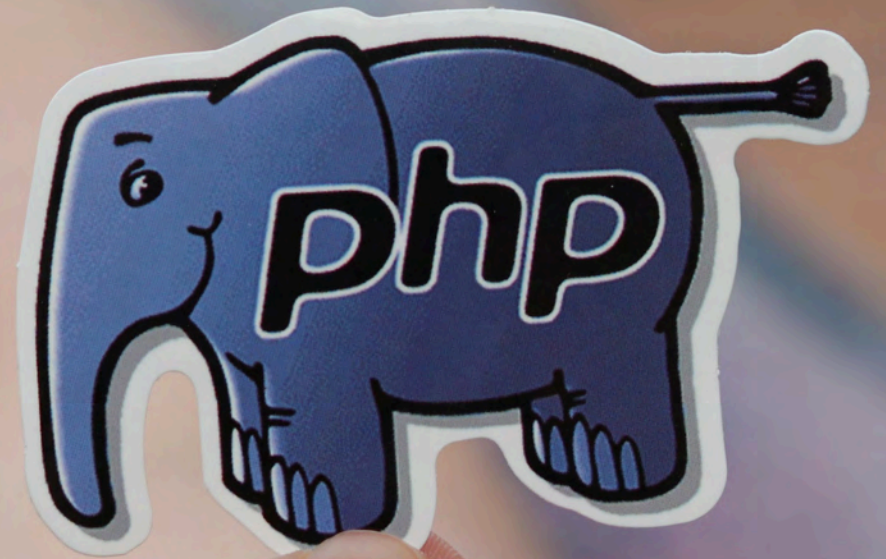


# Bloque B

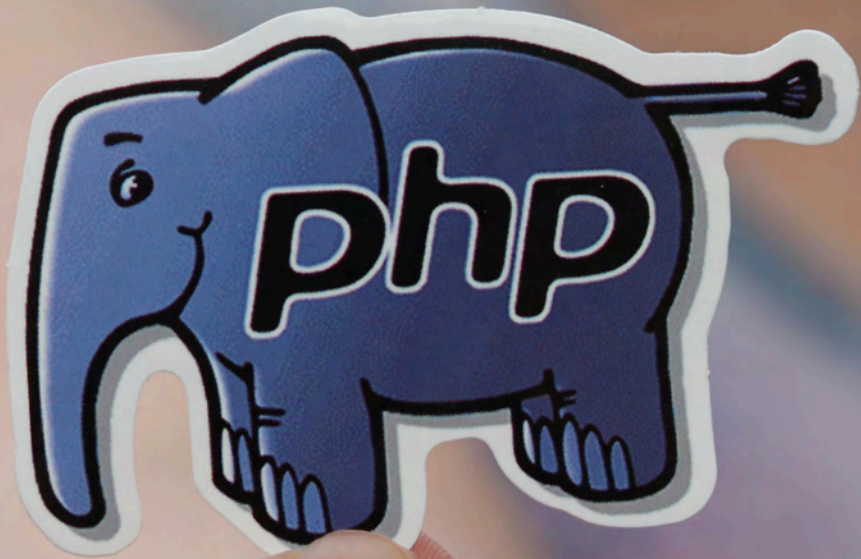
## Páginas Web Dinámicas

### Unidad 8. Fechas & Horas



# Contenidos

1. Introducción
2. Formatos de Fechas & Horas
3. Funciones integradas de fechas & horas
4. Objetos para representar fechas & horas
5. Uso de Intervalos
6. Generación de Eventos recurrentes
7. Gestión de diferentes Zonas horarias



# 1. Introducción

# Introducción

Las fechas y horas se pueden escribir de muchas maneras diferentes. PHP proporciona funciones y clases incorporadas que ayudan a procesar y mostrar fechas y horas en varios formatos.

# Introducción

En este capítulo, aprenderemos las diferentes formas en que el intérprete PHP puede aceptar fechas y horas como entrada, y cómo puede formatearlas como salida cuando son mostradas a los visitantes.

PHP puede trabajar con fechas y horas utilizando:

- Componentes como años, meses, días, horas, minutos y segundos
- Formatos como "1st June 2001", "1/6/2001" o "next Tuesday".
- Marcas de tiempo Unix (también conocido como *timestamp*) que cuentan el número de segundos desde el 1 de enero de 1970 (puede parecer una forma extraña de representar fechas/horas, pero muchos lenguajes de programación las utilizan)

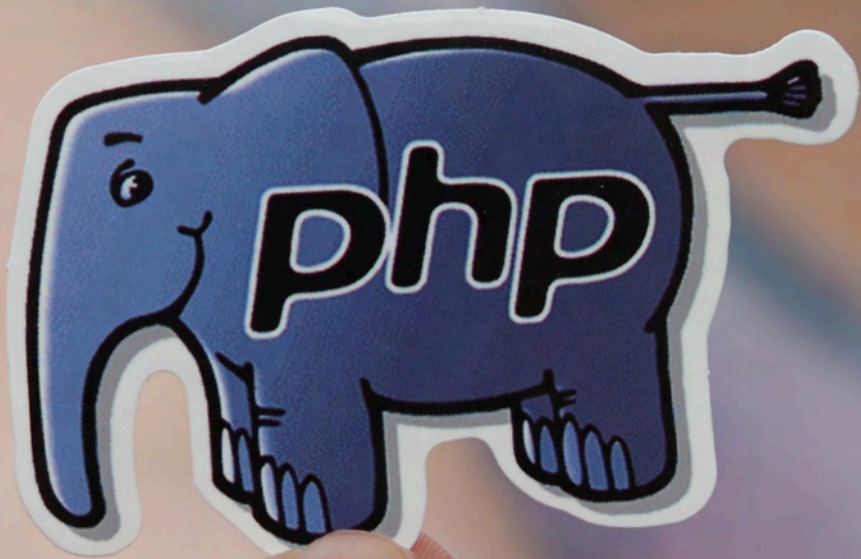
# Introducción

Una vez que hayamos comprendido cómo procesa PHP los formatos de fecha y hora, conoceremos un conjunto de **funciones integradas que generan marcas de tiempo Unix (timestamps) y las convierten a un formato legible** por humanos.

Luego aprenderás como las fechas y horas pueden ser representadas utilizando objetos que son creados utilizando cuatro **clases incorporadas**:

- `DateTime` crea objetos que **representan una fecha y hora específicas**.
- `DateInterval` crea objetos que representan un **intervalo de tiempo** (por ejemplo, una hora o una semana).
- `DatePeriod` crea objetos que representan **eventos recurrentes** que ocurren a intervalos regulares (por ejemplo, cada día, mes o año)
- `DateTimeZone` crea objetos que representan una **zona horaria**





## 2. Formatos de Fechas & Horas

# Formatos de fecha

Las fechas pueden mostrarse de muchas maneras diferentes. PHP utiliza un conjunto de **caracteres de formato** (o **format characters**) para describir cómo se escribe una fecha.

Las fechas pueden constar de los siguientes componentes

- Día de la semana
- Día del mes
- Mes
- Año



# Formatos de fecha

PHP utiliza caracteres de formato para representar cada uno de estos componentes. Por ejemplo, `m-d-Y` representa el formato de fecha `04-06-2022`. Los caracteres de formato le indican al intérprete de PHP cómo serán las fechas:

- Procesadas cuando son recibidas
- Formateadas cuando se muestran

# Formatos de fecha

DAY OF WEEK		
CHARACTER	DESCRIPTION	EXAMPLE
<b>D</b>	First three letters	Sat
.....		
<b>l</b>	Full name	Saturday
DAY OF MONTH		
CHARACTER	DESCRIPTION	EXAMPLE
<b>d</b>	Digits with leading zero	09
.....		
<b>j</b>	Digits no leading zero	9
.....		
<b>S</b>	Suffix	th



# Formatos de fecha

## MONTH

CHARACTER	DESCRIPTION	EXAMPLE
<b>m</b>	Digits with leading zero	04
<b>n</b>	Digits no leading zero	4
<b>M</b>	First three letters	Apr
<b>F</b>	Full name	April

## YEAR

CHARACTER	DESCRIPTION	EXAMPLE
<b>Y</b>	Four digits	2022
<b>y</b>	Two digits	22

# Formatos de fecha

Se pueden agregar espacios, barras inclinadas, guiones y puntos entre los caracteres de formato para separar visualmente cada componente.

A continuación, puede ver cómo los caracteres de formato describen diferentes formas de escribir la misma fecha:

FORMAT CHARACTERS	DATE FORMAT
<b>l m j Y</b>	Saturday April 6 2022
<b>D jS F Y</b>	Sat 6th April 2022
<b>n/j/Y</b>	4/6/2022
<b>m/d/y</b>	04/06/22
<b>m-d-Y</b>	04-06-2022

# Formatos de tiempo

El tiempo puede constar de los siguientes componentes:

- Horas
- Minutos
- Segundos
- am/pm (si no se utiliza la hora de 24 horas)

# Formatos de tiempo

Cada uno de ellos puede representarse utilizando caracteres de formato. Por ejemplo, `g:i a` representa una hora en el formato `8:09 am`. Estos caracteres de formato se utilizan para indicar al intérprete PHP cómo serán las horas:

- Procesadas cuando se reciben
- Formateadas cuando se muestran



# Formatos de tiempo

## HOUR

CHARACTER	DESCRIPTION	EXAMPLE
<b>h</b>	12-hour with leading zero	08
<b>g</b>	12-hour no leading zero	8
<b>H</b>	24-hour with leading zero	08
<b>G</b>	24-hour no leading zero	8

## MINUTE

CHARACTER	DESCRIPTION	EXAMPLE
<b>i</b>	Digits with leading zero	09

# Formatos de tiempo

## SECOND

CHARACTER	DESCRIPTION	EXAMPLE
<b>s</b>	Digits with leading zero	04

## AM/PM

CHARACTER	DESCRIPTION	EXAMPLE
<b>a</b>	Lowercase	am
<b>A</b>	Uppercase	AM



# Formatos de tiempo

Se puede agregar espacios, dos puntos, puntos y paréntesis entre los caracteres de formato para separar visualmente cada componente.

A continuación, puedes ver cómo los caracteres de formato pueden describir diferentes formas de escribir la misma hora:

FORMAT CHARACTERS	DATE FORMAT
<b>g:i a</b>	8:09 am
<b>h:i(A)</b>	08:09(AM)
<b>6:i</b>	08:09

# Especificando fechas y horas mediante Strings

Algunas funciones y métodos permiten especificar una fecha y hora utilizando una cadena de caracteres (String). La cadena debe tener un formato aceptado, como se describe a continuación.

# Especificando fechas y horas mediante Strings

El intérprete PHP puede aceptar fechas utilizando los siguientes formatos de cadena. Si se utilizan barras inclinadas ( / ), el intérprete de PHP espera el mes antes del día del mes (formato estadounidense). Si se utilizan guiones ( - ) o puntos ( . ) como separadores, espera el día del mes antes del mes (formato europeo).

# Especificando fechas y horas mediante Strings

DATE FORMAT	EXAMPLE
<b>d F Y</b>	04 September 2022
<b>jS F Y</b>	4th September 2022
<b>F j Y</b>	September 4 2022
<b>M d Y</b>	Sep 04 2022
<b>m/d/Y</b>	09/04/2022
<b>Y/m/d</b>	2022/09/04
<b>d-m-Y</b>	04-09-2022
<b>n-j-Y</b>	9-4-2022
<b>d.m.y</b>	04.09.22



# Especificando fechas y horas mediante Strings

El intérprete PHP puede aceptar tiempos utilizando los siguientes formatos de cadena. Pero además, también puede:

- Utilizar mayúsculas o minúsculas para *am* y *pm*.
- Utilizar una letra `T` para separar una fecha de una hora. Ejemplo: `2023-10-15T14:30:00` se corresponde con el patrón `YYYY-MM-DDTHH:MM:SS`.
- Añadir una zona horaria a continuación.

# Especificando fechas y horas mediante Strings

12-HOUR TIME FORMAT	EXAMPLE
<b>ga</b>	4am
<b>g:i a</b>	4:08 am
<b>g:i:s a</b>	4:08:37 am
<b>g.i.s a</b>	4.08.37 am
24-HOUR TIME FORMAT	EXAMPLE
<b>H:i</b>	04:08
<b>H:i:s</b>	04:08:37
<b>His</b>	040837
<b>H.i.s</b>	04.08.37

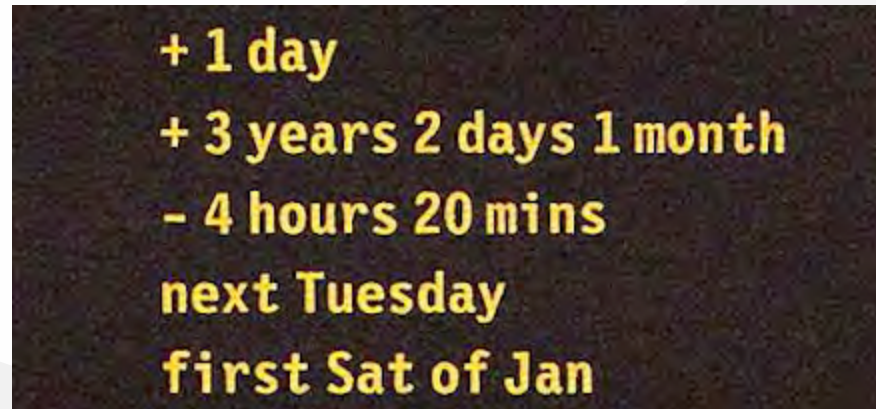


# Especificando fechas y horas mediante Strings

También se pueden usar los siguientes tiempos relativos:

TYPE	RELATIVE TIME
Add/subtract	<b>+</b> <b>-</b>
Quantity	<b>0 - 9</b>
Units of time (can be plural)	<b>day, fortnight, month, year, hour,</b> <b>min, minute, sec, second</b>
Day names	<b>Monday - Sunday</b> and <b>Mon - Sun</b>
Relative terms	<b>next, last, previous, this</b>
Ordinal terms	<b>first - twelfth</b>

# Especificando fechas y horas mediante Strings



**+ 1 day**  
**+ 3 years 2 days 1 month**  
**- 4 hours 20 mins**  
**next Tuesday**  
**first Sat of Jan**

**First / Last** sólo funciona para los días del mes.

Si no se especifica ninguna hora, se fijará a medianoche.

# Unix Timestamps

Las marcas de tiempo Unix (*timestamps*) representan **fechas y horas utilizando el número de segundos transcurridos desde la medianoche del 1 de enero de 1970.**

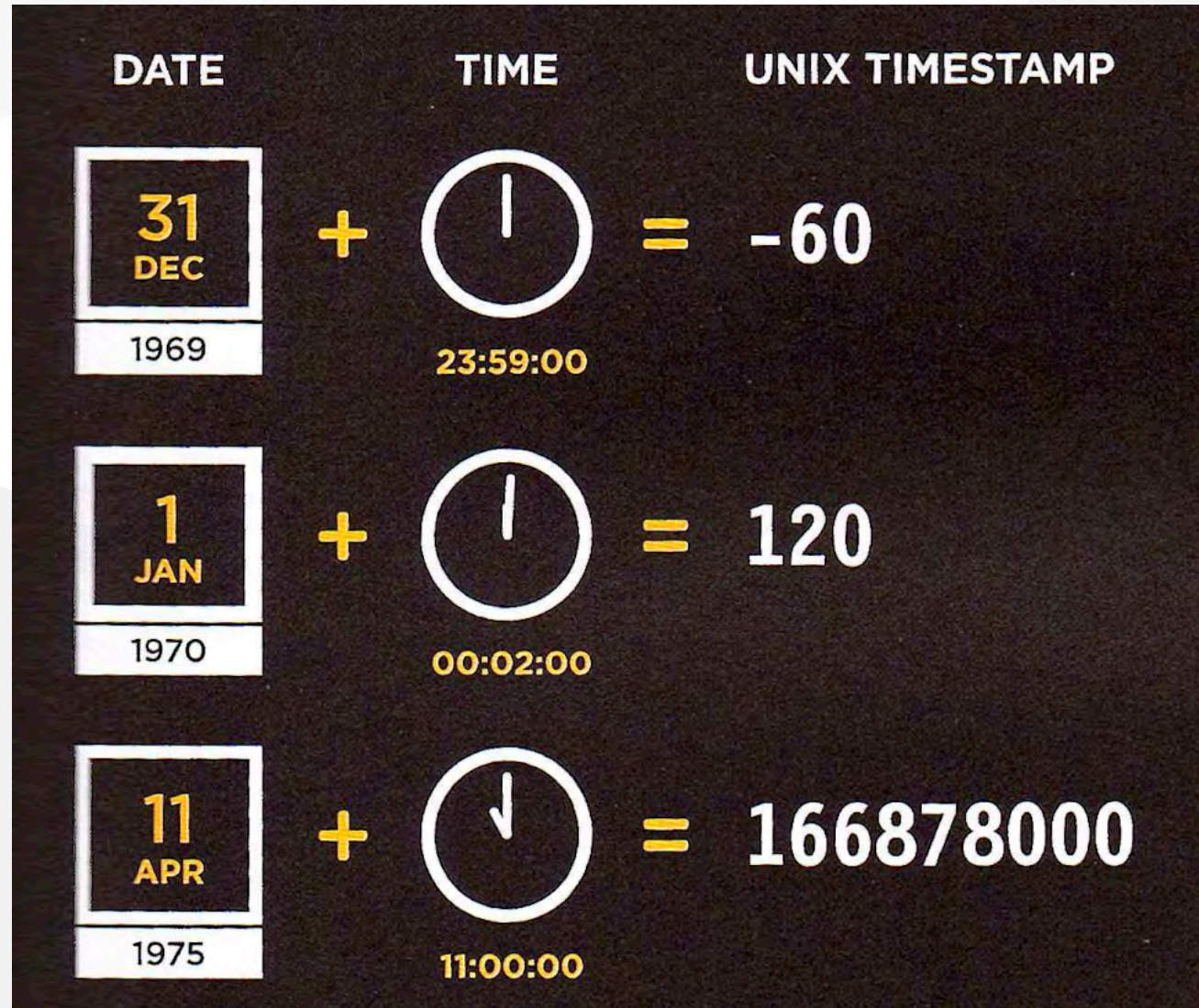
El intérprete PHP nos permite especificar y recuperar fechas y horas utilizando marcas de tiempo Unix.

# Unix Timestamps

En los siguientes ejemplos, se puede ver algunos ejemplos específicos de fechas y horas, seguidos de su correspondiente marca de tiempo Unix.

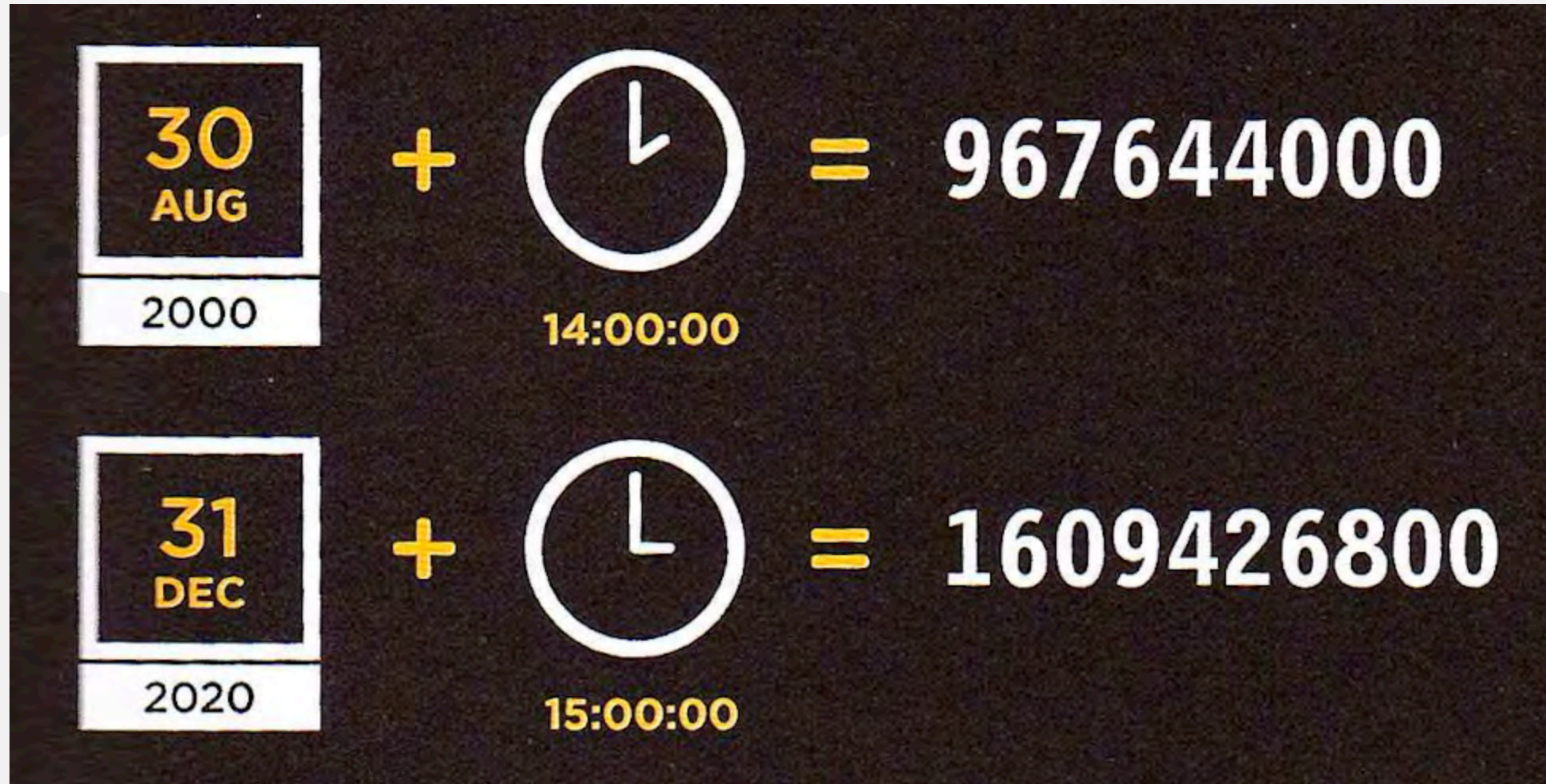


# Unix Timestamps





# Unix Timestamps



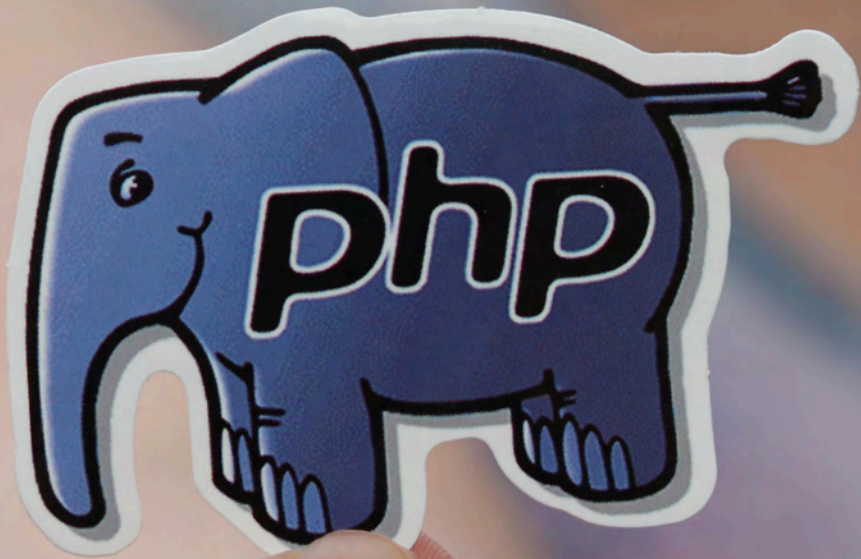
# Unix Timestamps

Las fechas **anteriores al 1 de enero de 1970** se escriben utilizando números **negativos**.

Como veremos a continuación, las funciones y clases incorporadas de PHP nos ayudan a trabajar con marcas de tiempo Unix. Utilizan los caracteres de formato que acabamos de conocer para describir como las funciones y clases deben **transformar una marca de tiempo Unix en algo que sea legible por humanos**.

La **fecha máxima** de una marca de tiempo Unix es el **19 de enero de 2038**.

Unix es un sistema operativo que se desarrolló en la década de 1970.



### 3. Funciones integradas de fechas & horas

# Funciones integradas de fecha y hora

PHP tiene funciones incorporadas que pueden **crear marcas de tiempo Unix para representar fechas y horas.**

También tiene funciones incorporadas para **convertir estas marcas de tiempo Unix en un formato que sea fácil de leer.**



# Funciones integradas de fecha y hora

Las tres funciones siguientes se utilizan para crear una marca de tiempo Unix.

Si no son capaces de crear una marca de tiempo, devuelven false.

Si no se especifica una hora para `strtotime()` o `mktime()`, la hora se establece a medianoche.



# Funciones integradas de fecha y hora

FUNCTION	DESCRIPTION
<code>time()</code>	Returns the current date and time as a Unix timestamp.
<code>strtotime(\$string)</code>	Converts a string to a Unix timestamp (accepts formats shown on p314).
	EXAMPLE
	<code>strtotime('December 1 2020');</code>
	<code>strtotime('1/12/2020');</code>
<code>mktime(H, i, s, n, j, Y)</code>	Converts date/time components (in arguments) into a Unix timestamp.
	EXAMPLE
	REPRESENTS
	<code>mktime(17, 01, 05, 2, 1, 2001);</code> February 1 2001 17:01:05
	<code>mktime(01, 30, 45, 4, 29, 2020);</code> April 29 2020 01:30:45

# Funciones integradas de fecha y hora

La función `date()` convierte las marcas de tiempo Unix a un formato legible por humanos.

El formato se especifica utilizando los **caracteres de formato** que hemos visto previamente en esta unidad.

Si no se indica ninguna marca de tiempo, se mostrará la **fecha y hora actuales**.

# Funciones integradas de fecha y hora

FUNCTION	DESCRIPTION
<code>date(\$format[, \$timestamp])</code>	Returns a Unix timestamp formatted in a human-readable way: The first parameter specifies how the date should be formatted. The second parameter is the Unix timestamp to format.
EXAMPLE	OUTPUT
<code>date('Y');</code>	Current year
<code>date('d-m-y h:i a', 1609459199);</code>	31-12-20 11:59 pm
<code>date('D j M Y H:i:a', 1609459199);</code>	Thu 31 Dec 2020 23:59:59

## Ejemplo: Funciones de fecha

A continuación vemos un ejemplo donde se calcula las fechas de inicio y fin de un periodo de ofertas (del 1 de enero al 1 de febrero de 2021), las formatea para mostrarlas con el día de la semana y el formato `d M Y`, e incluye una cabecera y pie de página desde archivos externos.



# Ejemplo: Funciones de fecha

1. La función `strtotime()` crea una marca de tiempo Unix para representar una fecha en el pasado. Se almacena en una variable llamada `$start`.
2. La función `mktime()` crea una marca de tiempo Unix para representar una fecha un mes más tarde. Se almacena en `$end`.
3. La función `date()` convierte estas marcas de tiempo Unix en un formato legible utilizando el:
  - Nombre del día
  - Día (con cero a la izquierda)
  - Mes (tres primeras letras)
  - Año (cuatro dígitos)

Estas se almacenan en las variables `$start_date` y `$end_date`.

## Ejemplo: Funciones de fecha

4. Se muestra la versión legible por humanos de cada fecha.
5. El archivo include para el pie de página añade un aviso de copyright. El año se escribe utilizando la función `date()`. Como no se indica ninguna marca de tiempo, utiliza la fecha actual.

# Ejemplo: Funciones de fecha

date-functions.php :

```
<?php
① $start      = strtotime('January 1 2021');
② $end        = mktime(0, 0, 0, 2, 1, 2021);
③ [ $start_date = date('l, d M Y', $start);
    $end_date   = date('l, d M Y', $end);
    ?>
    <?php include 'includes/header.php'; ?>

④ [ <p><b>Sale starts:</b> <?= $start_date ?></p>
    <p><b>Sale ends:</b> <?= $end_date ?></p>

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

includes/footer.php :

```
⑤ <footer>&copy; <?php echo date('Y')?></footer> ...
```

NOTA: Si la hora está desfasada unas horas, comprueba el ajuste de zona horaria por defecto en el archivo *php.ini*.

# Ejemplo: Funciones de fecha





# Actividad: Funciones de fecha

Vas a desarrollar un pequeño sistema para gestionar la **fecha de inicio y finalización de un evento** en un sitio web. El sistema permitirá al usuario ver las fechas actuales de los eventos, mostrar cuántos días faltan para que inicie o termine un evento, y ajustarse automáticamente según la fecha actual. También calcularás si el evento ya ha finalizado o está en curso.



# Actividad: Funciones de fecha

## Instrucciones:

1. **Fecha actual del sistema:** Utiliza `time()` para obtener la fecha y hora actual y formatearla usando `date()` para mostrar la fecha de hoy.

## 2. Fechas del evento:

- Define la **fecha de inicio** del evento con `strtotime()`. Usa una fecha que esté en el futuro.
- Define la **fecha de finalización** con `mktime()`, especificando manualmente el día, mes y año.



# Actividad: Funciones de fecha

## 3. Mostrar la cuenta regresiva:

- Calcula cuántos días faltan para que inicie el evento.
- Calcula cuántos días faltan para que termine el evento, desde la fecha actual.

## 4. Mensajes condicionales:

- Si la fecha actual es anterior al inicio del evento, muestra un mensaje que indique cuántos días faltan para que comience.
- Si la fecha actual está entre el inicio y el final del evento, muestra un mensaje indicando que el evento está en curso.
- Si el evento ya ha finalizado, muestra un mensaje indicando que ha concluido.





## 4. Objetos para representar fechas & horas



# Objetos para representar fechas y horas

La clase `DateTime` de PHP crea un **objeto que representa una fecha y hora**. Sus métodos pueden devolver la fecha y la hora que el objeto representa, ya sea en un formato legible por humanos o como una marca de tiempo Unix.

# Objetos para representar fechas y horas

Para crear un objeto `DateTime`, deberemos utilizar:

- Una variable que contenga el objeto
- El operador de asignación
- La palabra clave `new`
- El nombre de la clase `DateTime`
- Un par de paréntesis

# Objetos para representar fechas y horas

Entre paréntesis, añadiremos la **fecha/hora que debe representar el objeto**. Puedes utilizar cualquiera de los formatos de fecha y hora que se mostraron previamente en el apartado "*Especificando fechas y horas mediante Strings*" de esta unidad. El valor debe colocarse **entre comillas**.

# Objetos para representar fechas y horas

- Si no se especifica una fecha y hora, el objeto utilizará **la fecha y hora actuales**.
- Si se especifica una fecha pero no una hora, el objeto utilizará **la medianoche del día especificado**.

```
$date = new DateTime('2001-02-01 15:01:05');
```

The diagram illustrates the components of the code snippet above. Brackets are placed under each part of the code with labels below them: a bracket under '\$date' is labeled 'VARIABLE', a bracket under 'DateTime' is labeled 'CLASS NAME', and a bracket under the string '2001-02-01 15:01:05' is labeled 'DATE AND TIME'.



# Objetos para representar fechas y horas

También se puede utilizar la función `date_create_from_format()` para crear un objeto `DateTime`.

- El primer argumento es el formato en el que se suministrarán la fecha y la hora.
- El segundo argumento es la fecha y la hora en el formato especificado. Ambos argumentos van entre comillas.

```
$date = date_create_from_format('j-M-Y', '15-Jan-2020');
```

VARIABLE      FUNCTION      FORMAT      DATE/TIME

# Objetos para representar fechas y horas

Los siguientes métodos del objeto `DateTime` devuelven la fecha y la hora que representa el objeto.

- Para obtener la fecha/hora en un formato legible, utiliza el método `format()`.
- Para obtener la fecha/hora como una marca de tiempo Unix, utiliza el método `getTimestamp()`.

METHOD	DESCRIPTION
<code>format(\$format[, \$DateTimeZone])</code>	Gets the date and time in the specified format. The optional second parameter sets a time zone (see p326).
<code>getTimestamp()</code>	Returns the Unix timestamp for the date and time the object represents.

# Ejemplo: Objeto DateTime

1. Este ejemplo comienza creando un objeto utilizando la clase `DateTime` . Se almacena en una variable llamada `$start` .
2. Se crea un segundo objeto `DateTime` utilizando la función `date_create_from_format()` .
  - El primer parámetro especifica el formato en el que se proporcionará la fecha.
  - El segundo parámetro establece la fecha y la hora.
  - Este objeto se almacena en una variable llamada `$end` .

## Ejemplo: Objeto DateTime

3. La fecha y hora de inicio se escriben en la página utilizando el método `format()` del objeto `DateTime`. El argumento especifica el formato en el que deben escribirse la fecha y la hora.
4. La fecha (no la hora) final se escribe en la página utilizando el método `format()` del objeto `^`. Su parámetro especifica el formato en el que debe escribirse la fecha.
5. La hora final se escribe por separado y también utiliza el método `format()`. Esto muestra cómo se puede escribir simplemente la fecha o la hora que contiene el objeto.



# Ejemplo: Objeto DateTime

```
<?php
① $start = new DateTime('2021-01-01 00:00');
② $end   = date_create_from_format('Y-m-d H:i',
    '2021-02-01 00:00');
?>
<?php include 'includes/header.php'; ?>

<p><b>Sale starts:</b>
③ <?= $start->format('l, jS M Y H:i') ?></p>
<p><b>Sale ends:</b>
④ <?= $end->format('l, jS M Y') ?> <b>at</b>
⑤ <?= $end->format('H:i') ?></p>

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

# Ejemplo: Objeto DateTime





# Actividad: Objeto DateTime

Eres un desarrollador web y estás creando un sistema para gestionar eventos en una página web. En este sistema, los usuarios pueden crear eventos futuros, y el sistema debe procesar las fechas de los eventos, almacenarlas y mostrarlas en diferentes formatos dependiendo de las necesidades del cliente.



# Actividad: Objeto DateTime

## Requisitos:

1. Los usuarios ingresan las fechas en un formato específico: `d/m/Y H:i:s` (por ejemplo, `16/10/2024 15:30:00` ).
2. El sistema debe convertir esta fecha en un objeto `DateTime` utilizando la función `date_create_from_format()` .
3. Una vez que se ha creado el objeto `DateTime` , el sistema debe:
  - Mostrar la fecha en formato: `Y-m-d H:i:s` (por ejemplo, `2024-10-16 15:30:00` ).
  - Obtener el *timestamp* UNIX correspondiente a esa fecha usando el método `getTimestamp()` .
  - Mostrar la fecha en formato legible como "16 de octubre de 2024, 15:30".



# Actualizando fecha y hora en objetos DateTime

Una vez creado un objeto con la clase DateTime, puedes utilizar los siguientes métodos para establecer o actualizar la fecha/hora que representa.

- Los métodos que establecen una fecha/hora **sobrescriben** cualquier fecha/hora que el objeto represente actualmente.
- Los métodos `add()` y `sub()` utilizan un objeto `DateInterval` que introduciremos en la siguiente sección.

# Actualizando fecha y hora en objetos DateTime

METHOD	DESCRIPTION
<code>setDate(\$year, \$month, \$day)</code>	Sets a date for the object
<code>setTime(\$hour, \$minute [, \$seconds][, \$microseconds])</code>	Sets a time for the object
<code>setTimestamp(\$timestamp)</code>	Sets the date/time using a Unix timestamp
<code>modify(\$DateFormat)</code>	Updates the date/time using a string
<code>add(\$DateInterval)</code>	Adds an interval of time using a <code>DateInterval</code> object (see p322)
<code>sub(\$DateInterval)</code>	Subtracts an interval of time using a <code>DateInterval</code> object (see p322)

# Actualizando fecha y hora en objetos DateTime

Cuando el intérprete de PHP crea una variable, puede contener un valor escalar o un array en la variable. Cuando el intérprete de PHP crea un objeto, almacena el objeto en una ubicación independiente en su memoria. Entonces, si ese objeto es almacenado en una variable, **la variable almacena la ubicación de donde el objeto fue creado en la memoria del intérprete PHP** (en lugar de almacenar el objeto mismo).

Esto significa que, si se crea un objeto y se almacena en una variable, luego se declara una segunda variable y se asigna el mismo objeto como valor de esa variable, **ambas variables contendrán la ubicación del mismo objeto**.

# Actualizando fecha y hora en objetos DateTime

Por lo tanto, si actualiza el objeto en una variable, también se actualizará en la otra:

```
$start = new DateTime('2020/12/1');  
$end   = $start;  
// Both variables point to same object  
$end->modify('+1 day');
```

Para evitar esto, se puede utilizar la palabra clave llamada `clone` para crear una copia del objeto:

```
$start = new DateTime('2020/12/1');  
$end   = clone $start;  
// Only object in $end is modified  
$end->modify('+1 day');
```

# Ejemplo: Cómo ajustar la fecha y hora en un objeto `DateTime`

1. Se crea un nuevo objeto utilizando la clase `DateTime`. El objeto se almacena en una variable llamada `$start`; contiene la fecha y hora actuales.
2. La fecha se establece utilizando el método `setDate()` del objeto `DateTime`.
3. La hora se actualiza utilizando el método `setTime()` del objeto `DateTime`.



# Ejemplo: Cómo ajustar la fecha y hora en un objeto `DateTime`

4. El objeto que se almacena en `$start` se clona utilizando la palabra clave `clone` y el clon se almacena en una variable llamada `$end`.
5. El método `modify()` del objeto `DateTime` se utiliza para actualizar el objeto almacenado en `$end` de forma que represente una fecha y hora que sea 2 horas 15 minutos posterior a la fecha y hora que estaba representada por el objeto almacenado en `$start`.
6. Las fechas y horas que representan ambos objetos se escriben utilizando el método `format()`.

# Ejemplo: Cómo ajustar la fecha y hora en un objeto DateTime

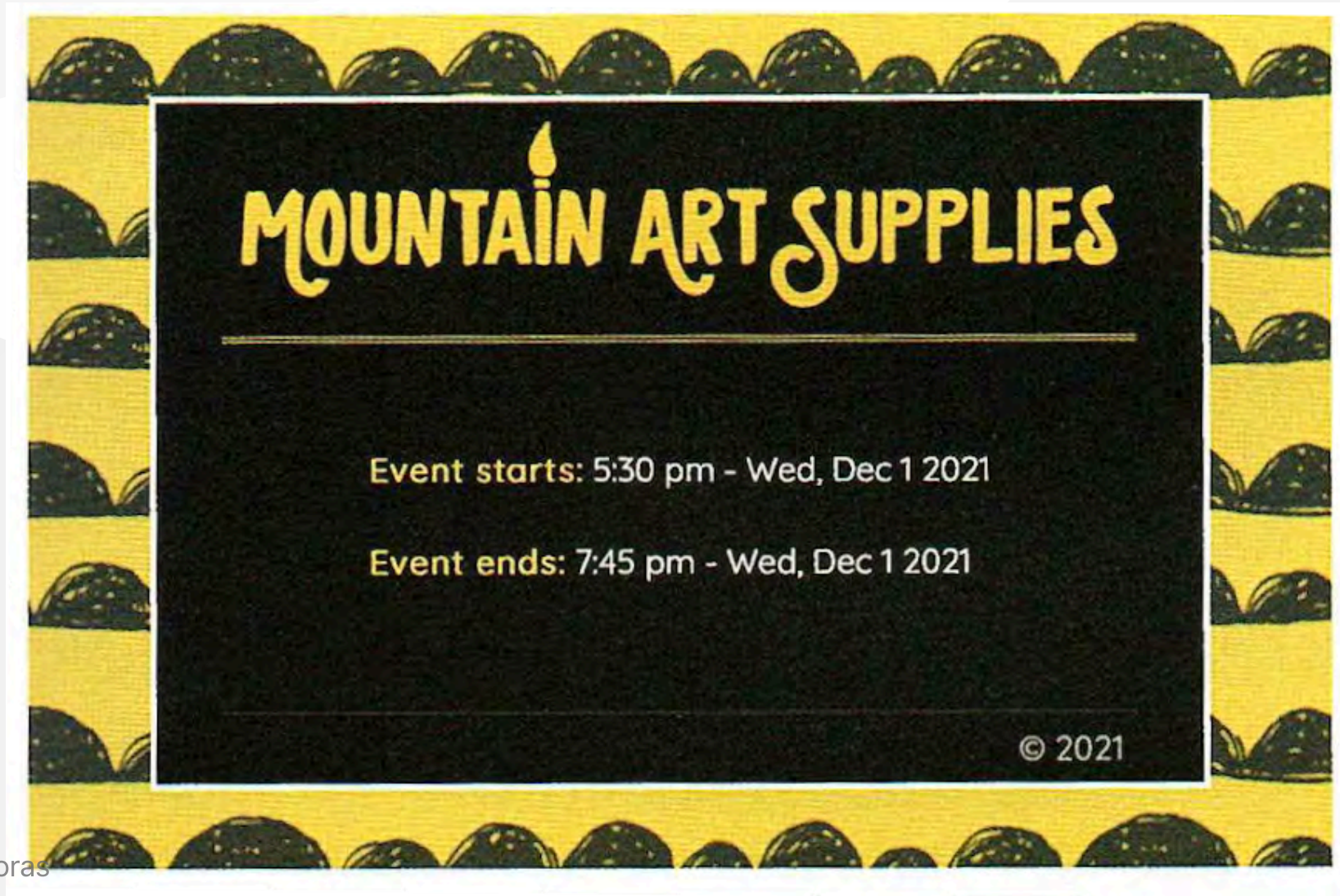
```
<?php
① $start = new DateTime();
② $start->setDate(2021, 12, 01);
③ $start->setTime(17, 30);
④ $end = clone $start;
⑤ $end->modify('+2 hours 15 min');
?>
<?php include 'includes/header.php'; ?>

<p><b>Event starts:</b>
⑥ <?= $start->format('g:i a - D, M j Y') ?></p>

<p><b>Event ends:</b>
⑥ <?= $end->format('g:i a - D, M j Y') ?></p>

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

# Ejemplo: Cómo ajustar la fecha y hora en un objeto DateTime





# Actividad: Ajustar fecha y hora en un objeto DateTime

Estás desarrollando un sistema de gestión de eventos para un sitio web, y los administradores del sistema necesitan poder ajustar las fechas de los eventos de manera dinámica en función de cambios de último momento. Entre los requisitos, los administradores necesitan cambiar la fecha y la hora del evento, ajustar la fecha en función de ciertos plazos, y convertir una fecha proporcionada en formato UNIX a una fecha legible.

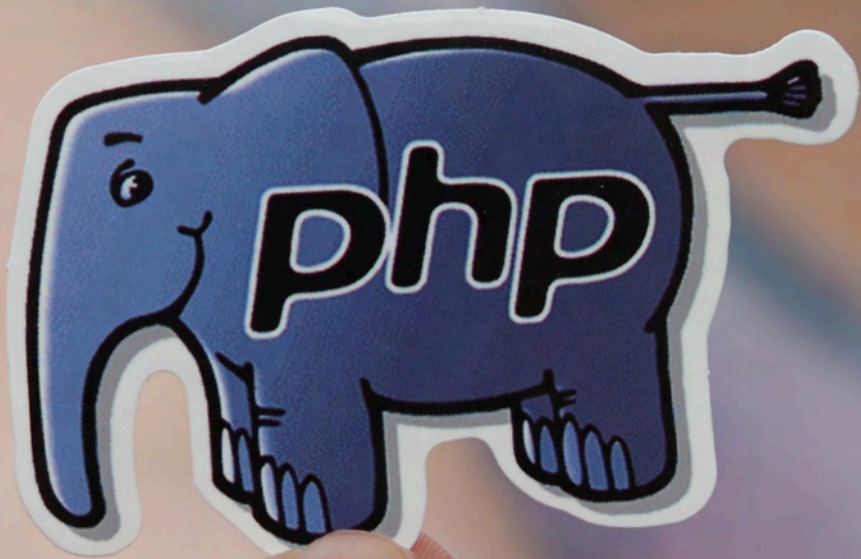


# Actividad: Ajustar fecha y hora en un objeto DateTime

## Requisitos:

1. Un evento tiene inicialmente una fecha y hora asignada: `16/10/2024 15:30:00`.
2. El administrador debe poder:
  - Cambiar la fecha del evento utilizando `setDate()`.
  - Cambiar la hora del evento utilizando `setTime()`.
  - Ajustar la fecha del evento a partir de un *timestamp* UNIX utilizando `setTimestamp()`.
  - Modificar la fecha utilizando `modify()` para sumar o restar días y horas.





## 5. Uso de Intervalos

# Representar un intervalo usando DateInterval

La clase `DateInterval` se utiliza para crear un objeto que **representa un intervalo de tiempo** medido en años, meses, semanas, días, horas, minutos y segundos.

# Representar un intervalo usando DateInterval

Los métodos `add()` y `sub()` del objeto `DateTime` utilizan un objeto `DateInterval` para **especificar un intervalo de tiempo que añadir o eliminar de la fecha/hora actual**. La duración del intervalo se especifica utilizando el formato que se muestra en la tabla siguiente.

La letra **P** precede a cada intervalo. La letra **T** precede a un periodo de tiempo.

```
$interval = new DateInterval('P1M');
```

VARIABLE                      CLASS NAME      INTERVAL



# Representar un intervalo usando DateInterval

INTERVAL	REPRESENTED
1 year	P1Y
2 months	P2M
3 days	P3D
1 year, 2 months, 3 days	P1Y2M3D
1 hour	PT1H
30 mins	PT30M
15 seconds	PT15S
1 hour, 30 minutes, 15 seconds	PT1H30M15S
1 year, 1 day, 1 hour, and 30 minutes	P1Y1DT1H30M

# Representar un intervalo usando DateInterval

El método `diff()` del objeto `DateTime` (abreviatura de *difference*) compara dos objetos `DateTime` y devuelve un objeto `DateInterval` que representa el intervalo entre ellos.

Para mostrar el intervalo almacenado en un objeto `DateInterval`, podemos utilizar su método `format()`. Su argumento es una cadena que utiliza los caracteres de formato que se muestran a continuación, donde queremos que se muestre el intervalo.

```
STRING TO DISPLAY
├──────────────────┤
$interval->format('%h hours %i minutes');
└──┬──┘   └──┬──┘
  INTERVAL INTERVAL
```

# Representar un intervalo usando DateInterval

INTERVAL	DESCRIPTION
%y	Years
%m	Months
%d	Days
%h	Hours
%i	Minutes
%s	Seconds
%f	Microseconds

## Ejemplo: Objeto DateInterval

Este ejemplo calcula la diferencia entre la fecha y hora actuales y un evento programado para el 31 de diciembre de 2025 a las 20:30, mostrando un "countdown" (cuenta regresiva) en años, meses y días, y además muestra una fecha límite de un mes a partir de hoy para comprar entradas con descuento del 50%, formateando ambas fechas para su visualización en la página.



# Ejemplo: Objeto DateTimeInterval

1. La fecha y hora actuales se representan utilizando un objeto `DateTime`, y se almacenan en `$today`.
2. La fecha de un evento se representa utilizando un objeto `DateTime` almacenado en una variable llamada `$event`.
3. El método `diff()` del objeto `DateTime` obtiene el intervalo de tiempo entre ahora y la fecha del evento. El objeto `DateTimeInterval` que se devuelve se almacena en `$countdown`.
4. La fecha y hora actuales se almacenan en `$earlybird`.

## Ejemplo: Objeto DateInterval

5. Un intervalo de un mes es representado por un objeto `DateInterval` almacenado en `$interval`.
6. El método `add()` del objeto `DateTime` añade el intervalo en el objeto `DateInterval` a la fecha actual en `$earlybird`.
7. Se escribe el intervalo almacenado en `$countdown`. Observa cómo se antepone el símbolo `%` a los caracteres de formato que representan los intervalos.
8. Se escribe la fecha almacenada en `$earlybird`.

# Ejemplo: Objeto DateTimeInterval

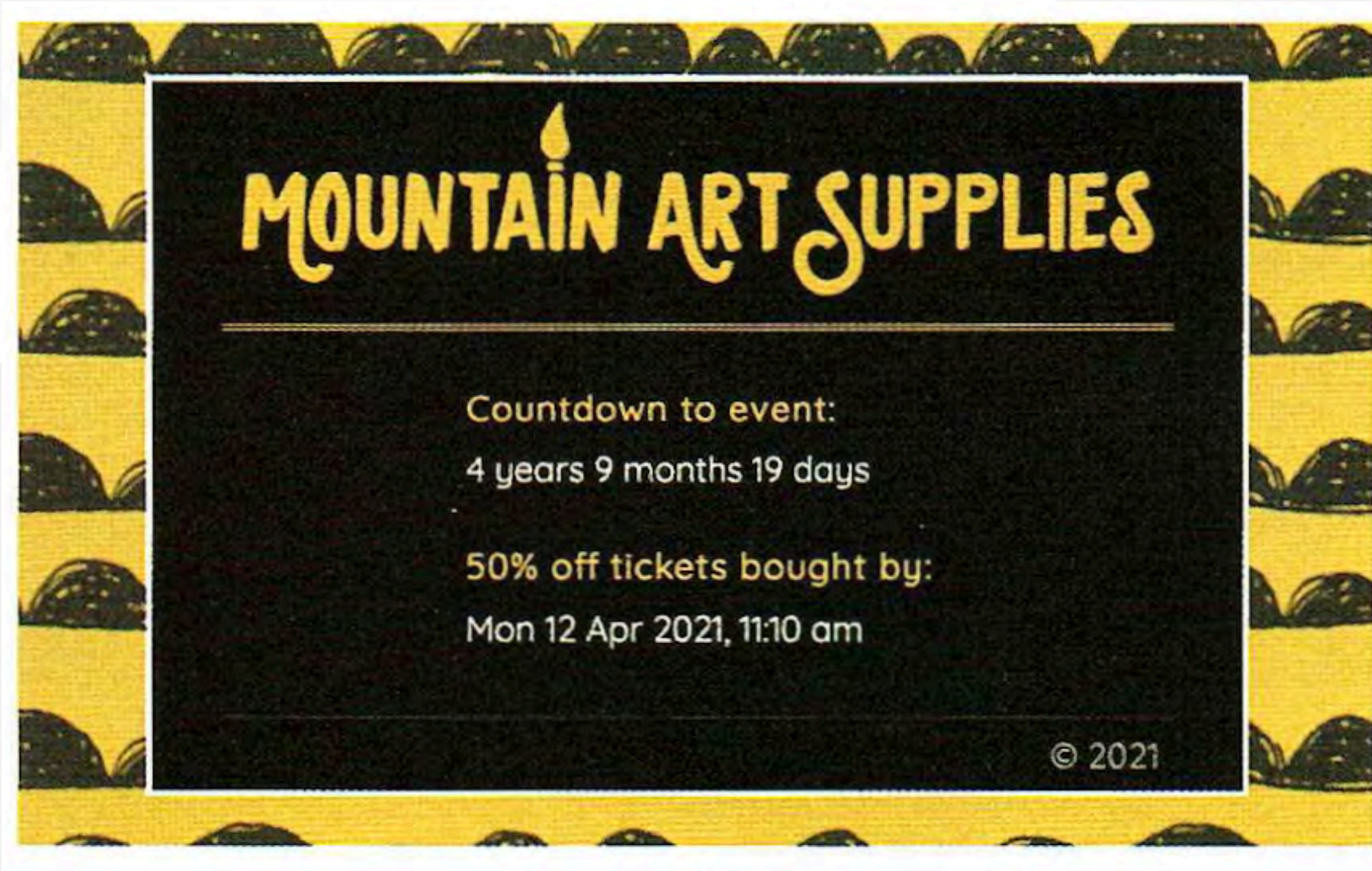
```
<?php
① $today      = new DateTime();
② $event      = new DateTime('2025-12-31 20:30');
③ $countdown  = $today->diff($event);

④ $earlybird  = new DateTime();
⑤ $interval   = new DateInterval('P1M');
⑥ $earlybird->add($interval);
?>
<?php include 'includes/header.php'; ?>

<p><b>Countdown to event:</b><br>
⑦ <?= $countdown->format('%y years %m months %d days') ?>
</p>
<p><b>50% off tickets bought by:</b><br>
⑧ <?= $earlybird->format('D d M Y, g:i a') ?>
</p>

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

# Ejemplo: Objeto DateInterval





# Actividad: Intervalos

Un cliente ha solicitado que su sistema de eventos no solo muestre las fechas de inicio y fin de los eventos, sino también la duración total del evento en un formato fácil de entender. Por ejemplo, si el evento dura 2 días y 3 horas, el sistema debe mostrarlo de manera clara.

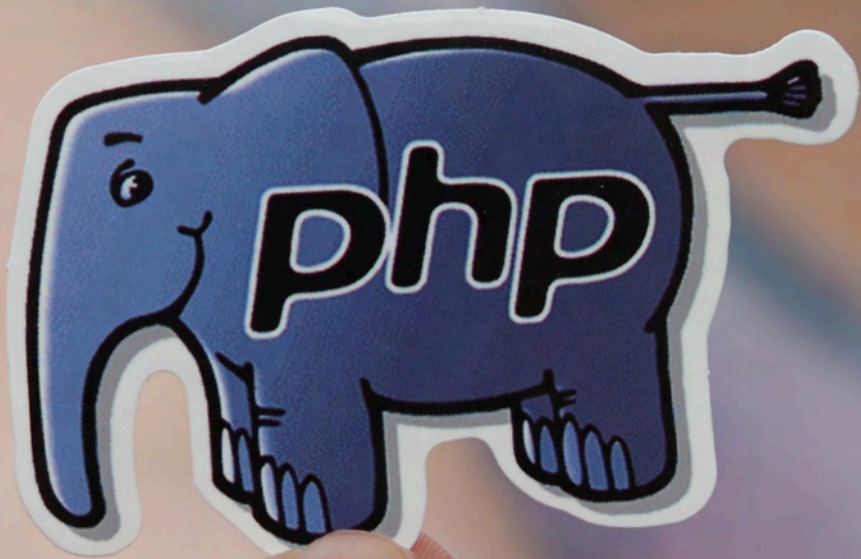


# Actividad: Intervalos

## Requisitos:

1. El sistema debe permitir ingresar dos fechas: la fecha y hora de inicio del evento y la fecha y hora de fin del evento.
2. El sistema debe calcular la duración total entre ambas fechas usando la clase `DateInterval()`.
3. El sistema debe mostrar el intervalo de tiempo en un formato personalizado como "X días, Y horas, Z minutos".
4. Utilizar el método `format()` de `DateInterval` para mostrar la duración.
5. Calcula el número de horas totales que dura el evento, mostrando el tiempo total en horas y minutos.





## 6. Generación de Eventos recurrentes



# Eventos recurrentes mediante DatePeriod

La clase `DatePeriod` puede crear un objeto que **almacena un conjunto de objetos `DateTime` que se producen a intervalos regulares entre una fecha de inicio y una fecha final**. A continuación, se puede realizar un **bucle** a través de cada uno de los objetos `DateTime` que crea.

# Eventos recurrentes mediante DatePeriod

Para crear un objeto `DatePeriod`, se necesitan tres cosas:

- Una fecha de inicio (objeto `DateTime`)
- La frecuencia del evento (un objeto `DateInterval`)
- Una fecha final para el periodo

La **fecha final** del periodo de fechas puede ser:

- Un objeto `DateTime` ; o
- Un número entero que indica **cuántas veces debe producirse el evento** (después de la fecha de inicio)

# Eventos recurrentes mediante DatePeriod

Cuando se crea un objeto `DatePeriod`, éste contiene una serie de objetos `DateTime`; cada uno representa un punto en el tiempo entre la fecha de inicio y la fecha final en el intervalo especificado en el objeto `DateInterval`.

```
$period = new DatePeriod($start, $interval, $end);
```

VARIABLE      CLASS NAME      START DATE/TIME      INTERVAL      END DATE/TIME



# Eventos recurrentes mediante DatePeriod

Es posible acceder a cada objeto `DateTime` de un objeto `DatePeriod` utilizando un bucle `foreach`.

Al igual que con todos los bucles, utiliza un nombre de variable para mantener cada objeto `DateTime` a medida que realiza bucles a través de ellos.

En el bloque de código, se puede utilizar los métodos del objeto `DateTime` para trabajar con esa fecha/hora.

DatePeriod OBJECT HOLDS  
DateTime OBJECTS

VARIABLE NAME TO REPRESENT  
EACH DateTime OBJECT

```
foreach($period as $occurrence) {  
    echo $occurrence->format('Y jS F');  
}
```

## Ejemplo: Objeto DatePeriod

Este ejemplo genera una lista de fechas, una por cada mes, desde el 1 de enero de 2025 hasta el 1 de enero de 2026, y las muestra formateadas con el día de la semana y la fecha completa, recorriendo cada una de ellas dentro de un bucle `foreach` .

# Ejemplo: Objeto DatePeriod

1. La variable `$start` contiene un objeto DateTime que representa el 1 de enero de 2025.
2. La variable `$end` contiene un objeto DateTime que representa el 1 de enero de 2026.
3. La variable `$interval` contiene un objeto DateInterval que representa un mes.
4. La variable `$period` contendrá el objeto DatePeriod. Requiere tres parámetros (los valores se definieron en los Pasos 1-3):
  - Una fecha de inicio
  - Un intervalo
  - Una fecha finalContendrá doce objetos `DateTime` (uno por cada mes del año 2025).



## Ejemplo: Objeto DatePeriod

5. Un bucle foreach recorre cada uno de los objetos Datetime. Dentro del bucle, `$event` representa cada objeto DateTime.

Para cada uno de ellos:

6. El método `format()` se utiliza para escribir el nombre del día y luego el mes, el día y el año.

# Ejemplo: Objeto DatePeriod

```
<?php
① $start      = new DateTime('2025-1-1');
② $end        = new DateTime('2026-1-1');
③ $interval   = new DateInterval('P1M');
④ $period     = new DatePeriod($start, $interval, $end);
?>
<?php include 'includes/header.php'; ?>

<p>
⑤ <?php foreach ($period as $event) { ?>
⑥ [    <b><?= $event->format('l') ?></b>,
      <?= $event->format('M j Y') ?></b><br>
    <?php } ?>
</p>

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

# Ejemplo: Objeto DatePeriod





# Actividad: Eventos recurrentes

En un sistema de gestión de eventos, se necesita permitir que los administradores generen eventos recurrentes (por ejemplo, una reunión que ocurre cada semana durante 2 meses).

El sistema debe poder:

1. Crear un evento inicial con una fecha y hora dadas.
2. Definir un intervalo de repetición, como "cada semana" o "cada 15 días".
3. Generar y mostrar todas las fechas futuras del evento durante un periodo determinado (por ejemplo, los próximos 2 meses).
4. Mostrar la duración de cada evento y formatear la fecha de manera clara y comprensible para los usuarios.



# Actividad: Eventos recurrentes

## Requisitos:

1. El administrador introduce la fecha y hora inicial del evento y la duración de este en días, horas y minutos.
2. Se define un intervalo de repetición, por ejemplo, 7 días (una vez por semana).
3. Se genera una lista de todas las fechas del evento durante un periodo de 2 meses.
4. Se muestra cada evento en un formato legible y se calcula la duración entre el inicio y el fin de cada uno.





## 7. Gestión de diferentes Zonas Horarias

# Gestión de zonas horarias mediante `DateTimeZone`

Al crear un objeto `DateTime`, se puede **especificar una zona horaria**. Esto se representa mediante un objeto creado utilizando la clase `DateTimeZone`.

La clase `DateTimeZone` crea un objeto que representa una zona horaria. **Almacenará información sobre esa zona horaria.**

Esto nos **ayuda a manejar las fechas y horas de manera precisa y coherente a nivel global o en contextos donde las zonas horarias influyen.**



# Gestión de zonas horarias mediante DateTimeZone

En el paréntesis, hay que indicar la zona horaria que utiliza una zona horaria IANA (para ver la lista completa, consulta <https://timeapi.io/documentation/iana-timezones>).

Puedes utilizar este objeto al crear un objeto DateTime para **especificar su zona horaria**, que también controla el horario de verano.

```
VARIABLE          CLASS NAME          TIME ZONE
┌──────────┐      ┌──────────┐      ┌──────────┐
$tz_LDN = new DateTimeZone('Europe/London');
$LDN = new DateTime('now', $tz_LDN);
                                └──────────┘
                                DateTimeZone
                                OBJECT
```

The diagram illustrates the components of the provided code snippets. It identifies three parts: 'VARIABLE' (pointing to `$tz_LDN` and `$LDN`), 'CLASS NAME' (pointing to `DateTimeZone` and `DateTime`), and 'TIME ZONE' (pointing to the string argument in the `DateTimeZone` constructor). A bracket under the `$tz_LDN` argument in the second line points to the label 'DateTimeZone OBJECT'.

# Gestión de zonas horarias mediante DateTimeZone

METHOD	DESCRIPTION										
<code>getName()</code>	Returns the name of this time zone										
<code>getLocation()</code>	Returns an indexed array holding the following information: <table><tr><th>KEY</th><th>VALUE</th></tr><tr><td><code>country_code</code></td><td>Short code for country</td></tr><tr><td><code>latitude</code></td><td>Latitude of this location</td></tr><tr><td><code>longitude</code></td><td>Longitude of this location</td></tr><tr><td><code>comments</code></td><td>Any comments about this location</td></tr></table>	KEY	VALUE	<code>country_code</code>	Short code for country	<code>latitude</code>	Latitude of this location	<code>longitude</code>	Longitude of this location	<code>comments</code>	Any comments about this location
KEY	VALUE										
<code>country_code</code>	Short code for country										
<code>latitude</code>	Latitude of this location										
<code>longitude</code>	Longitude of this location										
<code>comments</code>	Any comments about this location										
<code>getOffset()</code>	Returns the offset from UTC for this timezone in seconds (UTC is the same time as GMT but it is a standard, and is not tied to a country/territory)										
<code>getTransitions()</code>	Returns an array indicating when daylight savings time takes effect in the given time zone										

## Ejemplo: Objeto DateTimeZone

Este ejemplo obtiene y muestra las horas actuales en tres zonas horarias diferentes (Londres, Tokio y Sídney), incluyendo la diferencia horaria (en horas) respecto a UTC, y además muestra información geográfica sobre Londres, como su latitud y longitud, utilizando el objeto `DateTimeZone`.

# Ejemplo: Objeto `DateTimeZone`

1. Se crean dos objetos `DateTimeZone` para representar las zonas horarias de Londres y Tokio.
2. `getLocation()` devuelve los datos de localización de la zona horaria de Londres como un array. El array se almacena en `$location`.
3. Se crean dos objetos `DateTime` utilizando los objetos `DateTimeZone` del paso 1. Representan la hora actual en dos zonas horarias.
4. Se crea un tercer objeto `DateTime` para mostrar cómo el objeto `DateTimeZone` puede crearse al mismo tiempo que el objeto `DateTime`.



# Ejemplo: Objeto `DateTimeZone`

5. Para cada uno de los objetos `DateTime`

- `format()` muestra la hora actual en ese lugar.
- `getOffset()` muestra la diferencia horaria entre esas localizaciones y UTC.

Devuelve la diferencia horaria en segundos, por lo que se divide por  $60 * 60$  para mostrar el desfase en horas.

6. El nombre de la primera zona horaria se recupera utilizando `getName()`.

7. Se escriben la longitud y la latitud de la zona horaria.

# Ejemplo: Objeto DateTimeZone

```
<?php
① [ $tz_LDN    = new DateTimeZone('Europe/London');
    $tz_TYO    = new DateTimeZone('Asia/Tokyo');
② $location = $tz_LDN->getLocation();

③ [ $LDN      = new DateTime('now', $tz_LDN);
    $TYO      = new DateTime('now', $tz_TYO);
④ [ $SYD      = new DateTime('now',
    new DateTimeZone('Australia/Sydney'));
?> ...

⑤ [ <p><b>LDN: <?= $LDN->format('g:i a') ?></b>
    (<?= ($LDN->getOffset() / (60 * 60)) ?>)<br>
    <b>TYO: <?= $TYO->format('g:i a') ?></b>
    (<?= ($TYO->getOffset() / (60 * 60)) ?>)<br>
    <b>SYD: <?= $SYD->format('g:i a') ?></b>
    (<?= ($SYD->getOffset() / (60 * 60)) ?>)<br></p>

    <h1>Head Office</h1>
⑥ <p><?= $tz_LDN->getName() ?><br>
⑦ [ <b>Longitude:</b> <?= $location['longitude'] ?><br>
    <b>Latitude:</b>  <?= $location['latitude'] ?></p>
```

# Ejemplo: Objeto DateTimeZone





# Actividad: Zonas horarias

Eres un desarrollador web encargado de implementar un sistema de eventos global. Los eventos pueden tener participantes de diferentes partes del mundo, por lo que es necesario manejar zonas horarias. El sistema debe mostrar las fechas de los eventos en la zona horaria local de cada participante y manejar correctamente la conversión de tiempo. También necesitas ajustar las horas según las transiciones de horario de verano.



# Actividad: Zonas horarias

## Requisitos:

1. El administrador crea un evento con una fecha y hora en una zona horaria específica.
2. El sistema debe convertir esa fecha a otras zonas horarias, como la de Nueva York y Tokio.
3. El sistema debe mostrar información sobre la zona horaria local de cada ciudad, como el nombre, la ubicación geográfica, el *offset* respecto a UTC, y las transiciones de horario de verano.
4. Se debe generar una lista de eventos recurrentes y convertirlos automáticamente a las zonas horarias seleccionadas.

# Resumen

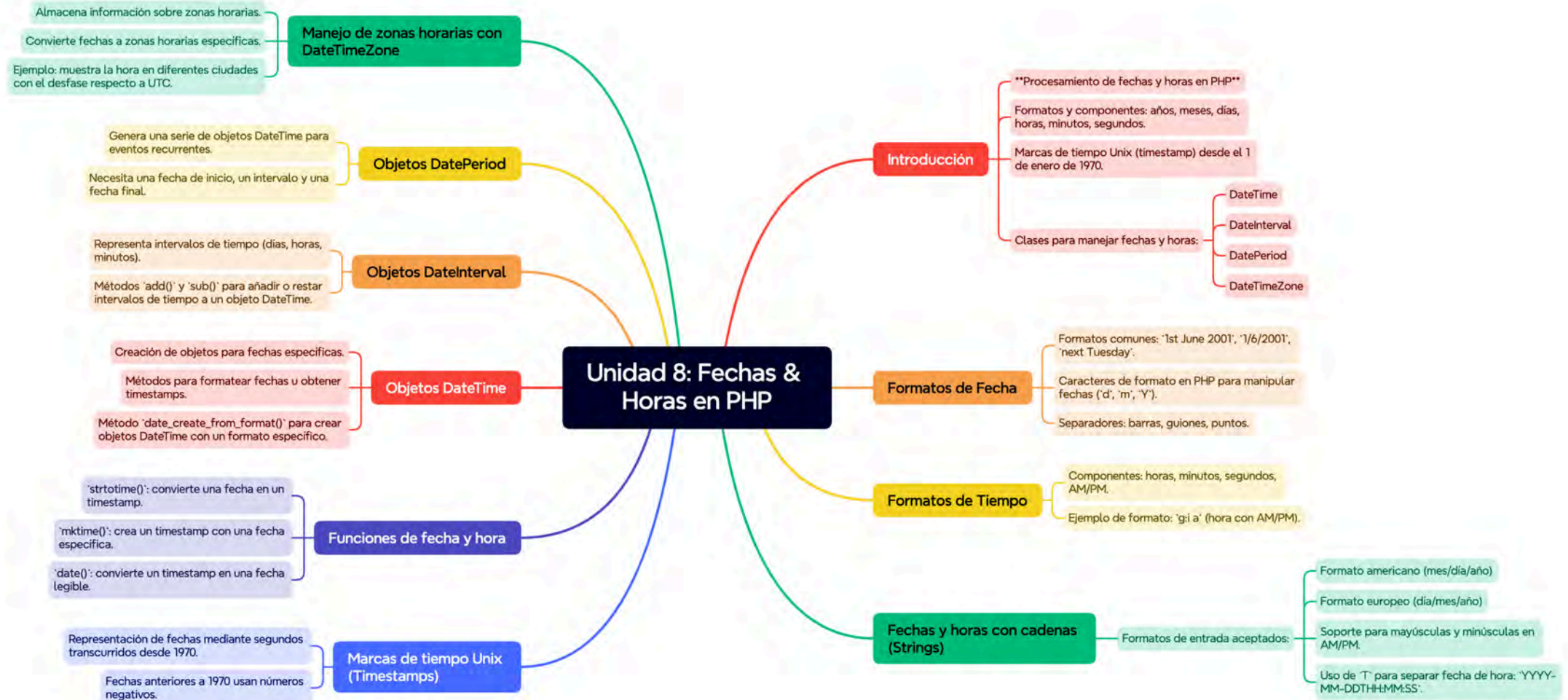
- Los caracteres de formato permiten especificar la forma en que debe formatearse una fecha o una hora.
- Una marca de tiempo Unix representa una fecha y hora utilizando los segundos transcurridos desde el 1 de enero de 1970.
- Las funciones `time()`, `strptime()` y `mktime()` crean marcas de tiempo Unix. La función `date()` convierte una marca de tiempo Unix a un formato legible por el ser humano.



# Resumen

- La clase `DateTime` crea objetos que representan fechas y horas. Dispone de métodos para modificar fechas y horas y mostrarlas en formatos legibles.
- La clase `DateInterval` crea objetos que representan un intervalo de tiempo, como un mes o un año.
- La clase `DatePeriod` crea un conjunto de objetos `DateTime` que representan eventos recurrentes.
- La clase `DateTimeZone` crea objetos que representan zonas horarias y contienen información sobre ellas.

# Resumen



Presented with mind

# Referencias

1. Una guía completa para manejar fechas y horas en PHP ([Honeybadger](#)).
2. Dominando las funciones de fecha y hora en PHP: Guía completa para desarrolladores ([Gyata](#)).
3. Trabajando con fechas y horas en PHP: Mejores prácticas ([Milind Daraniya Blog](#)).
4. Fecha y hora en PHP ([W3Schools](#)).
5. Trabajando con zonas horarias en PHP ([Codementor](#)).