Dev8D Javascript Workshop

Part 1: Objects, Functions and the Object Tree

1. Run the following code

```
var dump = function(obj) {
     for(var attr in obj) {
          println(attr+':'+obj[attr]);
     }
}

kangaroo1 = {name:'Kanga',
            jump : function(){println('whee!')}
           };
kangaroo1.name;
kangaroo1.jump();
dump(kangaroo1);
JSON.stringify(kangaroo1);

var Kangaroo = function(name) {
     this.name = name;
     this.jump= function(){println('whee!')}
}
kangaroo2 = new Kangaroo('Roo');
kangaroo2.name;
kangaroo2.jump();
dump(kangaroo2);
JSON.stringify(kangaroo2);
```

Try creating some more objects and running dump and JSON.stringify on them.

Run the following code:

```
dump(this);

JSON.stringify(this)
```

If you have time at the end, you might like to think about how you could improve the dump function.

2. Run the following code;

```
var create = function(parent) {
     var tmp = function(){};
     tmp.prototype = parent;
     var child = new tmp();
     return child;
}
bird.fly = function () {println('Flap flap!')}
owl = create(bird);
owl.fly();
dump(owl);
```

Experiment with using the create function to create other objects that inherit from eachother.

3. Run the following code:

```
function person(first_name, last_name) {
     return {first_name: first_name, last_name:last_name}
}
var person = person('Christopher', 'Robin');
dump(person);
```

Modify the person function to return an object that also has the method full_name that returns the person's full name based on `first_name` and `last_name`. Test that the function works.

Part 2: Scope and the 'this' keyword

1. Run the following code:

```
var i = 0;
function test_1() {
     for (i=0; i<10; i++) {}
}
test_1();
```

What is the value of the global variable `i`?

Run the following code:

```
var i = 0;
function test_2() {
     for (var i=0; i<10; i++) { }
}
test_2();
```

What is the value of the global variable `i`?

2.

Here is the code for a simple constructor function:

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

Run the following code:

```
p = new Point(1,2);
```

What is the value of `p.x`? What is the value of `p.y` ?

Now suppose somebody accidentally forgets the 'new' keywords and runs the following code:

```
q = Point(3,4);
```

What is `typeof q`? What is `q.x`? What is `q.y`? What is the value of the global variable `x`? What is the value of the global variable `y`?

Now suppose we have the following code instead:

```
function Point_v2(x,y) {
     this.x = x;
     this.y = y;
     return this;
}
q = Point_v2(5,6);
r = Point_v2(7,8);
```

What is `q.x`? What is `q === this`? What is `r.x`? Are q and r the same object?

3. Create a function `f` which returns `this`. Create an object `a`. Give a an attribute `b` than contains `f`. Does `a.b === f`? Does `a.b() === f()`?

4. Run the following code:

```
a = [0,1,2];
a.push(3,4,5);
b = [0,1,2];
p = b.push;
p(3,4,5);
```

What is `a[4]`? What is `b[4]`? What is the value of the global variable `length`?