Javascript Game Development - Keyboard Input

Arthur Schreiber · 27 February 2011

Now that we have our basic game loop layed out, we can focus on implementing other aspects of our javascript based game. What I want to show in this post is how you process keyboard input.

Keyboard Events

If you've done any previous web development with javascript, you'll most likely know the `keydown` and `keyup` events that get fired by the browser when a key is pressed down and released again.

Let's say we want to create a game where the player controls a small black rectangle and can move it around in a bigger area (very sophisticated, I know!).

First, we'll need a class that represents our player object:

```
function Player() {
  this.x = 0;
  this.y = 0;
}

Player.prototype.draw = function(context) {
  context.fillRect(this.x, this.y, 32, 32);
};

Player.prototype.moveLeft = function() {
  this.x -= 1;
};

Player.prototype.moveRight = function() {
  this.x += 1;
};

Player.prototype.moveUp = function() {
  this.y -= 1;
};

Player.prototype.moveRight = function() {
  this.y += 1;
};
```

Next, we'll need to hook up on instance of the Player class to our existing game

code.

```
Game.start = function() {
  ...

  Game.player = new Player();

  ...
};

Game.draw = function() {
  ...

  Game.player.draw(Game.context);

  ...
};
```

Last, we need to setup our event handling code.

```
window.addEventListener('keydown', function(event) {
  switch (event.keyCode) {
    case 37: // Left
      Game.player.moveLeft();
    break;

    case 38: // Up
      Game.player.moveUp();
    break;

    case 39: // Right
      Game.player.moveRight();
    break;

    case 40: // Down
      Game.player.moveDown();
    break;
  }
}, false);
```

This first version kinda works, but it is pretty choppy and only ever works for one key. Why?

The choppiness is due to the fact that the keydown event is fired repeatedly with small pauses between each event, and these pauses are not synced up with our game update code. So we'll have to figure out a way to get them in sync.

The problem that only ever one key is recognized comes from the fact that the browser only ever repeats keydown events for the last key that was pressed.

For working around these two problems, I'm using this super simple helper

object:

```
var Key = {
  _pressed: {},

  LEFT: 37,
  UP: 38,
  RIGHT: 39,
  DOWN: 40,

  isDown: function(keyCode) {
    return this._pressed[keyCode];
  },

  onKeydown: function(event) {
    this._pressed[event.keyCode] = true;
  },

  onKeyup: function(event) {
    delete this._pressed[event.keyCode];
  }
};
```

You can hook it up to keydown and keyup events like this:

```
window.addEventListener('keyup', function(event) { Key.onKeyup(event); }, false);
window.addEventListener('keydown', function(event) { Key.onKeydown(event); }, false);
```

This object tracks the key status of individual keys based on the keydown/keyup events fired by the browser. Using it, we can easily check in our game update code whether a specific key is currently pressed down or not. Now we add the following function to our Player class:

```
Player.prototype.update = function() {
  if (Key.isDown(Key.UP)) this.moveUp();
  if (Key.isDown(Key.LEFT)) this.moveLeft();
  if (Key.isDown(Key.DOWN)) this.moveDown();
  if (Key.isDown(Key.RIGHT)) this.moveRight();
};
```

Then we only need to hook this also up to the Game.update method:

```
Game.update = function() {
  ...
  Game.player.update();
  ...
};
```

Check out this final version.

Sort by Best Share Favorite

Join the discussion…

Geoff H • 4 years ago

Protip: Don't store "true", store the epoch time the event occurred. Now you have TWO pieces of useful information about the press, and can do smarter event handling because you know when it was pressed, so you can increase or decrease an effect, or just turn it off after a max time.

4 • Reply • Share ›

Guest Geoff H • 4 years ago

Good idea. :)

• Reply • Share ›

Arnor Heidar Sigurdsson • 2 years ago

Ok, this is great, I did something similar for a game I'm working on. I have one question, one that I'm trying to deal with which also seems to be the case with your implementation.

When you hold eg. the "Right" key down and then press [meta on mac, ctrl on pc, though i haven't tried on a PC], and then release the right key, the keyup event is not registered for releasing the Right key, so the box just continues moving, until you press right again.

Any idea how to prevent this behavior?

3 • Reply • Share ›

Gma85690 • 3 years ago

Awesome, Awesome, Awesome!

Maybe you plan write more article's on JavaScript Game Development?

3 • Reply • Share ›

Michael • 2 years ago

I just finished writing a bit that was strikingly similar to what you've done here, then stumbled upon your article. I feel reassured that other people are thinking about these problems in the same way. :) Great resource. I encourage you to share more!

1 • Reply • Share ›

Addison Jones-Mulaire • 3 years ago

Hi there, I'm just starting to work with html5 and javascript for a project, and I'm having some trouble putting the two steps together. Could you show me the overall structure of the file?

1 • Reply • Share ›

Adinan Cenci • a year ago

I did something like that but you see there is a problem

I implemented a function that open/close the interface of my game, in that fraction of time my finger is pressing the button, the Game.update would have run dozens of times, and so the function open, close the interface dozens of times...
A workaround would be use a onkeydown event apart from the game loop, but I rather use an single...?interface?...a single way to set events for the keyboard. any ideas?

▲ | ▼ • Reply • Share ›

Ibraheem Osama Mohamed • 2 years ago

Nice really helpful :)

▲ | ▼ • Reply • Share ›

Serj Nights • 2 years ago

TY MAN!!!
I am sooooo happy that i finally find the way to fix this "choppy" effect!!!
Thank you!

▲ | ▼ • Reply • Share ›

Dude • 2 years ago

This is very elegant. Thank you.

▲ | ▼ • Reply • Share ›

Melanke • 3 years ago

great post, great blog!

▲ | ▼ • Reply • Share ›

a. • 3 years ago

Helped a lot, cheers broseph ;)

▲ | ▼ • Reply • Share ›

louisstow • 4 years ago

You're giving my some great ideas for my JavaScript game engine, Crafty (http://craftyjs.com).
Cheers!

▲ | ▼ • Reply • Share ›

Arthur Schreiber Mod ↗ louisstow • 4 years ago

Thanks! :)

▲ | ▼ • Reply • Share ›

guest • 4 years ago

cool article; :-)

but, don't work IE?

▲ | ▼ • Reply • Share ›

Arthur Schreiber Mod ↗ guest • 4 years ago

Yes, it won't work in IE 6/7/8, as these versions don't support the

<canvas> element (nor window.addEventListener). You could work around this by using excanvas (http://code.google.com/p/explo.../) and creating/using a wrapper around window.attachEvent/window.addEventListener. But excanvas performance will probably not be good enough to implement a real game.

It works in IE 9, tho. :)

• Reply • Share ›

guest Arthur Schreiber • 4 years ago

used canvas thinking. I didn't check sorry.

anyway, good post :-)

• Reply • Share ›

abw • 4 years ago

You don't need the extra function wrappers around the keyboard event handlers:

window.addEventListener('keyup', Key.onKeyup, false);

• Reply • Share ›

Arthur Schreiber Mod abw • 4 years ago

Well, you're wrong there. :)

Removing the function wrapper will result in the `onKeyup` function to be bound to the `window` object instead of the `Key` object. I could've used Function#bind as in:

window.addEventListener('keyup', Key.onKeyup.bind(Key), false);

But Function#bind is not available in older browsers, so I didn't use it.

1 • Reply • Share ›

Anonymous Function abw • 2 years ago

I believe it does need to be called as an anonymous function if you want to pass in a parameter.

• Reply • Share ›

Xiaolu Xiong • 4 years ago

Cool!

• Reply • Share ›