Week 3

MORE ABOUT JAVASCRIPT & CANVAS

MORE ABOUT JAVASCRIPT

IF ... ELSE ...

```
var a = Math.random() // 0 <= a < 1

if (a >= 0.5) {
    // do something
} else if (a >= 0.25) {
    // do something else
} else {
    // this will catch everything else
}
```

SWITCH

```
var a = Math.floor(Math.random() * 5) // integer 0 ~ 4

switch (a) {
    case 0:
        // will do this if a == 0
        break // DO NOT forget the break for each case
    case 1:
        // will do this if a == 1
        break
    default:
        // any case not specified will go to here
        // e.g. 2, 3, 4
        break
}
```

FOR LOOPS

```
for (var i = 0; i < 10; i++) {
    console.log(i)
}
// 0,1,2,3,4,5,6,7,8,9
```

What each part means...

```
var i = 0; // this is the start condition

i < 10; // this is the end condition

i++ // this happens after each loop
```

WHILE LOOPS

```
var a = 0

while (a < 10) {
    console.log(a)
    a++
}
// 0,1,2,3,4,5,6,7,8,9
```

Things inside a while loop will be executed repeatedly until the condition evaluates to false. In this case, when a is equal to or larger than 10.

ARRAYS

```
var arr = [] // this is called 'Array literal', and is the
             // recommended way to create an array.
             // arr is now an empty array with length 0.
arr.length // 0

// you can also directly create an array with elements in it
var arr2 = ['a', 'b', 'c', 'd', 'e']
arr2.length // 5

// You access elements in an array with square brackets syntax
// using integer index
arr2[0] // 'a' index starts at 0!
arr2[3] // 'd'
arr2[arr2.length - 1] // 'e' the last element
```

ARRAYS

JavaScript Arrays can hold anything, even elements of different types.

```
var arr = [1, 'two', true, { data: 99 }]
```

To add new elements to an existing Array:

```
arr.push('new stuff') // 'new stuff' will be pushed to the end
                      //  of the array
arr.length // 5
arr[4] // 'new stuff'
```

ARRAYS

There are many other useful methods to manipulate Arrays.

Mutator Methods

`push(), pop(), shift(), unshift(), splice() ...`

Accessor Methods

`concat(), join(), slice() ...`

ES5+ only

`indexOf(), lastIndexOf(), forEach() ...`

TIMERS

Do something on an interval:

```
var intervalID = setInterval(function () {
    console.log('this will be logged every second')
}, 1000)

clearInterval(intervalID) // stop it
```

Do it only once:

```
var timeoutID = setTimeout(function () {
    console.log('this will happen after 1 second')
}, 1000)

clearTimeout(timeoutID) // cancel it before it happens
```

TIMERS

Do something on an interval:

```
var intervalID = setInterval(function () {
    console.log('this will be logged every second')
}, 1000)

clearInterval(intervalID) // stop it
```

Do it only once:

```
var timeoutID = setTimeout(function () {
    console.log('this will happen after 1 second')
}, 1000)

clearTimeout(timeoutID) // cancel it before it happens
```

TIMERS

```
function animate () {
    // e.g. move something by 1 pixel...
}
var interval = 1000 / 60 // 60 frames per second
```

Both can be used for animation:

```
var intervalID = setInterval(animate, interval)
animate ()
```

OR

```
function loop () {
    setTimeout(loop, interval)
    animate()
}
loop()
```

TIMERS

JavaScript timers are not always accurate. In short, the program could be busy when your timer function is called, resulting in a slight delay.

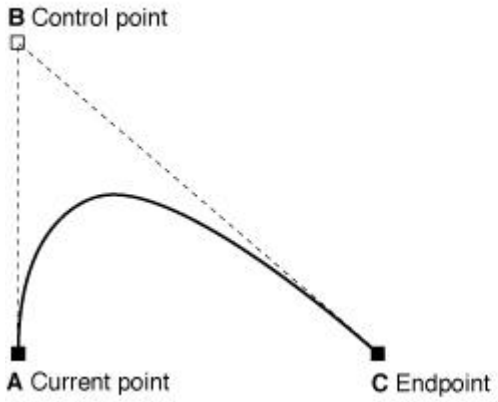
A new API automatically compensate for that delay:
`requestAnimationFrame()`

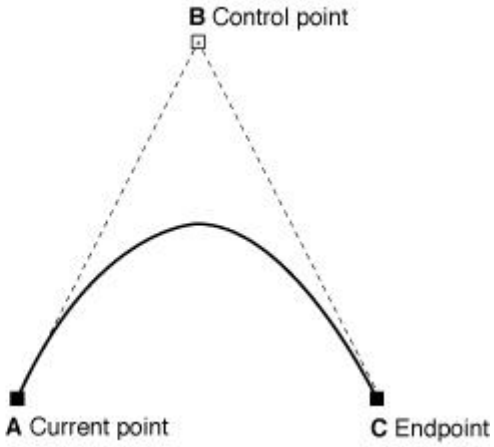
```
function animate () {
    requestAnimationFrame(animate)
    // move something by 1px...
    // or use the timestamp for accurate calculation
}
```

MORE ABOUT CANVAS

DRAWING CURVES

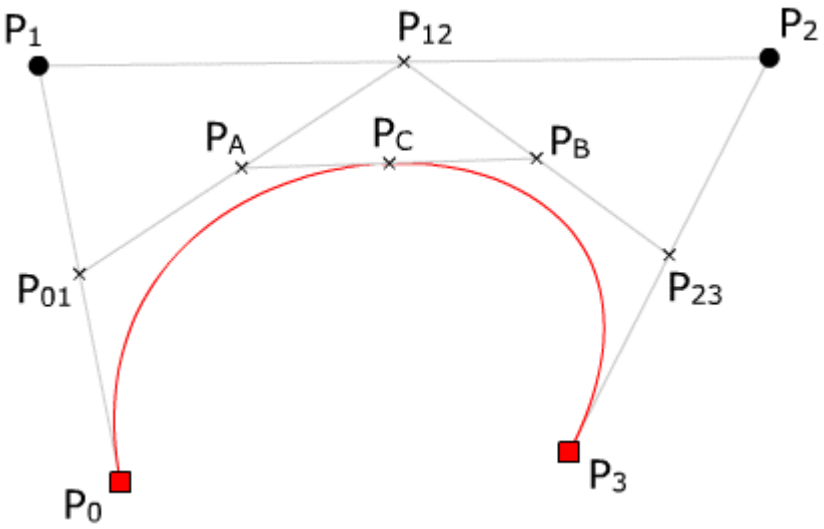
```
quadraticCurveTo(cpx, cpy, x, y)
```





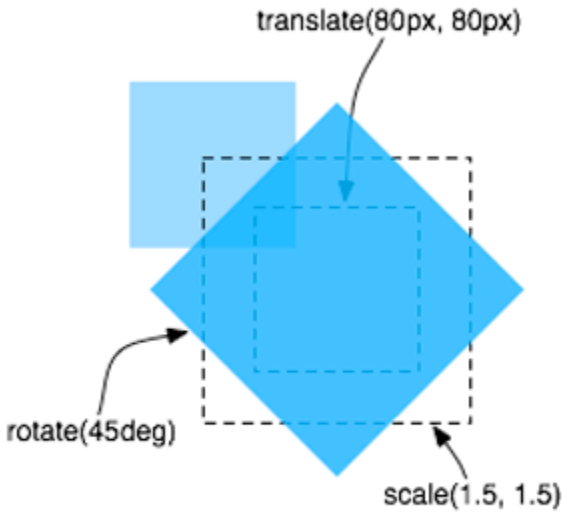
DRAWING CURVES

```
bezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y)
```



TRANSFORMATIONS

```
scale(x, y)
rotate(angle)
translate(x, y)
```



Compound Transform

CONTEXT STATE

```
ctx.save()

// do transforms
// set styles
// draw stuff

ctx.restore()

// transforms and styles are restored to the state before ctx.
save()
```

Design AND Technology

Think on both sides.

Generative Art Inspirations

Casey Reas
Joshua Davis
Justin Windle