IDEs and Development

Features that IDEs offer to assist with development.

IDEs, or Integrated Development Environments, are text editors with multiple features that assist with development of software (Walker, 2018). These editors are used with the source code of software, ranging from individual files in some environments, up to solutions and workspaces with many files and assets involved.

These are the tools that I utilise on a regular basis, seeking out IDEs or plugins that include these features.

Syntax Highlighting

```
if (window == NULL)
{
    printf("SDL could not create window: %s\n", SDL_GetError());
    SDL_Delay(2000);
    return 0;
}
```

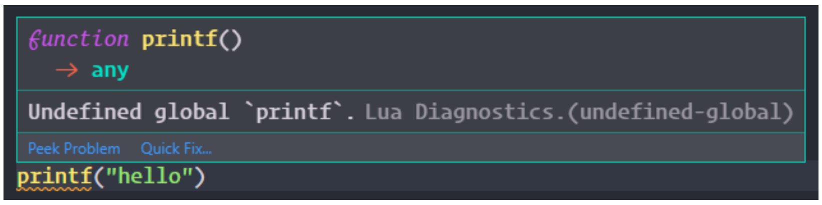
Syntax Highlighting is the process by which a script or document will have colours or typefaces to denote the differences between certain characters, words and strings in the document (Sarkar, 2015). Syntax highlighting depends on the language being used, with different syntax highlighting engines having different supported languages. Google Code Prettify, for example, has a set of languages that it supports, along with a generic highlighter that attempts to apply syntax highlighting on any language (amroamroamro, et al., 2015 - 2019).

The impact this has on readability and comprehension has been shown to accelerate the reading and understanding of code, with the improvement decreasing as an individual's programming experience increases (Sarkar, 2015). This implies that code with syntax highlighting is more legible and easier to understand because of the extra dimension of information provided by said highlighting.

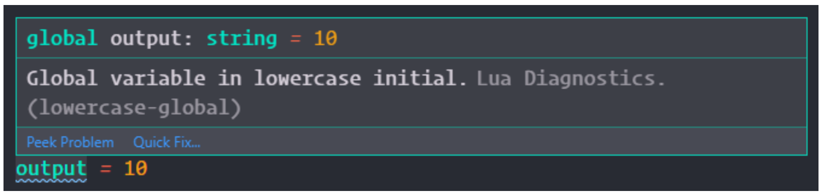
Syntax highlighting isn't limited to colour, with certain fonts and IDEs supporting typeface changes (italic, bold and cursive, as examples) to further illustrate syntax.

```lua
if (love.mouse.isDown(1) or love.keyboard.isDown("return")) then
    love.window.close()
end
```

Linting



```
function printf()
  → any
Undefined global `printf`. Lua Diagnostics.(undefined-global)
Peek Problem    Quick Fix...
printf("hello")
```



```
global output: string = 10
Global variable in lowercase initial. Lua Diagnostics.
(lowercase-global)
Peek Problem    Quick Fix...
output = 10
```

Linting is a form of static code analysis that analyses code and identifies any issues with it, such as programmatic issues, where variables and methods are called when they don't exist, or when they're invoked with the wrong arguments (Bellairs, 2019). As well as this, it highlights stylistic errors, like with issues relating to variable name conventions (i.e. PascalCase for global variables, and camelCase for local variables) (lua-users, 2017).

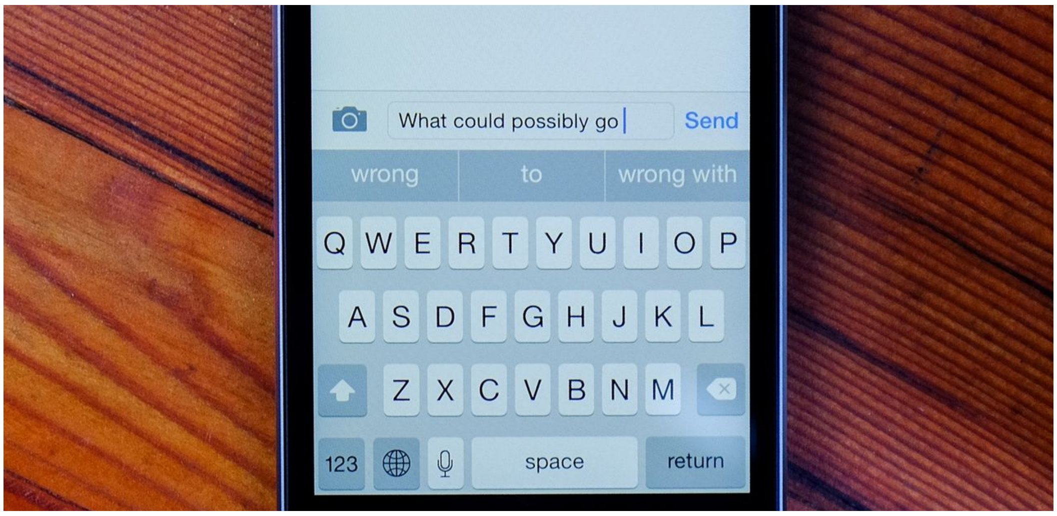
Linting serves a similar function to compiler errors in that they demonstrate errors and warnings in any given segment of code. The difference between the two is that Linting doesn't require full compilation, and instead looks at the code without compiling or executing it, listing any errors or warnings that exist as well as where they're located. This is what makes it a form of static code analysis (techopedia, 2019).

The utility of linting is that it allows errors to be identified as the code is being written or looked at, without requiring full compilation or execution to identify. The error highlighting and the popup windows that show the error are especially useful as it accelerates the identification of what the underlying issue is.

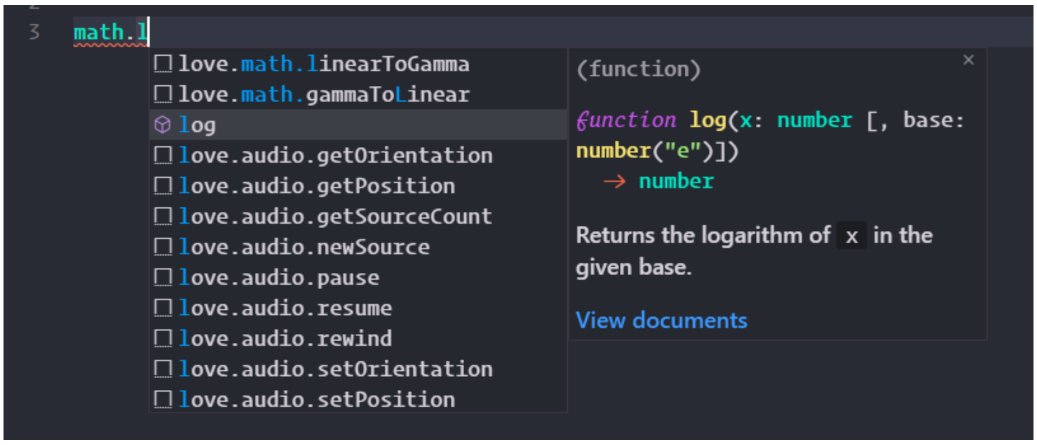
Intellisense

Intellisense is a feature in Visual Studio related IDEs that contains a few features, namely code completion, parameter info, quick info and member lists (Microsoft, 2019). While this is a copyrighted and proprietary feature, the set of features that Intellisense includes are more general and included under the term itself.

Code completion is a service that identifies and fills keywords that are available and relevant to the current context. A similar process is the autocomplete that keyboards on phones and tablets offer, differing in that those are suggested for the sentence, whereas with IDEs the suggestions come from the entire workspace.



(Lowensohn, 2014)



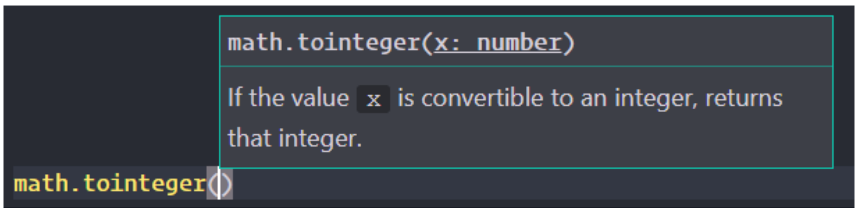
```
math.l
love.math.linearToGamma
love.math.gammaToLinear
log
love.audio.getOrientation
love.audio.getPosition
love.audio.getSourceCount
love.audio.newSource
love.audio.pause
love.audio.resume
love.audio.rewind
love.audio.setOrientation
love.audio.setPosition
```

```
(function)

function log(x: number [, base:
number("e")])
  -> number
```

Returns the logarithm of `x` in the given base.

View documents

As described before, Intellisense also includes parameter info and quick info, both of which explain the suggestions that are provided. The quick info is shown above (along with the code completion), explaining what the math.log() function does, as well as what parameters must be provided. Two overflows are shown, one with a single value that is equivalent to `ln(x)`, and another with two values that is equivalent to $\log_y(x)$. This information can be essential for speeding up the writing of code, reducing the amount of documentation that the reader needs to look through.



```
math.tointeger(x: number)
```

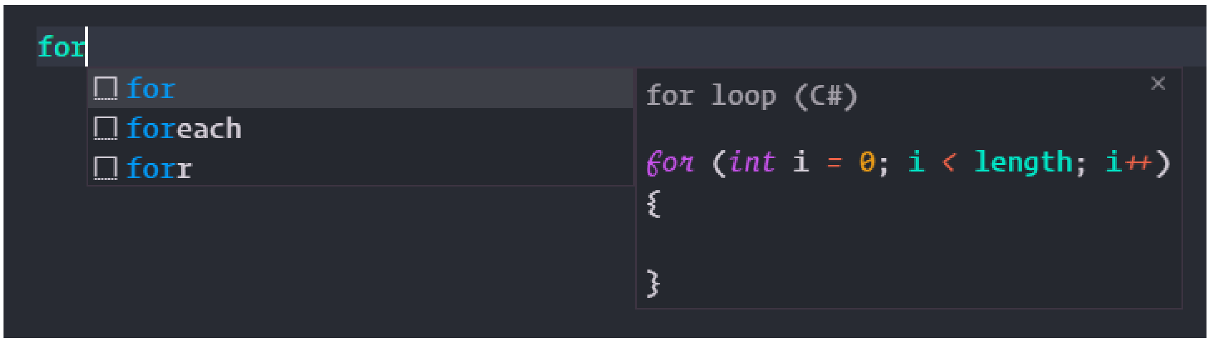
If the value `x` is convertible to an integer, returns that integer.

```
math.tointeger()
```

Parameter info describes what kind of parameters should be passed to a function or method, usually describing what type (i.e. integer or Boolean), and what the purpose of the function/method is. In the example above, x has no specified type, however the info shows that any value that could be converted to an integer, like a string or a bool, is accepted. It's unclear whether or not a float would work, however there are rounding functions for that.

Code Snippets

Snippets extend code completion by including keywords that fill blocks of reusable code. These snippets can greatly accelerate code production, whether automated with keywords or just a set of snippets in a file or on a webpage, by showing a block of code that performs a set purpose. There are online repositories with code snippets (CSS-Tricks, n.d.), as well as plugins and systems that include them with autocompletion (Microsoft, 2019).



```
for
```

for loop (C#)

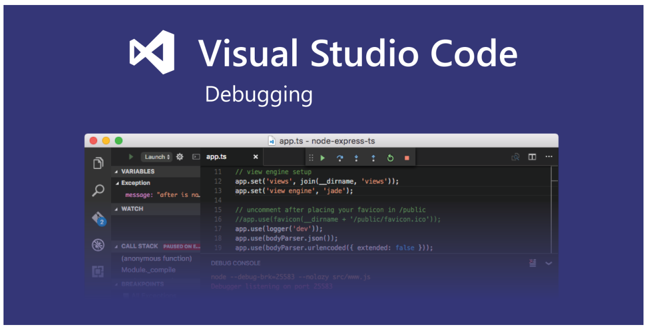
```
for (int i = 0; i < length; i++)
{

}
```

These snippets are incredibly useful for code development, as they automate a lot of the process by simplifying the production of certain chunks of code. For example, the screenshot above shows autocomplete for C#'s for loops, foreach loops, as well as a reversed for loop (i--). This is especially useful when the programmer knows a for loop is required, they can just fill the type they need and not worry about typing it out or potentially getting the order wrong.

More complex snippets are available for a variety of languages, including snippets that perform much more complex operations such as converting a polygon to path data in Javascript (Coyier, 2016).

Debugger

Many IDEs have debuggers built in or available as plugins, which allow for code to be executed under special conditions, such as simulated instruction-sets, allowing for halting and the view of memory in real-time, along with many other features which help to identify any potential issues within code (LeopardSkinPillBoxHat, 2009) (Techopedia, 2020). I don't use debuggers very often, only when there's a bug I can't identify without the assistance of a debugger.



```
// view engine setup
app.set('views', join(__dirname, 'views'));
app.set('view engine', 'jade');

// uncomment after placing your favicon in /public
//app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
```

```
node --debug-brk=25583 --nolazy src/www.js
Debugger listening on port 25583
```

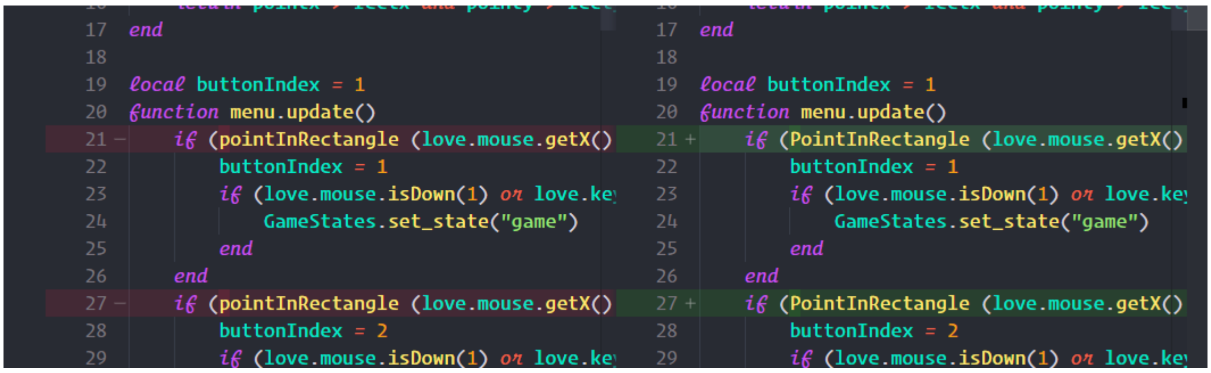
(Microsoft, 2019)

Plugins

Plugins are incredibly useful in an IDE, significantly expanding the potential functionality of the software. Plugins can offer features that are not packaged with the software, greatly extending the advantages of using an IDE, along with usability and coding speed/accuracy. Some of the main plugins that I use are as follows:

Github

Github is a source control repository that allows for version control, software sharing, collaboration, and many other features. It uses the Git version control system to handle uploads and downloads, with extended functionality for repositories allowing for collaboration, forks, pull requests and changelogs (Brown, 2019). I use Github as my primary source control system, with plugins in my IDEs that allow for commits, pushes, fetches and so on. This system is incredibly useful due to the integrated view for changes, showing what needs to be committed to the repository, as well as what was added, removed, or changed.

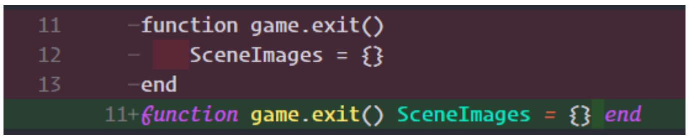


```
end

local buttonIndex = 1
function menu.update()
-   if (pointInRectangle (love.mouse.getX()
        buttonIndex = 1
        if (love.mouse.isDown(1) or love.ke
            GameStates.set_state("game")
        end
    end
-   if (pointInRectangle (love.mouse.getX()
        buttonIndex = 2
        if (love.mouse.isDown(1) or love.ke
```

```
end

local buttonIndex = 1
function menu.update()
+   if (PointInRectangle (love.mouse.getX()
        buttonIndex = 1
        if (love.mouse.isDown(1) or love.ke
            GameStates.set_state("game")
        end
    end
+   if (PointInRectangle (love.mouse.getX()
        buttonIndex = 2
        if (love.mouse.isDown(1) or love.ke
```

Code Formatters

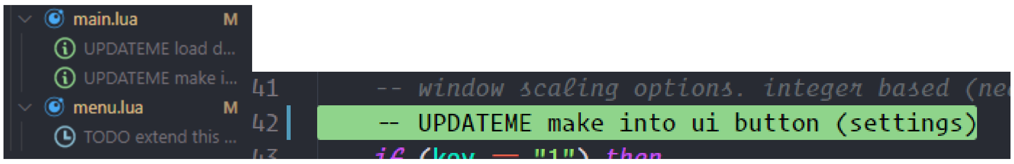
The specific formatter that I use, prettier, is designed to work with many languages, and formats code in a specified style, reducing or adding whitespace where necessary, and generally tidying up code. This helps fit the code I produce to style conventions, improving readability and standardisation (Petersen, 2017-2019).



```
-function game.exit()
-    SceneImages = {}
-end
+function game.exit() SceneImages = {} end
```

Todo Tree

This is another plugin that I use on a regular basis, referring to comments with any keywords in them, which I can specify. This is incredibly useful because it allows for a list of incomplete or broken sections of code, especially with the keyword FIXME (Gruntfuggly, 2017-2019).

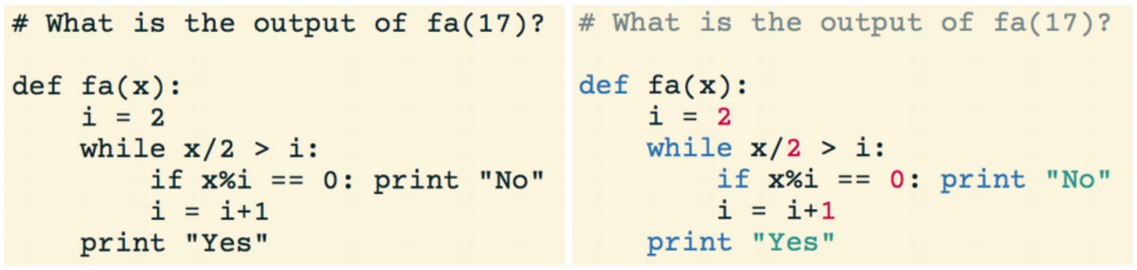


```
main.lua                M
    UPDATEME load d...
    UPDATEME make i...
menu.lua                M
    TODO extend this ...
```

```
    -- window scaling options. integer based (ne
    -- UPDATEME make into ui button (settings)
```

Utility of an IDE

IDEs are effectively just text editors with lots of expansion designed to assist developers. Whether or not an IDE is more effective than a regular text editor, for example notepad or even a more functional program like notepad++, is clear. The study I referred to in the syntax highlighting section alone outlines that syntax highlighting improves legibility, understanding and consistency of the interpretation of code for both beginners and experienced developers.



```
# What is the output of fa(17)?

def fa(x):
    i = 2
    while x/2 > i:
        if x%i == 0: print "No"
        i = i+1
    print "Yes"
```

```
# What is the output of fa(17)?

def fa(x):
    i = 2
    while x/2 > i:
        if x%i == 0: print "No"
        i = i+1
    print "Yes"
```

Fig. 1. Left: code without highlighting. Right: same code with syntax highlighting.

(Sarkar, 2015)

Other functionality clearly improves the efficiency of development using an IDE rather than without, with snippets allowing for huge amounts of automation, especially alongside linting and intellisense, cutting down the amount of compilation or interpretation required by huge amounts, instead having integrated depictions of whether code is written correctly or functional. For these reasons alone, I think it's clear that IDEs are incredibly effective when compared with regular text editors, especially because I find the features afforded by the usage of an IDE to be so useful.

References

amroamroamro, et al., 2015 - 2019. *google/code-prettify.* [Online]
Available at: https://github.com/google/code-prettify
[Accessed 31st December 2019].

Bellairs, R., 2019. *Why Is Linting Important? And How To Use Lint Tools.* [Online]
Available at: https://www.perforce.com/blog/qac/why-linting-important-and-how-use-lint-tools
[Accessed 31st December 2019].

Brown, K., 2019. *What Is GitHub, and What Is It Used For?.* [Online]
Available at: https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/
[Accessed 2nd January 2020].

Coyier, C., 2016. *Convert Polygon to Path Data.* [Online]
Available at: https://css-tricks.com/snippets/javascript/convert-polygon-path-data/
[Accessed 2nd January 2020].

CSS-Tricks, n.d.. *Snippets.* [Online]
Available at: https://css-tricks.com/snippets/
[Accessed 2nd January 2020].

Gruntfuggly, 2017-2019. *Todo Tree.* [Online]
Available at: https://marketplace.visualstudio.com/items?itemName=Gruntfuggly.todo-tree
[Accessed 2nd January 2020].

LeopardSkinPillBoxHat, 2009. *Why is debugging better in an IDE? [closed].* [Online]
Available at: https://stackoverflow.com/questions/426569/why-is-debugging-better-

in-an-ide
[Accessed 2nd January 2020].

Lowensohn, J., 2014. *I let Apple's QuickType keyboard take over my iPhone.* [Online]
Available at: https://www.theverge.com/2014/9/17/6337105/breaking-apples-quicktype-keyboard
[Accessed 2nd January 2020].

lua-users, 2017. *Lua Style Guide.* [Online]
Available at: http://lua-users.org/wiki/LuaStyleGuide
[Accessed 31st December 2019].

Microsoft, 2019. *Debugging.* [Online]
Available at: https://code.visualstudio.com/docs/editor/Debugging
[Accessed 2nd January 2020].

Microsoft, 2019. *IntelliSense.* [Online]
Available at: https://code.visualstudio.com/docs/editor/intellisense#_intellisense-for-your-programming-language
[Accessed 2nd January 2020].

Microsoft, 2019. *Snippets in Visual Studio Code.* [Online]
Available at: https://code.visualstudio.com/docs/editor/userdefinedsnippets
[Accessed 2nd January 2020].

Petersen, E., 2017-2019. *Prettier - Code formatter.* [Online]
Available at: https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode
[Accessed 2nd January 2020].

Sarkar, A., 2015. *The impact of syntax colouring on program comprehension.* [Online]
Available at: https://docs.google.com/viewerng/viewer?url=http://www.ppig.org/sites/ppig.org/files/2015-PPIG-26th-Sarkar.pdf
[Accessed 31st December 2019].

techopedia, 2019. *Static Code Analysis.* [Online]
Available at: https://www.techopedia.com/definition/24621/static-code-analysis
[Accessed 31st December 2019].

Techopedia, 2020. *Techopedia explains Debugger.* [Online]
Available at: https://www.techopedia.com/definition/597/debugger
[Accessed 2nd January 2020].

Walker, A., 2018. *What is an IDE (Integrated Development Environment)?.* [Online]
Available at: https://learn.g2.com/ide
[Accessed 31st December 2019].