JQUERY

```
//--- COMMON JQUERY SELECTORS ---//
          // get element by id
          $("#ElementID").whatever();

          // get element by css class
          $(".ClassName").whatever();

          // get elements where id contains a string
          $("[id*='value']").whatever();

          // get elements where id starts with a string
          $("[id^='value']").whatever();

          // get elements where id ends with a string
          $("[id$='value']").whatever();

          // get all elements of certain type (can use "p", "a", "div" -
any html tag)
          $("div").whatever();


          //--- JQUERY TOGGLE/SHOW/HIDE ---//
          // toggle hide/show of an element
          $("#DivID").toggle(1000);

          // do something when animation is complete
          $("#DivID").toggle(1000, function () {
              alert("Toggle Complete");
          });



          // hide an element
          $("#DivID").hide(1000);

          // do something when animation is complete
          $("#DivID").hide(1000, function () {
              alert("Hide Complete");
          });



          // show an element
          $("#DivID").show(1000);
```

```
  // do something when animation is complete
  $("#DivID").show(1000, function () {
      alert("Show Complete");
  });

//--- JQUERY SLIDE - SLIDE AN ELEMENT IN AND OUT ---//
  // toggle slide up and down
  $("#DivID").slideToggle(1000);

  // do something when animation complete
  $("#DivID").slideToggle(1000, function () {
      alert("Slide Toggle Complete");
  });

  // slide up
  $("#DivID").slideUp(1000);

  // do something when animation is complete
  $("#DivID").slideUp(1000, function () {
      alert("Slide Up Complete");
  });

  // slide down
  $("#DivID").slideDown(1000);

  // do something when animation is complete
  $("#DivID").slideDown(1000, function () {
      alert("Slide Down Complete");
  });

//--- JQUERY FADE - FADE AN ELEMENT IN, OUT & TO ---//
  // fade in
  $("#DivID").fadeIn(1000);

  // do something when animation complete
  $("#DivID").fadeIn(1000, function () {
      alert("Fade In Complete");
  });

  // fade out
  $("#DivID").fadeOut(1000);

  // do something when animation is complete
```

```
            $("#DivID").fadeOut(1000, function () {
                alert("Fade Out Complete");
            });

            // fade to (fades to specified opacity)
            $("#DivID").fadeTo(1000, 0.25);

            // do something when animation is complete
            $("#DivID").fadeTo(1000, 0.25, function () {
                alert("Fade To Complete");
            });


      //--- ANIMATE (EXAMPLE USES OPACITY, BUT CAN USE ANY CSS PROPERTY.
---//
            //--- NOTE SOME MY REQUIRE THE USE OF A PLUGIN SUCH AS JQUERY
COLOR ANIMATION PLUGIN. ---//
            $("#DivID").animate({ opacity: 0.25 }, 1000);

            // do something when animation complete
            $("#DivID").animate({ opacity: 0.25 }, 1000, function () {
                alert("Opacity Animation Complete");
            });

            //--- ADD & REMOVE CSS CLASSES ---///
            // add css class
            $("#DivID").addClass("newclassname");

            // remove css class
            $("#DivID").removeClass("classname");

            // add & remove class together
            $("#DivID").removeClass("classname").addClass("newclassname");

            // add & remove multiple classes
            $("#DivID").removeClass("classname
classname2").addClass("newclassname newclassname2");

           //--- GET & SET TEXTBOX VALUE ---//
            //--- CAN ALSO BE USED ON ANY OTHER ELEMENT THAT HAS A VALUE
PROPERTY ---//
            // get the value of a textbox
            var TextboxValue = $("#TextboxID").val();

            // set the value of a textbox
```

```
$("#TextboxID").val("New Textbox Value Here");

//--- GET & SET HTML OF ELEMENT ---//
// get element html
var DivHTML = $("#DivID").html();

// set element html
$("#DivID").html("<p>This is the new html</p>");

//--- GET & SET TEXT OF ELEMENT ---//
// get text of element
var DivText = $("#DivID").text();

// set text of element
$("#DivID").text("This is the new text.");


          //--- GET & SET ELEMENT'S WIDTH & HEIGHT
// get element height
var ElementHeight = $("#DivID").height();

// set element height
$("#DivID").height(300);


// get element width
var ElementWidth = $("#DivID").width();

// set element width
$("#DivID").width(600);


          //--- CHANGE AN ELEMENT'S CSS PROPERTY ---//
$("#DivID").css("background-color", "#000");
$("#DivID").css("border", "solid 2px #ff0000");
```

CANVAS
HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Solucao</title>
        <script src="jquery.min.js" type="text/javascript"></script>
        <script src="solucao.js" type="text/javascript"></script>
        <link href="solucao.css" type="text/css" rel="stylesheet"/>
```

```
	</head>
	<body>
		<canvas id="canvas" width="400" height="300"></canvas>
		<div>
			<button id="start">Start Animation</button>
			<button id="stop">Stop Animation</button>
		</div>
	</body>
</html>
```

CSS

```
body {
	font-family: Verdana;
	font-size: small;
}

canvas {
	cursor: pointer;
	border: 1px solid black;
}
```

JS

```
function Shape(canvas, x, y){
	if(canvas){
		this.context = canvas.getContext("2d");
		this.x = x;
		this.y = y;
	}
}

Shape.prototype.getColor = function(){
	if(this.color == undefined){
		this.color = "green";
	}
	return this.color;
}

function Square(canvas, x, y, side, rotation) {
	Shape.call(this, canvas, x, y);
	this.side = side;
	this.rotation = rotation;
}

Square.prototype = new Shape(); // clone(Shape.prototype);
Square.prototype.constructor = Square;

Square.prototype.draw = function () {
   // Draw the circle.
   this.context.save();
   this.context.beginPath();
   // Rotation
```

```
    this.context.translate((this.x)/2, (this.y)/2);
    this.context.rotate(this.rotation * Math.PI / 180);
    this.context.rect(0, 0, this.side, this.side);
    // Style
    this.context.fillStyle = this.getColor();
    this.context.strokeStyle = "black";
    this.context.lineWidth = 1;
    this.context.fill();
    this.context.stroke();

    this.context.restore();
};

// This array hold all the circles on the canvas.
var shapes = [];

var canvas;
var context;

$(document).ready(function() {
  /*

  canvas = $('#canvas');
  context = canvas.getContext("2d");
  $('#start').click = startAnimation;
  $('#stop').click = stopAnimation;
  initCanvas();
  */
  canvas = $("#canvas")[0];
  context = canvas.getContext("2d");
  $("#start").click(startAnimation);
  $("#stop").click(stopAnimation);
  initCanvas();
});

function initCanvas(){
        addShape(100, 100);
        addShape(150, 100);
}

function addShape(x, y) {
        var size = 20, rotation = 0;
        if(shapes.length >= 2){
                size = shapes[shapes.length-2].side * 1.25;
                rotation = shapes[shapes.length-2].rotation + 25;
        }

        // Create the new shape.
        var shape = new Square(canvas, x, y, size, rotation);
        // Store it in the array.
        shapes.push(shape);
        // Draw the canvas.
```

```
        drawShapes();
}


function drawShapes() {
  // Clear the canvas.
  context.clearRect(0, 0, canvas.width, canvas.height);

  // Go through all the shapes.
  for(var i = 0; i < shapes.length; i++) {
    var shape = shapes[i];
    shape.draw();
  }
}

var interval1;
var interval2;
var interval3;

function startAnimation(){
        interval1 = setInterval("addShape(100, 100)", 1000);
        interval2 = setInterval("addShape(150, 100)", 1000);
        interval3 = setInterval("restartAnimation()", 8000);
}

function restartAnimation(){
        clearInterval(interval1);
        clearInterval(interval2);
        clearInterval(interval3);
        shapes = [];
        initCanvas();
        startAnimation();
}

function stopAnimation(){
        clearInterval(interval1);
        clearInterval(interval2);
        clearInterval(interval3);
}

function randomFromTo(from, to) {
  return Math.floor(Math.random() * (to - from + 1) + from);
}
```

Java Script OO

```
function Mammal(name){
        this.name=name;
        this.offspring=[];
}
Mammal.prototype.haveABaby=function(){
        var newBaby=new Mammal("Baby "+this.name);
```

```
        this.offspring.push(newBaby);
        return newBaby;
}
Mammal.prototype.toString=function(){
        return '[Mammal "'+this.name+'"]';
}


Cat.prototype = new Mammal();       // Here's where the inheritance occurs
Cat.prototype.constructor=Cat;      // Otherwise instances of Cat would have a constructor of
Mammal
function Cat(name){
        this.name=name;
}
Cat.prototype.toString=function(){
        return '[Cat "'+this.name+'"]';
}


var someAnimal = new Mammal('Mr. Biggles');
var myPet = new Cat('Felix');
alert('someAnimal is '+someAnimal);   // results in 'someAnimal is [Mammal "Mr. Biggles"]'
alert('myPet is '+myPet);             // results in 'myPet is [Cat "Felix"]'

myPet.haveABaby();                    // calls a method inherited from Mammal
alert(myPet.offspring.length);        // shows that the cat has one baby now
alert(myPet.offspring[0]);            // results in '[Mammal "Baby Felix"]'
```