Server requirements

- software / development stack
  - database : MySQL
  - web server : Apache2
  - language : PHP
  - platform : Linux + Windows + macOS
- stacks
  - LAMP: done
  - WAMP: done
  - MAMP: done

PHP

- stands PHP: Hypertext Preprocessor
- server side language used to develop dynamic website
  - dynamic
    - the code will get executed on the server
    - the php will get convered on the server to html
- programming language
- has syntax like C language
  - semicolon mandatory
  - case sensitive
- interpreted language
- the PHP code must be written in

variables

- the variable must be declared with $ sign

- prefer using camel case while declaring a variable

- e.g.

  ```php
  $num = 100;
  print("num = " . $num);
  ```

constant

- to declare a constant use function define()

- use upper case letters to declare a constant

- e.g.

```php
define("PI", 3.14);
print("PI = " . PI);
```

data types

- in PHP, the data type of a variable is inferred
- the data type will be implciitly assigned by PHP (by looking at the value)
- no explicit data types can be assigned
- types
  - **integer**
    - is used to represent the whole numbers
    - e.g.

    ```php
    // integer
    $num = 100;
    ```

  - **double**
    - is similar to float in C
    - e.g.

    ```php
    // double
    $salary = 5.6;
    ```

  - **string**
    - used to represent a string
    - can be declared with single or double quotes
    - e.g.

    ```php
    // string
    $firstName = "steve";
    $lastName = 'jobs';
    ```

  - **boolean**

- used to declare a variable with values true or false
- e.g.

```
// boolean
$canVote = true;
```

- **NULL**
  - represents nothing
  - e.g.

```
// null
$myVar = NULL;
```

- **object**
- **resource**
- **array**

functions

- named block of code which can be reused
- in PHP a function can be called with
  - same number of parameters the function is expecting
  - more number of parameters the function is expecting
- **but a function can not be called with less number of parameters**
- types
  - **parameterless function**
    - e.g.

```
// function declaration
function helloWorld() {
    print("inside helloWorld");
}

// function call
helloWorld();
```

  - **parameterized function**

    - e.g.

    ```
    // function declaration
    function add($p1, $p2) {
        $addition = $p1 + $p2;
        print("addition = $addition");
    }

    // function call
    add(10, 20);
    ```

  - **function with return value**
    - a function which returns a value
    - e.g.

    ```
    function add($p1, $p2) {
      return $p1 + $p2;
    }

    $addition = add(10, 40);

    // 50
    print($addtion);
    ```

predefined values

- NAN
- INF

array

- collection of values
- to create an array
  - call array function

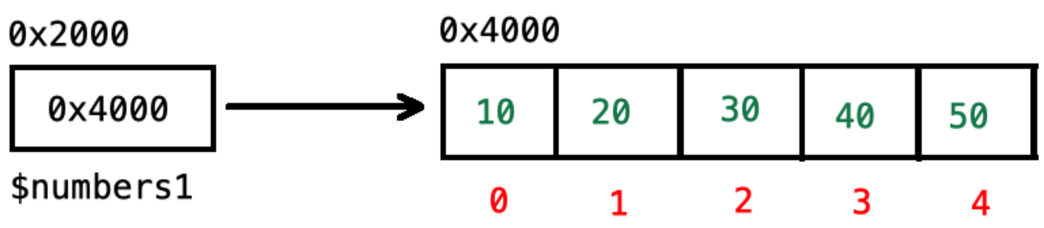
  ```
  $numbers = array();
  ```

  - use subscript

```
$numbers = [];
```

- types
  - indexed array
    - index position is managed automatically
    - index starts at 0 and get incremented till the last value in the array

```
$numbers1 = array(10, 20, 30, 40, 50);
```



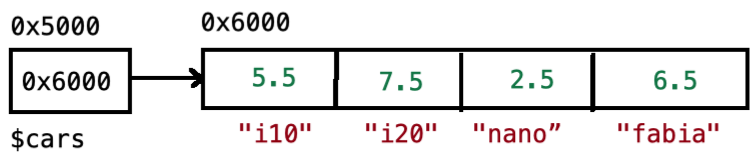
```
$numbers = [10, 20, 30, 40, 50];

// traditional for..loop
for ($index = 0; $index < count($numbers); $index++) {
  print("value at $index = " . $numbers[$index] . "<br>");
}

// for..each loop
foreach($numbers as $value) {
  print("value = $value <br>");
}
```

  - associative array
    - collection of associated values
    - the index position will be decided by the user
    - can have index position with any type (string, integer, double, boolean)
    - multiple values with same keys will have only the last value

```
$cars = [
   "i10" => 5.5,
   "i20" => 7.5,
   "nano" => 2.5,
   "fabia" => 6.5
 ];
```



```php
// the price will be stored in the array
// on the car name position
$cars = [
  "i10" => 5.5,
  "i20" => 7.5,
  "nano" => 2.5
];

foreach ($cars as $price) {
  print($price)
}

foreach ($cars as $car => $price) {
  print($car . "has a price = " .$price)
}
```

- multidimensional array
    - also known as array of arrays

```php
// indexed array of indexed arrays
$array = [
  ["i10", "i20", "nano", "fabia"],
  ["steve", "bill", "sundar", "mark"]
];

/*
Array
(
    [0] => Array
        (
            [0] => i10
            [1] => i20
            [2] => nano
            [3] => fabia
        )

    [1] => Array
        (
            [0] => steve
            [1] => bill
            [2] => sundar
            [3] => mark
        )
)
*/
```

```
// indexed array of associative arrays
$array = [
  [
    "i10" => 5.5,
    "i20" => 7.5,
    "nano" => 2.5,
    "fabia" => 6.5
  ],
  [
    "steve" => "apple",
    "bill" => "ms",
    "sundar" => "google",
    "jeff" => "amazon",
    "elisson" => "oracle"
  ]
];

/*

Array
(
    [0] => Array
        (
            [i10] => 5.5
            [i20] => 7.5
            [nano] => 2.5
            [fabia] => 6.5
        )

    [1] => Array
        (
            [steve] => apple
            [bill] => ms
            [sundar] => google
            [jeff] => amazon
            [elisson] => oracle
        )

)
*/
```

```
// associative array of associative arrays
$array = [
    "cars" => [
      "i10" => 5.5,
      "i20" => 7.5,
      "nano" => 2.5,
      "fabia" => 6.5
    ],
    "persons" => [
      "steve" => "apple",
      "bill" => "ms",
      "sundar" => "google",
      "jeff" => "amazon",
      "elisson" => "oracle"
    ]
];

/*
Array
(
    [cars] => Array
        (
            [i10] => 5.5
            [i20] => 7.5
            [nano] => 2.5
            [fabia] => 6.5
        )

    [persons] => Array
        (
            [steve] => apple
            [bill] => ms
            [sundar] => google
            [jeff] => amazon
            [elisson] => oracle
        )
)
*/
```

super global variables