|  |  |
| --- | --- |
| \*\*\*\*\* Bingo Sheet ① \*\*\*\*\*  function createColumn(col) {  const source = [];  for(let i = 0; i < 15; i++) {  source[i] = i + 1 + (15 \* col);  }  const column = [];  for(let i = 0; i < 5; i++) {  column[i] =  source.splice(Math.floor(Math.random() \* source.length ), 1)[0];  }  return column;  }  function createColmuns() {  const columns = [];  for(let i = 0; i < 5; i++) {  columns[i] = createColumn(i);  }  columns[2][2] = "Free";  return columns;  } | \*\*\*\*\* Bingo Sheet ② \*\*\*\*\*  function createBingo(columns) {  const bingo = [];  for(let row = 0; row < 5; row++) {  bingo[row] = [];  for(let col = 0; col < 5; col++) {  bingo[row][col] = columns[col][row];  }  }  return bingo;  }  function renderBingo(bingo) {  for(let row = 0; row < 5; row++) {  const tr = document.createElement('tr');  for(let col = 0; col < 5; col++) {  const td = document.createElement('td');  td.textContent = bingo[row][col];  tr.appendChild(td);  }  document.querySelector('tbody').appendChild(tr);  }  }  const columns = createColmuns();  const bingo = createBingo(columns);  renderBingo(bingo); |
| \*\*\*\*\* Stop Watch ① \*\*\*\*\*  const timer = document.getElementById('timer');  const start = document.getElementById('start');  const stop = document.getElementById('stop');  const reset = document.getElementById('reset');  let startTime;  let timeoutId;  let elapsedTime = 0;  function countUp() {  const d =  　　　　　　new Date(Date.now() - startTime + elapsedTime );  const m = String(d.getMinutes()).padStart(2, '0');  const s = String(d.getSeconds()).padStart(2, '0');  const ms = String(d.getMilliseconds()).padStart(3, '0');  timer.textContent = `${m}:${s}:${ms}`;  timeoutId = setTimeout(() => {  countUp();  }, 10);  } | \*\*\*\*\* Stop Watch ② \*\*\*\*\*  function setButtonStateInitial() {  start.disabled = false;  stop.disabled = true;  reset.disabled = true;  }  function setButtonStateRunning() {  start.disabled = true;  stop.disabled = false;  reset.disabled = true;  }  function setButtonStateStopped() {  start.disabled = false;  stop.disabled = true;  reset.disabled = false;  }  setButtonStateInitial(); |
| \*\*\*\*\* Stop Watch ③ \*\*\*\*\*  start.addEventListener('click', () => {  setButtonStateRunning();  startTime = Date.now();  countUp();  });  stop.addEventListener('click', () => {  setButtonStateStopped();  clearTimeout(timeoutId);  elapsedTime **+=** Date.now() - startTime;  });  reset.addEventListener('click', () => {  setButtonStateInitial();  timer.textContent = '00:00:000';  elapsedTime = 0;  }); |  |
| \*\*\*\*\* Calender ① \*\*\*\*\*  const today = new Date();  let year = today.getFullYear();  let month = today.getMonth();  function getCalenderHead() {  const dates = [];  **//今月の0日目を指定することで先月の末日を取得**  const d = new Date(year, month, 0).getDate();  **//今月初日の曜日の数値を取得（日曜0....土曜6）**  const n = new Date(year, month, 1).getDay();  for(let i = 0; i < n; i++) {  dates.unshift( {  date: d - i,  isTday: false,  isDisabled : true,  } );  }  return dates;  } | \*\*\*\*\* Calender ② \*\*\*\*\*  function getCalenderTail() {  const dates = [];  **//翌月の0日目を指定することで今月の末日を取得する事が出来る**  const lastDay = new Date(year, month + 1, 0).getDay();  for(let i = 1; i < 7 - lastDay; i++) {  dates.push({  date: i, isToday: false, isDisabled: true,  });  }  return dates;  }  function getCalenderBody() {  const dates = []; // date:日付、day:曜日  **//翌月の0日目を指定することで今月の末日を取得する事が出来る**  const lastDate = new Date(year, month + 1, 0).getDate();  for(let i = 1; i <= lastDate; i++) {  dates.push({  date: i, isToday: false, isDisabled: false,  });  }  if(year === today.getFullYear() && month === today.getMonth()) {  dates[today.getDate() - 1].isToday = true;  }  return dates;  } |
| \*\*\*\*\* Calender ③ \*\*\*\*\*  function clearCalender() {  const tbody = document.querySelector('tbody');    while(tbody.firstChild) {  tbody.removeChild(tbody.firstChild);  }  }  function renderWeeks() {  //スプレッド構文...で全ての配列を展開して格納  const dates = [  ...getCalenderHead(),  ...getCalenderBody(),  ...getCalenderTail(),  ];  const weeks = [];  const weeksCount = dates.length / 7;  for(let i = 0; i < weeksCount; i++) {  weeks.push(dates.splice(0, 7));  } | \*\*\*\*\* Calender ④ \*\*\*\*\*  weeks.forEach( week => {  const tr = document.createElement('tr');  week.forEach( date => {  const td = document.createElement('td');  td.textContent = date.date;  if(date.isToday) {  td.classList.add('today');  }  if(date.isDisabled) {  td.classList.add('disabled');  }  tr.appendChild(td);  });  document.querySelector('tbody').appendChild(tr);  });  }  function renderTitle() {  const title =  　　　　　　　`${year}/${String(month + 1).padStart(2, "0")}`;  document.getElementById('title').textContent = title;  }  function createCalender() {  clearCalender();  renderTitle();  renderWeeks();  } |
| \*\*\*\*\* Calender ⑤ \*\*\*\*\*    document.getElementById('prev').addEventListener('click', function() {  month --;  if( month < 0 ) {  year --;  month = 11;  }  createCalender();  });  document.getElementById('next').addEventListener('click', function() {  month ++;  if( month > 11 ) {  year ++;  month = 0;  }  createCalender();  });  document.getElementById('today').addEventListener('click', function() {  year = today.getFullYear();  month = today.getMonth();  createCalender();  });  createCalender(); |  |
| \*\*\*\*\* Typing Game ① \*\*\*\*\*  function setWord() {  **//spliceメソッドの返り値は結果が一つでも配列なので[0]で指定**  word =  　words.splice(Math.floor(Math.random() \* words.length), 1)[0];  target.textContent = word;  loc = 0;  }  const words = [ 'red', 'blue', 'pink', ];  let word;  let loc = 0;  let startTime;  let isPlaying = false;  const target = document.getElementById('target');  document.addEventListener('click', () => {  if(isPlaying === true) {  return;  }  isPlaying = true;  startTime = Date.now();  setWord();  }); | \*\*\*\*\* Typing Game ② \*\*\*\*\*  document.addEventListener('keydown', function(e) {  if(e.key !== word[loc]){  return;  }  loc++;  target.textContent =  　　　　　　　　　 '\_'.repeat(loc) + word.substring(loc);  if(loc === word.length) {  if(words.length === 0) {  const elapsedTime =  　　　　　　　　((Date.now() - startTime) / 1000 ).toFixed(2);  const result = document.getElementById('result');  result.textContent =  　　　　　　　　　　　　　`Finished! ${elapsedTime} seconds!`;  return;  }  setWord();  }  }); |
| \*\*\*\*\* Slideshow ① \*\*\*\*\*  const images = [　'Myslideshow/img/pic00.png',　'Myslideshow/img/pic01.png',  'Myslideshow/img/pic02.png',　'Myslideshow/img/pic03.png',  'Myslideshow/img/pic04.png',　'Myslideshow/img/pic05.png',  'Myslideshow/img/pic06.png',　 'Myslideshow/img/pic07.png',  ];  **let currentIndex = 0;**  const mainImage = document.getElementById('main');  mainImage.src = images[currentIndex];  images.forEach( (image, index) => {  const img = document.createElement('img');  img.src = image;  const li = document.createElement('li');  if(index === currentIndex) {  li.classList.add('current');  }  li.addEventListener('click', () => {  mainImage.src = image;  const thumbnails =  　　　　　　　　　　　document.querySelectorAll('.thumbnails > li');  thumbnails[**currentIndex**].classList.remove('current');  currentIndex = index;  thumbnails[**currentIndex**].classList.add('current');  });  li.appendChild(img);  document.querySelector('.thumbnails').  appendChild(li);  }); | \*\*\*\*\* Slideshow ② \*\*\*\*\*  next.addEventListener('click', () => {  let target = currentIndex + 1;  if(target === images.length) {  target = 0;  }  const thumbnails =  　　　　　　　 document.querySelectorAll('.thumbnails > li');  thumbnails[target].click();  });  const prev = document.getElementById('prev');  prev.addEventListener('click', () => {  let target = currentIndex - 1;  if(target < 0) {  target = images.length - 1;  }  const thumbnails =  　　　 document.querySelectorAll('.thumbnails > li');  thumbnails[target].click();  }); |
| \*\*\*\*\* Slideshow ③ \*\*\*\*\*  let timeoutId;  function playSlideshow() {  timeoutId = setTimeout(() => {  next.click();  playSlideshow();  }, 1000);  }  let isPlaying = false;  const play = document.getElementById('play');  play.addEventListener('click', () => {  if(isPlaying === false) {  playSlideshow();  play.textContent = 'Pause';  } else {  clearTimeout(timeoutId);  play.textContent = 'Play';  }  isPlaying = !isPlaying;  }); |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |