Comp 125 - Visual Information Processing

Spring Semester 2018 - week 14 - wednesday

Dr Nick Hayward

JavaScript - Object Prototype

basic idea of prototypes - part 1

- basic idea of *prototypes* in JS is straightforward
- every *object* can have a reference to its *prototype*
  - *a delegate object with properties*
  - *default behaviour for child objects (example on next slide)*
- i.e. *prototype* is a fall back object to search for a given property &c.

JavaScript - Object Prototype

basic idea of prototypes - part 2

```
const object1 = { title: 'the glass bead game' };
const object2 = { author: 'herman hesse' };

console.log(object1.title);

Object.setPrototypeOf(object1, object2);

console.log(object1.author);
```

- define two objects
  - *their properties can be called with standard object notation*
  - *they can be modified and mutated as standard*
- an object's internal *prototype* is not directly accessible
- we need to use the method `setPrototypeOf()`
  - *sets and updates an object's prototype*
- e.g. pass `object1` as the object to update
  - *`object2` as the object to set as prototype*
- if a requested property is not found on `object1`
  - *JS will then search its prototype.*
- e.g. request `author` as a property of `object1`
  - *it's not found on `object1`*
  - *but it's available as a property of its prototype (it's parent...)*

JavaScript - Object Prototype

object constructor & prototypes

- object-oriented languages, such as Java and C++, include a `class` constructor
  - *provides known encapsulation and structuring*
- the constructor is initialising an object to a known initial state
- this pattern allows us to consolidate a set of properties and methods
  - *for a `class` of objects in one place*
- JS offers such a mechanism
  - *although in a slightly different form to Java, C++ &c.*
- JS still uses the `new` operator to instantiate new objects via constructors
  - *JS does not include a true `class` definition*
- in JS, the `new` operator is applied to a constructor function
  - *this triggers the creation of a new object*

JavaScript - Object Prototype

prototype method

- in JS, every function includes their own prototype object
  - *set automatically as the prototype of any created objects*

```
//constructor for object
function libraryRecord() {
    //set default value on prototype
    libraryRecord.prototype.library = 'castalia';
}

const bookRecord = new libraryRecord();

console.log(bookRecord.library);
```

- we may also set a default method on an instantiated object's prototype
  - *his is what we'll be doing as we create the next animation*

HTML Canvas - basic animations

add some physics - collision detection

- random updates to a shape's movement around the canvas is useful
  - *we may want the shape to remain within the confines of the canvas as well*
- we need to check for basic collisions of the drawn shape
  - *collisions against the canvas edges &c.*
- need to implement custom **collision detection** for our shape

HTML Canvas - basic animations

pin ball with collision - part 1

- add canvas to HTML

```
<!-- add canvas -->
<canvas id="drawing" width="400" height="400"></canvas>
```

HTML Canvas - basic animations

pin ball with collision - part 2

- add function to create a circle
  - *define `x`, `y` coordinates for circle centre*
  - *define `radius`*
  - *draw with fill or stroke and colour*

```
// define circle function
function circle(x, y, radius, fillCircle, color) {
    // start recording
  context.beginPath();
    // define arc - x, y, radius, start posn, end posn, anticlockwise...
    context.arc(x, y, radius, 0, Math.PI * 2, false);
    // check fill or stroke
    if (fillCircle) {
        // colour for fill
      context.fillStyle = color;
        context.fill();
    } else {
        // set line width & line colour
        context.lineWidth = 2;
    context.strokeStyle = color;
        context.stroke();
    }
}
```

HTML Canvas - basic animations

pin ball with collision - part 3

- define the **constructor** for the pin ball

```
// ball constructor - name capitalised to denote constructor
function Ball() {
 this.x = 100;
 this.y = 100;
 this.xSpeed = -2;
 this.ySpeed = 3;
};
```

- negative value for `xSpeed`
  - *makes ball to left for each frame of the animation*

HTML Canvas - basic animations

pin ball with collision - part 4

- define `draw` method on `Ball` prototype
  - *draws pin ball as circle with colour &c.*
  - *always the same size, colour &c.*

```
// 1. update prototype - method to draw ball
Ball.prototype.draw = function () {
 circle(this.x, this.y, 10, true, 'green');
};
```

HTML Canvas - basic animations

pin ball with collision - part 5

- define `move` method on `Ball` prototype
  - *moves pin ball around `canvas`*
- updates `x` and `y` based on current speed
  - *adds horizontal speed for `this.x`*
  - *adds vertical speed for `this.y`*

```
// 2. update prototype -method to move a ball
Ball.prototype.move = function () {
 this.x += this.xSpeed;
 this.y += this.ySpeed;
};
```

- e.g. x = `100 - 2`, then `98 -2`, and so on...
- e.g. y = `100 + 3`, then `103 + 3`, and so on...

HTML Canvas - basic animations

pin ball with collision - part 6

- then need to check for collisions
- e.g. if the ball hits the bottom edge of the `canvas`
  - *negate speed for ball*
  - `this.ySpeed = -this.ySpeed`
- e.g. if the ball hits either side of the `canvas`
  - `this.xSpeed = -this.xSpeed`

HTML Canvas - basic animations

pin ball with collision - part 7

- collision detection may be added as follows

```
// 3. update prototype - check a collision
Ball.prototype.checkCollision = function () {
if (this.x < 0 || this.x > 400) {
 this.xSpeed = -this.xSpeed;
 }
if (this.y < 0 || this.y > 400) {
 this.ySpeed = -this.ySpeed;
 }
};
```

- check if the ball has hit either horizontal side of `canvas`
  - `if (this.x < 0 || this.x > 400)`
- also check if ball hits either vertical side of `canvas`
  - `if (this.y < 0 || this.y > 400)`

HTML Canvas - basic animations

pin ball with collision - part 8

- create new pin ball for animation
  - *instantiate `ball` using `Ball` constructor*
- `ball` object now includes `prototype` methods and properties
  - *`draw()`, `move()` and `checkCollision()` method*
  - *`x`, `y`, `xSpeed`, and `ySpeed` properties*

```
// instantiate a ball object using the Ball constructor
var ball = new Ball();
```

HTML Canvas - basic animations

pin ball with collision - part 9

- create animation with standard `setInterval` timer

```
// run the timer for the animation...
setInterval(function () {
 context.clearRect(0, 0, 400, 400);
 ball.draw();
 ball.move();
 ball.checkCollision();
}, 30);
```

- for each execution of `setInterval`
  - *clear `canvas` rectangle*
  - *call `draw()`, `move()` and `checkCollision()` methods*
- call `setInterval` every `30` milliseconds
- Example - pin ball with collision
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.4/*

HTML Canvas - Canvas

add text with fill style

- define font for use with text
  - *e.g.* `Sans Serif`
- define font size for text
  - *e.g.* `25px`
- draw text with `fillText()`
  - *add string to render as text*
  - *specify `x` and `y` coordinates*

```
// define font for text
context.font = "25px Sans Serif";
// draw text on canvas - string, x, y
context.fillText("Welcome to the wonderful world of canvas...", 50, 50);
```

- Example - add text with fill
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-text/basic-fill/*

HTML Canvas - Canvas

add text with stroke style

- define font for use with text
  - *e.g.* `Comic Sans MS`
- define font size for text
  - *e.g.* `30px`
- draw text with `strokeText()`
  - *add string to render as text*
  - *specify `x` and `y` coordinates*

```
// define font for text
context.font = "30px Comic Sans MS";
// draw text on canvas with stroke - string, x, y
context.strokeText("Welcome to the wonderful world of canvas...", 10, 50);
```

- Example - add text with stroke style
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-text/basic-stroke/*

HTML Canvas - Canvas

add text with colour

- define font for use with text
  - *e.g.* `Sans Serif`
- define font size for text
  - *e.g.* `25px`
- define colour for fill
  - *e.g.* `green`
- draw text with `fillText()`
  - *add string to render as text*
  - *specify `x` and `y` coordinates*

```
// define font for text
context.font = "25px Sans Serif";
// define fill colour
context.fillStyle = "green";
// draw text on canvas - string, x, y
context.fillText("Welcome to the wonderful world of canvas...", 50, 50);
```

- Example - add text with fill and colour
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-text/basic-fill-colour/*

HTML Canvas - Canvas

add text with alignment

- define font for use with text
  - *e.g.* `Sans Serif`
- define font size for text
  - *e.g.* `25px`
- define colour for fill
  - *e.g.* `green`
- define text align for point of rendering
  - *e.g.* `center` *draws text centred on* `x` *and* `y` *coordinates*
  - *other values such as* `right`, `left`
- draw text with `fillText()`
  - *add string to render as text*
  - *specify* `x` *and* `y` *coordinates based on height and width of canvas*
  - `width` *and* `height` *are properties of* `canvas` *object...*

```
// define font for text
context.font = "25px Sans Serif";
// define fill colour
context.fillStyle = "green";
// define text alignment
context.textAlign = "center";
// draw text on canvas - string, x, y
context.fillText("Welcome to the wonderful world of canvas...", canvas.width/2, canvas.height/2);
```

- Example - add text with fill colour and alignment
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-text/basic-align/*

HTML Canvas - Canvas interaction

move a ball with keyboard controls

- create a new example to allow a user to move a ball
  - *move ball around canvas using keyboard controls*
- requirements include
  - *need to draw a **ball***
  - *listen for specific keypress commands, e.g UP, DOWN, LEFT, RIGHT*
  - *then update animation of ball to reflect each keypress*
- allowing a user to directly control animation of shape on canvas
- setup our initial example with a `canvas` and `context`
  - *use `Ball` constructor and **prototype** methods*
  - *start to add logic to control the `ball`, update animation...*
  - *extend the prototype for user control of the `ball` object*

References

- MDN JS - keyboard event
- W3Schools - HTML5
  - *media elements*
  - *canvas element*
- W3Schools - JS
  - *event listener*