Command-line arguments

Here's the list of arguments you can pass to `rustdoc`:

`-h/--help`: help

Using this flag looks like this:

```
$ rustdoc -h
$ rustdoc --help
```

This will show `rustdoc`'s built-in help, which largely consists of a list of possible command-line flags.

Some of `rustdoc`'s flags are unstable; this page only shows stable options, `--help` will show them all.

`-V/--version`: version information

Using this flag looks like this:

```
$ rustdoc -V
$ rustdoc --version
```

This will show `rustdoc`'s version, which will look something like this:

```
rustdoc 1.17.0 (56124baa9 2017-04-24)
```

`-v/--verbose`: more verbose output

Using this flag looks like this:

```
$ rustdoc -v src/lib.rs
$ rustdoc --verbose src/lib.rs
```

This enables "verbose mode", which means that more information will be written to standard out. What is written depends on the other flags you've passed in. For example, with `--version`:

```
$ rustdoc --verbose --version
rustdoc 1.17.0 (56124baa9 2017-04-24)
binary: rustdoc
commit-hash: hash
commit-date: date
host: host-triple
release: 1.17.0
LLVM version: 3.9
```

`-o/--out-dir`: output directory path

Using this flag looks like this:

```
$ rustdoc src/lib.rs -o target/doc
$ rustdoc src/lib.rs --out-dir target/doc
```

By default, `rustdoc`'s output appears in a directory named `doc` in the current working directory. With this flag, it will place all output into the directory you specify.

`--crate-name`: controlling the name of the crate

Using this flag looks like this:

```
$ rustdoc src/lib.rs --crate-name mycrate
```

By default, `rustdoc` assumes that the name of your crate is the same name as the `.rs` file. `--crate-name` lets you override this assumption with whatever name you choose.

`--document-private-items`: Show items that are not public

Using this flag looks like this:

```
$ rustdoc src/lib.rs --document-private-items
```

By default, `rustdoc` only documents items that are publicly reachable.

```
pub fn public() {} // this item is public and will be documented
mod private { // this item is private and will not be documented
    pub fn unreachable() {} // this item is public, but unreachable, so it will not be docu
}
```

`--document-private-items` documents all items, even if they're not public.

`-L`/`--library-path`: where to look for dependencies

Using this flag looks like this:

```
$ rustdoc src/lib.rs -L target/debug/deps
$ rustdoc src/lib.rs --library-path target/debug/deps
```

If your crate has dependencies, `rustdoc` needs to know where to find them. Passing `--library-path` gives `rustdoc` a list of places to look for these dependencies.

This flag takes any number of directories as its argument, and will use all of them when searching.

`--cfg`: passing configuration flags

Using this flag looks like this:

```
$ rustdoc src/lib.rs --cfg feature="foo"
```

This flag accepts the same values as `rustc --cfg`, and uses it to configure compilation. The example above uses `feature`, but any of the `cfg` values are acceptable.

`--extern`: specify a dependency's location

Using this flag looks like this:

```
$ rustdoc src/lib.rs --extern lazy-static=/path/to/lazy-static
```

Similar to `--library-path`, `--extern` is about specifying the location of a dependency. `--library-path` provides directories to search in, `--extern` instead lets you specify exactly which dependency is located where.

`-C`/`--codegen`: pass codegen options to rustc

Using this flag looks like this:

```
$ rustdoc src/lib.rs -C target_feature=+avx
$ rustdoc src/lib.rs --codegen target_feature=+avx

$ rustdoc --test src/lib.rs -C target_feature=+avx
$ rustdoc --test src/lib.rs --codegen target_feature=+avx

$ rustdoc --test README.md -C target_feature=+avx
$ rustdoc --test README.md --codegen target_feature=+avx
```

When rustdoc generates documentation, looks for documentation tests, or executes documentation tests, it needs to compile some rust code, at least part-way. This flag allows you to tell rustdoc to provide some extra codegen options to rustc when it runs these compilations. Most of the time, these options won't affect a regular documentation run, but if something depends on target features to be enabled, or documentation tests need to use some additional options, this flag allows you to affect that.

The arguments to this flag are the same as those for the `-C` flag on rustc. Run `rustc -C help` to get the full list.

`--test`: run code examples as tests

Using this flag looks like this:

```
$ rustdoc src/lib.rs --test
```

This flag will run your code examples as tests. For more, see the chapter on documentation tests.

See also `--test-args`.

`--test-args`: pass options to test runner

Using this flag looks like this:

```
$ rustdoc src/lib.rs --test --test-args ignored
```

This flag will pass options to the test runner when running documentation tests. For more, see the chapter on documentation tests.

See also `--test`.

`--target`: generate documentation for the specified target triple

Using this flag looks like this:

```
$ rustdoc src/lib.rs --target x86_64-pc-windows-gnu
```

Similar to the `--target` flag for `rustc`, this generates documentation for a target triple that's different than your host triple.

All of the usual caveats of cross-compiling code apply.

`--default-theme`: set the default theme

Using this flag looks like this:

```
$ rustdoc src/lib.rs --default-theme=ayu
```

Sets the default theme (for users whose browser has not remembered a previous theme selection from the on-page theme picker).

The supplied value should be the lowercase version of the theme name. The set of available themes can be seen in the theme picker in the generated output.

Note that the set of available themes - and their appearance - is not necessarily stable from one rustdoc version to the next. If the requested theme does not exist, the builtin default (currently `light`) is used instead.

`--markdown-css`: include more CSS files when rendering markdown

Using this flag looks like this:

```
$ rustdoc README.md --markdown-css foo.css
```

When rendering Markdown files, this will create a `<link>` element in the `<head>` section of the generated HTML. For example, with the invocation above,

```
<link rel="stylesheet" type="text/css" href="foo.css">
```

will be added.

When rendering Rust files, this flag is ignored.

`--html-in-header`: include more HTML in

Using this flag looks like this:

```
$ rustdoc src/lib.rs --html-in-header header.html
$ rustdoc README.md --html-in-header header.html
```

This flag takes a list of files, and inserts them into the `<head>` section of the rendered documentation.

`--html-before-content`: include more HTML before the content

Using this flag looks like this:

```
$ rustdoc src/lib.rs --html-before-content extra.html
$ rustdoc README.md --html-before-content extra.html
```

This flag takes a list of files, and inserts them inside the `<body>` tag but before the other content `rustdoc` would normally produce in the rendered documentation.

`--html-after-content`: include more HTML after the content

Using this flag looks like this:

```
$ rustdoc src/lib.rs --html-after-content extra.html
$ rustdoc README.md --html-after-content extra.html
```

This flag takes a list of files, and inserts them before the `</body>` tag but after the other content `rustdoc` would normally produce in the rendered documentation.

`--markdown-playground-url`: control the location of the playground

Using this flag looks like this:

```
$ rustdoc README.md --markdown-playground-url https://play.rust-lang.org/
```

When rendering a Markdown file, this flag gives the base URL of the Rust Playground, to use for generating `Run` buttons.

`--markdown-no-toc`: don't generate a table of contents

Using this flag looks like this:

```
$ rustdoc README.md --markdown-no-toc
```

When generating documentation from a Markdown file, by default, `rustdoc` will generate a table of contents. This flag suppresses that, and no TOC will be generated.

`-e/--extend-css`: extend rustdoc's CSS

Using this flag looks like this:

```
$ rustdoc src/lib.rs -e extra.css
$ rustdoc src/lib.rs --extend-css extra.css
```

With this flag, the contents of the files you pass are included at the bottom of Rustdoc's `theme.css` file.

While this flag is stable, the contents of `theme.css` are not, so be careful! Updates may break your theme extensions.

`--sysroot`: override the system root

Using this flag looks like this:

```
$ rustdoc src/lib.rs --sysroot /path/to/sysroot
```

Similar to `rustc --sysroot`, this lets you change the sysroot `rustdoc` uses when compiling your code.

`--edition`: control the edition of docs and doctests

Using this flag looks like this:

```
$ rustdoc src/lib.rs --edition 2018
$ rustdoc --test src/lib.rs --edition 2018
```

This flag allows `rustdoc` to treat your rust code as the given edition. It will compile doctests with the given edition as well. As with `rustc`, the default edition that `rustdoc` will use is `2015` (the first edition).

`--theme`: add a theme to the documentation output

Using this flag looks like this:

```
$ rustdoc src/lib.rs --theme /path/to/your/custom-theme.css
```

`rustdoc`'s default output includes two themes: `light` (the default) and `dark`. This flag allows you to add custom themes to the output. Giving a CSS file to this flag adds it to your documentation as an additional theme choice. The theme's name is determined by its filename; a theme file named `custom-theme.css` will add a theme named `custom-theme` to the documentation.

`--check-theme`: verify custom themes against the default theme

Using this flag looks like this:

```
$ rustdoc --check-theme /path/to/your/custom-theme.css
```

While `rustdoc`'s HTML output is more-or-less consistent between versions, there is no guarantee that a theme file will have the same effect. The `--theme` flag will still allow you to add the theme to your documentation, but to ensure that your theme works as expected, you can use this flag to verify that it implements the same CSS rules as the official `light` theme.

`--check-theme` is a separate mode in `rustdoc`. When `rustdoc` sees the `--check-theme` flag, it discards all other flags and only performs the CSS rule comparison operation.

`--crate-version`: control the crate version

Using this flag looks like this:

```
$ rustdoc src/lib.rs --crate-version 1.3.37
```

When `rustdoc` receives this flag, it will print an extra "Version (version)" into the sidebar of the crate root's docs. You can use this flag to differentiate between different versions of your library's documentation.

`@path`: load command-line flags from a path

If you specify `@path` on the command-line, then it will open `path` and read command line options from it. These options are one per line; a blank line indicates an empty option. The file can use Unix or Windows style line endings, and must be encoded as UTF-8.

`--passes`: add more rustdoc passes

This flag is **deprecated**. For more details on passes, see the chapter on them.

`--no-defaults`: don't run default passes

This flag is **deprecated**. For more details on passes, see the chapter on them.

`-r`/`--input-format`: input format

This flag is **deprecated** and **has no effect**.

Rustdoc only supports Rust source code and Markdown input formats. If the file ends in `.md` or `.markdown`, `rustdoc` treats it as a Markdown file. Otherwise, it assumes that the input file is Rust.