語でよく使用される英単語や英文であるという特徴がある．また，出題された問題の解説をアプリケーション終了後に表示するといった言語学習支援機能を備えている．本論文では，2 章でアプリケーションの仕様，3 章でアプリケーションの設計，4 章でユーザテストの内容とその結果からアプリケーションの評価を行う．

2 アプリケーションの仕様

研究の目的として，自身がプログラミング学習を行った時，タイピングミスによるプログラムエラーに何度も苦しめられたという背景がある．プログラミングの授業を見ていると，私と同じような経験をしている人も非常に多く，指導教員に話を聞くと，最初はほとんどタイピングミスがエラーの原因になっていて，タイピングミスによるエラーは，エラー箇所を探すのが大変だから，みんな苦しめられていると仰っていた．

そこで私は，なぜプログラミング初心者はタイピングミスによるエラーを多発してしまうのかという原因について考えてみた．考え付いた原因として，「プログラミング初心者はパソコンの操作に慣れていない人が多い為，タイピングのミスが多いこと」や「慣れない英語のタイピングが多い」，「使用しているタイピング学習ツールのユーザインタフェースが良くない」がある．このうち，3 つ目の学習ツールのユーザインタフェースについては，私自身でどうにか出来るものではないと考え，原因の 1 つ目と 2 つ目に注目した．

まず，原因の 1 つ目である「プログラミング初心者はパソコン操作に慣れていない」という点に置いて，実際にプログラミング経験が有る人，無い人計 15 人にタイピングテストを実施した．その結果が，以下の図 1 である．ここで使用したタイピングアプリケーションは E-typing で，そのスコアを図で表したものである．[1]

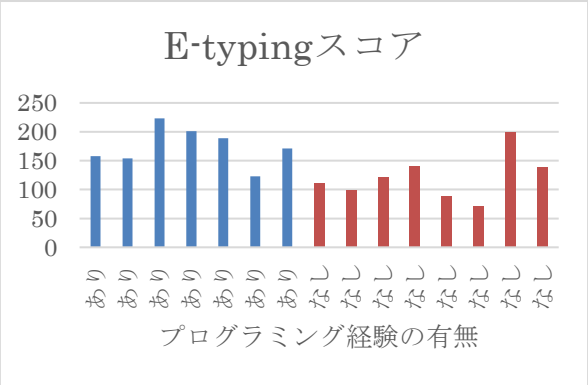


図 1 プログラム経験によるタイピング力の差

図から読み取れる通り，プログラミング経験者(図 1 左半分)のスコアが高く，プログラミング未経験者(図 2 右半分)のスコアが低いことが分かった．スコアの平均値を出すと，経験者組が 174．1，経験無し組が 121．1 とおよそ 50 程度の差があり，E-typing のランクが 2 から 3 程変わる大きな差があることが分かった．

次に，原因の 2 つ目である「慣れない英語のタイピングでミスが増えている」という点についても，実際にテストを行った．テスト方法は，1 つ目のテストと同様に，E-typing というアプリケーションを使用し，15 人にローマ字タイピングと英語タイピングの両方を行ってもらい，その結果を評価するというものである．実際のテスト結果を以下の図 2 に示した．

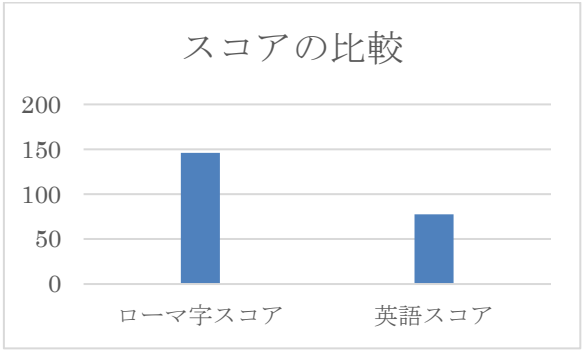


図 2 ローマ字と英語のスコア比較

このテストも，私の予想通り英語のタイピングに慣れていない人が多いという結果が出た．スコアの差は歴然としており，ローマ字入力のスコア平均が 146，英語入力のスコア平均が 77．7 とローマ字入力の半分ほどのスコアになるという結果になった．しかし，プログラミング経験者のスコアも初心者と同様に英語タイピングの結果がローマ字タイピングの半分になっており，プログラミングの経験と英文のタイピング力は必ずしも直結するわけではないということが分かった．

これらのテスト結果から，プログラミング初心者に足りないものとして，「タイピング力」と「コード入力の慣れ」があると考えられる．そこでこの 2 点を補うべく本研究で，問題文がプログラミングでよく使用する英単語や英文となったタイピングアプリケーションを開発した．次に，このアプリケーションの開発環境について説明する．

本アプリケーションは，HTML，CSS，JavaScript を用いて開発した．開発に使用したツールは Visual Studio Code を用い，私の研究室で徐教授が借りている heteml というレンタルサーバーに自身が書いたコードをアップロードするという形で作成した．アプリケーションにこれらの言語を使用した理由は，他の良く使用される寿司打やマナビジョンタイピングなどのタイピングアプリケーションが，全て Web ブラウザ上で動作するアプリケーションであり，タイピングアプリケーションは Web ブラウザ上で行うものという一般の認識が高いと考え，それらに適したプログラミング言語は何かを考えた結果である．次に，アプリケーションの仕様について説明する．

実際に制作したアプリケーションは、以下のリンクから使用できる。
https://buturi.heteml.net/student/2021/wada/kenkyunaiyo/sotsuken.html

アプリケーションの仕様は，実際のプレイ画面の図を用いて説明する．下の図 3 はタイピングアプリケーションの初期画面である．この画面のスタートボタンをクリックしてアプリケーションをスタートさせる．スタートボタンをクリックすると，図 4 のアプリケーションの設定画面に移る．この設定画面では，プレイするアプリケーションの設定を行う．図 4 の 1 つ目の言語選択は，出題される問題の言語を設定する．本アプリケーションは，C 言語と JavaScript の問題を用意した為，いずれかを選択する．図 4 の 2 つ目の難易度選択では，問題の難易度を選択する．「かんたん」は，それぞれの言語の英単語が問題となる．「ふつう」は，「かんたん」で問題となった英単語を実際にコード入力する例が問題となっている．「むずかしい」は，実際に 1 つのコードが問題となっており，最初から最後まで全てを入力する仕様となっている．図 4 の 3 つ目のプレイ時間は，アプリケーションのプレイ時間を設定するもので，3 分，2 分，1 分のいずれかを選択できる．設定を終えると，「タイピングを開始」ボタンを入力することで，ゲームがスタートする．図 4 の開始ボタンをクリックされた後の画面が，図 5 である．画面の指示通り，Enter キーか Space キーを入力することで，3，2，1 とカウントダウンが始まり，アプリケーションがスタートする．

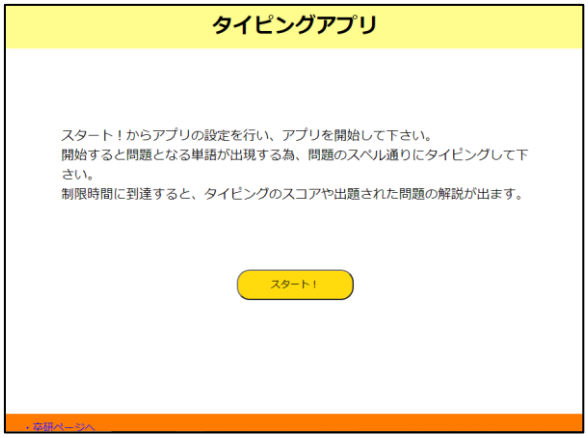


図 3 スタート画面

図 4 設定画面

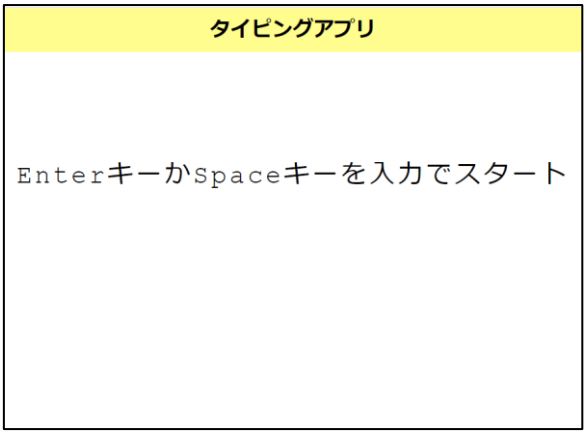


図 5 開始直前画面

タイピングアプリケーションが始まった画面が図 6 である．画面に出題された問題を入力していくことでアプリケーションが進んでいく．適切に文字が入力されていくと，図 7 のように，入力された文字の色が薄くなる．間違えて文字を入力すると，図 8 のように画面上に miss という文字と音での通知が設定されている．また，プレイ中，画面の左上には適切に入力された文字数，右上には，図 4 の設定画面で設定したプレイ時間の内，残り時間が常時表示されている．

図 6 プレイ画面 1

図 7 プレイ画面 2

図 8 ミス画面

プレイ時間が終わると，図 9，10 のようにプレイの結果画面が表示される．この画面では，今回のプレイの正タイプ率，ミスタイプ率とその割合，そこから導き出したスコアが表示される．また，出題された問題のプログラミング言語としての意味の解説が表示され，そこでプログラミングの学習も進めることができるといった仕様となっている．

次に，図 10 の「間違えた問題を復習」ボタンをクリックすると，図 5 と同様の画面になり，図 12 のように今回のプレイでタイピングミスが 1 回でもあった問題が再度出題される．間違えた問題が 1 問も無い場合は，図 11 のように終了画面が表示される．

復習問題をプレイ中は，特に制限時間やタイピングの正確さを求めない為，プレイ中の図 6 と異なり，図 12 ではそれらの表示はしていない．また，図 8 のようにタイピングミスの際も，文字や音での通知は行っていない．

復習問題を終えると，図 13 のように復習問題終了画面が表示され，アプリケーションが終了される．これらが，本研究で開発したタイピングアプリケーションの一連の流れである．

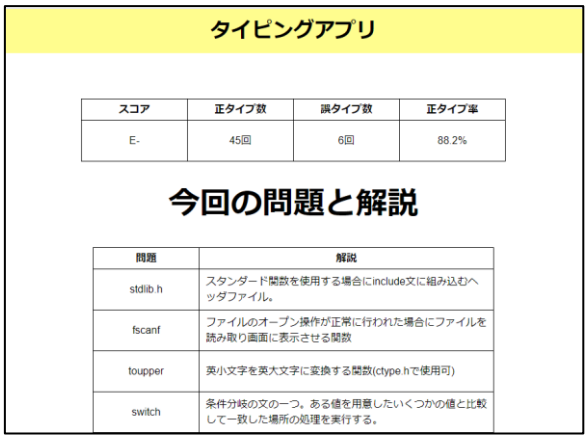


図 9 結果画面 1

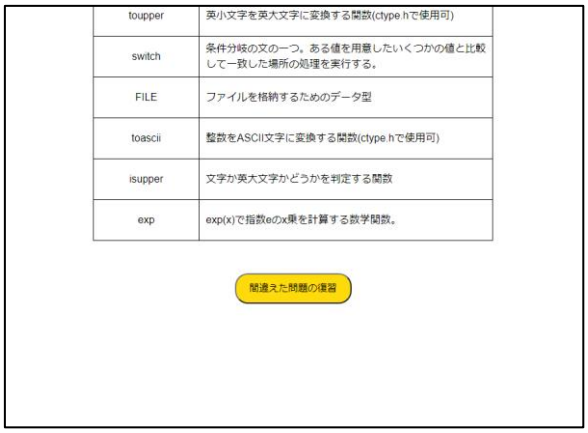


図 10 結果画面 2

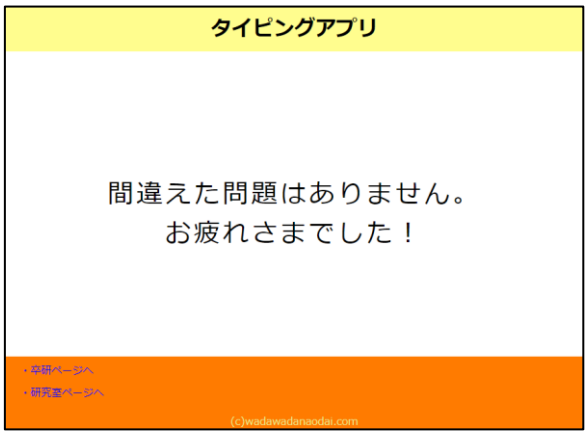


図 11 終了画面

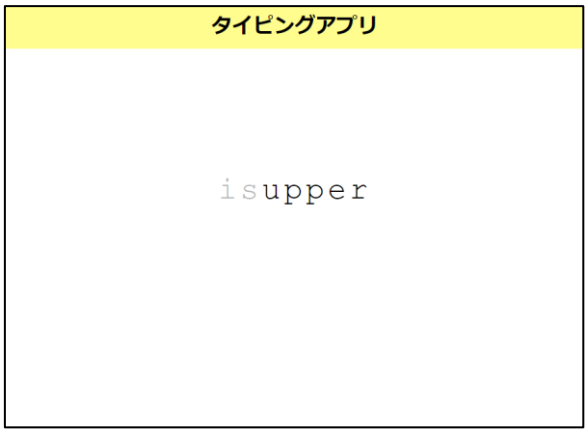


図 12 復習問題プレイ画面

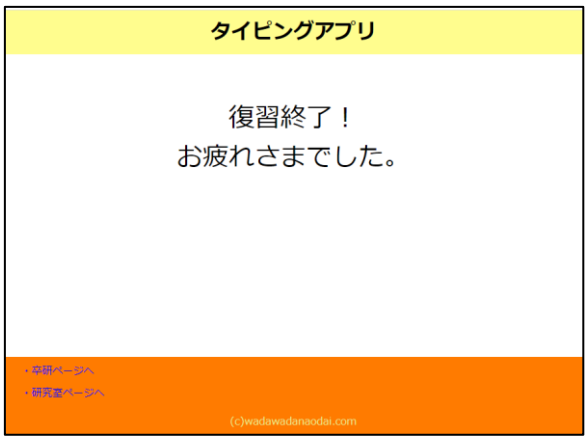


図 13 復習問題終了画面

このアプリケーション作成にあたり，研究室のメンバーや教授と多くの議論を行った．その議論で大きく変更した点は 3 つある．

1 つ目は，入力された文字の表示方法である．完成版では，図 7 のように入力された文字が透過されて表示されているが，議論前に独自で作成していた時は，図 14 のように，入力された文字を完全に消し，アンダーバーに変換することで入力したことが分かるようにしていた．私が図 14 のように作成した理由は，このタイピングアプリケーションに独自性を付け加えようと考えた為である．タイピングアプリケーションを作成するにあたり，複数のアプリケーションをプレイしてきたが，入力された文字の処理方法はアプリにより様々で，文字を透過させる形や入力された文字から表示していくものなどがあった．しかし，図 14 のような表示方法をしたアプリケーションは，私が調べた範囲では無かった．そこで私は，他のアプリケーションにはない特徴を付けるという目的で，図 14 のような入力された文字を消し，アンダーバーに置き換えて表示するという風に作成した．しかし，このアプリケーションの議論を進めていく中で，文字を消すのではなく透過させることで，単語としての認識が高まりより知識として定着していくのではないかという意見が挙がり，その意見に非常に共感出来た為，完成版は図 7 のように透過させて表示するという仕様にした．

2 つ目は，スクロールバーについてである．図で示しているアプリケーションは，画像を切り取っている為，画面サイズは認識できないが，このアプリケーションは，スクロールバーがほとんど表示されない形となっている．これも，議論によって，当初の形から変更となった点である．当初はアプリケーションの画面のサイズは特に気にしておらず，非常に縦長のアプリケーションで，スクロール先にも空白の部分があるという状態であった．しかし，この点も研究室でプレイしてもらった時に，スクロールバーが表示される，つまり余白を作ると，その先に何かコンテンツがあるのではとユーザに期待感を抱かせてしまうのではないかという指摘があった．私が他のタイピングアプリケーションをプレイした時は，特にスクロールバーについて気にしていな

かった為，この指摘はあまり刺さらなかった．そこで，私の周りの人 4 人に実際に簡単なテストを行った．テスト内容は，無駄に余白があるアプリ，常に画面下にフッターを固定したアプリ，無駄な余白を消し，スクロール先をフッターのみに調整したアプリの 3 つをテストユーザにプレイしてもらい，スクロールバーについて気にするようなしぐさがあるかを確認するものである．結果として，4 人中 3 人は特に気にしなかったものの，1 人はスクロールバーについて気にするようなしぐさを見せた．そこで，スクロールバーでコンテンツを期待する人もいるということが分かり，完成版の形通り，スクロールバーを出来るだけ無くし，スクロールバーが表示される時は，その先にコンテンツを用意するという表示方法に変更した．

3 つ目は，コンテンツ量についてだ．開発当初は，問題の量は，それぞれの単語で 1 問ずつとしていた．その理由は，言語の学習を幅広く行ってもらう事と全ての入力に慣れてもらう為である．しかし，研究室メンバーに制限時間 3 分でプレイしてもらった時，時間内に全ての問題を終わらせてしまう人がいた．そこで，私の全ての問題をプレイしてもらいたいという考えとコンテンツ量を増やすという問題を加味した結果，よく使用する英単語や英文の出題数を増やすのはどうかという意見が挙がり，それを採用した．研究室のメンバー内で，自分たちがその言語を学んだとき，特によく使用した英文，英単語をピックアップし，その問題の出題数を増やすという方法でコンテンツ量の少なさの問題を解決した．

図 14 修正前画像

3 アプリケーションの設計

ここからは，このタイピングアプリケーションの設計について説明する．アプリケーションの概要，処理の流れは右の図 14 のフローチャート通りである．このフローチャートを基に，アプリケーションを作成していった．このフローチャートとソースコードを基に，アプリケーションのプログラムについて説明していく．

まず，フローチャートの上部，アプリケーションのスタート画面について説明する．HTML のソースコードは図 16，CSS のソースコードは図 18，JavaScript のソースコードは図 17 で示しており，これらが図 3 のスタート画面へとつながる．図 16 を見ると，アプリケーションの説明をセクションタグで囲っている事が分かる．その理由は，スタート画面，設定画面，プレイ画面，結果画面という流れで画面が切り替わるアプリケーションにしたため，それらを場面ごとにセクションタグで区切り，ID を振ることで，JavaScript で表示非表示をコントロールするためである．実際に，図 17 の JavaScript ソースコードは，スタートボタンをクリックした時のイベントを設定したモノであるが，expDreat という関数を用いて，スタート画面の表示を非表示に切り替え，106 行目で設定画面を表示させる命令をしていることが分かる．

図 18 は，アプリケーションの画面スタイルについてのソースコードである．画面の色合いは，他のタイピングアプリケーションには無い色合いを採用したく，ヘッダーを黄色，フッターをオレンジとした．また，ヘッダーは常に表示される要素なので，薄めの黄色に，フッターは，スクロール先か最後の画面のみで表示される為，目立つように明るめのオレンジを採用した．

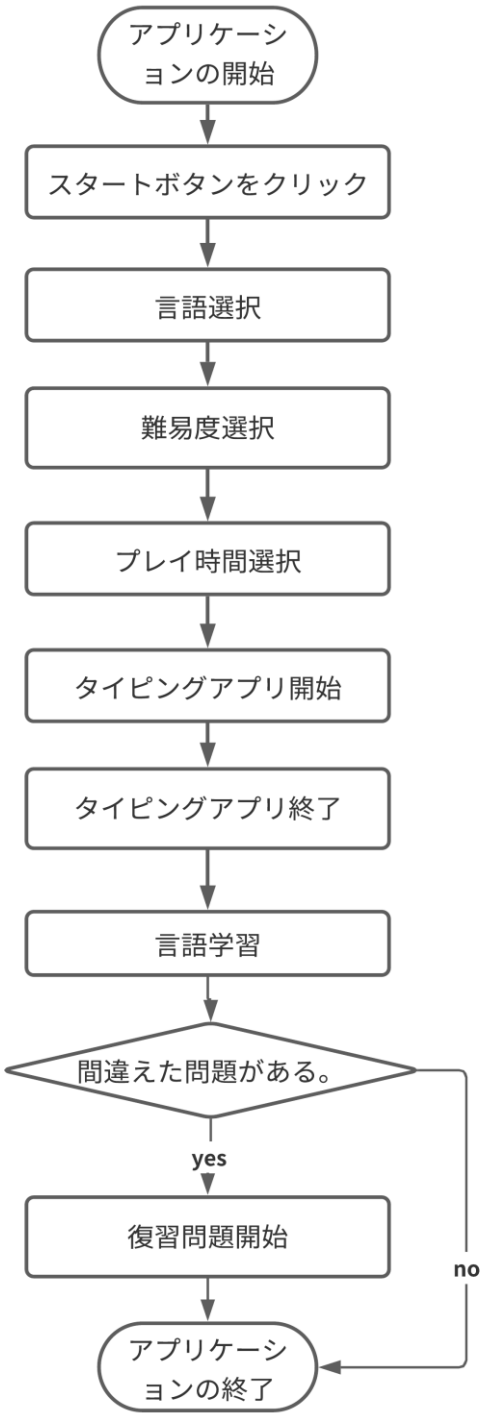


図 15 アプリのフローチャート

```
<body id="body">
    <header id="header">
        <h1>タイピングアプリ</h1>
    </header>

    <section id="explan">
        <div>スタート！からアプリの設定を行い、アプリを開始して下さい。</div>
        <div>開始すると問題となる単語が出現する為、問題のスペル通りにタイピングして下さい。</div>
        <div>制限時間に到達すると、タイピングのスコアや出題された問題の解説が出ます。</div>
    </section>
    <button id="startbtn">スタート！</button>


    <footer id="footer">
        <li><a href="../index.html" target="_blank">・卒研ページへ</a></li>
        <li><a href="https://buturi.hteml.net/student/2021/" target="_blank">・研究室ページへ</a></li>
        <p>(c)wadawadanaodai.com</p>
    </footer>
</body>
```

図 16 HTML ソースコード　スタート画面

```
    startbtn.addEventListener('click', () => {
        expDreat();
        settei.style.display = "block";
        const header = document.getElementById('header'); //ヘッダーのスタイル調整
        footer.style.marginTop = "0px";
        header.style.marginBottom = "35px";
    }); //スタート画面のあれこれ
```

図 17 JS ソースコード　スタート画面

```css
html {
  background: rgb(216, 213, 213);
  user-select: none;
}

* {
  margin-top: 0;
  margin-bottom: 0;
}

header {
  background: rgb(255, 253, 142);
  text-align: center;
  padding-top: 10px;
  padding-bottom: 10px;
  margin-bottom: 100px;
}

html,
body {
  width: 100%;
  margin: 0;
}

body {
  text-align: center;
  width: 900px;
  margin-left: auto;
  margin-right: auto;
  background: white;
  word-wrap: break-word;
}

button {
  background-color: rgb(255, 219, 14);
  border-radius: 20px;
  padding: 10px;
  width: 20%;
  font-size: 15px;
}

/* フッタースタイル */

footer {
  background: rgb(255, 123, 0);
  margin-top: 55px;
}

footer>li {
  padding-top: 10px;
  margin-left: 30px;
  list-style: none;
  text-align: initial;
}

footer>li>a {
  text-decoration: none;
  color: blue;
}

footer>li>a:hover {
  color: red;
}

footer>p {
  padding-top: 20px;
  padding-bottom: 10px;
  text-align: center;
  color: rgb(236, 236, 155);
}
```

図 18 CSS ソースコード　スタート画面

設定画面のソースコードは，図 19，図 20，図 21，図 22 であり，実際に表示される画面は図 4 である．図 4，図 19 の通り，言語設定，難易度設定はラジオボタンを，プレイ時間の設定はスライダーを用いた．理由は，選択する範囲の広さやユーザがより慣れているモノにするという点と，数字はスライド形式にすることで分かりやすくなるという点である．また，図 19 の 25~26，30~32 行目の通り，入力された言語，難易度に合わせて 0~2 数字を割り当てている．この数字を図 20 の JavaScript で文字列から整数に変え，図 21 で選択された言語と難易度を数字から判別し，それに対応した問題を出題するための変数 words に格納している．また，wordcheck，wordscome はそれぞれ，問題の正誤を判別する配列，問題の解説を格納する配列であり，この 2 つも同じタイミングで行っている．

問題の設定を終えると，タイピングを開始のボタンをクリックすることで，図 5 の開始直前画面に移る．この部分のプログラムが，図 22 である．ここで，先ほど記載した設定の読み取りが行われる．そして，エンターキーもしくはスペースキーが入力されると，アプリケーションが開始される．その処理を記載したのが，図 23 である．428 行目の条件分岐文でキーの入力判別を行い，その後，count という変数を用いて，開始前のカウントダウンを行っている．カウントダウンが終了すると，関数 playcount でプレイ時間の管理，関数 setWord で問題を用意，関数 playtyu でプレイ中の処理を行う．これらの関数内の処理は，後述する．

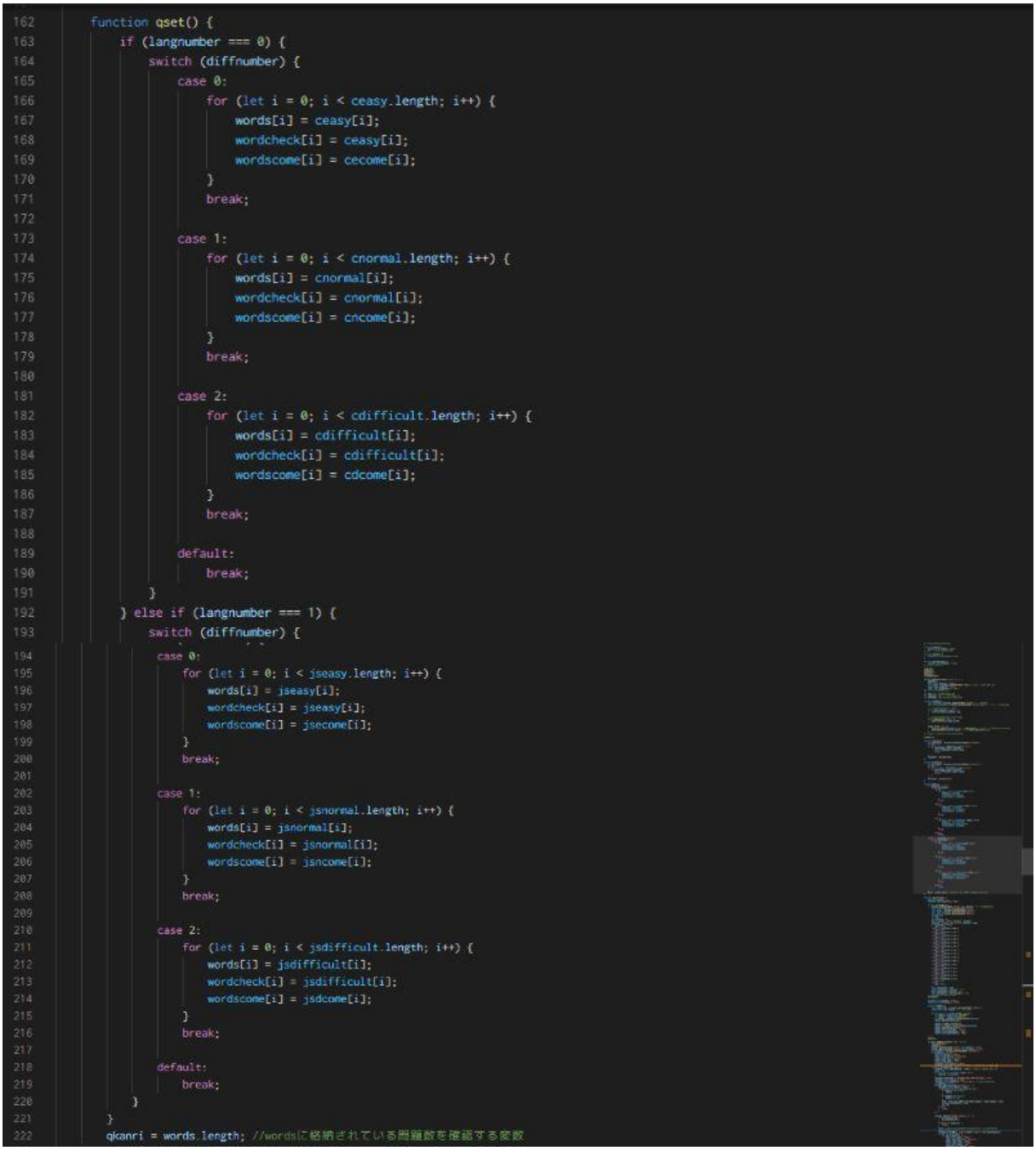
```html
22  <section id="settei">
23      <div id="a">
24          <div id="languagechoice">言語選択　</div>
25          <label><input type="radio" name="language" value="0" checked="checked">C　　　　 </label>
26          <label><input type="radio" name="language" value="1">JavaScript</label>
27      </div>
28      <div id="b">
29          <div id="difficultchoice">難易度選択</div>
30          <label><input type="radio" name="difficult" value="0" checked="checked">かんたん　</label>
31          <label><input type="radio" name="difficult" value="1">ふつう　　</label>
32          <label><input type="radio" name="difficult" value="2">むずかしい</label>
33      </div>
34      <div id="c">
35          <div id="playtime">プレイ時間</div>
36          <input type="range" id="example" list="tlist" min="1" max="3" step="1" value="3">
37          <datalist id="tlist">
38              <option value="1">
39              <option value="2">
40              <option value="3">
41          </datalist>
42          <div>　　<a id="current-value"></a>分</div>
43      </div>
44      <div id="btnposi">
45          <button id="checkButton">タイピングを開始</button>
46      </div>
47  </section>
```

図 19 HTML ソースコード　設定画面

```javascript
function langcheck() {
    let langelements = document.getElementsByName("language");
    let val = "";
    for (let i = 0; i < langelements.length; i++) {
        if (langelements.item(i).checked) {
            val = langelements.item(i).value;
            break;
        }
    }
    langnumber = parseInt(val);
}

function diffcheck() {
    let diffelements = document.getElementsByName("difficult");
    let val = "";
    for (let i = 0; i < diffelements.length; i++) {
        if (diffelements.item(i).checked) {
            val = diffelements.item(i).value;
            break;
        }
    }
    diffnumber = parseInt(val);
}
```

図 20 JS ソースコード　設定画面 1



```
function qset() {
    if (langnumber === 0) {
        switch (diffnumber) {
            case 0:
                for (let i = 0; i < ceasy.length; i++) {
                    words[i] = ceasy[i];
                    wordcheck[i] = ceasy[i];
                    wordscome[i] = cecome[i];
                }
                break;

            case 1:
                for (let i = 0; i < cnormal.length; i++) {
                    words[i] = cnormal[i];
                    wordcheck[i] = cnormal[i];
                    wordscome[i] = cncome[i];
                }
                break;

            case 2:
                for (let i = 0; i < cdifficult.length; i++) {
                    words[i] = cdifficult[i];
                    wordcheck[i] = cdifficult[i];
                    wordscome[i] = cdcome[i];
                }
                break;

            default:
                break;
        }
    } else if (langnumber === 1) {
        switch (diffnumber) {
            case 0:
                for (let i = 0; i < jseasy.length; i++) {
                    words[i] = jseasy[i];
                    wordcheck[i] = jseasy[i];
                    wordscome[i] = jsecone[i];
                }
                break;

            case 1:
                for (let i = 0; i < jsnormal.length; i++) {
                    words[i] = jsnormal[i];
                    wordcheck[i] = jsnormal[i];
                    wordscome[i] = jsncome[i];
                }
                break;

            case 2:
                for (let i = 0; i < jsdifficult.length; i++) {
                    words[i] = jsdifficult[i];
                    wordcheck[i] = jsdifficult[i];
                    wordscome[i] = jsdcome[i];
                }
                break;

            default:
                break;
        }
    }
    qkanri = words.length; //wordsに格納されている問題数を確認する変数
```

図 21 JS ソースコード　設定画面 2

```
checkButton.addEventListener('click', () => { //ゲーム設定画面のあれこれ
    time = parseInt(elementrange.value); //文字列を整数に変換
    langcheck();
    diffcheck();
    setDreat();
    qset();
    const footer = document.getElementById('footer'); //フッターのスタイル調整
    const timebtnstyle = document.getElementById('playnow')
    header.style.marginBottom = "100px";
    footer.style.marginTop = '285px';
    target.style.display = "block";
    target.style.fontFamily = 'Courier New'; //問題文のフォント変更
    target.style.fontSize = "40px"; //プレイ前スタート画面のフォントサイズ変更

    var playtime = time * 60; //ゲームプレイ時間
    function playcount() {
        pldreat.style.display = "block";
        pldreat.style.display = "flex";
        const timetext = document.getElementById('time');
        var timer, timer2;
        var countup = function() {
            playtime--;
            var kei = 60;
            var minute = playtime / kei;
            var second = playtime % kei;
            timetext.textContent = Math.floor(minute) + "分" + ("00" + second).slice(-2) + "秒";
            if (playtime === 0) {
                timetext.textContent = "終了!";
                clearInterval(timer);
                target.textContent = "終了!";
                footer.style.marginTop = '200px';
                isPlaying = false; //プレイ終了の命令
                    timer2 = setTimeout(() => {
                        tarDreat();
                        playDreat();
                        resultscreen();
                    }, 1000);
                }
            }
            timer = setInterval(countup, 1000);
        }
```

図 22 JS ソースコード　設定画面 3

```
    document.addEventListener('keydown', e => {
        if (e.keyCode === 13 || e.keyCode === 32) {
            if (countstart === true) {
                return;
            }
            if (countstart === false) {
                countstart = true;
            } //Enter Space の複数回入力の拒否(このやり方をフラグを立てるという)
            footer.style.marginTop = '340px';
            target.innerHTML = ""; //tarDreat()を使用すると問題文事消えてしまう為、要素を消す。
            var count = 3;
            var id = setInterval(function() {
                footer.style.marginTop = '270px';
                document.querySelector('#Stime').textContent = count;
                count--;
                if (count <= -1) {
                    header.style.marginBottom = "10px";
                    timebtnstyle.style.marginBottom = "90px";
                    target.style.marginBottom = "36px";
                    footer.style.marginTop = '250px';
                    StimeDreat();
                    if (isPlaying === true) {
                        return;
                    } //trueはゲームが始まっている為、クリックでの処理は止めたいのでreturnとする。
                    target.style.fontSize = "48px"; //問題文のフォント変更
                    playcount();
                    isPlaying = true; //クリックするとゲームがスタートする為、trueにしている。
                    setWord();
                    playtyu();
                }
            }, 1000); //ゲーム前のカウントダウン
        } else {
```

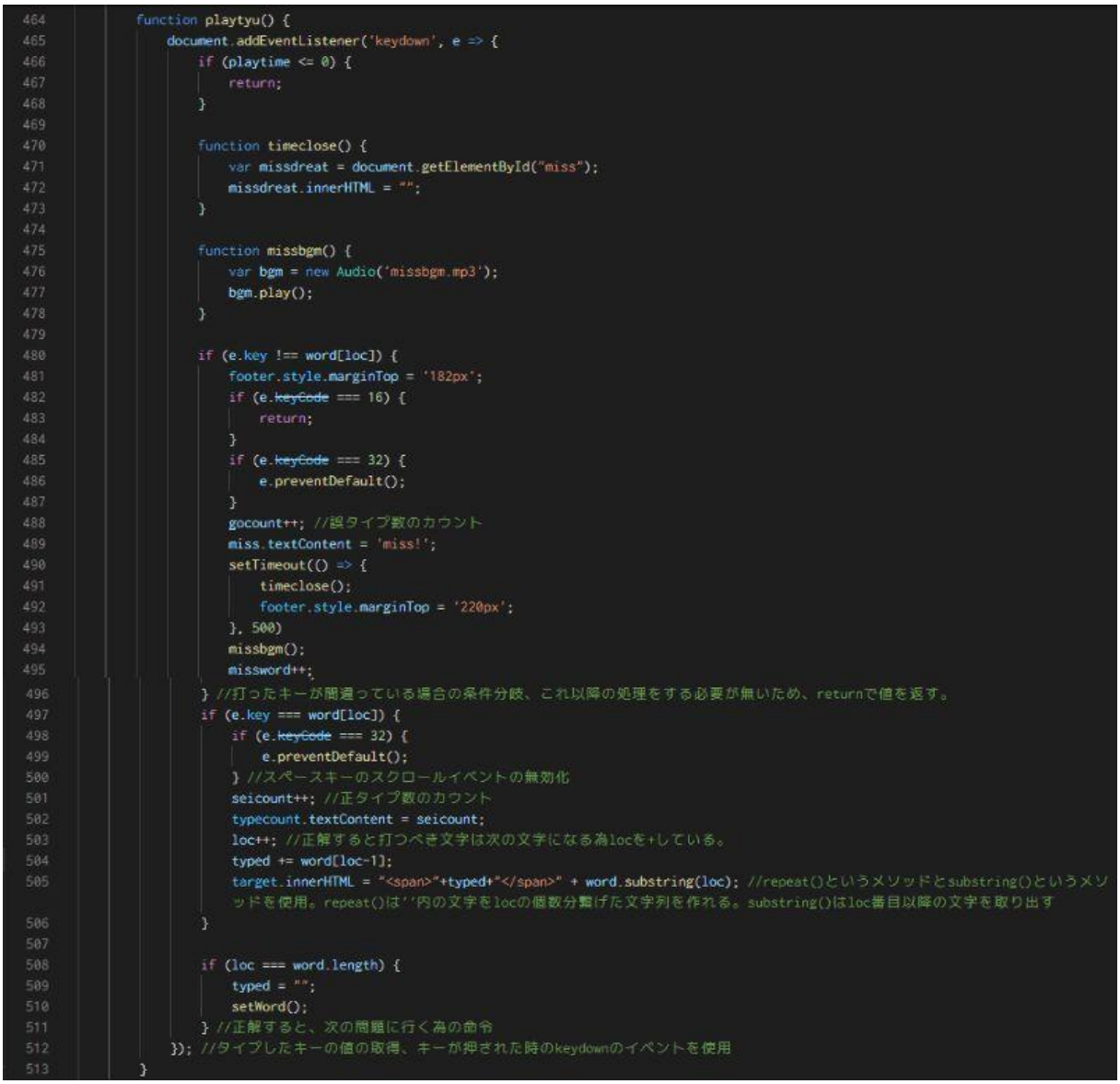
図 23 JS ソースコード開始直前画面

プレイ中の処理についてのプログラムは図 24，図 25 に記載がある．関数 SetWord は，先ほど記述した通り，出題される問題を用意するためのものである．10 行目がその処理に当たり，選ばれた問題が格納されている配列 words から，Math．floor，Math．random という 2 つのメソッドを使用し，ランダムに問題を抽出している．また，一度出題された問題を配列 words 内から排除する形で変数 word に格納し，それが問題となって出題されている．このようにして，問題がランダムに用意される処理が行われている．5~9 行目は間違えた問題を格納する処理であり，11~15 行目は解説を出題順通りに並べ替えるための処理である．

次の図 25 は，プレイ中の処理についての関数 playtyu の中身である．466~468 行目は，プレイ時間が終了すると，この関数内の処理も停止させるという処理を記載している．470~478 行目は，誤タイプ時の miss 文字と音の組み込みをするための関数の作成をしている．480~496 行目は，タイピングミスをしたときの処理で，470~478 行目で設定したミス時の処理の実行，タイピングミスの合計のカウント，タイピングミスをした問題を判別するための変数のカウントを行っている．497~506 行目は，正解のタイピングをした時の処理である．ここでは，正タイプ数のカウントや図 7 のように，入力された文字を透過させる処理を行っている．

```
4   function setWord() {
5       if (missword > 0) {
6           missque[missquecount] = word;
7           missquecount++;
8           missword = 0;
9       } //間違えた問題を抽出
10      word = words.splice(Math.floor(Math.random() * words.length), 1)[0]; //wordsの中からランダム番目を選び、wordに代入して問題を作
        成している。また、splice()というメソッドで選ばれた問題を削除しながらwordにセットしている。
11      for (let i = 0; i < qkanri; i++) {
12          if (word === wordcheck[i]) {
13              qcomment[qnumber] = wordscome[i];
14          }
15      }
16      qorder[qnumber] = word;
17      target.innerHTML = word;
18      loc = 0; //新しい単語をセットしたら、再びlocを0にしたい為、ここで宣言
19      qnumber++;
20      console.log(qcomment);
21      console.log(qorder);
22  } //問題となる単語を用意するための関数
```

図 24 JS ソースコード　関数 SetWord



```
464     function playtyu() {
465         document.addEventListener('keydown', e => {
466             if (playtime <= 0) {
467                 return;
468             }
469
470             function timeclose() {
471                 var missdreat = document.getElementById("miss");
472                 missdreat.innerHTML = "";
473             }
474
475             function missbgm() {
476                 var bgm = new Audio('missbgm.mp3');
477                 bgm.play();
478             }
479
480             if (e.key !== word[loc]) {
481                 footer.style.marginTop = '182px';
482                 if (e.keyCode === 16) {
483                     return;
484                 }
485                 if (e.keyCode === 32) {
486                     e.preventDefault();
487                 }
488                 gocount++; //誤タイプ数のカウント
489                 miss.textContent = 'miss!';
490                 setTimeout(() => {
491                     timeclose();
492                     footer.style.marginTop = '220px';
493                 }, 500)
494                 missbgm();
495                 missword++;
496             } //打ったキーが間違っている場合の条件分岐、これ以降の処理をする必要が無いため、returnで値を返す。
497             if (e.key === word[loc]) {
498                 if (e.keyCode === 32) {
499                     e.preventDefault();
500                 } //スペースキーのスクロールイベントの無効化
501                 seicount++; //正タイプ数のカウント
502                 typecount.textContent = seicount;
503                 loc++; //正解すると打つべき文字は次の文字になる為locを+している。
504                 typed += word[loc-1];
505                 target.innerHTML = "<span>"+typed+"</span>" + word.substring(loc); //repeat()というメソッドとsubstring()というメソ
                    ッドを使用。repeat()は''内の文字をlocの個数分繋げた文字列を作れる。substring()はloc番目以降の文字を取り出す
506             }
507
508             if (loc === word.length) {
509                 typed = "";
510                 setWord();
511             } //正解すると、次の問題に行く為の命令
512         }); //タイプしたキーの値の取得、キーが押された時のkeydownのイベントを使用
513     }
```

図 25 JS ソースコード　関数 Playtyu

アプリケーションが終了すると，図 22 の 408~419 行目の処理が行われる．ここで，プレイ中のタイマーを停止させる処理，「終了！」の文字を出現させる処理を行い，その後，HTML と JavaScript の resultscreen 関数で図 9，10 のように，スコアや解説の表示を行う．

結果画面の処理を表すのが，図 26，27，28 である．図 26 の 63~76 行目でスコア表の作成，78~83 行目で問題解説表の作成，84 行目で復習問題のボタンを作成している．スコアの取得は，JavaScript で行っており，図 27 の 229 行目の scoretable という関数内で処理をしている．スコア表の内，正タイプ数と誤タイプ数は，playtyu 関数内でカウントしている為，この関数では，スコアと正タイプ率の計算を行い，それらの結果を出力した．238 行目が正タイプ率の計算で，変数 rate に代入する形で行った．スコアの計算は，239 行目で行っており，正タイプ率，入力文字数を考慮するような計算式を e-typing のスコアから推測し用いた．また，そのスコアに応じて，e-typing のランクを基に，ランクを判別する式を 240 行目以降に記述している．これらの計算結果を表示するのが，図 28 の 275~279 行目である．このようにして，スコア表の作成を行った．

問題解説表は，図 28 の 285 行目に記述のある関数 comment 内で行った．解説表は，入力した問題数により表の数が変わる為，HTML では，図 26 の 78~83 行目の通り，表のタイトルのみの作成を行い，JavaScript で解いた問題数分の表を付け加える方法を取った．図 28 の 289 行目からが解説表を作成するプログラムであり，qorder という変数で解いた問題の管理を行っていたため，それを使用し，解説に必要な表の数，解説する問題の取得を行い，問題解説表を作成した．

この問題解説表を使い，言語の学習を進めていってもらう．

```
    <p id="result"></p>
    <table id="table">
        <tr>
            <th>スコア</th>
            <th>正タイプ数</th>
            <th>誤タイプ数</th>
            <th>正タイプ率</th>
        </tr>
        <tr>
            <td id="score"></td>
            <td id="seikai"></td>
            <td id="gotou"></td>
            <td id="arate"></td>
        </tr>
    </table>
    <p id="commenttext">今回の問題と解説</p>
    <table id="comment">
        <tr>
            <th id="qtable">問題</th>
            <th id="comtable">解説</th>
        </tr>
    </table>
    <button id="reviwebtn">間違えた問題の復習</button>
```

図 26 HTML ソースコード　結果画面の表

```
function resultscreen() {
    body.style.height = "";
    reviwetar.style.marginTop = "90px";

    function scoretable() {
        document.getElementById('table').style.display = ""; //スコア表の表示
        const score = document.getElementById('score');
        const seikai = document.getElementById('seikai');
        const gotou = document.getElementById('gotou');
        const arate = document.getElementById('arate');
        let rate = 0;
        let rank;
        let scorerank = 0;
        rate = seicount * 100 / (seicount + gocount);
        scorerank = ((rate / 100) ** 3) * (seicount / time);
        if (scorerank >= 277) {
            rank = "SS";
        } else if (scorerank >= 260) {
            rank = "S";
        } else if (scorerank >= 243) {
            rank = "A+";
        } else if (scorerank >= 226) {
```

図 27 JS ソースコード　関数 resultscreen 1

```
272             } else {
273                 rank = "E-";
274             }
275             score.textContent = rank;
276             seikai.textContent = seicount + "回";
277             gotou.textContent = gocount + "回";
278             arate.textContent = rate.toFixed(1) + "%";
279         } //score表を作成・表示する関数
280         scoretable();
281 
282         reviwebtn.style.display = "block";
283         commenttext.style.display = "block";
284 
285         function comment() {
286             const commenttable = document.getElementById('comment');
287             commenttable.style.display = ""; //解説の表示
288 
289             for (let i = 0; i < qorder.length - 1; i++) {
290                 let newRow = commenttable.insertRow();
291                 let newCell = newRow.insertCell();
292                 let newText = document.createTextNode(qorder[i]);
293                 newCell.appendChild(newText);
294 
295                 newCell = newRow.insertCell();
296                 newText = document.createTextNode(qcomment[i]);
297                 newCell.appendChild(newText);
298                 newCell.style.textAlign = "initial"
299                 newCell.style.paddingLeft = "10px"
300                 newCell.style.paddingRight = "10px"
301             }
302         }
303         comment();
```

図 28 JS ソースコード　関数 resultscreen 2

言語学習を終えると，間違えた問題の復習ボタンをクリックし，間違えた問題のタイピングを再度行う．その処理を記述したのが図 29，30 である．まず，間違えた問題の有無を調べる必要がある為，その処理を 311 行目に記載している．間違えた問題が無ければ，図 11 の画面になり，間違えた問題があれば，319 行目以降の処理が進む．

復習問題の進め方は，通常の問題と変わっておらず，words 配列に問題を格納し，それをランダムに word 変数に振り当てて出題するという方式である．しかし，復習問題は，間違えないようにタイピングすることをより意識してもらいたいと考え，制限時間や正タイプ率の表示，タイプミス時の演出等を消す方法を取った．その為，通常の問題に使用した playtyu や SetWord などの関数は使用せず，再度タイピングを行う為のプログラムを構築した．通常問題で格納された問題を空にするために，321 行目で words 配列を空にし，次の行で，間違えた問題を words 配列に代入する．そして，図 30 の 370 行目でランダムに問題を word 変数に代入し，次の行で出題している．復習問題を全て終えると，357 行目の中の処理に入り，図 13 のように終了画面が表示される．

このような流れで，本アプリケーションを構築した．

```
305        reviwebtn.addEventListener('click', () => {
306            reviwebtnDreat();
307            comDreat();
308            document.getElementById('table').style.display = "none";
309            document.getElementById('comment').style.display = "none";
310            var reviwetar = document.getElementById('reviwetar');
311            if (missquecount === 0) {
312                body.style.height = "630px";
313                footer.style.position = "absolute";
314                footer.style.left = "233px";
315                footer.style.right = "233px";
316                footer.style.bottom = "0";
317                reviwetar.style.fontSize = "40px";
318                reviwetar.innerText = '間違えた問題はありません。\nお疲れさまでした!';
319            } else if (missquecount > 0) {
320                reviwetar.style.paddingBottom = "150px"; //復習問題画面のスタイル調整
321                words = [];
322                for (let i = 0; i < missque.length; i++) {
323                    words[i] = missque[i];
324                }
325                reviwetar.textContent = "EnterキーかSpaceキーを入力でスタート";
326                reviwetar.style.fontSize = "40px";
327                reviwetar.style.fontFamily = 'Courier New'; //問題文のフォント変更
328                let keykanri = false;
329                document.addEventListener('keydown', e => {
330                    reviwetar.style.fontSize = "48px";
331                    if (e.keyCode === 13 || e.keyCode === 32) {
332                        if (keykanri === true) {
333                            return;
334                        }
335                        if (keykanri === false) {
```

図 29 JS ソースコード　復習問題 1

```
346                document.addEventListener('keydown', e => {
347                    if (e.keyCode === 32) {
348                        e.preventDefault();
349                    }
350                    if (e.key !== word[loc]) {
351                        return;
352                    }
353                    loc++; //正解すると打つべき文字は次の文字になる為locを+している。
354                    typed += word[loc-1];
355                    reviwetar.innerHTML = "<span>"+typed+"</span>" + word.substring(loc);
356                    if (loc === word.length) {
357                        if (words.length === 0) {
358                            body.style.height = "630px";
359                            footer.style.position = "absolute";
360                            footer.style.left = "233px";
361                            footer.style.right = "233px";
362                            footer.style.bottom = "0";
363                            reviwetar.style.display = "none";
364                            const result = document.getElementById('result');
365                            result.innerText = '復習終了!\nお疲れさまでした。';
366                            result.style.fontSize = "40px";
367                            return;
368                        } //wordsから単語が無くなった時の処理
369                        typed = "";
370                        word = words.splice(Math.floor(Math.random() * words.length), 1)[0];
371                        reviwetar.textContent = word;
372                        loc = 0;
373                    } //正解すると、次の問題に行く為の命令
374                }); //タイプしたキーの値の取得、キーが押された時のkeydownのイベントを使用
375            }
376        });
```

図 30 JS ソースコード　復習問題 2

4 ユーザテストについて

制作したアプリケーションの評価を行うべく，ユーザテストを実施した．ユーザテストの目的は，本アプリケーションの操作性や改善点の客観的な評価を得ることやタイピング能力の向上やプログラミング言語の学習にどの程度の効果があるのかを調査するためである．テストの実施手順は，開始前に，アンケートと言語能力テスト，タイピングテストを行い，本研究で制作したアプリケーションを計 10 回プレイしてもらう．テスト終了後に，開始前と同様の言語テスト，タイピングテストをプレイしてもらうという流れである．次に，このユーザテストの具体的な中身について説明する．

まず，開始前アンケートは，プログラム経験の有無，日々のパソコンに触れる回数を調査するものである．この調査をすることで，テストの目的である「どの層にどれくらい効果があるのか」を調べる事が出来る．また，言語テストは，タイピングアプリケーションの問題を中心に制作し，開始段階での言語の理解度を調査した．タイピングテストは，外部のタイピングアプリケーションである e-typing で行った．制作したアプリケーションでテストを行わなかった理由は，本アプリケーションに慣れてスコアが上昇する人がいる可能性があるという意見が議論であがり，それを考慮した為である．

次に，実施期間は，20 日とし，2 日に 1 回プレイしてもらうこととした．この理由は，参考文献 2，3 の 2 つの論文を参考にしたもので，週 4 回程度の継続的な練習に効果があるという点と，10 回程度の練習でタイピング精度が向上するという結果がでたという点，また，毎日はプレイできないというテストユーザの意見，これらを考慮したものである．[2][3]テスト時間は，同様の論文を参考に，1 回で 20 分程度の学習が適切と判断し，1 回のプレイで同様のテストを 2 回プレイしてもらうこととした．プレイする難易度は，最初の 5 回はかんたん，次に普通を 3 回，最後にむずかしいを 2 回プレイしてもらうこととした．このように設定した理由は，初心者により基礎を固めてもらう為である．本アプリケーションの目的の 1 つに，プログラミングの際の入力ミスによるエラーを減らすというものがあり，プログラミング初心者を当初のターゲットユーザに設定した為，それを反映する形で，プレイする難易度を設定した．

最後に，終了後アンケートについて説明する．終了後アンケートは，こちらが設定した時間通りプレイできたか，本アプリのやりやすさ，タイピング精度と言語能力の向上が感じられるかについて質問した．これらの質問で，操作性についての問題点や改善点，能力向上を実感できているか等を調査した．また，終了後の言語テストは，開始前のテストと同様のものを使い，言語能力の向上の有無やより効果があった問題の種類等を調査した．

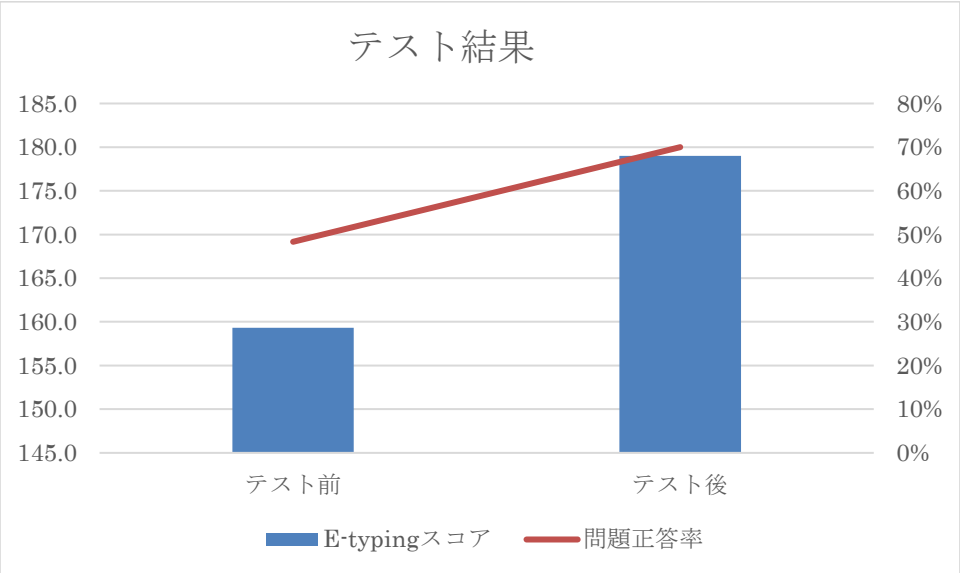


図 31 ユーザテスト結果(C 言語)

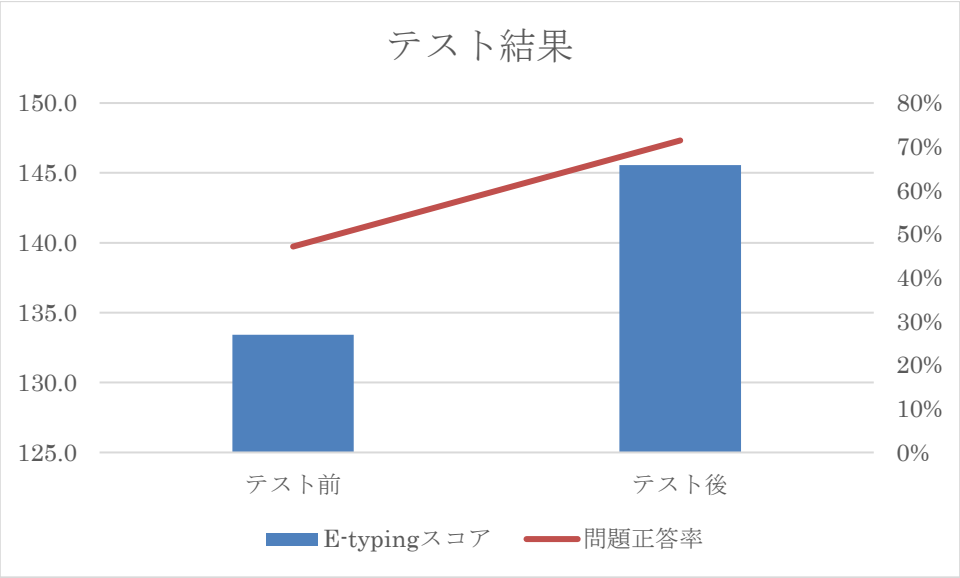


図 32 ユーザテスト結果(JavaScript)

このテストの結果が，図 31，32 である．まず，テストを指定した回数分プレイ出来たのは，テストユーザ 15 名中 13 名で，C 言語は 7 人中 6 人，JavaScript は 8 人中 7 人であった．そして，結果は，タイピング力，言語学習共に，能力が上がるという効果が見られた．この原因について，それぞれ分析してみた．

はじめに，E-typing のスコアは，C 言語のテスト受験者は 20，JavaScript のテスト受験者は 12 程度上がった．これを E-typing のスコア表に置き換えてみると，およそワンランク伸びたという結果になった．参考文献 2，3 の論文と比較すると，プレイ時間に対しての伸び率はそれほど大きくはなかった．[2][3]この原因として，問題文が英語のタイピングアプリケーションの為，ローマ字タイピングには，それほど慣れることが出来なかったことが考えられる．しかし，スコアは向上している為，英語のタイピングでもタイピング力の向上には繋がるということが，この研究で分かった．

次に，言語学習についてである．問題正答率は，C 言語・JavaScript のどちらも大きく向上しており，言語学習についての効果は大きくあったと考察できる．問題ごとの正答率を調べると，C 言語は，正答率が大きく上がった問題とあまり変わらなかった問題がそれぞれあったが，JavaScript は全体的に正答率が伸びていた．この原因は，C 言語に英語の意味を理解するとある程度問題が解けるという特徴があったことと考えられる．例えば，「数学関数を使用する際のヘッダファイル名は何か.」という問題では，答えに math という数学を表すワードが入っていればそれが正解と導き出せる．このように，この特徴を示す問題が C 言語には多く，問題の正答率が混在した原因と考えられる．それぞれの正答率が向上した要因は，テストユーザが問題の解説をよく読み，学習を進めたことにある．言語テストの問題は，タイピング問題の解説を基に作成していたため，解説を読みながら学習を進めることでより効果があるという特徴を持っていた．その特徴を，開始前のテストとアプリケーションをプレイしていく中で，気づいた人が多く，より学習が進み，能力が向上したと考えられる．

最後に，プログラミング未経験者でこのテストをプレイしてもらったユーザ 2 名に，テスト後に，プログラムを 1 から 1 人で書けるかというテストを行った．結果は，どちらもプログラムを書く事は出来なかった．C 言語でテストを行ったユーザは，8 割程度は，コードを書くことが出来ていたが，使用する文が一部分からず，完成できなかった．JavaScript のユーザは，ほとんど書くことが出来なかった．原因は，C 言語は，その言語のみで完成させることが出来る為，理解しやすかったことに対し，JavaScript は，HTML の知識も少し必要で，理解に時間を要してしまう事があると考えた．

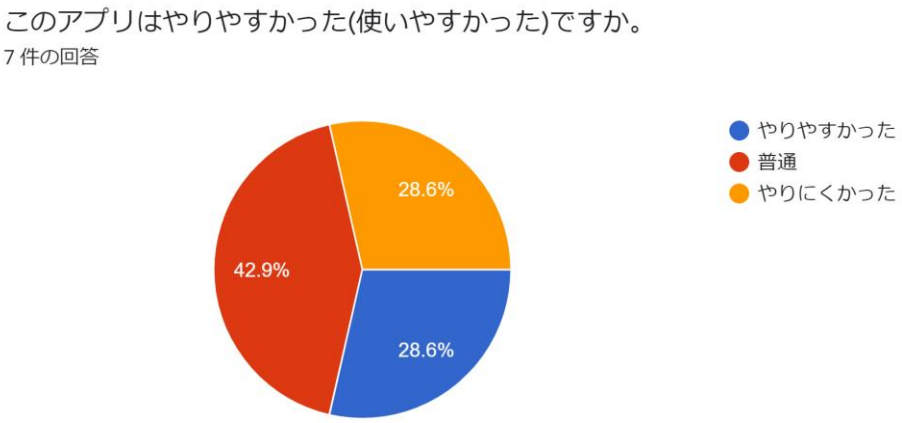


図 33 ユーザテスト使用感アンケート(C 言語)

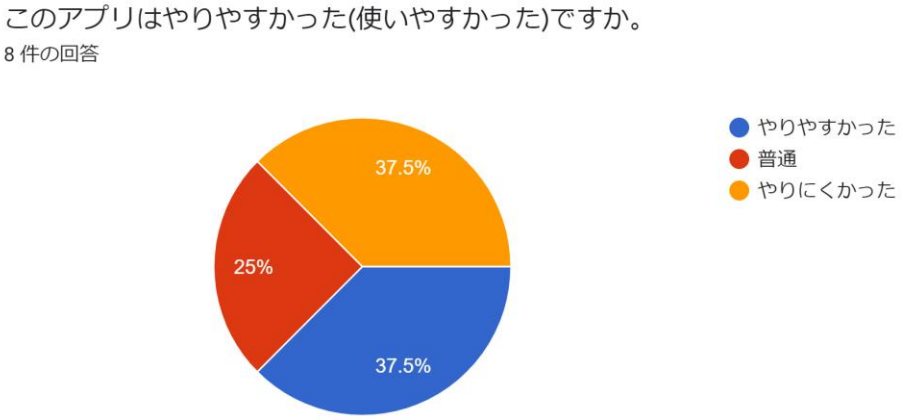


図 34 ユーザテスト使用感アンケート(JavaScript)

次に，使用感についてのアンケート結果を，図 33，34 に示した．図の通り，「やりやすかった」と感じたユーザは 3 割程度で，あまり多くなかった．「やりにくかった」と答えた人に，理由を聞くと，「既存のタイピングアプリの方がやりやすい」，「タイピングミス時の音がうるさかった」などの意見をいただいた．また，「普通」と答えている人も多く，既存のタイピングアプリケーションと比べて，特徴をうまく持たせることが出来なかったことも反省点の 1 つである．使いやすくかつ他にはない特徴を持たせるということをどのように実現するかということは，ユーザインタフェースの今後の課題になる．

5 おわりに

本論文では，プログラミング初心者のタイピングミスによるプログラムエラーの多さに注目し，その課題の解決に向けて，プログラミング初心者向けタイピングアプリケーションの開発，評価を行った．このタイピングアプリケーションは，タイピング力の向上と言語学習の推進を目指し，タイピングアプリケーションの問題を，よくあるローマ字入力やかな文字入力とは異なる，プログラミングで使用する英文や英単語に置き換えて，2 つの学習を同時に進めることが出来るように設計した．

ユーザテストでアプリケーションの評価を行った結果から，タイピング力，言語能力のどちらにも効果がある事が分かったが，ユーザの能力により効果に大きな差が出ており，当初の目標であったプログラミング初心者の能力向上より，プログラミング経験者の復習ツールとしての効果がより大きくあることが分かった．また，アプリケーションの使いやすさは，あまり良い評価を得ることはできず，他の既存のタイピングアプリケーションと差異はほとんど無いという結果となった．このことから，このアプリケーションの効果はあったが，まだまだ改善が必要な点が多くあることが分かる．

今後の課題は，より使いやすいと思われるユーザインタフェースの改善，言語学習機能の充実，プレイしたいと思われる付加価値の増加などが考えられる．これらの点について，どのように実装するか，どうすれば使いやすいと思われるかなどの考察が必要である．開発，テストのサイクルを複数回繰り返し，アプリケーションの改善に努めていきたい．

参考文献

1) インターネットでタイピング練習　イータイピング E-typing (参照 2022-02-07)
https://www.e-typing.ne.jp/

2) 吉長裕司・川端洋昭　タッチタイピングの習熟過程における初期熟達感の考察　(2001)
https://www.jstage.jst.go.jp/article/jmet/25/suppl/25_KJ00003905555/_pdf/-char/ja

3) 池村努　短期大学生におけるタイピング練習に関する研究　(2016)
https://hokurikugakuin.repo.nii.ac.jp/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=991&item_no=1&page_id=13&block_id=21