Asciidoc Test Article

Table of Contents

As far as I understood, the first paragraph after the main header is the preamble. I'm not sure what a preamble usually contains. Something like an abstract, I guess. I'll have to look that up and read some examples.

1. Sections, Titles & Paragraphs

This here is a first level section. Below is a second level section.

1.1. Second Level Section

First, I will try out a few things with paragraphs. Seemingly, paragraphs themselves can have titles. The following paragraph for example will have a short title. Let's see how it looks.

Short Title

This is the second paragraph which should have a small title. Let's see how that looks once I generate a PDF or an HTML for it.

Inline Literal

There are inline literals. I create here one mainly to see whether it looks nice in my themes. Here is an inline literal: `asciidoc2pdf simple-article.adoc`. Let's see how that looks if there is text around it. Does the border offset and the border radius look good?

Literal Paragraph

I can create literal paragraphs. That means it will be monospaced and any whitespaces will be kept and not removed. This can be useful for certain things like ascii art. The paragraph below is a literal paragraph.

```
  _   ,_,   _
 / `'=) (='` \
/.-.-.\ /.-.-.\
`      "      `
```

I hope one can see the ascii art above correctly.

2. Admonitions

One can create notes in to variations.

This is a note without a title. I add a bit more text to it to see whether vim properly formats this variation of writing a note. Unfortunately, vim isn't that smart. But it doesn't matter. Asciidoctor-pdf obviously can handle the note even if it isn't formatted perfectly.

Note Title

This is a note with a title.

This is a tip.

This is a warning.

This is important.

Here you need to be cautious.

3. Example Blocks

An example block is useful for visually delineating content that illustrates a concept or showing the result of an operation. An example can contain any type of content and AsciiDoc syntax. Normal substitutions are applied to example content.

If the example content is contiguous, i.e., not interrupted by empty lines, the block style name `example` can be placed directly on top of the text in an attribute list (`[]`).

If the example content includes multiple blocks or content separated by empty lines, place the content between delimiter lines consisting of four equals signs (`====`).

Example 1. Onomatopoeia

The book hit the floor with a **thud**.

He could hear doves *cooing* in the pine trees' branches.

4. Sidebars

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Optional Title

Sidebars are used to visually separate auxiliary bits of content that supplement the main text.

They can contain any type of content.

Source code block in a sidebar

```
const { expect, expectCalledWith, heredoc } = require('../test/test-utils')
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5. Source Code

Below is some source code:

```
ps -ef | grep java
```

I can also create multi-line souce code blocks:

Multi-Line Source Code Block

```
1 function log(message) { ①
2     console.log(getDateTime() + ": " + message); ②
```

```
3 }
```

① Declaration of the log function.

② Logging the current datetime followed by the log message.

There is autofit configuration that can either be set globally or on specific code blocks. It shrinks the code such that long code lines may still fit into the PDF. The code block below has this autofit enabled. Let's see how it looks. And to learn more about it, refer to https://asciidoctor.org/docs/asciidoctor-pdf/#autowidth-tables.

Long lined code block

```
function log(message) {
    console.log(getDateTime() + ": " + message); // And some rather long comment which should lead
to a shrink.
}
```

6. Icons

This is actually a feature documented at https://asciidoctor.org/docs/asciidoctor-pdf/#font-based-icons. Take a look at it.

Here is a fontawesome icon: .

And here is an icon from another icon set: .

You can even set the size of the icons:

7. Keyboard

The keyboard macro uses the short (no target) macro syntax `kbd:[key(+key)*]`. Each key is displayed as entered in the document. Multiple keys are separated by a plus (e.g., `Ctrl+T`) or a comma (e.g., `Ctrl,T`). The plus is preferred. It's customary to represent alpha keys in uppercase, though this is not enforced. If the last key is a backslash (`\`), it must be followed by a space. Without this space, the processor will not recognize the macro. If one of the keys is a closing square bracket (`]`), it must be proceeded by a backslash. Without the backslash escape, the macro will end prematurely. You can find example of these cases in the example below. Here are some examples:

Shortcut	Purpose
F11	Toggle fullscreen
Ctrl + T	Open a new tab
Ctrl + Shift + N	New incognito window
\	Used to escape characters
Ctrl +]	Jump to keyword
Ctrl + +	Increase zoom

8. Buttons

It can be difficult to communicate to the reader that they need to press a button. They can't tell if you are saying "OK" or they are supposed to look for a button labeled OK. It's all about getting the semantics right. The btn macro to the rescue! Here are some examples:

Press the `Ok` button when you are finished.
Select a file in the file navigator and click `Open`.

9. Menus

Trying to explain how to select a menu item can be a pain. With the menu macro, the symbols do the work. Here are some examples:

To save the file, select **File › Save**.
Select **View › Zoom › Reset** to reset the zoom level to the default setting.

10. Definitions

The typical definitions look as follows:

GPS
: Denotes that the position is a GPS position.

CELLULAR
: Denotes that the position is a cellular position.

However, you can also create horizontal definitions.

GPS	Denotes that the position is a GPS position.
CELLULAR	Denotes that the position is a cellular position. I'm adding here more text to it to see how the term gets verically aligned. Nice, the term is top aligned.

You can also create them so they have markers. These markers are either unordered:

- **GPS:** Denotes that the position is a GPS position.
- **CELLULAR:** Denotes that the position is a cellular position. I'm adding here more text to it to see how the term gets verically aligned. Nice, the term is top aligned.

Or ordered:

1. **GPS:** Denotes that the position is a GPS position.
2. **CELLULAR:** Denotes that the position is a cellular position. I'm adding here more text to it to see how the term gets verically aligned. Nice, the term is top aligned.

By default, the subject (i.e., the term) is followed immediately by a colon (still in bold) and offset from the description by a space. You can replace the colon with another character (or sequence of characters) using the block attribute named `subject-stop`.

- **GPS >** Denotes that the position is a GPS position.
- **CELLULAR >** Denotes that the position is a cellular position. I'm adding here more text to it to see how the term gets verically aligned. Nice, the term is top aligned.

A description list with marker uses a run-in layout by default. In other words, the subject appears on the same line as the description, separated by the subject stop and a space. To make the subject appear above the description, like in a normal description list, add the stack role to the list. In this case, the stop character is only added if specified explicitly.

- **GPS**
  Denotes that the position is a GPS position.
- **CELLULAR**
  Denotes that the position is a cellular position. I'm adding here more text to it to see how the term gets verically aligned. Nice, the term is top aligned.

11. Q&A

1. *What is the answer to the universe, god and everything?*

   The answer is 42.

2. *What time is today?*

   I don't know. Maybe 22:00?

3. *What day is it today?*

   It is Sunday 2020-12-13.

12. Tables

There are awesome table features. You can actually just paste CSV into your document ant it will create a table for it. So you don't have to wiggle and bring your data into the markdown specific format. Just awesome.

ID	FName	LName	Address	Phone
1	Vasya	**Pupkin**	London	+123
2	X	**Y**	A,B	45678

13. Lists

- ☐ Todo 1
- ☑ Todo 2 (done)

14. Anchors & Cross References

This is a cross-reference to an anchor defined earlier in the document: link.

Alternatively, one can link directly to any header in the document like this: Admonitions.

15. Blockquotes

After landing the cloaked Klingon bird of prey in Golden Gate park:

> *Everybody remember where we parked.*
>
> *— Captain James T. Kirk, Star Trek IV: The Voyage Home*

16. Verse

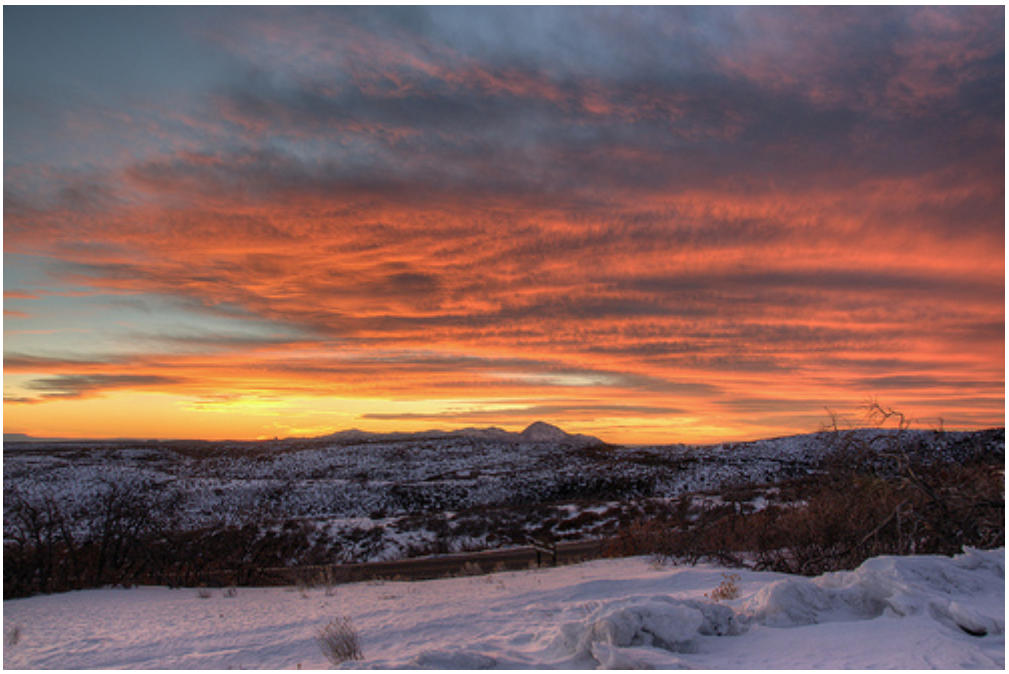
When you need to preserve indents and line breaks, use a verse block. Verses are defined by setting `verse` on a paragraph or an excerpt block delimited by four underscores.

> *The fog comes*
> *on little cat feet.*
>
> *It sits looking*
> *over harbor and city*
> *on silent haunches*
> *and then moves on.*
>
> *— Carl Sandburg, Fog*

17. Images

There are two AsciiDoc image macro types, block and inline. As with all macros, the block and inline forms differ by the number of colons that follow the macro name. The block form uses two colons (`::`), whereas the inline form only uses one (`:`).

Here is an example block image:



And here is an example of the same block image but with an ID, a caption, a link, and size attributes.

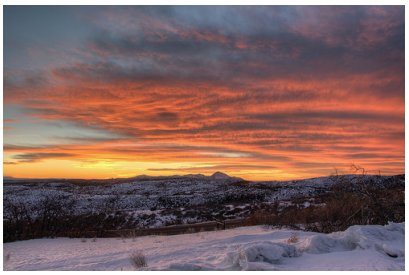


Figure 1. A mountain sunset

And here are two examples of inline images. The second one has a title that will be displaye when hovering over the image (at least in HTML).

Click  to get the party started.
Click  when you need a break.

18. Horizontal Rule

Here is some text that is followed by a horizontal rule.

And then the text continues.

19. Paragraph Summary Layout

The section has a layout that supports summaries right next to the paragraphs.

This section here makes an attempt at writing a text summaries right next to the paragraphs. The summaries are displayed in their own column. They are slightly less emphasized than they main text. The idea is that the main text still has the reader's focus. The summaries are merely there for the reader to recap what the paragraphs are about. Apart from being helpful for a recap, the summaries also help in finding a particular paragraph.

I anticipate a problem with page breaks.

I created this layout using a table. If a page breaks happen in the middle of a cell, then I guess the entire cell will be moved over to the next pages as cells usually do not break. Let's make this paragraph rather long and then add a few more paragraphs to it to see how it will behave.

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Morbi sagittis orci nisl, id suscipit enim accumsan ut. Nullam vitae lobortis velit. Fusce lobortis pharetra elit, posuere feugiat eros mattis quis. Aliquam tincidunt arcu eu mauris accumsan luctus. Morbi tincidunt eget mauris sit amet mattis. Vivamus eros augue, rutrum sit amet pulvinar ac, pretium nec ante.

Where does it come from?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Conclusion: on page break, paragraphs in this layout move to the next page.

Unfortunately, in this layout, entire paragraphs move over to the next page if a page break happens to be within the paragraph. That is because I created this layout using a table. So, each paragraph in this layout is a table cell. Cells obviously do not break in the sense that half of the cell stays on the current page and the rest of the cell moves over onto the next page. Instead, the entire cell moves over to the next page in case it doesn't have enough space on the current page.

Breakable doesn't help.

I tried solving this problem with the `%breakable` option although breakable is anyway the default. Adding breakable didn't help. It does not have the desired effect.

Keep paragraphs small.

The only workaround I can offer is to keep the paragraphs relatively small, so it doesn't hurt that much when an entire paragraph gets moved over to the next page.