CS472 WAP

Node Manipulation

DOM, jQuery

Main Point Preview

- The jQuery object returned by the selection mode of $() is a collection of DOM elements wrapped by jQuery functionality. This object can read style properties as well as set them by using the css method of jQuery.
- *Science of Consciousness: Node manipulation involves locating nodes, observing or reading values, and changing values. Our TM practice develops our ability to locate quiet states of awareness. Advanced techniques and the TM-Sidhi Program develop abilities to experience and manipulate different characteristics of pure consciousness*

Aspects of the DOM and jQuery

- **Identification:**
  - how do I obtain a reference to the node that I want.
  - using css-like selectors to get target nodes
- **Traversal:**
  - Find nodes by tree traversal relations
  - using children, sibling, parent, etc links to get target nodes
- **Node Manipulation:**
  - how do I get or set aspects of a DOM node.
  - e.g., style, attributes, innerHTML
- **Tree Manipulation:**
  - how do I change the structure of the page.

Looping over jQuery elements

```
$("li").each(function(idx, e) {
  // do stuff with e
});
```

- You cannot use a regular JavaScript for each loop on the jQuery object because it is not an array, just an array-like object

Inside the jQuery each loop

```
$("li").each(function(idx, e) {
  // do stuff with e
});
```

- return false to exit the loop early
- e is a plain old DOM object
  - We can upgrade it again using $ if we want

```
$("li").each(function(idx, e) {
  eJ = $(e);
  // do stuff with e
});
```

Modifying DOM nodes

- DOM nodes have fields that correspond to the attributes in HTML tags. There are a few exceptions

HTML attributes	DOM fields
title	.title
id	.id
class	.className
style="prop: value"	.style.prop = value

Getting/setting CSS classes in DOM

```
function highlightField() {
  // turn text yellow and make it bigger
  var elem = document.getElementById("id");

  if (!elem.className) {
    elem.className = "highlight";
  } else if (elem.className.indexOf("invalid") < 0) {
    elem.className += " highlight";
  }
}
```

- JS DOM's className property corresponds to HTML `class` attribute
- somewhat clunky when dealing with multiple space-separated classes as one big string
- className value is a string, not an array like we would want

Getting/setting CSS classes in jQuery

```
function highlightField() {
  // turn text yellow and make it bigger
  if (!$("#myid").hasClass("invalid")) {
    $("#myid").addClass("highlight");
  }
}
```

- addClass, removeClass, hasClass, toggleClass manipulate CSS classes
- similar to existing className DOM property, but don't have to manually split by spaces

Adjusting styles with the DOM

```
<button id="clickme">Color Me</button>
window.onload = function() {
  document.getElementById("clickme").onclick = changeColor;
};
function changeColor() {
  const clickMe = document.getElementById("clickme");
  clickMe.style.color = "red";
}
```

Property	Description
style	lets you set any CSS style property for an element

- contains same properties as in CSS, but with camelCasedNames
  - examples: backgroundColor, borderLeftWidth, fontFamily

Adjusting styles in jQuery

*

```
function biggerFont() {
   // turn text yellow and make it bigger
   $("#clickme").css("color", "yellow");
   const size = parseInt($("#clickme").css("font-size"));
   const newsize = size + 16 + "px";
   $("#clickme").css("fontSize", newsize );
}
```

- css function of the jQuery object works even if styles not previously set
- Accepts familiar font-size syntax in addition to fontSize
- css(*property*) gets the property value
- css(*property*, *value*) sets the property value

Common bug: incorrect usage of existing styles

```
// bad!
 ~~$("#main").css("top", $("#main").css("top") + 100 + "px");~~
```

- the above example computes e.g. "200px" + 100 + "px" , which would evaluate to "200px100px“

- a corrected version:

```
// correct
$("#main").css("top", parseInt($("#main").css("top")) + 100 + "px");
```

Recall: Unobtrusive styling

```
function okayClick() {
  ~~this.style.color = "red";~~
  this.className = "highlighted";
}
.highlighted { color: red; }
```

- well-written JavaScript code should contain as little CSS as possible
- use JS to set CSS classes/IDs on elements
- define the styles of those classes/IDs in your CSS file
- Conclusion: unobtrusive styling means avoid using the .css method in jQuery
  - Set classes in JavaScript code to handle any style changes

jQuery method behavior

- Getters typically operate only on the first of the jQuery object's selected elements.

```
<ul>
  <li style="font-size: 10px">10px font size</li>
  <li style="font-size: 20px">20px font size</li>
  <li style="font-size: 30px">30px font size</li>
</ul>
$("li").css("font-size"); // returns '10px'
```

- Setters typically operate on all of the selected DOM elements.

```
$("li").css("font-size", "15px"); // sets all selected elements to '15px'
<ul>
  <li style="font-size: 15px">10px font size</li>
  <li style="font-size: 15px">20px font size</li>
  <li style="font-size: 15px">30px font size</li>
</ul>
```

jQuery method parameters

- Many jQuery object methods are overloaded
- getter syntax:

```
 $("#myid").css(propertyName);
```

- setter syntax:

```
 $("#myid").css(propertyName, value);
```

- multi-setter syntax:

```
 $("#myid").css({
   'propertyName1': value1,
   'propertyName2': value2,
   ...
 });
```

- modifier syntax:

```
$("#myid").css(propertyName,
  function(idx, oldValue) {
    return newValue;
});
```

- function allows for computation based on the old value

common jQuery mistake

```
// bad jQuery
$("div").css("top", parseInt($("div").css("top")) +
 100 + "px");
```

- Likely to give bad results if multiple selected objects. Why?

- a corrected version:

```
$("div").css("top", function(idx, old) {
  return parseInt(old) + 100 + "px";
}); // good jQuery
```

jQuery method returns

- When there is no other return to make, jQuery methods return the same jQuery object back to you

method	return type
$("#myid");	jQuery object
$("#myid").children();	jQuery object
$("#myid").css("margin-left");	String
$("#myid").css("margin-left", "10px");	**jQuery object**
$("#myid").addClass("special");	**jQuery object**

jQuery chaining

```
~~$("img").css("color", "red");~~
~~$("img").attr("id", "themainarea");~~
~~$("img").addClass("special");~~
```

- The implictly returned jQuery object allows for chaining of method calls.

```
$("img") // good jQuery style
   .css("color", "red")
   .addClass("special")
   .attr("src", "foo.png");
// we could chain further right here
```

More node manipulation with jQuery

jQuery method	functionality
.hide()	set CSS display: none
.show()	set CSS display to original value, e.g., block, inline
.empty()	remove everything inside the element, innerHTML = ""
.html()	get/set the innerHTML without escaping html tags
.text()	get/set the innerHTML, HTML escapes the text first
.val()	get/set the value of a form input, select, textarea, ...
.height()	get/set the height in pixels, returns a Number
.width()	get/set the width in pixels, return a Number

Main Point

- The jQuery object returned by the selection mode of $() is a collection of DOM elements wrapped by jQuery functionality. This object can read style properties as well as set them by using the css method of jQuery.
- *Science of Consciousness: Node manipulation involves locating nodes, observing or reading values, and changing values. Our TM practice develops our ability to locate quiet states of awareness. Advanced techniques and the TM-Sidhi Program develop abilities to experience and manipulate different characteristics of pure consciousness*

MAHARISHI • INTERNATIONAL UNIVERSITY •

HIGHER EDUCATION • FOR HIGHER CONSCIOUSNESS •