# Derek Sivers

**Articles:**

# Memorizing a programming language using spaced repetition software

2013-01-06

I've been doing this for a year, and it's **the most helpful learning technique I've found in 14 years of computer programming**.

## Background:

I'm an intermediate programmer. I didn't go to school for it. I just learned by necessity because I started a website that just kept growing and growing, and I couldn't afford to hire a programmer, so I picked up a few books on PHP, SQL, Linux, and Apache, learned just enough to make it work, then used that little knowledge for years.

But later, when I worked along side a real programmer, I was blown away by his vocabulary! All of these commands and functions just flowing effortlessly out of his fingers. We were using the same language, but **he had memorized so much of it**, that I felt like a child next to a university professor. I really wanted to get that kind of fluency.

It made me think about how much I've **learned then immediately forgotten**, over the years. I read books or articles about some useful feature, try it once, but then I get distracted, forget about it, and go about my normal way of doing things.

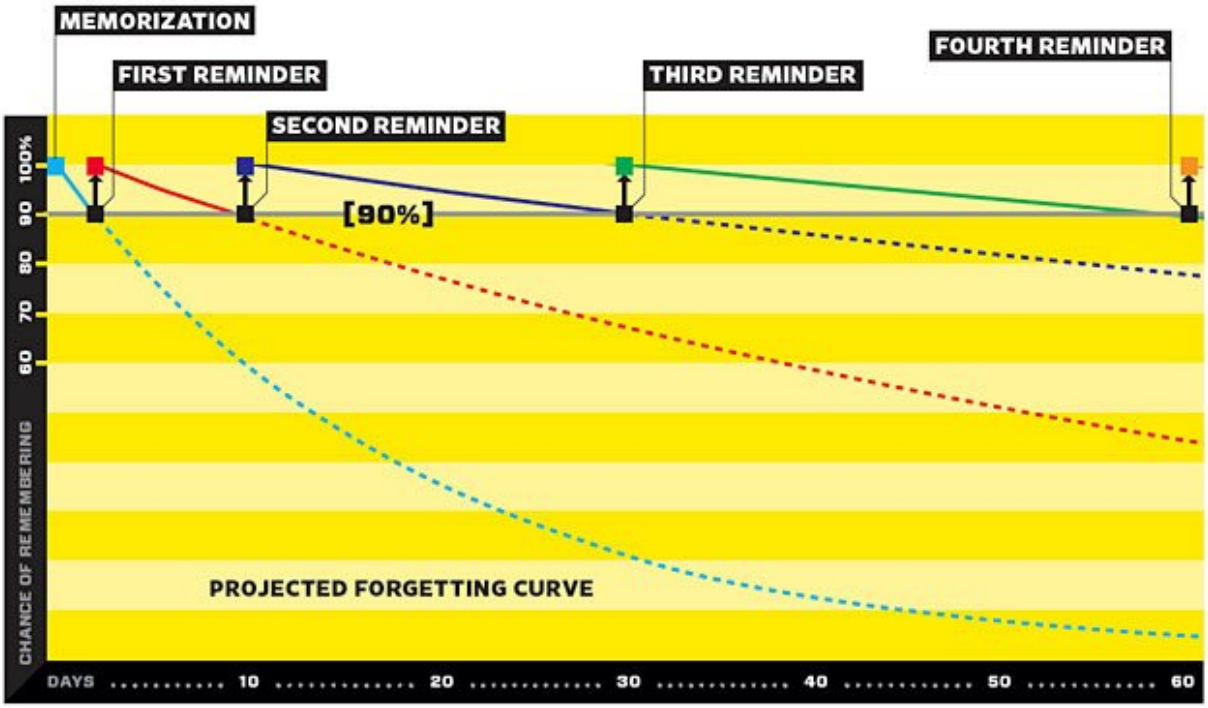
I wanted to deeply memorize the commands and techniques of the language, and not forget them, so that they stay at the forefront of my mind whenever I need them.

## Spaced Repetition:

When you hear a new fact, it's forgotten pretty quickly unless it's brought back to the forefront of your mind repeatedly.

You can do this haphazardly by immersing yourself in a language, for example, where the new words you learn will be brought up by chance occasionally.

But memory research shows that **the most effective and efficient time for a new fact to be remembered is right before you were about to forget it**.




Say if you learn a new word in a foreign language, you'd want to practice it again a few minutes after hearing it, then a few hours, then the next day, then in 2 days, then 5 days, then 10 days, 3 weeks, 6 weeks, 3 months, 8 months, etc. After a while it's basically permanently memorized with a rare reminder.

**Spaced Repetition Software does this for you**, so you can just **give it a bunch of facts you want to remember, then have it quiz you once a day, and it manages the intervals based on your feedback.** After each quiz question, if you say that one was easy, it won't be introduced for a long time, but if you were stumped, then it'll ask it again in a few minutes, until you've got it.

**Go to ankisrs.net and download Anki**. It's a free, open source, popular spaced repetition software program.

As for programming, you get where I'm going with this.

What if you could **memorize everything about the programming language** of your choice? Every command, every parameter, every function. Every solution to hundreds of the most common problems, all completely memorized at your fingertips? Imagine going through the documentation or a book, and **permanently remembering every single thing in it?**

Enough of the intro, let's get to the HOW-TO:

## First, learn!

**Flash cards are for *remembering* what you've learned.**

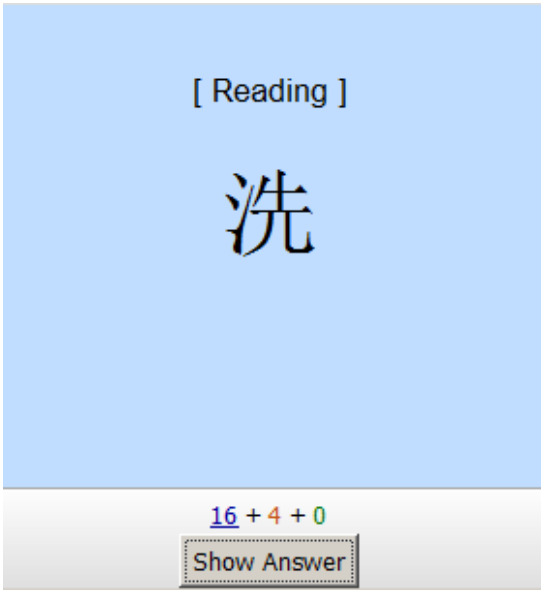
Before you create a flash card for something, you need to actually **learn and understand it.** Create the flash card after you've really understood.

(This is why it's not that helpful to look at someone else's deck. Those are just reminders.)

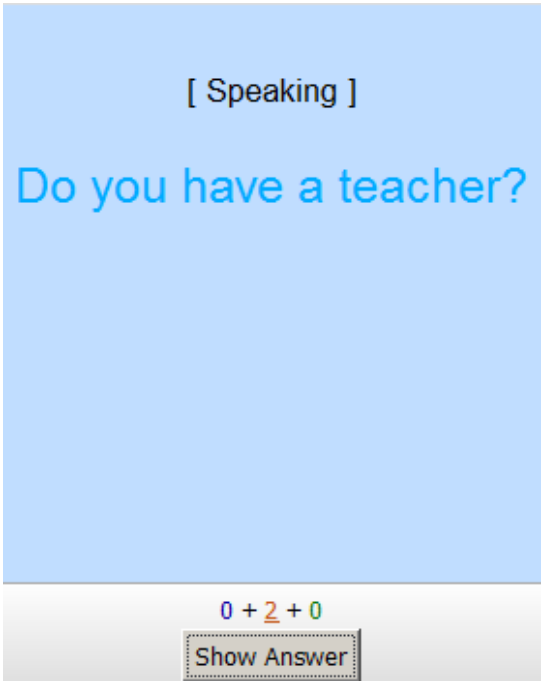
## Convert Knowledge into Small Facts:

You're going to be making a bunch of flash cards. Question on the front. Answer on the back.

If you were just using this to memorize foreign language vocabulary, then the formatting would be easy. The front would have a word or phrase, and the back would have its translation, and vice-versa.


[ Reading ]
洗
16 + 4 + 0
Show Answer


[ Reading ]
洗
xǐ
to wash
to bathe
<1m
<10m
4d
Again
Good
Easy


[ Speaking ]
Do you have a teacher?
0 + 2 + 0
Show Answer


But if you're learning anything else, you're going to have to put a little craft and creativity into making your own flash cards.

It takes some effort to read through paragraphs of stuff you want to remember, pick out the key facts, break them down into their smallest form, and turn them into questions for quizzing your future self.

Here are my best time-saving tips from a year of doing this:

## Turn prose into code

If you're reading a tutorial about programming, and come across a paragraph describing a feature.

"The add (+) operator... if only one operand is a string, the other operand is converted to a string and the result is the concatenation of the two strings."

You test it out yourself, play around with it, and understand it. So you make a flashcard to remember it.

```
var a = 5 + '5';
// what is a?

'55'
If either side of + is a string, the other is
converted to a string before adding like strings.
```

## Try to trick your future self

Sometimes you learn a “gotcha” — a common mistake or surprising feature.

“If the new Array constructor is passed a single number, it creates an empty Array with a length of that number. Any other combination of arguments creates an Array of those arguments.”

You test it out yourself, play around with it, and understand it. Then make two flash cards to try to trick your future self.

```
var a = new Array('5');
// what is a?

An array with one item, the string '5': ['5'];
```

... and then an almost-identical question ...

```
var a = new Array(5);
// what is a?

An empty array with a length of 5.
```

When the program quizzes you, it will shuffle the cards, so that hopefully your examples will intentionally catch you by surprise.

You can also try to trick yourself with more complicated examples, to keep these gotchas fresh in your mind:

```
var a = [20, 10, 5, 1];
// what is a.sort()?

[1, 10, 20, 5]
// sort treats all values as strings
```

Don’t forget to **quiz yourself on the solution, too:**

```
var a = [20, 10, 5, 1];
// sort these in numeric order

function compare(v1, v2) { return(v1 – v2); }
a.sort(compare);
```

## Save the cool tricks

If you find a cool trick you want to remember, turn it into the answer of a small challenge.

```
var albums = [
  {name: 'Beatles', title: 'White Album', price: 15},
  {name: 'Zeppelin', title: 'II', price: 7}];
// make this work:
albums.sort(by('name'));
albums.sort(by('title'));
albums.sort(by('price'));

function by(propName) {
  return function(obj1, obj2) {
    v1 = obj1[propName];
    v2 = obj2[propName];
    if (v1 < v2) { return -1; }
    else if (v1 > v2) { return 1; }
    else { return 0; }
  };
}
```

## Make the answer require multiple solutions

If there's more than one way of doing something, and you want to remember both, make your future self come up with more than one solution, so you can keep both alternatives in mind.

```
s = 'string like this'
# In Ruby, show two ways to turn it into 'String Like This'

s.split.map(&:capitalize).join(' ')
s.gsub(/\b\S/) {|x| x.upcase}
```

## Turn broad concepts into succinct examples

Say you just spent 20 minutes learning something that's more conceptual, and not as much about remembering specific functions. Sometimes all you need is one succinct example to remind yourself of the concept.

```
/(a(b)((c)d))/.match('abcd')
# What will $1, $2, $3, $4 be?
```

```
$1 = 'abcd'
$2 = 'b'
$3 = 'cd'
$4 = 'c'
```

Another example:

```
class C
  self
end
class D < C
end
d1 = D.new
# which object is self?


class D
```

## Read “20 Rules of Formulating Knowledge” by Piotr Wozniak

The best advice on this stuff is an article called “20 Rules of Formulating Knowledge” by Piotr Wozniak at supermemo.com/en/articles/20rules. So please read that one.

# Run Through it Daily

For most efficient results, turn on your spaced repetition software once a day. If you go too long without, you’ll screw up all the timings, and have to re-learn stuff you would have remembered.

**You can remember thousands of these facts in only 20 minutes a day.** I just make it a morning routine. Make a cup of boiling tea. Do my Anki. Drink my tea.

It’s fun when quizzing yourself to add a little adrenaline, and make yourself go as fast as you can.

It’s like a mental visit to the gym. A little intense 20 minutes a day is so worth it for the immediate and long-term results.

Add some new cards whenever you can, and you’ll be amazed that everything you saved stays fresh in your mind.

# Conclusion

I've been doing this for a year, and it's been a HUGE boost to my fluency. I highly recommend it, as you can tell.

If you're interested, you can also use this approach to learn all kinds of things. Read the interview with Piotr Wozniak at wired.com for more inspiration.

Feel free to email me anytime to let me know how it goes for you.

# UPDATE:

After I posted this, someone showed me the "Janki Method Refined" by Jack Kinsella, which is a great article on similar approach,and explains it even better than I did here, so **please read that one, too.**

Also I got many requests to share my Ruby and JavaScript Anki decks. I don't think it's very useful, because the whole point is that the Anki card is not where you learn. It's more of a reminder you give your future self about something you have already learned. The learning itself requires more context. But that said, if you want, **here are my JavaScript and Ruby decks** as of January 2013. You'll have to use Anki's "File → Import" function to import them.

- JavaScript Anki deck
- Ruby Anki deck

 ( « previous || next » )

# Copy & share: sivers.org/srs