UIClass-July-7-2016 Homework 24

1. event delegation, how on works in jquery when on method?

Event delegation allows us to avoid adding event listeners to specific nodes; instead, the event listener is added to one parent. That event listener analyzes bubbled events to find a match on child elements.

Event delegation refers to the process of using event propagation (bubbling) to handle events at a higher level in the DOM than the element on which the event originated. It allows us to attach a single event listener for elements that exist now or in the future.

<!DOCTYPE html>

<html>

<head>

<title>Event Delegation demo</title>

</head>

<body>

<div class="books">

<ul id="books">

<li><a id="book1" href="#">HTML 1</a></li>

<li><a id="book2" href="#">HTML 2 </a></li>

<li><a id="book3" href="#">HTML 3 </a></li>

<li><a id="book4" href="#">HTML 4</a></li>

<li><a id="book5" href="#">HTML 5</a></li>

</ul>

</div>

<script>

(function(){

'use strict';

var books= document.querySelector('#books');

books.addEventListener('click',function(e){

if(e.target){

if(e.target.matches('a#book4')){

console.log(e.target.innerText,'was clicked - buy it online');

} else{

console.log(e.target.innerText,'was clicked - with event delegation');

}

}

},false)

/\*

var book1=document.querySelector('#book1');

var book2=document.querySelector('#book2');

var book3=document.querySelector('#book3');

var book4=document.querySelector('#book4');

var book5=document.querySelector('#book5');

book1.addEventListener('click',function(){

console.log(book1.innerText,'was clicked');

},false);

book2.addEventListener('click',function(){

console.log(book2.innerText,'was clicked');

},false);

book3.addEventListener('click',function(){

console.log(book3.innerText,'was clicked');

},false);

book4.addEventListener('click',function(){

console.log(book4.innerText,'was clicked');

},false);

book5.addEventListener('click',function(){

console.log(book5.innerText,'was clicked');

},false);\*/

})();

</script>

</body>

</html>

Ex2:In this program ,it will add “New List Item” when we click on the add button and when we click on text “New List Item” it will fadeOut(500) ,I have used ”on” to perform this. Same can be achieved by using “delegate” in jquery version below 1.7;

<html>

<head>

<title>Event delegation demo!</title>

<script src="https://code.jquery.com/jquery-2.2.4.js"

integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="

crossorigin="anonymous"></script>

<script type="text/javascript">

$(document).ready(function(){

$('ul').on('click','li',function(){

$(this).fadeOut(500);

});

$('#btnAdd').on('click',function(){

$('ul').append('<li> New List Item </li>');

});

});

</script>

</head>

<body style="font-family:Arial">

<input id="btnAdd" type="button" value="Add a New Item"/>

<ul>

<li> List Item</li>

<li>List Item</li>

</ul>

</body>

</html>

1. bind vs on methods in jquery?

.bind() has been deprecated, for jquery 1.7 we are using “on”.

.bind() method is used for attaching an event handler directly to elements. Handlers are attached to the currently selected elements in the jQuery object, so those elements *must exist* at the point the call to .bind() occurs.

Bind() method: it will attach one or more event handlers for selected elements, and specifies a function to run when the event occurs.

The bind() example:

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>bind demo</title>

<style>

p {

background: tomato;

font-weight: bold;

cursor: pointer;

padding: 5px;

}

p.over {

background: blue;

}

span {

color: green;

}

</style>

<script src="https://code.jquery.com/jquery-1.10.2.js"></script>

</head>

<body>

<p>Click or double click here.</p>

<span></span>

<script>

$( "p" ).bind( "click", function( event ) {

var str = "( " + event.pageX + ", " + event.pageY + " )";

$( "span" ).text( "Click happened! " + str );

});

$( "p" ).bind( "dblclick", function() {

$( "span" ).text( "Double-click happened in " + this.nodeName );

});

$( "p" ).bind( "mouseenter mouseleave", function( event ) {

$( this ).toggleClass( "over" );

});

</script>

</body>

</html>

Bind() example2: when you click on the text “Please click me” ,bind() will be executed.

<head> <script src="https://code.jquery.com/jquery-1.10.2.js"></script>

<script>

$(document).ready(function(){

$("p").bind("click", function(){

alert("You have clicked on me.");

});

});

</script>

</head>

<body>

<p>Please click me!</p>

</body>

On() method: It will attach one or more event handlers for the selected elements and child elements.

Example:When you click on text , on() will be executed.

<head><script src="https://code.jquery.com/jquery-2.2.4.js"></script>

<script>

$(document).ready(function(){

$("p").on("click", function(){

alert("You Clicked Me. Thankyou!.");

});

});

</script>

</head>

<body>

<p>Please Click here!!.</p>

</body>

1. window.onload and jquery ready?

The load event fires when all the content on your page fully loaded including the DOM (document object model) content, asynchronous javascript, frames and images, we can also use **body onload=** both are same.

This method will run after everything has loaded and it attached an event handler to load event. This event mostly works with elements that are associated with URL such as image, iframe, frame or script and the window object.

Error: Refused to display 'https://www.google.com/' in a frame because it set 'X-Frame-Options' to 'SAMEORIGIN'.

In the below example, first the document will be loaded and then the window will be loaded after the page has been completely loaded.

<head>

<script src="https://code.jquery.com/jquery-2.2.4.js"></script>

<script>

$( document ).ready(function() {

alert( "document loaded" );

});

$( window ).load(function() {

alert( "window loaded" );

});

</script>

</head>

<body>

<iframe src="http://google.com"></iframe>

</body>

</body>

Ready() method: the ready method will run when the DOM is ready such as when all elements are there to be found/used are ready, but not necessarily the content. Since, this event occurs when document is ready, it is a good place to have all other jQuery event and functions.

**Jquery $document.ready**function event executes a bit earlier than window.onload and called once the DOM(Document object model) is loaded on your page. DOM means all the html tags/script I.e.(anchor tag, table, div tag, paragraph tag etc..). It will not wait for the images, frames to get fully load. It means that it is the earliest stage in page load process.

The main difference is that **$document.ready()** event gets called as soon as your DOM is loaded. It does not wait for the contents to get loaded fully. For example, there are very heavy images on any web page and takes time to load. If you have used **window.onload** then it will wait until all your images are loaded fully, hence it slows down the execution. On the other side, **$document.ready()** does not wait for elements to get loaded.

1. all questions explain with examples.