Guidelines and deliverables for tutorials

Your assignment is to write a tutorial or manual and give a talk on your topic. For the former, you should expect to write between 5 and 10 pages; however, your document may run longer depending on the number of screenshots, illustrations and/or code examples you include. Some of the topics are extremely broad and complex (such as EJBs and J2EE) and some are extremely focused (C# delegates and events). Obviously, in the former cases you will not be able to provide the coverage and detail that you might be able to with a topic of smaller scope.

The talk should be 12 to 15 minutes and will generally not be able to cover everything that is in your written document; do your best to convey the main ideas. Both will be graded on technical content, organization, and presentation (the writing for the former and the oral presentation for the latter). *Approximate* weights are as follows:

- 35% Written tutorial technical content and code examples
- 25% Written tutorial organization of document and quality of writing
- 20% Talk content and organization
- 20% Slides and oral presentation

Deliverables include your written manual in pdf form, an archive of accompanying code examples if applicable, and your presentation slides. **Deliverables are due at the time of your presentation, except that those who present during the first week will be allowed until the following Monday to turn in everything other than the slides.**

Manual/tutorial

Your document should be relatively formal. Specifically,

- Above all: be direct, clear, and concise.
- Avoid unnecessary words, irrelevant concepts, and any other forms of b.s.
- Be careful to define terminology and use it consistently.
- Understand who your audience is and make appropriate assumptions.
- Avoid using the passive voice.
- You are not writing a blog, so don’t use “I” or talk about your experiences and opinions (the use of “we” and “you” are appropriate, within reason, especially when you are writing how-to instructions).
- Avoid slang, sarcasm, flippancy, and evangelism.
No doubt you have had to read quite a bit of technical documentation, and you know how frustrating it can be. Hopefully you can draw on that experience to think of ways to smooth the learning curve for your readers as much as possible.

You may be presenting on a library, tool, language feature, or technique. However, in all cases you should expect that there will be some mix of explanations of concepts, examples, and step-by-step instructions. In general you should try to introduce concepts as needed and support them with concrete examples and step-by-step instructions. Don’t get these mixed up. (That is, don’t try to introduce a new concept in step 5c of a set of step-by-step instructions, and conversely, if what you’re really doing is giving step-by-step instructions, spell out the steps using an itemized list and don’t bury it in a paragraph).

In most cases, you should create some original code examples that the reader can try out, and include them separately. In general you should be extremely selective about including code listings in the text of your document, and try to highlight just what’s important for what you are discussing. If a code example is more than 10-12 lines, make it a separate figure.

You should expect to have:

- Table of contents (this is trivial to create if you use your word processor’s headings correctly).
- Description of prerequisites and who the document is written for.
- Introduction and overview. This is extremely important. Give the reader some context for the problem being solved by this tool or library. What is it for? Is it language feature for general-purpose programming, an enterprise tool, a technique, or what? What problem does it solve? Would it make sense to briefly describe some other ways to solve the same problem?
- Sections with accurate, descriptive titles. Separate introduction of concepts from how-to instructions.
- When including illustrations, use labeled figures and refer to them. If you use your word processor’s features for cross-references this is easy.
- A separate section describing your sources and further references for the reader. Be scrupulous in attributing quotations, illustrations, or code examples to sources.

**Talk**

You will have a strict 15 minute time limit to allow some time for questions, discussion, and editorializing by your instructor. Practice your talk and time yourself, aiming for about 13 minutes. If you have roommates or pets, present your talk to them for feedback. Try to be
comfortable with the material to the point that you don’t have to rely on memorization. (However, memorizing your talk is much better than reading it from your slides or notes!)

• It is very, very strongly suggested that you prepare slides (PowerPoint or other) to help present your main ideas. Do not use screens full of code in your IDE as a presentation format.

• Remember that the #1 mistake in preparing slides is to use too many words. You want the audience to be listening to you, not furiously reading your slides. Carefully choose a few words or phrases that will help the listener focus on what you are saying rather than distract from what you are saying.

• Use pictures or illustrations if you can. Choose code examples judiciously, make them large enough to read and visually highlight what it is that illustrates your point.

• Use animation only if it is helpful somehow to getting your point across; otherwise it’s usually just a distraction.

• Your presentation should include a live demo if at all possible. It is ok, of course, to bring up and/or use an IDE as part of your demo. This is where many presentations crash and burn, so do this sparingly. Be aware of what the audience can see and follow: make font large enough to see, make your mouse/cursor operations slow and deliberate, and be sure to pause and visually highlight the line or lines you are trying to emphasize or discuss.