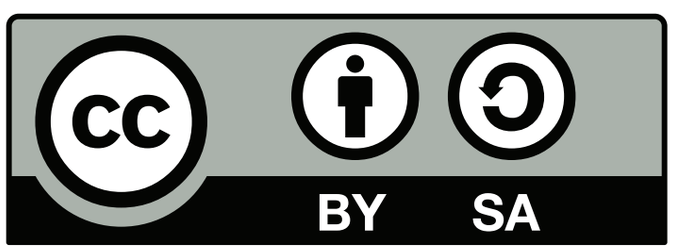
# Tecnologia e Applicazioni Internet 2008/9

Lezione 8 - Ajax

Matteo Vaccari
http://matteo.vaccari.name/
matteo.vaccari@uninsubria.it



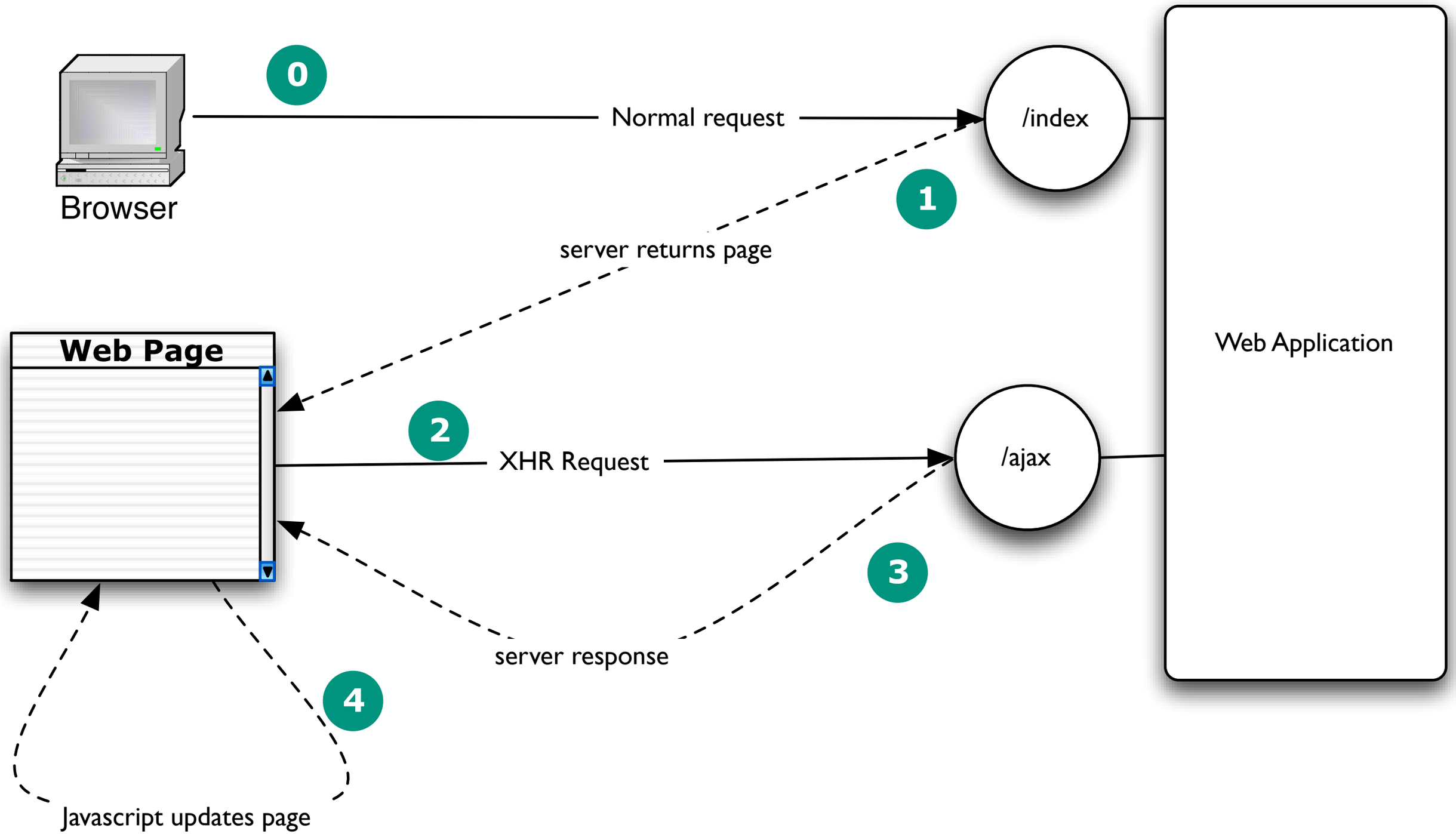
# Un avvertimento

Tempo necessario a sviluppare una feature con Ajax

—

10 * tempo necessario per la stessa feature *senza* Ajax

# XMLHttpRequest (XHR)


Browser
0
Normal request
/index
1
server returns page
Web Page
2
XHR Request
/ajax
3
server response
4
Javascript updates page
Web Application

# Ajax

- Load a page
- Make an XHR request
- When response arrives, update page
- Handle errors
- Handle timeouts

# Ottenere l'oggetto *XMLHttpRequest*

```
// non-Microsoft browser, IE >= 7
var request = new XMLHttpRequest();

// IE < 7
var request = new ActiveXObject("Msxml2.XMLHTTP");
var request = new ActiveXObject("Microsoft.XMLHTTP");


// prova in diverse maniere fino a che una funziona
var request = Try.these(
    function() { return new XMLHttpRequest(); },
    function() { return new ActiveXObject("Msxml2.XMLHTTP"); },
    function() { return new ActiveXObject("Microsoft.XMLHTTP"); }
);
```

# Chiamare il server

```
// settare metodo e url
request.open("GET", url);

// altri parametri opzionali
request.setRequestHeader("User-Agent", "XMLHttpRequest");
request.setRequestHeader("Accept-Language", "en");

// callback eseguita quando il server risponde
request.onreadystatechange = function() {...};

// eseguire la chiamata
request.send(null);
```

# Rispondere a una chiamata *asincrona*

```
// readyState meaning
// 0    open() has not been called yet.
// 1    open() has been called, but send() has not been called.
// 2    send() has been called, but the server has not responded yet.
// 3    Data is being received from the server.
// 4    The server's response is complete.

request.onreadystatechange = function() {
    if (request.readyState == 4) {  // If the request is finished
        if (request.status == 200)  // If it was successful
            alert(request.responseText);  // Display the server's response
    }
}
```

# XHR Limitations

- *Same Origin* Policy
- Non più di due chiamate aperte contemporaneamente

# The *Prototype* library

```
// Hide the element
$(element).hide();
```

```
// Add a class to the element
$(element).addClassName("myClass");

// All descendant nodes of the element
// with the id "article"
$('articles').descendants();
```

# Manipolare il DOM

```
<p id="myid">Ciao</p>

// rimpiazza il contenuto dell'elemento
$("myid").update("Hi!");

// rimpiazza tutto l'elemento e sostituiscilo
$("myid").replace("<h3>Hello!</h3>");

// elimina l'elemento
$("myid").remove();
```

# Event Handling

```
// execute when DOM is loaded
document.observe("dom:loaded", function() {
  // initially hide all these elements
  $$('div.tabcontent').invoke('hide');
});

// add an event handler that executes when
// the onclick event triggers on the element
$(element).observe('click', function(event){
  alert(Event.element(event).innerHTML);
 });

// *replace* the event handler function (deprecated)
$(element).onclick = myFunc;
```

# Esecuzione periodica

```
// esegui myFunc ogni 3 secondi
new PeriodicalExecuter(myfunc, 3);
```

```
function myfunc(periodicalUpdater) {
  if (!confirm('Want me to annoy you again later?'))
    periodicalUpdater.stop(); // stop the updater
}
```

# Ajax with prototype

```
new Ajax.Request('/some_url',
  {
    method:'get',
    onSuccess: function(transport){
      var response = transport.responseText || "no response text";
      alert("Success! \n\n" + response);
    },
    onFailure: function(){ alert('Something went wrong...') }
});
```

# Sending parameters

```
new Ajax.Request('/some_url', {
  method: 'get',
  parameters: {company: 'example', limit: 12}
  });

new Ajax.Request('/some_url', {
  parameters: $('id_of_form_element').serialize(true)
  });
```

# Updating a page element

```
<h2>Our fantastic products</h2>
<div id="products">(fetching product list ...)</div>
```

```
new Ajax.Updater('products', '/some_url', { method: 'get' });

new Ajax.Updater('products', '/some_url', {
  method: 'get',
  insertion: Insertion.Top   // don't replace; insert on top
  });
```

# Periodical update

```
new Ajax.PeriodicalUpdater('products', '/some_url',
  {
    method: 'get',
    insertion: Insertion.Top,
    frequency: 3,    // every 3 seconds
    decay: 2,        // factor by which the frequency
                     // is multiplied when response
                     // does not change
});
```

# How to unit test Ajax?

- Unit test the server-side calls
- Unit test the JavaScript calls

# Mocking Ajax

```
// mocking Ajax
this.mockAjax({
  text: 'some text you like',                  // '' by default
  xml: your_xml_object,                        // null by default
  status: 200,                                 // 200 by default
  headers: {'Content-type': 'text/xml'} // {} by default
});

...

// at the end of the test
this.undoAjaxMock();
```

# Mocking Ajax

```
<a href="#" onclick="ajaxed_update(); return false;">click me</a>

var ajaxed_update = function() {
  new Ajax.Updater('some-block', '/some/url');
};

test_ajax_load: function() {
  this.mockAjax({text: '<p>some text</p>'});

  ajaxed_update();

  // the block will be updated right now
  this.assertEqual(
    '<p>some text</p>', $('some-block').innerHTML
  );

  this.undoAjaxMock();
}
```