5. Introducing jQuery Manipulating content

Client-Side Web Programming

2022/2023

José Socuéllamos

Index

1. Iterating through elements
2. Manipulating content
3. Events

1.- Iterating through elements

Filter	Description	Example
length // ~~size()~~	Number of elements in the object	$("ul").length;
get()	Gets an array of all the DOM elements	$("p").get();
get(n)	Gets the specified DOM element	$("p").get(3);
find(selector | el)	Gets the descendant elements of the selected element.	$("body").find("p.classA");

each(function(i))	Runs a function for each matched element.	$("li").each(function(i) {
		  var text = $(this).text();
		  console.log(‘The text of the ' + i +
		    ' element is:' + text);
		});

2.- Manipulating Content

- Creating content
  - We can create HTML content by passing a tag as an argument:

```
let p = $('<p>');
```

  - This doesn't add the element to the document, it only creates a new element ready for us to add, which we'll learn later.
  - We can actually create more than one element. In fact, any tree of HTML elements we want:

```
$("<ul><li>one</li><li>two</li><li>three</li></ul>");
```

  - We can also use this format to create an element with attributes:

```
$('<img src="photo.jpg" alt="my pet" />');
```

2.- Manipulating Content

- Creating content
  - A more elegant way of creating jQuery elements is by setting their attributes without having to build the full HTML string ourselves:

```
let myId ="container";
let myText = "Lorem Ipsum";
$('<div>', {
    id: myId,
    text: myText
});
```

```
let time = new Date().getHours();
let photo;
if (time > 12) {
    photo = "afternoon.jpg";
 } else {
    photo = "morning.jpg";
 }
 $("<img>", { src: photo });
```

2.- Manipulating Content

- Creating content
  - We cannot add plain text to the document using the function $().
  - To get, append and set content we can use methods:

Filter	Description	Example
html()	Gets the content (innerHTML) of the first matched element	$("#myImg").html();
html(htmlString)	Sets the content of all matched elements, overwriting it (can include HTML)	$(".blue").html("My <b>list</b>");
append(htmlString)	appends the content to the end of all matched elements (can include HTML)	$(".blue").append("My <b>list</b>");
prepend(htmlString)	appends the content to the beginning of all matched elements (can include HTML)	$(".blue").prepend("My <b>list</b>");
text()	Gets the text content of all matched elements	$("#myP").text();
text(textString)	Sets only the text content of all matched elements, overwriting it (cannot include HTML)	$("p").text("My paragraph");

2.- Manipulating Content

- Manipulating attributes
  - We can manipulate the values of one or more attributes using the following functions.

Filter	Description	Example
attr(name)	Gets the attribute value of the first matched element	$("#myimg").attr("src");
attr(name,value)	Sets the attribute value for all matched elements.	$("#myimg").attr("src","http:…”);
attr({name:value})	Sets multiple values to multiple attributes.	$("#myimg").attr({ src: "http:…. ", alt: "my image"});
removeAttr(name)	Removes one or more attributes from the selected elements.	$("#myimg").removeAttr("width height");

2.- Manipulating Content

- Inserting and moving content
  - We can add/move existing or new content to the elements selected using the following functions:

Filter	Description	Example
appendTo(selector)	Inserts specified content at the end of the selected elements (also moves).	$("<li>Second element</li>").appendTo("ul");
prependTo(selector);	Inserts specified content at the beginning of the selected elements (also moves).	$("<p>Hello</p>").prependTo("div:first)");
insertBefore(selector);	Inserts HTML elements before the selected elements (also moves)	$("#myimg").insertBefore("ul:first");
insertAfter(selector);	inserts HTML elements after the selected elements (also moves)	$("#myimg").insertAfter("ul:eq(2)");

  - **BEWARE**: *append* and *prepend* methods work the other way around than *appendTo* and *prependTo*

2.- Manipulating Content

- Wrapping content
  - Wrapping means we can get an existing element inside a new element.

Filter	Description	Example
wrap(html)	Wraps a specified HTML element around each selected element	$(".a").wrap("<div style= 'border: 3px solid red'></div>");
wrapAll(html)	Wraps a specified HTML element around all selected element	$(".a").wrapAll("<div style= 'border: 3px solid red'></div>");
wrapInner(html);	Wraps a specified HTML element around the content (innerHTML) of each selected element	$("td").wrapInner("<b></b>");
unwrap();	Removes the parent element of the selected elements	$("p").unwrap ();

2.- Manipulating Content

- Wrapping content – `wrap` vs `wrapAll`

```
<div class="foo"></div>
<div class="foo"></div>
<div class="foo"></div>
```

```
$('.foo').wrap('<div class="bar" />');
```

```
<div class="bar"><div class="foo"></div></div>
<div class="bar"><div class="foo"></div></div>
<div class="bar"><div class="foo"></div></div>
```

```
$('.foo').wrapAll('<div class="bar" />');
```

```
<div class="bar">
  <div class="foo"></div>
  <div class="foo"></div>
  <div class="foo"></div>
</div>
```

2.- Manipulating Content

- Wrapping content – `wrapInner`

```
<div class="container">
  <div class="inner">Hello</div>
  <div class="inner">Goodbye</div>
</div>
```

```
$( ".inner" ).wrapInner( "<div class='new'></div>");
```

```
<div class="container">
  <div class="inner">
    <div class="new">Hello</div>
  </div>
  <div class="inner">
    <div class="new">Goodbye</div>
  </div>
</div>
```

2.- Manipulating Content

- Replacing content
    - With jQuery, we can replace the content of an element by another

Filter	Description	Example
replaceWith(content)	Replaces selected elements with new content (can contain HTML tags).	$("p:first").replaceWith("<img>…</img>");
replaceAll(selector)	Replace selected elements with new HTML elements.	$("<h2>New text</h2>").replaceAll("p");

2.- Manipulating Content

- Removing and Cloning elements

Filter	Description	Example
empty()	Removes all child nodes and content from the selected elements.	$(".foo").empty();
remove()	Removes the selected elements, including all text and child nodes.	$("#foo").remove();
clone()	Makes a copy of selected elements (including child nodes, text and attributes) and returns them for a further use	$("#foo").first().clone().appendTo("#bar");

2.- Manipulating Content

- Adding, removing and checking class names
  - In JavaScript, adding and removing class names to elements of a set was a quite long process.

```
let elements = document.getElementsByClassName('my-class');
for (let i = 0; i < elements.length; i++) {
    elements[i].classList.add('hidden');
}
```

2.- Manipulating Content

- Adding, removing and checking class names
  - Now in jQuery, it’s an easy operation.

```
var elements = document      El    ntsByClassName('my-class');
for (let i = 0; i < eleme         ngth; i++) {
    elements[i].classList         hidden');
}
```

```
$('.my-class').addClass('hidden');
```

2.- Manipulating Content

- Adding, removing and checking class names

Filter	Description	Example
addClass(className/s)	Adds one or more class names (separating them with spaces) to the selected elements.	$("#foo").addClass("bar title");
removeClass(className/s)	Removes one or more class names from the selected elements (all classes if empty).	$("#foo").removeClass("bar");
hasClass(className)	Checks if ANY of the selected elements have a specified class name.	$("#foo").hasClass("bar");
is(selectorElement)	Checks if one of the selected elements matches the selectorElement.	if ($("ul").parent().is(".bar")) {...}

```
if (aValue === 10) {
    $('p').addClass('hidden');
} else {
    $('p').removeClass('hidden');
}
```

```
$('p:first').is('.surprise');
```

```
$('p:first').hasClass('.surprise');
```

2.- Manipulating Content

- Toggling classes

Filter	Description	Example
toggleClass(className)	Adds the class if it's not set to the matched elements or removes it if it's already set.	$("#foo").toggleClass("bar");

```css
.hidden {
    display: none;
}
```

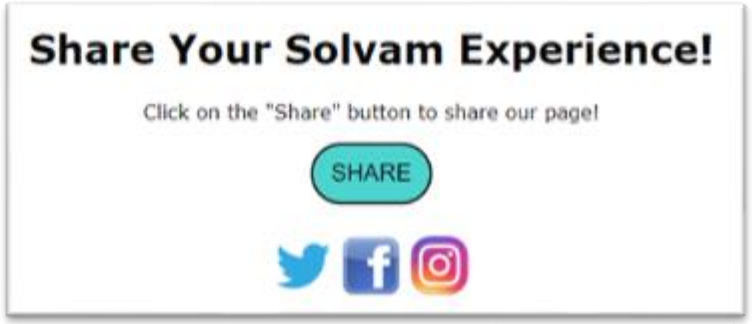
CSS

```html
<button class="share-button">Share</button><br>
<img src="twitter.png" class="socials">
<img src="facebook.jpg" class="socials">
<img src="instagram.png" class="socials">
```

HTML

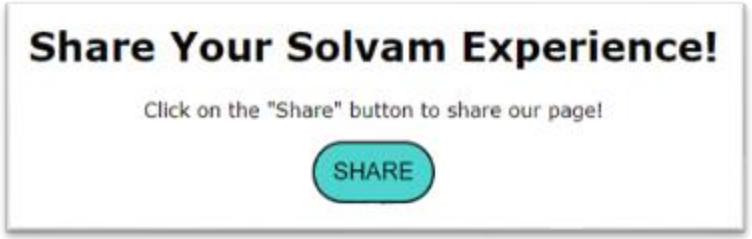
```javascript
$('.share-button').click(function () {
    $('.socials').toggleClass('hidden');
});
```

jQuery



Share Your Solvam Experience!

Click on the "Share" button to share our page!



Share Your Solvam Experience!

Click on the "Share" button to share our page!

2.- Manipulating Content

- Toggling classes with conditions

Filter	Description	Example
toggleClass(className, switch/condition)	If the condition is met (the switch is true), the class is set. If the condition is not met (the switch is false), the class is removed.	$("#foo").toggleClass("bar", a===3);

```
if (aValue === 10) {
    $('p').addClass('hidden');
} else {
    $('p').removeClass('hidden');
}
```

```
$('p').toggleClass('hidden', aValue === 10);
```

2.- Manipulating Content

- Getting and setting styles

Filter	Description	Example

css(property)	Gets the CSS property value of the FIRST matched element (beware of shorthand properties: border…).	`$("#foo").css("font-family");`
		`$("#foo").css("border");`
		`//different result in different browsers`

css(property,value)	Set the CSS property and value.	`$("#foo").css("width", "20");`

css({property:value, …})	Sets multiple CSS properties and values.	`$("#foo").css({`
		`"border": "3px solid green",`
		`"background-color": "red"`
		`});`

- Getting and setting dimensions
  - width(), height(), innerHeight(), innerWidth(), offset(), position()…

2.- Manipulating Content

- Dealing with form element values
  - Because form elements have special properties, jQuery contains some functions to get and set their values:

Filter	Description	Example
val()	Gets the value of the value attribute of the FIRST matched element.	$('input[type="radio"][name="radio-group"]:checked').val();
val(value)	Sets the value of the value attribute for ALL matched elements.	$('input[type="select"]').val(["one","two","three"]);

```
<label>John<input type="checkbox" name="Beatles" value="John"></label>
<label>Paul<input type="checkbox" name="Beatles" value="Paul"></label>
<label>George<input type="checkbox" name="Beatles" value="George"></label>
<label>Ringo<input type="checkbox" name="Beatles" value="Ringo"></label>
```

```
var checkboxValues =
    $('input[type="checkbox"][name="Beatles"]:checked').map(function () {
        return $(this).val();
    }).toArray();
```

3.- Events

- jQuery has its own event implementation that hides the differences between browsers from us.
- We have a unified method for setting event handlers.
- It allows multiple handlers for each event type in each element.
- Event-type names are standard (i.e. *click*, *mouseover*).
- The Event instance is passed as the first argument of the handlers.

3.- Events

- It normalizes the Event instance for the most often used properties.
- It provides unified methods for event cancelling and default action blocking.
- With jQuery, you can select a set of elements and then attach the same handler to all of them in one statement.

```
$('img').on('click', function (e) {
    alert("Hi there!");
});
```

3.- Events

- As you can see, the way to attach a handler to an event is using the following syntax.

```
$("div").on('click', function () { ... });
```

- Besides, the on() method, we can also attach a function to a given event by using “event-named” methods:

```
$("div").click(function () { ... });
```

3.- Events

- We can attach the event handler only to the specified child elements, and not the selector itself:

```
$("div").click('p', function () { ... });
```

  - This event applies only to p elements inside the div element.

- Finally, we can attach multiple events at once:

```
$('button')
    .on('click', function (e) {
        console.log('Button clicked!');
    })
    .on('mouseenter mouseleave', function (e) {
        $(this).toggleClass('test');
    })
```

3.- Events

- The events available to listen to are:

Events			
blur	focusout	mousedown	mouseup
change	hover	mouseenter	ready
click	keydown	mouseleave	resize
dblclick	keypress	mousemove	scroll
focus	keyup	mouseout	select
focusin	hover	mouseover	submit

3.- Events

- jQuery provides a specialized version of the on() method, called one(), which sets a single use event handler.
- The event will only run once and then it will remove itself.

```
$("p").one("click", function () {
    $(this).css('font-size', '12px');
});
```

- To remove an event, we just have to use the off() function.

```
$("button").off("click");
```

3.- Events

- Event object
  - We can find information about the triggered event wrapped in the event object.
  - We can retrieve the Event object as we did in JavaScript, as arguments of the triggered function.
  - https://api.jquery.com/category/events/event-object

3.- Events

- Event object
  - The most important properties and methods are:
    - `type`: event type (click, mouseover,etc)
    - `target`: the element that triggered the event
    - `pagex`, `pagey`: the mouse position relative to the document
    - `timestamp`: time when the event has triggered
    - `preventdefault()`: avoid to run the default action in the browser

3.- Events

- Triggering event handlers
  - Event handlers are designed to be invoked when the browser or user activity triggers the events.
  - jQuery has provided methods to automatically trigger event handlers.

```
trigger(eventName)
```

3.- Events

```
var foo = function (e) {
    if (e) {
        console.log(e.type);
    } else {
        console.log("This function wasn't triggered by an event")
    }
};

$('p').click(foo);

foo(); //instead of $('p').trigger('click');
```