Lecture 7: jQuery

CS472 Web Programming

Maharishi University of Management

Department of Computer Science

Assistant Professor Asaad Saad

Maharishi University of Management - Fairfield, Iowa © 2016

Why jQuery?

JavaScript is a powerful language, but it coding to browsers is difficult:

- Code that works great in Chrome, Firefox, Safari, ... will fail in IE and vice versa
- Browsers have different implementation of specs in each version
- With the rise of mobile devices maintaining browsers compatibility becomes difficult.

jQuery Framework

The jQuery JavaScript library adds many useful features to JavaScript:

- Many useful extensions to the DOM (enhancing/upgrading DOM objects)
- Adds utility functions for built-in types String, Array, Object, Function
- Improves event-driven programming
- Many cross-browser compatibility fixes
- Makes Ajax programming easier

```
jQuery 1.x:
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>

jQuery 2.x:
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
```

Your best friend: http://api.jquery.com/ 😜

jQuery Design Principles

jQuery is so powerful because of these design principles

- An expressive method for defining a set of elements
- A superset of CSS selectors
- Useful and commonly needed methods for navigating the DOM tree
- Heavily overloaded APIs
- Functional programming techniques that apply operations to sets of elements at a time
- Method chaining for succinct operations

Aspects of the DOM and jQuery

Identification: (Identifying elements signature)

- How do I obtain a reference to the node that I want.
- Using css-like selectors to get target nodes

Traversal: (Identifying elements signature)

- How do I move around the DOM tree.
- Using children, sibling, parent, etc links to get target nodes

Node Manipulation: (Identifying elements signature)

- How do I get or set aspects of a DOM node.
- e.g., style, attributes, innerHTML

Tree Manipulation: (Creating new elements signature)

- How do I change the structure of the page.

jQuery $ function signatures

Responding to the page ready event

```
   $(function);
```

Identifying elements

```
   $("selector", [context]);
```

Upgrading DOM elements

```
   $(elements);
```

Creating new elements

```
   $("<html>", [properties]);
```

Page ready event -- `window.onload`

We cannot use the DOM before the page has been constructed. jQuery gives us a more compatible way to do this.

The DOM way

```
    window.onload = function() {
      // do stuff with the DOM
    }
```

The direct jQuery translation

```
    $(document).ready(function() {
      // do stuff with the DOM
    });
```

The jQuery way

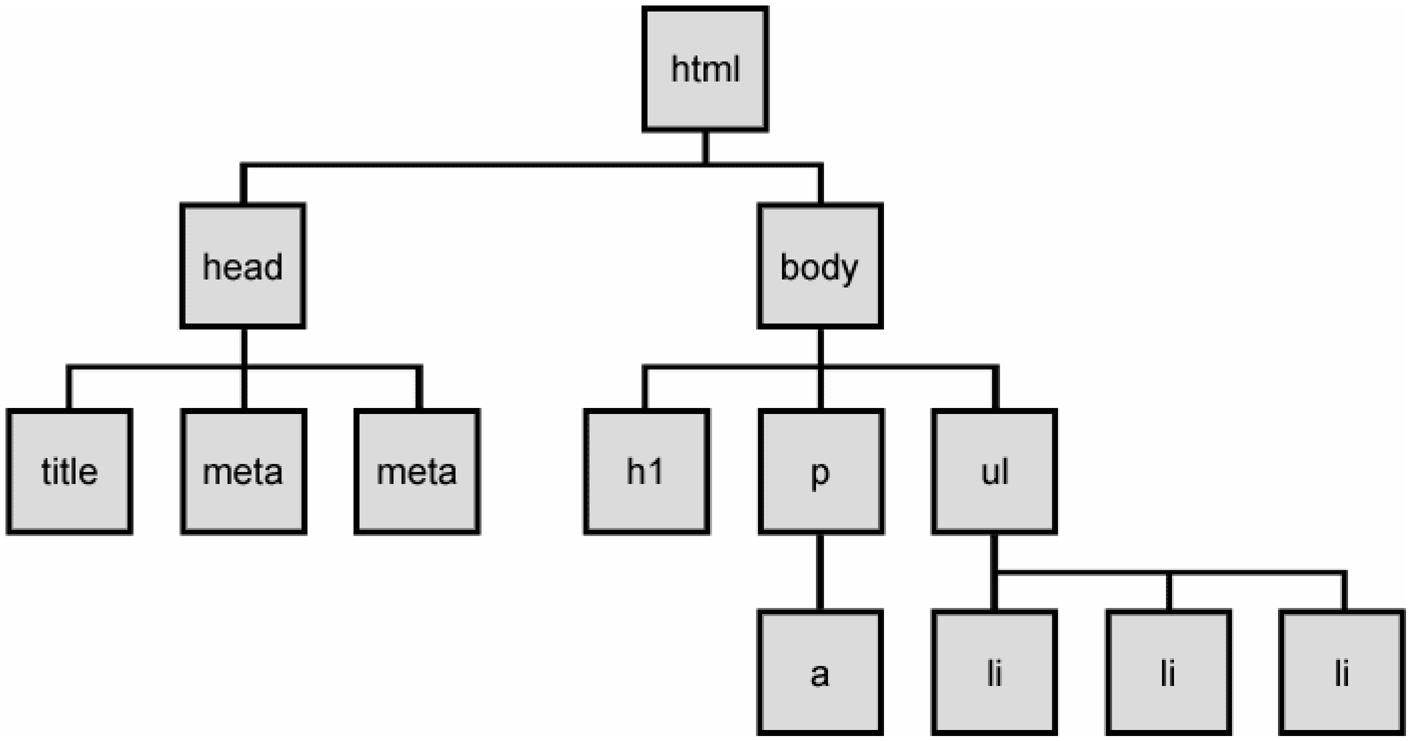
```
    $(function() {
      // do stuff with the DOM
    });
```

Main Points

It is important to realize that the $() function is heavily overloaded. It will be a different function depending on the arguments it has, including running a callback function on page load, selecting DOM elements, wrapping DOM elements, and creating new DOM elements. Science of Consciousness: In ordinary waking state consciousness we might perceive ourselves as very different people depending on surface characteristics such as our job or skills. By having the experience of pure awareness we realize that these are all different aspects of our true non-changing Self.

The DOM tree

The elements of a page are nested into a tree-like structure of objects



Selecting groups of DOM objects

- **`getElementById`** returns the first element with the specified id.
- **`getElementsByTagName`** returns array of all elements with the given tag, such as "div"
- **`getElementsByName`** returns array of all elements with the given name attribute
- **`querySelector`** returns the first element that would be matched by the given CSS selector string
- **`querySelectorAll`** returns an array of all elements that would be matched by the given CSS selector string

```
$("selector", [context]);
```

Example

```html
<body>
       <p>This is the first paragraph</p>
       <p>This is the second paragraph</p>
       <p>You get the idea...</p>
</body>
```

```javascript
var allParas = document.querySelectorAll("p");
for (var i = 0; i < allParas.length; i++) {
       allParas[i].style.backgroundColor = "yellow";
}
```

```javascript
$("p").css("background-color", "yellow");
```

jQuery node identification

The $ aka jQuery function selects elements from the DOM using most any CSS selector.

```
// id selector
var elem = $("#myid");

// group selector
var elems = $("#myid, p");

// context selector
var elems = $("#myid > div p");

// complex selector
var elems = $("#myid > h1.special:not(.classy)");
```

jQuery / DOM comparison

`getElementById("id")`	`$("#id")`
`getElementsByTagName("tag")`	`$("tag")`
`getElementsByName("somename")`	`$("[name='somename']")`
`querySelector("selector")`	`$("selector")`
`querySelectorAll("selector")`	`$("selector")`

`$("[name='somename']")` is instance of the general selector `$("[someAtt='someVal']")`

jQuery Terminology

The jQuery function

Refers to the global jQuery function/object or the $ function depending on the context

jQuery object

The object returned by the jQuery function that often represents a group of elements

Selected elements

The DOM elements that you have selected, most likely by some CSS selector passed to the jQuery function and possibly later filtered further

jQuery object

The $ function always (even for ID selectors) returns an array-like object called a jQuery object.

This returned jQuery object wraps the originally selected DOM objects.

You can access the actual DOM object by accessing the elements of the jQuery object.

```
// Not the same
document.getElementById("myid") !== $("#myid");
document.querySelectorAll("p") !== $("p");

// true
document.getElementById("myid") === $("#myid")[0];
document.getElementById("myid") === $("#myid").get(0);
document.querySelectorAll("p")[0] === $("p")[0];
```

Using `$` as a wrapper (upgrade)

- `$` adds extra functionality to DOM elements
- Passing an existing DOM object to `$` will give it the jQuery upgrade

```
// convert regular DOM objects to a jQuery object
var elem = document.getElementById("myelem");
elem = $(elem);

var elems = document.querySelectorAll(".special");
elems = $(elems);
```

Main Points

When the argument to $() is a CSS selector the function will return a “jQuery object” that contains a group of selected DOM elements. CSS selectors are a simple, natural, and powerful tool used by jQuery to identify groups of DOM elements. **Science of Consciousness:** A mantra is a simple, natural, and powerful tool that we use in the TM Technique.

Types of DOM nodes

```
<p>This is a paragraph of text with a <a href="/path/page.html">link in it</a>.</p>
```

Element nodes (HTML tag)

 can have children and/or attributes

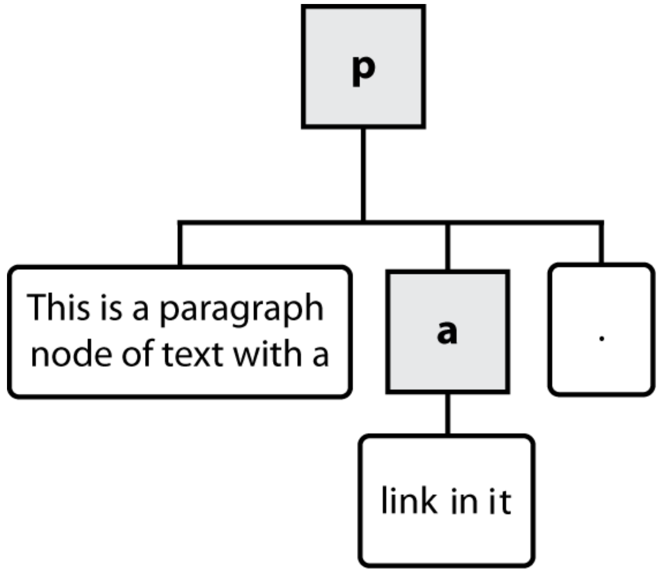
Text node text nodes (text in a block element)

Attribute node attribute nodes (attribute/value pair)

 text/attributes are children in an element node

 cannot have children or attributes

 not usually shown when drawing the DOM tree



DOM tree traversal example

```
<div>
     <p id="foo">This is a paragraph of text with a
     <a href="/path/to/another/page.html">link</a>.</p>
</div>
```

Q: How many children does the <div> above have?

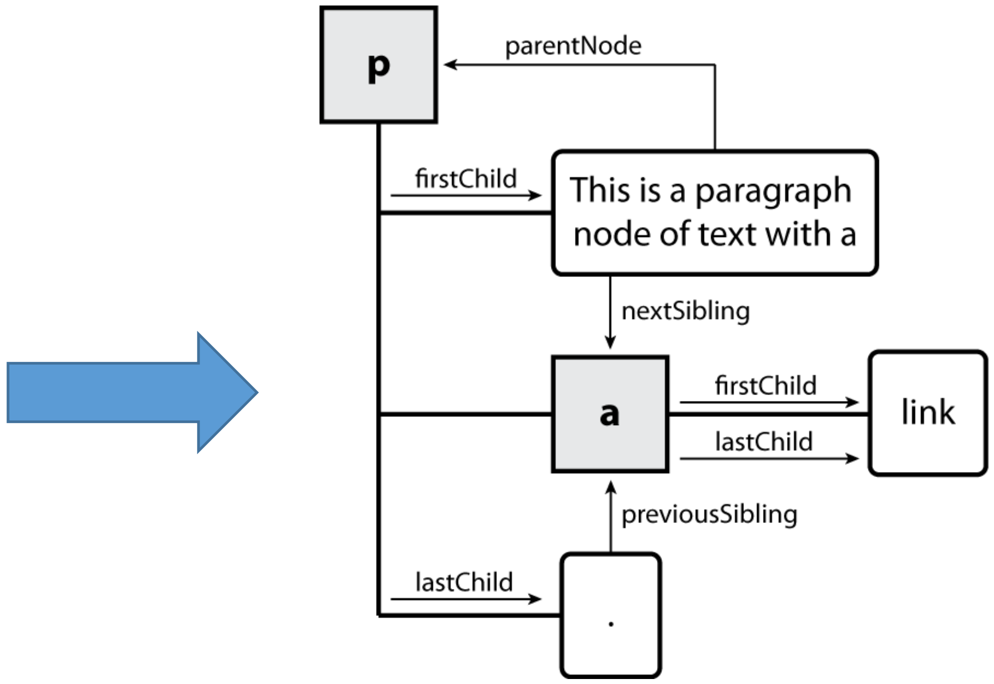
1. Text nodes representing "\n\t"
2. An element node representing the <p>
3. Text nodes representing "\n\t"

Q: How many children does the <p> above have?

1. This is a paragraph of text with a
2. An element node representing the <a>
3. The dot (.)

Q: How many children does the <a> have?

1. The anchor has 1.



jQuery Traversing - Ancestors

An ancestor is a parent, grandparent, great-grandparent, and so on.

- **`parent()`** - returns the direct parent element of the selected element.
- **`parents()`** - returns all ancestor elements of the selected element.
  - You can also use an optional parameter to filter the search for ancestors.
- **`parentsUntil()`** - returns all ancestor elements between two given arguments.

```
$("span").parent();
$("span").parents();
$("span").parents("ul"); // returns all ancestors of all <span> elements that are <ul> elements
$("span").parentsUntil("div");
```

jQuery Traversing - Descendants

A descendant is a child, grandchild, great-grandchild, and so on.

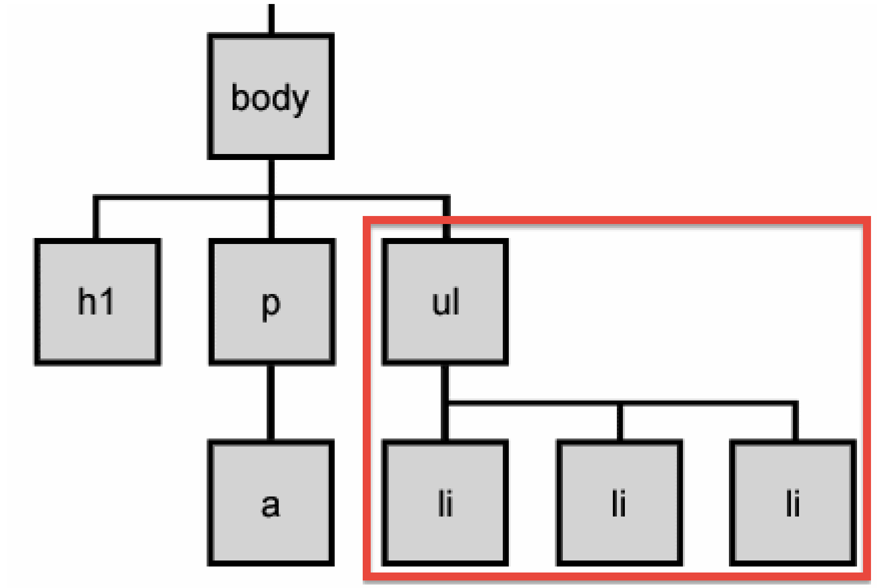
- **`children()`** - returns all direct children of the selected element.
  - You can also use an optional parameter to filter the search for children.
- **`find()`** - returns descendant elements of the selected element

```
$("div").children();

// returns all <p> elements with the class name "first", that are direct children of <div>
$("div").children("p.first");

// returns all <span> elements that are descendants of <div>:
$("div").find("span");

// returns all descendants of <div>
$("div").find("*");
```

`.find()` Contextual Element Identification

jQuery gives two identical ways to do contextual element identification:

```
var elem = $("#myUL");

// These are identical
var specials = $("li.special", elem);
var specials = elem.find("li.special");
```



Notice:

- `h1, p, ul` are children to `body` (they are related to each other as siblings)
- `a` element is descendent of `body`, and child of `p`
- `p` is parent to `a`

jQuery Traversing - Siblings

- **`siblings()`** - returns all sibling elements of the selected element
  - You can also use an optional parameter to filter the search for siblings
- **`next()`** - returns the next sibling element of the selected element.
- **`nextAll()`** - returns all next sibling elements of the selected element
- **`nextUntil()`** - returns all next sibling elements between two given arguments
- **`prev()`**
- **`prevAll()`**
- **`prevUntil()`**

jQuery Traversing - Filtering

```
// selects the first <p> element inside the first <div> element
$("div p").first();

// selects the last <p> element inside the last <div> element
$("div p").last();

// returns an element with a specific index number of the selected elements
$("p").eq(1);
```

The filter() method lets you specify a criteria. Elements that do not match the criteria are removed from the selection, and those that match will be returned

```
// returns all <p> elements with class name "intro"
$("p").filter(".intro");

// returns all elements that do not match the criteria - opposite to filter
$("p").not(".intro");
```

jQuery - Get and Set CSS Classes

- **`addClass()`** - Adds one or more classes to the selected elements
- **`removeClass()`** - Removes one or more classes from the selected elements
- **`toggleClass()`** - Toggles between adding/removing classes from the selected elements
- **`css()`** - Sets or returns the style attribute

```
$("h1, h2, p").addClass("blue");
$("div").addClass("important");
$("#div1").addClass("important blue");
$("h1, h2, p").removeClass("blue");
$("h1, h2, p").toggleClass("blue"); // add if not found, remove if exists
```

.css()

The css() method sets or returns one or more style properties for the selected elements.

```
$("p").css("background-color"); // returns the background-color value
$("p").css("background-color", "yellow"); // sets the value

// set a background-color and a font-size for ALL matched elements
$("p").css({
   "background-color": "yellow",
   "font-size": "200%"
});

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

`.css()` 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>
</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>
</ul>
```

Common bug: incorrect usage of existing styles

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

The above example computes `"200px" + 100 + "px"` which evaluates to `"200px100px"`

A corrected version:

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

Common jQuery mistake

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

The above example does not take full advantage of jQuery syntax.
Does not work if there are multiple selected objects. Why is that?

A corrected version:

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

jQuery method returns

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

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

jQuery chaining

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

The implicitly returned jQuery object allows for chaining of method calls.

```
$("img").css("color", "red")
        .addClass("special")
        .src = "foo.png";
```

Expression return value at each line:

```
// [<img />, ...]
// [<img style="color: red" />, ...]
// [<img class="special" style="color: red" />, ...]
// cannot chain further because this is an assignment :(
```

`.attr()`

- jQuery has a wrapper function for getting/setting various attributes of selected elements.
- Allows us to chain our method calls.

```
// poor jQuery style
$("img").css("color", "red")
         .addClass("special")
         .src = "foo.png";

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

More node manipulation with jQuery

`.hide()`	toggle CSS display: none on
`.show()`	toggle CSS display: none off
`.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 Points

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:** Our TM practice develops our ability to locate or experience quiet states of awareness. Advanced techniques and the TM-Sidhi Program develop abilities to experience and manipulate different characteristics of pure consciousness.

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

jQuery

1. jQuery is a powerful and widely used JavaScript library for working with the DOM that provides cross-browser compatibility.
2. The jQuery function returns a jQuery object, which contains a collection of DOM elements for manipulation.

3. **Transcendental consciousness**. The TM Technique is a convenient and cross-platform API for experiencing transcendental consciousness.
4. **Impulses within the transcendental field**: Thoughts and actions at quiet levels of awareness are simple, natural, and effective because they are in accord with all the laws of nature that reside at these deep levels.
5. **Wholeness moving within itself**: Advanced TM Techniques and the TM-Sidhi Programs are powerful APIs for bringing the calm dynamism and bliss of pure consciousness into daily activity.