

Bloque A

Instrucciones básicas de programación

Unidad 2. Estructuras de control



Contenidos

1. Introducción
2. Sentencias condicionales
3. Bucles
4. Archivos *include*
5. Ejemplo final
6. Resumen de la unidad



Introducción

Introducción

Esta unidad muestra cómo indicar al intérprete PHP:

- **si debe ejecutar un bloque de código o no**
- **cuándo repetir un conjunto de sentencias**
- **cuándo incluir código de otro archivo.**

Introducción

Existen tres formas de controlar cuándo el intérprete de PHP ejecutará las sentencias de un archivo PHP:

- **Secuencia:** El intérprete PHP ejecuta las sentencias en el orden en que están escritas. Línea 1, línea 2, línea 3, y así sucesivamente hasta llegar a la última línea. Todos los ejemplos que hemos visto hasta ahora en el libro ejecutan el código en secuencia.
- **Selección:** El intérprete de PHP utiliza una condición para comprobar si se ejecutará o no algún código. Por ejemplo, la condición podría ser: "¿Ha iniciado sesión el usuario?". Si la respuesta es "no" podría mostrar un enlace a la página de inicio de sesión. Si la respuesta es "sí", podría mostrar un enlace a la página de perfil del usuario. Los programadores llaman a estas instrucciones sentencias condicionales porque seleccionan qué conjunto de sentencias ejecutar en función de una condición.

Introducción

- **Repetición / iteración:** El intérprete de PHP puede repetir el mismo conjunto de código varias veces. Por ejemplo, si un array contuviera una lista de la compra, las mismas instrucciones podrían ejecutarse para cada elemento de la lista (tanto si contiene 1 como 100 elementos). Los bucles se utilizan para repetir conjuntos de instrucciones.

Introducción

Cuando cambias el orden en el que se ejecutan las sentencias, estás **cambiando el flujo de control**.

En esta unidad, también aprenderás **cómo utilizar archivos *include*** (*include files*) para contener código que es utilizado por múltiples páginas. Este método te permite **incluir un archivo en varias páginas en lugar de repetir el mismo código** una y otra vez en cada archivo.



Sentencias condicionales

Sentencias condicionales

Las **sentencias condicionales** prueban una **condición** para determinar si se ejecuta o no un bloque de código. Son similares a decir: "*Si esta situación es cierta, ejecuta la tarea 1 (y opcionalmente, si no, ejecuta la tarea 2)*".

Algunas tareas sólo se realizan **si se cumple una condición**. Consideremos un sitio web en el que los usuarios pueden iniciar sesión. Si el usuario

- **ha iniciado sesión**, se muestra un enlace a su página de perfil.
- **no ha iniciado sesión**, se muestra un enlace a la página de inicio de sesión.

En este caso, la condición sería: "*¿Ha iniciado sesión el usuario?*" y se utiliza para determinar qué enlace mostrar.

Sentencias condicionales

Las condiciones son expresiones que siempre dan como resultado un valor *verdadero* o *falso*. A menudo utilizan **operadores de comparación** para comparar dos valores.

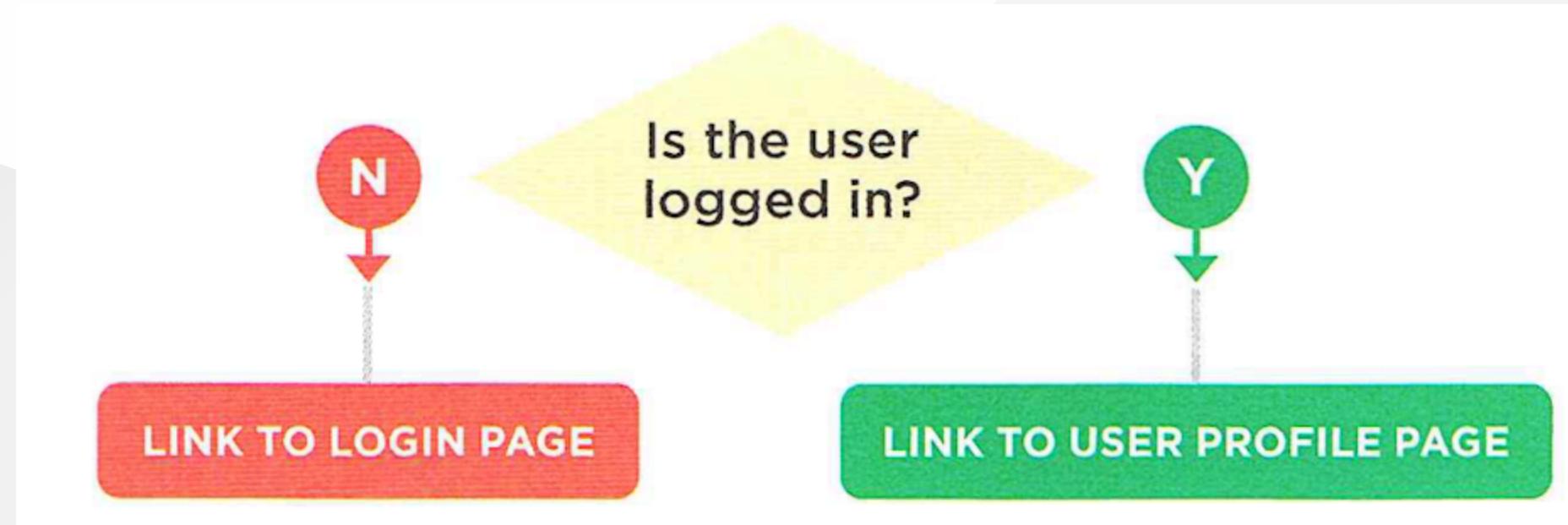
Si una variable llamada `$logged_in` almacena un valor de *true* cuando un usuario está logueado y *false* cuando no lo está, lo siguiente actuará como una condición:

```
($logged_in === true)
```

Si la condición resulta en un valor de:

- *true* podría ejecutar sentencias que muestren un enlace a la página de perfil del usuario.
- Si el valor es *false*, podría ejecutar sentencias que muestren un enlace a la página de inicio de sesión.

Sentencias condicionales



Sentencias condicionales

IF

Una sentencia `if` sólo ejecuta un conjunto de sentencias si se cumple una condición.

Las sentencias a utilizar si se cumple la condición están rodeadas de **llaves**.

Si la condición no se cumple, las sentencias entre llaves **se omiten** y el intérprete de PHP pasa a la siguiente línea de código.

```
if ($logged_in === true) {  
    // Sentencias a ejecutar si la condición se cumple  
}
```

Sentencias condicionales

IF...ELSE

Una sentencia `if... else` comprueba una condición.

- Si devuelve verdadero, **se ejecuta el primer conjunto** de sentencias.
- En caso contrario, se ejecuta el segundo conjunto de sentencias.

También se utiliza el **operador ternario** para abbreviar una sentencia `if... else`.

```
if ($logged_in === true) {  
    // Sentencias a ejecutar si la condición se cumple  
} else {  
    // Sentencias a ejecutar si la condición no se cumple  
}
```

Sentencias condicionales

IF...ELSEIF...

En una sentencia `if... elseif`, si no se cumple la primera condición, puede añadir una segunda. Las sentencias que siguen a la segunda condición **sólo se ejecutan si se cumple la segunda condición.**

Puedes proporcionar un conjunto predeterminado de sentencias que **se ejecuten si no se cumple ninguna de las dos condiciones** utilizando la opción `else` al final.

```
if ($logged_in === true) {  
    // Sentencias a ejecutar si la condición 1 se cumple  
} elseif ($time > 12) {  
    // Sentencias a ejecutar si la condición 1 no se cumple y la condición 2 se cumple  
} else {  
    // Sentencias que se ejecutan si no se cumplen ninguna de las condiciones previas  
}
```

Sentencias condicionales

SWITCH

Una sentencia `switch` no depende de una condición; **se especifica una variable** y luego se proporcionan **opciones** que podrían **coincidir con el valor de la variable**.

Si ninguna de las opciones coincide, un **conjunto predeterminado de sentencias** puede ser ejecutado en su lugar (valor por defecto).

Si no hay ningún valor por defecto y ninguna coincidencia, el intérprete de PHP **pasa a la siguiente línea después de la sentencia switch**.

Sentencias condicionales

SWITCH

```
switch ($option) {  
    case 'option_1':  
        // Sentencias a ejecutar  
        break;  
    case 'option_2':  
        // Sentencias a ejecutar  
        break;  
    default:  
        // Sentencias a ejecutar por defecto  
}
```

Sentencias condicionales

MATCH

PHP 8 añadió la expresión `match` (una variación de la sentencia `switch`). Si se encuentra una **coincidencia exacta para una variable** (es el **mismo valor y tipo de datos** que la variable) entonces se ejecuta una expresión y se devuelve el valor que crea la expresión.

Se pueden especificar **múltiples opciones en una línea**, así como proporcionar un **valor por defecto** en caso de que ningún valor coincida.

Sin embargo, **si no hay ninguna coincidencia y ningún valor por defecto**, se producirá un **error**.

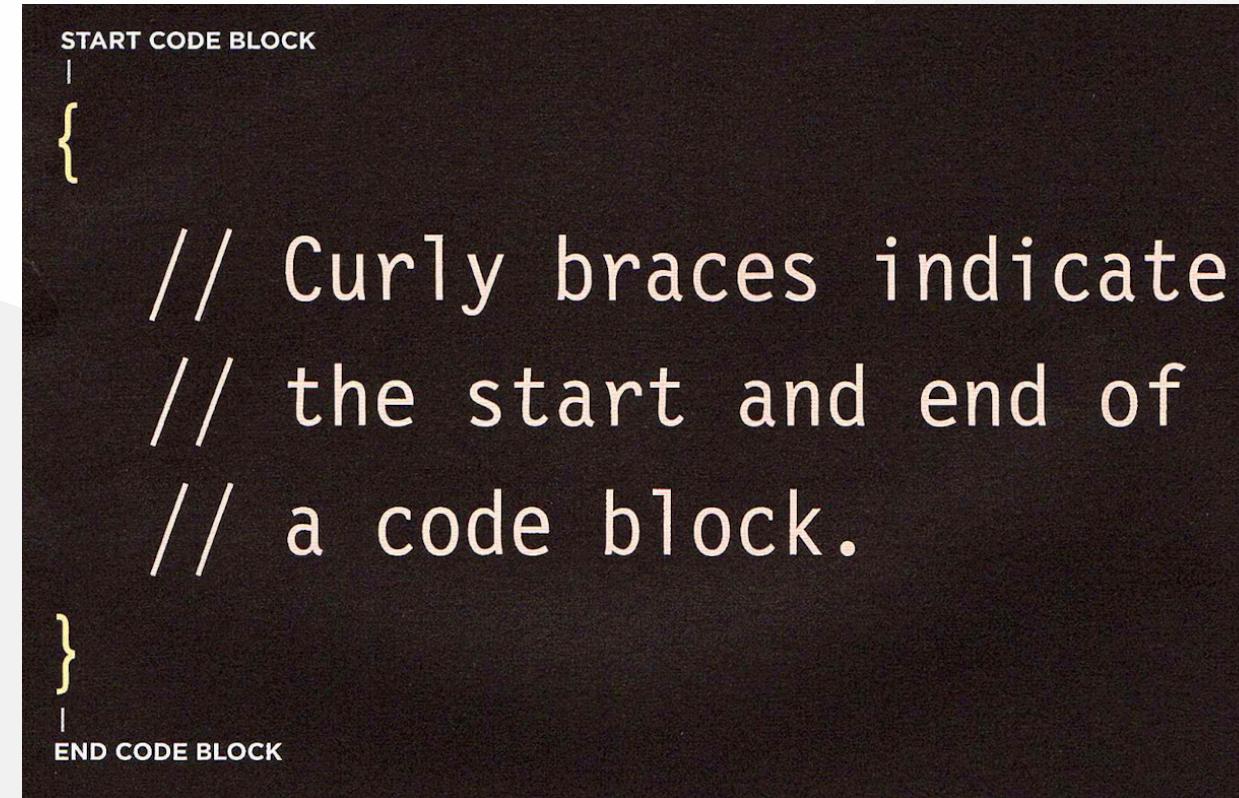
Sentencias condicionales

MATCH

```
$result = match($option) {  
    'option_1'          => /* Expresión */,  
    'option_2', 'option_3' => /* Expresión */,  
    default             => /* Expresión */,  
};
```

Las llaves ({}) forman bloques de código

PHP utiliza llaves para rodear un conjunto de sentencias relacionadas. Las llaves y las sentencias dentro de ellas forman un **bloque de código**.



Las llaves ({}) forman bloques de código

Las llaves permiten al intérprete de PHP saber dónde empieza y termina un bloque de código:

- Una llave izquierda o de apertura { indica el comienzo de un bloque de código.
- Una llave derecha o de cierre } indica el final de un bloque de código.

No hay límite en el número de sentencias que pueden aparecer dentro de las llaves de un bloque de código.

Los bloques de código permiten al intérprete de PHP **ejecutar, omitir o repetir las sentencias que contienen.**

Las llaves ({}) forman bloques de código

No es estrictamente necesario que exista un punto y coma después de la llave } de cierre al final de un bloque de código porque los bloques de código sólo indican dónde comienza y termina un conjunto de sentencias relacionadas; el bloque de código en sí mismo no es una instrucción que el intérprete PHP ejecute.

Un punto y coma es necesario cuando el bloque de código forma parte de una instrucción que requiere un punto y coma al final, como una asignación o una declaración de una función anónima.

Las llaves ({}) forman bloques de código

Algunos ejemplos que ilustran cuando es obligatorio el uso del ; y cuando no es necesario:

```
<?php
if (true) {
    echo "Esto es verdadero";
} // No se necesita punto y coma aquí

function saludar() {
    echo "¡Hola!";
} // No se necesita punto y coma aquí

$saludo = function() {
    return "¡Hola, de nuevo!";
}; // Punto y coma necesario porque es una asignación

echo $saludo(); // Llama a la función anónima e imprime "¡Hola, de nuevo!"
?>
```

Estructura de las sentencias condicionales

Una condición **siempre resulta en, o se evalúa en, un valor booleano** de verdadero o falso. El resultado determina qué bloque de código se ejecuta.

La siguiente condición comprueba si el valor de la variable `$logged_in` es verdadero:

- Si lo es, la condición resulta en un valor de true.
- Si no lo es, el resultado es falso.

Estructura de las sentencias condicionales

```
CONDITION TO TEST
if ($logged_in === true) {
    $link = '<a href="member.php">My Profile</a>';
} else {
    $link = '<a href="login.php">Login</a>';
}
```

CODE TO EXECUTE IF VALUE IS TRUE

CODE TO EXECUTE IF VALUE IS FALSE

Estructura de las sentencias condicionales

Cuando el resultado de la condición es **verdadero**, se ejecuta el **primer bloque de código**. A continuación, el intérprete de PHP **ignora la palabra clave else** y se salta el segundo bloque de código. Luego, **se mueve a la primera línea de código después de la sentencia condicional**.

Cuando el resultado de la condición es **falso**, el intérprete de PHP **omite el primer bloque de código** y se mueve a la palabra clave **else**. Entonces ejecuta las sentencias del bloque de código que le siguen. A continuación, **se mueve a la primera línea de código después de la sentencia condicional**.

Ejemplo: Sentencia if

Este ejemplo muestra un **saludo personalizado si un visitante ha iniciado sesión**.

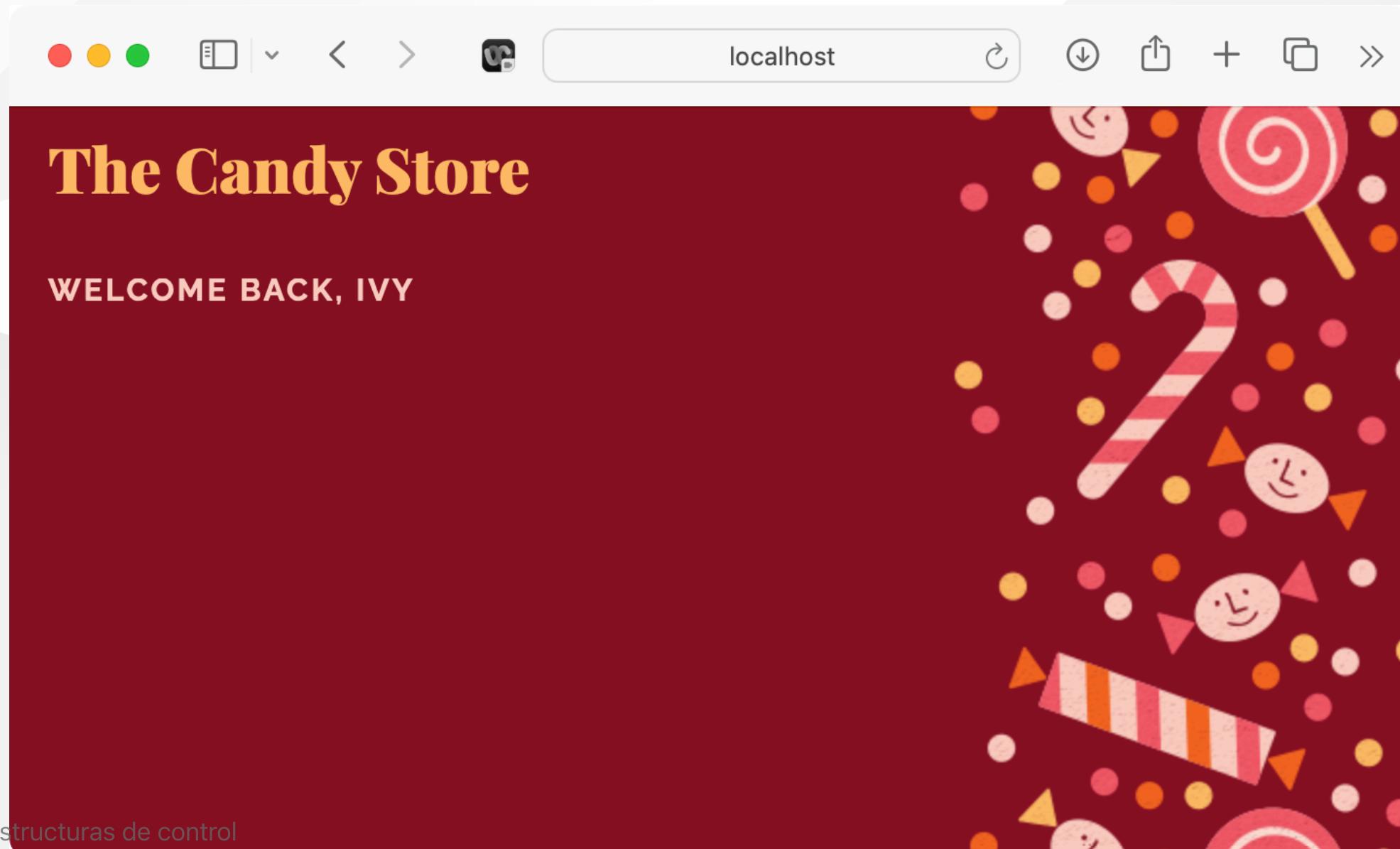
Comienza creando dos variables y almacenando valores en ellas:

1. \$name contiene el nombre del visitante.
2. La variable \$greeting se inicializa; lo que significa que se le da un valor inicial que se utilizará si no se actualiza en los Pasos 3 y 4. El saludo dice simplemente "Hello".
3. Una sentencia if utiliza una **condición** para comprobar si la variable \$name no es una cadena vacía.
Si la variable no está vacía entonces se ejecuta el bloque de código siguiente.
4. El valor de la variable \$greeting se actualiza para decir "Welcome back, "seguido del nombre del visitante.
5. Finalmente, el valor almacenado en \$greeting se escribe en la página.

Ejemplo: Sentencia if

```
<?php  
① $name      = 'Ivy';  
② $greeting = 'Hello';  
  
③ if ($name !== '') {  
④     $greeting = 'Welcome back, ' . $name;  
}  
?  
<!DOCTYPE html>  
<html>  
    <head> ... </head>  
    <body>  
        <h1>The Candy Store</h1>  
⑤        <h2><?= $greeting ?></h2>  
    </body>  
</html>
```

Ejemplo: Sentencia if



Ejemplo: Sentencia if

Es importante saber que una condición puede contener **sólo un nombre de variable**, por ejemplo:

```
if ($name) {  
    $greeting = 'Hi, ' . $name;  
}
```

Esta condición comprobaría si el valor almacenado en la variable `$name` sería tratado como *veradero* después de que se haya producido la **manipulación de tipos** que vimos en la unidad anterior.

Como ya sabemos, una cadena que contenga cualquier texto, o un número distinto de 0, se trataría como un valor de *true*.

Ejemplo: Sentencia if

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 1, cambia el valor del saludo `$name` para almacenar una cadena vacía.
Actualiza la página y observa cómo cambia el saludo.

Ejemplo: sentencia if...else

Este ejemplo comprueba el nivel de existencias de un artículo y muestra el mensaje correspondiente.

1. Una variable llamada `$stock` contiene el número de artículos en stock.
2. Una sentencia `if` utiliza una condición para comprobar si la cantidad almacenada en `$stock` es mayor que 0.
3. Si la condición resulta *verdadera*, una variable llamada `$message` recibe el valor "*In stock*".

El intérprete PHP omite entonces la palabra clave `else` y el bloque de código subsiguiente.

Ejemplo: sentencia if...else

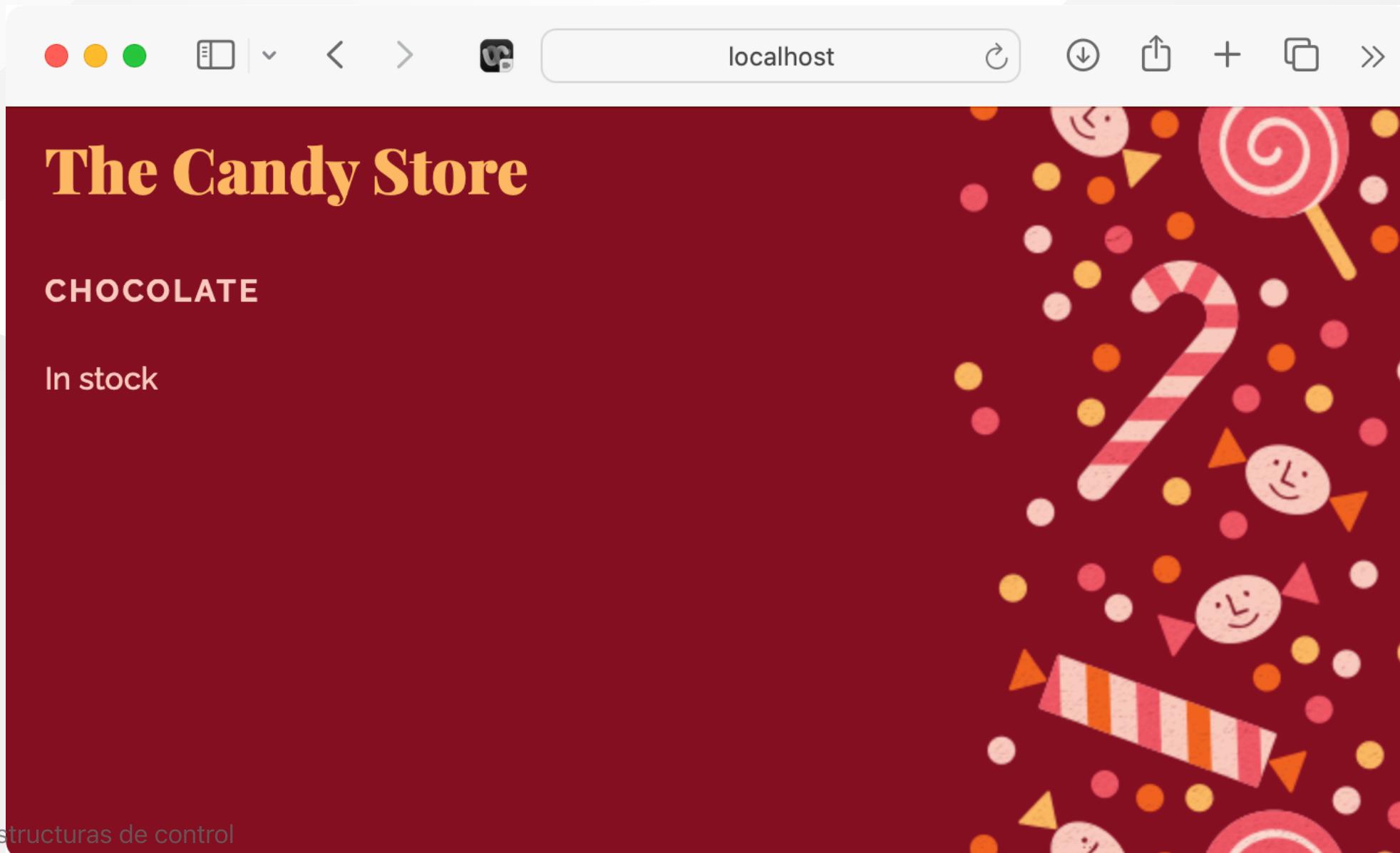
4. Si la condición del paso 2 resulta *falsa*, la palabra clave `else` indica al intérprete de PHP que ejecute el bloque de código que le sigue.
5. A una variable llamada `$message` se le da el valor de "Sold out".
6. El valor almacenado en la variable `$message` se escribe en la página.

Ejemplo: sentencia if...else

```
<?php
$stock = 5;

if ($stock > 0) {
    $message = 'In stock';
} else {
    $message = 'Sold out';
}
?>
<!DOCTYPE html>
<html>
    <head>
        <title>if else Statement</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Chocolate</h2>
        <p><?= $message ?></p>
    </body>
</html>
```

Ejemplo: sentencia if...else



Ejemplo: sentencia if...else

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el paso 1, cambia el valor almacenado en `$stock` a 0.
- En el paso 5, cambia el mensaje a "*More stock coming soon*".

Operador ternario

Los operadores ternarios **comprueban una condición y proporcionan un valor para utilizar si la condición resulta verdadera y otro valor si resulta falsa.**

Suelen emplearse como una versión abreviada de una sentencia `if...else`.

Operador ternario

La siguiente condición de la sentencia `if...else` comprueba si la edad del usuario es inferior a 16 años.

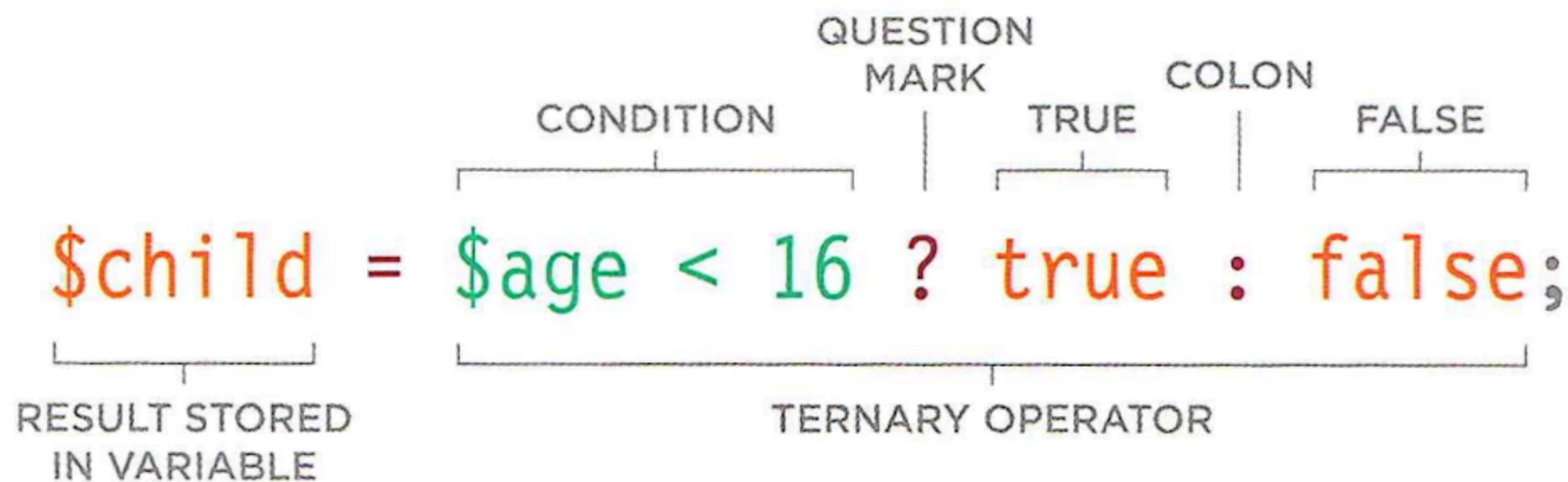
Si la condición resulta

- *true* entonces se asigna a `$child` un valor de *true*.
- *false* entonces se asigna a `$child` un valor de *false*.

```
if ($age < 16) {  
    $child = true;  
} else {  
    $child = false;  
}
```

Operador ternario

A continuación, se puede ver cómo un operador ternario hace esto en sólo una línea de código.



Operador ternario

Un **signo de interrogación** (`?`) separa la condición que se comprueba de los valores que se utilizarán.

Dos puntos (`:`) separan el valor que se devuelve si la condición resulta verdadera del valor que se devuelve si la condición resulta falsa.

Aquí, el resultado devuelto por el operador ternario se almacena en una variable llamada `$child` .

A veces se colocan **paréntesis alrededor de la condición** para mostrar que dará como resultado un único valor, aunque no son necesarios.

Ejemplo: Operador ternario

Este ejemplo reproduce el anterior pero utiliza un operador ternario (en lugar de una sentencia `if...else`).

1. Una variable llamada `$stock` contiene el número de artículos en stock.
2. Se utiliza un operador ternario para asignar un valor a la variable `$message` .

La condición comprueba si el valor en `$stock` es mayor que 0. Si esta condición se evalúa como:

- `true` entonces En existencia se almacena en `$message` .
- `false` entonces Agotado se almacena en `$message` .

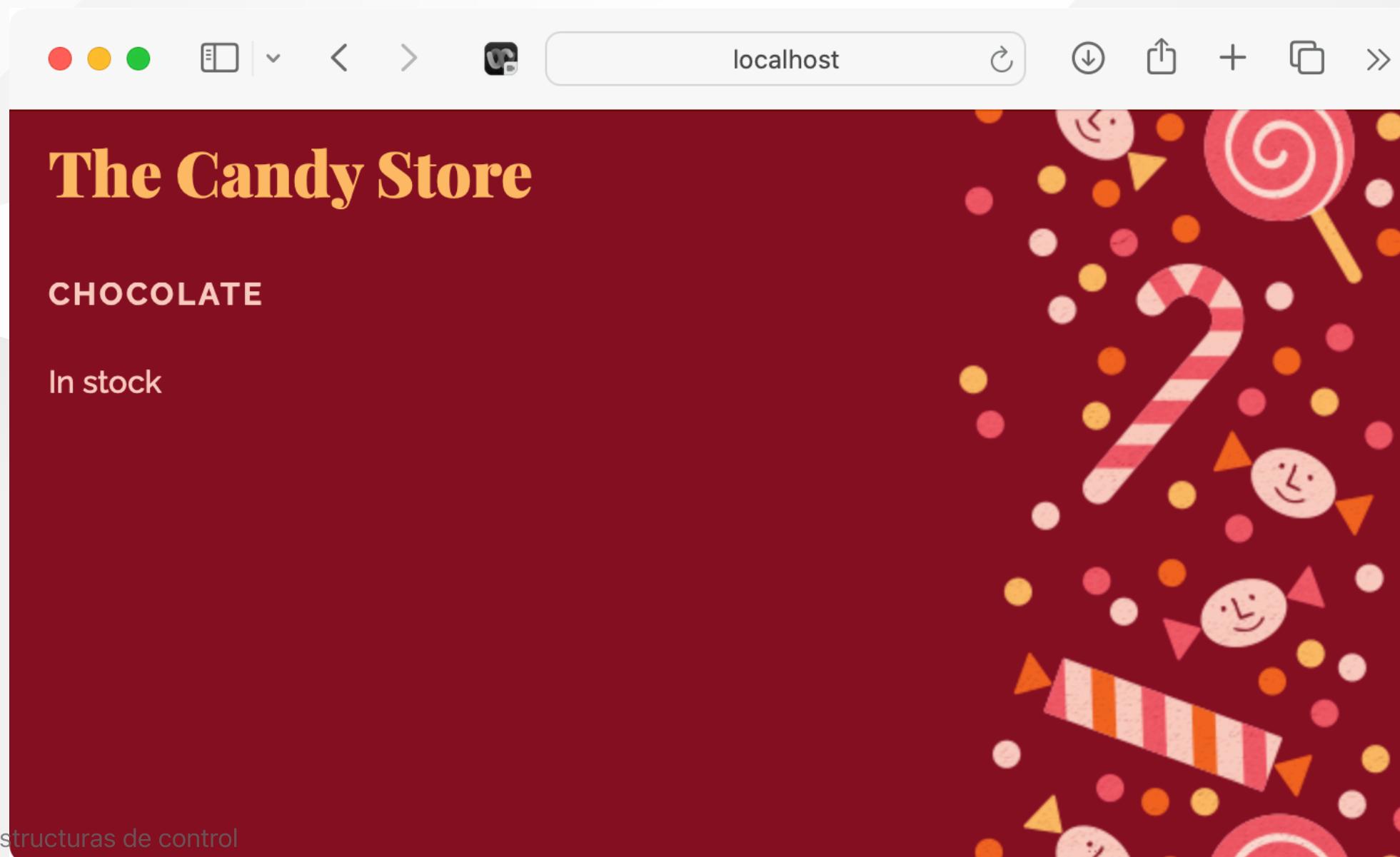
3. Por último, el valor almacenado en la variable `$message` se escribe en la página.

Ejemplo: Operador ternario

```
<?php
$stock    = 5;

$message = ($stock > 0) ? 'In stock' : 'Sold out';
?>
<!DOCTYPE html>
<html>
  <head>
    <title>Ternary Operator</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Chocolate</h2>
    <p><?= $message ?></p>
  </body>
</html>
```

Ejemplo: Operador ternario



Ejemplo: Operador ternario

Sobre el ejemplo anterior:

- En el Paso 1, cambia el valor almacenado en `$stock` a 0.
- En el paso 2, cambia el mensaje a "*More stock coming soon*".

Ejemplo: Sentencias if...elseif

Este ejemplo se basa en los anteriores.

1. Una variable llamada `$ordered` indica el número de artículos que la tienda ha pedido para reponer sus existencias.
2. Una sentencia `if` utiliza una condición para comprobar si el valor en `$stock` es mayor que 0. Si lo es, entonces una variable llamada `$message` contendrá "*In stock*", y el intérprete de PHP se mueve al final de la sentencia `if... elseif`.
3. Si la primera condición no se cumple, una sentencia `elseif...` utiliza una segunda condición para comprobar si el valor en `$ordered` es mayor que 0. Si lo es, la variable llamada `$message` contendrá "*Coming soon*", y el intérprete de PHP se mueve al final de la sentencia `if... elseif`.

Ejemplo: Sentencias if...elseif

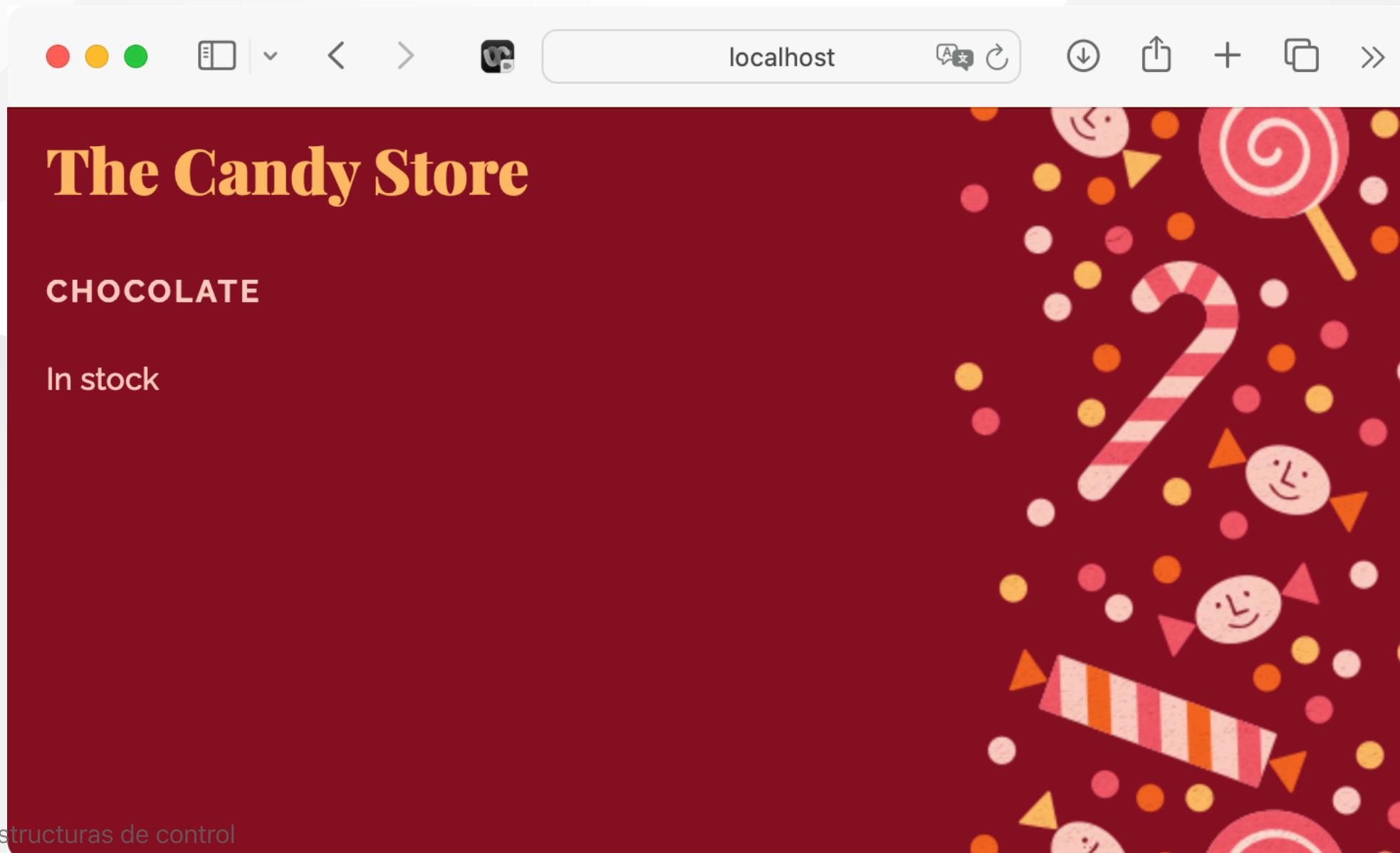
4. Si ninguna de las dos condiciones resultó en un valor de true, el intérprete de PHP ejecuta la cláusula else y el bloque de código que le sigue, que almacena el mensaje "Sold out" en la variable \$message .
5. Por último, el valor almacenado en \$message se escribe en la página.

Ejemplo: Sentencias if...elseif

```
<?php
$stock    = 5;
① $ordered = 3;

② if ($stock > 0) {
    $message = 'In stock';
}
③ elseif ($ordered > 0) {
    $message = 'Coming soon';
}
④ else {
    $message = 'Sold out';
}
?>
<!DOCTYPE html>
<html>
    <head> ... </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Chocolate</h2>
⑤        <p><?= $message ?></p>
    </body>
</html>
```

Ejemplo: Sentencias if...elseif



Ejemplo: Sentencias if...elseif

Sobre el ejemplo anterior:

- En el paso 1, cambia el valor de la variable `$stock` a 0. Cuando actualices la página, el mensaje debería decir "*Coming soon*".

Ejemplo: sentencias switch

1. Se establece una variable llamada `$day` para contener un día de la semana.
2. La sentencia `switch` comienza con el comando `switch` y un nombre de variable entre paréntesis. La variable contiene lo que se conoce como el valor del switch. Esto es seguido por un par de llaves que contienen opciones que pueden coincidir con el valor del switch.
3. Hay dos opciones. Ambas:
 - Empiezan con la palabra `case`
 - Seguido de un valor
 - Despues dos puntos (`:`)

Ejemplo: sentencias switch

4. Si el valor del switch coincide con una opción, se ejecutan las sentencias que le siguen. (Establecen valores para la variable llamada `$offer`).
5. `break` le dice al intérprete PHP que vaya al final de la sentencia `switch` .
6. La última opción es `default` , a la cual le siguen las sentencias a ejecutar si ninguna de las opciones anteriores coincide. (No hay `break` después de la opción `default`).
7. Por último, en el HTML se muestra el valor de `$offer` .

Ejemplo: sentencias switch

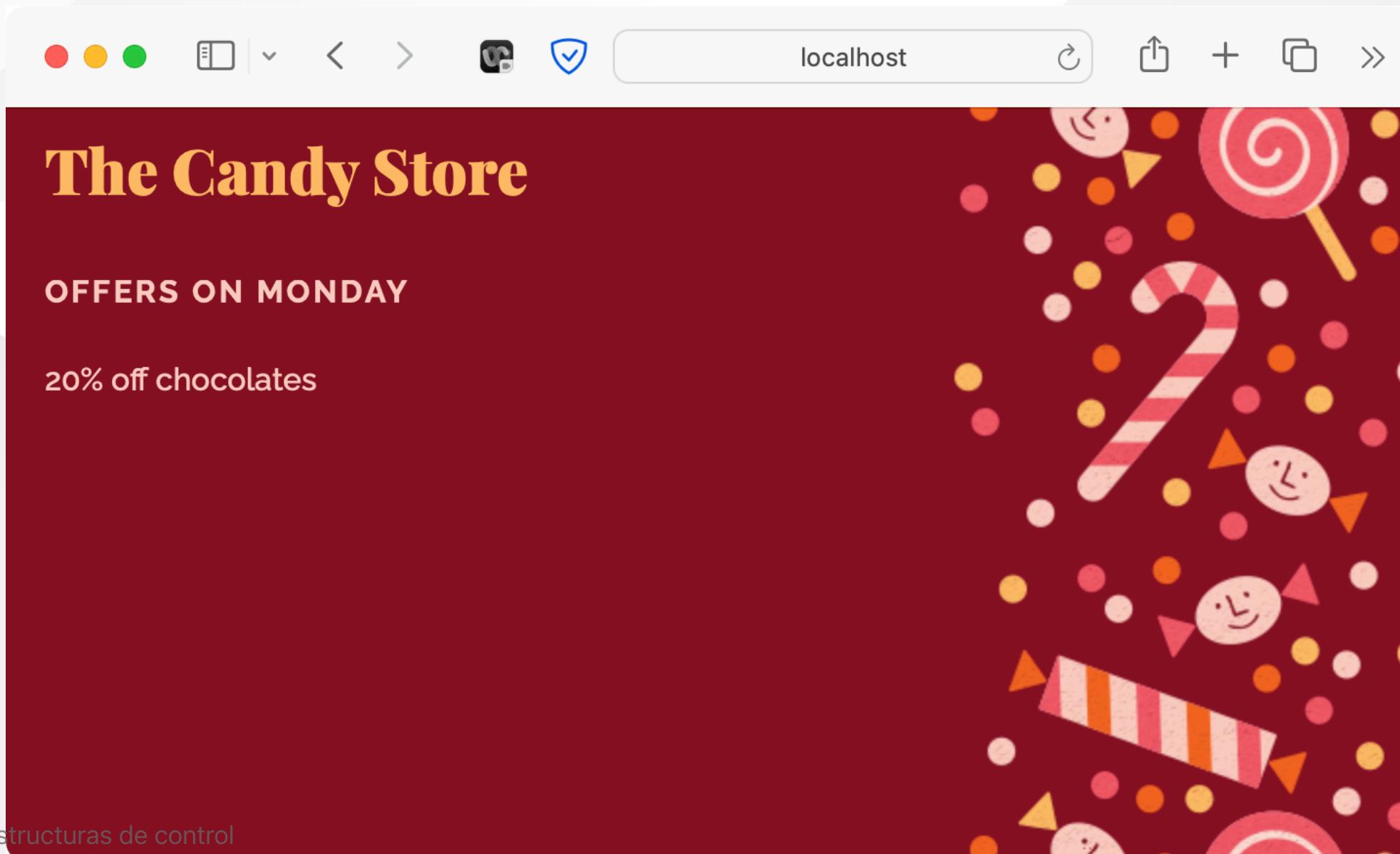
```
<?php
$day = 'Monday';

switch ($day) {
    case 'Monday':
        $offer = '20% off chocolates';
        break;
    case 'Tuesday':
        $offer = '20% off mints';
        break;
    default:
        $offer = 'Buy three packs, get one free';
}
?>
```

Ejemplo: sentencias switch

```
<!DOCTYPE html>
<html>
  <head>
    <title>switch Statement</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Offers on <?= $day; ?></h2>
    <p><?= $offer ?></p>
  </body>
</html>
```

Ejemplo: sentencias switch



Ejemplo: sentencias switch

Sobre el ejemplo anterior:

- En el Paso 1, cambia el valor a "*Wednesday*".
- Después del paso 5, añade una opción a la sentencia `switch` para "*Wednesday*".

Ejemplo: Expresión match

NOTA: Este ejemplo sólo funciona con PHP 8+.

1. Una variable llamada `$day` se establece para contener un día de la semana.
2. Se utiliza una expresión `match` para asignar un valor a la variable `$offer`. Comienza con la palabra `match`, seguida de un paréntesis que contiene el nombre de una variable, y luego una llave de apertura.
3. Las llaves contienen brazos que comienzan con valores que se comprueban para ver si coinciden con el valor almacenado en la variable `$day`.
 - Si se encuentra una coincidencia, se ejecuta la expresión a la derecha del operador de doble flecha.
 - Cada brazo sólo puede ejecutar una expresión y termina con una coma. Además, observa que la expresión `match` utiliza una comparación de tipos **estricta**; no realizará manipulación de tipos.

Ejemplo: Expresión match

4. El último brazo utiliza el valor por defecto (`default`), seguido de una expresión a ejecutar si no hay coincidencia. (Si no hay ninguna coincidencia y ningún brazo por defecto, se produce un error).
5. Finalmente se muestra el valor contenido en `$offer` en la parte de la página que construye el código HTML.

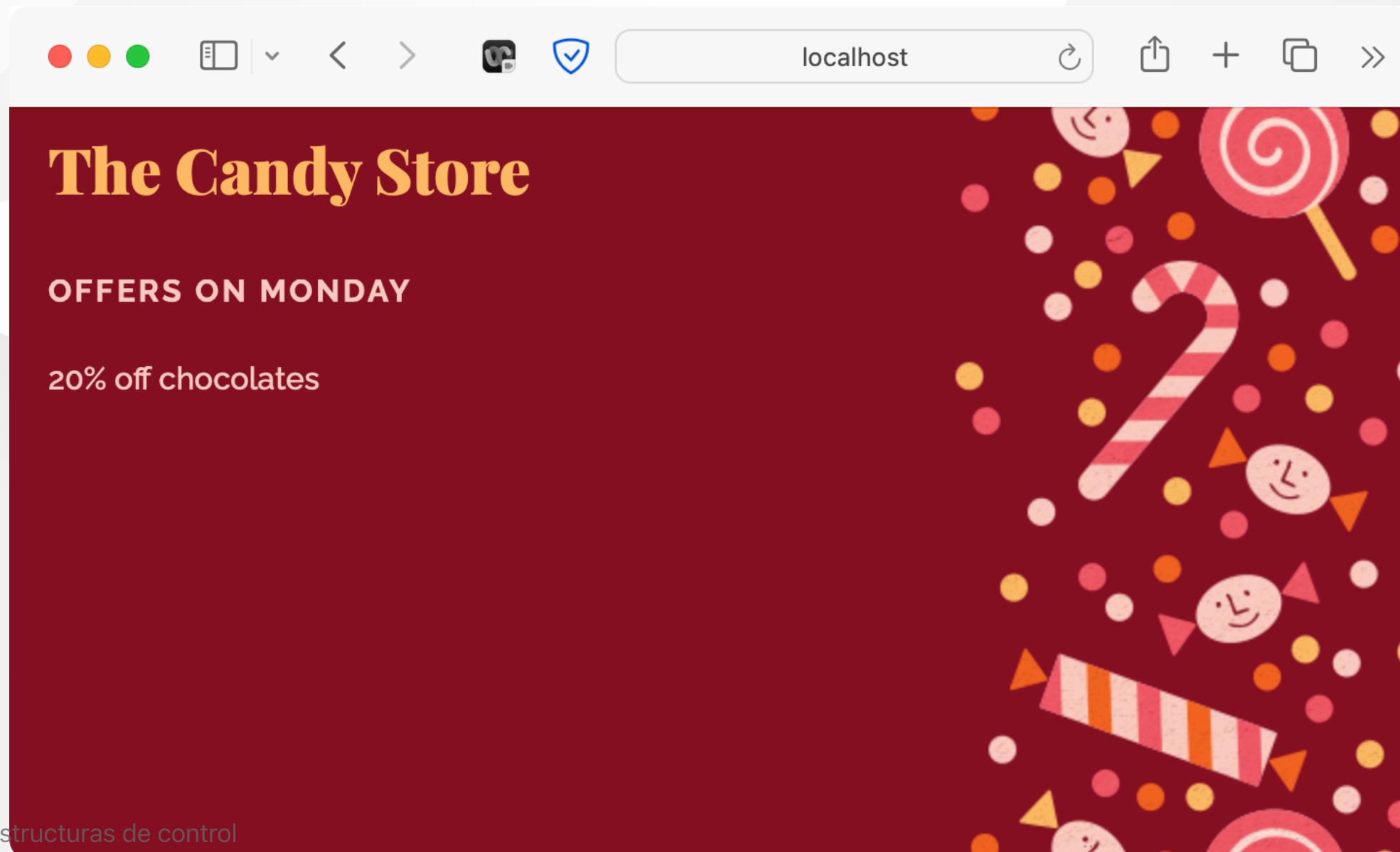
Ejemplo: Expresión match

```
<?php
$day = 'Monday';

$offer = match($day) {
    'Monday'              => '20% off chocolates',
    'Saturday', 'Sunday'  => '20% off mints',
    default                => '10% off your entire order',
};

?>
<!DOCTYPE html>
<html>
    <head>
        <title>match Expression</title>
        <link rel="stylesheet" href="css/styles.css">
    </head>
    <body>
        <h1>The Candy Store</h1>
        <h2>Offers on <?= $day ?></h2>
        <p><?= $offer ?></p>
    </body>
</html>
```

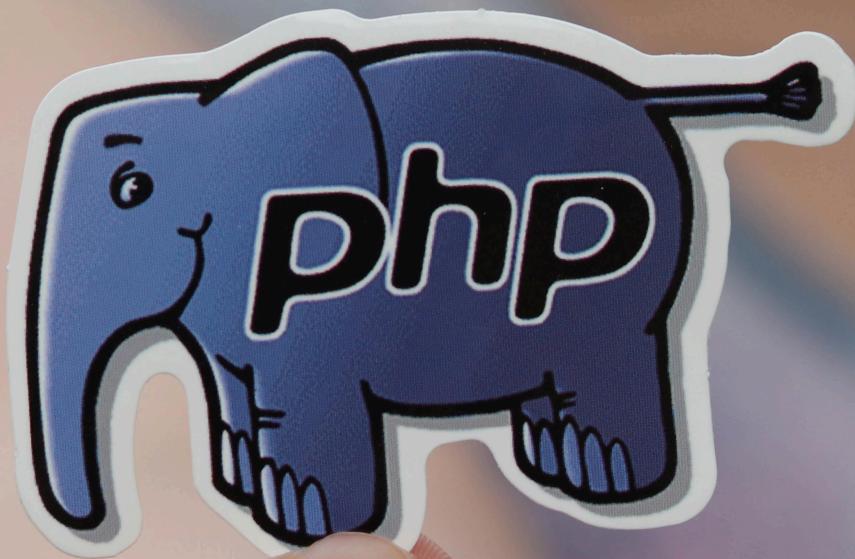
Ejemplo: Expresión match



Ejemplo: Expresión match

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el paso 1, cambia el valor a *Tuesday*.
- En el paso 3, añade una oferta a la expresión `match` para martes (*Tuesday*).
- En el paso 1, cambia el valor a *Wednesday*. A continuación, elimina la opción por defecto. Debería aparecer un error.



Bucles

Bucles

Los bucles permiten **escribir un conjunto de instrucciones una vez y luego repetirlas un número fijo de veces o hasta que se cumpla una condición.**

Si quieras que una persona haga la misma tarea diez veces, en lugar de escribir las mismas instrucciones diez veces, puedes escribir las instrucciones una vez y decirle que repita la tarea diez veces.

Bucles

En PHP, los bucles nos permiten:

- **Escribir las instrucciones** para realizar una tarea una vez dentro de un par de llaves que crean un bloque de código.
- Utilizar una **condición** para determinar si ejecutar o no esas sentencias. Si la condición es *verdadera*, el bloque de código se ejecuta; si es *falsa*, no.
- Una vez que las sentencias se han ejecutado, **la condición se comprueba de nuevo**. Si es *verdadera*, se repiten las sentencias y se vuelve a comprobar la condición.

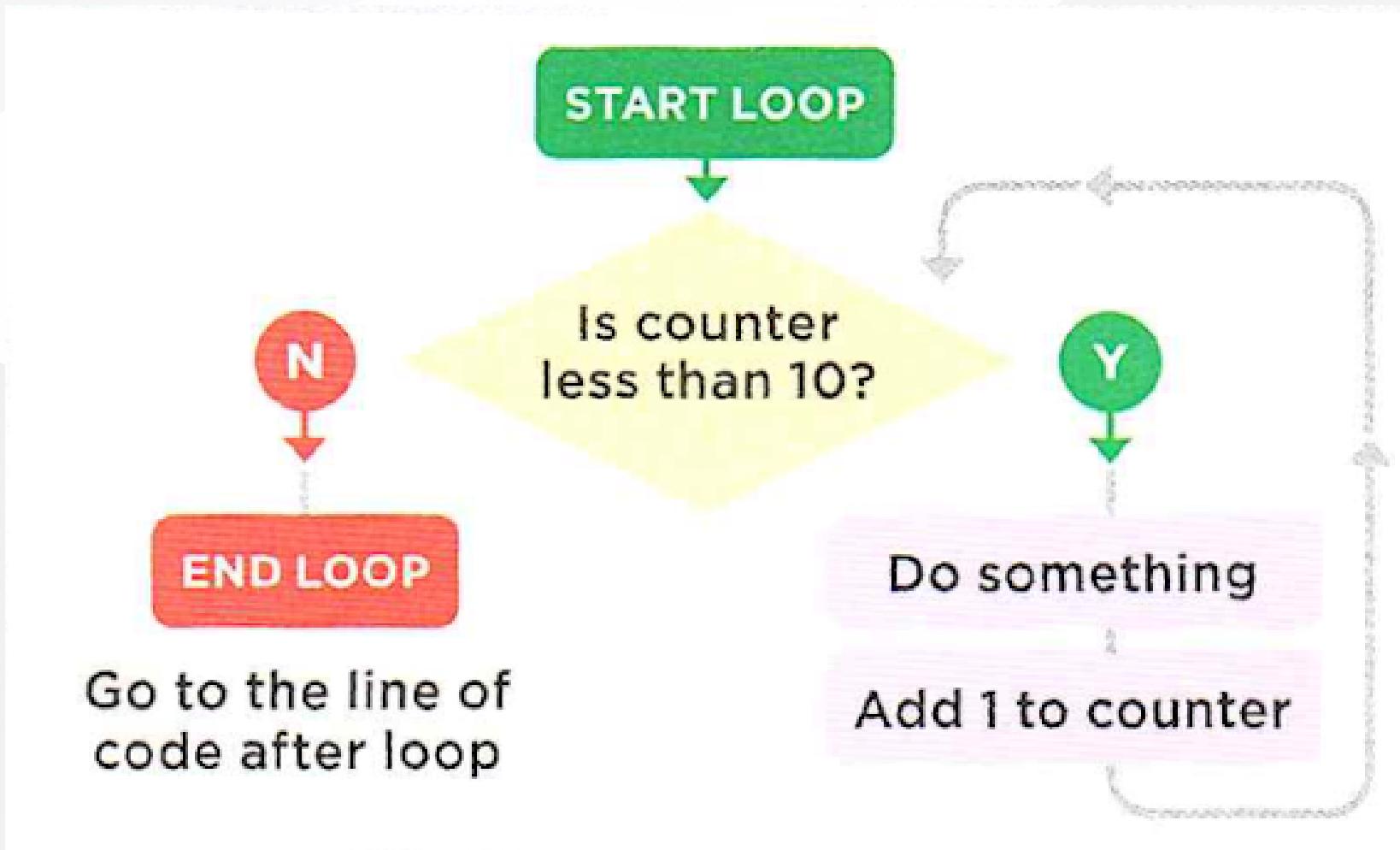
Cuando la condición es *falsa*, **el intérprete pasa a la línea de código siguiente al bucle**.

Bucles

Para **realizar una tarea diez veces**, se puede utilizar una variable que actúe como **contador**, darle un valor de 1, y luego:

1. Comprobar si el valor del contador es menor que 10.
2. Si lo es, se ejecutan las sentencias del bloque de código.
3. El **valor almacenado en el contador se incrementa en 1**.
4. El intérprete PHP vuelve al Paso 1.

Bucles



Bucles

While

Un bucle `while` repite las sentencias del bucle **mientras la condición tenga como resultado un valor *verdadero***.

Do-While

Un bucle `do...while` es como un bucle while, excepto que la condición se comprueba después de que el grupo de sentencias se haya ejecutado, lo que significa que **las sentencias se ejecutarán al menos una vez, incluso si la condición se evalúa como falsa.**

Bucles

For

Un bucle `for` permite repetir el bloque de código un número determinado de veces. La condición va seguida de instrucciones que crean el contador y lo actualizan cada vez que se procesa el bucle.

Foreach

Un bucle `foreach` recorre cada uno de los elementos de un array y repite la misma serie de sentencias para cada uno de ellos. (También puede trabajar con propiedades de objetos, que conoceremos en la unidad 4).

Bucle while

Un bucle `while` comprueba una condición; si devuelve *verdadero*, se ejecuta un bloque de código. A continuación, la condición se comprueba de nuevo; si es *verdadera*, el bloque de código se ejecuta de nuevo. **El bucle se repite hasta que la condición es falsa.**

Bucle while

Tipo de bucle

Todos los bucles comienzan con una **palabra clave** que le indica al intérprete de PHP qué **tipo de bucle** está utilizando. Un bucle while comienza con la palabra clave `while`.

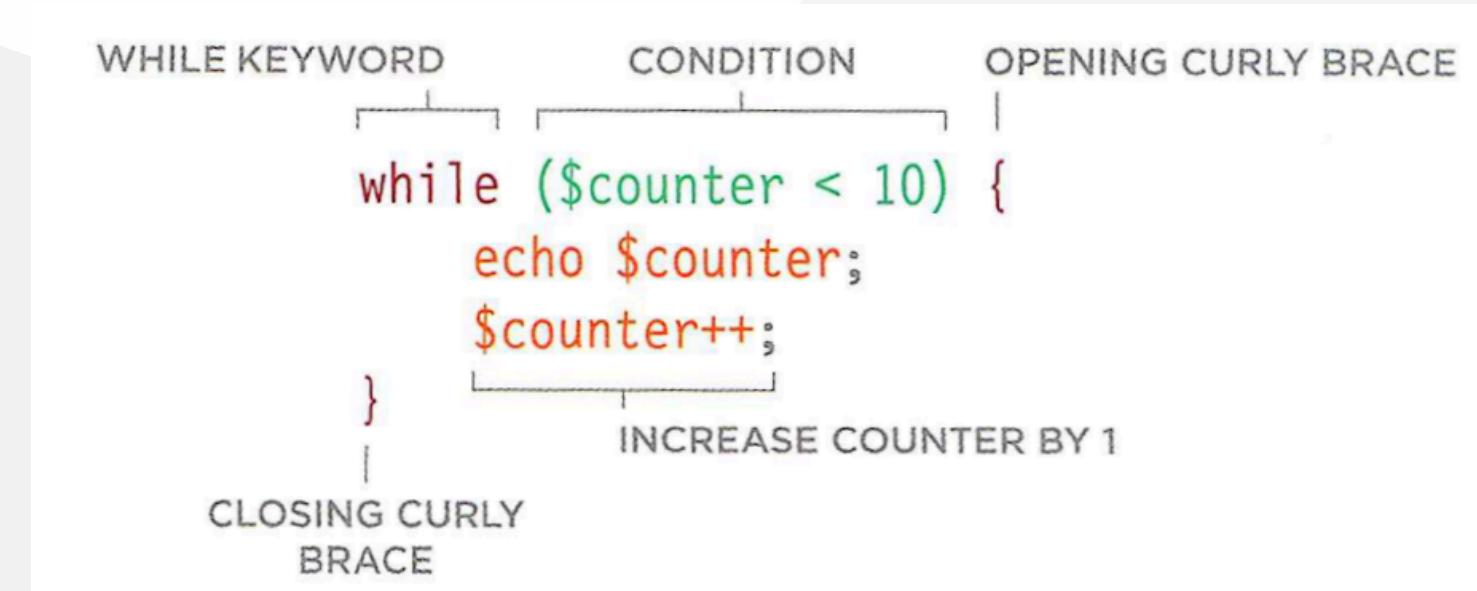
Condición

La condición **verifica valores en el código**. (El siguiente ejemplo verifica si el valor en la variable `$counter` es menor a 10.) Si la condición se evalúa como *verdadera*, las sentencias entre llaves se ejecutan.

Bucle while

Sentencias a ejecutar

Las sentencias que realizan la tarea que se va a repetir **se colocan dentro de las llaves ({ }).** Los bucles repiten el código en estas llaves hasta que la condición es falsa.



Bucle while

En el ejemplo anterior, mientras el valor de la variable `$counter` sea menor que 10, repite las instrucciones entre llaves.

El código entre llaves:

1. Escribe el valor almacenado en la variable `$counter`
2. Suma 1 al valor de `$counter` utilizando el operador `++`.

Si `$counter` tuviera el valor 1 al iniciarse este código, se mostraría 123456789 (ya que escribe el contenido de `$counter` hasta llegar a 10).

Ejemplo: bucle while

El siguiente ejemplo muestra cuánto cuestan varios paquetes de caramelos.

1. Se establece una variable llamada `$counter` para que contenga el valor 1.
2. `$packs` contiene el número de paquetes para los que se muestran los precios.
3. `$price` es el coste por paquete.
4. El bucle while comienza con una condición. Comprueba si el valor de `$counter` es menor o igual que el valor almacenado en `$packs`. Si lo es, se ejecutan las sentencias entre llaves.

Ejemplo: bucle while

5. Se escribe el número del contador.
6. Se muestra el texto '*packs cost \$*'.
7. El valor en `$price` se multiplica por el valor en `$counter` y se escribe.
8. Se añade un salto de línea.
9. El número en `$counter` se incrementa en 1 utilizando el operador de incremento.

Después del paso 9, el intérprete de PHP comprueba la condición del paso 4 de nuevo.
Repite el proceso hasta que esta condición devuelva *falso*.

Ejemplo: bucle while

```
<?php
$counter = 1;
$packs   = 5;
$price    = 1.99;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>while Loop</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Prices for Multiple Packs</h2>
    <p>
      <?php
      while ($counter <= $packs) {
        echo $counter;
        echo ' packs cost $';
        echo $price * $counter;
        echo '<br>';
        $counter++;
      }
      ?>
    </p>
  </body>
</html>
```

Ejemplo: bucle while

The screenshot shows a web browser window with the address bar set to "localhost". The main content is a page titled "The Candy Store" featuring a section titled "PRICES FOR MULTIPLE PACKS" with a list of prices for different quantities of candy packs. To the right of the text is a colorful, cartoonish illustration of various candies, including a large striped candy cane, a lollipop, and several smaller circular and triangular candies.

PRICES FOR MULTIPLE PACKS

- 1 packs cost \$1.99
- 2 packs cost \$3.98
- 3 packs cost \$5.97
- 4 packs cost \$7.96
- 5 packs cost \$9.95

DWES - U2. Estructuras de control

Ejemplo: bucle while

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el paso 2, aumenta el número de paquetes a 10.
- En el paso 4, cambia el operador a < en lugar de <= .

Bucle do-while

Un bucle do while **ejecuta un conjunto de sentencias entre llaves antes de comprobar la condición**, por lo que el bloque de código siempre se ejecuta al menos una vez.

Tipo de bucle

Un bucle do while comienza con la palabra clave `do`. La palabra clave `while` aparece después de la llave de cierre que contiene las sentencias a ejecutar.

Sentencias a ejecutar

Las sentencias que deben repetirse se colocan **entre llaves**. Éstas se ejecutan al menos una vez porque la condición viene después de las llaves.

Bucle do-while

Condición

La condición comprueba los valores actuales del código. Si se evalúa como *verdadero*, el intérprete de PHP **vuelve al principio del bucle y repite las sentencias**.

```
DO KEYWORD    OPENING CURLY BRACE
|           |
do {          |
echo $counter;      |
$counter++;      |
} while ($counter < 10);|
|           |
CLOSING     WHILE      CONDITION
CURLY BRACE KEYWORD
```

Bucle do-while

Las sentencias de este bucle escriben el valor de la variable `$counter` y luego suman 1 a ese número utilizando el operador `++`.

Las sentencias se ejecutan antes de la condición, por lo que siempre escribirá el valor en `$counter` y le sumará 1 al menos una vez.

Si `$counter` tuviera un valor de 3, entonces este código escribiría 3456789. Si `$counter` tuviera un valor de 1, escribiría 123456789.

Ejemplo: bucle do-while

En este ejemplo, el código entre llaves se ejecuta antes de que se compruebe la condición, por lo que el bloque de código se ejecuta una vez aunque la condición sea falsa.

1. Se establecen dos variables. El número de paquetes de caramelos se almacena en `$packs`. El precio por paquete se almacena en `$price`.
2. El bucle `do while` comienza con la palabra clave `do` y una llave de apertura. El bloque de código viene antes de la condición, por lo que se ejecuta una vez se cumpla o no la condición.
3. Se escribe el número de paquetes (almacenados en `$packs`), seguido de las palabras "`packs cost $`".

Ejemplo: bucle do-while

4. El coste se calcula multiplicando `$packs` por `$price`. A continuación se hace un salto de línea.
5. El número almacenado en `$packs` se reduce en uno utilizando el operador `--` decremento.
6. El bloque de código termina con una llave de cierre, seguido por la palabra clave `while` y luego la condición, la cual comprueba si el número almacenado en `$packs` es mayor que 0.

Ejemplo: bucle do-while

```
<?php
$packs = 5;
$price = 1.99;
?>
<!DOCTYPE html>
<html lang="en-us">
  <head>
    <title>do while Loop</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Prices for Multiple Packs</h2>
    <p>
      <?php
      do {
        echo $packs;
        echo ' packs cost $';
        echo $price * $packs;
        echo '<br>';
        $packs--;
      } while ($packs > 0);
      ?>
    </p>
  </body>
</html>
```

Ejemplo: bucle do-while

The screenshot shows a web browser window with the address bar set to "localhost". The main content is a page titled "The Candy Store" featuring a list of prices for multiple candy packs. To the right of the text is a decorative background with a red candy cane, lollipops, and colorful confetti.

The Candy Store

PRICES FOR MULTIPLE PACKS

5 packs cost \$9.95
4 packs cost \$7.96
3 packs cost \$5.97
2 packs cost \$3.98
1 packs cost \$1.99

DWES - U2. Estructuras de control

82

Ejemplo: bucle do-while

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 1, cambia el valor en `$packs` a 10 y cambia el valor almacenado en `$price` a 2.99.

Bucle for

Un bucle `for` repite un conjunto de sentencias un número determinado de veces. Para ello, crea un contador y lo actualiza cada vez que se ejecuta el bucle.

Tipo de bucle

Un bucle for comienza con la palabra clave `for`.

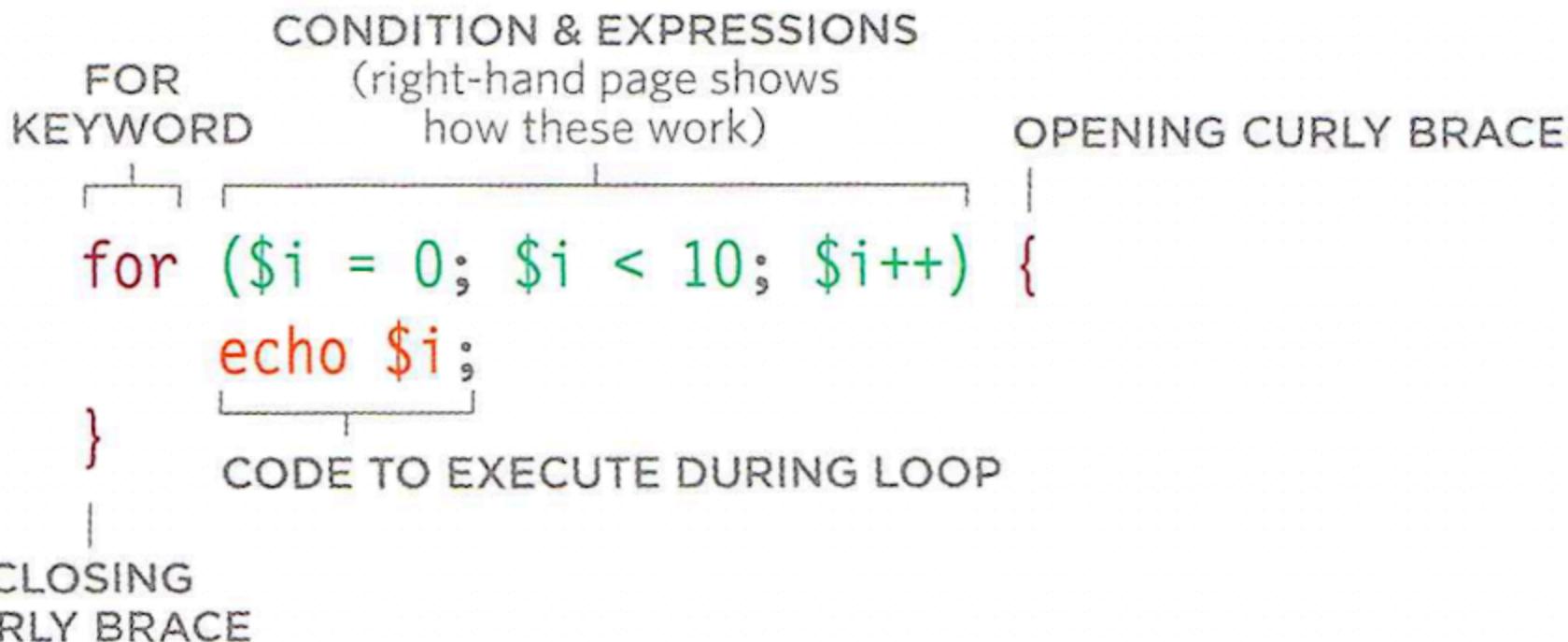
Condición

En un bucle for, la condición se sitúa junto al código para crear y actualizar el contador. Esto se muestra en detalle a continuación.

Bucle for

Sentencias a ejecutar

Las sentencias que realizan la tarea que debe repetirse se colocan **dentro de las llaves**. Se ejecutan **un número fijo de veces**.



Bucle for

El nombre de variable `$i` o `$index` se utiliza a menudo para un contador.

La sentencia entre llaves escribe el valor almacenado en `$i`.

En este caso, se escribiría: 0123456789.

Los bucles `for` utilizan tres expresiones

El bucle `for` necesita dos expresiones adicionales además de la condición. Una **crea un contador** y la otra **lo actualiza**.

Expresión 1: Inicialización

Esta expresión sólo se ejecuta una vez; la primera vez que se ejecuta el bucle. **Crea la variable para el contador y establece su valor a 0.**

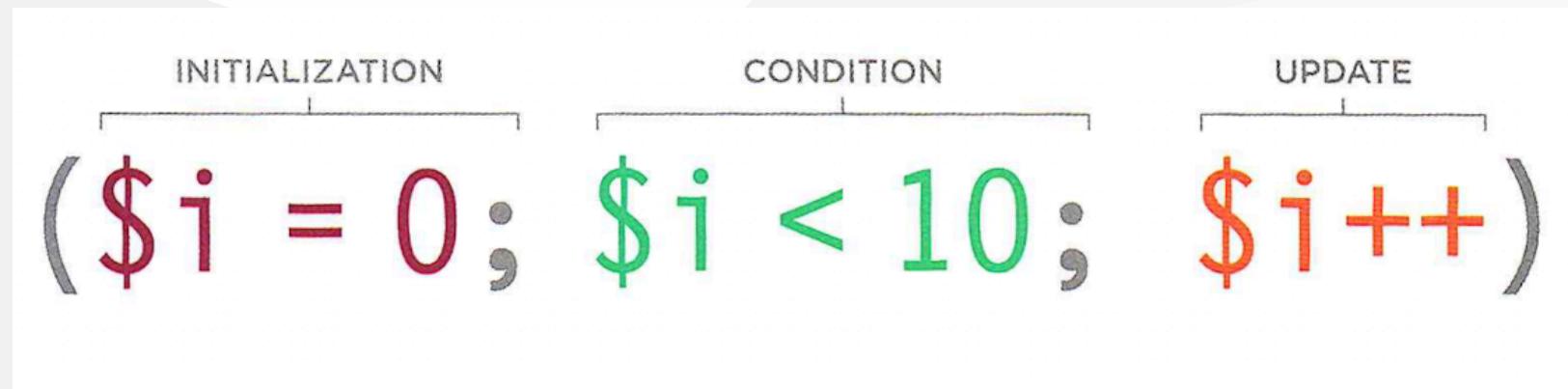
Expresión 2: Condición

La segunda expresión es la **condición**. Las sentencias en el bucle `for` se repetirán hasta que la condición sea falsa.

Los bucles for utilizan tres expresiones

Expresión 3: Actualizar

Una vez que las sentencias entre llaves se han ejecutado, la tercera expresión añade 1 al número almacenado en el contador.



Los bucles for utilizan tres expresiones

En el ejemplo anterior, la variable llamada `$i` se utiliza como **contador**. Se le asigna un valor inicial de 0.

La **condición** comprueba si el valor almacenado en `$i` es menor que 10. Si lo es, **se ejecutan las sentencias del bloque de código**.

En lugar de utilizar el número 10, podría ser una variable que contenga un valor, por ejemplo, `$i < Smax;`

Cada vez que el bucle se ha ejecutado, **el contador se actualiza** utilizando el operador de incremento `++` para añadir 1 al valor almacenado en `$i`.

Ejemplo: bucle for

El siguiente ejemplo **utiliza un bucle for para repetir una tarea diez veces.**

1. Una variable llamada `$price` almacena el coste de un solo paquete de caramelos.
2. La palabra clave `for` indica el tipo de bucle, seguida de las tres expresiones entre paréntesis:
 - Expresión uno: `$i = 1;` Esto **inicializa el contador** con un valor de 1.
 - Expresión dos: `$i <= 10;` Esta es la **condición**. Dice que el código debe repetirse mientras el valor del contador sea menor o igual que 10.
 - Expresión tres: `$i++` Esto **incrementa el número en el contador en 1** cada vez que se ejecuta el bucle.

Ejemplo: bucle for

3. Las llaves contienen las **expresiones utilizadas cada vez que se ejecuta el bucle**.

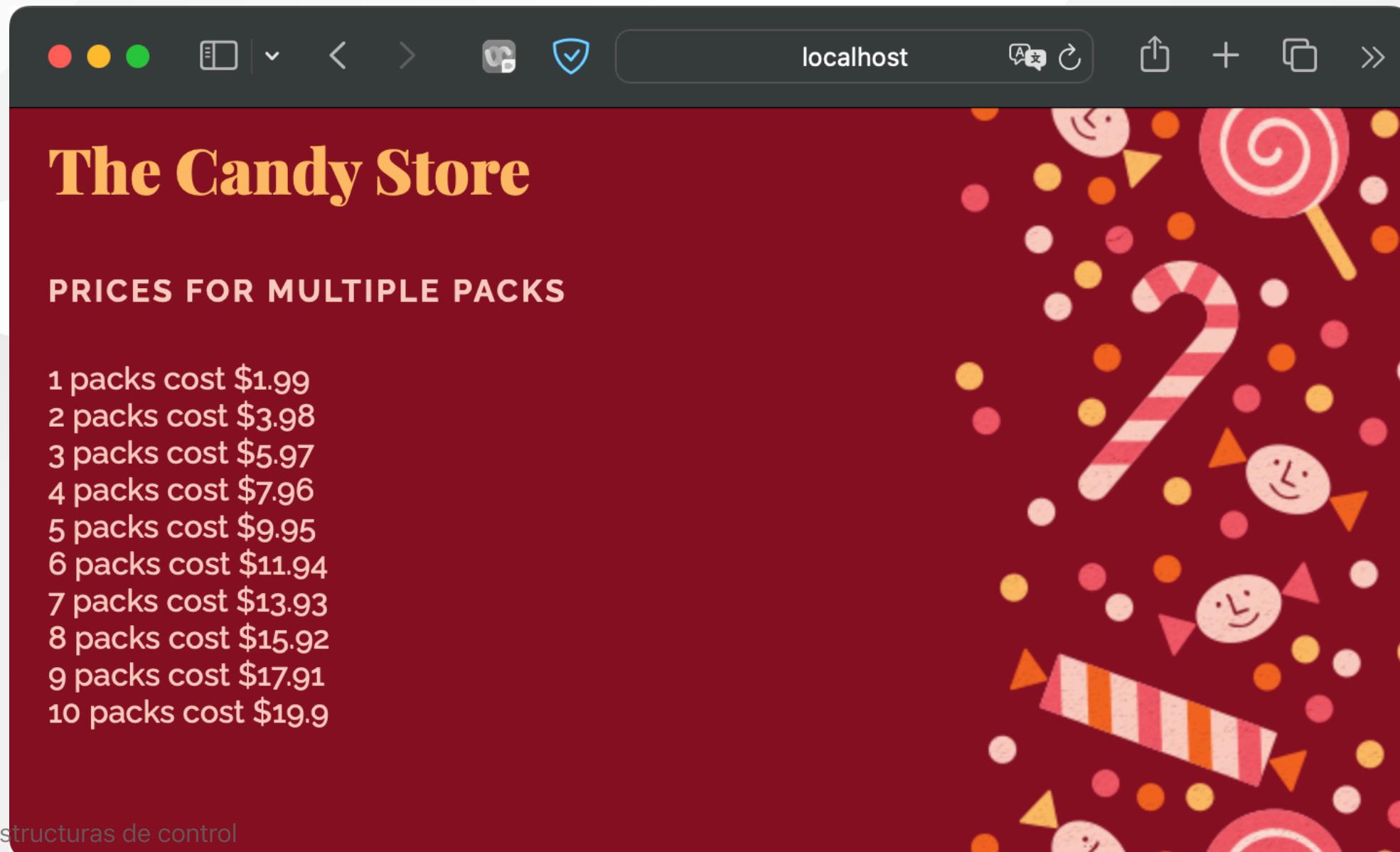
Como antes, escribe el número de paquetes (el valor almacenado en el contador), y el precio (el valor en el contador multiplicado por el valor en `$price`).

Una vez ejecutadas las sentencias de las llaves, la tercera expresión (en el paso 2) actualiza el contador incrementando en 1 el número almacenado en `$i` .

Ejemplo: bucle for

```
<?php
$price = 1.99;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>for Loop</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Prices for Multiple Packs</h2>
    <p>
      <?php
        for ($i = 1; $i <= 10; $i++) {
          echo $i;
          echo ' packs cost $';
          echo $price * $i;
          echo '<br>';
        }
      ?>
    </p>
  </body>
</html>
```

Ejemplo: bucle for



The screenshot shows a web browser window with the address bar set to "localhost". The main content is a page titled "The Candy Store" featuring a red background decorated with various candies like lollipops, a candy cane, and a party hat. On the left, there's a section titled "PRICES FOR MULTIPLE PACKS" listing the cost for different quantities of packs:

Packs	Cost
1	\$1.99
2	\$3.98
3	\$5.97
4	\$7.96
5	\$9.95
6	\$11.94
7	\$13.93
8	\$15.92
9	\$17.91
10	\$19.9

DWES - U2. Estructuras de control

93

Ejemplo: bucle for

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el paso 1, aumenta el precio de 1,99 a 2,99.
- En el Paso 2, haz que el bucle se repita 20 veces.

Ejemplo: bucle for (2)

Este ejemplo muestra descuentos por comprar varios paquetes de caramelos.

1. Una variable almacena el precio de un solo paquete de caramelos.
2. La palabra clave `for` indica el tipo de bucle, seguida de las tres expresiones entre paréntesis:
 - Expresión uno: `$1 = 10;` . Inicializa el contador con un valor de 10.
 - Expresión dos: `$i <= 100;` Esta es la condición. Indica que el código debe repetirse mientras el valor del contador sea menor o igual a 100.
 - Expresión tres: `$i = $i + 10` Esto incrementa el valor en el contador en 10 cada vez que se ejecuta el bucle.
3. Las llaves contienen las expresiones que se ejecutan cada vez que se ejecuta el bucle.
Son las mismas que las del ejemplo anterior.

Ejemplo: bucle for (2)

```
<?php
$price = 1.99;
?>
<!DOCTYPE html>
<html>
  <head>
    <title>for Loop – Higher Counter</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Prices for Large Orders</h2>
    <p>
      <?php
        for ($i = 10; $i <= 100; $i = $i + 10) {
          echo $i;
          echo ' packs cost $';
          echo $price * $i;
          echo '<br>';
        }
      ?>
    </p>
  </body>
</html>
```

Ejemplo: bucle for (2)

The screenshot shows a web browser window with the address bar set to "localhost". The main content area displays a page titled "The Candy Store" with a heading "PRICES FOR LARGE ORDERS". Below this, a list of price points for large orders is shown, generated by a for loop:

- 10 packs cost \$19.9
- 20 packs cost \$39.8
- 30 packs cost \$59.7
- 40 packs cost \$79.6
- 50 packs cost \$99.5
- 60 packs cost \$119.4
- 70 packs cost \$139.3
- 80 packs cost \$159.2
- 90 packs cost \$179.1
- 100 packs cost \$199

The background of the page features a red pattern with various candies, including lollipops and a large striped candy cane.

Ejemplo: bucle for (2)

Sobre el ejemplo anterior, realiza la siguiente modificación:

- En el Paso 2, actualiza la condición para mostrar los precios de hasta 200 paquetes de caramelos.

Bucle `foreach`

Un bucle `foreach` está diseñado para trabajar con tipos de datos compuestos como los arrays. Recorre cada elemento del array, uno a uno, y ejecuta el mismo bloque de código para cada elemento.

Los tipos de datos compuestos, como los arrays, contienen una serie de elementos relacionados. Cada elemento está formado por un **par clave/valor**.

- En un array **asociativo**, la clave es una **cadena**.
- En un array **indexado**, la clave es un número de **índice**.

Un bucle `foreach` recorre cada elemento de un array, uno a uno, ejecuta las sentencias del bloque de código y pasa al siguiente elemento.

Bucle foreach

Cada vez que se ejecuta el bloque de código, puedes acceder a la **clave** y al **valor** del **elemento actual del array** y utilizarlo dentro del bloque de código.

Para ello, en los paréntesis después de la palabra clave foreach, se especifica:

- El nombre de la **variable que contiene el array**
- Un nombre de **variable para representar la clave actual**
- Un nombre de **variable para representar el valor actual**

Bucle foreach

Tipo de bucle

Un bucle foreach comienza con la palabra clave `foreach`.

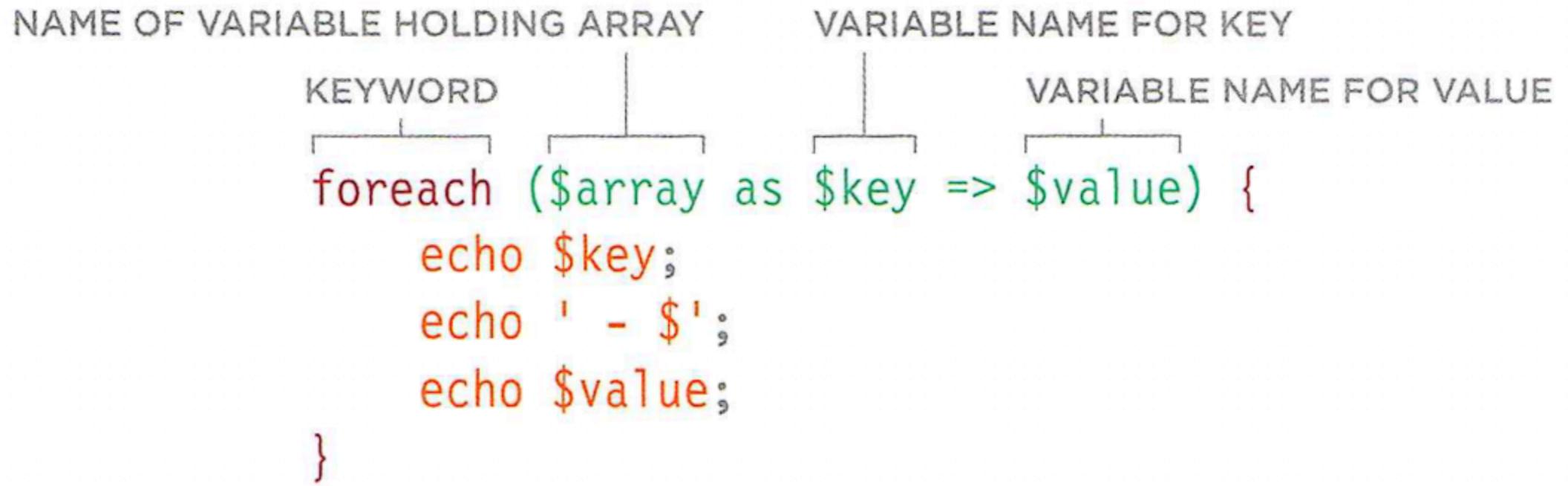
Nombres de las variables

Los paréntesis contienen tres nombres de variables (mostrados en detalle a continuación).

Sentencias a ejecutar

Las sentencias que realizan la tarea que se va a repetir están dentro de las llaves.

Bucle foreach



Bucle foreach

En el ejemplo anterior hay tres sentencias dentro de las llaves, las cuales escriben:

- La clave
- Un guión y un símbolo de dólar
- El valor del elemento

Si sólo quieres utilizar los valores del array, puedes omitir la clave:

```
foreach ($array as $value) {  
    // Las sentencias van aquí  
}
```

Bucle foreach

Una vez que se han repetido las mismas sentencias para cada elemento del array, el intérprete de PHP pasa a la siguiente línea de código después del bucle.

A continuación, `$products` almacena un array de nombres de productos y precios.

```
$products = ['toffee' => 2.99, 'mints' => 1.99, 'fudge' => 3.40,];
```

VARIABLE KEY VALUE KEY VALUE KEY VALUE

Un bucle `foreach` es capaz de mostrar un nombre y un precio para cada artículo.

El bucle funcionará independientemente del número de elementos que contenga el array.

Bucle foreach

En el ejemplo siguiente, el bucle comienza con la palabra clave `foreach`.

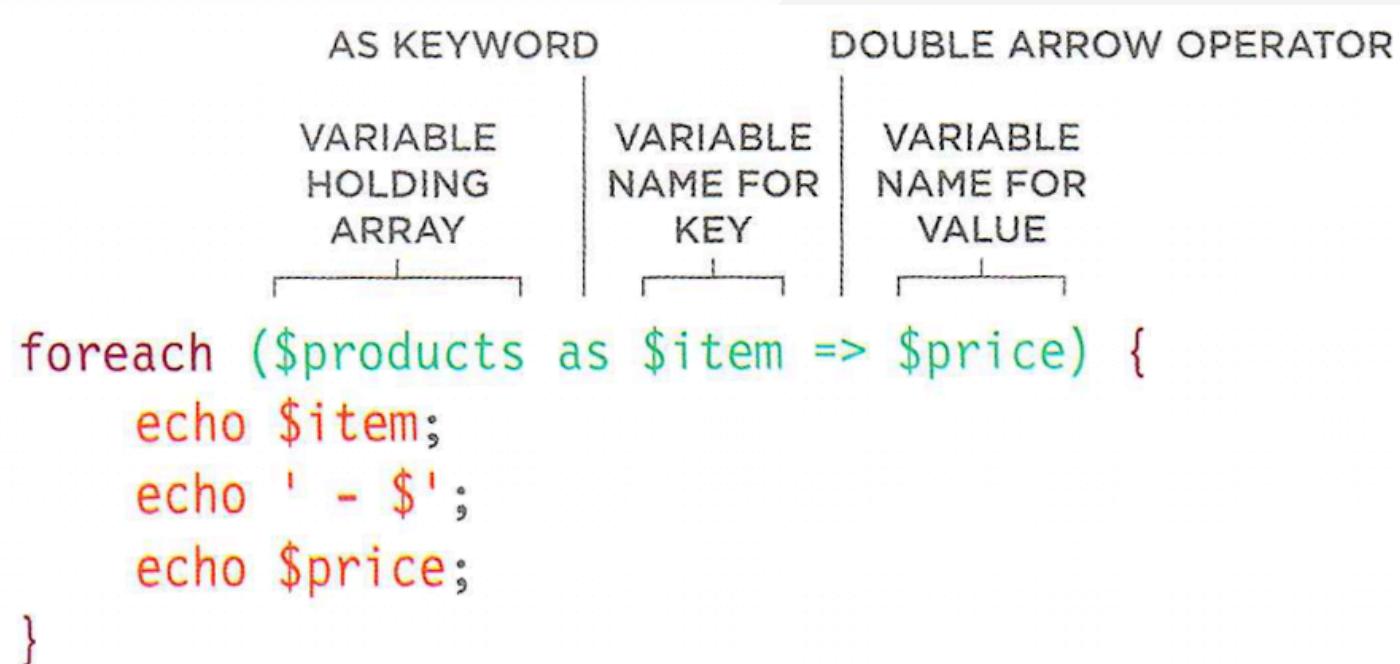
Luego, en el paréntesis

- `$productos` es el nombre de la **variable que contiene el array**.
- Le sigue la palabra clave `as`.
- `$item` es el nombre de la **variable que contendrá la clave del elemento actual del array**.
- Va seguido de un **operador de flecha doble** (`=>`).
- `$price` es el nombre de la **variable que contendrá el valor del elemento actual en el array**.

Bucle foreach

En el bloque de código, los nombres de variable `$item` y `$price` se utilizarán para representar la clave y el valor del elemento actual de la matriz.

Primero, se escribe el nombre del producto (almacenado en `$item`), seguido de un dólar y un guión, y luego el precio (almacenado en `$price`).



Bucle foreach

Los bucles se utilizan a menudo **en la parte de la página que genera HTML**.

Si fuese el caso, la primera y la última línea del bucle anterior de ejemplo podrían colocarse en sus propios bloques de código, y los datos de las claves y valores del array pueden entonces escribirse utilizando la abreviatura de echo (`<?= ?>`), como se observa en el siguiente ejemplo:

```
<?php foreach ($products as $item => $price) { ?>
<li>
    <b><?= $item ?></b> - $<?= $price ?>
</li>
<?php } ?>
```

Ejemplo: bucle foreach

Este ejemplo muestra una **tabla con los nombres y precios de los productos almacenados en un array**.

1. Una variable llamada `$products` almacena un array asociativo que contiene una lista de productos, junto con sus precios.
2. En la parte HTML de la página, se escribe un encabezado y el inicio de la tabla HTML.
3. Se crea un bucle `foreach`.

Ejemplo: bucle foreach

En los paréntesis que siguen a la palabra clave `foreach`, se ve:

- `$productos` - el nombre de la variable que contiene el array
- La palabra clave `as`
- `$item` - nombre de la variable utilizada para representar la clave del elemento actual del array
- El operador de doble flecha
- `$precio` - nombre de la variable utilizada para representar el valor del elemento actual de la matriz.

Esto es seguido por una llave de apertura para iniciar un bloque de código.

4. Se escribe una fila de tabla para cada elemento del array mostrando su nombre y precio.

5. Por último, se cierra el bloque de código.

Ejemplo: bucle foreach

```
<?php
$products = [
    'Toffee' => 2.99,
    'Mints'   => 1.99,
    'Fudge'   => 3.49,
];
?>

...
<h2>Price List</h2>
<table>
    <tr>
        <th>Item</th>
        <th>Price</th>
    </tr>
<?php foreach ($products as $item => $price) { ?>
    <tr>
        <td><?= $item ?></td>
        <td>$<?= $price ?></td>
    </tr>
<?php } ?>
</table> ...
```

Ejemplo: bucle foreach

The screenshot shows a web browser window with the address bar set to "localhost". The main content is a page titled "The Candy Store" featuring a "PRICE LIST" table and a decorative background.

PRICE LIST

ITEM	PRICE
Toffee	\$2.99
Mints	\$1.99
Fudge	\$3.49

The background of the page is red and features a whimsical illustration of a large candy cane, lollipops, and various colorful shapes like circles and triangles.

Ejemplo: bucle

Sobre el ejemplo anterior, realiza la siguiente modificación:

- En el paso 1, añade dos elementos más al array.

Ejemplo: bucle foreach (2)

En un array **indexado**, los números de índice suelen representar el orden de los elementos en el array.

Se puede utilizar un bucle `foreach` para escribir los valores en ese orden, como se puede ver en el siguiente ejemplo:

1. La variable `$best_sellers` contiene un array indexado que representa los productos más vendidos.

Ejemplo: bucle foreach (2)

2. Se utiliza un bucle `foreach` para mostrar los productos más vendidos. Después de la palabra clave `foreach`, en el paréntesis, se ve:

- `$best_sellers` - el nombre de la **variable que contiene el array**
- La palabra clave `as`
- `$product` - el nombre de la **variable que se utilizará para representar el valor del elemento actual** del array
- **No hay variable para el nombre de la clave** (el número de índice)
Esto es seguido por una llave de apertura para iniciar un bloque de código.

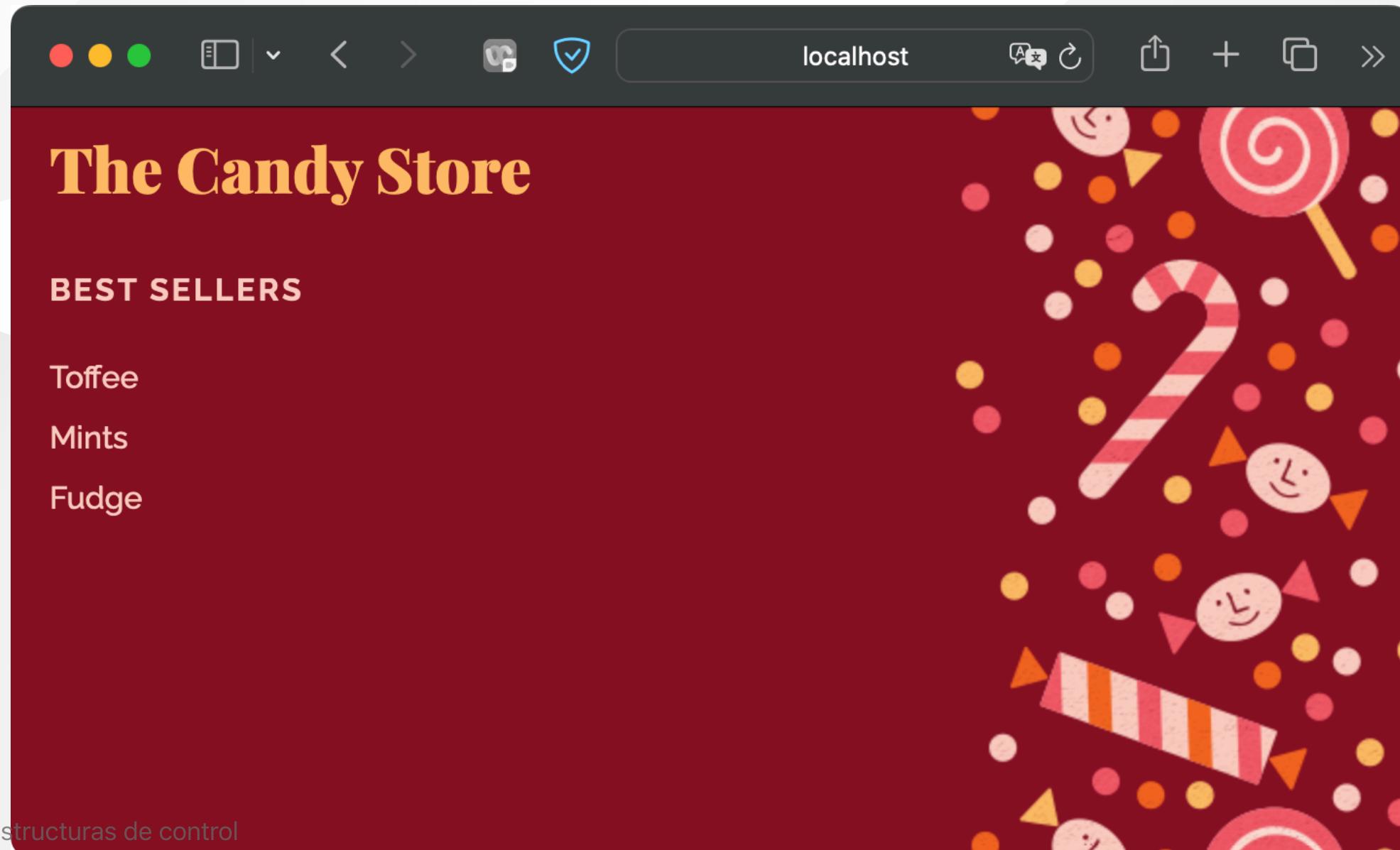
3. El **valor del elemento actual** se muestra en etiquetas `<p>`.

4. Por último, se cierra el bloque de código.

Ejemplo: bucle foreach (2)

```
<?php
$best_sellers = ['Toffee', 'Mints', 'Fudge'];
?>
<!DOCTYPE html>
<html>
  <head>
    <title>foreach Loop – Just Accessing Values</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <h2>Best Sellers</h2>
    <?php foreach ($best_sellers as $product) { ?>
      <p><?= $product ?></p>
    <?php } ?>
  </body>
</html>
```

Ejemplo: bucle foreach (2)



Ejemplo: bucle foreach (2)

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 1, añade dos elementos más al array.
- Luego, en los Pasos 2 y 3, cambia el nombre de la variable de `$product` a `$candy`.



Archivos *include*

Archivos *include*

La mayoría de los sitios web necesitan repetir código idéntico en varias páginas.

Por ejemplo, la **cabecera** y el **pie de página** suelen ser iguales en todas las páginas. Los archivos *include* te ahorran repetir las mismas líneas de código en varios archivos.

Archivos *include*

En lugar de duplicar el código para el encabezado de un sitio en cada página, se puede:

- Poner el código para la cabecera en un archivo PHP separado que se conoce como archivo *include*.
- Utilizar la sentencia *include* de PHP para **añadir ese código a cada página que utiliza la cabecera**.

Cuando el intérprete de PHP se encuentra con una sentencia *include*, **obtiene el contenido del archivo *include* y ejecuta ese código como si estuviera colocado donde se utiliza la sentencia *include***.

Archivos *include*

En el siguiente ejemplo, un archivo llamado `candy.php` incluye dos archivos:

- `header.php` contiene el **encabezado** del sitio.
- `footer.php` contiene el **pie de página** del sitio.

Entre las dos declaraciones de inclusión se encuentra el **contenido principal de la página**.

Utilizar archivos de inclusión:

- **Evita duplicar o repetir** el mismo código.
- **Facilita el mantenimiento del código** porque, cuando se modifica un archivo include, se actualizan todas las páginas que lo utilizan.

Archivos *include*

candy.php

```
<?php include 'includes/header.php'; ?>
```

```
<h1>The Candy Store</h1>
```

```
<h2>Welcome</h2>
```

```
<p>A wide selection of delicious candy  
handmade in our kitchen...</p>
```

```
<?php include 'includes/footer.php'; ?>
```

Archivos *include*

includes/header.php

```
<h1>The Candy Store</h1>
<nav>
    <a href="index.php">Home</a> |
    <a href="candy.php">Candy</a> |
    <a href="about.php">About</a> |
    <a href="contact.php">Contact</a>
</nav>
```

Archivos *include*

includes/footer.php

```
<footer>
    &copy; <?php echo date('Y')?>
</footer>
```

Archivos *include*

Se pueden utilizar **cuatro palabras clave** para incluir código de un archivo de inclusión. Cada una se comporta de forma ligeramente diferente, pero utiliza la misma sintaxis.

La palabra clave `include` le dice al intérprete de PHP que **obtenga otro archivo del servidor y lo trate como si el contenido de ese archivo hubiera sido escrito donde está escrita la sentencia `include`**.

Va seguida de la **ruta al archivo escrita entre comillas**. (A veces se ve el nombre del archivo y las comillas entre paréntesis, pero no son necesarios). El archivo de inclusión debe utilizar la extensión `.php`.

Archivos *include*

INCLUDE STATEMENT

```
<?php include 'includes/filename.php'; ?>
```

RELATIVE PATH TO FILE

Archivos *include*

include / require

Las palabras clave `include` y `require` añaden el código del archivo cuya ruta sigue a la palabra clave.

La diferencia entre ellas es **cómo se comporta el intérprete de PHP si el archivo incluido no puede ser encontrado o leído:**

- `include` : el intérprete **genera un error, pero sigue intentando procesar el resto de la página.**
- `require` : el intérprete **genera un error, y deja de intentar procesar el resto de la página.**

Archivos *include*

`include_once` / `require_once`

Las palabras clave `include_once` y `require_once` realizan **exactamente el mismo trabajo que include y require**, pero **aseguran que el intérprete PHP sólo incluya el código una vez en cualquier página**.

Una vez que un archivo ha sido incluido en la página utilizando las palabras clave `include_once` y `require_once`, si la página utiliza las mismas palabras clave para incluir el mismo archivo, **no será incluido una segunda vez**.

El intérprete de PHP utiliza recursos adicionales para comprobar si el archivo ya ha sido incluido, por lo que estas opciones **sólo deben utilizarse cuando exista riesgo de duplicación**.

Ejemplo: Archivos *include*

A continuación se muestran dos archivos *include*:

- `header.php` contiene el marcado **HTML** de apertura, el **nombre del sitio** y la **navegación** que aparece en la parte superior de cada página del sitio.
- `footer.php` contiene un **aviso de copyright** con el año en curso y el **HTML** de cierre de cada página.

Ambos archivos utilizan la extensión `.php`. Esto asegura que el código PHP se ejecuta a través del intérprete PHP.

Los archivos *include* suelen almacenarse en una carpeta llamada ***includes***.

Como muestra este ejemplo, si tuvieramos que actualizar la navegación, sólo tendríamos que actualizar el archivo `header.php`, y **se actualizaría automáticamente cada página que incluyera este archivo**.

Ejemplo: Archivos *include*

```
<!-- includes/header.php -->

<!DOCTYPE html>
<html>
  <head>
    <title>The Candy Store</title>
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <h1>The Candy Store</h1>
    <nav>
      <a href="index.php">Home</a> |
      <a href="candy.php">Candy</a> |
      <a href="about.php">About</a> |
      <a href="contact.php">Contact</a>
    </nav>
```

Ejemplo: Archivos *include*

```
<!-- includes/footer.php -->

<footer>&copy; <?php echo date('Y')?></footer>
</body>
</html>
```

Ejemplo: Archivos *include*

La siguiente página utiliza los archivos *include* que acabamos de ver.

1. La página comienza creando un **mensaje sobre los niveles de existencias** y almacenándolo en una variable llamada `$message`. Si el nivel de existencias es:
 - Mayor o igual a 10, el mensaje es *Good availability*.
 - Entre 1 y 9, el mensaje es *Low stock*.
 - 0, el mensaje es *Out of stock*.
2. La parte HTML de la página comienza incluyendo el fichero de cabecera (que contiene el código que se utiliza al principio de cada página).

Ejemplo: Archivos *include*

La sentencia `require_once` indica que el fichero **sólo debe incluirse una vez**. Se coloca donde debe aparecer la cabecera, y el intérprete de PHP lo trata como si este código se copiara y pegara allí.

3. A continuación aparece el **contenido de la página**, que muestra el mensaje de nivel de existencias.
4. Finalmente, la sentencia `include` indica al intérprete PHP que añada el código del fichero `footer.php`.

Ejemplo: Archivos *include*

```
<?php
$stock = 25;

if ($stock >= 10) {
    $message = 'Good availability';
}
if ($stock > 0 && $stock < 10) {
    $message = 'Low stock';
}
if ($stock == 0) {
    $message = 'Out of stock';
}?
<?php require_once 'includes/header.php'; ?>

<h2>Chocolate</h2>
<p><?= $message ?></p>

<?php include 'includes/footer.php'; ?>
```

Ejemplo: Archivos *include*

The screenshot shows a web browser window with the address bar set to "localhost". The main content area displays a website for "The Candy Store". The header features the store's name in a large, gold-colored serif font. Below the header is a navigation bar with links for "HOME", "CANDY", "ABOUT", and "CONTACT". A section titled "CHOCOLATE" contains the text "Good availability". The background of the page is a vibrant red color decorated with various colorful, cartoonish candy illustrations, including lollipops, a wrapped candy cane, and smiling circles.

The browser interface includes standard controls like back and forward buttons, a search bar, and a refresh button. The overall design is clean and modern, utilizing a sans-serif font for most text elements.

Ejemplo: Archivos *include*

Sobre el ejemplo anterior, realiza la siguiente modificación:

- Añade un nuevo enlace a la navegación en la cabecera .php .



Ejemplo final de la unidad

Ejemplo final de la unidad

Este ejemplo muestra un saludo, seguido de los descuentos aplicables cuando un cliente compra varios paquetes de caramelos.

Utiliza varias de las técnicas que se han presentado a lo largo de esta unidad.

- Una **variable** almacena el nombre del visitante.
- Un **operador condicional** crea un saludo para el visitante.
- Se utiliza un **bucle for** para crear un **array indexado** que contiene los precios con descuento que se ofrecen cuando los clientes compran varios paquetes de caramelos.

Ejemplo final de la unidad

- La cabecera y el pie de página de la página se encuentran en archivos **include**.
- Se utiliza un **bucle foreach** para mostrar los precios con descuento del array. Para un paquete hay un descuento del 4%, para dos paquetes el descuento es del 8%, para tres paquetes el descuento es del 12% y así sucesivamente...
- Un **operador ternario** garantiza que la página muestre la palabra pack cuando muestre el precio de un solo paquete de caramelos, o la palabra packs cuando muestre el precio de varios paquetes.

Ejemplo final de la unidad

El fichero comienza creando los valores que se mostrarán en la página y almacenándolos en variables.

1. Una variable llamada `$name` contiene el nombre del usuario.
2. Se inicializa la variable `$greeting`, que almacena el mensaje "Hello" que se puede mostrar a cualquier persona.
3. La condición de una sentencia `if` comprueba si la variable `$name` tiene algún valor.
4. Si lo tiene, el valor de `$greeting` se actualiza para contener un **mensaje personal utilizando el nombre del visitante**.

Ejemplo final de la unidad

5. El nombre de un producto se almacena en `$product`.

6. El coste de un paquete se almacena en `$cost`.

7. Un bucle `for` **crea un array para almacenar los precios de múltiples paquetes de caramelos.** El contador representa el número de paquetes de caramelos. En el paréntesis del bucle:

- El **contador** se pone a 1 (que representa 1 paquete de caramelos)
- La **condición** comprueba si el contador es menor o igual que 10 (que representa 10 paquetes de caramelos)
- El contador **se incrementa en 1** cada vez que se ejecuta el bucle

Ejemplo final de la unidad

Dentro del **bucle**:

8. La variable `$subtotal` contiene el precio de un paquete individual de caramelos multiplicado por el contador (que representa el número actual de paquetes).
9. La variable `$discount` contiene el descuento al comprar este número de paquetes. Se calcula dividiendo el coste entre 100 y multiplicando esa cifra por el número del contador multiplicado por 4.
10. La variable `$totals` contiene un array; la **clave** es el valor actual del contador (que representa el número de paquetes) y el **valor** es el precio de este número de paquetes de caramelos menos el descuento.
11. Se cierra el bucle `for`.

Ejemplo final de la unidad

En la parte de la página que genera el HTML que se devuelve al navegador:

12. La **cabecera** de la página se incluye utilizando la palabra clave *require* porque es necesaria para que el resto de la página se muestre correctamente (el [archivo de cabecera](#) se mostró previamente).
13. **Se escribe el saludo** en la página.
14. **El nombre del producto se escribe** en la página.
15. Se crea una **tabla HTML** y se añaden **encabezados** de columna a la primera fila de la tabla.

Ejemplo final de la unidad

16. Se utiliza un bucle `foreach` para mostrar los datos almacenados en el array que se creó en los Pasos 1-7. Se añade una nueva fila de datos a la tabla por cada elemento de ese array. En el paréntesis del bucle:
- El **array** a utilizar se almacena en una variable llamada `$totales`
 - La **clave** se almacena en una variable llamada `$quantity`
 - El **valor** se almacena en una variable llamada `$precio`
17. Se escribe la clave de este elemento del array (es el número de paquetes de caramelos).

Ejemplo final de la unidad

18. Se escribe el texto "`pack`". La condición de un operador ternario comprueba entonces si el valor en `$cantidad` es 1. Si lo es, no se añade nada. Si es cualquier valor distinto de 1, se añade una "s" (por lo que se lee "`packs`" en lugar del singular "`pack`").
19. Se escribe el precio de esta cantidad de caramelos* (con el descuento).
20. Se cierra el bucle `foreach`.
21. El pie de página de la página se incluye utilizando la palabra clave `include` (el pie de página se mostró en ejemplos anteriores).

Ejemplo final de la unidad

```
<?php
$name = 'Ivy';                                // Store the user's name

$greeting = 'Hello';                            // Create initial value for greeting
if ($name) {                                     // If $name has a value
    $greeting = 'Welcome back, ' . $name;        // Create personalized greeting
}

$product = 'Lollipop';                          // Product name
$cost    = 2;                                    // Cost of single pack

for ($i = 1; $i <= 10; $i++) {
    $subtotal   = $cost * $i;                    // Total for this quantity
    $discount   = ($subtotal / 100) * ($i * 4);  // Discount for this quantity
    $totals[$i] = $subtotal - $discount;         // Add discounted price to indexed array
}
?>
```

Ejemplo final de la unidad

```
<?php require 'includes/header.php'; ?>

<p><?= $greeting ?></p>
<h2><?= $product ?> Discounts</h2>
<table>
  <tr>
    <th>Packs</th>
    <th>Price</th>
  </tr>
  <?php foreach ($totals as $quantity => $price) { ?>
  <tr>
    <td>
      <?= $quantity ?>
      pack<?= ($quantity === 1) ? '' : 's'; ?>
    </td>
    <td>
      $<?= $price ?>
    </td>
  </tr>
  <?php } ?>
</table>

<?php include 'includes/footer.php' ?>
```

Ejemplo final de la unidad

The screenshot shows a web browser window with the address bar set to 'localhost'. The main content area displays a website titled 'The Candy Store' with a dark red background. The header includes navigation links for 'HOME | CANDY | ABOUT | CONTACT' and a welcome message 'Welcome back, Ivy'. Below the header, there is a section titled 'LOLLIPOP DISCOUNTS' featuring a table of discounts. To the right of the table is a decorative graphic of lollipops and candy canes. The footer contains copyright information.

PACKS	PRICE
1 pack	\$1.92
2 packs	\$3.68
3 packs	\$5.28
4 packs	\$6.72
5 packs	\$8
6 packs	\$9.12
7 packs	\$10.08
8 packs	\$10.88
9 packs	\$11.52
10 packs	\$12

DWES - U2. Estructuras de control © 2024

148

Ejemplo final de la unidad

Sobre el ejemplo anterior, realiza las siguientes modificaciones:

- En el Paso 6, cambia el valor de \$cost a 10.
- En el Paso 7, actualiza el bucle para que se ejecute 20 veces, mostrando los precios de hasta 20 paquetes de caramelos.

Actividad propuesta

Desarrolla una página web sencilla para un club deportivo, que muestre un saludo personalizado y una tabla con descuentos en la cuota mensual según el número de meses que un socio se inscriba.

Deberás utilizar las técnicas que hemos estudiado en esta unidad:

- sentencias condicionales,
- operador ternario,
- bucles y
- archivos "include" para estructurar y manejar el código.



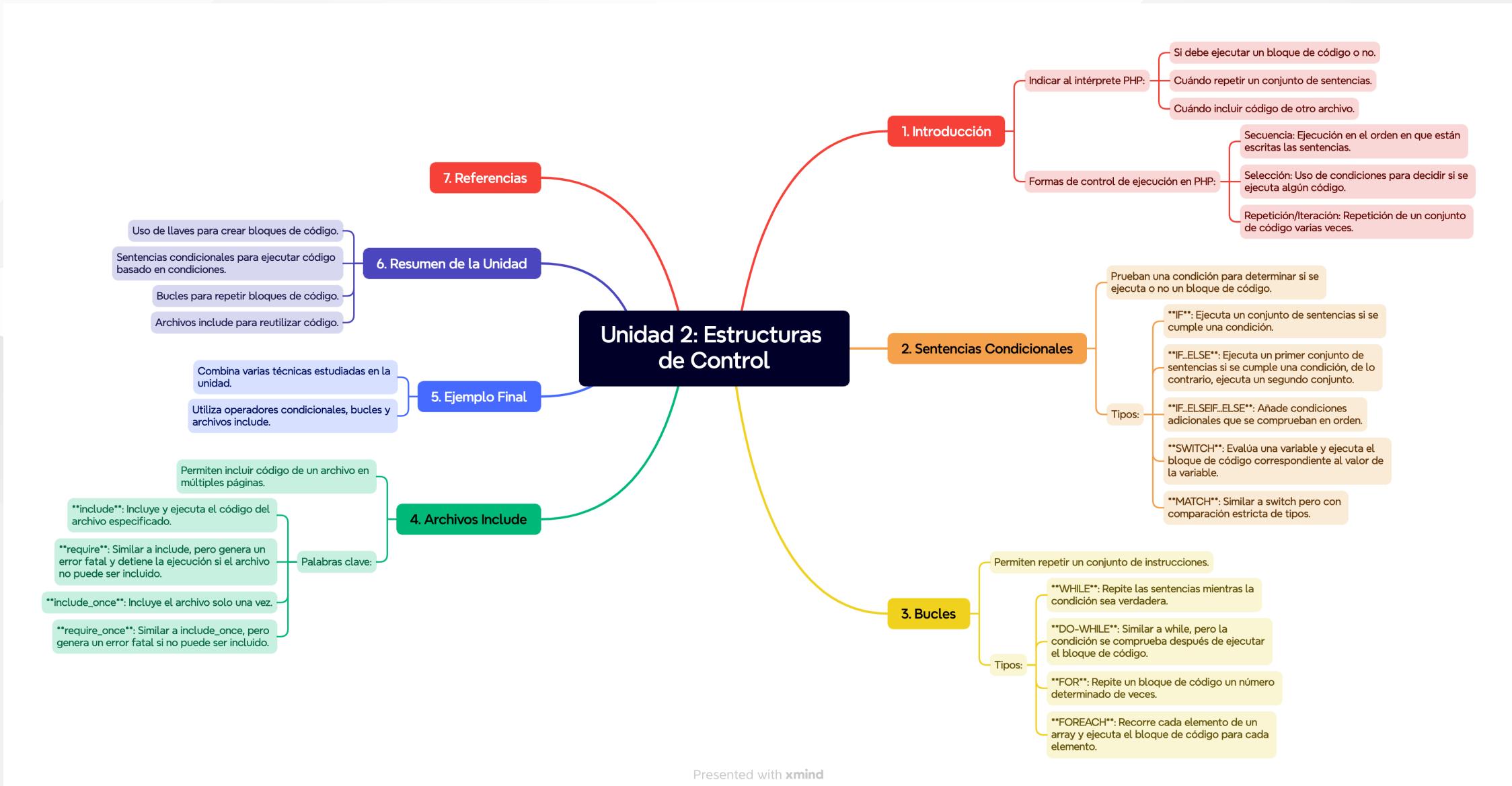
Resumen de la unidad

Resumen de la unidad

- Se pueden utilizar llaves alrededor de un grupo de sentencias relacionadas para formar un **bloque de código**.
- Las **sentencias condicionales** utilizan una condición para determinar si se ejecutan o no las sentencias de un bloque de código.
- Las **condiciones** dan como resultado un **valor verdadero o falso**.
- Existen cinco tipos de sentencias condicionales: `if` , `if... else` , `if... elseif` , `switch` y `match` .

Resumen de la unidad

- Los **bucles** permiten repetir el mismo bloque de código varias veces mientras una condición sigue siendo verdadera.
- Existen cuatro tipos de bucles: `while` , `do... while` , `for` y `foreach` .
- Los **archivos *include*** almacenan código que se utilizará en varias páginas para evitar tener que duplicarlo.



Presented with **xmind**

Referencias

- [Estructuras de control \(php.net\)](#)
- [Estructuras de control en PHP: Guía para programadores \(codigoroot.net\)](#)
- [Estructuras de control if-else y switch \(andresvillar.dev\)](#)
- [Referencia operador match \(php.net\)](#)

Bloque A

Instrucciones básicas de programación

Unidad 2. Estructuras de control

