06

Para saber mais: API de datas do JavaScript

Os navegadores permitem que nós nos comuniquemos com eles através de uma **interface**, que possui uma lista de métodos de objetos que conseguimos acessar com JavaScript.

Dentro desta lista vamos acessar um objeto chamado **Date()**, mas primeiro precisamos instanciar esse objeto.

```
const data = new Date()
\\ Wed Oct 14 2020 14:14:24 GMT-0300 (Brasilia Standard Time)
```

Podemos ver que acessamos várias informações da data. O objeto **Date()** possui vários métodos para lidar com datas. Por exemplo, se quisermos editar essa primeira informação para um formato que vai exibir a data separada por barra podemos usar o método **toLocaleDateString**:

```
data.toLocaleDateString('pt-BR')
\\ "10/14/2020"
```

Esse formato pode ser configurável: por exemplo, podemos criar um objeto que vai conter a chave que diz respeito à data e ao valor para definir como queremos exibir a data:

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

Agora basta chamar `dataOptions` como segundo parâmetro:

```
data.toLocaleDateString('pt-BR', dataOptions)

\\ 28 de agosto de 2020
```

E o horário? O navegador possui um método chamado **toLocaleTimeString()**, que mostra o horário do navegador e, assim como no método de data, passamos pt-br como parâmetro. Assim, a data é formatada para o padrão que usamos aqui no Brasil.

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

O resultado é configurável assim como o de data, com o mesmo processo de criar um objeto com chave e valor, que depois passamos como parâmetro.

```
const horarioOptions = {
   hour12: false,
```

```
    hour: 'numeric',
    minute: '2-digit',
    second: '2-digit',
    timeZone: 'America/Sao_Paulo'
}
```

Usando `horarioOptions` como argumento da função **toLocaleTimeString**, temos:

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

Podemos combinar todas essas opções utilizando o método toLocaleString(). Usando esses três pontos antes do objeto, estamos indicando que todas as chaves/valor do objeto vão ser passadas para esse novo objeto. Essa sintaxe chama-se **spread operator**.

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

Se precisarmos usar essa formatação em vários lugares do código, podemos utilizar o objeto **Intl.DateTimeFormat** que é um **construtor**, ou seja, ele vai receber informações iniciais de como queremos que a data esteja formatada.

```
const formataData = new Intl.DateTimeFormat('pt-BR', {
    ...dataOptions,
    ...horarioOptions
})
```

Por fim, chamando o método **format**, conseguimos formatar diferentes datas caso necessário.

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

Ficou claro que trabalhar com datas utilizando API do navegador traz vantagens e desvantagens, e depende do seu projeto aproveitar essa flexibilidade de customizações.