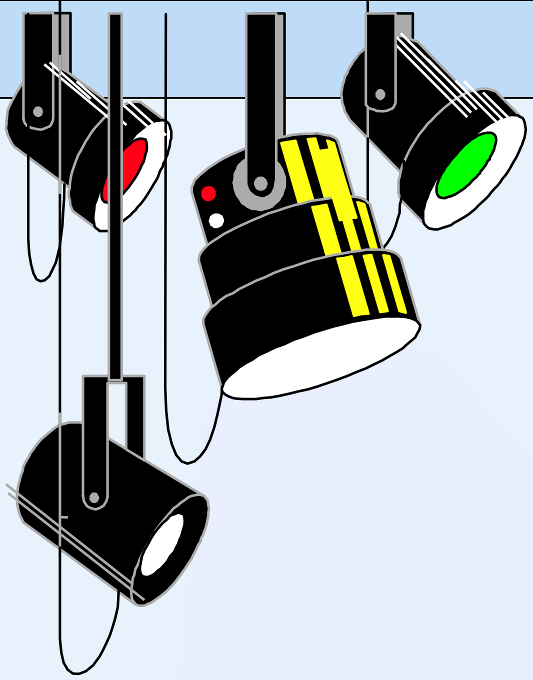


Web Databases

- Web database
- System architecture
- Web programming language:
  - PHP
  - Node.js

Web Databases

- What is a web database?
  - A database accessed from the Internet
  - E-commence and other Internet applications are designed to interact with the user through *web interfaces*
  - An online flight ticket booking system

    web interface:

    input - customer information: time, location, airport, destination

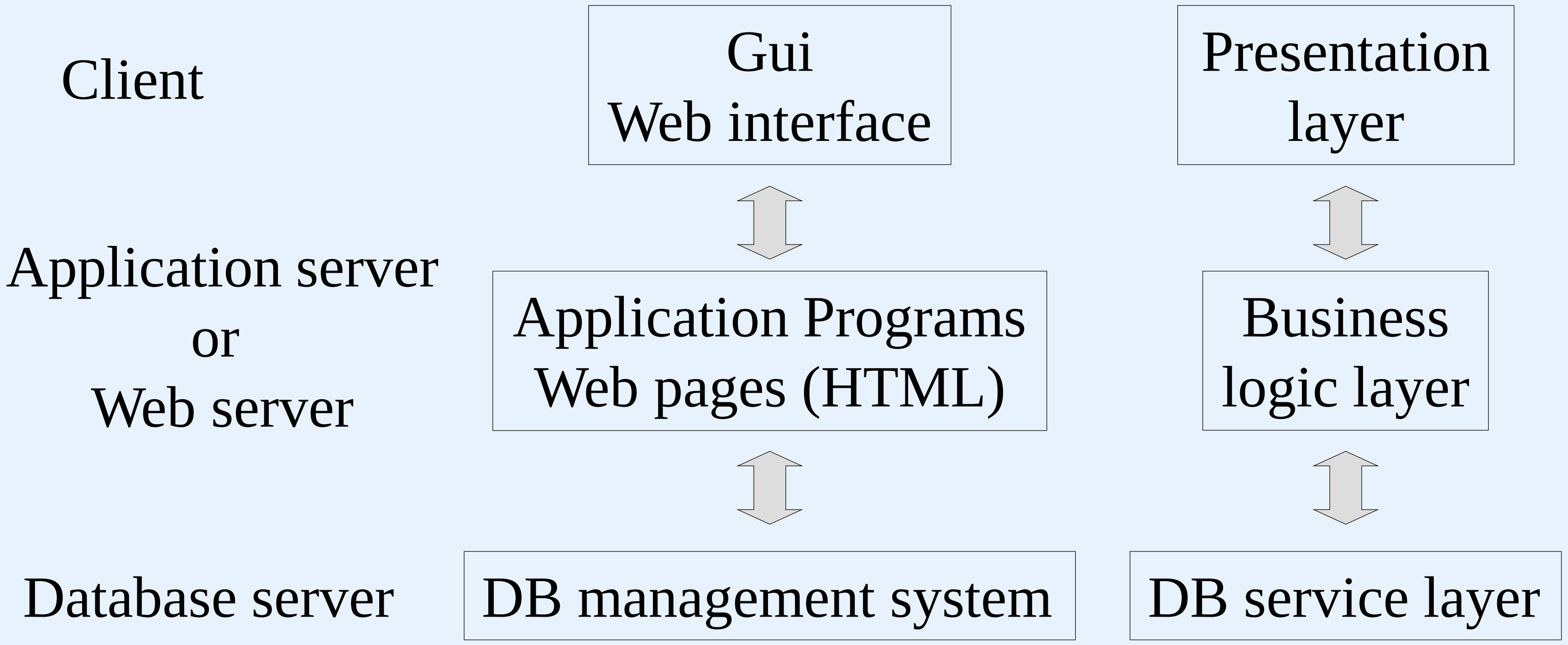
    output – departure time, arrival time, flight number, price

    database access:

    query evaluation

Web Databases

- Three-tier architecture:



Web Databases

- **Web server language (script language): PHP**

  - PHP – a script language, used to generate *dynamic HTML pages*.

    PHP programs are executed on Web server computers. (This is in contrast to some scripting languages, such as JavaScript, which are executed on client computers.)

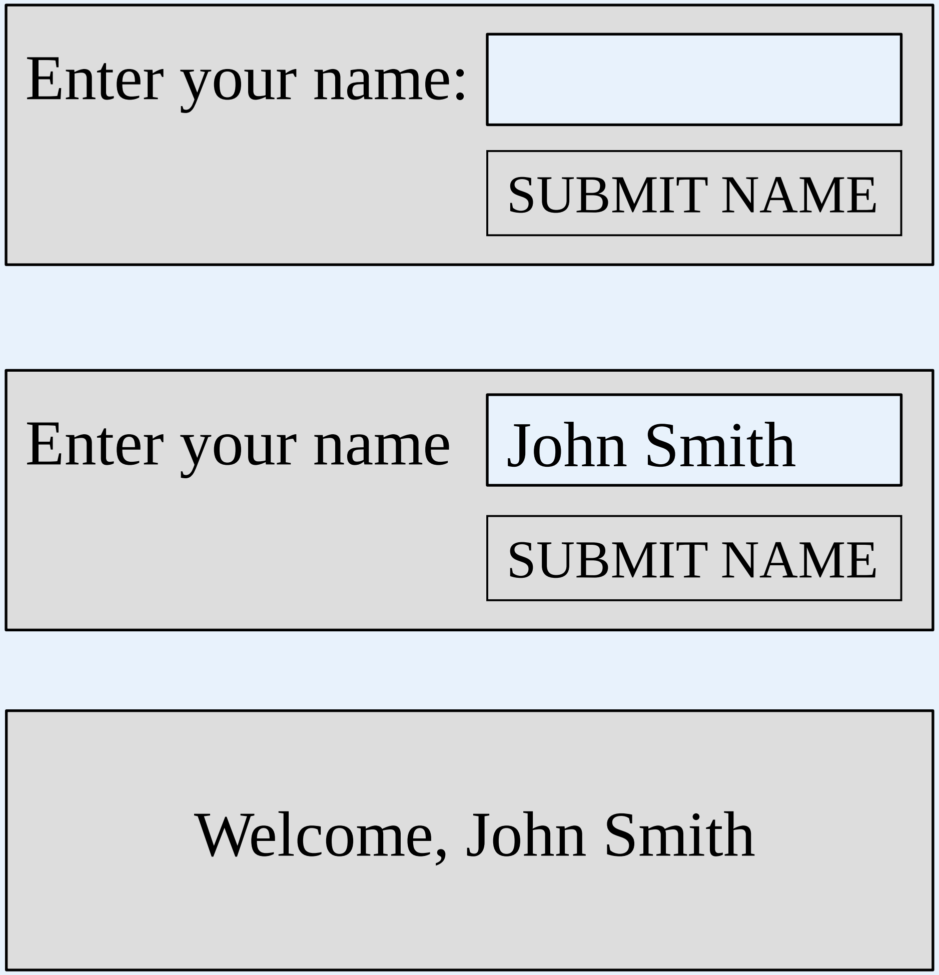
  - The official PHP website has installation instructions for PHP: http://PHP.org.net

  - PHP 5 and later versions can work with a MySQL database using:
    - MySQLi extension (the 'i' stands for improved)
    - PDO (PHP Data Objects)

- **A simple PHP example**
  - The program prompts a user to enter the first and last name and then prints a welcome message to that user.

```
<?PHP
    //Printing a welcome message if the user submitted his/her name
    //through the PHP form
    if ($_post['user_name']) {
            print("Welcome, ");
            print($_post['user_name']); }
    else { print <<<_HTML_
            <FORM method="post" action="$_SERVER['PHP_SELF']">
            Enter your name: <input type="text" name="user_name">
            <BR/>
            <INPUT type="submit" value="SUBMIT NAME"></FORM>
             _HTML_;
            }
?>
```

Web Databases



Enter your name:

SUBMIT NAME

Enter your name John Smith

SUBMIT NAME

Welcome, John Smith

Web Databases

- A PHP script is enclosed with a pair of tags:

  start tag: <?php

  end tag: ?>

  Stored in a file, named, for example,

  greeting.php,

  and located in an address, for example,

  http://www.myserver.com/examples/greeting.php.

- You can also put it is a HTML file.

Web Databases

```
<!DOCTYPE html>
  <html><body>
      <h1>My first PHP page</h1>
      <?PHP
            if ($_post['user_name']) {
            print("Welcome, ");
            print($_post['user_name']); }
            else { print <<<_HTML_
                     <FORM method="post" action="$_SERVER['PHP_SELF']">
                     Enter your name: <input type="text" name="user_name">
                     <BR/>
                     <INPUT type="submit" value="SUBMIT NAME">
                     </FORM>
                     _HTML_;
                     }
      ?>
    </body></html>
```

Web Databases

- Single line Comments started by //, or started by #
  Multiple-line comments start with /* and end with */

- The predefined PHP variable $post is an array that hold all the variables entered through form parameters

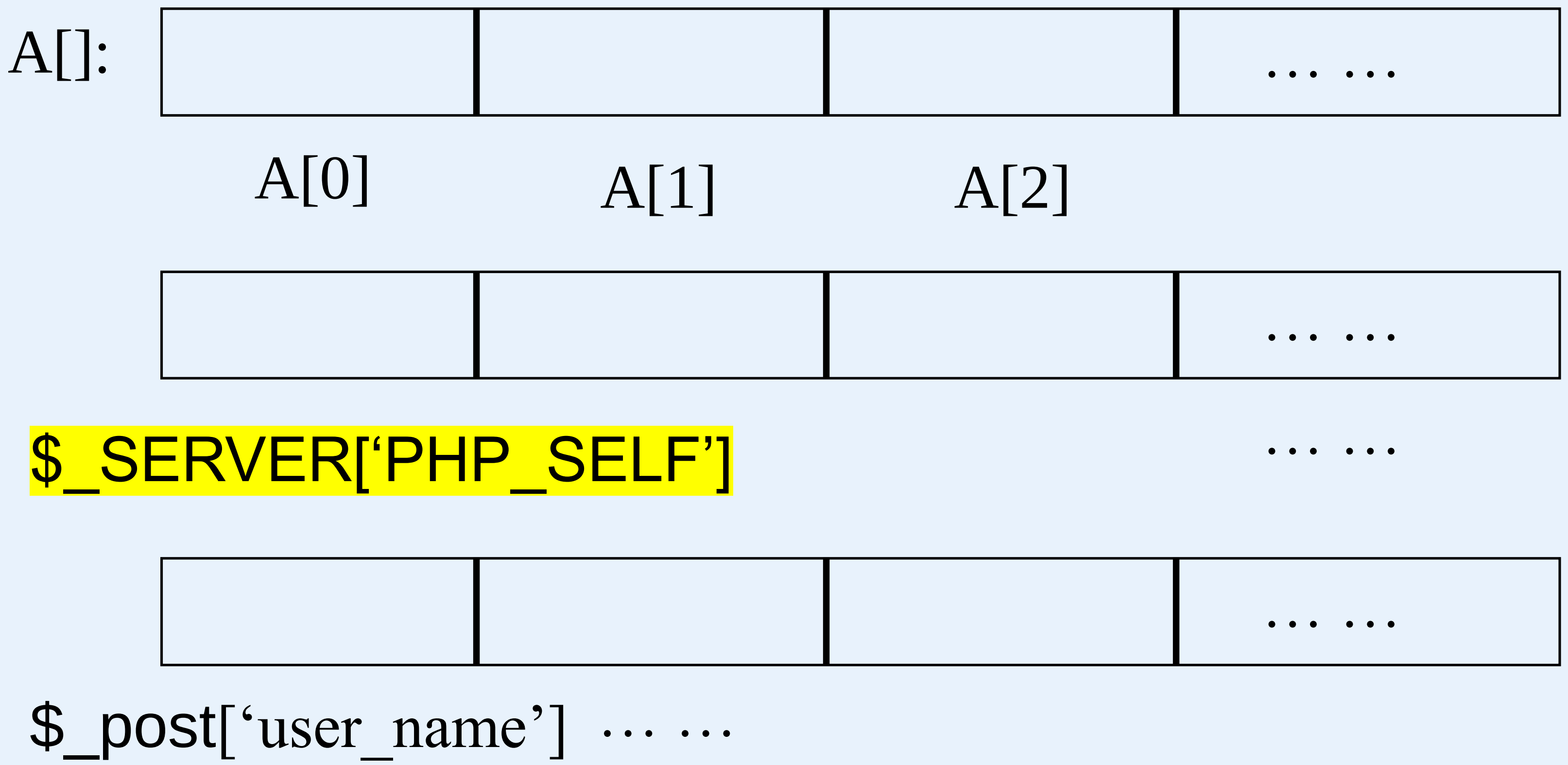
```
<!DOCTYPE html>
  <html>
    <body>
      <h1>My first PHP page</h1>
          <FORM method="post" action="$_SERVER['PHP_SELF']">
          Enter your name: <input type="text" name="user_name">
          <BR/>
          <INPUT type="submit" value="SUBMIT NAME">
          </FORM>
    </body>
</html>
```

Web Databases

```
<!DOCTYPE html>
  <html>
    <body>
      <h1>My first PHP page</h1>
          welcome, John Smith
     </body>
</html>
```

Web Databases

- Arrays in PHP are dynamic with no fixed number of elements.
- They can be indexed by numbers, or strings.



Web Databases

- When the web page at

  http://www.myserver.com/examples/greeting.php

  is first opened, $_post["user_name"] is *empty*. Then, the if-condition will evaluate to *false*.
- In this case, the else-part will be executed, by which a long text in an HTML will be created.

  All text between an opening

  <<<_HTML_

  and a closing

  HTML;

  will be put into the HTML file as is.

  In addition, the closing _HTML_; must be alone on a single line.

Web Databases

- This HTML file will be sent to the client to create a form shown before. It will be created to collect inputs from the user.
    - method = “post” indicates that the input will be sent to $_post array through a form.
    - Another way to send the input is the “get” method, by which the input of a user is sent to $_get array as a string associated with the corresponding URL address.
    - action = $_SERVER[‘PHP_SELF’] indicates that after the server has received the input, it will react according the value of $_SERVER[‘PHP_SELF’].
    - This value is in fact the path name of the PHP file currently being executed on the server. Then, the file will be executed once again.

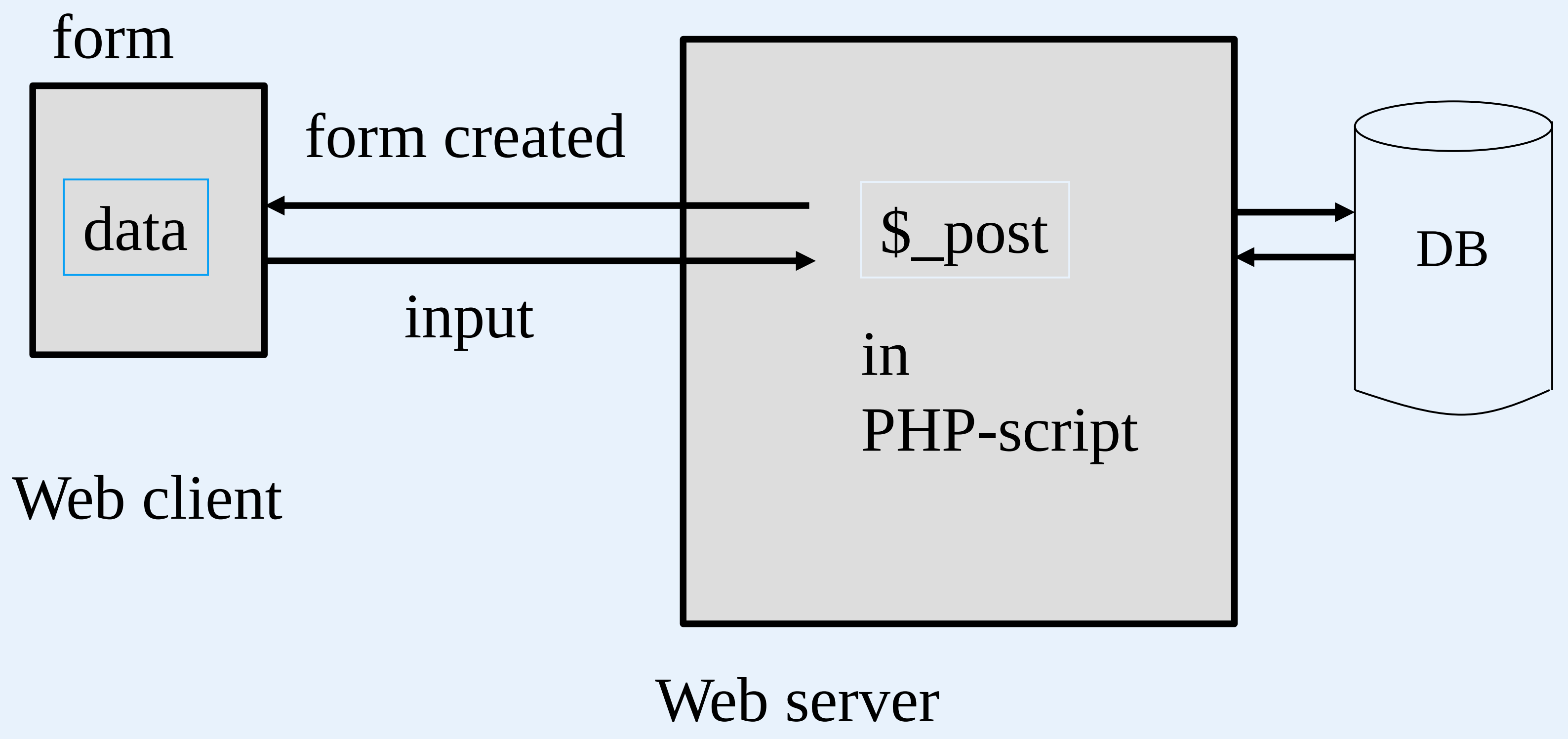
Web Databases

- Once the user types the name *John Smith* in the text box of the form and clicks on *the SUBMIT NAME* button, the PHP script will be reprocessed.
  - This time, $_post['user_name'] will include the string "John Smith" and
  - the if-condition is satisfied.
  - In this case, another HTML file will be constructed, which contains

    Welcome, John Smith.

post:

http://www.myserver.com/examples/greeting.php

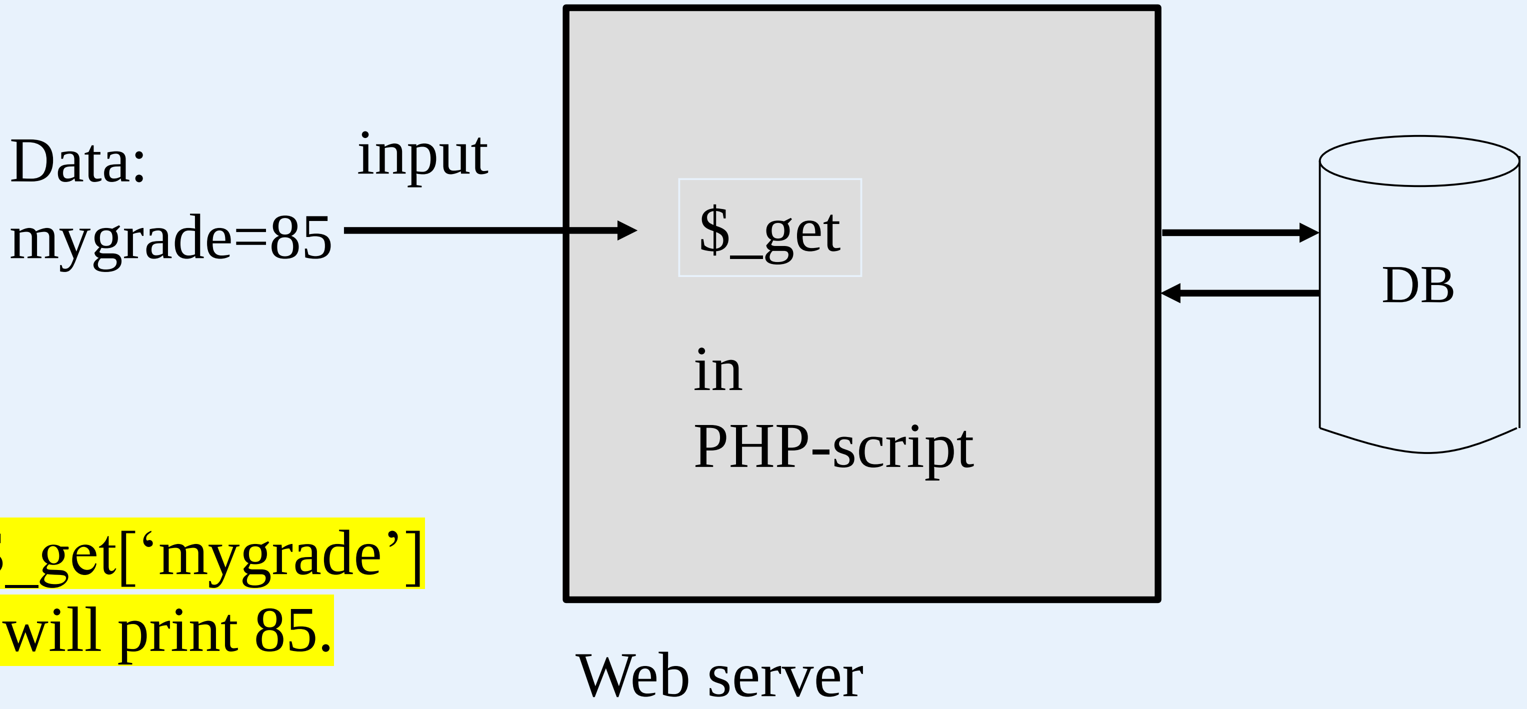


echo $_GET['myget']; // this will print "value"
echo $_GET['myget']; // this will print "value"

Web Databases

get:

http://www.myserver.com/examples/another.php?mygrade=85



echo $_get['mygrade']
//This will print 85.

Web Databases

HTTP: Hypertext Transfer Protocol

HTML: Hypertext Markup Language

URL: Uniform Resource Locator

Example: http://www.fredsco.com → 150.188.003.001

IP address

protocol

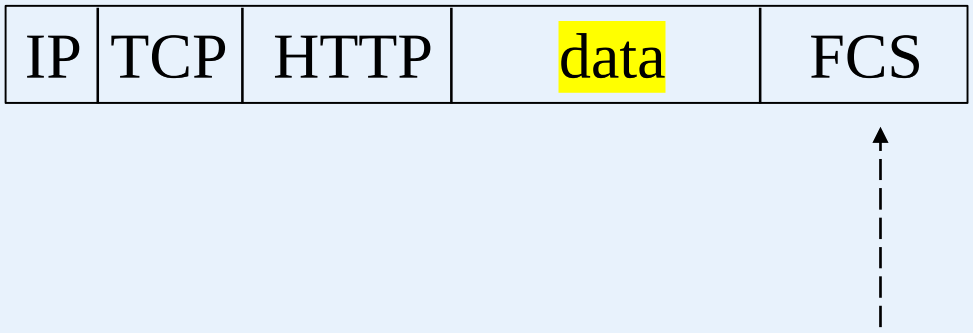
Computer name
(named network address)

To reach a computer (with a name address), the name address should first be replaced by its IP address by visiting a local DNS server.

DNS- domain name system

About data transmission (between web client and server) over the network:

TCP/IP package:



frame check sequence
(4 bytes, used to find error bits)

Web Databases

- **Connecting to a database**

```
require ‘DB.php’
$d = DB::connect{‘mysqli://acct1:pass12@www.host.com/db1’};
if (DB:isError($d)){die(“cannot connect …”, $d->getMessage());}
    …
$q = $d->query(“CREATE TABLE EMPLOYEE
    (       Emp_id INT,
            Name VARCHAR(15),
            Job VARCHAR(10),
            Dno INT)”   );
if (DB:isError($q)) {die(“table creation not successful …”, $d->getMessage());
    …
$d ->setErrorHandling( );
    …        ◄- - - - - - - - - - - - - - - - - - - - - - - - -
$eid = $d->nextID(‘EMPLOYEE’);
$q = $d->query(“INSERT INTO EMPLOYEE VALUES
    ($eid,$_post[‘emp_name’], $_post[‘emp_job’], $_post[‘emp_dno’])”);
```

A form should be displayed here to receive input data.

Web Databases

- First, load the DB.php (the PEAR DB module)
- Using the DB library function to establish the connection to a DB:

  DB::connect(<DB name>)

  - DB name:

    <DBMS software>://<user account>:<password>@<DB server>

  - DBMS software package that are accessible are:

    MySQL, Oracle, SQLite,

    Microsoft SQL Server , Mini SQL, Informix, Sybase,

    any ODBC compliant system

Web Databases

- $d -> query(<DB command>)

  DB command

  - DDL statement
  - DML statement
  - SQL statement

- We assume that the user entered valid values in the input parameters called emp_name, emp_job, and emp_dno. These would be accessible via the PHP array $_post as discussed before.

  $_post['emp_name']

  $_post['emp_job']

  $_post['emp_dno']

Web Databases

- **Safe way to execute a DB command**: use of placeholder (specified by the ? Symbol in a statement)

Example:

```
$eid = $d->nextID(‘EMPLOYEE’);
$q = $d->querry(“INSERT INTO EMPLOYEE VALUES (?, ?, ?, ?)”,
array($eid,$_post[‘emp_name’],$_post[‘emp_job’],$_post[‘emp_dno’]));
```

Web Databases

- **Retrieval queries from database tables**

```
require ‘DB.php’
$d = DB:connect{‘mysqli://acct1;pass12@www.host.com/db1’};
if (DB:isError($d)){die(“cannot connect …”, $d->getMessage());}
    …
$q = $d->query(“SELECT Name FROM EMPLOYEE WHERE
    Job = ? AND Dno = ?”,
array($_post[‘emp_job’], $_post[‘emp_dno’]) );
print “employee in dept $_post[‘emp_dno’] whose job is
    $_post[‘emp_job’]. \n”;
while ($r = $q->fetchRow())) {
    print “employee $r[0] \n”;
}
```

What is Node.js?

- Node.js is a script language
- Node.js is an open source server environment
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the serve

Web Databases

- **Web server language (script language): Node.js**

  - A common task for a web server can be to open a file on the server and return the content to the client.
  - Here is how Node.js handles a file request:

    1. Sends the task to the computer's file system.
    2. Ready to handle the next request.
    3. When the file system has opened and read the file, the server returns the content to the client.

  - Node.js does not wait for the response from the file system, and simply continues with the next request.
  - Node.js runs single-threaded, non-blocking, asynchronous programming, which is very memory efficient.

What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data (user inputs)
- Node.js can add, delete, modify data in databases (in DB server)

What is a Node.js File?

- Node.js files contain programs that will be executed on certain events. A typical event is someone trying to access a port (a number assigned to a communication process) on the server.
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

- **Download and install Node.js**

  The official Node.js website has installation instructions for Node.js: http://nodejs.org

  Getting Started

  - Once you have downloaded and installed Node.js on your computer, let's try to display "Hello World" in a web browser.
  - Create a Node.js file named "myfirst.js", and add the following code:

    myfirst.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
 res.end('Hello World!');
}).listen(8080);
```

myfirst.js

middleware function

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World!’); //write a response and then
  res.end( );                          //end the response
}).listen(8080);
```

a port number

Save the file on your computer:

C:\Users\Your Name\myfirst.js

The code tells the computer to write "Hello World!" if anyone (e.g. a web browser) tries to access your computer on port 8080.

- **Command Line Interface**

  -Node.js files must be initiated in the "Command Line Interface" program of your computer.

  -Navigate to the folder that contains the file "myfirst.js", the command line interface window should look something like this:

```
C:\Users\Your Name>_
```

```
C:\Users\Your Name>node myfirst.js
```

- **Execution of myfirst.js**

  - Now, your computer works as a server!
  - If anyone tries to access your computer on port 8080, they will get a "Hello World!" message in return!
  - Start your internet browser, and type in address:

    http://localhost:8080

- Built-in Modules

  - Node.js has a set of built-in modules which you can use without any further installation.
  - Look at Built-in Modules Reference for a complete list of modules.

- Include Modules

  To include a module, use the require() function with the name of the module:

```
var http = require('http');
```

Now your application has access to the HTTP module, and is able to create a server.

- Create your own modules

  - Create a module that returns the current date and time:

```
exports.myDateTime = function ( ) {
    return Date( );
};
```

- Use the exports keyword to make properties and methods available outside the module file.
- Save the code above in a file called

  "myfirstmodule.js"

- Include Your Own Module
  - Now you can include and use the module in any of your Node.js files.

Use the module "myfirstmodule" in a Node.js file:

```
var http = require('http');
var dt = require('./myfirstmodule');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

- Add an HTTP Header
  - If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

  - The first argument of the res.writeHead() method is the status code, 200 means that all is OK, the second argument is an object containing the response header.

- Read the Query String
  - The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).
  - This object has a property called "url" which holds the part of the url that comes after the domain name: demo_http_url.js

```
  var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end( );
}).listen(8080);
```

http://localhost:8080/summer
will produce this result:
/summer

Web Databases

- MySQL databases in a web server

  - You can download a free MySQL database at http://www.mysql.com/downloads/

  - Install MySQL Driver

- Once you have MySQL up and running on your computer, you can access it by using Node.js.

- To access a MySQL database with Node.js, you need a MySQL driver.

- Install MySQL from nmp.

Web Databases

- To download and install the "mysql" module, open the Command Terminal and execute the following:

```
C:\Users\Your Name>npm install mysql
```

npm - a package manager for installing Node.js packages.

- Create Connection

demo_db_connection.js

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

```
C:\Users\Your Name>node demo_db_connection.js
```

- Creating a Database
  - Create a database named "mydb”

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("CREATE DATABASE mydb", function (err,
result) {if (err) throw err;
    console.log("Database created"); }); });
```

Save the code above in a file called "demo_create_db.js“
C:\Users\Your Name>node demo_create_db.js

- Creating a table
  - Create a table named “customers”

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost", user: "yourusername ",
  password: "yourpassword", database: "mydb"});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (name
  VARCHAR(255), address VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");});
```

Web Databases

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address)
            VALUES ('Company Inc', 'Highway 37')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
}).listen(8080);
```

Web Databases

- Query a Database
  - Use SQL statements to read from (or write to) a MySQL database

```
… …
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
 database: “mydb”
 var sql = “select * from customers where   name = 'David'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

Web Databases

How to construct a form?

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<form action="fileupload" method="post"
   enctype="multipart/form-data">');
  res.write('<input type="file" name="filetoupload"><br>');
  res.write('<input type="submit">');
  res.write('</form>');
  return res.end( );
}).listen(8080);
```