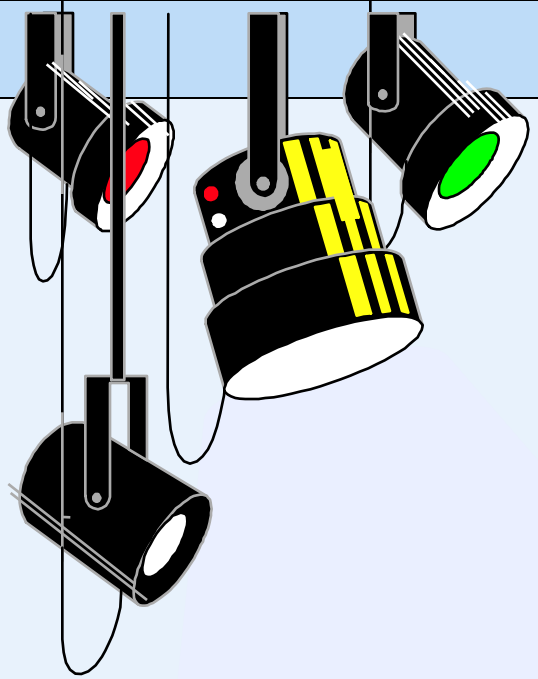
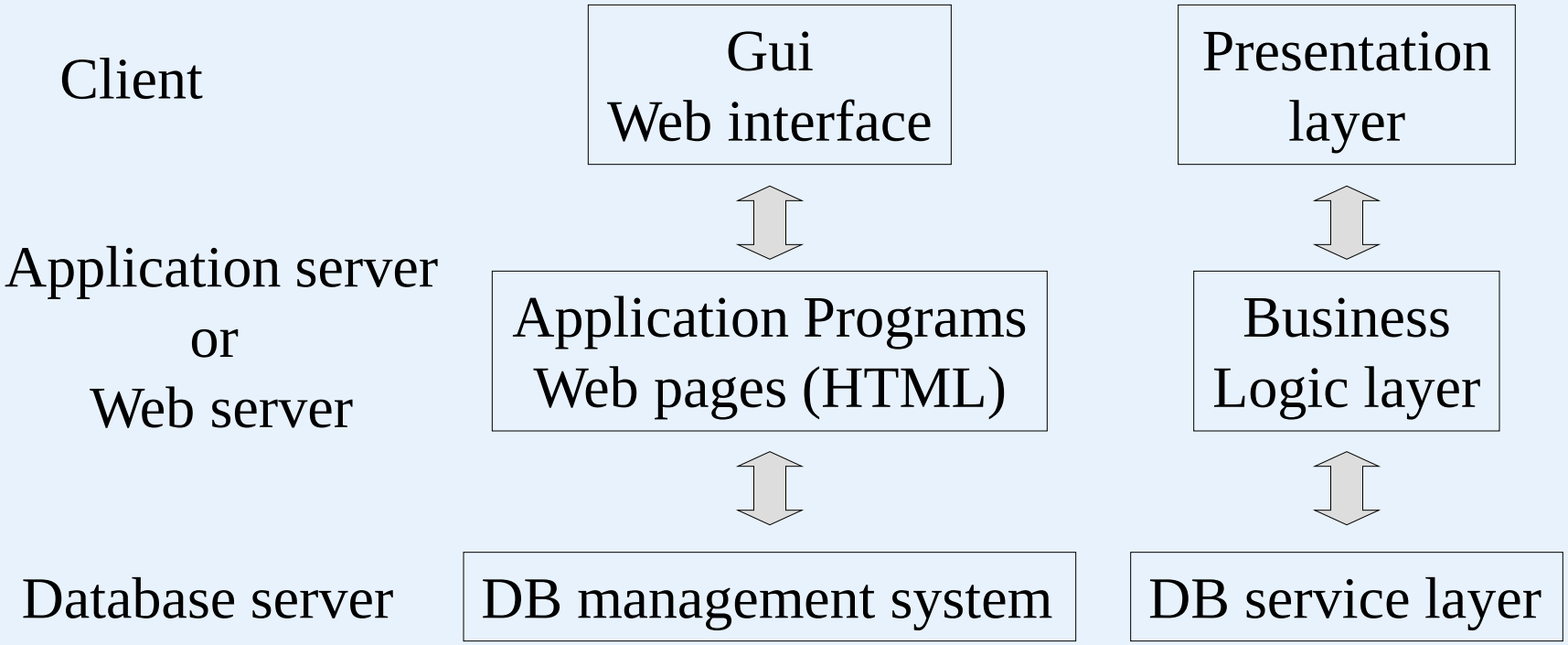


Web Databases

- What is a web database
- Web database system architecture
- Web programming language: PHP and Node.js

Web Databases

- What is a web database
  - A database accessed from the Internet
  - Three-tier architecture:



- **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 are executed on client computers.)

  - PHP 5 and later can work with a MySQl database using:

    - MySQLi extension (the ‘i’ stands for improved)
    - PDO (PHP Data Objects)

- **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 are executed on client computers.)

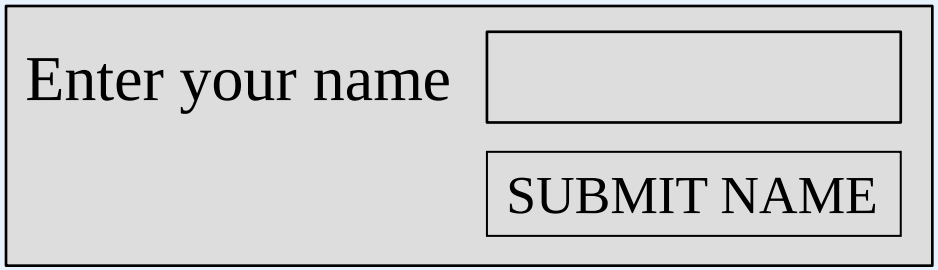
  - PHP 5 and later can work with a MySQl database using:

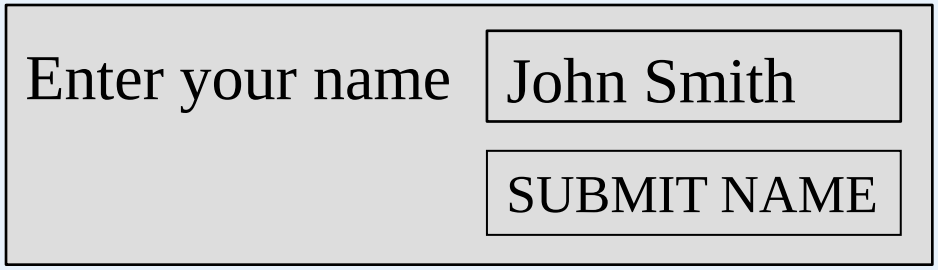
    - MySQLi extension (the ‘i’ stands for improved)
    - PDO (PHP Data Objects)

- **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” section=“$_SERVER[‘PHP_SELF’]”>
            Enter your name: <input type=“text” name=“user_name”>
            <BR/>
            <INPUT type=“submit” value=“SUBMIT NAME”>
            </FORM> _HTML_;} ?>
```

Web Databases





- **Connecting to a database**

```
require ‘DB.php’
$d = DB:connect{‘mysqli;//acct1;pass12@www.host.com/db1’};
if (DB:isError($d)){die(“canno connect …”, $d->getMessage());}
    …
$q = $d->query(“CREATE TABLE EMPLOYEE
    (Emp_id INT,
    Name VARCHAR(15),
    Job VARCHAR(10),
    Dno INT)”);
        print(“Welcome, “);
        print($_post[‘user_name’]);
if (DB:isError($q)) {… …}
$eid = $d->nextID(‘EMPLOYEE’);
$q = 4d->querry(“INSERT EMPLOYEE VALUES
    ($eid,$_post[‘emp_name’],$_post[‘emp_job’],$_post[‘emp_dno’])”);
```

- **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”;
}
```

- **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);
```

Save the file on your computer: C:\Users*Your Name*\myfirst.js

myfirst.js

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

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. This tutorial will use the "mysql" module, downloaded from NPM.
- To download and install the "mysql" module, open the Command Terminal and execute the following:

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

Here npm is package manager for 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

- 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!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

- 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