

## **Tutorial Assignment**

For this assignment, write brief, clear, to-the-point instructions for carrying out a technical task using software or writing code. Pick something that you learned during the process of Project 1 or something that you know you want to learn in preparation for Project 2—how to perform a task in Dreamweaver, Composer, NVU, Frontpage, Photoshop, or a multimedia program such as Flash or Moviemaker. Or if you learned to carry out a technical task directly with code—html, java script, CSS—that you think an audience of future 505 students might need to carry out, then that task is appropriate for the tutorial. Their primary purpose will be to build a web portfolio just as you are. What are the key tasks they need to carry out that assignment?

In some cases you can remake instructions from other tutorials (the DW book, workshop tutorials, or other tutorials on the web). One of the key reasons for this assignment is to solidify your learning. Even though you may have already carried out a task using another tutorial, writing your own version of those instructions reinforces what you've learned. It also could provide a better tutorial for other students because as a student you understand the particular rhetorical situation of the class and know the kinds of tasks and detail that other student's need.

The tutorial should be in html format. Resist the temptation to use PDF, a Word doc, or some other format. Part of the assignment is to get more experience designing usable web pages.

### **Define the Task:**

Begin the process by defining the task you want users to perform. Experts at writing instructions always begin with a task analysis that outlines the tasks a user will need to perform in a particular situation.

Start by identifying the main task: assemble a grill, for example. Such a general task can be broken down into several first-level sub-tasks:

1. Locate all parts.
2. Get the required tools.
3. Lay out the parts in order.
4. Assemble parts into smaller units.
5. Assemble smaller units into larger units.

These first-level sub-tasks can also have various sub-tasks. The first-level sub-tasks can function as headers in your tutorial with the second-level sub-tasks bulleted underneath them.

In larger technical communication situations, you might interview or observe users to determine the key tasks involved. To get started on your instructions, brainstorm (list) the first-level sub-tasks involved in your chosen procedure and think specifically about their proper order or sequence. Number those and then work through each one in the software program. Add in the

second-level sub-tasks as you work through the sequence. Be as detailed as possible. Revise your first-level sub-tasks if needed when working through the process.

To have a successful tutorial, you want a main task that requires more than 2 or 3 first-level sub-tasks. Your initial task-analysis should allow you to determine whether or not your main task is appropriate for the assignment. In general, 4 to 6 first-level sub-tasks with a few second-level sub-tasks under each will be sufficient.

### **Follow this Format:**

The typical elements for most tutorials are:

Title: Keep the title brief and most importantly to the point. Use it to describe the main task to be carried out (e.g., How to Assemble a Grill). For this assignment, be really clear about what version of the software you are using.

Overview: When the tutorial carries more than 2 or 3 tasks, as this one should, you may need a brief statement that summarizes the process the user is about to work through. Sometimes this is as basic as setting up the main task and first-level sub-tasks (e.g., To assemble a grill you will need to locate all parts, get the required tools, lay out the parts in order, assemble parts into smaller units, and assemble smaller units into larger units). For something like a tutorial for class, you may also need to address the rhetorical context, purposes, or design issues related to carrying out this task. These might come out in the choice of task, how you set up the task in the overview, or even the actual steps you have the users work through. (For example, I often include downloading material as a part of getting ready for tutorials. Or I will have students look at other examples to get a sense of what the tutorial is after. Or, I will build in setting up particular tables because I know they will be used later for particular design functions.)

Instructions: Step-by-step instructions should be as detailed as possible. Try to fill in all gaps for all possible scenarios. Certainly consider what assumptions the user will make about the process, but also what quirks in the program might arise or what layout or design issues in the interface might contribute to an inability to interpret your instructions. Also consider:

- **Audience**: Imagine the users' prior knowledge with regard to the task. Are they novice or expert? Do they know past versions of this particular software? Are they familiar with other similar software but don't know this one? Take this particular 505 class as representative of the range of students who may use the tutorial in the future.
- **Structure**: Most instructions are written in numbered lists so that users can easily see the sequence and remember where they left off in the sequence. Headers are also used to highlight key steps in the process. Don't overuse them. Make sure they have optimum rhetorical impact. Also use bulleted lists to highlight steps in a process. Use bullets when the second-level sub-task list is short. Like links in navigation, 4 to 7 bullets are optimal for usability.

- **Length:** Brief instructions should not be too long. For this assignment, you will probably want to keep the tutorial to a single web page. Multiple pages will present a variety of usability issues. When working with one page, keep the basic web design rules in mind. Users will have to scroll, so make sure they don't get lost in the process. Use bold for clearly articulated headers, use numbered sequences so users don't get lost, and make sure long sequences don't stretch beyond the length of one screen without some kind of orienting signposts.
- **Language:** Use the same terms to refer to the same parts or steps. Don't confuse users by using terms inconsistently. For example, you might use *click* to refer to clicking a "button" in a program and use *select* when selecting an item from a drop down menu. Make sure that you pay careful attention to this level of detail in your writing. Also use imperative voice and action verbs. Always leave out the second person "you" and begin with the verb to highlight the action involved.

**Diagrams:** Almost all instructions or tutorials have diagrams or screen shots to better illustrate the process or the parts involved. Screen shots should be used when a button to click or a series to follow is difficult to describe in words. You don't want to overuse screen shots or misuse screen shots. Too many screen shots can make it difficult to see a numbered process if the image takes up too much space above the fold. Random screen shots that don't carry a direct and clear rhetorical or informational purpose just take up space and detract from images that do carry a rhetorical or informational purpose. (To capture a screen shot on a PC: hold down the Alt and Print Screen keys, open an image-editing application, start a new file, and paste the image into the new file. Crop down to the desired image in an image editing program.)

**Follow up:** It is always a good idea to show the user where to go for further information. If you are designing instructions for a company, the company contact info or web site would go here. For our purposes, give the user links to related tutorials online. These might be for the same procedure you are covering, or they might cover related material to your main task. It is always a good idea to give multiple links because sites come and go everyday and you want to increase the chance that some of your links will still be good when future students use the tutorial. (Use Google liberally to find examples to follow and links to include.) Also, be sure to put a date at the end of your tutorial. This will let a future audience know how current the tutorial is.

### **Test Your Tutorial:**

Most technical documentation should be proofread and tested by an audience. This is especially the case with instructions—you won't know if they work unless you have a group of users use it. For this assignment, have at least one friend or relative try out the tutorial before you submit it. Watch him/her closely and write down the things that show up as confusing or that cause roadblocks in the process. Resist intervening in the user's process. If you have to explain something verbally, then write it down because it should be worked back into the tutorial. After taking notes, go back and revise the document accordingly.