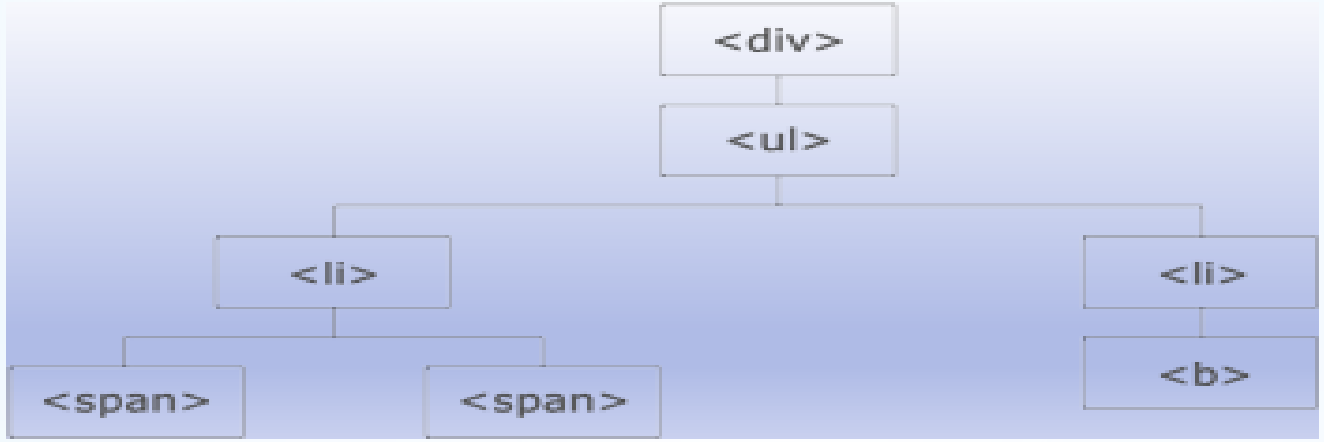
Ahmed Elashry
aelashry@outlook.com

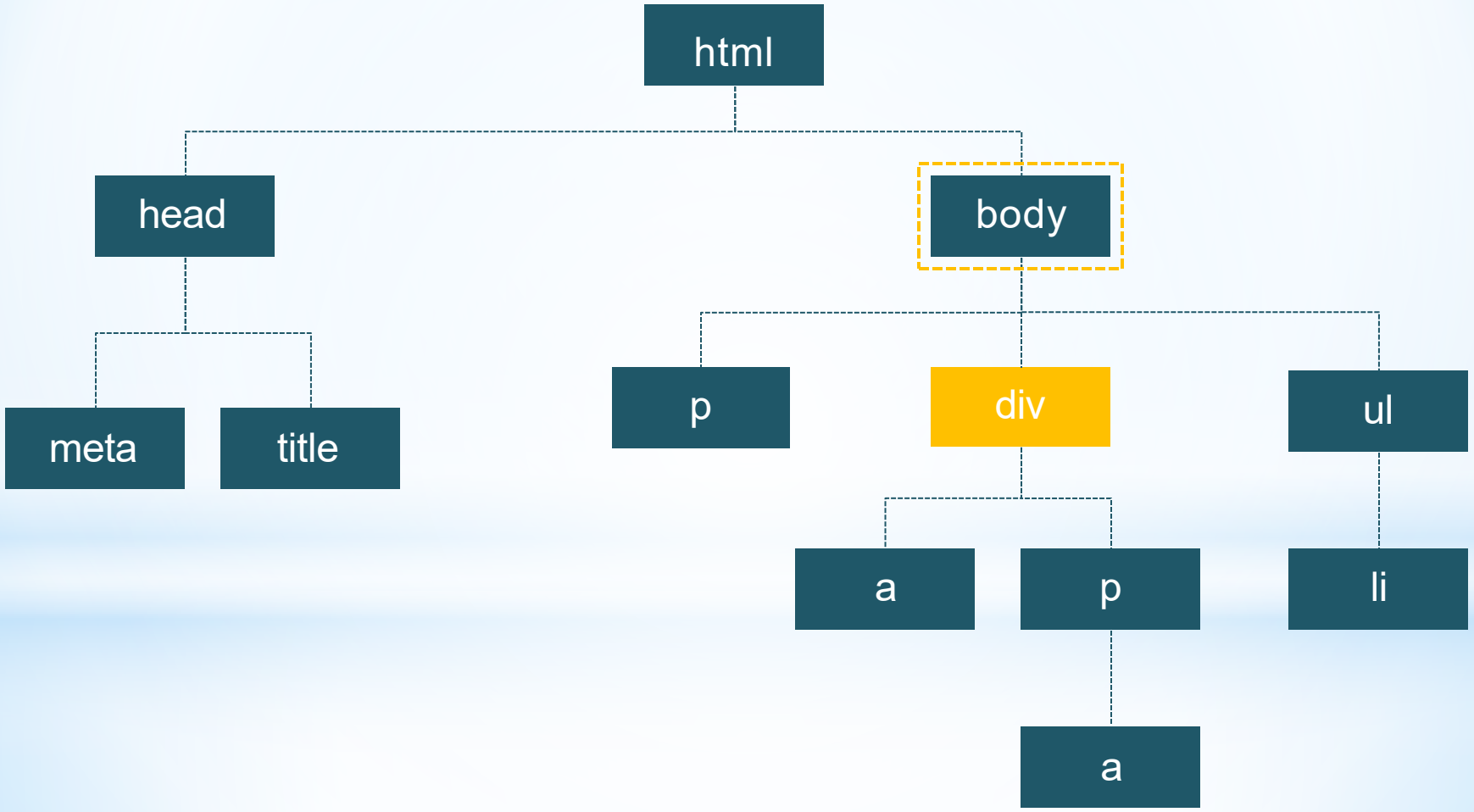
* jQuery traversing, which means "move through", are used to "find" (or select) HTML elements based on their relation to other elements. Start with one selection and move through that selection until you reach the elements you desire.



* jQuery Traversing

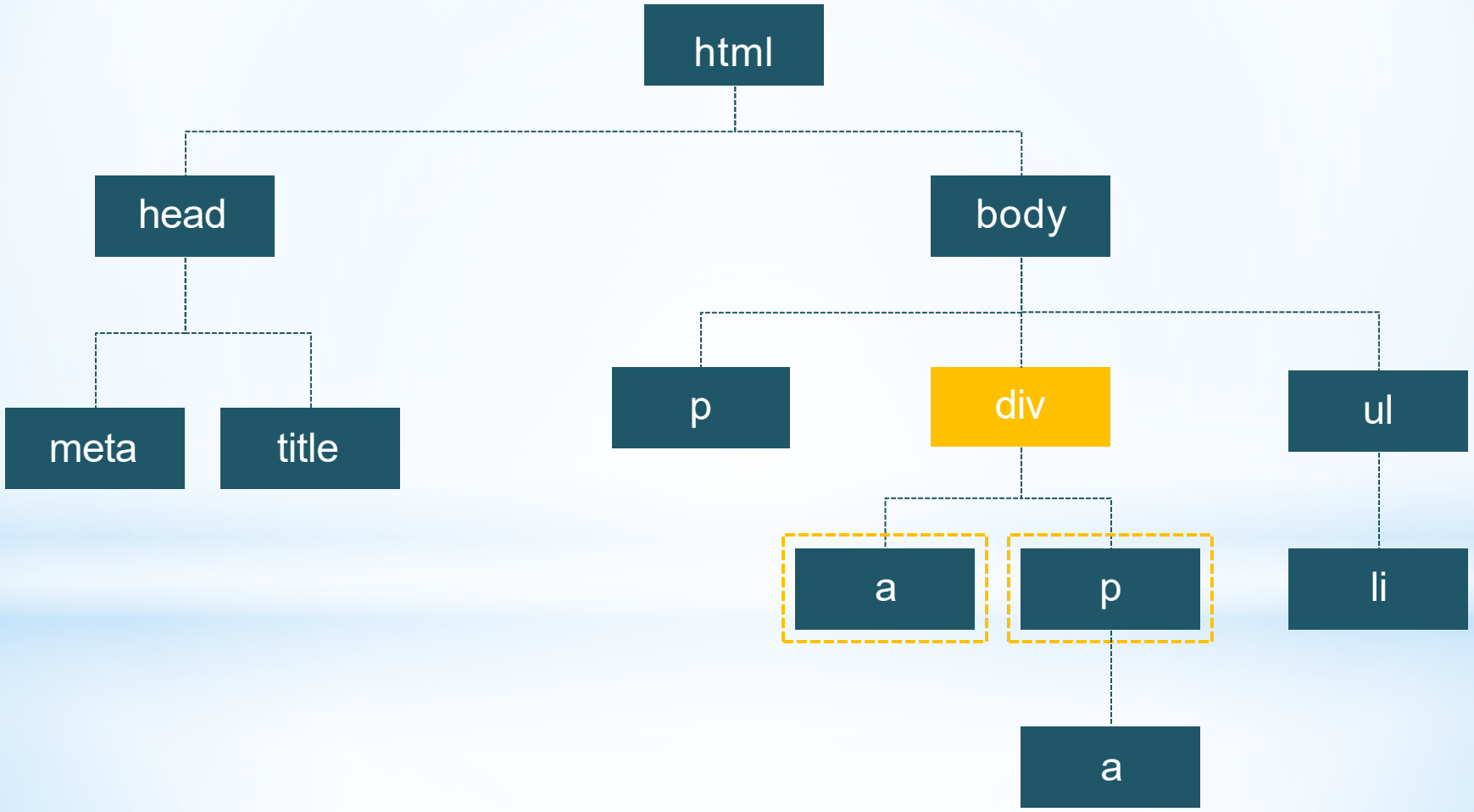
DOM Traversing

```
$("div").parent();
```



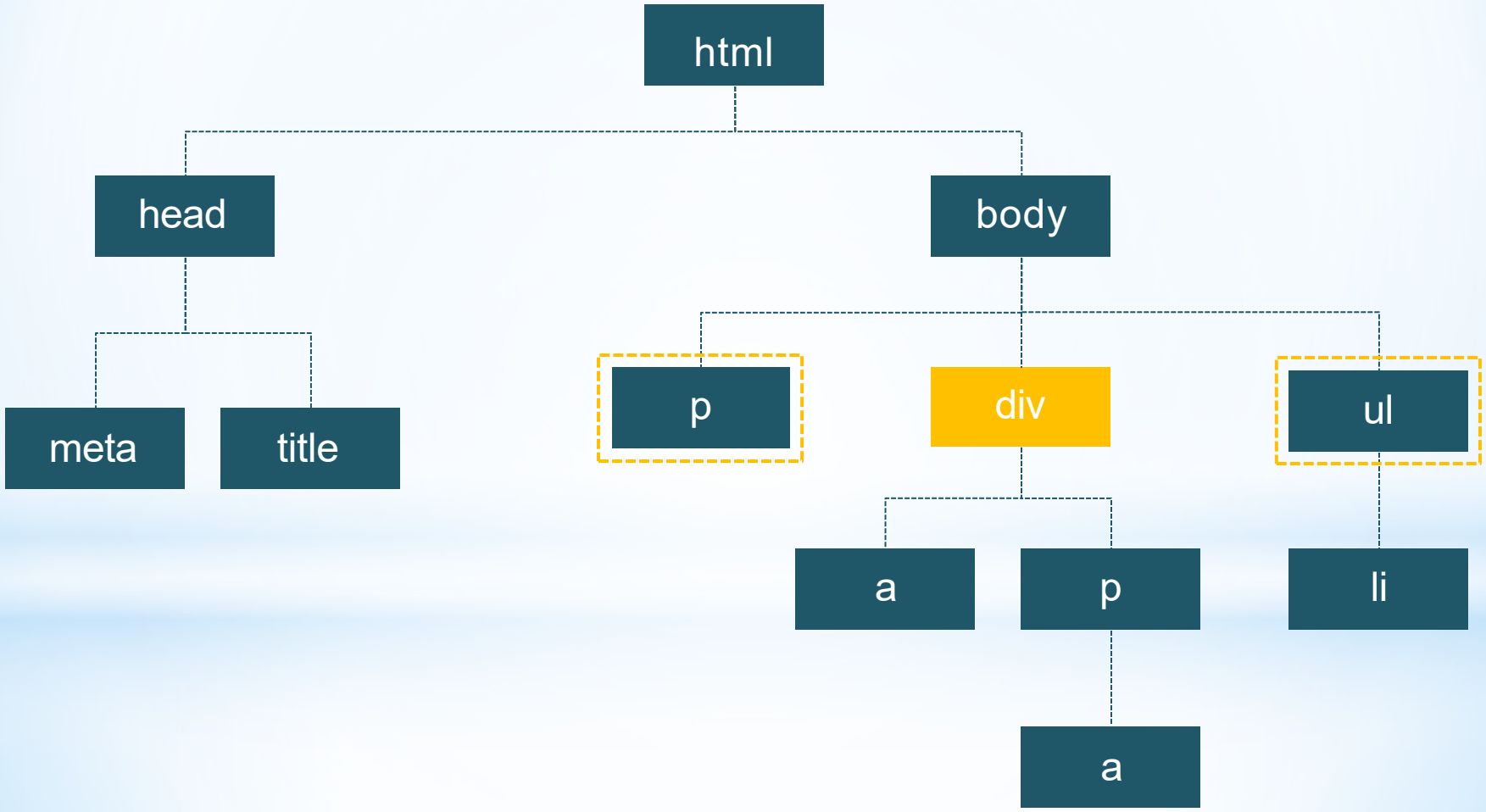
DOM Traversing

```
$("div").children();
```



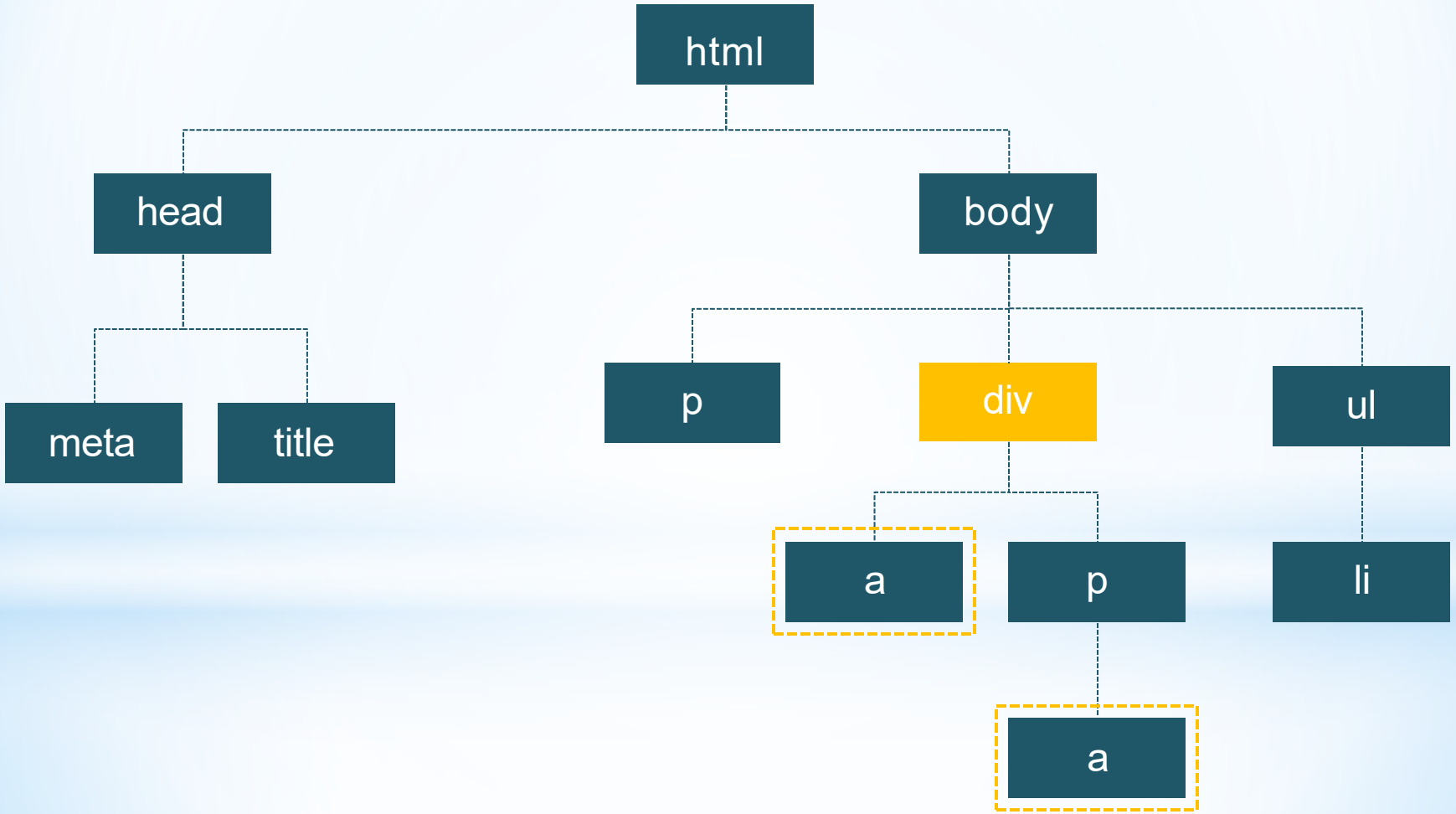
DOM Traversing

```
$("div").siblings();
```



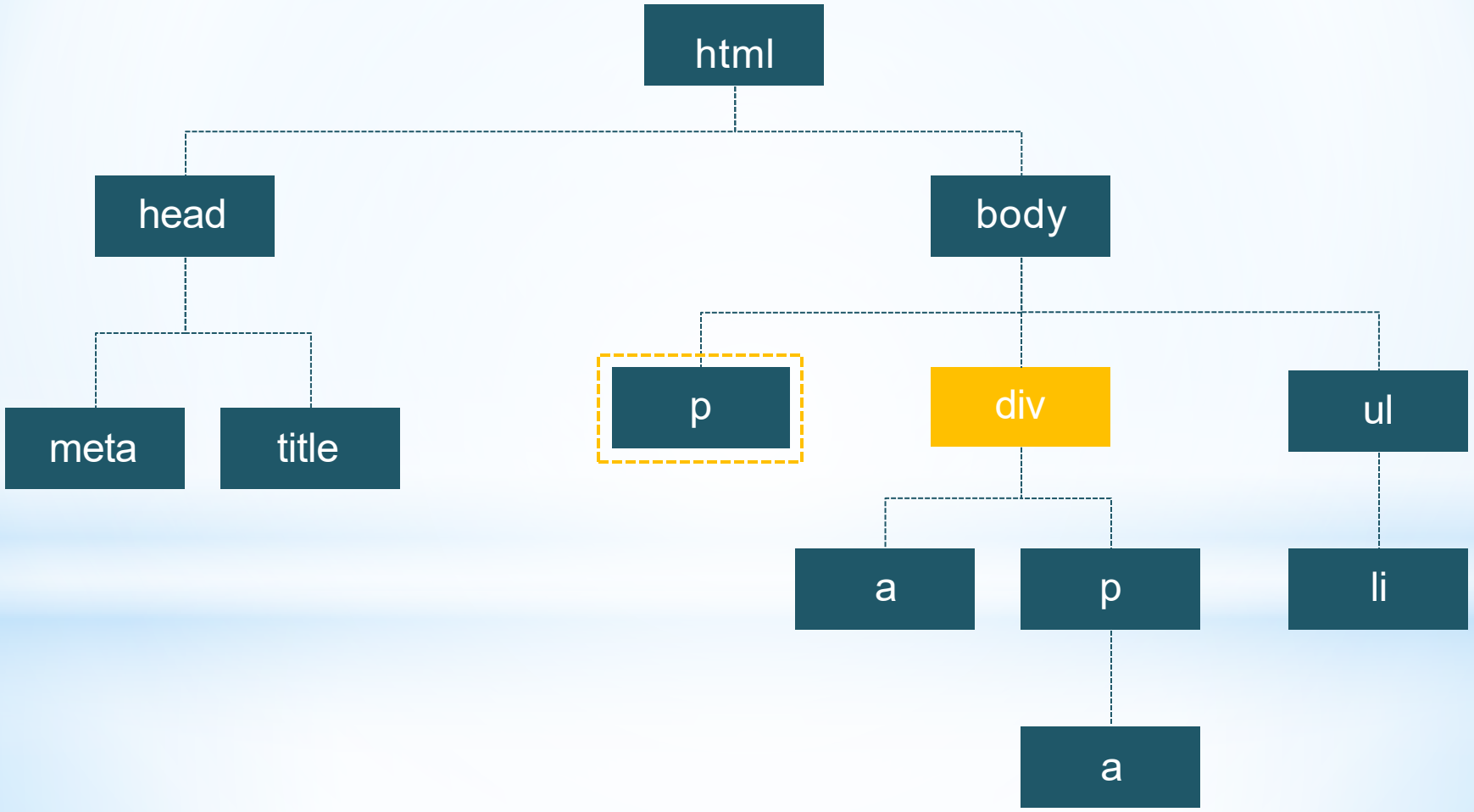
DOM Traversing

```
$("div").find("a");
```



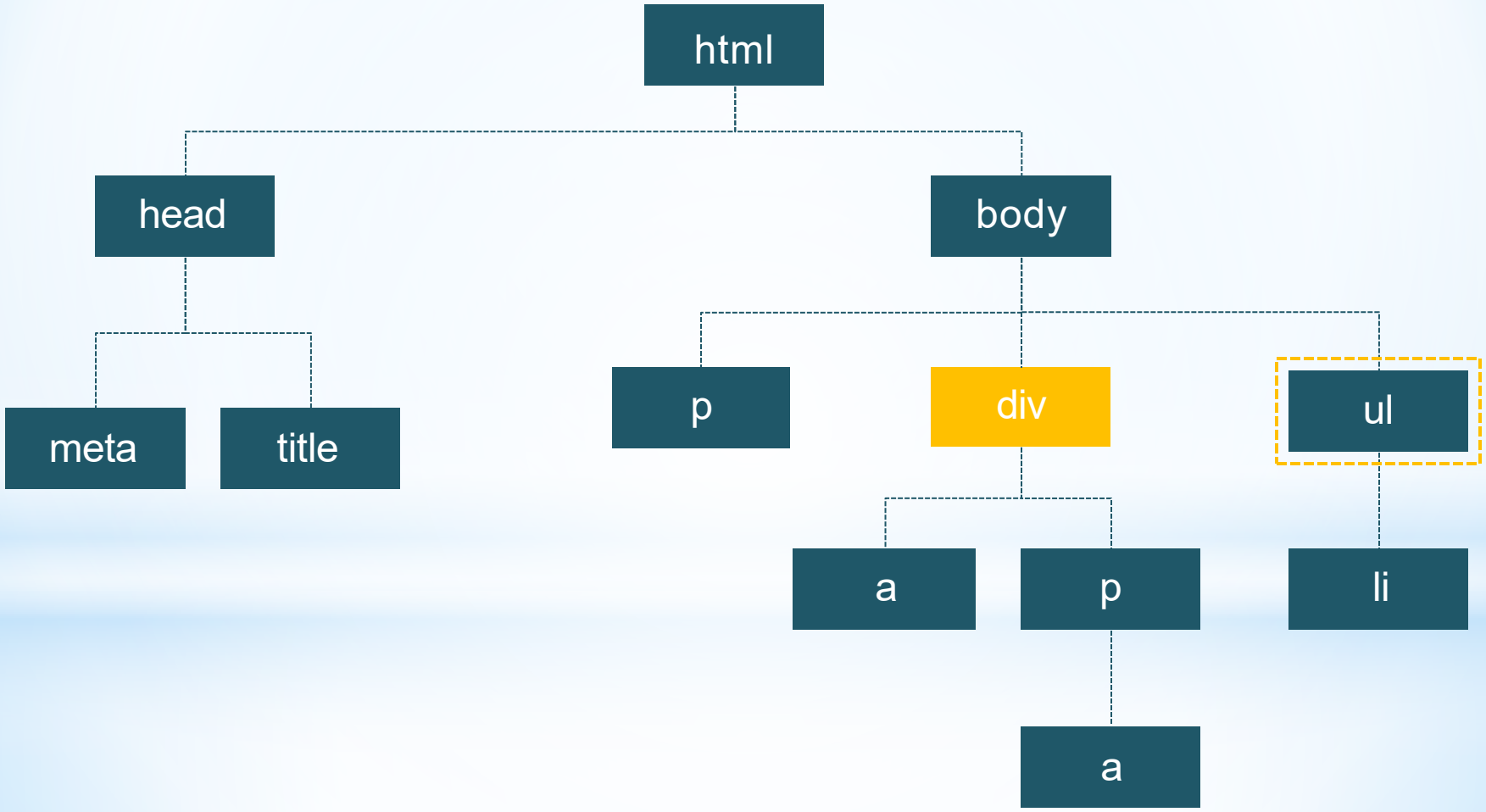
DOM Traversing

```
$("div") .prev();
```



DOM Traversing

```
$("div") .next();
```



Applying Styles

- The **addClass**(classes) method can be used to apply defined style sheets onto all the matched elements. You can specify multiple classes separated by space.
- The **removeClass**(**class**) method removes all or the specified class(es) from the set of matched elements.
- The **hasClass**(class) method returns true if the specified class is present on at least one of the set of matched elements otherwise it returns false.
- The **toggleClass**(**class**) method adds the specified class if it is not present, removes the specified class if it is present.

*Treating with Classes

```
$("div").addClass("my-cls")
```

```
<div class="my-cls">
</div>
```

*Treating with Classes

```
$("div").addClass("my-cls")
        .removeClass("my-cls")
```

```
<div >
```

```
</div>
```

*Treating with Classes

```
$("div").
        addClass("my-cls")
        removeClass("my-cls")
        toggleClass("my-cls")
```

```
<div class="my-cls">
</div>
```

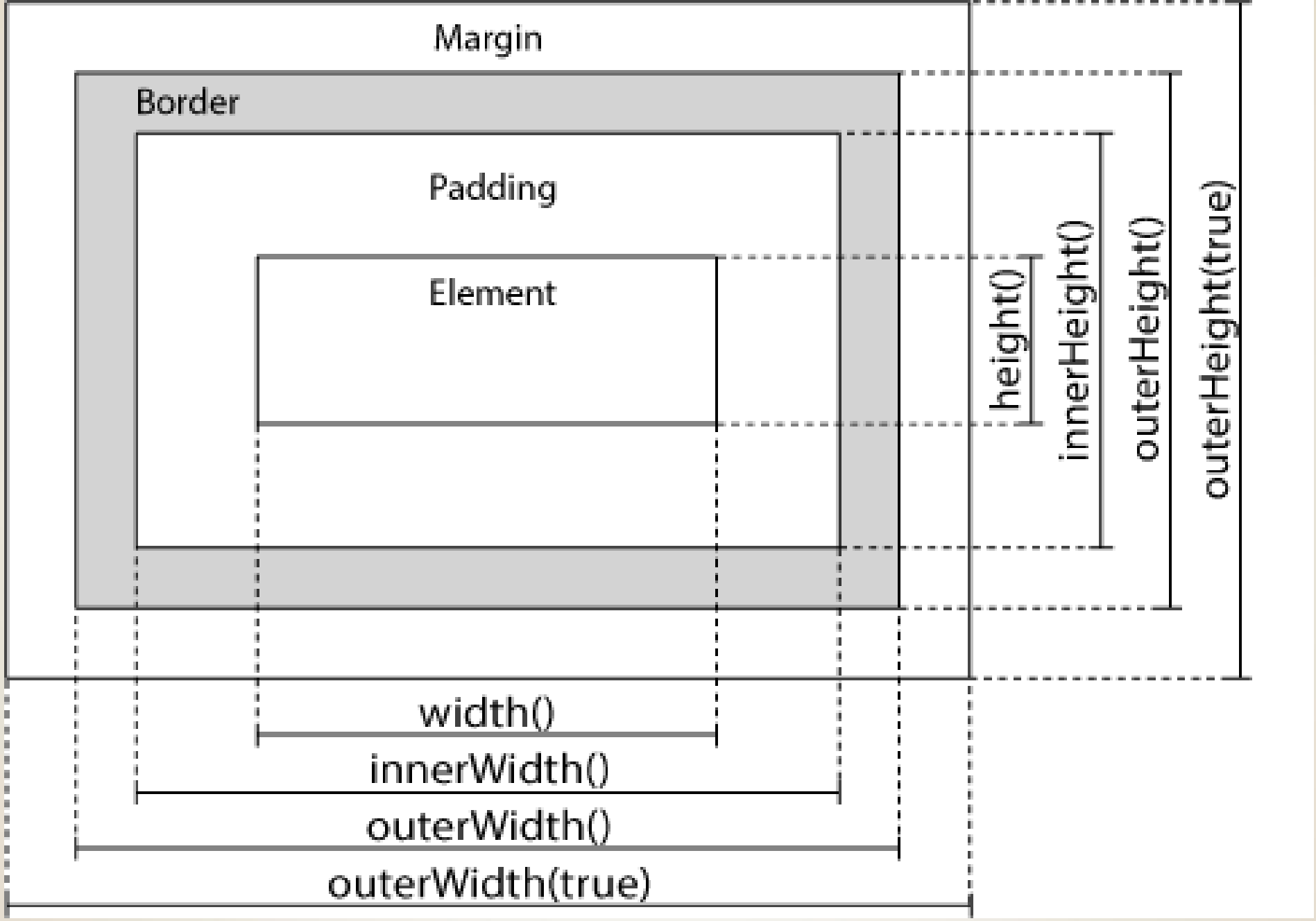
*Treating with Classes

```
$("div").
        addClass("my-cls")
        removeClass("my-cls")
        toggleClass("my-cls")
        toggleClass("my-cls")
```

```
<div >
</div>
```

* The height(val) method sets the CSS height of every matched element.
* The innerHeight( ) method gets the inner height (excludes the border and includes the padding) for the first matched element.
* width(val) Set the CSS width of every matched element.
* outerWidth([margin])Get the outer width (includes the border and padding by default) for the first matched element.
  * margin – This optional argument when set to true the margin of the element will be included in the calculations.
* The position( ) method gets the top and left position of an element relative to its offset parent.

* Applying Styles


Margin
Border
Padding
Element
height()
innerHeight()
outerHeight()
outerHeight(true)
width()
innerWidth()
outerWidth()
outerWidth(true)

*DOM Manipulation

* append(content) Append content to the inside of every matched element.
* appendTo(selector) Append all of the matched elements to another, specified, set of elements
* before(content) Insert content before each of the matched elements.
* empty( ) method removes all child nodes from the set of matched elements.
* remove(expr)Removes all matched elements from the DOM.
* replaceWith(content) Replaces all matched elements with the specified HTML or DOM elements.

*Appending Elements

```
$(".inner").append("<p>append text</p>")
```

```
<div class="container">
    <div class="inner">
        <p> some text</p>
        <p> some text</p>
    </div>
</div>
```

*Appending Elements

```
$(".inner").append("<p>append text</p>")
```

```
<div class="container">
    <div class="inner">
        <p> some text</p>
        <p> some text</p>
        <p> append Text</p>
    </div>
</div>
```

* Add New Elements With append()

The append() methods can take an infinite number of new elements as parameters.

The new elements can be generated with text/HTML with JQuery, or with JavaScript code and DOM elements.

Example:

```
function appendText()
{
   var txt1 = "<p>Text.</p>";
   var txt2 = $("<p></p>").text("Text.");
    var txt3 = document.createElement("p");
    txt3.innerHTML = "Text.";
    $("body").append(txt1, txt2, txt3);
}
```

*JQuery after() and before() Methods:

The jQuery after() method inserts content AFTER the selected HTML elements.

The jQuery before() method inserts content BEFORE the selected HTML elements.

Example:

```
$("img").after("Some text after");

$("img").before("Some text before");
```

*Inserting Elements

```
$(".inner").before("<p>before inner</p>")
```

```
<div class="container">
    <div class="inner">
        <p> some text</p>
        <p> some text</p>
    </div>
</div>
```

*Inserting Elements

```
$(".inner").before("<p>before inner</p>")
```

```
<div class="container">
    <p>before inner</p>
    <div class="inner">
        <p> some text</p>
        <p> some text</p>
    </div>
</div>
```

*Replacing Elements

```
$("#np").replaceWith("<p>I'm the new one</p>")
```

```
<div class="container">
    <p id="np">I warn you, Don't Replace Me </p>
</div>
```

Replacing Elements

```
$("#np").replaceWith("<p>I'm the new one</p>")
```

```
<div class="container">
    <p>I'm the new one</p>
</div>
```

JQuery - Remove Elements:

With jQuery, it is easy to remove existing HTML elements.

Remove Elements/Content:

To remove elements and content, there are mainly two jQuery methods.

remove() - Removes the selected element (and its child elements)

empty() - Removes the child elements from the selected element

Example:

```
$("#div1").remove();
```

*JQUERY EMPTY() METHOD:

The JQuery empty() method removes the child elements of the selected element(s).

Example:

```
$("#div1").empty();
```

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.

Example:

```
$("p").remove(".test");
```

DOM Manipulation Example

jQuery Traversing - Filtering

* The 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.

* 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 not() method returns all elements that do not match the criteria.
* The find() search through all the child elements only.

*//update

*jQuery filter() - not() Method

*jQuery Traversing Methods

* children([selector])Get a set of elements containing all of the unique immediate children of each of the matched set of elements.
* next([selector])Get a set of elements containing the unique next siblings of each of the given set of elements.
* parent([selector])Get the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements.

*Events Handling

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your Web Application.

* Following are the examples events –

* A mouse click
* A web page loading
* Taking mouse over an element
* Submitting an HTML form
* A keystroke on your keyboard, etc.

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

Remember the events

. click(fn)

. dblclick(fn)

. **hover(fnIn , fnOut)**

. submit(fn)

. blur(fn)

. focus(fn)

. change()

. select()

. keyup(fn)

. keydown(fn)

. keypress(fn)

. mouseover (fn)

. mousemove (fn)

. mouseout (fn)

. scroll()

. resize()

. ready()

*Event Methods

- *click() method attaches an event handler function to an HTML element.
- *dblclick() The function is executed when the user double-clicks on the HTML element
- *mouseenter() The function is executed when the mouse pointer enters the HTML element.
- *mouseleave() The function is executed when the mouse pointer leaves the HTML element.
- *mousedown() The function is executed, when the left, middle or right mouse button is pressed down.
- *hover() 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.

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

*Event Methods on()

on

```
$("selector").on(event,[childSelector],[data],  function,  map)
```

Example

```
//Basic Example
$("div").on("click", function(){
        alert("Hi, I'm DIV");
});
```

on

```
$("selector").on(event,[childSelector],[data],  function,  map)
```

Example

```
//Map Example
$("div").on({
                click: function(){
                                alert("You clicked me");
                },
                mouseover: function(){
                                alert("You hovered over me");
                }
});
```

one

```
$("selector").one(event,[data], function)
```

Example

```
//Multiple Events Example
$("div").one("click dbclick", function(){
        alert("Hi, I'm DIV");
});

//Data Example
$("div").one("click", {name: "ahmed"},function(event){
        var name = event.data.name;
        alert("Hi, I'm" + name);
});
```

*preventDefault() Method

* The preventDefault() method prevents the browser from executing the default action.

* You can use the method isDefaultPrevented to know whether this method was ever called (on that event object).

What is AJAX?

* AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.
* the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page.

*jQuery load() Method

*The load() method loads data from a server and puts the returned data into the selected element.

```
[selector].load( URL, [data], [callback] );
```

*The required URL parameter specifies the URL you wish to load.

*The optional data parameter specifies a set of query string key/value pairs to send along with the request.

*The optional callback parameter is the name of a function to be executed after the load() method is completed.

jQuery.ajax(options) Method

* The jQuery.ajax(options) method loads a remote page using an HTTP request.
* Options
  * **Async** A Boolean indicating whether to perform the request asynchronously. The default value is true.
  * **Data** A map or string that is sent to the server with the request.
  * **Success** callback function that is executed if the request succeeds.
  * **Error** A callback function that is executed if the request fails.
  * **Type** A string defining the HTTP method to use for the request (GET or POST). The default value is GET.
  * **url** A string containing the URL to which the request is sent.

```
$("#btn1").click(function (event) {
        alert("test");
        $.ajax({
          type: "GET",
          url: "http://jsonplaceholder.typicode.com/users",
          async: true,
          contentType: "application/json;charset=utf-8",
          dataType: "json",
          success: function (data) {
            $.each(data, function () {
              $("<ul/>")
                .append($("<li/>").val(this.id).text(this.name))
                .appendTo("#div1");
            });
          },
          error: function () {
            alert("Failed!");
          },
        });
      });
```

* jQuery provides a trivially simple interface for doing various kind of amazing effects. jQuery methods allow us to quickly apply commonly used effects with a minimum configuration.

* jQuery - Effects

JQuery Effect Methods

* hide( ) Hides each of the set of matched elements if they are shown.
* show( ) Displays each of the set of matched elements if they are hidden.
* toggle( ) Toggle displaying each of the set of matched elements.
* fadeIn(speed) Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.
* fadeOut(speed) Fade out all matched elements by adjusting their opacity to 0, then setting display to "none" and firing an optional callback after completion.

Show / Hide

* `$("selector").show ([duration][,easing] [,callbackFn])`

Example

```
$("div") .
        hide()

        show(1000)

        toggle(500)

        toggle(500,function({
              $(this).css("background", 'blue');
        }));
```

* The slideUp(duration, callback) is used to slide the matched element up by hiding. It has two optional parameters (duration, callback). The first parameter “duration” is a speed (slow, normal, fast) or you can set a time in milliseconds. The second parameter “callback” is a function, which is executed after the slideUp() method is completed.

* slideUp-slideDown

*slideToggle

* The slideToggle(duration, callback) works between slideUp( ) and slideDown( ) for the selected elements. It has two optional parameters(duration, callback). The first parameter “duration” is a speed (slow, normal, fast) or you can set time in milliseconds. The second parameter “callback” is a function, which is executed after the slideToggle() method is completed.

*Slide

```
$("selector").slideUp([duration][,easing]  [,callbackFn])
```

Example

```
$("div") •  slideUp(500, "swing")
            slideDown("slow")
            slideToggle()
```

Animations

* 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.

*jQuery - noConflict()

*Many JavaScript libraries use $ as a function or variable name, just as jQuery does. In jQuery's case, $ is just an alias for jQuery, so all the functionality is available without using $.

*Run $.noConflict() method to give control of the $ variable back to whichever library first implemented it. This helps us to make sure that jQuery doesn't conflict with the $ object of other libraries.

```
// Import other library
// Import jQuery
 $.noConflict();
 jQuery(document).ready(function($) {
// Code that uses jQuery's $ can follow here.});
// Code that uses other library's $ can follow here.
```