Pre-Class material
Introduction (2 min)
Self-editing (10 min)
Organizing large docs (20 min)
Illustrating (10 min)
Creating sample code (10 min)
Summary (1 min)

Organizing large documents

On this page
When to write large documents
Organize a document
Outline a document
Outline exercise
Introduce a document
Introduction exercise
Add navigation
Prefer task-based headings
•••

Estimated Time: 20 minutes

How do you organize a large collection of information into a cohesive document or website? Alternatively, how do you reorganize an existing messy document or website into something approachable and useful? The following tactics can help:

- Choosing to write a single, large document or a set of documents
- Organizing a document
- Adding navigation
- Disclosing information progressively

When to write large documents

You can organize a collection of information into longer standalone documents or a set of shorter interconnected documents. A set of shorter interconnected documents is often published as a website, wiki, or similar structured format.

Some readers respond more positively than others to longer documents. Consider the following perspectives from two hypothetical readers you're writing documentation for:

- Hong finds reading long documents difficult and disorientating. He prefers to use site search to find answers to his questions.
- Rose is comfortable navigating large documents. She often uses the built-in page search feature in her web browser to find useful information on the current page.

So, should you organize your material into a single document or into a set of documents in a website? Consider the following guidelines:

- How-to guides, introductory overviews, and conceptual guides often work better as shorter documents when aimed at readers who are new to the subject matter. For example, a reader who is completely new to your subject matter might struggle to remember lots of new terms, concepts, and facts. Remember that your audience might be reading your documentation to gain a quick and general overview of the topic.
- In-depth tutorials, best practice guides, and command-line reference pages can work well as lengthier documents, especially when aimed at readers who already have some experience with the tools and subject matter.
- A great tutorial can rely on a narrative to lead the reader through a series of related tasks in a longer document. However, even large tutorials can sometimes benefit from being broken up into smaller parts.
- Many longer documents aren't designed to be read in one sitting. For example, users typically scan through a reference page to search for an explanation of a command or flag.

The remainder of this unit covers techniques that can be useful for writing longer documents, such as tutorials and some conceptual guides.

Organize a document

This section suggests some techniques for planning a longer document, including creating an outline and drafting an introduction. After you've completed the first draft of a document, you can review it against your outline and introduction to make sure you haven't missed anything you originally intended to cover.

Outline a document

Starting with a structured, high-level outline can help you group topics and determine where more detail is needed. The outline helps you move topics around before you get down to writing.

You might find it useful to think of an outline as the narrative for your document. There is no standard approach to writing an outline, but the following guidelines provide practical tips you might find useful:

- Before you ask your reader to perform a task, explain to them why they are doing it. For example, the following bullet points illustrate a section of an outline from a tutorial about auditing and improving the accessibility of web pages:
  - Introduce a browser plugin that audits the accessibility of web pages; explain that the reader will use the results of the audit report to fix several bugs.
  - List the steps to run the plugin and audit the accessibility of a web page.
- Limit each step of your outline to describing a concept or completing a specific task.
- Structure your outline so that your document introduces information when it's most relevant to your reader. For example, your reader probably doesn't need to know (or want to know) about the history of the project in the introductory sections of your document when they're just getting started with the basics. If you feel the history of the project is useful, then include a link to this type of information at the end of your document.
- Consider explaining a concept and then demonstrating how the reader can apply it either in a sample project or in their own work. Documents that alternate between conceptual information and practical steps can be a particularly engaging way to learn.
- Before you start drafting, share the outline with your contributors. Outlines are especially useful if you're working with a team of contributors who are going to review and test your document.

Outline exercise

Review and update the following high-level outline of an introduction to a long tutorial. To solve this exercise, you can do any of the following:

- Rearrange the existing topics.
- Add any missing topics you feel should be in an introduction.
- Remove any topics you feel are irrelevant for an introduction.

```
## The history of the project

Describes the history of the development of the project.

## Prerequisites

Lists concepts the reader should be familiar with prior to starting, as well as
any software or hardware requirements.

## The design of the system

Describes how the system works.

## Audience

Describes who the tutorial is aimed at.

## Setting up the tutorial

Explains how to configure your environment to follow the tutorial.

## Troubleshooting

Explains how to diagnose and solve potential problems that might occur when
working through the tutorial.

## Useful terminology

Lists definitions of terms that the reader needs to know to follow the
tutorial.
```

Click the icon to see a possible answer.

Introduce a document

If readers of your documentation can't find relevance in the subject, they are likely to ignore it. To set the ground rules for your users, we recommend providing an introduction that includes the following information:

- What the document covers.
- What prior knowledge you expect readers to have.
- What the document doesn't cover.

Remember that you want to keep your documentation easy to maintain, so don't try to cover everything in the introduction.

The following paragraph demonstrates the ideas from the preceding list as an overview for a hypothetical document publishing platform called Froobus:

```
This document explains how to publish Markdown files using the Froobus system.
Froobus is a publishing system that runs on a Linux server and converts
Markdown files into HTML pages. This document is intended for people who are
familiar with Markdown syntax. To learn about the syntax, see the Markdown
reference. You also need to be comfortable running simple commands in a
Linux terminal. This document doesn't include information about installing or
configuring a Froobus publishing system. For information on installing Froobus,
see Getting started.
```

After you've completed the first draft, check your entire document against the expectations you set in your overview. Does your introduction provide an accurate overview of the topics you cover? You might find it useful to think of this review as a form of documentation quality assurance (QA).

Introduction exercise

For this exercise, review and revise the following introduction for a best practices guide for a hypothetical programming language called F@. Remove any information you feel is irrelevant in this context and add any information you feel is missing.

```
This guide lists best practices for working with the F@ programming language.
F@ was developed in 2011 as an open source community project. This guide
supplements the F@ style guide. In addition to the best practices in this guide,
make sure you also install the F@ command-line linter and run it on your code.
The programming language is widely adopted in the health industry. If you have
suggestions for additions to the list of best practices, file an issue in the
F@ documentation repository.
```

Click the icon to see a possible answer.

The following is one possible solution:

```
This guide lists best practices for working with the F@ programming language.
Before you review this guide, complete the introductory tutorial for new F@
developers. This guide supplements the F@ style guide. In addition to the best
practices in this guide, make sure you also install the F@ command-line linter
and run it on your code. If you have suggestions for additions to the list of
best practices, file an issue in the F@ documentation repository.
```

Add navigation

Providing navigation and signposting for your readers ensures they can find what they are looking for and the information they need to get unstuck.

Clear navigation includes:

- introduction and summary sections
- a clear, logical development of the subject
- headings and subheadings that help users understand the subject
- a table of contents menu that shows users where they are in the document

- links to related resources or more in-depth information
- links to what to learn next

The tips in the following sections can help you plan the headings in your documentation.

Prefer task-based headings

Choose a heading that describes the task your reader is working on. Avoid headings that rely on unfamiliar terminology or tools. For example, suppose you are documenting the process for creating a new website. To create the site, the reader must initialize the Froobus framework. To initialize the Froobus framework, the reader must run the `carambola` command-line tool. At first glance, it might seem logical to add either of the following headings to the instructions:

- Running the carambola command
- Initializing the Froobus framework

Unless your readers are already very experienced with the terminology and concepts for this topic, a more familiar heading might be preferable, such as *Creating the site*.

Provide text under each heading

Most readers appreciate at least a brief introduction under each heading to provide some context. Avoid placing a level three heading immediately after a level two heading, as in the following example:

```
## Creating the site
### Running the carambola command
```

In this example, a brief introduction can help orient the reader:

```
## Creating the site

To create the site, you run the `carambola` command-line tool. The command
displays a series of prompts to help you configure the site.

### Running the carambola command
```

Heading exercise

Helping readers navigate through your documentation helps them find the information they need to successfully use your tool. Often, a clear and well-organized table of contents or outline acts like a map that helps your users navigate the functionality of your tool.

For this exercise, improve the following outline. You can rearrange, add, and delete topics and create secondary entries too.

```
About this tutorial
Advanced topics
Build the asset navigation tree
Define resource paths
Defining and building projects
Launch the development environment
Defining and building resources
What's next
Define image resources
Audience
See also
Build an image resource
Define an image project
Build an image project
Setting up the tutorial
Select the tutorial asset root
About this guide
```

Click the icon to see a possible answer.

The following is one possible solution:

```
## About this tutorial

### Audience
```

```
### About this guide

### Advanced topics

## Setting up the tutorial

### Select the tutorial asset root

### Launch the development environment

### Build the asset navigation tree

### Define resource paths

## Defining and building resources

### Define image resources

### Build an image resource

## Defining and building projects

### Define an image project

### Build an image project

## Defining and building databases

### Define a database

### Build a database

## Pushing, publishing, and viewing a database

### Push a database

### Publish a database

### View a database

## Configuring display rules for point data

### Define, configure, and build vector data

## See also

### Sample data files

## What's next
```

Disclose information progressively

Learning new concepts, ideas, and techniques can be a rewarding experience for many readers who are comfortable reading through documentation at their own pace. However, being confronted with too many new concepts and instructions too quickly can be overwhelming. Readers are more likely to be receptive to longer documents that progressively disclose new information to them when they need it. The following techniques can help you incorporate progressive disclosure in your documents:

- Where possible, try introducing new terminology and concepts near to the instructions that rely on them.
- Break up large walls of text. To avoid multiple large paragraphs on a single page, aim to introduce tables, diagrams, lists, and headings where appropriate.
- Break up large series of steps. If you have a particularly long list of complicated steps, try to re-arrange them into shorter lists that explain how to complete sub-tasks.
- Start with simple examples and instructions, and add progressively more interesting and complicated techniques. For example, in a tutorial for creating forms, start by explaining how to handle text responses, and then introduce other techniques to handle multiple choice, images, and other response types.

Next unit: Illustrating

Was this helpful?

Send feedback

Based on this session

Lists and tables

Educational resources about technical writing.

Updated Aug 31, 2021

Technical writing resources

Educational resources about technical writing.

Updated Oct 12, 2021

Overview of technical writing courses

Educational resources about technical writing.

Updated Nov 2, 2021

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License, and code samples are licensed under the Apache 2.0 License. For details, see the Google Developers Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2021-02-19 UTC.