Mark Needham

Thoughts on Software Development

Javascript: Confusing ‘call’ and ‘apply’

4 Comments

I wrote a couple of weeks ago about using the ‘call’ and ‘apply’ functions in Javascript when passing functions around and while working on our IE6 specific code I realised that I’d got them mixed up.

We were writing some code to override one of our functions so that we could call the original function and then do something else after that.

The code was roughly like this:

```
Foo = {
    bar : function(duck) {
      console.log("bar " + duck.quack());
    }
};
```

The code that I originally wrote to capture the original function, call it and then do the additional behaviour was like this:

```
(function() {
  var originalBar = Foo.bar;

   Foo.bar = function(duck) {
        originalBar.call(this, arguments);
        console.log("new bar");
   };
})();
```

When we call the function:

```
Foo.bar({ quack : function() { return "quacking" } });
```

We get the following error:

```
TypeError: duck.quack is not a function
```

‘arguments’ is a local variable in any Javascript function which contains all the arguments passed to the function stored in an array type structure.

However, I had forgotten that when using the ‘call’ function we need to pass the full list of parameters individually rather than as an array so in this case we would need to pass ‘duck’ in specifically:

```
(function() {
  var originalBar = Foo.bar;

   Foo.bar = function(duck) {
        originalBar.call(this, duck);
        console.log("new bar");
    };
})();
```

Now when we run the function we get the expected behaviour:

```
Foo.bar({ quack : function() { return "quacking" } });
```

```
bar quacking
new bar
```

This is where apply comes in handy because apply allows us to pass in ‘arguments’ as the second parameter and it will send all the arguments of the function that we’re inside to the function that we’re calling which is exactly what we want in this case.

Using ‘apply’ we would end up with the following code:

```
(function() {
  var originalBar = Foo.bar;

   Foo.bar = function(duck) {
        originalBar.apply(this, arguments);
        console.log("new bar");
   };
})();
```

In this case the function only takes in one argument so there’s not much noticeable improvement in the code but when a function takes multiple arguments then using ‘apply’ is certainly a cleaner approach.

Be Sociable, Share!

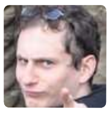
Written by Mark Needham

February 28th, 2010 at 1:45 am

Posted in Javascript

Tagged with Javascript

« Javascript: Isolating browser specific code
A reminder about context switching »



marek • 3 years ago

Thanks for the article. I elaborated bit more on my blog JavaScript call() and apply(). Not going to copy+paste my code here but if you looking for more information and examples, give it a go and just read the post.

• Reply • Share ›

Reducir Cintura • 5 years ago

Encontré tu blog por pura casualidad y me gustó mucho el contenido que publicas. Ya llevo un rato navegando por aquí. Agregaré la dirección a mi lector de noticias. Si tienes minutos libres, comenta en mi blog. Abrazo!

• Reply • Share ›

qFox • 6 years ago

It might be better to convert "arguments" to an Array first, because it is not an Array when supplied by JS. Array.prototype.slice.call(arguments) will do that trick.

Note also that arguments is a very special object that can do magic voodoo on your named variables. It's mere usage will also slow down the function dramatically on most browsers.

• Reply • Share ›

Gonzalo Casas • 6 years ago

If your goal is to alter the invocation of an external function, you could try the jquery AOP plugin: http://code.google.com/p/jquer...

Bye!

• Reply • Share ›

ALSO ON **MARK NEEDHAM** WHAT'S THIS?

Python NLTK/Neo4j: Analysing the transcripts of How I Met Your Mother

1 comment • 8 months ago

Avat salty-horse — The function count_words isn't required, since collections.Counter already knows how to count a list of words.

Spark: Write to CSV file

3 comments • 9 months ago

Avat Mark Needham — Knight Fu Ah good call, that's a much better solution! Thanks for sharing :)

The Willpower Instinct: Reducing time spent mindlessly scrolling for things to

1 comment • 3 months ago

Avat Alex — I haven't read the book but I think I'm going to check it out. I do this all the time too, flitting between my handful of

R/dplyr: Extracting data frame column value for filtering with %in%

1 comment • 6 months ago

Avat Antonios K — You can also do: highScoringPeople = df %>% filter(score > 3) %>% select(userId) df %>% filter(userId

Subscribe   Add Disqus to your site   Privacy

Twitter
Email
435 readers BY FEEDBURNER

Subscribe by email

Search for: Search

Archives

Archives Select Month

Categories

- .NET (67)
- Agile (61)
  - Distributed Agile (14)
- Algorithms (12)
- Android (11)
- Book Club (24)
- Books (36)
- Build (25)
- Coding (113)
  - Code Katas (2)
  - Incremental Refactoring (9)
- Coding Dojo (23)
- Communication (21)
- Conferences (13)
  - QCon (4)
  - XP 2011 (5)
  - XP Day (4)
- Data Science (10)
- Databases (148)
  - CouchDB (4)
  - Mark Logic (2)
  - neo4j (141)
- Deliberate Practice (2)
- DevOps (21)
- Domain Driven Design (12)
- Feedback (9)
- Graph Processing (2)
- Hibernate (7)
- Hiring (3)
- iPad (2)
- jQuery (8)
- Languages (443)
  - Clojure (44)
  - F# (54)
  - Haskell (46)

- J (1)
- Java (42)
- Javascript (20)
- Objective C (4)
- Python (50)
- R (90)
- Ruby (63)
- Scala (34)

- Lean (3)
- Learning (32)
- Machine Learning (16)
- Messaging (2)
- Micro Services (5)
- Networking (6)
- OOP (7)
- Organisational Patterns (1)
- Pair Programming (28)
- Product Development (1)
- QTB (5)
- Ranking Systems (4)
- Reading Code (7)
- Scripting (40)
  - Batch Scripting (4)
  - Shell Scripting (36)
- SICP (1)
- Software Development (264)
- Spark (7)
- Systems Thinking (7)
- Testing (66)
- ThoughtWorks University (21)
- Version Control (17)