Make sure you can log into the server via an FTP client like Filezilla!

Download “PHP File Loading files.zip” from the class website and unzip into a subdirectory called *files*!

1. Write the bare XHTML markup

Start with just a plain file with the following markup:

```
<!DOCTYPE html>
<html>
<head>
     <title>MyProject</title>
</head>

<body>

</body>
</html>
```

2. Loading a Plaintext File

Start by opening a file as a **read**, and checking to make sure it worked right:

```
$filepath = "files/plaintext.txt";

$fileHandler = fopen( $filepath, "r" );
if ( $fileHandler == false )
{
     print_r( error_get_last() );  // Error message (PHP5)
     exit();   // Stop script
}
```

Next, all we need to do is call **fgets** to store the file's contents in a variable:

```
$contents = fgets( $fileHandler );

fclose( $fileHandler );
```

Then, make sure to output the information

```
<p>File contents:</p>

<p><?=$contents?></p>
```

3. Loading a JSON File

Reading a JSON file is very similar – we have to load the file into $contents before we can use the **json_decode** function (or we could do it all at once).

Make sure to change the filename:

```
$filepath = "files/text.json";
```

and then add this line after the **fgets**:

```
$contents = fgets( $fileHandler );
$jsonContents = json_decode( $contents, true ); // add this
```

The **true** argument here is whether or not it should be an associative array. If this is set to false, the JSON will be read in as an object instead of an array.

And remember to use **print_r** to output:

```
<p>File contents:</p>

<pre>
     <? print_r( $jsonContents ); ?>
</pre>
```

4. Loading a CSV File

Reading the CSV file will be the most tricky, since you have to read in one row at a time into an array.

Keep the code using **fopen** and the error checking after that. But, then create a new array called $rows.

```
$rows = array();
```

We will use a **while** loop to get all the rows from the CSV file like this:

```
while ( $r = fgetcsv( $fileHandler ) )
{
     array_push( $rows, $r );
}
```

Make sure to close the $fileHandler once we're done.

Note that **array_push** has two arguments: The first one is the array to be added to, and the second one is the value of the new element.

Now, if we want to output all the data in each row, we can do:

```
<p>File contents:</p>

<?
foreach( $rows as $row ) {
     echo( "<p>" );
     print_r( $row );
     echo( "</p>" );
}
?>
```

5. Challenge Problem

Can you set up the CSV reading page to instead use a table? It should look like the top display, rather than the bottom:

File contents:

DAY	START TIME	END TIME
TUESDAY	5:30 pm	6:45 pm
THURSDAY	5:30 pm	6:45 pm
TUESDAY	3:00 pm	5:00 pm
THURSDAY	3:00 pm	5:00 pm

Array ([0] => TASK [1] => DAY [2] => START TIME [3] => END TIME)

Array ([0] => CS 490 WD [1] => TUESDAY [2] => 5:30 pm [3] => 6:45 pm)

Array ([0] => CS 490 WD [1] => THURSDAY [2] => 5:30 pm [3] => 6:45 pm)

Array ([0] => OFFICE [1] => TUESDAY [2] => 3:00 pm [3] => 5:00 pm)

Array ([0] => OFFICE [1] => THURSDAY [2] => 3:00 pm [3] => 5:00 pm)

Fill code for read-plaintext.php5:

```
<!DOCTYPE html>
<html>
<head>
     <title>MyProject</title>
</head>

<body>

     <?
     $filepath = "files/plaintext.txt";

     $fileHandler = fopen( $filepath, "r" );
     if ( $fileHandler == false )
     {
          print_r( error_get_last() );  // Error message (PHP5)
          exit();   // Stop script
     }

     $contents = fgets( $fileHandler );

     fclose( $fileHandler );
     ?>

     <p>File contents:</p>

     <p><?=$contents?></p>

</body>
</html>
```