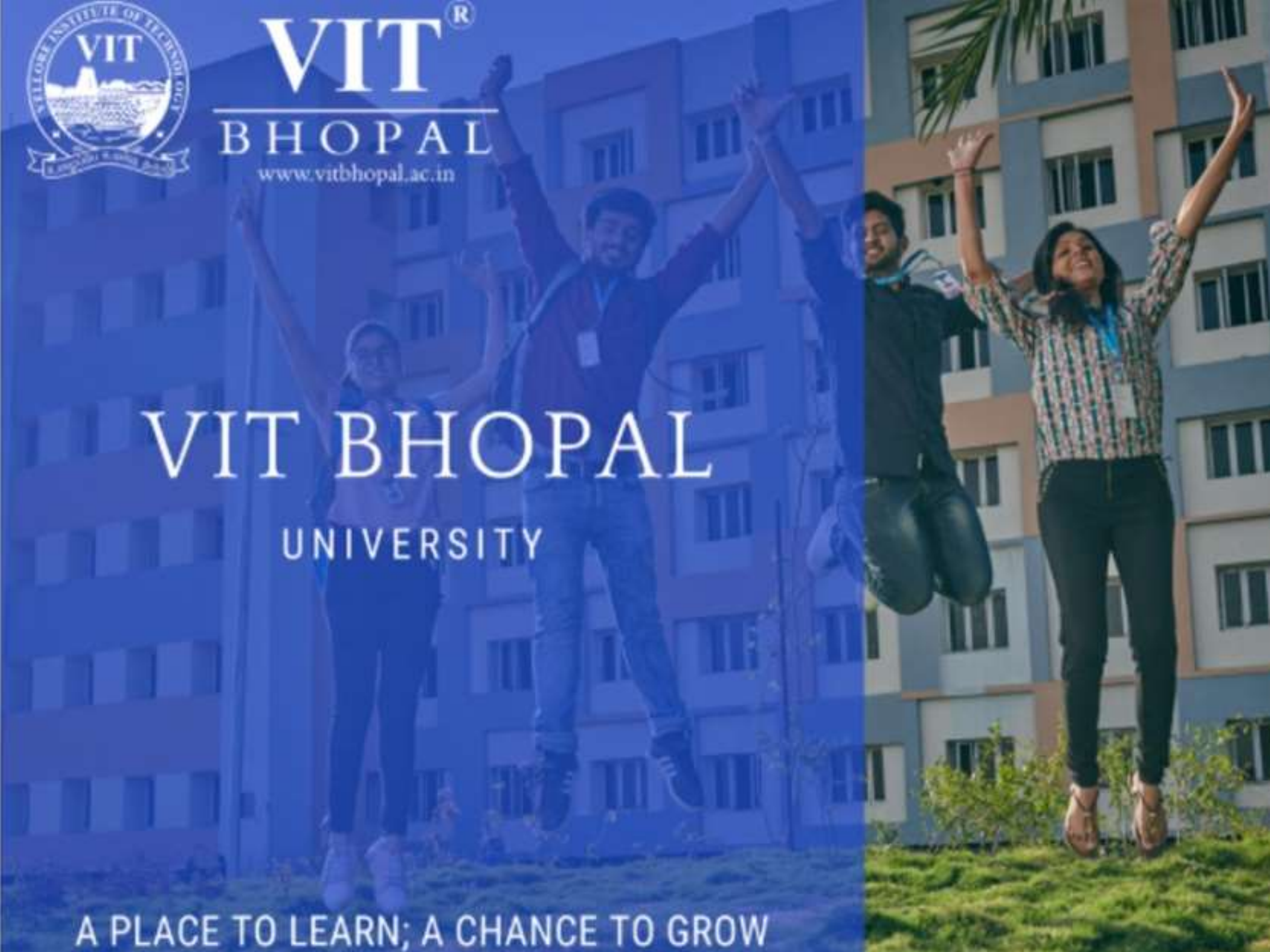
VIT®

BHOPAL

www.vitbhopal.ac.in



VIT BHOPAL

UNIVERSITY

A PLACE TO LEARN; A CHANCE TO GROW

WELCOME TO

CSE 4001 - INTERNET AND WEB PROGRAMMING

Unit 3

Java Script and JQuery

JavaScript Basics –Functions – Arrays

DOM - Built-in Objects -Regular

Expression - Event handling – Validation

– JSON Basics– JQuery Basics - plugins.

Text Books:

- 1. Thomas Powell, HTML and CSS, Complete Reference, Fifth Edition, Mc Graw Hill, 2010
- 2. Thomas Powell, Fritz Schneider , JavaScript The complete reference, Mc Graw Hill, 2013
- 3. Tom Christiansen, Nathan Torkington, Perl Cookbook, O'Reilly, 2012
- 4. David Powers, PHP Solutions, Dynamic web page design made easy, Apress, 2010
- 5. Joe Fawcett, Danny Ayers, Liam R. E. Quin, Beginning XML, 5th Edition, Wrox, 2012

Reference Books:

- 1. Paul Dietel, Harvey Dietel and Abbey Dietel, Internet and World Wide Web How to program, 5th International Edition, Pearson, 2012

Java script language

- JavaScript is the world's most popular programming language.
- JavaScript is the programming language of the Web.
- JavaScript is easy to learn.
- This tutorial will teach you JavaScript from basic to advanced

Why to study JavaScript?

- JavaScript is one of the 3 languages all web developers must learn:
- 1. HTML to define the content of web pages
- 2. CSS to specify the layout of web pages
- 3. JavaScript to program the behavior of web pages

JavaScript versions

- The Original JavaScript ES1 ES2 ES3 (1997-1999)
- The First Main Revision ES5 (2009)
- The Second Revision ES6 (2015)
- The Yearly Additions (2016, 2017, 2018)

- Commonly Asked Questions
- How do I get JavaScript?
- Where can I download JavaScript?
- Is JavaScript Free?
- You don't have to get or download JavaScript.
- JavaScript is already running in your browser on your computer, on your tablet, and on your smart-phone.
- JavaScript is free to use for everyone.

JavaScript Introduction

- JavaScript Can Change HTML Content
- One of many JavaScript HTML methods is getElementById().
-
- The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript": Ex:1

- JavaScript Can Change HTML Attribute Values
- In this example JavaScript changes the value of the src (source) attribute of an <img> tag:
- Ex: 2
- JavaScript Can Change HTML Styles (CSS)
- Changing the style of an HTML element, is a variant of changing an HTML attribute:
- Ex:3

Example

- document.getElementById("demo").style.fontSize = "35px";
- JavaScript Can Hide HTML Elements
- Hiding HTML elements can be done by changing the display style:

Example

- document.getElementById("demo").style.display = "none";
- JavaScript Can Show HTML Elements
- Showing hidden HTML elements can also be done by changing the display style:

JavaScript Versions and history

Brendan Eich created **JavaScript** in 1995 while he was at Netscape Communications Corporation, the creators of the legendary Netscape Navigator web browser. At the time, the Java coding language was rapidly gaining traction and Netscape Communications was working to make it available in Netscape Communicator.

JavaScript Syntax

- JavaScript syntax is the set of rules, how JavaScript programs are constructed:

// How to create variables:

- var x;
- let y;

// How to use variables:

- x = 5;
- y = 6;
- let z = x + y;
- JavaScript Values
- The JavaScript syntax defines two types of values:

- Fixed values
- Variable values
- Fixed values are called Literals.

- Variable values are called Variables.

- JavaScript Literals
- The two most important syntax rules for fixed values are:
- Ex:4

JavaScript Variables

- In a programming language, variables are used to store data values.
- JavaScript uses the keywords var, let and const to declare variables.
- An equal sign is used to assign values to variables.
- In this example, x is defined as a variable. Then, x is assigned (given) the value 6:
- Example
- var x = 5;
- var y = 6;
- var z = x + y;

- From the example above, you can expect:
- x stores the value 5
- y stores the value 6
- z stores the value 11
- Much Like Algebra
- In this example, price1, price2, and total, are variables:
- Example
- var price1 = 5;
- var price2 = 6;
- var total = price1 + price2;   Ex

- In programming, just like in algebra, we use variables (like price1) to hold values.
- In programming, just like in algebra, we use variables in expressions (total = price1 + price2).
- From the example above, you can calculate the total to be 11.
- JavaScript variables are containers for storing data values

JavaScript Operators

- JavaScript uses arithmetic operators (+ - * /) to compute values:
- (5 + 6) * 10
- JavaScript uses an assignment operator (=) to assign values to variables:
- let x, y;
- x = 5;
- y = 6;

JavaScript Expressions

- An expression is a combination of values, variables, and operators, which computes to a value.
- The computation is called an evaluation.
- For example, 5 * 10 evaluates to 50:
- 5 * 10
- Expressions can also contain variable values:
- x * 10
- The values can be of various types, such as numbers and strings.
- For example, "John" + " " + "Doe", evaluates to "John Doe":
- "John" + " " + "Doe"

JavaScript Keywords

- JavaScript keywords are used to identify actions to be performed.
- The let keyword tells the browser to create variables:
- let x, y;
- x = 5 + 6;
- y = x * 10;
- The var keyword also tells the browser to create variables:
- var x, y;
- x = 5 + 6;
- y = x * 10;

JavaScript Comments

- Not all JavaScript statements are "executed".
- Code after double slashes // or between /* and */ is treated as a comment.
- Comments are ignored, and will not be executed:
- let x = 5;   // I will be executed
- // x = 6;   I will NOT be executed
- You will learn more about comments in a later chapter.

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function is executed when "something" invokes it (calls it).
- function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
  }

JavaScript Function Syntax

- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas: **(*parameter1, parameter2, ...*)**
- The code to be executed, by the function, is placed inside curly brackets: **{}**
- function *name*(*parameter1, parameter2, parameter3*) {

  // code to be executed

  }

- Function **parameters** are listed inside the parentheses () in the function definition.
- Function **arguments** are the **values** received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables.
- A Function is much the same as a Procedure or a Subroutine, in other programming languages.

- Example
- Calculate the product of two numbers, and return the result:

-
  ```
  let x = myFunction(4, 3);   // Function is

  called, return value will end up in x

  function myFunction(a, b) {

    return a * b;             // Function returns the

  product of a and b

  }
  ```

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Return

- When JavaScript reaches a return statement, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**. The return value is "returned" back to the "caller":

- Ex

JavaScript Objects

- JavaScript objects are written with curly braces {}.
- Object properties are written as name:value pairs, separated by commas.
- Example
- const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
- The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.
- You will learn more about objects later in this tutorial.

The typeof Operator

- You can use the JavaScript typeof operator to find the type of a JavaScript variable.
- The typeof operator returns the type of a variable or an expression:
- Example
- typeof ""   // Returns "string"
- typeof "John"   // Returns "string"
- typeof "John Doe"   // Returns "string"
- Example
- typeof 0   // Returns "number"
- typeof 314   // Returns "number"
- typeof 3.14   // Returns "number"
- typeof (3)   // Returns "number"
- typeof (3 + 4)   // Returns "number"
- You will learn more about typeof later in this tutorial.

- Undefined
- In JavaScript, a variable without a value, has the value undefined. The type is also undefined.
- Example
- let car;   // Value is undefined, type is undefined
- Any variable can be emptied, by setting the value to undefined. The type will also be undefined.
- Example
- car = undefined;   // Value is undefined, type is undefined
- Empty Values
- An empty value has nothing to do with undefined.
- An empty string has both a legal value and a type.
- Example
- let car = "";   // The value is "", the typeof is "string"

Why Functions?

- You can reuse code: Define the code once, and use it many times.
- You can use the same code many times with different arguments, to produce different results.
- The () Operator Invokes the Function
- Functions Used as Variable Values
- Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

- Local Variables
- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables can only be accessed from within the function.
- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
- Local variables are created when a function starts, and deleted when the function is completed.

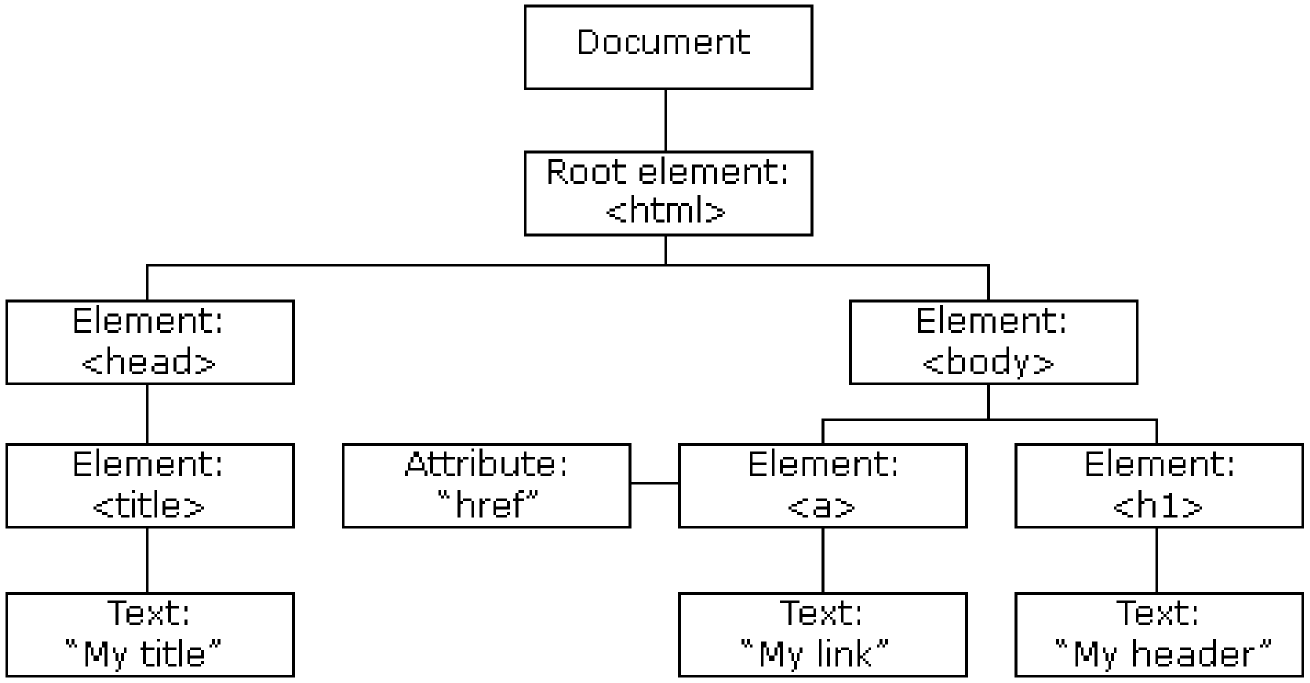
JavaScript Arrays

- JavaScript arrays are written with square brackets.
- Array items are separated by commas.
- The following code declares (creates) an array called cars, containing three items (car names):
- Example
- const cars = ["Saab", "Volvo", "BMW"];
- Array indexes are zero-based, which means the first item is [0], second is [1], and so on.
- You will learn more about arrays later in this tutorial.

JavaScript DOM

- Document Object Model
- When a web page is loaded, the browser creates

  a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:
- The HTML DOM Tree of Objects



- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the DOM?

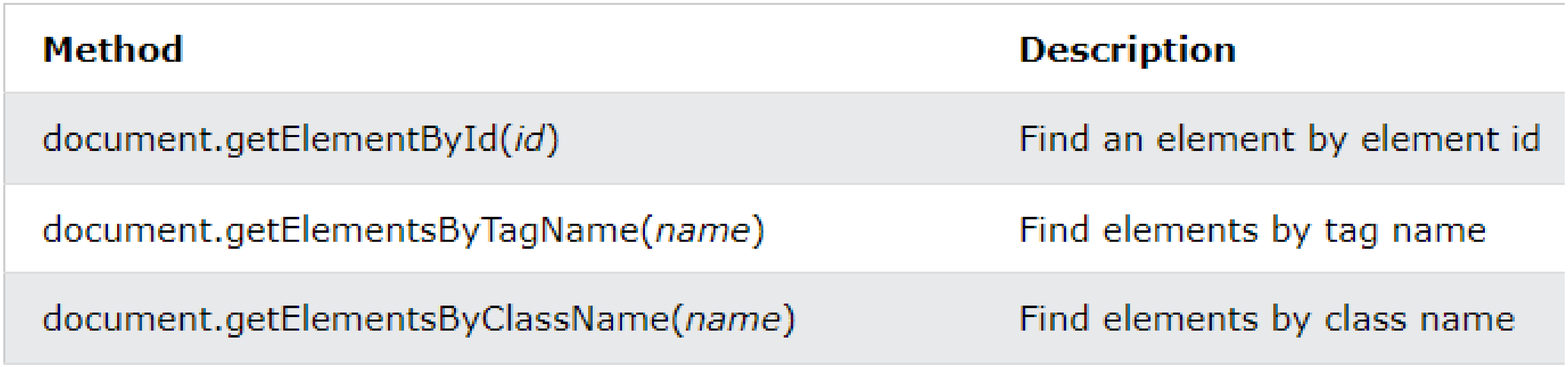
- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- The W3C DOM standard is separated into 3 different parts:
- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

The DOM Programming Interface

- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).
- Example
- The following example changes the content (the innerHTML) of the <p> element with id="demo":

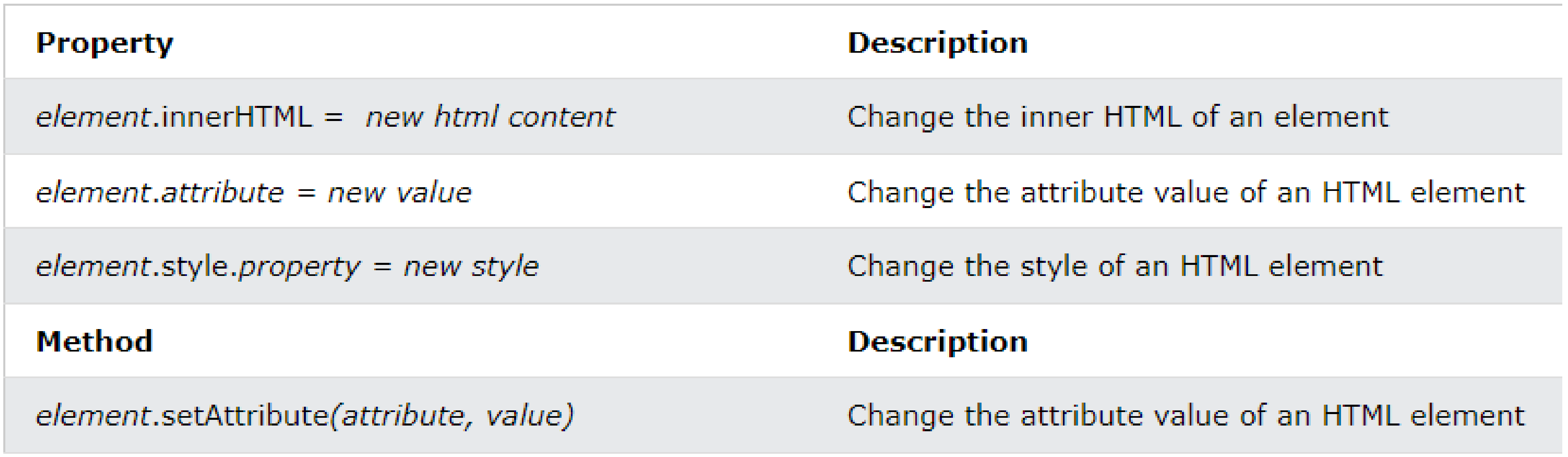
- In the example above, getElementById is a **method**, while innerHTML is a **property**.
- The getElementById Method
- The most common way to access an HTML element is to use the id of the element.
- In the example above the getElementById method used id="demo" to find the element.
- The innerHTML Property
- The easiest way to get the content of an element is by using the innerHTML property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

Finding HTML Elements



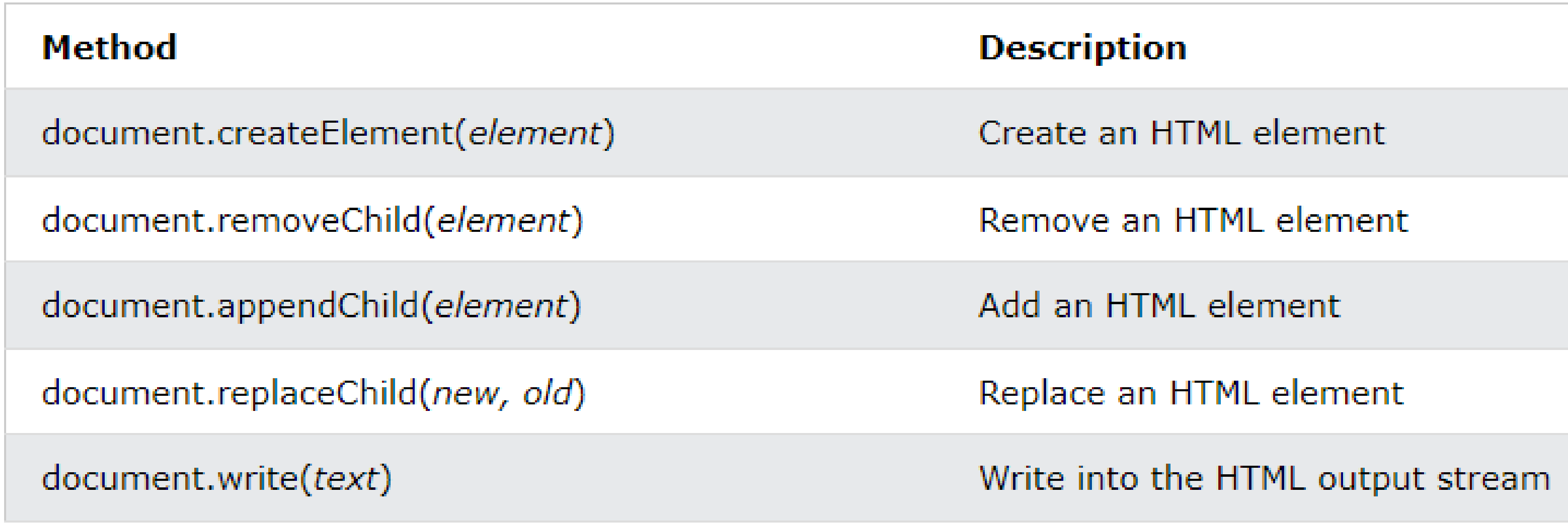
Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements



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

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(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Finding HTML Elements

- Often, with JavaScript, you want to manipulate HTML elements.
- To do so, you have to find the elements first. There are several ways to do this:
- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors
- Finding HTML elements by HTML object collections
- Finding HTML Element by Id
- The easiest way to find an HTML element in the DOM, is by using the element id.
- This example finds the element with id="intro": EX

Finding HTML Elements by Tag Name

- This example finds all <p> elements:

- This example finds the element with id="main", and then finds all <p> elements inside "main":

Changing HTML Content

- The easiest way to modify the content of an HTML element is by using the innerHTML property.
- To change the content of an HTML element, use this syntax:
- document.getElementById(*id*).innerHTML = *new HTML*
- This example changes the content of a <p> element:

Changing the Value of an Attribute

- To change the value of an HTML attribute, use this syntax:
- document.getElementById(*id*).*attribute = new value*
- This example changes the value of the src attribute of an <img> element:

Dynamic HTML content

- JavaScript can create dynamic HTML content:
- Date : Tue Feb 22 2022 21:58:43 GMT+0530 (India Standard Time)

JavaScript Form Validation

- HTML form validation can be done by JavaScript.
- If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:
- JavaScript Example

-
  ```
  function validateForm() {
    let x = document.forms["myForm"]["fname"].value;
    if (x == "") {
      alert("Name must be filled out");
      return false;
    }
  }
  ```

JSON - Introduction

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a **text format** for storing and transporting data
- JSON is "self-describing" and easy to understand
- This example is a JSON string:
- '{"name":"John", "age":30, "car":null}'
- It defines an object with 3 properties:
- name
- age
- car
- Each property has a value.

- If you parse the JSON string with a JavaScript program, you can access the data as an object:
- let personName = obj.name;
  let personAge = obj.age;

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent

Why Use JSON?

- The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.
- Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.
- JavaScript has a built in function for converting JSON strings into JavaScript objects:
- JSON.parse()
- JavaScript also has a built in function for converting an object into a JSON string:
- JSON.stringify()

- Storing Data
- When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.
- JSON makes it possible to store JavaScript objects as text.

JSON Syntax Rules

- JSON syntax is derived from JavaScript object notation syntax:
- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

- JSON Data - A Name and a Value
- JSON data is written as name/value pairs (aka key/value pairs).
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:
- Example
- "name":"John“
- JSON - Evaluates to JavaScript Objects
- The JSON format is almost identical to JavaScript objects.
- In JSON, *keys* must be strings, written with double quotes:
- JSON
- {"name":"John"}

- In JavaScript, keys can be strings, numbers, or identifier names:
- JavaScript
- {name:"John"}
- JSON Values

In **JSON**, *values* must be one of the following data types:

- a string
- a number
- an object
- an array
- a boolean
- null

- In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:
- a function
- a date
- undefined
- In JSON, *string values* must be written with double quotes:
- JSON
- {"name":"John"}

- JavaScript Objects
- Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.
- With JavaScript you can create an object and assign data to it, like this:
- Example
- person = {name:"John", age:31, city:"New York"};
- You can access a JavaScript object like this:

- Example
- // returns John

  person.name;
- It can also be accessed like this:
- Example
- // returns John

  person["name"];
- Data can be modified like this:

- Example
- person.name = “Subash";
- It can also be modified like this:
- Example
- person["name"] = “Subash";
- You will learn how to convert JavaScript objects into JSON later in this tutorial.
- JavaScript Arrays as JSON
- The same way JavaScript objects can be written as JSON, JavaScript arrays can also be written as JSON.
- You will learn more about objects and arrays later in this tutorial.

- JSON Files
- The file type for JSON files is ".json"
- The MIME type for JSON text is "application/json"

jQuery Introduction

What is jQuery?

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

- The jQuery library contains the following features:
- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities
- **Tip:** In addition, jQuery has plugins for almost any task out there.

- Why jQuery?
- There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable.
- Many of the biggest companies on the Web use jQuery, such as:
- Google
- Microsoft
- IBM
- Netflix

- Adding jQuery to Your Web Pages
- There are several ways to start using jQuery on your web site. You can:
- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google
- Downloading jQuery
- There are two versions of jQuery available for downloading:
- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

- Both versions can be downloaded from jQuery.com.
- The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):
- <head>

  <script src="jquery-3.5.1.min.js"></script>

  </head>
- jQuery CDN
- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- Google is an example of someone who host jQuery:

- Google CDN:
- <head>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

  </head>

- jQuery Syntax
- The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).
- Basic syntax is: **$(*selector*).*action*()**
- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)
- Examples:
- $(this).hide() - hides the current element.
- $("p").hide() - hides all <p> elements.
- $(".test").hide() - hides all elements with class="test".
- $("#test").hide() - hides the element with id="test".

- The Document Ready Event
- You might have noticed that all jQuery methods in our examples, are inside a document ready event:
- $(document).ready(function(){

  // jQuery methods go here...

  });
- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

- Here are some examples of actions that can fail if methods are run before the document is fully loaded:
- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet
- **Tip:** The jQuery team has also created an even shorter method for the document ready event:

- $(function(){

  // jQuery methods go here...

  });

jQuery Selectors

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- All selectors in jQuery start with the dollar sign and parentheses: $().

- The element Selector
- The jQuery element selector selects elements based on the element name.
- You can select all <p> elements on a page like this:
- $("p")

- **Example**
- When a user clicks on a button, all <p> elements will be hidden:
- Example
-
```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```