# jQueryはもっと効率よく書ける！ 押さえておきたいトラバースメソッドのすべて

2017/03/22 **Baljeet Rathi**



sitepoint

**多彩なセレクターはjQueryの特徴の1つですが、DOMトラバースメソッドを使って絞り込む方法を身につければ、もっと効率よくコードが書けます。**

本記事ではjQueryで使えるDOMトラバースメソッドを説明していくとともに、ページのある要素との関係を基に別の要素を簡単に選択するさまざまな方法を解説します。

## 要素の絞り込み

はじめに、選択肢を具体的に絞り込んでいく方法を説明します。たくさんの条件、たとえば、ほかの要素との関連性や特定のクラスに属するか否かで要素を絞り込めます。たいていは当初より絞り込んだ要素を選択できるはずです。

絞り込みに使うフィルターメソッドはこちらです。

- eq：選択された要素の中から指定した位置（インデックス値）にある要素を選び出す。インデックスは0から始まるので、先頭の要素の選択には$("selector").eq(0)を使う。バージョン1.4以降では負の整数を指定することで前からでなく後ろから選択する方法も使える
- first、last：firstメソッドは条件に合致した要素のなかから単に最初の要素を返し、lastは最後の要素を返す。どちらのメソッドも引数はない
- slice：特定のインデックスの範囲にある複数要素を一度に抜き出したいときはslice()を使う。このメソッドは2つの引数を取る。抜き出したい部分の開始インデックスと、その部分の終わりを示す終端インデックス。2つ目の引数は省略でき、省略した場合は最初の引数startよりもインデックスの大きいものがすべて選択される

- filter：セレクターに合致した要素だけを抜き出すか、あるいは引数としてセットした関数の実行によって要素を抜き出す。次のコードはセレクターを使ったこのメソッドの1例となる。

```
    $("li").filter(":even").css( "font-weight", "bold" );
```

関数を使う方法でも同じ要素を取り出せる。

```
    $("li")
.filter(function( index ) {
  return index % 2 === 0;
})
.css( "font-weight", "bold" );
```

以下のように、関数を用いるともっと複雑な条件で選択できる。

```
    .filter(function( index ) {
 return $( "span", this ).length >= 2;
})
```

次のようにすると、最低でも2つ以上のspanタグをもつ要素だけが選択される

- map：関数を通じて現在選択されている要素を渡すことで、最終的に新たなjQueryオブジェクト（要素の配列）が生成されて返される。要素の配列を持つjQueryのオブジェクトを返すので、通常の配列と同じようにgetメソッドが使える


## DOMを横断して要素を選択する

いろいろな要素にアクセスするためのセレクターは分かっているけども親要素にアクセスしたい、という状況を考えてください。加えて、親要素には特定のクラス名も無く、共通の特なるタグも無いので特定が難しいとします。現在アクセスしている要素の親である、とい

得できる

- find：一致する要素の中からセレクタまたは要素で絞り込んだ子孫をすべて取得する。この場合セレクタとしてfind()に渡す引数は必須。もしすべての子孫を取得したい場合はセレクタとして引数にワイルドカード('*')を指定する。
  デモを見ると分かるように、children()メソッドでは直接の子要素だけに下線を引いているがfind()メソッドではセレクタに合致するすべての子孫要素に背景色を付けている。
  デモはこちら
- parent：現在の各要素の親要素を取得する。親要素はセレクタで絞り込める
- parents：現在の各要素のすべての祖先を取得する。parent()とparents()との違いは、前者はDOMツリーの1階層上の要素だけを取得するのに対し、後者はドキュメントルートまでさかのぼってすべての祖先を取得する
- closest：現在の要素からDOMツリーを上にたどって、与えられたセレクタに最初に合致した要素を取得する。parents()メソッドとclosest()メソッドとの間には大きな違いがある。parents()が要素の親要素から探索開始するのに対して、closest()では現在のその要素自身から開始される。また別の違いとしてclosest()では一致するものを見つけるまでDOMツリーをたどるが、parents()では一致する要素を見つけたあともドキュメントルート要素まですべての祖先を探索する。
  デモで、使用されているこの2行のコードを確認できる。

```
    $("i").closest("span").css("background", "yellow");
$("b").parent().css("color","blue");
```

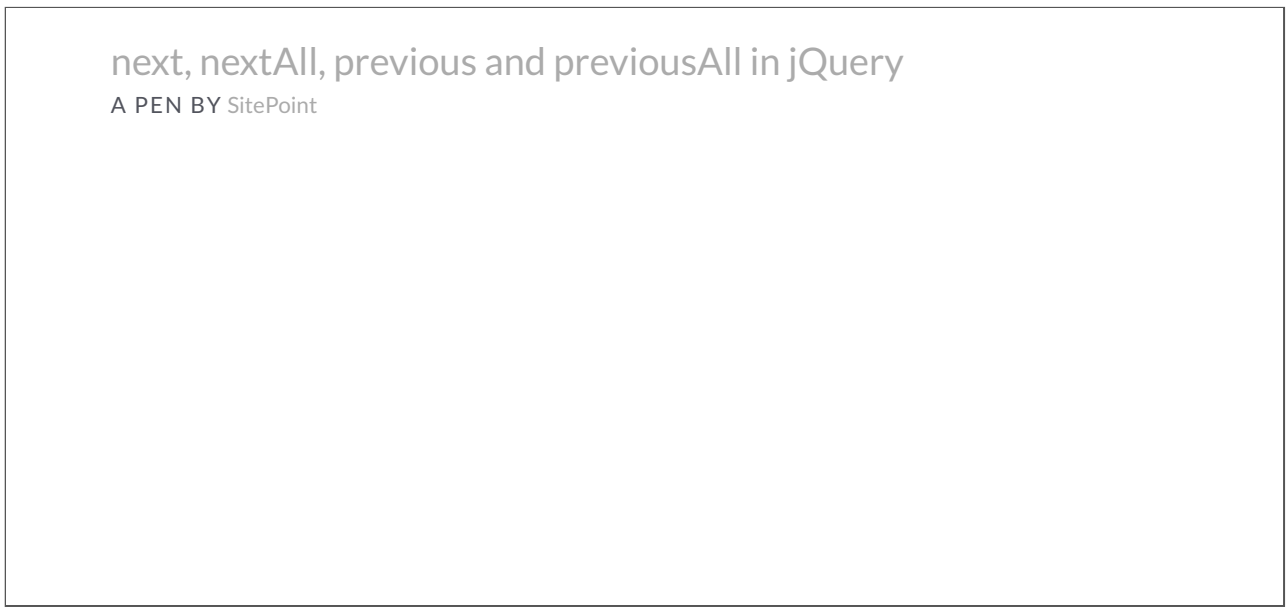
  デモの最後の段落にあるイタリック体の単語にはタグを持つ祖先がない。よって背景色は変更されない。
  同じく2行目では、太字の単語をはさむタグの色が変わっている。最後の段落は段落全体が<b>タグの親要素なので、すべてが青色に変わる。
  デモはこちら
- siblings：合致した各要素の、兄弟要素を取得する。また兄弟要素のうちセレクタに合致したものだけ取得するために引数としてセレクタをセットできる
- prev：各要素の直前の兄弟要素を取得する。もしセレクタをセットした場合はセレクタに合致した場合のみその要素を取得する
- prevAll：各要素の前の兄弟要素すべてを取得する。ほかのメソッドと同じくセレク

兄弟要素を絞り込める。

デモで、使用されているこの2行のコードを確認できる。

```
    $("selector").next(".nextall").css("color","orange");
$("selector").nextAll(".nextall").css("color","red");
```

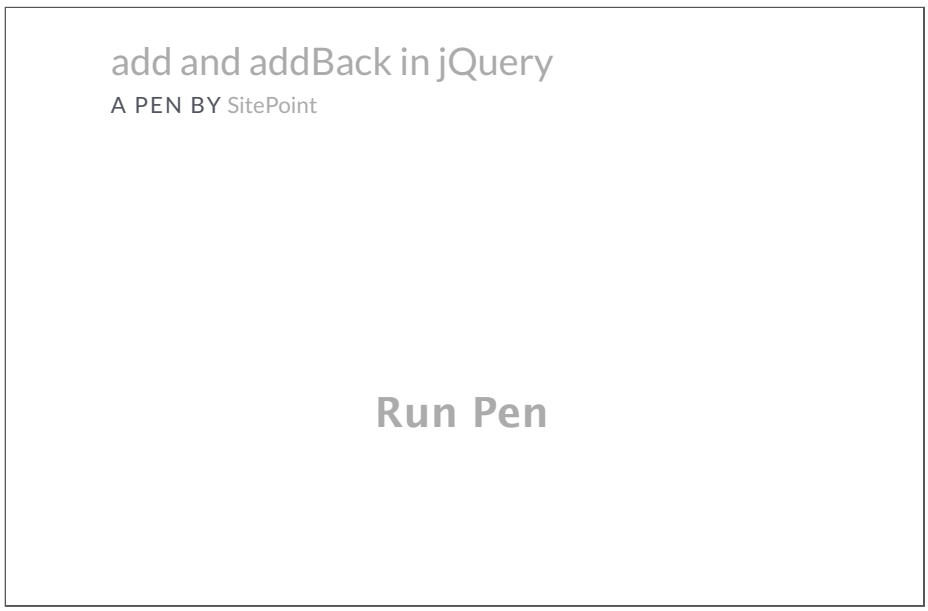
デモではリストのどの項目もオレンジ色になっていないのは、対象要素の直後の兄弟要素がnextallクラスではないため。
もう一度確認すると、next()とprev()にセレクタをセットする場合、どちらのメソッドもセレクタに合致する最初の要素を見つけるまで後ろまたは前にある兄弟要素をたどるわけではない。これらのメソッドはあくまで直前・直後の兄弟要素を参照するだけであり、もしそれがセレクタに合致しなければ空のjQueryオブジェクトが返る



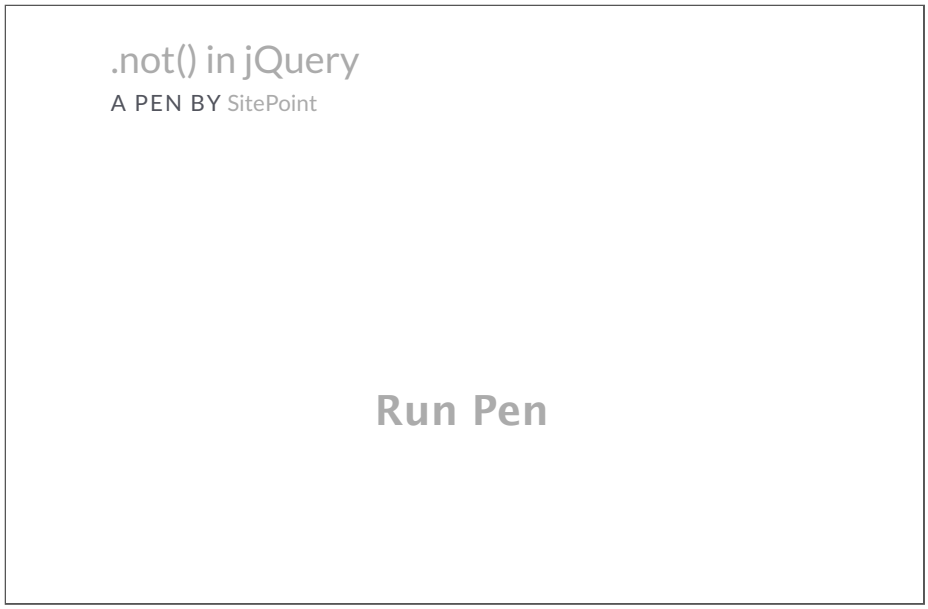

## DOMトラバースに関係するほかのメソッド

DOMトラバースを使うときには、元の要素とは関係しない要素を追加で選択したり、あるいは再び元の要素を選択し直したりする必要が生じるかもしれません。jQueryにはこれに対応した関数があります。

- addBack：jQueryには要素に加えた変更を記録する内部スタックがある。このようなトラバースメソッドを呼んでも、内部スタックに変更を加えた要素が追加される。もし以前の要素と新たな要素の両方にアクセスしたいならaddBackメソッドを使用する。
  addBackのデモの最初のケースでは、段落を選択してからchildren()を呼んでいる。2つ目のケースではchildren()を呼んだあとさらにaddBack()を呼んでいる。addBack()によって2つ目の段落が選択範囲として追加され、ここに赤い境界線を加えている




- end：直近の絞り込み操作を終了させ、要素選択をそれ以前の状態に戻す。たとえば、現在選択している要素に関連した要素を操作したあと、いったん元に戻して別の要素を操作するような場合に便利。
  デモでは最初に.find("b")メソッドですべての<b>タグを選択していったん緑色に変更したのち.end()を呼んでいる。これにより前の状態に戻ってから、改めて今度はすべての<i>タグを選択している。
  デモはこちら
- contents：現在のすべての要素のテキストやコメントノードを含むすべての子要素を取得したいときにはcontentsメソッドが使える。自分のWebページと同一ドメインにある<iframe>からコンテンツを取得するのにも使える
- not：もしも多数の要素のうちセレクタと「合致しない」要素だけを抜き出したいときには、not()が使える。jQueryのバージョン1.4以降ではこのメソッドの引数として関セットすることで特定の条件を指定できる。現在の要素のうち、この条件に合致す




## 最後に

説明してきたようなjQueryのメソッドにより、1組の要素からまた別の要素へと簡単に遷移できます。とてもよく似ているメソッドもあるので注意をしたほうが良いでしょう。parents()とclosest()との違いや、next("selector")とnextAll("selector").eq(0)との違いを知っているだけでも何時間か節約できます。

※本記事はJoan Yin、Chris Perryが査読を担当しています。最高のコンテンツに仕上げるために尽力してくれたSitePointの査読担当者のみなさんに感謝します。

（原文：A Comprehensive Look at jQuery DOM Traversal）

［翻訳：西尾健史／編集：Livit］

## Baljeet Rathi

インドを拠点に活動するライター、Web開発者です。フルスタック開発者ですが、フロントエンドに関係したものが大好き。現在まで5年以上Web開発に携わっています。

jQuery

WPJに戻る