  
ACTIVIDADES  
6 DE 10 06

## API de fechas del JavaScript

Los navegadores permiten que nosotros nos comuniquemos con ellos a través de una *interface* que posee una lista de métodos de objetos a los que podemos acceder con JavaScript.

Dentro de esta lista vamos a acceder a un objeto llamado *Date()*, pero primero necesitamos instanciar ese objeto.

```
const data = new Date()  
\\ Wed Oct 14 2020 14:14:24 GMT-0600 (Ciudad de México)
```

[COPIA EL CÓDIGO](#)


Podemos ver que accedimos a varias informaciones de la fecha. El objeto *Date()* posee varios métodos para trabajar con fechas. Por ejemplo, si quisiéramos editar esa primera información para un formato que exhibirá la fecha separada por diagonales podemos utilizar el método *toLocaleDateString*:

  
DISCORD  
ALURA  
FORO DEL  
CURSO  
VOLVER AL  
DASHBOARD  
MODO  
NOCTURNO

66.0k xp



13%

  
ACTIVIDADES  
6 DE 10  
DISCORD  
ALURA  
FORO DEL  
CURSO  
VOLVER AL  
DASHBOARD  
MODO  
NOCTURNO

66.0k xp



13%

```
data.toLocaleDateString('es-MX')  
\\ "10/14/2020"
```

COPIA EL CÓDIGO

Ese formato puede ser configurable: por ejemplo, podemos crear un objeto que va a contener la llave de la fecha y al valor para definir cómo queremos exhibir la fecha:

```
const dataOptions = {  
  weekend: 'long',  
  year: 'numeric',  
  month: 'long',  
  day: 'numeric'  
}
```

COPIA EL CÓDIGO

Ahora necesitamos llamar `dataOptions` como segundo parámetro:

```
data.toLocaleDateString('es-MX', dataOptions)  
  
\\ 28 de agosto de 2020
```

COPIA EL CÓDIGO

¿Y el horario? El navegador posee un método llamado `toLocaleTimeString()` que muestra el horario del navegador y, así como en el método de fecha, pasemos es-mx como parámetro. Así, la fecha será formateada para el patrón utilizado en México.

```
data.toLocaleTimeString()  
\\ "9:04:54 AM"
```

[COPIA EL CÓDIGO](#)

El resultado es configurable así como el de la fecha, con el mismo proceso de crear objeto con llave y valor, que después pasamos como parámetro.

```
const horarioOptions = {  
  hour12: false,  
  hour: 'numeric',  
  minute: '2-digit',  
  second: '2-digit',  
  timeZone: 'America/Sao_Paulo'  
}
```

[COPIA EL CÓDIGO](#)

ACTIVIDADES  
6 DE 10



DISCORD  
ALURA



FORO DEL  
CURSO



VOLVER AL  
DASHBOARD



MODULO  
NOCTURNO



66.0k xp



13%

Usando `horarioOptions` como argumento de la función *ToLocaleTimeString*, tenemos:

```
data.toLocaleTimeString('es-MX', horarioOptions)
\\ "9:04:54"
```

[COPIA EL CÓDIGO](#)

Podemos combinar todas esas opciones utilizando el método `toLocaleString()`. Usando esos tres puntos antes del objeto, estamos indicando que todas las llaves/valor del objeto pasarán para ese nuevo objeto. Esa sintaxis se llama *spread operator*.

```
data.toLocaleString('es-MX', {
  ...dataOptions,
  ...horarioOptions
})
\\ "28 de agosto de 2020 9:04:54"
```

[COPIA EL CÓDIGO](#)

Si necesitamos usar ese formato en varios lugares del código, podemos utilizar el objeto *Intl.DateTimeFormat* que es un

ACTIVIDADES  
6 DE 10

DISCORD  
ALURA

FORO DEL  
CURSO

VOLVER AL  
DASHBOARD

MODO  
NOCTURNO



66.0k xp



13%

*constructor*, o sea, recibirá informaciones iniciales de cómo queremos que la fecha esté formateada.

```
const formataData = new Intl.DateTimeFormat('es-MX',  
  ...dataOptions,  
  ...horarioOptions  
))
```

[COPIA EL CÓDIGO](#)

Para terminar, llamando el método *format*, podemos formatear diferentes fechas caso sea necesario.

```
formataData.format(data)  
\\ “28 de agosto de 2020 9:04:54”
```

[COPIA EL CÓDIGO](#)

Está claro que trabajar con fechas utilizando API del navegador nos trae ventajas y desventajas, y depende de tu proyecto aprovechar esa flexibilidad de customizaciones.

ACTIVIDADES  
6 DE 10

DISCORD  
ALURA

FORO DEL  
CURSO

VOLVER AL  
DASHBOARD


MODO  
NOCTURNO




66.0k xp




13%




ACTIVIDADES  
6 DE 10




DISCORD  
ALURA




FORO DEL  
CURSO



VOLVER AL  
DASHBOARD



MODO  
NOCTURNO



66.0k xp

