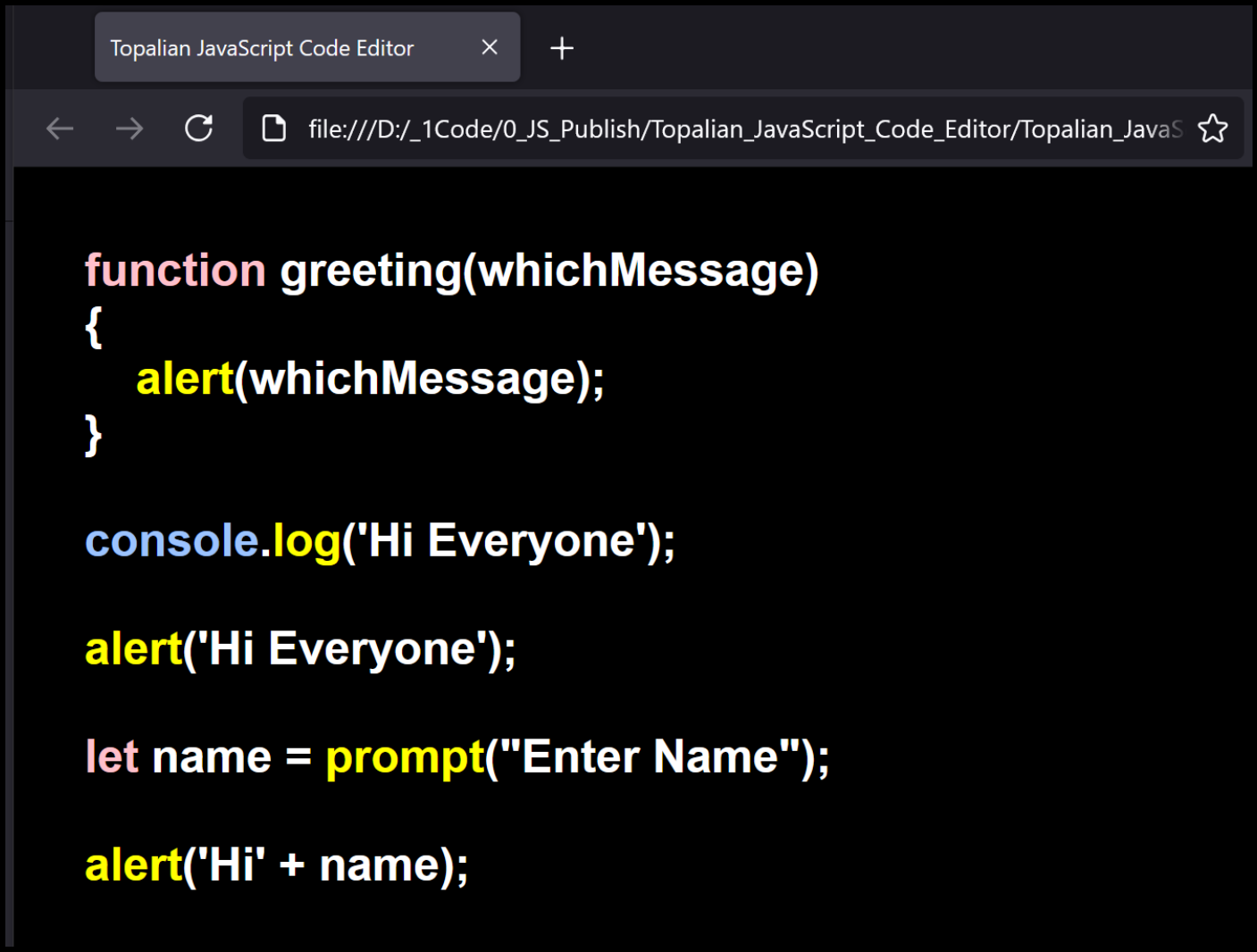
Topalian JavaScript Code Editor

by

Christopher Andrew Topalian

Dedicated to God the Father


Topalian JavaScript Code Editor
file:///D:/_1Code/0_JS_Publish/Topalian_JavaScript_Code_Editor/Topalian_JavaS
function greeting(whichMessage)
{
alert(whichMessage);
}
console.log('Hi Everyone');
alert('Hi Everyone');
let name = prompt("Enter Name");
alert('Hi' + name);

```
<!-- Dedicated to God the Father -->
<!-- All Rights Reserved Christopher Andrew Topalian Copyright 2000-2024 -->
<!-- https://github.com/ChristopherTopalian -->
<!-- https://github.com/ChristopherAndrewTopalian -->
<!-- Topalian_JavaScript_Code_Editor.html -->
<!-- Version 001 - (2024-07-04) -->

<html>
<head>
<title> Topalian JavaScript Code Editor </title>

<style>
```

```
body
{
    padding: 20px;
    background-color: rgb(0, 0, 0);
    font-family: Arial;
    color: white;
}

#editorContainer
{
    height: 300px;
}

#codeInput, #codeOutput
{
    position: absolute;
    top: 0;
```

```
    left: 0;
    width: 100%;
    height: 100%;
    padding: 10px;
    font-family: Arial;
    font-size: 24px;
    font-weight: bold;
    border: none;
    resize: none;
    tab-size: 4;
    overflow-y: scroll;
}

#codeInput
{
    background-color: rgba(0, 0, 0, 0);
    /* hide text in textarea */
    color: transparent;
```

```
    caret-color: white;
    z-index: 2;
    overflow-y: scroll;
}

#codeOutput
{
    /* if font size increaes, decrease top pos
corrospondingly */
    top: -22px;
    background-color: rgb(0, 0, 0);
    color: white;
    white-space: pre-wrap;
    word-break: break-word;
    /* make div not selectable */
    pointer-events: none;
    overflow-y: scroll;
    z-index: 1;
```

```
}

.pink
{
    color: pink;
}

.skyblue
{
    color: skyblue;
}

.yellow
{
    color: yellow;
}

.lightGray
```

```
{
    color: rgb(200, 200, 200);
}

.magenta
{
    color: rgb(255, 138, 255);
}

.objects
{
    color: rgb(155, 195, 255);
}

</style>

</head>
```

```
<body>

<div id = "editorContainer">
    <textarea id = "codeInput"></textarea>
    <pre id = "codeOutput"></pre>
</div>

<script>

let theEvents =
[
    'onabort',
    'onafterprint',
    'onanimationend',
    'onanimationiteration',
    'onanimationstart',
    'onbeforeprint',
    'onbeforeunload',
```

```
'onblur',
'oncanplay',
'oncanplaythrough',
'onchange',
'onclick',
'oncontextmenu',
'oncopy',
'oncut',
'ondblclick',
'ondrag',
'ondragend',
'ondragenter',
'ondragleave',
'ondragover',
'ondragstart',
'ondrop',
'ondurationchange',
'onemptied',
```

```
'onended',
'onerror',
'onfocus',
'onfocusin',
'onfocusout',
'onfullscreenchange',
'onfullscreenerror',
'onhashchange',
'oninput',
'oninvalid',
'onkeydown',
'onkeypress',
'onkeyup',
'onLine',
'onload',
'onloadeddata',
'onloadedmetadata',
'onloadstart',
```

'onmessage',
'onmousedown',
'onmouseenter',
'onmouseleave',
'onmousemove',
'onmouseover',
'onmouseout',
'onmouseup',
'onmousewheel',
'onoffline',
'ononline',
'onopen',
'onpagehide',
'onpageshow',
'onpaste',
'onpause',
'onplay',
'onplaying',

```
'onpopstate',
'onprogress',
'onratechange',
'onresize',
'onreset',
'onscroll',
'onsearch',
'onseeked',
'onseeking',
'onselect',
'onshow',
'onstalled',
'onstorage',
'onsubmit',
'onsuspend',
'ontimeupdate',
'ontoggle',
'ontouchcancel',
```

```
    'ontouchend',
    'ontouchmove',
    'ontouchstart',
    'ontransitionend',
    'onunload',
    'onvolumechange',
    'onwaiting',
    'onwheel'
];

let theFunctions =
[
    'abs',
    'acos',
    'acosh',
    'adoptNode',
    'alert',
    'appendChild',
```

```
'asin',
'asinh',
'assert',
'assert',
'assign',
'atan',
'atan2',
'atanh',
'atob',
'back',
'blur',
'btoa',
'cbrt',
'ceil',
'charAt',
'charCodeAt',
'clear',
'clearInterval',
```

```
'clearTimeout',
'clearWatch',
'click',
'close',
'closest',
'clz32',
'compile',
'concat',
'confirm',
'constructor',
'contains',
'copyWithin',
'cos',
'cosh',
'count',
'createAttribute',
'createComment',
'createDocumentFragment',
```

```
'createElement',
'createEvent',
'createTextNode',
'decodeURI',
'decodeURIComponent',
'encodeURI',
'encodeURIComponent',
'endsWith',
'entries',
'error',
'escape',
'eval',
'every',
'exec',
'execCommand',
'exitFullscreen',
'exp',
'expm1',
```

```
'fill',
'filter',
'find',
'fintIndex',
'floor',
'focus',
'forEach',
'forward',
'from',
'fromCharCode',
'fround',
'fullscreenEnabled',
'getDate',
'getDay',
'getAttribute',
'getAttributeNode',
'getBoundingClientRect',
'getComputedStyle',
```

'getCurrentPosition',
'getElementById',
'getElementsByClassName',
'getElementsByName',
'getElementsByTagName',
'getFullYear',
'getHours',
'getItem',
'getMilliseconds',
'getMinutes',
'getModifierState',
'getMonth',
'getNamedItem',
'getSeconds',
'getSelection',
'getTargetRanges',
'getTime',
'getTimezoneOffset',

```
'getUTCDate',
'getUTCDay',
'getUTCFullYear',
'getUTCHours',
'getUTCMilliseconds',
'getUTCMinutes',
'getUTCMonth',
'getUTCSeconds',
'getYear',
'go',
'group',
'groupCollapsed',
'groupEnd',
'hasAttribute',
'hasAttributes',
'hasChildNodes',
'hasFocus',
'importNode',
```

```
'includes',
'indexOf',
'info',
'insertAdjacentElement',
'insertAdjacentHTML',
'insertAdjacentText',
'insertBefore',
'isArray',
'isDefaultNamespace',
'isEqualNode',
'isFinite',
'isInteger',
'isNaN',
'isSafeInteger',
'isSameNode',
'isSupported',
'item',
'join',
```

```
'key',
'keys',
'lastIndexOf',
'localeCompare',
'log',
'log10',
'log1p',
'log2',
'map',
'match',
'matches',
'matchMedia',
'max',
'min',
'moveBy',
'moveTo',
'namedItem',
'normalize',
```

```
'normalizeDocument',
'now',
'Number',
'open',
'parse',
'parseFloat',
'parseInt',
'pop',
'preventDefault',
'print',
'push',
'querySelector',
'querySelectorAll',
'random',
'reduce',
'reduceRight',
'reload',
'remove',
```

```
'removeAttribute',
'removeAttributeNode',
'removeChild',
'removeEventListener',
'removeItem',
'repeat',
'replace',
'replaceChild',
'requestAnimationFrame',
'requestFullscreen',
'resizeBy',
'resizeTo',
'reverse',
'round',
'pow',
'prompt',
'removeEventListener',
'removeNamedItem',
```

```
'renameNode',
'scroll',
'scrollBy',
'scrollIntoView',
'scrollTo',
'search',
'setAttribute',
'setAttributeNode',
'setDate',
'setFullYear',
'setHours',
'setInterval',
'setItem',
'setMilliseconds',
'setMinutes',
'setMonth',
'setNamedItem',
'setSeconds',
```

```
'setTime',
'setTimeout',
'setUTCDate',
'setUTCFullYear',
'setUTCHours',
'setUTCMilliseconds',
'setUTCMinutes',
'setUTCMonth',
'setUTCSecond',
'setYear',
'shift',
'sign',
'sin',
'sinh',
'slice',
'some',
'sort',
'splice',
```

```
'split',
'startsWith',
'String',
'stop',
'stopImmediatePropagation',
'stopPropagation',
'stringify',
'substr',
'substring',
'table',
'tan',
'tanh',
'test',
'time',
'timeEnd',
'toDateString',
'toGMTString',
'toExponential',
```

```
'toFixed',
'toJSON',
'toISOString',
'toLocaleDateString',
'toLocaleLowerCase',
'toLocaleString',
'toLocaleTimeString',
'toLocaleUpperCase',
'toLowerCase',
'toPrecision',
'toString',
'toTimeString',
'toUpperCase',
'toUTCString',
'trace',
'trim',
'trunc',
'unescape',
```

```
    'unshift',
    'UTC',
    'valueOf',
    'warn',
    'watchPosition',
    'write',
    'writeln'
];

let theObjects =
[
    'console',
    'document',
    'window',
    'Date',
    'Math',
    'JSON',
];
```

```
let codeInput =
document.getElementById('codeInput');

let codeOutput =
document.getElementById('codeOutput');

function escapeHtml(code)
{
    return code.replace(/&/g,
'&amp;').replace(/</g, '&lt;').replace(/>/g,
'&gt;');
}

function highlightCode()
{
    let code = escapeHtml(codeInput.value);
```

```
    let functionPattern = new RegExp(`\\b(${theFunctions.join('|')})\\b`, 'g');

    let eventPattern = new RegExp(`\\b(${theEvents.join('|')})\\b`, 'g');

    let objectPattern = new RegExp(`\\b(${theObjects.join('|')})\\b`, 'g');

    let highlightedCode = code.replace(functionPattern, '<span style="color: yellow;">$1</span>')

    // events
    .replace(eventPattern, '<span class="flowControl">$1</span>')

    // objects
```

```
    .replace(objectPattern, '<span
class="objects">$1</span>')

    // function
    .replace(/\bfunction\b/g, '<span
class="pink">function</span>')

    // let
    .replace(/\blet\b/g, '<span
class="pink">let</span>')

    // new
    .replace(/\bnew\b/g, '<span
class="pink">new</span>')

    // digits
    .replace(/\d/g, '<span
class="lightGray">$&</span>')
```

```
    // flowControl
    .replace(/\b(if|else|do|while|for|forEach|
break|continue)\b/g, '<span
class="magenta">$1</span>');

    codeOutput.innerHTML = highlightedCode;
}

codeInput.addEventListener('input',
highlightCode);
codeInput.addEventListener('scroll',
function()
{
    codeOutput.scrollTop = this.scrollTop;
});

highlightCode();
```

```
codeInput.addEventListener("keydown",
function(event)
{
    if (event.key === "Tab")
    {
        // prevent default tab behavior
        event.preventDefault();

        // get cursor position
        let start = this.selectionStart;
        let end = this.selectionEnd;

        // insert 4 spaces at cursor position
        this.value = this.value.substring(0, start)
+ "    " + this.value.substring(end);
```

```
        // move cursor position forward by 4
spaces
        this.selectionStart = this.selectionEnd =
start + 4;
    }

    highlightCode();
});

</script>

</body>

</html>
```

Christopher Topalian

How to Combine .js files into one main.js file using Command Prompt

Version for when we have only ONE folder of .js files that we want to combine.

// HowToCombineJSFilesOneFolder.js

First, we add two new lines at the end of every script. This way they will later combine nicely with a line break in between each script.

We open our js folder.
In our js project folder, we type
cmd
in the address bar of the folder and then press enter

This opens our js folder in the Command prompt

We type in the words
copy *.js main.js
and then press enter

This creates a new file that is named main.js
This new file contains all .js files in ONE file.
But, there is a junk character at the end of the main.js script that we have to delete. In VSCode the character might be called SUB

```
    titleContainer.append(titleOfApp);
}
```

We remove this junk SUB character and the code will now run.

```
    titleContainer.append(titleOfApp);
}
```

As we can see, the junk character is removed.

<!-- Now, in our html code, we can type -->
<script src = 'main.js'></script>

This makes it much easier to upload our js code to our website.

Christopher Topalian

How to Combine .js files into one main.js file using Command Prompt

Version for when we have js scripts in subfolders in our js project folder, that we want to combine.

// HowToCombineJSFiles.js
TUTORIAL:
How to Combine all .js files in all folders that are in our js folder.

Getting things ready:
We should add two new lines at the end every script. This way they will combine nicely with a line break in between each script.

Step One: Open our js folder

Step Two: Type in the address bar of the js folder, cmd, press Enter

This opens our js folder in the command prompt

Step Three: Type the command shown below in the command prompt and then press Enter

```
for /r "%CD%" %i in (*.js) do type "%i" >> main.js
```

Now we have a newly created .js file named main.js that has all of our js files included into one file.

This makes it easy to upload our application and easy to find out how many lines of code our project is.

To use our main.js file, we include it in our html file code:

```
<script src = 'js/main.js'></script>
```

Happy Scripting :-)

Christopher Topalian

How to Combine .js files into one main.js file using a batch file

Version for when we have js scripts in subfolders in our js project folder, that we want to combine.

// HowToCombineJSFilesUsingBatFile.js

We can combine all of the .js files
that are located in our js folder
into one main.js file, using either:

- The Command Prompt Method
- or
- The .bat File Method

The .bat file method is very easy.
We double click the bat file, which is located in our js folder, and it will make a main.js file, which includes all .js files in the js folder, including all .js files in all subdirectories of our js folder.

This is a very easy way to combine our .js files, because we can double click the .bat file

anytime, and it will again generate the main.js file, which includes all .js files in the js folder, including all .js files in all subdirectories of our js folder. This makes uploading our application online much easier.

Happy Scripting :-)

```
:: Topalian_Combine_JS_Files.bat

:: This .bat File Combines All .js files in all folders of our project folder, into one main.js file.

:: To activate this .bat file, we double click the .bat file, while it is located in our js folder.

@echo off
:: set the output file name
```

```
set "output=main.js"

:: clear existing output file
type nul > "%output%"

:: loop through all JavaScript files in
subdirectories
for /r %%i in (*.js) do (
    :: append the content of each file to the
output file
    type "%%i" >> "%output%"
)

echo "JavaScript files combined into
%output% successfully."
```

Christopher Topalian

How to Combine .js files into one main.js file using Node.js

This version will successfully combine a single folder of js files.
It also works to combine js files in all subdirectories.

```
// Topalian_Combine_JS_Files.js

let fs = require('fs');
let path = require('path');

function combineJSFiles(directory,
scriptFilename)
{
    let outputFilePath = path.join(directory,
'main.js');

    let fileContents = [];

    function traverseFolder(folder)
    {
        let files = fs.readdirSync(folder);

        for (let i = 0; i < files.length; i++)
```

```
    {
        let file = files[i];

        let filePath = path.join(folder, file);

        let stats = fs.statSync(filePath);

        if (stats.isDirectory())
        {
            traverseFolder(filePath);
        }
        else if (path.extname(filePath) === '.js')
        {
            let content =
fs.readFileSync(filePath, 'utf8');
            // check if file is not script file itself
            if (filePath !== scriptFilename)
            {
```

```
                fileContents.push(content);
            }
        }
    }
}

traverseFolder(directory);

fs.writeFileSync(outputFilePath,
fileContents.join('\n'), 'utf8');

console.log(`Combined ${
{fileContents.length} .js files into ${
{outputFilePath}`);
}

// get current directory of script
let currentDirectory = process.cwd();
```

```
// get filename of script
let scriptFilename = __filename;

combineJSFiles(currentDirectory,
scriptFilename);
```

Christopher Topalian

How to Combine .js files into one main.js file using Python

This version will successfully combine a single folder of js files.
It also works to combine js files in all subdirectories.

```
# Topalian_Combine_JS_Files.py

import os

def combineJSFiles(directory,
scriptFileName):
    outputFilePath = os.path.join(directory,
'main.js')
    fileContents = []

    def traverseFolder(folder):
        for root, dirs, files in os.walk(folder):
            for file in files:
                filePath = os.path.join(root, file)
                if filePath != scriptFileName and
filePath.endswith('.js'):
                    with open(filePath, 'r',
encoding='utf-8') as f:
```

```
                fileContents.append(f.read())

    traverseFolder(directory)

    with open(outputFilePath, 'w',
encoding='utf-8') as f:
        f.write('\n'.join(fileContents))
    print(f"Combined {len(fileContents)} .js
files into {outputFilePath}")

# get current directory of script
currentDirectory =
os.path.dirname(os.path.abspath(__file__))

# get filename of script
scriptFileName = os.path.abspath(__file__)
```

```
combineJSFiles(currentDirectory,
scriptFileName)
```

What other file types can we combine?

We have combined .js files in this book, but we might choose to instead combine:

.py or .html or .txt

This is very useful for book making.

In each of the scripts shown in this book, we can manually change the parts where it says .js, with .py, if we wanted to, for instance, copy all .py files into one main.py file.

We can do the same thing for .html files, where we change the file type it will be combining to .html and it will combine all .html files into one main.html file.

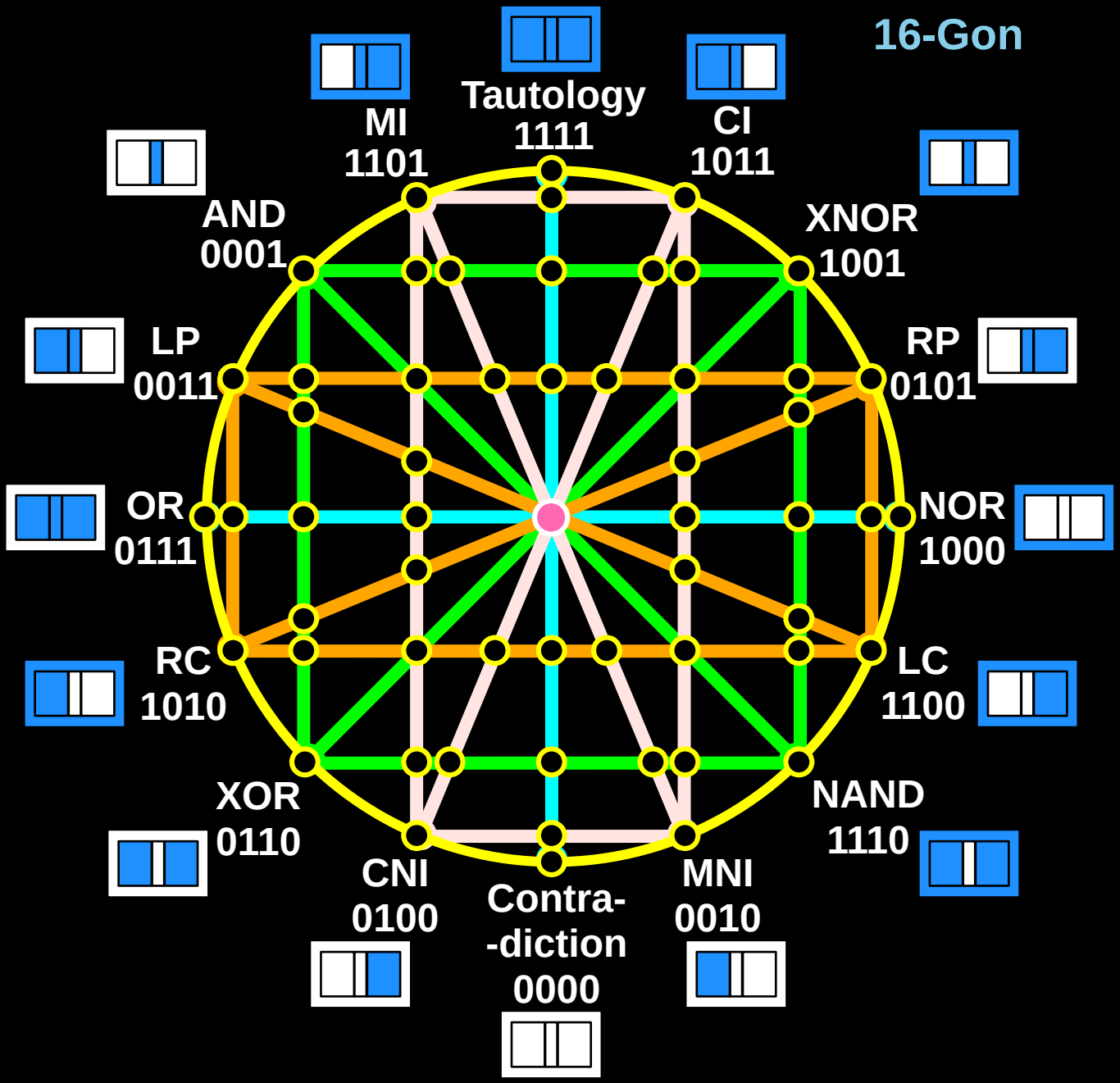
We add two line breaks at the end of all files, so that there is space between files, when they are combined.

Remember too, that not all file types will combine, but the ones above will.

The original files are not changed. The content from the original files is only copied from.

Christopher Topalian

True Artificial Intelligence System



For More Tutorials:

GitHub.com/ChristopherTopalian

GitHub.com/ChristopherAndrewTopalian

Sites.google.com/view/CollegeOfScripting

CollegeOfScripting.weebly.com

CollegeOfScripting.wordpress.com

Youtube.com/ScriptingCollege

Twitter.com/CollegeOfScript

Rumble.com/user/CollegeOfScripting

Dedicated to God the Father

This book is created by the
College of Scripting Music & Science.
Always remember, that each time you write a script with a pencil and paper, it becomes imprinted so deeply in memory that the material and methods are learned extremely well.

When you Type the scripts, the same is true. The more you type and write out the scripts by keyboard or pencil and paper, the more you will learn programming!

Write and Type every example that you find.
Keep all of your scripts organized.
Every script that you create increases your programming abilities.
SEEING CODE, is one thing,
but WRITING CODE is another.
Write it, Type it, Speak it, See it, Dream it.

CollegeOfScripting.weebly.com