中山大學
SUN YAT-SEN UNIVERSITY

Lecture 19
Asynchronous JavaScript and XML (AJAX)

SE-805 Web 2.0 Programming (supported by Google)

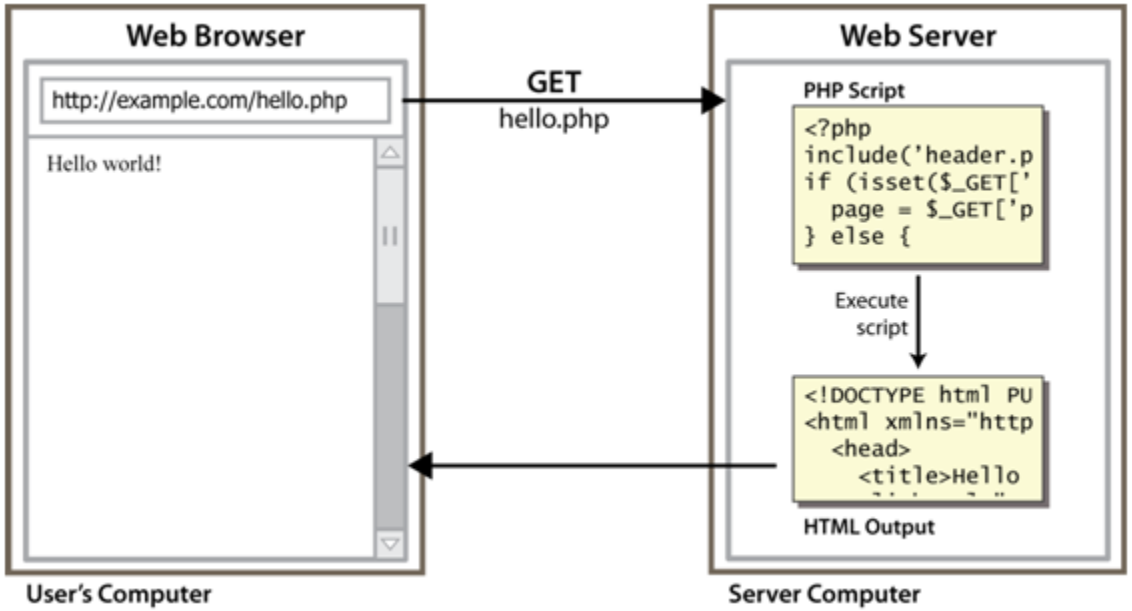
http://my.ss.sysu.edu.cn/courses/web2.0/

School of Software, Sun Yat-sen University

Outline

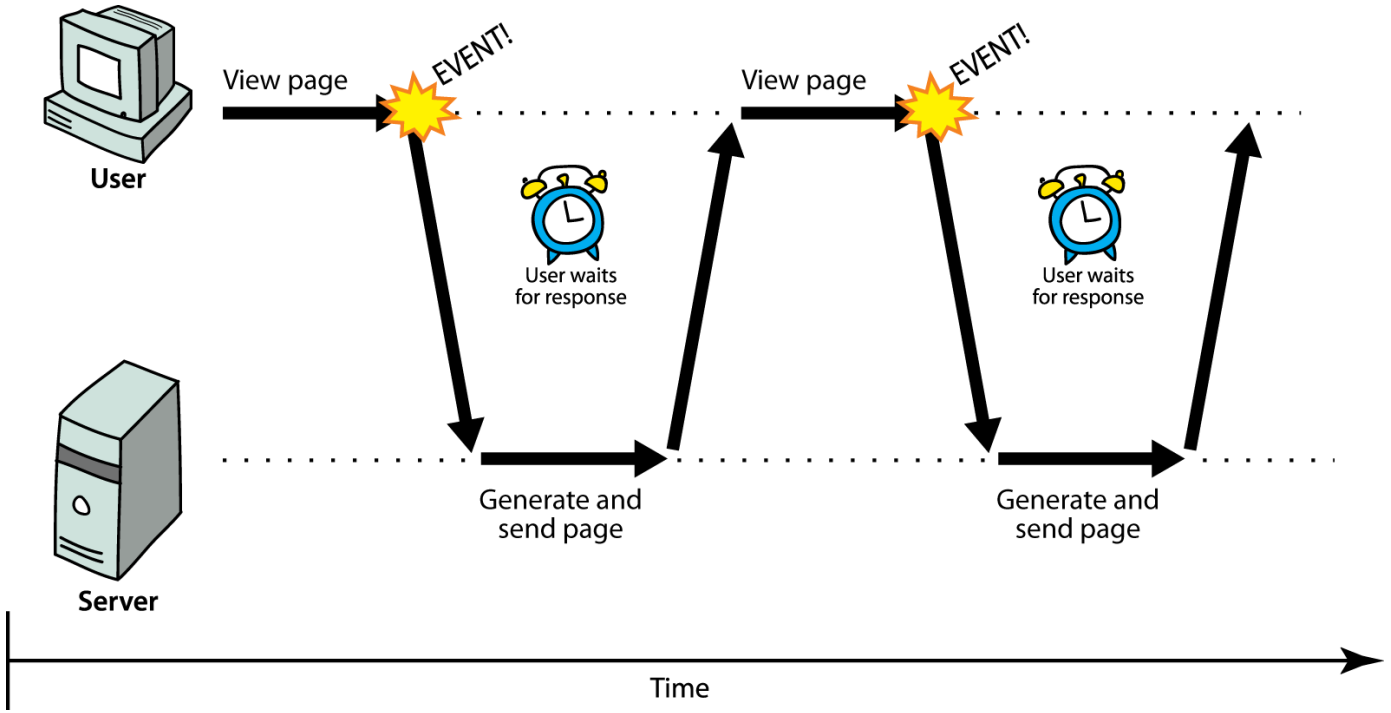
- **Synchronous vs. Asynchronous**
- XMLHttpRequest
- AJAX in Prototype
- Limits of AJAX
- Debugging AJAX

Server Browser Interaction



- How does a browser interact with a user?
- When will it issue a request?

Synchronous Web Communication

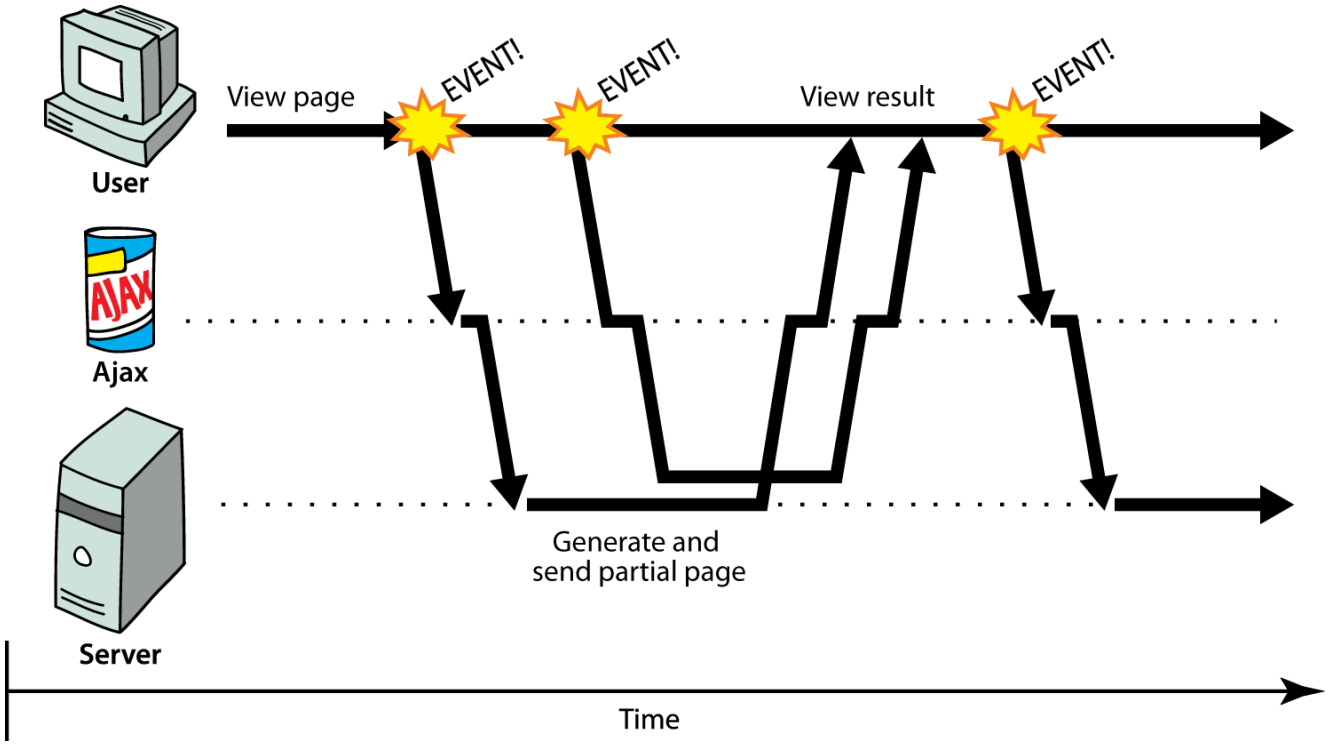


- **Synchronous**: user must wait while new pages load
  - The typical communication pattern used in web pages (click, wait, refresh)
- Almost all changes with new data lead to page refresh

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
- **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
  - avoids the "click-wait-refresh" pattern
  - examples: Google Suggest

Asynchronous Web Communication



- **Asynchronous**: user can keep interacting with page while data loads
  - Communication pattern made possible by AJAX
- Changing with new data but without page refresh

Outline

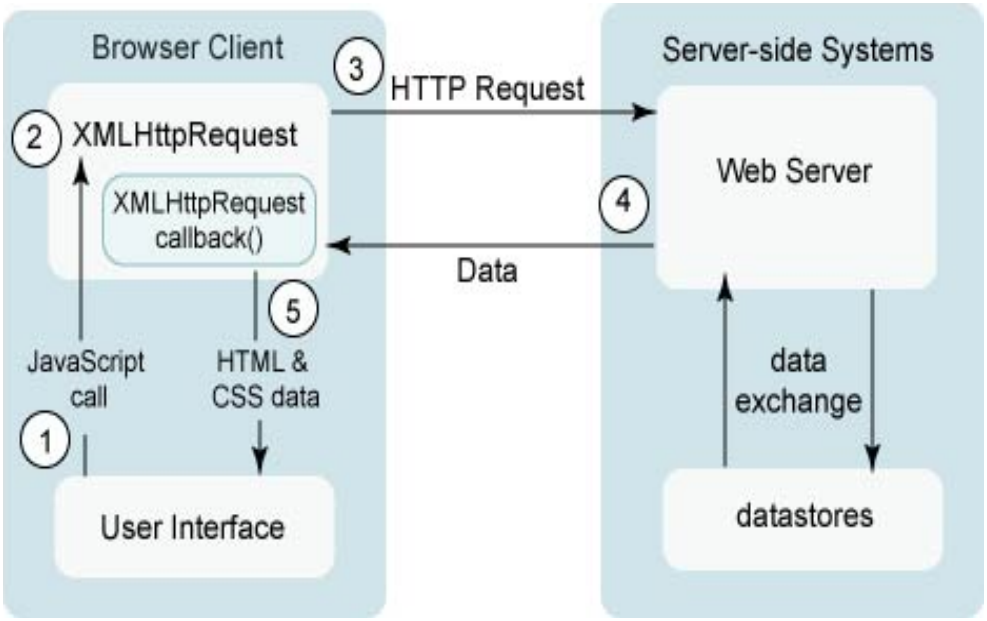
- Synchronous vs. Asynchronous
- **XMLHttpRequest**
- AJAX in Prototype
- Limits of AJAX
- Debugging AJAX

XMLHttpRequest

- **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**
- 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

- A user clicks, invoking an event handler
- The handler creates an **`XMLHttpRequest`** object
- The **`XMLHttpRequest`** object requests page from the server



- The server retrieves appropriate data, sends it back
- The **`XMLHttpRequest`** fires an **`event`** when data arrives
  - You can attach a **handler** function to this event, which is often called a **callback**
- Your callback event handler processes the data and displays it

Outline

- Synchronous vs. Asynchronous
- XMLHttpRequest
- **AJAX in Prototype**
- Limits of AJAX
- Debugging AJAX

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:
  - The **URL** to fetch, as a **String**,
  - A set of **options**, as an array of *key* : *value* pairs in **{}** braces (an anonymous JS object)
- Hides icky details from the raw `XMLHttpRequest`; works well in all browsers

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

Event	Description
`onSuccess`	request completed successfully
`onFailure`	request was unsuccessful
`onException`	request has a syntax error, security error, etc.
`onCreate`, **`onComplete`**, **`on###`** (for HTTP error code ###)	

- Events in the `Ajax.Request` object that you can handle

Basic Prototype AJAX Template

```
  new Ajax.Request("url",
     {
       method: "get",
       onSuccess: functionName
     }
  );
  ...

function functionName(ajax) {
  do something with ajax.responseText;
}
```
JS

- Most AJAX requests we'll do in this course are **GET** requests
- Attach a handler to the request's **onSuccess** event
- the handler takes an AJAX response object, which we'll name **ajax**, as a parameter

The AJAX response Object

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

- Most commonly property is **responseText**, to access the fetched page

Protype's AJAX Updater

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

- `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
- `onSuccess` handler not needed (but `onFailure`, `onException` handlers may still be useful)

Creating a POST Request

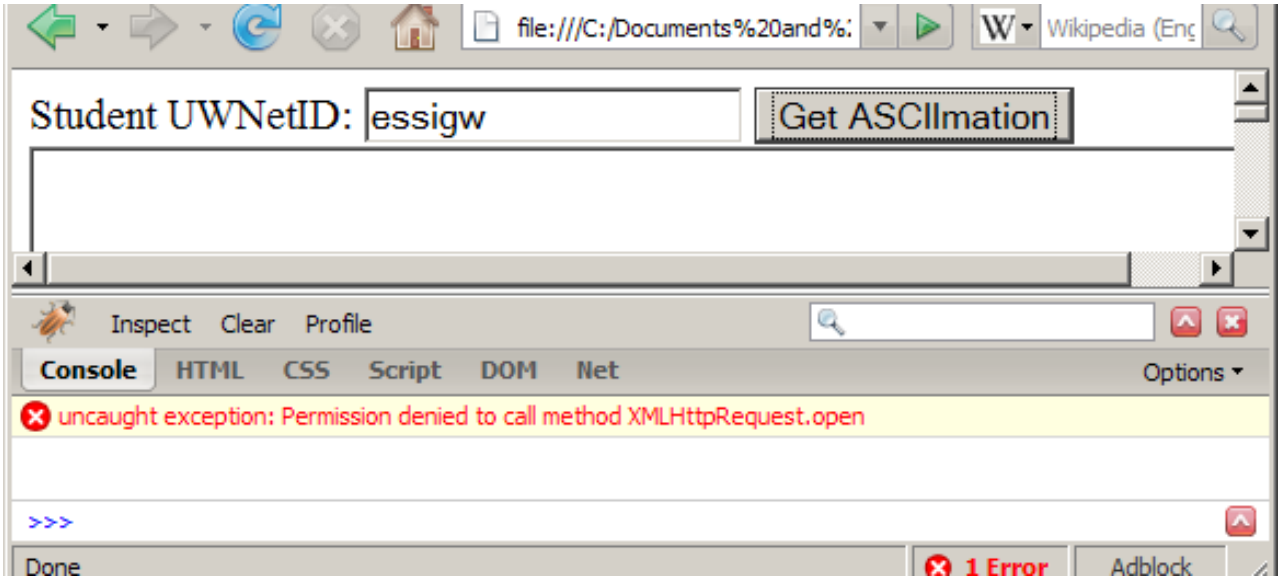
```
new Ajax.Request("url",
   {
     method: "post",    // optional
     parameters: { name: value, name: value, ..., name: value },
     onSuccess: functionName,
     onFailure: functionName,
     onException: functionName
   }
);
```
JS

- `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

Outline

- Synchronous vs. Asynchronous
- XMLHttpRequest
- AJAX in Prototype
- **Limits of AJAX**
- Debugging AJAX

XMLHttpRequest Security Restrictions



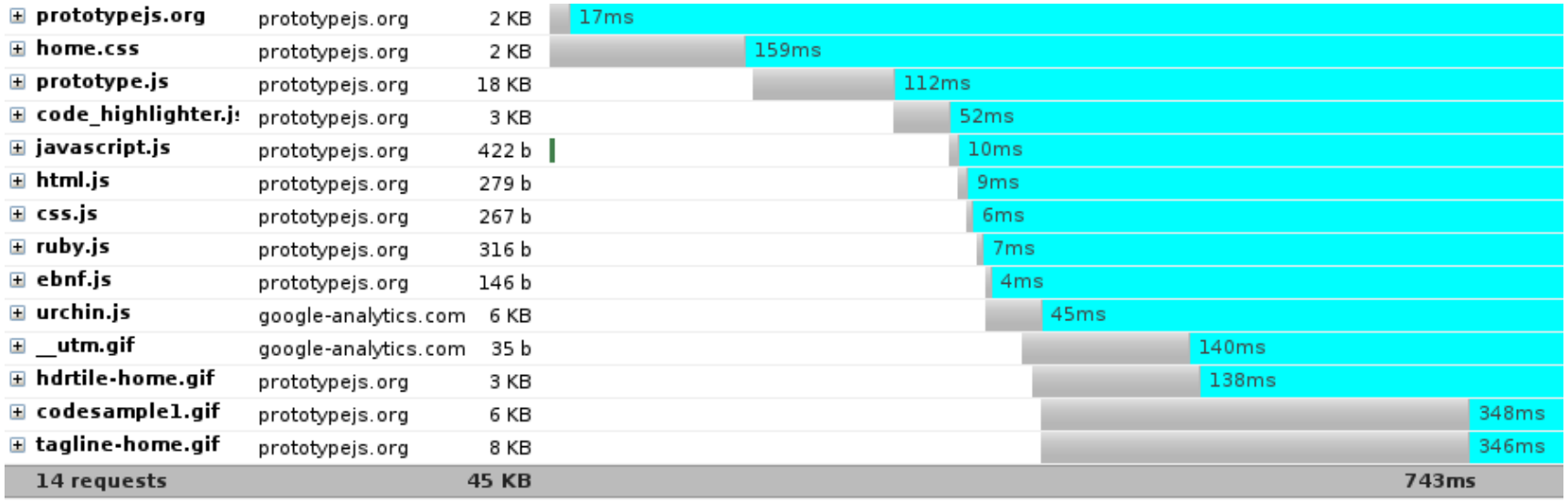
- Cannot be run from a web page stored on your hard drive
- Can only be run on a web page stored on a web server
- SOP

Same Origin Policy

- The Same Origin Policy (SOP) limits browsers only fetching content from the same origin site.
  - Except resources: images, scripts, videos, etc.
- The Same Origin Policy (SOP) essentially mandates that AJAX requests cannot access another **fully qualified domain** than the page from which they're running.
  - Even the same domain on another port!
- SOP is all about XSS (Cross Site Script)

Two-request Limit

- The HTTP 1.1 (RFC 2616) recommends that a single-user client SHOULD NOT maintain more than 2 connections with any server or proxy.
- Most browsers (including IE) abide by this rule



Outline

- Synchronous vs. Asynchronous
- XMLHttpRequest
- AJAX in Prototype
- Limits of AJAX
- **Debugging AJAX**

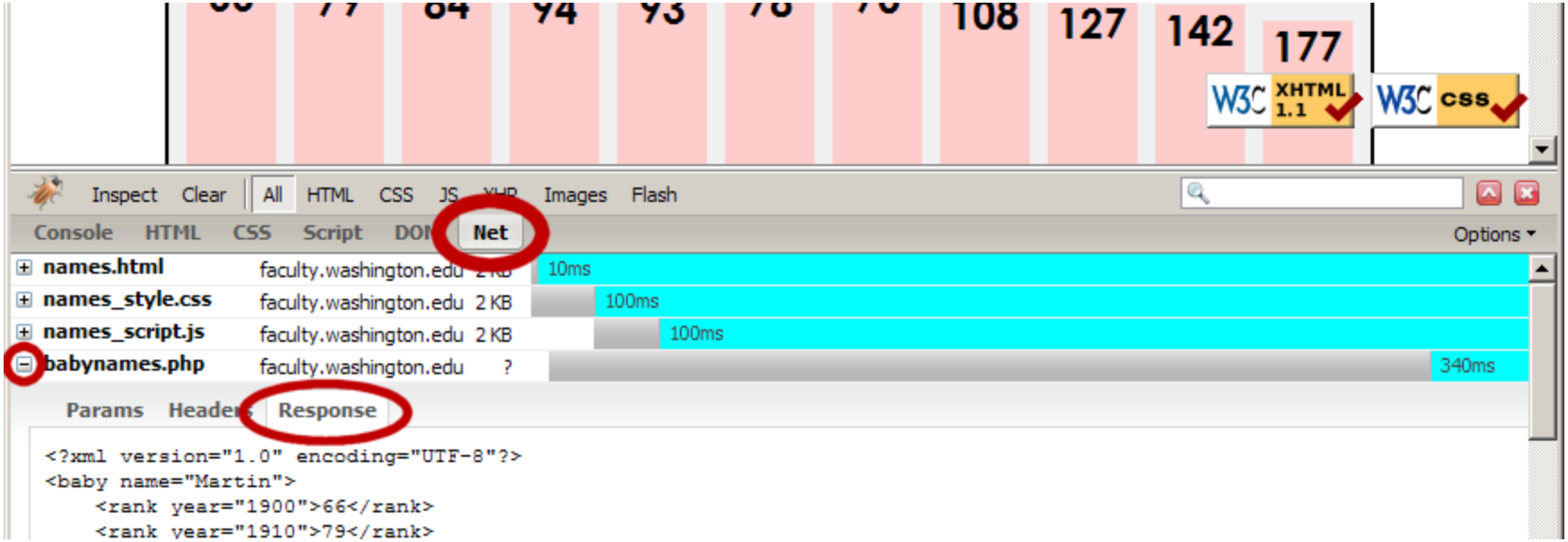
Handling AJAX Errors

```
  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

- For user's (and developer's) benefit, show an error message if a request fails

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

Summary

- Synchronous vs. Asynchronous
- XMLHttpRequest
- AJAX in Prototype
  - Ajax.Request, Ajax.Updater
- Limits of AJAX
  - SOP, Two-request limit
- Debugging AJAX

Exercises

- Write a simple **AJAX** to-do list application as a web page.
  - A element wraps all html elements
  - A form for adding new items and a list of all items
  - Buttons of "select all", "deselect all", "remove"
  - When the "add" button is clicked the new to-do item will be inserted to the bottom of the list in an AJAX way
    - Create a php script generating html the fragment for a new to-do item (something likes YOUR_NEW_TO-DO_ITEM)
    - Alter the onclick handler to issue a XMLHttpRequest by using prototype.js' Ajax.Request
      - what's the "method", "GET" or "POST"?
      - "onSuccess", "onFailure", "onException"
    - use Ajax.Updater to rewrite the handler

Further Readings

- W3C XMLHttpRequest Specification
  http://www.w3.org/TR/XMLHttpRequest
- W3School XMLHttpRequest reference
  http://www.w3schools.com/dom/dom_http.asp
- W3School AJAX tutorial
  http://www.w3schools.com/ajax/default.asp
- Google Code University AJAX tutorial
  http://code.google.com/edu/ajax/tutorials/ajax-tutorial.html
- Prototype Learning Center
  http://www.prototypejs.org/learn
- Developer Notes for prototype.js
  http://www.sergiopereira.com/articles/prototype.js.html

Thank you!