JavaScript: Functions, Arrays, Objects, and Events

JavaScript Functions

- Built-in functions
  - JavaScript provides several objects that have a rich collection of methods for performing common math calculations, string manipulations, date and time manipulations, and manipulations of collections of data called arrays
- Customized functions
  - You can define your own functions that perform specific tasks

Customized Function template

```
function name (input parameters, if any) {
        // function code goes here
}
```

- Three ways to return control to the point at which a function was invoked
  - Reaching the function-ending right brace
  - Executing the statement *return*;
  - Executing the statement "*return expression*;" to return the value of *expression* to the caller script

Functions are Objects

- A function can be considered as an object and referenced by a variable

  e.g., var obj = function(){
                  console.log("Hello");};
- A function without a name is an **anonymous function**
- A function can be used as an argument to another function

  e.g., window.setTimeOut(obj, 5000);

Demo 1: Functions

- Define a function that takes a person's height in inches and weight in pounds and calculates the BMI (rounded to an integer).

  BMI = 703 * weight / (height * height)

Arrays

- An array is a group of variables that have the same name and normally are of the same type
- Each individual variable is called an element
- We may refer to any one of these elements by giving the array’s name followed by the position number of the element in square brackets (`[]`)

Arrays (Cont.)

- The first element in every array is the zeroth element.
- The *i*th element of array `c` is referred to as `c[i-1]`.
- Every array in JavaScript knows its own length, which it stores in its `length` attribute and can be found with the expression *`arrayname`*`.length`

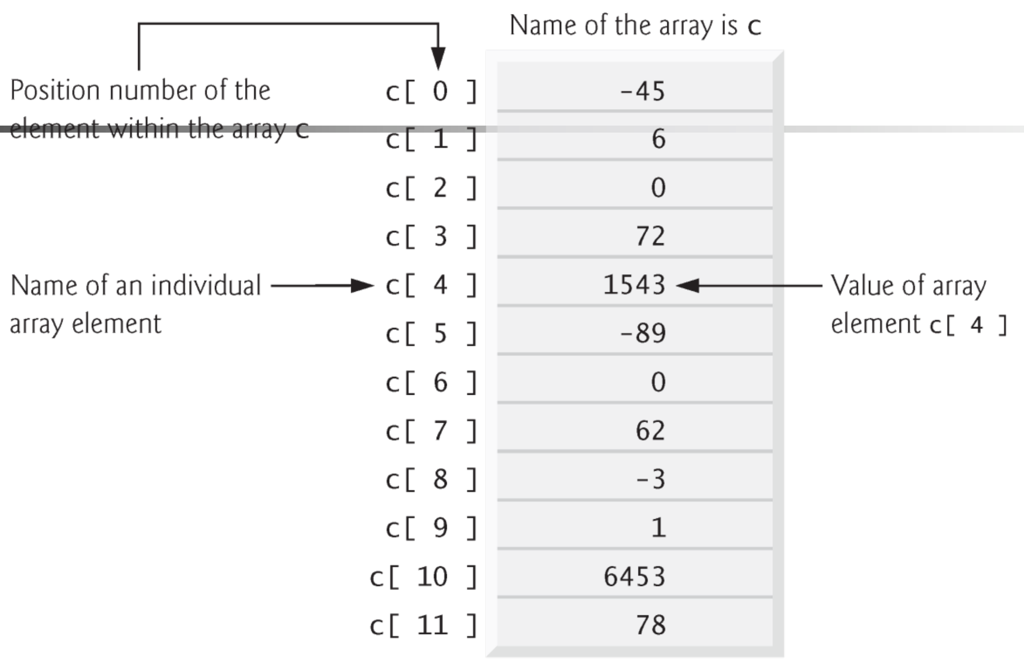

Fig. 10.1 | Array with 12 elements.

Declaring and Allocating Arrays

- JavaScript arrays are `Array` **objects**.
- You use the ***new*** operator to create a new array and to specify the number of elements in an array.

E.g., var n1= new Array(3);

        var n3 = new Array();

        var n2 = ["Ford", "Toyota", "Honda"];

Array Methods

- push(): adds new element to the end of array
- pop(): removes last element in array and returns the removed element
- shift(): removes first element in array and returns the removed element
- concat(): concatenates two arrays into one
- sort(): sorts an array
- indexOf(): search array for an element and returns its position index

Examples:

```
var nums = [5, 3, 6, 2];
nums.push(1);
console.log(nums);               //[5,3,6,2,1]
console.log(nums.pop());         //1
console.log(nums);               //[5,3,6,2]
console.log(nums.shift());       //5
console.log(nums);               //[3,6,2]
console.log(nums.concat([3,5])); //[3,6,2,3,5]
console.log(nums.sort());        //[2,3,6]
console.log(nums.indexOf(6));        //2
```

Events

- JavaScript events
  - allow scripts to respond to user interactions and modify the page accordingly
- Events and event handling
  - help make web applications more dynamic and interactive

Event Examples

- Click: When the user single clicks an HTML element
- Dblclick: When the user double clicks an element
- Change: When the user makes a selection change in a Select element
- Submit: When a form’s data is submitted
- Mouseover: When the mouse cursor enters an element, an `mouseover` event occurs for that element
- Mouseout: When the mouse cursor leaves the element, a `mouseout` event occurs for that element

Event Handlers

- An **event handler** is a function that responds to an event.
- Assigning an event handler to an event on a DOM node is called **registering an event handler**
- Method ***`addEventListener`*** can be called multiple times on a DOM node to register more than one event-handling method for an event.
- If a script in the `head` attempts to get a DOM node for an HTML element in the body, **`getElementById`** returns `null` because the body has not yet loaded

The Window Object

- The window object represents an open window in a browser
- If a document contains frames (<iframe> tag), there is a window object for the HTML document, and one additional window for each frame

The `load` Event

- The `window` object's `load` event fires when the window finishes loading successfully
  - i.e., all its children are loaded and all external files referenced by the page are loaded
- *Every* DOM element has a `load` event, but it's most commonly used on the `window` object.

Window Methods

- alert(): display an alert message and an OK butotn
- setTimeout(): call a function a specified number of miliseconds
- setInterval(): call a function at the specified interval in miliseconds
- Complete window properties and methods can be found here

Document Object

- The root of an HTML document
- Methods:
  - getElementById(): returns the value of the element at the specified id
  - writeln(): writes a line of output to the document (adds a new line at the end)
  - write(): writes output to the document
- Will talk more about this object in DOM

The DOMContentLoaded Event

- The DOMContentLoaded event fires when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading
- Window’s load event should be used only to detect a fully-loaded page

Demo 2: A running clock