# Gestione delle date in Java Script

Rev. 1.0 del 10/12/2021

javascript memorizza le date nel formato **ISODate** (standard ISO 8601 del 1988) che è un **Object** che memorizza al suo interno due informazioni principali:

- il valore della data memorizzato mediante un numero intero a 64 bit che rappresenta il cosiddetto timestamp unix, cioè il numero di **millisecondi** trascorsi dalla data zero dei sistemi unix (1/1/1970)
  Su 64 bit sono rappresentabili 290 milioni di anni in positivo e in negativo.
- Un **offset** che indica la differenza tra l'ora locale e l'ora **GMT** del meridiano 0 di Greenwich (il cosiddetto Greenwich Mean Time). L'Italia adotta l'ora standard dell'Europa centrale per cui l'ora solare invernale è GMT+0**1**:00, quella legale estiva GMT+0**2**:00

Per creare una nuovo oggetto Date si utilizza il costruttore dell'oggetto:

```
var dataCorrente = new Date();
```

che restituisce la data corrente in formato **ISODate**, con dentro i msec e l'offset.
Il metodo **`Date()`** **senza il new** restituisce la data corrente come semplice stringa !

E' anche possibile creare un oggetto **ISODate** a partire da una generica data scritta come stringa in formato **UTC**:

```
var dataDiNascita = new Date("2004-02-25");
```

Gli **UTC Format** validi sono i seguenti:

```
YYYY
YYYY-MM
YYYY-MM-DD
MM-DD-YYYY
YYYY-MM-DDThh:mm±HH:MM
YYYY-MM-DDThh:mm:ss±HH:MM
YYYY-MM-DDThh:mm:ss.mmm±HH:MM
mmm di un timestamp unix
```

**I parametri passati al costruttore sono da intendersi come espressi in timeZone locale**.
Il **±** finale rappresenta l'offset dell'ora locale rispetto al meridiano di Greenwich. Se non espressamente indicato viene automaticamente letto ed utilizzato quello impostato all'interno del sistema operativo

Nel momento in cui si esegue un **new Date** (con o senza parametri), la funzione stessa richiede al sistema operativo il timeZone della macchina e provvede a salvare all'interno dell'oggetto Date:

- La data corrente espressa come timestamp unix **ricalcolata al valore GMT**
- L'offset da applicare al timestamp precedente per ottenere l'ora locale.

## Metodi per la visualizzazione di una data

Oltre al metodo base .toString() esistono diversi altri metodi di visualizzazione della data contenuta all'interno di un oggetto ISODate. Tutti questi metodi provvedono a ricalcolare l'ora locale aggiungendo / sottraendo il valore di timeZone e a visualizzare la data/ora in vari formati differenti.

### toIsoString()

Restituisce la data nel tipico **UTC format** che è un formato **fisso** lungo sempre **24 caratteri** che può essere poi utilizzato nel costruttore new Date. L'ora viene espressa come ora GMT (attestata dalla Z finale che indica il meridiano zero). Rappresenta il metodo utilizzato per default nella **serializzazione JSON** dell'oggetto ISODate e rappresenta anche il formato stringa normalmente utilizzato per salvare una data all'interno di un qualunque DB.

Non esistono forme brevi con soltanto la data oppure con l'ora espressa in formato locale.

```
console.log(dataDiNascita.toISOString())
2004-02-25T00:00:00.000Z
```

**Elenco Completo delle funzioni di visualizzazione:**

```
var dataCorrente = new Date("2020-10-13T17:37:56");
```

| | | |
|---|---|---|
| dataCorrente.**toISOString**() | 2020-10-13T15:37:56.000Z | // GMT |
| dataCorrente.**toISOString().substr(0,10)** | 2020-10-13 | // GMT |
| dataCorrente.toString() | Tue Oct 13 2020 **17:37:56** GMT+0200 | // locale |
| dataCorrente.toUTCString() ( | Tue, 13 Oct 2020 15:37:56 GMT | // GMT |
| dataCorrente.toDateString() | Tue Oct 13 2020 | // solo data GMT |
| dataCorrente.toLocaleString() | 13/10/2020, **17:37:56** | // data/ora locali |
| dataCorrente.**toLocaleDateString**() | 13/10/2020 | // data locale |
| dataCorrente.toLocaleTimeString('it-IT', { hour12: false }) , **17:37:56** | | // ora locale |

Il metodo di default `.toString()` restituisce come ora l'ora locale con l'indicazione successiva del time zone locale utilizzato per il calcolo : **`GMT+0200.`** (vedasi nota successiva)

**Nota**

Notare la differenza tra le seguenti righe:

```
var date = new Date("2020-10-13T00:00:00")
console.log(date.toString())
Tue Oct 13 2020 00:00:00 GMT+0200   // locale
console.log(date.toISOString())
2020-10-12T22:00:00.000Z            // GMT

var date = new Date("2020-10-13")
console.log(date.toString())
Tue Oct 13 2020 02:00:00 GMT+0200   // locale
console.log(date.toISOString())
2020-10-13T00:00:00.000Z            // GMT
```

Cioè, in fase di creazione dell'oggetto Date:

- Se il time viene indicato esplicitamente, viene assunto come ora locale
- Se il time viene omesso viene automaticamente assunta come ora l'ora zero di Greenwich

**Metodi che restituiscono un timestamp**

msec=dataCorrente.**getTime( )** Restituisce il **timestamp unix interno** (msec dal 1/1/70)**)**
msec=dataCorrente.**valueOf( )** equivalente al precedente
msec=**Date.now( )** Metodo statico equivalente ai precedenti. Restitusice la data corrente in msec

**Metodi per l'accesso ai singoli campi di una data**

| | |
|---|---|
| ss = dataCorrente.getSeconds ( ) | Restituisce i secondi 0-59 |
| mm = dataCorrente.getMinutes( ) | Restituisce i minuti 0-59 |
| hh = dataCorrente.getHours( ) | Restituisce l'ora da 0-23 riferita al timeZone locale |
| hh = dataCorrente.getUTCHours( ) | Restituisce l'ora da 0-23 calcolata in UTC (1 o 2 ore in meno del precedente) |
| gg = dataCorrente.**getDate**( ) | Restituisce il giorno come numero intero da **1-31** riferita al timeZone locale |
| gg = dataCorrente.getUTCDate( ) | Restituisce il giorno come numero intero da **1-31**espresso in UTC |
| gg = dataCorrente.getDay ( ) | Giorno della settimana 0-6 (Domenica = 0, Lunedì = 1, …… Sabato = 6) |
| gg = dataCorrente.getUTCDay ( ) | Giorno della settimana 0-6 (Domenica = 0, Lunedì = 1, …… Sabato = 6) UTC |

me = dataCorrente.getMonth( ) Mese dell'anno 0-11 (attenzione gennaio = 0)
yy = dataCorrente.getFullYear( ) Anno a 4 cifre

offset= dataCorrente.**getTimezoneOffset**() Restituisce l'offset (in minuti) del meridiano corrente (+60 per l'Italia)

### Metodi che consentono di modificare una data

I seguenti metodi non restituiscono alcunché, ma provvedono semplicemente a modificare il contenuto di date:
dataCorrente.setTime(val) Si aspetta come parametro i msec complessivi e reimposta l'intera data
dataCorrente.setSeconds (val) Modifica soltanto i secondi
dataCorrente.setMinutes(val) Modifica soltanto i Minuti
dataCorrente.setHours(val) Modifica soltanto l'ora (interpretata come locale)
dataCorrente.setUTCHours(val)Modifica soltanto l'ora (interpretata come UTC)
dataCorrente.setDay(val) Modifica soltanto il giorno della settimana 0-6
dataCorrente.setUTCDay(val) Modifica soltanto il giorno della settimana 0-6 (UTC)
dataCorrente.setMonth(val) mese dell'anno 0-11
dataCorrente.setUTCMonth(val) mese dell'anno 0-11
dataCorrente.setYear(val) Anno a 4 cifre

dataCorrente.**setDate(**val) Modifica soltanto il giorno del mese 1-31. Se come parametro viene passato un numero non previsto nel mese (cioè un numero minore di 1 oppure un numero maggiore di 30 o 31, automaticamente viene aggiornato di conseguenza il numero del **mese** ed eventualmente il numero dell'**anno**.

dataCorrente.setUTCDate(val) Modifica soltanto il giorno del mese 1-31 calcolato come giorno UTC

### Somma, sottrazione e confronto fra date

Poiché il timestamp è un semplice numero intero, **è possibile sommare e sottrarre i timestamp.**

- Il **confronto** può sempre essere eseguito in forma diretta: `if(data1> data2) …`
- Nel caso della **sottrazione** può essere eseguita direttamente sull'oggetto ISODate, **senza** l'utilizzo di .getTime().
- Nel caso della **somma** occorre sempre passare attraverso .getTime()

**1) Data che si avrà fra 7 giorni.**

```
var dataCorrente = new Date()
var nextWeek = dataCorrente.getTime()    +    7 * 24 * 3600 * 1000
alert(new Date(nextWeek))
```

**2) Differenza fra due date**

La differenza fra due oggetti ISODate è un semplice numero indicante i msec di differenza fra le due date.

```
var diff = new Date(_txtN2.value) - new Date(_txtN1.value)
var nGiorni = Math.floor(diff/(24*3600*1000))
```

Se la differenza da valutare è inferiore al mese si può utilizzare la seguente soluzione.

```
var dateDiff = new Date(diff);
var giorni = dateDiff.getUTCDate();  var hours = dateDiff.getUTCHours()
var minutes = dateDiff.getUTCMinutes() // senza UTC verrebbe aggiunto il timeZone
```

**3) Età di una persona**

```
var age=Math.floor((new Date() - dob)/(365*24*3600*1000))
```

4) Come impostare una data 'locale' partendo da una data UTC

```
var offset = dataCorrente.getTimezoneOffset() / 60;
var hours = dataCorrente.getHours();
dataCorrente.setHours(hours - offset);
```

### Nota

Se si imposta una certa data tramite new Date() e poi si vanno a leggere i msec contenuti all'interno della data e li si visualizza all'interno di un time converter, si legge in realtà una data GMT cioè, in inverno, si vede un'ora in meno rispetto all'ora italiana utilizzata nel costruttore (due ore in estate).

Questo perché new Date() salva i millisecondi come GMT, e poi aggiunge l'offset di 1 ora all'interno dell'apposito campo relativo alle informazioni locali

## Cenni sulla libreria moment.js

```
src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js"
var moment = require('moment');  // nodejs
```

La libreria moment, mediante i metodi .**add**() e .**subtract()** semplifica notevolmente le elaborazioni sulle date. L'oggetto **`moment`** è un oggetto articolato, simile all'oggetto ISODate di javascript, contenente molte informazioni sulla data memorizzata.
Il suo funzionamento è comunque simile a quello dell'oggetto ISODate

### Il costruttore moment()

Funzionamento analogo al **`new Date()`** di javascript. Nel caso di moment() **non** si utilizza il **new** che viene eseguito all'interno della libreria

| | |
|---|---|
| **`let date = moment()`** | restituisce un oggetto moment contenente la data corrente |
| **`let date=moment("2004-02-25")`** | restituisce un oggetto moment contenente la data 25-02-2004 |

### La conversione in stringa

Gli oggetti moment dispongono del metodo **format()** che restituisce una stringa in formato UTC esattamente come il metodo **.toISOString()** di javascript

```
console.log(date.format())
```

### I metod add() e subtract()

I metodi .add() e .subtract() applicati ad un oggetto moment consentono una rapida elaborazione

```
var date = moment().subtract(1, 'year')
var date = moment().subtract(1, 'day')
var date = moment().add(1, 'day')
```

| Key | Shorthand | |
|---|---|---|
| years | y | |
| quarters | Q | |
| months | M | // partendo da 0 ! |
| weeks | w | |
| days | d | |
| hours | h | |
| minutes | m | |
| seconds | s | |
| milliseconds | ms | |

- Le chiavi precedenti possono essere scritte indifferentemente in forma singolare oppure in forma plurale.

- La Shorthand è sempre la stessa.
- Tutte le chiavi precedenti (sia in forma singolare che in forma plurale) sono disponibili anche sotto forma di metodo per leggere / assegnare un singolo campo. Esempi:

```
var date = moment().seconds(0)   // azzero i secondi del moment corrente
let annoCorrente = date.year()
moment().add(7,'days').subtract(1,'months').year(2009).minutes(0).seconds(0)
moment().add(7, 'days').add(1, 'months');
moment().add( {days:7, months:1} )
```

### Il metodo diff()

Restitusice la differenza in msec fra due "moment date" (esattamente come la differenza diretta fra le ISODate). Però esiste la possibilità di specificare un secondo parametro che indica di esporre la differenza non in msec, ma in secondi, minuti, ore, giorni, etc

```
let ggPermanenza=dataFine.diff(dataInizio, "days")
```

### Il confronto

Le date di moment, così come le ISODate, possono sempre essere confrontate in modo diretto:

```
if(data1 > data2) …
```

### il metodo toDate() e la conversione in ISODate

Dato un oggetto moment, è possibile convertirlo in ISODate usando il metodo .toDate()

```
var date = moment().subtract(1, 'year')
var jsdate = date.toDate()
```

### I metodi timeTo() e timeFrom()

Consentono di calcolare in modo molto fine la differenza fra due date.
Si rimanda alla sezione DOCS del sito ufficiale