Handle format Date Time and Currency in Javascript

1. Date Time

When you want to display the Date and Time, you should follow the steps below.

1. Get the time stamps or the ISO formatted date from REST API
2. Creates a `Date` object
3. We have various ways to handle show datetime.

a) The `toLocaleString()` or `toLocaleDateString()` and `toLocaleTimeString()`

First, we can use the `toLocaleString()` or `toLocaleDateString()` and `toLocaleTimeString()` methods or a date library to display the local time.

`toLocaleTimeString()` always puts time in there, using default time format if you don't specify it.

```
const dateTime = new Date();
const timeString = dateTime.toLocaleTimeString();

console.log(timeString) ==> 11:08:11 AM
```

`toLocaleDateString()` does the same thing, but for date instead of time:

```
const dateTime = new Date();
const timeString = dateTime.toLocaleDateString();

console.log(timeString) ==> 8/12/2023
```

`toLocaleString()` allows you to format your date the way you like, it won't put anything extra.

```
const dateTime = new Date();
const timeString = dateTime.toLocaleString()

console.log(timeString) ==> '8/18/2023, 11:08:11 AM'
```

b) The `Intl.DateTimeFormat` function

It has two parameters, `locales` and `options`

Locales

The locale is an optional parameter that can be passed as a string. It represents a specific geographical, political, or cultural region. It just formats the number based on the locale and is not the currency formatting.

```
const date = new Date();

Intl.DateTimeFormat('ko').format(date) ==> '2023. 8. 18.'
Intl.DateTimeFormat('en-EN').format(date) ==> '8/18/2023'
```

Options

This is the options parameter and you can use it to apply more formatting for showing date. This is a JavaScript object that holds other parameters like

```
const date = new Date();

console.log(
  new Intl.DateTimeFormat('en-GB', { dateStyle: 'full', timeStyle: 'long',
  timeZone: 'Australia/Sydney' }).format(date),
); ==> 'Friday, 18 August 2023 at 16:44:26 GMT+10'
```

here are more options you can use in the official documentation.

c) The Date libraries

Finally, we can use lib such as momentjs, or dayjs to format date

```
const date = new Date();
// basic
const dateMoment = moment(date).format("DD/MM/YYYY")
const dateJS = dayjs(date).format('DD/MM/YYYY')

console.log(dateMoment) ==> '18/08/2023'
console.log(dateJS) ==> '18/08/2023'

// use locale
moment.locale('en-au')
const dateLocaleAUMoment = moment(date).format('L');

console.log(dateLocaleAuMoment) ==> '18/08/2023'
```

Common Date Formatting Patterns

1. **Specific Date Format**: To display a date in a specific format, such as `DD/MM/YYYY`, you can use `Intl.DateTimeFormat` with the appropriate options.

```
const date = new Date();
// frist way
const formatter = new Intl.DateTimeFormat('en-AU',
{ day: '2-digit', month: '2-digit', year: 'numeric' });
const formattedDate = formatter.format(date);

// second way
const formatterDateString = date.toLocaleDateString('en-AU');

console.log(formattedDate); ==> '18/08/2023'
console.log(formatterDateString); ==> '18/08/2023'
```

2. **Time Format**: To format the time portion of a date, you can use the `hour`, `minute`, and `second` options.

```
const date = new Date();

// first way
const formatter = new Intl.DateTimeFormat('en-AU', { hour: '2-digit', minute: '2-digit', second: '2-digit' });
const formattedTime = formatter.format(date);

// second way
const formatterTimeString = date.toLocaleTimeString('en-AU');

console.log(formattedTime); ==> '12:00:00 AM'
console.log(formatterTimeString); ==> '12:00:00 AM'
```

2. Currency

For the currency format, we should use `Intl.NumberFormat` function to show currency. It has two major parameters, `locales` and `options`

Note: `Intl.NumberFormat()` can be called with or without new. Both will create a new `Intl.NumberFormat` instance.

When you use the `Intl.NumberFormat()` constructor without passing any locale or option, it will only format the number by adding commas.

```
const price = 14340;
console.log(new Intl.NumberFormat().format(price)); // 14,340
```

Locales

The locale is an optional parameter that can be passed as a string. It represents a specific geographical, political, or cultural region. It just formats the number based on the locale and is not the currency formatting.

```
const price = 143450;

console.log(new Intl.NumberFormat('en-US').format(price)); // 143,450
console.log(new Intl.NumberFormat('en-IN').format(price)); // 1,43,450
console.log(new Intl.NumberFormat('en-DE').format(price)); // 143.450
```

Options

This is the main parameter and you can use it to apply more formatting like that of currency. This is a JavaScript object that holds other parameters like:

- `style`: You use this to specify the type of formatting you want. This takes in values like decimals, currency, and units. For this article, you will use **currency** because that is the style in which you want to format the number.
- `currency`: Another option is currency. You can use this option to specify the currency you want to format to, such as `'USD'`, `'CAD'`, `'GBP``'`, `'INR'` and lots more. This will also help provide the symbol in the appropriate position based on the locale.

```
const price = 143450;

// format number to US dollar
const USDollar = new Intl.NumberFormat('en-US', {
    style: 'currency',
    currency: 'USD',
});

// format number to US dollar
const AUDollar = new Intl.NumberFormat('en-AU', {
    style: 'currency',
    currency: 'AUD',
});

// format number to British pounds
let pounds = Intl.NumberFormat('en-GB', {
    style: 'currency',
    currency: 'GBP',
});

console.log('Dollars: ' + USDollar.format(price));
// Dollars: $143,450.00

console.log('AU Dollars: ' + AUDollar.format(price));
```

```
// AUDollars: $143,450.00

console.log(`Pounds: ${pounds.format(price)}`);
// Pounds: £143,450.00
```