JavaScript 3: Extending JavaScript with jQuery

Chapter 10

Randy Connolly and Ricardo Hoar

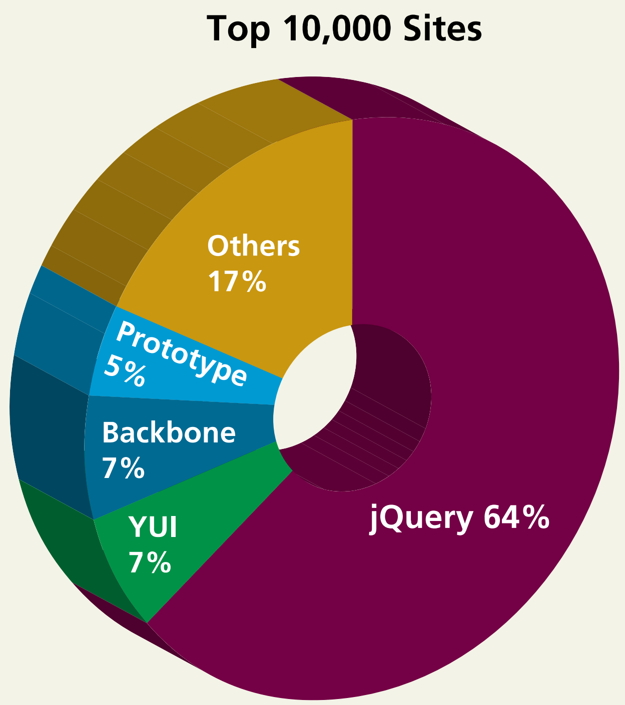
Fundamentals of Web Development

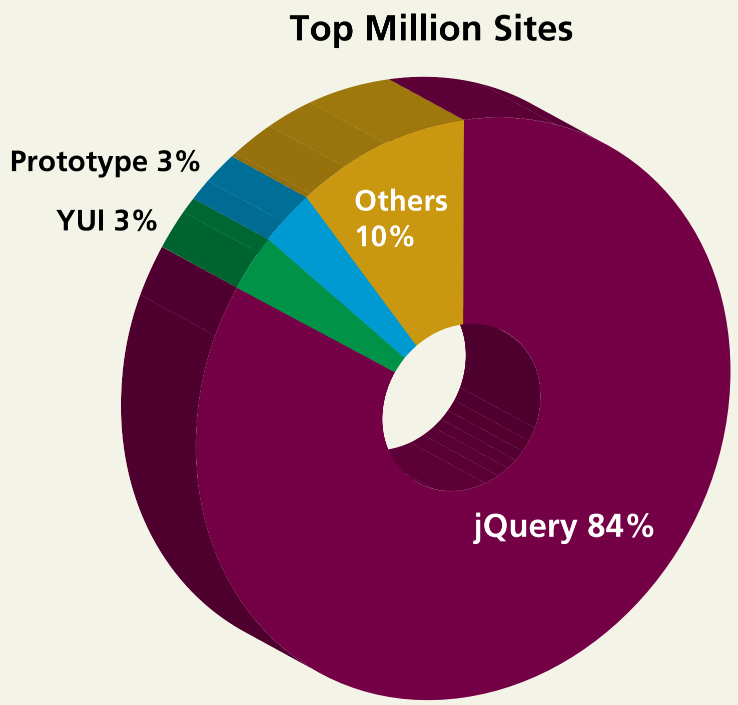
Chapter 10

Chapter 10

jQuery Foundations

A popular framework





jQuery Foundations

Including jQuery

Use a Content Delivery Network (CDN)

```
<script src="http://code.jquery.com/jquery-3.1.0.min.js">
</script>
```

Use a failsafe in case the CDN is down

jQuery Foundations

jQuery Selector

Remember getElementByID()…

- The power of jQuery resides in the function named **jQuery()**.
- There's also an alias for this function named **$()** .
- You can combine CSS selectors with the $() notation to select DOM objects that match CSS attributes

jQuery Foundations

jQuery Selector

```
/* selecting using regular JavaScript */

var node = document.getElementById("here");
var link = document.querySelectorAll("ul li");

/* equivalent selection using jQuery */

var node = $("#here");

var link = $("ul li");
```

- The **$()** function always returns a set of results

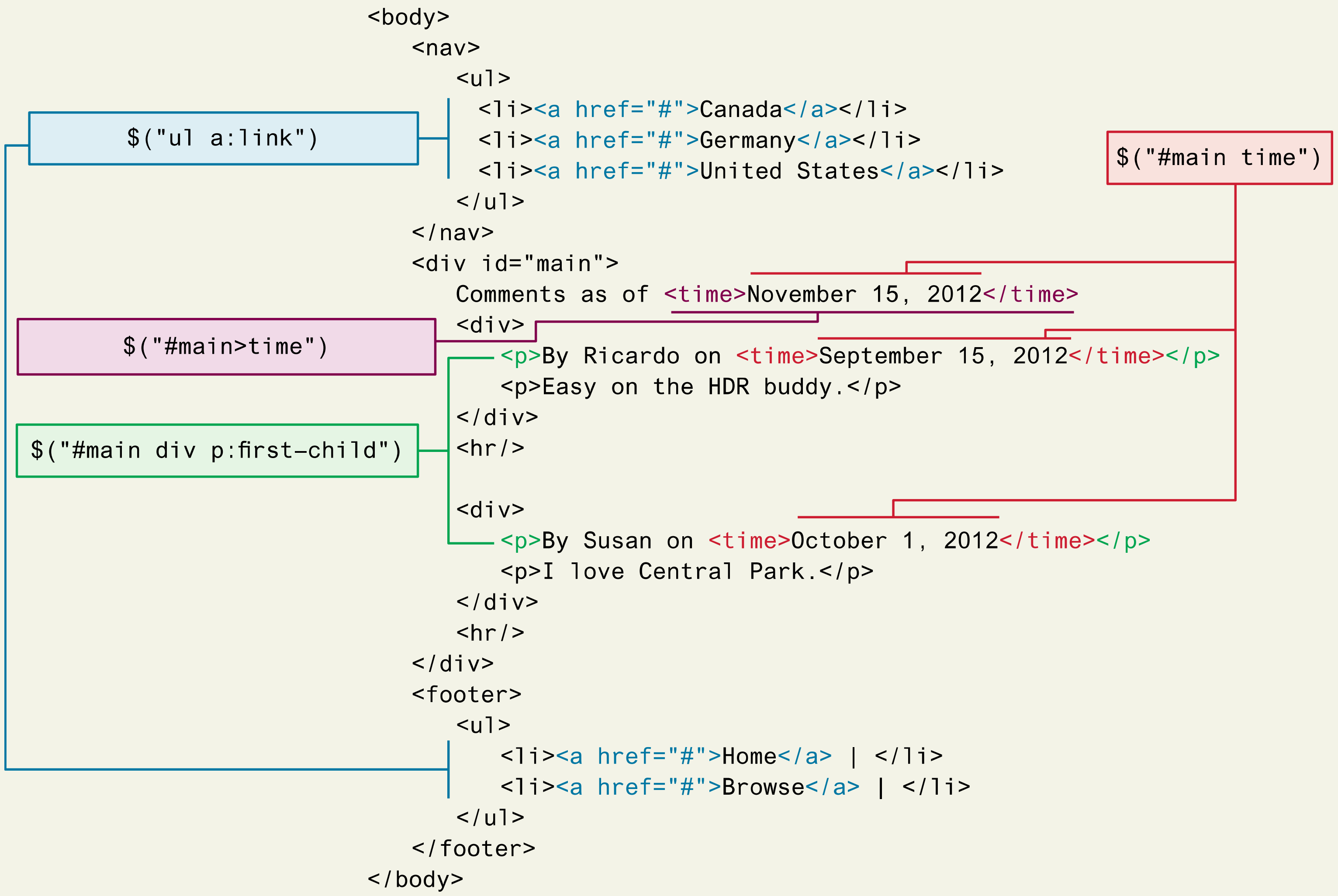
jQuery Foundations

Basic Selectors

- **$("*")**—Universal selector matches all elements (and is slow).
- **$("tag")**—Element selector matches all elements with the given element name.
- **$(".class")**—Class selector matches all elements with the given CSS class.
- **$("#id")**—Id selector matches all elements with a given HTML id attribute.

jQuery Foundations

Some examples



```
<body>
   <nav>
      <ul>
        <li><a href="#">Canada</a></li>
        <li><a href="#">Germany</a></li>
        <li><a href="#">United States</a></li>
      </ul>
   </nav>
   <div id="main">
      Comments as of <time>November 15, 2012</time>
      <div>
         <p>By Ricardo on <time>September 15, 2012</time></p>
         <p>Easy on the HDR buddy.</p>
      </div>
      <hr/>

      <div>
         <p>By Susan on <time>October 1, 2012</time></p>
         <p>I love Central Park.</p>
      </div>
      <hr/>
   </div>
   <footer>
      <ul>
         <li><a href="#">Home</a> | </li>
         <li><a href="#">Browse</a> | </li>
      </ul>
   </footer>
</body>
```

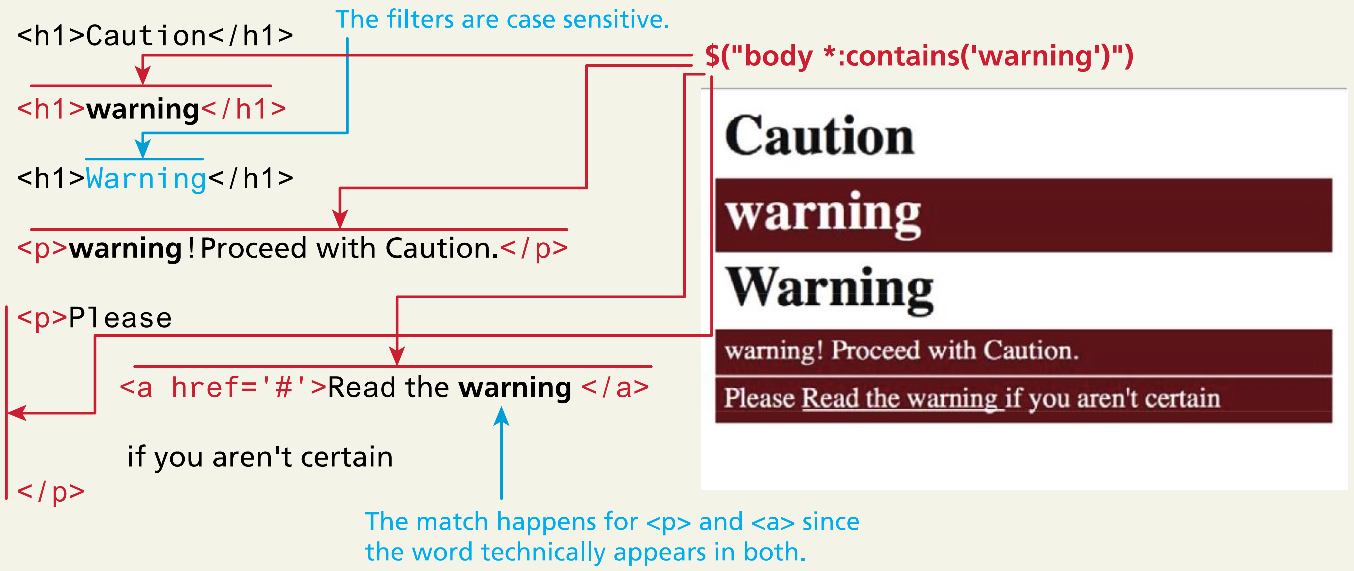
jQuery Foundations

Advanced Selectors (Go back to Chapter 4 in CSS for a refresher)

- Attribute Selector
- Pseudo-Element Selector
- Contextual Selector
- jQuery Filters
- Form Selectors

jQuery Foundations

jQuery's content filter selection



The filters are case sensitive.
$("body *:contains('warning')")
The match happens for <p> and <a> since the word technically appears in both.

```
<h1>Caution</h1>
<h1>warning</h1>
<h1>Warning</h1>
<p>warning! Proceed with Caution.</p>
<p>Please
    <a href='#'>Read the warning </a>
    if you aren't certain
</p>
```

Caution

warning

Warning

warning! Proceed with Caution.

Please Read the warning if you aren't certain

jQuery Foundations

Common element Manipulations - **HTML attributes**

- We can both set and get an attribute value by using the **attr()** method.

```
// link is assigned the href attribute of the first <a> tag

var link = $("a").attr("href");

// change all links in the page to http://funwebdev.com

$("a").attr("href","http://funwebdev.com");

// change the class for all images on the page to fancy

$("img").attr("class","fancy");
```

jQuery Foundations

Common element Manipulations - **HTML properties**

- The **prop()** method is the preferred way to retrieve and set the value of a property.

```
<input class="meh" type="checkbox" checked="checked">
```

```
var theBox = $(".meh");

theBox.prop("checked"); // evaluates to TRUE
```

jQuery Foundations

Common element Manipulations – **Changing CSS**

- jQuery provides the extremely intuitive **css()** method.

```
var color = $("#element").css("background-color"); // get the color

$("#element").css("background-color", "red"); // set color to red
```

Chapter 10

Event Handling in jQuery

Just like JavaScript, jQuery supports creation and management of listeners/handlers for JavaScript events.

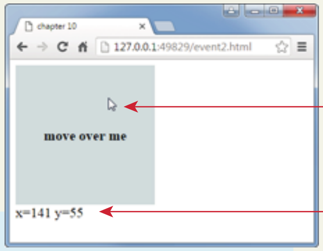
While pure JavaScript uses the **addEventListener()** method, jQuery has **on()** and **off()** methods as well as shortcut methods to attach events.

Event Handling in jQuery

Binding and Unbinding Events

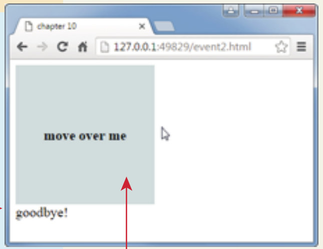
Notice that we are chaining together multiple event handlers in one statement. This is a common programming style used by jQuery programmers.

```
$(".panel")
   .on("mousemove",function (e) {
      $("#message").html("x=" + e.pageX + " y=" + e.pageY);
   })

   .on("mouseleave",function (e) {
      $("#message").html("goodbye!");
   })

   .on("click",function () {
      $("#message").html("stopped move reporting");
      $(".panel").off("mousemove");
   });
```

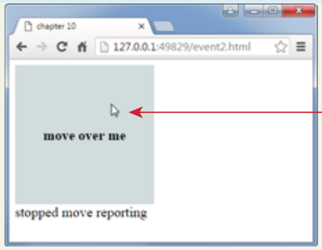


When user moves mouse over element, then display x, y coordinates.

When user moves mouse outside of element, then indicate this.



But even though the mouse is gone, the panel is still listening for future mouse over events.



However, when the user clicks on the panel, we turn off its listener for mouse moves. Thus future moves will not trigger the mouse move event.

Event Handling in jQuery

Page Loading

```
$(document).ready(function() {
        // set up listeners knowing page loads before this runs
        $("#example").click(function () {
        $("#message").html("you clicked");
      });
});
```

Or the even simpler

```
      $(function () {
      …
      });
```

Chapter 10

DOM Manipulation

Creating Nodes

```
// pure JavaScript way

var jsLink = document.createElement("a");
jsLink.href = "http://
www.funwebdev.com";
jsLink.innerHTML = "Visit Us";
jsLink.title = "JS";



// jQuery version 1

var link1 = $('<a href="http://funwebdev.com"
                    title="jQuery">Visit Us</a>');
```

DOM Manipulation

Creating Nodes

```
// jQuery version 2
var link2 = $('<a></a>');
link2.attr("href","http://funwebdev.com");
link2.attr("title","jQuery verbose");
link2.html("Visit Us");


// version 3
$('<a>', {
                href: 'http://funwebdev.com',
                title: 'jQuery',
                text: 'Visit Us'
});
```

DOM Manipulation

Adding DOM Elements

```
<div class="dest">
existing content
</div>
```

```
var link = $('<a href="http://funwebdev.com">Fun</a>');
```

`$(".dest").append(link);`

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

`$(".dest").prepend(link);`

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

`link.appendTo($(".dest"));`

```
<div class="dest">
existing content
<a href="http://funwebdev.com">Fun</a>
</div>
```

`link.prependTo($(".dest"));`

```
<div class="dest">
<a href="http://funwebdev.com">Fun</a>
existing content
</div>
```

`$(".dest").before(link);`

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

`$(".dest").after(link);`

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

`link.insertBefore($(".dest"));`

```
<a href="http://funwebdev.com">Fun</a>
<div class="dest">
existing content
</div>
```

`link.insertAfter($(".dest"));`

```
<div class="dest">
existing content
</div>
<a href="http://funwebdev.com">Fun</a>
```

DOM Manipulation

Wrapping Existing DOM in New Tags

- Wrap all elements matched by a selector within a new element using wrap().

DOM Manipulation

Wrapping Existing DOM in New Tags

```
<div class="external-links">
    <div class="gallery">Uffuzi Museum</div>
    <div class="gallery">National Gallery</div>
    <div class="link-out">funwebdev.com</div>
</div>
```

```
$(".gallery").wrap('<div class="galleryLink"><div>');
```

```
<div class="external-links">
<div class="galleryLink">
<div class="gallery">Uffuzi Museum</div>
</div>
<div class="galleryLink">
<div class="gallery">National Gallery</div>
</div>
<div class="link-out">funwebdev.com</div>
</div>
```

DOM Manipulation

Wrapping Existing DOM in New Tags

```
<div class="external-links">
    <div class="gallery">Uffuzi Museum</div>
    <div class="gallery">National Gallery</div>
    <div class="link-out">funwebdev.com</div>
</div>
```

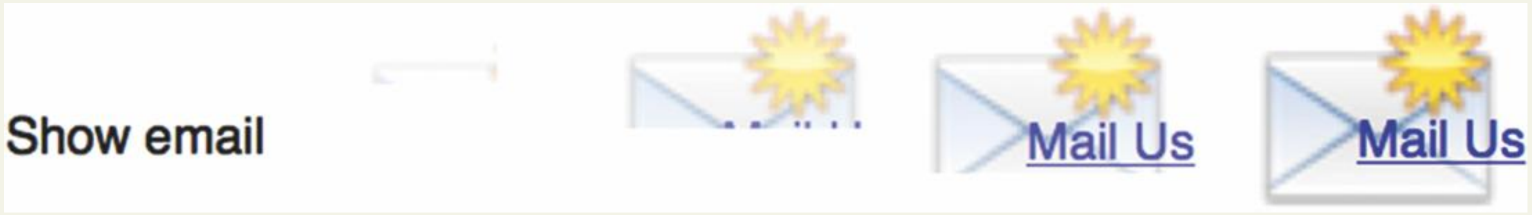
```
$(".gallery").wrap(function() {
    return "<div class='galleryLink' title='Visit " +  $(this).html() + "'></div>";
});
```

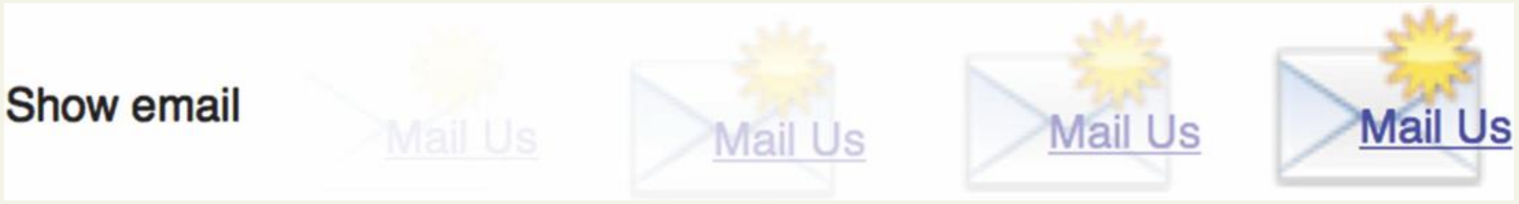
```
<div class="external-links">
  <div class="galleryLink" title="Visit Uffuzi Museum">
  <div class="gallery">Uffuzi Museum</div>
  </div>
  <div class="galleryLink" title="Visit National Gallery">
  <div class="gallery">National Gallery</div>
  </div>
  <div class="link-out">funwebdev.com</div>
</div>
```

Chapter 10

Effects and Animation

Show() and fadeIn()



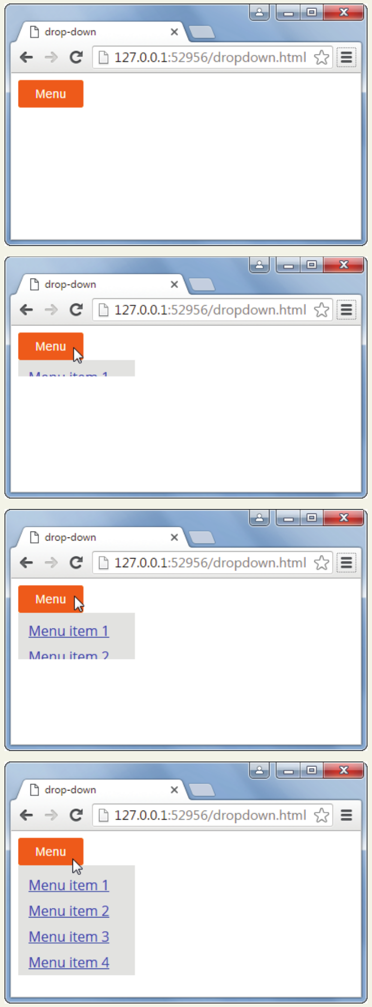


Effects and Animation

Using slide()

```
<button id="menuBtn">Menu</button>
<ul id="menu">
    <li><a href="#">Menu item 1</a></li>
    <li><a href="#">Menu item 2</a></li>
    <li><a href="#">Menu item 3</a></li>
    <li><a href="#">Menu item 4</a></li>
</ul>
```

```
$(function () {
    $("#menu").hide();      When page loads, hide the list.

    $("#menuBtn").on("mouseenter", function () {
        $("#menu").slideDown(500);
    });                Slide list down in 0.5 sec when mouse
                       hovers over it.


    $("#menuBtn").on("mouseleave", function () {
        $("#menu").slideUp(300);
    });                Slide list up faster when mouse is no
                       longer hovering over it.

});
```



Effects and Animation

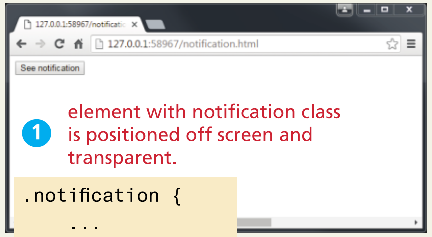
Raw Animation

$("#box").**animate**({left: '495px'});

- Describes a final state in CSS.
- The state before defines where the animations starts.
- Animate() has many parameters including:
  - Duration
  - Step
  - Done
  - …

Effects and Animation

Raw Animation

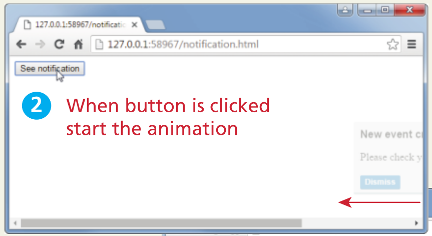


1 element with notification class is positioned off screen and transparent.

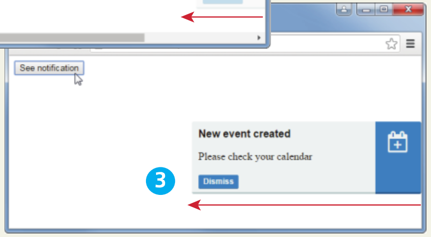
```
.notification {
    ...
    right: -350px;
    top: 100px;
    opacity: 0;
}
```

These are the three properties that will be animated.

These are the **before** values.



2 When button is clicked start the animation



```
$(function() {
    $('#notifyBtn').on("click", function () {

        $('.notification')
           .animate({right:'0px', opacity: "1"},500)
           .animate({top: "0"});

       window.setTimeout(function() {
           dismissNotification();
       }, 4000);
    });

    $('#dismissBtn').on("click", function () {
       dismissNotification();
    });
});
```

3 Over 0.5 sec, first animate these two properties

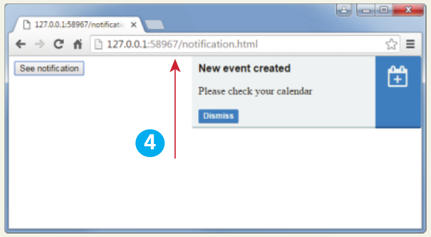
These are the **after** values

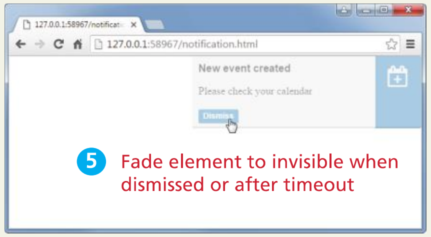
4 and then animate this property

After 4 seconds, call the dismiss notification function

```
function dismissNotification() {
    $('.notification').fadeOut(500);
    $('#notifyBtn').fadeOut(500);
}
```

5

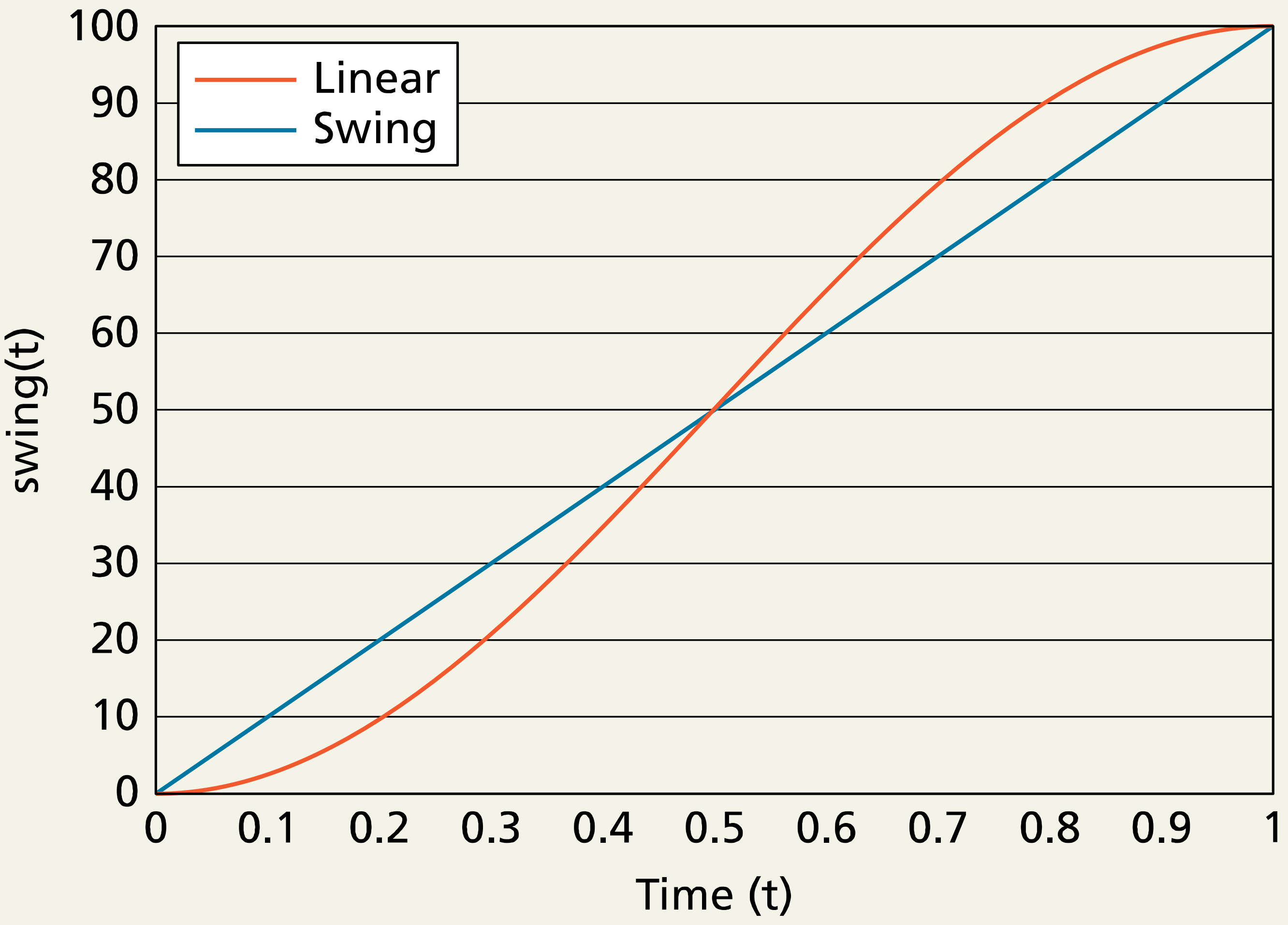




5 Fade element to invisible when dismissed or after timeout

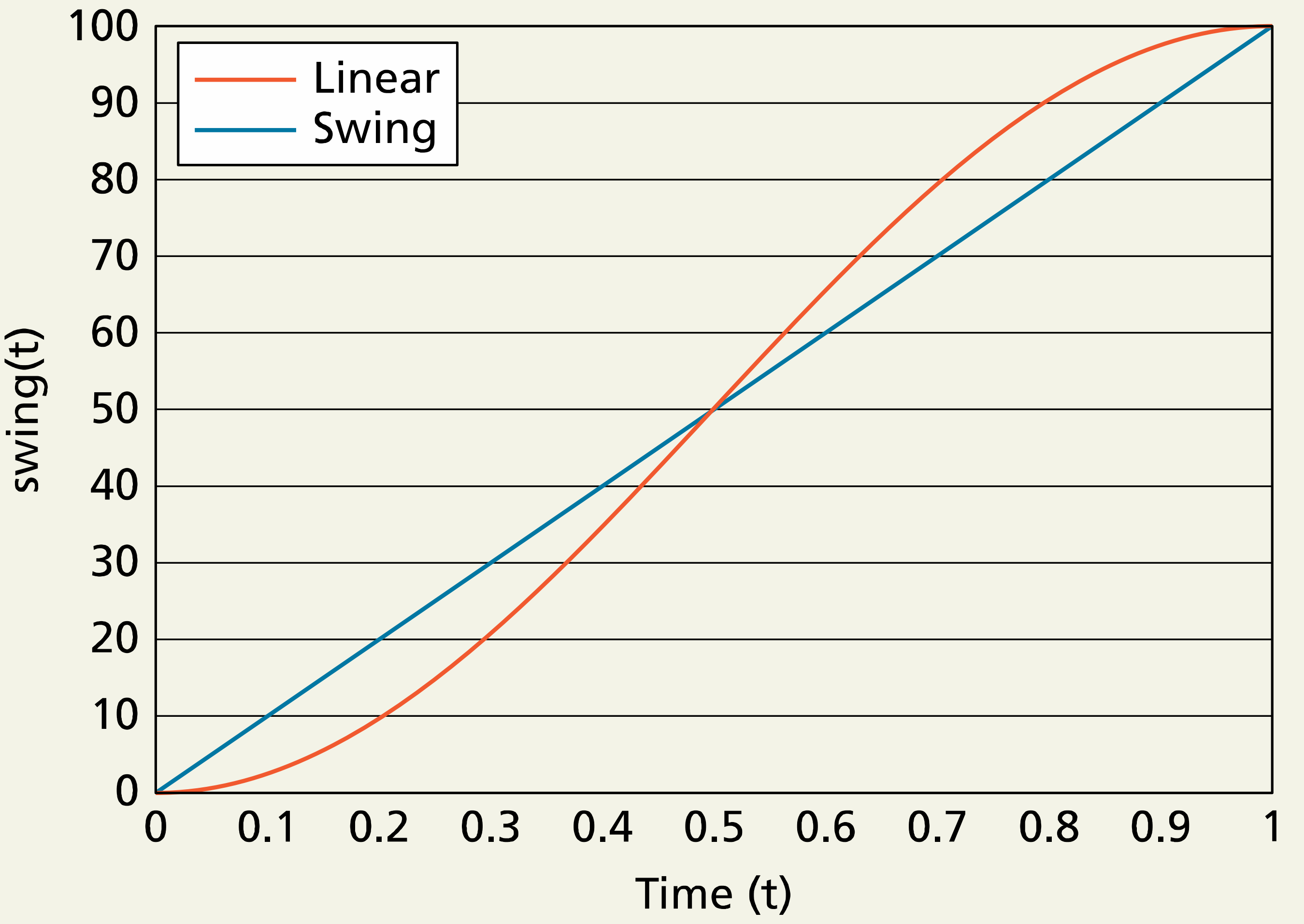
Effects and Animation

Easing Functions



Effects and Animation

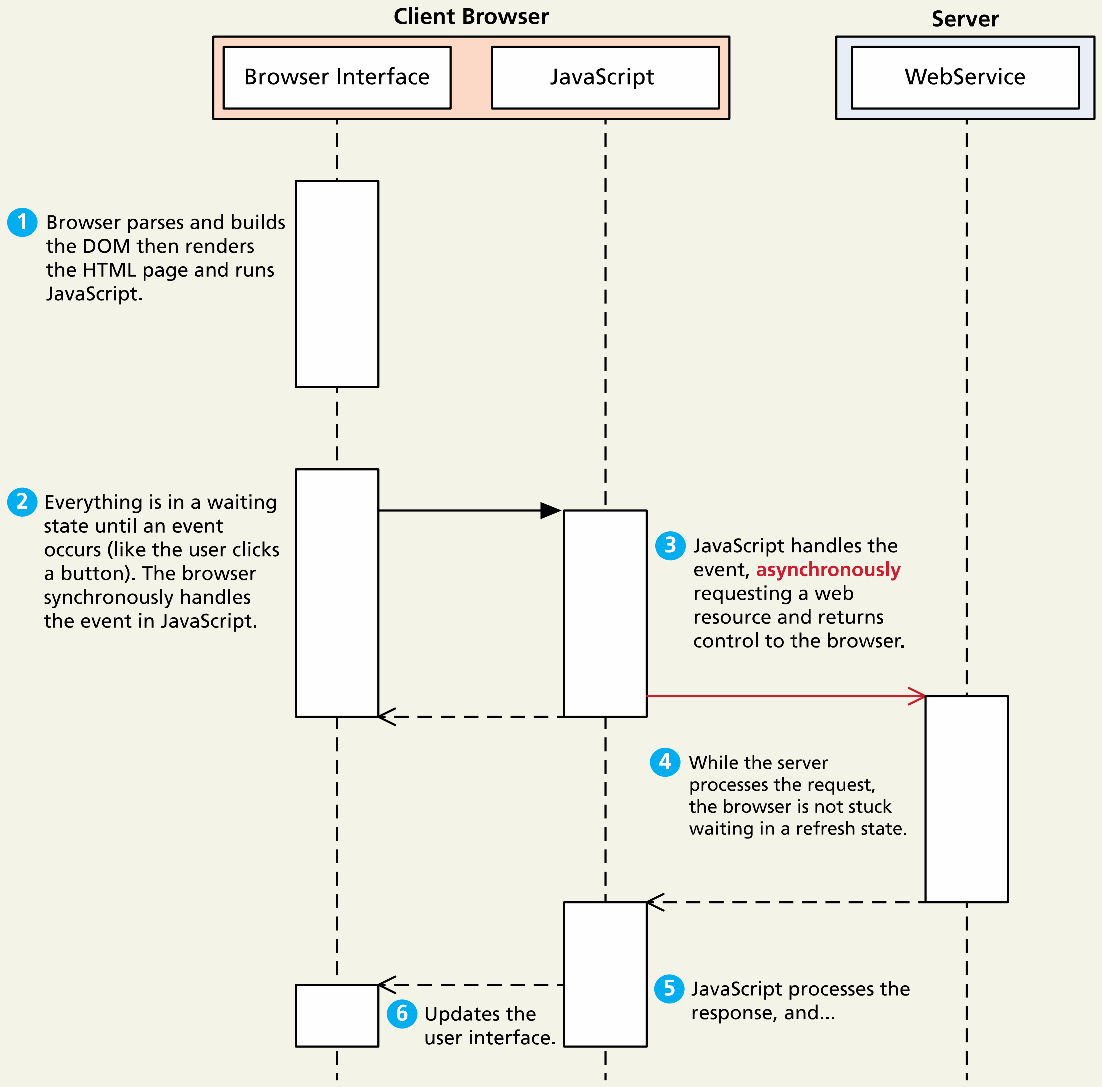
Easing Functions



Chapter 10

AJAX

Asynchronous JavaScript with XML (AJAX)



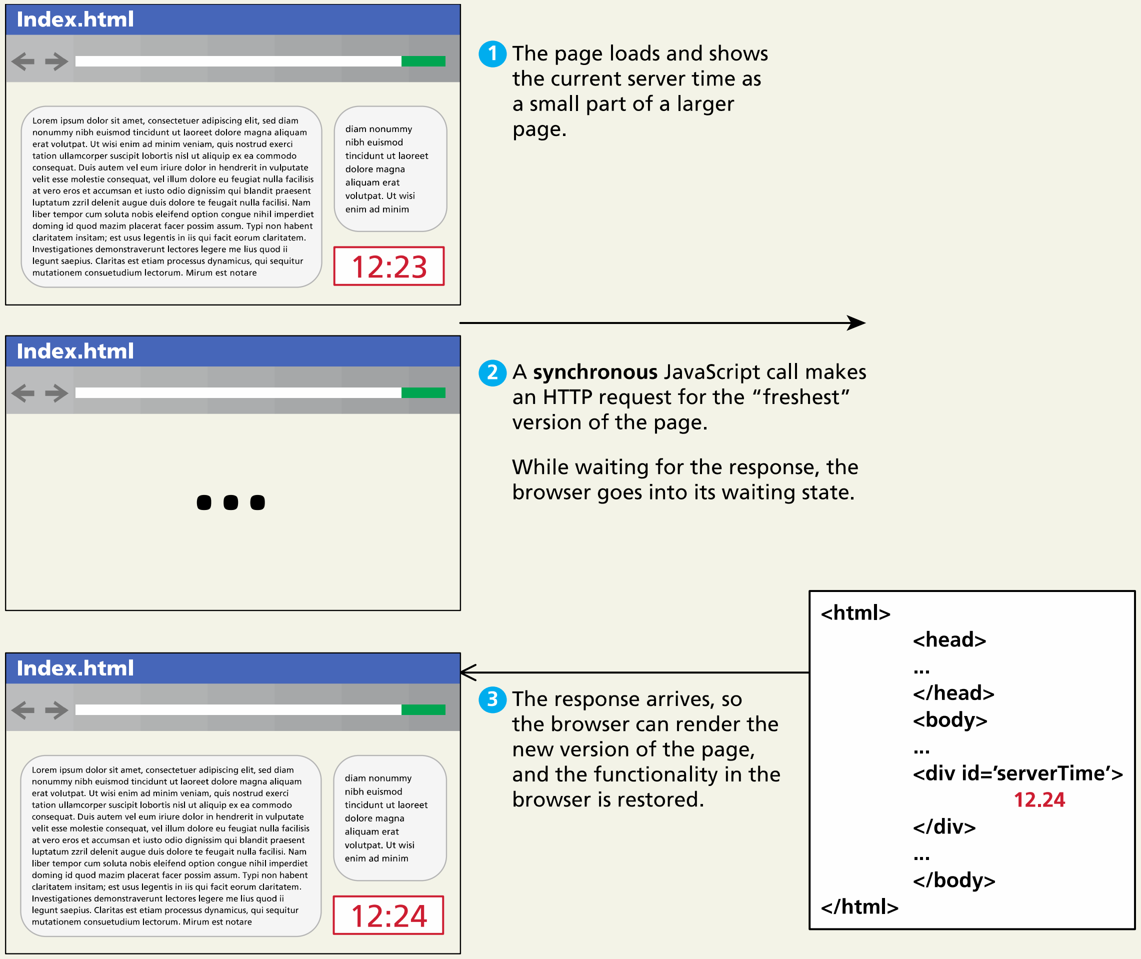
Client Browser
Server
Browser Interface
JavaScript
WebService
1 Browser parses and builds the DOM then renders the HTML page and runs JavaScript.
2 Everything is in a waiting state until an event occurs (like the user clicks a button). The browser synchronously handles the event in JavaScript.
3 JavaScript handles the event, asynchronously requesting a web resource and returns control to the browser.
4 While the server processes the request, the browser is not stuck waiting in a refresh state.
5 JavaScript processes the response, and...
6 Updates the user interface.

AJAX

Making Asynchronous Requests

AJAX

Synchronous example



1 The page loads and shows the current server time as a small part of a larger page.

2 A **synchronous** JavaScript call makes an HTTP request for the "freshest" version of the page.

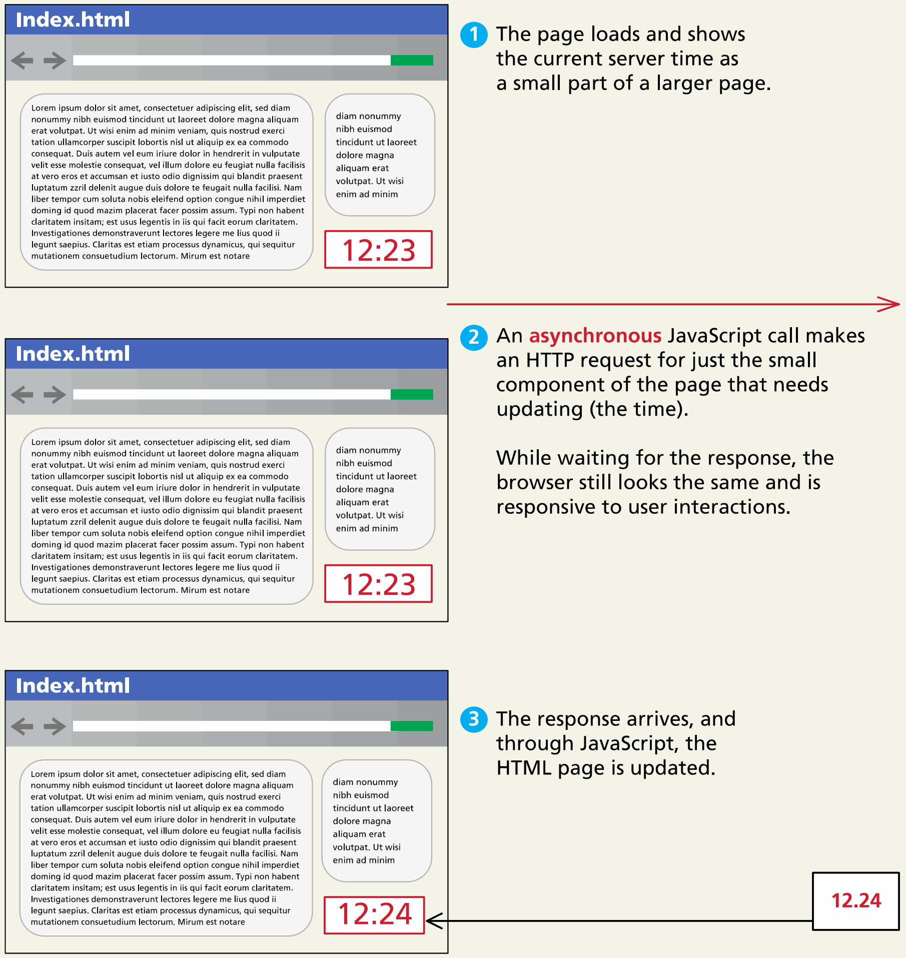
While waiting for the response, the browser goes into its waiting state.

3 The response arrives, so the browser can render the new version of the page, and the functionality in the browser is restored.

```
<html>
        <head>
        …
        </head>
        <body>
        …
        <div id='serverTime'>
                12.24
        </div>
        …
        </body>
</html>
```

AJAX

Asynchronous example – what's changed?



1 The page loads and shows the current server time as a small part of a larger page.

2 An **asynchronous** JavaScript call makes an HTTP request for just the small component of the page that needs updating (the time).

While waiting for the response, the browser still looks the same and is responsive to user interactions.

3 The response arrives, and through JavaScript, the HTML page is updated.

AJAX

Making Asynchronous requests – load()

Easy shortcut functions like **load()**

```
$("#timeDiv").load("currentTime.php");
```

- Asynchronously calls currentTime.php and puts the returned content into the selected div with id timeDiv

AJAX

Making Asynchronous requests - GET

Easy shortcut functions **.get()**

```
$.get("serviceTravelCountries.php?name=Italy");
```

Note that the $ symbol is followed by a dot.

AJAX

Making Asynchronous requests – GET formal

jQuery.get (url [, data] [, success([data, textStatus, jqXHR])] [, dataType])

- **url** is a string that holds the location to send the request.
- **data** is an optional parameter that is a query string or a JavaScript object literal.
- **success(data,textStatus,jqXHR)** is an optional callback function
    - **data** holding the body of the response as a string.
    - **textStatus** holding the status of the request (i.e., “success”).
    - **jqXHR** holding a jqXHR object
- **dataType** is an optional parameter to hold the type of data expected

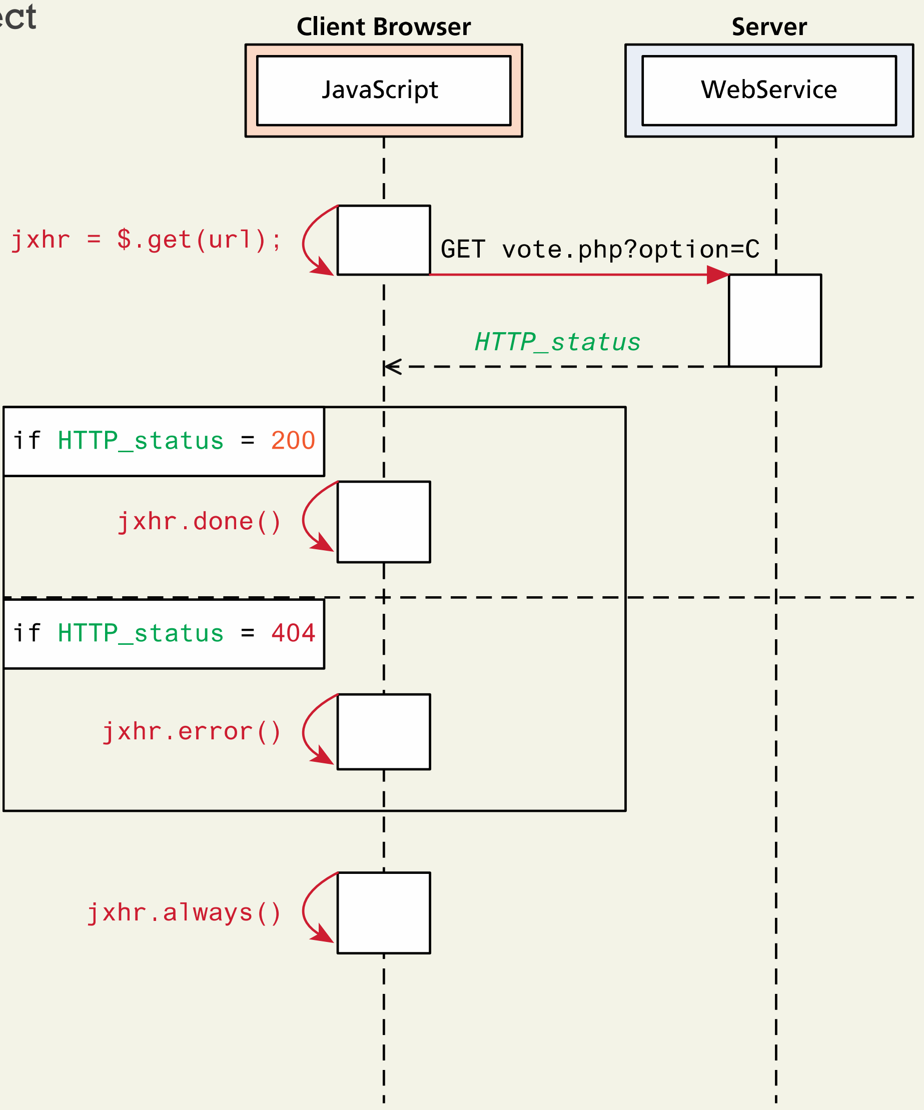
AJAX

The jqXHR Object

- $.get() requests made by jQuery return **a jqXHR object**
- jqXHR objects implement the methods
    - done(),
    - fail(), and
    - always(),

AJAX

The jqXHR Object



AJAX

The POST Method

jQuery handles POST almost as easily as GET, with the need for an added field to hold our data.

```
$.get("serviceTravelCities.php", param)
```

to

```
$.post("serviceTravelCities.php", param)
```

AJAX

The POST Method – form serialization

The **serialize()** method can be called on a DOM form element to encode it into a query string

```
var postData = $("#someForm").serialize();

$.post("formHandler.php", postData);
```

AJAX

Complete Control over AJAX

Over 30 fields to customize control. Here we add headers

```
$.ajax({ url: "vote.php",
         data: $("#voteForm").serialize(),
         async: true,
         type: post,
         headers: {"User-Agent" : "Homebrew Vote Engine",
                   "Referer": "http://funwebdev.com"
         }
});
```

AJAX

Cross-Origin Resource Sharing

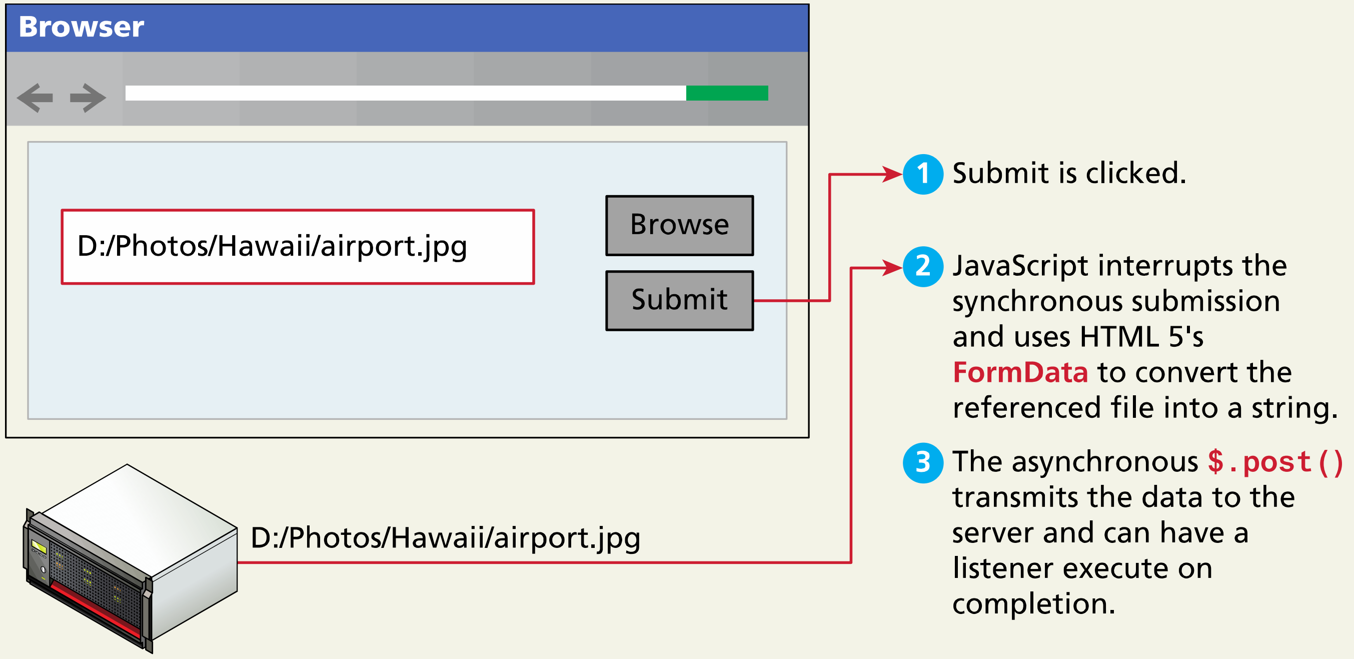
cross-origin resource sharing described in greater detail in Chapter 18.

- a way by which some malicious software can gain access to the content of other web pages you are surfing despite the scripts being hosted on another domain
- sharing content legitimately between two domains becomes harder
- *images.funwebdev.com* and *www.funwebdev.com* are considered different origins
- Access-Control-Allow-Origin header

Chapter 10

Asynchronous File Transmission

The FormData Interface



1 Submit is clicked.

2 JavaScript interrupts the synchronous submission and uses HTML 5's **FormData** to convert the referenced file into a string.

3 The asynchronous `$.post()` transmits the data to the server and can have a listener execute on completion.

Asynchronous File Transmission

The FormData Interface

```
function uploadFile () {
        // get the file as a string
        var formData = new FormData($("#fileUpload")[0]);
        var xhr = new XMLHttpRequest();
        xhr.addEventListener("load", transferComplete, false);
        xhr.addEventListener("error", transferFailed, false);
        xhr.addEventListener("abort", transferCanceled, false);
        xhr.open('POST', 'upload.php', true);
        xhr.send(formData); // actually send the form data
        function transferComplete(evt) { // stylized upload complete
                $("#progress").css("width","100%");
                $("#progress").html("100%");
        }
        function transferFailed(evt) {
                alert("An error occurred while transferring the file.");
        }
        function transferCanceled(evt) {
                alert("The transfer has been canceled by the user.");
        }
}
```

Asynchronous File Transmission

Appending Files to a POST

```
var allFiles = $(":file")[0].files;

for (var i=0;i<allFiles.length;i++) {

        formData.append('images[]', allFiles[i]);

}
```

Chapter 10

Summary

Key Terms

Animation

Asynchronous JavaScript with XML (AJAX)

content delivery network (CDN)

content filters

cross-origin resource sharing (CORS)

easing function

filters

framework

FormData

graceful degredation

jQuery

jqXHR

library

progressive enhancement

Summary

Questions?