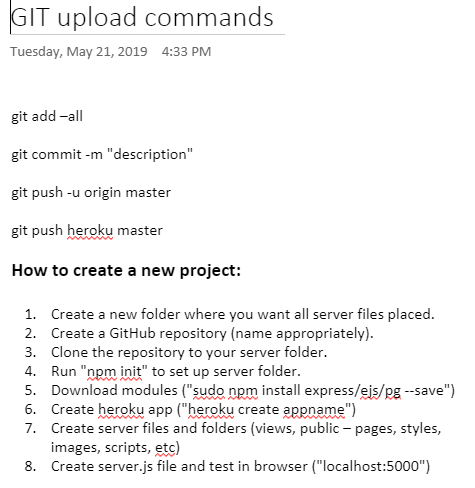


Shopping Cart:

<https://stackoverflow.com/questions/41973682/create-a-one-page-shopping-cart-using-php-session-and-no-sql>

**Start Server:**

php –S localhost:8000

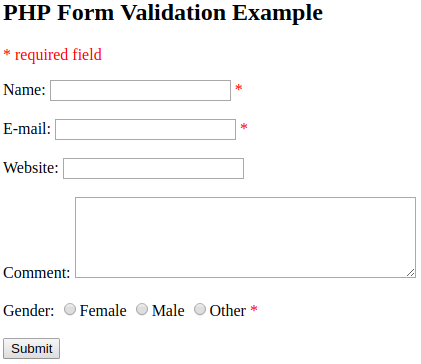




Name: <input type="text" name="name">   
E-mail: <input type="text" name="email">   
Website: <input type="text" name="website">   
Comment: <textarea name="comment" rows="5" cols="40"></textarea>

Gender:   
<input type="radio" name="gender" value="female">Female   
<input type="radio" name="gender" value="male">Male   
<input type="radio" name="gender" value="other">Other

Output:



The validation rules for the form above are as follows:

|  |  |
| --- | --- |
| **Field** | **Validation Rules** |
| Name | Required. + Must only contain letters and whitespace |
| E-mail | Required. + Must contain a valid email address (with @ and .) |
| Website | Optional. If present, it must contain a valid URL |
| Comment | Optional. Multi-line input field (textarea) |
| Gender | Required. Must select one |

The above code would user the following HTML/PHP:

<form method="post" action=

"<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

A special function that replaces certain HTML characters with ASCII code to prevent cross-site scripting (XSS) attacks. Always use this command to pass your variables through:

htmlspecialchars()

Read more about this [HERE](https://www.w3schools.com/php/php_form_validation.asp). More security codes include

trim() and stripslashes()

You should take all 3 of these functions and put them in another function that performs all of the checks on each variable parameter you run through it. For the above, it would [look like this](https://www.w3schools.com/php/showphp.asp?filename=demo_form_validation_escapechar):

<?php   
// define variables and set to empty values   
$name = $email = $gender = $comment = $website = "";   
   
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {   
  $name = test\_input($\_POST["name"]);   
  $email = test\_input($\_POST["email"]);   
  $website = test\_input($\_POST["website"]);   
  $comment = test\_input($\_POST["comment"]);   
  $gender = test\_input($\_POST["gender"]);   
}   
   
function test\_input($data) {   
  $data = trim($data);   
  $data = stripslashes($data);   
  $data = htmlspecialchars($data);   
  return $data;   
}   
?>

**Check input field contents:**

This function checks to assure that input fields only contain approved letters/characters and whitespace:

$name = test\_input($\_POST["name"]);

if (!preg\_match("/^[a-zA-Z ]\*$/",$name)) {

$nameErr = "Only letters and white space allowed";

}

The preg\_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

**Check email format:**

$email = test\_input($\_POST["email"]);

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

$emailErr = "Invalid email format";

}

The filter\_var() function validates email addresses.

**Validate URLs:**

$website = test\_input($\_POST["website"]);

if (!preg\_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~\_|!:,.;]\*[-a-z0-9+&@#\/%=~\_|]/i",$website)) {

$websiteErr = "Invalid URL";

}

Put all of the above together and it will look like this: <https://www.w3schools.com/php/showphp.asp?filename=demo_form_validation_special>

Here is a lecture on PHP security and filtering:

<https://phptherightway.com/#data_filtering>

**Check for empty fields:**

The empty() function checks to see if a field is empty and returns a boolean.

You can use this to display error messages if a field should be required.

**Retain the values in the form input fields after submission:**

See: <https://www.w3schools.com/php/php_form_complete.asp>

PHP can load, edit, and save files like any other language. Be careful when doing this because PHP is error prone.

**Read a file and display its contents:**

readfile() has a limited number of options and can be used for basic actions.

<?php

echo readfile("webdictionary.txt");

?>

**PHP Open File**

This gives you more options than readfile()

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

echo fread($myfile,filesize("webdictionary.txt"));

fclose($myfile);

?>

The**fopen()** function has 2 parameters:

1.) The name of the file to be opened

2.) Which mode the file should be opened in.

The options for parameter 2 are:



The fread() function reads from an open file. This function also has 2 parameters:

1.) the file to read from

2.) the maximum number of bytes to read

On the 2nd parameter, an easy way to have it read the entire file is as follows:

fread($myfile,filesize("webdictionary.txt"));

**PHP Close File**

As with C++, you should close files after you open them or they will hog system resources.

<?php

$myfile = fopen("webdictionary.txt", "r");

// some code to be executed....

fclose($myfile);

?>

**PHP Read Single Line**

Use the fgets() function:

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

echo fgets($myfile);

fclose($myfile);

?>

This will output only the first line of a php file.

The below function will read the entire file line by line:

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

// Output one line until end-of-file

while(!feof($myfile)) {

echo fgets($myfile) . "<br>";

}

fclose($myfile);

?>

Each line is displayed when retrieved, then the line break drops the display to the next line.

**Read a Single Character**

The following will read a file character by character until the end of the file:

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

// Output one character until end-of-file

while(!feof($myfile)) {

echo fgetc($myfile);

}

fclose($myfile);

?>

Unlike the previous example that breaks after each line is read, this will continue a consistent stream of text, not separating any of the display by breaks.

**Create a new file**

The fopen() function creates a new file if there is not already a file with that name.

**Write to a file**

The fwrite() function is used to write to a file:

<?php

$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");

$txt = "John Doe\n";

fwrite($myfile, $txt);

$txt = "Jane Doe\n";

fwrite($myfile, $txt);

fclose($myfile);

?>

In the above example, a file is opened, written to, and written to again appending new information. The above example will also overwrite an existing file with the new information, erasing all previous data. Be careful how you use this.

A cookie is a small file that is created when a user visits a website that stores information about that user. Cookies can be set to expire. Cookies can be set to be made available for the entire site or just a directory you select.

**Creating a Cookie**

Use setcookie() function to create a cookie:

setcookie(name, value, expire, path, domain, secure, http only);

<!DOCTYPE html>

<?php

$cookie\_name = "user";

$cookie\_value = "John Doe";

setcookie($cookie\_name, $cookie\_value, time() + (86400 \* 30), "/"); // 86400 = 1 day

?>

<html>

<body>

<?php

if(!isset($\_COOKIE[$cookie\_name])) {

     echo "Cookie named '" . $cookie\_name . "' is not set!";

} else {

     echo "Cookie '" . $cookie\_name . "' is set!<br>";

     echo "Value is: " . $\_COOKIE[$cookie\_name];

}

?>

<p><strong>Note:</strong> You might have to reload the page to see the value of the cookie.</p>

 </body>

</html>

Notice in the above example that a cookie needs to be declared **outside (before) of the <html> tags.** Usually cookies are only set upon 2nd visits. Refresh your web page F5 when testing.

**Modify a Cookie**

Modifying a cookie is like changing a variable. Just re-declare it and everything will be overwritten.

**Delete a Cookie**

Just set a cookie with the expiration date in the past.

**Check if Cookies are Enabled**

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the setcookie() function, then count the $\_COOKIE array variable:

<?php

setcookie("test\_cookie", "test", time() + 3600, '/');

?>

<html>

<body>

<?php

if(count($\_COOKIE) > 0) {

    echo "Cookies are enabled.";

} else {

    echo "Cookies are disabled.";

}

?>

</body>

</html>

**Google Fonts:**

1. Add to html head: <script src=https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js" type="text/javascript"></script>
2. Visit <https://fonts.google.com/>
3. Search/select a font and click the red + button.
4. Copy and paste the html/CSS code from the selected family window.

**Custom Fonts:**

1. Find a custom font online and download the .ttf file.
2. Upload the .ttf file to a webfont  kit generator like <https://www.fontsquirrel.com/tools/webfont-generator>, and download the webfont kit file.
3. Create a folder in your website root directory called ".fonts" (must have the dot for heroku).
4. Copy and paste the webfont kit files into the ".fonts" folder.
5. Copy and paste the CSS code from the "stylesheet.css" file in the webfont kit into your main CSS.
6. Modify the CSS urls/directories from the default to your .fonts directory location.

Node.js allows you to execute a .js (javascript) file in the terminal without having to call it with HTML code. Just run with

node filename.js

**Set up the fs module** from the Node core library:

var fs = require('fs')

**Read a file:**

\*Note: All synchronous (or blocking) filesystem methods in the fs module end with

  'Sync'.

fs.readFileSync('/path/filename');+

Documentation for fs.library:

[file:///usr/local/lib/node\_modules/learnyounode/node\_apidoc/fs.html](file:///\\usr\local\lib\node_modules\learnyounode\node_apidoc\fs.html)

**Set up and configure a new database on your host server** open your terminal to the repository location and enter the following commands:

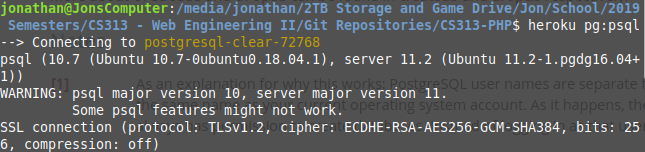
heroku addons: create heroku-postgresql:hobby-dev

heroku config –s

**Connect to your database on your server**

Open your terminal to the repository file location and enter:

heroku pg:psql



**Create / Delete a Database**

createdb databaseName

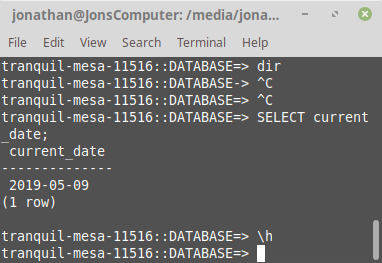
dropdb databaseName      <-- Warning! No undos!

**Open a pre-existing database:**

psql databaseName

\h will show you a list of commands

\q to get out of the list



After installation...

**Start PostgreSQL:**

Sudo -i -u postgres

psql

**Backout:**

\q

exit

**View a list of all databases:**

\l

**Select a database to edit:**

\c databasename

**View a list of all tables on a database:**

SELECT \* FROM pg\_catalog.pg\_tables;

WHERE schemaname = 'public';

Displays stuff in the terminal:

netstat -tpln

**Steps I took to get it working on a local environment:**

Install the php-pgsql package after PHP and PostgreSQL were already installed:

sudo apt-get install php-pgsql

Set up a superuser and password- log into psql and enter the following command in the database:

CREATE USER yourname WITH SUPERUSER PASSWORD 'yourpassword';

**Current User name:**postgres

**Current password:** password

! The name of your database file is cap-sensitive in PHP.

foreach ($db->query("SELECT \* FROM players WHERE id='7' or id='9';") as $row)

            {

                $pn = $row['playername'];

                $fn = $row['realfirstname'];

                $ln = $row['reallastname'];

                $bio = $row['bio'];

                $age = $row['age'];

                $num = $row['playernumber'];

                echo "<p>Player Name: </p><e>$pn</e>" .

                   "<p>First Name: </p><e>$fn</e>" .

                   "<p>Last Name: </p><e>$ln</e>" .

                   "<p>Catch Phrase: </p><e>$bio</e>" .

                   "<p>Age: </p><e>$age</e>" .

                   "<p>Number: </p><e>$num</e><br>" .

                   "<p class='line'></p>";

            }

<https://www.postgresql.org/docs/9.6/tutorial-sql-intro.html>

<https://www.w3schools.com/sql/default.asp>

**Quick Reference**

\*Note – SQL is not cap-sensitive, but all caps are used as good style.

**Create a Table**

CREATE TABLE tableName (param1, param2, param3,…);

**Delete Table**

DROP TABLE tableName;

**Insert Row Into Table**

INSERT INTO tableName VALUES(value1, value2, value3, …);

The above will insert the values in the order of the columns going left to right. If you want to do something special, you need to specify the parameters as shown below. This is generally considered better etiquette rather than relying on column order:

INSERT INTO tablename(param1, param2, param27)

VALUES(1stValue, 2ndValue, 27thValue);

**Select Item From Table**

Individual Column:

SELECT columnName FROM tableName;

**All Columns:**

SELECT \* FROM tableName;

**Multiple Columns:**

SELECT columnName1, columnName2, …

Multiple Rows:

SELECT \* FROM tableName

WHERE id = '3' or id='7' ...

**Select Specific Item Matching Another:**

SELECT columnName FROM tableName

WHERE columnName = 'itemName';

**Modify Items in Table**

update tableName

set columnName = 'itemName'

where columnName = 'itemName';

Be very careful when updating things as it is easy to accidentally overwrite data!

**Delete Item in Table**

DELETE FROM tableName WHERE columnName = 'itemName';

DELETE FROM tableName;

This will wipe your whole table. Be careful!

**See**<https://code.tutsplus.com/articles/sql-for-beginners-part-3-database-relationships--net-8561>

**Displays all contents in multiple tables together:**

The statement

**SELECT \* FROM table1 JOIN table2;**

Is the same as

**SELECT \* FROM table1, table2;**

Display rows where 2+ tables have a matching column name and the values into those columns match. Usually this is where a foreign key column name matches the primary key column name in another table. It then displays all matching values.

**SELECT \* FROM table1 NATURAL JOIN table2;**

Similar to a NATURAL JOIN is this:

**SELECT \* FROM table1 JOIN table2**

**WHERE table1.table1\_ID = table2.table1\_ID;**

For conditionals, you can use either

**WHERE condition;**

**or**

**ON (condition);**

**or**

**USING (condition); <-- This one eliminates redundancies**

LEFT JOINS will display everything from the first table. If there is no match in the 2nd table, it will show a NULL value for those fields.

**Start PHP Local Host Server:**

**Open website *root location* in terminal.**

**php –S localhost:8000**

**Enter:**[**http://localhost:8000**](http://localhost:8000/)**in web browser.**

**Start PostgreSQL Local Database:**

**sudo -i -u postgres (optional database name)**

**Psql**

**Username:**postgres

**Password:**password

**Start PostgreSQL on Heroku:**

**heroku pg:psql**

**View all databases:**

**\l**

**View all tables**

SELECT \* FROM pg\_catalog.pg\_tables

WHERE schemaname = 'public';

Or

\dt   from inside database

**PostgresSQL Server Startup:**

sudo apt-get update   
sudo apt-get install postgresql-10.4 

By default, the postgres user has no password and can hence only connect if ran by the postgres system user. The following command will assign it:

sudo -u postgres psql -c "ALTER USER postgres PASSWORD 'postgres';"   
sudo -u postgres psql -c "CREATE DATABASE testdb;" 

**Start the PostgreSQL server**

sudo service postgresql start 

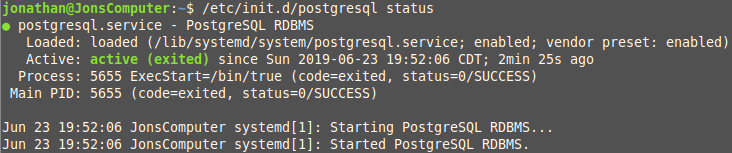
**Stop the PostgreSQL server:**

sudo service postgresql stop

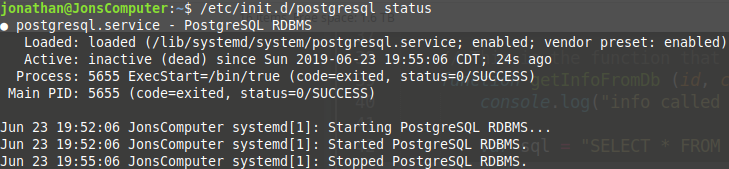
**Check status:**

/etc/init.d/postgresql status

**Active:**



**Inactive:**



**To log in:**

psql (user\_name)  -h 127.0.0.1 -d (db\_name)