Based on student mid-semester evaluations, the following is a quick reference. I'm going to organize it around the lecture topics, which is perhaps maybe not the best way to organize information, but I figure it will be the easiest way for you to get ahold of what you need and relate it to the screencasts if you need more detail.

Set #1 - Introduction to the course. Flash Basics, Personal introductions

- Huge difference between .swf and .fla files—make sure you save and submit your .**fla** files

Set #2 - Drawing Tools, Animation (Tweens)

- Naming conventions:
  - For: variables, layers, files, library items (graphics, components, movie clips, buttons, etc . . .)
  - No spaces
  - Start with letter
  - Use "_" or capital letters to separate words:
    - home_button
    - homeButton
  - Flash is case sensitive
- Tween differences
  - Classic = object moves/scales/changes alpha etc . . . and maintains basic shape. This is the old motion tween from AS3, you use it with grouped objects. It can come in handy if you want several things to follow the same motion guide.
  - Motion = object moves/scales/changes alpha etc . . . but maintains basic shape. Use with: grouped object, symbols, pseudo-symbols. *Only way* to use the advanced motion editor tools.
  - Shapte = object moves/scales/changes alpha etc . . . and can "morph" into a different shape. Use with: simple vector art

Set #3 - Advanced Drawing Tools, Lighting Effects, Shadows, Alpha & Gradient Transform

- Creating depth with changes in size, "bur" effect, two kinds of lighting effects "spot" lights with gradient fills on an object, or gradients on a separate overlay object, similar use for shadows.
- Gradient transform tool, remember if you want to avoid hard edges, the grandient needs to be complete within the shape of the object you want.
- Don't forget to select **both** the fill (or stroke) on your object and then select the fill (or stroke) in your color window if you want to actually apply changes to your object.

Set #4 - Graphic Symbols, Nesting Symbols, Onion Skinning & Edit Multiple Frames, Motion Guides, Easement & Custom Easement

- Symbols, reuse and "blueprint" metaphore
- Tools for manipulating the timeline and adjusting/modifying tweens.
- Best bet for moving a shape/classic tween is to edit multiple frames, lock things you don't want to adjust, then draw a selection rectangle around it all.

Actionscript

- Next week (#5) marks a major shift away from graphical kinds of things and into ActionScripting. While we start of somewhat slow we do hit the ground running and things ramp up quickly from here. If you follow these simple guidelines you can avoid getting yourself into a world of hurt down the road (some won't make sense right now, but get them into your head anyway):
  1. write a **small bit** of code **then test** and make sure it works
  2. **copy|paste** your variable names and instance names or use the target path and have Flash write them for you. 4 out of 5 dentists agree that **typing is bad for you**.
  3. when you start a function or an if statement put in your () and {} and **make sure they match** before moving on with your life.

Set #5 - Button Symbols, Actionscripting (including using functions, creating custom functions, and listeners)

- Creating buttons, different visual states.
- Buttons scripting, each button needs:
- Instance name – select and apply in properties window. For example:

```
windowDocBtn
```

- A listener applied to that instance. For example:

```
windowDocBtn.addEventListener(MouseEvent.CLICK,
onWindowDocBtnClick);
```

- A function that is run when that listener is triggered. For example:

```
function onWindowDocBtnClick(e:MouseEvent) {
   gotoAndStop(2);
}
```

- Timeline functions

```
stop(); //stops playhead
nextFrame(); // takes you to the next frame in the timeline.
prevFrame(); // takes you to the previous frame in the
timeline.
gotoAndStop(1); // takes you to the first frame and stops the
timeline. (can replace 1 with any number you want).
```

Set #6 - Movie Clip Symbols, Dot Syntax, Object Heirarchy and Pathing.

- Movie clip timelines run *independently* of the main timeline and each other.
- Can change properties of objects (like the text inside a text field):

```
faceState.text = "I'm sad!";
```

- Can path "up" the hierarchy *relative* to where you are:

```
(parent as MovieClip).faceState.text = "sad";
```

- Can path "up" the hierarchy by an *absolute* jump to the top:

```
(root as MovieClip).faceState.text = "sad";
```

- Pathing works for anything, updating something like a text property or changing the timeline of a movie clip:

```
(parent as MovieClip).sky.stop();
```

Set #7 - Variables, Simple Control Structures (if/then/else), Comments, Debugging

- Variables store information, they have a name, a type, and they have the information they store.
- Properties are variables that are already built into Flash (or they're variables in custom classes you've created—but we won't cover that here). They have a name that you can't change, they have a type that you can't change, they have information that updates automatically which you can "read". Some you can change (e.g. "write" to, others are "read" only). Following are two example properties of the MovieClip class built into Flash.

```
currentFrame //this gives you the playhead's frame
```

```
totalFrames //this gives you total number of frames in a
            //timeline.
```

- For **variables** . . . You declare them once:

```
var recentPhoto:int;
```

- For **variables** and **some properties** you update (or "write") information they contain any time you like:

```
recentPhoto = currentFrame;
```

- For **variables** you can do both at the same time:

```
var recentPhoto:int = 1;
```

- If statements let you run code only when certain conditions are met, they have a condition and a block of code:

```
if(face.currentFrame <= 3) {
   faceState.text = "I'm sad.";
}
```

- If statements can be combined with "else if" and "else". "else if" lets you specify another condition if the first one is not true, can do this as many times as you like. "else" becomes the catch all if all conditions fail:

```
if(face.currentFrame < 4) {
   faceState.text = "I'm sad.";
} else if (face.currentFrame > 8) {
   faceState.text = "I'm happy.";
} else {
   faceState.text = "I'm ok.";
}
```

- You need to be careful with variables and setting initial values—if you run through that code again it may write over the top of values you had later updated. To avoid this, add the kickoff value only if the variable hasn't already been given something that's not a default:

```
if(recentPhoto == 0) {
   var recentPhoto:int = 1;
}
```

- See the lecture handouts for a complete list of default variables for each type of variable.

Set #8 – Embedded Items

- Idea is to keep things low stress, check learner's understanding right after you give them something new.
- Need 3 types of feedback: invalid, correct, incorrect
- Incorrect feedback should be: 1) clearly state what happened (wrong), 2) give a hint about what the correct choice is (be corrective) and 3) encourage them to try again (be recursive).