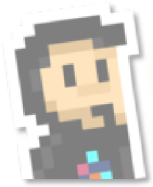


MANZ.DEV
LenguajeJS.com

Array, módulos, Regexp...

Formatear fechas: DateTimeFormat

Internacionalización y conversión de fechas

Categoría

Fechas y horas

Javascript Fechas y horas

Intl.DateTimeFormat

Compatibilidad segura (2016)

Una de las partes más complejas de trabajar con fechas es **formatear una fecha** para mostrarla de forma adecuada (`o de la forma que nos interesa`). Una fecha tiene múltiples representaciones posibles: numéricas, alfabéticas,

abreviaciones, con diferentes idiomas, estilo, orden, combinaciones entre sí, etc...

- Numéricas: `25/1/2021`, `25/01/2021`, `1/25/2021`, `01/25/2021`, `2021/01/25`, `25-1-2021`...
- Idiomáticas: `25 de Enero de 2021`, `25 de Ene. de 2021`, `25 January, 2021`, `25 Jan 2021`...
- Orden/combinaciones: `Jan/25/2021`, `25/Febrero/2021`, `25-Febrero-2021`...

Es muy habitual, que queramos formatear esa fecha con la representación de la región a la que pertenecemos (`o a la que pertenece el usuario`), por lo que la cosa se puede complicar bastante.

Por suerte, el objeto **Intl** posee `DateTimeFormat`, un sistema que unido a lo que ya sabemos sobre fechas del objeto Date nativo, pueden hacernos la vida más fácil para dar formato a fechas:

Objeto	Descripción
`Intl.DateTimeFormat`	Crea un objeto de formato con las preferencias de tu región (o la región indicada).

Veamos que podemos hacer con ella.

Métodos de formato

Dicho objeto, tiene una serie de métodos interesantes, como por ejemplo `.format()`, en el cuál nos centraremos para saber como formatear una fecha.

Los métodos de los que dispone son los siguientes:

Método	Descripción
S `.format(date)`	Formatea la fecha `date` con la configuración de región iniciada.
A `.formatToParts(date)`	Idem, dividiendo sus partes en un array de objetos.
S `.formatRange(a, b)`	Crea un rango con las fechas `a-b` usando la configuración de región.
A `.formatRangeToParts(a, b)`	Idem, pero divide sus partes en un array de objetos.
O `.resolvedOptions()`	Devuelve las opciones de región definidas en la instancia.

> Observa que tanto `.format()` como `.formatRange()` tienen una versión `*ToParts()` que hace exactamente lo mismo, sólo que en lugar de devolver un STRING , devuelven un ARRAY de OBJECT con cada parte diferenciada.

Crear una fecha

Cuando trabajamos con el objeto `Date` nativo de Javascript, tenemos opciones (`aunque limitadas`) para personalizar las fechas con las que trabajamos. Métodos como `.toString()`, `.toDateString()`, `.toGMTString()` o `.toISOString()` se pueden utilizar para personalizar el modo de representación de una fecha. Incluso podemos utilizar `.toLocaleDateString()` para formatearla dependiendo de la configuración regional del sistema del usuario:

JS

```
// 30 de Enero de 2021
const date = new Date(2021, 0, 30);

date.toString();           // "Sat Jan 30 2021 00:00:00 GMT
date.toDateString();       // "Sat Jan 30 2021"
date.toGMTString();        // "Sat, 30 Jan 2021 00:00:00 GM
date.toISOString();        // "2021-01-30T00:00:00.000Z"
date.toLocaleDateString(); // "30/1/2021"
```

En caso de querer un **tipo de representación** diferente a las anteriores, tendríamos que optar por usar librerías externas o por crear una función personalizada que devuelva el tipo de representación buscada utilizando getters nativos, lo que puede llegar a ser una tarea tediosa.

Formatear fecha Personalizar Estilo de fecha Opciones

En su lugar, podemos utilizar el objeto `Intl`, creando una nueva instancia de `DateTimeFormat()`. Se trata de un objeto que nos permitirá formatear fechas, indicando la configuración regional a seguir, e indepedientemente de la que tenga el usuario en su sistema. Observa el siguiente ejemplo, donde se muestra la fecha del ejemplo anterior, formateada en localización de **España**

(`es`), **Estados Unidos** (`en-US`), **Alemania** (`de`), **Azerbaiyán** (`az`) o **Mauritania** (`mr`):

JS

```
const esDate = new Intl.DateTimeFormat("es").format(date);
"30/1/2021"

const enDate = new Intl.DateTimeFormat("en-US").format(date
"1/30/2021"

const deDate = new Intl.DateTimeFormat("de").format(date);
"30.1.2021"

const azDate = new Intl.DateTimeFormat("az").format(date);
"2021-1-30"

const mrDate = new Intl.DateTimeFormat("mr").format(date);
"३०/१/२०२१"
```

Si al instanciar `new Intl.DateTimeFormat()` no indicamos ningún parámetro, se indicará por defecto el código del país del sistema, por lo que si tenemos un navegador con el sistema en Español, sería como si se hiciera un `new Intl.DateTimeFormat("es-ES")`.

SUBIR A TABS

Formato personalizado

A diferencia de los estilos predefinidos que podemos seleccionar rápidamente con las opciones `dateStyle` y `timeStyle`, tenemos un segundo modo de personalización. Antes de nada, tener en cuenta que debemos seleccionar uno de los dos, es decir, si utilizamos `dateStyle` o `timeStyle`, no podremos utilizar ninguna de las siguientes opciones.

Las opciones son las siguientes:

Opción	Descripción
`weekday`	Día de la semana: `long`, `short` o `narrow`.
`era`	Era actual: `long`, `short` o `narrow`.
`year`	Año: `numeric` o `2-digit`.
`month`	Mes: `numeric`, `2-digit`, `long`, `short` o `narrow`.
`day`	Día: `numeric` o `2-digit`.
`dayPeriod`	Periodo del día: `narrow`, `short` o `long`. Solo en **inglés** de momento.
`hour`	Hora: `numeric` o `2-digit`.

Opción	Descripción
B hour12	Activa el formato de 12 horas (01:00 p.m.) o lo desactiva (13:00).
S hourCycle	Formato de 12 horas (h11 o h12) o de 24 horas (h23 o h24).
S minute	Minutos: numeric o 2-digit.
S second	Segundos: numeric o 2-digit.
S fractionalSecondDigits	Dígitos de las fracciones de segundos: 1, 2 o 3.
S timeZoneName	Nombre de la zona horaria: long o short.

El funcionamiento de estas opciones es muy sencillo, vamos a echar un vistazo.

Personalizar formato Formatear partes Formatear rangos

Simplemente se trata de añadir la característica que queremos mostrar en la representación de la hora, con el valor que más nos interese. Si no queremos

mostrar alguno, simplemente lo omitimos:

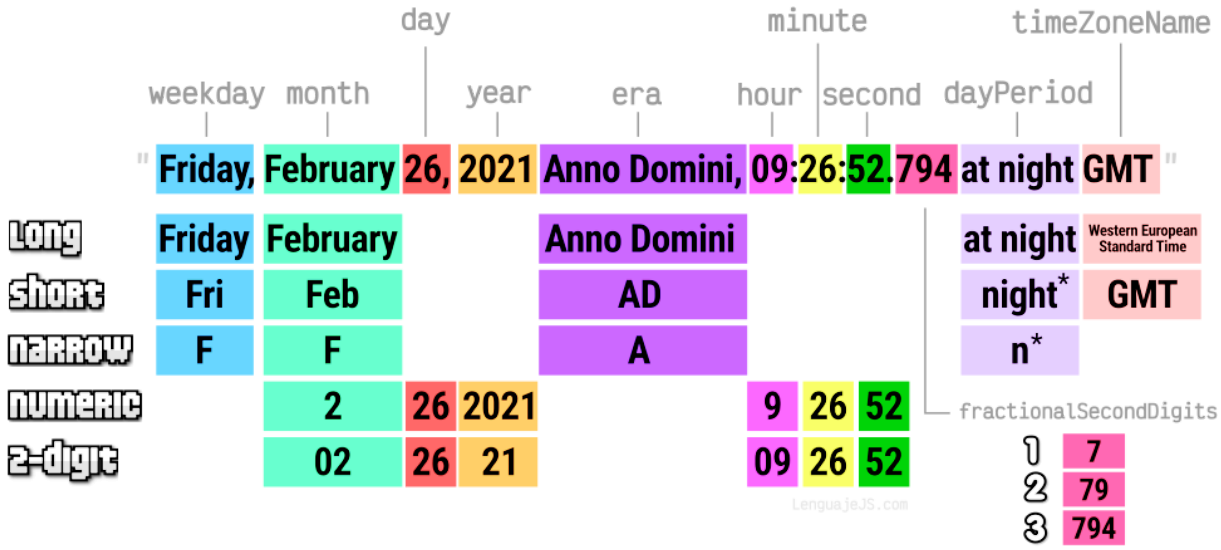
JS

```
// Localización: España
// Día de la semana, día numérico e inicial del mes
new Intl.DateTimeFormat("es", {
  weekday: "long",
  day: "2-digit",
  month: "narrow"
}).format(new Date());
"viernes, 26 F"

// Localización : Inglesa
// Día en 2 dígitos, periodo del día, mes, hora y minutos
new Intl.DateTimeFormat("en", {
  day: "2-digit",
  dayPeriod: "long",
  month: "short",
  hour: "2-digit",
  minute: "2-digit"
}).format(new Date());
"Feb 26, 03:07 at night"
```

Es importante destacar que el **orden de los parámetros** en el OBJECT de opciones no importa, puesto que se colocarán en el lugar apropiado en la representación final generada. También es importante observar que cada parámetro tiene un valor que hace que la fecha se representa de una forma particular. Dependiendo de los demás valores presentes, la representación puede variar ligeramente (`por ejemplo, la opción era en formato largo necesita mostrar año y otros datos aunque no se indiquen`).

A continuación puedes ver un gráfico donde se indican los diferentes valores de estas opciones, para una fecha concreta:



SUBIR A TABS

Fechas relativas: RelativeTimeFormat
Capítulo anterior

Fechas con Temporal API
Capítulo siguiente

Volver
Al índice

Acceder a Discord
Comunidad de Manz.dev

RELACIONADOS

En mis canales de Youtube @ManzDev y ManzDevTv, tienes más contenido...

APRENDER MÁS

Si lo prefieres, puedes aprender también sobre estas temáticas:

¿QUIÉN SOY YO?

Soy Manz, vivo en Tenerife (España) y soy streamer partner en Twitch y profesor. Me apasiona el universo de la programación web, el diseño y desarrollo web y la tecnología en general. Aunque soy full-stack, mi pasión es el front-end, la terminal y crear cosas divertidas y locas.

Puedes encontrar más sobre mi en Manz.dev

Twitch Youtube Twitter

Instagram Tiktok Linkedin

GitHub Discord

Creado por Manz con ❤ Alojado en DigitalOcean.