**Difference between Zend session and Zend registry.**

**Zend\_Registry** is used to store objects/values for the current request. In short, anything that you commit to Registry in index.php can be accessed from other controllers/actions (because EVERY request is first routed to the index.php bootstrapper via the .htaccess file). Config parameters and db parameters are generally prepped for global use using the Zend\_Registry object.

**Zend\_Session\_Namespace** actually uses PHP sessions. Data stored using Zend\_Session can be accessed in different/all pages. So, if you want to create a variable named ‘UserRole’ in the /auth/login script and want it to be accessible in /auth/redirect, you would use Zend\_Session.

***For Example***

if you have a DB object stored in the registry, and when you are access a page say login page, this particular DB object will be accessible to any of the files used in that page (to construct), meaning global access (same keyword in PHP [Scope Variable](http://in2.php.net/manual/en/language.variables.scope.php#language.variables.scope.global)).

Whereas, Session scope means you can access anywhere on any page until you close your browser (until your session get’s destroyed).

**Zend\_Session** is for working with the session extension in PHP.This would be for tracking loggin in user(s), etc.

**Zend\_Registry** is used to store objects / resources / etc in at runtime. The idea is that you want to use maybe 1 config object, or 1 database throughout your entire application. So what you would do, when you create these objects, you would assign them to the registry.

**Benefits using composer**

With Composer we get the following advantages:

1. The dependencies required by the package we are pulling in are automatically taken care by Composer itself, leaving we free to focus on the programming instead of dependency management.
2. When the package we are using gets a new version, a simple composer update will do everything for us, without ever needing to do any file management manually.
3. With Composer we get a centralized autoload.php file which also be optimized with Composer. It loads everything we need and all we do is include one file. The optimized version is super fast! Simply do composer dump-autoload -o
4. We can use psr-4 namespaces to load a specific path on our application and have it be included in the autoloader file. Then we can simple use the namespace and it's available application wise!

**If else and switch case query in MySQL**

SELECT col1, col2, IF( action = 2 AND state = 0, 1, 0 ) AS state from tbl1;

CASE

WHEN condition\_1 THEN result\_1

WHEN condition\_2 THEN result\_2

…

ELSE result END

SELECT col1, col2, (case

when (action = 2 and state = 0) then 1 else 0 end) as state from tbl1;

**2nd highest salary + MySQL limit in detail**

SELECT DISTINCT(salary) FROM emp\_tbl ORDER BY salary DESC limit 2,1;

**Clone function in jQuery**

## Definition and Usage

The clone() method makes a copy of selected elements, including child nodes, text and attributes.

## Syntax

$(*selector*).clone(true|false)

### Example

Clone all <p> elements and insert them at the end of the <body> element:

$("button").click(function(){  
    $("p").clone().appendTo("body");  
});

**Strstr, substr and strpos in php**

Find the first occurrence of "world" inside "Hello world!" and return the rest of the string:

echo strstr("Hello world!","world");

**Output :** world!

Return "world" from the string:

echo substr("Hello world",6);

**Output** : world

Find the position of the first occurrence of "php" inside the string:

echo strpos("I love php, I love php too!","php");

**Output** : 7

**How to define constant in php?**

define("GREETING", "Welcome to W3Schools.com!");  
echo GREETING;