**WordWebNav Users' Guide**

v1.0, 7/2021

This users' guide describes how to use WordWebNav (WWN) to create a web-page from a Word-doc.

The quick-start section describes the minimal steps in using WWN to create a web-page. Subsequent sections describe the other WWN web-page features and tools. The Appendix provides info about troubleshooting, and about Word-HTML's limitations and bugs.

# Pre-requisite technical skills

In using WWN, files must be specified as full-paths. Full-paths can be obtained via File Explorer, and its "Copy path" feature (shift+right-click).

Using WWN requires basic skills in system-administration, such as working with directories and file-paths, creating configuration-files, installing software, and running Python programs.

Installing WWN web-pages on a web-server requires basic skills in web-site creation, such as creating directories, uploading web-pages, and creating hyperlinks to internal web-pages.

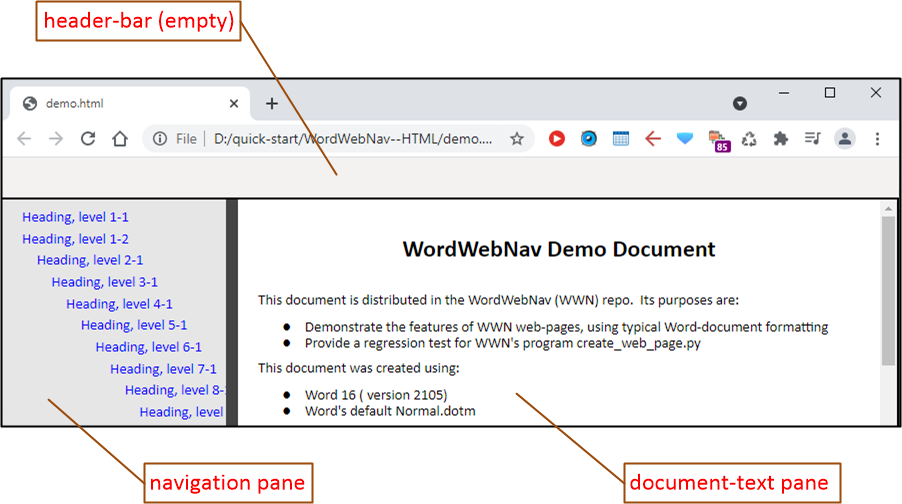
Tutorials for such tasks can be found on the Internet.

# Quick-start: creating a WWN web-page

First, install WWN as described in the installation doc. It includes downloading the WWN repo.

The present instructions describe how to create a WWN web-page, from a Word-doc. The web-page to be created is shown below. It includes a navigation pane with hyperlinks to the document's headings. The web-page is configured to be opened from the file-system, i.e., by clicking on it.

The Word-doc used is in the WWN repo at: tests\tests-for-create\_web\_page\_py\demo.docx



The first two steps are: 1) adding a TOC to the Word-doc, and 2) saving the Word-doc as a Word HTML-file. The manual process for these steps is described here. WWN provides a program that automates these two steps, and it is described in section 5 "Tools".

Make a directory named "quick-start". It will be used in creating the WWN web-page. Copy the Word-doc demo.docx from the repo, to the quick-start directory. Also, in the quick-start directory, create two directories for storing the web-pages that will be created:

* WordWebNav--HTML: used to store the WWN web-page
* WordWebNav--Word-HTML: used to store Word HTML-file

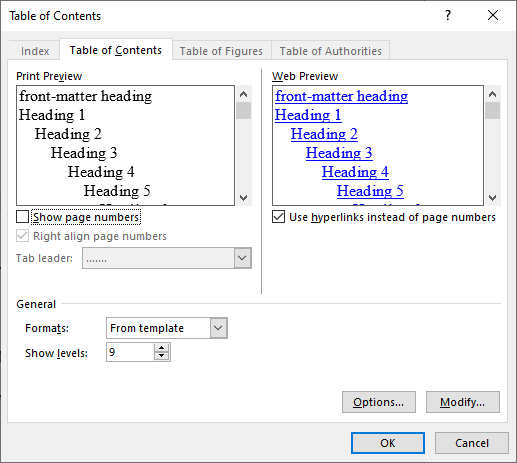
## Insert table-of-contents

In this step, the Word-doc is opened, a table-of-contents (TOC) is inserted at the top, and the doc is saved as a Word HTML-file. The TOC is just used in creating the Word HTML-file, and it's not saved to the Word-doc.

Word creates the TOC from the document's headings. If a Word-doc does not have headings, this step can be skipped. demo.docx does have headings.

When the WWN web-page is constructed, the TOC at the top of the Word HTML-file will be put in the WWN web-page's navigation pane. If a Word-doc already has a TOC at the top, another TOC can still be added to the top, but the two TOCs should be separated by an empty paragraph. The second TOC will be in the WWN web-page's document-text pane.

* Open the Word-doc, and go to the top of the doc (ctrl+home)
* Create the TOC:
  + Open Word's TOC dialog window, by clicking on:
    - Review : Table of Contents : Custom Table of Contents
  + In the dialog window (shown below):
    - Click to turn-off "Show page numbers"
    - Click to turn-on "User hyperlinks instead of page numbers"
    - In the "Formats:" drop-down box, select the option "From template"
    - In the "Show levels" drop-down box, enter "9", to show all levels.
  + Click on "OK"
  + For the pop-up window asking, "Replace this table of contents?", click on "No".



A screen-shot of the Word-doc, with the added TOC, is in the [Appendix](#_Quick-start:_WWN_web-page).

The TOC added to the top of the Word-doc is referred to as the *navigation-pane TOC*. If a Word-doc has existing TOCs, before adding the navigation-pane TOC, the existing TOCs must use Word's default TOC-style. If the existing TOCs use any other TOC-style, then the WWN web-page's navigation-pane could be misformatted. In short, this will not be a problem if the navigation-pane TOC is the only TOC, or if the existing TOCs were created using the "Formats:" option "From template". In the present example, demo.docx has an existing TOC, and it uses Word's default TOC-style. WWN's TOC requirements are further described in the Appendix (section 6.1.3 ).

## Create the Word HTML-file

The next step is to save the Word-doc as a Word HTML-file.

Click on: File : Save As

* Click Browse, and select the directory WordWebNav--Word-HTML
* Set "Save as type" to "Web Page, Filtered (\*.htm, \*.html)"
  + Be sure to select the "Filtered" web-page type, shown here:

Graphical user interface, text, application, email

Description automatically generated

* Set the file-name extension to ".htm" or ".html"
* Click "Save"
* If prompted about "Office-specific tags", click "Yes" to save.
* Close the Word-HTML doc

A Word HTML-file will have been created. If the Word-doc had embedded images, they will be converted to image files, and saved in a directory. The directory name is the same name as the Word HTML-file, but with the suffix "\_files". For this example, the file and directory created are:

* demo.htm
* demo\_files

A screen-shot of the Word HTML-file is in the [Appendix](#_Create_the_Word).

Some Word features do not get rendered well in HTML format. So, it's prudent to open the Word HTML-file in a browser, to check for problems. WWN can fix some Word-HTML bugs, in the next step. Also, some rendering problems can be fixed by using alternative Word features. More info on Word-HTML's limitations and bugs is in the Appendix (sections 6.1.4 and 6.2 ).

## Create the WWN web-page

The next step is to run the WWN program create\_web\_page. It converts a Word HTML-file to a WWN web-page.

### Create the parameter-file needed by create\_web\_page

To run create\_web\_page, a parameter-file is needed. It specifies the input Word HTML-file, the directory for the output WWN web-page, etc. The present example shows the minimal set of parameters needed. The WWN web-page that is created can be opened from the local hard-drive (not a web-server).

* Use the example parameter-file that is in the WWN repo, at:
  + templates\web\_page\_create--parameters--minimum.yml
* Copy that parameter-file to the directory with the Word-doc, i.e., quick-start
* Rename the copied file to the same name as the Word-doc, but with the extension ".yml":
  + demo.yml

The parameter-file is in YAML format, and proper YAML syntax must be used. YAML uses key/value pairs, and the keys must be properly indented, e.g., by 2 spaces. If create\_web\_page encounters YAML syntax errors, console messages are displayed. The parameter-file has links to tutorials on YAML syntax.

In the parameter-file, edit the values for the keys. (A "full-path" starts with the drive-letter, e.g., "C:\")

* input\_html\_path: specifies the Word HTML-file
  + Set the value to the full-path for demo.htm
* output\_directory\_path: specifies where the WWN web-page will be written.
  + Set the value to the full-path for WordWebNav--HTML.
* scripts\_directory\_url: specifies the directory with WWN's CSS and JavaScript files
  + Set the value to the full-path for this directory in the WWN repo: /assets
  + (In using that directory, the WWN web-page can be opened from the local hard-drive.)

### Run create\_web\_page

The program create\_web\_page can be run by clicking on it in File Explorer, or by calling it from the Windows command prompt.

The program is in the WWN repo at: createwebpage\create\_web\_page.py

create\_web\_page.py can be run by clicking on it in File Explorer. This works if Windows associates the file-extension (.py) with Python. Alternatively, it may be possible to run create\_web\_page.py by right-clicking on the file, and selecting: "Open with" : "Python". When create\_web\_page.py is run from File Explorer, the program will prompt for the full-path of the parameter-file.

To run create\_web\_page from the Windows command prompt:

> cd <directory with create\_web\_page.py>

> python create\_web\_page.py <full-path of parameter-file>

For demo.docx, when create\_web\_page runs successfully, the last console message will be:

"INFO. Processing completed. No errors. Warning messages: 1"

This warning message can be ignored: "WARNING. Span tag(s) found...".

create\_web\_page will create the WWN web-page, and put it in the output directory. If the Word HTML-file has a directory for images, it will be copied to the output directory. For the example, WordWebNav--HTML will contain:

* demo.htm : the WWN web-page
* demo\_files : the WWN web-page's directory for images

When create\_web\_page loads the Word HTML-file, the file is decoded and converted to Unicode. If the file's encoding-type cannot be determined, the file-load will fail. The decoding process is described in the Appendix (section 6.1.2 ). The output WWN web-page is encoded in UTF-8 format.

To view the WWN web-page, just click on it, and it should open in the browser. Again, it's prudent to check for rendering problems.

## Directory and file organization

The present example includes a set of directories and files. That directory and file organization are recommended, in general, for using WWN:

* DIRECTORY: contains one or more Word-docs processed by WWN, e.g., the directory quick-start
  + FILES: <word-doc-name>.doc\* : Word-docs, e.g., demo.docx
  + FILES: <word-doc-name>.yml : parameter-files for create\_web\_page, e.g., demo.yml
  + DIRECTORY: WordWebNav--Word-HTML: used to store Word HTML-files
    - <word-doc-name>.html : Word HTML-files, e.g., demo.html
    - <word-doc-name>\_files : directories with pictures for the associated Word HTML-file, e.g., demo\_files
  + DIRECTORY: WordWebNav--HTML: used to store WWN web-pages
    - <word-doc-name>.html : WWN web-pages, e.g., demo.html
    - <word-doc-name>\_files : directories with pictures for the associated WWN web-page, e.g., demo\_files

The directory WordWebNav--HTML is used when the WWN web-page is opened from the local file-system. When opening a WWN web-page from a web-server, the WWN web-page is typically stored in a different directory.

# Creating a WWN web-page, for a web-server

The quick-start instructions showed how to create a WWN web-page that can be opened from the local file-system, e.g., by clicking on the ".htm" file. Creating a WWN web-page that can be opened from a web-server is the same, except for two key/value pairs in the create\_web\_page parameter-file:

* output\_directory\_path: specifies where the WWN web-page will be written.
  + Typically, the local file-system would have a mirror of the web-server's directories and files
  + Set the key's value to the full-path of the appropriate directory in the web-server mirror.
* scripts\_directory\_url: specifies the directory with WWN's CSS and JavaScript files
  + Set the value to the URL of the web-server directory that contains these files.
  + A relative path is suggested, e.g., /assets/WordWebNav

Also, WWN's CSS and JavaScript files will need to be stored on the web-server and its mirror on the local file-system, e.g., in /assets/WordWebNav.

It's prudent to test a WWN web-page before copying it to a production web-server. This can be done by setting-up a test web-server on the local workstation (e.g., just enable IIS). The test web-server would use the production web-server's mirror, on the local file-system.

The WWN web-page will need to be copied from the mirror to the production web-server. If the WWN web-page has a "\_files" directory, it will need to be copied to the same directory as the WWN web-page.

# Additional web-page features

There are four additional WWN web-page features that can be used, and they are described in the following sections:

* 4.1 Header-bar
* 4.2 Readers' comments
* 4.3 Adding HTML to the <head> section
* 4.4 Fixes for common Word-HTML bugs

In those sections, the examples given are at this web-page:

<https://jimyuill.com/software/www/WordWebNav/demo.html>.

It is referred to as "*the example web-page*".

The example web-page was created from the Word-doc that was used earlier, in the "quick-start" example. That Word-doc is in the WWN repo at: tests\tests-for-create\_web\_page\_py\demo.docx.

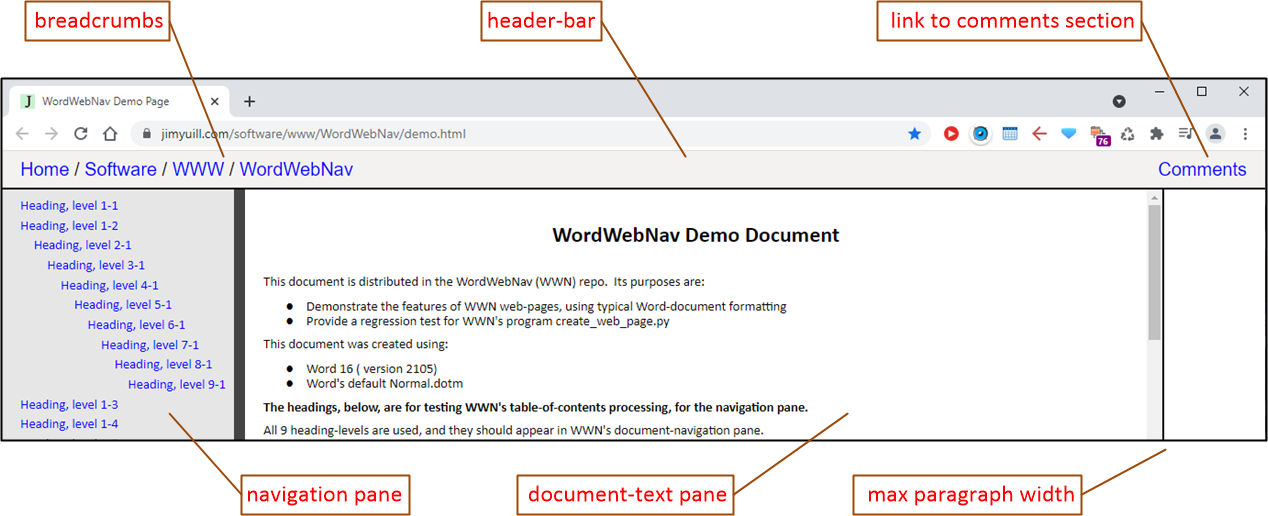
Also, for the example web-page, its create\_web\_page parameter-file is in the WWN repo at:

templates\web\_page\_create--parameters--all.yml

That file is referred to as "*the example parameter-file*", or simply, "*the parameter file*".

## Header-bar

The WWN web-page has a header-bar at the top. The figure below shows the example web-page's header-bar. It contains breadcrumbs for web-site navigation, on the left. On the right is a link to a "comments" section, which is at the bottom of the document-text pane, and readers can submit comments there.



The next two sections describe how to configure the header-bar.

### Header-bar structure

For a WWN web-page, the header-bar is intended primarily for navigation links, as in the example web-page. Also, the header-bar is intended to have a single line of text, or be empty.

The header-bar contents are specified in the create\_web\_page parameter-file, under the key web\_page\_header\_bar.

The header-bar layout is divided into ***sections***. Each header-bar section is specified in the parameter-file, using these keys:  
 - section:

contents:

contents\_alignment:

In the example web-page, the header-bar has two sections, which can also be seen in the example parameter-file.

Each section has the same width in the header-bar. The example web-page has two sections, so each takes-up half of the header-bar's total width. If there were three sections, each would take-up one-third of the header-bar's total width.   
  
In the WWN web-page's HTML, the header-bar is formatted as an HTML table (<table>), with no border. The table has one table-row (<tr>). Each section's contents are placed in a table-cell (<td>), in the row. The example web-page's header-bar has two sections, so it uses two table-cells.

### Header-bar contents and alignment

For each header-bar section, five types of contents are supported: breadcrumbs, hyperlink, html, text, and empty. In the create\_web\_page parameter-file, for each header-bar section, the contents-type is specified as a key under "contents:". This example shows the breadcrumbs key:

- section:

contents:

breadcrumbs:

contents\_alignment:

For the breadcrumbs key, additional keys and values are needed to specify the breadcrumb links. This is shown in the example parameter-file. Use of the hyperlink key is also shown in the example parameter-file.

The key contents-alignment is used to specify alignment of a section's text in the table-cell, e.g., left, right, or center. The contents-alignment key is optional, and the default is left.

For the keys html and text, the value is specified after the key. The value is put in the section's table-cell. This example displays "Hello World" in the center of the header-bar (assuming it's the only section).

- section:

contents:

text: Hello World

contents\_alignment: center

For the key empty, an empty table-cell is created. It can be used to center text in the header-bar. For instance, in the example web-page, to display the Comments link in the center of the header-bar: add a third section with contents-type empty, and for the second section, with the Comments link, set its "contents\_alignment:" to "center".

## Readers' comments

In the WWN web-page, a web-interface for readers' comments can be added after the document-text. The example web-page uses the Commento service for reader-comments. Other such services could be used instead.

In the example parameter-file, the key "document\_text\_trailer:" is used to specify the HTML for implementing the reader-comments.

In addition, the header-bar can have a link to reader-comments, as shown in the example web-page and parameter-file . In the example, the link's text is "Comments". The link address for reader-comments should always be "#word\_web\_nav\_document\_text\_trailer".

## Adding HTML to the <head> section

HTML can be added to the <head> section in the WWN web-page. The example parameter-file specifies the related keys, and it shows how to use them: title, description, and additional\_html.

The web-page's title and description are displayed by search engines, in the search results. The title is also displayed in the browser tab for the web-page.

## Fixes for common Word-HTML bugs

The WWN overview describes some of the Word-HTML bugs that are fixed by WWN, e.g., misformatted lists. The bugs, and their fixes, are further described in the Appendix (section 6.2 ).  
  
Those bugs are fixed by create\_web\_page. For an input Word HTML-file, all of the fixes are made. The only exception is fixes for the Word-HTML bug which incorrectly makes text white that should be black. That white text is invisible on a white background.

Word-HTML can incorrectly use the style-attribute "color:white" to make text white, when the text should be black. That style-attribute is specified in an HTML span-tag. WWN's bug-fix is to remove the style attribute "color:white" from span-tags. To apply the bug-fix, the create\_web\_page parameter-file must include these keys:

word\_html\_edits:

white\_colored\_text:

For the key white\_colored\_text, permissible values are: doNotRemove (the default), removeInParagraphs, and removeAll. If the Word-doc does not have white-text intentionally, the value removeAll can be used. For Word-docs that specify white text, but not in paragraphs, the value removeInParagraphs can be used.

If create\_web\_page finds Word-HTML with the style-attribute "color:white", this message is displayed:

WARNING. Span tag(s) found, with attribute "style" and value "color:white".

Prior INFO messages provide more details, including the number of instances of those span-tags. If the Word-doc is supposed to have white text, the INFO messages' counts can be used to ensure that such span-tags are only used where needed.

# Tools

## Automated creation of Word HTML, for a Word-doc

A program is provided to automate the initial steps in creating a WWN web-page: insert a table-of-contents at the top of the Word-doc, and save the doc as a Word HTML-file. The user can specify one or more Word-docs to be processed.

The program is in the WWN repo, in the Word-doc at: tools\generate\_word\_html.docm

The program is a Word macro (VBA program) within the Word-doc.

Also, the Word-doc describes how to use the program.

# Appendices

## Appendix A: Troubleshooting

### Error and warning messages from create\_web\_page.py

This section contains additional info for selected error and warning messages from create\_web\_page.

* ERROR. Could not open the config-file for yamllint.
  + The yamllint config-file is distributed with the WWN repo, and it should be in the same directory as create\_web\_page.py.
* ERROR. yamllint found syntax errors in the parameter-file.
  + yamllint is used to check for syntax errors relative to the YAML standard. yamllint doesn't check for syntax errors relative to create\_web\_page's parameter requirements, e.g., misspelled keys.
* ERROR. The YAML-loader was not able to load the parameter-file.
  + The problem is probably not due to YAML syntax errors, as yamllint is called before loading the parameter file.
* ERROR. An exception was raised in Cerberus.
  + Cerberus is used to verify the parameter-file's syntax relative to create\_web\_page's parameter requirements. This includes misspelled keys, missing keys, and missing values.
  + Before calling Cerberus, yamllint was called to check for YAML syntax errors, so the reported error is probably not a YAML syntax error.
  + Cerberus's error messages can be difficult to read.
  + Example error messages:

{'required': [{'scripts\_directory\_url': ['required field']}]}

* + - Meaning: the key scripts\_directory\_url is missing. It should be specified under the key required.

{'required': [{'scripts\_directory\_url': ['null value not allowed']}]}

* + - Meaning: the key scripts\_directory\_url is specified, but the value is missing. That key is specified under the key required.

{'word\_html\_edits': [{'white\_colored\_text': ['unallowed value removeAl']}]}

* + - Meaning: the key white\_colored\_text is specified, with the value removeAl, but that value is not allowed. (removeAll is allowed.) That key is specified under the key word\_html\_edits
* ERROR. Could not open the expected jinja template-file.
* ERROR. The jinja template-file does not have the expected signature:
  + The jinja template-file is distributed with the WWN repo, and it should be in the same directory as create\_web\_page.py.
* ERROR. Could not load the input Word HTML-file.
  + The file is loaded by BeautifulSoup, and its error message is also displayed. The problem is likely due to the file's encoding not being recognized. More info on file encodings is in the Appendix (section 6.1.2 ).
* ERROR. The input Word-HTML does not have exactly one <head> element.
* ERROR. The input Word-HTML does not have exactly one <body> element.
* ERROR. The input Word-HTML does not have the expected MS-Word signature:
* ERROR. The expected HTML <head> tag was not found.
* ERROR. The expected HTML </head> tag was not found.
* ERROR. The expected HTML </body> tag was not found.
  + A properly-created Word HTML-file should have those HTML elements.
* ERROR. Unexpected error, while fixing unordered-list list-items.
* ERROR. Editing HTML span tags within a paragraph, having span attribute "style" and value "color:white".
* ERROR. Decoding the ... Number decoded () is not equal to the number encoded ()
  + If these error messages are displayed, it's likely due to a bug in create\_web\_page.

* WARNING. Span tag(s) found, with attribute "style" and value "color:white".
  + There is a Word-HTML bug that incorrectly makes some text white. Other sections describe the bug, WWN's bug-fix, and this warning message (sections 4.4 , 6.2.4 ).

### Supported Word HTML-file encodings

The program create\_web\_page loads a Word HTML-file. Many file encodings are recognized. If the encoding is not recognized, or if it's misidentified, then the file cannot be processed.

create\_web\_page loads the file using the program BeautifulSoup. BeautifulSoup uses a sub-library called "*Unicode, Dammit*" (UD), to detect the file's encoding and convert it to Unicode. UD guesses the file's encoding type. It usually guesses correctly, but sometimes it makes mistakes.

If BeautifulSoup cannot load the file, an error message is displayed. For decoding errors, the error message will often contain, "... codec can't decode byte...".

### The navigation-pane: requirements and formatting-problems

This section describes WWN's requirements for the Word-doc's tables-of-contents. The Word "style" used for tables-of-contents must be Word's default style. For a WWN web-page, if its navigation-pane is not formatted properly, it is probably due to the Word-doc using a different table-of-contents (TOC) style.

The process for creating a WWN web-page starts with inserting a TOC at the top of the Word-doc (described in section 2.1 ). This TOC gets used to create the WWN web-page's navigation-pane. In the Word HTML-file, WWN moves the TOC at the top of the page to the navigation-pane. (If there is no such TOC, the navigation-pane will be empty.)  
  
When using WWN, if there is a TOC at the top of the Word-doc, the document's TOC-style must be the default Word TOC-style. If a different TOC-style is used, the navigation-pane might not be formatted correctly.

Word provides a default TOC-style, and several other predefined TOC-styles, e.g., Classic. As shown below, when a TOC is created, its style is selected, e.g., From template, Classic, Distinctive, etc.   
Graphical user interface, application

Description automatically generated  
  
A Word-doc has a single TOC-style, and it is used for all TOCs in the document. When a TOC is created, the option "From template" specifies to use the document's current TOC-style. For the first TOC created, "From template" selects the default Word TOC-style.

There are three cases in which a Word-doc does not use the default Word TOC-style:

* The Word-doc has a TOC that was created with a "Formats:" option other than "From template", e.g., the option "Fancy". When that TOC was created, the document's TOC-style was set to that TOC's style. (Deleting that TOC does not change the document's TOC-style.)
* In the Word-doc, if a heading's formatting is manually modified, that formatting will appear in the heading's TOC-entry. For example, with the default Word TOC-style, TOC-entries do not use italics. However, for a particular heading, if its text is manually changed to be italics (e.g., Ctrl+i), then the heading's TOC-entry will be in italics.
* For a Word-doc, the default TOC-style is defined in Normal.dot. Usually, that TOC-style is the default Word TOC-style. However, Normal.dot can be modified to use a different default TOC-style. Though, that is unusual.

If the navigation-pane is misformatted, it will likely require fixing the Word-doc's TOC-style.

For problems due to manually-formatted heading-text, that formatting can be removed. Word has a Styles window that shows the document's styles. That window can be used to find manually-formatted heading-text.  
  
If the Word-doc is not using the default Word TOC-style, the document can be reset to use the default. (This assumes Normal.dot specifies the default Word TOC-style.)

* Delete the Word-doc's existing TOCs.
* Open Word's Styles window, then click to open the "Manage Styles" window.
* Delete the current TOC-styles:
  + Click to sort the styles alphabetically
  + Delete styles "TOC 1" to "TOC 9"
* Copy in the TOC-styles from Normal.dot:
  + Click on "Import/Export"
* The deleted TOCs can be recreated, but they must all use the option "Formats: From template".

### Word-HTML problems and work-arounds

This section describes Word features that do not render well in Word-HTML, and ways to work around them.  
  
WWN has been tested with a number of Word-docs downloaded from the Internet. They were mostly technical reports and papers, and they used typical Word features. Their Word-HTML was usually well-formatted.

When the Word-HTML was not well-formatted, it could be fixed by using alternative Word features. For some of the formatting problems encountered, Word was being used in convoluted and unintended ways. Those problems could be fixed by using Word in simpler and intended ways.

Below are examples of Word-HTML problems encountered, and work-arounds:

* **Problem**: There are Word features for which there is no comparable HTML feature.

An example is a page-number reference inserted by Word, e.g., "Section 4.2 on page 10". HTML does not have page numbers.  
  
An obvious solution is to not use such features, and use formatting and lay-out that works with Word-HTML.

* **Problem**: The Word-HTML has unexpected formatting that does not appear in the Word-doc.

Examples are extra blank lines, or an unwanted shift in the left margin.

One cause can be formatting in the Word-doc that is unneeded or incorrect, but which does not visibly affect the Word-doc. Examples are an unneeded section-break or incorrect text-wrapping for a table or image (see below).

The Word feature "Show paragraph marks" can be helpful for finding hidden formatting symbols, e.g., section breaks. The feature can be turned-on in Word's ribbon, at Home : Paragraph.

* **Problem**: A table, or image, has unwanted text next to it.

Change the table properties, or image layout, to have no text-wrapping, e.g.,

Graphical user interface, application

Description automatically generated

* **Problem**: The Word-HTML includes a line that only has an underscore, i.e., "\_".

The problem can be caused by a paragraph in the Word-doc that is just a space and underline formatting is on. A solution is to delete the space, or turn-off underline formatting.

* **Problem**: Word-doc text-boxes tend to be blurred in the Word-HTML, or they can be missing.

When a text-box is rendered in Word-HTML, it is converted to a picture in PNG format. The picture is often blurred.

One solution for text-box problems is to use a table instead, e.g., a table with one cell. This avoids converting the content to picture format.

* **Problem**: In a Word-doc, a hyperlink specified as just a file-name may be transformed to an invalid link in the Word HTML.

In HTML, a link that's relative to the current page is specified as just a file-name, without directories, e.g., "demo.html". In a Word-doc, a link that's specified as just a file-name may be treated differently by Word, e.g., it may be turned into an absolute link.

An alternative is to not specify a link as just a file-name in a Word-doc. In HTML, links that start with a "/" are relative to the root of the current web-server. Those links may be specified in the Word-doc, e.g., /software/www/WordWebNav/demo.html"

How Word treats absolute and relative hyperlinks appears to vary between Word versions, e.g., how Word transforms relative links into absolute links. Also, the documentation for that feature is often unclear. Additional investigation is needed. Additional info:

* + <https://answers.microsoft.com/en-us/msoffice/forum/all/relative-addressed-hyperlinks-become-full/41e085fd-6d40-47a9-825c-ed468ef87d22>
  + <https://word.tips.net/T001527_Maintaining_Proper_Hyperlinks_in_Word_2000_and_Later.html>
  + <https://sites.google.com/site/hardwaremonkey/blog/relativeandabsolutelinksforwordoffice2013>
  + <https://support.microsoft.com/en-us/topic/how-to-create-absolute-hyperlinks-and-relative-hyperlinks-in-word-documents-d6339b5a-98ba-a483-bac5-95bd33ff5a80>
* **Problem**: A paragraph with text-alignment "Justify" can have incorrect indentation in Word-HTML.

"Justify" evenly distributes text between margins. For each line in the paragraph (except the last), the line's last letter is at the right margin. In the Word HTML, the first line can be indented too far from the left margin, by a few spaces. This has only been observed in paragraphs in a bulleted-list's list-item.

One solution for fixing the misaligned paragraph is to use left-alignment instead of "Justify".

* **Problem**: A math equation is misplaced on the Word-HTML page.

This problem can occur if the equation was created with an older version of Word's equation editor.

One solution is to right click on the equation, then select the option to upgrade the equation.

## Appendix B: Word-HTML bugs fixed by WWN

The program create\_web\_page.py creates a WWN web-page. This includes fixing common bugs in Word's HTML. Those bugs are summarized here, and more info is in the sections below.

* Word-HTML's paragraphs span the browser's width, which makes the paragraphs very difficult to read.
* Word-HTML's multi-level bulleted-lists are misformatted, as well as numbered-lists.
* Word-HTML can incorrectly make text white that should be black. The text is invisible on a white background.

The bugs are shown using a Word-doc provided in the WWN repo, at: tests\tests-for-create\_web\_page\_py\demo.docx

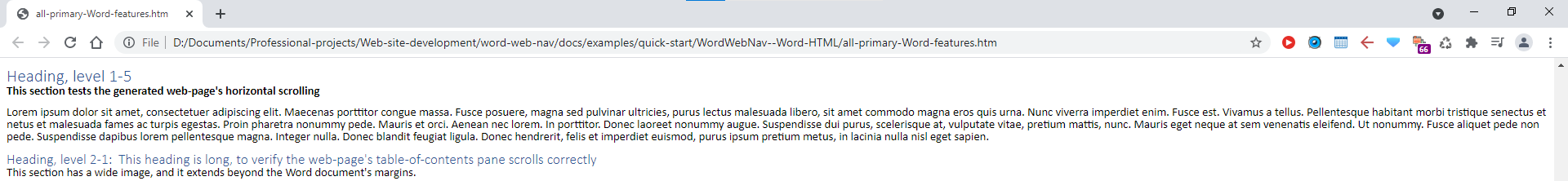
That Word-doc's WWN web-page is [here](https://jimyuill.com/software/www/WordWebNav/demo.html).

### Word-HTML's paragraphs span the browser's width

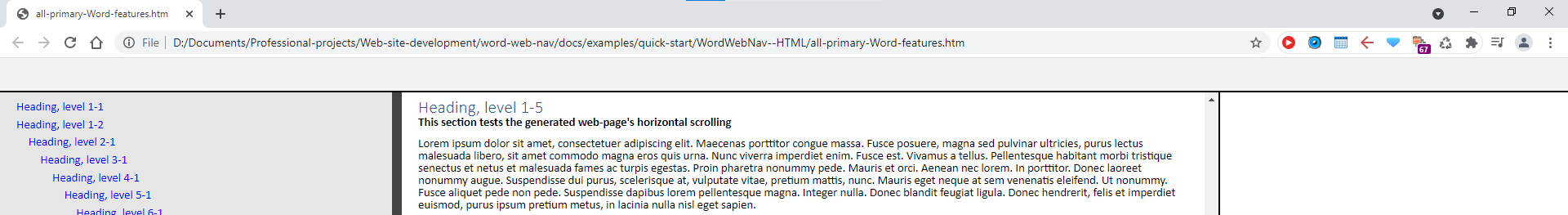
Word-HTML's paragraphs span the browser's width, which makes the paragraphs very difficult to read.

The bug is fixed by limiting the document-text's pane to a maximum of 960 pixels. The screen-shots are of a browser that's using the full-width of a display, which is 1920x1080 pixels. In both screen-shots, the same long paragraph is displayed.

* The Word HTML-file. The paragraph width is the browser's width, which is 1920 pixels in this example.

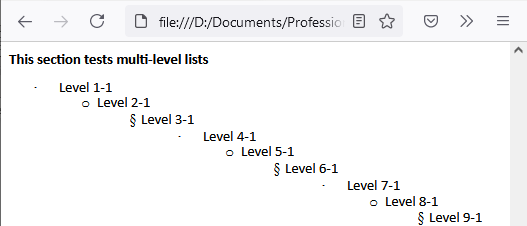


* The WWN web-page. The paragraph width is the width of the document-text pane. The document-text pane is at its maximum width, which is 960 pixels.

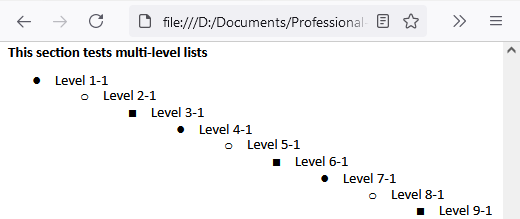


### Word-HTML's multi-level bulleted-lists are misformatted

In Word HTML, multi-level bulleted lists are misformatted. The spacing between the symbol and text is inconsistent. In Firefox on Windows, two of the bullet symbols are malformed.

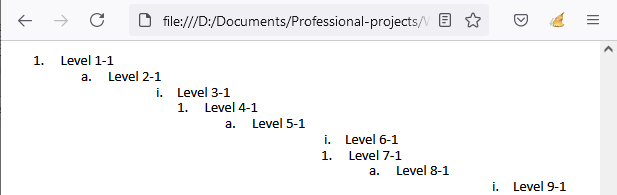


Those problems are fixed in the WWN web-page.

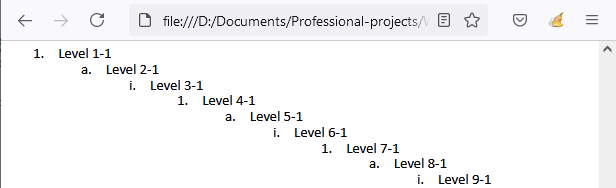


### Word-HTML's multi-level ordered-lists are misformatted

In Word HTML, multi-level ordered lists are misformatted. The indentation before the list symbols is inconsistent and incorrect.



Those problems are fixed in the WWN web-page.

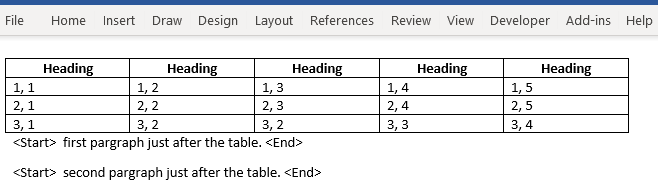


### Word-HTML can incorrectly make text white

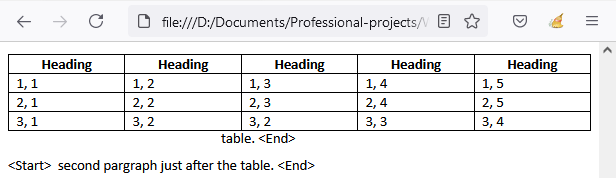
There is a Word-HTML bug which incorrectly makes text white that should be black. More info is in the earlier section that describes the create\_web\_page parameter-file, and its key "white\_colored\_text" (section 4.4 ).

This bug has primarily been observed in a paragraph when it immediately follows a table, as shown below. Though the bug does not occur in all such paragraphs.

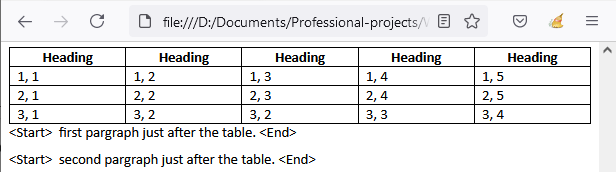
An example of a paragraph immediately after a table, in a Word-doc:



In the Word-HTML, some of the paragraph's text is incorrectly made white.



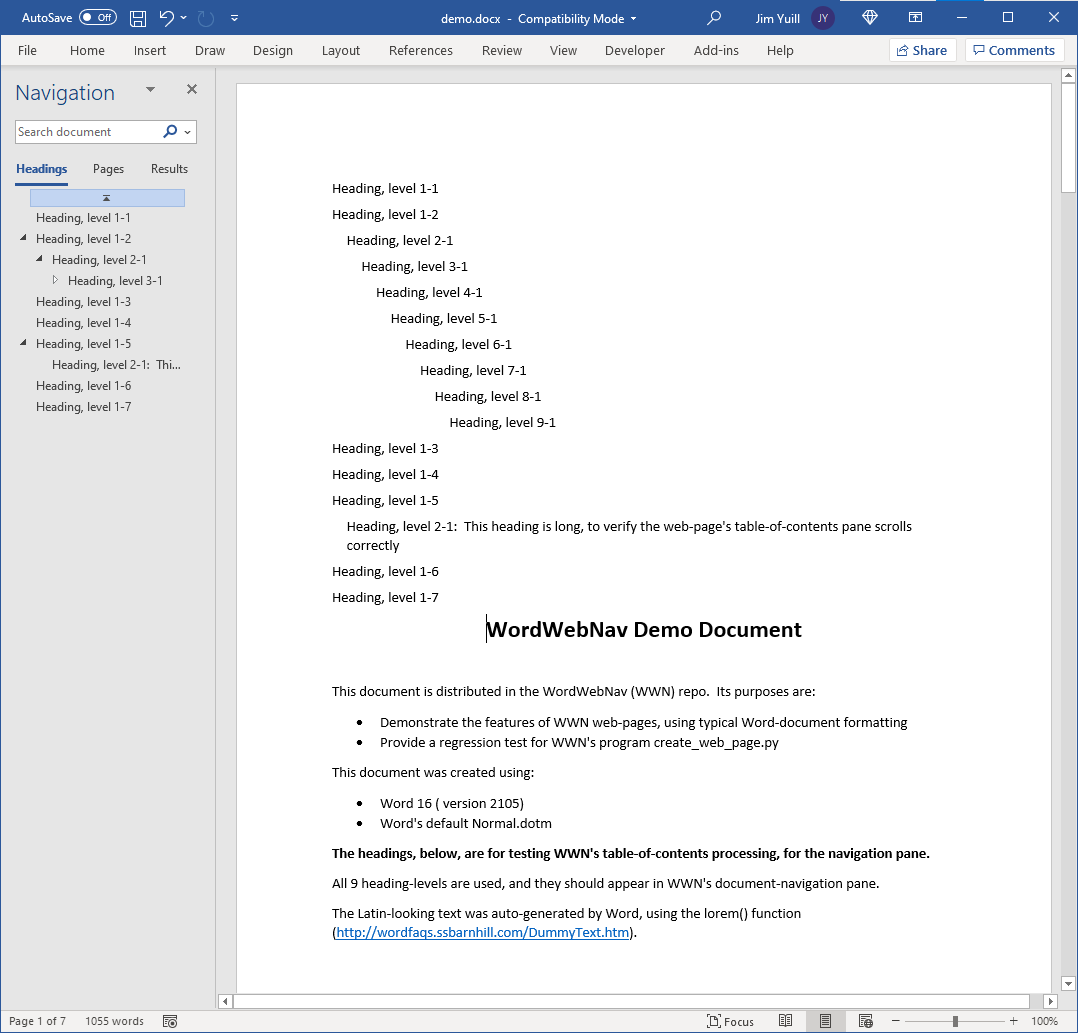
In the WWN web-page, the Word-HTML was fixed by removing the paragraph's style-attribute "color:white".



## Appendix C: Additional figures

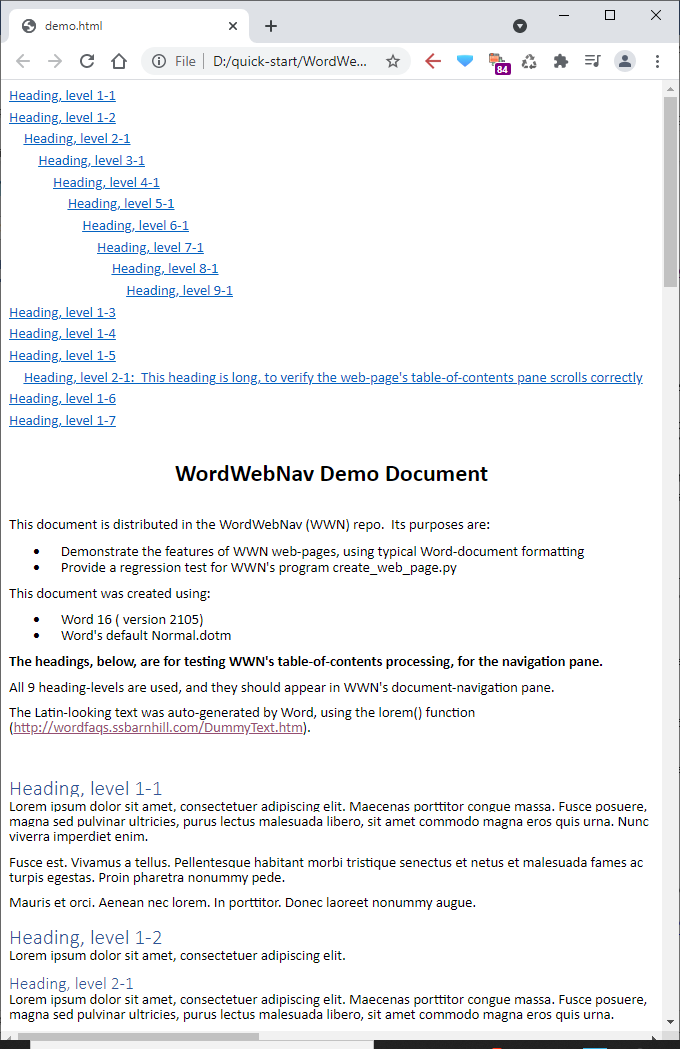
### Insert table-of-contents

The example Word-doc, after adding the TOC.



### Create the Word HTML-file

The Word HTML-file:



*Copyright (c) 2021-present by Jim Yuill, under the license at* [*https://github.com/jimyuill/word-web-nav*](https://github.com/jimyuill/word-web-nav)