Exercises ▼ Get Certified ▼ PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R JQUERY DJANGO TYPESCRIPT NODEJS MYSQL HTML Graphics Graphics HOME HTML Plotting Game Controllers Plot Graphics Plot Canvas Plot Plotly Previous Next 🕽 Plot Chart.js Plot Google Plot D3.js Push the buttons to move the red square: Google Maps Maps Intro Maps Basic Maps Overlays Maps Events Maps Controls Maps Types Maps Reference UP SVG Tutorial SVG Intro LEFT RIGHT SVG in HTML SVG Rectangle DOWN SVG Circle SVG Ellipse SVG Line SVG Polygon Get in Control SVG Polyline SVG Path Now we want to control the red square. SVG Text SVG Stroking Add four buttons, up, down, left, and right. SVG Filters Intro Write a function for each button to move the component in the selected direction. SVG Blur Effects SVG Drop Shadows Make two new properties in the component constructor, and call them speedX and speedY. These properties are being used as speed indicators. SVG Linear Add a function in the component constructor, called newPos(), which uses the speedX and speedY properties to change the component's position. **SVG Radial** SVG Examples The newpos function is called from the updateGameArea function before drawing the component: SVG Reference Canvas Tutorial Example Canvas Intro Canvas Drawing <script> Canvas Coordinates function component(width, height, color, x, y) { Canvas Lines this.width = width; this.height = height; Canvas Shapes this.speedX = 0; Canvas Rectangles this.speedY = 0; Canvas Circles this.x = x;Canvas Curves this.y = y;Canvas Gradients this.update = function() { Canvas Text ctx = myGameArea.context; Canvas Images ctx.fillStyle = color; ctx.fillRect(this.x, this.y, this.width, this.height); Canvas Clock this.newPos = function() { Clock Intro this.x += this.speedX; Clock Face this.y += this.speedY; Clock Numbers Clock Hands Clock Start function updateGameArea() { HTML Game myGameArea.clear(); myGamePiece.newPos(); Game Intro myGamePiece.update(); Game Canvas Game Components Game Controllers function moveup() { Game Obstacles myGamePiece.speedY -= **1**; Game Score Game Images function movedown() { Game Sound myGamePiece.speedY += 1; Game Gravity Game Bouncing Game Rotation function moveleft() { Game Movement myGamePiece.speedX -= 1; function moveright() { myGamePiece.speedX += 1; </script> <button onclick="moveup()">UP</button> <button onclick="movedown()">DOWN</button> <button onclick="moveleft()">LEFT</button> <button onclick="moveright()">RIGHT</button> Try it Yourself » **Stop Moving** If you want, you can make the red square stop when you release a button. Add a function that will set the speed indicators to 0. To deal with both normal screens and touch screens, we will add code for both devices: Example function stopMove() { myGamePiece.speedX = 0;myGamePiece.speedY = 0;</script> <button onmousedown="moveup()" onmouseup="stopMove()" ontouchstart="moveup()">UP</button> <button onmousedown="movedown()" onmouseup="stopMove()" ontouchstart="movedown()">DOWN</button> <button onmousedown="moveleft()" onmouseup="stopMove()" ontouchstart="moveleft()">LEFT</button> <button onmousedown="moveright()" onmouseup="stopMove()" ontouchstart="moveright()">RIGHT</button> Try it Yourself » Keyboard as Controller We can also control the red square by using the arrow keys on the keyboard. Create a method that checks if a key is pressed, and set the key property of the myGameArea object to the key code. When the key is released, set the key property to false: Example var myGameArea = { canvas : document.createElement("canvas"), start : function() { this.canvas.width = 480; this.canvas.height = 270; this.context = this.canvas.getContext("2d"); document.body.insertBefore(this.canvas, document.body.childNodes[0]); this.interval = setInterval(updateGameArea, 20); window.addEventListener('keydown', function (e) { myGameArea.key = e.keyCode; window.addEventListener('keyup', function (e) { myGameArea.key = false; clear : function(){ this.context.clearRect(0, 0, this.canvas.width, this.canvas.height); Then we can move the red square if one of the arrow keys are pressed: Example function updateGameArea() { myGameArea.clear(); myGamePiece.speedX = 0;myGamePiece.speedY = 0;if (myGameArea.key && myGameArea.key == 37) {myGamePiece.speedX = -1; } if (myGameArea.key && myGameArea.key == 39) {myGamePiece.speedX = 1; } if (myGameArea.key && myGameArea.key == 38) {myGamePiece.speedY = -1; } if (myGameArea.key && myGameArea.key == 40) {myGamePiece.speedY = 1; } myGamePiece.newPos(); myGamePiece.update(); Try it Yourself » Multiple Keys Pressed What if more than one key is pressed at the same time? In the example above, the component can only move horizontally or vertically. Now we want the component to also move diagonally. Create a keys array for the myGameArea object, and insert one element for each key that is pressed, and give it the value remains true untill the key is no longer pressed, the value becomes false in the keyup event listener function:

your professional

To All Current & Future Courses

COLOR PICKER

All Courses & Certificates

Shop now!

Does not include Bootcamps & Subscriptions

> From 7th of July 2023 -1st of August 2023

development!

Get Lifelong Access

Get Full Access

Save \$1118

Invest in

Tutorials ▼ Exercises ▼ Get Certified ▼ Services ▼ HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R JQUERY DJANGO TYPESCRIPT NODEJS MYSQL canvas : document.createElement("canvas"), HTML Graphics start : function() { Graphics HOME this.canvas.width = 480; this.canvas.height = 270; HTML Plotting this.context = this.canvas.getContext("2d"); document.body.insertBefore(this.canvas, document.body.childNodes[0]); Plot Graphics this.interval = setInterval(updateGameArea, 20); Plot Canvas window.addEventListener('keydown', function (e) { Plot Plotly myGameArea.keys = (myGameArea.keys || []); Plot Chart.js myGameArea.keys[e.keyCode] = true; Plot Google window.addEventListener('keyup', function (e) { Plot D3.js myGameArea.keys[e.keyCode] = false; Google Maps Maps Intro clear : function(){ Maps Basic this.context.clearRect(0, 0, this.canvas.width, this.canvas.height); Maps Overlays Maps Events

```
Maps Controls
                             function updateGameArea() {
Maps Types
                              myGameArea.clear();
Maps Reference
                              myGamePiece.speedX = 0;
                              myGamePiece.speedY = 0;
SVG Tutorial
                              if (myGameArea.keys && myGameArea.keys[37]) {myGamePiece.speedX = -1; }
                              if (myGameArea.keys && myGameArea.keys[39]) {myGamePiece.speedX = 1; }
SVG Intro
SVG in HTML
                              if (myGameArea.keys && myGameArea.keys[38]) {myGamePiece.speedY = -1; }
                              if (myGameArea.keys && myGameArea.keys[40]) {myGamePiece.speedY = 1; }
SVG Rectangle
                              myGamePiece.newPos();
SVG Circle
                              myGamePiece.update();
SVG Ellipse
SVG Line
SVG Polygon
                            Try it Yourself »
SVG Polyline
SVG Path
SVG Text
SVG Stroking
                          Using The Mouse Cursor as a Controller
SVG Filters Intro
SVG Blur Effects
SVG Drop Shadows
                          If you want to control the red square by using the mouse cursor, add a method in myGameArea object that updates the x and y coordinates of the mouse cursor:.
SVG Linear
SVG Radial
                          Example
SVG Examples
SVG Reference
                            var myGameArea = {
Canvas Tutorial
                              canvas : document.createElement("canvas"),
                              start : function() {
Canvas Intro
                                this.canvas.width = 480;
Canvas Drawing
                                this.canvas.height = 270;
Canvas Coordinates
                                this.canvas.style.cursor = "none"; //hide the original cursor
Canvas Lines
                                this.context = this.canvas.getContext("2d");
Canvas Shapes
                                document.body.insertBefore(this.canvas, document.body.childNodes[0]);
                                this.interval = setInterval(updateGameArea, 20);
Canvas Rectangles
                                window.addEventListener('mousemove', function (e) {
Canvas Circles
                                  myGameArea.x = e.pageX;
Canvas Curves
                                  myGameArea.y = e.pageY;
Canvas Gradients
Canvas Text
Canvas Images
                              clear : function(){
                                this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
Canvas Clock
Clock Intro
Clock Face
Clock Numbers
Clock Hands
                           Then we can move the red square using the mouse cursor:
Clock Start
HTML Game
                          Example
Game Intro
Game Canvas
                            function updateGameArea() {
Game Components
                              myGameArea.clear();
Game Controllers
                              if (myGameArea.x && myGameArea.y) {
Game Obstacles
Game Score
Game Images
                              myGamePiece.update();
Game Sound
Game Gravity
Game Bouncing
                            Try it Yourself »
Game Rotation
Game Movement
                          Example
                            var myGameArea = {
                              start : function() {
                             clear : function(){
                          Example
                              myGameArea.clear();
                             myGamePiece.update();
                           Try it Yourself »
                          Example
                            function startGame() {
                              myGameArea.start();
                          Example
                            var myGameArea = {
                             canvas : document.createElement("canvas"),
                              start : function() {
                               this.canvas.width = 480;
                                this.canvas.height = 270;
                                this.context = this.canvas.getContext("2d");
                                document.body.insertBefore(this.canvas, document.body.childNodes[0]);
                                this.interval = setInterval(updateGameArea, 20);
```

window.addEventListener('mousedown', function (e) {

window.addEventListener('mouseup', function (e) {

window.addEventListener('touchstart', function (e) {

window.addEventListener('touchend', function (e) {

this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);

myGameArea.x = e.pageX; myGameArea.y = e.pageY;

myGameArea.x = false; myGameArea.y = false;

myGameArea.x = e.pageX; myGameArea.y = e.pageY;

myGameArea.x = false; myGameArea.y = false;

clear : function(){

```
myGamePiece.x = myGameArea.x;
      myGamePiece.y = myGameArea.y;
Touch The Screen to Control The Game
We can also control the red square on a touch screen.
Add a method in the myGameArea object that uses the x and y coordinates of where the screen is touched:
    canvas : document.createElement("canvas"),
      this.canvas.width = 480;
      this.canvas.height = 270;
      this.context = this.canvas.getContext("2d");
      document.body.insertBefore(this.canvas, document.body.childNodes[0]);
      this.interval = setInterval(updateGameArea, 20);
      window.addEventListener('touchmove', function (e) {
        myGameArea.x = e.touches[0].screenX;
        myGameArea.y = e.touches[0].screenY;
      this.context.clearRect(0, 0, this.canvas.width, this.canvas.height);
Then we can move the red square if the user touches the screen, by using the same code as we did for the mouse cursor:
  function updateGameArea() {
   if (myGameArea.x && myGameArea.y) {
      myGamePiece.x = myGameArea.x;
      myGamePiece.y = myGameArea.y;
Controllers on The Canvas
We can also draw our own buttons on the canvas, and use them as controllers:
   myGamePiece = new component(30, 30, "red", 10, 120);
    myUpBtn = new component(30, 30, "blue", 50, 10);
    myDownBtn = new component(30, 30, "blue", 50, 70);
    myLeftBtn = new component(30, 30, "blue", 20, 40);
    myRightBtn = new component(30, 30, "blue", 80, 40);
Add a new function that figures out if a component, in this case a button, is clicked.
Start by adding event listeners to check if a mouse button is clicked (mousedown and mouseup). To deal with touch screens, also add event listeners to check if the screen is clicked on (touchstart and touchend):
```

Tutorials ▼ Exercises ▼ Get Certified ▼ Services ▼ Menu ▼ Sign Up

0 Q Q

FORUM | ABOUT

```
HTML Graphics
Graphics HOME
HTML Plotting
Plot Graphics
```

Plot Canvas

Plot Chart.js

Plot Google

Plot Plotly

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP BOOTSTRAP HOW TO W3.CSS C C++ C# REACT R JQUERY DJANGO TYPESCRIPT NODEJS MYSQL

```
Plot D3.js

Google Maps

Maps Intro

Maps Basic

Maps Overlays

Maps Events

Maps Controls

Maps Types

Maps Reference
```

Maps Types
Maps Reference

SVG Tutorial

SVG Intro

SVG in HTML

SVG Rectangle

SVG Circle

SVG Ellipse

SVG Line

SVG Ellipse
SVG Line
SVG Polygon
SVG Polyline
SVG Path
SVG Text
SVG Stroking
SVG Filters Intro
SVG Blur Effects
SVG Drop Shadows
SVG Linear
SVG Radial
SVG Examples
SVG Reference

Canvas Tutorial
Canvas Intro
Canvas Drawing
Canvas Coordinates
Canvas Lines
Canvas Shapes
Canvas Rectangles
Canvas Circles
Canvas Curves
Canvas Gradients

Canvas Images

Canvas Clock

Clock Intro

Clock Face

Clock Numbers

Clock Hands

Clock Start

Canvas Text

Previous

HTML Game
Game Intro
Game Canvas
Game Components

Game Controllers

Game Obstacles

Game Score

Game Images

Game Sound

Game Gravity

Game Bouncing
Game Rotation
Game Movement

```
In the updateGameArea function, we take the neccessarry actions if one of the blue buttons is clicked:
Example
 function component(width, height, color, x, y) {
   this.width = width;
  this.height = height;
   this.speedX = 0;
   this.speedY = 0;
   this.x = x;
   this.y = y;
   this.update = function() {
     ctx = myGameArea.context;
    ctx.fillStyle = color;
    ctx.fillRect(this.x, this.y, this.width, this.height);
   this.clicked = function() {
     var myleft = this.x;
     var myright = this.x + (this.width);
     var mytop = this.y;
     var mybottom = this.y + (this.height);
     var clicked = true;
    if ((mybottom < myGameArea.y) || (mytop > myGameArea.y) || (myright < myGameArea.x) || (myleft > myGameArea.x)) {
       clicked = false;
     return clicked;
 function updateGameArea() {
   myGameArea.clear();
   if (myGameArea.x && myGameArea.y) {
    if (myUpBtn.clicked()) {
       myGamePiece.y -= 1;
    if (myDownBtn.clicked()) {
       myGamePiece.y += 1;
    if (myLeftBtn.clicked()) {
       myGamePiece.x += -1;
    if (myRightBtn.clicked()) {
       myGamePiece.x += 1;
   myUpBtn.update();
   myDownBtn.update();
   myLeftBtn.update();
   myRightBtn.update();
   myGamePiece.update();
 Try it Yourself »
```

Log in to track progress

Upgrade Newsletter **Get Certified** Spaces **Report Error Top Tutorials Top References** Top Examples **Get Certified HTML** Tutorial HTML Reference HTML Examples HTML Certificate **CSS Tutorial** CSS Reference CSS Examples CSS Certificate JavaScript Tutorial JavaScript Reference JavaScript Examples JavaScript Certificate SQL Reference **How To Tutorial** How To Examples Front End Certificate SQL Tutorial Python Reference SQL Examples SQL Certificate Python Tutorial W3.CSS Reference Python Examples Python Certificate W3.CSS Tutorial Bootstrap Reference W3.CSS Examples PHP Certificate **Bootstrap Tutorial** PHP Reference **Bootstrap Examples** jQuery Certificate PHP Tutorial HTML Colors PHP Examples Java Certificate Java Tutorial Java Reference Java Examples C++ Certificate C++ Tutorial Angular Reference XML Examples C# Certificate jQuery Tutorial jQuery Reference jQuery Examples XML Certificate

Next 🕽

W3Schools is optimized for learning and training. Examples might be simplified to improve reading and learning. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using W3Schools, you agree to have read and accepted our terms of use, cookie and privacy policy.

Copyright 1999-2023 by Refsnes Data. All Rights Reserved.

W3Schools is Powered by W3.CSS.

V³

