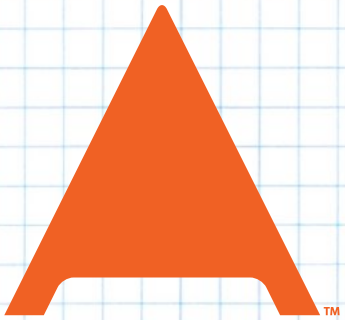
www.phparch.com

March 2017
VOLUME 16 - Issue 3



php[architect]

# Back to the Drawing Board

## ALSO INSIDE

**Education Station:** Validate the Complexity of Your Writing With TextStatistics

**Community Corner:** Three Tech Lessons I Learned Cleaning the Kitchen

**Leveling Up:** Exploring Object Immutability

**finally{}:** Working Together

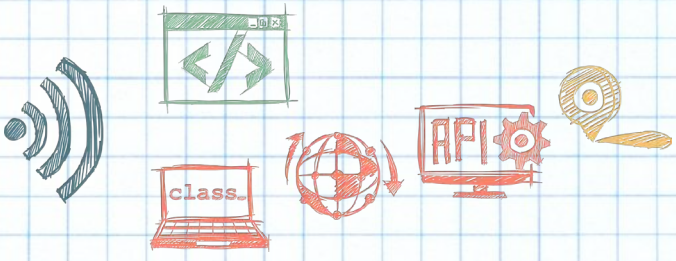
Learning to Code with Minecraft, Part Two

Deploying to Docker Swarm

How to Use SELinux (No, I Don't Mean Turn It Off)

Pursuing a Graduate Degree as Professional Development





# PHP[TEK] 2017

The Premier PHP Conference
12th Annual Edition

**May 24-26 — ATLANTA**

## Keynote Speakers:

Gemma Anible
**WonderProxy**

Keith Adams
**Slack**

Alena Holligan
**Treehouse**

Larry Garfield
**Platform.sh**

Mitch Trale
**PucaTrade**

Samantha Quiñones
**Etsy**

**Sponsored By:**

php women · ShootProof · Zeamster · Drupal Sourcing Mental Illness · platform.sh · stickermule

tek.phparch.com

php[architect]

# CONTENTS

MARCH 2017
Volume 16 - Issue 3

# Back to the Drawing Board

## Features



## Columns





**Subscriptions**
Print, digital, and corporate subscriptions are available. Visit https://www.phparch.com/magazine to subscribe or email contact@phparch.com for more information.

**Advertising**
To learn about advertising and receive the full prospectus, contact us at ads@phparch.com today!

**Managing Partners**
Kevin Bruce, Oscar Merida, Sandy Smith

php[**architect**] is published twelve times a year by:
musketeers.me, LLC
201 Adams Avenue
Alexandria, VA 22301, USA



**Contact Information:**
**General mailbox:** contact@phparch.com
**Editorial:** editors@phparch.com

**Print ISSN** 1709-7169
**Digital ISSN** 2375-3544

# Validate the Complexity of Your Writing With TextStatistics

*Matthew Setter*

**While code can often be the foundation of what we do on a daily basis; it's not the only thing which is important. Well, it shouldn't be, if it is. Another, dare I say, equally important part is documentation. This can be class and function documentation, end user documentation, tutorials—even creating broader content such as screencasts and online training.**

Given that, we have to regularly ask ourselves:

> *Is the documentation we're writing able to be read and comprehended by the majority of people who will read it?*

## Welcome to TextStatistics

Perhaps you've not stopped to consider this previously. If not (or even if you have), in this month's edition of Education Station, I'm going to introduce you to a library which can help you gauge the complexity of your writing. If you work with a content management system, these tools can help your content authors assess the complex nature of their writing, and hopefully simplify it to reach site visitors.

The library is called TextStatistics[1], by Dave Child. The library, as the repository describes it:

> *Generates information about text including syllable counts and Flesch-Kincaid, Gunning-Fog, Coleman-Liau, SMOG and Automated Readability scores.*

These terms might not mean a lot to you. So, before we dive into the library, let's look at what they mean. Collectively, Flesch-Kincaid, the Gunning Fog Index, Coleman-Liau, and SMOG are all ways of calculating a readability score.

### What is a Readability Score?

According to readable.io[2], a readability score is:

> *A computer-calculated index which can tell you roughly what level of education someone will need to be able to read a piece of text easily.*

It goes on to say that:

> *There are several algorithms available for measuring scores, and these use factors like sentence length, syllable count, percentage of multi-syllable words and so on to calculate their own readability score.*

To summarize, these algorithms are all means of estimating the complexity of a given piece of text. Once you've determined its complexity, you can then know, with a rough degree of certainty, what level of education a person will require to fully understand it. You can then more accurately write to suit your reader.

## Gunning Fog

But what are these tests? Let's take a more detailed look, starting with Gunning Fog[3].

> *In linguistics, the Gunning fog index is a readability test for English writing. The index estimates the years of formal education a person needs to understand the text on the first reading. A fog index of 12 requires the reading level of a U.S. high school senior (around 18 years old).*

## Coleman-Liau

Next, there's Coleman-Liau[4]:

> *The Coleman–Liau index was designed to be easily calculated mechanically from samples of hard-copy text. Unlike syllable-based readability indices, it does not require that the character content of words be analyzed, only their length in characters.*

## SMOG

Then, there's SMOG[5]:

> *SMOG Readability Formula estimates the years of education a person needs to understand a piece of writing.*

1 *TextStatistics:* http://phpa.me/gh-text-statistics

2 *According to readable.io:* https://readable.io

3 *Gunning Fog:* http://phpa.me/gunning-fox-index

4 *Coleman-Liau:* http://phpa.me/coleman-liau-index

5 *SMOG:* http://phpa.me/smog-readability-formula

*McLaughlin created this formula as an improvement over other readability formulas.*

## Flesch-Kincaid

Finally, there's theFlesch-Kincaid, which we'll focus on, primarily, for the rest of the column.

*The Flesch–Kincaid readability tests are readability tests designed to indicate how difficult a passage in English is to understand. There are two tests, the Flesch Reading Ease, and the Flesch–Kincaid Grade Level.*

In a nutshell, the higher a piece of text scores on the *Reading Ease* test, then the lower it will score on the *Grade Level* test. Said another way, the easier a piece of text is to read, then the lower the level of education is required to read it. You can see the formula for how the Reading Ease test is calculated in Figure 1.

Figure 1.

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$

And in the table below, you can see how the scores marry-up with the grade levels.

| Score | School Level | Notes |
|---|---|---|
| 100.00–90.00 | 5th grade | Very easy to read. Easily understood by an average 11-year-old student. |
| 90.0–80.0 | 6th grade | Easy to read. Conversational English for consumers. |
| 80.0–70.0 | 7th grade | Fairly easy to read. |
| 70.0–60.0 | 8th & 9th grade | Plain English. Easily understood by 13–15-year-old students. |
| 60.0–50.0 | 10th to 12th grade | Fairly difficult to read. |
| 50.0–30.0 | College | Difficult to read. |
| 0–30.0 | College Graduate | Very difficult to read. Best understood by university graduates. |

Now that we have all this, how do we go about measuring it (or any of the other tests)? Good question. That's why we have the subject of this month's edition: `TextStatistics`.

It provides methods for all of the tests above, along with a range of other methods, for judging the complexity of a piece of text in simpler ways. These include the number of words, sentences, syllables and so on. We're going to work through the tests first, and then finish up with the more simple methods of text analysis.

## Installing It

It should go without saying, but we're going to install the library using Composer. To do so, run the following command from the command line, in the root directory of your project:

```
composer require davechild/textstatistics
```

This will, after a short period, import the package into the `vendor/` directory for your project (or make it the first one, if you're just experimenting with it).

## Let's Write Some Code

Now we're ready to see how it works. Honestly, there's not a lot to it. To sum it up, all we need to do is pass a piece of text to one of the respective functions, and it will report back a score.

To get started, in a new file, let's be unoriginal and call it `index.php`, add the following code:

```
<?php

require_once ('vendor/autoload.php');

use DaveChild\TextStatistics as TS;
```

With that, we're ready to go, but we'll need a piece of text to analyze. To get a proper gauge, we'll need at least two. We'll take the following quote, from Iron and the Soul[6], by the legendary Henry Rollins.

*When I was young I had no sense of myself. All I was, was a product of all the fear and humiliation I suffered. Fear of my parents. The humiliation of teachers calling me "garbage can" and telling me I'd be mowing lawns for a living. And the very real terror of my fellow students. I was threatened and beaten up for the color of my skin and my size. I was skinny and clumsy, and when others would tease me I didn't run home crying, wondering why.*

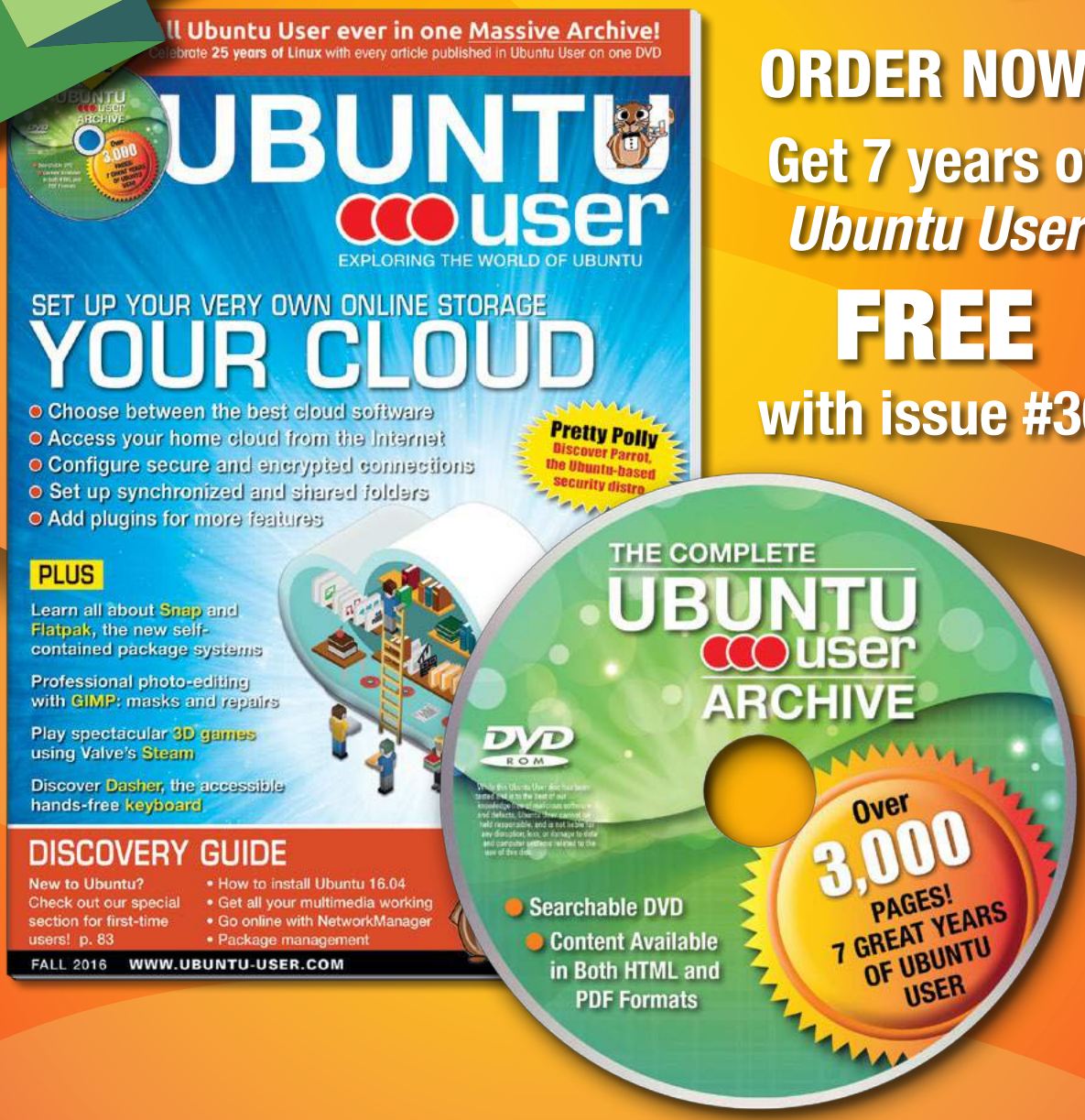
Now, let's get a piece of text to contrast it with, something which is classically considered to be more sophisticated, more complicated. The excerpt below is from Act 2, Scene 1 of The Tempest by William Shakespeare.

*Beseech you, sir, be merry; you have cause, So have we all, of joy; for our escape Is much beyond our loss. Our hint of woe Is common; every day some sailor's wife, The masters of some merchant and the merchant Have just our theme of woe; but for the miracle, I mean our preservation, few in millions Can speak like us: then wisely, good sir, weigh Our sorrow with our comfort.*

Update the code to include variables for both the pieces of

6 *Iron and the Soul:* http://www.artofmanliness.com/trunk/?p=1748

# Celebrating 25 Years of Linux!




## ORDER NOW!

Get 7 years of *Ubuntu User*

**FREE**

with issue #30

***Ubuntu User* is the only magazine for the Ubuntu Linux Community!**

**BEST VALUE:** Become a subscriber and save 35% off the cover price!
The archive DVD will be included with the Fall Issue, so you must act now!

**Order Now! Shop.linuxnewmedia.com**

text, as in Listing 1.

Then, add the code in Listing 2, to run the Flesch-Kincaid Reading Ease test on them.

This will render the following output.

```
Flesch-Kincaid Reading Ease:
  Henry Rollins: 77.8
  Shakespeare: 61
```

Given these scores, it can be reasonably expected that you'll need at least a 7th-grade education to read the text by Henry Rollins, but a 10th- 12th-grade school education to read the text from Shakespeare. Let's validate that assumption by calling another function: `fleschKincaidGradeLevel()`, to calculate the grade level.

To do so, add the code snippet from Listing 3 to `index.php`.

When you run it, it will output the following results, confirming my quick calculation:

```
Flesch-Kincaid Grade Level:
  5.5
  12
```

It shows that I was wrong. I both estimated too high for the piece by Henry Rollins and too generally for Shakespeare.

## One Thing I Want to Say

*I want it to be clear that I'm* **not** *inferring the information contained in one piece of text is any better nor worse than the other. Specifically, I have enormous respect for Henry Rollins and have learned a great deal from him.*

*What I'm seeking to convey here is that we're only considering how the information is framed and conveyed. If we want to debate the value of the information itself, we can do that another day.*

## Another Algorithm

Let's have a look at another test, the Spache Readability Score[7]. The score:

> *Compares words in a text to a set list of everyday words. The number of words per sentence and the percentage of unfamiliar words determine the reading age.*

Here's the formula:

```
Grade Level = (0.121 x Average sentence length)
   + (0.082 x Percentage of unique unfamiliar words)
   + 0.659
```

TextStatistics contains the method `spacheReadabilityScore`. Let's run it by adding the Listing 4 code to `index.php`.

7 *the Spache Readability Score:* http://phpa.me/spache-readability-score

**Listing 1**

```
$textHenryRollins = 'When I was young I had no sense of myself.
All I was, was a product of all the fear and humiliation I
suffered. Fear of my parents. The humiliation of teachers
calling me "garbage can" and telling me I\'d be mowing lawns
for a living. And the very real terror of my fellow students.
I was threatened and beaten up for the color of my skin and my
size. I was skinny and clumsy, and when others would tease me
I didn\'t run home crying, wondering why.';

$textShakespeare = 'Beseech you, sir, be merry; you have cause,
So have we all, of joy; for our escape
Is much beyond our loss. Our hint of woe
Is common; every day some sailor\'s wife,
The masters of some merchant and the merchant
Have just our theme of woe; but for the miracle,
I mean our preservation, few in millions
Can speak like us: then wisely, good sir, weigh
Our sorrow with our comfort.';
```

**Listing 2**

```
$txtStats = new TS\TextStatistics();

echo "Flesch-Kincaid Reading Ease: \n";
printf(
  "  Henry Rollins: %s\n",
  $txtStats->fleschKincaidReadingEase($textHenryRollins)
);
printf(
  "  Shakespeare: %s\n",
  $txtStats->fleschKincaidReadingEase($textShakespeare)
);
```

**Listing 3**

```
echo "Flesch-Kincaid Grade Level: \n";
printf(
  "  %s\n",
  $txtStats->fleschKincaidGradeLevel($textHenryRollins)
);
printf(
  "  %s\n",
  $txtStats->fleschKincaidGradeLevel($textShakespeare)
);
```

**Listing 4**

```
echo "Spache Readability Score: \n";
printf(
  "  %s\n",
  $txtStats->spacheReadabilityScore($textHenryRollins)
);
printf(
  "  %s\n",
  $txtStats->spacheReadabilityScore($textShakespeare)
);
```

Running it produces the following output:

```
Spache Readability Score:
  5
  4.1
```

Interestingly, Shakespeare has the lower grade level for this test.

## And Just One More

Let's run one more algorithm, the Spache Readability Formula[8]. But, before we do, here's how it works:

1. Count the following as 'familiar' or 'known' words:

- Words appearing on the Spache Revised Word List.
- Variants of words appearing on the Spache Revised Word List that have regular verb form endings –ing, -ed, -es.
- Plural and possessive endings of nouns from the Spache Revised Word List.
- First Names
- Single letters standing alone as words. E.g., "C is the third letter of the alphabet."

2. "Difficult Words," i.e., the words not appearing on the Spache Word List are counted only once, even if they appear later with other endings. Count the following as 'unfamiliar' or 'unknown' words:

- Words not appearing on the Spache Revised Word List (other than First Names).
- Variants of words appearing on the Spache Revised Word List that have irregular verb form endings—unless those variant forms also appear on the Spache List.
- Variants of words appearing on the Spache Revised Word List that have adverbial, comparative, or superlative endings –ly, -er, -est.

With that in mind, Listing 5 shows how we run it using the library.

Listing 5

```
1. echo "Spache Difficult Word Count: \n";
2. printf(
3.   "  %s \n",
4.   $txtStats->spacheDifficultWordCount($textHenryRollins)
5. );
6. printf(
7.   "  %s \n",
8.   $txtStats->spacheDifficultWordCount($textShakespeare)
9. );
```

When run, we'll get the following output:

```
Spache Difficult Word Count:
  23
  18
```

## What About Some More Basic Analysis?

Now that we've looked at a series of the core methods, which implement some of the larger algorithms, let's look at a series of the contained methods, which calculate more accurate text statistics. The available methods calculate things such as:

| Calculation | Method |
|---|---|
| Letter count | `letterCount()` |
| Word count | `wordCount()` |
| Sentence count | `sentenceCount()` |
| Syllable count | `totalSyllables()` |
| Average syllables per/word | `averageSyllablesPerWord()` |
| Average words per/sentence | `averageWordsPerSentence()` |
| Text length | `textLength()` |
| Percentage of words with three or more syllables | `percentageWordsWithThreeSyllables()` |

For these final examples, we'll just analyze the text from Henry Rollins (let's not be bashful, I'm totally biased). Copy the snippet from Listing 6 into `index.php`.

Listing 6

```
1. printf("Letter count: %d \n",
2.       $txtStats->letterCount($textHenryRollins));
3. printf("Word count: %d \n",
4.       $txtStats->wordCount($textHenryRollins));
5. printf("Sentence count: %d \n",
6.       $txtStats->sentenceCount($textHenryRollins));
7. printf("Total syllables: %d \n",
8.       $txtStats->totalSyllables($textHenryRollins));
9. printf("Average syllables per/word: %f \n",
10.      $txtStats->averageSyllablesPerWord($textHenryRollins));
11. printf("Average words per/sentence: %f \n",
12.      $txtStats->averageWordsPerSentence($textHenryRollins));
13. printf("Text length: %d \n",
14.      TS\Text::textLength($textHenryRollins));
15. printf("Words with three syllables: %f \n",
16.      $txtStats->percentageWordsWithThreeSyllables(
17.         $textHenryRollins
18.      )
19. );
```

8 *the Spache Readability Formula:* http://phpa.me/spache-readability-score

When you run it, you'll get the following output:

```
Letter count: 346
Word count: 85
Sentence count: 7
Total syllables: 121
Average syllables per/word: 1.375000
Average words per/sentence: 12.142857
Text length: 458
Words with three syllables: 3.409091
```

## In Conclusion

That's been a rapid run-through of the TextStatistics package by Dave Child, along with some of the algorithms for calculating text complexity, and some background on what readability scores are and how text readability works.

Perhaps extracting statistics about text isn't your thing, or you haven't considered how useful it could be, until now. If nothing else, this month's edition has given you a greater appreciation of how you write, and how that impacts the ability of others to understand what you're trying to say.

As documentation is essential to the ability of users to use the software which we write and maintain, it's important that we consider our audience's ability to understand and process that information.

You now have a wealth of information about the theory behind it, as well as a library which you can use to monitor and improve your efforts. All the best to your efforts to write excellent documentation, to have happy users, and a massive uptake of your software projects.

*Matthew Setter is an independent software developer, specializing in creating test-driven applications, and a technical writer http://www.matthewsetter.com/services/. He's also editor of Master Zend Framework, which is dedicated to helping you become a Zend Framework master? Find out more*[9].

9 http://www.masterzendframework.com/welcome-from-phparch


# Borrowed this magazine?

## Get php[architect] delivered to your doorstep or digitally every month!

Each issue of php[**architect**] magazine focuses on an important topic that PHP developers face every day.

We cover topics such as frameworks, security, ecommerce, databases, scalability, migration, API integration, devops, cloud services, business development, content management systems, and the PHP community.

php[architect]

Digital and Print+Digital
Subscriptions
Starting at $49/Year

http://phpa.me/mag_subscribe