CSE 341 Section Handout #10
JavaScript Cheat Sheet 2

Variadic functions (var-args)

```
// use the arguments array to refer to all parameters passed
function addAll() {
    var sum = 0;
    for (var i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

Anonymous functions (lambdas)

```
function(parameters) { statements; }
```

Example:

```
[1, 2, 3, 4].map(function(n) { return n * n; }    // returns [1, 4, 9, 16]
```

Function methods

Function Method	Description
`toString()`	string representation of the function's code
`apply(`**thisObj**`, `**args**`)`	calls the function, using the given object as this
`call(`**thisObj**`, `**arg1**`, `**arg2**`, `**...**`)`	similar to apply but passes args as var-args rather than array
`bind(`**thisObj**`)`	a version of the function that uses the given object as `this`

Underscore library (http://documentcloud.github.com/underscore/)

- **Collections**: `each`, `map`, `reduce`, `reduceRight`, `detect`, `select`, `reject`, `all`, `any`, `include`, `invoke`, `pluck`, `max`, `min`, `sortBy`, `sortedIndex`, `toArray`, `size`
- **Arrays**: `first`, `rest`, `last`, `compact`, `flatten`, `without`, `uniq`, `intersect`, `zip`, `indexOf`, `lastIndexOf`, `range`
- **Functions**: `bind`, `bindAll`, `memoize`, `delay`, `defer`, `wrap`, `compose`
- **Objects**: `keys`, `values`, `functions`, `extend`, `clone`, `tap`, `isEqual`, `isEmpty`, `isElement`, `isArray`, `isArguments`, `isFunction`, `isString`, `isNumber`, `isBoolean`, `isDate`, `isRegExp`, `isNaN`, `isNull`, `isUndefined`
- **Utility**: `noConflict`, `identity`, `times`, `breakLoop`, `mixin`, `uniqueId`, `template`
- **Chaining**: `chain`, `value`

Example:

```
_([1, 4, 2, 7, 3, 5]).max()     // returns 7
_.range(10, 15)                 // returns [10, 11, 12, 13, 14]
```

Testing types

```
// typeof does not help because it always returns "object" for all objects!
object instanceof ConstructorName
object.constructor
```

Using Java classes in JavaScript, with Rhino

```
importPackage(Packages.JavaPackageName);
importClass(Packages.JavaPackageName);
var name = new JavaClassName(parameters);
var name = new InterfaceOrSubclass(object);     // implementing an interface
```

CSE 341 Section Handout #10
JavaScript Cheat Sheet 2 (continued)

Object-oriented programming and prototypes

```
function ConstructorName(parameters) {
    statements;
}
ConstructorName.prototype.methodName = function(parameters) {
    statements;
};
```

Example:
```
function Point(x, y) {
    this.x = x;
    this.y = y;
}
Point.prototype.distanceFromOrigin = function() {
    return Math.sqrt(this.x * this.x + this.y * this.y);
};
String.prototype.contains = function(text) {  // built-in types can be modified!
    return this.indexOf(text) >= 0;
};
```

Prototypal inheritance

```
function Superclass(parameters) {
    statements;
}
function Subclass(parameters) {
    statements;
}
Subclass.prototype = new SuperClass(parameters);
```

Example:
```
function Point3D(x, y, z) {
    this.x = x;
    this.y = y;
    this.z = z;
}
Point3D.prototype = new Point();   // "subclass" of Point
```

Regular expressions

```
/pattern/flags          // flags: g (global), i (case insensitive), m (multi-line)
var name = new RegExp("pattern", "flags");
```

`.`	any character	`^, $`	beginning/end of line/string	`\<, \>`	word boundaries
`|`	or	`()`	grouping/capturing	`\`	escape sequence
`*`	0 or more	`+`	1 or more	`?`	0 or 1
`{min,max}`	given number of occurrences	`[chars]`	character set	`[char-char]`	character range
`[^chars]`	invert character set	`\b,\B,\d,\D, \s,\S,\w,\W`	predefined char sets for word boundaries, digits, spaces, and word characters	`\0, \1, ...` `"$0", "$1"`	back-references

Example:
```
var s = "mississippi";
s = s.replace(/i(.)\1/g, "ee$1");    // "meeseeseepi"
```