

Bloque A

Instrucciones básicas de programación

Unidad 1. Variables, expresiones y operadores



Contenidos

1. Introducción
2. Variables
3. Arrays
4. Abreviatura de echo
5. Expresiones y operadores
6. Manipulación de datos
7. Creación de una página PHP básica
8. Resumen de la unidad
9. Referencias



Introducción

Introducción

Esta primera unidad muestra **cómo las variables almacenan datos que pueden cambiar** cada vez que una página PHP es solicitada, y cómo las **expresiones y operadores** trabajan con los valores almacenados en dichas variables.

Las variables utilizan un nombre para representar un valor que puede cambiar cada vez que se solicita una página PHP:

- El **nombre** describe el tipo de datos que contiene la variable.
- El **valor** es el valor que la variable debe tener cada vez que la página es solicitada.

Una vez que la página ha terminado de ejecutarse y el HTML ha sido enviado de vuelta al navegador, **el intérprete de PHP olvida la variable** (así que la próxima vez que la página se ejecute, puede contener un valor diferente).

Introducción

PHP distingue entre diferentes tipos de valores que puede almacenar en una variable (como texto y números); estos se conocen como **tipos de datos**:

- Un trozo de texto se llama ***string***.
- Un número entero se denomina ***integer***
- Un número fraccionario se representa mediante un ***float***.
- Un valor verdadero o falso se denomina ***boolean***.
- Una serie de nombres y valores relacionados pueden almacenarse en un ***array***.

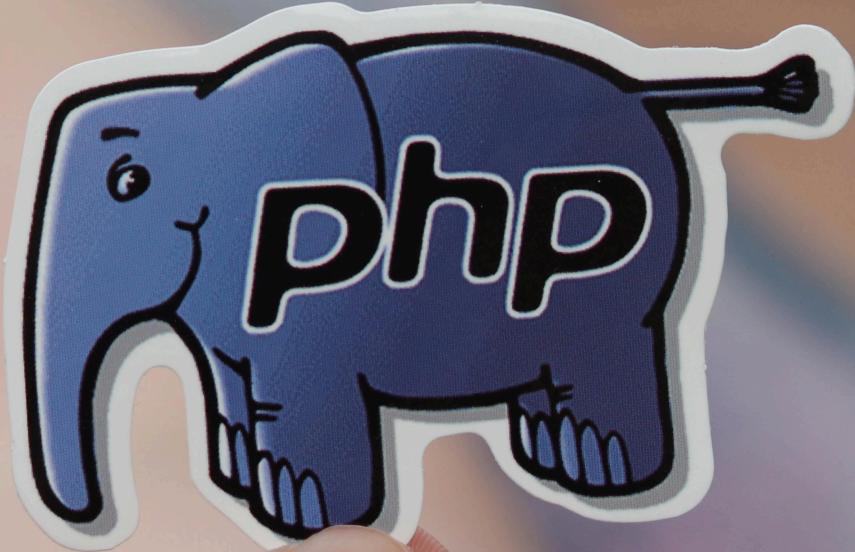
Introducción

Una vez que hayamos aprendido lo que son las variables, veremos cómo las **expresiones** pueden utilizar varios valores para crear uno solo.

Por ejemplo, el texto en dos variables puede unirse para formar una frase, o un número almacenado en una variable puede multiplicarse por un número en otra.

Las **expresiones** se basan en operadores para crear un único valor.

Por ejemplo, el operador `+` se utiliza para sumar dos valores y el operador `-` para restar un valor de otro.



Variables

Variables

Las variables **almacenarán datos que pueden cambiar** (o variar) cada vez que se solicita una página PHP.

Utilizan un **nombre** para representar un **valor** que puede cambiar.

Para crear una variable y almacenar un valor en ella, se necesita:

- Un **nombre** de variable que debe **empezar por un signo de dólar** (`$`), seguido de una o varias palabras que **describan el tipo de información** que puede contener la variable.
- Un **signo igual** (`=`), conocido como operador de asignación porque **asigna un valor al nombre de la variable**.
- El **valor que desea que contenga** la variable.

Variables

Si la variable contiene **texto**, éste se escribe entre comillas. Se puede utilizar comillas **simples** (' ') o **dobles** (" "), pero deben coincidir.

Por ejemplo, no es correcto empezar con comillas simples y cerrar el texto con comillas dobles.

Si la variable almacena un **número** o un **valor booleano** (verdadero o falso), no se coloca entre comillas.

Cuando se crea una variable, los programadores llamamos a esto **declarar una variable**.

Cuando se le asigna un valor, dicen que se está **asignando un valor** a la variable.

Variables

Ejemplo de declaración y asignación de valor

| NAME | VALUE |
|---------|----------|
| \$name | = 'Ivy'; |
| \$price | = 5; |

ASSIGNMENT OPERATOR

Variables

Una vez que se ha declarado una variable y se le ha asignado un valor, **el nombre de la variable se puede utilizar en el código PHP** siempre que se desee utilizar el valor que actualmente contenga la variable.

Cuando el intérprete PHP se encuentra con un nombre de variable, **reemplaza el nombre con el valor que contiene**. A continuación, se utiliza el comando `echo` para mostrar el valor almacenado en la variable `$name` mostrada previamente.

```
echo $name;
```



DISPLAY VALUE IN VARIABLE

Creando y accediendo a variables

En el siguiente ejemplo, se crean dos variables y se les asignan valores en la parte superior de la página PHP:

1. \$name contiene el **nombre del visitante actual del sitio**. Se trata de texto, por lo que se escribe entre comillas.
2. \$price contiene el **precio de un paquete de caramelos**. Es un número, por lo que el valor no se escribe entre comillas.

```
<?php  
$name = 'Ivy';  
$price = 5;  
?>
```

Creando y accediendo a variables

A continuación, podemos ver el HTML que se envía al navegador del visitante, en el cual:

- El **nombre del visitante** se escribe en la página utilizando el comando `echo`.
- El **precio del caramelo** se escribe en la página de la misma forma.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
      $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

Creando y accediendo a variables

```
<?php
$name  = 'Ivy';
$price = 5;
?>
<!DOCTYPE html>
<html>
<head>
    <title>Variables</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <h1>The Candy Store</h1>
    <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
        $<?php echo $price; ?> per pack.</p>
</body>
</html>
```

Creando y accediendo a variables

The screenshot shows a web browser window with the URL "localhost" in the address bar. The page itself has a dark red background and features a large, stylized title "The Candy Store" in a gold-colored font. Below the title, the text "WELCOME IVY" is displayed in white. A message follows: "The cost of your candy is \$5 per pack." To the right of the text, there is a vibrant, colorful illustration of various candies, including a large striped candy cane, several lollipops, and smaller circular and triangular candies, all set against a dark red background.

Creando y accediendo a variables

En esta primera unidad, asignaremos los valores a las variables directamente en el código PHP.

En unidades posteriores, los valores asignados a las variables podrán **proceder de otras fuentes**, como formularios HTML que envían los visitantes, datos en URLs o de bases de datos.

Creando y accediendo a variables

Realiza las siguientes modificaciones sobre el ejemplo anterior:

- En el Paso 1, cambia el valor de la variable `$name` para que contenga tu nombre. Guarda el archivo y actualiza la página en tu navegador. Verás que se muestra tu nombre.
- En el Paso 2, cambia el precio a 2€. Guarda el archivo y luego actualiza la página. Verás el nuevo precio.

Cómo nombrar a las variables

El nombre de una variable debe **describir apropiadamente los datos que almacena**. Utilizaremos las siguientes **reglas** para crear un nombre de variable.

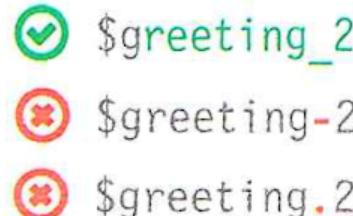
1. Empezar con un **signo de dólar** (`$`).



2. Seguido de una **letra o un guión bajo** (no un número).



3. Después, utilizaremos **cualquier combinación de letras de la A a la Z (mayúsculas y minúsculas)**, **números** y **guiones bajos**. (No se permiten guiones ni puntos).



Cómo nombrar a las variables

Utilizar nombres de variables que describan los datos que almacenan facilita la comprensión y el seguimiento del código.

Si utilizas más de una palabra para describir los datos que almacena una variable, es habitual **separar cada palabra con un guión bajo**.

Ojo: `$this` tiene un significado especial (hace referencia a la instancia actual dentro de una clase). No se puede utilizar como nombre de variable.

Cómo nombrar a las variables

Los nombres **distinguen entre mayúsculas y minúsculas** (case-sensitive), por lo que `$Puntuación` y `$puntuación` serían dos nombres diferentes.

Sin embargo, en general **debes evitar crear dos variables que utilicen la misma palabra y diferentes combinaciones de mayúsculas/minúsculas**, ya que es probable que confundan a otras personas que lean tu código.

Además, técnicamente, **se pueden utilizar caracteres de distintos conjuntos de caracteres** (como caracteres chinos por ejemplo), pero a menudo se considera una **buenas práctica utilizar sólo letras de la A a la Z, números y guiones bajos** (ya que es posible que surjan algunos problemas complicados a la hora de admitir otros caracteres).

Tipos de datos escalares (básicos)

PHP distingue entre tres tipos de datos escalares que contienen **texto**, **números** y **booleanos**:

Tipo de datos cadena de texto (*String*)

Los programadores denominamos cadena (o *string*) a un **fragmento de texto**. El tipo de datos *string* puede estar formado por letras, números y otros caracteres, pero se utilizan para representar texto.

```
$name = 'Carlos';
```

Los strings **siempre van entre comillas simples o dobles**. La comilla inicial debe coincidir con la comilla final.

Tipos de datos escalares (básicos)

Ejemplos uso correcto de comillas en *String*

```
$name = 'Ivy'; // Bien  
$name = "Ivy"; // Bien  
$name = "Ivy'; // Error  
$name = 'Ivy"'; // Error
```

Tipos de datos escalares (básicos)

Tipos de datos numéricos

Los tipos de datos numéricos **permiten realizar operaciones matemáticas** con los valores que contienen, como sumas o multiplicaciones.

```
$price = 15;
```

Los números **no se escriben entre comillas**. Si colocamos números entre comillas, pueden ser tratados como cadenas de texto en lugar de números.

PHP tiene dos tipos de datos numéricos:

- **int** representa enteros, que son números enteros (por ejemplo, 275).
- **float** contiene números de punto flotante, que representan fracciones (por ejemplo,

2.75).

Tipos de datos escalares (básicos)

Tipo de dato booleano (*boolean*)

El tipo de datos booleano sólo puede tener uno de dos valores: **verdadero o falso**. Estos valores son habituales en la mayoría de los lenguajes de programación.

```
$logged_in = true;
```

`true` y `false` deben escribirse en **minúsculas** y **no se colocan entre comillas**. Al principio, los valores booleanos pueden parecer abstractos, pero **podemos representar muchas cosas utilizando este tipo de valores**, como por ejemplo:

- ¿Ha iniciado sesión un visitante?
- ¿Ha aceptado los términos y condiciones?
- ¿Tiene un producto derecho a envío gratuito?

Tipos de datos escalares (básicos)

Tipo de dato nulo (*null*)

PHP, como muchos otros lenguajes de programación, también tiene un tipo de datos llamado *null*, el cual **indica que no se ha especificado un valor para una variable**.

Conversión de tipos

Más adelante veremos cómo el intérprete de PHP puede **convertir un valor de un tipo de datos a otro** (por ejemplo, una cadena a un número).

Actualizando el valor de una variable

Podemos cambiar o sobrescribir el valor almacenado en una variable **asignándole un nuevo valor**. Esto **se hace de la misma forma que se asigna un valor a la variable al crearla**.

A continuación veremos un **ejemplo de sobreescritura del valor de una variable**:

Actualizando el valor de una variable

1. La variable `$name` se inicializa. Esto significa que **se declara y se le asigna un valor inicial**, que se utilizará si la variable no se actualiza más adelante en la página.

El valor inicial es "Guest"; se escribe entre comillas ya que es texto.

2. A la variable `$name` se le asigna entonces el **nuevo valor** de "Ivy".
3. La variable `$price` contiene el precio de un paquete individual de caramelos.

```
<?php  
$name = 'Guest';  
$name = 'Ivy';  
$price = 5;  
?>
```

Actualizando el valor de una variable

A continuación, podemos ver el HTML que se enviará al navegador del visitante. En el HTML:

4. El nombre se escribe en la página utilizando el comando `echo` . Muestra el valor actualizado que se asignó a la variable `$name` en el paso 2, **no el original con el que se inicializó la variable.**
5. El coste del caramelo se escribe en la página también mediante `echo` .

Actualizando el valor de una variable

```
<!DOCTYPE html>
<html>
  <head>
    <title>Updating Variables</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Welcome <?php echo $name; ?></h2>
    <p>The cost of your candy is
      $<?php echo $price; ?> per pack.</p>
  </body>
</html>
```

Actualizando el valor de una variable

```
<?php  
① $name = 'Guest';  
② $name = 'Ivy';  
③ $price = 5;  
?>  
<!DOCTYPE html>  
<html>  
    <head>  
        <title>Updating Variables</title>  
        <link rel="stylesheet" href="css/styles.css">  
    </head>  
    <body>  
        <h1>The Candy Store</h1>  
        ④ <h2>Welcome <?php echo $name; ?></h2>  
        <p>The cost of your candy is  
        ⑤     $<?php echo $price; ?> per pack.</p>  
    </body>  
</html>
```

Actualizando el valor de una variable

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area has a dark red background with a festive candy-themed illustration on the right side featuring a lollipop, a candy cane, and several smiling circular faces. The text 'The Candy Store' is displayed in large, gold-colored letters at the top left. Below it, the message 'WELCOME IVY' is shown in white. A text block states 'The cost of your candy is \$5 per pack.' The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with various icons.

The Candy Store

WELCOME IVY

The cost of your candy is \$5 per pack.

Actualizando el valor de una variable

Realiza las siguientes modificaciones sobre el ejemplo anterior:

- En el Paso 2, cambia el valor de la variable \$name para que contenga tu nombre. Guarda el archivo y actualiza la página en tu navegador. Verás que se muestra tu nombre.
- Añade una nueva línea después del Paso 2 y cambia el valor de la variable \$name por otro nombre. Guarda el archivo y actualiza la página. Verás que se muestra el nuevo nombre.



Arrays

Arrays

Una variable también puede contener una **matriz** (o **array/vector**) que **almacena una serie de valores relacionados**.

Las matrices se conocen como **tipos de datos compuestos** porque pueden **almacenar más de un valor**.

Arrays

Un array es como un **contenedor que contiene un conjunto de variables relacionadas**. Cada ítem del array se denomina **elemento**. De la misma manera que una variable utiliza un nombre para representar un valor, cada elemento de un array tiene una:

- **Clave** que actúa igual que el nombre de una variable
- **Valor** que es el dato que representa el nombre

PHP tiene dos tipos de arrays:

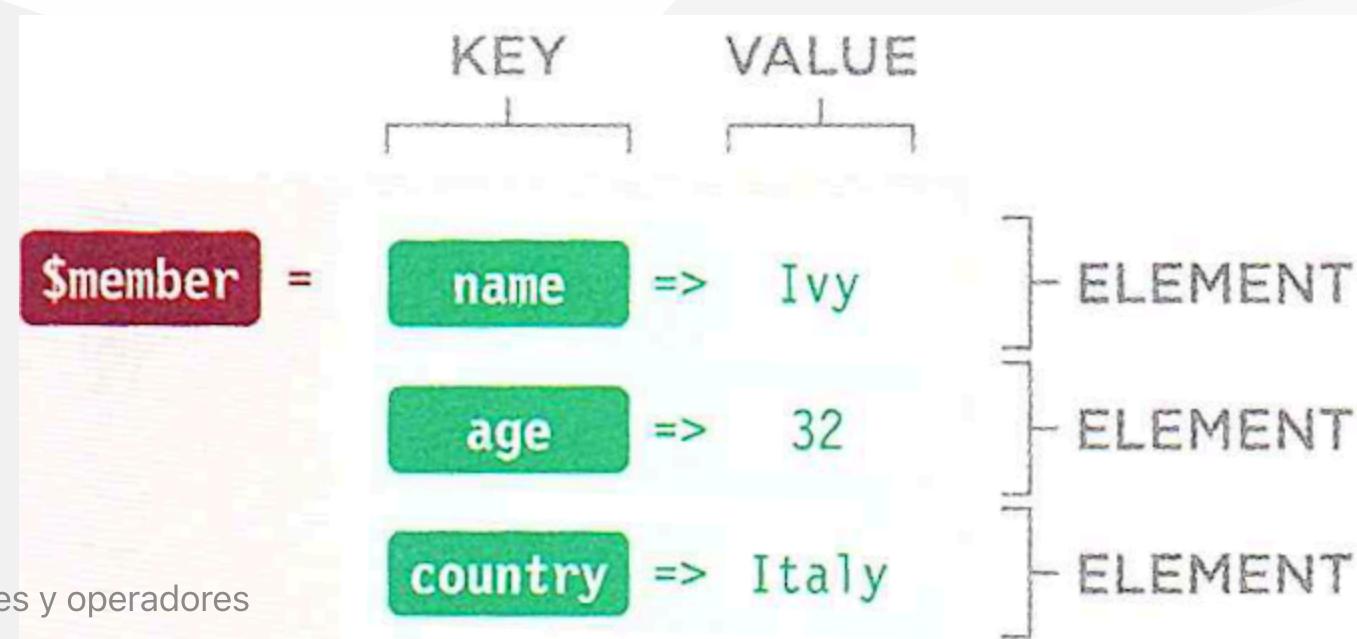
- En los arrays **asociativos**, la **clave de cada elemento es un nombre** que describe los datos que representa.
- En los arrays **indexados**, la **clave de cada elemento es un número** conocido como **número índice**.

Arrays

Array asociativo

El array que se muestra a continuación está diseñado para **almacenar datos que representan a un miembro de un sitio web**.

Cada vez que se utilice este array, los nombres utilizados en las claves (que describen los datos almacenados en cada elemento de la matriz) **seguirán siendo los mismos**.

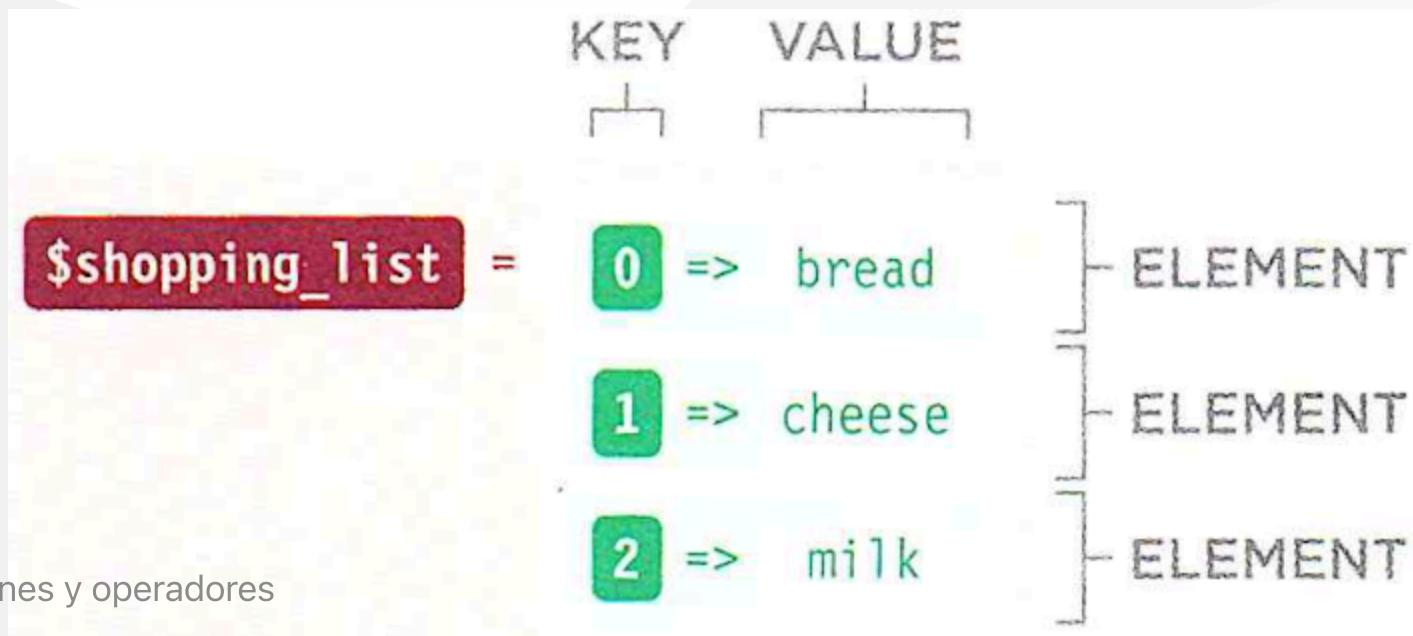


Arrays

Array indexado

El siguiente array está diseñado para contener una lista de la compra. Las listas de este tipo pueden contener un número diferente de elementos cada vez que se utilizan.

La clave no utiliza un nombre para describir cada elemento de la lista; **utiliza un número de índice** (que es un **número entero y empieza por 0**).



Arrays

En estos dos ejemplos, cada valor almacenado en el array es un tipo de dato escalar (un dato individual).

Pero también podemos encontrarnos ejemplos de arrays en los que un elemento del array contiene otro array.

Importante: Los números de índice en un array indexado comienzan en 0, no en 1. El primer elemento de la lista tiene un número de índice 0. El segundo elemento se identifica por el número de índice 1, y así sucesivamente.

Arrays asociativos

Para crear un array asociativo, **asignaremos a cada elemento (o ítem) del array una clave que describa los datos que contiene.**

Para **almacener un array asociativo en una variable**, se deberá utilizar:

- Un nombre de variable que describa el conjunto de valores que contendrá el array.
- El operador de asignación.
- Corchetes para crear el array.

Arrays asociativos

Dentro de los corchetes, utilizaremos:

- El **nombre de la clave** entre comillas.
- El **operador de doble flecha** `=>`.
- El **valor de este elemento** (las cadenas van entre comillas; los números y los booleanos no)
- Una **coma después de cada elemento**.

Arrays asociativos

| VARIABLE | CREATE ARRAY |
|-----------------------|------------------------|
| | |
| \$member = [| |
| 'name' => 'Ivy', | |
| 'age' => 32, | |
| 'country' => 'Italy', | |
|]; | |
| | KEY OPERATOR VALUE |

Arrays asociativos

También se puede crear un array asociativo utilizando la sintaxis que se muestra a continuación, con la palabra **array** seguida de paréntesis (en lugar de corchetes).

```
$member = array(  
    'name'    => 'Ivy',  
    'age'     => 32,  
    'country' => 'Italy',  
);
```

Arrays asociativos

Para acceder a un elemento de un array asociativo, se utiliza:

- El nombre de la **variable que contiene el array**.
- Seguido de **corthetes y comillas**.
- Dentro de las comillas, escribimos la **clave del elemento que deseamos recuperar**.

| VARIABLE | KEY |
|-------------------|-----|
| | |
| \$member['name']; | |

Arrays asociativos

1. En el siguiente ejemplo, **se crea un array asociativo** y se almacena en una variable llamada `$nutrition`.

El array se crea entre corchetes. Tiene **tres elementos** (cada elemento tiene un par clave/valor). El operador `=>` asigna los valores a cada una de las claves.

```
<?php
$nutrition = [
    'fat'  => 16,
    'sugar' => 51,
    'salt'  => 6.3,
];
?>
```

Arrays asociativos

2. Para mostrar los datos almacenados en el array, se utiliza:

- El comando `echo` para indicar que se escriba en la página web el siguiente valor.
- Seguido del **nombre de la variable que contiene el array**.
- Seguido de corchetes y comillas que contienen el **nombre de la clave a la que se quiere acceder**.

Por ejemplo, para escribir el contenido de azúcar en la página utilizaremos: `echo $nutrition ['sugar'] ;`

Arrays asociativos

```
<!DOCTYPE html>
<html>
  <head>
    <title>Associative Arrays</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Nutrition (per 100g)</h2>
    <p>Fat: <?php echo $nutrition['fat']; ?>%</p>
    <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>
    <p>Salt: <?php echo $nutrition['salt']; ?>%</p>
  </body>
</html>
```

Arrays asociativos

The screenshot shows a web browser window with the URL 'localhost' in the address bar. The page itself has a dark red background and features the title 'The Candy Store' in large, gold-colored letters. Below the title, there is a section titled 'NUTRITION (PER 100G)' in white. Underneath this section, three items are listed with their respective percentages: 'Fat: 16%', 'Sugar: 51%', and 'Salt: 6.3%'. To the right of the text, there is a vibrant, colorful illustration of various candies, including a large lollipop, several wrapped candies, and some smiley faces, all set against a dark red background.

The Candy Store

NUTRITION (PER 100G)

Fat: 16%

Sugar: 51%

Salt: 6.3%

localhost

DWES - U1. Variables, expresiones y operadores

47

Arrays asociativos

Realiza las siguientes modificaciones en el ejemplo anterior:

En el paso 1, cambia los valores del array. Dale a la clave

- *fat* un valor de 42.
- *sugar* un valor de 60.
- *salt* un valor de 3,5.

Guarda y actualiza la página para ver los valores actualizados.

En el paso 1, añade otro elemento al array. Utiliza la clave *protein* y asígnale un valor de 2,6.

Luego, en el Paso 2, muestra el valor de la proteína en la página.

Arrays indexados

Cuando se crea un array, si no se proporciona una clave para cada elemento, el intérprete de PHP le asignará un número llamado número de índice (o simplemente *índice*). Los índices comienzan en cero (0), no en uno (1).

Para **almacenar un array indexado en una variable**, se utiliza:

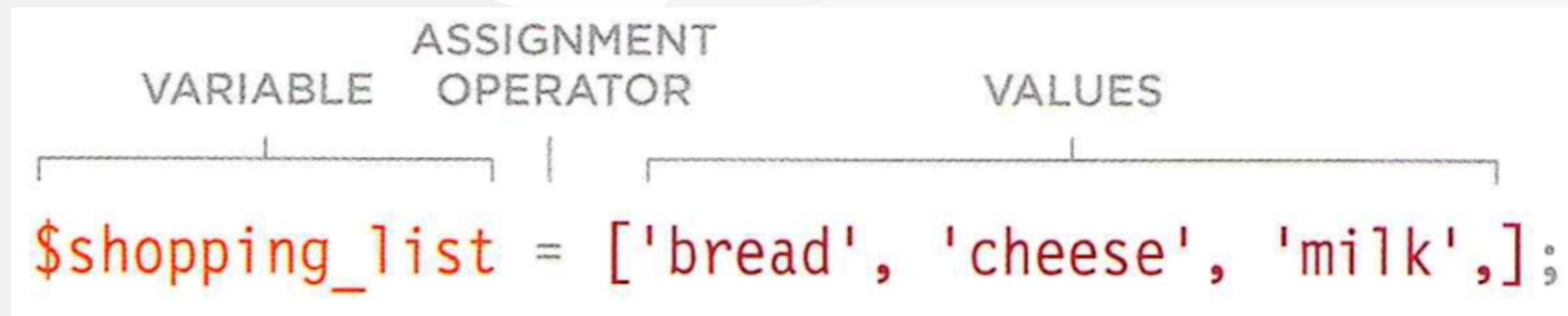
- Un nombre de variable que describa el conjunto de valores que contendrá la matriz
- El operador de asignación
- Corchetes para crear la matriz

Arrays indexados

Dentro de los corchetes, se utiliza:

- La lista de valores que contendrá el array (las cadenas van entre comillas, los números y los booleanos no)
- Una coma después de cada valor

A cada elemento se le asignará un número de índice.



Arrays indexados

En el ejemplo anterior, al pan se le asignaría el número de índice 0, al queso 1 y a la leche 2. Los números de índice se utilizan a menudo para **indicar el orden de los elementos listados en el array**.

También se puede crear un array indexado utilizando la sintaxis que se muestra a continuación, con la palabra `array` seguida de paréntesis (en lugar de corchetes).

```
$shopping_list = array('bread',
                       'cheese',
                       'milk');
```

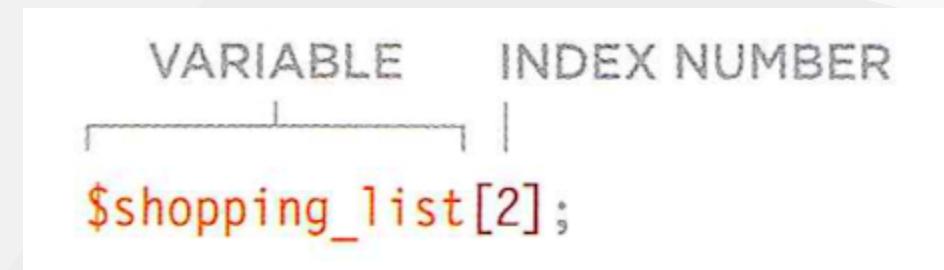
Arrays indexados

Cada valor que se añade al array puede estar en la misma línea o en una nueva línea (como se muestra en el último ejemplo).

Para **acceder a los elementos de un array indexado**, se utiliza:

- El nombre de la variable que contiene la matriz.
- Seguido de corchetes (sin comillas).
- El número de índice del elemento al que desea acceder (entre corchetes)

El código siguiente obtiene el tercer elemento de la matriz, por lo que en este ejemplo obtendríamos el valor "*milk*".



Arrays indexados

1. En el siguiente ejemplo se comienza creando una variable llamada `$best_sellers`. Su valor es un **array que contiene una lista de los artículos más vendidos en el sitio web.**

Este array se crea utilizando corchetes, y los elementos se añaden al array dentro de esos corchetes. Como los elementos de la matriz son texto, se colocan entre comillas (los números y los booleanos no irían entre comillas). Cada elemento va seguido de una coma.

```
<?php  
$best_sellers = ['Chocolate', 'Mints', 'Fudge',  
    'Bubble gum', 'Toffee', 'Jelly beans',];  
?>
```

Arrays indexados

2. Los tres artículos más vendidos se escriben en la página:

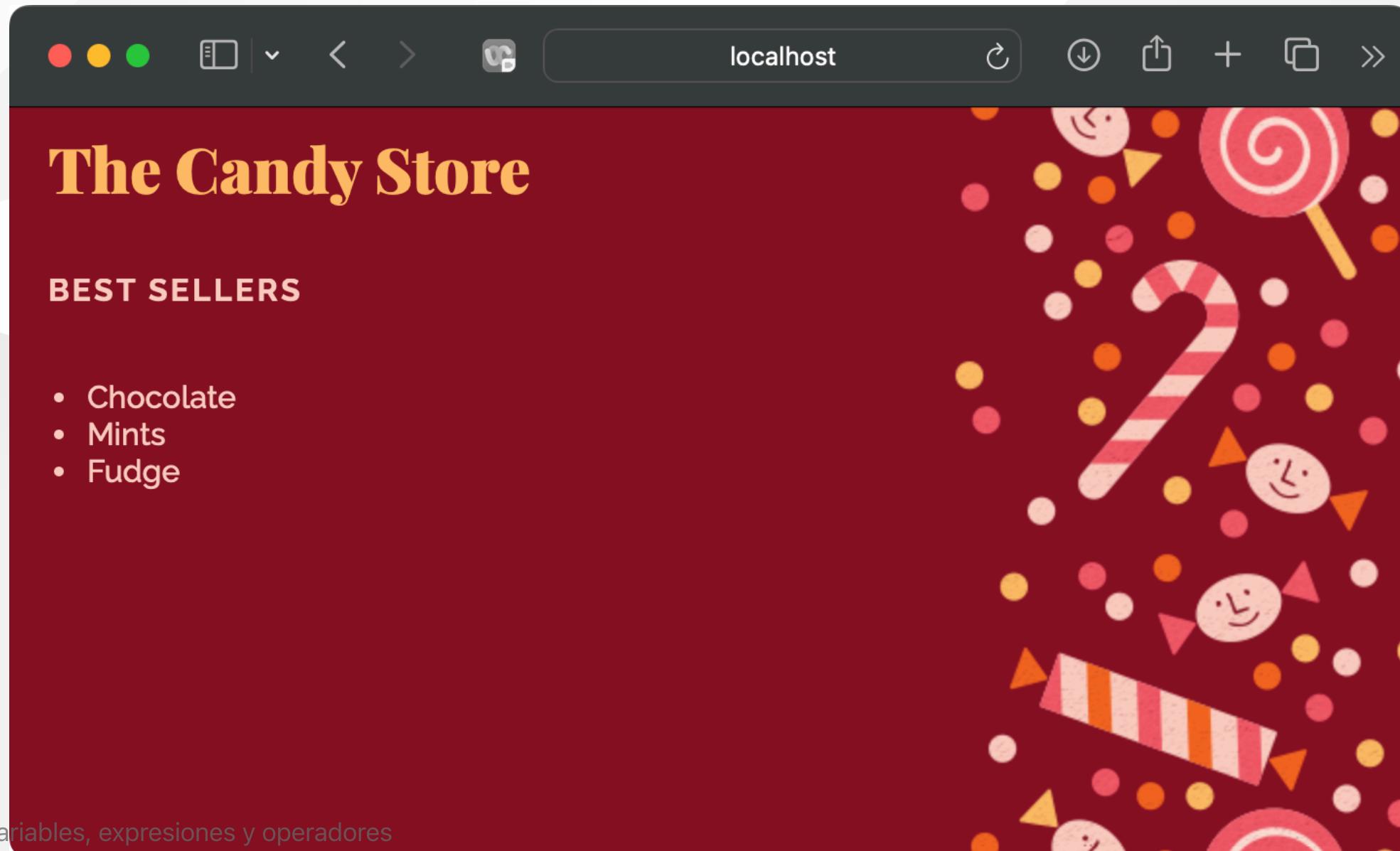
- El comando `echo` indica que se escriba el siguiente valor.
- Seguido del nombre de la variable que contiene la matriz.
- A continuación, los corchetes que contienen el número de índice del artículo que desea recuperar.

¡Recuerda que los números de índice empiezan en 0, no en 1!

Arrays indexados

```
<!DOCTYPE html>
<html>
  <head>
    <title>Indexed Arrays</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Best Sellers</h2>
    <ul>
      <li><?php echo $best_sellers[0]; ?></li>
      <li><?php echo $best_sellers[1]; ?></li>
      <li><?php echo $best_sellers[2]; ?></li>
    </ul>
  </body>
</html>
```

Arrays indexados



Arrays indexados

Sobre el ejemplo anterior, en el Paso 1, añade un nuevo elemento al array después de elemento "*Fudge*" (caramelo). En el Paso 2, muestra también el 4º y 5º elemento del array.

Actualizando arrays

Una vez creado un array, puedes añadirle nuevos elementos o actualizar el valor de cualquiera de los elementos que lo componen.

Para **actualizar un valor almacenado en un array asociativo**, se utiliza:

- El nombre de la variable que contiene el array
- Seguido de corchetes
- A continuación, el nombre de la clave entre comillas
- Un operador de asignación
- El nuevo valor que debe contener

The diagram illustrates the structure of the assignment statement: \$member['name'] = 'Tom';. It uses brackets and arrows to label parts of the code:

- A bracket under '\$member' is labeled "VARIABLE".
- A bracket under 'name' is labeled "KEY".
- A bracket under "'Tom'" is labeled "NEW VALUE".

```
$member['name'] = 'Tom';
```

VARIABLE KEY NEW VALUE

Actualizando arrays

Para **añadir un nuevo elemento a un array asociativo**, haz exactamente lo mismo que antes, pero utiliza un nuevo nombre de clave (no uno que ya se haya utilizado en el array).

Las comillas van alrededor del nombre de la clave cuando es un string porque las comillas indican un tipo de datos string. También podemos utilizar enteros como clave de un array asociativo.

Actualizando arrays

Para **actualizar un valor almacenado en un array indexado**, se utiliza:

- El nombre de la variable que contiene el array - Seguido de corchetes
- El número de índice (sin comillas)
- Un operador de asignación
- El nuevo valor que debe contener

The diagram shows the code: `$shopping_list[2] = 'butter';`. Below the code, three horizontal arrows point to the corresponding parts: the first arrow points to `$shopping_list` and is labeled "VARIABLE"; the second arrow points to `[2]` and is labeled "INDEX NUMBER"; the third arrow points to `'butter'` and is labeled "NEW VALUE".

```
$shopping_list[2] = 'butter';
```

VARIABLE INDEX NUMBER NEW VALUE

Actualizando arrays

Estudiaremos **cómo añadir elementos a arrays indexados** más adelante. El proceso es diferente porque puedes especificar la posición del nuevo elemento en el array.

Las comillas no se colocan alrededor del índice porque los tipos de datos numéricos no se entrecorren.

¿Qué tipo de array utilizar?

Los **arrays asociativos** son mejores cuando:

- Se sabe exactamente qué información contendrá el array. Esto es necesario para proporcionar un nombre clave para cada uno de los elementos.
- Necesita obtener datos individuales utilizando un nombre de clave.

Los **arrays indexados** son útiles cuando:

- No se sabe cuántos datos se van a almacenar en el array. (Los números de índice aumentan a medida que se añaden más elementos a la lista).
- Desea almacenar una serie de valores en un orden específico.

Ejemplo: Modificando valores almacenados en arrays

1. El siguiente ejemplo comienza **almacenando un array** en una variable llamada `$nutrition` .

No es necesario que las claves y los valores que componen cada elemento del array estén en una nueva línea (como se muestra aquí), pero facilita la lectura si lo están.

2. El valor que se almacena para el elemento "fat" **se actualiza** de 38 a 36.
3. Se añade un **nuevo elemento** al array. La clave es "fiber" y su valor es 2.1.
4. Los **valores del array se escriben** en la página.

Ejemplo: Modificando valores almacenados en arrays

```
<?php
$nutrition = [
    'fat'    => 38,
    'sugar'  => 51,
    'salt'   => 0.25,
];
$nutrition['fat']  = 36;
$nutrition['fiber'] = 2.1;
?>
```

Ejemplo: Modificando valores almacenados en arrays

```
<!DOCTYPE html>
<html>
  <head>
    <title>Updating Arrays</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Nutrition (per 100g)</h2>
    <p>Fat: <?php echo $nutrition['fat']; ?>%</p>
    <p>Sugar: <?php echo $nutrition['sugar']; ?>%</p>
    <p>Salt: <?php echo $nutrition['salt']; ?>%</p>
    <p>Fiber: <?php echo $nutrition['fiber']; ?>%</p>
  </body>
</html>
```

Ejemplo: Modificando valores almacenados en arrays

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area features a red background with a candy-themed illustration on the right side. The illustration includes a large striped candy cane, several smaller candies, and some smiling faces. On the left, the text 'The Candy Store' is displayed in a large, stylized font. Below it, the heading 'NUTRITION (PER 100G)' is shown in a bold, white, sans-serif font. Underneath this heading, there is a list of nutritional information:

- Fat: 36%
- Sugar: 51%
- Salt: 0.25%
- Fiber: 2.1%

DWES - U1. Variables, expresiones y operadores

66

Ejemplo: Modificando valores almacenados en arrays

En el ejemplo anterior, después del paso 3, añade otra clave para la proteína (*protein*) y asígnale un valor de 7,3.

Arrays multidimensionales

El **valor de cualquier elemento de un array puede ser otro array**. Cuando cada elemento de un array contiene otro array, se denomina **array multidimensional** y puede utilizarse para representar datos que pueden verse en tablas.

Hay ocasiones en las que es necesario **almacenar un conjunto de valores relacionados en un elemento de un array** (por ejemplo, cuando se representan datos que tradicionalmente se han visto en tablas).

Consideremos la siguiente tabla con tres miembros, sus edades y sus países.

Arrays multidimensionales

| NAME | AGE | COUNTRY |
|------|-----|---------|
| Ivy | 32 | UK |
| Emi | 24 | Japan |
| Luke | 47 | USA |

Arrays multidimensionales

Cada fila de esta tabla (cada miembro) puede representarse utilizando un elemento de un array indexado.

Cada elemento puede contener un array asociativo que almacene el nombre, la edad y el país de cada miembro.

Los índices para el array indexado son asignados automáticamente por el intérprete de PHP. La coma después de cada array asociativo indica el final del valor para ese elemento.

```
$members = [  
    ['name' => 'Ivy', 'age' => 32, 'country' => 'UK'],  
    ['name' => 'Emi', 'age' => 24, 'country' => 'Japan'],  
    ['name' => 'Luke', 'age' => 47, 'country' => 'USA'],  
];
```

Arrays multidimensionales

Para obtener el array que contiene los datos sobre *Emi*, utiliza la función:

- Nombre de la variable que contiene el array indexado
- Número de índice del elemento al que quieras acceder entre corchetes (recuerda que los arrays indexados empiezan en 0, y que los números no se ponen entre comillas).

```
$members [1] ;
```

Arrays multidimensionales

Para obtener la edad de *Luke*, utilizaremos:

- Nombre de la variable que contiene el array indexado
- Número de índice del elemento que contiene el array de datos sobre *Luke*, entre corchetes
- Clave del elemento al que quieras acceder en el array sobre *Luke* en un segundo conjunto de corchetes (como la clave es una cadena, ponla entre comillas).

```
$members [2] ['age'] ;
```

Ejemplo: Arrays multidimensionales

1. Este ejemplo comienza almacenando un array indexado en una variable llamada `$offers`.

Cada elemento del array almacena un array asociativo que contiene el nombre, precio y nivel de stock de un artículo que está en oferta.

2. Se escribe el nombre del primer producto. (El número de índice del primer producto es 0).
3. Se escribe el precio del primer producto.
4. Se escribe el nombre y el precio del segundo producto.
5. Se escriben el nombre y el precio del tercer producto.

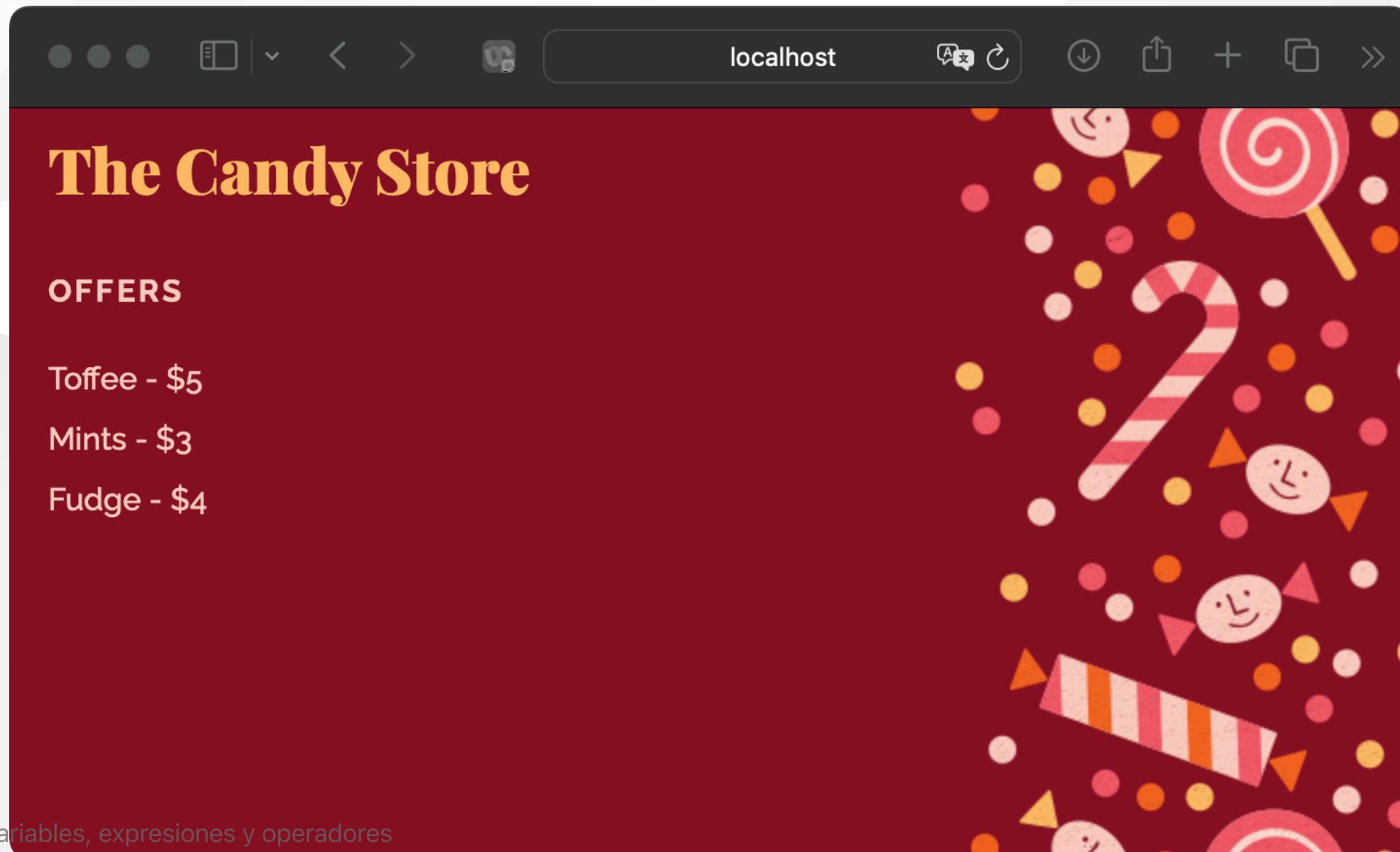
Ejemplo: Arrays multidimensionales

```
<?php
$offers = [
    ['name' => 'Toffee', 'price' => 5, 'stock' => 120,],
    ['name' => 'Mints', 'price' => 3, 'stock' => 66,],
    ['name' => 'Fudge', 'price' => 4, 'stock' => 97,],
];
?>
```

Ejemplo: Arrays multidimensionales

```
<!DOCTYPE html>
<html>
  <head>
    <title>Multidimensional Arrays</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Offers</h2>
    <p><?php echo $offers[0]['name']; ?> -
      $<?php echo $offers[0]['price']; ?> </p>
    <p><?php echo $offers[1]['name']; ?> -
      $<?php echo $offers[1]['price']; ?> </p>
    <p><?php echo $offers[2]['name']; ?> -
      $<?php echo $offers[2]['price']; ?> </p>
  </body>
</html>
```

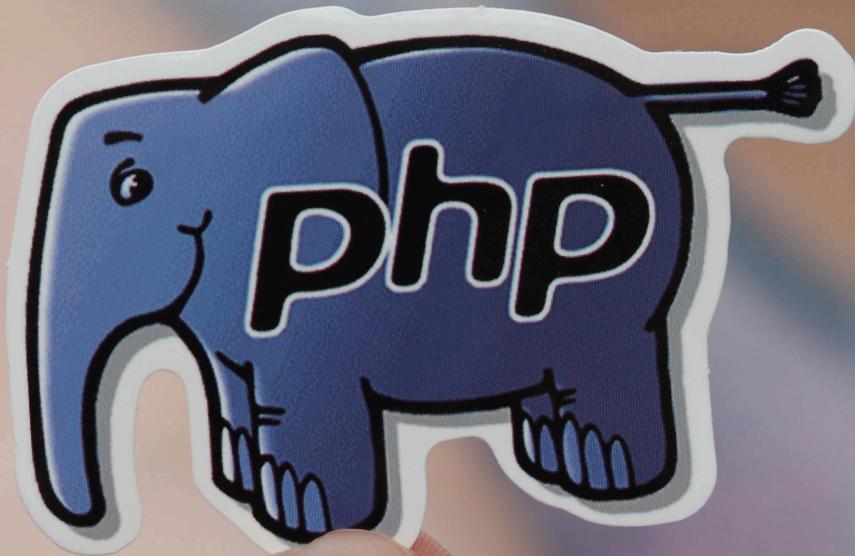
Ejemplo: Arrays multidimensionales



Ejemplo: Arrays multidimensionales

Sobre el ejemplo anterior, en el Paso 1, añade otro producto con el nombre Chocolate al array. Fija el precio en 2 y asígnale un nivel de existencias de 83. A continuación, después del Paso 5, escribe el nombre y el precio del nuevo producto que acabas de añadir.

En la próxima unidad veremos cómo se puede utilizar un bucle para escribir el nombre y el precio de cada producto en el array `$offers`, sin importar cuántos productos contenga el array.



Abreviatura de "echo"

Abreviatura de "echo"

Cuando un bloque PHP sólo se utiliza para escribir un valor en el navegador, podemos utilizar la **abreviatura en lugar de** `<?php echo ?>`.

En lugar de escribir `<?php echo $variable; ?>` se puede utilizar la **abreviatura**:

```
<?= $variable ?>
```

Esta es la única vez que no es necesario utilizar el delimitador de la apertura completa `<?php`.

Usando la abreviatura, **nos ahorramos**:

- Las letras php en la etiqueta de apertura
- El comando echo
- Un punto y coma antes de la etiqueta de cierre

Abreviatura de "echo"

SHORTHAND
FOR ECHO



<?= \$username ?>

<?= \$list[0] ?>

CLOSING TAG



VALUE TO DISPLAY

Abreviatura de "echo"

En muchos de los ejemplos que estudiaremos, verás que cada archivo PHP tiene dos partes:

- Primero, **el código PHP almacena valores en variables o arrays**. (También puede realizar tareas con los datos que contienen).
- Luego, **el código HTML que es enviado de vuelta al navegador**. Esta segunda parte de la página mostrará valores que fueron almacenados en variables utilizando la sintaxis abreviada que acabamos de ver.

Abreviatura de "echo"

Si comienzas cada página creando los valores que la página necesitará mostrar, y almacenándolos en variables, ayudarás a crear una **clara separación entre el código PHP que se ejecuta en el servidor, y el código HTML que el visitante acabará viendo.**

La segunda parte del fichero, donde se crea la página HTML, **debería utilizar la mínima cantidad posible de código PHP**. En los primeros ejemplos que estudiaremos en este módulo, el código PHP en esta parte de la página sólo escribirá valores almacenados en variables en el HTML.

Ejemplo: Abreviatura de "echo"

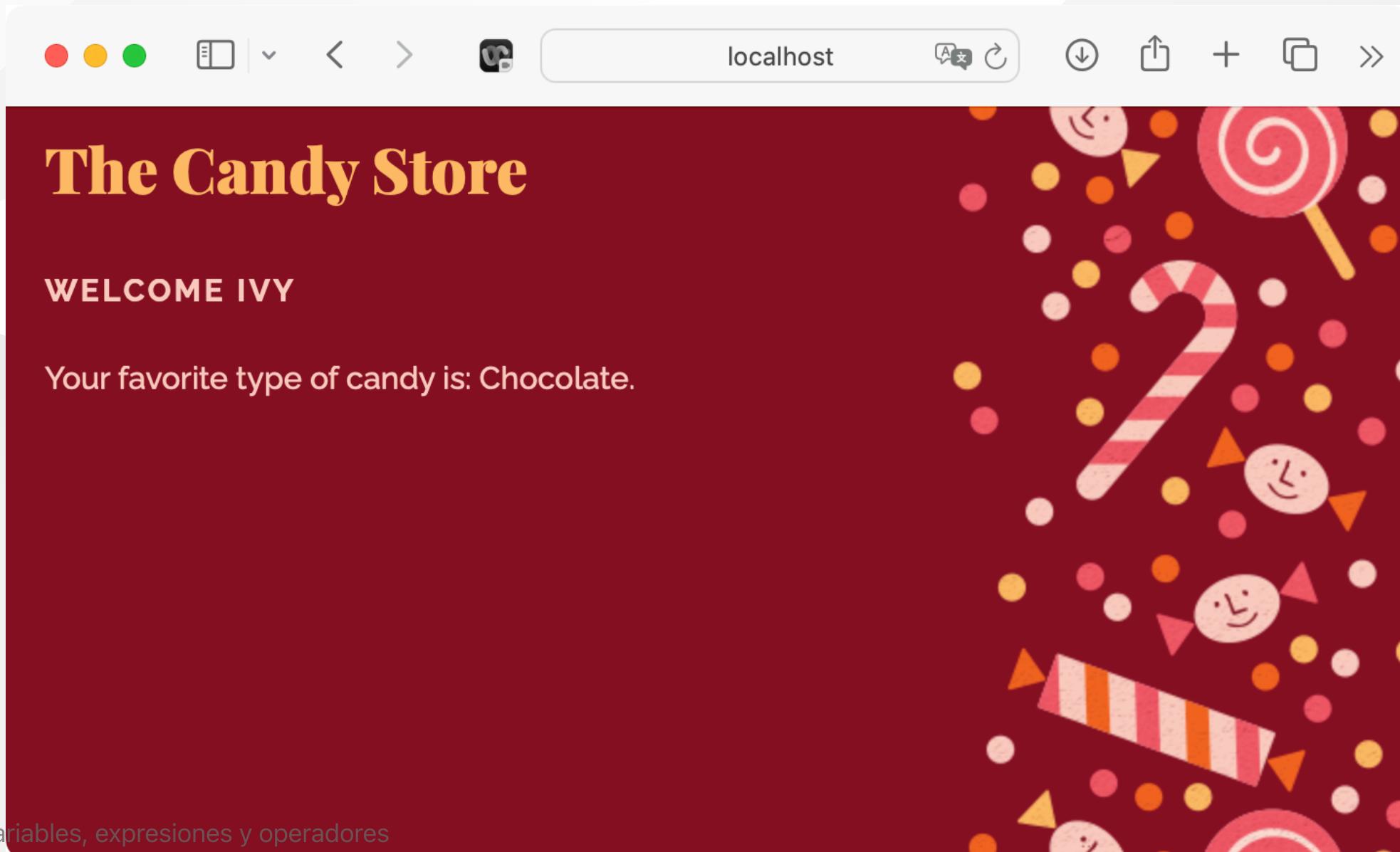
En el siguiente ejemplo, podemos ver que se han creado dos variables diferentes y se les han asignado valores en la parte superior de la página antes de que comience el HTML:

1. `$name` contiene el nombre de un miembro del sitio. Esto es texto por lo que se almacena entre comillas.
2. `$favorites` contiene un array de los tipos de caramelos favoritos del miembro.
3. El nombre y el tipo de caramelo favorito del miembro se escriben en la página utilizando la abreviatura del comando `echo`.

Ejemplo: Abreviatura de "echo"

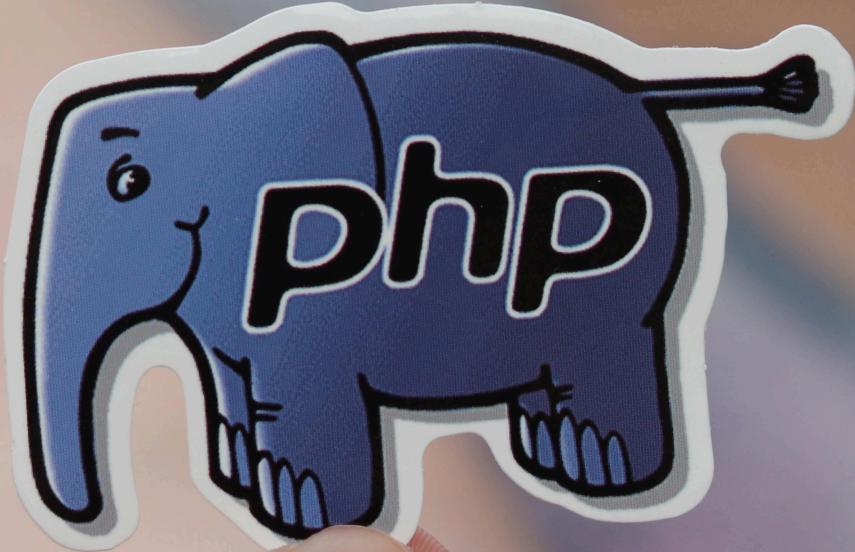
```
<?php
$name      = 'Ivy';
$favorites = ['Chocolate', 'Toffee', 'Fudge',];
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Echo Shorthand</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Welcome <?= $name ?></h2>
    <p>Your favorite type of candy is:
      <?= $favorites[0] ?>.</p>
  </body>
</html>
```

Ejemplo: Abreviatura de "echo"



Ejemplo: Abreviatura de "echo"

Sobre el ejemplo anterior, en el Paso 1, cambia el valor almacenado en la variable `$name` por tu nombre. En el Paso 2, añade tu tipo favorito de caramelo al principio del array. Guarda el archivo y actualiza la página en tu navegador. Verás que el contenido cambia.



Expresiones y operadores

Expresiones y operadores

A menudo se utilizan dos (o más) valores para crear uno nuevo.

Las **expresiones** consisten en una o más construcciones que se evalúan (dan como resultado) un único valor.

Las expresiones utilizan **operadores** para crear un único valor.

Expresiones y operadores

Las **matemáticas básicas** (suma, resta, multiplicación y división) utilizan dos valores para crear uno nuevo.

La siguiente expresión multiplica el número 3 por el número 5 para crear un valor de 15:

```
3 * 5
```

Los programadores decimos que las expresiones se **evalúan** a un único valor. A continuación, el nuevo valor que se crea se almacena en una variable llamada `$total`:

```
$total = 3 * 5;
```

Los símbolos `+ - * /` se denominan **operadores**.

Expresiones y operadores

Puedes **unir dos o más cadenas** para crear un texto más largo utilizando un operador de cadena llamado **operador de concatenación**.

La siguiente expresión une los valores 'Hi ' e 'Ivy' para crear una cadena:

```
$greeting = 'Hola ' . 'Marcos';
```

La unión de estas dos cadenas se evalúa en un único valor de "*Hola Marcos*", que se almacena en la variable llamada `$greeting`.

A continuación profundizaremos en los distintos tipos de operadores que podemos emplear en PHP.

Operadores aritméticos

Los operadores aritméticos te permiten trabajar con números, realizando tareas como **sumas, restas, multiplicaciones y divisiones**.

Por ejemplo, si alguien compra 3 paquetes de caramelos y cada paquete cuesta 5\$, puedes utilizar un operador de multiplicación para calcular el coste total de esos tres paquetes de caramelos.

PHP permite utilizar los siguientes **operadores matemáticos**. Pueden ser utilizados con números literales o variables que almacenan números.

Operadores aritméticos

| NAME | OPERATOR | PURPOSE | EXAMPLE | RESULT |
|----------------|----------|---|---------------------|--------|
| Addition | + | Add one value to another | 10 + 5 | 15 |
| Subtraction | - | Subtract one value from another | 10 - 5 | 5 |
| Multiplication | * | Multiply two values (NOTE: This is an asterisk and not the letter x) | 10 * 5 | 50 |
| Division | / | Divide two values | 10 / 5 | 2 |
| Modulus | % | Divide two values and return the remainder | 10 % 3 | 1 |
| Exponentiation | ** | Raise a value to the power of another | 10 ** 5 | 100000 |
| Increment | ++ | Add one to the number and return the new value | \$i = 10; \$i++; | 11 |
| Decrement | -- | Subtract one and return the new value | \$i = 10; \$i--; | 9 |

Operadores aritméticos

Orden de ejecución

Es posible realizar varias operaciones aritméticas en una misma expresión, pero es importante comprender el orden en que se calculará el resultado: **la multiplicación y la división se realizan antes que la suma y la resta.**

Esto puede afectar al número que esperas ver. Por ejemplo, los números aquí se calculan de izquierda a derecha. El resultado es 16:

```
$total = 2 + 4 + 10;
```

En el siguiente, sin embargo, el resultado es 42 (no 60):

```
$total = 2 + 4 * 10;
```

Operadores aritméticos

Orden de ejecución

Los **paréntesis** te permiten indicar qué cálculo quieras que se realice primero, de modo que lo siguiente da como resultado un total de 60:

```
$total = (2 + 4) * 10;
```

Los paréntesis indican que 2 se suma a 4 antes de multiplicarse por 10.

Ejemplo: Operadores aritméticos

El siguiente ejemplo muestra **cómo se utilizan operadores matemáticos con números para calcular el coste de un pedido**. En primer lugar, se crean dos variables para almacenar:

1. Número total de artículos pedidos (`$items`)
2. Coste de cada paquete de caramelos (`$cost`)

A continuación, se realizan los cálculos y los resultados se almacenan en variables antes de crear el HTML. Esto ayuda a separar el código PHP del contenido HTML.

3. El coste del pedido se calcula **multiplicando el número de artículos por el coste de un paquete de caramelos**.

Ejemplo: Operadores aritméticos

4. Hay que **añadir un 20% de impuestos**. Para ello, el subtotal se divide por 100. (Esto se hace entre paréntesis). (Esto se hace entre paréntesis para asegurarse de que se calcula en primer lugar.) A continuación, el resultado se multiplica por 20.
5. Por último, **se añade el impuesto al subtotal** para hallar el coste total.
6. Los resultados almacenados en las variables **se escriben en la página HTML**.

Ejemplo: Operadores aritméticos

```
<?php
① $items      = 3;
② $cost       = 5;
③ $subtotal   = $cost * $items;
④ $tax        = ($subtotal / 100) * 20;
⑤ $total      = $subtotal + $tax;
?>
<!DOCTYPE html>
<html>
<head> ... </head>
<body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Items: <?= $items ?></p>
    <p>Cost per pack: $<?= $cost ?></p>
    <p>Subtotal: $<?= $subtotal ?></p>
    <p>Tax: $<?= $tax ?></p>
    <p>Total: $<?= $total ?></p>
</body>
</html>
```

Ejemplo: Operadores aritméticos

The screenshot shows a web browser window with the URL "localhost" in the address bar. The page itself has a dark red background with a festive candy-themed illustration on the right side featuring a large lollipop, a candy cane, and several smiling faces made of circles and triangles.

The Candy Store

SHOPPING CART

Items: 3

Cost per pack: \$5

Subtotal: \$15

Tax: \$3

Total: \$18

Ejemplo: Operadores aritméticos

Sobre el ejemplo anterior, cambia el coste de los artículos en el Paso 1 y la cantidad en el Paso 2.

Operadores de cadenas (strings)

Los operadores de cadena permiten trabajar con texto. Existen dos operadores de cadena que se utilizan para combinar distintos fragmentos de texto en una sola cadena.

Por ejemplo, si tienes el nombre de un miembro almacenado en una variable y su apellido almacenado en una segunda variable, puedes unir las dos variables para crear su nombre completo.

El proceso de unir dos o más cadenas se denomina **concatenación**.

Operadores de cadenas (strings)

Operador de concatenación .

El operador de concatenación es un **símbolo de punto** (.).

Une el valor de una cadena con el valor de otra. En el ejemplo siguiente, la variable llamada \$name contendría la cadena 'Sergio Díaz':

```
$forename = 'Sergio';
$surname = 'Díaz';
$name = $forename . ' ' . $surname;
```

Operadores de cadenas (strings)

Operador de concatenación .

Observa que se añade un espacio entre las variables `$forename` y `$surname` ; si no existiera el espacio, la variable `Sname` contendría el valor *SergioDíaz*.

Se pueden **concatenar tantas cadenas como se desee en una sentencia** siempre que se utilice el operador de concatenación entre cada cadena.

Operadores de cadenas (strings)

Operador de asignación de concatenación `.=`

Si deseas **añadir texto a una variable existente**, puedes utilizar el operador de asignación de concatenación. Se trata de una forma abreviada de crear una cadena actualizada:

```
$greeting = 'Hola ';  
$greeting .= 'Pablo';
```

En este caso, la cadena '*Hola*' se almacena en una variable llamada `$greeting`. En la siguiente línea, el operador de asignación de concatenación añade la cadena '*Pablo*' al final del valor de la variable llamada `$greeting`.

Ahora, la variable `$greeting` contiene el valor '*Hola Pablo*'. Como puedes ver, utiliza una línea de código menos que el ejemplo anterior.

Operadores de cadenas (strings)

Se pueden unir cadenas almacenadas en variables sin un operador de concatenación.

Si se asigna un valor con comillas dobles (en lugar de comillas simples), el intérprete de PHP sustituye los nombres de las variables entre comillas dobles por los valores que contienen.

A continuación, `$name` contendría el valor "Sergio Díaz".

```
$forename = 'Sergio';
$surname = 'Díaz';

$name = "$forename $surname";
```

Ejemplo: Operadores de cadenas (strings)

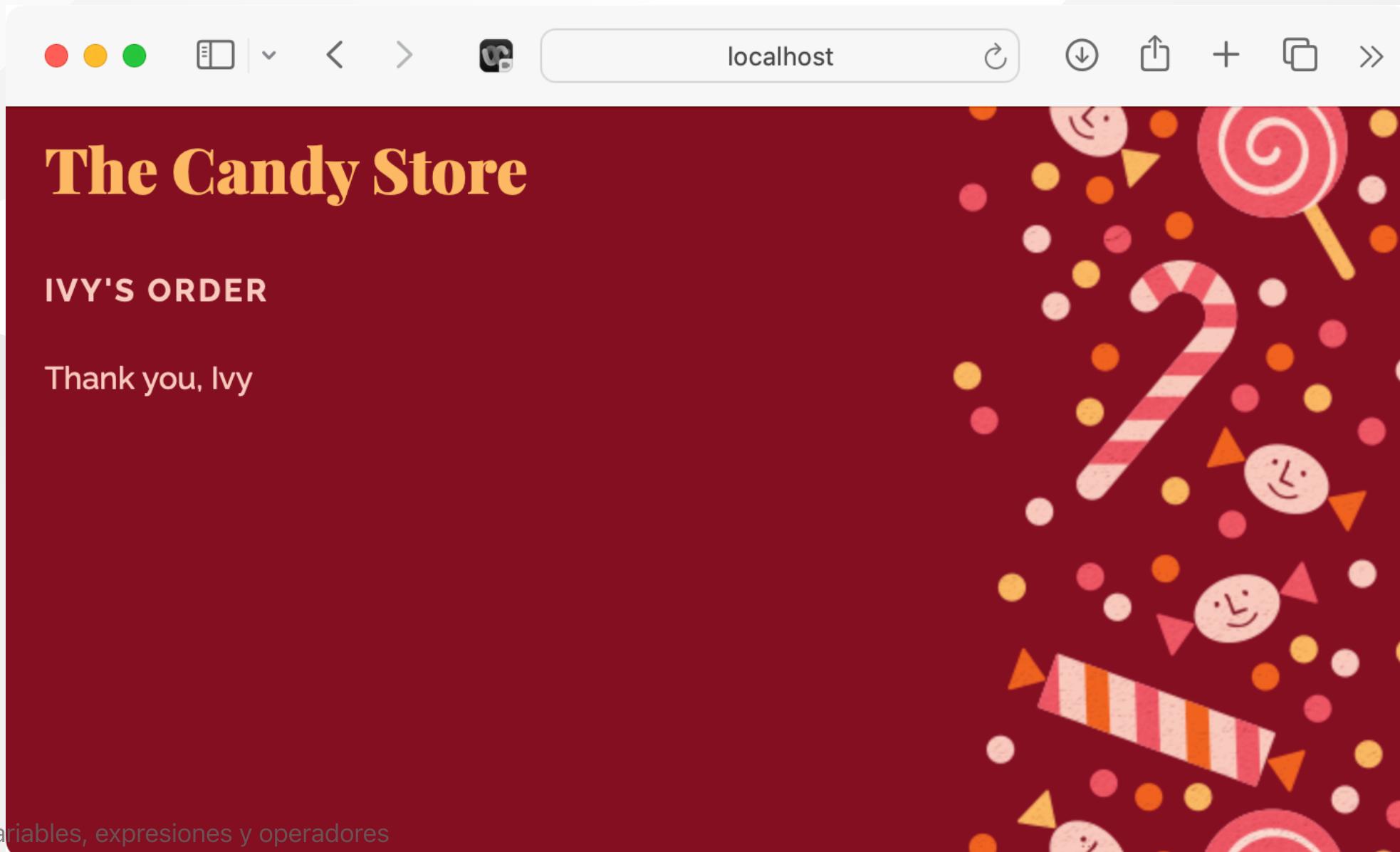
En este ejemplo se mostrará un mensaje personalizado.

1. En primer lugar, se crea una variable llamada `$prefix` para almacenar el **inicio del mensaje para el visitante**. Contiene las palabras '*Thank you*'.
2. Se crea una segunda variable para almacenar el **nombre del visitante**. La variable se llama `$name`, y el visitante se llama *Ivy*.
3. El **mensaje personal se crea concatenando (o uniendo) tres valores y almacenando el nuevo valor** en una variable llamada `$message` :
 - En primer lugar, se añade a `$message` el valor almacenado en `$prefix` .
 - A continuación, se añade una coma y un espacio.
 - Por último, se añade el valor almacenado en `$name` .

Ejemplo: Operadores de cadenas (strings)

```
<?php
$prefix  = 'Thank you';
$name    = 'Ivy';
$message = $prefix . ', ' . $name;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>String Operator</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2><?= $name ?>'s Order</h2>
    <p><?= $message ?></p>
  </body>
</html>
```

Ejemplo: Operadores de cadenas (strings)



Ejemplo: Operadores de cadenas (strings)

Sobre el ejemplo anterior, en el Paso 2, cambia el valor almacenado en `$name` por tu nombre.

En el Paso 3, asigna el valor de la variable `$message` utilizando comillas dobles (y sin operador de concatenación).

Operadores de comparación

Como su nombre indica, los operadores de comparación **comparan dos valores y devuelven un valor booleano** de verdadero o falso.

Por ejemplo, si tomamos los números 3 y 5, podemos compararlos para ver si

- 3 es mayor que 5 (*falso*)
- 3 es igual a 5 (*falso*)
- 3 es menor que 5 (*verdadero*)

También puedes **comparar cadenas para ver si un valor es mayor o menor que otro**:

- 'Manzana' es mayor que 'Plátano' (*falso*)
- "A" es igual a "B" (*falso*)
- "A" es menor que "B" (*verdadero*)

Operadores de comparación

Es igual a (==)

Este operador compara dos valores para ver si son iguales.

- '`Hello`' == '`Hello`' da como resultado *verdadero* porque son la misma cadena.
- '`Hello`' == '`Goodbye`' da como resultado *false* porque no son la misma cadena.

Si utilizamos `echo` para escribir el valor de un booleano en la página, `true` mostrará 1 y `false` no mostrará nada.

Operadores de comparación

Es distinto de (!= o <>)

Estos operadores comparan dos valores para ver si no son iguales.

- '`Hello`' `!=` '`Hello`' da como resultado *falso* porque son la misma cadena.
- '`Hello`' `!=` '`Goodbye`' da como resultado *verdadero* porque no son la misma cadena.

Operadores de comparación

Los operadores anteriores permiten al intérprete de PHP **determinar si los dos valores son equivalentes o no.**

Los operadores que veremos a continuación son **más estrictos** porque **comprueban tanto el valor como el tipo de datos.**

Los operadores anteriores tratarían 3 (un entero) como equivalente a 3.0 (un float). Los operadores siguientes no lo harían.

Operadores de comparación

Es idéntico a (`==`)

Este operador compara dos valores para comprobar que tanto el valor como el tipo de datos son iguales.

- `'3' == 3` da como resultado *falso* porque **no son del mismo tipo de datos**.
- `'3' == '3'` da como resultado *verdadero* porque **son el mismo tipo de datos y valor**.

Operadores de comparación

No es idéntico a (`!==`)

Este operador compara dos valores para comprobar que tanto el valor como el tipo de datos no son iguales.

- `3.0 ! == 3` da como resultado *verdadero* porque **no son del mismo tipo de datos**.
- `3.0 !== 3.0` da como resultado *falso* porque **son el mismo tipo de datos y valor**.

Operadores de comparación

Menor que (<) y mayor que (>)

< comprueba si el valor de la izquierda es menor que el de la derecha.

```
4 < 3 // false  
3 < 4 // true
```

> comprueba si el valor de la izquierda es mayor que el de la derecha.

```
z > a // true  
a > z // false
```

Operadores de comparación

Menor o igual que (`<=`) y mayor o igual que (`>=`)

`<=` comprueba si el valor de la izquierda es menor o igual que el valor de la derecha.

```
4 <= 3 // false  
3 <= 4 // true
```

`>=` comprueba si el valor de la izquierda es mayor o igual que el valor de la derecha.

```
z >= a // true  
z >= z // true
```

Operadores de comparación

Operador de nave espacial (`<=>` spaceship operator)

El operador nave espacial compara los valores a su izquierda y derecha y da como resultado:

- `0` si ambos valores son iguales
- `1` si el valor de la izquierda es mayor
- `-1` si el valor de la derecha es mayor

Este operador se introdujo en PHP 7 (y no funciona con versiones anteriores de PHP).

```
1 <=> 1 // 0
2 <=> 1 // 1
2 <=> 3 // -1
```

Operadores lógicos

Los tres **operadores lógicos** *AND* (`&&`), *OR* (`||`) y *NOT* (`!`) funcionan con dos valores: *verdadero* o *falso*.

Para entender cómo funcionan, considera las dos preguntas siguientes; ambas pueden responderse con verdadero o falso:

¿Hace calor? ¿Hace sol?

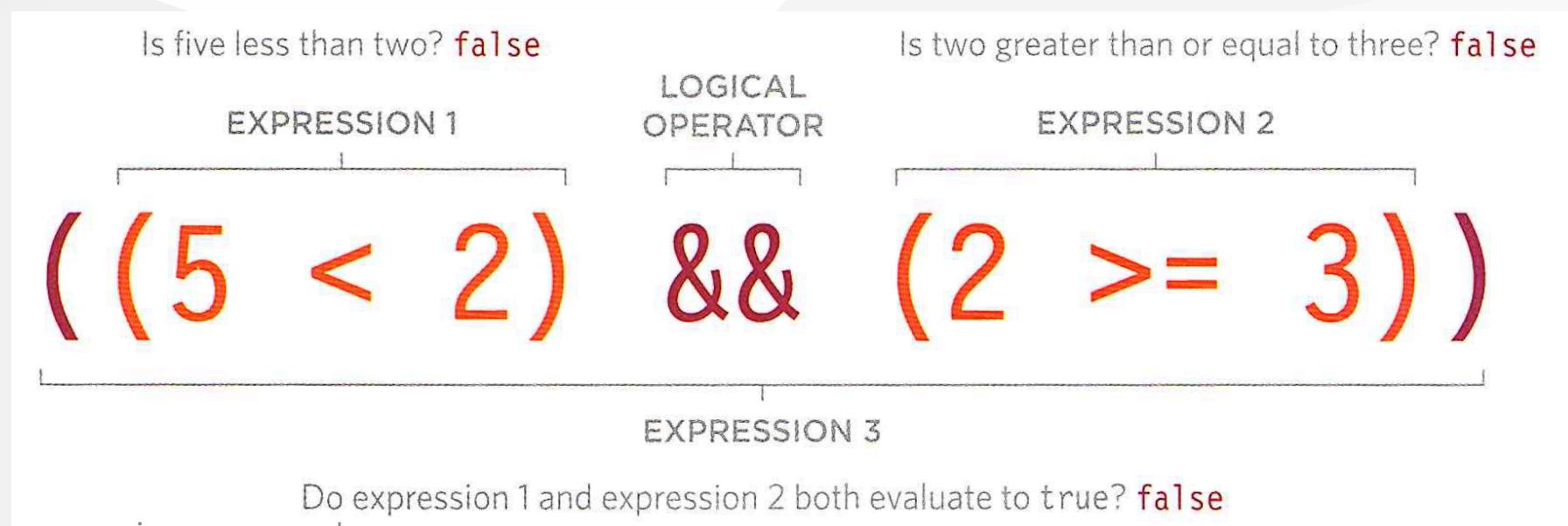
- El operador *and* comprueba si la temperatura es alta **y** hace sol.
- El operador *or* comprueba si la temperatura es alta **o** si hace sol.
- El operador *not* puede comprobar si la respuesta a sólo una de estas preguntas a la vez **no es cierta**. (p.e: ¿**no** hace sol?, ¿**no** hace calor?)

Operadores lógicos

Cada uno de estos operadores da como resultado un valor *verdadero* o *falso*.

Los operadores lógicos **pueden utilizarse con varios operadores de comparación** para comparar los resultados de varias expresiones.

En esta línea de código, hay tres expresiones, cada una de las cuales se resolverá con un único valor de verdadero o falso.



Operadores lógicos

La expresión 1 (a la izquierda) y la expresión 2 (a la derecha) utilizan operadores de comparación, y ambas expresiones dan como resultado el valor *falso*.

La expresión 3 utiliza un operador lógico (en lugar de un operador de comparación).

El operador lógico ***and*** (***&&***) **comprueba si ambas expresiones (a ambos lados) devuelven verdadero.** En este caso, no lo hacen, por lo que toda la expresión se evaluará a un valor de *falso*.

Operadores lógicos

Las expresiones 1 y 2 se evalúan antes que la 3.

Cada expresión se ha puesto en su propio conjunto de paréntesis.

Esto ayuda a mostrar que el código en cada conjunto de paréntesis debe evaluarse en un único valor.

Funciona igualmente sin paréntesis (**los operadores de comparación tienen mayor prioridad que los operadores lógicos a la hora de evaluarse**), pero eso hace que sea más difícil de leer.

Operadores lógicos

AND (&&)

Este operador comprueba más de una condición. P.e: `((2 < 5) && (3 >= 2))` El resultado es *verdadero*.

Si ambas expresiones se evalúan como *verdadero*, la expresión devuelve *verdadero*.

Si una de ellas resulta *falsa*, la expresión resulta *falsa*.

```
true && true // true  
true && false // false  
false && true // false  
false && false // false
```

Puedes utilizar la palabra `and` en lugar de dos ampersands (`&&`).

Operadores lógicos

OR (||)

Este operador comprueba al menos una condición. P.e: `((2 < 5) || (2<1))` El resultado es *verdadero*.

Si cualquiera de las expresiones da como resultado *verdadero*, la expresión devuelve *verdadero*. Si ambas expresiones resultan *falsas*, la expresión resulta *falsa*.

```
true || true // true  
true || false // true  
false || true // true  
false || false // false
```

Puedes utilizar la palabra `or` en lugar de dos caracteres de barra vertical (`||`).

Operadores lógicos

NOT (!)

Este operador toma un único valor booleano y lo niega. P.e: `!(2 < 1)` Da como resultado un valor de *verdadero*.

El operador `!` **niega una expresión**. Si es falsa (sin la `!` delante), el resultado es *verdadero*. Si la expresión es verdadera, da como resultado *falso*.

```
!true // false
!false // true
```

No se puede utilizar la palabra `not` en lugar del signo de exclamación.

Operadores lógicos

Las expresiones lógicas **se evalúan de izquierda a derecha**.

Una vez que la primera expresión ha sido evaluada y el intérprete de PHP conoce el operador lógico, **puede que no necesite evaluar la segunda condición**, como se puede ver en los ejemplos siguientes:

Operadores lógicos

```
((5 < 2) && (2 >= 2))
```



A **false** value was found.

No tiene sentido seguir probando la segunda condición porque **ambas no pueden dar como resultado el valor verdadero**.

Operadores lógicos

```
((2 < 5) || (2 >= 2))
```



A **true** value was found.

No tiene sentido seguir comprobando la segunda condición porque **al menos uno de los valores es verdadero**.

Ejemplo: Usando operadores de comparación

1. En el siguiente ejemplo, se crean tres variables:

- La primera contiene el **tipo de caramelo que quiere el cliente.**
- La segunda muestra que hay 5 paquetes en **stock.**
- La tercera muestra que el cliente quiere **8 paquetes.**

2. Un operador de comparación **comprueba si la cantidad deseada es menor o igual que la cantidad en stock.** El resultado se almacena en una variable llamada `$can_buy`.

Ejemplo: Usando operadores de comparación

3. **Casi nunca se escribe un booleano directamente en la página web** (como se muestra aquí). Es más probable que el valor se utilice en lógica condicional, que conoceremos en la próxima unidad. Pero es importante ver lo que obtienes cuando intentas escribir un valor booleano como este; si el valor es:
- *true*, la página mostrará 1
 - *false*, la página no mostrará nada

Ejemplo: Usando operadores de comparación

```
<?php
① $item      = 'Chocolate';
    $stock     = 5;
    $wanted    = 8;
② $can_buy  = ($wanted <= $stock);
    ?>

<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Shopping Cart</h2>
        <p>Item:   <?= $item ?></p>
        <p>Stock:   <?= $stock ?></p>
        <p>Wanted:  <?= $wanted ?></p>
③   <p>Can buy: <?= $can_buy ?></p>
    </body>
</html>
```

Ejemplo: Usando operadores de comparación

The screenshot shows a web browser window with the URL "localhost" in the address bar. The page itself has a dark red background with a festive candy-themed illustration on the right side featuring a large lollipop, a candy cane, and several smiling circular faces.

The Candy Store

SHOPPING CART

Item: Chocolate

Stock: 5

Wanted: 8

Can buy:

Ejemplo: Usando operadores de comparación

Sobre el ejemplo anterior, en el Paso 1, intercambia los valores en `$stock` y `$wanted`. El valor de `$can_buy` cambiará.

Ejemplo: Usando operadores lógicos

Este ejemplo se basa en el anterior.

1. Esta vez, el cliente sólo quiere **3 paquetes** de caramelos.
2. Se añade una variable llamada `$deliver` ; almacena un valor booleano para **indicar si se puede hacer o no una entrega**.
3. Una **expresión** utiliza dos operadores de comparación:
 - El primero **comprueba si hay suficientes artículos en stock**.
 - El segundo **comprueba si el artículo puede entregarse**.

Un operador lógico `&&` **comprueba si ambos operadores resultan verdaderos**. En caso afirmativo, el valor de `$can_buy` será *verdadero* y la página mostrará el número 1.

Si ambos operadores no son verdaderos, el valor de `$can_buy` será *falso* y no se mostrará nada.

Ejemplo: Usando operadores lógicos

```
<?php
$item      = 'Chocolate';
$stock     = 5;
$wanted    = 3;
$deliver   = true;
$can_buy  = (($wanted <= $stock) && ($deliver == true));
?>
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Shopping Cart</h2>
    <p>Item: <?= $item ?></p>
    <p>Stock: <?= $stock ?></p>
    <p>Ordered: <?= $wanted ?></p>
    <p>Can buy: <?= $can_buy ?></p>
  </body>
</html>
```

Ejemplo: Usando operadores lógicos

The screenshot shows a web browser window with the URL "localhost" in the address bar. The page itself has a dark red background featuring a colorful illustration of various candies and sweets. On the left side of the page, there is text related to a shopping cart item:

The Candy Store
Shopping Cart

Item: Chocolate
Stock: 5
Wanted: 3
Can buy: 1

The browser interface includes standard navigation buttons (back, forward, search, etc.) and a tab labeled "localhost".

Ejemplo: Usando operadores de comparación

Sobre el ejemplo anterior, en el Paso 1, intercambia los valores en `$stock` y `$wanted`. El valor en `$can_buy` cambiará.



Manipulación de tipos

Manipulación de tipos

El intérprete de PHP **puede convertir un valor de un tipo de datos a otro**. Esto se conoce como **manipulación de tipos** (del inglés [type juggling](#)) y **puede conducir a resultados inesperados**.

PHP es conocido como un **lenguaje débilmente tipado** porque, cuando se crea una variable, **no es necesario especificar el tipo de datos del valor que contendrá**.

A continuación, la variable `$title` contiene una cadena y luego un entero:

```
$title = 'Ten'; // String  
$title = 10; // Integer
```

Manipulación de tipos

El enfoque de PHP puede compararse con los lenguajes de programación *fuertemente tipados* (como C++ o C#), que **requieren que los programadores especifiquen el tipo de datos que tendrá cada variable cuando se declara**.

Cuando el intérprete de PHP se encuentra con un valor que no utiliza el tipo de datos que espera recibir, **puede intentar convertir el valor al tipo de datos esperado**. Este es un proceso llamado **manipulación de tipos**.

Manipulación de tipos

Las manipulaciones de tipos pueden causar confusión porque el intérprete de PHP **puede generar resultados o errores sorprendentes**. Por ejemplo, el operador de suma a continuación suma dos valores. El número 1 es un entero, pero 2 es una cadena porque está entre comillas.

```
$total = 1 + '2';
```

En este caso, el intérprete de PHP **intentará convertir automáticamente la cadena en un número para poder realizar la operación aritmética**. Como resultado, la variable `$total` contendrá el número 3.

Manipulación de tipos

Cuando el tipo de datos de un valor es cambiado a un tipo de datos diferente, los programadores decimos que el tipo de datos del valor es **convertido** (cast) de un tipo a otro. El malabarismo de tipos se conoce como **conversión implícita** porque el intérprete de PHP realiza la conversión.

Cuando un programador cambia explícitamente el tipo de datos de un valor utilizando código, se conoce como **conversión (o casting) explícito** porque se le ha dicho explícitamente al intérprete de PHP que cambie el tipo de datos.

A continuación veremos las **reglas** que especifican cómo se convierte un valor de un tipo de datos a otro.

Manipulación de tipos

Números

Cuando el intérprete de PHP espera dos números, puede realizar una operación matemática sobre ellos.

A continuación, puedes ver lo que sucede cuando:

- Se añade una cadena a un número
- Se añade un booleano a un número

Manipulación de tipos

Números + String

| NUMBER + STRING | TREATED AS | RESULT | DESCRIPTION |
|-----------------|------------|--------------|---|
| 1 + '1' | 1 + 1 | 2 (int) | String holds a valid integer. It is treated as an integer. |
| 1 + '1.2' | 1 + 1.2 | 2.2 (float) | String holds a float. It is treated as a float. |
| 1 + '1.2e+3' | 1 + 1200 | 1201 (float) | String holds a float using an e (exponent of 10). Treated as a float. |
| 1 + '5star' | 1 + 5 | 6 (int) | String holds an integer followed by other characters. The number is treated as an integer. Later characters are ignored. |
| 1 + '3.5star' | 1 + 3.5 | 4.5 (float) | String holds a float followed by other characters. The number is treated as a float. Later characters are ignored. |
| 1 + 'star9' | 1 + 0 | 1 (int) | String starts with anything other than an integer or a float. It is treated as the number 0. |

Manipulación de tipos

Números + Boolean

| NUMBER + BOOLEAN TREATED AS | RESULT | DESCRIPTION |
|-----------------------------|--------|---|
| 1 + true | 1 + 1 | 2 (int) Boolean true is treated as the integer 1. |
| 1 + false | 1 + 0 | 1 (int) Boolean false is treated as the integer 0. |

Manipulación de tipos

Cadenas (Strings)

Cuando el intérprete PHP intenta concatenar dos cadenas, seguirá estas reglas.

A continuación, podemos ver lo que ocurre cuando PHP:

- Concatena una cadena con un número.
- Concatena una cadena con un booleano.

Manipulación de tipos

String + Número

| STRING . NUMBER | TREATED AS | RESULT | DESCRIPTION |
|-----------------|----------------|------------------|---------------------------------|
| 'Hi ' . 1 | 'Hi ' . '1' | Hi 1 (string) | Integer is treated as a string. |
| 'Hi ' . 1.23 | 'Hi ' . '1.23' | Hi 1.23 (string) | Float is treated as a string. |

String + Boolean

| STRING . BOOLEAN | TREATED AS | RESULT | DESCRIPTION |
|------------------|-------------|---------------|---|
| 'Hi ' . true | 'Hi ' . '1' | Hi 1 (string) | Boolean true is treated as an integer of 1. |
| 'Hi ' . false | 'Hi ' . '' | Hi (string) | Boolean false is treated as a blank string. |

Manipulación de tipos

Booleans

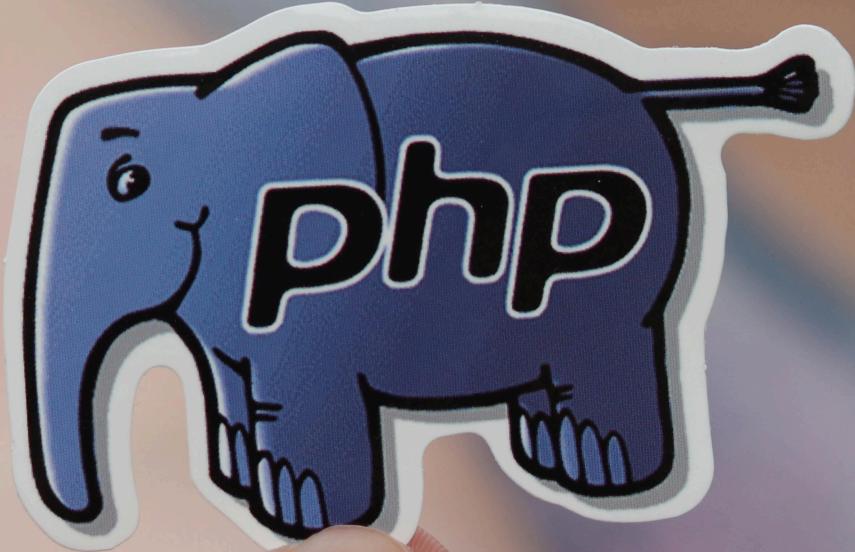
Cuando el intérprete PHP espera un valor booleano, todos los valores mostrados en la siguiente tabla serán tratados como *false*.

Cualquier otro valor es tratado como *true* (cualquier texto, un número distinto de 0, o el propio valor booleano *true*).

Manipulación de tipos

Booleans

| VALUE | DATA TYPE | TREATED AS |
|---------|--------------------------|------------|
| false | Boolean | false |
| 0 | Integer | false |
| 0.0 | Float | false |
| '0' | String with a value of 0 | false |
| '' | Empty string | false |
| array[] | Empty array | false |
| null | Null | false |



Creación de una página PHP básica

Creación de una página PHP básica

Este ejemplo reúne varias de las técnicas que se han visto en esta unidad.

El archivo PHP crea una página HTML que informa a los visitantes sobre un descuento disponible cuando compran varios paquetes de caramelos.

Creación de una página PHP básica

Comprobaremos cómo:

- Almacenar información en variables y arrays.
- Utilizar el operador de concatenación para unir texto en variables para crear un saludo personalizado para un visitante.
- Utilizar operadores aritméticos para realizar cálculos que determinen los precios que se muestran en la página.
- Escribir nuevos valores creados por el intérprete PHP en el contenido HTML de la página.

Además, si se actualizan los valores almacenados en las variables, la página reflejará automáticamente los nuevos productos y precios.

Creación de una página PHP básica

Cuando comenzamos a escribir archivos PHP, a menudo estos contienen una mezcla de código HTML y PHP. Es una buena práctica separar este código tanto como sea posible:

- **Comenzaremos utilizando PHP para crear los valores que se mostrarán en la página HTML, y almacenarlos en variables.** (A la derecha, esto está por encima de la línea horizontal punteada.)
- Luego, la parte inferior de la página puede **centrarse en el contenido HTML**. El **código PHP sólo debe utilizarse en esta parte de la página para mostrar los valores** que se han almacenado en las variables.

Creación de una página PHP básica

En este ejemplo, el código PHP al principio de la página:

1. Comienza declarando una variable para guardar el **nombre de usuario** del visitante.
Se llama `$username` porque el nombre de una variable debe empezar siempre con el símbolo del dólar, seguido de un nombre que describa el tipo de datos que contiene.
2. Se declara una variable llamada `$greeting` para contener un **saludo para el visitante**. Para ello se utiliza el operador de concatenación para unir la cadena "Hello," y el nombre del visitante.

Creación de una página PHP básica

3. Se crea una variable llamada `$offer` para guardar los detalles de un artículo que está en oferta especial. Su valor es un array con cuatro elementos:

- El artículo en oferta
- La cantidad que deben comprar
- El precio normal por paquete (sin descuento)
- El precio con descuento por paquete

El primer elemento, que describe el artículo en oferta, utiliza un tipo de datos **cadena de caracteres (string)**. Los demás valores son **enteros**.

Creación de una página PHP básica

4. Se crea una variable llamada `$usual_price`. Su valor es el **coste de los artículos sin descuento**. Se calcula multiplicando dos de los valores almacenados en el array: la *cantidad* y el *precio*.
5. Se crea una variable llamada `$offer_price`. Su valor es el **coste de los artículos con el descuento aplicado**. Se calcula multiplicando la *cantidad* y el *precio con descuento* almacenados en el array.
6. Se crea una variable llamada `$saving` para **guardar el ahorro total del cliente**. Se calcula restando el valor almacenado en la variable `$offer_price` (creada en el paso 5) del valor almacenado en `$usual_price` (creada en el paso 4).

Creación de una página PHP básica

```
<?php
$username = 'Ivy';                                // Variable to hold username

$greeting = 'Hello, ' . $username . '.';           // Greeting is 'Hello' + username

$offer = [
    'item'      => 'Chocolate',                    // Create array to hold offer
    'qty'        => 5,                            // Item on offer
    'price'      => 5,                            // Quantity to buy
    'discount'   => 4,                            // Usual price per pack
    'discount'   => 4,                            // Offer price per pack
];
$usual_price = $offer['qty'] * $offer['price'];       // Usual total price
$offer_price = $offer['qty'] * $offer['discount'];    // Offer total price
$saving      = $usual_price - $offer_price;          // Total saving
?>
```

Creación de una página PHP básica

La segunda mitad de la página creará el HTML que se envía al navegador. Comienza con la declaración *HTML DOCTYPE*. PHP sólo se utiliza para escribir los valores que se almacenaron en variables en los pasos anteriores:

7. El saludo, que es la palabra "Hello" seguida del nombre del visitante, se escribe en la página utilizando la abreviatura del comando `echo`.
8. El **ahorro total**, que se almacena en la variable `$saving` (creada en el paso 6) se muestra en un círculo amarillo. Se utiliza CSS para colocar este círculo en la esquina superior derecha de la ventana del navegador.
9. Un párrafo explica los **detalles de la oferta**. Muestra la cantidad de caramelos que el visitante tiene que comprar y el nombre de los caramelos.
10. A continuación se muestra el precio con descuento almacenado en `$offer_price` y el precio habitual en `$usual_price`.

Creación de una página PHP básica

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>

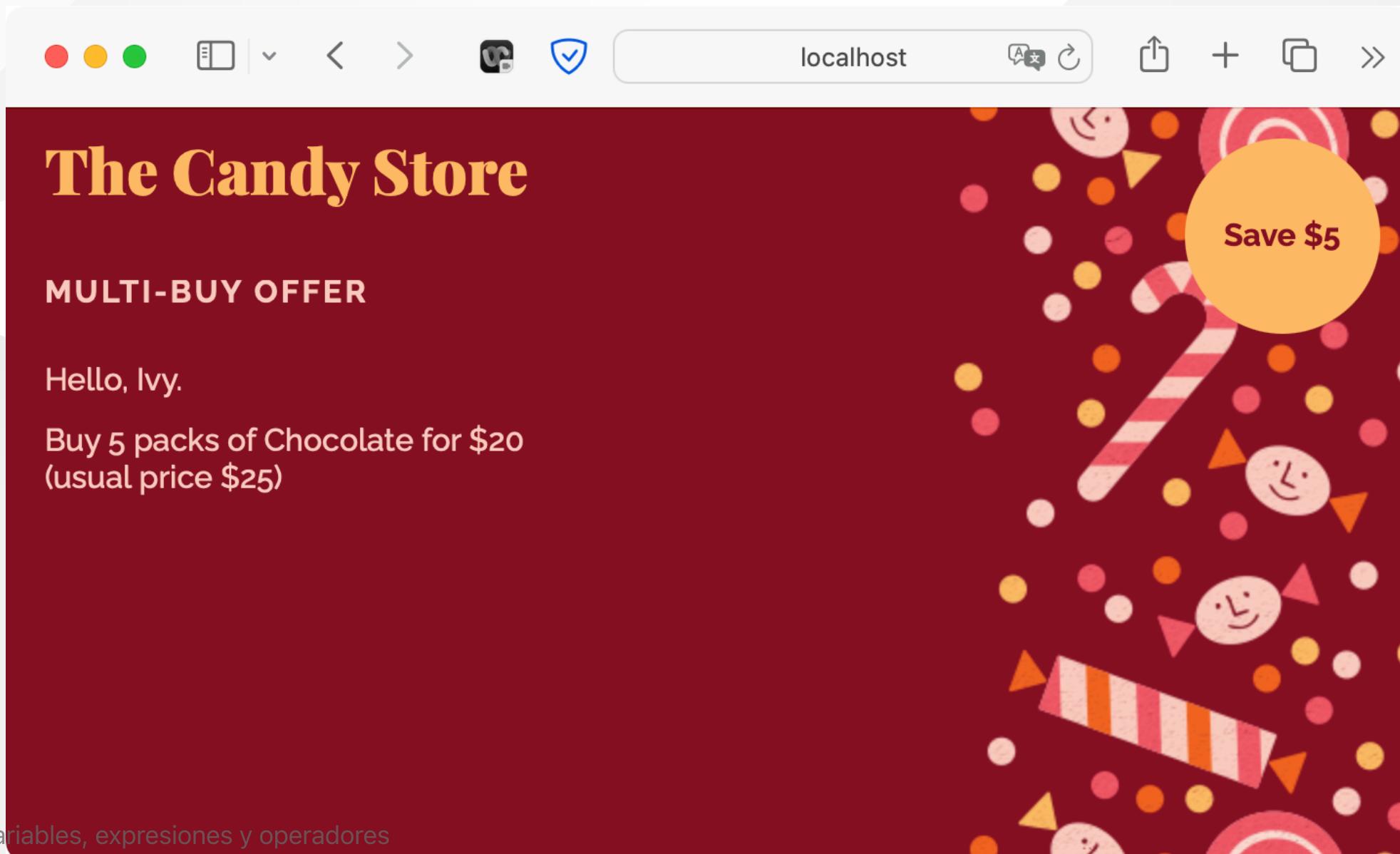
    <h2>Multi-buy Offer</h2>

    <p><?= $greeting ?></p>

    <p class="sticker">Save $<?= $saving ?></p>

    <p>Buy <?= $offer['qty'] ?> packs of <?= $offer['item'] ?>
      for $<?= $offer_price ?><br> (usual price $<?= $usual_price ?>)</p>
  </body>
</html>
```

Creación de una página PHP básica



Creación de una página PHP básica

Sobre la página de ejemplo, en el Paso 1, cambia el nombre de usuario por tu nombre.

En el Paso 2, actualiza el saludo que se muestra al visitante para que diga "Hi" (en lugar de "Hello").

En el paso 3, actualiza el número de paquetes de caramelos en la clave *qty* del array

`$offer` a 3.

En el paso 3, actualiza el precio de los caramelos a 6.

Actividad propuesta

Desarrolla una página similar al ejemplo que hemos estudiado pero con temática diferente, aplicando los conceptos sobre variables, expresiones y operadores de PHP que hemos estudiado en esta unidad.

Algunos ejemplos de temática que podríais utilizar para este desarrollo pueden ser:

- **Promoción de un restaurante:** Desarrolla una página web para un restaurante que muestre una oferta especial en un menú.
- **Rebajas en una tienda de ropa:** Crea una página web para una tienda de ropa que muestra una oferta en artículos de vestir.

Actividad propuesta

- **Promoción de una librería:** Desarrolla una página web para una librería que muestra una oferta en libros.
- **Oferta de un gimnasio:** Crea una página web para un gimnasio que muestra una oferta en membresías.
- **Promoción de electrónica (*black friday*):** Desarrolla una página web para una tienda de electrónica que muestra una oferta en cualquier tipo de electrónica.



Resumen de la unidad

Resumen de la unidad

- Las **variables** almacenan datos que varían cada vez que se ejecuta un script.
- Los **tipos de datos escalares** contienen texto, números enteros, números de coma flotante y valores booleanos de verdadero o falso.
- Un **array** es un tipo de datos compuesto, utilizado para almacenar un conjunto de valores relacionados.
- Cada ítem de un array se denomina **elemento**. Los elementos de un array **asociativo** tienen una clave y un valor. Los elementos de un array **indexado** tienen un número de índice y un valor.

Resumen de la unidad

- Los **operadores de cadena** unen (concatenan) texto en cadenas.
- Los **operadores matemáticos** realizan operaciones matemáticas utilizando números.
- Los **operadores de comparación** comparan dos valores para ver si uno es igual, mayor o menor que el otro.
- Los **operadores lógicos** pueden combinar los resultados de varias expresiones utilizando y, o y no.





Referencias

Referencias

- [PHP Manual \(php.net\)](#)
- [Tutoriales sobre PHP \(w3schools\)](#)
- [Documentación de PHP sobre operadores \(php.net\)](#)

Bloque A

Instrucciones básicas de programación

Unidad 1. Variables, expresiones y operadores

