

The project documentation has two purposes. The first is functional, and the goal here is to provide enough information about your project that another developer or team of developers could easily come in and make revisions or extensions (many of you discuss in the scope section of your work plan more features and/or content than you intent to implement). The second purpose of your project documentation is to serve as a portfolio piece when you go out and look for jobs. Following are some strong suggestions of things that will be good to have in your project documentation:

- Give credit where credit is due, you should probably mention any subject matter experts you worked with (most likely your clients) and any material, such as actionScript from an outside source that you used or adapted as part of the project.
- In that same spirit *if you are working on the final project as a group*, talk about the team members and the role(s) each of you played.
- Next steps – discuss the next logical area of development for your project. This should be pretty easy to put together from the scope section of your work plan, but might also include ideas that emerged as you put the project together. Finally, this is a great place to highlight things that were promised in your work plan that did not get done due to time constraints.

The remaining sections are a little more technical. This portion can be more of a “living” document if you choose, relying on good comments embedded within your code rather than a separate explanation of what your code does (although in both cases you will need a still need a broad overview separate from your .fla file of what is going on). Make the assumption that your end reader is familiar with Flash and knows a little bit about ActionScript, but *do not* make the assumption that they are familiar with your code. Some good things to include here:

- Timeline Structure – How is your project organized? (both in terms of layers, and major “events”).
- Flow chart – This can be an excellent means for diagramming some of the main system interactions. Depending on the complexity of your project, you might want to use a series of small flowcharts or one large flow chart. (If your layers are well organized and named well, you might collapse the timeline into the flow chart, by providing the frame number(s) of the various portions of your project). Pg. 54 in the option text has an example flow chart, in addition to the exemplar project that is on the course web page.
- Naming conventions – What are your naming conventions for variables, layers, library items, etc . . . Some of these might be general (all lower case, with subsequent words capitalized) and some might be specific to particular items in your project (Dynamic text fields used to show text feedback are given an instance name in of “feedBackTextXX”, where “XX” refers to the current frame number)
- Important variables – List and discuss the key variables in your project (this will likely be limited to any session, client, or application variables).
- Explanation of critical code segments (again, using comments is acceptable). Walk a programmer through how your code works for complex sequences (such as drag and drop interactions w/ feedback).
- Known bugs – Discuss known limitations of your program (ways that users can “break” it), or ways that it performs in an unexpected manner.
- Video compression – *If you use video in your project*, write down the settings you selected when you embedded it in your flash development file.

Other thoughts:

- Be concise, if you find you are repeating yourself—you should probably just explain it once and talk about other times that it happens (for instance, don't walk through how you implemented drag and drop fifteen times).
- Similarly, don't feel the need to give a blow by blow on precise pixels for where you placed elements. If you set up your project timeline well and you have a good overview, someone can figure out what they need to know.
- If you find yourself spending more than 2-3 hours on your project documentation you are probably going into too much detail.

**Project Documentation Front End** – As you have seen with the exemplars, you will be responsible for creating a simple web page (named index.html) that allows other class members (as well as other interested parties) to take a look at your hard work. This front end should consist of the following parts:

- Project Title
- System requirements – you can pull this out of your work plan if you already wrote this into your limitations section, but let your audience know this up front.
- Primary development platform – Was the work done on a PC or on a MAC? This is more for me as a grader, so that I can look at your development file(s) through the same lens.
- Link to work plan
- Link to project documentation
- Link to related media files (assuming that you used graphics, sound, and/or video, this would just be a link to a folder containing these files prior to import into flash so they can be edited and adjusted). I have had at least two students as me for their full projects after taking the class, and they were both grateful to have the original media as well.
- Link to development (.fla) file(s).
- Link to exported (.swf) file(s) – if you promised web delivery, then link them to a web page with the .swf placed in-line. Flash will do this for you, but we did not cover this in class, I'm happy to help with this and with the project documentation front end if you need it, either during class consultation time or a scheduled time outside of class.

**Final Project** – These are the development files for your final project along with the exported .swf (or .swfs). Most of what I want is described below in the rubric. I will say that although you won't lose points for using scenes as a way to break up your file this does cause performance problems. The preferred method is to use movie clips as opposed to scenes.

- Deliverables: A folder (which will likely have sub-folders) of everything mentioned in the *project documentation front end* section above.
- File Naming convention: Use the following naming convention for your folder name (zip archives should have this folder at the top level, with all files and sub/folders contained within) finalProjectYourName, so if your name was Sam Walker your folder would be named finalProjectSamWalker.

### Assessment Rubrics

Your final project **documentation** will be graded according to the following criteria:

Criteria	Points
Is your project documentation complete (would someone be able to recreate the structure of your project looking only at the final .swf and your documentation—more importantly, would they be able to easily find what they were looking for if they were forced to come in and make changes?) Does it contain all of the relevant portions as outlined above?	30 points
Is your project documentation accurate?	25 points
Is your project documentation professional (free of typographical and grammatical errors)?	25 points
Does your project documentation contain a functional front-end as described above?	20 points
Total	100 points

Your **final project** will be graded according to the following criteria:

Criteria	Points
Do you use a consistent naming convention for layers, symbols, and pseudo-symbols? Do all of your layers have a meaningful name? (e.g. "layer 1" is not an option)	10 points
Is your project easy to change and update? <ul style="list-style-type: none"> <li>• you should have only the number of instances you absolutely need for each symbol or element of the project.</li> <li>• you should use consistent tab stops for your code—don't be shy about using the autoformat button in the actions</li> </ul>	50 points

<p>window.</p> <ul style="list-style-type: none"> <li>Finally, you should not have any "magic numbers." For the purposes of this class, a magic number is defined as a value in ActionScript that is used in more than one piece of code, but not updatable in one place. If you find yourself typing out a number in more than one location, create a variable and grab the value from the variable instead.</li> </ul>	
Do you have a well organized timeline (related layers are near each other, elements are where they are promised).	30 points
Is your project free of all syntax errors, and major logic errors (operating in ways that are unexpected)?	10 points
Total	100 points