Tutorial 13 B
Exploring the Document Object Model

HTML, CSS,
and Dynamic HTML
5TH EDITION

Carey, P. (2013). New perspectives on HTML, CSS, and Dynamic HTML: Comprehensive (5th ed.). Australia ; United Kingdom: Course Technology Cengage Learning.

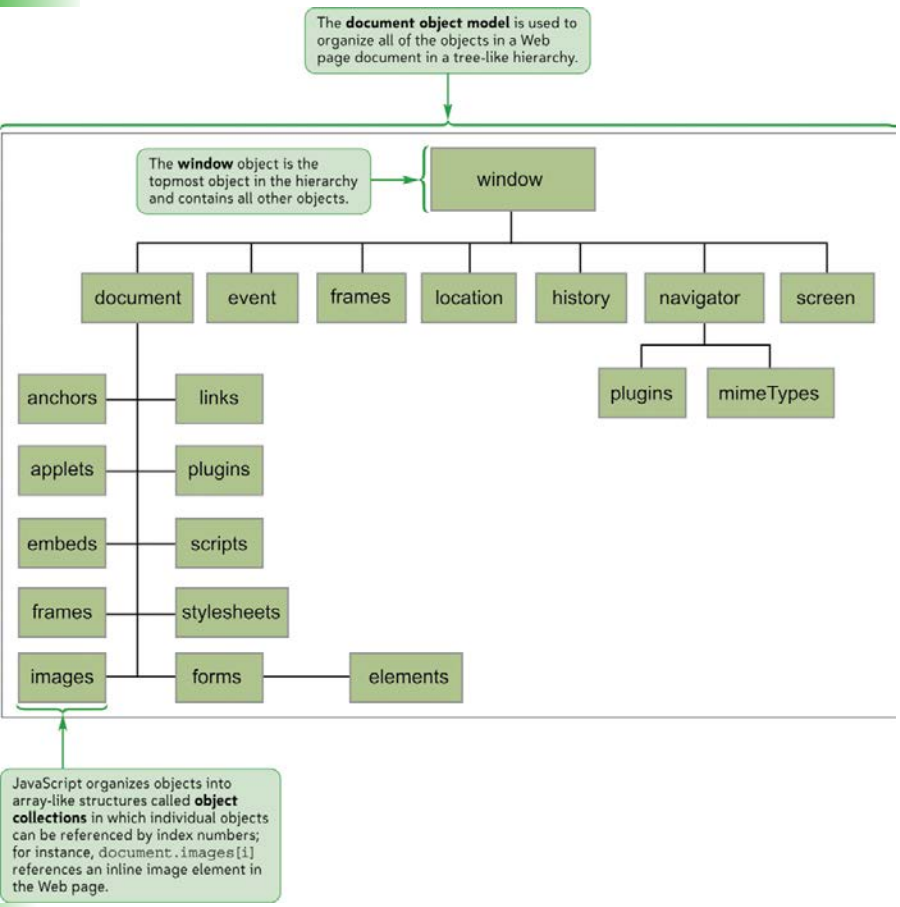
Objectives

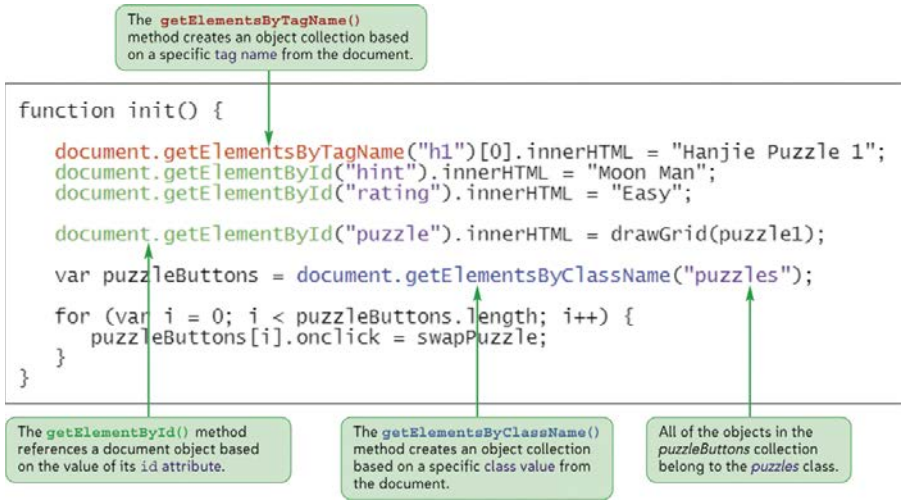
- Learn about objects and the document object model
- Reference document objects by ID, name, and tag name
- Write HTML code to a document object
- Write an event handler as an object property

Objectives

- Change the inline style of a document object
- Use a CSS selector in an object reference
- Loop through an object collection
- Create alert, confirm, and prompt dialog boxes

Document Object Model





The Document Object Model

- JavaScript is an **object-based language**—that is, it's based on manipulating objects by changing each object's properties or by applying a method or an action to each object, often in response to an event
- JavaScript supports four kinds of objects
  - Built-in objects
  - Document objects
  - Browser objects
  - Customized objects

The Document Object Model

Figure 13-4 Versions of the document object model

Document Object Model	Description
Internet Explorer DOM	A document object model provided for the Internet Explorer browser and including features not found in any W3C DOM
W3C DOM Level 1	The first DOM specification by the W3C, which supported all page and browser elements and handled all events occurring within the browser
W3C DOM Level 2	The second DOM specification, allowing for the capture of events, manipulation of CSS styles, working with element text, and document subsets
W3C DOM Level 3	The third DOM specification, providing a framework for working with document loading and saving, as well as working with DTDs and document validation
HTML5 DOM	The latest document object model based on HTML5 (still in development at the time of this writing)

Referencing Objects

Figure 13-5 Object names

Object Name	Description
`window`	The browser window
`document`	The Web document displayed in the window
`document.body`	The body of the Web document displayed in the browser window
`event`	Events or actions occurring within the browser window
`history`	The list of previously visited Web sites within the browser window
`location`	The URL of the document currently displayed in the browser window
`navigator`	The browser itself
`screen`	The screen displaying the document

Referencing Object Collections

- When more than one of the same type of object exists, these objects are organized into structures called **object collections**

Figure 13-6 Object collections

Object Collection	References
`document.anchors`	All anchors marked with the <a> tag
`document.applets`	All applets
`document.embeds`	All embed elements
`document.forms`	All Web forms
`document.`*`fname`*`.elements`	All form elements within the form *fname*
`document.images`	All inline images
`document.links`	All hypertext links
`document.plugins`	All plug-ins in the document
`document.styleSheets`	All style sheet elements
`navigator.plugins`	All plug-ins supported by the browser
`navigator.mimeTypes`	All MIME types supported by the browser
`window.frames`	All frames within the current browser window

Referencing Objects

- To reference an object as part of the collection in a document, use either

  collection[*idref*]

  or

  collection.*idref*

  where *idref* is either an index number representing the position of the object in the collection, or the value of the id assigned to that element

- To reference an array of elements based on the tag name, use

  object.getElementsByTagName(*tag*)

  where *object* is an object reference and *tag* is the name of the element tag.

Referencing Objects

- To reference an array of elements based on the value of the class attribute, use

  object.getElementsByClassName(*class*)

  where *class* is the class attribute value.

- To reference a document object based on the value of its id attribute, use

  document.getElementById(*id*)

  where *id* is the id value.

- To reference an array of elements based on the value of the name attribute, use

  document.getElementsByName(*name*)

  where *name* is the value of the name attribute.

Writing HTML Content

- The content within an HTML element is also part of the object tree
  - innerHTML property
  - outerHTML property

Figure 13-7 Creating the init() function

displays the first puzzle's title, hint, and difficulty rating

uses the drawGrid() function to generate the HTML code for the first puzzle's grid

```
function init() {
   /* Write the first puzzle data into the Web page */
   document.getElementsByTagName("h1")[0].innerHTML = "Hanjie Puzzle 1";
   document.getElementById("hint").innerHTML = puzzle1Hint;
   document.getElementById("rating").innerHTML = puzzle1Rating;

   /* Display the first puzzle using the drawGrid() function  */
   document.getElementById("puzzle").innerHTML = drawGrid(puzzle1);
}
```

Writing Event Handlers as Object Properties

- Any document object can be assigned an event handler from within your JavaScript program using the expression

```
object.onevent = function;
```

- To run a function when the page is initially loaded by the browser, use the statement

```
window.onload = function;
```

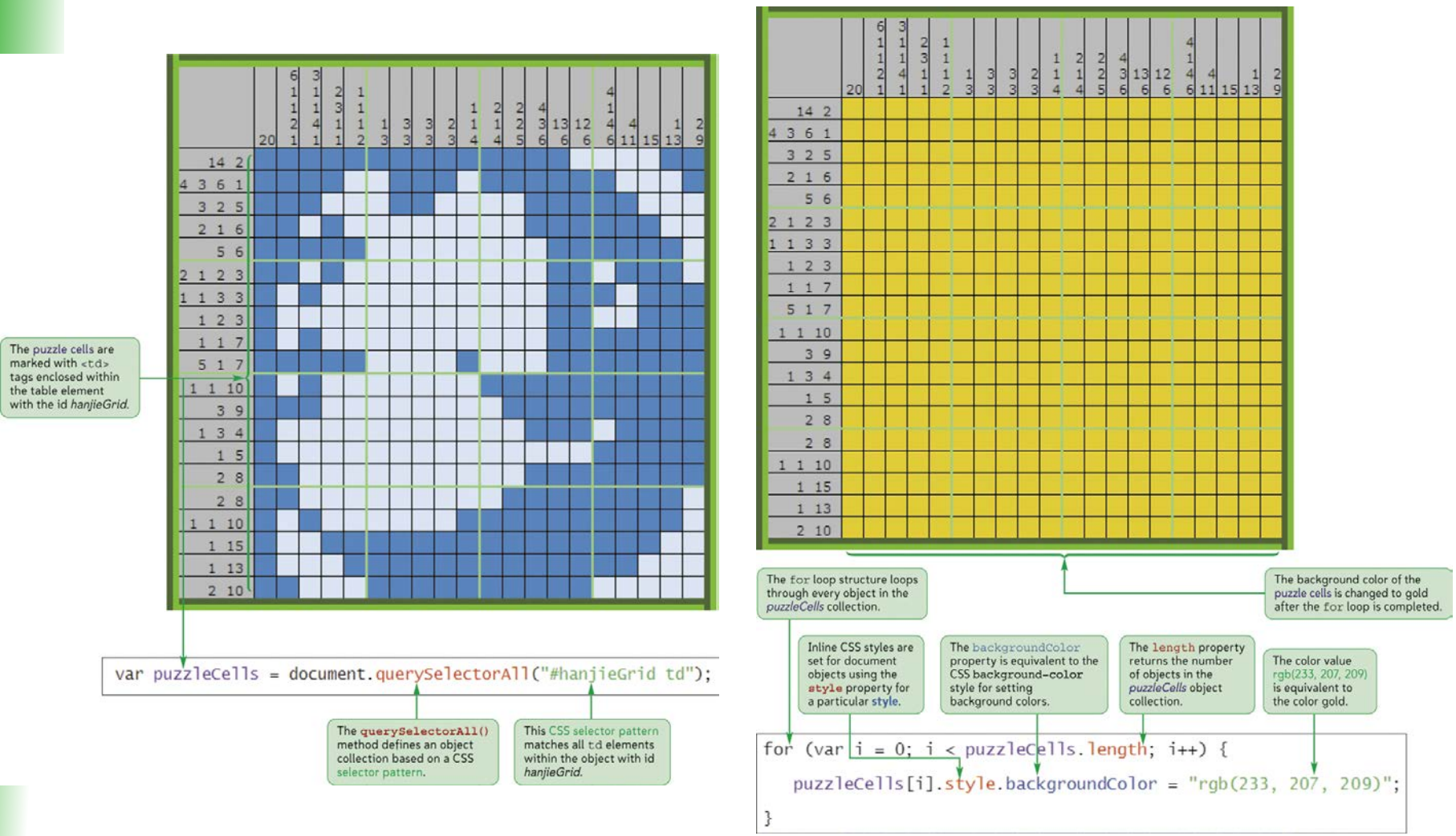
Writing Event Handlers as Object Properties

- To run a function in response to a mouse click, use

```
object.onclick = function;
```

- To reference the object that initiated the event, use the this keyword
- The **this keyword** is a special JavaScript keyword that refers to the object that calls or owns a particular function or method
  - `this.id`
  - `this.value`

JavaScript and CSS



Setting Inline Styles with JavaScript

- To apply an inline style to a document object, use

  ```
  object.style.property = text
  ```

- To retrieve the integer value of an inline style property, use

  ```
  parseInt(object.style.property)
  ```

- To retrieve the numeric value of an inline style property, use the following:

  ```
  parseFloat(object.style.property)
  ```

Creating Object Collections Using CSS Selections

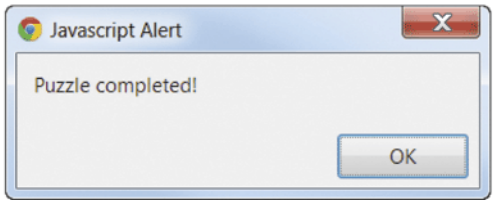
- To create an object collection based on a CSS selector pattern, use
  - `document.querySelectorAll(`*`selector`*`)`
- To return only the first object based on a CSS selector pattern, use the following:
  - `document.querySelector(`*`selector`*`)`

Displaying Browser Window Dialog Boxes

- JavaScript supports three types of dialog boxes
  - Alert
  - Confirm
  - Prompt
- The alert dialog box, which you've already seen, is created using the method

```
alert(message)
```

Figure 13-29 Browser window alert dialog box



Displaying Browser Window Dialog Boxes

- When an **alert dialog box** is being displayed, the execution of the program code halts until the user clicks the *OK* button
- The **confirm dialog box** prompts the user for a yes or no response

```
confirm(message)
```

- If you need a text string returned instead of a Boolean value, you can display a **prompt dialog box** by using the method

```
prompt(message, default)
```