

- jQuery?

- Features of jQuery

- fast, small, and feature-rich JavaScript library.
- jQuery greatly simplifies JavaScript programming.
- easy to learn and code.
- it also requires much less lines of code to achieve the same feature in comparison.
- lightweight, "write less, do more",
- It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.
- Popular JavaScript framework.
- **Cross Browser Friendly** – jQuery is currently the most popular JavaScript library and works in all browsers.
- **Plug-ins** – There are an abundance of plug-ins on the web that make creating special effects simple and fast for web developers.

- The jQuery library contains the following features:
- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations

Adding jQuery to Your Web Pages

- Download the jQuery library from [jQuery.com](http://jquery.com)
- Or
- Include jQuery from a CDN, like Google

- 1. Downloading jQuery
- The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag:
- `<head>`
`<script src="jquery-1.11.2.min.js"></script>`
`</head>`
- Place the downloaded file in the same directory as the pages where you wish to use it.

2.jQuery CDN

- Include it from a CDN (Content Delivery Network).Both Google and Microsoft host jQuery.
- If many different web sites use the same JavaScript framework, it makes sense to host the framework library in a common location for every web page to share. A CDN (Content Delivery Network) solves this. A CDN is a network of servers containing shared code libraries Google provides a free CDN for a number of JavaScript libraries, including:
 - jQuery
 - Prototype
 - MooTools
 - Dojo
 - Yahoo! YUI

- To use jQuery from Google or Microsoft, use one of the following or To use a JavaScript framework library in your web pages, just include the library in a `<script>` tag:
- `<head>`
`<script`
`src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>`
`</head>`

Or

- `<head>`
`<script`
`src="http://ajax.aspnetcdn.com/ajax/jquery/j`
`query-1.11.2.min.js"></script>`
`</head>`

Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

jQuery Syntax

\$(selector).action()

sign to access jQuery

query (or find)" HTML elements

action() to be performed on the element(s)

\$("p").hide() - hides all <p> elements.

The Document Ready Event

- all jQuery methods are inside a document ready event:
- `$(document).ready(function(){`

// jQuery methods go here...

`});`

- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- We should wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

Or you can simply write:

```
$(function(){
```

```
    // jQuery methods go here...
```

```
});
```

jQuery Selectors

- allow us to select and manipulate HTML element(s).
- All selectors in jQuery start with the dollar sign and parentheses: \$().

- **The element Selector:**selects elements based on the element name.For eg.

`$("p")`

- **The #id Selector:**uses the id attribute of an HTML tag to find the specific element.

`$("#test")`

- **The .class Selector:** finds elements with a specific class.
- `$(".test")`

Element selector

- `<!DOCTYPE html>`
- `<html>`
- `<head>`
- `<script`
`src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>`
-
- Or `Src="jquery-1.11.2.min.js"`
- `<script>`
- `$(document).ready(function(){`
- `$("button").click(function(){`
- `$("p").hide();`
- `});`
- `});`
- `</script>`
- `</head>`

- `<body>`
- `<h2>This is a heading</h2>`
- `<p>This is a paragraph.</p>`
- `<p>This is another paragraph.</p>`
- `<button>Click me</button>`
- `</body>`
- `</html>`

Id selector

- <!DOCTYPE html>
- <html>
- <head>
- <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
- <script>
- \$(document).ready(function(){
- \$("button").click(function(){
- \$("#test").hide();
- });
- });
- </script>
- </head>

- `<body>`
- `<h2>This is a heading</h2>`
- `<p>This is a paragraph.</p>`
- `<p id="test">This is another paragraph.</p>`
- `<button>Click me</button>`
- `</body>`
- `</html>`

jQuery Selectors

- `$("*")`: Selects all elements
- `$(this)`: Selects the current HTML element
- `$("p.intro")`: Selects all `<p>` elements with `class="intro"`
- `$("p:first")`: Selects the first `<p>` element
- `$("ul li:first")`: Selects the first `` element of the first ``
- `$("ul li:first-child")`: Selects the first `` element of every ``
- `$("[href]")`: Selects all elements with an href attribute

- `$("a[target='_blank']")`:Selects all `<a>` elements with a target attribute value equal to `"_blank"`
- `$("a[target!='_blank']")`:Selects all `<a>` elements with a target attribute value NOT equal to `"_blank"`
- `$(":button")`:Selects all `<button>` elements and `<input>` elements of `type="button"`
- `$("tr:even")`:Selects all even `<tr>` elements
- `$("tr:odd")`:Selects all odd `<tr>` elements

jQuery Event Methods

- An event represents the precise moment when something happens.
- The term "**fires**" is often used with events.

Events

- **Mouse Events:**
- click,dblclick,mouseenter,mouseleave
- **Keyboard Events:**
- keypress,keydown,keyup
- **Form Events:**
- submit,change,focus,blur
- **Document/Window :**
- load,resize,scroll,unload

jQuery Syntax For Event Methods

- To assign a click event to all paragraphs on a page,
- `$("p").click();`
- The next step is to define what should happen when the event fires. You must pass a function to the event:
- `$("p").click(function(){
 // action goes here!!
});`

Commonly Used jQuery Event Methods

- **`$(document).ready():`**


The `$(document).ready()` method allows us to execute a function when the document is fully loaded.

- **`click():`**

The `click()` method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element.

- The following example says: When a click event fires on a `<p>` element; hide the current `<p>` element:

Click event

- <!DOCTYPE html>
- <html>
- <head>
- <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
- <script>
- \$(document).ready(function(){
- \$("p").click(function(){
- \$(this).hide();  current element will be hidden i.e button
- });
- });
- </script>
- </head>

- `<body>`
- `<p>HELLO WORLD</p>`
- `<p>Click me away!</p>`
- `<p>Click me too!</p>`
- `</body>`
- `</html>`

- **dblclick()**

The `dblclick()` method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

- ```
$("#p").dblclick(function(){
 $(this).hide();
});
```

- **mouseenter()**
- The mouseenter() method attaches an event handler function to an HTML element.
- The function is executed when the mouse pointer enters the HTML element:
- ```
$("#p1").mouseenter(function(){  
    alert("You entered p1!");  
});
```

- **mouseleave()**

- The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

```
$("#p1").mouseleave(function(){  
    alert("Bye! You now leave p1!");  
});
```

- **mouseup()**
- The mouseup() method attaches an event handler function to an HTML element.
- The function is executed, when the left mouse button is released, while the mouse is over the HTML element:
- ```
$("#p1").mouseup(function(){
 alert("Mouse up over p1!");
});
```



- **mousedown()**
- The mousedown() method attaches an event handler function to an HTML element.
- The function is executed, when the left mouse button is pressed down, while the mouse is over the HTML element:
- ```
$("#p1").mousedown(function(){  
    alert("Mouse down over p1!");  
});
```

- **hover()**
- The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.
- The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example of hover

- ```
$("#p1").hover(function(){
 alert("You entered p1!");
},
function(){
 alert(" You now leave p1!");
});
```

- **focus()**
- The focus() method attaches an event handler function to an HTML form field.
- The function is executed when the form field gets focus:
- ```
$("#input").focus(function(){  
    $(this).css("background-color", "#cccccc");  
});
```

- **blur()**
- The blur() method attaches an event handler function to an HTML form field.
- The function is executed when the form field loses focus:
- ```
$("#input").blur(function(){
 $(this).css("background-color", "#ffffff");
});
```

- **The on() Method:**
- The on() method attaches one or more event handlers for the selected elements.

- <!DOCTYPE html>
- <html>
- <head>
- <script  
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.  
js"></script>
- <script>
- \$(document).ready(function(){
- \$("p").on({
- mouseenter: function(){
- \$(this).css("background-color", "lightgray");
- },
- mouseleave: function(){
- \$(this).css("background-color", "lightblue");
- },
- click: function(){
- \$(this).css("background-color", "yellow");

- }
- });
- });
- </script>
- </head>
- <body>
- <p>Click or move the mouse pointer over this paragraph.</p>
- </body>
- </html>



# jQuery Effects - Hide and Show

- jQuery hide() and show()

- `$(selector).hide(speed,callback);`

`$(selector).show(speed,callback);`

- The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the hide() or show() method completes

# example

- `$("#hide").click(function(){  
 $("p").hide();` → all paragraph elements will be hidden  
`});`

```
$("#show").click(function(){
 $("p").show();
});
```

# jQuery toggle()

- can toggle between the hide() and show() methods with the toggle() method.
- `$(selector).toggle(speed,callback);`
- `$("#button").click(function(){  
 $("#p").toggle();  
});`

# jQuery Effects - Fading

- **Fading Methods:** we can fade an element in and out of visibility.
- **fadeIn() Method:** is used to fade in a hidden element.
- **Syntax:**
- ***\$(selector).fadeIn(speed,callback);***
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the fading completes.

# example

- `$(document).ready(function(){`
  - `$("button").click(function(){`
  - `$("#div1").fadeIn();`
  - `$("#div2").fadeIn("slow");`
  - `$("#div3").fadeIn(3000);`
  - `});`
  - `});`
- no of milliseconds

- **fadeOut() Method**
- The jQuery fadeOut() method is used to fade out a visible element.
- **Syntax:**
- `$(selector).fadeOut(speed,callback);`

- **fadeToggle() Method**
- The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.
- If the elements are faded out, fadeToggle() will fade them in.
- If the elements are faded in, fadeToggle() will fade them out.
- **Syntax:**
- `$(selector).fadeToggle(speed,callback);`

- **fadeTo() Method**
- The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).
- **Syntax:**
- ***`$(selector).fadeTo(speed,opacity,callback)`***
- The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).



# Sliding Methods

- can create a sliding effect on elements.
- jQuery has the following slide methods:
- `slideDown()`
- `slideUp()`
- `slideToggle()`
-

- **slideDown() Method:** is used to slide down an element.
- **Syntax:**
- *\$(selector).slideDown(speed,callback);*
- `$("#flip").click(function(){  
 $("#panel").slideDown();  
});`

- **slideUp() Method**
- The jQuery slideUp() method is used to slide up an element.
- **Syntax:**
- *\$(selector).slideUp(speed,callback);*

- **slideToggle() Method**
- The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.
- If the elements have been slid down, slideToggle() will slide them up.
- If the elements have been slid up, slideToggle() will slide them down.
- ***\$(selector).slideToggle(speed,callback);***

# Animation

- **The animate() Method**
- The jQuery animate() method is used to create custom animations.
- **Syntax:**
- `$(selector).animate({params},speed,callback);`
- The required params parameter defines the CSS properties to be animated.
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the animation completes.

# animate() - Manipulate Multiple Properties

- it moves a <div> element to the right, until it has reached a left property of 250px:
- Example
- ```
$("#button").click(function(){  
    $("#div").animate({  
        left: '250px',  
        opacity: '0.5',  
        height: '150px',  
        width: '150px'  
    });  
});
```

- **animate() - Using Relative Values**
- It is also possible to define relative values (the value is then relative to the element's current value). This is done by putting += or -= in front of the value:
- Example
- ```
$("#button").click(function(){
 $("#div").animate({
 left: '250px',
 height: '+=150px',
 width: '+=150px'
 });
});
```

- **animate()** - Using Pre-defined Values
- You can even specify a property's animation value as "show", "hide", or "toggle":
- Example
- ```
$("#button").click(function(){  
    $("#div").animate({  
        height: 'toggle'  
    });  
});
```


- **animate() - Uses Queue Functionality**
- By default, jQuery comes with queue functionality for animations.
- This means that if you write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.
- So, if you want to perform different animations after each other, we take advantage of the queue functionality:

example

- `$("#button").click(function(){
 var div = $("#div");
 div.animate({height: '300px', opacity: '0.4'},
"slow");
 div.animate({width: '300px', opacity: '0.8'},
"slow");
 div.animate({height: '100px', opacity: '0.4'},
"slow");
 div.animate({width: '100px', opacity: '0.8'},
"slow");
});`

Previous example first moves the <div> element to the right, and then increases the font size of the text:

```
$("#button").click(function(){  
    var div = $("#div");  
    div.animate({left: '100px'}, "slow");  
    div.animate({fontSize: '3em'}, "slow");  
});
```

-

stop() Method

- The jQuery stop() method is used to stop an animation or effect before it is finished.
- The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.
- **Syntax:**
- *\$(selector).stop();*

Callback Functions

- JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
- To prevent this, you can create a callback function.
- A callback function is executed after the current effect is finished.
- syntax: ***\$(selector).hide(speed,callback);***

- The example below has a callback parameter that is a function that will be executed after the hide effect is completed:
- ```
$("#button").click(function(){
 $("#p").hide("slow", function(){
 alert("The paragraph is now hidden");
 });
});
```

# Without callback

- The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

```
$("#button").click(function(){
 $("#p").hide(1000);
 alert("The paragraph is now hidden");
});
```

# Chaining

- We can chain together actions/methods.
- Chaining allows us to run multiple jQuery methods (on the same element) within a single statement.



# jQuery Method Chaining

- Until now we have been writing jQuery statements one at a time (one after the other).
- However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).
- This way, browsers do not have to find the same element(s) more than once.
- To chain an action, you simply append the action to the previous action.
- The following example chains together the `css()`, `slideUp()`, and `slideDown()` methods. The "p1" element first changes to red, then it slides up, and then it slides down:

# example

- `$("#p1").css("color", "red").slideUp(2000).slideDown(2000);`

# jQuery - Get Content and Attributes

- jQuery contains powerful methods for changing and manipulating HTML elements and attributes.
- **jQuery DOM Manipulation**
- One very important part of jQuery is the possibility to manipulate the DOM.
- jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

# Get Content - text(), html(), and val()

- 3 jQuery methods for DOM manipulation are:
- **text()** - Sets or returns the text content of selected elements
- **html()** - Sets or returns the content of selected elements (including HTML markup)
- **val()** - Sets or returns the value of form fields

# how to get content with the jQuery text() and html() methods:

- ```
$("#btn1").click(function(){  
    alert("Text: " + $("#test").text());  
});  
$("#btn2").click(function(){  
    alert("HTML: " + $("#test").html());  
});
```

how to get the value of an input field
with the jQuery val() method:

- `$("#btn1").click(function(){
 alert("Value: " + $("#test").val());
});`

Get Attributes - attr()

- The jQuery attr() method is used to get attribute values.
- The following example demonstrates how to get the value of the href attribute in a link:
- Example
- ```
$("#button").click(function(){
 alert($("#ashoka").attr("href"));
});
```
-

# Set Content - text(), html(), and val()

- ```
$("#btn1").click(function(){  
    $("#test1").text("Hello world!");  
});  
$("#btn2").click(function(){  
    $("#test2").html("<b>Hello world!</b>");  
});  
$("#btn3").click(function(){  
    $("#test3").val("Dolly Duck");  
});
```


Set Attributes - attr()

- is also used to set/change attribute values.
- The following example demonstrates how to change (set) the value of the href attribute in a link:
- ```
$("#button").click(function(){
 $("#ashoka").attr("href",
 "http://www.google.com");
});
```

# jQuery - Add Elements

- Add New HTML Content
- four jQuery methods that are used to add new content:
- **append()** - Inserts content at the end of the selected elements
- **prepend()** - Inserts content at the beginning of the selected elements
- **after()** - Inserts content after the selected elements
- **before()** - Inserts content before the selected elements

- **append() Method**
- `$("p").append("Some appended text.");`
- **prepend() Method**
- `$("p").prepend("Some prepended text.");`

# example

- <!DOCTYPE html>
- <html>
- <head>
- <script  
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
- <script>
- \$(document).ready(function(){
- \$("#btn1").click(function(){
- \$("p").append(" <b>hello world</b>.");
- });
- \$("#btn2").click(function(){
- \$("ol").append("<li>Appended list item</li>");
- });
- });
- </script>
- </head>

- `<body>`
- `<p>This is a paragraph.</p>`
- `<p>This is another paragraph.</p>`
- `<ol>`
- `<li>List item 1</li>`
- `<li>List item 2</li>`
- `<li>List item 3</li>`
- `</ol>`
- `<button id="btn1">Append text</button>`
- `<button id="btn2">Append list items</button>`
- `</body>`
- `</html>`

# Add Several New Elements With append() and prepend()

- `<script>`
- `function appendText() {`
- `var txt1 = $("<p></p>").text("orange");`
- `var txt2 = $("<p></p>").text("apple");`
- `var txt3 = document.createElement("p");`
- `txt3.innerHTML = "watermelon";`
- `$("p").append(txt1, txt2, txt3);`
- `}`
- `</script>`

- The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the `append()` method (this will work for `prepend()` too)

# after() and before() Methods

- `$(“element”).after("Some text after");`  
`$(“element”).before("Some text before");`



# Remove Elements/Content

- **remove()** - Removes the selected element (and its child elements).for e.g
    - `$("#div1").remove();`
    - In this example, the element with id div1 and its children will be removed
  - **empty()** - Removes the child elements from the selected element for e.g
    - `$("#div1").empty();`
- Only the children will be removed.

# Filter the Elements to be Removed

- The jQuery `remove()` method also accepts one parameter, which allows you to filter the elements to be removed.
- The parameter can be any of the jQuery selector syntaxes.
- The following example removes all `<p>` elements with `class="raman"`:

- `$("p").remove(".raman");`
- This example removes all `<p>` elements with `class="test"` and `class="demo"`:
- **Example**
- `$("p").remove(".test, .demo");`

# jQuery Manipulating CSS

- following methods:
- `addClass()` - Adds one or more classes to the selected elements
- `removeClass()` - Removes one or more classes from the selected elements
- `toggleClass()` - Toggles between adding/removing classes from the selected elements
- `css()` - Sets or returns the style attribute

- **addClass() Method**

- The following example shows how to add class attributes to different elements. We can select multiple elements, when adding classes:

- **Example**

- ```
$("#button").click(function(){  
    $("#h1, h2, p").addClass("ashoka");  
    $("#div").addClass("raman");  
});
```
- Here we have added a class named as ashoka to 3 elements, h1, h2, p.
- Raman class is added to div element.
- We can also specify multiple classes within the addClass() method.

- **removeClass() Method**
- The following example shows how to remove a specific class attribute from different elements:
- **Example**
- ```
$("#button").click(function(){
 $("#h1, h2, p").removeClass("ashoka");
});
```
- ashoka class has been removed from h1,h2,p

- **toggleClass() Method**
- The following example will show how to use the jQuery toggleClass() method. This method toggles between adding/removing classes from the selected elements:
- **Example**
- ```
$("#button").click(function(){  
    $("#h1, h2, p").toggleClass("blue");  
});
```

- **css() Method**
- The css() method sets or returns one or more style properties for the selected elements.

- **Return a CSS Property**
- To return the value of a specified CSS property, use the following syntax:
- *css("propertyname");*
- The following example will return the background-color value of the FIRST matched paragraph element:
- **Example**
- `$("p").css("background-color");`

- **Set a CSS Property**
- To set a specified CSS property, use the following syntax:
- `css("propertyname","value");`
- The following example will set the background-color value for ALL matched elements:
- **Example**
- `$("p").css("background-color", "yellow");`

- **Set Multiple CSS Properties**
- To set multiple CSS properties, use the following syntax:
- `css({"propertyname":"value","propertyname":"value",...});`
- The following example will set a background-color and a font-size for ALL matched elements:
- **Example**
- `$("p").css({"background-color": "yellow", "font-size": "200%"});`

Dimension Methods

- **width() and height() Methods**
- The width() method sets or returns the width of an element (excludes padding, border and margin).
- The height() method sets or returns the height of an element (excludes padding, border and margin).

- **innerWidth() and innerHeight() Methods**
- The innerWidth() method returns the width of an element (includes padding).
- The innerHeight() method returns the height of an element (includes padding).

- **outerWidth() and outerHeight() Methods**
- The outerWidth() method returns the width of an element (includes padding and border).
- The outerHeight() method returns the height of an element (includes padding and border).

- The `outerWidth(true)` method returns the width of an element (includes padding, border, and margin).
- The `outerHeight(true)` method returns the height of an element (includes padding, border, and margin).

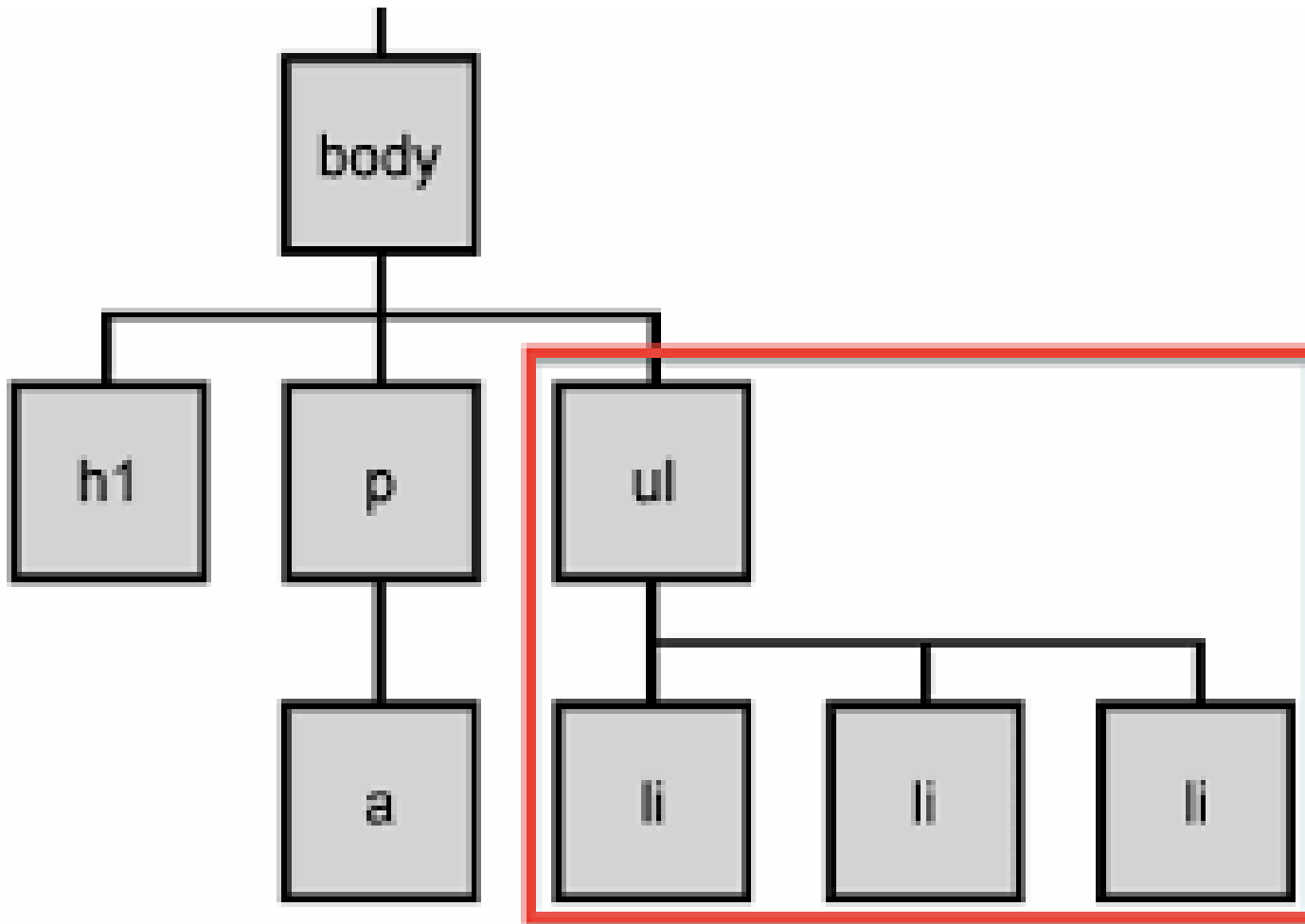
- **More width() and height()**
- The following example returns the width and height of the document (the HTML document) and window (the browser viewport):

- `$("#button").click(function(){
 $("#div1").width(500).height(500);
});`
- The given example sets the width and height of a specified `<div>` element.

Traversing?

- 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.

DOM(family tree)



- We 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.
- **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.

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.

jQuery methods for traversing up the DOM tree are:

- **parent() Method**
- This 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 elements:
- **Example**
- ```
$(document).ready(function(){
 $("span").parent();
});
```

- **parents() Method**

- This 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 `` elements that are `` elements:
- **Example**
- ```
$(document).ready(function(){
 $("span").parents("ul");
});
```

- **parentsUntil() Method**
- This 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");  
});
```

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.

Traversing Down the DOM Tree

- 2 jQuery methods for traversing down the DOM tree are:
- **children()** :This 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");  
});
```

- **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 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("*");  
});
```

Traversing - Siblings

- jQuery methods for traversing sideways in the DOM tree:
 - `siblings()`
 - `next()`
 - `nextAll()`
 - `nextUntil()`
 - `prev()`
 - `prevAll()`
 - `prevUntil()`

- **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");  
});
```

- **next() Method**
- The next() method returns the next sibling element of the selected element.
- The following example returns the next sibling of <h2>:
- **Example**
- ```
$(document).ready(function(){
 $("h2").next();
});
```

- **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();  
});
```

- **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");
});
```

- **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).

# Traversing - Filtering

- **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 do not match, a certain criteria.

- **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();  
});
```


- **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();
});
```

- **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);  
});
```

- **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");
});
```

- **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");  
});
```

- **How do I select elements when I already have a DOM element?**
- If you have a variable containing a DOM element, and want to select elements related to that DOM element, simply wrap it in a jQuery object.
- **var myDomElement = document.getElementById("foo");** *// A plain DOM element.*
- **\$(myDomElement).find("a");** *// Finds all anchors inside the DOM element.*