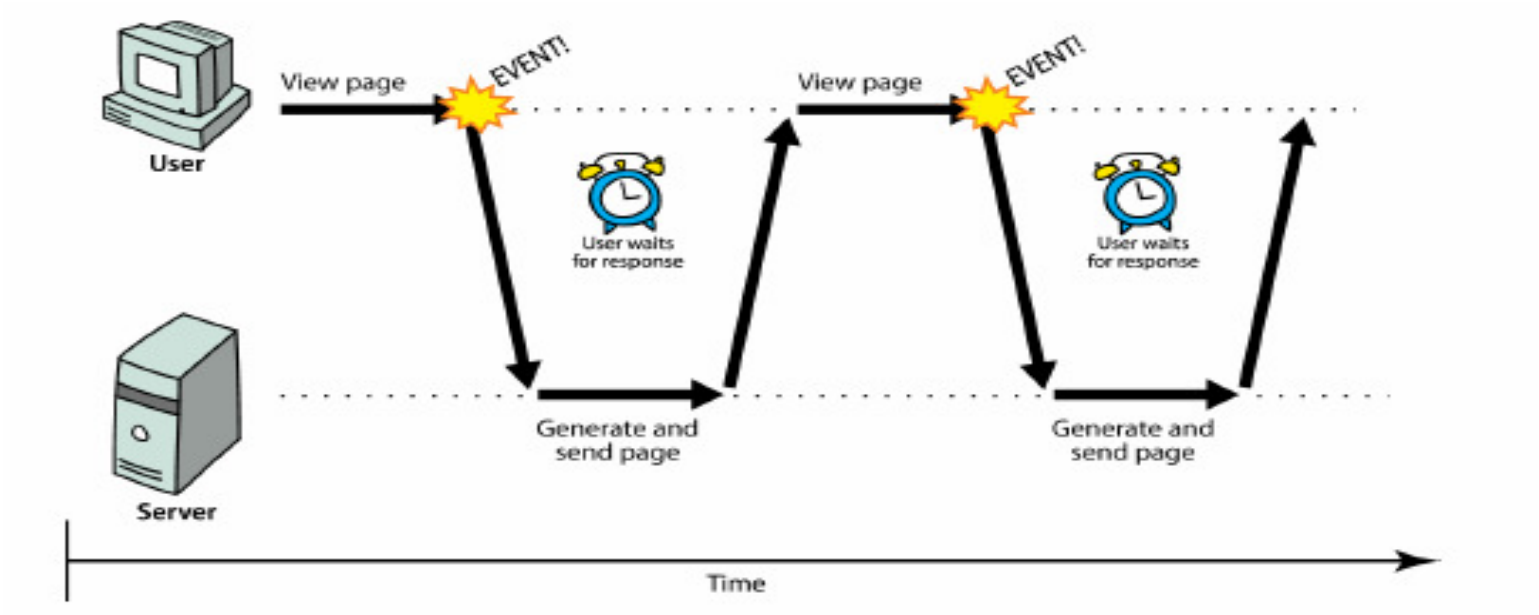
Ajax

Synchronous web communication



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

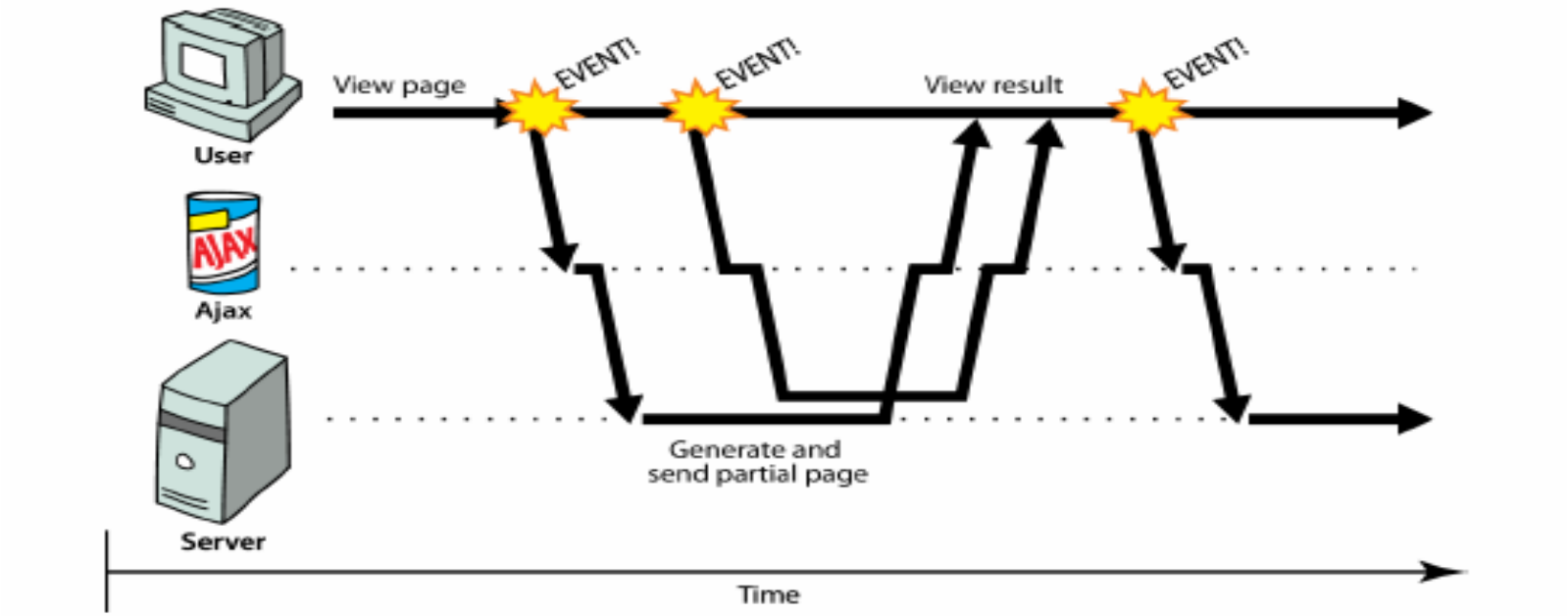
Web applications and Ajax

- **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

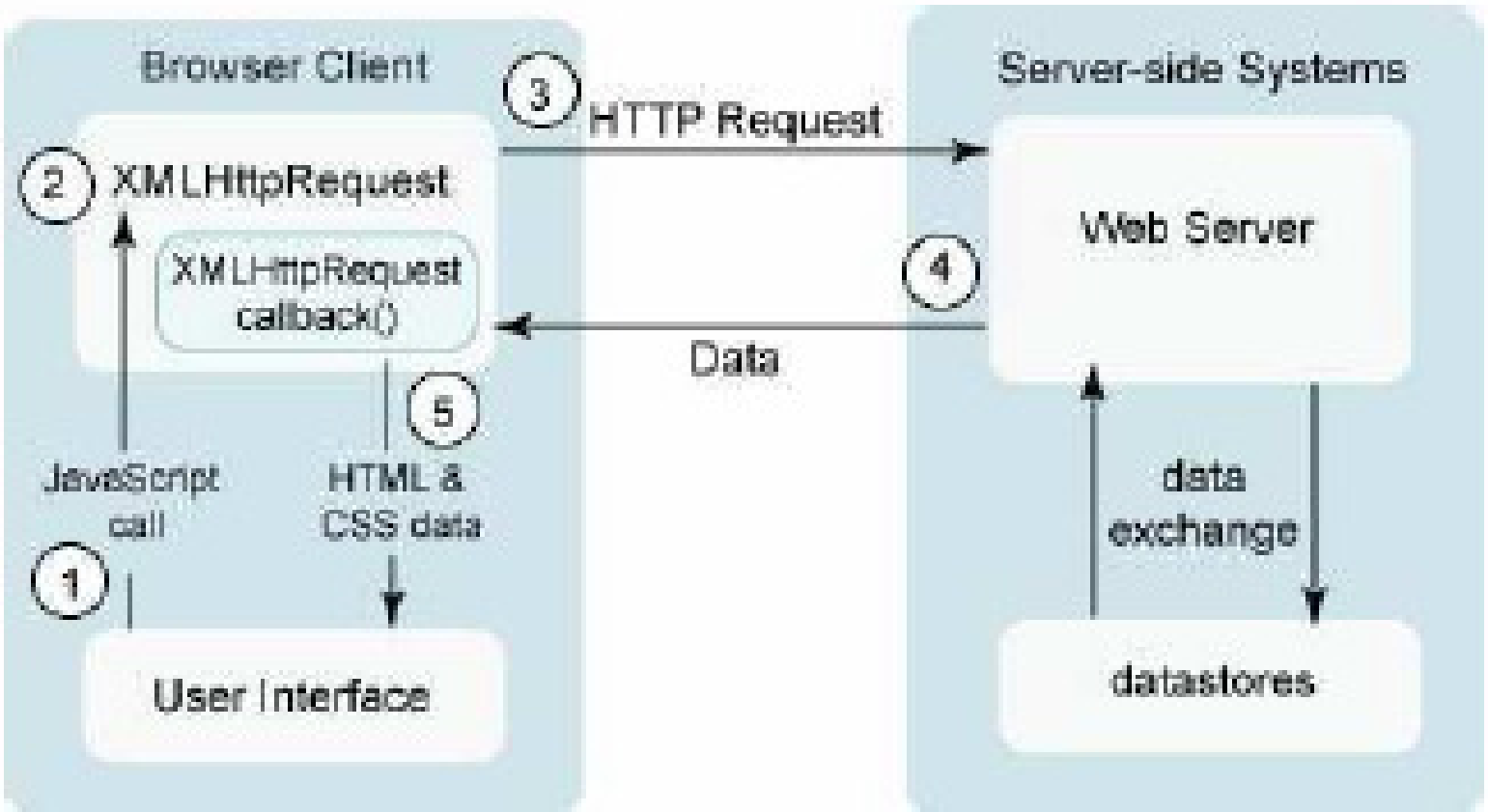
`XMLHttpRequest` (and why we won't use it)

- sounds great!...
- ... but it is clunky to use, and has various browser incompatibilities
- Prototype 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

A typical Ajax request



Prototype's Ajax model


```
new Ajax.Request("url",
{
        option : value,
        option : value,
        ...
        option : value
}
);
```
JS

- construct a Prototype Ajax.Request object to request a page from a server using Ajax
- constructor accepts 2 parameters:
  1. the URL to 1. fetch, as a String,
  2. a set of options, as an array of key : value pairs in {} braces (an anonymous JS object)

Prototype Ajax methods and properties

option	description
method	how to fetch the request from the server (default "post")
parameters	query parameters to pass to the server, if any
asynchronous (default true), contentType, encoding, requestHeaders	

options that can be passed to the Ajax.Request constructor

Prototype Ajax methods and properties

event	description
onSuccess	request completed successfully
onFailure	request was unsuccessful
onException	request has a syntax error, security error, etc.

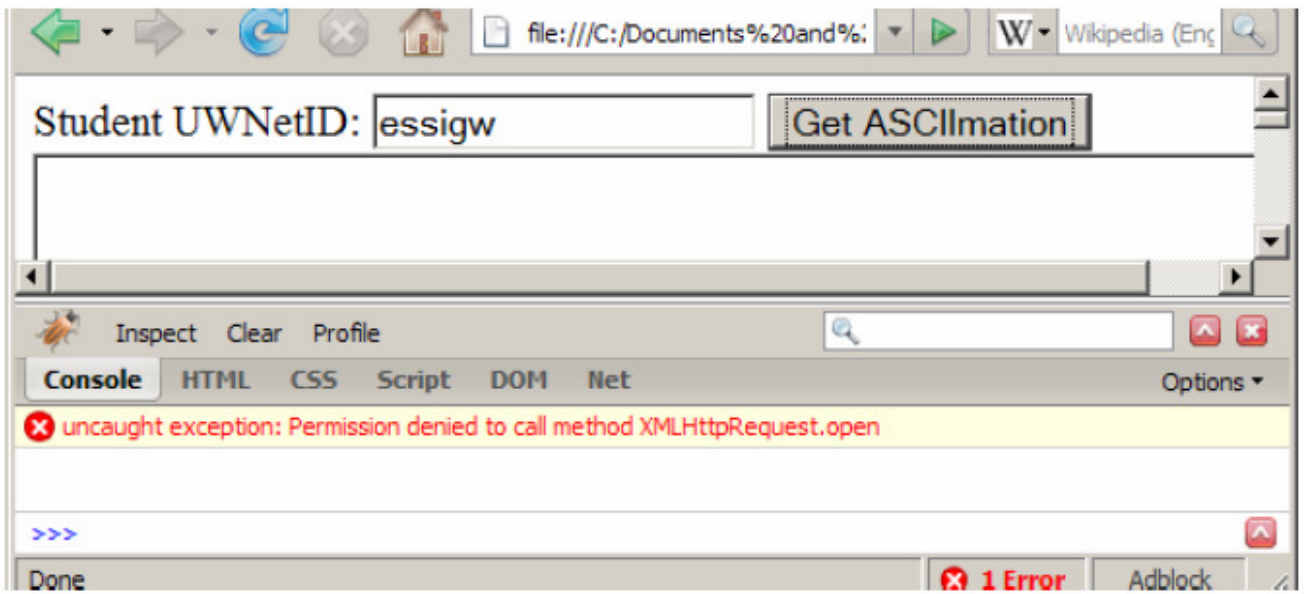
events in the Ajax.Request object that you can handle

Basic Prototype Ajax template

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 (seen later)

```
function handleRequest(ajax) {
        alert(ajax.responseText);
}
```
JS

XMLHttpRequest security restrictions



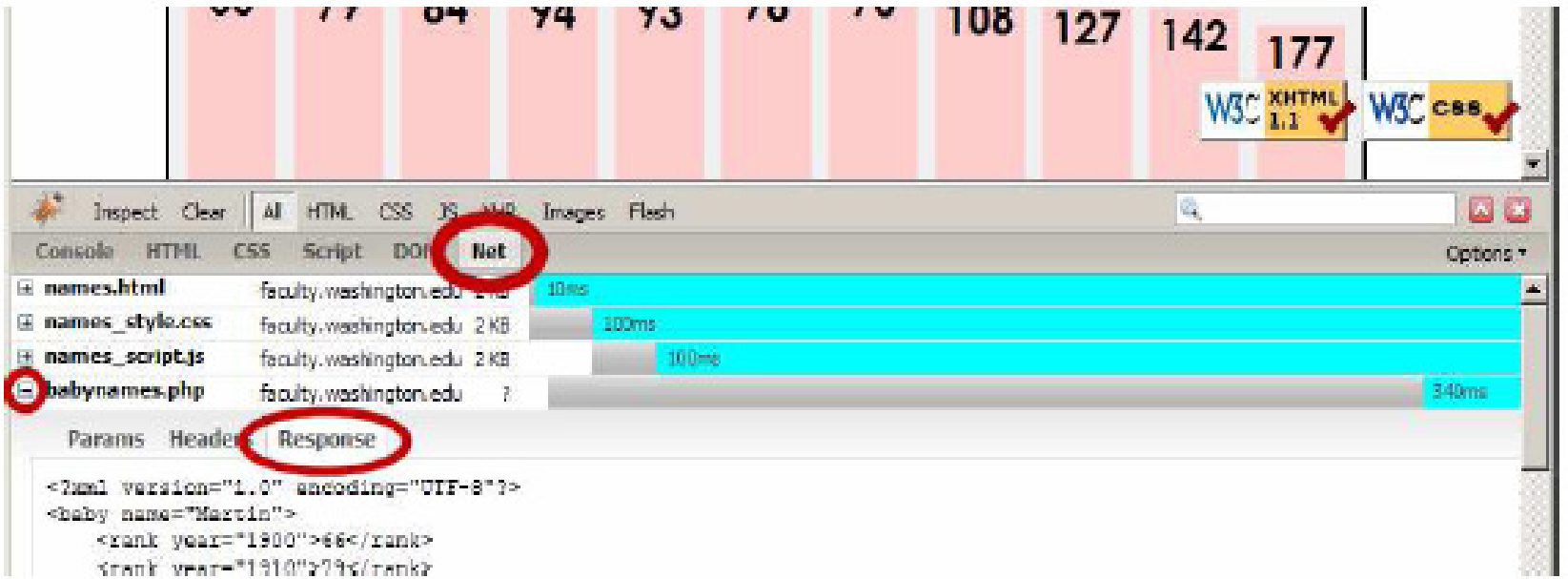
- cannot be run from a web page stored on your hard drive
- can only be run on a web page stored on a web server
- can only fetch files from the same site that the page is on www.foo.com/a/b/c.html can only fetch from www.foo.com

Handling Ajax errors


```
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

Prototype's Ajax Updater


```
new Ajax.Updater(
        "id",
        "url",
        {
                method: "get"
        }
);
```
JS

- Ajax.Updater fetches a file and injects its content into an element as innerHTML
- additional (1st) parameter specifies the id of element to inject into