jQuery

write less, do more.

Topics

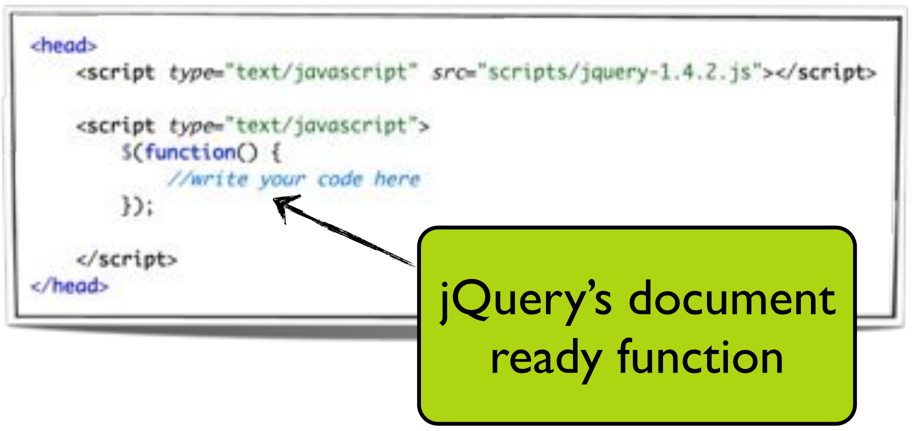
- Selectors
- The wrapped element set
- Manipulating elements
- Events
- Animations & Effects
- Utility functions
- jQuery plugins

jQuery UI topics

- Themes
- Effects
- Drag ‘n drop
- Widgets

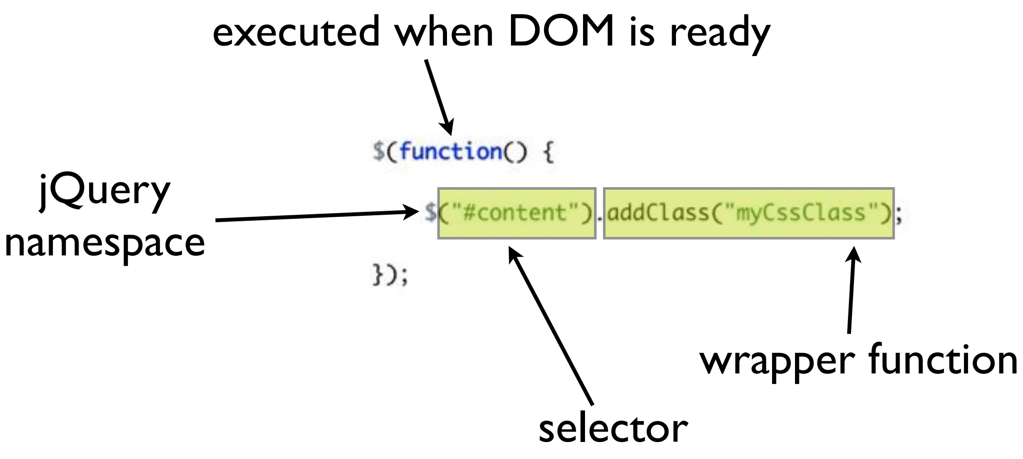
Introducing jQuery

Installing jQuery



```
<head>
    <script type="text/javascript" src="scripts/jquery-1.4.2.js"></script>

    <script type="text/javascript">
        $(function() {
            //write your code here
        });

    </script>
</head>
```

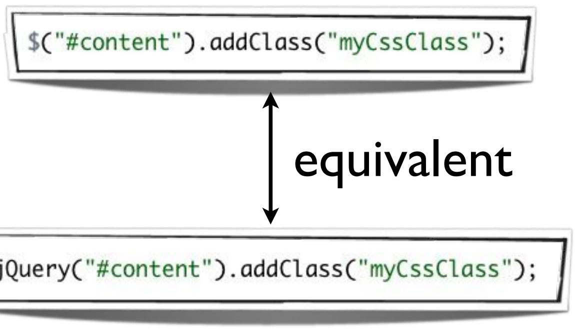
Understanding the Syntax



```
$(function() {
    $("#content").addClass("myCssClass");
});
```

Understanding the Syntax

- $ is just an alias for jQuery



```
$("#content").addClass("myCssClass");
```

```
jQuery("#content").addClass("myCssClass");
```

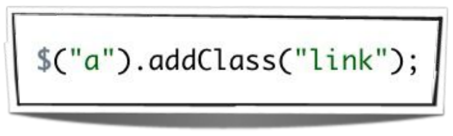
Utility functions

- Methods that are not executed on wrapped elements

```
$.trim("  some string  ");
```

Wrapped element set

- Selectors return a wrapped set of elements
- The wrapped set provides common methods

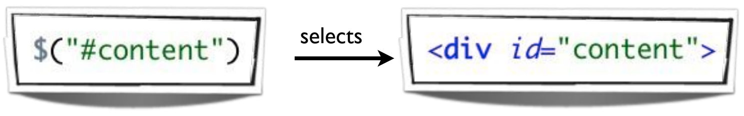


```
$("a").addClass("link");
```

```
1. Select all <a> elements
2. Add a CSS class to the wrapped set
```

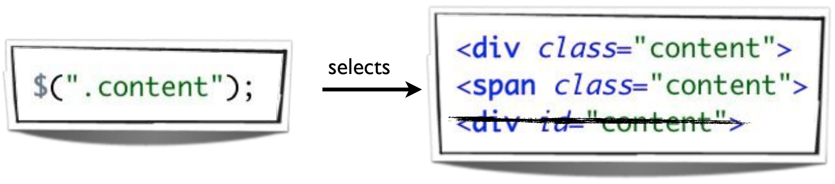
Selecting elements

By id



`$("#content")` → `<div id="content">`

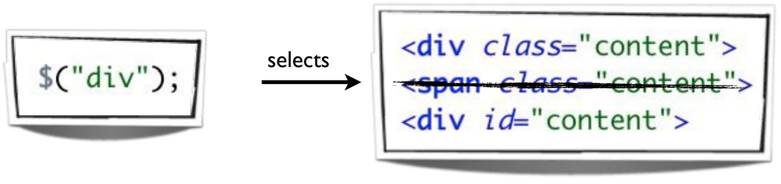
By CSS class



`$(".content");` →

`<div class="content">`
`<span class="content">`
~~`<div id="content">`~~

By type



`$("div");` →

`<div class="content">`
~~`<span class="content">`~~
`<div id="content">`

Selecting elements

Links within divs	`$("div a");`
Only direct children	`$("div > a");`
Divs with CSS class	`$("div.content");`
Input elements with attribute type set to ‘text’	`$("input[type='text']");`

Selecting by position

the first <a> element	`$("a:first");`
the last <a> element	`$("a:last");`
odd table rows	`$("tr:odd");`
even table rows	`$("tr:even");`
fifth table row	`$("tr:nth-child(5)");`

Special selectors

:button	All buttons <input type="submit"> <button>
:checkbox	All checkboxes input[type=checkbox]
:checked	Checkboxes and radio buttons that are checked
:contains(someString)	Elements containing text 'someString'
:disabled	Disabled elements
:enabled	Enabled elements
:file	File input elements
:selected	Selected <option> elements

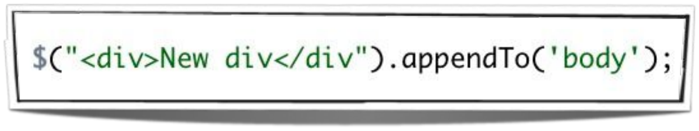
More selectors: http://api.jquery.com/category/selectors/

Lab

Selectors

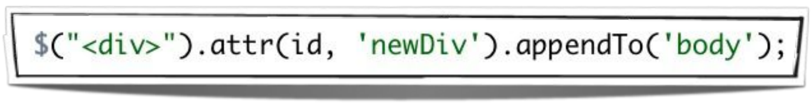
Generating HTML

- New elements can be created dynamically



```
$("<div>New div</div>").appendTo('body');
```

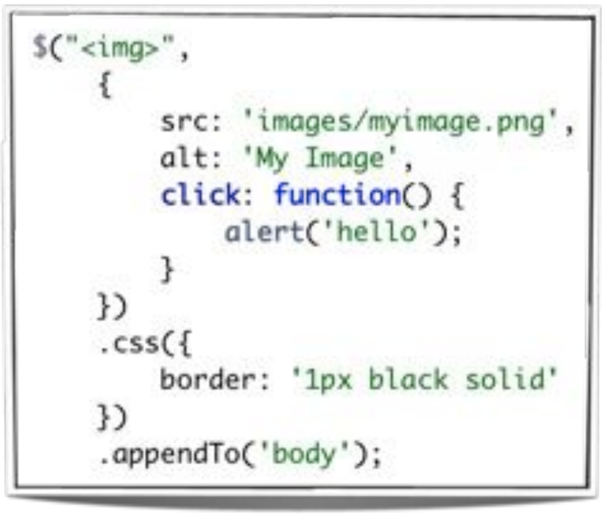
Empty elements can be written with single tag



```
$("<div>").attr(id, 'newDiv').appendTo('body');
```

Generating HTML

Complex elements can be created using an object argument



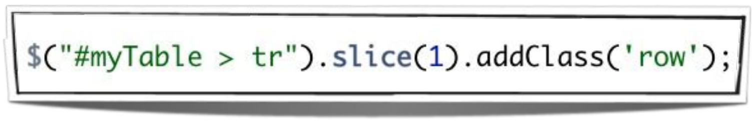
```
$("<img>",
    {
        src: 'images/myimage.png',
        alt: 'My Image',
        click: function() {
            alert('hello');
        }
    })
    .css({
        border: '1px black solid'
    })
    .appendTo('body');
```

Working with the Wrapped Element Set

size()	The number of elements in the set
get(index)	Get a specific element by index
eq(index)	Get a specific element by index as a new wrapped set
first()	Get the first element of the set
last()	Get the last element of the set
slice(start, end)	New wrapped set containing a slice of the original set
add(expression, context)	Add the elements of expression to the wrapped set
not(expression)	Creates a new wrapped set and removes elements matched by expression

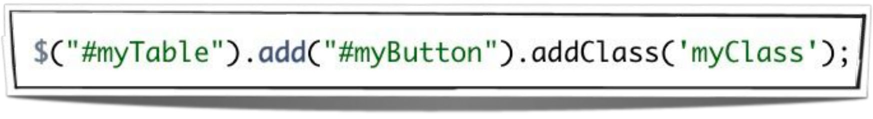
Wrapped set examples

Add class to all table rows except the header row



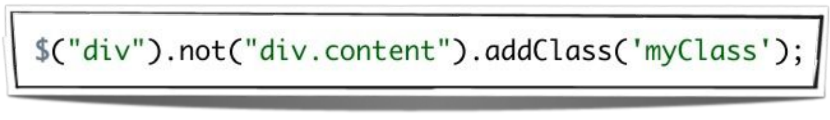
```
$("#myTable > tr").slice(1).addClass('row');
```

Select two elements by id



```
$("#myTable").add("#myButton").addClass('myClass');
```

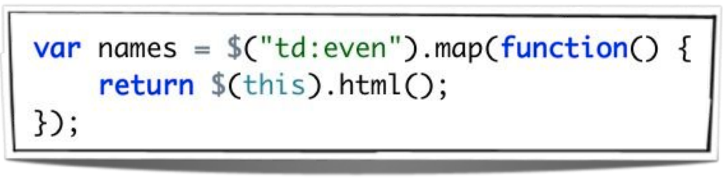
Divs that don’t have CSS class ‘content’



```
$("div").not("div.content").addClass('myClass');
```

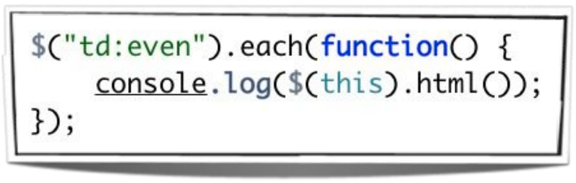
Traversing a wrapped set

Collects the returned values into
a new jQuery object



```
var names = $("td:even").map(function() {
    return $(this).html();
});
```

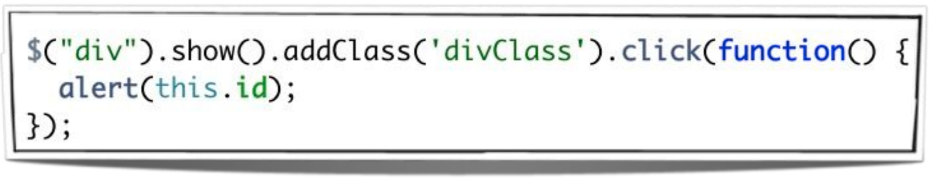
Execute function for each element



```
$("td:even").each(function() {
    console.log($(this).html());
});
```

Chaining functions

- Most function on the wrapped set can be chained



```
$("div").show().addClass('divClass').click(function() {
  alert(this.id);
});
```

Element attributes

Retrieve href attribute from links

```
var href = $("a").attr('href');
```

Set href attribute at links

```
$("a").attr('href', 'http://google.nl');
```

Set multiple attributes at once

```
$("input").attr(
{
    value: '',
    title: 'Please enter a value'
});
```

Custom element data

- Add data to any DOM element
- Prevents the need for global variables

set data

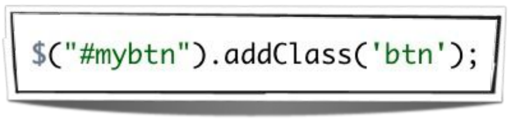
```
$("#mybtn").data('name', 'My Name');
```

get data

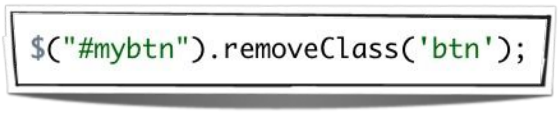
```
var value = $("#mybtn").data('name');
```

CSS classes

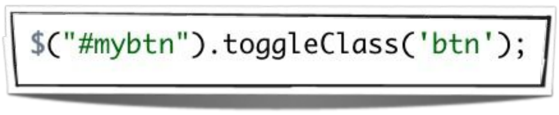
- Best way to change element visuals is adding/removing CSS classes



```
$("#mybtn").addClass('btn');
```



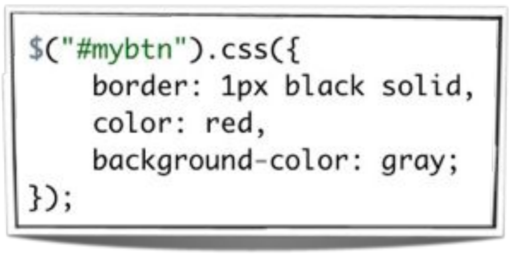
```
$("#mybtn").removeClass('btn');
```



```
$("#mybtn").toggleClass('btn');
```

Setting styles directly

- Prefer CSS classes because of re-usability

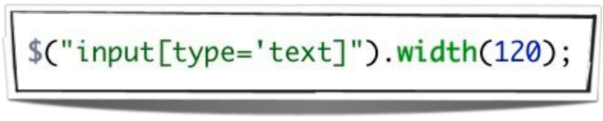


```
$("#mybtn").css({
    border: 1px black solid,
    color: red,
    background-color: gray;
});
```

Width and height

- Convenience method for setting/ getting width and height
  - identical to .css(“width”)

Set all text input fields to a width of 120px

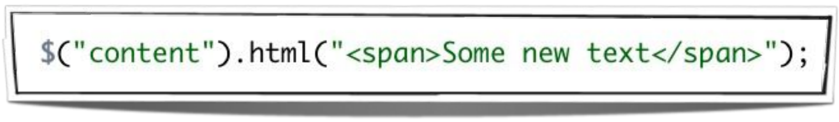


```
$("input[type='text]").width(120);
```

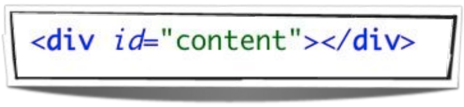
Element content

- Changing element content is often used with Ajax
  - show result in empty div

Set the content div’s content



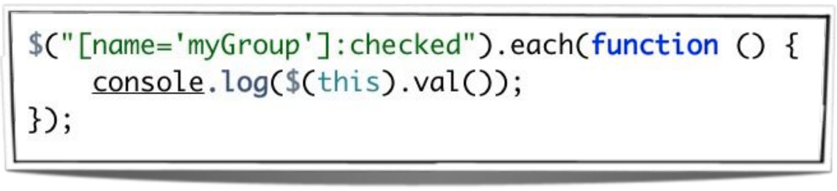
```
$("content").html("<span>Some new text</span>");
```



```
<div id="content"></div>
```

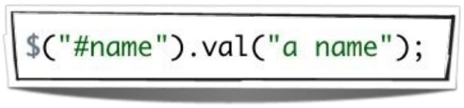
Working with forms

Get values of checked checkboxes



```
$("[name='myGroup']:checked").each(function () {
    console.log($(this).val());
});
```

Set value of a text input

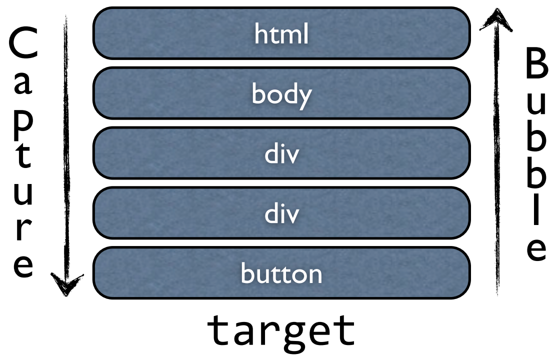


```
$("#name").val("a name");
```

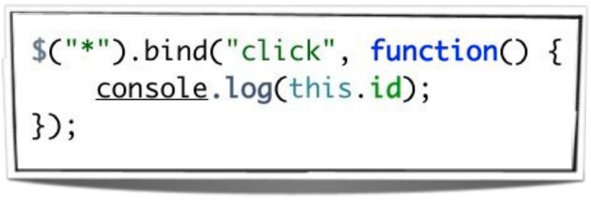
jQuery
write less, do more.

Events

Event bubbling

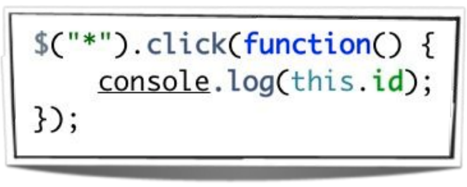


Binding events

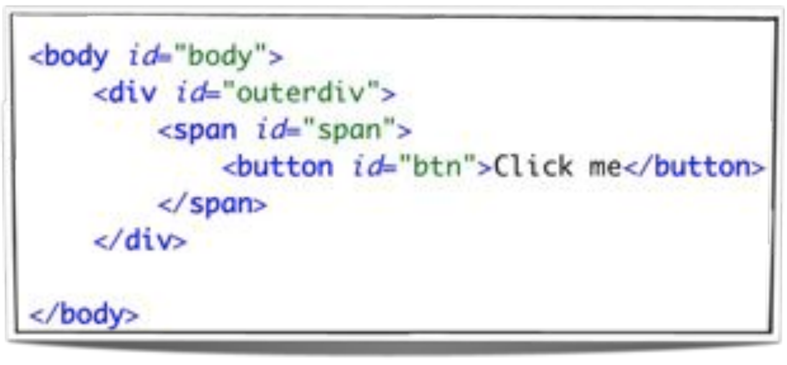


```
$("*").bind("click", function() {
    console.log(this.id);
});
```

or



```
$("*").click(function() {
    console.log(this.id);
});
```



```
<body id="body">
    <div id="outerdiv">
        <span id="span">
            <button id="btn">Click me</button>
        </span>
    </div>

</body>
```

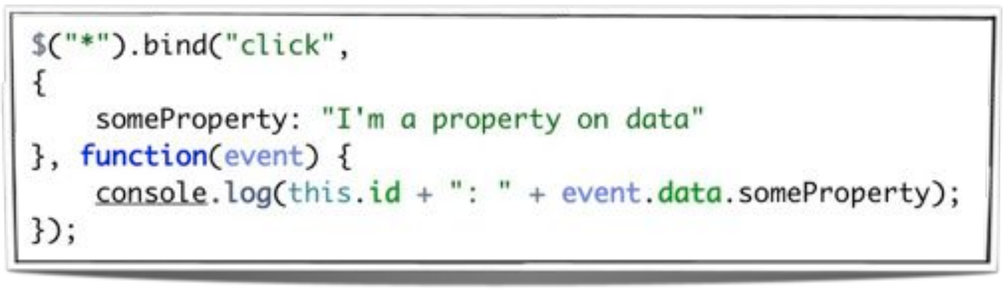
Prints:

```
html
body
outerdiv
span
btn
```

The bind method

```
bind(eventType, data, handler)
```

eventType	Space separated list of event types to respond to
data	Caller supplied data that's attached to the Event instance (optional)
handler	Function to handle the event



```
$("*").bind("click",
{
    someProperty: "I'm a property on data"
}, function(event) {
    console.log(this.id + ": " + event.data.someProperty);
});
```

event types

```
blur        focusout    mouseleave  scroll
change      keydown     mousemove   select
click       keypress    mouseout    submit
dblclick    keyup       mouseover   unload
error       load        mouseup
focus       mousedown   ready
focusin     mouseenter  resize
```

Removing handlers

```
$("btn").unbind("click");
```

Execute once, unbinds automatically after handling

```
$("btn").one("click", function() {
    //do something
});
```

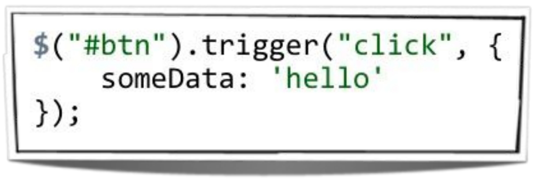
Live event handling

- Bind handlers to elements that might not be on the page yet
- Handler is added to newly created elements that match the selector

```
$(".btn").live("click", function() {
    //do something
});

$("<button>").addClass("btn").appendTo("body");
```

Trigger events from code

- Mimic user interaction by triggering events from code



```
$("#btn").trigger("click", {
    someData: 'hello'
});
```

Lab

TODO application

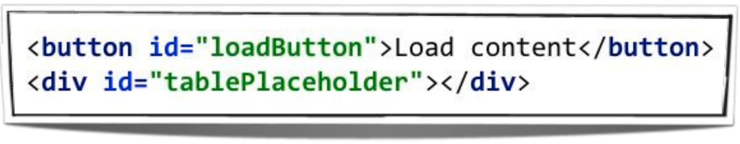
jQuery write less, do more.

Ajax

Two types of Ajax

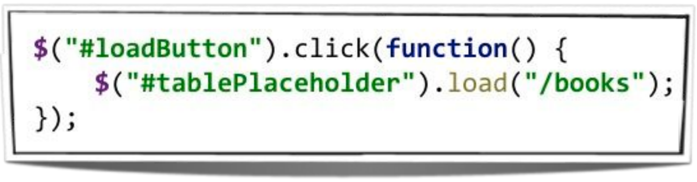
- Data centric
  - server sends back JSON
  - client does all the rendering
- Content centric
  - server sends back HTML
  - client just places the new content in the page

Content centric Ajax

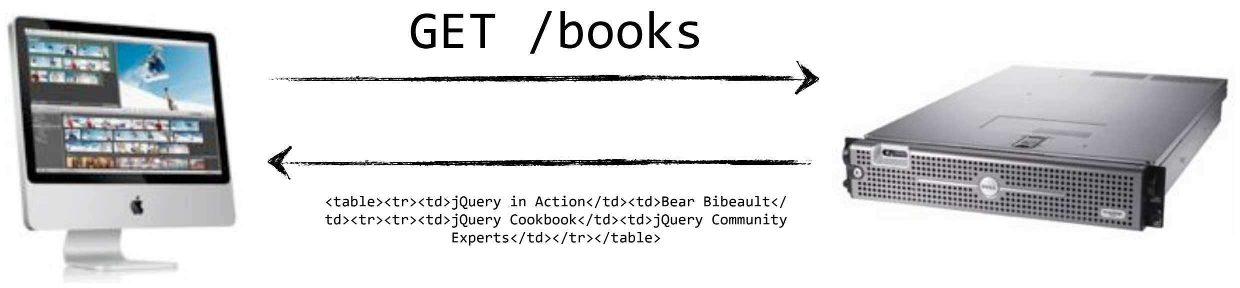


```html
<button id="loadButton">Load content</button>
<div id="tablePlaceholder"></div>
```

```
load(url, params, callback)
```



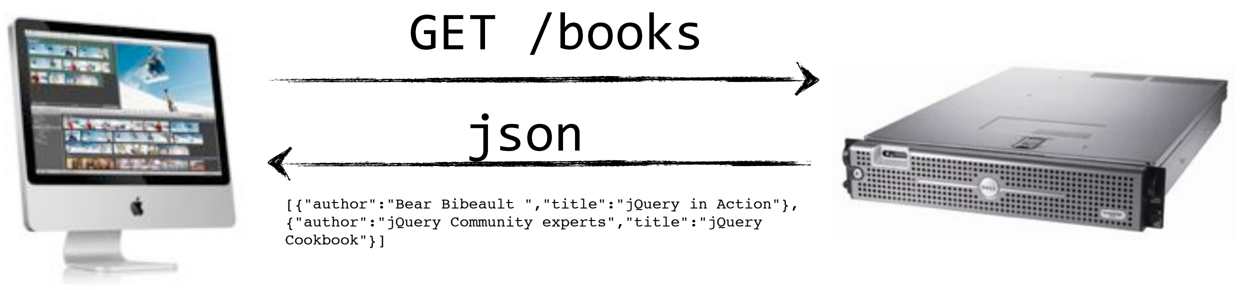
```javascript
$("#loadButton").click(function() {
    $("#tablePlaceholder").load("/books");
});
```



```
<table><tr><td>jQuery in Action</td><td>Bear Bibeault</
td><tr><tr><td>jQuery Cookbook</td><td>jQuery Community
Experts</td></tr></table>
```

Data centric Ajax

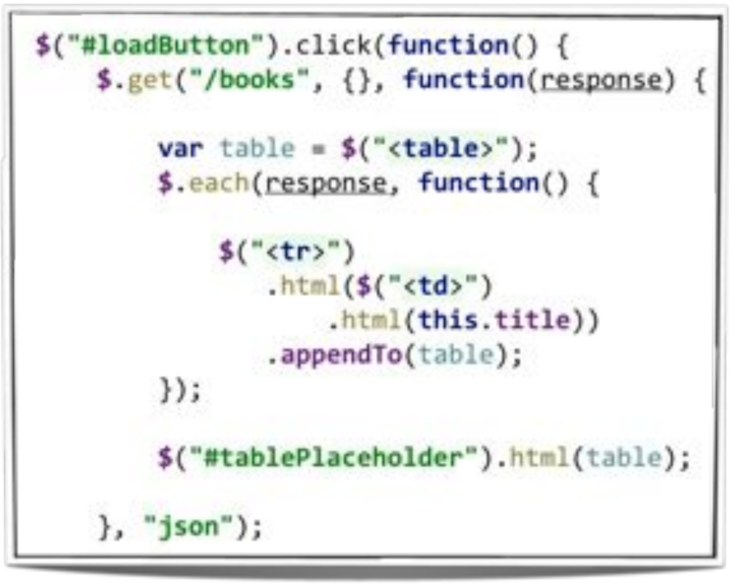
- Now the client is responsible for generating html
  - more work (client side)
  - more flexible



```
[{"author":"Bear Bibeault ","title":"jQuery in Action"},
{"author":"jQuery Community experts","title":"jQuery
Cookbook"}]
```

Data centric Ajax

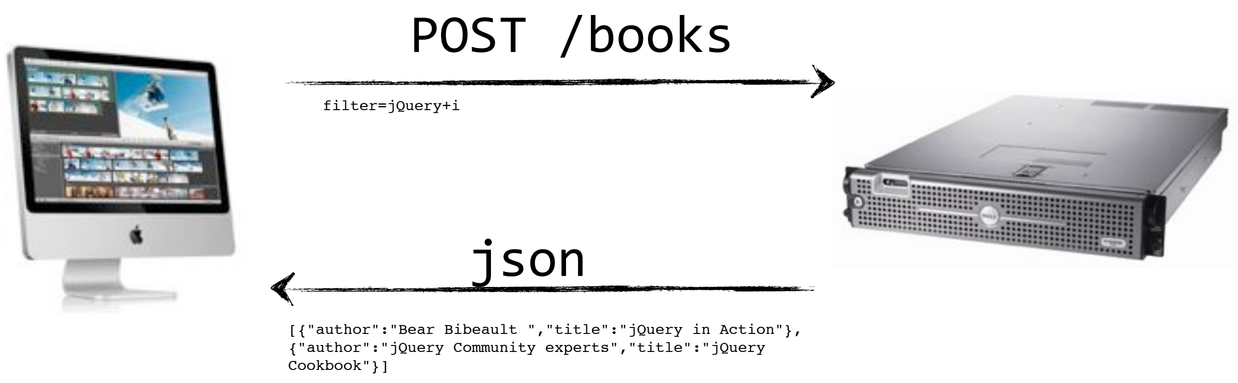
```
$.get(url, params, callback, format)
$.getJSON(url, params, callback)
```



```
$("#loadButton").click(function() {
    $.get("/books", {}, function(response) {

        var table = $("<table>");
        $.each(response, function() {

            $("<tr>")
                .html($("<td>")
                    .html(this.title))
                .appendTo(table);
        });

        $("#tablePlaceholder").html(table);

    }, "json");
```

Data centric Ajax

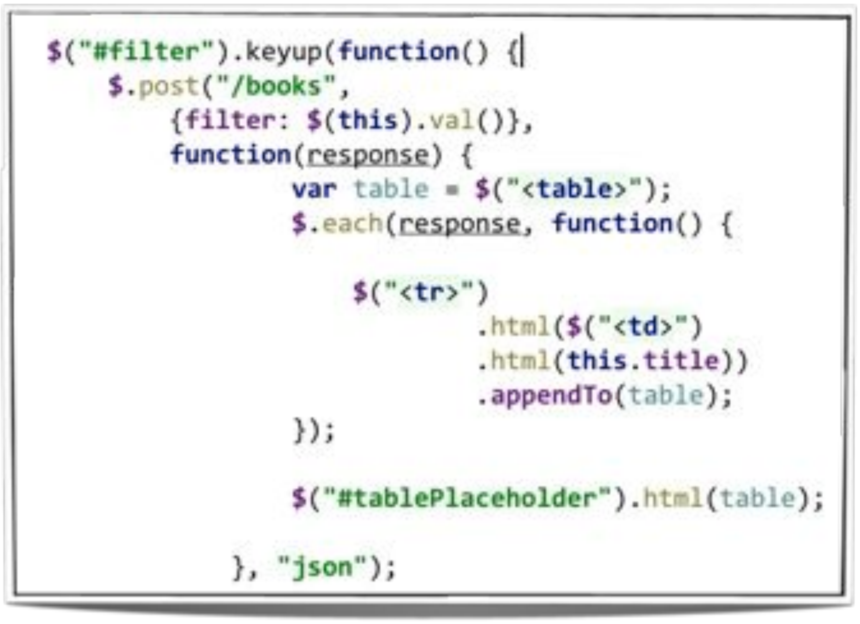
- Send URL encoded data to the server
- Retrieve JSON data for rendering



```
[{"author":"Bear Bibeault ","title":"jQuery in Action"},
{"author":"jQuery Community experts","title":"jQuery
Cookbook"}]
```

Posting data

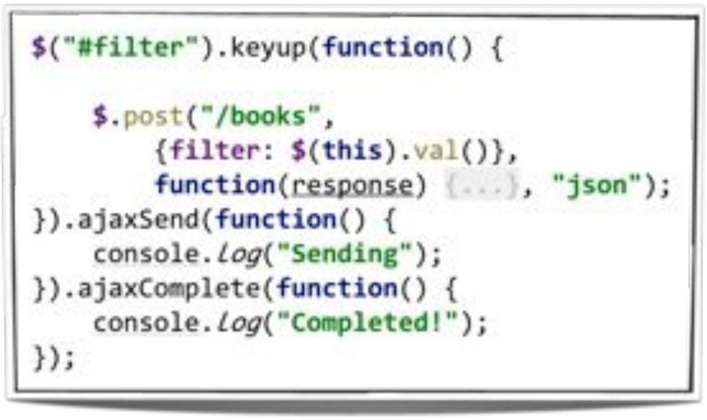
```
$.post(url, params, callback, format)
```



```
$("#filter").keyup(function() {
    $.post("/books",
        {filter: $(this).val()},
        function(response) {
                var table = $("<table>");
                $.each(response, function() {

                    $("<tr>")
                            .html($("<td>")
                            .html(this.title))
                            .appendTo(table);
                });

                $("#tablePlaceholder").html(table);

            }, "json");
```

Ajax Events

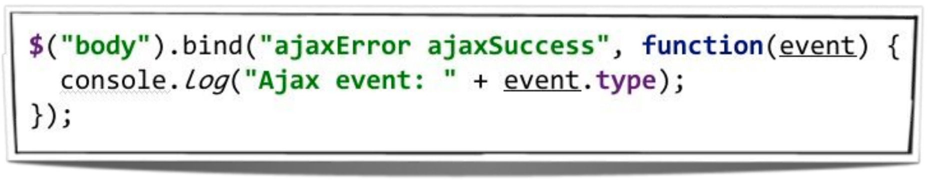
- Bind events using wrapper functions



```
$("#filter").keyup(function() {

    $.post("/books",
        {filter: $(this).val()},
        function(response) {...}, "json");
}).ajaxSend(function() {
    console.log("Sending");
}).ajaxComplete(function() {
    console.log("Completed!");
});
```

Ajax Events

- Bind events using bind
  - `ajaxStart, ajaxSend, ajaxSuccess, ajaxError, ajaxComplete, ajaxStop`



```
$("body").bind("ajaxError ajaxSuccess", function(event) {
  console.log("Ajax event: " + event.type);
});
```

Lab

Twitter client

Animation

Showing and hiding elements

```
$("#hideme").hide();
```

```
$("#hideme").show();
```

```
$("#hideme").toggle();
```

Animated

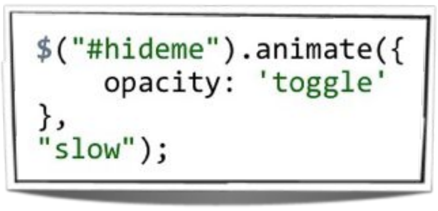
```
$("#hideme").toggle("slow");
```

```
$("#hideme").fadeOut("slow");
```

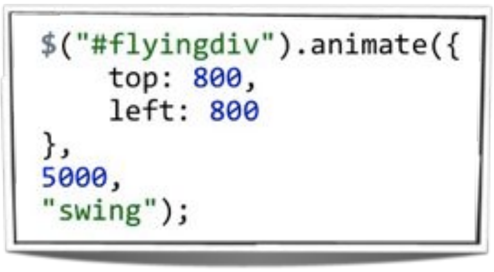
```
$("#hideme").fadeIn("slow");
```

Custom animations

```
animate(properties, duration, easing, callback)
```



```
$("#hideme").animate({
    opacity: 'toggle'
},
"slow");
```

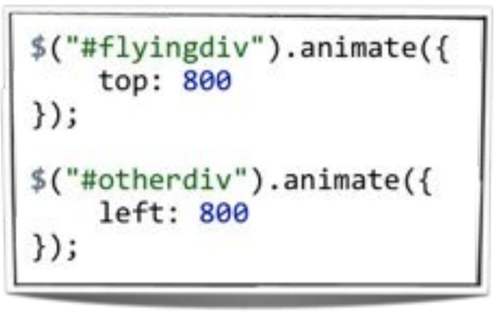


```
$("#flyingdiv").animate({
    top: 800,
    left: 800
},
5000,
"swing");
```

Custom animations

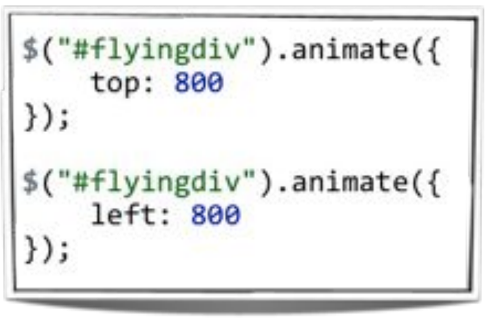
- The animate method is non-blocking
  - multiple animations on the same object are queued however

Executed in parallel



```
$("#flyingdiv").animate({
    top: 800
});

$("#otherdiv").animate({
    left: 800
});
```

Executed in serial



```
$("#flyingdiv").animate({
    top: 800
});

$("#flyingdiv").animate({
    left: 800
});
```

jQuery
write less, do more.

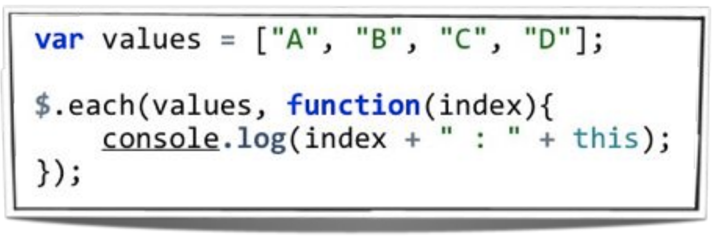
Utilities

Utility functions

- Operate on other data than a wrapped set
- `$.functionName()`
  - e.g. `$.trim(“ a string “)`

Iterating collections

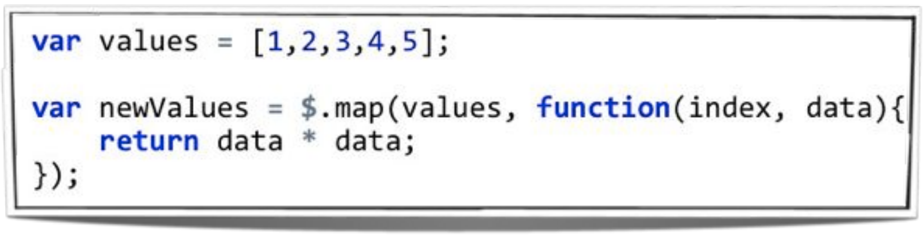
```
$.each(container, callback)
```



```
var values = ["A", "B", "C", "D"];

$.each(values, function(index){
    console.log(index + " : " + this);
});
```

Iterating collections

```
$.map(container, callback)
```

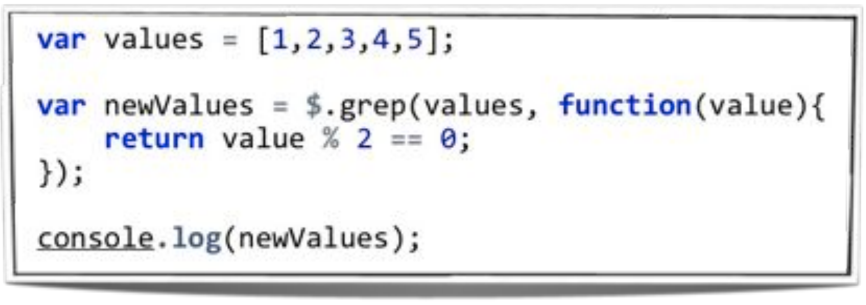


```
var values = [1,2,3,4,5];

var newValues = $.map(values, function(index, data){
    return data * data;
});
```

- Creates a new collection containing the returned values from the map function

Filtering collections

```
$.grep(container, callback)
```



```
var values = [1,2,3,4,5];

var newValues = $.grep(values, function(value){
    return value % 2 == 0;
});

console.log(newValues);
```

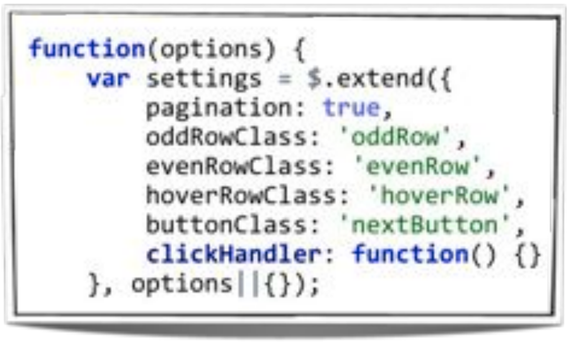
```
[2, 4]
```

Extending objects

- Mimics inheritance
  - copy all values from the base object to the extending object
  - often used for settings defaults in argument objects

Extending objects

```
$.extend(deep, target, source1,
    source 2, ... sourceN)
```



```
function(options) {
    var settings = $.extend({
        pagination: true,
        oddRowClass: 'oddRow',
        evenRowClass: 'evenRow',
        hoverRowClass: 'hoverRow',
        buttonClass: 'nextButton',
        clickHandler: function() {}
    }, options||{});
```

- Creates a new object ‘settings’ based on the ‘options’ argument object

jQuery
write less, do more.

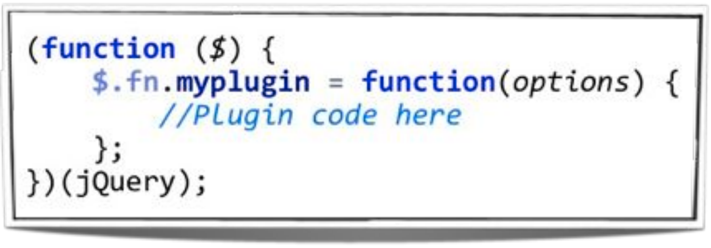
Writing Plugins

Introduction

- jQuery is designed to be extendible
  - a small core
  - lots of plugins available
  - write your own plugin easily
- Plugins can provide wrapper functions or utility functions

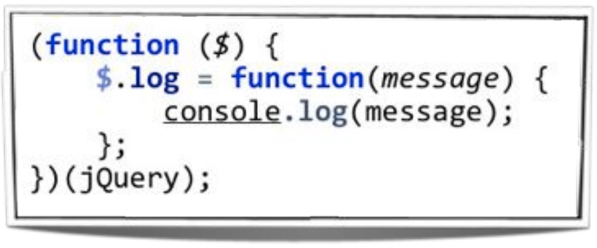
Development guidelines

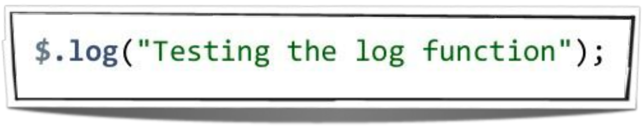
- Files should be named as follows
  - `jquery.[plugin-name].1.0.js`
- Avoid $ namespace collisions



```
(function ($) {
    $.fn.myplugin = function(options) {
        //Plugin code here
    };
})(jQuery);
```

Adding utility functions


```
(function ($) {
    $.log = function(message) {
        console.log(message);
    };
})(jQuery);
```

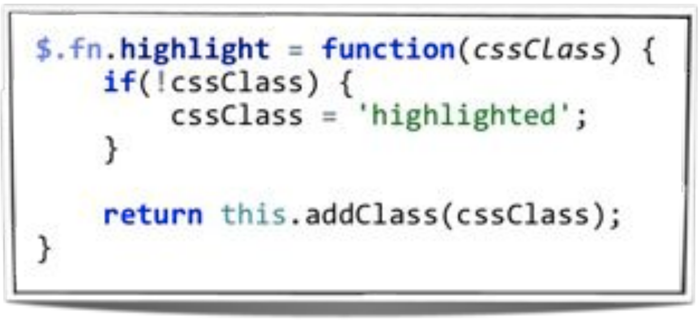

```
$.log("Testing the log function");
```

Wrapper functions

```
$.fn.[function_name] = function(arguments) {
 //Do something
 return this;
};
```

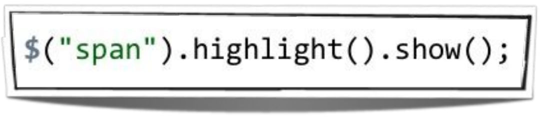
- ‘this’ represents the wrapped set
- return the (modified) wrapped set to support chaining

Wrapper function example



```
$.fn.highlight = function(cssClass) {
    if(!cssClass) {
        cssClass = 'highlighted';
    }

    return this.addClass(cssClass);
}
```

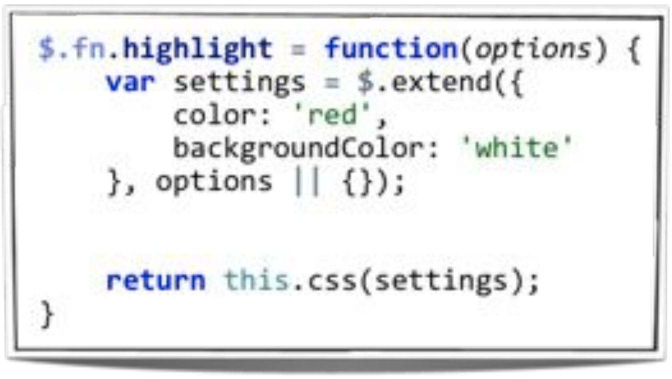
Chaining still works



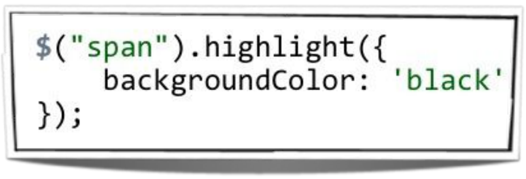
```
$("span").highlight().show();
```

Flexible arguments

- Use an Object to pass in arguments
  - use `$.extend` to provide default values



```
$.fn.highlight = function(options) {
    var settings = $.extend({
        color: 'red',
        backgroundColor: 'white'
    }, options || {});

    return this.css(settings);
}
```

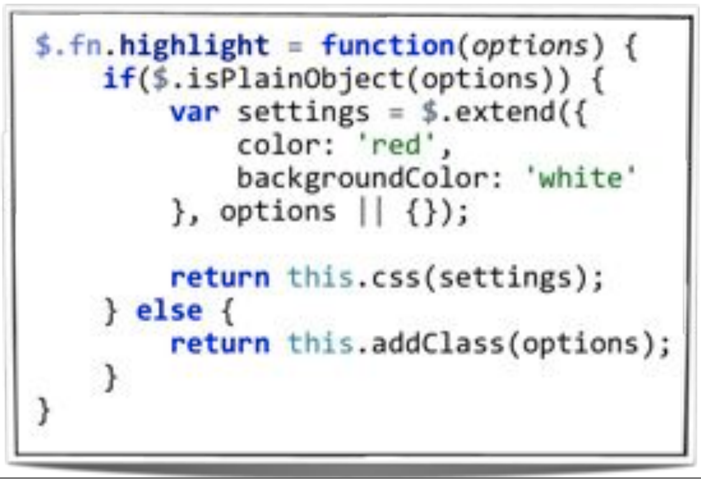


```
$("span").highlight({
    backgroundColor: 'black'
});
```

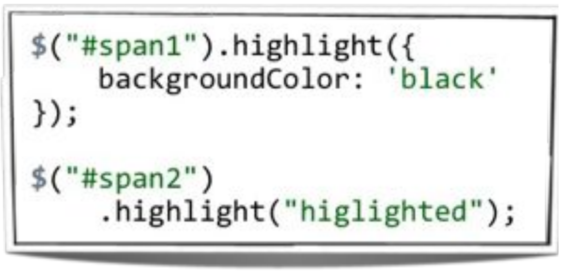
More flexible arguments

```
highlight(cssClass)
highlight({options})
```

Emulate operator overloading



```
$.fn.highlight = function(options) {
    if($.isPlainObject(options)) {
        var settings = $.extend({
            color: 'red',
            backgroundColor: 'white'
        }, options || {});

        return this.css(settings);
    } else {
        return this.addClass(options);
    }
}
```



```
$("#span1").highlight({
    backgroundColor: 'black'
});

$("#span2")
    .highlight("higlighted");
```

Lab

Table plugin

Introduction

jQuery UI

- Official plugin
  - widgets
  - theming framework
  - effects
  - interactions (e.g. drag & drop)

Downloading jQuery UI

- Create a customized download
  - choose or create a theme
  - select components to include

Build Your Download

UI Core

Interactions

Widgets

Download

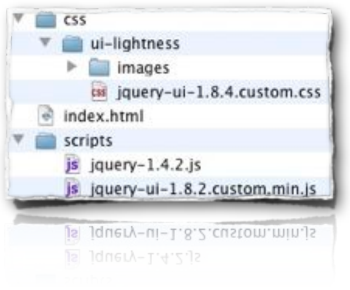
Downloading jQuery UI

- Create a customized download
  - choose or create a theme
  - select components to include

Configuring jQuery UI

```
<link type="text/css"
      href="css/ui-lightness/jquery-ui-1.8.4.custom.css"
      rel="stylesheet" />
```

```
<script type="text/javascript"
        src="scripts/jquery-ui-1.8.2.custom.min.js">
</script>
```



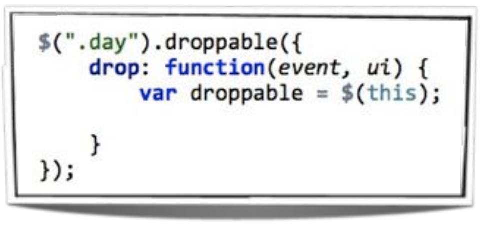
Drag & Drop

- Draggable and Droppable
  - draggables can be dropped anywhere by default
  - use a droppable to get an event handler

```
$(".todrag").draggable();
```

```
$(".droparea").droppable();
```

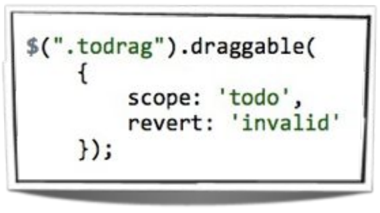
Handling events



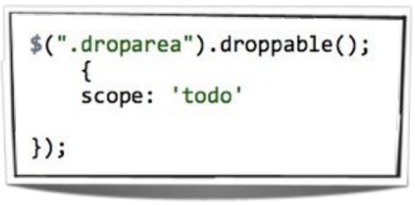
```
$(".day").droppable({
    drop: function(event, ui) {
        var droppable = $(this);

    }
});
```

Limit dropping

- Drops can be limited to specific drop points



```
$(".todrag").draggable(
    {
        scope: 'todo',
        revert: 'invalid'
    });
```



```
$(".droparea").droppable();
    {
    scope: 'todo'

});
```