# Chapter 1: The ABC of Programming

-A script is a series of instructions that a computer can follow to achieve a goal.

-To write a script, you need to 1st state your goal and then list the tasks that need to be completed in order to achieve it:  
+Define the goal.

+Design the script.

+Code each step.

-Designing a script: tasks: The high-level view of tasks can be presented using a flowchart.

-Designing a script: steps: When you ready to code the script, steps can then be translated into lines of code.

-The document object represents an HTML page. Using document object, we can access and change what content users see on page and respond. Like other objects, document has: properties + methods + events

-How a browser sees a web page

+The browser receives an HTML page

+It creates a model of page and store it in memory.

+It shows the page on screen using a rendering engine (may use CSS)

-How HTML, CSS & JavaScript fit together

+HTML (Content Layer): give the page structure and adds semantics

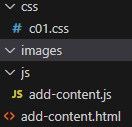
+CSS (Presentation Layer): enhances the HTML page with rules that state how HTML content is presented

+JS (Behavior Layer): where we change how the page behaves, adding interactivity.

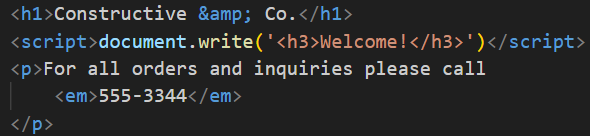
-Progressive enhancement: 3 layers form the basis of this popular approach to building web page: Start with HTML, then add CSS rules, JS is added last.

-Sample code: www.javascriptbook.com

-Creating a basic JS:



-Linking to a JS file from HTML page: Use <script src>  
-Placing the script in the page:



-JS runs where it is found in the HTML: When browser comes across a <script>, it stops to load the script and then checks to see if it needs to do anything.

# Chapter 2: Basic JavaScript Instructions

-Statements: each individual instruction or step, should end with a semicolon.

-Comment: //, /\* \*/

-Variable: store data

-Variables: how to declare them

+variable keyword + variable name: var quantity;

-Varibles: how to assign them a value

+variable name + assignment operator + variable value: quantity = 3;

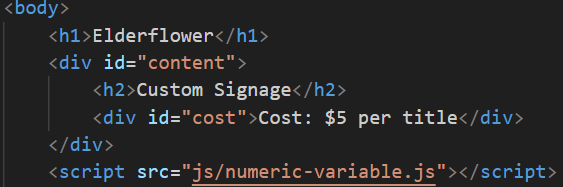
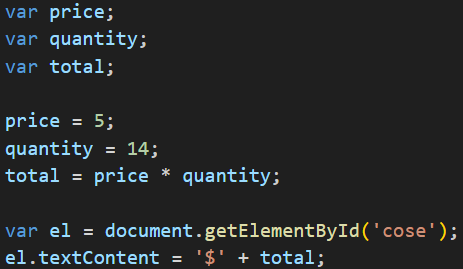
-Data types

+Numeric data type

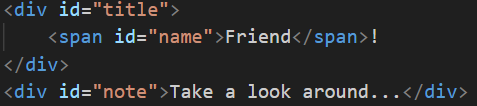
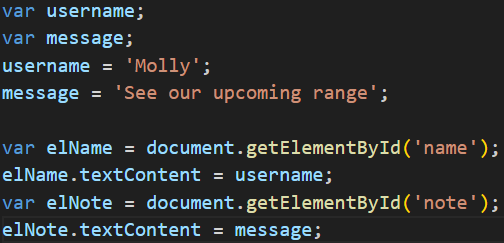
+String data types: can be single or double quotes

+Boolean data type

-Using a variable to store a number



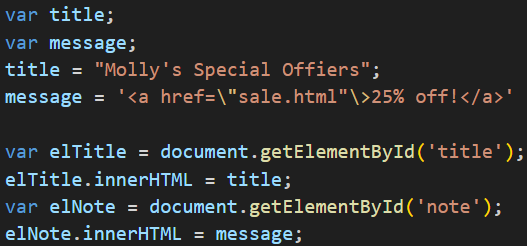
-Using a variable to store a string



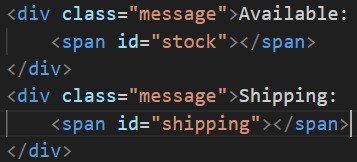
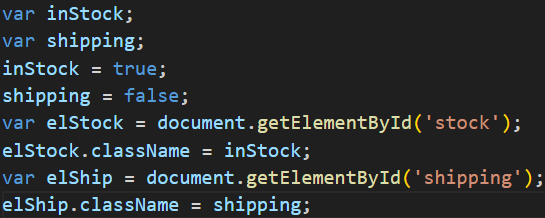
-Using quotes inside a string

+” ‘’ ” or ‘ “ “ ’

+Escaping the quotation characters: use backward slash before any type of quote mark that appears with a string ->the following character is part of string.

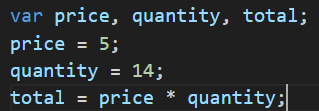
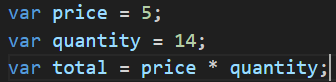


-Using a variable to store a Boolean



+The value are used in class attributes of HTML elements. These values trigger CSS class rules: true show a check, false show a cross.

-Shorthand for creating variable

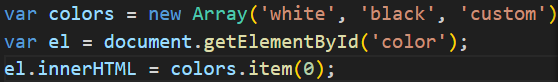
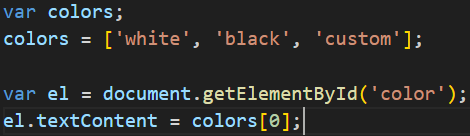




-Changing the value of a variable

-Rules for naming variable: Same in Java

-Arrays: store a list of values



+Number of items: colors.length;

-Expressions: evaluates into a single value. There are 2:

+assign a value to a variable

+use 2 or more values to return a single value

-Operators

+Assignment

+Arithmetic operators

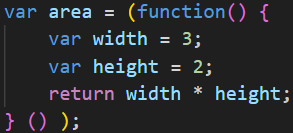
+String operators

+Comparison operators

+Logical operators

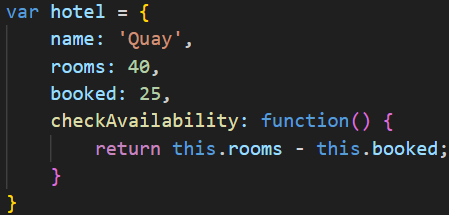
# Chapter 3: Functions, Methods & Objects

-Immediately invoked function expressions:

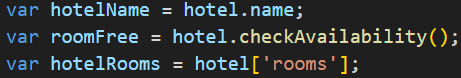


-Variable scope: local variable + global variable

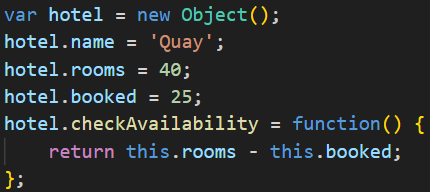
-Creating object:



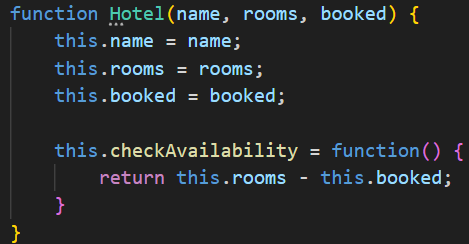
+Access object and dot notation:



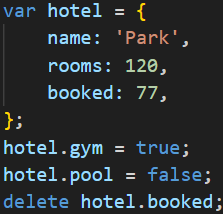
+Creating object: constructor notation



+Create many objects: constructor notation



+Add and remove properties:



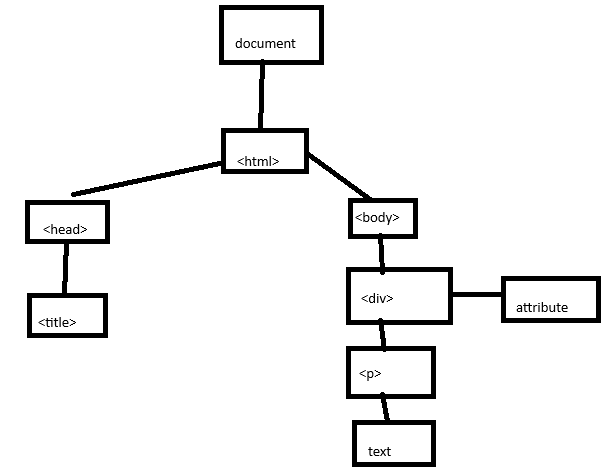
-Arrays are objects: They hold a related set of key/value pairs: costs = [420, 460, 230, 620];

-3 groups of built-in objects:

+Browser object model:

Window (current browser window or tab)-> document (current web page) + history (pages in browser history) + location (url of current page) + navigator (information about browser) + screen (device’s display information)

+Document object model:



-Global JS objects:

+String: for working with string values

+Number: For working with numeric values

+Boolean: for working with Boolean values

+Date: represent and handle dates

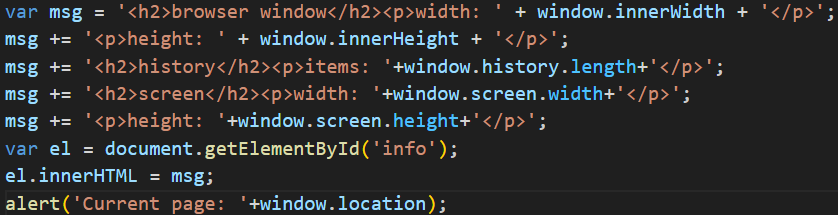
+Math: work with numbers and calculations

+Regex: match patterns within strings of text

-Browser object model: The window object

|  |  |
| --- | --- |
| Property | Description |
| window.innerHeight | Height of window |
| window.innerWidth | Width of window |
| window.pageXOffset | Distance document has been scrolled horizontally |
| window.pageYOffset | Distance document has been scrolled vertically |
| window.screenX | X-coordinate of pointer, relative to top left corner of screen |
| window.screenY | Y-coordinate of pointer, relative to top left corner of screen |
| window.location | Current URL of window object (or local file path) |
| window.document | Reference to document object, which is used to represent the current page contained in window |
| window.history | Reference to history object for browser window or tab, which contains details of pages that have been viewed in that window or tab |
| window.history.length | Number of items in history object for browser window or tab |
| window.screen | Reference to screen object |
| window.screen.width | Access screen object and find value of its width property |
| window.screen.height | Access screen object and find value of its height property |

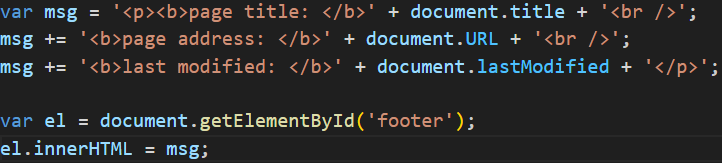
|  |  |
| --- | --- |
| Method | Description |
| window.alert() | Create dialog box with message (user must click OK button to close it) |
| window.open() | Open new browser window with URL specified as parameter |
| window.print() | Tell browser that user wants to print contents of current page |



-The document object model (**DOM**): The **document** object

|  |  |
| --- | --- |
| **Property** | **Description** |
| document.title | Title of current document |
| document.lastModified | Date on which document was last modified |
| document.URL | Return string containing URL of current document |
| document.domain | Return domain of current document |

|  |  |
| --- | --- |
| **Method** | **Description** |
| document.write() | Write text to document |
| document.getElementById() | Return element, if there is an element with value of id attribute that matches |
| document.querySelectorAll() | Return list of element that match a CSS selector, which is specified as a parameter |
| document.createElement() | Create new element |
| document.createTextNode() | Create new text node |



-Global objects: string object

+Property: length

+methods: toUpperCase(), toLowerCase(), charAt(), indexOf(), lastIndexOf(), subString(), split(), trim(), replace()

-Data types revisited:

+5 primitive data types: String, Number, Boolean, Undefined, Null

+Complex data type: Object, Arrays, Functions

-Global object: Number Object

|  |  |
| --- | --- |
| **Method** | **Description** |
| isNaN() | Check if the valiue is not number |
| toFixed() | Rounds to specified number of decimal places (return string) |
| toPrecision() | Rounds to total number of places (return string) |
| toExpotential() | Return string representing the number in exponential notation |

-Global objects: Math object

+Property: Math.PI

+Methods: Math.round(), Math.sqrt(), Math.ceil(), Math.floor(), Math.random()

-Global object: Date object

+var today = new Date();

+getDate()/setDate(), getDay() 0-6, getFullYear()/setFullYear(), getHours()/setHours() 0-23, getMilliseconds()/setMilliseconds() 0-999, getMinutes()/setMinutes() 0-59, getMonth()/setMonth() 0-11, getSeconds()/setSeconds() 0-59, getTime()/setTime(), getTimeZoneOffset(), toDateString(), toTimeString(), toString()

+Use format: YYYY, MM, DD, HH, MM, SS or

MMM (Mon/Tue) DD, YYYY HH:MM:SS

# Chapter 4: Decisions & Loops

-Evaluating conditions & conditional statement: if else

-Comparison operators: == != === !== > >= < <=

-Logical operators: && || !  
-Loops: for, while, do while

# Chapter 5: Document Object Model

-Document object model (DOM) specifies how browsers should create a model of HTML page and how JS can access and update content of web page while it is in the browser window.

-DOM is neither part of HTML or JS, it’s separate set of rules. It’s implemented by all major browser makers, ane covers 2 primary areas:

+Make a model of HTML page: When browser loads a web page, it creates a model of page in memory. DOM specifies the way in which browser should structure this model using DOM tree.

+Access and change HTML page: DOM defines methods and properties to access and update each object in this model.

-DOM is an API: user interfaces let human interact with programs, APIs let programs (and scripts) talk to each other.

-DOM tree is a model of a web page

+As browser loads a web page, it creates a model of page. The model is DOM tree, and it’s stored in browser’s memory. It’s consists of 4 main types of nodes

+Document node: represent the entire page

+Element nodes: relationships between document and all element nodes are: parent, children, siblings, ancestors and descendants.

+Attribute nodes: are not children of element that carries them, they are part of that element.

+Text nodes: Can’t have children, if element contains text and another child element, child element isn’t a child of text node but rather a child of containing element.

-**Working with DOM tree**

+**Step** **1**: **Access the elements**

Select an **individual element node**: getElementById() + querySelector(css selector) , traverse from one element to another within DOM tree

Select **multiple elements** (nodelists): getElementsByClassName(class), getElementByTagName(tag name), querySelectorAll(css selector)

**Traverse** between element nodes: parentNode, previousSibling/nextSibling, firstChild/lastChild

+**Step 2: Work with those elements**

Access/update **text nodes**: firstChild, nodeValue

Work with **HTML content**: innerHTML, textContent, createElement(), createTextNode(), appendChild()/removeChild()

Access or update **attribute values**: className/id, hasAttribute(), getAttribute(), setAttribute(), removeAttribute()

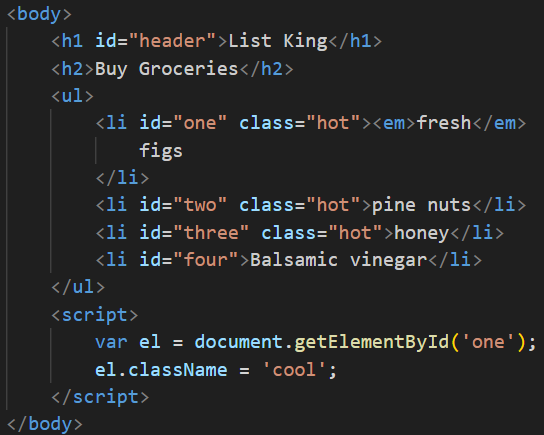
**-Caching DOM queries**

+DOM queries: methods that find elements in DOM tree

+When a script select an element to access/update, then interpreter must find the element(s) in Dom tree.

+variable doesn’t store the element, it stores a reference to where that node is in DOM tree.

-Select element using ID attributes:



+Use a property className to update value of class attribute of element stored in el variable.

-Nodelists: DOM queries that return more than one element

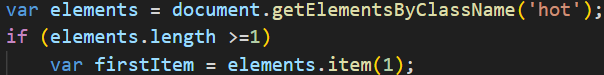
+getElementsByTagName(‘h1)

+getElementsByClassName(‘hot’)

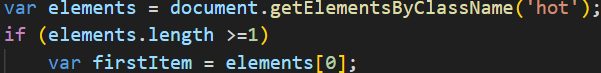
+querySelectorAll(‘li[id]’): return elements <li> having [id] attribute

-**Select** an **element** from a **nodelist**:

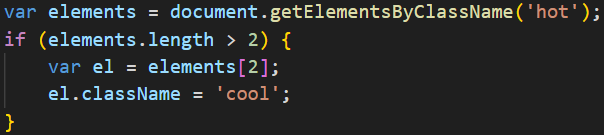
+**item()**



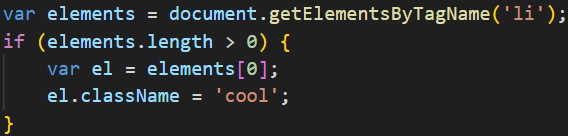
+**Array syntax**:



-Select elements using **class attributes**



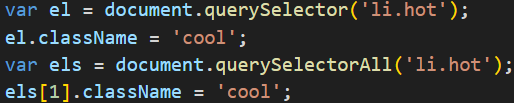
-Select elements by **tag name**



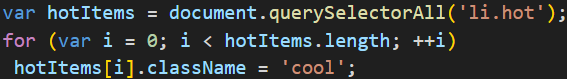
-Select elements using **CSS selectors**:

+**querySelector()**: return 1st element node that matches CSS style selector

+**querySelectorAll():** return NodeList of all matches



-**Repeat** actions for an entire **nodelist**:



-**Traverse** the **DOM**

+**parentNode**: finds the element node for the containing element in HTML

+**previousSibling, nextSibling**

+**firstChild, lastChild**

-**whitespace nodes**: Some browsers add a text node whenever they com across whitespace between elements. So properties above return different elements in different browsers. You could **strip all whitespace** out of HTML page -> harder to read.

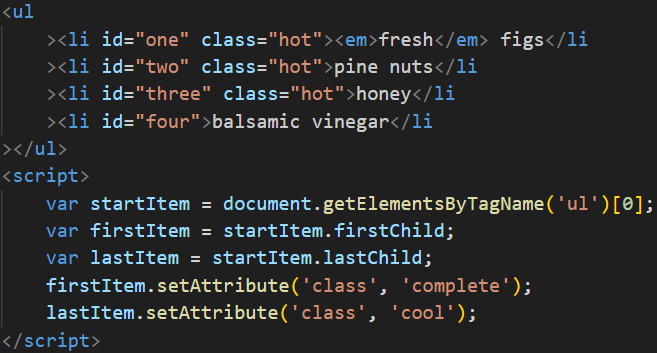
+One of the most popular ways to address this kind of problem is use **jQuery**.

-Previous & next sibling



-First & last child:

+These properties also return consistent results if there is **whitespace** between elements. We can make **closing tags put next to opening tags** of next element



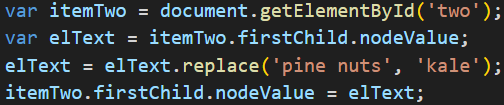
-**Get/Update** **element content**

+**Text nodes**: nodeValue: access text from node

+**Contain element**

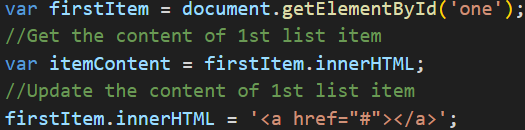
|  |  |
| --- | --- |
| **Property** | **Description** |
| innerHTML | Get/set text & markup |
| textContent | Get/set text only |
| innerText | Get/set text only |

-Access & Update a text node with **nodeValue**:

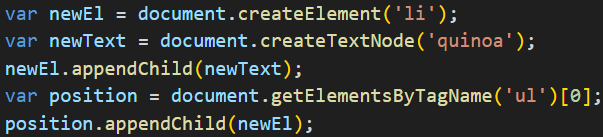


-Add or remove HTML content

+**innerHTML**: update entire fragment



+**DOM manipulation methods**: target individual nodes in DOM tree: **createElement()**, **createTextNode()**, **appendChild()**



# Chapter 6: Events

# Chapter 7: jQuery

# Chapter 8: Ajax & JSON

# Chapter 9: APIs