Implement Static Properties and Methods in JavaScript Labs

Perform these labs on your own computer using Visual Studio 2022 or later, or VS Code 1.8x or later, to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Clone an RV Object

Create a new folder named **classes**.

Right mouse-click on the **classes** folder and add a new file named **RV.js**.

Add the following code into this file.

```
// Define an RV class
class RV {
  // Define constructor
  constructor(rvid, year, make, model, listPrice,
releaseDate) {
    // Initialize properties
    this.rvid = rvid;
    this.year = year;
    this.make = make;
    this.model = model;
    this.listPrice = listPrice;
    this.releaseDate = releaseDate;
  }

  // Static method to clone an
  // existing RV instance
  static clone(rv) {
    // Create a new instance of an RV class


    // Clone the original data

    // Return the cloned object
    return clone;
  }
}
```

Create a file named **rvs.html** and add the following HTML into the new file.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />

  <title>Clone an RV Object</title>

  <link href="styles/site.css" rel="stylesheet" />
</head>

<body>
  <header>
    <h1>Clone an RV Object</h1>
  </header>

  <main>
    <p>Open your F12 Browser Tools to view the
results.</p>
  </main>

  <script src="classes/RV.js"></script>
  <script>
    'use strict';

    // Create an RV object
    // Passing in 1, 2025, "Newmar", "New Aire", 500000,
new Date("2024-09-01") to the constructor
    let rv = new RV(1, 2025, "Newmar", "New Aire",
500000, new Date("2024-09-01"));
    // Display the RV object
    console.log("RV Object");
    console.log(rv);
    console.log("");

    // Clone the RV object here
    // Make a couple of changes to the clone
    newRv.make = "Tiffin";
    newRv.model = "Phaeton";

    // Display the cloned object
    console.log("Cloned RV Object");
    console.log(newRv);
  </script>
</body>

</html>
```

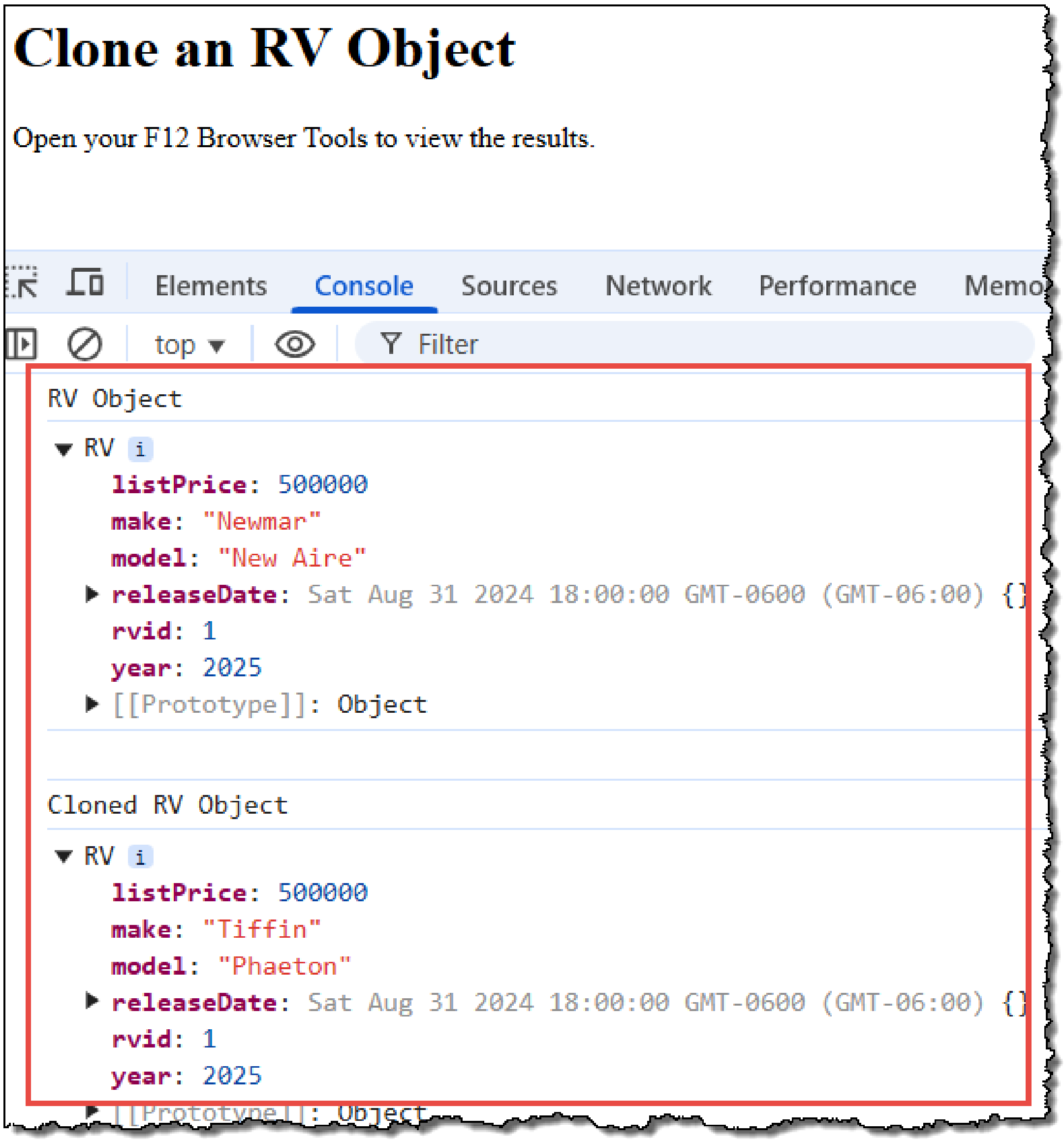
Write the **clone()** method of the RV class, create a new instance of an RV object and assign this new instance to a variable named *clone*.

Write the code to clone the *rv* parameter coming into this clone() method into the *clone* variable.

Within the **rvs.html** file, write the appropriate code to clone the exiting RV object into a new variable named *newRv*.

Try It Out

Display the **rvs.html** file in your browser and your page should look like Figure 1.



Clone an RV Object

Open your F12 Browser Tools to view the results.

```
RV Object
▼ RV i
    listPrice: 500000
    make: "Newmar"
    model: "New Aire"
  ▶ releaseDate: Sat Aug 31 2024 18:00:00 GMT-0600 (GMT-06:00) {}
    rvid: 1
    year: 2025
  ▶ [[Prototype]]: Object

Cloned RV Object
▼ RV i
    listPrice: 500000
    make: "Tiffin"
    model: "Phaeton"
  ▶ releaseDate: Sat Aug 31 2024 18:00:00 GMT-0600 (GMT-06:00) {}
    rvid: 1
    year: 2025
  ▶ [[Prototype]]: Object
```

Figure 1: Clone an RV Object.

Lab 2: Create a Number Helper Class

Right mouse-click on the **classes** folder and add a new file named **NumberHelper.js**.

Add the following code into this file.

```
class NumberHelper {
  static localeCode = 'en-US';
  static isoCurrencyCode = 'USD';
  static toCurrency(value, locale, isoCode) {
    // Check parameters passed in
    locale ??= NumberHelper.localeCode;
    isoCode ??= NumberHelper.isoCurrencyCode;

    // Convert value to a currency formatted
    // according to the locale specified
    return value.toLocaleString(locale,
      {
        style: 'currency',
        currency: isoCode
      });
  }
}
```

Create a file named **numberHelper.html** and add the following HTML into the new file.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />

  <title>Using the NumberHelper Class</title>

  <link href="styles/site.css" rel="stylesheet" />
</head>

<body>
  <header>
    <h1>Using the NumberHelper Class</h1>
  </header>

  <main>
    <p>Open your F12 Browser Tools to view the
results.</p>
  </main>

  <script src="classes/NumberHelper.js"></script>
  <script>
    'use strict';

    let listPrice = 42.99
    // Display listPrice as US Currency


    // Display listPrice as German Euros
    // Locale Code = 'de-DE'
    // ISO Currency Code = 'EUR'


    // Set the localeCode property to 'de-DE'
    // Set the isoCurrencyCode property to 'EUR'
    // These properties are used until changed again


    // Display the listPrice in the default currency

  </script>
</body>

</html>
```

Write a line of code to display the *listPrice* variable as US currency.

Write a line of code to display the *listPrice* variable as German Euros passing in 'de-DE' and 'EUR' to the toCurrency() method.

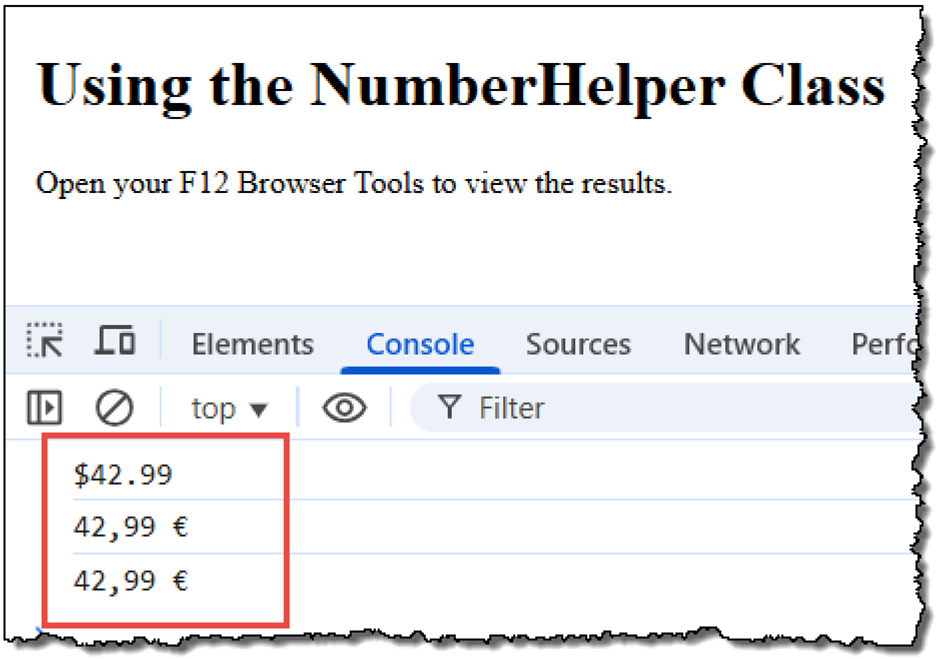
Set the *localeCode* property in the NumberHelper class to 'de-DE'.

Set the *isoCurrencyCode* property in the NumberHelper class to 'EUR'.

Display the *listPrice* by just calling toCurrency(listPrice).

Try It Out

Display the **numberHelper.html** file in your browser and your page should look like Figure 2.


Using the NumberHelper Class
Open your F12 Browser Tools to view the results.
Elements
Console
Sources
Network
Perfo
top
Filter

```
$42.99
42,99 €
42,99 €
```

Figure 2: Use the NumberHelper class to display currency values.

Lab 3: Create a DOM Helper Class

Right mouse-click on the **classes** folder and add a new file named **DOMHelper.js**. Add the following code into this file.

```
class DOMHelper {
  static getElement(id) {
    return document.getElementById(id);
  }

  static getValue(id) {
    return DOMHelper.getElement(id).value;
  }

  static getDateValue(id) {
    return new Date(DOMHelper.getValue(id) + " ");
  }

  static setDateValue(id, dt) {
    DOMHelper.getElement(id).value =
`${dt.getFullYear()}-${String(dt.getMonth() +
1).padStart(2, '0')}-${String(dt.getDate()).padStart(2,
'0')}`;
```

```
  }

  static setValue(id, value) {
    DOMHelper.getElement(id).value = value;
  }

  static setText(id, value) {
    DOMHelper.getElement(id).innerText = value;
  }
}
```

Create a file named **domHelper.html** and add the following HTML into the new file.

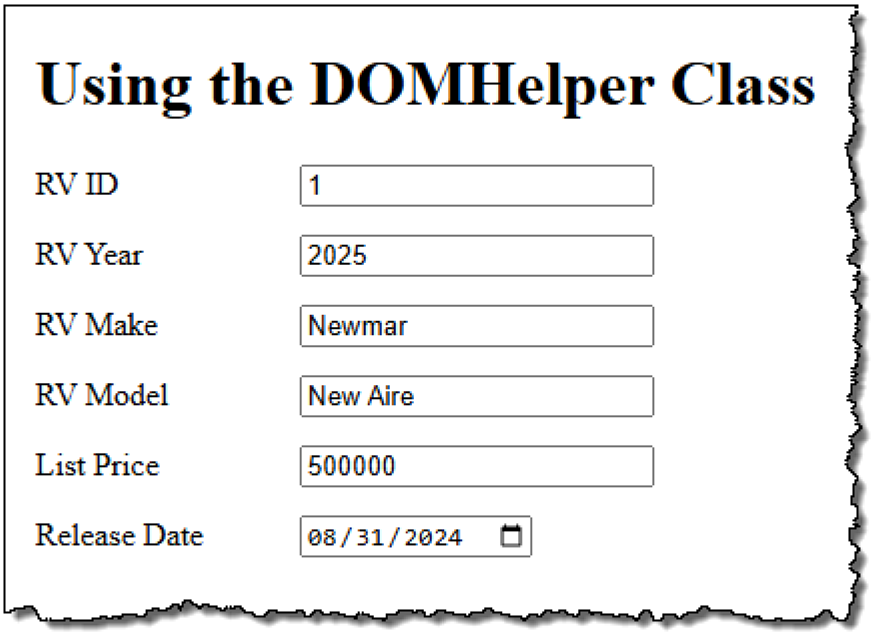
```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />

  <title>Using the DOMHelper Class</title>

  <link href="styles/site.css" rel="stylesheet" />
</head>

<body>
  <header>
    <h1>Using the DOMHelper Class</h1>
  </header>

  <main>
    <form>
      <div>
        <label for="rvid">RV ID</label>
        <input type="number" id="rvid" name="rvid"
readonly />
      </div>
      <div>
        <label for="year">RV Year</label>
        <input type="number" id="year" name="year" />
      </div>
      <div>
        <label for="make">RV Make</label>
        <input type="text" id="make" name="make" />
      </div>
      <div>
```

```
        <label for="model">RV Model</label>
        <input type="text" id="model" name="model" />
      </div>
      <div>
        <label for="listPrice">List Price</label>
        <input type="number" id="listPrice"
name="listPrice" />
      </div>
      <div>
        <label for="releaseDate">Release Date</label>
        <input type="date" id="releaseDate"
name="releaseDate" />
      </div>
    </form>
  </main>

  <script src="classes/RV.js"></script>
  <script src="classes/DOMHelper.js"></script>
  <script>
    'use strict';

    // Create an RV object
    // Passing in 1, 2025, "Newmar", "New Aire", 500000,
new Date("2024-09-01") to the constructor
    let rv = new RV(1, 2025, "Newmar", "New Aire",
500000, new Date("2024-09-01"));

    // Use the DOMHelper class to set the
    // appropriate RV object values into
    // their respective input fields


  </script>
</body>

</html>
```

Write the appropriate methods in the **DOMHelper** class to set the property values from the **RV** object into their respective input fields.

Try It Out

Display the **domHelper.html** file in your browser and your page should look like Figure 3.



Using the DOMHelper Class

RV ID	1
RV Year	2025
RV Make	Newmar
RV Model	New Aire
List Price	500000
Release Date	08/31/2024

Figure 3: Use the DOMHelper class to manipulate the DOM.

Lab 4: Create a Date Helper Class

Right mouse-click on the **classes** folder and add a new file named **DateHelper.js**. Add the following code into this file.

```
class DateHelper {
  static getMonthNames() {
    return ['January', 'February', 'March', 'April',
'May', 'June', 'July', 'August', 'September', 'October',
'November', 'December'];
  }

  static getShortMonthNames() {
    return ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
  }

  static createDayOptions(month, year) {
    let html = "";

    // Calculate last day of the month
    // Zero for the last day means one day less than the
first day of the month
    let dt = new Date(year, month + 1, 0);
    for (let day = 1; day <= dt.getDate(); day++) {
      html += `<option value="${day}">${day}</option>`;
    }

    return html;
  }

  static createMonthOptions(useShortNames) {
    let html = "";
    let months;

    if (useShortNames) {
      months = DateHelper.getShortMonthNames();
    }
    else {
      months = DateHelper.getMonthNames();
    }

    for (let index = 0; index < months.length; index++)
{
      html += `<option
value="${index}">${months[index]}</option>`;
    }

    return html;
  }

  static createYearOptions(begin, end, reverse) {
    let html = "";

    if (reverse) {
      for (let year = end; year >= begin; year--) {
        html += `<option
value="${year}">${year}</option>`;
      }
    }
    else {
      for (let year = begin; year <= end; year++) {
        html += `<option
value="${year}">${year}</option>`;
      }
    }

    return html;
  }
}
```

Create a file named **dateHelper.html** and add the following HTML into the new file.

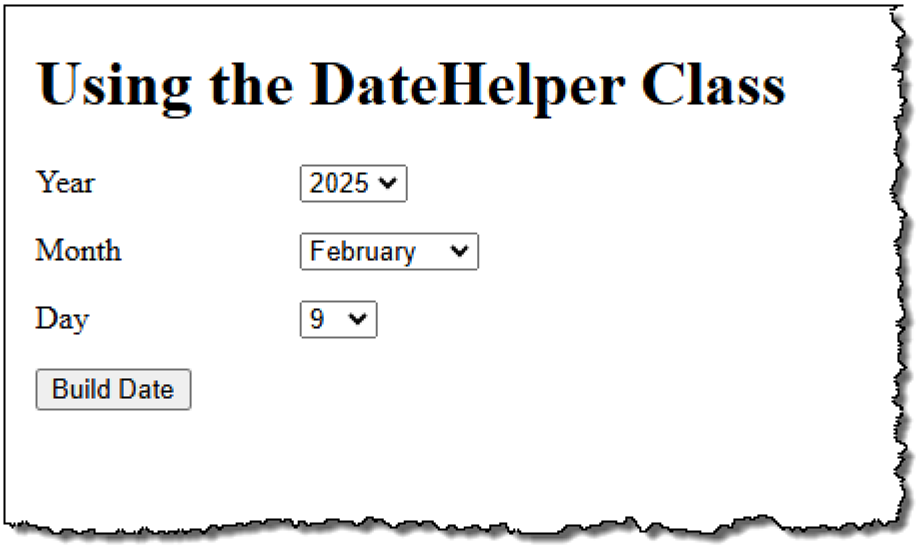
```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />

  <title>Using the DateHelper Class</title>

  <link href="styles/site.css" rel="stylesheet" />
</head>

<body>
  <header>
    <h1>Using the DateHelper Class</h1>
  </header>

  <main>
    <form>
      <div>
        <label for="years">Year</label>
        <select id="years" name="years"
onchange="loadDays();">
        </select>
      </div>
      <div>
        <label for="months">Month</label>
        <select id="months" name="months"
onchange="loadDays();">
        </select>
      </div>
      <div>
        <label for="days">Day</label>
        <select id="days" name="days">
        </select>
      </div>
    </form>
    <button type="button" onclick="buildDate();">Build
Date</button>
    <p id="output"></p>
  </main>

  <script src="classes/DOMHelper.js"></script>
  <script src="classes/DateHelper.js"></script>
  <script>
    'use strict';

    // Load years and months first
    loadYears();
    loadMonths();

    // Load days after years/months
    loadDays();

    // Set each select to today's date
    let dt = new Date();
    DOMHelper.setValue("years", dt.getFullYear());
    DOMHelper.setValue("months", dt.getMonth());
    DOMHelper.setValue("days", dt.getDate());

    function buildDate() {
      let day = DOMHelper.getValue("days");
      let month = DOMHelper.getValue("months");
      let year = DOMHelper.getValue("years");

      let dt = new Date(year, month, day);

      setText("output", dt.toLocaleDateString());
    }

    function loadDays() {
      let month = Number(DOMHelper.getValue("months"));
      let year = DOMHelper.getValue("years");

      let html = DateHelper.createDayOptions(month,
year);

      DOMHelper.getElement("days").innerHTML = html;
    }

    function loadMonths() {
      DOMHelper.getElement("months").innerHTML =
DateHelper.createMonthOptions();
    }

    function loadYears() {
      DOMHelper.getElement("years").innerHTML =
DateHelper.createYearOptions(2000, 2026);
    }
  </script>
</body>
```

```
</html>
```

Try It Out

There is **no code** to write here. Just pay attention to the various code highlighted in bold.

Display the **dateHelper.html** file in your browser and it should look like Figure 4. Try changing the **Month** drop-down and then look at the **Day** drop-down. The number of days will change based on the month.



Using the DateHelper Class

Year	2025
Month	February
Day	9

Figure 4: Use the DateHelper class to build the appropriate date select elements.

THIS PAGE INTENTIONALLY LEFT BLANK

Answers

Lab 1

In the **classes\RV.js** file, add the following method.

```
// Static method to clone an
// existing RV instance
static clone(rv) {
  // Create a new instance of an RV class
  let clone = new RV();

  // Clone the original data
  Object.assign(clone, rv);

  // Return the cloned object
  return clone;
}
```

In the **rvs.html** file add the following line of code.

```
// Clone the RV object
let newRv = RV.clone(rv);
```

Lab 2

```
let listPrice = 42.99

// Display listPrice as US Currency
console.log(NumberHelper.toCurrency(listPrice));

// Display listPrice as German Euros
// Locale Code = 'de-DE'
// ISO Currency Code = 'EUR'
console.log(NumberHelper.toCurrency(listPrice, 'de-DE',
'EUR'));

// Set the localeCode property to 'de-DE'
// Set the isoCurrencyCode property to 'EUR'
// These properties are used until changed again
NumberHelper.localeCode = 'de-DE';
NumberHelper.isoCurrencyCode = 'EUR';
// Display the listPrice in the default currency
```

```
console.log(NumberHelper.toCurrency(listPrice));
```

Lab 3

```
// Use the DOMHelper class to set the
// appropriate RV object values into
// their respective input fields
DOMHelper.setValue("rvid", rv.rvid);
DOMHelper.setValue("year", rv.year);
DOMHelper.setValue("make", rv.make);
DOMHelper.setValue("model", rv.model);
DOMHelper.setValue("listPrice", rv.listPrice);
DOMHelper.setDateValue("releaseDate", rv.releaseDate);
```