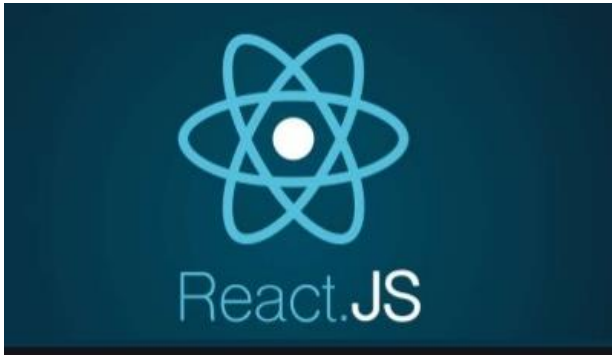

Presentation by Uplatz

Contact Us: https://training.uplatz.com/

Email: info@uplatz.com

Phone:+44 7836 212635

Uplatz

Table Of Contents:

Element Visibility

Parameter	Details
Duration	When passed, the effects of .hide(), .show() and .toggle() are animated; the element(s) will gradually fade in or out.

Overview

$(element).hide() // sets display: none

$(element).show() // sets display to original value

$(element).toggle() // toggles between the two

$(element).is(':visible') // returns true or false

$('element:visible') // matches all elements that are visible

$('element:hidden') // matches all elements that are hidden

$('element').fadeIn(); // display the element

$('element').fadeOut(); // hide the element

$('element').fadeIn(1000); // display the element using timer

$('element').fadeOut(1000); // hide the element using timer

// display the element using timer and a callback function

$('element').fadeIn(1000, function(){

// code to execute

});

// hide the element using timer and a callback function

$('element').fadeOut(1000, function(){

// code to execute });

Toggle possibilities

Simple toggle() case

```
function toggleBasic() {
 $(".target1").toggle();
}
```

With specific duration

```
function toggleDuration() {
 $(".target2").toggle("slow"); // A millisecond duration value is also acceptable
}
```

...and callback

```
function toggleCallback() {
 $(".target3").toggle("slow",function(){alert('now do something');});
}
```

...or with easing and callback.

```
function toggleEasingAndCallback() {
 // You may use jQueryUI as the core only supports linear and swing easings
 $(".target4").toggle("slow","linear",function(){alert('now do something');});
}
```

...or with a variety of options.

```
function toggleWithOptions() {
 $(".target5").toggle(
 {
// See all possible options in: api.jquery.com/toggle/#toggle-options
 duration:1000, // milliseconds
 easing:"linear",
```

```
done:function(){
 alert('now do something');
 } }); }
```

It's also possible to use a slide as animation with **slideToggle()**

```
function toggleSlide() {
 $(".target6").slideToggle(); // Animates from top to bottom, instead of top corner
}
```

...or fade in/out by changing opacity with **fadeToggle()**

```
function toggleFading() {
 $( ".target7" ).fadeToggle("slow")
}
```

...or toggle a class with toggleClass()

```
function toggleClass() {
 $(".target8").toggleClass('active');
}
```

- A common case is to use toggle() in order to show one element while hiding the other (same class)

```
function toggleX() {
 $(".targetX").toggle("slow");
}
```

Append

Parameters	**Details**
content	Possible types: Element, HTML string, text, array, object or even a function returning a string.

Efficient consecutive .append() usage

Starting out:

HTML

```
<table id='my-table' width='960' height='500'></table>
```

JS

```
var data = [
 { type: "Name", content: "John Doe" },
 { type: "Birthdate", content: "01/01/1970" },
 { type: "Salary", content: "$40,000,000" },
 // ...300 more rows...
 { type: "Favorite Flavour", content: "Sour" }
];
```

Appending inside a loop

You just received a big array of data.

- Now it's time to loop through and render it on the page.
- Your first thought may be to do something like this:

```
var i; // <- the current item number
var count = data.length; // <- the total
var row; // <- for holding a reference to our row object
// Loop over the array
for ( i = 0; i < count; ++i ) {
 row = data[ i ];
 // Put the whole row into your table
 $('#my-table').append(
 $('<tr></tr>').append(
 $('<td></td>').html(row.type),
 $('<td></td>').html(row.content)
```

)); }

This is perfectly valid and will render exactly what you'd expect, but...

DO NOT do this.

- Remember those 300+ rows of data?
- Each one will force the browser to re-calculate every element's width, height and positioning values, along with any other styles - unless they are separated by a layout boundary, which unfortunately for this example (as they are descendants of a <table> element), they cannot.
- At small amounts and few columns, this performance penalty will certainly be negligible.
- But we want every millisecond to count.

Better options

- Add to a separate array, append after loop completes

```
/**
 * Repeated DOM traversal (following the tree of elements down until you reach
 * what you're looking for - like our <table>) should also be avoided wherever possible.
 */
// Keep the table cached in a variable then use it until you think it's been removed
var $myTable = $('#my-table');
// To hold our new <tr> jQuery objects
var rowElements = [];
```

```
var count = data.length;
var i;
var row;
// Loop over the array
for ( i = 0; i < count; ++i ) {
 rowElements.push(
 $('<tr></tr>').append(
 $('<td></td>').html(row.type),
 $('<td></td>').html(row.content)
 )); }
// Finally, insert ALL rows at once
$myTable.append(rowElements);
```

- Out of these options, this one relies on jQuery the most.

Using modern Array.* methods

```
var $myTable = $('#my-table');
// Looping with the .map() method
// - This will give us a brand new array based on the result of our callback function
var rowElements = data.map(function ( row ) {
 // Create a row
 var $row = $('<tr></tr>');
 // Create the columns
 var $type = $('<td></td>').html(row.type);
 var $content = $('<td></td>').html(row.content);
 // Add the columns to the row
 $row.append($type, $content);
// Add to the newly-generated array
```

```
return $row;
});
// Finally, put ALL of the rows into your table
$myTable.append(rowElements);
```

- Functionally equivalent to the one before it, only easier to read.

Using strings of HTML (instead of jQuery built-in methods)

```
// ...
var rowElements = data.map(function ( row ) {
 var rowHTML = '<tr><td>';
 rowHTML += row.type;
 rowHTML += '</td><td>';
```

```
rowHTML += row.content;
 rowHTML += '</td></tr>';
 return rowHTML;
});
```

// Using .join('') here combines all the separate strings into one

$myTable.append(rowElements.join(''));

- Perfectly valid but again, not recommended.
- This forces jQuery to parse a very large amount of text at once and is not necessary.
- jQuery is very good at what it does when used correctly.

Manually create elements, append to document fragment

```
var $myTable = $(document.getElementById('my-table'));
/**
 * Create a document fragment to hold our columns
 * - after appending this to each row, it empties itself
 * so we can re-use it in the next iteration.
 */
var colFragment = document.createDocumentFragment();
/**
 * Loop over the array using .reduce() this time.
 * We get a nice, tidy output without any side-effects.
 * - In this example, the result will be a
 * document fragment holding all the <tr> elements
 */
```

```
var rowFragment = data.reduce(function ( fragment, row ) {
 // Create a row
 var rowEl = document.createElement('tr');
 // Create the columns and the inner text nodes
 var typeEl = document.createElement('td');
 var typeText = document.createTextNode(row.type);
 typeEl.appendChild(typeText);
 var contentEl = document.createElement('td');
 var contentText = document.createTextNode(row.content);
 contentEl.appendChild(contentText);
 // Add the columns to the column fragment
 // - this would be useful if columns were iterated over separately
```

```
// but in this example it's just for show and tell.
 colFragment.appendChild(typeEl);
 colFragment.appendChild(contentEl);
rowEl.appendChild(colFragment);
 // Add rowEl to fragment - this acts as a temporary buffer to
 // accumulate multiple DOM nodes before bulk insertion
 fragment.appendChild(rowEl);
 return fragment;
}, document.createDocumentFragment());
// Now dump the whole fragment into your table
$myTable.append(rowFragment);
```

Document.createDocumentFragment()

- Creates a new empty DocumentFragment into which DOM nodes can be added to build an offscreen DOM tree.

Syntax

```
var fragment = document.createDocumentFragment();
```

Value

- A newly created, empty, DocumentFragment object, which is ready to have nodes inserted into it.

Usage notes

- DocumentFragments are DOM Node objects which are never part of the main DOM tree.

- The usual use case is to create the document fragment, append elements to the document fragment and then append the document fragment to the DOM tree.
- In the DOM tree, the document fragment is replaced by all its children.
- Since the document fragment is in memory and not part of the main DOM tree, appending children to it does not cause page reflow (computation of element's position and geometry).
- Historically, using document fragments could result in better performance.
- You can also use the DocumentFragment constructor to create a new fragment:

let fragment = new DocumentFragment();

Example

- This example creates a list of major web browsers in a DocumentFragment, then adds the new DOM subtree to the document to be displayed.

HTML

```
<ul id="ul">
</ul>
```

JavaScript

```
var element  = document.getElementById('ul'); // assuming ul exists
var fragment = document.createDocumentFragment();
var browsers = ['Firefox', 'Chrome', 'Opera',
    'Safari', 'Internet Explorer'];
browsers.forEach(function(browser) {
```

```
var li = document.createElement('li');
    li.textContent = browser;
    fragment.appendChild(li);
});
element.appendChild(fragment);
```

Result

Firefox

Chrome

Opera

Safari

Internet Explorer

Document.createTextNode()

Syntax

var text = document.createTextNode(data);

text is a Text node.

- data is a string containing the data to be put in the text node.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>createTextNode example</title>
<script>
function addTextNode(text) {
  var newtext = document.createTextNode(text),
      p1 = document.getElementById("p1");
p1.appendChild(newtext);
}
```

```
</script>
</head>
<body>
  <button onclick="addTextNode('YES!
');">YES!</button>
  <button onclick="addTextNode('NO!
');">NO!</button>
  <button onclick="addTextNode('WE CAN! ');">WE
CAN!</button>
 <hr />
 <p id="p1">First line of paragraph.</p>
</body>
</html>
```

jQuery append

HTML

```
<p>This is a nice </p>
<p>I like </p>
<ul>
 <li>List item 1</li>
 <li>List item 2</li>
 <li>List item 3</li>
</ul>
<button id="btn-1">Append text</button>
<button id="btn-2">Append list item</button>
```

Script

```
$("#btn-1").click(function(){
 $("p").append(" <b>Book</b>.");
```

```
});
$("#btn-2").click(function(){
 $("ul").append("<li>Appended list item</li>");
});
});
```

Appending an element to a container

Solution 1:

```
$('#parent').append($('#child'));
```

Solution 2:

```
$('#child').appendTo($('#parent'));
```

- Both solutions are appending the element #child (adding at the end) to the element #parent.

Before:

```
<div id="parent">
```

other content

** **

** **

** **

After:

other content

- **Note**: When you append content that already exsists in the document, this content will be removed from its original parent container and appended to the new parent container.

- So you can't use .append() or .appendTo() to clone an element.

Prepend

Prepending an element to a container

Solution 1:

$('#parent').prepend($('#child'));

Solution 2:

$('#child').prependTo($('#parent'));

- Both solutions are prepending the element #child (adding at the beginning) to the element #parent.

Before:

```
<div id="parent">
 <span>other content</span>
</div>
```

** **

** **

After:

** **

** **

** **

other content

** **

Prepend method

- prepend() - Insert content, specified by the parameter, to the beginning of each element in the set of matched elements.

1. prepend(content [, content])

// with html string

```
jQuery('#parent').prepend('<span>child</span>');
// or you can use jQuery object
jQuery('#parent').prepend($('#child'));
// or you can use comma separated multiple elements to prepend
jQuery('#parent').prepend($('#child1'),$('#child2'));
```

prepend(function)

- JQuery version: 1.4 onwards you can use callback function as the argument. Where you can get arguments as index position of the element in the set and the old HTML value of the element.
- Within the function, this refers to the current element in the set.

```
jQuery('#parent').prepend(function(i,oldHTML){
```

// return the value to be prepend

return 'child';

)}

Getting and setting width and height of an element

Getting and setting width and height (ignoring border):

Get width and height:

var width = $('#target-element').width();

var height = $('#target-element').height();

Set width and height:

$('#target-element').width(50);

$('#target-element').height(100);

Getting and setting innerWidth and innerHeight

(ignoring padding and border)

Get width and height:

```
var width = $('#target-element').innerWidth();
var height = $('#target-element').innerHeight();
```

Set width and height:

```
$('#target-element').innerWidth(50);
$('#target-element').innerHeight(100);
```

Getting and setting outerWidth and outerHeight (including padding and border)

Get width and height (excluding margin):

```
var width = $('#target-element').outerWidth();
var height = $('#target-element').outerHeight();
```

Get width and height (including margin):

```
var width = $('#target-element').outerWidth(true);
var height = $('#target-element').outerHeight(true);
```

Set width and height:

```
$('#target-element').outerWidth(50);
$('#target-element').outerHeight(100);
```

jQuery .animate() Method

Parameter	Details
properties	An object of CSS properties and values that the animation will move toward
duration (default: 400)	A string or number determining how long the animation will run
easing (default: swing)	A string indicating which easing function to use for the transition
complete	A function to call once the animation is complete, called once per matched element.
start	specifies a function to be executed when the animation begins.

step specifies a function to be executed for each step in the animation.

queue a Boolean value specifying whether or not to place the animation in the effects queue.

progress specifies a function to be executed after each step in the animation.

done specifies a function to be executed when the animation ends.

fail specifies a function to be executed if the animation fails to complete.

specialEasing a map of one or more CSS properties from the styles parameter, and their corresponding easing functions.

always specifies a function to be executed if the animation stops without completing.

Animation with callback

- Sometimes we need to change words position from one place to another or reduce size of the words and change the color of words automatically to improve the attraction of our website or web apps.
- JQuery helps a lot with this concept using fadeIn(), hide(), slideDown() but its functionality are limited and it only done the specific task which assign to it.
- Jquery fix this problem by providing an amazing and flexible method called .animate().
- This method allows to set custom animations which is used css properties that give permission to fly over borders.
- for example if we give css style property as width:200; and current position of the DOM element is 50, animate method reduce current

- position value from given css value and animate that element to 150.
- But we don't need to bother about this part because animation engine will handle it.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.
1/jquery.min.js"></script>
<script>
 $("#btn1").click(function(){
 $("#box").animate({width: "200px"});
 });
</script>
<button id="btn1">Animate Width</button>
<div id="box"
style="background:#98bf21;height:100px;width:1(
margin:6px;"></div>
```

List of css style properties that allow in .animate() method.

backgroundPositionX, backgroundPositionY, borderWidth, borderBottomWidth, borderLeftWidth,

borderRightWidth, borderTopWidth, borderSpacing, margin, marginBottom, marginLeft, marginRight,

marginTop, outlineWidth, padding, paddingBottom, paddingLeft, paddingRight, paddingTop, height,

width, maxHeight, maxWidth, minHeight, minWidth, fontSize, bottom, left, right, top, letterSpacing,

wordSpacing, lineHeight, textIndent,

Speed specified in .animate() method.

milliseconds (Ex: 100, 1000, 5000, etc.),

"slow",

"fast"

Easing specified in .animate() method.

"swing"

"linear"

Here is some examples with complex animation options.

Eg 1:

```
$( "#book" ).animate({
 width: [ "toggle", "swing" ],
 height: [ "toggle", "swing" ],
 opacity: "toggle"
}, 5000, "linear", function() {
 $( this ).after( "<div>Animation complete.</div>" );
});
```

Eg 2:

```
$("#box").animate({
height: "300px",
width: "300px"
}, {
duration: 5000,
easing: "linear",
complete: function(){
$(this).after("<p>Animation is complete!</p>");
}
});
```

jQuery Deferred objects and Promises

- jQuery promises are a clever way of chaining together asynchronous operations in a building-block manner.
- This replaces old-school nesting of callbacks, which are not so easily reorganised.

jQuery ajax() success, error VS .done(), .fail()

- **success and Error** : A success callback that gets invoked upon successful completion of an Ajax request.
- A failure callback that gets invoked in case there is any error while making the request.

Example:

```
$.ajax({
 url: 'URL',
 type: 'POST',
```

data: yourData,

datatype: 'json',

success: function (data) { successFunction(data); },

error: function (jqXHR, textStatus, errorThrown) { errorFunction(); }

});

.done() and .fail() :

- .ajax().done(function(data, textStatus, jqXHR){}); Replaces method .success() which was deprecated in jQuery
- This is an alternative construct for the success callback function above.

.ajax().fail(function(jqXHR, textStatus, errorThrown){}); Replaces method .error() which was deprecated in

jQuery

- 1.8.This is an alternative construct for the complete callback function above.

Example:

```
$.ajax({
 url: 'URL',
 type: 'POST',
 data: yourData,
 datatype: 'json'
})
```

- .done(function (data) { successFunction(data); })
- .fail(function (jqXHR, textStatus, errorThrown) { serrorFunction(); });

Basic promise creation

- Here is a very simple example of a function that "promises to proceed when a given time elapses".
- It does that by creating a new Deferred object, that is resolved later and returning the Deferred's promise:

```
function waitPromise(milliseconds){
 // Create a new Deferred object using the jQuery static method
 var def = $.Deferred();
 // Do some asynchronous work - in this case a simple timer
 setTimeout(function(){
```

// Work completed... resolve the deferred, so it's promise will proceed

```
 def.resolve();
 }, milliseconds);
 // Immediately return a "promise to proceed when the wait time ends"
 return def.promise();
}
```

And use like this:

```
waitPromise(2000).then(function(){
 console.log("I have waited long enough");
});
```

Ajax

Parameter Details

url Specifies the URL to which the request will be sent settings an object containing numerous values that affect the behavior of the request

type The HTTP method to be used for the request

data Data to be sent by the request

success A callback function to be called if the request succeeds

error A callback to handle error

statusCode An object of numeric HTTP codes and functions to be called when the response has the corresponding code

dataType The type of data that you're expecting back from the server

contentType Content type of the data to sent to the server.

Default is "application/x-www-form-urlencoded;

charset=UTF-8“ context Specifies the context to be used inside callbacks, usually this which refers to the current target.

Handling HTTP Response Codes with $.ajax()

- In addition to .done, .fail and .always promise callbacks, which are triggered based on whether the request was successful or not, there is the option to trigger a function when a specific HTTP Status Code is returned from the server.

This can be done using the statusCode parameter.

```
$.ajax({
 type: {POST or GET or PUT etc.},
 url: {server.url},
 data: {someData: true},
```

```
statusCode: {
 404: function(responseObject, textStatus, jqXHR) {
 // No content found (404)
 // This code will be executed if the server returns a 404 response
 },
 503: function(responseObject, textStatus, errorThrown) {
 // Service Unavailable (503)
 // This code will be executed if the server returns a 503 response
 } } })
.done(function(data){
 alert(data);
})
```

```
fail(function(jqXHR, textStatus){
 alert('Something went wrong: ' + textStatus);
})
.always(function(jqXHR, textStatus) {
 alert('Ajax request was finished')
});
```

As official jQuery documentation states:

- If the request is successful, the status code functions take the same parameters as the success callback;
- if it results in an error (including 3xx redirect), they take the same parameters as the error callback

Using Ajax to Submit a Form

- Sometimes you may have a form and want to submit it using ajax.

Suppose you have this simple form -

```
<form id="ajax_form" action="form_action.php">
 <label for="name">Name :</label>
 <input name="name" id="name" type="text" />
 <label for="name">Email :</label>
 <input name="email" id="email" type="text" />
 <input type="submit" value="Submit" />
</form>
```

- The following jQuery code can be used (within a $(document).ready call) -

```
$('#ajax_form').submit(function(event){
 event.preventDefault();
 var $form = $(this);
 $.ajax({
 type: 'POST',
```

```
url: $form.attr('action'),
 data: $form.serialize(),
 success: function(data) {
 // Do something with the response
 },error: function(error) {
 // Do something with the error
 } });
});
```

Explanation

var $form = $(this) - the form, cached for reuse

$('#ajax_form').submit(function(event){

- \- When the form with ID "ajax_form" is submitted run this
- function and pass the event as a parameter.

event.preventDefault();

- Prevent the form from submitting normally (Alternatively we can use return
- false after the ajax({}); statement, which will have the same effect)
- **url**: $form.attr('action'), - Get the value of the form's "action" attribute and use it for the "url" property.
- data: $form.serialize(), - Converts the inputs within the form into a string suitable for sending to the

server.

- In this case it will return something like "name=Bob&email=bob@bobsemailaddress.com"

All in one examples

Ajax Get:

Solution 1:

```
$.get('url.html', function(data){
 $('#update-box').html(data);
});
```

Solution 2:

```
$.ajax({
 type: 'GET',
 url: 'url.php',
}).done(function(data){
 $('#update-box').html(data);
}).fail(function(jqXHR, textStatus){
 alert('Error occurred: ' + textStatus);
});
```

Ajax Load: Another ajax get method created for simplcity

$('#update-box').load('url.html');

- .load can also be called with additional data. The data part can be provided as string or object.

$('#update-box').load('url.php', {data: "something"});

$('#update-box').load('url.php', "data=something");

- If .load is called with a callback method, the request to the server will be a post

```
$('#update-box').load('url.php', {data: "something"},
function(resolve){
 //do something
});
```

Ajax Post:

Solution 1:

```
$.post('url.php',
 {date1Name: data1Value, date2Name: data2Value},
//data to be posted
 function(data){
 $('#update-box').html(data);
 }
);
```

Solution 2:

```
$.ajax({
 type: 'Post',
 url: 'url.php',
 data: {date1Name: data1Value, date2Name:
data2Value} //data to be posted
}).done(function(data){
```

```
$('#update-box').html(data);
}).fail(function(jqXHR, textStatus){
 alert('Error occurred: ' + textStatus);
});
```

Ajax Post JSON:

```
var postData = {
 Name: name,
 Address: address,
 Phone: phone
};
$.ajax({
 type: "POST",
 url: "url.php",
```

```
dataType: "json",
 data: JSON.stringfy(postData),
 success: function (data) {
 //here variable data is in JSON format
 }
});
```

Ajax Get JSON:

Solution 1:

```
$.getJSON('url.php', function(data){
 //here variable data is in JSON format
});
```

Solution 2:

```
$.ajax({
 type: "Get",
 url: "url.php",
 dataType: "json",
 data: JSON.stringfy(postData),
 success: function (data) {
 //here variable data is in JSON format
 },
 error: function(jqXHR, textStatus){
 alert('Error occurred: ' + textStatus);
 }
 });
```

Summary:

- Element Visibility
- Append
- Prepend
- Getting and setting width and height of an element
- jQuery Deferred objects and Promises
- Ajax

Thank You.....

If you have any quries please write to info@uplatz.com".

Uplatz