#Cursed-socials

Cursed Lesson #1 — Gravity Text

a Cursed Socials tutorial by Priyanshu Singh learn real JavaScript physics by building something dumb on purpose

#Cursed-socials

What are we building?

The idea: when a user stops scrolling, all the text on your page slowly falls down like gravity is pulling it. when they scroll again, everything snaps back.

What you will learn:

- how requestAnimationFrame works (the engine behind every web animation)
- detecting when someone stops scrolling using setTimeout
- moving elements around with transform: translateY()
- basic physics: velocity + acceleration

#Cursed-socials

The setup (plain HTML)

You only need 3 files. No frameworks, no npm, nothing fancy.



Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Gravity Text Demo</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>My Cursed Social</h1>
  <p>stop scrolling and watch what happens</p>

  <div class="post">
    <h2>first post</h2>
    <p>this text is about to fall lol</p>
  </div>
```

```html
  <div class="post">
    <h2>second post</h2>
    <p>gravity is coming for me</p>
  </div>

  <div class="post">
    <h2>third post</h2>
    <p>i am not safe either</p>
  </div>

  <div style="height: 200vh;"></div>

  <script src="gravity.js"></script>
</body>
</html>
```

(the `200vh` div is just so the page is scrollable)

Quick Css (nothing special)

```css
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Comic Sans MS', cursive, sans-serif;
  background: #f5f0ff;
  color: #1e1750;
  padding: 40px 20px;
}

h1 {
  font-size: 2.5rem;
  margin-bottom: 10px;
}
```

```css
.post {
  border: 2px solid #483b8a;
  padding: 20px;
  margin: 20px 0;
  border-radius: 8px;
  background: white;
}

.post h2 {
  margin-bottom: 8px;
}
```

Nothing cursed here yet. Just a normal looking page.

The cursed part is all JavaScript.

Step 1: Grab all the text

First we need to tell JavaScript which elements should fall. We use querySelectorAll to grab every heading and paragraph on the page.

```
var textElements = document.querySelectorAll('h1, h2, p');
```

What this does: finds every <h1>, <h2>, and <p> tag on the page and puts them in a list.

You could also target specific classes like:

```
var textElements = document.querySelectorAll('.post h2, .post p');
```

Step 2: Build a "drop" for each element

Each text element needs its own physics. We store how far it has fallen (y), how fast it is going (speed), and how quickly it speeds up (accel).

```
var drops = [];

for (var i = 0; i < textElements.length; i++) {
  drops.push({
    el: textElements[i],
    y: 0,
    speed: 0,
    accel: 0.15 + Math.random() * 0.25,
    maxFall: 60 + Math.random() * 90
  });
}
```

Why `Math.random()` ? So each piece of text falls at a slightly different speed and distance. If they all fell the same way it would look boring and robotic.

- `accel` = how fast it picks up speed (like real gravity but tiny)
- `maxFall` = maximum pixels it can fall before stopping

Step 3: The animation loop

This is the core. requestAnimationFrame is how browsers run smooth animations. It calls your function ~60 times per second.

```
var falling = false;
var animationId = null;

function animateFall() {
  if (!falling) return;

  var allDone = true;

  for (var i = 0; i < drops.length; i++) {
    var d = drops[i];
    if (d.y < d.maxFall) {
      d.speed = d.speed + d.accel;
      d.y = Math.min(d.y + d.speed, d.maxFall);
      allDone = false;
    }
    d.el.style.transform = 'translateY(' + d.y + 'px)';
  }

  if (!allDone) {
    animationId = requestAnimationFrame(animateFall);
  }
```

How the physics work:

- each frame, `speed` increases by `accel` (it accelerates, just like real falling)
- `y` increases by `speed` (it moves down)
- it stops when `y` reaches `maxFall`
- `translateY` moves the element down without breaking the layout

What is requestAnimationFrame?

The boring way (don't do this):

```
setInterval(function() {
  // move stuff
}, 16);
```

The right way:

```js
requestAnimationFrame(function() {
  // move stuff
});
```

Why?

- `requestAnimationFrame` syncs with your screen refresh rate (usually 60fps)
- it pauses when the tab is hidden (saves battery)
- it is smoother because the browser knows exactly when to paint
- every real game and animation library uses this

This is what powers every smooth animation you see on the web. Learning it here means you can build games, parallax effects, loading spinners, anything.

Step 4: Detect when scrolling stops

There is no "scroll stop" event in JavaScript. So we fake one using a timer trick.

```
var scrollTimer = null;

function startFalling() {
  falling = true;

  for (var i = 0; i < drops.length; i++) {
    drops[i].y = 0;
    drops[i].speed = 0;
  }

  animationId = requestAnimationFrame(animateFall);
}

function stopFalling() {
  falling = false;

  if (animationId) {
    cancelAnimationFrame(animationId);
  }

  for (var i = 0; i < drops.length; i++) {
    drops[i].el.style.transition = 'transform 0.4s ease';
    drops[i].el.style.transform = '';
  }
```

```
  setTimeout(function() {
    for (var i = 0; i < drops.length; i++) {
      drops[i].el.style.transition = '';
    }
  }, 500);
}
```

The timer trick: every time the user scrolls, we clear the old timer and start a new 1.5 second timer. If 1.5 seconds pass with no scrolling, the timer fires and we start the gravity.

```
window.addEventListener('scroll', function() {
  if (falling) {
    stopFalling();
  }
  clearTimeout(scrollTimer);
  scrollTimer = setTimeout(function() {
    startFalling();
  }, 1500);
});
```

Step 5: Start it on page load too

```
scrollTimer = setTimeout(function() {
  startFalling();
}, 3000);
```

Add this at the bottom of your gravity.js file.

What you just learned (for real)

Concept	Where you used it
`querySelectorAll`	grabbing multiple elements at once
`requestAnimationFrame`	running smooth 60fps animations
`cancelAnimationFrame`	stopping animations cleanly
velocity + acceleration	basic physics simulation
`transform: translateY()`	moving elements without breaking layout
`setTimeout` / `clearTimeout`	detecting when an event stops firing
`Math.random()`	adding natural variation
`style.transition`	smooth snap-back animation

These are the same building blocks used in real games, scroll animations, and interactive websites. You just learned them by making text fall off a page.

Make it your own

Ideas to remix this:

- Sideways gravity: use translateX instead and make text slide off to the right
- Bounce: when y hits maxFall, reverse the speed and multiply by 0.6 so it bounces
- Rotate while falling: add a random rotation angle that increases as it falls
- Sound effect: play a thud sound when each piece of text hits its max fall

Closing

You just built a physics engine in 60 lines of JavaScript.

No React. No libraries. No AI. Just you, a text editor, and `requestAnimationFrame`.

Now go put this in your cursed social and watch reviewers lose their minds.

Ship it. Claim your grant.

make sure to try this out on #cursed-socials website!