# Heading 1

This paragraph belongs to heading 1.

## Heading 2

This paragraph belongs to heading 2.

### Heading 3

#### Heading 4

##### Heading 5

###### *Heading 6*

Some **bold** and underlined text with code in heading.

This paragraph has **bold** or *italic* or underlined text.

This paragraph has ***bold-italic***, *underlined-italic* and **underlined-bold** text.

This paragraph has code and [links](https://www.adobe.com/).

A bold **paragraph with**   
**manual** breaks.

What happens if **bold spawns**

**several paragraphs?**

This has even more formatting like ~~strikethrough~~ and somesupscript andsuperscript.

See different monospace fonts: courier or **Roboto Mono Bold** or maybe Source Code Pro.

Let’s see if spaces after **bold** are correctly handled. Also **bold** *italic* switcherroos. And spaces  **before bold** text?

What about **bold*italicbold****italic* switches?

Here’s some example code:

async function getFile(drive, parentId, name, isFolder) {

const query = [

`'${parentId}' in parents`,

`and name = ${JSON.stringify(name)}`,

'and trashed=false',

`and mimeType ${isFolder ? '=' : '!='} }

And some code with soft breaks

// first, turn styles off that are not in text style  
 for (let i = currentStyles.length - 1; i >= 0; i -= 1) {  
 const s = currentStyles[i];  
 if (!ts[s.name]) {  
 md += s.off;  
 currentStyles.splice(i, 1);  
 }  
 }

### And a one liner:

$ npm install

Should work. And then code at the end  
Before a soft break and at the end

Of a paragraph.

Here code with soft breaks and **formatting:**  
$ npm install  
  
$ cd …  
**And** it continues….

Here code with soft breaks only two lines  
$ npm install  
$ cd …  
And it continues….

And code that changes code font.

Page Break:

Section Break (next page):

Section Break (continous):

Horizontal Line:

Empty Heading:

Empty Bold Heading:

Empty heading with image:

## 

Empty heading with hr

## 

Empty heading with section break

Empty Heading with page break

## Heading with Soft Breaks

## Then numbered lists:

1. First install the dependencies:  
   $ npm install
2. Bar **bold**
3. Trello
   1. Apple
   2. Oranges
   3. Lemons
      1. Roman
      2. Greek  
         Muiltiline  
         With code.
      3. Spanish

## And unordered lists:

* Todo
* And so more
* Should do the trick
  + Nested 1
  + Nested 2
* Mixed with ordered

1. One
2. Two
3. Three

### Special List

1. Codeline
2. More code
3. Some more code

## All List Types

1. One
2. Two
3. Three
4. Four
5. Five
6. Six

* Seven
* Eight
* Nine
* Ten
* Eleven
* Twelfe

### Demos

You can watch the entire recording here

1. Server timing & DOM based HTL engine
2. Helix Pages & auto-generated sequence diagrams
3. Performance analysis of Helix OpenWhisk actions
4. Authoring user journey
5. Dev experience: Helix 6 months ago and today

## Let’s try a simple table

|  |  |  |  |
| --- | --- | --- | --- |
| a0 | b0 | c0 | d0 |
| a1 | b1 | c1 | d1 |
| a2 | b2 | c2 | d2 |

## And a more complex table

|  |  |  |  |
| --- | --- | --- | --- |
| **Country** | **Abbrev** | **Amount** | **Example** |
| Switzerland | CH | 5 | const a=1; let b=5; |
| USA | US | 2.5 | *n/a -* or |
| Japan | JP | 3.14 | Math.PI; |

## Table with lists

Totally useless...

|  |  |
| --- | --- |
| List | Comment |
| 1. Apple 2. Banana 3. Orange | fruits |
| * Car * Airplane * Ship | transportation |

# Inline Images

Here is a simple happy face!



# This is heading after image

# This is heading with image.

1. 
2. 

## Final Code

/\*\*  
 \* Main function  
 \* @param params Action params

\* @returns {Promise<\*>} The response

\*/

async function run(params) {

const disclosed = { ...params };

Object.keys(disclosed).forEach((key) => {

if (key.match(/^[A-Z0-9\_]+$/)) {

delete disclosed[key];

}

});

log.trace('%s', JSON.stringify(disclosed));

return fetchViaDoclet(params);

}