16. jQuery Traversing

16.1. What is Traversing?

jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.

The image below illustrates a family tree. With jQuery traversing, you can easily move up (ancestors), down (descendants) and sideways (siblings) in the family tree, starting from the selected (current) element. This movement is called traversing - or moving through - the DOM.

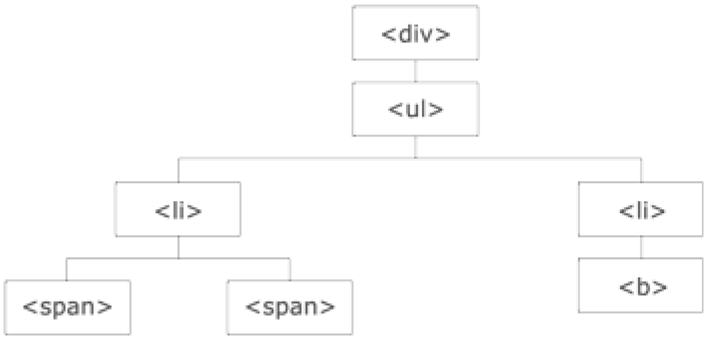

Illustration explained:

- The <div> element is the **parent** of <ul>, and an **ancestor** of everything inside of it
- The <ul> element is the **parent** of both <li> elements, and a **child** of <div>
- The left <li> element is the **parent** of <span>, **child** of <ul> and a **descendant** of <div>
- The <span> element is a **child** of the left <li> and a **descendant** of <ul> and <div>
- The two <li> elements are **siblings** (they share the same parent)
- The right <li> element is the **parent** of <b>, **child** of <ul> and a **descendant** of <div>
- The <b> element is a **child** of the right <li> and a **descendant** of <ul> and <div>

An ancestor is a parent, grandparent, great-grandparent, and so on.
A descendant is a child, grandchild, great-grandchild, and so on.
Siblings share the same parent.

16.2. Traversing the DOM

jQuery provides a variety of methods that allows us to traverse the DOM.

The largest category of traversal methods are tree-traversal.

The next chapters will show us how to travel up, down and sideways in the DOM tree.

jQuery Traversing - Ancestors

An ancestor is a parent, grandparent, great-grandparent, and so on.

With jQuery you can traverse up the DOM tree to find ancestors of an element.

16.3. Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()
- parentsUntil()

a) jQuery parent() Method

The parent() method returns the direct parent element of the selected element.

This method only traverse a single level up the DOM tree.

The following example returns the direct parent element of each <span> elements:

Example

```
$(document).ready(function(){
  $("span").parent();
});
```

b) jQuery parents() Method

The parents() method returns all ancestor elements of the selected element, all the way up to the document's root element (<html>).

The following example returns all ancestors of all <span> elements:

Example

```
$(document).ready(function(){
  $("span").parents();
});
```

You can also use an optional parameter to filter the search for ancestors.

The following example returns all ancestors of all <span> elements that are <ul> elements:

Example

```
    $(document).ready(function(){
      $("span").parents("ul");
    });
```

c) jQuery parentsUntil() Method

The parentsUntil() method returns all ancestor elements between two given arguments.

The following example returns all ancestor elements between a <span> and a <div> element:

Example

```
    $(document).ready(function(){
      $("span").parentsUntil("div");
    });
```

jQuery Traversing - Descendants

A descendant is a child, grandchild, great-grandchild, and so on.

With jQuery you can traverse down the DOM tree to find descendants of an element.

16.4. Traversing Down the DOM Tree

Two useful jQuery methods for traversing down the DOM tree are:

- children()
- find()

a) jQuery children() Method

The children() method returns all direct children of the selected element.

This method only traverse a single level down the DOM tree.

The following example returns all elements that are direct children of each <div> elements:

Example

```
$(document).ready(function(){
  $("div").children();
});
```

You can also use an optional parameter to filter the search for children.

The following example returns all <p> elements with the class name "1", that are direct children of <div>:

Example

```
$(document).ready(function(){
  $("div").children("p.1");
});
```

b) jQuery find() Method

The find() method returns descendant elements of the selected element, all the way down to the last descendant.

The following example returns all <span> elements that are descendants of <div>:

Example

```
$(document).ready(function(){
  $("div").find("span");
});
```

The following example returns all descendants of <div>:

Example

```
$(document).ready(function(){
  $("div").find("*");
});
```

jQuery Traversing - Siblings

Siblings share the same parent.

With jQuery you can traverse sideways in the DOM tree to find siblings of an element.

16.5. Traversing Sideways in The DOM Tree

There are many useful jQuery methods for traversing sideways in the DOM tree:

- siblings()
- next()
- nextAll()
- nextUntil()
- prev()
- prevAll()
- prevUntil()

a) jQuery siblings() Method

The siblings() method returns all sibling elements of the selected element.

The following example returns all sibling elements of <h2>:

Example

```
$(document).ready(function(){
  $("h2").siblings();
});
```

You can also use an optional parameter to filter the search for siblings.

The following example returns all sibling elements of <h2> that are <p> elements:

Example

```
$(document).ready(function(){
  $("h2").siblings("p");
});
```

b) jQuery next() Method

The next() method returns the next sibling element of the selected element.

This method only returns one element.

The following example returns the next sibling of <h2>:

Example

```
$(document).ready(function(){
  $("h2").next();
});
```

c) jQuery nextAll() Method

The nextAll() method returns all next sibling elements of the selected element.

The following example returns all next sibling elements of <h2>:

Example

```
$(document).ready(function(){
  $("h2").nextAll();
});
```

d) jQuery nextUntil() Method

The nextUntil() method returns all next sibling elements between two given arguments.

The following example returns all sibling elements between a <h2> and a <h6> element:

Example

```
$(document).ready(function(){
  $("h2").nextUntil("h6");
});
```

e) jQuery prev(), prevAll() & prevUntil() Methods

The prev(), prevAll() and prevUntil() methods work just like the methods above but with reverse functionality: they return previous sibling elements (traverse backwards along sibling elements in the DOM tree, instead of forward).

jQuery Traversing - Filtering

16.6. Narrow Down The Search For Elements

The three most basic filtering methods are first(), last() and eq(), which allow you to select a specific element based on its position in a group of elements.

Other filtering methods, like filter() and not() allow you to select elements that match, or not match, a certain criteria.

a) jQuery first() Method

The first() method returns the first element of the selected elements.

The following example selects the first <p> element inside the first <div> element:

Example

```
$(document).ready(function(){
  $("div p").first();
});
```

b) jQuery last() Method

The last() method returns the last element of the selected elements.

The following example selects the last <p> element inside the last <div> element:

Example

```
$(document).ready(function(){
  $("div p").last();
});
```

c) jQuery eq() method

The eq() method returns an element with a specific index number of the selected elements.

The index numbers start at 0, so the first element will have the index number 0 and not 1. The following example selects the second <p> element (index number 1):

Example

```
$(document).ready(function(){
  $("p").eq(1);
});
```

d) jQuery filter() Method

The filter() method lets you specify a criteria. Elements that do not match the criteria are removed from the selection, and those that match will be returned.

The following example returns all <p> elements with class name "intro":

Example

```
$(document).ready(function(){
  $("p").filter(".intro");
});
```

e) jQuery not() Method

The not() method returns all elements that do not match the criteria.

Tip: The not() method is the opposite of filter().

The following example returns all <p> elements that do not have class name "intro":

Example

```
$(document).ready(function(){
  $("p").not("intro");
});
```

16.7. jQuery Traversing Reference

For a complete overview of all jQuery Traversing methods, please go to our jQuery Traversing Reference.