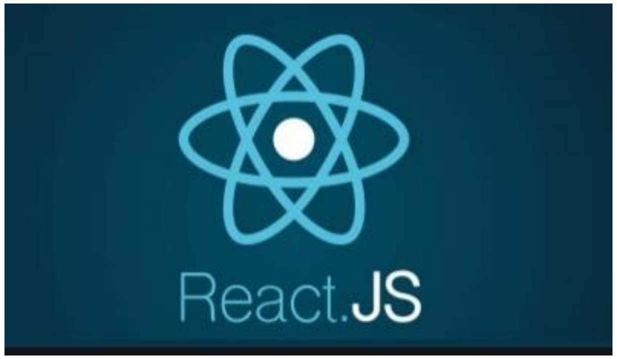

Presentation by Uplatz

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

Email: info@uplatz.com

Phone:+44 7836 212635

Uplatz

Table Of Contents:

Background information - Event propagation

- Delegated events are only possible because of event propagation (often called event bubbling).
- Any time an event is fired, it will bubble all the way up (to the document root).
- They delegate the handling of an event to a non-changing ancestor element, hence the name "delegated" events.
- So in example above, clicking <a> element link will trigger 'click' event in these elements in this order:

a

li

ul

body

html

document root

Solution

- Knowing what event bubbling does, we can catch one of the wanted events which are propagating up through our HTML.
- A good place for catching it in this example is the <ul> element, as that element does is not dynamic:

```
$('ul').on('click', 'a', function () {
 console.log(this.href); // jQuery binds the event function to the targeted DOM element
 // this way `this` refers to the anchor and not to the list
 // Whatever you want to do when link is clicked
});
```

In above:

- We have 'ul' which is the recipient of this event listener
- The first parameter ('click') defines which events we are trying to detect.
- The second parameter ('a') is used to declare where the event needs to originate from (of all child elements under this event listener's recipient, ul).
- Lastly, the third parameter is the code that is run if first and second parameters' requirements are fulfilled.

In detail how solution works

1. User clicks <a> element
2. That triggers click event on <a> element.
3. The event start bubbling up towards document root

4. The event bubbles first to the <li> element and then to the <ul> element.

5. The event listener is run as the <ul> element has the event listener attached.

6. The event listener first detects the triggering event. The bubbling event is 'click' and the listener has 'click', it is a pass.

7. The listener checks tries to match the second parameter ('a') to each item in the bubble chain.

As the last item in the chain is an 'a' this matches the filter and this is a pass too.

8. The code in third parameter is run using the matched item as it's this.

If the function does not include a call to stopPropagation(), the event will continue propagating upwards towards the root (docume

- **Note:** If a suitable non-changing ancestor is not available/convenient, you should use document.
- As a habit do not use 'body' for the following reasons:
- body has a bug, to do with styling, that can mean mouse events do not bubble to it.
- This is browser dependant and can happen when the calculated body height is 0 (e.g. when all child elements have absolute positions).
- Mouse events always bubble to document.

document always exists to your script, so you can attach delegated handlers to document outside of a DOMready handler and be certain they will still work

Attach and Detach Event Handlers

Attach an Event Handler

- jQuery has the event API .on().
- This way any standard javascript event or custom event can be bound on the currently selected jQuery element.
- There are shortcuts such as .click(), but .on() gives you more options.

HTML

```
<button id="foo">bar</button>
```

jQuery

```
$( "#foo" ).on( "click", function() {
 console.log( $( this ).text() ); //bar
});
```

Detach an Event Handler

- Naturally you have the possibility to detach events from your jQuery objects too.

You do so by using .off(events [, selector] [, handler]).

HTML

```
<button id="hello">hello</button>
```

jQuery

```
$('#hello').on('click', function(){
 console.log('hello world!');
 $(this).off();
});
```

- When clicking the button $(this) will refer to the current jQuery object and will remove all attached event handlers from it.

You can also specify which event handler should be removed.

jQuery

```
$('#hello').on('click', function(){
 console.log('hello world!');
 $(this).off('click');
});
$('#hello').on('mouseenter', function(){
 console.log('you are about to click');
});
```

In this case the mouseenter event will still function after clicking.

Switching specific events on and of via jQuery. (Named Listeners)

➢ Sometimes you want to switch off all previously registered listeners.

```
//Adding a normal click handler
$(document).on("click",function(){
 console.log("Document Clicked 1")
});
//Adding another click handler
$(document).on("click",function(){
 console.log("Document Clicked 2")
});
//Removing all registered handlers.
$(document).off("click")
```

- An issue with this method is that ALL listeners binded on document by other plugins etc would also be removed.
- More often than not, we want to detach all listeners attached only by us.
- To achieve this, we can bind named listeners as,

```
//Add named event listener.
$(document).on("click.mymodule",function(){
 console.log("Document Clicked 1")
});
$(document).on("click.mymodule",function(){
 console.log("Document Clicked 2")
});
//Remove named event listener.
```

$(document).off("click.mymodule");

- This ensures that any other click listener is not inadvertently modified.

originalEvent

- Sometimes there will be properties that aren't available in jQuery event.
- To access the underlying properties use

Event.originalEvent

Get Scroll Direction

$(document).on("wheel",function(e){

console.log(e.originalEvent.deltaY)

// Returns a value between -100 and 100 depending on the direction you are scrolling

})

- Events for repeating elements without using ID's

Problem

- There is a series of repeating elements in page that you need to know which one an event occurred on to do something with that specific instance.

Solution

Give all common elements a common class

- Apply event listener to a class.
- this inside event handler is the matching selector element the event occurred on
- Traverse to outer most repeating container for that instance by starting at this
- Use find() within that container to isolate other elements specific to that instance

HTML

```
<div class="item-wrapper" data-item_id="346">
 <div class="item"><span
class="person">Fred</span></div>
 <div class="item-toolbar">
 <button class="delete">Delete</button>
 </div>
</div>
<div class="item-wrapper" data-item_id="393">
 <div clss="item"><span
class="person">Wilma</span></div>
 <div class="item-toolbar">
 <button class="delete">Delete</button>
 </div>
</div>
```

jQuery

```
$(function() {
 $('.delete').on('click', function() {
```

```
 // "this" is element event occurred on
 var $btn = $(this);
 // traverse to wrapper container
 var $itemWrap = $btn.closest('.item-wrapper');
 // look within wrapper to get person for this button
instance
 var person = $itemWrap.find('.person').text();
 // send delete to server and remove from page on
success of ajax
```

```
$.post('url/string', { id:
$itemWrap.data('item_id')}).done(function(response) {
$itemWrap.remove()
}).fail(function() {
alert('Ooops, not deleted at server');
});
}); });
```

Document Loading Event .load()

- If you want your script to wait until a certain resource was loaded, such as an image or a PDF you can use .load(), which is a shortcut for shortcut for .on("load", handler).

HTML

```
<img src="image.jpeg" alt="image" id="image">
```

```
jQuery
$( "#image" ).load(function() {
 // run script
});
```

DOM Manipulation

Creating DOM elements

- The jQuery function (usually aliased as $) can be used both to select elements and to create new elements.

var myLink = $('<a href="http://stackexchange.com"></a>');

- You can optionally pass a second argument with element attributes:

var myLink = $('<a>', { 'href': 'http://stackexchange.com' });

- '<a>' --> The first argument specifies the type of DOM element you want to create.
- In this example it's an anchor but could be anything on this list. See the specification for a reference of the a element.
- **{ 'href': 'http://stackexchange.com' }** --> the second argument is a JavaScript Object containing attribute
- name/value pairs.
- the 'name':'value' pairs will appear between the < > of the first argument, for example <a name:value> which for our example would be
- **<a href="http://stackexchange.com"></a>**

Manipulating element classes

Assuming the page includes an HTML element like:

```
<p class="small-paragraph">
 This is a small <a
href="https://en.wikipedia.org/wiki/Paragraph">paragr
aph</a>
 with a <a class="trusted"
href="http://stackexchange.com">link</a> inside.
</p>
```

- jQuery provides useful functions to manipulate DOM classes, most notably hasClass(), addClass(), removeClass() and toggleClass().
- These functions directly modify the class attribute of the matched elements.

```
$('p').hasClass('small-paragraph'); // true
$('p').hasClass('large-paragraph'); // false
// Add a class to all links within paragraphs
```

$('p a').addClass('untrusted-link-in-paragraph');

// Remove the class from a.trusted

$('a.trusted.untrusted-link-in-paragraph')

.removeClass('untrusted-link-in-paragraph')

.addClass('trusted-link-in-paragraph');

➢ Toggle a class Given the example markup, we can add a class with our first .toggleClass():

$(".small-paragraph").toggleClass("pretty");

Now this would return true: $(".small-paragraph").hasClass("pretty")

➢ toggleClass provides the same effect with less code as:

if($(".small-paragraph").hasClass("pretty")){

$(".small-paragraph").removeClass("pretty");}

else {

 $(".small-paragraph").addClass("pretty"); }

toggle Two classes:

$(".small-paragraph").toggleClass("pretty cool");

Boolean to add/remove classes:

$(".small-paragraph").toggleClass("pretty",true);
//cannot be truthy/falsey

$(".small-paragraph").toggleClass("pretty",false);

Function for class toggle (see example further down to avoid an issue)

$("div.surface").toggleClass(function() {

 if ($(this).parent().is(".water")) {

 return "wet";

```
} else {
 return "dry";
 }
});
```

Used in examples:

```
// functions to use in examples
function stringContains(myString, mySubString) {
 return myString.indexOf(mySubString) !== -1;
}
function isOdd(num) { return num % 2;}
var showClass = true; //we want to add the class
```

Examples:

Use the element index to toggle classes odd/even

```
$( "div.gridrow"
).toggleClass(function(index,oldClasses, false),
showClass ) {
 showClass
 if ( isOdd(index) ) {
 return "wet";
 } else {
 return "dry";
 } });
```

- ➢ More complex toggleClass example, given a simple grid markup

```
<div class="grid">
 <div class="gridrow">row</div>
 <div class="gridrow">row</div>
 <div class="gridrow">row</div>
```

```
<div class="gridrow">row</div>
 <div class="gridrow">row</div>
 <div class="gridrow gridfooter">row but I am
footer!</div>
</div>
```

Simple functions for our examples:

```
function isOdd(num) {
 return num % 2;
}
function stringContains(myString, mySubString) {
 return myString.indexOf(mySubString) !== -1;
}
var showClass = true; //we want to add the class
```

Add an odd/even class to elements with a gridrow class

```
$("div.gridrow").toggleClass(function(index,
oldClasses, showThisClass) {
 if (isOdd(index)) {
 return "odd";
 } else {
 return "even";
 }
 return oldClasses;
}, showClass);
```

- If the row has a gridfooter class, remove the odd/even classes, keep the rest.

```
$("div.gridrow").toggleClass(function(index,
oldClasses, showThisClass) {
 var isFooter = stringContains(oldClasses, "gridfooter");
 if (isFooter) {
 oldClasses = oldClasses.replace('even', '
').replace('odd', ' ');
 $(this).toggleClass("even odd", false);
 }
 return oldClasses;
}, showClass);
```

- The classes that get returned are what is effected. Here, if an element does not have a gridfooter, add a class for even/odd.
- This example illustrates the return of the OLD class list.
- If this else return oldClasses; is removed, only the new classes get added, thus the row with a gridfooter class would have all classes removed had we not returned those old ones -

- they would have been toggled (removed) otherwise.

```
$("div.gridrow").toggleClass(function(index,
oldClasses, showThisClass) {
 var isFooter = stringContains(oldClasses, "gridfooter");
 if (!isFooter) {
 if (isOdd(index)) {
 return "oddLight";
 } else {
 return "evenLight";
 }
 } else return oldClasses;
}, showClass);
```

Other API Methods

- jQuery offers a variety of methods that can be used for DOM manipulation.

The first is the .empty() method.

Imagine the following markup:

<div id="content">

<div>Some text</div>

</div>

- By calling $('#content').empty();, the inner div would be removed. This could also be achieved by using

$('#content').html('');.

Another handy function is the .closest() function:

<tr id="row_1">

<td><button type="button"

```
class="delete">Delete</button>
</tr>
```

- If you wanted to find the closest row to a button that was clicked within one of the row cells then you could do this:

```
$('.delete').click(function() {
 $(this).closest('tr');
});
```

- Since there will probably be multiple rows, each with their own delete buttons, we use $(this) within the .click() function to limit the scope to the button we actually clicked.
- If you wanted to get the id of the row containing the Delete button that you clicked, you could so something like this:

```
$('.delete').click(function() {
 var $row = $(this).closest('tr');
 var id = $row.attr('id');
});
```

- It is usually considered good practise to prefix variables containing jQuery objects with a $ (dollar sign) to make it clear what the variable is.
- An alternative to .closest() is the .parents() method:

```
$('.delete').click(function() {
 var $row = $(this).parents('tr');
 var id = $row.attr('id');
});
```

and there is also a .parent() function as well:

```
$('.delete').click(function() {
```

```
var $row = $(this).parent().parent();
 var id = $row.attr('id');
});
```

- .parent() only goes up one level of the DOM tree so it is quite inflexible, if you were to change the delete button to be contained within a span for example, then the jQuery selector would be broken.

DOM Traversing

Select children of element

- To select the children of an element you can use the children() method.

```
<div class="parent">
 <h2>A headline</h2>
 <p>Lorem ipsum dolor sit amet...</p>
```

<p>Praesent quis dolor turpis...</p>

</div>

Change the color of all the children of the .parent element:

$('.parent').children().css("color", "green");

- The method accepts an optional selector argument that can be used to filter the elements that are returned.

// Only get "p" children

$('.parent').children("p").css("color", "green");

Get next element:

- To get the next element you can use the .next() method.

<ul>

<li>Mark</li>

```
<li class="anna">Anna</li>
 <li>Paul</li>
</ul>
```

- ➢ If you are standing on the "Anna" element and you want to get the next element, "Paul", the .next() method will allow you to do that.

```
// "Paul" now has green text
$(".anna").next().css("color", "green");
```

- ➢ The method takes an optional selector argument, which can be used if the next element must be a certain kind of element.

```
// Next element is a "li", "Paul" now has green text
$(".anna").next("li").css("color", "green");
```

- If the next element is not of the type selector then an empty set is returned, and the modifications will not do anything.

```
// Next element is not a ".mark", nothing will be done in this case
$(".anna").next(".mark").css("color", "green");
```

Get previous element

- To get the previous element you can use the .prev() method.

```
<ul>
<li>Mark</li>
 <li class="anna">Anna</li>
 <li>Paul</li>
</ul>
```

- If you are standing on the "Anna" element and you want to get the previous element, "Mark", the .prev() method will allow you to do that.

// "Mark" now has green text

$(".anna").prev().css("color", "green");

- The method takes an optional selector argument, which can be used if the previous element must be a certain kind of element.

// Previous element is a "li", "Mark" now has green text

$(".anna").prev("li").css("color", "green");

- If the previous element is not of the type selector then an empty set is returned, and the modifications will not do anything.

// Previous element is not a ".paul", nothing will be done in this case

$(".anna").prev(".paul").css("color", "green");

Filter a selection

- To filter a selection you can use the .filter() method.
- The method is called on a selection and returns a new selection.
- If the filter matches an element then it is added to the returned selection, otherwise it is ignored.
- If no element is matched then an empty selection is returned.

The HTML

This is the HTML we will be using.

```
<ul>
 <li class="zero">Zero</li>
 <li class="one">One</li>
```

<li class="two">Two</li>

<li class="three">Three</li>

</ul>

Selector

- Filtering using selectors is one of the simpler ways to filter a selection.

$("li").filter(":even").css("color", "green"); // Color even elements green

$("li").filter(".one").css("font-weight", "bold"); // Make ".one" bold

Function

- Filtering a selection using a function is useful if it is not possible to use selectors.

- The function is called for each element in the selection.
- If it returns a true value then the element will be added to the returned selection.

```
var selection = $("li").filter(function (index, element) {
 // "index" is the position of the element
 // "element" is the same as "this"
 return $(this).hasClass("two");
});
selection.css("color", "green"); // ".two" will be colored green
```

Elements

- You can filter by DOM elements. If the DOM elements are in the selection then they will be included in the returned selection.

```
var three =
document.getElementsByClassName("three");
$("li").filter(three).css("color", "green");
```

Selection

- You can also filter a selection by another selection. If an element is in both selections then it will be included in the returned selection.

```
var elems = $(".one, .three");
$("li").filter(elems).css("color", "green");
```

find() method

- .find() method allows us to search through the descendants of these elements in the DOM tree and construct a new jQuery object from the matching elements.

HTML

```
<div class="parent">
 <div class="children" name="first">
 <ul>
 <li>A1</li>
 <li>A2</li>
 <li>A3</li>
 </ul>
 </div>
 <div class="children" name="second">
 <ul>
 <li>B1</li>
 <li>B2</li>
 <li>B3</li>
 </ul>
```

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

jQuery

```
$('.parent').find('.children[name="second"] ul
li').css('font-weight','bold');
```

Iterating over list of jQuery elements

- When you need to iterate over the list of jQuery elements.

Consider this DOM structure:

```
<div class="container">
 <div class="red one">RED 1 Info</div>
 <div class="red two">RED 2 Info</div>
 <div class="red three">RED 3 Info</div>
</div>
```

- To print the text present in all the div elements with a class of red:

```
$(".red").each(function(key, ele){
 var text = $(ele).text();
 console.log(text);
});
```

- Tip: key is the index of the div.red element we're currently iterating over, within its parent.
- ele is the HTML element, so we can create a jQuery object from it using $() or jQuery(), like so: $(ele).
- After, we can call any jQuery method on the object, like css() or hide() etc. In this example, we just pull the text of the object.

Selecting siblings

- To select siblings of an item you can use the .siblings() method.
- A typical example where you want to modify the siblings of an item is in a menu:

```
<ul class="menu">
 <li class="selected">Home</li>
 <li>Blog</li>
 <li>About</li>
</ul>
```

- When the user clicks on a menu item the selected class should be added to the clicked element and removed from its siblings:

```
$(".menu").on("click", "li", function () {
 $(this).addClass("selected");
 $(this).siblings().removeClass("selected");
});
```

- The method takes an optional selector argument, which can be used if you need to narrow down the kinds of siblings you want to select:

```
$(this).siblings("li").removeClass("selected");
```

closest() method

- Returns the first element that matches the selector starting at the element and traversing up the DOM tree

HTML

```
<div id="abc" class="row">
 <div id="xyz" class="row">
 </div>
 <p id="origin">
```

Hello

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

jQuery

```
var target = $('#origin').closest('.row');
console.log("Closest row:", target.attr('id') );
var target2 = $('#origin').closest('p');
console.log("Closest p:", target2.attr('id') );
```

Output

"Closest row: abc"

"Closest p: origin"

first() method :

- The first method returns the first element from the matched set of elements.

HTML

```
<div class='.firstExample'>
 <p>This is first paragraph in a div.</p>
 <p>This is second paragraph in a div.</p>
 <p>This is third paragraph in a div.</p>
 <p>This is fourth paragraph in a div.</p>
 <p>This is fifth paragraph in a div.</p>
</div>
```

JQuery

```
var firstParagraph = $("div p").first();
console.log("First paragraph:", firstParagraph.text());
```

Output:

```
First paragraph: This is first paragraph in a div.
```

CSS Manipulation

CSS – Getters and Setters

CSS Getter

- The .css() getter function can be applied to every DOM element on the page like the following:

// Rendered width in px as a string. ex: `150px`

// Notice the `as a string` designation - if you require a true integer,

// refer to `$.width()` method

$("body").css("width");

- This line will return the computed width of the specified element, each CSS property you provide in the parentheses will yield the value of the property for this $("selector") DOM element, if you ask for CSS attribute

- that doesn't exist you will get undefined as a response.

You also can call the CSS getter with an array of attributes:

$("body").css(["animation","width"]);

this will return an object of all the attributes with their values:

Object {animation: "none 0s ease 0s 1 normal none running", width: "529px"}

CSS Setter

- The .css() setter method can also be applied to every DOM element on the page.

$("selector").css("width", 500);

- This statement set the width of the $("selector") to 500px and return the jQuery object so you can more

methods to the specified selector.

The .css() setter can also be used passing an Object of CSS properties and values like:

$("body").css({"height": "100px", width:100, "padding-top":40, paddingBottom:"2em"});

- All the changes the setter made are appended to the DOM element style property thus affecting the elements' styles (unless that style property value is already defined as !important somewhere else in styles).

Increment/Decrement Numeric Properties

- Numeric CSS properties can be incremented and decremented with the += and -= syntax, respectively, using the

.css() method:

// Increment using the += syntax

$("#target-element").css("font-size", "+=10");

// You can also specify the unit to increment by

$("#target-element").css("width", "+=100pt");

$("#target-element").css("top", "+=30px");

$("#target-element").css("left", "+=3em");

// Decrementing is done by using the -= syntax

$("#target-element").css("height", "-=50pt");

Set CSS property

Setting only one style:

$('#target-element').css('color', '#000000');

Setting multiple styles at the same time:

$('#target-element').css({

'color': '#000000',

```
 'font-size': '12pt',
 'float': 'left',
});
```

Get CSS property:

- To get an element's CSS property you can use the .css(propertyName) method:

```
var color = $('#element').css('color');
var fontSize = $('#element').css('font-size');
```

Summary:

- Events
- DOM Manipulation
- DOM Traversing
- CSS Manipulation

Thank You………

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