| **format character** | **Description** | **Example returned values** |
| --- | --- | --- |
| **Day** | | |
| d | Day of the month, 2 digits with leading zeros | 01 to 31 |
| D | A textual representation of a day, three letters | Mon through Sun |
| j | Day of the month without leading zeros | 1 to 31 |
| l | A full textual representation of the day of the week | Sunday through Saturday |
| N | ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0) | 1 (for Monday) through 7 (for Sunday) |
| S | English ordinal suffix for the day of the month, 2 characters | st, nd, rd or th. Works well with j |
| w | Numeric representation of the day of the week | 0 (for Sunday) through 6 (for Saturday) |
| z | The day of the year (starting from 0) | 0 through 365 |
| **Week** | | |
| W | ISO-8601 week number of year, weeks starting on Monday (added in PHP 4.1.0) | Example: 42 (the 42nd week in the year) |
| **Month** | | |
| F | A full textual representation of a month, such as January or March | January through December |
| m | Numeric representation of a month, with leading zeros | 01 through 12 |
| M | A short textual representation of a month, three letters | Jan through Dec |
| n | Numeric representation of a month, without leading zeros | 1 through 12 |
| t | Number of days in the given month | 28 through 31 |
| **Year** | | |
| L | Whether it’s a leap year | 1 if it is a leap year, 0 otherwise. |
| o | ISO-8601 year number. This has the same value as Y, except that if the ISO week number (W) belongs to the previous or next year, that year is used instead. (added in PHP 5.1.0) | Examples: 1999 or 2003 |
| Y | A full numeric representation of a year, 4 digits | Examples: 1999 or 2003 |
| y | A two digit representation of a year | Examples: 99 or 03 |
| **Time** | | |
| a | Lowercase Ante meridiem and Post meridiem | am or pm |
| A | Uppercase Ante meridiem and Post meridiem | AM or PM |
| B | Swatch Internet time | 000 through 999 |
| g | 12-hour format of an hour without leading zeros | 1 through 12 |
| G | 24-hour format of an hour without leading zeros | 0 through 23 |
| h | 12-hour format of an hour with leading zeros | 01 through 12 |
| H | 24-hour format of an hour with leading zeros | 00 through 23 |
| i | Minutes with leading zeros | 00 to 59 |
| s | Seconds, with leading zeros | 00 through 59 |
| **Timezone** | | |
| e **(unsuported)** | Timezone identifier (added in PHP 5.1.0) | Examples: UTC, GMT, Atlantic/Azores |
| I | Whether or not the date is in daylights savings time | 1 if Daylight Savings Time, 0 otherwise. |
| O | Difference to Greenwich time (GMT) in hours | Example: +0200 |
| P | Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3) | Example: +02:00 |
| T | Timezone setting of this machine | Examples: EST, MDT … |
| Z | Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive. | -43200 through 43200 |
| **Full Date/Time** | | |
| c | ISO 8601 date (added in PHP 5) | 2004-02-12T15:19:21+00:00 |
| r | [RFC 2822](http://www.faqs.org/rfcs/rfc2822) formatted date | Example: Thu, 21 Dec 2000 16:01:07 +0200 |
| U | Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT) | See also [time()](http://us2.php.net/manual/en/function.time.php) |

// Simulates PHP's date function  
Date.prototype.format = function(format) {  
    var returnStr = '';  
    var replace = Date.replaceChars;  
    for (var i = 0; i < format.length; i++) {       var curChar = format.charAt(i);         if (i - 1 >= 0 && format.charAt(i - 1) == "\\") {  
            returnStr += curChar;  
        }  
        else if (replace[curChar]) {  
            returnStr += replace[curChar].call(this);  
        } else if (curChar != "\\"){  
            returnStr += curChar;  
        }  
    }  
    return returnStr;  
};  
  
Date.replaceChars = {  
    shortMonths: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],  
    longMonths: ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'],  
    shortDays: ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'],  
    longDays: ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'],  
  
    // Day  
    d: function() { return (this.getDate() < 10 ? '0' : '') + this.getDate(); },  
    D: function() { return Date.replaceChars.shortDays[this.getDay()]; },  
    j: function() { return this.getDate(); },  
    l: function() { return Date.replaceChars.longDays[this.getDay()]; },  
    N: function() { return this.getDay() + 1; },  
    S: function() { return (this.getDate() % 10 == 1 && this.getDate() != 11 ? 'st' : (this.getDate() % 10 == 2 && this.getDate() != 12 ? 'nd' : (this.getDate() % 10 == 3 && this.getDate() != 13 ? 'rd' : 'th'))); },  
    w: function() { return this.getDay(); },  
    z: function() { var d = new Date(this.getFullYear(),0,1); return Math.ceil((this - d) / 86400000); }, // Fixed now  
    // Week  
    W: function() { var d = new Date(this.getFullYear(), 0, 1); return Math.ceil((((this - d) / 86400000) + d.getDay() + 1) / 7); }, // Fixed now  
    // Month  
    F: function() { return Date.replaceChars.longMonths[this.getMonth()]; },  
    m: function() { return (this.getMonth() < 9 ? '0' : '') + (this.getMonth() + 1); },  
    M: function() { return Date.replaceChars.shortMonths[this.getMonth()]; },  
    n: function() { return this.getMonth() + 1; },  
    t: function() { var d = new Date(); return new Date(d.getFullYear(), d.getMonth(), 0).getDate() }, // Fixed now, gets #days of date  
    // Year  
    L: function() { var year = this.getFullYear(); return (year % 400 == 0 || (year % 100 != 0 && year % 4 == 0)); },   // Fixed now  
    o: function() { var d  = new Date(this.valueOf());  d.setDate(d.getDate() - ((this.getDay() + 6) % 7) + 3); return d.getFullYear();}, //Fixed now  
    Y: function() { return this.getFullYear(); },  
    y: function() { return ('' + this.getFullYear()).substr(2); },  
    // Time  
    a: function() { return this.getHours() < 12 ? 'am' : 'pm'; },  
    A: function() { return this.getHours() < 12 ? 'AM' : 'PM'; },  
    B: function() { return Math.floor((((this.getUTCHours() + 1) % 24) + this.getUTCMinutes() / 60 + this.getUTCSeconds() / 3600) \* 1000 / 24); }, // Fixed now  
    g: function() { return this.getHours() % 12 || 12; },  
    G: function() { return this.getHours(); },  
    h: function() { return ((this.getHours() % 12 || 12) < 10 ? '0' : '') + (this.getHours() % 12 || 12); },  
    H: function() { return (this.getHours() < 10 ? '0' : '') + this.getHours(); },  
    i: function() { return (this.getMinutes() < 10 ? '0' : '') + this.getMinutes(); },  
    s: function() { return (this.getSeconds() < 10 ? '0' : '') + this.getSeconds(); },  
    u: function() { var m = this.getMilliseconds(); return (m < 10 ? '00' : (m < 100 ?  
'0' : '')) + m; },  
    // Timezone  
    e: function() { return "Not Yet Supported"; },  
    I: function() {  
        var DST = null;  
            for (var i = 0; i < 12; ++i) {  
                    var d = new Date(this.getFullYear(), i, 1);  
                    var offset = d.getTimezoneOffset();  
  
                    if (DST === null) DST = offset;  
                    else if (offset < DST) { DST = offset; break; }                     else if (offset > DST) break;  
            }  
            return (this.getTimezoneOffset() == DST) | 0;  
        },  
    O: function() { return (-this.getTimezoneOffset() < 0 ? '-' : '+') + (Math.abs(this.getTimezoneOffset() / 60) < 10 ? '0' : '') + (Math.abs(this.getTimezoneOffset() / 60)) + '00'; },  
    P: function() { return (-this.getTimezoneOffset() < 0 ? '-' : '+') + (Math.abs(this.getTimezoneOffset() / 60) < 10 ? '0' : '') + (Math.abs(this.getTimezoneOffset() / 60)) + ':00'; }, // Fixed now  
    T: function() { var m = this.getMonth(); this.setMonth(0); var result = this.toTimeString().replace(/^.+ \(?([^\)]+)\)?$/, '$1'); this.setMonth(m); return result;},  
    Z: function() { return -this.getTimezoneOffset() \* 60; },  
    // Full Date/Time  
    c: function() { return this.format("Y-m-d\\TH:i:sP"); }, // Fixed now  
    r: function() { return this.toString(); },  
    U: function() { return this.getTime() / 1000; }  
};