🤔

Data Type

- **Data Types Overview**
  - PHP supports various data types including:
    - **boolean** (`true`, `false`)
    - **integer** (e.g., `42`)
    - **float** (e.g., `3.14`)
    - **string** (e.g., `"Hello"`)
    - **array** (indexed or associative)
    - **object** (instances of classes)
    - **NULL** (no value)
    - **resource** (external resources like files or database connections)
    - **callable** (functions or methods that can be called)
    - **iterable** (arrays or objects that can be looped over)

1. Boolean

A **boolean** represents a logical value, either `true` or `false`.

Example:

```
$flag = true;
echo gettype($flag);  // Outputs: boolean
```

2. Integer

An **integer** represents whole numbers, both positive and negative.

Example:

```
$age = 25;
echo gettype($age);  // Outputs: integer
```

3. Float (or Double)

A **float** (also known as a **double**) represents numbers with decimal points.

Example:

```
$price = 19.99;
echo gettype($price);  // Outputs: double
```

4. String

A **string** represents text, which can be enclosed in single quotes (`'`) or double quotes (`"`).

Example:

```
$name = "John";
echo gettype($name);  // Outputs: string
```

5. Array

An **array** holds multiple values in a single variable. Arrays can be **indexed** (numeric keys) or **associative** (named keys).

Example of indexed array:

```
$fruits = ["apple", "banana", "cherry"];
echo gettype($fruits);  // Outputs: array
```

Example of associative array:

```
$person = ["name" => "John", "age" => 25];
```

```
echo gettype($person);  // Outputs: array
```

6. Object

An **object** is an instance of a class. It can hold properties and methods defined within the class.

Example:

```
class Person {
    public $name;
    public $age;

    public function __construct($name, $age) {
        $this->name = $name;
        $this->age = $age;
    }
}

$person = new Person("John", 25);
echo gettype($person);  // Outputs: object
```

7. NULL

The **NULL** type represents a variable with no value. A variable can be explicitly set to `null`, or it may be uninitialized.

Example:

```
$var = null;
echo gettype($var);  // Outputs: NULL
```

8. Resource

A **resource** represents an external resource, such as a database connection or file handle.

Example:

```
$file = fopen("example.txt", "r");
echo gettype($file);  // Outputs: resource
fclose($file);  // Close the file after use
```

9. Callable

A **callable** is a function or method that can be called. It can be a string (function name) or an array (object and method).

Example:

```
function greet() {
    echo "Hello!";
}

$func = "greet";
$func();  // Outputs: Hello!
```

10. Iterable

The **iterable** type represents any variable that can be looped over, such as arrays or objects implementing the `Traversable` interface.

Example:

```
function printArray(iterable $arr) {
    foreach ($arr as $value) {
        echo $value . "<br>";
    }
}

printArray([1, 2, 3]);  // Outputs: 1 2 3
```

11. Type Juggling

PHP automatically converts data between types when necessary. This is called **type juggling**.

Example:

```
echo 1 + "2.5";  // Outputs: 3.5 (String "2.5" is converted to a float)
```

12. Type Casting

You can explicitly convert (or **cast**) one data type to another using type casting.

Example:

```
$str = "123.45";
$int = (int) $str;  // Casts the string to an integer
echo gettype($int);  // Outputs: integer
```

13. String Escaping

You can escape characters inside a string using backslashes, such as `\'`, `\"`, and `\\`.

Example:

```
echo 'Hello "PHP"';  // Outputs: Hello "PHP"
echo "Hello 'PHP'";  // Outputs: Hello 'PHP'
echo 'Hello \'PHP\'';  // Outputs: Hello 'PHP'
```

14. Heredoc

Heredoc allows for creating strings that span multiple lines, and variables are parsed inside them.

Example:

```
$name = "John";
echo <<<EOD
Hello, $name!
```

```
This is a Heredoc string.
EOD
```

15. Nowdoc

Nowdoc is similar to Heredoc, but it doesn't parse variables inside the string.

Example:

```
$name = "John";
echo <<<'EOD'
Hello, $name!
This is a Nowdoc string.
EOD;
```