Klass Documentation

Release 0.1

Jonas Obrist

April 06, 2013

CONTENTS

CHAPTER

ONE

INTRODUCTION

Klass provides class like objects in JavaScript that resemble Python classes, including explicit self arguments to methods, inheritance and the ability to call methods on super classes.

Warning: This project is mostly me playing around. Think very good about it before using this in a project.

CHAPTER

TWO

USAGE

2.1 Create a new class

Classes are created by calling `Klass()`, which optionally takes any number of parents as arguments and returns a function to define your class. That function can be called with an object of attributes and methods to be defined on your class.

Whenever a class is initialized, the `__init__` method is called.

A simple class can be created like this:

```
var Animal = Klass()({
    '__init__': function(self, name){
        self.name = name;
    }
});
```

You can now create instances of `Animal` by calling it, for example like this:

```
var my_animal = Animal('A Name');
```

2.2 Creating subclasses

Now let's make a few different animals, and give each of them a speak method:

```
var Cat = Klass(Animal)({
    'speak': function(self){
        console.log(self.name + ' says: Nyan~');
    }
});

var Fish = Klass(Animal){
    'speak': function(self){
        console.log(self.name + ' says: Blubb');
    }
});
```

Those two classes both inherit `Animal` and re-use it's `__init__` method.

They can be used just like animals:

```
var neko = Cat('nekochan');
neko.speak() // Writes "nekochan says: Nyan~" to the console
var ponyo = Fish('Ponyo');
ponyo..speak() // Writes "Ponyo says: Blubb" to the console
```

2.3 $uper

If you use inheritance, you can use the special method `$uper` on an instance to call methods on parent classes, without necessarily knowing what parent classes your instance has.

A silly example could be:

```
var Counter = Klass()({
    '__init__': function(self){
        self.value = 0;
    },
    'increment': function(self){
        self.value++;
    }
});
var DoubleCounter = Klass(Counter)({
    'increment': function(self){
        self.$uper('increment')();
        self.$uper('increment')();
    }
});
```

In this example, `DoubleCounter` just calls the `increment` method on its parent class twice when `increment` is called.

2.4 Helpers

There are two helpers `Klass.issubclass()` and `Klass.isinstance()` which behave like the `issubclass` and `isinstance` builtins in Python.

Using them using the `Animal`, `Fish` and `Cat` example from above:

```
var animal = Animal('an animal');
var neko = Cat('neko');
var fish = Fish('nemo');
Klass.issubclass(Cat, Animal); // true
Klass.issubclass(Fish, Animal); // true
Klass.issubclass(Animal, Cat); // false
Klass.issubclass(Cat, Fish); // false

Klass.isinstance(neko, Animal); // true
Klass.isinstance(neko, Cat); // true
Klass.isinstance(neko, Fish); // false
```

CHAPTER

THREE

REFERENCE

Klass ([*parent*, ...])

Returns a function to define your class. That function takes an object of attributes and methods for the class and returns a the class constructor.

If the class constructor is invoked, it returns a new instance of that class. Instances have two special methods: `__init__` which is called when the class is instantiated and `$uper` which can be used to call functions on parent classes.

`Klass.`**issubclass** (*class1*, *class2*)

Returns whether `class1` is a subclass of `class2`.

`Klass.`**isinstance** (*instance*, *klass*)

Returns whether `instance` is an instance of `klass` or a subclass of `klass`.