CS1116/CS5018

Web Development 2

Dr Derek Bridge

School of Computer Science & Information Technology
University College Cork

(Acknowledgment: This way of introducing JavaScript is inspired by the methods of Seb Lee-Delisle.)

Conditional statements

Python	JavaScript

`if x < y:`
`  print('x is smaller than y')`
`elif x == y:`
`  print('x is equal to y')`
`else:`
`  print('x is larger than y')`
	`if (x < y) {`
	`  console.log('x is smaller than y')`
	`} else if (x === y) {`
	`  console.log('x is equal to y')`
	`} else {`
	`  console.log('x is larger than y')`
	`}`

In the JavaScript, note:

- the round parentheses
- the curly braces and
- the *three* equal signs!

Also, where Python uses `and`, `or` and `not`, JavaScript uses `&&`, `||` and `!`

Revised version of `draw()`

```
function draw() {
    context.clearRect(0, 0, width, height);
    context.fillStyle = 'yellow';
    context.fillRect(x, y, size, size);
    x = x + xChange;
    y = y + yChange;
    if ( x < 0 ) {
        xChange = xChange * - 1;
    } else if ( x + size > width ) {
        xChange = xChange * - 1;
    }
    if ( y < 0 ) {
        yChange = yChange * - 1;
    } else if ( y + size > height ) {
        yChange = yChange * - 1;
    }
}
```

Type coercion

Python is a **strongly-typed** language, whereas JavaScript is **weakly-typed**

Python (they all produce error messages)	JavaScript (they all do type coercions)
`x = 'abc' + 12`	`x = 'abc' + 12;`
`x = 'abc' - 12`	`x = 'abc' - 12;`

`if 'abc' == 3:`
`  #do something`
`else:`
`  #do something else`
	`if ('abc' == 3) {`
	`  # do something`
	`} else {`
	`  # do something else`
	`}`

`if '3' == 3:`
`  #do something`
`else:`
`  #do something else`
	`if ('3' == 3) {`
	`  # do something`
	`} else {`
	`  # do something else`
	`}`

Type coercion

- JavaScript's type coercion is bizarre and causes many programming errors.
  - charlieharvey.org.uk/page/javascript_the_weird_parts
  - wtfjs
- Avoid JavaScript type coercion in equality tests by using **identity** (===), instead of **equality** (==)
  - Then in the JavaScript on the previous slide, both tests would be false

Lists and arrays

The 'equivalent' of a Python **list** is a JavaScript **array**

Python	JavaScript
`groceries = ['eggs', 'milk', 'tea']`	`let groceries = ['eggs', 'milk', 'tea'];`
`len(groceries)`	`groceries.length`
`groceries.append('bread')`	`groceries.push('bread')`

(JavaScript arrays are very similar to Python lists, but not so similar to arrays in languages such as C or Java)

JavaScript objects

- At their simplest, **objects** in JavaScript are bundles of comma-separated properties, e.g.:

```
let twinA = {
    firstname : 'John',
    surname : 'Grimes',
    age : 25
}

let twinB = {
    firstname : "Edward",
    surname : "Grimes",
    age : 25
}
```

- To refer to an object's properties, use the **dot notation**, e.g. `twinA.firstname`

for loops

- Using a loop to 'visit' each item in a list or array:

Python	JavaScript

`for item in groceries:`
`  print(item)`
	`for (let item of groceries) {`
	`  console.log(item);`
	`}`

- Using a loop to count, e.g. from 0 to 9 inclusive:

Python	JavaScript

`for i in range(10):`
`  print(i)`
	`for (let i = 0; i < 10; i += 1) {`
	`  console.log(i);`
	`}`

A new version of `particles.js`

```
let canvas;
let context;
let width;
let height;
let ps = [];
let gravity = 0.5;

document.addEventListener('DOMContentLoaded', init, false);

function init() {
    canvas = document.querySelector('canvas');
    context = canvas.getContext('2d');
    width = canvas.width;
    height = canvas.height;
    window.setInterval(draw, 33);
}

function draw() {
    for (let i = 0; i < 30; i += 1) {
        let p = {
            x : 250,
            y : 150,
            size : 10,
            xChange : getRandomNumber(-10, 10),
            yChange : getRandomNumber(-10, 10)
        };
        ps.push(p)
    }
    context.clearRect(0, 0, width, height);
    context.fillStyle = 'yellow';
    for (let p of ps) {
        context.fillRect(p.x, p.y, p.size, p.size);
        p.x = p.x + p.xChange;
        p.y = p.y + p.yChange;
        p.yChange = p.yChange + gravity;
    }
}

function getRandomNumber(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
```