How to embed GeoGebra into Web

藤本 光史（福岡教育大学）*

概要

ユーザーインタフェース作成のための JavaScript ライブラリである React を用いて GeoGebra を Web へ埋め込むための手法について解説する。この手法は実行前に JavaScript コードのエラーチェックができるため、GeoGebra のスクリプトを利用した教材開発環境を改善することに繋がる。また、他の Web 技術と連携することで高度で柔軟なユーザーインタフェースを作成することが可能となる。

1 はじめに

2017 年及び 2019 年の本研究会において、筆者は GeoGebra でのスクリプトの利用法について解説（GeoGebra スクリプトの利用法 [1]、JavaScript の利用法 [2]）した。ボタンや入力ボックスなどを設置した GeoGebra 教材を作成するにはスクリプトの利用が不可欠である。特に、JavaScript を利用する方法ではイベントリスナーが使用可能であるため、より複雑な教材の作成が可能となる。しかし、GeoGebra に内蔵されたスクリプト入力ダイアログは決して使いやすいものではない。スクリプト入力ダイアログについて改善が必要と思われる項目を以下に挙げる。

- 点や線分などのオブジェクトを作成しないとスクリプト入力ダイアログへアクセスできない。
- コードの整形機能や補完機能が貧弱[*1]である。
- JavaScript コードの修正を反映させるには GeoGebra の再起動が必要である。
- JavaScript コードにエラーがあっても、何のメッセージも表示されない[*2]。
- 作成したコードのバージョンを管理する仕組みが提供されておらず、既存のバージョン管理システムにも対応できていない。

以上の項目を改善するために「**React を用いた Web アプリ化**」を提案する。React を用いて GeoGebra を Web に埋め込むことで上記が改善できるだけでなく、WebAssembly などの他の Web 技術と組み合わせることにより、既存の様々な数学ソフトウェアと連携できるようになるこ

* fujimoto@fukuoka-edu.ac.jp

*1 括弧の強調表示やインデントの自動挿入は可能であるが、インデントの挿入は直ちに行われず GeoGebra の再起動が必要である。

*2 起きるはずのことが起きないことでコードに誤りがあることを理解しないといけない。

とが期待できる。

2　スクリプトを利用した例

Web への埋め込むを行う前に、JavaScript を利用した GeoGebra ワークシートを作成しておく。ここで作成するのは点と線分で描かれた（連結）図形が一筆書き可能かどうかを判定するワークシートであり、これを次節以降で Web に埋め込んでいく。

まず GeoGebra で JavaScript のコードを入力するには何らかのオブジェクトが必要である。そこで、ツールバーの右から 2 つ目のツールボタンの▽から「ボタン」を選択後、描画エリア（グラフィックスビュー）をクリックして判定ボタンを作成する。

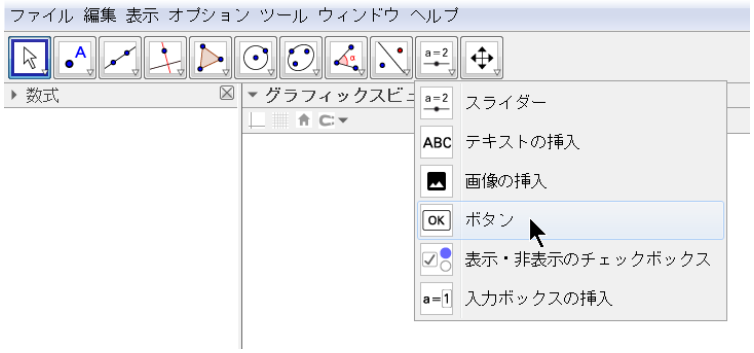

図 1　判定ボタンの作成

そして、判定ボタンを右クリック→「オブジェクトのプロパティ」→「スクリプト記述」→「**グローバル Java スクリプト**」欄に以下のコードを入力する。ただし、枠外の数字は行番号であり入力してはいけない。

コード 1　「**グローバル Java スクリプト**」欄への入力

```
1  function ggbOnInit() {
2      ggbApplet.registerAddListener(ListenerA);
3      ggbApplet.registerRemoveListener(ListenerRM);
4      ggbApplet.registerRenameListener(ListenerRN);
5      objectList();
6  }
7  function ListenerA(P1) {
8      objectList();
9  }
10 function ListenerRM(P1) {
11     objectList();
12 }
13 function ListenerRN(P1, P2) {
14     objectList();
15 }
```

```
16 var Pname = [];
17 var Sname = [];
18 function objectList() {
19     ggbApplet.unregisterAddListener(ListenerA);
20     ggbApplet.unregisterRemoveListener(ListenerRM);
21     Pname = [];
22     Sname = [];
23     for (var i = 0; i < ggbApplet.getObjectNumber(); i++) {
24         var obj = ggbApplet.getObjectName(i);
25         if (ggbApplet.getObjectType(obj) === "point") Pname.push(obj);
26         else if (ggbApplet.getObjectType(obj) === "segment") Sname.push(obj);
27     }
28     ggbApplet.registerAddListener(ListenerA);
29     ggbApplet.registerRemoveListener(ListenerRM);
30 }
```

objectList() 関数は、描画エリアに存在する全てのオブジェクトを調べ、それが点ならば配列 Pname に、線分ならば配列 Sname にそのオブジェクト名を格納していく。この関数は 2〜4 行目のオブジェクトの追加・削除・リネームに対応したイベントリスナーから呼び出されるようになっている。

次に、「グローバル Java スクリプト」欄の左にある「**クリックして**」欄に以下のコードを入力する。

コード 2 「**クリックして**」欄への入力

```
1  var connect = [];
2  for (var i = 0; i < Pname.length; i++) {
3      ggbApplet.evalCommand('tmpC=Circle(' + Pname[i] + ',0.5)');
4      var cnt = 0;
5      for (var j = 0; j < Sname.length; j++) {
6          ggbApplet.evalCommand('tmpP=Intersect(tmpC, ' + Sname[j] + ')');
7          if (ggbApplet.isDefined('tmpP')) cnt++;
8          ggbApplet.deleteObject('tmpP');
9      }
10     connect.push(cnt);
11     ggbApplet.deleteObject('tmpC');
12 }
13 for (var even = 0, odd = 0, k = 0; k < Pname.length; k++) {
14     ggbApplet.setCaption(Pname[k], connect[k]);
15     ggbApplet.setLabelStyle(Pname[k], 3);
16     if (connect[k] % 2 !== 0) {
17         ggbApplet.setColor(Pname[k], 255, 0, 0);
18         ggbApplet.setPointSize(Pname[k], 7);
19         odd++;
20     } else {
```

```
21          ggbApplet.setColor(Pname[k], 0, 0, 255);
22          ggbApplet.setPointSize(Pname[k], 5);
23          even++;
24      }
25  }
26  ggbApplet.evalCommand('odd=FormulaText("\\Large 奇点: ' + odd + '")');
27  ggbApplet.setCoords('odd', 0, 5.5);
28  ggbApplet.evalCommand('even=FormulaText("\\Large 偶点: ' + even + '")');
29  ggbApplet.setCoords('even', 3, 5.5);
30  if (odd === 0 || odd === 2) {
31      ggbApplet.evalCommand('result=FormulaText("\\Large \\Rightarrow 一筆書き可能")');
32      ggbApplet.setCoords('result', 0, 4.5);
33  } else {
34      ggbApplet.evalCommand('result=FormulaText("\\Large \\Rightarrow 一筆書き不可能")');
35      ggbApplet.setCoords('result', 0, 4.5);
36  }
```

2〜12 行目で各点に接続されている線分の数（接続数）をカウントし、その結果をグローバル変数 connect に保存している。接続数が奇数の点を**奇点**、偶数の点を**偶点**と呼ぶ。14 行目で各点の接続数をキャプションに設定し、16〜24 行目で奇点と偶点の個数をカウントし、奇点の色を赤に変更している。26〜29 行目で奇点と偶点の個数を描画エリアに出力し、30〜36 行目でその図形が一筆書き可能か不可能かを出力している[*3]。

注意事項：「クリックして」欄では、下部のドロップダウンリストで「GeoGebra Script」を「JavaScript」に変更する必要がある。

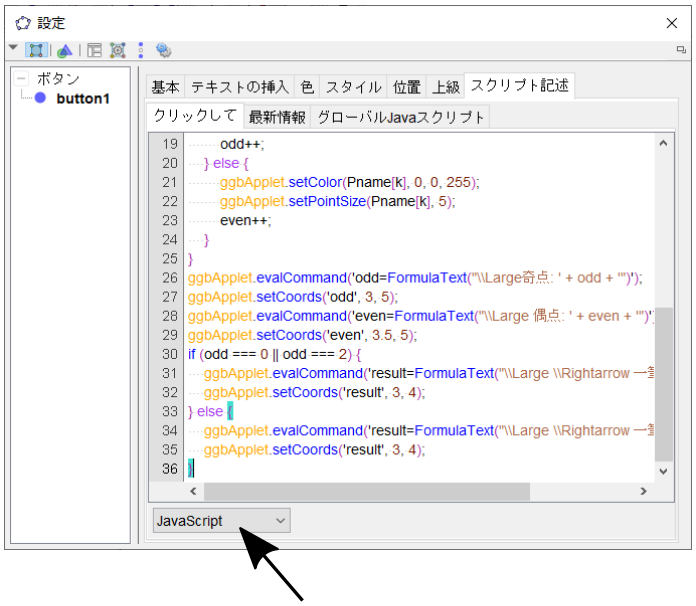

図 2　入力コードを JavaScript に変更

[*3] 連結な図形が一筆書き可能であるための必要十分条件は、「**奇点の個数が 0 個または 2 個であること**」である。

作成したワークシートに名前を付けて保存し、一旦 GeoGebra を終了する。そして、再度 GeoGebra を起動してそのファイルを読み込む[*4]。描画エリアに点と線分で図形を描いた後、判定ボタンをクリックすると、次の図ように判定結果が表示される。

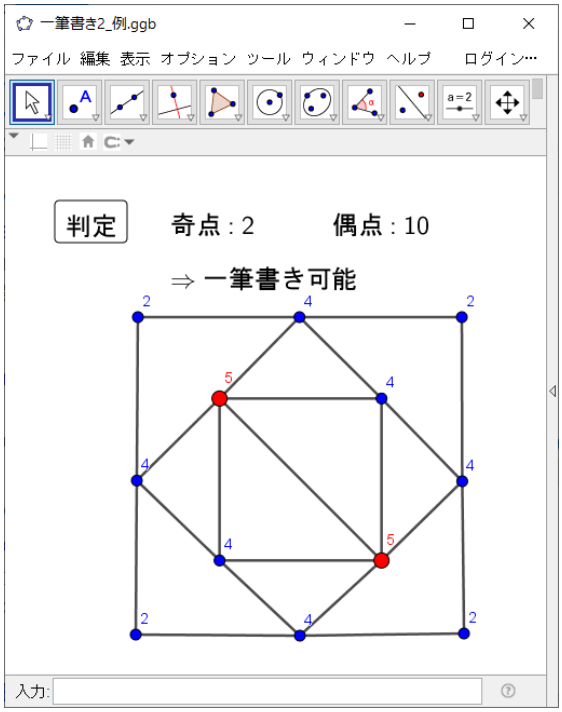

図 3　一筆書き可能な図の判定

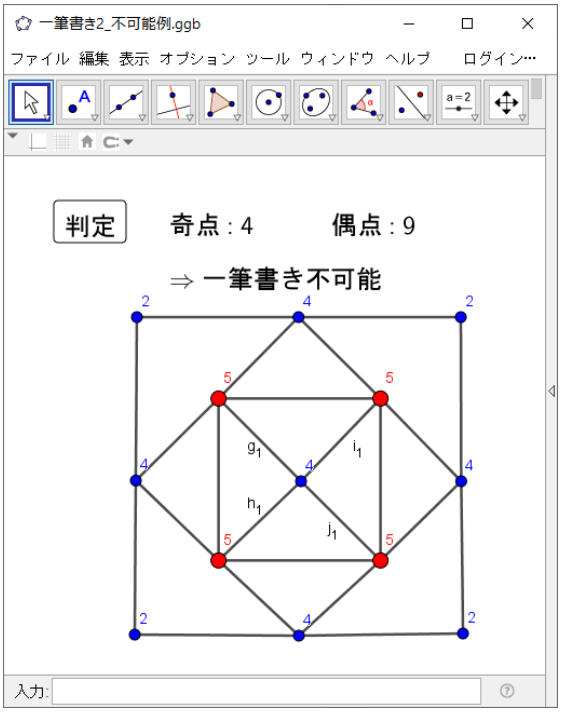

図 4　一筆書き不可能な図の判定

3　Web への埋め込み：公式の方法

GeoGebra バージョン 4.2 までは HTML エクスポート機能があったので、ワークシートから直接 HTML ファイルを生成し、それを加工して Web に公開することができた。しかし、現在のバージョンではこの機能が削除されている。削除された理由の詳細についてはフォーラム上の議論 [3] を参照して欲しい。

ワークシートを Web 上で公開するには公式サイトの **GeoGebra Materials**（旧名称 GeoGebraTube）[4] を利用する。まず Web ブラウザで `https://www.geogebra.org/materials` にアクセスする。ここでは、2 節で作成した GeoGebra ワークシートを公開してみよう。

[*4] GeoGebra スクリプトと異なり、JavaScript を使用した場合はファイルを保存して再読み込みしないと修正は反映されない。

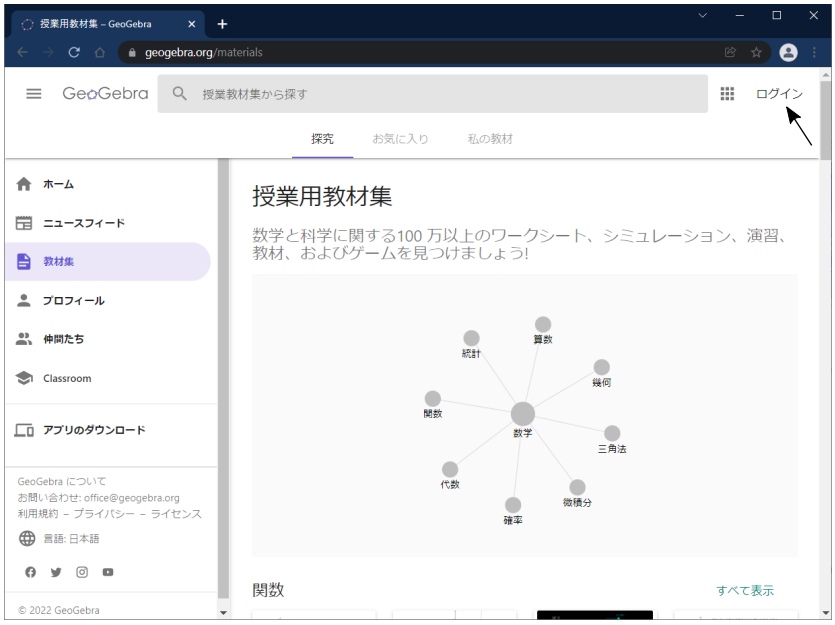

図 5　GeoGebra Materials

右上の**ログイン**をクリックしてログイン[*5]する。

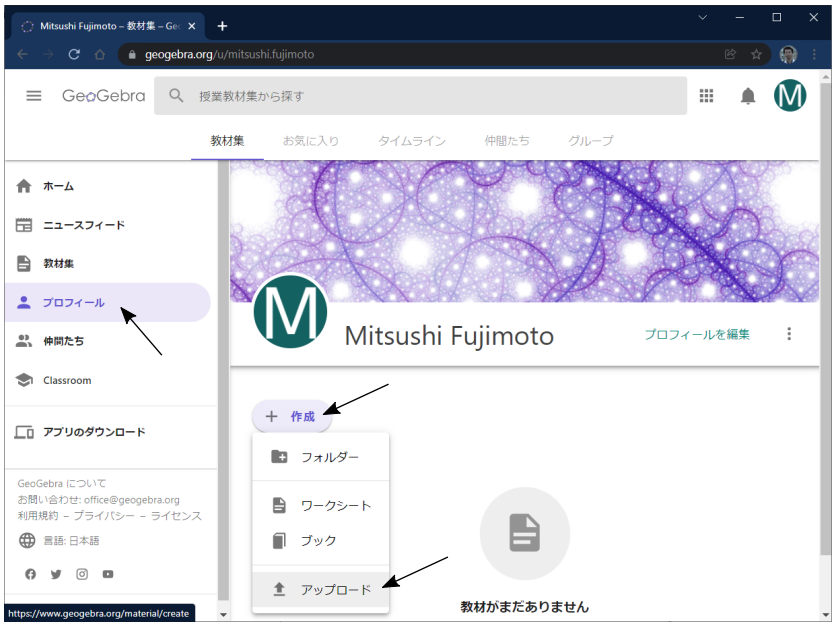

図 6　アップロード時の画面

そして、**プロフィール** → ＋作成 ボタン → **アップロード**の順でクリックする。

[*5] はじめてアクセスする際は「アカウントを作成」をクリックしてアカウントを作成しなければならない。

図 7　アップロードするファイルを選択

ファイルを選択ボタンをクリックして、2 節で作成したワークシートの .ggb ファイルを選びアップロードする。

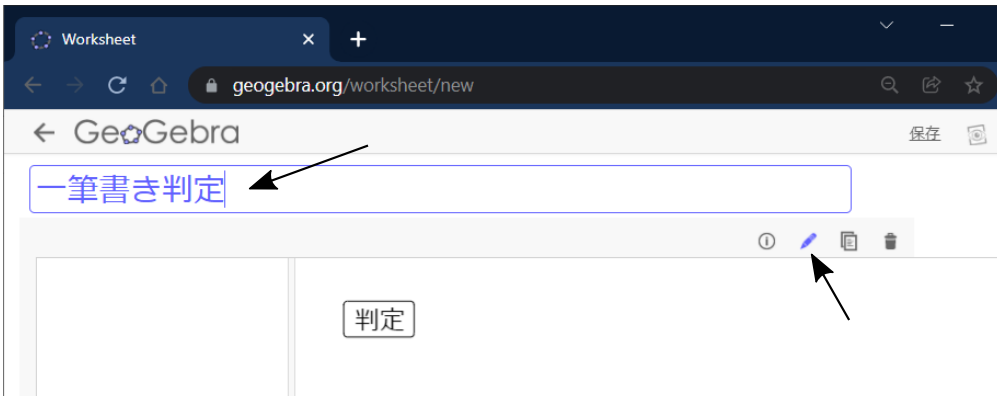

図 8　タイトルの設定と編集ボタンのクリック

次に、タイトル（ここでは「一筆書き判定」とした）を入力し、マウスポインタをタイトル欄右下に移動すると現れる編集ボタン（鉛筆型のアイコン）をクリックする。

図 9　ツールバーの設置

画面を下方にスクロールして、「**高度な設定...**」をクリックして、「**ツールバーを表示**」にチェックを入れて、完了ボタンをクリックする。

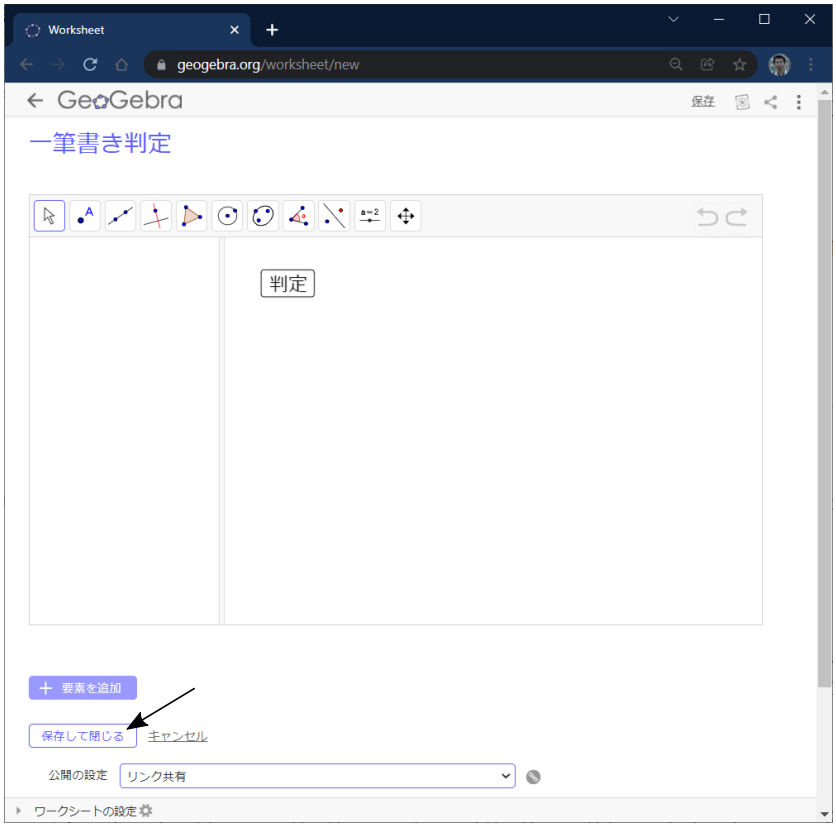

図 10　編集完了画面

最後に、保存して閉じるをクリックすると、アップロードしたワークシートが「教材集」に並ぶ。

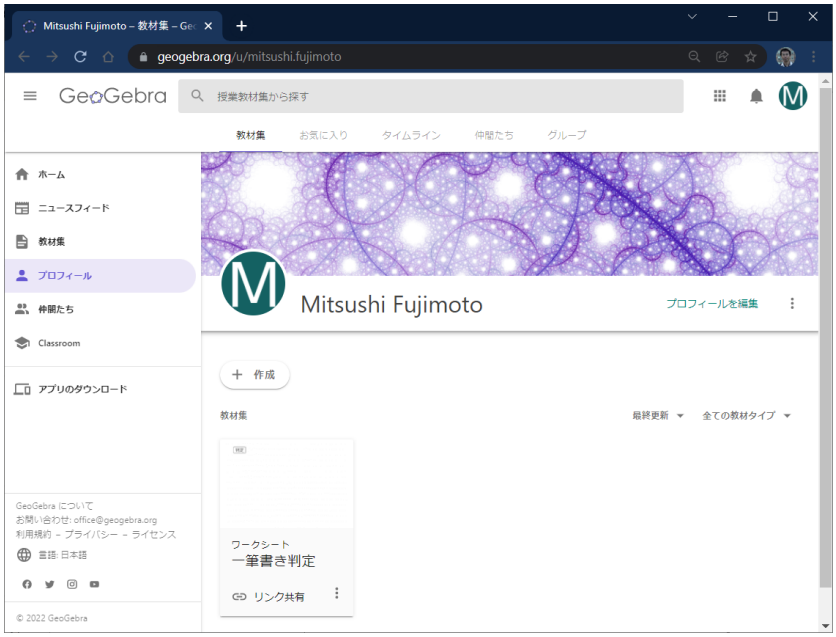

図 11　アップロードされたワークシート

これをクリックすると、Web に埋め込まれたワークシートが表示される。

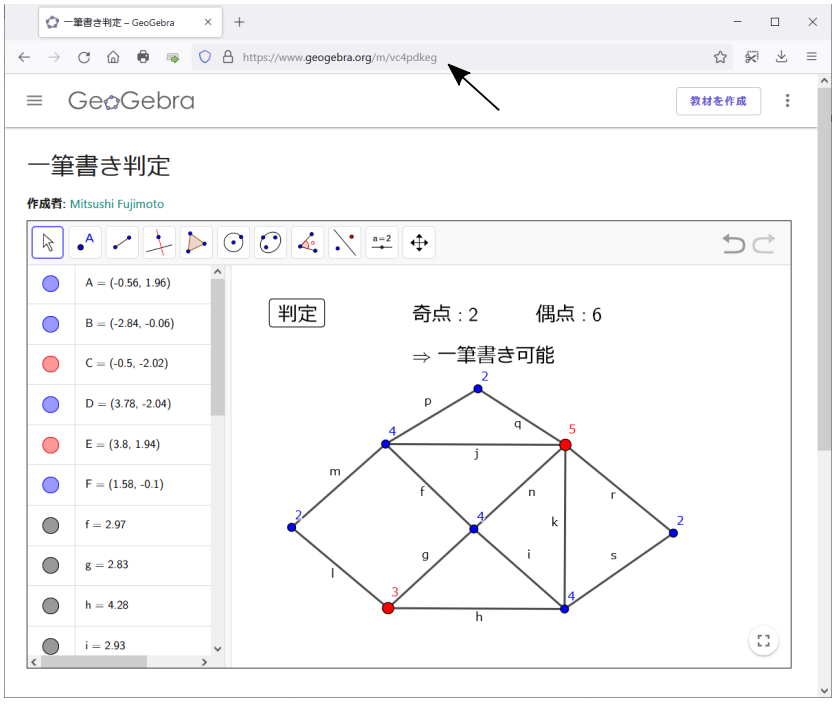

図 12　Web に埋め込まれたワークシート

Web ページの URL は固有のものが自動的に生成される。この URL から誰でもこのワークシート

にアクセスできる[*6]。

4 Web への埋め込み：React を用いた方法

3 節で解説した GeoGebra Materials を利用した Web への埋め込み方法は比較的簡単であるが、ボタンや入力ボックスをワークシートの外に配置するなど高度なユーザーインタフェースを作成したい場合には適さない。GeoGebra の公式サイトにはそのような要望に応えるための方法が記載されている [5]。しかし、ここでは 1 節で述べた様々な点を改善するために **react-geogebra** を用いた方法を紹介する。

4.1 React とは

React [6] とは、Facebook によって開発された Web アプリのユーザーインタフェース作成のための JavaScript ライブラリである。JavaScript を拡張して HTML の Document Object Model(DOM) を直接操作可能にしたスクリプト言語 JSX が利用できるため、短くわかりやすいコードの記述が可能である。

例えば、HTML において `id` が `root` の`<div>`要素内に、与えられた `id` と文字列を持つ`<p>`要素を追加する際、JavaScript では通常次のように記述する。

コード 3　JavaScript による要素の追加

```
1 function text(id, children) {
2   const elem = document.createElement('p');
3   elem.className = id;
4   elem.className = 'text';
5   elem.innerHTML = children;
6   return elem;
7 }
8 const root = document.getElementById('root');
9 root.appendChild(text('sample','Hello'));
```

これを JSX で記述すると次のようになる。

コード 4　JSX による要素の追加

```
1 function Text({ id, children }) {
2   return <p id={id} className="text">{children}</p>;
3 }
4 const root = document.getElementById('root');
5 ReactDOM.render(<Text id="sample">Hello</Text>, root);
```

[*6] ここで作成した一筆書き判定教材の URL は `https://www.geogebra.org/m/vc4pdkeg` である。

このように JavaScript の 2〜6 行目をわずか 1 行で記述することができる。ただし、JSX はそのままでは Web ブラウザに搭載された JavaScript エンジンに理解されないため、通常の JavaScript に変換 (トランスパイル) する必要がある。このトランスパイルの過程で JSX だけでなく JavaScript についてもエラーがないかチェックされるので、C 言語などのコンパイラ型言語のように実行前にコードのミスを見つけることが可能となる。

4.2 React 利用の準備

React を利用するためには Node.js [7] のインストールが必要である。Node.js は JavaScript の実行環境の一つであり、通常は Web ブラウザ上で動作させる JavaScript コードをコマンドライン上で実行可能にするものである。Node.js が「環境」と呼ばれるのは、非常に多くの有用なパッケージが開発されていて、それらを管理するパッケージマネージャの npm (Node Package Manager) と共に利用されるためである。

Node.js と npm は頻繁にバージョンアップされており、バージョンの差異で動作に支障が出る場合がある。そのため nvm (Node Version Manager) をインストールした上で、nvm に Node.js と npm を管理させることが推奨されている。以下では、Windows 10 の WSL2 上の Linux 環境を用いることにする。

4.2.1 nvm のインストール

GitHub にある nvm の公式サイト `https://github.com/nvm-sh/nvm` に従い、以下を実行する[*7]。

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

これによって、ホームディレクトリの `.bashrc` に nvm 関連の設定が追加される。一旦、ターミナルを終了し再起動すると、nvm が使えるようになる。

```
$ nvm -v
0.39.1
```

このように nvm のバージョンが表示されることを確認しておく[*8]。

4.2.2 Node.js のインストール

Node.js の最新安定バージョンである LTS (Long Term Support) 版と、それに対応した最新の npm をインストールする。

```
$ nvm install --lts
$ nvm install-latest-npm
```

[*7] `v0.39.1` の部分はアップデートされる度に変更される部分なので公式サイトを必ずチェックすること。

[*8] nvm のバージョンが表示されない場合は、ターミナルの起動時に `.bashrc` を読み込むように設定すること。

これで Node.js と npm がインストールされる。念のため、バージョンをチェックしておく。

```
$ node -v
v16.13.2
$ npm -v
8.3.2
```

4.3 React アプリの例

Node.js 環境の動作確認のために、React アプリのサンプルを動かしてみよう。React 用の作業ディレクトリを作成し、テンプレートを用意するためにターミナルで以下を実行する。

```
$ mkdir ~/react
$ cd ~/react
$ npx create-react-app my-app
```

npx は npm パッケージを（ディレクトリに）インストールせずに実行するコマンドである。これによって my-app ディレクトリが作成され、その中に React アプリのテンプレートが生成される。

次に、ターミナルで以下を実行する。

```
$ cd my-app
$ export BROWSER=none
$ npm start
```

この操作によりテンプレートがトランスパイルされ、同時に簡易 Web サーバが起動する[*9]。Web ブラウザから http://localhost:3000 にアクセスして次の画面が表示されれば動作確認完了である。

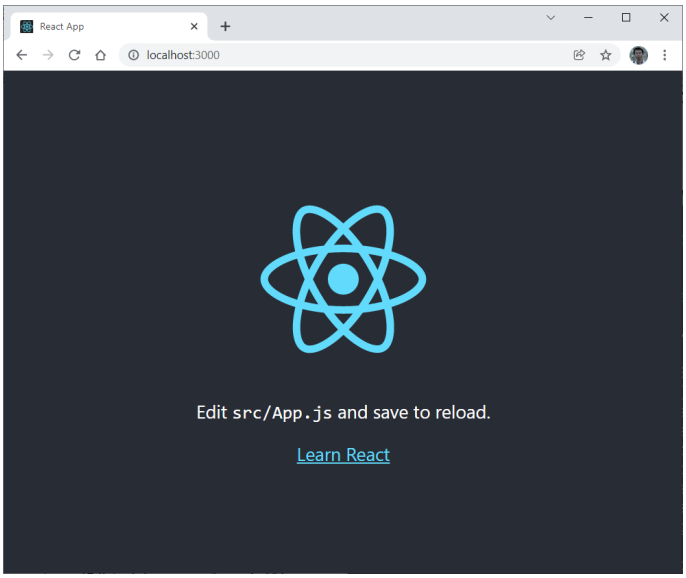

図 13　React アプリのテンプレートの実行画面

[*9] トランスパイル結果はファイルに保存されず、メモリ上にのみ生成され、それが直接簡易 Web サーバに提供される。

簡易 Web サーバを終了するにはターミナルで Ctrl + C を入力すればよい。

4.4 最小構成の React プロジェクト

React アプリを開発するにはまず「プロジェクト」を用意する必要がある。プロジェクトとは React アプリをビルドするために必要なソースやパッケージをまとめたものであり、非常に多くのファイルから成るためツールを利用して生成するのが一般的である。4.3 節で用いた create-react-app もプロジェクト生成ツールの一つであり、シンプルな構成の React アプリを開発する際に推奨されているものである。しかし、デフォルトで生成されるものには React のロゴ画像や CSS ファイルなどが含まれており、単純な React アプリを開発する場合にはそれらが邪魔になることもある。ここでは、create-react-app が生成したファイルをベースに最小構成のテンプレートを作成する。

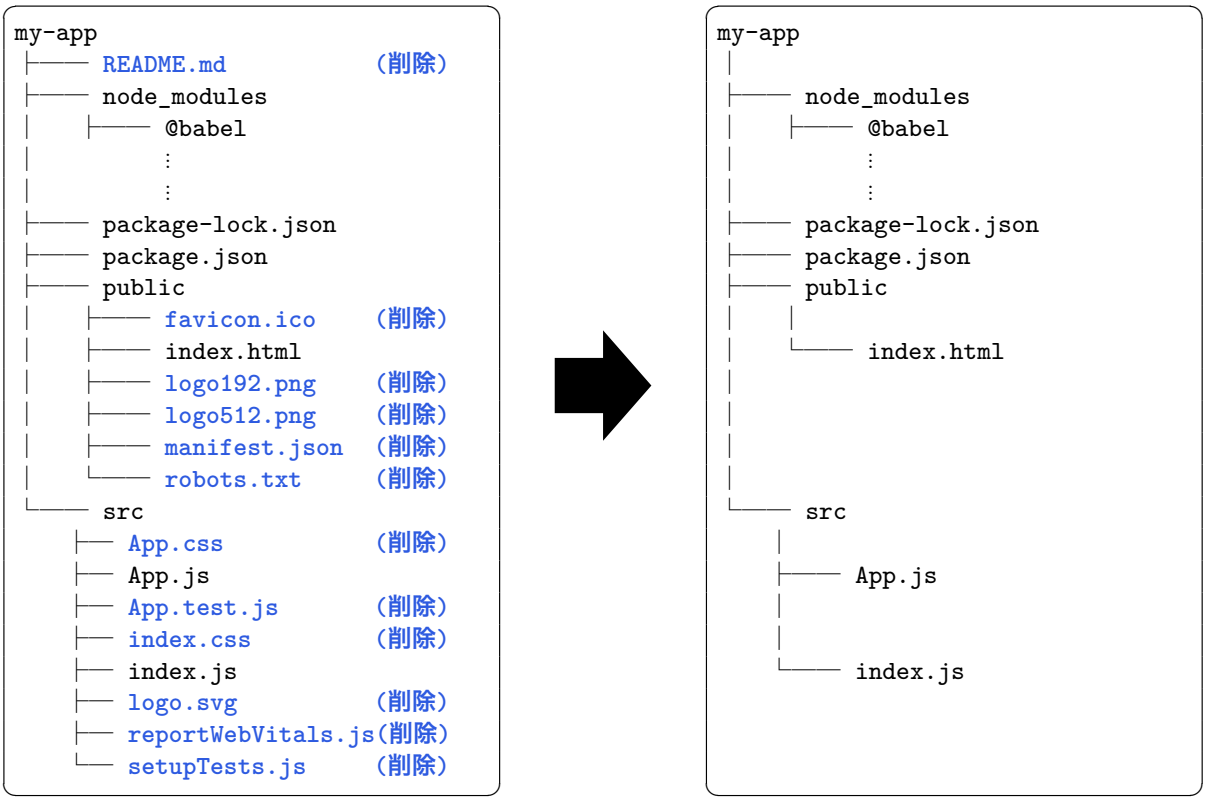

図 14　最小構成の React プロジェクト

上の図の左側が create-react-app が生成したディレクトリとファイルであり、右側が修正後のものである。`node_modules` ディレクトリにはインストールされたパッケージが保存され、`package-lock.json` 及び `package.json` ファイルにはパッケージの依存関係やバージョン情報が保存されているので削除してはいけない。残る `public/index.html`、`src/index.js` 及び `src/App.js` ファイルは以下の内容に変更する。

コード 5　public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

<body>には、id が root の 要素が置かれているだけである。

コード 6　src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

1〜3 行目の import 文により、React のコアライブラリと DOM ライブラリ、App.js ファイルで定義された<App>コンポーネントのインポートを行っている。そして、5〜10 行目の ReactDOM.render() メソッドで、<App>コンポーネントを index.html ファイルの 要素にレンダリングしている。

コード 7　src/App.js

```
function App() {
  return (
    <div className="App">
      <h1>react-geogebra Demo</h1>
    </div>
  );
}
export default App;
```

1〜7 行目で<App>コンポーネントを関数として定義している。8 行目の記述によって、index.js ファイルにおいて<App>コンポーネントが利用可能になる。この App.js ファイルを修正していくことが React アプリ開発のメインの作業になる。

4.5 GGB ワークシートの React アプリ化

既存の JavaScript ライブラリを React に対応させたものが npm パッケージとして The npm registry [8] で公開されている[*10]。そして、GeoGebra を React コンポーネントとして利用できるようにしたものが **react-geogebra** [9] である。4.4 節で用意した最小構成の React プロジェクトに次のように react-geogebra をインストールする。

```
$ cd ~/react/my-app
$ npm install react-geogebra --legacy-peer-deps --no-audit
```

--legacy-peer-deps オプションはパッケージの依存関係に問題があった場合に自動的に解決するためのもので、--no-audit オプションは脆弱性に関する警告を無視する[*11]ためのものである。

そして、2 節で作成した GeoGebra ワークシートを Web に埋め込むために、src/App.js ファイルを次のように修正する。

コード 8　src/App.js（修正）

```
1  import Geogebra from 'react-geogebra';
2  function App() {
3    return (
4      <div className="App">
5        <h1>react-geogebra Demo</h1>
6        <Geogebra
7          width="800" height="600" showToolBar allowStyleBar showMenuBar="false"
8          showAlgebraInput="false" appletOnLoad={ggbOnInit}
9        />
10       <button type="button" onClick={judgement}><big>判定</big></button>
11     </div>
12   );
13 }
14 export default App;
```

1 行目で react-geogebra から<Geogebra>コンポーネントをインポートし、6〜9 行目で「サイズ 800 × 600 ピクセル・ツールバー有り・スタイルバー有り・メニューバー無し・入力バー無し」で設置している。また、appletOnLoad で ggbOnInit 関数を指定することにより、<Geogebra> コン

[*10] ライブラリ名が MathJax の場合は react-mathjax のように、react- がパッケージ名の接頭辞として使用されることが多い。

[*11] npm が出力する脆弱性の警告は非常に多く、個別に対処するのはほとんど不可能であるため、ここではすべて無視する。

ポーネントがロードされた際に自動的に ggbOnInit 関数が実行される。そして、10 行目で 判定 ボタンを設置し、クリックすると judgement 関数が実行されるように onClick を設定している。

次に、GeoGebra ワークシートの「**グローバル Java スクリプト**」欄のコードを src/App.js ファイルに追加コピーする。

コード 9　src/App.js（追加）

```
15 function ggbOnInit() {
16     const ggbApplet = window.ggbApplet;
17     ggbApplet.setAxesVisible(false,false);
18     ggbApplet.setGridVisible(false);
19     ggbApplet.refreshViews();
20     ggbApplet.registerAddListener(ListenerA);
21     ggbApplet.registerRemoveListener(ListenerRM);
22     ggbApplet.registerRenameListener(ListenerRN);
23     objectList();
24 }
25 function ListenerA(P1) {
26     objectList();
27 }
28 function ListenerRM(P1) {
29     objectList();
30 }
31 function ListenerRN(P1, P2) {
32     objectList();
33 }
34 var Pname = [];
35 var Sname = [];
36 function objectList() {
37     const ggbApplet = window.ggbApplet;
38     ggbApplet.unregisterAddListener(ListenerA);
39     ggbApplet.unregisterRemoveListener(ListenerRM);
40     Pname = [];
41     Sname = [];
42     for (var i = 0; i < ggbApplet.getObjectNumber(); i++) {
43         var obj = ggbApplet.getObjectName(i);
44         if (ggbApplet.getObjectType(obj) === "point") Pname.push(obj);
45         else if (ggbApplet.getObjectType(obj) === "segment") Sname.push(obj);
46     }
47     ggbApplet.registerAddListener(ListenerA);
48     ggbApplet.registerRemoveListener(ListenerRM);
49 }
```

16 行目と 37 行目に const ggbApplet = window.ggbApplet; という行が追加されることに注意すること。17〜18 行目で座標軸とグリッドを非表示に設定し、19 行目でそれを描画エリアに反

映させている。

さらに、GeoGebra ワークシートの「**クリックして**」欄のコードを src/App.js ファイルに次のように judgement 関数として追加コピーする。

コード 10　src/App.js（追加）

```
50 function judgement(){
51   const ggbApplet = window.ggbApplet;
52   var connect = [];
53   for (var i = 0; i < Pname.length; i++) {
54       ggbApplet.evalCommand('tmpC=Circle(' + Pname[i] + ',0.5)');
55       var cnt = 0;
56       for (var j = 0; j < Sname.length; j++) {
57           ggbApplet.evalCommand('tmpP=Intersect(tmpC, ' + Sname[j] + ')');
58           if (ggbApplet.isDefined('tmpP')) cnt++;
59           ggbApplet.deleteObject('tmpP');
60       }
61       connect.push(cnt);
62       ggbApplet.deleteObject('tmpC');
63   }
64   for (var even = 0, odd = 0, k = 0; k < Pname.length; k++) {
65       ggbApplet.setCaption(Pname[k], connect[k]);
66       ggbApplet.setLabelStyle(Pname[k], 3);
67       if (connect[k] % 2 !== 0) {
68           ggbApplet.setColor(Pname[k], 255, 0, 0);
69           ggbApplet.setPointSize(Pname[k], 7);
70           odd++;
71       } else {
72           ggbApplet.setColor(Pname[k], 0, 0, 255);
73           ggbApplet.setPointSize(Pname[k], 5);
74           even++;
75       }
76   }
77   ggbApplet.evalCommand('odd=FormulaText("\\Large 奇点: ' + odd + '")');
78   ggbApplet.setCoords('odd', 0, 5.5);
79   ggbApplet.evalCommand('even=FormulaText("\\Large 偶点: ' + even + '")');
80   ggbApplet.setCoords('even', 3, 5.5);
81   if (odd === 0 || odd === 2) {
82     ggbApplet.evalCommand('result=FormulaText("\\Large\\Rightarrow 一筆書き可能")');
83     ggbApplet.setCoords('result', 0, 4.5);
84   } else {
85     ggbApplet.evalCommand('result=FormulaText("\\Large\\Rightarrow 一筆書き不可能")');
86     ggbApplet.setCoords('result', 0, 4.5);
87   }
88 }
```

最後に、package.json ファイルの最終行を以下のように修正する。

コード 11　package.json（修正）

```
38   },
39   "homepage": "."
40 }
```

これによって、生成される新しい index.html 内のパスが相対パスになり、Web サーバの任意のディレクトリに公開用ファイルを配置できるようになる。

以上の修正の後、ターミナルで次を実行すると、プロジェクトがトランスパイルされ公開用ファイルが my-app ディレクトリ内の build ディレクトリに生成される。

```
$ npm run build

> my-app@0.1.0 build
> react-scripts build

Creating an optimized production build...
Compiled successfully.
```

簡易 Web サーバを起動するために、build ディレクトリの中で python3 -m http.server & を実行後、Web ブラウザで http://localhost:8000 にアクセスすると、React アプリ化された GeoGebra ワークシートが表示される。

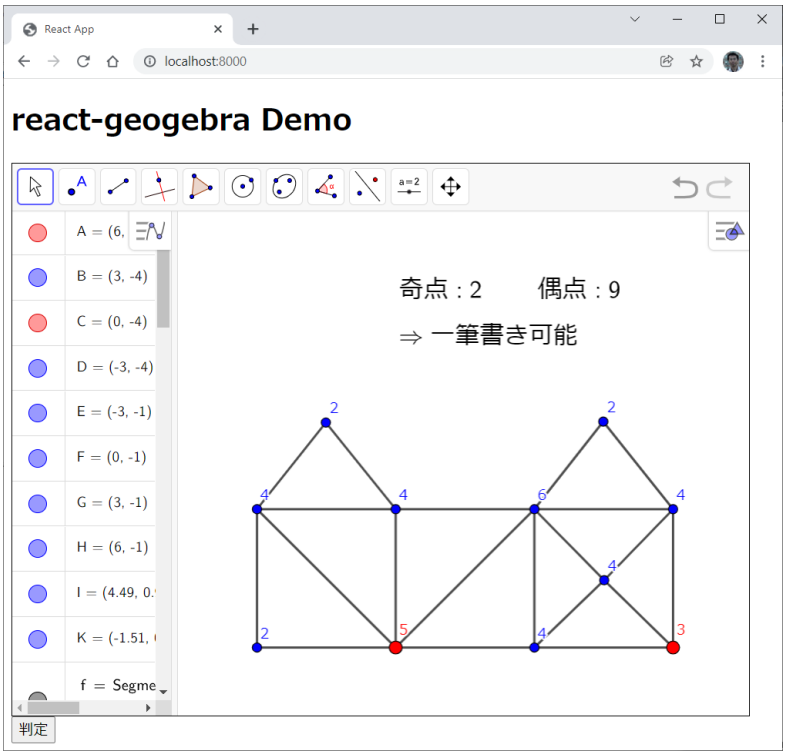

図 15　React アプリ化されたワークシート

5 おわりに

本稿では、GeoGebra ワークシートを React アプリ化して Web に埋め込む手法について解説した。本手法を用いることで GeoGebra のスクリプト入力ダイアログの束縛から離れ、任意のエディタでコードの編集が可能となる。これは GitHub などのバージョン管理システムでコードを管理できるようになることを意味する。また、トランスパイルにより JavaScript のエラーを実行前にチェックすることが可能になるため、JavaScript を用いた GeoGebra 教材の開発環境を改善できると考えられる。

今後の課題としては、WebAssembly を用いることにより、C/C++ などで開発されている既存の数学ソフトウェアと GeoGebra を連携できるようにすることなどが挙げられる。

謝辞 本研究は JSPS 科研費 JP17K01133 及び JP21K12157 の助成を受けたものである。

参考文献

[1] 藤本光史, How to create sliding puzzles on GeoGebra – A tutorial of GeoGebra Script –, 統計数理研究所共同研究リポート 396, 動的幾何学ソフトウェア GeoGebra の整備と普及 (3), (2018) 33–42.

[2] 藤本光史, How to make an app using JavaScript on GeoGebra, 統計数理研究所共同研究リポート 430, 動的幾何学ソフトウェア GeoGebra の整備と普及 (5), (2020) 34–47.

[3] Export to HTML for offline and export dialogue, `https://help.geogebra.org/topic/export-to-html-for-offline-and-export-dialogue`

[4] GeoGebra Materials platform, `https://www.geogebra.org/materials`

[5] GeoGebra Apps Embedding, `https://wiki.geogebra.org/en/Reference:GeoGebra_Apps_Embedding`

[6] React – A JavaScript library for building user interfaces, `https://reactjs.org/`

[7] Node.js, `https://nodejs.org/`

[8] The npm registry, `https://www.npmjs.com/`

[9] react-geogebra, `https://www.npmjs.com/package/react-geogebra`