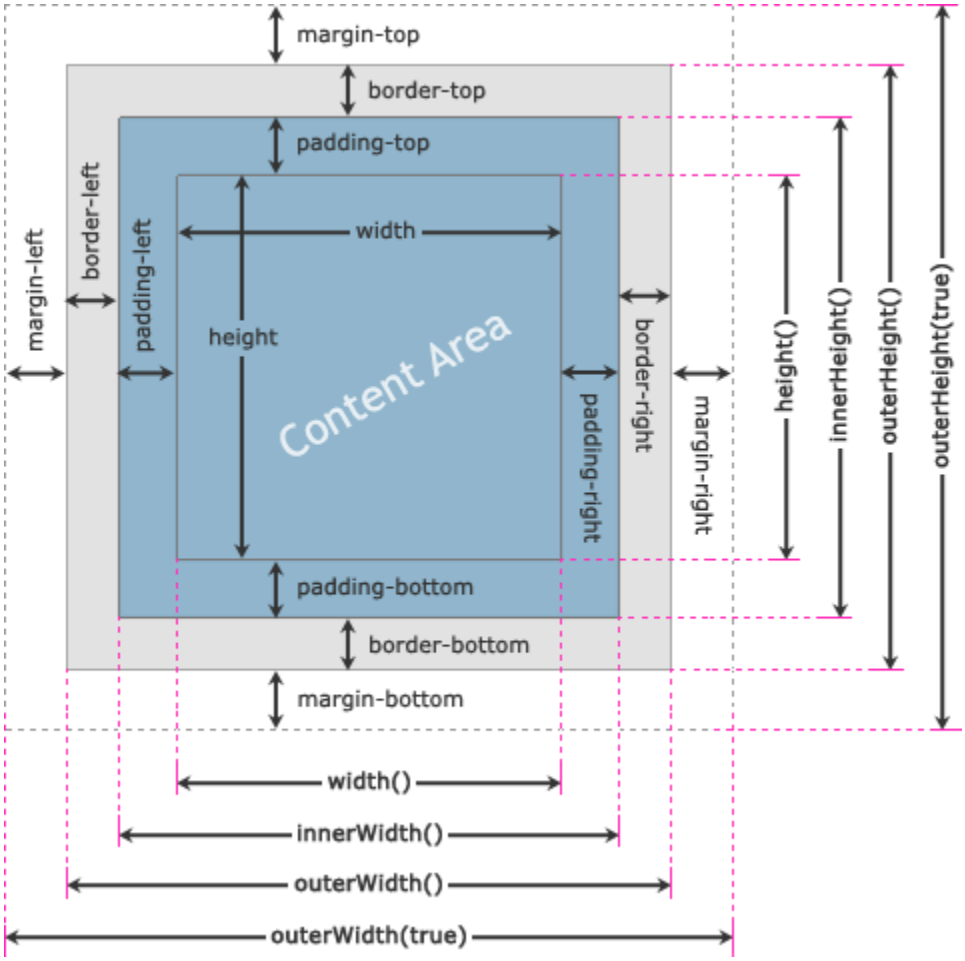
jQuery Dimensions

Understanding the jQuery Dimensions

jQuery provides several methods, such as `height()`, `innerHeight()`, `outerHeight()`, `width()`, `innerWidth()` and `outerWidth()` to get and set the CSS dimensions for the elements. Check out the following illustration to understand how these methods are calculating the dimensions of an element's box.



---

jQuery `width()` and `height()` Methods

The jQuery `width()` and `height()` methods get or set the `width` and the `height` of the element respectively. This width and height doesn't include `padding`, `border` and `margin` on the element. The following example will return the width and height of a `<div>` element.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        var divWidth = $("#box").width();
        var divHeight = $("#box").height();
        $("#result").html("Width: " + divWidth + ", " +
"Height: " + divHeight);
    });
});
</script>
```

Similarly, you can set the width and height of the element by including the value as a parameter within the `width()` and `height()` method. The value can be either a string (number and unit e.g. 100px, 20em, etc.) or a number. The following example will set the width of a `<div>` element to 400 pixels and height to 300 pixels respectively.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").width(400).height(300);
    });
});
</script>
```

Note: Use the jQuery `width()` or `height()` method if you want to use an element's width or height in a mathematical calculation, since it returns the `width` and `height` property value as a unit-less pixel value (e.g. 400). Whereas, the `css("width")` or `css("height")` methods returns value with units (e.g. 400px).

---

jQuery `innerWidth()` and `innerHeight()` Methods

The jQuery `innerWidth()` and `innerHeight()` methods get or set the *inner width* and the *inner height* of the element respectively. This inner width and height includes the `padding` but excludes `border` and `margin` on the element. The following example will return the inner width and height of a `<div>` element on the click of a button.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        var divWidth = $("#box").innerWidth();
        var divHeight = $("#box").innerHeight();
        $("#result").html("Inner Width: " + divWidth + ", "
+ "Inner Height: " + divHeight);
    });
});
</script>
```

Similarly, you can set the element's inner width and height by passing the value as a parameter to the `innerWidth()` and `innerHeight()` method. These methods only alter the width or height of the element's content area to match the specified value.

For example, if the current width of the element is 300 pixels and the sum of the left and right padding is equal to 50 pixels than the new width of the element after setting the inner width to 400 pixels is 350 pixels i.e. `New Width = Inner Width - Horizontal Padding`. Similarly, you can estimate the change in height while setting the inner height.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").innerWidth(400).innerHeight(300);
    });
});
</script>
```

jQuery `outerWidth()` and `outerHeight()` Methods

The jQuery `outerWidth()` and `outerHeight()` methods get or set the *outer width* and the *outer height* of the element respectively. This outer width and height includes `padding` and `border` but excludes the `margin` on the element. The following example will return the outer width and height of a `<div>` element on the click of a button.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        var divWidth = $("#box").outerWidth();
        var divHeight = $("#box").outerHeight();
        $("#result").html("Outer Width: " + divWidth + ", "
+ "Outer Height: " + divHeight);
    });
});
</script>
```

You can also get the outer width and height that includes `padding` and `border` as well as the `margin` of the element. For that just specify the `true` parameter for the outer width methods, like `outerWidth(true)` and `outerHeight(true)`.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        var divWidth = $("#box").outerWidth(true);
        var divHeight = $("#box").outerHeight(true);
        $("#result").html("Outer Width: " + divWidth + ", "
+ "Outer Height: " + divHeight);
    });
});
</script>
```

Similarly, you can set the element's outer width and height by passing the value as a parameter to the `outerWidth()` and `outerHeight()` methods.

These methods only alter the width or height of the element's content area to match the specified value, like the `innerWidth()` and `innerHeight()` methods.

For example, if the current width of the element is 300 pixels, and the sum of the left and right padding is equal to 50 pixels, and the sum of the width of the left and right border is 20 pixels than the new width of the element after setting the outer width to 400 pixels is 330 pixels i.e. `New Width = Outer Width - (Horizontal Padding + Horizontal Border)`. Similarly, you can estimate the change in height while setting the outer height.

Example

Try this code »

```
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").outerWidth(400).outerHeight(300);
    });
});
</script>
```

jQuery Traversing

In this tutorial you will learn how to traverse through HTML DOM using jQuery.

What is Traversing

The jQuery selectors we've seen so far only allow us to select the elements down the DOM tree. But there are many occasions when you need to select a parent or ancestor element; that is where jQuery's DOM traversal methods come into play. With these traversal methods, we can go up, down, and all around the DOM tree very easily.

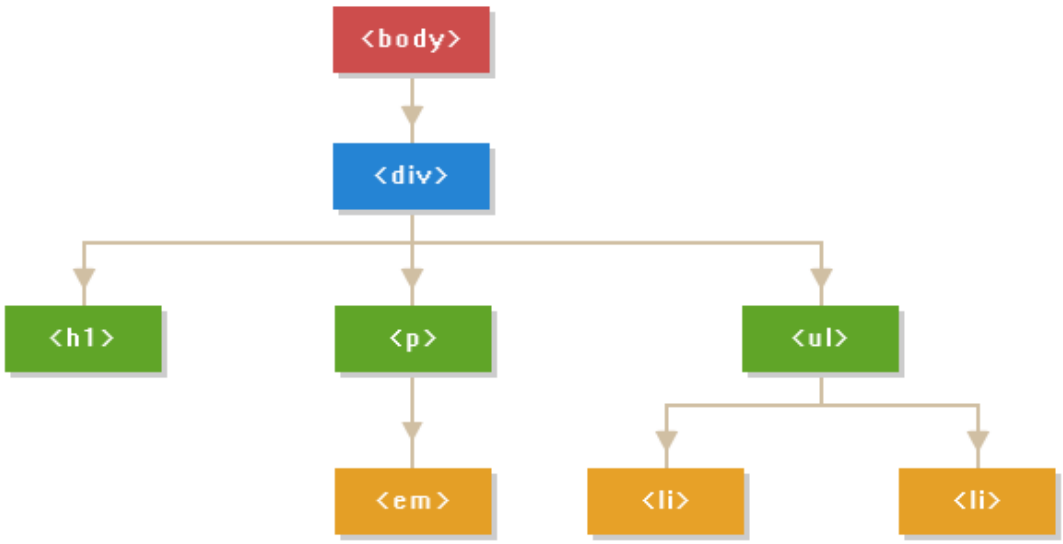
DOM traversing is one of the prominent features of the jQuery. To make the most it you need to understand the relationships between the elements in a DOM tree.

Example

Try this code »

```
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
```

The HTML code in the example above can be represented by the following DOM tree:



The above diagram showing the parent/child relationships between the elements:

- The `<body>` element is the ***parent*** of the `<div>` element, and an ***ancestor*** of everything inside of it. The enclosed `<div>` element is the ***parent*** of `<h1>`, `<p>` and `<ul>` elements, and a ***child*** of the `<body>` element.
- The elements `<h1>`, `<p>` and `<ul>` are ***siblings***, since they share the same parent.
- The `<h1>` element is a ***child*** of the `<div>` element and a ***descendant*** of the `<body>` element. This element does not have any children.
- The `<p>` element is the ***parent*** of `<em>` element, ***child*** of the `<div>` element and a ***descendant*** of the `<body>` element. The containing `<em>` element is a ***child*** of this `<p>` element and a ***descendant*** of the `<div>` and `<body>` element.
- Similarly, the `<ul>` element is the ***parent*** of the `<li>` elements, ***child*** of the `<div>` element and a ***descendant*** of the `<body>` element. The containing `<li>` elements are the ***child*** of this `<ul>` element and a ***descendant*** of the `<div>` and `<body>` element. Also, both the `<li>` elements are ***siblings***.

Note: In logical relationships, an ancestor is a parent, grandparent, great-grandparent, and so on. A descendant is a child, grandchild, great-

grandchild, and so on. Sibling elements are those which share the same parent.

Traversing the DOM Tree

Now that you have understood the logical relationships between the elements in a DOM tree. In the upcoming chapters you will learn how to perform various traversing operations such as traversing up, down and sideways the DOM tree using the jQuery.

In the next chapter you will learn how to select upper elements in a DOM tree.

jQuery Traversing Ancestors

In this tutorial you will learn how to traversing up the DOM tree using jQuery.

Traversing Up the DOM Tree

In logical relationships an ancestor is a parent, grandparent, great-grandparent, and so on.

jQuery provides the useful methods such as `parent()`, `parents()` and `parentsUntil()` that you can use to traverse up in the DOM tree either single or multiple levels to easily get the parent or other ancestors of an element in the hierarchy.

jQuery `parent()` Method

The jQuery `parent()` method is used to get the direct parent of the selected element.

The following example will highlight the direct parent element of the `<li>` which is `<ul>` by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery parent() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
```

```
    $("li").parent().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

---

jQuery `parents()` Method

The jQuery `parents()` method is used to get the ancestors of the selected element.

The following example will add a border around all the ancestor elements of the `<li>` which are `<ul>`, `<div>`, `<body>` and the `<html>` elements.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery parents() Demo</title>
<style>
    *{
        margin: 10px;
    }
    .frame{
        border: 2px solid green;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
```

```
<script>
$(document).ready(function(){
    $("li").parents().addClass("frame");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

You can optionally include one or more selector as a parameter within the `parents()` method to filter your search for the ancestors. The following example will apply the border around all the ancestors of the `<li>` that are `<div>` elements.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery parents() Demo</title>
<style>
    *{
        margin: 10px;
    }
    .frame{
        border: 2px solid green;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("li").parents("div").addClass("frame");
});
</script>
</head>
```

```
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

---

jQuery `parentsUntil()` Method

The jQuery `parentsUntil()` method is used to get all the ancestors up to but not including the element matched by the selector. In simple words we can say it returns all ancestor elements between two given elements in a DOM hierarchy.

The following example will add a border around all the ancestor elements of the `<li>` excluding `<html>` element i.e. add a border to `<ul>`, `<div>` and `<body>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery parentsUntil() Demo</title>
<style>
    *{
        margin: 10px;
    }
    .frame{
        border: 2px solid green;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
```

```
$(document).ready(function(){
    $("li").parentsUntil("html").addClass("frame");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

jQuery Traversing Descendants

In this tutorial you will learn how to traversing down the DOM tree using jQuery.

Traversing Down the DOM Tree

In logical relationships a descendant is a child, grandchild, great-grandchild, and so on.

jQuery provides the useful methods such as `children()` and `find()` that you can use to traverse down in the DOM tree either single or multiple levels to easily find or get the child or other descendants of an element in the hierarchy.

jQuery `children()` Method

The jQuery `children()` method is used to get the direct children of the selected element.

The following example will highlight the direct children of the `<ul>` element which is `<li>` by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery children() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
```

```
    $("ul").children().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

---

jQuery `find()` Method

The jQuery `find()` method is used to get the descendant elements of the selected element.

The `find()` and `children()` methods are similar, except that the `find()` method search through multiple levels down the DOM tree to the last descendant, whereas the `children()` method only search a single level down the DOM tree. The following example will add a border around all the `<li>` elements that are descendants of the `<div>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery find() Demo</title>
<style>
    *{
        margin: 10px;
    }
    .frame{
        border: 2px solid green;
    }
```

```
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("div").find("li").addClass("frame");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

However, if you want to get all the descendant elements you can use the universal selector.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery find() Demo</title>
<style>
    *{
        margin: 10px;
    }
    .frame{
        border: 2px solid green;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("div").find("*").addClass("frame");
});
</script>
</head>
```

```
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

jQuery Traversing Siblings

In this tutorial you'll learn how to traverse sideways in a DOM tree using jQuery.

Traversing Sideways in DOM Tree

In logical relationships siblings are those elements that share the same parent.

jQuery provides several methods such as `siblings()`, `next()`, `nextAll()`, `nextUntil()`, `prev()`, `prevAll()` and `prevUntil()` t hat you can use to traverse sideways in the DOM tree.

jQuery `siblings()` Method

The jQuery `siblings()` method is used to get the sibling elements of the selected element.

The following example will highlight the siblings of the `<p>` element which are `<h1>` and `<ul>` by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery siblings() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").siblings().addClass("highlight");
});
```

```
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

You can optionally include one or more selector as a parameter within the `siblings()` method to filter your search for the siblings. The following example will only apply the border around the siblings of the `<p>` that are `<ul>` elements.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery siblings() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").siblings("ul").addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
```

```
        </ul>
    </div>
</body>
</html>
```

---

jQuery `next()` Method

The jQuery `next()` method is used to get the immediately following sibling i.e. the next sibling element of the selected element. The following example will highlight the next sibling of the `<p>` element which is the `<ul>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery next() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").next().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
```

```
</html>
```

---

jQuery `nextAll()` Method

The jQuery `nextAll()` method is used to get all following siblings of the selected element.

The following example will highlight all the siblings of the `<p>` element that comes next to it.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery nextAll() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("p").nextAll().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <p>This is another paragraph.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
```

```
</html>
```

---

jQuery `nextUntil()` Method

The jQuery `nextUntil()` method is used to get all the following siblings up to but not including the element matched by the selector. In simple words we can say it returns all the next siblings elements between two given elements in a DOM hierarchy.

The following example will highlight all the following sibling elements of the `<h1>` element excluding the `<ul>` element i.e. highlight both the `<p>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery nextUntil() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("h1").nextUntil("ul").addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <p>This is another paragraph.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
```

```
        </ul>
    </div>
</body>
</html>
```

---

jQuery `prev()` Method

The jQuery `prev()` method is used to get the immediately preceding sibling i.e. the previous sibling element of the selected element. The following example will highlight the previous sibling of the `<ul>` element which is the `<p>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery prev() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul").prev().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <p>This is another paragraph.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
```

```
    </div>
</body>
</html>
```

---

jQuery `prevAll()` Method

The jQuery `prevAll()` method is used to get all preceding siblings of the selected element.

The following example will highlight all siblings of the `<ul>` element that comes prior to this.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery prevAll() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul").prevAll().addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <p>This is another paragraph.</p>
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
```

```
    </div>
</body>
</html>
```

---

jQuery `prevUntil()` Method

The jQuery `prevUntil()` method is used to get all the preceding siblings up to but not including the element matched by the selector. In simple words we can say it returns all the previous siblings elements between two given elements in a DOM hierarchy.

The following example will highlight all the previous sibling elements of the `<ul>` element excluding the `<h1>` element i.e. highlight both the `<p>` element.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery prevUntil() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul").prevUntil("h1").addClass("highlight");
});
</script>
</head>
<body>
    <div class="container">
        <h1>Hello World</h1>
        <p>This is a <em>simple paragraph</em>.</p>
        <p>This is another paragraph.</p>
        <ul>
```

```
            <li>Item One</li>
            <li>Item Two</li>
        </ul>
    </div>
</body>
</html>
```

jQuery Filtering

In this tutorial you will learn how to filter element's selection using jQuery.

Filtering the Elements Selection

jQuery provides several methods such as `filter()`, `first()`, `last()`, `eq()`, `slice()`, `has()`, `not()`, etc. that you can use to narrow down the search for elements in a DOM tree.

jQuery `first()` Method

The jQuery `first()` method filters the set of matched elements and returns the first element from the set. The following example will only highlight the first `<li>` element within the `<ul>` element by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery first() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").first().addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
```

```
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Last list item</li>
    </ul>
</body>
</html>
```

---

jQuery `last()` Method

The jQuery `last()` method filters the set of matched elements and returns the last element from the set. The following example will only highlight the last `<li>` element within the `<ul>` element by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery last() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").last().addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Last list item</li>
    </ul>
```

```
</body>
</html>
```

---

jQuery `eq()` Method

The jQuery `eq()` method filters the set of matched elements and returns only one element with a specified index number. The following example will highlight the second `<li>` element within the `<ul>` element by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery eq() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").eq(1).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Last list item</li>
    </ul>
</body>
</html>
```

Note: The index supplied to the `eq()` method indicates the 0-based position of the element that means the index 0 target the first element, the

index 1 targets the second element and so on. Also this index refers to the position of the element within the jQuery object, not within the DOM tree.

You can also specify a negative index number. A negative index number indicates a position starting from the end of the set, rather than the beginning. For example, the `eq(-2)` indicates the second last element within the set of matched elements.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery eq() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").eq(-2).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

---

jQuery `filter()` Method

The jQuery `filter()` method can take the selector or a function as its argument to filters the set of matched elements based on a specific criteria.

The supplied selector or function to the `filter()` method is tested against each element in the set of matched elements and all the elements that *matching* the supplied selector or pass the function's test will be included in the result.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery filter() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").filter(":even").addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

As stated earlier you can also pass a function to the `filter()` method to filter the set of matched elements based on certain conditions. The following example will test each `<li>` element within the `<ul>` and highlight those `<li>` elements whose indexes are odd numbers i.e. highlights only second and fourth list item as the index is zero-based.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery filter() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").filter(function(index){
        return index % 2 !== 0;
    }).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Last list item</li>
    </ul>
</body>
</html>
```

---

jQuery `has()` Method

The jQuery `has()` method filters the set of matched elements and returns only those elements that has the specified descendant element. The following example will highlight all the `<li>` elements that has the descendant `<ul>` elements.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery filter() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").has("ul").addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>Section 1</li>
        <li>Section 2</li>
        <li>
            <ul>
                <li>Section 2.1</li>
                <li>Section 2.2</li>
                <li>Section 2.3</li>
            </ul>
        </li>
        <li>Section 4</li>
    </ul>
</body>
</html>
```

---

jQuery `not()` Method

The jQuery `not()` method filters the set of matched elements and returns all elements that does not met the specified conditions. It can take the selector or a function as its argument.

The supplied selector or function to the `not()` method is tested against each element in the set of matched elements and all the elements that is *not matching* the supplied selector or pass the function's test will be included in the result.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery not() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").not(":even").addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

The `not()` method can take a function as its argument in the same way that `filter()` does, but it works just opposite of the `filter()` method i.e. the elements that pass the function's test are excluded and rest the elements are included in the result.

The following example will test each `<li>` element within the `<ul>` and highlight those `<li>` elements whose indexes are not the odd numbers i.e. highlights first and third list item.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery not() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").not(function(index){
        return index % 2 !== 0;
    }).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

---

jQuery `slice()` Method

The jQuery `slice()` method filters the set of matched elements specified by a range of indices. This method accepts *start* and *end* (optional) index number as arguments, where the start index specifies the position at which the elements begin to be selected and the end index specify the position at which the elements stop being selected.

The following example will highlight the first and second `<li>` elements within the `<ul>` element by adding the class `.highlight` on document ready.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery slice() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").slice(0, 2).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

You can also specify a negative index numbers. A negative index number indicates a position starting from the end of the set, rather than the beginning. For example, the `slice(-2, -1)` only highlight the third list item, since it is the only item in the range between two from the end (-2) and one from the end (-1), as end position is not included in result.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="utf-8">
<title>jQuery slice() Demo</title>
<style>
    .highlight{
        background: yellow;
    }
</style>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("ul li").slice(-2, -1).addClass("highlight");
});
</script>
</head>
<body>
    <ul>
        <li>First list item</li>
        <li>Second list item</li>
        <li>Third list item</li>
        <li>Fourth list item</li>
    </ul>
</body>
</html>
```

jQuery Ajax

In this tutorial you will learn what Ajax is and how it works.

What is Ajax

Ajax stands for **A**synchronous **J**avascript **A**nd **X**ml. Ajax is just a means of loading data from the server to the web browser without reloading the whole page.

Basically, what Ajax does is make use of the JavaScript-based XMLHttpRequest object to send and receive information to and from a web server asynchronously, in the background, without interfering with the user's experience.

Ajax has become so popular that you hardly find an application that doesn't use Ajax to some extent. The example of some large-scale Ajax-driven online applications are: Gmail, Google Maps, Google Docs, YouTube, Facebook, Flickr, etc.

Note: Ajax is not a new technology, in fact, Ajax is not even really a technology at all. Ajax is just a term to describe the process of exchanging data from a web server asynchronously through JavaScript, without a page refresh.

Ajax with jQuery

Different browsers implement the Ajax differently that means if you're adopting the typical JavaScript way to implement the Ajax you have to write the different code for different browsers to ensure that Ajax would work cross-browser.

But, fortunately jQuery simplifies the process of implementing Ajax by taking care of those browser differences. It offers simple methods such

as `load()`, `$.get()`, `$.post()`, etc. to implement the Ajax that works seamlessly across all the browsers.

In the upcoming chapters you will learn how to load data from the server as well as how to send and receive data using HTTP GET and POST method through jQuery Ajax.

Tip: Ajax requests are triggered by the JavaScript code; your code sends a request to a URL, and when the request completes, a callback function can be triggered to handle the response. Further, since the request is asynchronous, the rest of your code continues to execute while the request is being processed.

jQuery Ajax Load

In this tutorial you will learn how to load data from server using jQuery.

jQuery `load()` Method

The jQuery `load()` method loads data from the server and place the returned HTML into the selected element. This method provides a simple way to load data asynchronous from a web server. The basic syntax of this method can be given with:

```
$(selector).load(URL, data, complete);
```

The parameters of the `load()` method has the following meaning:

- The required *URL* parameter specifies the URL of the file you want to load.
- The optional *data* parameter specifies a set of query string (i.e. key/value pairs) that is sent to the web server along with the request.
- The optional *complete* parameter is basically a callback function that is executed when the request completes. The callback is fired once for each selected element.

Let's put this method into real use. Create a blank HTML file "test-content.html" and save it somewhere in your web server. Now place the following HTML code inside of this file:

Example

Try this code »

```
<h1>Simple Ajax Demo</h1>
<p id="hint">This is a simple example of Ajax loading.</p>
<p><img src="sky.jpg" alt="Cloudy Sky"></p>
```

Now, create one more HTML file say "load-demo.html", and save it at the same location where you've saved the previous file. Now put the following HTML code inside of it:

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery load() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").load("test-content.html");
    });
});
</script>
</head>
<body>
    <div id="box">
        <h2>Click button to load new content inside DIV
box</h2>
    </div>
    <button type="button">Load Content</button>
</body>
</html>
```

Finally, open this page in your browser and click the "Load Content" button. You'll see the content of DIV box is replaced by the HTML content of the "test-content.html" file.

Tip: To test this Ajax example you need to place the HTML files on a web server. You can set up a local web server on your PC by installing WampServer or XAMPP. You must open the demo file using "http://" since Ajax makes HTTP requests.

Note: Ajax request can be made only to the files that exist on the same web server that servers the page from which the Ajax request is sent, not to external or remote servers for security reasons. This is called same-origin policy.

Further, the callback function can have three different parameters:

- `responseTxt` — Contains the resulting content if the request succeeds.
- `statusTxt` — Contains the status of the request such as success or error.

- `jqXHR` — Contains the XMLHttpRequest object.

Here's the modified version of the previous example that will display either the success or error message to the user depending on the status of the request.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery load() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").load("test-content.html",
function(responseTxt, statusTxt, jqXHR){
            if(statusTxt == "success"){
                alert("New content loaded successfully!");
            }
            if(statusTxt == "error"){
                alert("Error: " + jqXHR.status + " " +
jqXHR.statusText);
            }
        });
    });
});
</script>
</head>
<body>
    <div id="box">
        <h2>Click button to load new content inside DIV
box</h2>
    </div>
    <button type="button">Load Content</button>
</body>
</html>
```

Loading Page Fragments

The jQuery `load()` also allows us to fetch only a portion of the document. This is simply achieved by appending the `url` parameter with a space followed by a jQuery selector, let's check out the following example to make it more clear.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery load() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#box").load("test-content.html #hint");
    });
});
</script>
</head>
<body>
    <div id="box">
        <h2>Click button to load new content inside DIV
box</h2>
    </div>
    <button type="button">Load Content</button>
</body>
</html>
```

The jQuery selector `#hint` within the `url` parameter (*line no-10*), specify the portion of the "test-content.html" file to be inserted inside the DIV box, which is an element that has the ID attribute with a value `hint` i.e. `id="hint"`, see the first example.

jQuery
Ajax GET and POST Requests

In this tutorial you will learn how to send and receive data from a web server through Ajax via HTTP GET or POST methods using jQuery.

jQuery `$.get()` and `$.post()` Methods

The jQuery's `$.get()` and `$.post()` methods provide simple tools to send and retrieve data asynchronously from a web server. Both the methods are pretty much identical, apart from one major difference — the `$.get()` makes Ajax requests using the HTTP GET method, whereas the `$.post()` makes Ajax requests using the HTTP POST method.

The basic syntax of these methods can be given with:

```
$.get(URL, data, success);   –Or–   $.post(URL, data, success);
```

The parameters in the above syntax have the following meaning:

- The required *URL* parameter specifies the URL to which the request is sent.
- The optional *data* parameter specifies a set of query string (i.e. key/value pairs) that is sent to the web server along with the request.
- The optional *success* parameter is basically a callback function that is executed if the request succeeds. It is typically used to retrieve the returned data.

Note: The HTTP GET and POST methods are used to send request from a browser to a server. The main difference between these methods is the way in which the data is passed to the server. Check out the tutorial on GET and POST methods for the detailed explanation and comparison between these two methods.

Performing GET Request with AJAX using jQuery

The following example uses the jQuery `$.get()` method to make an Ajax request to the "date-time.php" file using HTTP GET method. It simply retrieves the date and time returned from the server and displays it in the browser without refreshing the page.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery get() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $.get("date-time.php", function(data){
            // Display the returned data in browser
            $("#result").html(data);
        });
    });
});
</script>
</head>
<body>
    <div id="result">
        <h2>Content of the result DIV box will be replaced
by the server date and time</h2>
    </div>
    <button type="button">Load Date and Time</button>
</body>
</html>
```

Here's our "date-time.php" file that simply output the current date and time of the server.

Example

Download

```
<?php
```

```
// Return current date and time from the server
echo date("F d, Y h:i:s A");
?>
```

Tip: If you face any difficulty while running these examples locally on your PC, please check out the tutorial on jQuery Ajax load for the solution.

You can also send some data to the server with the request. In the following example the jQuery code makes an Ajax request to the "create-table.php" as well as sends some additional data to the server along with the request.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery get() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        // Get value from input element on the page
        var numValue = $("#num").val();

        // Send the input data to the server using get
        $.get("create-table.php", {number: numValue} ,
function(data){
            // Display the returned data in browser
            $("#result").html(data);
        });
    });
});
</script>
</head>
<body>
    <label>Enter a Number: <input type="text"
id="num"></label>
    <button type="button">Show Multiplication
Table</button>
    <div id="result"></div>
</body>
</html>
```

Here's the PHP script of our "create-table.php" file that simply output the multiplication table for the number entered by the user on button click.

Example
Download

```php
<?php
$number = htmlspecialchars($_GET["number"]);
if(is_numeric($number) && $number > 0){
    echo "<table>";
    for($i=0; $i<11; $i++){
        echo "<tr>";
            echo "<td>$number x $i</td>";
            echo "<td>=</td>";
            echo "<td>" . $number * $i . "</td>";
        echo "</tr>";
    }
    echo "</table>";
}
?>
```

Performing POST Request with AJAX using jQuery

POST requests are identical to GET requests in jQuery. So, generally which method you should use either `$.get()` or `$.post()` is basically depends on the requirements of your server-side code. If you have large amount of data to be transmitted (e.g. form data) you need to use POST, because GET has a stringent limit on the data transfer.

Example
Try this code »

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery post() Demo</title>
<script src="https://code.jquery.com/jquery-
3.5.1.min.js"></script>
```

```
<script>
$(document).ready(function(){
    $("form").submit(function(event){
        // Stop form from submitting normally
        event.preventDefault();

        /* Serialize the submitted form control values to
be sent to the web server with the request */
        var formValues = $(this).serialize();

        // Send the form data using post
        $.post("display-comment.php", formValues,
function(data){
            // Display the returned data in browser
            $("#result").html(data);
        });
    });
});
</script>
</head>
<body>
    <form>
        <label>Name: <input type="text"
name="name"></label>
        <label>Comment: <textarea cols="50"
name="comment"></textarea></label>
        <input type="submit" value="Send">
    </form>
    <div id="result"></div>
</body>
</html>
```

Here's our "display-comment.php" file that simply output the data entered by the user.

Example

Download

```
<?php
$name = htmlspecialchars($_POST["name"]);
$comment = htmlspecialchars($_POST["comment"]);
echo "Hi, $name. Your comment has been received
successfully." . "";
echo "Here's the comment what you've entered: $comment";
?>
```

Now that you have learnt how to perform various Ajax operations such as loading data, submitting form, etc. asynchronously using jQuery. Before

concluding this chapter check out one more classic example of Ajax that will show you how to populate the state or city dropdown based on the option selected in the country dropdown using jQuery.

jQuery No-Conflict Mode

In this tutorial you will learn how to avoid conflicts between jQuery and other JavaScript library or framework.

Using jQuery with Other JavaScript Libraries

As you already know, jQuery uses the dollar sign (`$`) as a shortcut or alias for `jQuery`. Thus, if you use another JavaScript library that also uses the `$` sign as a shortcut, along with the jQuery library on the same page, conflicts could occur. Fortunately, jQuery provides a special method named `noConflict()` to deal with such situation.

jQuery noConflict() Method

The `jQuery.noConflict()` method return the control of the `$` identifier back to other libraries. The jQuery code in the following example (*line no-10*) will put the jQuery into no-conflict mode immediately after it is loaded onto the page and assign a new variable name `$j` to replace the `$` alias in order to avoid conflicts with the prototype framework.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery noConflict() Demo</title>
<script src="js/prototype.js"></script>
<script src="js/jquery.js"></script>
<script>
// Defining the new alias for jQuery
var $j = jQuery.noConflict();
$j(document).ready(function(){
    // Display an alert message when the element with ID
foo is clicked
    $j("#foo").click(function(){
```

```
        alert("jQuery is working well with prototype.");
    });
});

// Some prototype framework code
document.observe("dom:loaded", function(){
    // Display an alert message when the element with ID
bar is clicked
    $("bar").observe("click", function(event){
        alert("Prototype is working well with jQuery.");
    });
});
</script>
</head>
<body>
    <button type="button" id="foo">Run jQuery Code</button>
    <button type="button" id="bar">Run Prototype
Code</button>
</body>
</html>
```

Note: Many JavaScript libraries use the `$` as a function or variable name, just like the jQuery. Some of these libraries are: mootools, prototype, zepto etc.

However, if you don't want to define another shortcut for jQuery, may be because you don't want to modify your existing jQuery code or you really like to use `$` because it saves time and easy to use, then you can adopt another quick approach — simply pass the `$` as an argument to your `jQuery(document).ready()` function, like this:

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery noConflict() Demo</title>
<script src="js/prototype.js"></script>
<script src="js/jquery.js"></script>
<script>
jQuery.noConflict();
jQuery(document).ready(function($){
    // The dollar sign in here work as an alias to jQuery
    $("#foo").click(function(){
```

```
        alert("jQuery is working well with prototype.");
    });
});

// Some prototype framework code
document.observe("dom:loaded", function(){
    // The dollar sign in the global scope refer to
prototype
    $("bar").observe("click", function(event){
        alert("Prototype is working well with jQuery.");
    });
});
</script>
</head>
<body>
    <button type="button" id="foo">Run jQuery Code</button>
    <button type="button" id="bar">Run Prototype
Code</button>
</body>
</html>
```

Including jQuery Before Other Libraries

The above solutions to avoid conflicts rely on jQuery is being loaded after prototype.js is loaded. However, if you include jQuery before other libraries, you may use full name `jQuery` in your jQuery code to avoid conflicts without calling the `jQuery.noConflict()`. But in this scenario the `$` will have the meaning defined in the other library.

Example

Try this code »

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery noConflict() Demo</title>
<script src="js/jquery.js"></script>
<script src="js/prototype.js"></script>
<script>
jQuery(document).ready(function($){
    // Use full jQuery function name to reference jQuery
```

```
    jQuery("#foo").click(function(){
        alert("jQuery is working well with prototype.");
    });
});

// Some prototype framework code
document.observe("dom:loaded", function(){
    // The dollar sign here will have the meaning defined
in prototype
    $("bar").observe("click", function(event){
        alert("Prototype is working well with jQuery.");
    });
});
</script>
</head>
<body>
    <button type="button" id="foo">Run jQuery Code</button>
    <button type="button" id="bar">Run Prototype
Code</button>
</body>
</html>
```