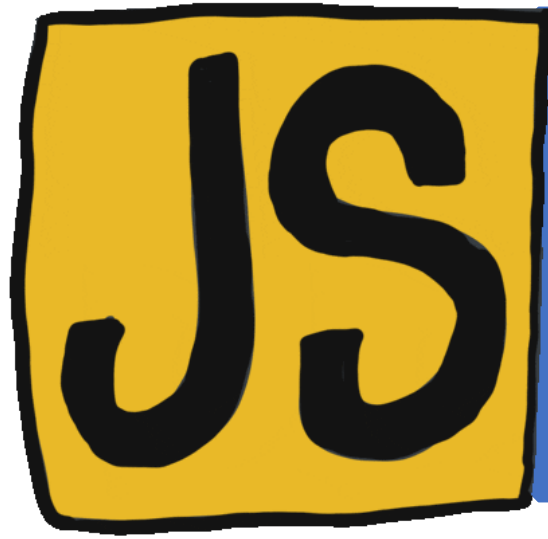


Desarrollo Web en Entorno Cliente

Tema 11





Dates



[date]

Date proporciona un objeto útil para trabajar con fechas y horas.

Su sintaxis es la siguiente:

```
new Date()  
new Date(milisegundos)  
new Date(cadenaFecha)  
new Date(año_num,mes_num,dia_num[,hor_num,min_num,seg_num,mils_num])
```

Si no se proporciona argumentos se crea un objeto *Date* con la hora y fecha actual extraída de la hora local del sistema.

Los argumentos vacíos se establecen a 0 (o a 1 si hace referencia al día).

La fecha se mide en milisegundos desde la media noche exacta del 01 de enero de 1970 en formato UTC. Un día contiene 86.400.000 milisegundos. El rango del objeto *Date* va desde -100,000,000 días hasta 100,000,000 días respecto del 01 de enero de 1970 UTC.

[date]

```
new Date()  
new Date(milisegundos)  
new Date(cadenaFecha)  
new Date(año_num, mes_num, día_num[, hor_num, min_num, seg_num, mils_num])
```

milisegundos es un valor entero que representa el número de milisegundos desde las 00:00:00 UTC del 1 de enero de 1970. Se conoce como **timestamp**: número entero que representa la cantidad de milisegundos transcurridos desde el inicio de 1970 y una fecha y hora concreta.

cadenaFecha es un valor de tipo cadena que representa una fecha. La cadena debe estar en un formato reconocido por el método *Date.parse()*.

año_num, *mes_num*, *día_num* son valores enteros que representan las partes de una fecha, como son el año, el mes y el día. Pero hay que tener presente, en la parte de los meses, que el 0 representa a enero y 11 a diciembre.

hor_num, *min_num*, *seg_num*, *mils_num* son valores enteros que representan las partes de una hora.

[date]

Ejemplo:

```
const fechaActual = new Date();  
console.log(fechaActual);
```

```
Mon Aug 07 2023 09:08:05 GMT+0200 (hora de verano de Europa central)
```

Los *timestamp* se usan con el objeto *Date* porque son una forma sencilla de representar numéricamente una fecha.

Las fechas anteriores al 1 de enero de 1970 tienen *timestamps* negativos::

```
// 31 de diciembre de 1969  
let Dic31_1969 = new Date(-24 * 3600 * 1000);
```

[date]

El objeto *Date* dispone de métodos para obtener solamente el año, el mes, el día, la hora o los minutos, entre otros, de la fecha que se le pase como argumento o en caso de no pasarle argumento alguno de la hora y fecha actual extraída del sistema.

Ejemplo:

```
const fechaActual = new Date();  
console.log(fechaActual.getFullYear());
```

2023

A continuación, se enumeran, de forma resumida, algunos métodos útiles del objeto *Date*, si no se indica dichos métodos no reciben parámetros:

Date.now() devuelve el valor numérico correspondiente a la hora actual.

getFullYear() devuelve el valor numérico correspondiente al año.

getMonth() devuelve el valor numérico correspondiente al mes (desde el 0 al 11).

[date]

getDate() devuelve el valor numérico correspondiente al día del mes (desde 1 a 31).

getDay() devuelve el valor numérico correspondiente al día de la semana (desde el 0, domingo, al 6, sábado).

getHours() devuelve el valor numérico correspondiente a la hora (desde el 0 al 23).

getMinutes() devuelve el valor numérico correspondiente a los minutos (desde el 0 al 59).

getSeconds() devuelve el valor numérico correspondiente a los segundos (desde 0 a 59)

getTime() devuelve el *timestamp* (cantidad de milisegundos transcurridos a desde el 1 de enero de 1970 UTC+0) correspondiente a un objeto *Date*.

getTimezoneOffset() devuelve la diferencia entre UTC y el huso horario de la zona actual, en minutos.

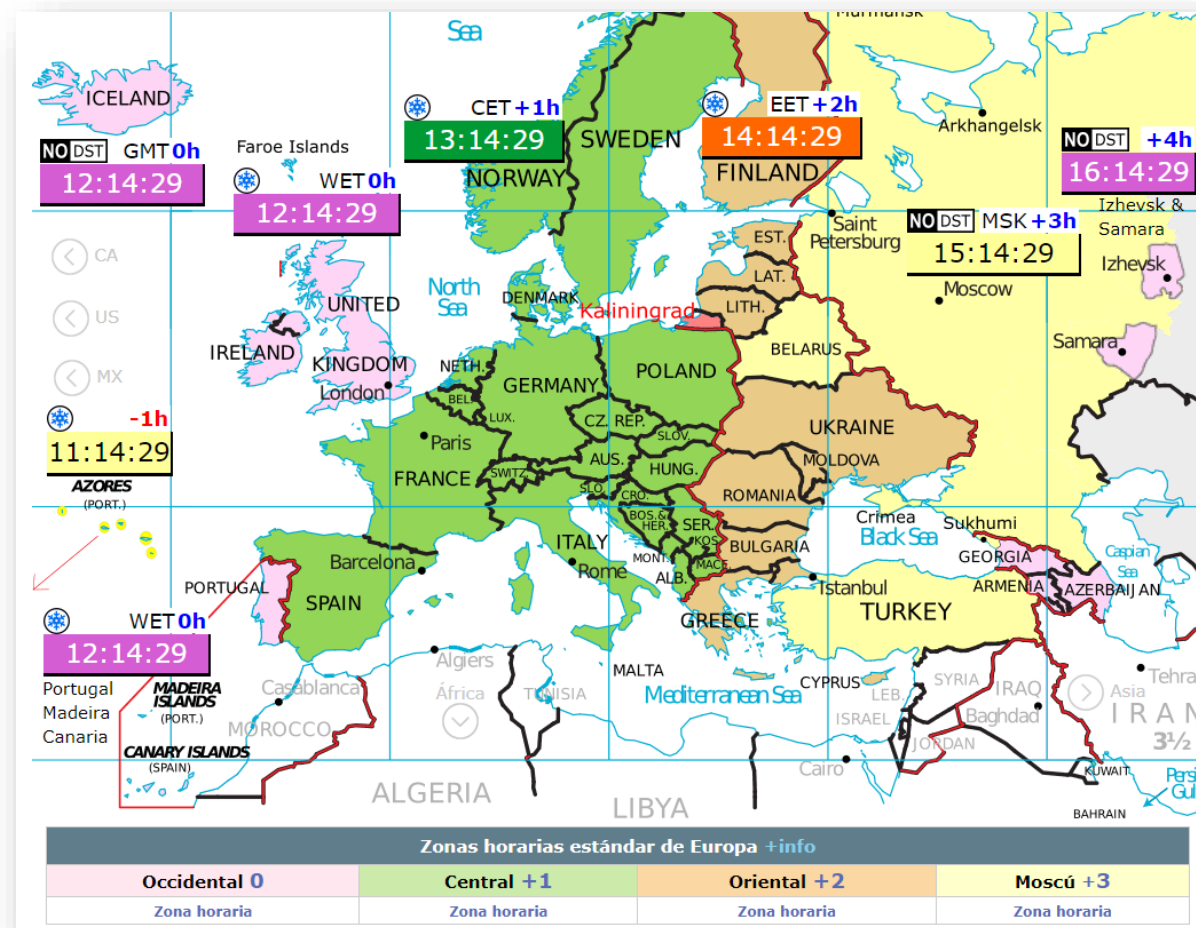
[date]

Los métodos mencionados anteriormente devuelven los componentes correspondientes a la zona horaria local.

También existen métodos para obtenerlo en formato UTC, devuelven el día, mes, año,... para la zona horaria UTC+0, estos métodos se llaman igual que los correspondientes a la zona horaria local, pero llevan las letras *UTC* justo después de *get*:

`getUTCFullYear()`, `getUTCMonth()`,
`getUTCDay()`, `getUTCSeconds()`...

El tiempo universal coordinado o UTC es el principal estándar de tiempo que regula los relojes y el tiempo en el mundo.



[date]

Ejemplos:

```
// fecha actual  
let fecha = new Date();  
  
console.log("la hora en la zona horaria actual: " + fecha.getHours());  
console.log("la hora respecto de la zona horaria UTC+0: " + fecha.getUTCHours());
```

```
la hora en la zona horaria actual: 13
```

```
la hora respecto de la zona horaria UTC+0: 12
```

```
// Si se está en la zona horaria UTC-1, devuelve 60  
// Si se está en la zona horaria UTC+3, devuelve -180  
console.log(new Date().getTimezoneOffset());
```

```
-60
```

[date]

Además, el objeto *Date* dispone de métodos que permiten establecer los componentes de la fecha y hora que almacena:

setFullYear(año, [mes], [día])

setMonth(mes, [día])

setDate(día)

setHours(hora, [min], [seg], [ms])

setMinutes(min, [seg], [ms])

setSeconds(seg, [ms])

setMilliseconds(ms)

setTime(ms) Establece la fecha en función de la cantidad de segundos transcurridos desde 1-1-1970 GMT+0)

[date]

A excepción de `setTime()`, todos los demás métodos poseen una variante UTC, por ejemplo: `setUTCHours()`.

Algunos métodos permiten fijar varios componentes al mismo tiempo, como, por ejemplo:

`setHours(hora, [min], [seg], [ms])`

Los componentes que no son mencionados no se modifican.

Ejemplo:

```
let fecha=new Date();
fecha.setHours(0, 0, 0, 0);
fecha.setMonth(11,24);
console.log("La Navidad se ha adelantado:"+ fecha);
```

```
La Navidad se ha adelantado:Sun Dec 24 2023
00:00:00 GMT+0100 (hora estándar de Europa central)
```

[date]

La autocorrección es una característica muy útil de los objetos *Date*.

Si se establecen valores fuera de rango, como 32 de agosto.... se ajustarán automáticamente.

Normalmente se usa esta característica para obtener la fecha, a partir de un período de tiempo específico.

Ejemplo, se quiere obtener la fecha de hoy pero dentro de 35 días a partir de este preciso instante.

```
//¿32 de agosto 2023?  
let fecha = new Date(2023, 7, 32);  
//Se corrigió a 1 de septiembre de 2023  
console.log(fecha);
```

```
Fri Sep 01 2023 00:00:00 GMT+0200 (hora de verano  
de Europa central)
```

```
let fecha=new Date();  
fecha.setDate(fecha.getDate() + 35);  
  
console.log(fecha);
```

[date]

Cuando se convierte un objeto *Date* a número toma el valor del *timestamp* actual, al igual que el método *date.getTime()*.

Ejemplo, de script que muestra cuantos días quedan para Navidad:

```
const fechaInicio = new Date().getTime();  
const fechaFin    = new Date('2023-12-24').getTime();  
  
let resultado = fechaFin - fechaInicio;  
resultado = Math.ceil(resultado / (1000*60*60*24));  
  
console.log(`Quedan ${resultado} días para navidad`);
```

(date)

Además, se dispone de los siguientes métodos para convertir un objeto *Date* a una cadena:

toDateString() devuelve una cadena correspondiente a la fecha de un objeto *Date*.

toTimeString() devuelve una cadena correspondiente a la hora de un objeto *Date*.

toString() devuelve una cadena de texto que representa la fecha y hora almacenada en un objeto *Date*.

toISOString() devuelve una cadena de texto en formato ISO que representa la fecha y hora almacenada en un objeto *Date*.

toJSON() devuelve una cadena de texto en formato JSON que representa la fecha y hora almacenada en un objeto *Date*.

toLocaleDateString() devuelve una cadena, en el formato que use el país local, correspondiente a la fecha de un objeto *Date*.

toLocaleTimeString() devuelve una cadena, en el formato que use el país local, correspondiente a la hora de un objeto *Date*.

[date]

Por último, el método *Date.parse()* devuelve el número de milisegundos desde las 00:00:00 del 1 de enero de 1970, hora local, que han pasado desde la fecha que se le pasa como argumento en formato cadena de caracteres (*string*).

Así permite leer un objeto tipo fecha desde una cadena de caracteres (*string*) que se le pasa como argumento.

El formato de la cadena debe ser:

YYYY-MM-DDTHH:mm:ss.sssZ

Donde:

YYYY-MM-DD es la fecha: año-mes-día.

El carácter *T* se usa como delimitador.

HH:mm:ss.sss es la hora: horas, minutos, segundos y milisegundos.

El carácter *Z* es opcional y especifica la zona horaria, con el formato *+ -hh:mm*. Si se incluye únicamente la letra *Z* equivale a UTC+0.

También es posible pasar como *string* variantes abreviadas, tales como *YYYY-MM-DD* o *YYYY-MM* o incluso *YYYY*.

[date]

El método `Date.parse(str)` convierte el *string* en el formato especificado y devuelve un *timestamp* (cantidad de milisegundos transcurridos desde el 1 de enero de 1970 UTC+0).

Si el formato del *string* no es válido devuelve un *NaN*.

Ejemplos:

```
let fecha= new Date(Date.parse("2023-12-03T14:51:50.417+01:00"));  
console.log (fecha);
```

```
Sun Dec 03 2023 14:51:50 GMT+0100 (hora  
estándar de Europa central)
```

```
Date.parse("Hola")
```

```
NaN
```


[momentjs]

Cuando se trabaja con fechas se obtiene un formato o palabras en inglés:

```
const fecha= new Date();  
console.log (fecha);
```

```
Fri Dec 03 2021 14:36:17 GMT+0100  
Europa central)
```

Y a pesar de que el objeto *Date*, como ya se ha visto, dispone de métodos como *toLocaleDateString()*, *toLocaleTimeString()* o *toLocaleString()* que muestran la fecha en el formato que se usa en el país desde donde se ejecuta no es muy versátil:

```
const fecha= new Date();  
console.log (fecha.toLocaleString());
```

```
3/12/2021 14:36:17
```

momentJS es una librería especialmente creada para trabajar con fechas: <https://momentjs.com>

[momentjs]

momentJs permite mostrar las fechas y las horas en un formato más personalizado y haciendo uso de la fecha y hora en el formato del país para el que se configura.

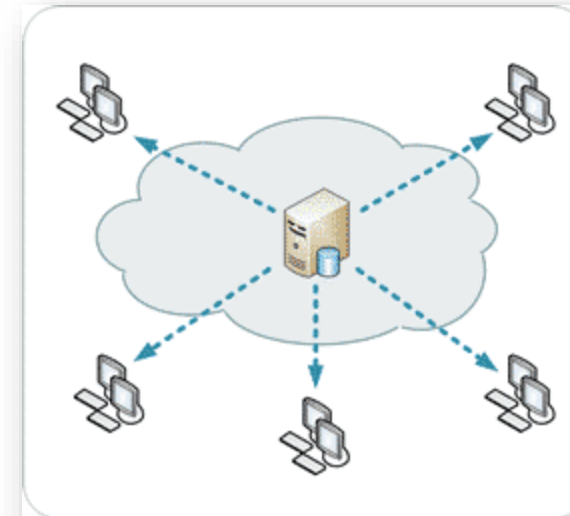
En la página web de *momentJS* puede consultarse todos los métodos que están disponibles.

```
moment().format('MMMM Do YYYY, h:mm:ss a'); // diciembre 3º 2021, 2:46:42 pm
moment().format('dddd');                    // viernes
moment().format("MMM Do YY");               // dic. 3º 21
moment().format('YYYY [escaped] YYYY');     // 2021 escaped 2021
moment().format();                          // 2021-12-03T14:46:42+01:00
```

Para hacer uso de esta librería hay que descargarla y añadirla como cualquier otro script de JS, con la excepción de que las librerías se suelen ubicar en la sección *header* de la página web, o usar un CDN.

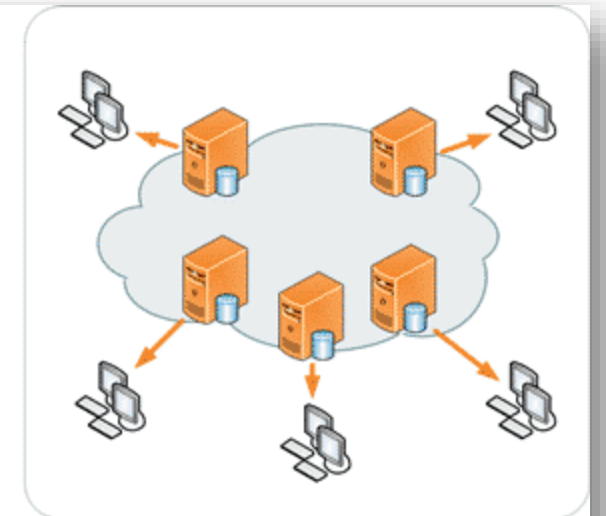
Un CDN (*Content Delivery Network*) o Red de Distribución de Contenido es un servicio que permite acceder a contenidos estáticos (como imágenes, PDF, vídeos, CSS, JS, etc.) desde servidores que están más cercanos geográficamente al visitante que entra a la página web.

La principal finalidad del CDN es que las personas que visitan un sitio web reciban la información desde el servidor más cercano y de esta forma accedan al contenido más rápidamente.



Sin CDN

Los archivos del sitio web se sirven desde un único servidor (el hosting)



Con CDN

Los archivos del sitio web se sirven desde varios servidores repartidos por todo el mundo.



Muchas de las librerías de JS, como *jQuery*, *Bootstrap*... están disponibles mediante CDN para que los usuarios puedan hacer uso de ellas más eficientemente.

Para hacer uso de CDN en las páginas web hay que indicar donde se encuentra el archivo, pero a diferencia del método tradicional, no será una ruta local sino una URL.

Los programadores de JS normalmente usan *cdnjs* para acceder a las librerías de JS.


cdnjs es una red de distribución de contenido (CDN) de software libre y de código abierto alojada por *Cloudflare* (una empresa estadounidense que proporciona una red de entrega de contenido, servicios de seguridad de Internet y servicios de servidores de nombres de dominio distribuidos,...)

cdnjs ofrece más de 4.095 bibliotecas de JavaScript y CSS, que se almacenan de manera pública en GitHub.

El 12,4% de los sitios web en Internet usan *cdnjs*, lo que lo convierte en el segundo CDN más popular para JavaScript.

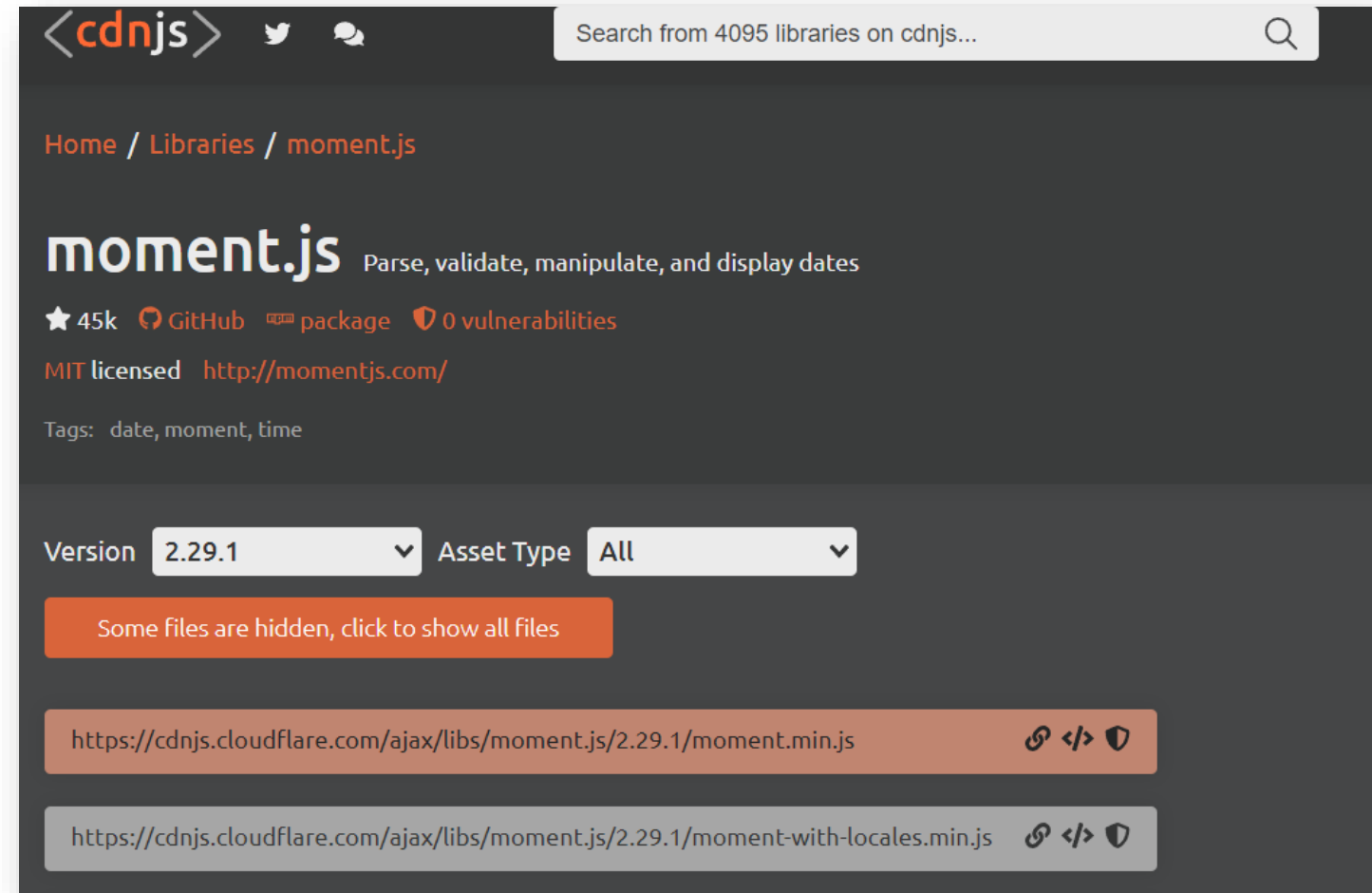
Para hacer uso de *momentjs* mediante CDN hay que visitar la siguiente página y escoger la librería que se desea utilizar.

<https://cdnjs.com/libraries/moment.js>

A continuación, se hace clic en el icono `</>` para copiar la etiqueta entera (URL,hash...) o en el icono de la cadena  para coger únicamente el enlace y se pega dentro de la sección *header* de la página web.

Hay que tener en cuenta que si se va a usar por algún script de JS la librería debe cargarse o ejecutarse en primer lugar.

El procedimiento es igual para otro tipo de librerías disponibles en cdnjs.



[momentjs]

Ejemplo:

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title></title>
  <link href="css/styles.css" rel="stylesheet">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment.min.js" integrity="sha512-qTXRIMy"
  <script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.29.1/moment-with-locales.min.js"></script>
</head>
```

[momentjs]

Una vez que se carga la librería *momentjs* en *header*, ya se tiene acceso desde JS al objeto *momentjs* que proporciona muchos métodos de utilidad.

```
console.log (moment());
```

```
moment()
```

```
▼ k {_isAMomentObject: true, _isUTC: false, _pf: {...}, _locale: S, _d: Fri  
  Dec 03 2021 17:44:34 GMT+0100 (hora estándar de Europa central), ...} i  
  ► _d: Fri Dec 03 2021 17:44:34 GMT+0100 (hora estándar de Europa centr...  
    _isAMomentObject: true  
    _isUTC: false  
    _isValid: true  
  ► _locale: S {_calendar: {...}, _longDateFormat: {...}, _invalidDate: 'Inv...  
  ► _pf: {empty: false, unusedTokens: Array(0), unusedInput: Array(0), o...  
  ► [[Prototype]]: Object
```

[momentjs]

Para configurarlo para que esté en español y para España se utiliza el método `moment.locale("es")`.

A continuación, ya se puede hacer uso de los diferentes métodos, siempre poniendo primero `moment()`.

Ejemplo:

```
moment.locale("es");  
console.log(`Estamos en ${moment().format("MMMM")}`);
```

Estamos en diciembre

```
moment.locale("es");  
console.log(`El 24 de diciembre es el día  
${moment("2021-12-24").dayOfYear()} del año`);
```

El 24 de diciembre es el día 358 del año

[momentjs]

momentjs dispone de muchos métodos, los hay que permiten añadir o sumar fechas y horas,... se recomienda hacer una lectura de la documentación y guía oficial disponibles en:

<https://momentjs.com/docs/> y <https://momentjs.com/guides/>

Ejemplos:

```
moment.Locale("es");  
console.Log(`Dentro de tres días será  
${moment().add(3, 'days').calendar()}`);  
  
console.Log(`y el 25 de diciembre será  
${moment("2021-12-25").format("dddd")}`);
```

```
Dentro de tres días será lunes a las 18:24  
El 25 de diciembre será sábado
```

Bibliografía y recursos online

- https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Date
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/Date
- <https://www.xatakamovil.com/conectividad/cdn-que-es-para-que-sirve-y-por-que-no-rompe-con-la-neutralidad-de-la-red>
- <https://momentjs.com/>
- <https://programacion.net/articulo/manejar-fechas-y-horas-como-un-profesional-con-moment.js-1301>