What is "Web 2.0"?

- **Web 2.0**: A set of ideas and technologies for creating modern, interactive web applications
  - Ajax, multimedia, streaming, stateful pages, cookies, user-generated content, web services, ...
What is a web service?

**web service**: software functionality that can be invoked through the internet using common protocols

- like a remote function(s) you can call by contacting a program on a web server
- many web services accept parameters and produce results
- can be written in PHP and contacted by the browser in XHTML and/or Ajax code
- service's output is often not HTML but rather text, XML, or other content types

Content ("MIME") types (1.2.3)

<table>
<thead>
<tr>
<th>MIME type</th>
<th>related file extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>.txt</td>
</tr>
<tr>
<td>text/html</td>
<td>.html, .htm, ...</td>
</tr>
<tr>
<td>text/css</td>
<td>.css</td>
</tr>
<tr>
<td>text/javascript</td>
<td>.js</td>
</tr>
<tr>
<td>text/xml</td>
<td>.xml</td>
</tr>
<tr>
<td>image/gif</td>
<td>.gif</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>.jpg, .jpeg</td>
</tr>
<tr>
<td>video/quicktime</td>
<td>.mov</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>.exe</td>
</tr>
</tbody>
</table>

- Lists of MIME types: [by type](#), [by extension](#)
Setting content type with `header`

```php
header("Content-type: type/subtype");

header("Content-type: text/plain");
print("This output will appear as plain text now!
");
```

- by default, a PHP script's output is assumed to be HTML
- use the `header` function to specify non-HTML output
  - must appear before any other output generated by the script

Example: Exponent web service

- Write a web service that accepts a `base` and `exponent` and outputs `base` raised to the `exponent` power. For example, the following query should output 81:

  ```url
  http://example.com/exponent.php?base=3&exponent=4
  ```

- solution:

```php
header("Content-type: text/plain");
$base = $_REQUEST["base"]; $exp = $_REQUEST["exponent"]; $result = pow($base, $exp); print $result;
```
Recall: HTTP GET vs. POST (6.3.3)

- **HTTP**: the set of commands understood by a web server and sent from a browser
  - **GET**: asks a server for a page or data
    - if the request has parameters, they are sent in the URL as a query string
  - **POST**: submits data to a web server and retrieves the server's response
    - if the request has parameters, they are embedded in the request's HTTP packet, not
      the URL
  - For submitting data, a POST request is more appropriate than a GET
    - GET requests embed their parameters in their URLs
    - URLs are limited in length (~ 1024 characters)
    - URLs cannot contain special characters without encoding
    - private data in a URL can be seen or modified by users

---

The **$_SERVER** superglobal array

<table>
<thead>
<tr>
<th>index</th>
<th>description</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$_SERVER[&quot;SERVER_NAME&quot;]</td>
<td>name of this web server</td>
<td>&quot;webster.cs.washington.edu&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;SERVER_ADDR&quot;]</td>
<td>IP address of web server</td>
<td>&quot;128.208.179.154&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;REMOTE_HOST&quot;]</td>
<td>user's domain name</td>
<td>&quot;hsdl.wa.comcast.net&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;REMOTE_ADDR&quot;]</td>
<td>user's IP address</td>
<td>&quot;57.170.55.93&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;HTTP_USER_AGENT&quot;]</td>
<td>user's web browser</td>
<td>&quot;Mozilla/5.0 (Windows; ...&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;HTTP_REFERER&quot;]</td>
<td>where user was before this page</td>
<td>&quot;<a href="http://www.google.com/">http://www.google.com/</a>&quot;</td>
</tr>
<tr>
<td>$_SERVER[&quot;REQUEST_METHOD&quot;]</td>
<td>HTTP method used to contact server</td>
<td>&quot;GET&quot; or &quot;POST&quot;</td>
</tr>
</tbody>
</table>
• call `phpinfo();` to see a complete list
GET or POST?

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    # process a GET request
    ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    # process a POST request
    ...
}
```

- some PHP web services process both GET and POST requests
- you can find out which kind of request we are currently processing by looking at the "REQUEST_METHOD" key of the global $_SERVER array
- you can also access query parameters through $_GET and $_POST rather than $_REQUEST

Emitting partial-page HTML data

```php
# suppose my web service accepts a "type" query parameter ...
if ($_REQUEST["type"] == "html") {
    # client wants their output to be HTML format
    ~> <ul>
    <?php
    foreach ($students as $kid) {
        ~> <li> <?= $kid ?> </li>
        <?php
    }
    ~> </ul>
    <?php
}
```

- some web services do output HTML, but not a complete page
- the partial-page HTML is meant to be fetched by Ajax and injected into an existing page
Emitting XML data

header("Content-type: text/xml");
print("<?xml version="1.0" encoding="UTF-8"?>\n");
print("<books>\n");
foreach ($books as $title) {
  print("<book title="$title" />\n");
}
print("</books>\n");

- specify a content type of text/xml or application/xml
- print an XML prologue (the <?xml line) first
  - important: no whitespace output can precede the prologue
- then print each line of XML data/tags as output
- some PHP libraries automatically generate XML for you from other data (e.g. databases)

Reporting errors

- how does a web service indicate an error to the client?
  - error messages (print) are not ideal, because they could be confused for normal output
- web service should return an HTTP "error code" to the browser, possibly followed by output
  - these are the codes you see in Firebug's console and in your Ajax request's .status property

<table>
<thead>
<tr>
<th>HTTP code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
</tr>
<tr>
<td>301-303</td>
<td>page has moved (permanently or temporarily)</td>
</tr>
<tr>
<td>400</td>
<td>illegal request</td>
</tr>
<tr>
<td>403</td>
<td>you are forbidden to access this page</td>
</tr>
<tr>
<td>404</td>
<td>page not found</td>
</tr>
<tr>
<td>500</td>
<td>internal server error</td>
</tr>
</tbody>
</table>

complete list
Using headers for HTTP error codes

```php
header("HTTP/1.1 code description");

if ($_REQUEST['foo'] != 'bar') {
    # I am not happy with the value of foo; this is an error
    header("HTTP/1.1 400 Invalid Request");
    die("An HTTP error 400 (invalid request) occurred.");
}

if (!file_exists($input_file_path)) {
    header("HTTP/1.1 404 File Not Found");
    die("HTTP error 404 occurred: File not found ($input_file_path)");
}
```

- `header` can also be used to send back HTTP error codes
  - `header("HTTP/1.1 403 Forbidden");`
  - `header("HTTP/1.1 404 File Not Found");`
  - `header("HTTP/1.1 500 Server Error");`