Summary

In this lesson, we will summarize all the concepts we learned in this chapter.

JavaScript has a number of great features that allow you to implement advanced scenarios. In this chapter you learned the most powerful of them.

You can use the **JSON** syntax within your code to create objects, serialize them to strings, and parse JSON strings back to objects.

JavaScript **functions** are the cornerstones of useful programming patterns. Functions can have an arbitrary number of arguments, and you have the opportunity to check their value and type during runtime. You can be very flexible with how you take the parameters provided by a function call into account. JavaScript allows you to provide **function expressions**, accept functions as arguments, and retrieve functions as return values. **Anonymous functions** and **closures** let you work with JavaScript in a manner that resembles **functional programming**.

Objects are really first-class citizens in JavaScript. You can create properties programmatically and implicitly and you can influence their behavior. With the help of **constructor functions** and **function prototypes**, you can implement **object inheritance** with multiple techniques, including **prototype chaining** and **pseudo-classical inheritance**, among others.

The **ECMAScript** specification defines the **Browser Object Model (BOM)** that allows you to interact with the browser your document is displayed in. The window, history, location, navigator, and screen objects all provide you properties and methods to query browser values, such as the current window's position, the parts of the documents URL, etc., and initiate actions such as displaying dialogs, opening or closing browser windows, and many more.

Just like other curly-brace programming languages, JavaScript has its own **error handling** mechanism based on the `try-catch` construct.