JavaScript 3

Working with the Document Object Model (DOM) and DHTML

Objectives

When you complete this lesson, you will be able to:

- Access elements by id, tag name, class, name, or selector
- Access element content, CSS properties, and attributes
- Add and remove document nodes
- Create and close new browser tabs and windows with an app

Objectives (cont'd.)

When you complete this lesson, you will be able to:

- Use the `setTimeout()` and `setInterval()` methods to specify a delay or a duration
- Use the `History`, `Location`, `Navigation`, and `Screen` objects to manipulate the browser window

Understanding the Browser Object Model and the Document Object Model

- JavaScript treats web page content as set of related components
  - objects
- Every element on web page is an object
- You can also create objects
  - a function is an object

Understanding the Browser Object Model

- Browser object model (BOM) or client-side object model
  - Hierarchy of objects
  - Each provides programmatic access
    - To a different aspect of the web browser window or the web page
- `Window` object
  - Represents a Web browser window
  - Called the global object
    - Because all other BOM objects contained within it

Understanding the Browser Object Model (cont′d.)

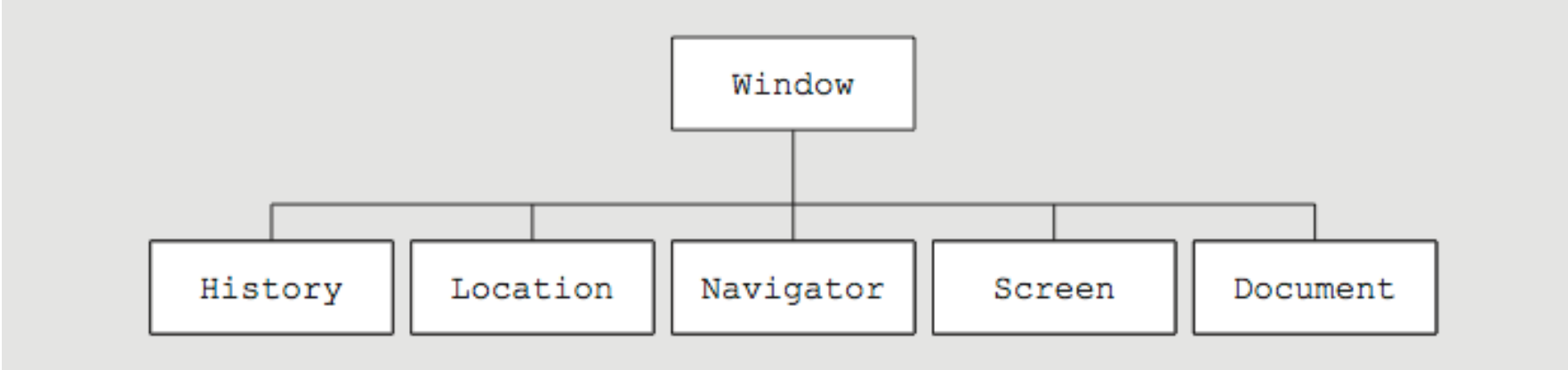

Figure: Browser object model

The Document Object Model

- `Document` object
  - Represents the Web page displayed in a browser
  - Contains all Web page elements
  - JavaScript represents each element by its own object

The DOM and DHTML

- Dynamic HTML (DHTML)
  - Interaction can change content of web page without reloading
  - Can also change presentation of content
  - Combination of HTML, CSS, and JavaScript
- DOM
  - example of an application programming interface (API)
  - structure of objects with set of properties and methods

The DOM tree

- The DOM hierarchy depends on a document's contents

```
1   <html lang="en">
2      <head>
3         <meta charset="utf-8" />
4         <title>Photo Gallery</title>
5      </head>
6      <body>
7         <header>
8            <h1>Garden Photo</h1>
9         </header>
10        <article>
11           <figure>
12              <figcaption>Butterfly bush</figcaption>
13              <img src="image016.jpg" alt="Butterfly bush" />
14           </figure>
15        </article>
16     </body>
17  </html>
```

The DOM tree (cont'd.)

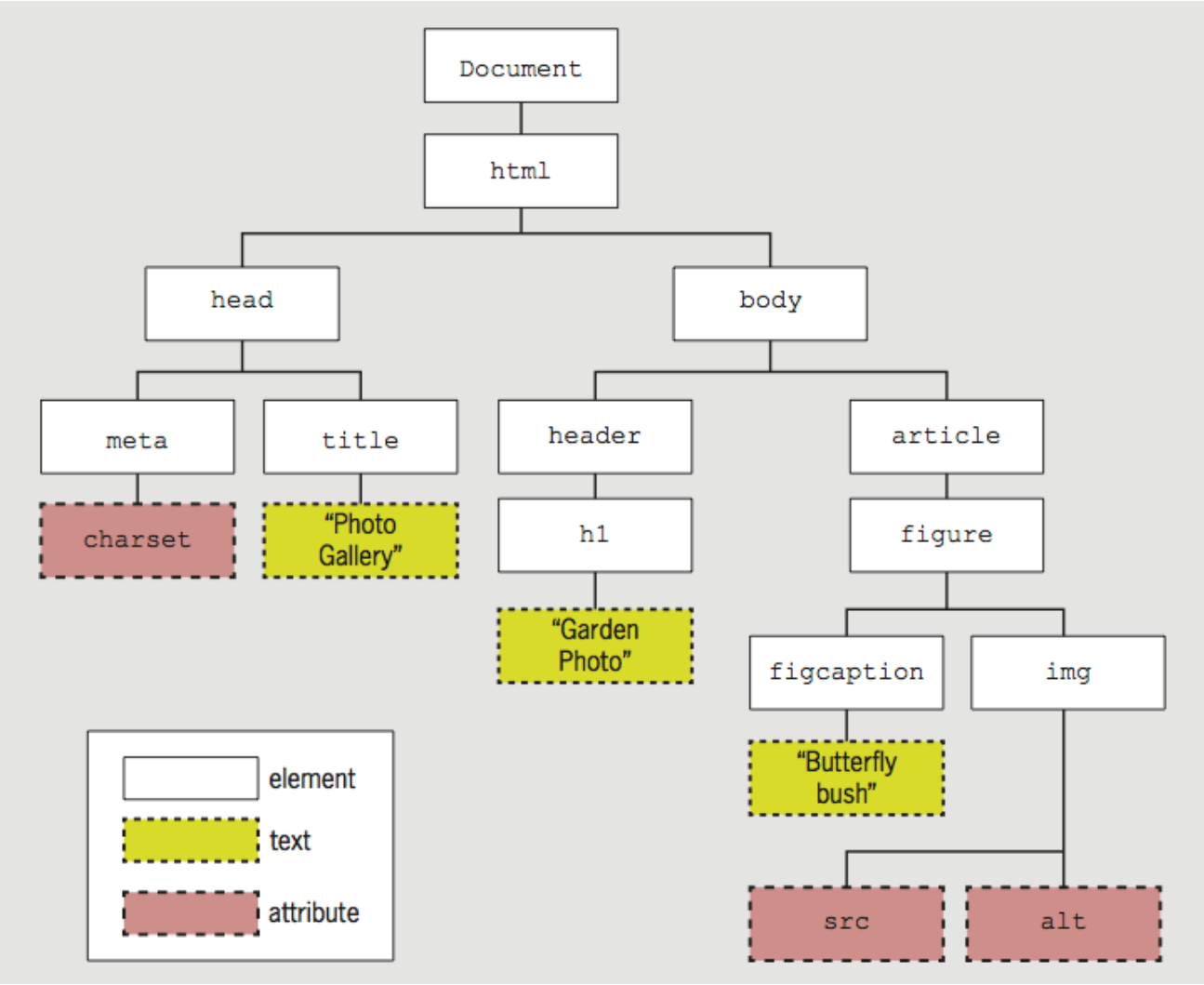

Figure Example DOM tree

The DOM tree

- Each item in the DOM tree is a node
- Element, attribute, and text content nodes are most commonly used

DOM `Document` Object Methods

METHOD	DESCRIPTION
`getElementById(ID)`	Returns the element with the `id` value *ID*
`getElementsByClassName(class1 [class2 ...])`	If one class name, `class1`, is specified, returns the collection of elements that belong to `class1`; if two or more space-separated class names are specified, the returned collection consists of those elements that belong to all specified class names
`getElementsByName(name)`	Returns the collection of elements with the name *name*
`getElementsByTagName(tag)`	Returns the collection of elements with the tag (element) name *tag*
`querySelectorAll(selector)`	Returns the collection of elements that match the CSS selector specified by *selector*
`write(text)`	Writes *text* to the document

Table HTML DOM `Document` object methods

DOM `Document` Object Properties

PROPERTY	DESCRIPTION
`body`	Document's `body` element
`cookie`	Current document's cookie string, which contains small pieces of information about a user that are stored by a web server in text files on the user's computer
`domain`	Domain name of the server where the current document is located
`lastModified`	Date the document was last modified
`location`	Location of the current document, including its URL
`referrer`	URL of the document that provided a link to the current document
`title`	Title of the document as specified by the `title` element in the document head section
`URL`	URL of the current document

Table Selected DOM `Document` object properties

Accessing Document Elements, Content, Properties, and Attributes

- Methods such as `getElementById()` are methods of the `Document` object
- Several methods available for JavaScript to reference web page elements

Accessing Elements by `id` value

- Set the `id` value in HTML
- `getElementById()` method
  - Returns the first element in a document with a matching `id` attribute
- Example:

HTML element with id value

```
<input type="number" id="zip" />
```

JavaScript to reference HTML element

```
var zipField = document.getElementById("zip");
```

Accessing Elements by Tag Name

- `getElementsByTagName()` method
  - Returns array of elements matching a specified tag name
  - Tag name is name of element
- Method returns a set of elements
  - Node list is indexed collection of nodes
  - HTML collection is indexed collection of HTML elements
  - Either set uses array syntax

Accessing Elements by Tag Name (cont'd.)

- Example:
  - Index numbers start at 0, so second element uses index number 1
  - To work with the *second* h1 element on a page:

```
var secondH1 = document.getElementsByTagName("h1")[1];
```

Accessing Elements by Class Name

- `getElementsByClassName()` method
  - Returns node list or HTML collection of elements with a `class` attribute matching a specified value
- Example
  - All elements with `class` value `side`:

```
var sideElements = document.getElementsByClassName("side");
```

Accessing Elements by Class Name (cont'd.)

- `class` attribute takes multiple values, so `getElementsByClassName()` method takes multiple arguments
- Arguments enclosed in single set of quotes, with class names separated by spaces
- Example
  - All elements with `class` values `side` and `green`:

```
var sideGreenElements = document.getElementsByClassName("side green");
```

Accessing Elements by Name

- `getElementsByName()` method
  - Returns node list or HTML collection of elements with a `name` attribute matching a specified value
- Not as useful as preceding options
  - But creates more concise code when accessing set of option buttons or check boxes in a form:

```
var colorButtons = document.getElementsByName("color");
```

- Not standard in IE9 and earlier versions of IE, so important to test

Accessing Elements with CSS Selectors

- `querySelector()` method
  - References elements using CSS syntax
  - Returns first occurrence of element matching a CSS selector
- Example:

HTML:

```
<header>
   <h1><img class="logo" src="images/logo.png"↲
      alt="Blue Jay Photography" /></h1>

</header>
```

JavaScript to reference `img` element

```
querySelector("header h1 img")
```

Accessing Elements with CSS Selectors (cont'd)

- IE8 supports only simple selectors
  - Can use different approach in creating CSS selector
  - Previous example could be rewritten as

```
querySelector("img.logo")
```

Accessing Elements with CSS Selectors (cont'd)

- `querySelectorAll()` method
  - Returns collection of elements matching selector
  - Different from `querySelector()` method, which returns only first occurrence
  - Example:

HTML:

```
<nav>
  <ul>
    <li>About Us</li>
    <li>Order</li>
    <li>Support</li>
  </ul>

</nav>
```

JavaScript to reference all three `li` elements:

```
querySelectorAll("nav ul li")
```

Accessing an Element's Content

- `textContent` property
  - Accesses and changes text that an element contains
  - Unlike `innerHTML`, `textContent` strips out HTML tags
- Example:

HTML:

```
<ul>
  <li class="topnav"><a href="aboutus.htm">About Us</a></li>
  <li class="topnav"><a href="order.htm">Order</a></li>
  <li class="topnav"><a href="support.htm">Support</a></li>

</ul>
```

JavaScript to reference and access first `li` element:

```
var button1 = querySelectorAll("li.topNav")[0];
var allContent = button1.innerHTML;
  // <a href="aboutus.htm">About Us</a>
var justText = button1.textContent;

  // About Us
```

Accessing an Element's Content (cont'd)

- `textContent` property is more secure
  - Not supported by IE8 or earlier
  - Some developers use `if/else` construction to implement `textContent` only on supported browsers

Accessing an Element's CSS Properties

- Can access CSS properties through DOM
  - Use dot notation
  - Reference element's style property followed by name of CSS property
  - Example: change value of CSS `display` property to `none` for element with `id` value `logo`:

```
document.getElementById("logo").style.display = "none";
```

Accessing an Element's CSS Properties (cont'd.)

- When CSS property includes hyphen (-), remove hyphen and capitalize letter following hyphen
  - Use dot notation
  - `font-family` becomes `fontFamily`
  - Example:

```
var font = document.getElementById("logo").style.fontFamily;
```

- CSS value specified using DOM reference is an inline style
  - Higher priority than embedded or external styles

Accessing an Element's CSS Properties (cont'd.)

- To remove a style you previously added with a DOM reference
  - set its value to an empty string
  - Example:

```
document.getElementById("navbar").style.color = "";
```

Accessing Element Attributes

- Access element attribute with period and name of attribute after element reference
  - Reference element with `id` value `homeLink`:

```
document.getElementById("homeLink")
```

  - Reference `href` attribute of same element:

```
document.getElementById("homeLink").href
```

- Can use to look up attribute value and assign to variable, or to assign new value to attribute

Accessing Element Attributes (cont'd)

- One exception for accessing element attributes
  - Must use property name `className` to refer to `class` attribute values
  - Single class value returned like standard attribute value
  - Multiple class values returned in single string, separated by spaces

Adding and Removing Document Nodes

- DOM includes methods to change DOM tree
  - Can create brand new elements
  - Can add/remove elements from DOM tree

Creating Nodes

- `createElement()` method
  - Creates a new element
  - Syntax:

```
document.createElement("element")
```

  - *`element`* is an element name

  - Example:
    - To create a new `div` element:

```
document.createElement("div");
```

Attaching Nodes

- Newly created node is independent of DOM tree
- `appendChild()` method
  - Attaches node to DOM tree
  - Syntax:

    parentNode.appendChild(*childNode*)

  - *`childNode`* is node to be attached
  - *`parentNode`* is node to attach child node to

Attaching Nodes (cont'd.)

- Example:
  - Create new `li` element and attach to element with `id` value `navList`:

```
var list = document.getElementById("navList");
var contact = document.createElement("li");
list.appendChild(contact);
```

- Document fragment
  - Set of connected nodes not part of document
  - Can use `appendChild()` to add document fragment to DOM tree for a document

Attaching Nodes (cont'd.)

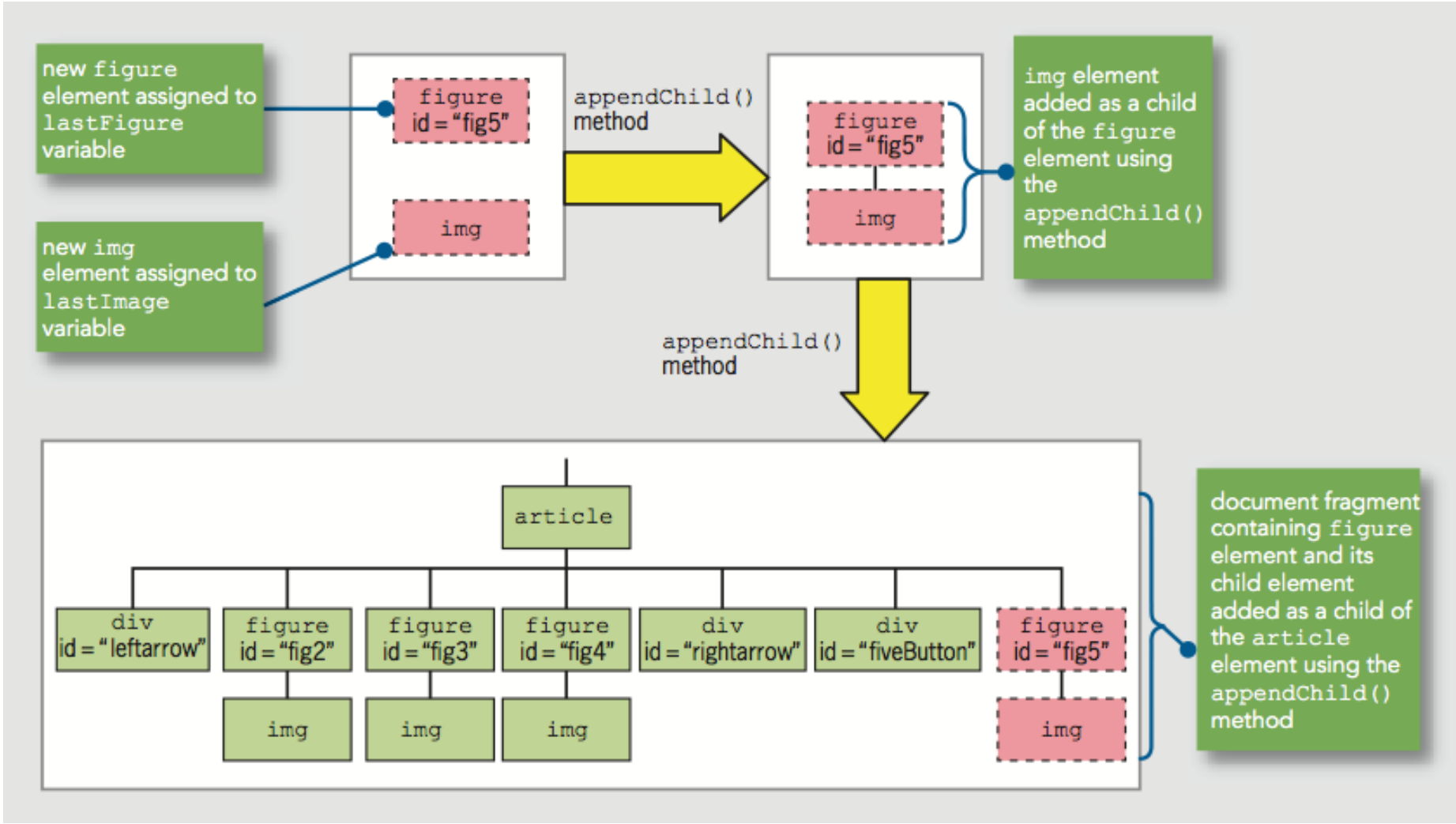

Table Using the `appendChild()` method to attach nodes

Cloning Nodes

- Create new node same as existing node
- `cloneNode()` method
- Syntax:
  - *existingNode*.cloneNode(true | false)
    - `true` argument clones child nodes
    - `false` argument clones only specified parent node
- Example:

```
document.createElement("div");
```

Cloning Nodes (cont'd.)

- Example:

```
var contact = document.createElement("li");
contact.className = "mainNav";
var directions = contact.cloneNode(true);
```

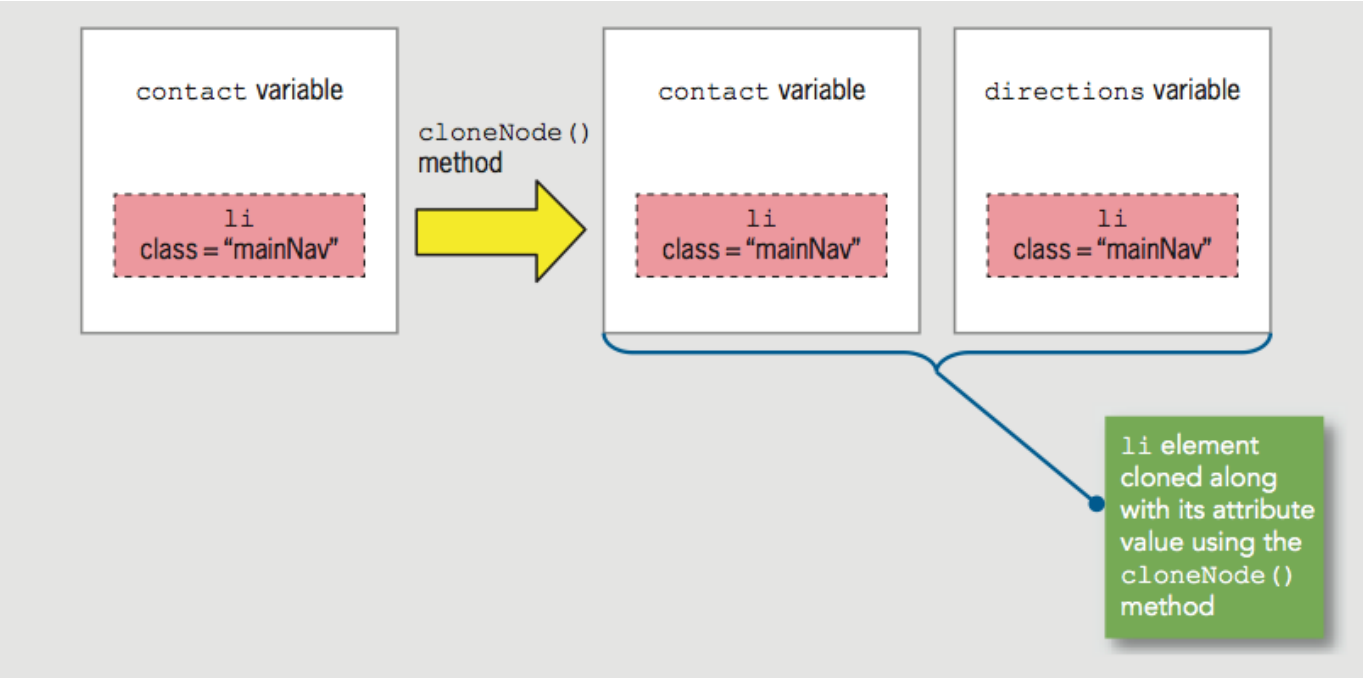

Figure Using the `cloneNode()` method

Inserting Nodes at Specific Positions in the Document Tree

- New node created with `createElement()` is not attached to document tree
- `appendChild()` adds node after existing child nodes
- To specify a different position, use `insertBefore()`
- Syntax:

  _ *parentNode*.insertBefore(*newChildNode*, *existingChildNode*)

Inserting Nodes at Specific Positions in the Document Tree (cont'd.)

- Example:
  - HTML:

```
<ul id="topnav">
  <li><a href="aboutus.htm">About Us</a></li>
  <li><a href="order.htm">Order</a></li>
  <li><a href="support.htm">Support</a></li>

</ul>
```

  - JavaScript:

```
var list = document.getElementById("topnav");
var directions = document.createElement("li");
directions.innerHTML = "Directions";
var aboutus = document.querySelectorAll("#topnav li")[0];
list.insertBefore(directions, aboutus);
```

Inserting Nodes at Specific Positions in the Document Tree (cont'd.)

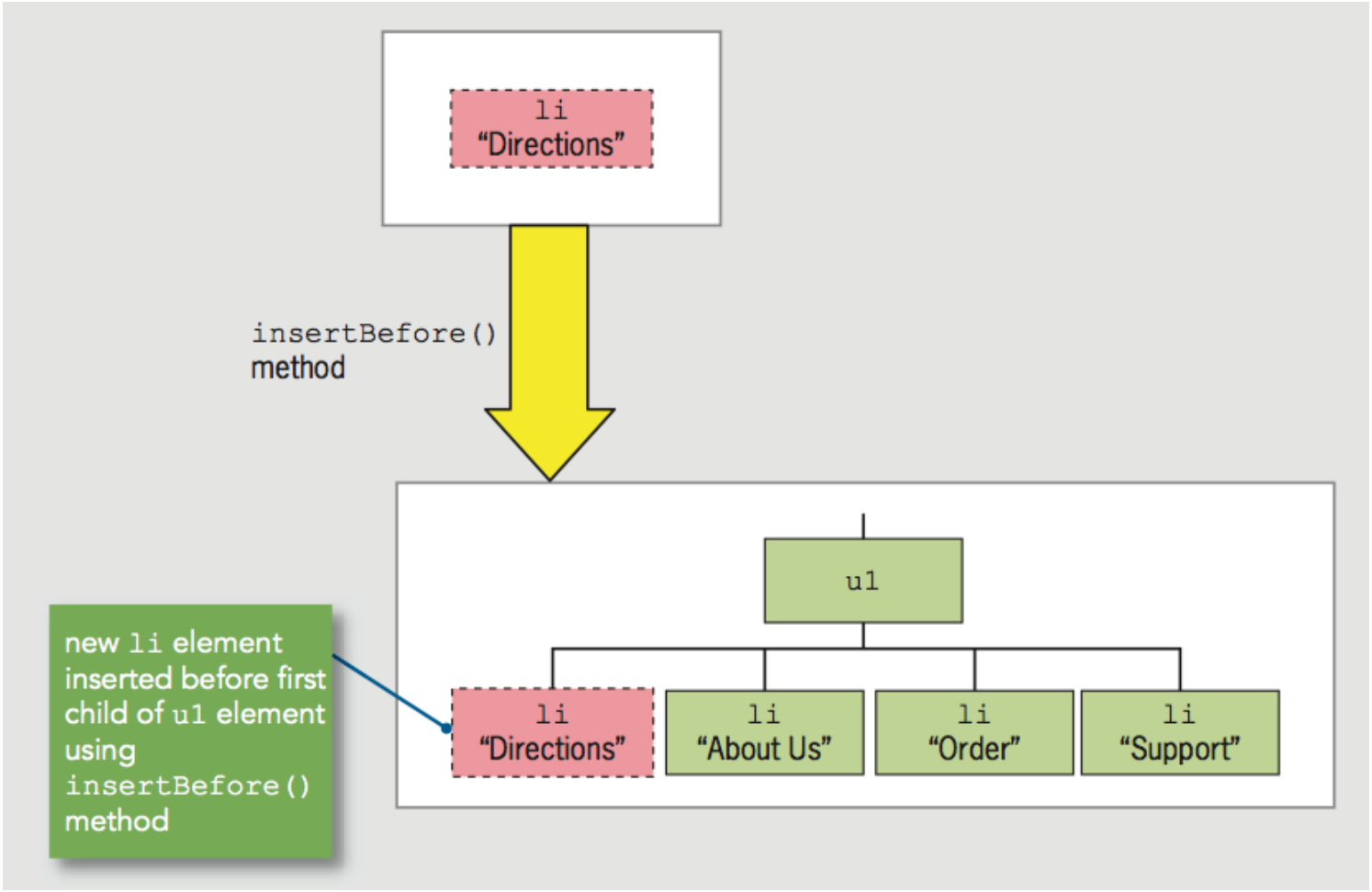

Figure Using the `insertBefore()` method

Removing Nodes

- `removeNode()` removes node from DOM tree
- Syntax:
  - *parentNode*.removeChild(*childNode*)
- Can assign removed node to variable:

```
var list = document.getElementById("topnav");

var aboutus = document.querySelectorAll("#topnav li")[0];

var aboutNode = list.removeChild(aboutus);
```

  - Node removed without being assigned to a variable is deleted during garbage collection

Manipulating the Browser with the `Window` Object

- `Window` object
  - Includes properties containing information about the web browser window or tab
  - Contains methods to manipulate the web browser window or tab itself

Manipulating the Browser with the Window Object (cont'd.)

PROPERTY	DESCRIPTION
`closed`	Boolean value that indicates whether a window or tab has been closed
`document`	Reference to the `Document` object
`history`	Reference to the `History` object
`innerHeight`	Height of the window area that displays content, including the scrollbar if present
`innerWidth`	Width of the window area that displays content, including the scrollbar if present
`location`	Reference to the `Location` object
`name`	Name of the window or tab
`navigator`	Reference to the `Navigator` object
`opener`	Reference to the window that opened the current window or tab
`outerHeight`	Height of the entire browser window
`outerWidth`	Width of the entire browser window
`screen`	Reference to the `Screen` object
`self`	Self-reference to the `Window` object; identical to the `window` property
`status`	Temporary text that is written to the status bar
`window`	Self-reference to the `Window` object; identical to the `self` property

Table `Window` object properties

Manipulating the Browser with the Window Object (cont'd.)

METHOD	DESCRIPTION
`alert()`	Displays a simple message dialog box with an OK button
`blur()`	Removes focus from a window or tab
`clearInterval()`	Cancels an interval that was set with `setInterval()`
`clearTimeout()`	Cancels a timeout that was set with `setTimeout()`
`close()`	Closes a web browser window or tab
`confirm()`	Displays a confirmation dialog box with OK and Cancel buttons
`focus()`	Makes a `Window` object the active window or tab
`moveBy()`	Moves the window relative to the current position
`moveTo()`	Moves the window to an absolute position
`open()`	Opens a new web browser window or tab
`print()`	Prints the document displayed in the current window or tab
`prompt()`	Displays a dialog box prompting a user to enter information

Table `Window` object methods *(continues)*

Manipulating the Browser with the Window Object (cont'd.)

METHOD	DESCRIPTION
`resizeBy()`	Resizes a window by a specified amount
`resizeTo()`	Resizes a window to a specified size
`scrollBy()`	Scrolls the window or tab by a specified amount
`scrollTo()`	Scrolls the window or tab to a specified position
`setInterval()`	Repeatedly executes a function after a specified number of milliseconds have elapsed
`setTimeout()`	Executes a function once after a specified number of milliseconds have elapsed

Table `Window` object methods

Manipulating the Browser with the `Window` Object (cont′d.)

- `self` property
  - Refers to the current `Window` object
  - Identical to using the `window` property to refer to the `Window` object
  - Examples:
    - window.close();
    - self.close();
- Web browser assumes reference to global object
- Good practice
  - Use `window` or `self` references
    - When referring to a `Window` object property or method

Opening and Closing Windows

- Reasons to open a new Web browser window
  - To launch a new Web page in a separate window
  - To use an additional window to display information
- When new Web browser window opened:
  - New `Window` object created
    - Represents the new window
- Know how to open a link in a new window using the a element's target attribute

```
<a href="http://www.wikipedia.org/"↲
  target="wikiWindow">Wikipedia home page</a>
```

Opening a Window

- `open()` method of the `Window` object
  - Opens new windows
- Syntax

  window.open(*url*, *name*, *options*, *replace*);

METHOD	DESCRIPTION
`resizeBy()`	Resizes a window by a specified amount
`resizeTo()`	Resizes a window to a specified size
`scrollBy()`	Scrolls the window or tab by a specified amount
`scrollTo()`	Scrolls the window or tab to a specified position
`setInterval()`	Repeatedly executes a function after a specified number of milliseconds have elapsed
`setTimeout()`	Executes a function once after a specified number of milliseconds have elapsed

Table Arguments of the `Window` object's `open()` method

Opening a Window (cont′d.)

- Include all (or none) `window.open( )` method arguments
- Example:
  - `window.open("http://www.wikipedia.org");`

Opening a Window (cont'd.)

- Customize new browser window or tab appearance
  - Use `window.open()` method `options` argument

NAME	DESCRIPTION
`height`	Sets the window's height
`left`	Sets the horizontal coordinate of the left of the window, in pixels
`location`	Includes the URL Location text box
`menubar`	Includes the menu bar
`personalbar`	Includes the bookmarks bar (or other user-customizable bar)
`resizable`	Determines if the new window can be resized
`scrollbars`	Includes scroll bars
`status`	Includes the status bar
`toolbar`	Includes the Standard toolbar
`top`	Sets the vertical coordinate of the top of the window, in pixels
`width`	Sets the window's width

Table Common options of the `Window` object's `open()` method

Opening a Window (cont′d.)

- `window.open()` method `name` argument
  - Same as value assigned to the `target` attribute
    - Specifies window name where the URL should open
  - If `name` argument already in use
    - JavaScript changes focus to the existing Web browser window instead of creating a new window

Opening a Window (cont′d.)

- `Window` object′s `name` property used to specify a target window with a link
  - Cannot be used in JavaScript code
- Assign the new `Window` object created with the `window.open()` method to a variable to control it
- `focus()` method
  - Makes a window the active window

Closing a Window

- `close()` method
  - Closes a web browser window
- `window.close()` or `self.close()`
  - Closes the current window

Working with Timeouts and Intervals

- `Window` object's timeout and interval methods
  - Creates code that executes automatically
- `setTimeout()` method
  - Executes code after a specific amount of time
  - Executes only once
  - Syntax
    - `var variable = setTimeout("code", milliseconds);`
- `clearTimeout()` method
  - Cancel `setTimeout()` before its code executes
- Example on next slide

Working with Timeouts and Intervals (cont′d.)

```
var buttonNotPressed = setTimeout("window.alert('Your↲
   changes have been saved')",10000);
function buttonPressed() {
   clearTimeout(buttonNotPressed);
   window.open(index.htm);
}
```

Working with Timeouts and Intervals (cont′d.)

- `setInterval()` method
  - Repeatedly executes the same code after being called only once
  - Syntax:
    - var *variable* = setInterval("*code*", *milliseconds*);
- `clearInterval()` method
  - Used to clear `setInterval()` method call

The `History` Object

- `History` object
  - Maintains internal list (history list)
    - All documents opened during current web browser session
- Security features
  - Will not display URLs contained in the history list

The `History` Object (cont'd.)

METHOD	DESCRIPTION
`back()`	Produces the same result as clicking a browser's Back button
`forward()`	Produces the same result as clicking a browser's Forward button
`go()`	Opens a specific document in the history list

Table Methods of the `History` object

The `History` Object (cont'd.)

- `go()` method
  - Allows navigation to a specific previously visited web page
- `History` object `length` property
  - Provides specific number of documents opened during the current browser session
  - Example:
    - Return to first document opened in current browser session:

```
history.go(-(history.length - 1));
```

The `Location` Object

- `Location` object
  - Allows changes to a new web page from within JavaScript code
- `Location` object properties allow modification of URL individual portions
  - Web browser automatically attempts to open that new URL

The Location Object (cont'd.)

PROPERTIES	DESCRIPTION
hash	URL's anchor
host	Host and domain name (or IP address) of a network host
hostname	Combination of the URL's host name and port sections
href	Full URL address
pathname	URL's path
port	URL's port
protocol	URL's protocol
search	URL's search or query portion

Table Properties of the Location object

METHOD	DESCRIPTION
assign()	Loads a new web page
reload()	Causes the page that currently appears in the web browser to open again
replace()	Replaces the currently loaded URL with a different one

Table Methods of the Location object

The `Location` Object (cont′d.)

- `Location` object′s `assign()` method
  - Same action as changing the `href` property
  - Loads a new web page
- `Location` object′s `reload()` method
  - Equivalent to the browser Reload or Refresh button
  - Causes current page to open again
- `Location` object′s `replace()` method
  - Replaces currently loaded URL with a different one

The `Navigator` Object

- `Navigator` object
  - Obtains information about current web browser
  - Example: determine type of web browser running

PROPERTIES	DESCRIPTION
`appName`	Name of the web browser displaying the page
`appVersion`	Version of the web browser displaying the page
`geolocation`	API for accessing the user's current location and user permission settings denying or allowing access to that information
`onLine`	Whether the browser currently has a network connection
`platform`	Operating system in use on the client computer
`userAgent`	String stored in the HTTP user-agent request header, which contains information about the browser, the platform name, and compatibility

Table Properties of the `Navigator` object

The `Navigator` Object (cont'd.)

```
console.log("Web browser name: " + navigator.appName);
console.log("Web browser version: " + navigator.appVersion);
console.log("Operating platform: " + navigator.platform);

console.log("User agent: " + navigator.userAgent);
```

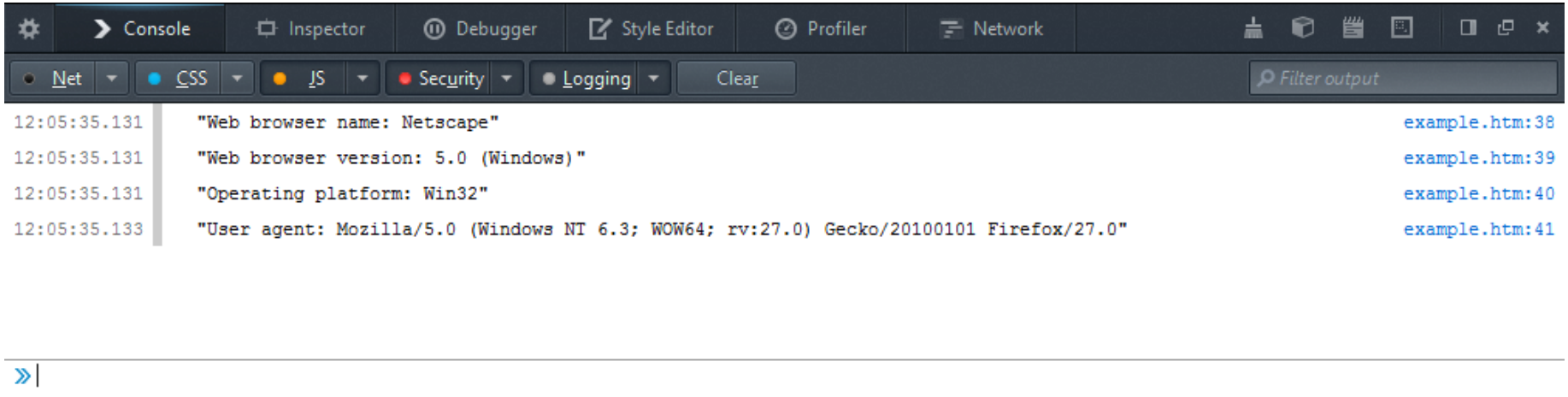

Figure `Navigator` object properties in Firefox console

The `Screen` Object

- `Screen` object
  - Obtains information about display screen's size, resolution, color depth
- Common use of `Screen` object properties
  - Centering a web browser window in the middle of the display area

The Screen Object (cont'd.)

PROPERTIES	DESCRIPTION
availHeight	Height of the display screen, not including operating system features such as the Windows taskbar
availWidth	Width of the display screen, not including operating system features such as the Windows taskbar
colorDepth	Display screen's bit depth if a color palette is in use; if a color palette is not in use, returns the value of the pixelDepth property
height	Height of the display screen
pixelDepth	Display screen's color resolution in bits per pixel
width	Width of the display screen

Table Properties of the Screen object

The Screen Object (cont'd.)

- Common Screen object properties uses
  - Center a web browser window
  - Example:

```
var winWidth = 300;
var winHeight = 200;
var leftPosition = (screen.width - winWidth) / 2;
var topPosition = (screen.height - winHeight) / 2;
var optionString = "width=" + winWidth + ",height="↲
   + winHeight + ",left=" + leftPosition + ",top="↲
   + topPosition;

var openWin = window.open("", "CtrlWindow", optionString);
```

Summary

- Browser object model (BOM) or client-side object model
  - Hierarchy of objects
- Top-level object in the browser object model
  - `Window` object
- `Document` object: most important object
- DOM represents web page displayed in window

Summary (cont'd.)

- Access elements with `getElementById()`, `getElementsByTagName()`, `getElementsByClassName()`, `getElementsByName`, `querySelector()`, or `querySelectorAll()`
- Access element content with `textContent()` or `innerHTML()` property
- Access CSS properties using an element's JavaScript `style` property

Summary (cont'd.)

- Create new node with `createElement()`
- Attach node with `appendChild()`
- Clone node with `cloneNode()`
- Attach node in specific place with `insertBefore()`
- Remove node with `removeNode()`
- Open new window with `window.open()`
- Close window with `window.close()`

Summary (cont'd.)

- `setTimeout()` executes code after specific amount of time
- `clearTimeout()` cancels `setTimeout()`
- `setInterval()` executes code repeatedly
- `clearInterval()` cancels `setInterval()`

Summary (cont′d.)

- `History` object maintains an opened documents history list
- `Location` object allows changes to a new web page from within JavaScript code
- `Navigator` object obtains information about the current web browser
- `Screen` object obtains information about the display screen′s size, resolution, color depth