Chapter 2: Interactive Web Applications

Example: Fibonacci Function in PHP (Version 1)

```
<body> ...
  <h2>
    <?php
        function fib($n){
            if ($n==0)
                return 0;
            else
                if ($n==1)
                    return 1;
                else
                    return fib($n-1)+fib($n-2);
        };
        echo "fib(3) = ", fib(3), "<br>";
        echo "fib(8) = ", fib(8), "<br>";
    ?>
  </h2>
</body>
</html>
```
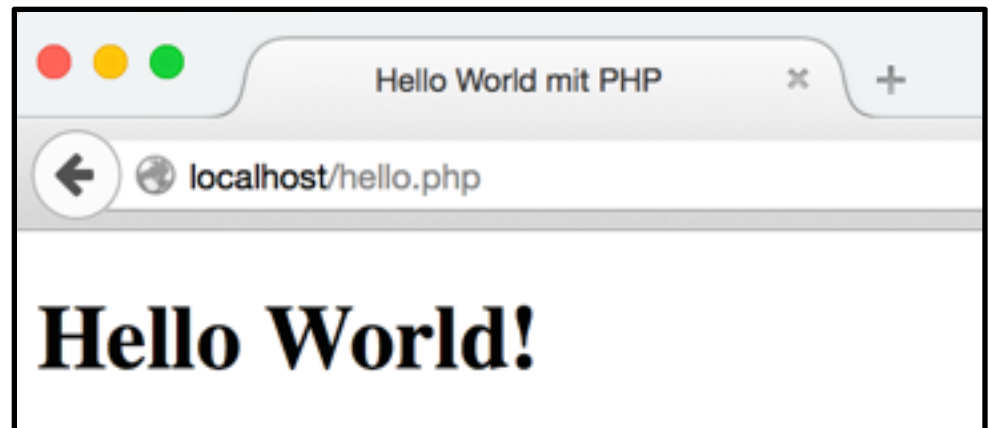
fibonacci1.php

HTTP Basics

- HTTP = HyperText Transfer Protocol, see http://www.w3.org/Protocols/
- Client-Server communication:
  - Client opens (TCP) connection to server (usually on port 80)
  - Client sends request (as text lines)
  - Server sends response (as text lines)
  - Client closes connection (HTTP is *stateless*)
- Format of all HTTP messages (requests and responses):

  Initial line

  Header lines (zero or more)

  Blank line

  Message body (optional)

- Example HTTP request:

  `GET /lehre/ws1516/mmn/index.html HTTP/1.1`

  `Host: www.medien.ifi.lmu.de:80`

  <blank line!>

Sample HTTP Request (GET)

```
GET /~hussmann/hello.php HTTP/1.1

ACCEPT: text/html,application/xhtml+xml,application/
xml;q=0.9,*/*;q=0.8

ACCEPT_ENCODING: gzip, deflate

ACCEPT_LANGUAGE: en-GB,en;q=0.5

CONNECTION: keep-alive

HOST: localhost

USER_AGENT: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10;
rv:41.0) Gecko/20100101 Firefox/41.0

CONTENT_TYPE:
```

HTTP Server Responses

- Message sent back from HTTP server always contains an initial response line which gives the *status* of the request processing.

- Example (success):

  `HTTP/1.1 200 OK`

- Example (error):

  `HTTP/1.1 404 Not found`

- Status codes:
  - 1xx: Informational message
  - 2xx: Success of some kind
  - 3xx: Redirection to other URL
    - e.g. 303: See other URL (given in Location: header)
  - 4xx: Client side error
  - 5xx: Server side error
    - e.g. 500: Server error

Example HTTP Response

- Experimenting manually with HTTP client/server dialogues:
  - "`telnet <host> 80`" in UNIX shell

- Retrieving a HTML page:

```
GET /~hussmann/hello.php HTTP/1.1
Host: localhost:80
```

- Response:

```
HTTP/1.1 200 OK
Date: Wed, 28 Oct 2015 14:02:38 GMT
Server: Apache/2.4.16 (Unix) PHP/5.5.29
X-Powered-By: PHP/5.5.29
Content-Length: 126
Content-Type: text/html

<!DOCTYPE html> … <html> … </html>
```

CGI-Style Coding for Parameters in GET Request

- Convention for passing parameter values to server-side programs
  - Introduced by the *Common Gateway Interface (CGI)*
  - Not part of the HTML protocol!
  - Interpreted by server programs, e.g. PHP module
- Syntax:
  - Parameter data stream is appended to URL after a "?"
  - Keyword/value pairs, separated by "=", e.g. "`fibinput=12`"
  - Multiple parameter groups are separated by "&"
  - Spaces in strings are replaced by "+"
  - Non-ASCII characters (and special characters "&", "+", "=", "%") are replaced by "%*xx*" (hexadecimal code of character in used character set)

Fibonacci Function in PHP: Using Request Data

> **Think about this code:**
> `"fib($fibinput) = "`

```
<body>
  <h1>
    Fibonacci Function (Result)
  </h1>
  <h2>
    <?php
        $fibinput = $_REQUEST['fibinput'];
        function fib($n){ as in version 1 };
        echo "fib($fibinput) = ";
        echo fib($fibinput);
        echo "<br>";
    ?>
    <br>
    <a href="fibonacci2a.html">New Computation</a>
  </h2>
</body>
```

fibonacci2b.php

Example GET Request with Parameter

- Request:

```
GET /~hussmann/fibonacci2b.php?fibinput=10 HTTP/1.1
Host: localhost
```

- Response:

```
Date: Wed, 28 Oct 2015 14:09:15 GMT
Server: Apache/2.4.16 (Unix) PHP/5.5.29
X-Powered-By: PHP/5.5.29
Content-Length: 245
Content-Type: text/html

<!DOCTYPE html>

<html>
<head> … fib(10) = 55 … </html>
```

GET and POST Methods in HTTP

Hypertext Transfer Protocol (HTTP) supports two request methods for passing parameter values to called documents/scripts:

- **GET Method:**
  - Parameter values transmitted within URL:

    `http://host.dom/path/fibonacci2.php?fibinput=10`

  - CGI-Style coding (see above)

- **POST Method:**
  - Parameter values transmitted in the HTTP message body
  - Parameter values not visible in URL

- Coding options for POST method:

  - Not part of HTTP (but specified for HTML forms)!
  - Coding method given in the `Content-Type` header
    - » `application/x-www-form-urlencoded` (CGI style)
    - » `multipart/form-data` (segmented data, better for large data blocks)

Example POST Request with Parameter

- Request:

```
POST /~hussmann/fibonacci2b.php HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

fibinput=12
```

- Response:

```
HTTP/1.1 200 OK
Date: Wed, 28 Oct 2015 14:17:54 GMT
...
Content-Type: text/html

<!DOCTYPE html>

<html>
<head> … fib(12) = 144 … </html>
```

PHP: Variables, Parameter Passing and Security

- Global arrays `$_REQUEST`, `$_GET`, `$_POST`
  - for accessing external values determined at call time (like form input)
  - `$_REQUEST` contains all parameters given in request,
    `$_GET` and `$_POST` contains all parameters passed by the resp. method
  - Obtaining individual variable values by array lookup:
    `$_REQUEST['var'];`

- Older PHP versions (up to 4.2.0):
  - External values were directly accessible through variables
    (like "`$fibinput`")
  - ***Where is the problem?***

HTML Reminder: Forms

- User input in HTML:

  `<form>` Element

- Sub-element:

  - `<input type=ty name=name>`

    Selected classic (HTML 4) types (*ty)*:

`checkbox`	Check box (Attribute `checked`)
`radio`	Radio button (Attribute `checked`)
`text`	Text input line
`textarea`	Multi-line text input area
`password`	Text input area not displaying the input
`file`	File selection
`button`	General button
`submit`	Button to send form contents
`reset`	Button to reset form contents

  - `<select name=name>` Pop-up menu for selection from options

    List of options: Sub-elements `<option>`

    `<option selected>` defines "pre-selected" values

HTML Form Example

```
<body>
    <form action="test.php"
        method="GET"
        enctype="application/x-www-form-urlencoded">
        <label>Name <input type="text" name="name"
            maxlength="10"/></label><br>
        Sex:
        <input type="radio" name="sex"
            value="male"></input> male
        <input type="radio" name="sex"
            value="female"></input> female <br>
        <input type="checkbox" name="married"
            value="yes">Married</input><br>
        <input type="submit" value="Submit" />
    </form>
</body>
```

/test.php?name=Max+Muster&sex=male&married=yes

HTML Forms and Server-Side Scripts

- HTML page containing forms
  - calls separate script page
  - transfers form data as variable values
- **action** attribute for HTML tag **<form>**
  - Specifies the server page to process the input
  - Can contain embedded script
- **method** attribute for HTML tag **<form>**
  - Specifies the HTTP method to be used to transfer form data to the server
  - Possible values: GET (default), POST
- **enctype** attribute for HTML tag **<form>**
  - Specifies the encoding method to be used for form data
  - Possible values:
    » application/x-www-form-urlencoded (CGI conventions) (default)
    » multipart/form-data (segmented data)

Example: POST Request with Multipart Encoding

- HTML:

```
<form action="test.php"
    method="POST" enctype="multipart/form-data">
```

- Generated HTTP request:

```
POST /test.php HTTP/1.1
Host: localhost …
Content-Type: multipart/form-data;
boundary=---------------------------103832778631715
Content-Length: 355

-----------------------------103832778631715
Content-Disposition: form-data; name="name"

Max Muster
-----------------------------103832778631715
Content-Disposition: form-data; name="sex"

male
-----------------------------103832778631715
Content-Disposition: form-data; name="married"

yes
-----------------------------103832778631715--
```

Fibonacci Function in PHP (Version 2): Input Form Calling PHP Script

```html
<body>
  <h1>
     Fibonacci Function (Input)
  </h1>
  <h2>
     Please enter number:
     <form name="fibform" action="fibonacci2b.php">
         <input type="text" name="fibinput"
            value="0"><br>
         <input type="submit" value="Compute">
     </form>
  </h2>
</body>
</html>
```

fibonacci2a.html

Combination of Input and Result Pages

fibonacci2.php

```php
<body>
  <h1>
    Fibonacci Function
  </h1>
  <h2>
    <?php
      function fib($n){  as above  };
      if (isset($_REQUEST['fibinput']) && $_REQUEST['fibinput']!="") {
        $fibparam = $_REQUEST['fibinput'];
        echo "fib($fibparam) = ";
        echo fib($fibparam);
        echo "<br>";
      }
    ?>
    <br/>
    Please enter number:
    <form name="fibform" action="fibonacci2.php">
      <input type="text" name="fibinput" value="0"><br>
      <input type="submit" value="Compute">
    </form>
  </h2>
</body>
```

> action="fibonacci2.php" can be omitted

Form Validation, Traditional Style

- Constraints for data entered into input forms:
  - required / optional
  - Special formats:
    Date
    URL
    Email address
- Checking the constraints ("validating" the input)
  - Client-side script code (JavaScript)
  - Catching the "submit" event
  - Data submitted only if validation returns true
- Client- vs. server-side validation:
  - Advantages, disadvantages?

Example: Traditional Form Validation

```html
<form id="blogentry">
  <label for="name">Name: </label>
  <input name="name" type="text"></br>
  <label for="email">Email: </label>
  <input name="email" type="text">
  <input type="submit" value="Submit">
</form>
<script type="text/javascript">
  blogentry = document.getElementById("blogentry");
  blogentry.addEventListener("submit", validateForm, false);
  function validateForm() {
    if (blogentry.name.value =="") {
      alert("Name is required");
      return false;
    };
    var emailinput=blogentry.email.value;
    var atpos=emailinput.indexOf("@");
    var dotpos=emailinput.lastIndexOf(".");
    if (atpos<1 || dotpos<atpos+2 || dotpos+2>=emailinput.length) {
      alert("Not a valid e-mail address");
      return false;
    };
    return true;
  }
</script>
```

html/formvalidate.html
Email validation code taken from w3schools.org

Detour: Accessing HTML Elements in JavaScript

- Old-fashioned JavaScript document tree:
    - Array access: `document.forms[f].elements[e]`
    - Shorthand: `document.forms.f.elements.e` (associative array)
    - Even shorter: `document.f.e`
- Strict DOM style:
    - `document.getElementById("f")`
- HTML5 Recommendation (Oct 28, 2014), Sect. 5.2.4:

    The Window interface <u>supports named properties</u>. The <u>supported property names</u> at any moment consist of the following, in <u>tree order</u>, ignoring later duplicates:

    - the browsing context name of any child browsing context of the active document whose name is not the empty string,

    - the value of the <u>name</u> content attribute for all <u>a</u>, <u>applet</u>, <u>area</u>, <u>embed</u>, <u>form</u>, <u>frameset</u>, <u>img</u>, and <u>object</u> elements in the active document that have a non-empty <u>name</u> content attribute, and

    - the value of the <u>id</u> content attribute of any <u>HTML element</u> in the active document with a non-empty <u>id</u> content attribute.

- Note that `window` is equivalent to `self` in JavaScript and can be omitted!
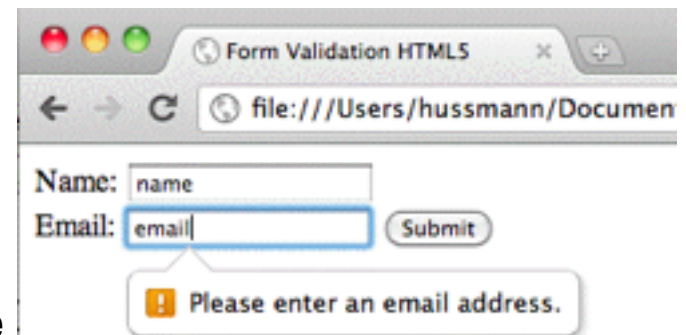
Form Validation with HTML5

- Standard scenarios of form validation are integrated into HTML5 standard
  - Input types: email, URL, date, time, number, range, search, phone number, color
  - Attributes: Required, min, max, step, pattern
- **Procedural** features are transformed to **declarative** features
- Declarative HTML5 replacing JavaScript code:
  - less error-prone
  - more precise (regarding definition of input syntax)
  - automatically benefits from upgrades
  - devices (e.g. smartphones) can choose best representation
- Transition problem!

Example: Form Validation with HTML5

```html
<!DOCTYPE html>

<html>
  <head>
    <title>Form Validation HTML5</title>
  </head>

  <body>
    <form name="blogentry">
      <label for="name">Name: </label>
      <input id="name" type="text" required></br>
      <label for="email">Email: </label>
      <input id="email" type="email" required>
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

formvalidate5.html   Google Chrome

Chapter 2: Interactive Web Applications

Literature:

    B. Lawson, R. Sharp: Introducing HTML5, New Riders 2011
    S. Fulton, J. Fulton: HTML5 Canvas, 2nd ed., O'Reilly 2013

Data Storage Options in the Web: Overview

- Client-side storage (implemented in browser):
  - Session level: Linking consecutive request/response pairs
  - Long-term level: Personalization, preferences
  - Traditional solution: Cookies
  - Modern solutions (HTML5): Web Storage, Web SQL Databases
- Server-side storage (implemented on server):
  - Access and modify external/global information
  - Simple solution: Server files (see PHP example "forum" below)
  - Powerful solution: Database access from server scripts
- Using relational databases, SQL-based:
  - Traditional solution, based on tables
  - Mixture of languages and paradigms
- Using non-SQL databases, e.g. "MongoDB":
  - Trending solution, based on document trees
  - Fully coherent with JavaScript

A Simple Discussion Forum (1)

- Interactive submission of text contributions

- Display of all submissions available on server

- Server uses simple text file for storage

- Altogether approx. 50 lines of HTML+PHP !

Discussion Forum

New Contribution:

Name: _____

Contribution (one line): _____

(Enter new contribution) (Reset)

Current discussion:

3 contributions

Contribution # 1:

Name: Max
Text: I have an idea

Contribution # 2:

Name: Peter
Text: I like this idea

Contribution # 3:

Name: Janet
Text: I don\'t like it

A Simple Discussion Forum (2)

Contents of file "forum.txt":

- – Each two consecutive lines represent one contribution.
- – First line: Name
- – Second line: Text

```
Max
I have an idea
Peter
I like this idea
```

Beware of access rights…

A Simple Discussion Forum (3)

Display of the full content of the file 'forum.txt'

- Used file function:
  - `file():` Converts file content to string array
- Used array function:
  - `count():` Length of array

```php
<h2>Current discussion:</h2>
<?php
   $content = file("forum.txt");
   echo "<h3>", count($content)/2, " contributions</h3>";
   echo "<hr>";
   $i = 0;
   while ($i < count($content)) {
       echo "<h3>Contribution # ", ($i+2)/2, ":</h3>";
       echo "<b>Name: </b>", $content[$i++], "<br>";
       echo "<b>Text: </b>", $content[$i++], "<br>";
       echo "<hr>";
   }
?>
```

forum.php

A Simple Discussion Forum (4)

Input interface (HTML form):

```html
<h1>Discussion Forum</h1>
 <hr>
 <h2>New Contribution:</h2>
 <form method="post">
    <table border="0">
       <colgroup>…</colgroup>
        <tr>
          <td>Name:</td>
          <td><input type="text" name="name"></td>
        </tr>
        <tr>
          <td>Contribution (one line):</td>
          <td><input type="text" name="contrib" size="60"></td>
        </tr>
    </table>
    <input type="submit" name="newcontrib"
          value="Enter new contribution">
    <input type="reset"">
 </form>
```

forum.php

A Simple Discussion Forum (5)

Extending the file 'forum.txt' with a new contribution

- **`$newcontrib`**: was the "enter contribution" button pressed?
- Used file functions:
    - **`fopen(), fclose():`** Open file ("a"=append), close file
    - **`fputs():`** Write string to file

```php
<?php
  $newcontrib = $_REQUEST['newcontrib'];
  $name = $_REQUEST['name'];
  $contrib = $_REQUEST['contrib'];
  if ($newcontrib != "" && $name != "" && $contrib != "") {
      $file = fopen("forum.txt", "a");
      if ($file) {
          fputs($file,$name . "\n");
          fputs($file,$contrib . "\n");
          fclose($file);
      }
  }
?>
```
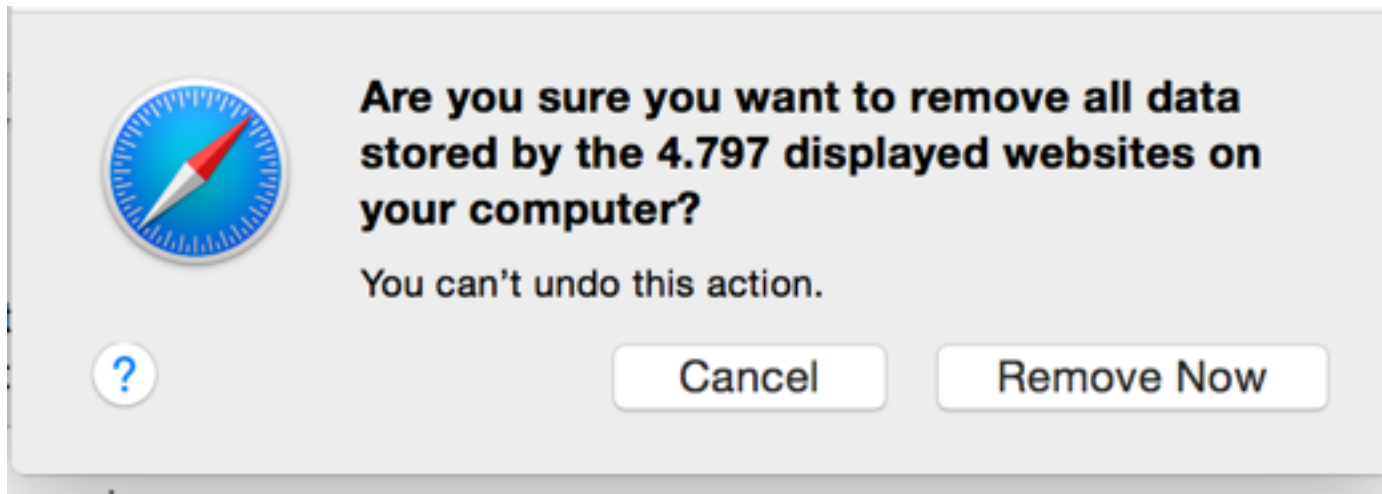
Sessions and States

- HTTP is stateless
  - Server does not "remember" any data from previous transactions
- Linking several transactions to a "session" with common data storage
  - Client-side: Storing all data on client and re-transmit for every transaction
  - Server-side: Storing all data on server, client has to identify the session
- Common solution:
  - Server-side software offers session support
    - » E.g. session support in PHP
  - Client stores "session id"
  - Methods for linking request to session id:
    - » Variable/value pair in GET or POST request
    - » HTTP "Cookie"

Cookies in HTTP

- Small data units stored in the browser storage area, controlled by browser

- Cookie contains:
  - *Name* (String), also called *key*
  - *Value* (String)
  - *Expiration date*
  - optional: domain, path, security information

- HTTP transfers cookies between client and server
  - In response, server can include header line "Set-Cookie:"
    - » Further information: name + value pair, expiration time
  - Cookie is stored by the browser
  - In further requests to the same server, client includes header line "Cookie:"
    - » Further information: name + value pair
  - Only cookies related to the requested server are transferred

What Will Change After This Step?

Are you sure you want to remove all data stored by the 4.797 displayed websites on your computer?

You can't undo this action.

Cancel   Remove Now

Types of Cookies

- Session cookie
  - Deleted on browser termination
  - No expiration date given = session cookie
- Persistent cookie
  - For tracking, personalization
- Secure cookie
  - Only transmitted when secure connection to server is used
- HttpOnly cookie
  - Access only for HTTP, not for script APIs
- Third party cookie
  - Cookies set for different domain than currently visited server
  - Used for tracking and cross-domain advertising

Cookies in PHP: Listing Current Cookies

Cookies currently set:

- cookie1=text1
- Test=test_text

cookie_list.php

Accessing Cookies

Displaying a list of all cookies currently set (for this application) by reading from global array $_COOKIE:

```php
<html>
  <h2>Cookies currently set:</h2>
  <ul>
  <?php
    while (list($k, $v) = each($_COOKIE))
      echo "<li>", $k, "=", $v, "</li>";
  ?>
  </ul>
...
</html>
```

cookie_list.php

HTML Form for Setting a Cookie

```html
<form>

    <input type="text" name="key" value="name">
       Cookie Name<br>

    <input type="text" name="val" value="text">
       Cookie Content<br>

    <input type="text" name="tim" value="10">
       Lifetime (minutes)<br>

    <input type="submit" name="set"
       value="Set Cookie"><br>

</form>
```

- Page loaded via **action** is identical to page containing the form
  – when omitting the **action** attribute.
- Server-side execution: actual setting action carried out when *next* page is loaded!

cookie_set.php

Setting the Cookie

```php
<?php
  if if (isset($_GET['set']) && $_GET['set']!="") {
     $key = $_GET['key'];
     $val = $_GET['val'];
     $tim = $_GET['tim'];
     $exp = time() + $tim * 60;
     setcookie($key, $val, $exp);
  }
?>
<!DOCTYPE html>

<html>

...
```

- "name" attribute of submit button ('set') is used to decide whether set button was pressed
- setcookie() call has to be very first output of page, to be transmitted together with the headers (HTTP requirement).

Client-Side Storage using *Web Storage*

- Web Storage/DOM Storage:
  - Standardized by W3C, intended as improvement over Cookies
  - Purely client-side storage
  - Not transmitted to server with each request
  - Javascript code can issue read and write requests
- Types of storage:
  - Session storage: Related to window/tab (!)
  - Local storage: Related to domain and maintained after browser termination
- Data structure:
  - Simple associative array (key/value pairs, both of string type)
  - Similar to Cookies

Web Storage Example

http://www.braekling.de/testlab/html5-webstorage-demo.html

Web Storage Interface (W3C)

- Interface `Storage` (defined independently of implementation language):

  ```
  String getItem(String key);
  void setItem(String key, String value);
  void removeItem (String key);
  void clear();
  ```

- Top-level browsing context contains two attributes:

  ```
  Storage sessionStorage;
  Storage localStorage;
  ```

- Shorthand notation in JavaScript due to associative array, example:

  ```
  var firstName = localStorage.firstName;
  var lastName = localStorage.lastName;
  ```

- When a storage area changes, an event is fired:

  ```
  StorageEvent storage;
  ```

JSON Stringification

- Converting data objects to a String representation
  - XML based
  - For JavaScript: Space-effective JSON notation
    (= JavaScript Object Notation)
- APIs:
  - JavaScript: `JSON.stringify(), JSON.parse()`
  - PHP: `json_encode(), json_decode()`
- JSON Example:

```
{"student": {
 "identification": [
      {"name": "firstname",
       "value": "Max"
      },
      {"name": "lastname",
       "value": "Muster"
      }],
  "grades": […]
  }
}
```

See php/json_php.php
php/json_php_js.php

Working Offline in Web Applications

- Why using Web applications offline?
    - …

- Working offline with server-based applications:
    - Client needs a significant amount of logic to give sense to offline work
    - Specify which parts of the application data is to be kept locally *(cached)*
        » Usually a set of files
        » *Cache manifest* (= list of files)
    - Browser needs to support access to cached data
        » interpret cache manifest
        » maintain application cache

Potential Enabled by Server-Side Scripts

- Receive and store user input
    - In various forms of persistent storage
- Process input and compute results
    - Depending on various information available on server side
- Create output suitable for being displayed in Web browsers
    - HTML, may include JavaScript
- Make use of advanced features offered by Web browsers
    - Examples: Cookies, user agent identification