**EXO OF EVENTS LISTENERS**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>Event objects example</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<main>

<h3><span class="js-click-counter">0</span> clicks</h3>

<button class="js-clicker">Click me</button>

</main>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

// this function does two things.

// First it creates a variable

// called `clickCount`. This variable

// is meant to keep track of

// the number of times the user

// has clicked the "click me"

// button. Initially we set its value

// to zero.

**function handleClicks() {**

**let clickCount = 0;**

// this line sets the inner text

// of the `.js-click-counter`

// element to the current value

// of `clickCount` (which is 0)

**$('.js-click-counter').text(clickCount);**

// this line says when the

// `.js-clicker` element is

// clicked, run the instructions

// inside the anonymous function

// (that is, the instructions

// between the {...} brackets).

**$('.js-clicker').click(function(event) {**

// any time the user clicks,

// we add 1 to the value of

// `clickCount ...

**clickCount += 1**;

// ...and display the updated

// click count in the

// `.js-click-count` element.

**$('.js-click-counter').text(clickCount);**

**});**

**}**

// this code just says that when

// the browser is done loading the

// page, it should run the

// `handleClicks` function

// we've defined above.

**$(handleClicks);**

**DOM event objects.**

The solution is to use DOM event objects. The callback functions in your event listeners always get access to an object representing the triggering event. This contains information about where you typed in the text, what keys you pressed, or which buttons were clicked. In fact, just about every piece of information you could ever want to know about *how* you performed an action is contained within the event object.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>jquery hello world</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<input type="text" placeholder="Type here">

<button type="button">Click me</button>

<div class="output"></div>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

$(function() {

$('button').mousedown(event =>

$('.output').text(`Button clicked: ${event.which}`)

);

$('input').keydown(event =>

$('.output').text(`Key pressed: ${event.key}`)

);

});

[**event.which**](https://api.jquery.com/event.which/) (<https://api.jquery.com/event.which/>) contains information about which button was pressed in a mousedown event and the key property contains information about which key was pressed in a keydown event. Different types of events will contain different information. For example, it wouldn't make sense for there to be a key property when you had clicked a mouse button.

**event.currentTarget**

One of the most powerful properties of an event object is the currentTarget. This contains information about which DOM element the user has interacted with.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>event.currentTarget example</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<h2>this is how <code>event.currentTarget</code> works</h2>

<div class="foo">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

<ul>

<li>fee</li>

<li>fie</li>

<li>foh</li>

<li>fum</li>

</ul>

</div>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

**$(function() {**

**$(".foo, p, ul, li").click(function(event) {**

**event.stopPropagation();**

**$("h2").text("event.currentTarget's text is: " + $(event.currentTarget).text());**

**});**

**});**

By using event.stopPropagation(), we're telling the browser to stop the event from "bubbling up" the DOM. It will run for the innermost clicked element (in this case, the li that we've clicked), but it won't run again for the ul or div.foo.

CSS:

h2 {

background-color: lemonchiffon;

padding: 20px;

margin-bottom: 0;

border: 1px solid black;

}

p {

margin: 0;

background-color: lightsalmon;

padding: 20px;

border: 1px solid black;

}

ul {

margin: 0;

background-color: lightcyan;

padding: 20px;

list-style: none;

border: 1px solid black;

}

li {

border: 1px dashed black;

padding: 20px;

text-align: center;

}

## Preventing default element behaviors

Exo:

event.preventDefault

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>event.preventDefault example</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<form class="js-form">

<label for="user-text">Add an item</label>

<input type="text" id="user-text" placeholder="Type something!" required>

<input type="submit">

</form>

<p class="js-display-user-text"></p>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

**$(function() {**

**$('.js-form').submit(event => {**

**// this stops the default form submission behavior**

**event.preventDefault();**

**const userTextElement = $(event.currentTarget).find('#user-text');**

**$(".js-display-user-text").text(`user text is: ${userTextElement.val()}`);**

**userTextElement.val("");**

**});**

**})**

## The this keyword

EXO:

When writing event listeners using jQuery, the this keyword plays an important role in the callback functions that run when an event happens. Inside a callback function on an event listener, this refers to event.currentTarget. So anywhere that you have been typing out event.currentTarget, you can replace it with the this keyword.

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>`this` example</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<h2>this is how <code>this</code> works</h2>

<div class="foo">

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>

<ul>

<li>fee</li>

<li>fie</li>

<li>foh</li>

<li>fum</li>

</ul>

</div>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

$(function() {

$(".foo, p, ul, li").click(function(event) {

event.stopPropagation();

$("h2").text("this's text is: " + $(this).text());

});

});

## Event delegation

*allows us to attach a single event listener, to a parent element, that will fire for all descendants matching a selector, whether those descendants exist now or are added in the future.*

<https://learn.jquery.com/events/event-delegation/>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>event delegation example 1</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet" type="text/css" />

</head>

<body>

<button>Make a pet!</button>

<p>Click an animal to remove it.</p>

<ul>

<li class="original">tiger</li>

<li class="original">lion</li>

<li class="original">bear</li>

</ul>

<script

src="https://code.jquery.com/jquery-3.2.1.min.js"

integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4="

crossorigin="anonymous"></script>

<script src="index.js"></script>

</body>

</html>

**$(function() {**

**$("button").click(function(event) {**

**$("ul").append(**

**"<li>" +**

**["cat", "dog", "rock"][Math.floor(Math.random()\*3)] + "</li>"**

**);**

**});**

**$('li').click(function(event) {**

**this.remove();**

**});**

**});**

**$(function() {**

**$("button").click(function(event) {**

**$("ul").append(**

**"<li>" +**

**["cat", "dog", "rock"][Math.floor(Math.random()\*3)] + "</li>"**

**);**

**});**

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

**this.remove();**

**});**

**});**

This code uses the [**.on() method from jQuery**](http://api.jquery.com/on/). Up until now, you’ve seen .click(), and .hover() which are shortcuts for .on(‘click… and .on(‘hover’. The code above says that when the <ul> element is clicked (i.e., *on* any *click* of a <ul>), if that click is specifically on an <li> element within the <ul>, then run the callback function to remove the clicked element. Because the <ul> was there at the time our JavaScript code first ran, the browser is able to bind the event listener that will respond to all click events that happen inside that <ul>. By using .on('click', 'li', function(){}), we filter all click events within the <ul> so that the function runs only if the clicked item was an <li>.

**https://api.jquery.com/on/**

## The a11y benefits of starting with HTML

Consider a common real-world interface: a toggle button. Below, we've turned the image of a lightbulb — marked up in an <img> tag — into a kind of toggle button.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>&lt;img&gt; lightbulb switches</title>

<link rel="shortcut icon" href="#">

<link href="index.css" rel="stylesheet"/>

</head>

<body>

<main role="main">

<img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb" class="lightbulb js-lightbulb">

<img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb" class="lightbulb js-lightbulb">

<img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb" class="lightbulb js-lightbulb">

<img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb" class="lightbulb js-lightbulb">

</main>

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<script src="index.js"></script>

</body>

</html>

\* {

box-sizing: border-box;

}

main {

min-width: 250px;

max-width: 960px;

padding: 30px;

margin: 0 auto;

}

.lightbulb {

float: left;

width: 25%;

padding: 10px;

border: 1px solid grey;

cursor: pointer;

}

.bulb-on {

background-color: yellow;

}

function handleBulbClicks() {

$('.js-lightbulb').on('click', event => {

// save reference to this bulb,

const targetBulb = $(event.currentTarget);

// and reference to all other bulbs

const otherBulbs = $('.js-lightbulb').not(targetBulb);

// Remove 'bulb-on' class from other bulbs

otherBulbs.removeClass('bulb-on');

// toggle the presence of 'bulb-on' on this bulb;

targetBulb.toggleClass('bulb-on');

});

}

$(handleBulbClicks);

The toggleClass();

<https://api.jquery.com/toggleclass/>

what kind of image *is* interactive? Once again, form elements come to our rescue. The repl.it below replaces our wannabe <img> switches with [**input type="image"**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/image).

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/image

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>ARIA-Friendly Lightbulbs</title>

<link rel="shortcut icon" href="#">

<link rel="stylesheet" href="index.css">

</head>

<body>

<main>

<input type="image" src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb one" aria-pressed="false" class="lightbulb js-lightbulb">

<input type="image" src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb two" aria-pressed="false" class="lightbulb js-lightbulb">

<input type="image" src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb three" aria-pressed="false" class="lightbulb js-lightbulb">

<input type="image" src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/lightbulb.svg" alt="Lightbulb four" aria-pressed="false" class="lightbulb js-lightbulb">

</main>

<script src="https://code.jquery.com/jquery-3.2.1.js"></script>

<script src="index.js"></script>

</body>

</html>

\* {

box-sizing: border-box;

}

main {

min-width: 250px;

max-width: 960px;

padding: 30px;

margin: 0 auto;

}

.lightbulb {

float: left;

position: relative;

width: 25%;

padding: 10px;

border: 1px solid grey;

cursor: pointer;

}

.bulb-on {

background-color: yellow;

}

/\* Manually set outline color

for a11y \*/

input[type="image"]:focus {

outline: auto 2px Highlight;

outline: auto 5px -webkit-focus-ring-color;

z-index: 1;

}

**function handleBulbClicks() {**

**$('.js-lightbulb').on('click', event => {**

// save reference to this bulb,

**const targetBulb = $(event.currentTarget);**

// and reference to all other bulbs

**const otherBulbs = $('.js-lightbulb').not(targetBulb);**

// store whether this bulb is aria-pressed**.**

// because attr() returns strings, we convert it

// back to a boolean by making a comparison**.**

**const pressedBool = $(targetBulb).attr('aria-pressed') === 'true';**

// Remove 'bulb-on' and uncheck bulbs that are not this one

**otherBulbs.removeClass('bulb-on').attr('aria-pressed', false);**

// toggle the presence of 'bulb-on' on this bulb;

// toggle aria-pressed by inverting pressedBool

// using the ! operator.

**targetBulb.toggleClass('bulb-on').attr('aria-pressed', !pressedBool);**

**});**

**}**

**$(handleBulbClicks);**

[**https://api.jquery.com/attr/**](https://api.jquery.com/attr/)

**Practice Exo:s**

#### Cat carousel

In this challenge, you need to write JavaScript that provides the following experience: when a user selects one of the thumbnail images using the mouse or keyboard, that image should be displayed in the full-size image container at the top. Updating the image should also update its alt attribute. To test keyboard interactivity, you can Tab to one of the thumbnails and press the Enter key.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>cat carousel drill</title>

<link rel="shortcut icon" href="#">

<link rel="stylesheet" href="index.css">

</head>

<body>

<main role="main">

<h1>Image Carousel</h1>

<div class="hero">

<img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/cat1.jpg" alt="An orange-eyed grey cat stretches toward the camera."/>

</div>

<div class="thumbnails">

<a class="thumbnail" href="#"><img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/cat1.jpg" alt="An orange-eyed grey cat stretches toward the camera."/></a>

<a class="thumbnail" href="#"><img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/cat2.jpg" alt="Closeup of a blue-eyed, grey cat with markings."/></a>

<a class="thumbnail" href="#"><img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/cat3.jpg" alt="An orange cat licks its paw."/></a>

<a class="thumbnail" href="#"><img src="https://s3-us-west-2.amazonaws.com/s.cdpn.io/t-65/cat4.jpg" alt="A content brown cat lounges with eyes closed."/></a>

</div>

</main>

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<script src="index.js"></script>

</body>

</html>

function handleThumbnailClicks() {

$('.thumbnail').on('click', function(event) {

const imgSrc = $(event.currentTarget).find('img').attr('src');

const imgAlt = $(event.currentTarget).find('img').attr('alt');

$('.hero img').attr('src', imgSrc).attr('alt', imgAlt);

// Accomplishes the same as line 6 with only one use off .attr()

// by passing it an object with multiple properties.

// See: http://api.jquery.com/attr/#attr-attributes

// $('.hero img').attr({'src': imgSrc, 'alt': imgAlt});

});

}

$(handleThumbnailClicks);

**The default interactive shell is now zsh.**

**To update your account to use zsh, please run `chsh -s /bin/zsh`.**

**For more details, please visit https://support.apple.com/kb/HT208050.**

**souleymanes-MBP:www desolo$ ssh-keygen -t rsa -b 4096 -C "desolo74@gmail.com"**

**Generating public/private rsa key pair.**

**Enter file in which to save the key (/Users/desolo/.ssh/id\_rsa): www/my notes**

**Enter passphrase (empty for no passphrase):**

**Enter same passphrase again:**

**Saving key "www/my notes" failed: No such file or directory**

**souleymanes-MBP:www desolo$ Users/desolo/www**

**bash: Users/desolo/www: No such file or directory**

**souleymanes-MBP:www desolo$ Users/desolo**

**bash: Users/desolo: No such file or directory**

**souleymanes-MBP:www desolo$**

**souleymanes-MBP:www desolo$**

**souleymanes-MBP:www desolo$ ssh-add**

**souleymanes-MBP:www desolo$ ssh-add -K ~/.ssh/id\_rsa**

**/Users/desolo/.ssh/id\_rsa: No such file or directory**

**souleymanes-MBP:www desolo$**

**souleymanes-MBP:www desolo$ eval "$(ssh-agent -s)"**

**Agent pid 6627**

**souleymanes-MBP:www desolo$**

**souleymanes-MBP:www desolo$ ~/.ssh/config**

**bash: /Users/desolo/.ssh/config: No such file or directory**

**souleymanes-MBP:www desolo$**

**souleymanes-MBP:www desolo$ ssh-keygen -t rsa -b 4096 -C "desolo74@gmail.com"**

**Generating public/private rsa key pair.**

**Enter file in which to save the key (/Users/desolo/.ssh/id\_rsa): www**

**Enter passphrase (empty for no passphrase):**

**Enter same passphrase again:**

**Your identification has been saved in www.**

**Your public key has been saved in www.pub.**

**The key fingerprint is:**

**SHA256:6JQqzGMh1dQjxAsttLyHpta7QKEixwpP4aUVg1ej0Q0 desolo74@gmail.com**

**The key's randomart image is:**

**+---[RSA 4096]----+**

**| ..=\*+Eo |**

**| .++==+.. |**

**| .=+=o . |**

**|.+.B. o |**

**|\*.@ . + S |**

**|=@.o + |**

**|ooB.. . |**

**|...o. |**

**| o. |**

**+----[SHA256]-----+**

**souleymanes-MBP:www desolo$**