

الوحدة السادسة

- مفهوم المدى: المكان في البرنامج الذي يسمح استخدام المتغير فيه بدون اخطاء

- بدون دوال عند الاعلان عن متغير يكون متاح ومسموح لكل العناصر تحته

- وجود الدوال مختلف:
(تمرير المتغير مع **عدم** التاثر بالتعديلات عليه داخل الدالة)

```
<?php
$var1 = 'value';
function test()
{
    echo $var1;
}
test();
?>
```

error



```
function test()
{
    $var1 = 'value';
}
echo $var1;
?>
```

error

```
<?php
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <?php

$var1='value';
function test(){ $var1="hi"; echo $var1;}

test();
echo $var1;
```

 google  v

hivalue

- تمرير المتغير مع التاثر بالتعديلات عليه داخل الدالة:

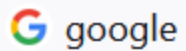
➤ العنوان

➤ global

```
<html>
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <?php

$var1='value';
function test($var1){$var1="hi"; echo $var1;}

test($var1);
echo $var1;
```



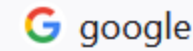
hivalue

```
?php
var1 = 'value';
function test()
{
  global $var1;
  $var1 = 'another value <br>';
  echo $var1;
}
```

```
<html>
<head>
  <meta charset="UTF-8">
  <title></title>
</head>
<body>
  <?php

$var1='value';
function test(&$var1){$var1="hi"; echo $var1;}

test($var1);
echo $var1;
```





hihi

With Global

```
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
```

```
$var1='value';
function test(){global $var1;$var1="hi"; echo $var1;}
```

```
echo $var1;
test();
```



 google  w

valuehi

```
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
```

```
$var1='value';
function test(){global $var1;$var1="hi"; echo $var1;}
```

```
test();
echo $var1;
```

 google 

hihi

json

- **مفهوم:** طريقة مختلفة لتخزين البيانات أو جلبها (مختلفة عن قاعدة البيانات)

- **قواعد:**

← يتم حفظ البيانات في تنسيق JSON على شكل كائن وتوضع العناصر بين الأقواس {}

```
{"var1":10,"var2":true,"var3":null,"var4":"value","var5":12.55}
```

شكل مصفوفة وتوضع عناصر المصفوفة بين الأقواس []

```
[10,20.25,"value",null,true]
```

← القيم التي يتم حفظها داخل الكائن أو المصفوفة هي أعداد صحيحة وأعداد كسرية وسلاسل

نصية وقيم منطقية وكائنات أخرى أو مصفوفات أخرى ويمكن الجمع بين جميع هذه الأنواع

كائن يحتوي على مصفوفة :

```
{"var1":10,"var2":[10,20,30]}
```

مصفوفة تحتوي على كائن :

```
[10,20,{"var1":"value1","var2":900},"value2"]
```

داخل كائن واحد أو مصفوفة واحدة

← ويتم إسناد القيم للعناصر باستخدام الرمز ":"

بين العناصر باستخدام الفاصلة ","

يجب أن يكون اسم العنصر بين علامتي إقتباس

دالة [json_encode](#) للتحويل إلى تنسيق الـ JSON.

دالة [json_decode](#) لتحويل تنسيق JSON إلى كائنات ومصفوفات يمكن التعامل معها من خلال لغة php.

دالة json_encode للتحويل إلى تنسيق الـ JSON.

يتم تحويل البيانات إلى ملف Jason ام على شكل:

كائن وله خصائص {

او

مصفوفة بها قيم مباشرة [

متى يتم التحويل إلى الشكل الأول ومتى تحول إلى الشكل الثاني؟؟؟؟؟؟؟؟ حسب القواعد اللاحقة

أولاً : تحويل البيانات إلى صيغة JSON باستخدام دالة `json_encode`

أمثلة عملية لإستخدام JSON :

1- لدينا مصفوفة ترابطية بها قيم مختلفة سيتم تحويلها لتنسيق JSON كالتالي :

```
<?php
$data['var1'] = 10;
$data['var2'] = 20.13;
$data['var3'] = null;
$data['var4'] = true;
$data['var5'] = 'value';
echo json_encode($data);
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
{"var1":10,"var2":20.13,"var3":null,"var4":true,"var5":"value"}
```

لاحظ: يتم
تحويل
المصفوفة
الترابطية
الى
كائن json

حالة 2 : مصفوفة عادية

مصفوفة عادية - أى معرفاتها عبارة عن أرقام- وتحتوى على قيم مختلفة

لاحظ: يتم
تحويل
المصفوفة
العادية الى
مصفوفة
json

```
<?php  
$data[] = 10;  
$data[] = 20.13;  
$data[] = null;  
$data[] = true;  
$data[] = 'value';  
echo json_encode($data);
```

المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
[10,20.13,null,true,"value"]
```

3- مصفوفة عادية تحتوي على قيم وعلى مصفوفة ترابطية وعلى مصفوفة عادية أخرى كالتالي :

```
<?php
    $data[] = 300;
    $data[] = array(10,20,30);
    $data[] =
array("var1"=>12.3,12.8,"var2"=>"value",9000,"var3"=>array(true,false));
    echo json_encode($data);
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
[300,[10,20,30],
{"var1":12.3,"0":12.8,"var2":"value","1":9000,"var3":
[true,false]]}]
```

لاحظ: يتم
تحويل كل
مصفوفة
عادية الى
مصفوفة
json وكل
ترابطية الى
كائن json

4- مصفوفة ترابطية تحتوي على قيم وعلى مصفوفة عادية كالتالي :

```
<?php
    $data =
    array("var1"=>12.3,12.8,"var2"=>array("value1","value2","value3"),
    9000);
    echo json_encode($data);
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

```
{"var1":12.3,"0":12.8,"var2":
["value1","value2","value3"],"1":9000}
```

لاحظ: يتم
تحويل كل
مصفوفة
عادية الى
مصفوفة
json وكل
ترابطية الى
كائن json

تحويل تنسيق JSON إلى كائنات ومصفوفات يمكن التعامل معها من خلال

لغة php .

ثانياً : تحويل صيغة JSON إلى كائنات ومصفوفات يمكن التعامل معها من خلال لغة php باستخدام دالة json_decode :

ملاحظة : بما أننا لم نتطرق للتعامل مع الكائنات حتى الآن فدالة json_encode تأخذ وسيط ثاني في حالة إعطائه القيمة true يتم تحويل كائنات الـ JSON إلى مصفوفات ترابطية hash table وإن أردت استخدام الكائن بدون تحويله لمصفوفة يمكنك الوصول للعناصر باستخدام الرمز <

1- جلب كائن في تنسيق JSON وتحويله إلى مصفوفة ترابطية في لغة PHP وبه الشكلان إما استخدام الكائن مباشراً أو تحويله لمصفوفة ترابطية واستخدامه كالتالي :

```
<?php
$json =
'{"var1":10,"var2":true,"var3":null,"var4":"value","var5":12.55}';
$data1 = json_decode($json);
$data2 = json_decode($json,true);
// الوصول للعناصر من خلال الكائن
echo $data1->var4;
echo "<br>";
// الوصول للعناصر عن طريق مصفوفة ترابطية
echo $data2['var4'];
?>
```

عند تنفيذ المثال السابق سيعطي نتيجة مماثلة للنتيجة التالية :

value
value