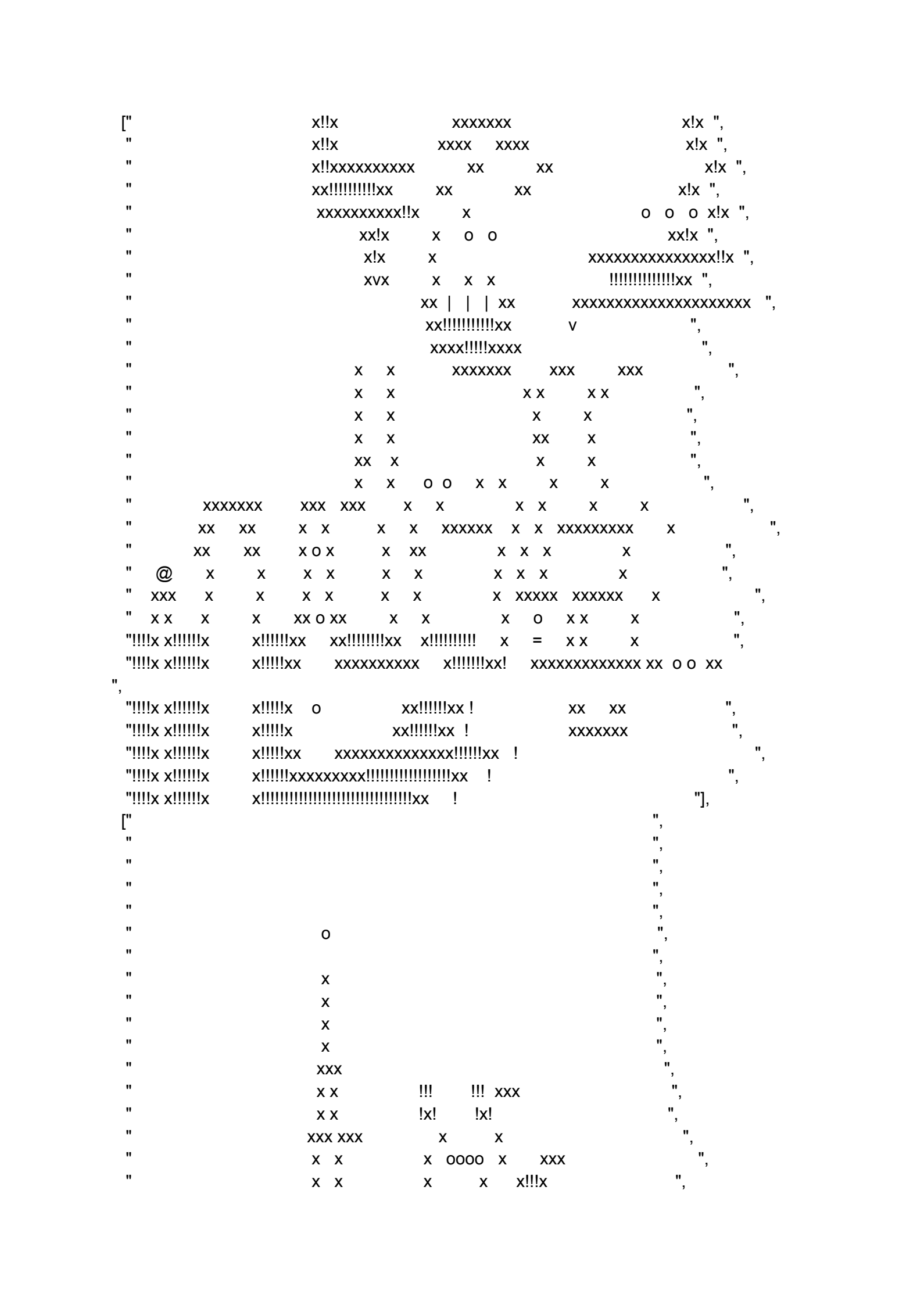
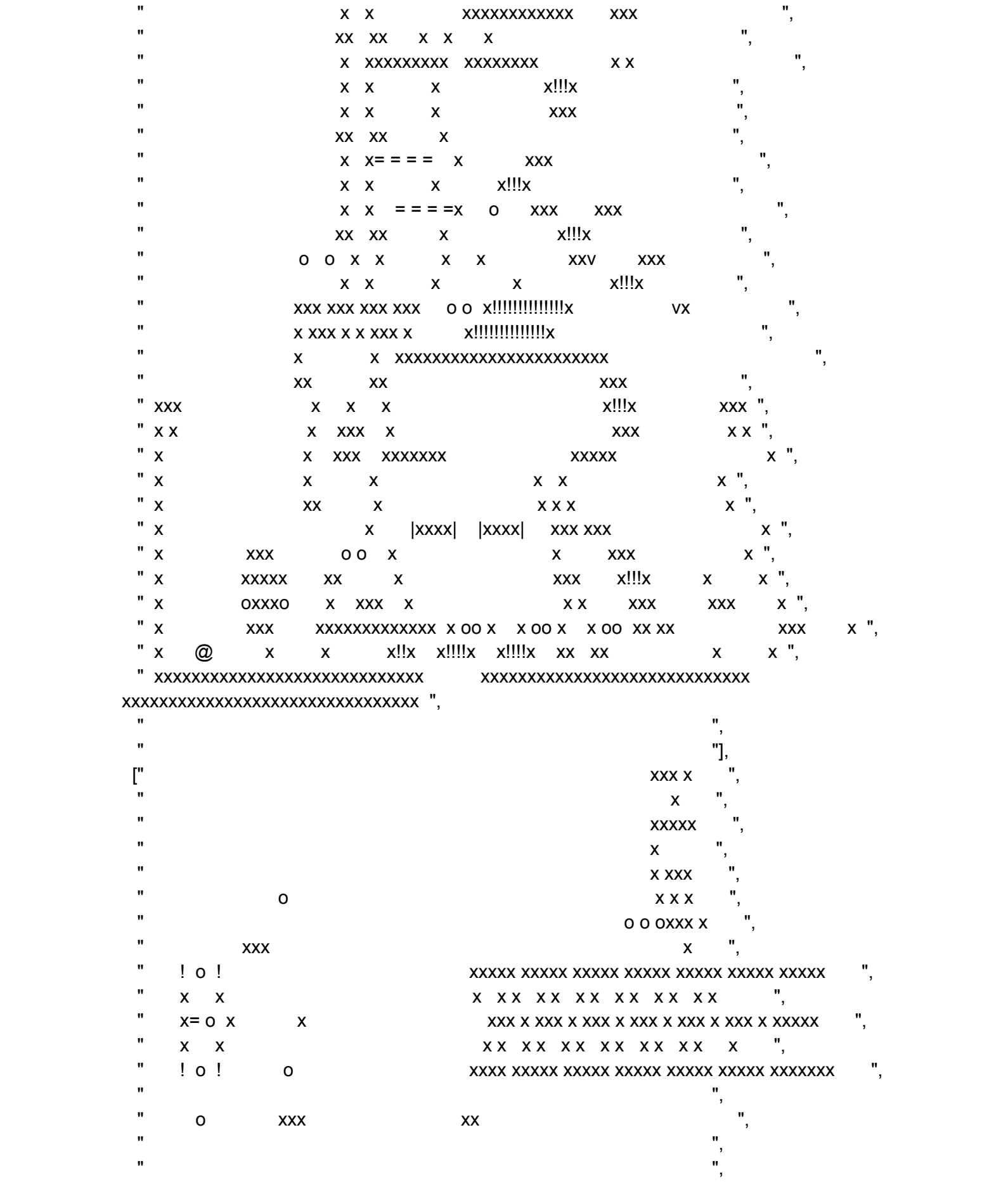
```
<!DOCTYPE html>
<html lang="en" >
<head>
  <meta charset="UTF-8">
  <title>game</title>
  <link rel="stylesheet" href="./style.css">
<syle>
body {
  background: #222;
}

h2 {
  color: #666;
  font-family: monospace;
  text-align: center;
}

.background {
  table-layout: fixed;
  border-spacing: 0;
}

.background td {
  padding: 0;
}

.lava, .actor {
  background: #e55;
}

.wall {
  background: #444;
  border: solid 3px #333;
  box-sizing: content-box;
}

.actor {
  position: absolute;
}

.coin {
  background: #e2e838;
  border-radius: 50%;
}

.player {
  background: #335699;
```

```
  box-shadow: none;
}

.lost .player {
  background: #a04040;
}

.won .player {
  background: green;
}

.game {
  position: relative;
  overflow: hidden;
}
</style>
</head>
<body>
<!-- partial:index.partial.html -->
<h2>Game Petualangan Informatika</h2>
<!-- partial -->
  <script  src="./script.js"></script>
<script>
var LEVELS = [
  ["                                                ",
   "                                                ",
   "                                                ",
   "                                                ",
   "                                                ",
   "                                                ",
   "                                                ",
   "                                      xxx       ",
   "                                 xx   xx  xx!xx     ",
   "                       o o     xx          x!!!x      ",
   "                                      xx!xx      ",
   "                       xxxxx             xvx      ",
   "                                           xx  ",
   "  xx                       o o                x  ",
   "  x              o                           x  ",
   "  x                       xxxxx                o x  ",
   "  x       xxxx     o                            x  ",
   "  x  @     x  x                          xxxxx     x  ",
   "  xxxxxxxxxxxx xxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxxxx   xxxxxxx  xxxxxxxxx  ",
   "                    x  x           x    x           ",
   "                    x!!!x          x!!!!!x          ",
   "                    x!!!x          x!!!!!x          ",
   "                    xxxxx            xxxxxxx            ",
   "                                                ",
   "                                                "],
```





```
  "                                   xx                                    ",
  "             xxx       xxx                                                 ",
  "                                                                           ",
  "                 o                                 x     x                 ",
  "                                     xx    xx                              ",
  "         xxx       xxx       xxx                     x           x         ",
  "                                                                           ",
  "                                       ||                                  ",
  "  xxxxxxxxxxx                                                              ",
  "  x       x o xxxxxxxxx o xxxxxxxxx o xx                                   x            ",
  "  x       x  x     x  x     x  x           ||            x    x                ",
  "  x  @    xxxxx  o  xxxxx  o  xxxxx                                              ",
  "  xxxxxxx                       xxxxx     xx    xx    xxx                      ",
  "       x=             =           =x  x             xxx                       ",
  "       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
x!!!!!!!!!!!!!!!!!!!xxx!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!",
  "
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "                                                                         "]
];


function Vector(x, y) {
        this.x = x; this.y = y;
}

Vector.prototype.plus = function(other) {
        return new Vector(this.x + other.x, this.y + other.y);
};

Vector.prototype.times = function(scale) {
        return new Vector(this.x * scale, this.y * scale);
};

// Note: uppercase words are used that means constructor are values
var actorchars =  {
        "@": Player,
        "o": Coin,
        "=": Lava,
        "|": Lava,
        "v": Lava
};

function Player(pos) {
        this.pos = pos.plus(new Vector(0, -.5));
        this.size = new Vector(.5, 1);
        this.speed = new Vector(0, 0);
}
```

```
Player.prototype.type = "player";
//IT Komputer MAN Kapuas
function Lava(pos, ch) {
	this.pos = pos;
	this.size = new Vector(1, 1);
	if (ch === "=")
		this.speed = new Vector(2, 0);
	else if (ch === '|')
		this.speed = new Vector(0, 2);
	else if (ch === 'v'){
		this.speed = new Vector(0, 3);
		this.repeatPos = pos;
	}
}
Lava.prototype.type = "Lava";

function Coin(pos) {
	this.basePos = this.pos = pos;
	this.size = new Vector(.6, .6);
	// take a look back
	this.wobble = Math.random() * Math.PI * 2;
}
Coin.prototype.type = "coin";

Level.prototype.isFinished = function() {
  return this.status != null && this.finishDelay < 0;
};

function Level(plan) {
	this.width = plan[0].length;
	this.height = plan.length;
	this.grid = [];
	this.actors = [];

	for (var y = 0; y < this.height; y++) {
		var line = plan[y],  gridLine = [];
		for (var x = 0; x < this.width; x++) {
			var ch = line[x], fieldType = null;
			var Actor = actorchars[ch];
			if (Actor)
				this.actors.push(new Actor(new Vector(x, y), ch));
			else if (ch === "x")
				fieldType = "wall";
			else if (ch === "!")
				fieldType = "lava";
			else if (ch === "|")
				fieldType = "lava";
			else if (ch === "=")
```

```
                        fieldType = "lava";
                    else if (ch === "v"){
                        fieldType = "lava";
                        console.log(fieldType);
                    }
                    gridLine.push(fieldType)
                }
                this.grid.push(gridLine);
        }
        this.player = this.actors.filter(function(actor) {
                return actor.type === "player"
        })[0];
        this.status = this.finishDelay = null;
}

function element(name, className) {
        var elem = document.createElement(name);
        if(className) elem.className = className;
        return elem;
}

function DOMDisplay(parent, level) {
        this.wrap = parent.appendChild(element("div", "game"));
        this.level = level;

        this.wrap.appendChild(this.drawBackground());
        this.actorLayer = null;
        this.drawFrame();
}


var scale = 15;


DOMDisplay.prototype.drawBackground = function() {
        var table = element("table", "background");
        table.style.width = this.level.width * scale + "px";
        table.style.height = this.level.height * scale + "px";
        this.level.grid.forEach(function(row) {
  var rowElement = table.appendChild(element("tr"));
                rowElement.style.height = scale + "px";
                row.forEach(function(type) {
                        rowElement.appendChild(element("td", type));
                });
        });
        return table;
};
```

```
DOMDisplay.prototype.drawActors = function() {
	var wrap = element("div");
	this.level.actors.forEach(function(actor) {
		var rect = wrap.appendChild(element("div", "actor " + actor.type));
		rect.style.width = actor.size.x * scale + "px";
		rect.style.height = actor.size.y * scale + "px";
		rect.style.left = actor.pos.x * scale + "px";
		rect.style.top = actor.pos.y * scale + "px";
	});
	return wrap;
}

DOMDisplay.prototype.drawFrame = function() {
	if (this.actorLayer)
		this.wrap.removeChild(this.actorLayer);
	this.actorLayer = this.wrap.appendChild(this.drawActors());
	this.wrap.className = "game " + (this.level.status || "");
	this.scrollPlayerIntoView();
};


// clear it later
DOMDisplay.prototype.scrollPlayerIntoView = function() {
  var width = this.wrap.clientWidth;
  var height = this.wrap.clientHeight;
  var margin = width / 3;

  // The viewport
  var left = this.wrap.scrollLeft, right = left + width;
  var top = this.wrap.scrollTop, bottom = top + height;

  var player = this.level.player;
  var center = player.pos.plus(player.size.times(0.5))
                 .times(scale);

  if (center.x < left + margin)
    this.wrap.scrollLeft = center.x - margin;
  else if (center.x > right - margin)
    this.wrap.scrollLeft = center.x + margin - width;
  if (center.y < top + margin)
    this.wrap.scrollTop = center.y - margin;
  else if (center.y > bottom - margin)
    this.wrap.scrollTop = center.y + margin - height;
};

DOMDisplay.prototype.clear = function() {
	this.wrap.parentNode.removeChild(this.wrap);
};
```

```
Level.prototype.obstacleAt = function(pos, size) {
  var xStart = Math.floor(pos.x);
  var xEnd = Math.ceil(pos.x + size.x);
  var yStart = Math.floor(pos.y);
  var yEnd = Math.ceil(pos.y + size.y);

  if (xStart < 0 || xEnd > this.width || yStart < 0)
    return "wall";
  if (yEnd > this.height)
    return "lava";
  for (var y = yStart; y < yEnd; y++) {
    for (var x = xStart; x < xEnd; x++) {
      var fieldType = this.grid[y][x];
      if (fieldType) return fieldType;
    }
  }
};

Level.prototype.actorAt = function(actor) {
  for (var i = 0; i < this.actors.length; i++) {
    var other = this.actors[i];
    if (other != actor &&
        actor.pos.x + actor.size.x > other.pos.x &&
        actor.pos.x < other.pos.x + other.size.x &&
        actor.pos.y + actor.size.y > other.pos.y &&
        actor.pos.y < other.pos.y + other.size.y)
      return other;
  }
};

var maxStep = 0.05;

Level.prototype.animate = function(step, keys) {
  if (this.status != null)
    this.finishDelay -= step;

  while (step > 0) {
    var thisStep = Math.min(step, maxStep);
    this.actors.forEach(function(actor) {
      actor.act(thisStep, this, keys);
    }, this);
    step -= thisStep;
  }
};


Lava.prototype.act = function(step, level) {
```

```
  var newPos = this.pos.plus(this.speed.times(step));
  if (!level.obstacleAt(newPos, this.size))
    this.pos = newPos;
  else if (this.repeatPos)
    this.pos = this.repeatPos;
  else
    this.speed = this.speed.times(-1);
};

var wobbleSpeed = 8, wobbleDist = 0.07;

Coin.prototype.act = function(step) {
  this.wobble += step * wobbleSpeed;
  var wobblePos = Math.sin(this.wobble) * wobbleDist;
  this.pos = this.basePos.plus(new Vector(0, wobblePos));
};


var playerXSpeed = 10;

Player.prototype.moveX = function(step, level, keys) {
  this.speed.x = 0;
  if (keys.left) this.speed.x -= playerXSpeed;
  if (keys.right) this.speed.x += playerXSpeed;

  var motion = new Vector(this.speed.x * step, 0);
  var newPos = this.pos.plus(motion);
  var obstacle = level.obstacleAt(newPos, this.size);
  if (obstacle)
    level.playerTouched(obstacle);
  else
    this.pos = newPos;
};

var gravity = 30;
var jumpSpeed = 17;

Player.prototype.moveY = function(step, level, keys) {
  this.speed.y += step * gravity;
  var motion = new Vector(0, this.speed.y * step);
  var newPos = this.pos.plus(motion);
  var obstacle = level.obstacleAt(newPos, this.size);
  if (obstacle) {
    level.playerTouched(obstacle);
    if (keys.up && this.speed.y > 0)
      this.speed.y = -jumpSpeed;
    else
      this.speed.y = 0;
```

```
  } else {
    this.pos = newPos;
  }
};

Player.prototype.act = function(step, level, keys) {
  this.moveX(step, level, keys);
  this.moveY(step, level, keys);

  var otherActor = level.actorAt(this);
  if (otherActor)
    level.playerTouched(otherActor.type, otherActor);

  // Losing animation
  if (level.status == "lost") {
    this.pos.y += step;
    this.size.y -= step;
  }
};

Level.prototype.playerTouched = function(type, actor) {
  if (type == "lava" && this.status == null) {
    this.status = "lost";
    this.finishDelay = 1;
  } else if (type == "coin") {
    this.actors = this.actors.filter(function(other) {
      return other != actor;
    });
    if (!this.actors.some(function(actor) {
      return actor.type == "coin";
    })) {
      this.status = "won";
      this.finishDelay = 1;
    }
  }
};

var arrowCodes = {37: "left", 38: "up", 39: "right"};

function trackKeys(codes) {
  var pressed = Object.create(null);
  function handler(event) {
    if (codes.hasOwnProperty(event.keyCode)) {
      var down = event.type == "keydown";
      pressed[codes[event.keyCode]] = down;
      event.preventDefault();
    }
  }
```

```
  addEventListener("keydown", handler);
  addEventListener("keyup", handler);
  return pressed;
}

function runAnimation(frameFunc) {
  var lastTime = null;
  function frame(time) {
    var stop = false;
    if (lastTime != null) {
      var timeStep = Math.min(time - lastTime, 100) / 1000;
      stop = frameFunc(timeStep) === false;
    }
    lastTime = time;
    if (!stop)
      requestAnimationFrame(frame);
  }
  requestAnimationFrame(frame);
}

var arrows = trackKeys(arrowCodes);

function runLevel(level, Display, andThen) {
  var display = new Display(document.body, level);
  runAnimation(function(step) {
    level.animate(step, arrows);
    display.drawFrame(step);
    if (level.isFinished()) {
      display.clear();
      if (andThen)
        andThen(level.status);
      return false;
    }
  });
}

function runGame(plans, Display) {
  function startLevel(n) {
    runLevel(new Level(plans[n]), Display, function(status) {
      if (status == "lost")
        startLevel(n);
      else if (n < plans.length - 1)
        startLevel(n + 1);
      else
        alert("You win!");
    });
  }
  startLevel(0);
```

```
}

runGame(LEVELS, DOMDisplay);
<script>
</body>
</html>
```