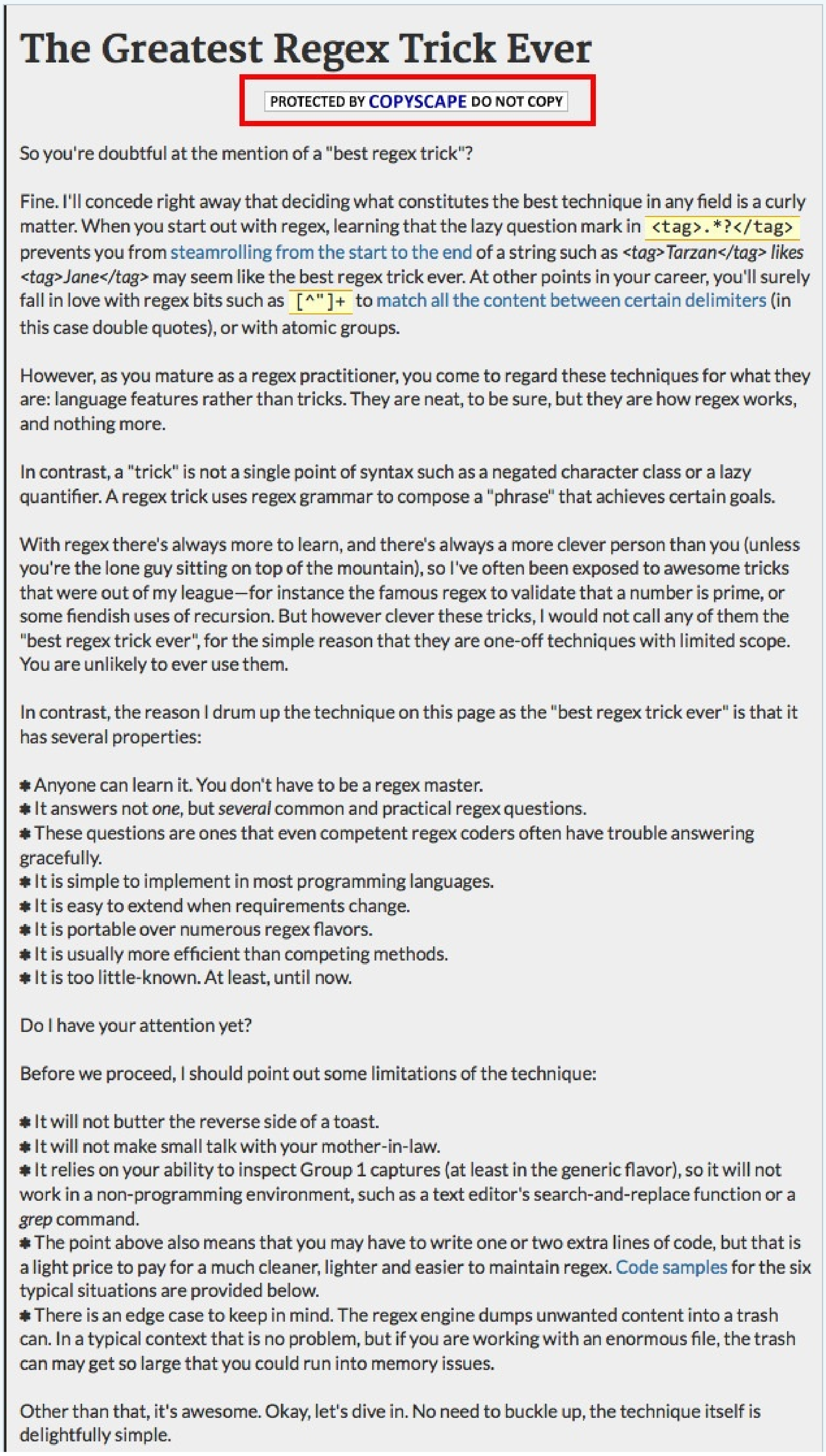
ILDER

转发微博

@网路冷眼
【The Greatest Regex Trick Ever】 http://t.cn/RyoGRRv 史上最棒的正则表达式技巧。请注意，本文很长并受Copyscape保护，不知翻译是否需要授权？强烈建议收藏！ @Linux中国 @慕课网 @左耳朵耗子



The Greatest Regex Trick Ever

PROTECTED BY COPYSCAPE DO NOT COPY

So you're doubtful at the mention of a "best regex trick"?

Fine. I'll concede right away that deciding what constitutes the best technique in any field is a curly matter. When you start out with regex, learning that the lazy question mark in `<tag>.*?</tag>` prevents you from steamrolling from the start to the end of a string such as *<tag>Tarzan</tag> likes <tag>Jane</tag>* may seem like the best regex trick ever. At other points in your career, you'll surely fall in love with regex bits such as `[^"]+` to match all the content between certain delimiters (in this case double quotes), or with atomic groups.

However, as you mature as a regex practitioner, you come to regard these techniques for what they are: language features rather than tricks. They are neat, to be sure, but they are how regex works, and nothing more.

In contrast, a "trick" is not a single point of syntax such as a negated character class or a lazy quantifier. A regex trick uses regex grammar to compose a "phrase" that achieves certain goals.

With regex there's always more to learn, and there's always a more clever person than you (unless you're the lone guy sitting on top of the mountain), so I've often been exposed to awesome tricks that were out of my league—for instance the famous regex to validate that a number is prime, or some fiendish uses of recursion. But however clever these tricks, I would not call any of them the "best regex trick ever", for the simple reason that they are one-off techniques with limited scope. You are unlikely to ever use them.

In contrast, the reason I drum up the technique on this page as the "best regex trick ever" is that it has several properties:

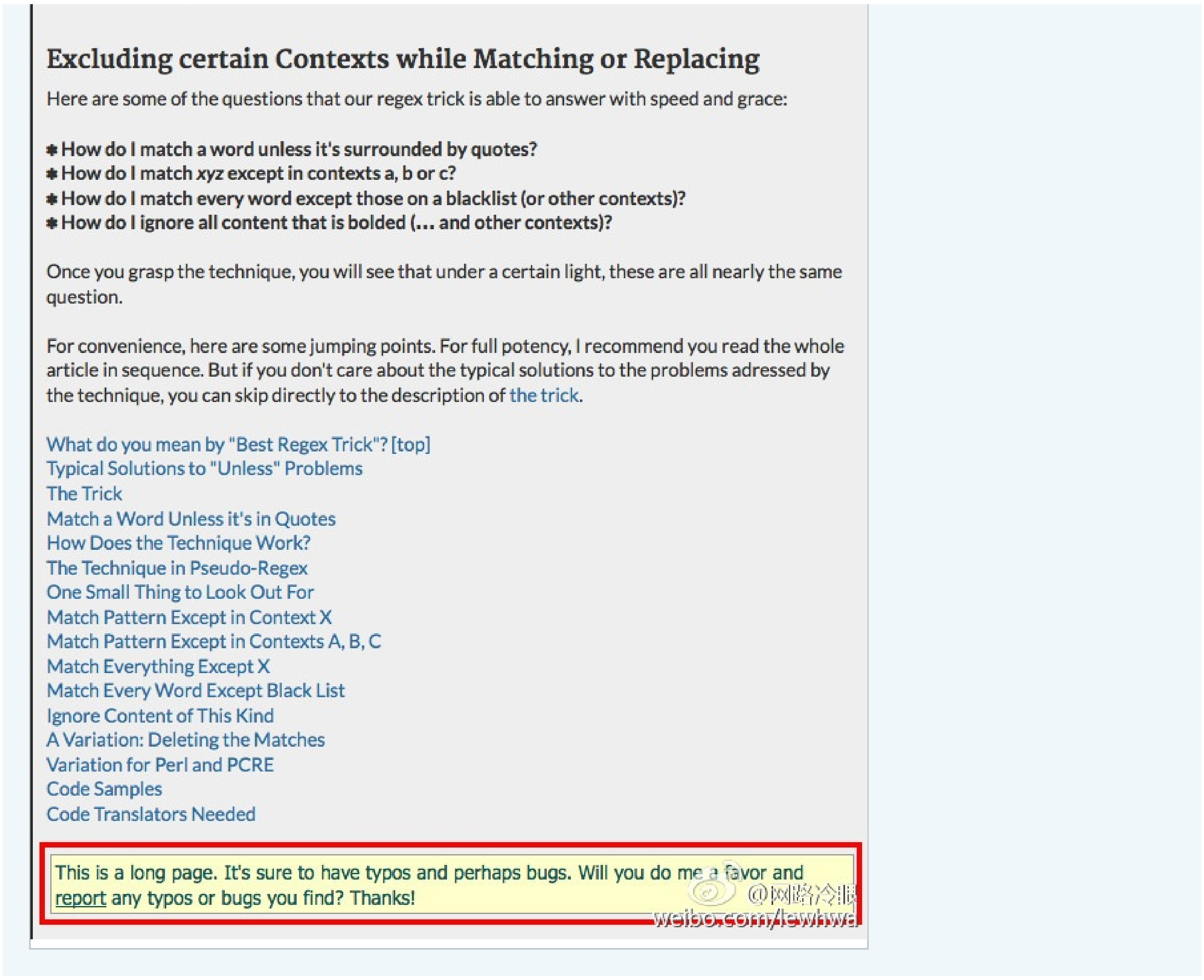
- Anyone can learn it. You don't have to be a regex master.
- It answers not *one*, but *several* common and practical regex questions.
- These questions are ones that even competent regex coders often have trouble answering gracefully.
- It is simple to implement in most programming languages.
- It is easy to extend when requirements change.
- It is portable over numerous regex flavors.
- It is usually more efficient than competing methods.
- It is too little-known. At least, until now.

Do I have your attention yet?

Before we proceed, I should point out some limitations of the technique:

- It will not butter the reverse side of a toast.
- It will not make small talk with your mother-in-law.
- It relies on your ability to inspect Group 1 captures (at least in the generic flavor), so it will not work in a non-programming environment, such as a text editor's search-and-replace function or a *grep* command.
- The point above also means that you may have to write one or two extra lines of code, but that is a light price to pay for a much cleaner, lighter and easier to maintain regex. Code samples for the six typical situations are provided below.
- There is an edge case to keep in mind. The regex engine dumps unwanted content into a trash can. In a typical context that is no problem, but if you are working with an enormous file, the trash can may get so large that you could run into memory issues.

Other than that, it's awesome. Okay, let's dive in. No need to buckle up, the technique itself is delightfully simple.


Excluding certain Contexts while Matching or Replacing

Here are some of the questions that our regex trick is able to answer with speed and grace:

- **How do I match a word unless it's surrounded by quotes?**
- **How do I match *xyz* except in contexts a, b or c?**
- **How do I match every word except those on a blacklist (or other contexts)?**
- **How do I ignore all content that is bolded (… and other contexts)?**

Once you grasp the technique, you will see that under a certain light, these are all nearly the same question.

For convenience, here are some jumping points. For full potency, I recommend you read the whole article in sequence. But if you don't care about the typical solutions to the problems adressed by the technique, you can skip directly to the description of the trick.

This is a long page. It's sure to have typos and perhaps bugs. Will you do me a favor and report any typos or bugs you find? Thanks!

2015年09月29日 12:58 来自 荣耀畅玩4X·快科技