Documentation

Spec Organization and Prose
---

1. Define the overall model of the technology and shape of the problem
   before defining the API and technical details.

2. Start with an overview / high-level description, push fiddly details
   and more boring material further down in the section, so that a reader
   gets an overall understanding before adding details and so that authors
   can stop reading once they get bored and have read everything they need.

3. Algorithmic vs Constraint based spec approaches: understand which to
   use when -- they each have their strengths and weaknesses, and you
   can use either or both as appropriate.

4. Use diagrams/figures/examples generously to illustrate, but not replace,
   normative prose.

5. Keep examples and figures close to the point they're illustrating.

6. Use the Infra data types and literals when you need to reference data
   types in the abstract. https://infra.spec.whatwg.org/

7. Periodically read your spec from top to bottom, to make sure it makes
   sense when read in order.

8. Also review your table of contents:
   * Ensure the spec and its heading structure is well-organized
   * Have headings (and thus the TOC links) use evocative language
     (not just titled by bits of code) to help readers quickly find
     the right section from the TOC.
   * Be generous with subheadings, to break up long sections and
     facilitate linking and easier ToC usage.

Spec Tooling / Formatting
---

1. Use version control.

2. Use a preprocessor like ReSpec or Bikeshed

3. Use manual IDs so that IDs remain stable as you adjust the heading text;
   add old IDs (via empty elements with IDs, or e.g. Bikeshed's oldids attribute)
   when removing or changing IDs so that links to your spec don't break.

4. Consider semantic line breaks
   (We looked at the source code to CSS Grid as an example*, seehttps://github.com/w3c/csswg-drafts/blob/master/css-grid-1/Overview.bs)