The symbols `${`, `}`, and `}` are part of a JavaScript template literal, which is a way to embed expressions inside string literals using backticks (`).

Here's an explanation of each symbol's purpose within the template literal:

Javascript:

```
countryIcon.src = `https://flagcdn.com/144x108/${weatherInfo?.sys
```

- The `$` symbol is used to signal the start of a template literal expression.
- The `{}` curly braces are used to enclose an expression that will be evaluated and inserted into the string. In this case, the expression is `weatherInfo?.sys?.country.toLowerCase()`, which will evaluate to the lowercase two-letter country code used to generate the URL for the flag image.
- The `}` symbol marks the end of the expression and the end of the template literal.

Template literals are a powerful way to dynamically generate strings in JavaScript, allowing you to easily insert variables and expressions without having to concatenate multiple strings together.

What is Optional Chaining Operator ?

Optional chaining is a new JavaScript operator introduced in ECMAScript 2020 that makes it easier to safely access nested properties and methods of an object, without worrying about encountering errors if a property or method does not exist or is undefined. The operator is denoted by the question mark (`?`) symbol and is often referred to as the "nullish coalescing operator".

Here's an example of how optional chaining can be used:

Javascript:

```
let user = { name: 'Alice', address: { city: 'New York', state: 'NY', zip: '10001' } };

let zipCode = user?.address?.zip;

console.log(zipCode); // '10001'
```

In this example, we have an object called `user` with two properties: `name` and `address`. The `address` property is itself an object that contains three sub-properties: `city`, `state`, and `zip`. We want to access the `zip` property of the `address` object, but we don't want to risk running into an error if the `address` object is null or undefined.

Using optional chaining, we can add a `?` symbol after each object property or method call, indicating that we want to perform the operation only if the property or method exists and is not null or undefined. In this case, if the `address` property or the `zip` property do not exist, the expression will simply evaluate to `undefined`, without throwing an error.

Here's another example:

Javascript:

```
let user = null;

let zipCode = user?.address?.zip;

console.log(zipCode); // undefined
```

In this case, the `user` object is null, so attempting to access the `address` property would result in a TypeError if we were not using optional chaining. With optional chaining, however, the expression simply evaluates to `undefined`, indicating that the property does not exist.

Optional chaining is a powerful tool for avoiding errors in complex object hierarchies, and it can make your code much more robust and easier to maintain. However, it's important to use optional chaining judiciously and to understand its limitations, as it can sometimes mask underlying issues with your code.

More Examples:

here is a breakdown of each symbol or operator that we will use in next lecture:

```
cityName.innerText = weatherInfo?.name;
```

- `cityName` is the name of an HTML element that will display the city name.
- `.innerText` is a property of the HTML element that sets its visible text content.
- `weatherInfo` is an object that contains information about the weather.
- `?.` is the optional chaining operator, which allows you to access properties of an object that may be null or undefined without causing an error. In this case, it is used to access the `name` property of the `weatherInfo` object.

```
countryIcon.src = `https://flagcdn.com/144x108/${weatherInfo?.sys
```

- `countryIcon` is the name of an HTML element that will display the country flag.
- `.src` is a property of the HTML element that sets the URL of the image to display.
- `weatherInfo` is an object that contains information about the weather.
- `?.` is the optional chaining operator, which allows you to access properties of an object that may be null or undefined without causing an error. In this case, it is used to access the `sys` property of the `weatherInfo` object, and then the `country` property of the `sys` object.
- `.toLowerCase()` is a method that converts the value of a string to lowercase characters.

```
desc.innerText = weatherInfo?.weather?.[0]?.main;
```

- `desc` is the name of an HTML element that will display the weather description.
- `.innerText` is a property of the HTML element that sets its visible text content.
- `weatherInfo` is an object that contains information about the weather.
- `?.` is the optional chaining operator, which allows you to access properties of an object that may be null or undefined without causing an error. In this case, it is used to access the `weather` property of the `weatherInfo` object, and then the first element of the `weather` array, and then the `main` property of that element.

```
weatherIcon.src = `http://openweathermap.org/img/w/${weatherInfo?[0]?.icon}.png`;
```

- `weatherIcon` is the name of an HTML element that will display the weather icon.
- `.src` is a property of the HTML element that sets the URL of the image to display.
- `weatherInfo` is an object that contains information about the weather.
- `?.` is the optional chaining operator, which allows you to access properties of an object that may be null or undefined without causing an error. In this case, it is used to access the `weather` property of the `weatherInfo` object, and then the first element of the `weather` array, and then the `icon` property of that element.

```
temp.innerText = `${weatherInfo?.main?.temp.toFixed(2)} °C`;
```

- `temp` is the name of an HTML element that will display the temperature.
- `.innerText` is a property of the HTML element that sets its visible text content.
- `weatherInfo` is an object that contains information about the weather.
- `?.` is the optional chaining operator, which allows you to access properties of an object that may be null or undefined without causing an error. In this case, it is used to access the `main` property of the `weatherInfo` object, and then the `temp` property of that object.
- `.toFixed()` is a method that formats a number with a specified number of digits after the decimal point.