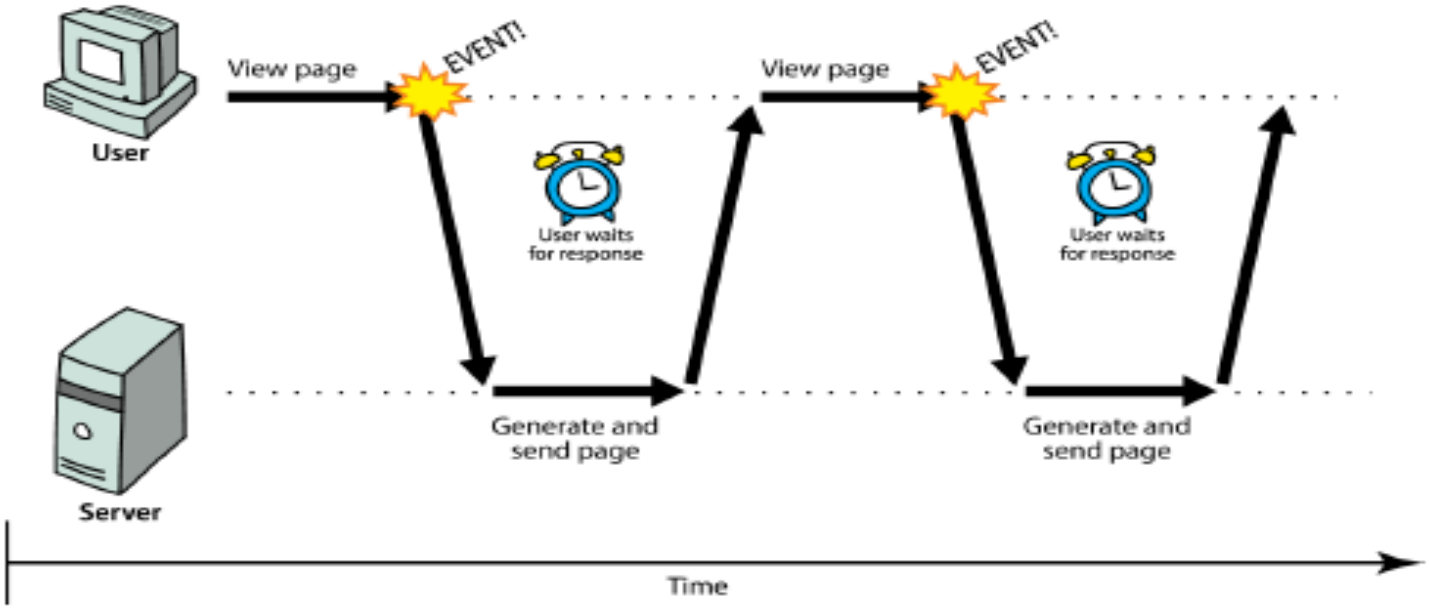
Ajax

Synchronous web communication



- synchronous: user must wait while new pages load
  - the typical communication pattern used in web pages (click, wait, refresh)

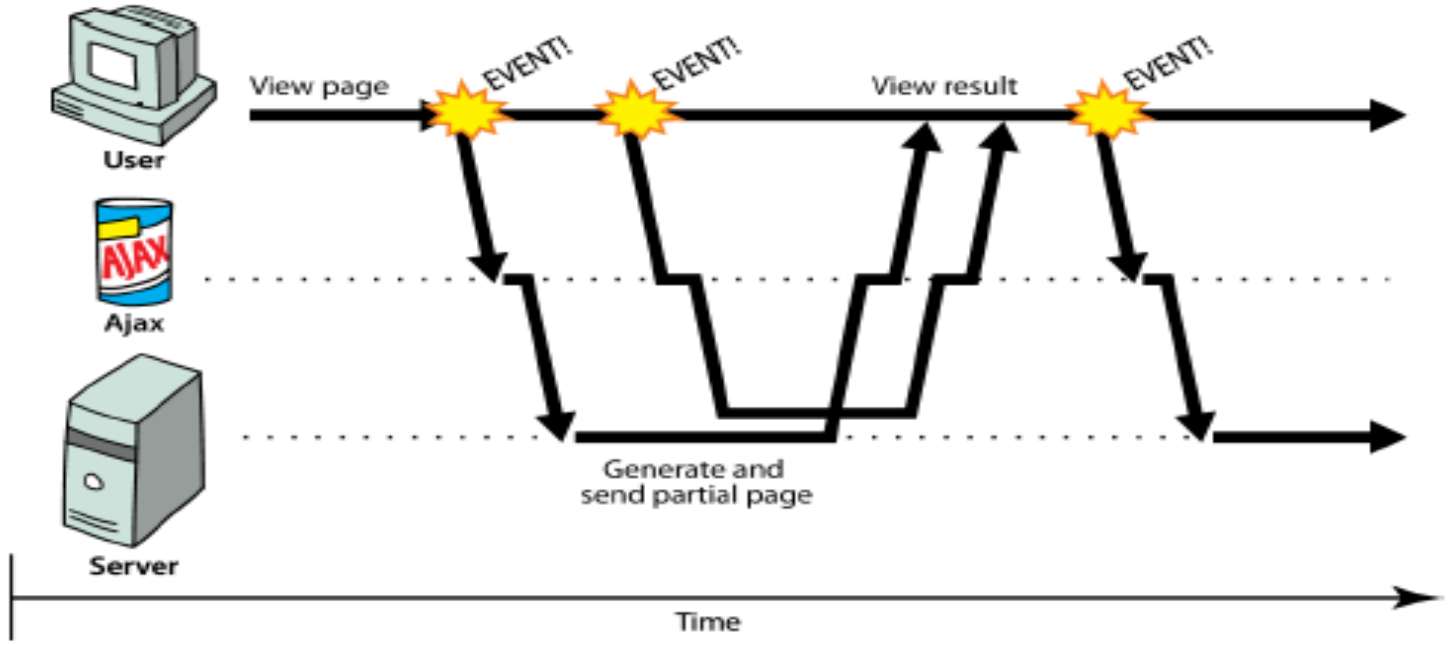
Web applications and Ajax

- **web application:** a dynamic web site that mimics the feel of a desktop app
  - presents a continuous user experience rather than disjoint pages
  - examples: Gmail, Google Maps, Google Docs and Spreadsheets, Flickr, A9

Web applications and Ajax

- **Ajax**: Asynchronous JavaScript and XML
  - not a programming language; a particular way of using JavaScript
  - downloads data from a server in the background
  - allows dynamically updating a page without making the user wait
  - avoids the "click-wait-refresh" pattern
  - Example: Google Suggest

Asynchronous web communication



- **asynchronous:** user can keep interacting with page while data loads
  - communication pattern made possible by Ajax

`XMLHttpRequest` (and why we won't use it)

- JavaScript includes an XMLHttpRequest object that can fetch files from a web server
  - supported in IE5+, Safari, Firefox, Opera, Chrome, etc. (with minor compatibilities)
- it can do this asynchronously (in the background, transparent to user)
- the contents of the fetched file can be put into current web page using the DOM

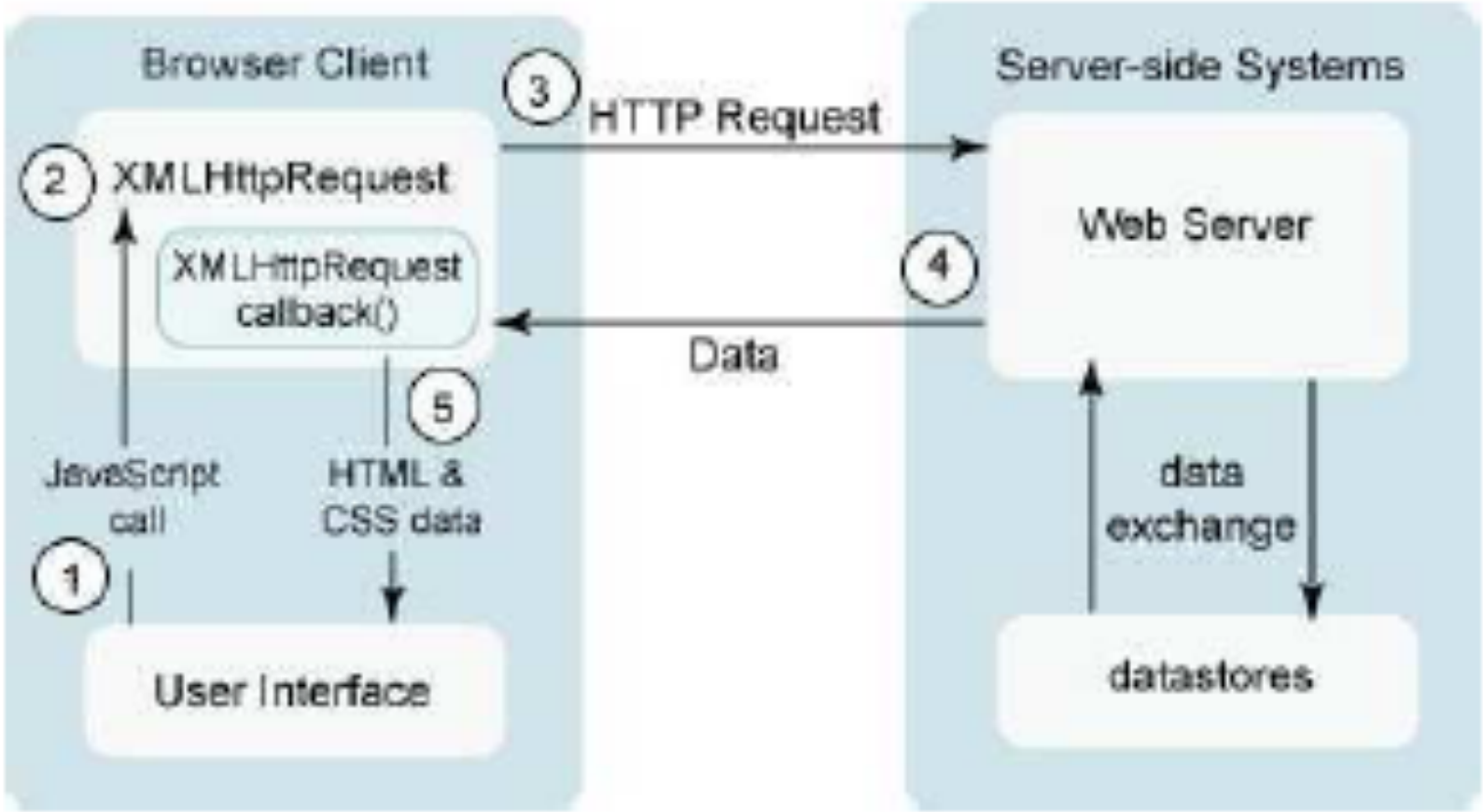
`XMLHttpRequest` (and why we won't use it)

- sounds great!...
- ... but it is clunky to use, and has various browser incompatibilities
- Prototype/JQuery provides a better wrapper for Ajax, so we will use that instead

A typical Ajax request

1. user clicks, invoking an event handler
2. handler's code creates an `XMLHttpRequest` object
3. `XMLHttpRequest` object requests page from server
4. server retrieves appropriate data, sends it back
5. `XMLHttpRequest` fires an event when data arrives
   - this is often called a callback
   - you can attach a handler function to this event
6. your callback event handler processes the data and displays it

A typical Ajax request



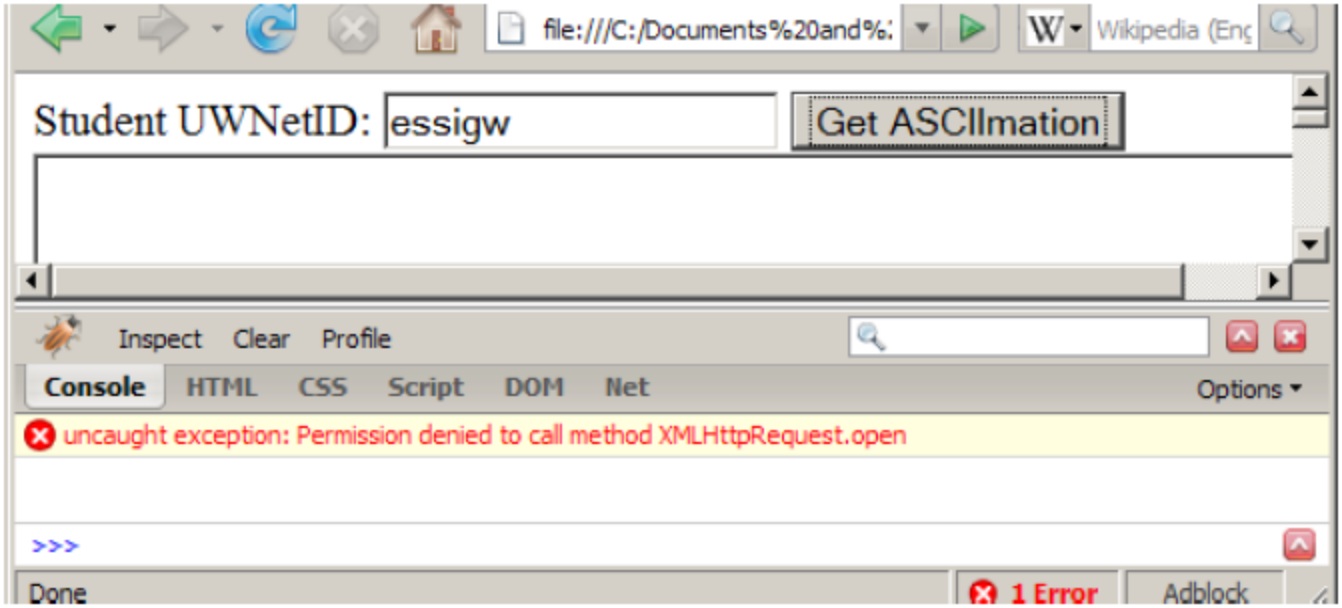
Example


```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        document.getElementById("txtHint").innerHTML = xmlhttp.responseText;
    }
}
xmlhttp.open("GET", "gethint.php?q=" + str, true);
xmlhttp.send();
```

Properties

property	description
status	the request's HTTP error code (200 = OK, etc.)
statusText	HTTP error code text
responseText	the entire text of the fetched page, as a String
responseXML	the entire contents of the fetched page, as an XML DOM tree

XMLHttpRequest security restrictions



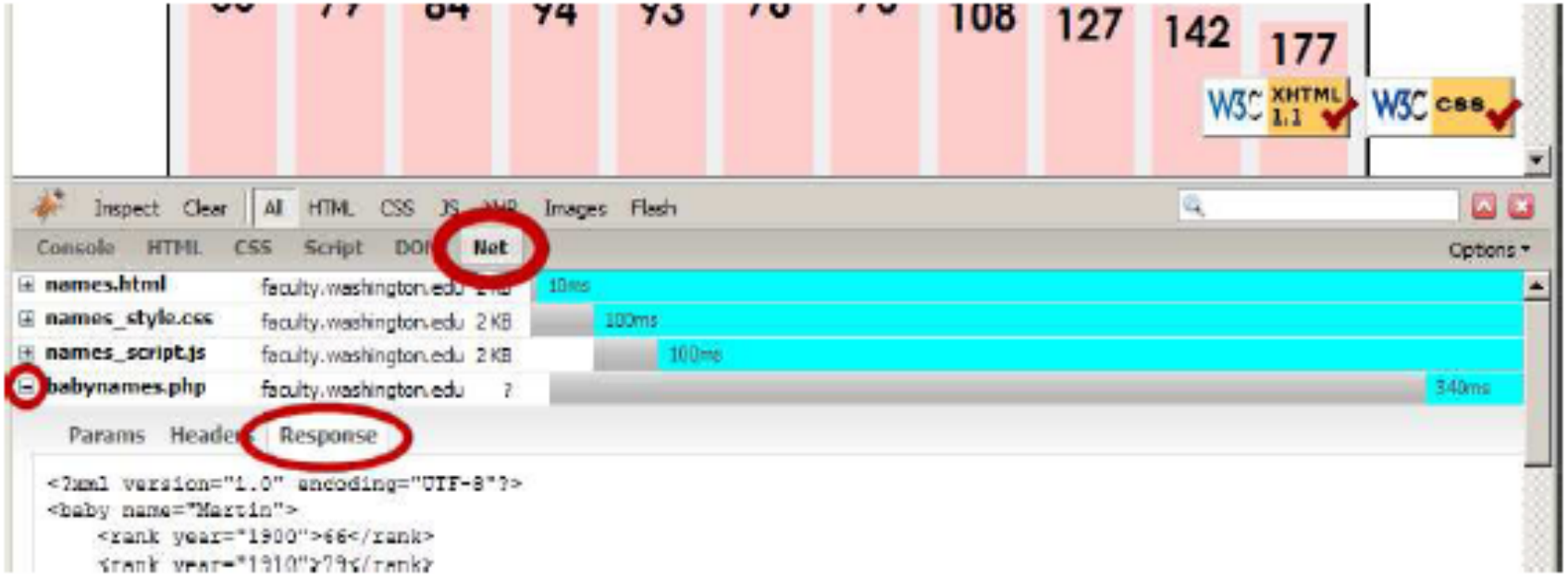
- cannot be run from a web page stored on your hard drive
- can only be run on a web page stored on a web server

Handling Ajax errors with Prototype


```
new Ajax.Request("url",
{
	method: "get",
	onSuccess: functionName,
	onFailure: ajaxFailure,
	onException: ajaxFailure
}
);
...
function ajaxFailure(ajax, exception) {
	alert("Error making Ajax request:" + "\n\nServer
status:\n" + ajax.status + " " + ajax.statusText +
"\n\nServer response text:\n" + ajax.responseText);
	if (exception) {
		throw exception;
	}
}
```

JS

Debugging Ajax code



- Net tab shows each request, its parameters, response, any errors
- expand a request with + and look at Response tab to see Ajax result

Creating a POST request


```
new Ajax.Request("url",
{
        method: "post", // optional
        parameters: { name: value, name: value, ..., name:
value },
        onSuccess: functionName,
        onFailure: functionName,
        onException: functionName
}
);
```

JS

Creating a POST request

- Ajax.Request can also be used to post data to a web server
- method should be changed to "post" (or omitted; post is default)
- any query parameters should be passed as a parameters parameter
  - written between {} braces as a set of name : value pairs (another anonymous object)
  - get request parameters can also be passed this way, if you like

From Javascript Tutorial

```
var s = document.getElementsByClassName('someclass');
var o = s[0].getElementsByTagName('someotherclass');
```

This method finds the element with the given id:

```
document.getElementById("id").onClick = function(e){
document.getElementById("id").innerHTML('<p>Clicked!</p>');
};
```

This method will fire an event if the mouse is move over the element.

JQuery write less, do more

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
    <script>
      $(document).ready(function() {
        $('#mo').click(function() {
          $('#mo').append('<p>Clicked!</p>');
        });
      });
</script>
```

- Cross Browser.
- Lots of helpers and utilities.
- Very active community

http://jquery.com/

UNIVERSITY OF TORONTO

DOM Traversal and Manipulation

Get the <button> element with the class 'continue' and change its HTML to 'Next Step...'

```
$( "button.continue" ).html( "Next Step..." )
```

Event Handling: JQuery write less, do more

Show the #banner-message element that is hidden with display:none in its CSS when any button in #button-container is clicked.

```
var hiddenBox = $( "#banner-message" );

$( "#button-container button" ).on( "click", function( event ) {

  hiddenBox.show();

});
```

AJAX How it works

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("weather-temp").innerHTML="<strong>" +
xmlhttp.responseText + "</strong>";
    }
  }
xmlhttp.open("GET","/api/getWeather",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("zipcode=97201");
```

Ajax: JQuery write less, do more

Call a local script on the server /api/getWeather with the query parameter zipcode=97201 and replace the element #weather-temp's html with the returned text.

```
$.ajax({
 url: "/api/getWeather",
 data: {
   zipcode: 97201
 },
 success: function( data ) {
   $( "#weather-temp" ).html( "<strong>" + data + "</strong>
degrees" );
 }
});
```

UNIVERSITY OF TORONTO