2021年度　情報理工学実験II

実験テーマ：　　　　モバイル

実験日：

氏名：　　　高柳奏和

学生番号：　　　02190367

班名：

提出日：

課題1-1

|  |
| --- |
| <!-- 10個の10進数をそれぞれ16進数に変換し表示するプログラム -->  <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex110: 16進数への変換</title>  </head>  <body>  <script>  function to\_hex(decimal) {  //ここに10進数を16進数に変換する関数の中身を書くこと。  if (decimal>=0 && decimal<=255) return decimal.toString(16)  else return 0  }  //decimalsは10進数を10個格納する配列  var decimals = new Array(10);  decimals[0] = -10;  decimals[1] = 0;  decimals[2] = 10;  decimals[3] = 15;  decimals[4] = 16;  decimals[5] = 100;  decimals[6] = 200;  decimals[7] = 255;  decimals[8] = 256;  decimals[9] = 1000;  //10進数(0) = 0x00 のように与えられた10進数を16進数に変換して表示する  for (var i = 0; i < 10; i++)  document.write("10進数(" + decimals[i] + ") = 0x" + to\_hex(decimals[i]) + "<br>");  </script>  </body>  </head> |

テーブル

自動的に生成された説明

課題1-2

|  |
| --- |
| <!-- 16進数(1~F)の掛け算表を表示するプログラム -->  <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex120: 16進数の掛け算表</title>  </head>  <body>  <script>  //ここに課題1-1で作成した10進数→16進数に変換する関数を定義すること。  function to\_hex(decimal) {  //ここに10進数を16進数に変換する関数の中身を書くこと。  return decimal.toString(16).toUpperCase();  }  var n = 16; //二次元配列の行、列の要素数(16×16の配列を作る)  var tableData; //掛け算表のデータを格納するn×nの二次元配列データ  //掛け算表のデータを格納するn×nの二次元配列を作成するコードをここに書くこと。  //以下の手順で二次元配列tableData[n][n]を作成できる  //１．要素数nの配列tableData[n]をまず作成  //２．列の各番地tableData[i]に要素数nの配列を格納  var tableData = new Array(n);  for(var i=0;i<n;i++){  tableData[i] = new Array(n);  for(var j=0;j<n;j++){  if(i==0&&j==0) {  tableData[i][j]="＼";  }else if(i==0){  tableData[i][j]=to\_hex(j);  }else if(j==0){  tableData[i][j]=to\_hex(i);  }else{  tableData[i][j]=to\_hex(i\*j);  }  }  }  //掛け算表用のデータを二次元配列に格納するコードをここに書くこと(for文を用いること)。  //表の左上部分(0行0列目)には区切りを表す"＼"を格納すること  //表の上段部分(0行1~15列目)に1~15の16進数文字列を格納すること  //表の左端部分(1~15行0列目)に1~15の16進数文字列を格納すること  //残りのi行j列目には掛け算(i\*j)の結果の16進数文字列を格納すること  function generate\_table() {  // get the reference for the body  var body = document.getElementsByTagName("body")[0];  // creates a <table> element and a <tbody> element  var tbl = document.createElement("table");  var tblBody = document.createElement("tbody");  // creating all cells  for (var i = 0; i < n; i++) {  // creates a table row  var row = document.createElement("tr");  for (var j = 0; j < n; j++) {  // Create a <td> element and a text node, make the text  // node the contents of the <td>, and put the <td> at  // the end of the table row  var cell = document.createElement("td");  var cellText = document.createTextNode(tableData[i][j]);  cell.appendChild(cellText);  row.appendChild(cell);  }  // add the row to the end of the table body  tblBody.appendChild(row);  }  // put the <tbody> in the <table>  tbl.appendChild(tblBody);  // appends <table> into <body>  body.appendChild(tbl);  // sets the border attribute of tbl to 2;  tbl.setAttribute("border", "2");  }  generate\_table();  //表の出力  </script>  </body>  </html> |

パソコンの画面

中程度の精度で自動的に生成された説明

課題2-1

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex210: 背景色を書き換える</title>  <style>  body {  background-color: #ffffff;  }  </style>  <script type="text/javascript">  var color\_id = 0; // 0は初期色の白を示している  var max\_colors = 10;    var colors = new Array(max\_colors);  colors[0] = "#ffffff"; // colors[0]は必ず白とすること  colors[1] = "#ffff00"; // 黄色  colors[2] = "#0000ff"; // 青色  colors[3] = "#ff00ff"; // pink  colors[4] = "#00ff00"; // lightgreen  colors[5] = "#0ff0ff"; // lightblue  colors[6] = "#f0f0f0"; // pink  colors[7] = "#fedacb"; // brown  colors[8] = "#aaff00"; // yellow  colors[9] = "#000000"; // black  function changeBgForward() {  // var node = document.createElement('p');  // node.innerHTML = color\_id.toString();  // document.body.appendChild(node);  if(color\_id==max\_colors-1) {  color\_id=0;  document.body.style.background = colors[color\_id];  } else {  color\_id=color\_id+1;  document.body.style.background = colors[color\_id];  }  }  function changeBgBackward() {  if(color\_id==0) {  color\_id=max\_colors-1;  document.body.style.background = colors[color\_id];  } else {  color\_id=color\_id-1;  document.body.style.background = colors[color\_id];  }  }  </script>  </head>  <body>  <div id="main\_content">  <button id="forward" onclick="changeBgForward()">背景色を変更する（前方）</button>  <button id="backward" onclick="changeBgBackward()">背景色を変更する（後方）</button>  </div>  </body>  </html> |

前方へ色を変化させた結果が以下。



















課題2-2

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex220-sample: テキスト行を追加する</title>  <script>  var i = 0;  function addText(event){  i++;  var node = document.createElement('p')  node.id = "id: " + i.toString();  node.innerHTML = i.toString() + ': あ';  document.body.appendChild(node);  }  function deleteText(event) {  if(i>0)  {  var node = document.getElementById("id: " + i.toString());  document.body.removeChild(node);  i--;  }  }  window.addEventListener("load", function(event){  var addButton = document.getElementById("add-text");  var deleteButton = document.getElementById("delete-text");  //クリックイベントを監視  addButton.addEventListener("click", addText, false);  deleteButton.addEventListener("click", deleteText, false);  }, false);  </script>  </head>  <body>  <div id="box">  <button id="add-text">テキストを追加する</button>  <button id="delete-text">テキストを削除する</button>  </div>  </body>  </html> |

グラフィカル ユーザー インターフェイス, テキスト

自動的に生成された説明

グラフィカル ユーザー インターフェイス, テキスト, チャットまたはテキスト メッセージ

自動的に生成された説明

グラフィカル ユーザー インターフェイス, テキスト

自動的に生成された説明

課題3-1

|  |
| --- |
| //アプリケーション中に使う変数を定義  var timer;  var start;  var isStarted = false;  //DOMで操作する要素を代入  var startButton = document.getElementById('start');  var stopButton = document.getElementById('stop');  var resetButton = document.getElementById('reset');  var watch = document.querySelector('.stopwatch p');  //イベント監視  startButton.addEventListener('click', watchStart, false);  stopButton. addEventListener('click', watchStop, false);  resetButton.addEventListener('click', watchReset, false);  //開始ボタンのイベントハンドラー  function watchStart(){  if (! isStarted) {  // Date() はミリ秒単位の現在時刻を出力する関数  start = new Date();  // setIntervalは第２引数で指定した間隔（ミリ秒単位）ごとに  // 第１引数で指定した関数を呼び出す関数  timer = setInterval(updateWatch, 1000/10);  // ここでは 1/60 秒に１度 updateWatch を呼び出すと指定  isStarted = true;  }  }  //停止ボタンのイベントハンドラー（仕様(7)を満たすためには改造が必要）  function watchStop(){  if ( isStarted ) {  // clearIntervalはsetIntervalを停止する関数  clearInterval(timer);  isStarted = false;  }  }  //リセットボタンのイベントハンドラー（仕様(6)を満たすためには改造が必要）  function watchReset() {  if (!isStarted) {  watchStop();  watch.innerHTML = "00:00:00:0";  }  // DOM の watch ノードのHTML記述を更新することでゼロにリセット  }  //計測中の時刻計算用関数  function updateWatch() {  //現在時間を date に代入  var date = new Date();  //ここで diff という変数を定義し、現在時刻とスタートした時刻の差分を代入  //するコードを以下に書くこと。具体的には date と start オブジェクトの  //getTime() メソッドの返り値を用いて計算すること。 getTime() メソッドは、  //オブジェクト内に格納されているミリ秒単位の時刻を出力する。  var diff = date.getTime() - start;  var ms = Math.floor((diff % 1000)/100);  var sec = Math.floor(diff / 1000) % 60;  var min = Math.floor(diff / 60000);  var hour = Math.floor(diff / 360000);  //表示用に桁数を合わせるコードを以下に書くこと。  if (hour < 10) hour = "0" + String(hour);  if (min < 10) min = "0" + String(min);  if (sec < 10) sec = "0" + String(sec);  //DOMの watch ノードのHTML記述を書き換えることでスタートからの経過時間を  //表示するコードを以下に書くこと。  watch.innerHTML = hour + ":" + min + ":" + sec + ":" + ms;  } |

テキスト

自動的に生成された説明

課題4-1

|  |
| --- |
| <!-- 省略しない記法で書かれたプログラムを省略記法に書き直す課題 -->  <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex410: ajaxの実験</title>  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <script>  //大域変数の定義：左の状態 (false, true)と右の状態 (false, true)  var left\_state = false;  var right\_state = false;  $(function(){  //ボタン1が押されたら、left\_functionを呼び出す  $('#button1').click(function(){  //left\_stateがfalseならAjaxでダウンロードを試みる  if (left\_state == false) {  $.ajax('ex410-load.txt').done(function(data) {  $('#status-area1').text('読み込み成功');  $('#display-area1').html(data);  }).fail(function() {  $('#status-area1').text('読み込み失敗');  });  //状態の変更  left\_state = true;  } else {  //停止状態に移行  //text()メソッドは、DOMノードのテキストを変更する  $('#status-area1').text('');  $('#display-area1').text('');  left\_state = false;  }  } );  //ボタン2が押されたら、right\_functionを呼び出す  $('#button2').click(function() {  if(right\_state == false) {  $.ajax('ex410-load.html').done(function(data) {  $('#status-area2').text('読み込み成功');  $('#display-area2').html(data);  }).fail(function() {  $('#status-area2').text('読み込み失敗');  });  right\_state = true;  } else {  $('#status-area2').text('');  //html()メソッドは、DOMノードのテキストを変更する  $('#display-area2').html('');  right\_state = false;  }  }  );  })  </script>  </head>  <body bgcolor="#FFFFFF">  <table>  <tr>  <td>  <div style='clear:both;border: 2px dotted #080;margin:10px; padding:10px;'>  <p>  <button id='button1'>Ajax で取得したテキストファイルを表示</button>  </p>  <span id='status-area1'></span><br>  <div id='display-area1' style='border:1px solid #00f;padding:10px;font-size:20px;font-weight:bold; height: 100px'></div>  </div>  </td>  <td>  <div style='clear:both;border: 2px dotted #080;margin:10px; padding:10px;'>  <p>  <button id='button2'>Ajax で取得した HTML を表示</button>  </p>  <span id='status-area2'></span><br>  <div id='display-area2' style='border:1px solid #00f;padding:10px;font-size:20px;font-weight:bold; height: 100px'></div>  </div>  </td>  </tr>  </table>  </body>  </html> |

グラフィカル ユーザー インターフェイス が含まれている画像

自動的に生成された説明

課題4-2

|  |
| --- |
| <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex421: 天気情報取得</title>  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <script src="./mamewaza/mamewaza\_weather.min.js"></script>  <script>  jQuery(document).ready(init);  function init(){  jQuery('#region').change(select\_region);  }  function select\_region(){  var region = jQuery(this).children('option:selected').val();  var city\_name = jQuery(this).children('option:selected').text();  jQuery("#city").html("");  jQuery.mamewaza\_weather({  selector: '#city',  region: region,  layout: 'horizontal',  when: '7days',  explanation: true,  cssPath: './mamewaza/mamewaza\_weather.css'  });  jQuery('#city\_name').html(city\_name);  }  </script>  </head>  <body>  <select id="region">  <option value="">選択してください</option>  <option value="360000">徳島県</option>  <option value="370000">香川県</option>  <option value="380000">愛媛県</option>  <option value="390000">高知県</option>  </select>  <div>  四国地方の天気予報  <p id="city"></p>  </div>  </body>  </html> |

モニター画面に映るグラフ

自動的に生成された説明

モニター画面に映るゲーム画面

低い精度で自動的に生成された説明

モニター画面に映るグラフ

自動的に生成された説明

モニター, 屋内, 画面, 座る が含まれている画像

自動的に生成された説明

課題4-3

|  |
| --- |
| <!-- jQueryを利用したアニメーションの課題 -->  <!DOCTYPE HTML>  <html>  <head>  <meta charset="UTF-8">  <title>ex430: パラパラアニメーション</title>  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <style type="text/css"> /\*style内の書式はcss\*/  /\*myImageというID属性が付けられたHTML要素に対してスタイルを指定\*/  #myImage {  position: absolute; /\*配置する位置を絶対位置で指定  (absoluteの場合はブラウザウインドウの左上が基準位置(0,0)となる)\*/  top: 20px; /\*ウインドウ左上から20px下に配置\*/  left: 100px; /\*ウインドウ左上から100px右に配置\*/  -moz-user-select: none; /\*ダブルクリックやドラッグで選択されない  ようにする(Firefox版)\*/  -webkit-user-select: none; /\*(Safari, Chrome版)\*/  }  </style>  <script type="text/javascript">  var NumOfImage = 4; //パラパラアニメーションに用いる画像の総数  var INTERVAL = 200; //パラパラアニメーションの速度  //(画像が切り替わる時間間隔)  var currentImg = 0; //現在表示している画像の番号  //(デフォルトでは0番目の画像を表示)  var timerId; //clearTimeoutで使われる数値のID  var isAnimating = false; //パラパラアニメーションの再生/終了状態を  //表すフラグ  var stopped = false; //パラパラアニメーションの一時停止状態を  //表すフラグ  //readyメソッド：この関数はHTML文書の読み込みが完了した後に呼ばれる  $(function(){  //画像のプリロード：img要素を作成し, images/img[画像の番号].png  //の画像ファイルを読み込んでおく  for(var i=0; i<NumOfImage; i++)  $("<img>").attr("src", "images/img" + i + ".png");  //最初の画像をdiv要素に追加  $("<img id='myImage' width='80' height='120'>") //ID=myImageとした幅80px,  //高さ120pxのimg要素を作成  .attr("src", "images/img0.png") //img要素への0番目画像の  //割り当て  .appendTo("#myDiv"); //ID=myDivであるdiv要素に,  //作成したimg要素を追加  //(メソッドチェーンを利用)  //「スタート」ボタンをクリックしたときに呼ばれる関数  $("#startBtn").click(function(){  //パラパラアニメーション中なら終了させる  if (isAnimating) {  clearTimeout(timerId);  isAnimating = false;  $("#startBtn").val("スタート"); //ボタンのラベルを変更  //そうでなければアニメーション開始  }else{  isAnimating = true;  flipAnimate();  $("#startBtn").val("ストップ"); //ボタンのラベルを変更  }  });  });  //Webブラウザ上をダブルクリックしたときに呼ばれる関数  $(document).dblclick(function(event){  //以下の手順1, 2を参考にして, ダブルクリック時にimg要素が直線移動  //アニメーションを行うコードをここに書くこと  //手順1.パラパラアニメーションを一時停止  //現在img要素がパラパラアニメーション再生状態にあるかを判定  //もし再生中ならば, そのアニメーションを一時停止する  if(isAnimating) {  stopped=true;  clearTimeout(timerId);  }  //手順2.ID=myImageを持つimg要素に対して直線移動アニメーション効果を適用  //アニメーションの内容は, ダブルクリックを行った位置までimg要素(の中心部分)  //を直線移動させるものとする  $("#myImage").animate({  left: event.pageX-$("#myImage").width()/2,  top: event.pageY-$("#myImage").height()/2,  }, "slow", "swing", flipAnimate);  //animateメソッドを用いたアニメーション効果の書式は以下の通り  //[ID=myImageを持つimg要素].animate(params, [duration], [easing], [callback])  //[]で書かれた引数は省略できる  //第１引数：params … アニメーションで数値を変化させたいプロパティとその値を  // JSON型式  // {[プロパティ] : [変化後の値], [プロパティ] : [変化後の値] ...}で指定する  //第２引数：duration … アニメーションの動作時間  //第３引数：easing … 数値が変化する度合  //第４引数：callback … アニメーション終了時に呼ばれる関数  //上記の4パラメータのうち, paramsにはダブルクリックを行った位置までimg要素  //(の中心部分)を直線移動させるよう, 適切なプロパティと数値を設定すること  //ヒント1：leftプロパティ及びtopプロパティを利用する  //ヒント2：ダブルクリックを行った位置はdbclick関数の引数event内に格納されて  //おり, その座標(x, y)はevent.pageX, event.pageYで取得できる  //ヒント3：img要素の幅と高さはそれぞれ, [ID=myImageを持つimg要素].width(),  // [ID=myImageを持つimg要素].height()で取得できる  //callback内には手順1の一時停止状態を解除させるコードを書くこと  });  //パラパラマンガのアニメーションを行う関数  function flipAnimate() {  if(stopped) {  stopped=false;  }  if(!isAnimating)return;  //表示したい番号の画像をimg要素に設定  $("#myImage").attr("src", "images/img" + currentImg + ".png");  currentImg++; //次に表示したい画像の番号を設定(ローテーションにする)  if(currentImg >= NumOfImage) currentImg = 0;  //INTERVALで指定したミリ秒後に再びflipAnimate()関数を呼び出す  timerId = setTimeout("flipAnimate()", INTERVAL);  }  </script>  </head>  <body>  <form name="form1"><p>  <input type="button" id="startBtn" value="スタート">  </p></form>  <div id="myDiv"></div>  </body>  </html> |

グラフィカル ユーザー インターフェイス, アプリケーション, Teams

自動的に生成された説明

挿絵, 時計 が含まれている画像

自動的に生成された説明

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

課題5-1

|  |
| --- |
| <!DOCTYPE html>  <html lang="ja">  <head>  <meta charset="UTF-8">  <title>ex510: jQuery Mobileのフォームページのサンプル</title>  <meta name="viewport" content="width=device-width,initial-scale=1">  <!-- 2020年11月18日現在のjQuery Mobileの最新安定板は1.4.5 -->  <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.css" />  <!-- jQuery 最新版を導入 -->  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <!-- 2020年11月18日現在のjQuery Mobileの最新安定板は1.4.5 -->  <script src="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.js"></script>  </head>  <body>  <div data-role="page">  <div data-role="header">  <h1>ワクチン接種後アンケート</h1>  </div>  <div data-role="content">  <form id="inquiry\_form" method="post">  <div data-role="fieldcontain">  <fieldtest data-role="controlgroup">  <input type="radio" name="radio-choice" id="radio-choice-1" value="choise-1" />  <label for="radio-choice-1">1回目</label>  <input type="radio" name="radio-choice" id="radio-choice-2" value="choise-2" />  <label for="radio-choice-2">2回目</label>  </fieldset>  </div>  <br>  <br>  <div data-role="fieldcontain">  <label for="inquiery">副反応があった方は記入をお願いします</label>  <br>  <textarea name="inquiery" id="inquirery"></textarea>  </div>  <br>  <input type="submit" id="submit" data-thema="b" value="送信">  </form>  </div>  </div>  </body>  </html> |

テキスト

自動的に生成された説明

課題5-2

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>ex521: スワイプで左右に遷移する(モバイルデバイス向け)</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <!-- 2020年11月18日現在のjQuery Mobileの最新安定板は1.4.5 -->  <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.css" />  <!-- jQuery 最新版を導入 -->  <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <!-- 2020年11月18日現在のjQuery Mobileの最新安定板は1.4.5 -->  <script src="https://ajax.googleapis.com/ajax/libs/jquerymobile/1.4.5/jquery.mobile.min.js"></script>  <script>  // #page1swipe上で右から左にスワイプした時の動作（page2view()を呼び出す）  $(document).on('pagecreate', '#page1', function() {  $('#page1swipe').on('swipeleft', page2view);  });  // #page2swipe上で左から右にスワイプした時の動作（page1view()を呼び出す）  // #page2swipe上で右から左にスワイプした時の動作（page3view()を呼び出す）  $(document).on('pagecreate', '#page2', function() {  $('#page2swipe').on('swiperight', page1view);  $('#page2swipe').on('swipeleft', page3view);  });  // #page3swipe上で左から右にスワイプした時の動作（page2view()を呼び出す）  $(document).on('pagecreate', '#page3', function() {  $('#page3swipe').on('swiperight', page2view);  });  // ページ1(#page1)へ進む  function page1view() {  $.mobile.changePage('#page1');  }  // ページ2(#page2)へ進む  function page2view() {  $.mobile.changePage('#page2');  }  // ページ3(#page3)へ進む  function page3view() {  $.mobile.changePage('#page3');  }  </script>  </head>  <body>  <!-- ページ1 -->  <div data-role="page" id="page1" data-theme="a">  <div data-role="header">  <!-- ナビゲーションバー(左右ページへのリンク) -->  <div data-role="navbar" class="navbar">  <ul>  <li>  <a href="#page2" data-icon="arrow-r"></a>  </li>  </ul>  </div>  </div>  <div data-role="content" id="page1swipe" style="min-height:500px;">  <p>ここはページ1です。</p>  <p>＞ボタンをクリックするとページ2に移動します。</p>  </div>  </div>  <!-- ページ2 -->  <div data-role="page" id="page2" data-theme="a">  <div data-role="header">  <!-- ナビゲーションバー(左右ページへのリンク) -->  <div data-role="navbar" class="navbar">  <ul>  <li>  <a href="#page1" data-icon="arrow-l"></a>  </li>  <li>  <a href="#page3" data-icon="arrow-r"></a>  </li>  </ul>  </div>  </div>  <div data-role="content" id="page2swipe" style="min-height:500px;">  <p>ここはページ2です。</p>  <p>＞ボタンをクリックするとページ3に移動します。</p>  <p>＜ボタンをクリックするとページ1に移動します。</p>  </div>  </div>  <!-- ページ3 -->  <div data-role="page" id="page3" data-theme="a">  <div data-role="header">  <!-- ナビゲーションバー(左右ページへのリンク) -->  <div data-role="navbar" class="navbar">  <ul>  <li>  <a href="#page2" data-icon="arrow-l"></a>  </li>  </ul>  </div>  </div>  <div data-role="content" id="page3swipe" style="min-height:500px;">  <p>ここはページ3です。</p>  <p>＜ボタンをクリックするとページ2に移動します。</p>  </div>  </div>  </body>  </html> |

テキスト が含まれている画像

自動的に生成された説明テキスト

自動的に生成された説明

テキスト

自動的に生成された説明

PCとモバイルデバイス双方に対応するサイトを構築する際に気を付けるべきこと

Webで使える部品のうちすべてをそのまま使うのではなく、スマホ用に別の部品を使ったりする（スクロールバー、ポップアップなど）

幅を一定ではなくリキッドウィンドウを使って柔軟に変えられるようにすべき。

課題6-1

Firefox38.8.0 ESR

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

Firefox

テキスト

中程度の精度で自動的に生成された説明

Chrome

テキスト が含まれている画像

自動的に生成された説明

課題6-2

仕様１

|  |
| --- |
|  |

仕様２

|  |
| --- |
|  |