PHP continue …

Sessions

Sessions

- What it is?
  - An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.
  - A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

Sessions

- Where the file Stored:
  - The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**.
- What Happens
  - HP first creates a unique identifier for that particular session which is a random string of **32 hexadecimal** numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
  - A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
  - A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

Sessions

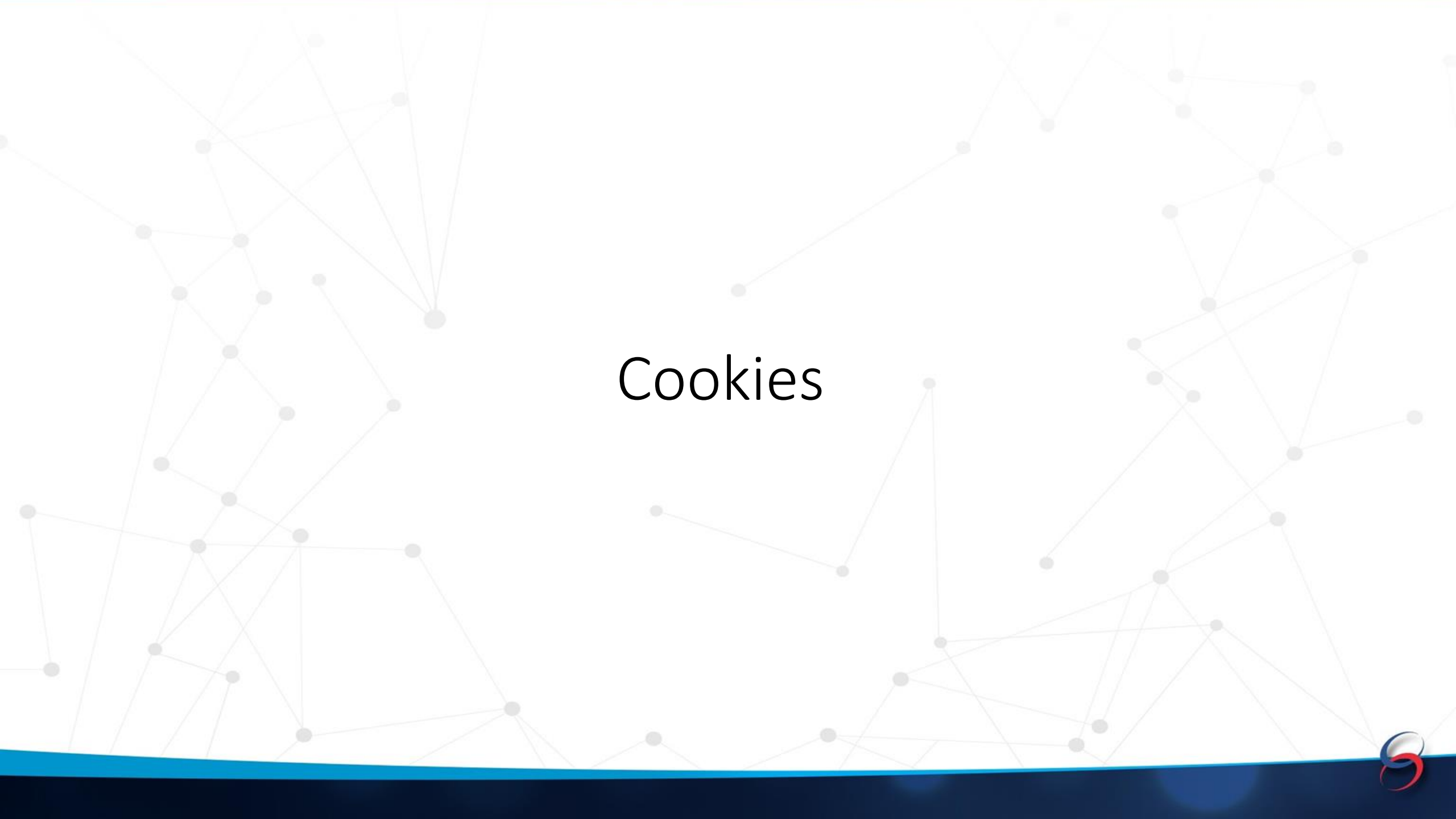
- Starting a session
  - session_start()
- Variables in sessions
  - $_SESSIONS
- Destroying a session
  - session_destroy()

Sessions(Example)

- Example:

```
<?php
session_start();
if( isset( $_SESSION['university'] ) ) {
      $_SESSION['counter']+= 1;
  }else {
        $_SESSION['counter']= 1;
  }
?>
```

Cookies



Cookies

- What it is?
  - Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.
- Use of Cookies
  - Remembering the end users data
  - Validating returning users

Cookies

- Determination of returned customer
  - Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
  - Browser stores this information on local machine for future use.
  - When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

`

Cookies

- Setting Cookies with PHP

Syntax:

- setcookie(name, value, expire, path, domain, security);

Here is the detail of all the arguments −

- **Name** − This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value** − This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** − This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** − This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** − This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** − This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

`

Cookies(Example)

- Example:

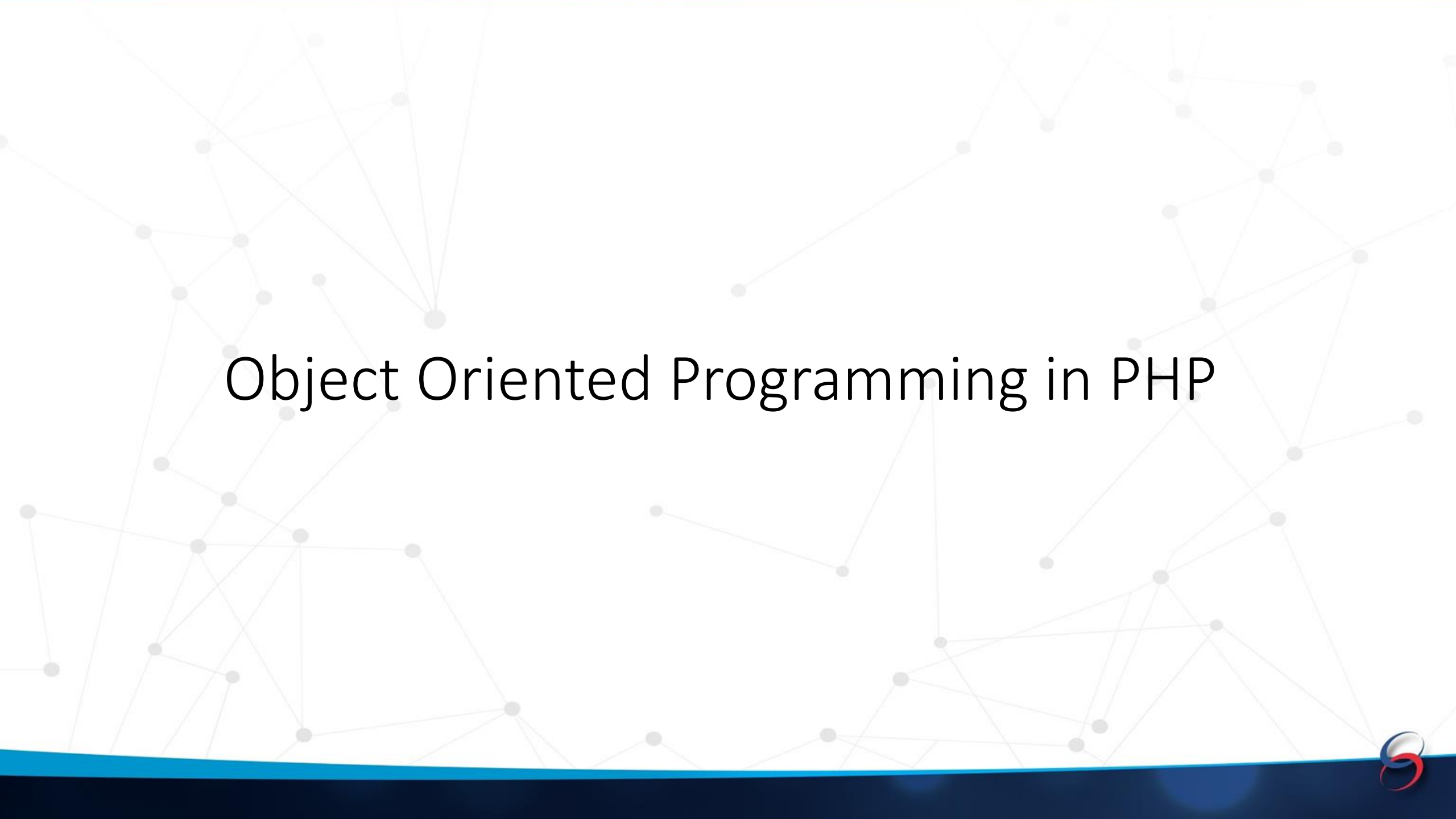
```
<?php
  setcookie("name", “Test", time()+3600, "/","", 0);
  setcookie("age", “66", time()+3600, "/", "",  0);
?>
<html>

  <head>
    <title>Setting Cookies with PHP</title>
  </head>

  <body>
    <?php echo "Set Cookies"?>
  </body>

</html>
```

Cookies(Example)

- Example(Accessing Cookies with PHP):

```
<head>
   <title>Accessing Cookies with PHP</title>
 </head>

 <body>

   <?php
     echo $_COOKIE["name"]. "<br />";

     /* is equivalent to */
     echo $HTTP_COOKIE_VARS["name"]. "<br />";

     echo $_COOKIE["age"] . "<br />";

     /* is equivalent to */
     echo $HTTP_COOKIE_VARS["name"] . "<br />";
   ?>

 </body>
</html>
```

Object Oriented Programming in PHP



Classes

- Definition

  This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.

  Example:

```
<?php
        class phpClass {
         var $var1;
        var $var2 = "constant string";

        function myfunc ($arg1, $arg2) {
                [..]
               }
                [..]
                }
?>
`
```

Classes

- Definition

  This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.

  Example:

```
<?php
        class phpClass {
         var $var1;
        var $var2 = "constant string";

        function myfunc ($arg1, $arg2) {
                [..]
               }
                [..]
                }
?>
`
```

Classes(Some terms)

- **Object** – An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable** – These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function** – These are the function defined inside a class and are used to access object data.
- **Inheritance** – When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class** – A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class** – A class that inherits from another class. This is also called a subclass or derived class.
- **Polymorphism** – This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it make take different number of arguments and can do different task.
- **Overloading** – a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **Data Abstraction** – Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation** – refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Constructor** – refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructor** – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

Examples of Class and Object

Class Definition

```
<?php
  class Books {
    /* Member variables */
    var $price;
    var $title;

    /* Member functions */
    function setPrice($par){
      $this->price = $par;
    }

    function getPrice(){
      echo $this->price ."<br/>";
    }

    function setTitle($par){
      $this->title = $par;
    }

    function getTitle(){
      echo $this->title ." <br/>";
    }
  }
?>
```

Creating Object

```
<?php
$physics = new Books;
$maths = new Books;
$chemistry = new Books;
//Calling Member Functions
$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );

$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );

$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

Output

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Constructor Functions

Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.

PHP provides a special function called __construct() to define a constructor. We can pass as many as arguments you like into the constructor function.

Example:

```
function __construct( $par1, $par2 ) {
  $this->title = $par1;
  $this->price = $par2;
}
```

Examples with Constructor Function

Class Definition

```
<?php
  class Books {
function __construct( $par1, $par2 ) {
  $this->title = $par1;
  $this->price = $par2;
}
    /* Member variables */
    var $price;
    var $title;

    /* Member functions */
    function setPrice($par){
      $this->price = $par;
    }

    function getPrice(){
      echo $this->price ."<br/>";
    }

    function setTitle($par){
      $this->title = $par;
    }

    function getTitle(){
      echo $this->title ." <br/>";
    }
  }
?>
```

Creating Object

```
<?php
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
?>
```

Output

Physics for High School
Advanced Chemistry
Algebra
10
15
7