PHP – IntlChar

In PHP7, a new IntlChar class has been introduced. It provides access to a number of utility methods that can be used to access information about Unicode characters. There are a number of static methods and constants in Intl class. They adhere closely to the names and behavior used by the underlying ICU (International Components for Unicode) library.

Note that you need to enable the Intl extension in the PHP installation in your system. To enable, open php.ini file and uncomment (remove the leading semicolon from the line)

```
extension=intl
```

Some static functions from Intl class are explained with examples as below −

IntlChar::charAge

This function gets the "age" of the code point

```
public static IntlChar::charAge(int|string $codepoint): ?array
```

The "age" is the Unicode version when the code point was first designated (as a non-character or for Private Use) or assigned a character.

Example

Take a look at the following example −

```
<?php
   var_dump(IntlChar::charage("\u{2603}"));
?>
```

It will produce the following **output** −

```
array(4) {
  [0]=>
  int(1)
  [1]=>
  int(1)
  [2]=>
  int(0)
  [3]=>
  int(0)
}
```

IntlChar::charFromName

The charFromName() function finds Unicode character by name and return its code point value

```
public static IntlChar::charFromName(string $name,
   int $type = IntlChar::UNICODE_CHAR_NAME): ?int
```

The type parameter sets of names to use for the lookup. Can be any of these constants −

- IntlChar::UNICODE_CHAR_NAME (default)
- IntlChar::UNICODE_10_CHAR_NAME
- IntlChar::EXTENDED_CHAR_NAME
- IntlChar::CHAR_NAME_ALIAS
- IntlChar::CHAR_NAME_CHOICE_COUNT

Example

Take a look at the following example −

```
<?php
   var_dump(IntlChar::charFromName("LATIN CAPITAL LETTER A"));
   var_dump(IntlChar::charFromName("SNOWMAN"));
?>
```

It will produce the following **output** −

```
int(65)
int(9731)
```

IntlChar::charName

The charName() function retrieves the name of a Unicode character

```
public static IntlChar::charName(int|string $codepoint,
   int $type = IntlChar::UNICODE_CHAR_NAME): ?string
```

Example

Take a look at the following example –

```
<?php
   var_dump(IntlChar::charName(".", IntlChar::UNICODE_CHAR_NAME));
   var_dump(IntlChar::charName("\u{2603}"));
?>
```

It will produce the following **output** –

```
string(9) "FULL STOP"
string(7) "SNOWMAN"
```

IntlChar::isalpha

The isalpha() function determines whether the specified code point is a letter character. true for general categories "L" (letters).

```
public static IntlChar::isalpha(int|string $codepoint): ?bool
```

Example

Take a look at the following example –

```
<?php
   var_dump(IntlChar::isalpha("A"));
   var_dump(IntlChar::isalpha("1"));
?>
```

It will produce the following **output** –

```
bool(true)
bool(false)
```

The Intl class defines similar static methods such as isdigit(), isalnum(), isblank(), etc.

IntlChar::islower

The islower() function determines whether the specified code point has the general category "Ll" (lowercase letter).

```
public static IntlChar::islower(int|string $codepoint): ?bool
```

Example

Take a look at the following example −

```
<?php
   var_dump(IntlChar::islower("A"));
   var_dump(IntlChar::islower("a"));
?>
```

It will produce the following **output** −

```
bool(false)
bool(true)
```

Similarly, there are functions such as isupper(), istitle(), iswhitespace() etc.

IntlChar::toupper

The given character is mapped to its uppercase equivalent.

```
public static IntlChar::toupper(int|string $codepoint): int|string|null
```

If the character has no uppercase equivalent, the character itself is returned.

Example

Take a look at the following example −

```
<?php
   var_dump(IntlChar::toupper("A"));
   var_dump(IntlChar::toupper("a"));
?>
```

It will produce the following **output** −

```
string(1) "A"
string(1) "A"
```