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`

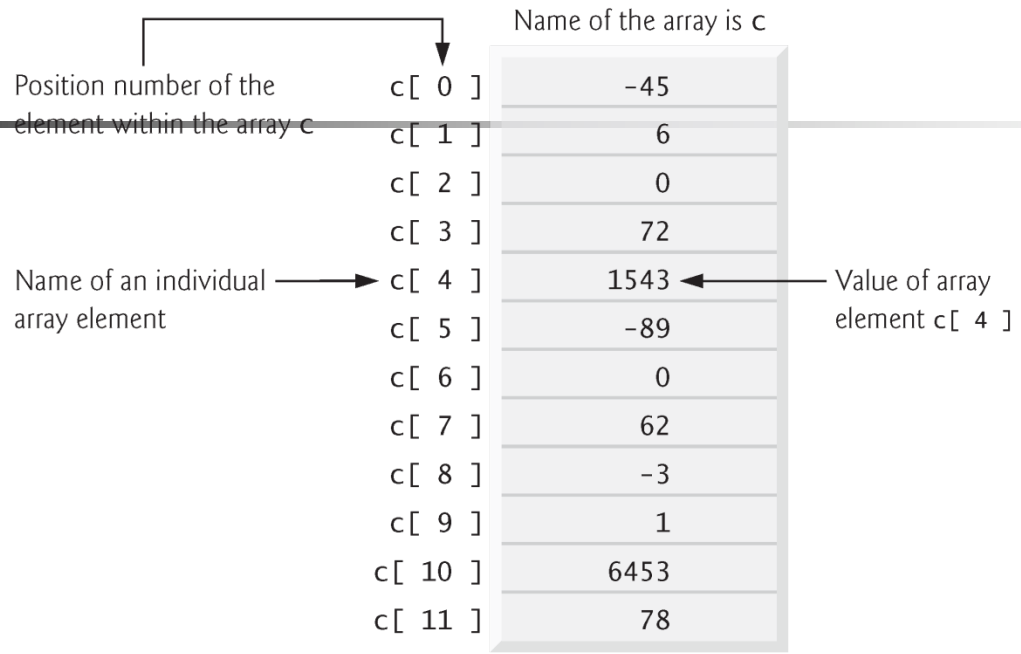


Fig. 10.1 | Array with 12 elements.

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();

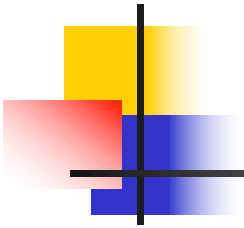
var n2 = ["Ford", "Toyota", "Honda"];

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

12.1 Document Object Model

- The Document Object Model gives you scripting access to *all* the elements on a web page. It defines
  - HTML5 elements as **objects**
  - The **Properties** of all HTML elements
  - The **methods** to access all HTML elements
  - The **events** for all HTML elements

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

What can JavaScript do?

- Using JavaScript, you can create, modify and remove elements in the page dynamically.
  - Change all the HTML elements in the page
  - Change all the HTML attribute values in the page
  - Change all the CSS styles in the page
  - Remove existing/add new HTML elements and attributes
  - React to all existing HTML elements' events in the page
  - Create new HTML events

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

12.2 Modeling a Document: DOM Nodes and Trees

- The nodes in a document make up the page's **DOM tree**, which describes the relationships among elements
- Nodes are related to each other through child-parent relationships
- A node can have multiple children, but **only one parent**
- Nodes with the same parent node are referred to as siblings
- The `html` node in a DOM tree is called the root node, which is the parent node of all HTML elements in the page

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

```
var myElement = document.getElementByID("intro");        //returns the element object with id="intro"

var x = document.getElementsByTagName("p");    //returns all <p> elements as an array

var x = document.getElementsByClassName("major");        //returns all elements with class="major" as
                                                                                an array
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Finding HTML Elements by CSS Selectors

- Example: Returns all <p> elements with class="intro"

```
var x=document.querySelectorAll("p.intro");
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Finding HTML Elements Using DOM Collections

- DOM has collections—groups of related objects on a page
- The `document` object has properties containing the `images` collection, `links` collection, `forms` collection, and `anchors` collection
  - Contain all the elements of the corresponding type on the page
- DOM collections are **links**, **images**, **forms**, and **tables**.
- The collection's **`length`** property specifies the number of items in the collection

Example:

```
document.links[0]          //the first <a> element
document.forms[0]          //the first <form> element
document.images[1]         //the second <img> element
document.tables[0]         //the first <table> element
document.tables.length     //the number of <table> elements
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Finding HTML Attributes

- Use the dot (.) operator to access element attribute values

Example:

```
<html>…
        <img id="myImage" src="oldpic.jpg"> …
</html>

var x=document.getElementById('myImage');
x.src = "newpic.jpg";
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Legacy Form Input Shortcut Accessor

```
<form name="myForm" method="post" action="http://www.vt.edu">
  <strong>Full Name:</strong><br />
  <input type="text" name="fullName" size="20" /><br />
  <strong>Address:</strong><br/>
  <textarea name="address" rows="3" cols="25"></textarea><br />
```

var x = document.forms[0].fullName //references the input element named “fullName” in the first <form> element

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

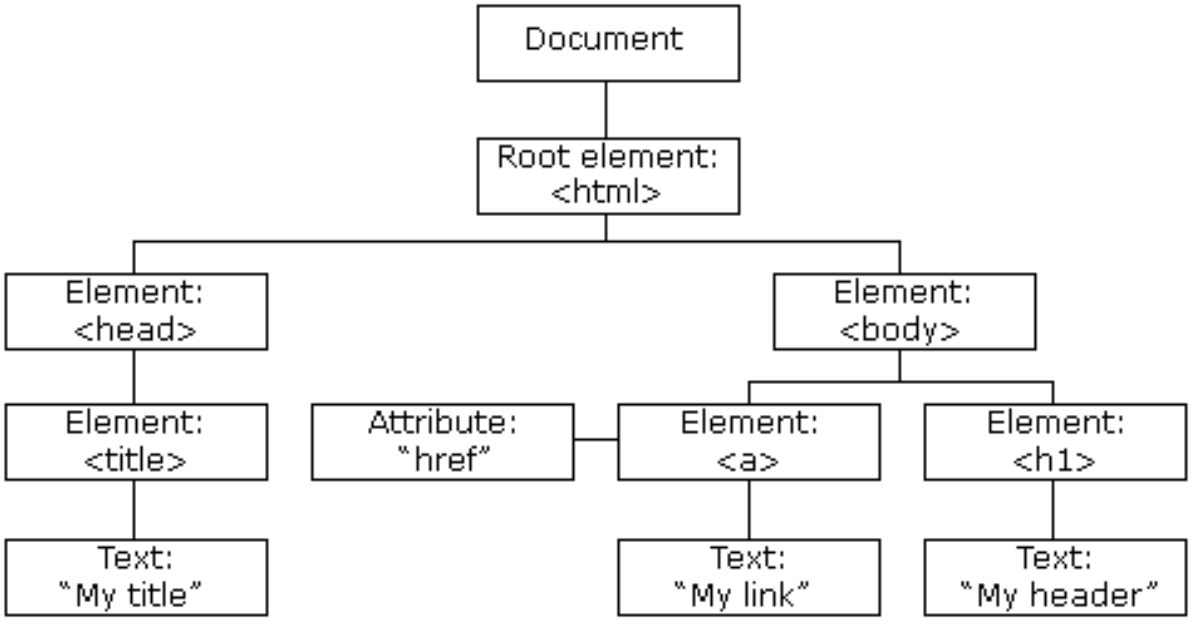
Node Properties

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Note: A whitespace character is an empty space without any visual representation in your page. A child/sibling node can be a whitespace.

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

The HTML DOM Tree of Objects



© w3schools.com.

Examples

```
document.getElementsByTagName("h1")[0].parentNode
//body element
document.getElementsByTagName("html")[0].firstChild
//head element
document.getElementsByTagName("html")[0].lastChild
//body element
document.getElementsByTagName("h1")[0].previousSibling

//a element
document.getElementsByTagName("a")[0].nextSibling
//h1 element
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Changing HTML Elements

Method	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = *new value*	Change the attribute value of an HTML element
element.setAttribute(*attribute, value*)	Change the attribute value of an HTML element
element.style.*property* = *new style*	Change the style of an HTML element

```
//the content of the element with id="p1" is updated to New text
var x=document.getElementById("p1");
x.innerHTML = "New text";
x.style.visibility = "hidden";      //hide the p1 element
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Adding and Deleting Elements

Method	Description
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Example

```
var newNode=document.createElement("p");
newNode.id="new";
var txt="I love Tech";

// attach a text to the new paragraph node as a child node
var textNode = document.createTextNode(txt);
newNode.appendChild(textNode);
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Adding Event Handlers

Method	Description
document.getElementById(*id*).onclick = function() {*code*}	Adding event handler code to an onclick event

```
//The tecommended way of creating an event handler
document.getElementById("p1").addEventListener("click",
myFunction, false);

//An alternative way
document.getElementById("p1").onclick = myFunction;
```

©1992-2012 by Pearson Education, Inc. All Rights Reserved.

Demo 1: Input form validation

©1992-2012 by Pearson Education, Inc. All Rights Reserved.