

1. **What is JavaScript?**

JavaScript is a client-side as well as server side scripting language that can be inserted into HTML pages and is understood by web browsers. JavaScript is also an Object based Programming language

2. **Enumerate the differences between Java and JavaScript?**

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for the different intent. Java is an object - oriented programming (OOPS) or structured programming language like C++ or C whereas JavaScript is a client-side scripting language.

3. **What are JavaScript Data Types?**

Following are the JavaScript Data types:

- Number
- String
- Boolean
- Object
- Undefined

4. **What is the use of isNaN function?**

isNaN function returns true if the argument is not a number otherwise it is false.

5. **Which company developed JavaScript?**

Netscape is the software company who developed JavaScript.

6. What are undeclared and undefined variables?

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

7. What is a prompt box?

A prompt box is a box which allows the user to enter input by providing a text box. Label and box will be provided to enter the text or number.

8. What is 'this' keyword in JavaScript?

'This' keyword refers to the object from where it was called.

9. Which symbol is used for comments in Javascript?

// for Single line comments and

/* MultiLine

Comment */

10. What are all the looping structures in JavaScript?

Following are looping structures in Javascript:

- For
- While
- do-while loops

11. What is called Variable typing in Javascript?

Variable typing is used to assign a number to a variable and the same variable can be assigned to a string.

- Example

```
o i = 30;  
o i = "string";
```

- This is called variable typing.

12. How can you convert the string of any base to integer in JavaScript?

The `parseInt()` function is used to convert numbers between different bases. `parseInt()` takes the string to be converted as its first parameter, and the second parameter is the base of the given string.

In order to convert 4A (of base 16) to integer, the code used will be:

```
parseInt ("4A", 16);
```

13. Explain the difference between "==" and "==="?"

"==" checks only for equality in value whereas "===" is a stricter equality test and returns false if either the value or the type of the two variables that are different.

14. What are all the types of Pop up boxes available in JavaScript?

- Alert
- Confirm and
- Prompt

15. What is the data type of variables of in JavaScript?

All variables in the JavaScript are object data types.

16. What is the difference between an alert box and a confirmation box?

An alert box displays only one button which is the OK button.

But a Confirmation box displays two buttons namely OK and cancel.

17. What is the way to get the status of a CheckBox?

The status can be acquired as follows -

```
alert(document.getElementById('checkbox1').checked);
```

If the CheckBox will be checked, this alert will return TRUE.

18. Why it is not advised to use innerHTML in JavaScript?

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML and, therefore, it is easier to insert rouge code in the document and, thus, make the web page unstable.

Javascript Operators

Disha H. Parekh

What is an operator?

- Let us take a simple expression $4 + 5$ is equal to 9. Here 4 and 5 are called operands and '+' is
- called the **operator**. JavaScript supports the **following types of operators**.
 - Arithmetic Operators
 - Comparison Operators
 - Logical *or* Relational Operators
 - Assignment Operators
 - Conditional *or* ternary Operators
- Lets have a look on all operators one by one.

Arithmetic Operators

- JavaScript supports the following arithmetic operators
 - Addition (+)
 - Subtraction (-)
 - Multiplication (*)
 - Division (/)
 - Modulus (%)
 - Increment (++)
 - Decrement (--)

Arithmetic Operators

- <Z:\IWT\PPts\first.html>

Comparison Operators

- JavaScript supports the following comparison operators
 - `==`
 - `!=`
 - `<=`
 - `<`
 - `>=`
 - `>`

Comparison Operators

- Z:\IWT\PPts\second_comparision.html

Beginner's essential

JavaScript Cheat Sheet

The Language of the Web.

TABLE OF CONTENTS

JavaScript Basics	3
Variables in JavaScript	3
The Next Level: Arrays	4
Operators	5
Functions	6
JavaScript Loop	7
If - Else Statements	8
Strings	8
Regular Expression Syntax	9
Numbers and Math	11
Dealing with Dates in JavaScript	13
DOM Mode	14
Working with the User Browser	17
JavaScript Events	19

JAVASCRIPT BASICS

Including JavaScript in an HTML Page

```
<script type="text/javascript">
```

```
    //JS code goes here
```

```
</script>
```

Call an External JavaScript File

```
<script src="myscript.js"></script><code></code>
```

Including Comments

Single line comments - //

Multi-line comments - /* comment here */

VARIABLES IN JAVASCRIPT

var, const, let

var – The most common variable. Can be reassigned but only accessed within a function. Variables defined with var move to the top when code is executed.

const – Can not be reassigned and not accessible before they appear within the code.

let – Similar to const, however, let variable can be reassigned but not re-declared.

Data Types

Numbers – **var** age = 23

Variables – **var** x

Text (strings) – **var** a = "init"

Operations – **var** b = 1 + 2 + 3

True or false statements – `var c = true`

Constant numbers – `const PI = 3.14`

Objects – `var name = {firstName:"John", lastName:"Doe"}`

Objects

```
var person = {  
  firstName:"John",  
  lastName:"Doe",  
  age:20,  
  nationality:"German"  
};
```

THE NEXT LEVEL: ARRAYS

```
var fruit = ["Banana", "Apple", "Pear"];
```

Array Methods

`concat()` – Join several arrays into one

`indexOf()` – Returns the primitive value of the specified object

`join()` – Combine elements of an array into a single string and return the string

`lastIndexOf()` – Gives the last position at which a given element appears in an array

`pop()` – Removes the last element of an array

`push()` – Add a new element at the end

`reverse()` – Sort elements in descending order

`shift()` – Remove the first element of an array

`slice()` – Pulls a copy of a portion of an array into a new array

`sort()` – Sorts elements alphabetically

`splice()` – Adds elements in a specified way and position

`toString()` – Converts elements to strings

`unshift()` – Adds a new element to the beginning

`valueOf()` – Returns the first position at which a given element appears in an array

OPERATORS

Basic Operators

`+` – Addition

`-` – Subtraction

`*` – Multiplication

`/` – Division

`(...)` – Grouping operator, operations within brackets are executed earlier than those outside

`%` – Modulus (remainder)

`++` – Increment numbers

`--` – Decrement numbers

Comparison Operators

`==` – Equal to

`===` – Equal value and equal type

`!=` – Not equal

`!==` – Not equal value or not equal type

`>` – Greater than

`<` – Less than

`>=` – Greater than or equal to

`<=` – Less than or equal to

`?` – Ternary operator

Logical Operators

`&&` – Logical and

`||` – Logical or

`!` – Logical not

Bitwise Operators

`&` – AND statement

`|` – OR statement

`~` – NOT

`^` – XOR

`<<` – Left shift

`>>` – Right shift

`>>>` – Zero fill right shift

FUNCTIONS

```
function name(parameter1, parameter2, parameter3) {
```

```
    // what the function does
```

```
}
```

Outputting Data

`alert()` – Output data in an alert box in the browser window

`confirm()` – Opens up a yes/no dialog and returns true/false depending on user click

`console.log()` – Writes information to the browser console, good for debugging purposes

`document.write()` – Write directly to the HTML document

`prompt()` – Creates an dialogue for user input

Global Functions

`decodeURI()` – Decodes a Uniform Resource Identifier (URI) created by `encodeURIComponent` or similar

`decodeURIComponent()` – Decodes a URI component

`encodeURIComponent()` – Encodes a URI into UTF-8

`encodeURIComponent()` – Same but for URI components

`eval()` – Evaluates JavaScript code represented as a string

`isFinite()` – Determines whether a passed value is a finite number

`isNaN()` – Determines whether a value is NaN or not

`Number()` – Returns a number converted from its argument

`parseFloat()` – Parses an argument and returns a floating point number

`parseInt()` – Parses its argument and returns an integer

JAVASCRIPT LOOPS

```
for (before loop; condition for loop; execute after loop) {  
    // what to do during the loop  
}
```

for – The most common way to create a loop in JavaScript

while – Sets up conditions under which a loop executes

do while – Similar to the while loop, however, it executes at least once and performs a check at the end to see if the condition is met to execute again

break – Used to stop and exit the cycle at certain conditions

continue – Skip parts of the cycle if certain conditions are met

IF - ELSE STATEMENTS

```
if (condition) {  
    // what to do if condition is met  
} else {  
    // what to do if condition is not met  
}
```

STRINGS

```
var person = "John Doe";
```

Escape Characters

```
\' - Single quote  
\" - Double quote  
\\ - Backslash  
\b - Backspace  
\f - Form feed  
\n - New line  
\r - Carriage return  
\t - Horizontal tabulator  
\v - Vertical tabulator
```

String Methods

```
charAt() - Returns a character at a specified position inside a  
string  
charCodeAt() - Gives you the unicode of character at that position  
concat() - Concatenates (joins) two or more strings into one
```

fromCharCode() – Returns a string created from the specified sequence of UTF-16 code units

indexOf() – Provides the position of the first occurrence of a specified text within a string

lastIndexOf() – Same as `indexOf()` but with the last occurrence, searching backwards

match() – Retrieves the matches of a string against a search pattern

replace() – Find and replace specified text in a string

search() – Executes a search for a matching text and returns its position

slice() – Extracts a section of a string and returns it as a new string

split() – Splits a string object into an array of strings at a specified position

substr() – Similar to `slice()` but extracts a substring depended on a specified number of characters

substring() – Also similar to `slice()` but can't accept negative indices

toLowerCase() – Convert strings to lower case

toUpperCase() – Convert strings to upper case

valueOf() – Returns the primitive value (that has no properties or methods) of a string object

REGULAR EXPRESSION SYNTAX

Pattern Modifiers

e – Evaluate replacement

i – Perform case-insensitive matching

g – Perform global matching

m – Perform multiple line matching

s – Treat strings as single line

x – Allow comments and whitespace in pattern

U – Ungreedy pattern

Brackets

[abc] – Find any of the characters between the brackets

[^abc] – Find any character not in the brackets

[0-9] – Used to find any digit from 0 to 9

[A-z] – Find any character from uppercase A to lowercase z

(a|b|c) – Find any of the alternatives separated with |

Metacharacters

. – Find a single character, except newline or line terminator

\w – Word character

\W – Non-word character

\d – A digit

\D – A non-digit character

\s – Whitespace character

\S – Non-whitespace character

\b – Find a match at the beginning/end of a word

\B – A match not at the beginning/end of a word

\0 – NUL character

\n – A new line character

\f – Form feed character

\r – Carriage return character

\t – Tab character

\v – Vertical tab character

\xxx – The character specified by an octal number xxx

`\xdd` – Character specified by a hexadecimal number `dd`

`\uxxxx` – The Unicode character specified by a hexadecimal number `xxxx`

Quantifiers

`n+` – Matches any string that contains at least one `n`

`n*` – Any string that contains zero or more occurrences of `n`

`n?` – A string that contains zero or one occurrences of `n`

`n{X}` – String that contains a sequence of `X` `n`'s

`n{X,Y}` – Strings that contains a sequence of `X` to `Y` `n`'s

`n{X,}` – Matches any string that contains a sequence of at least `X` `n`'s

`n$` – Any string with `n` at the end of it

`^n` – String with `n` at the beginning of it

`?=n` – Any string that is followed by a specific string `n`

`?!n` – String that is not followed by a specific string `n`

NUMBERS AND MATH

Number Properties

`MAX_VALUE` – The maximum numeric value representable in JavaScript

`MIN_VALUE` – Smallest positive numeric value representable in JavaScript

`NaN` – The "Not-a-Number" value

`NEGATIVE_INFINITY` – The negative Infinity value

`POSITIVE_INFINITY` – Positive Infinity value

Number Methods

`toExponential()` – Returns a string with a rounded number written as exponential notation

`toFixed()` – Returns the string of a number with a specified number of decimals

`toPrecision()` – String of a number written with a specified length

`toString()` – Returns a number as a string

`valueOf()` – Returns a number as a number

Math Properties

`E` – Euler's number

`LN2` – The natural logarithm of 2

`LN10` – Natural logarithm of 10

`LOG2E` – Base 2 logarithm of E

`LOG10E` – Base 10 logarithm of E

`PI` – The number PI

`SQRT1_2` – Square root of 1/2

`SQRT2` – The square root of 2

Math Methods

`abs(x)` – Returns the absolute (positive) value of x

`acos(x)` – The arccosine of x, in radians

`asin(x)` – Arcsine of x, in radians

`atan(x)` – The arctangent of x as a numeric value

`atan2(y,x)` – Arctangent of the quotient of its arguments

`ceil(x)` – Value of x rounded up to its nearest integer

`cos(x)` – The cosine of x (x is in radians)

`exp(x)` – Value of E^x

`floor(x)` – The value of x rounded down to its nearest integer

`log(x)` – The natural logarithm (base E) of x

`max(x,y,z,...,n)` – Returns the number with the highest value

`min(x,y,z,...,n)` – Same for the number with the lowest value

`pow(x,y)` – X to the power of y

`random()` – Returns a random number between 0 and 1

`round(x)` – The value of x rounded to its nearest integer

`sin(x)` – The sine of x (x is in radians)

`sqrt(x)` – Square root of x

`tan(x)` – The tangent of an angle

DEALING WITH DATES IN JAVASCRIPT

Setting Dates

`Date()` – Creates a new date object with the current date and time

`Date(2017, 5, 21, 3, 23, 10, 0)` – Create a custom date object. The numbers represent year, month, day, hour, minutes, seconds, milliseconds. You can omit anything you want except for year and month.

`Date("2017-06-23")` – Date declaration as a string

Pulling Date and Time Values

`getDate()` – Get the day of the month as a number (1-31)

`getDay()` – The weekday as a number (0-6)

`getFullYear()` – Year as a four digit number (yyyy)

`getHours()` – Get the hour (0-23)

`getMilliseconds()` – The millisecond (0-999)

`getMinutes()` – Get the minute (0-59)

`getMonth()` – Month as a number (0-11)

`getSeconds()` – Get the second (0-59)

`getTime()` – Get the milliseconds since January 1, 1970

`getUTCDate()` – The day (date) of the month in the specified date according to universal time (also available for day, month, fullyear, hours, minutes etc.)

`parse` – Parses a string representation of a date, and returns the number of milliseconds since January 1, 1970

Set Part of a Date

`setDate()` – Set the day as a number (1-31)

`setFullYear()` – Sets the year (optionally month and day)

`setHours()` – Set the hour (0-23)

`setMilliseconds()` – Set milliseconds (0-999)

`setMinutes()` – Sets the minutes (0-59)

`setMonth()` – Set the month (0-11)

`setSeconds()` – Sets the seconds (0-59)

`setTime()` – Set the time (milliseconds since January 1, 1970)

`setUTCDate()` – Sets the day of the month for a specified date according to universal time (also available for day, month, fullyear, hours, minutes etc.)

DOM MODE

Node Properties

`attributes` – Returns a live collection of all attributes registered to and element

`baseURI` – Provides the absolute base URL of an HTML element

`childNodes` – Gives a collection of an element's child nodes

`firstChild` – Returns the first child node of an element

`lastChild` – The last child node of an element

`nextSibling` – Gives you the next node at the same node tree level

`nodeName` – Returns the name of a node

nodeType – Returns the type of a node

nodeValue – Sets or returns the value of a node

ownerDocument – The top-level document object for this node

parentNode – Returns the parent node of an element

previousSibling – Returns the node immediately preceding the current one

textContent – Sets or returns the textual content of a node and its descendants

Node Methods

appendChild() – Adds a new child node to an element as the last child node

cloneNode() – Clones an HTML element

compareDocumentPosition() – Compares the document position of two elements

getFeature() – Returns an object which implements the APIs of a specified feature

hasAttributes() – Returns true if an element has any attributes, otherwise false

hasChildNodes() – Returns true if an element has any child nodes, otherwise false

insertBefore() – Inserts a new child node before a specified, existing child node

isDefaultNamespace() – Returns true if a specified namespaceURI is the default, otherwise false

isEqualNode() – Checks if two elements are equal

isSameNode() – Checks if two elements are the same node

isSupported() – Returns true if a specified feature is supported on the element

lookupNamespaceURI() – Returns the namespaceURI associated with a given node

lookupPrefix() – Returns a DOMString containing the prefix for a given namespaceURI, if present

normalize() – Joins adjacent text nodes and removes empty text nodes in an element

removeChild() – Removes a child node from an element

replaceChild() – Replaces a child node in an element

Element Methods

getAttribute() – Returns the specified attribute value of an element node

getAttributeNS() – Returns string value of the attribute with the specified namespace and name

getAttributeNode() – Gets the specified attribute node

getAttributeNodeNS() – Returns the attribute node for the attribute with the given namespace and name

getElementsByTagName() – Provides a collection of all child elements with the specified tag name

getElementsByTagNameNS() – Returns a live HTMLCollection of elements with a certain tag name belonging to the given namespace

hasAttribute() – Returns true if an element has any attributes, otherwise false

hasAttributeNS() – Provides a true/false value indicating whether the current element in a given namespace has the specified attribute

removeAttribute() – Removes a specified attribute from an element

removeAttributeNS() – Removes the specified attribute from an element within a certain namespace

removeAttributeNode() – Takes away a specified attribute node and returns the removed node

setAttribute() – Sets or changes the specified attribute to a specified value

setAttributeNS() – Adds a new attribute or changes the value of an attribute with the given namespace and name

setAttributeNode() – Sets or changes the specified attribute node

`setAttributeNodeNS()` – Adds a new namespaced attribute node to an element

WORKING WITH THE USER BROWSER

Window Properties

`closed` – Checks whether a window has been closed or not and returns true or false

`defaultStatus` – Sets or returns the default text in the statusbar of a window

`document` – Returns the document object for the window

`frames` – Returns all `<iframe>` elements in the current window

`history` – Provides the History object for the window

`innerHeight` – The inner height of a window's content area

`innerWidth` – The inner width of the content area

`length` – Find out the number of `<iframe>` elements in the window

`location` – Returns the location object for the window

`name` – Sets or returns the name of a window

`navigator` – Returns the Navigator object for the window

`opener` – Returns a reference to the window that created the window

`outerHeight` – The outer height of a window, including toolbars/scrollbars

`outerWidth` – The outer width of a window, including toolbars/scrollbars

`pageXOffset` – Number of pixels the current document has been scrolled horizontally

`pageYOffset` – Number of pixels the document has been scrolled vertically

`parent` – The parent window of the current window

`screen` – Returns the Screen object for the window

`screenLeft` – The horizontal coordinate of the window (relative to screen)

`screenTop` – The vertical coordinate of the window

`screenX` – Same as `screenLeft` but needed for some browsers

`screenY` – Same as `screenTop` but needed for some browsers

`self` – Returns the current window

`status` – Sets or returns the text in the statusbar of a window

`top` – Returns the topmost browser window

Window Methods

`alert()` – Displays an alert box with a message and an OK button

`blur()` – Removes focus from the current window

`clearInterval()` – Clears a timer set with `setInterval()`

`clearTimeout()` – Clears a timer set with `setTimeout()`

`close()` – Closes the current window

`confirm()` – Displays a dialogue box with a message and an OK and Cancel button

`focus()` – Sets focus to the current window

`moveBy()` – Moves a window relative to its current position

`moveTo()` – Moves a window to a specified position

`open()` – Opens a new browser window

`print()` – Prints the content of the current window

`prompt()` – Displays a dialogue box that prompts the visitor for input

`resizeBy()` – Resizes the window by the specified number of pixels

`resizeTo()` – Resizes the window to a specified width and height

`scrollBy()` – Scrolls the document by a specified number of pixels

`scrollTo()` – Scrolls the document to specified coordinates

setInterval() – Calls a function or evaluates an expression at specified intervals

setTimeout() – Calls a function or evaluates an expression after a specified interval

stop() – Stops the window from loading

Screen Properties

availHeight – Returns the height of the screen (excluding the Windows Taskbar)

availWidth – Returns the width of the screen (excluding the Windows Taskbar)

colorDepth – Returns the bit depth of the color palette for displaying images

height – The total height of the screen

pixelDepth – The color resolution of the screen in bits per pixel

width – The total width of the screen

JAVASCRIPT EVENTS

Mouse

onclick – The event occurs when the user clicks on an element

oncontextmenu – User right-clicks on an element to open a context menu

ondblclick – The user double-clicks on an element

onmousedown – User presses a mouse button over an element

onmouseenter – The pointer moves onto an element

onmouseleave – Pointer moves out of an element

onmousemove – The pointer is moving while it is over an element

onmouseover – When the pointer is moved onto an element or one of its children

onmouseout – User moves the mouse pointer out of an element or one of its children

onmouseup – The user releases a mouse button while over an element

Keyboard

onkeydown – When the user is pressing a key down

onkeypress – The moment the user starts pressing a key

onkeyup – The user releases a key

Frame

onabort – The loading of a media is aborted

onbeforeunload – Event occurs before the document is about to be unloaded

onerror – An error occurs while loading an external file

onhashchange – There have been changes to the anchor part of a URL

onload – When an object has loaded

onpagehide – The user navigates away from a webpage

onpageshow – When the user navigates to a webpage

onresize – The document view is resized

onscroll – An element's scrollbar is being scrolled

onunload – Event occurs when a page has unloaded

Form

onblur – When an element loses focus

onchange – The content of a form element changes (for `<input>`, `<select>` and `<textarea>`)

onfocus – An element gets focus

onfocusin – When an element is about to get focus

onfocusout – The element is about to lose focus

oninput – User input on an element

oninvalid – An element is invalid

onreset – A form is reset

onsearch – The user writes something in a search field
(for `<input="search">`)

onselect – The user selects some text (for `<input>` and `<textarea>`)

onsubmit – A form is submitted

Drag

ondrag – An element is dragged

ondragend – The user has finished dragging the element

ondragenter – The dragged element enters a drop target

ondragleave – A dragged element leaves the drop target

ondragover – The dragged element is on top of the drop target

ondragstart – User starts to drag an element

ondrop – Dragged element is dropped on the drop target

Clipboard

oncopy – User copies the content of an element

oncut – The user cuts an element's content

onpaste – A user pastes content in an element

Media

onabort – Media loading is aborted

oncanplay – The browser can start playing media (e.g. a file has buffered enough)

oncanplaythrough – When browser can play through media without stopping

ondurationchange – The duration of the media changes

onended – The media has reach its end

onerror – Happens when an error occurs while loading an external file

onloadeddata – Media data is loaded

onloadedmetadata – Meta data (like dimensions and duration) are loaded

onloadstart – Browser starts looking for specified media

onpause – Media is paused either by the user or automatically

onplay – The media has been started or is no longer paused

onplaying – Media is playing after having been paused or stopped for buffering

onprogress – Browser is in the process of downloading the media

onratechange – The playing speed of the media changes

onseeked – User is finished moving/skipping to a new position in the media

onseeking – The user starts moving/skipping

onstalled – The browser is trying to load the media but it is not available

onsuspend – Browser is intentionally not loading media

ontimeupdate – The playing position has changed (e.g. because of fast forward)

onvolumechange – Media volume has changed (including mute)

onwaiting – Media paused but expected to resume (for example, buffering)

Animation

animationend – A CSS animation is complete

animationiteration – CSS animation is repeated

animationstart – CSS animation has started

Other

transitionend – Fired when a CSS transition has completed

onmessage – A message is received through the event source

onoffline – Browser starts to work offline

ononline – The browser starts to work online

onpopstate – When the window's history changes

onshow – A <menu> element is shown as a context menu

onstorage – A Web Storage area is updated

ontoggle – The user opens or closes the <details> element

onwheel – Mouse wheel rolls up or down over an element

ontouchcancel – Screen touch is interrupted

ontouchend – User finger is removed from a touch screen

ontouchmove – A finger is dragged across the screen

ontouchstart – Finger is placed on touch screen

Errors

try – Lets you define a block of code to test for errors

catch – Set up a block of code to execute in case of an error

throw – Create custom error messages instead of the standard JavaScript errors

finally – Lets you execute code, after try and catch, regardless of the result

Error Name Values

name – Sets or returns the error name

message – Sets or returns an error message in string form

EvalError – An error has occurred in the eval() function

RangeError – A number is "out of range"

ReferenceError – An illegal reference has occurred

SyntaxError – A syntax error has occurred

TypeError – A type error has occurred

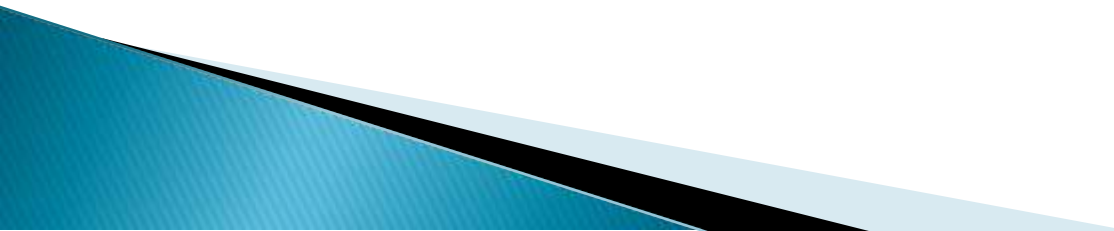
URIError – An encodeURI() error has occurred

Java Script in HTML

Disha H. Parekh



Why 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
- 

The `<script>` Tag

- ▶ JavaScript code must be inserted between `<script>` and `</script>` tags.

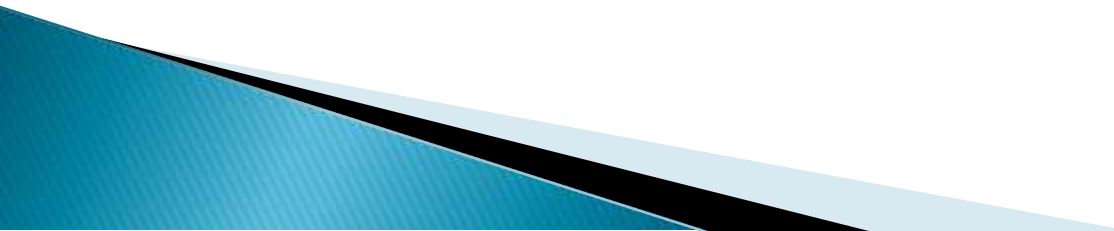
```
<script>
```

```
    //java script code
```

```
</script>
```



JavaScript Functions and Events

- ▶ A JavaScript **function** is a block of JavaScript code, that can be executed when "called" for.
 - ▶ For example, a function can be called when an **event** occurs, like when the user clicks a button.
- 

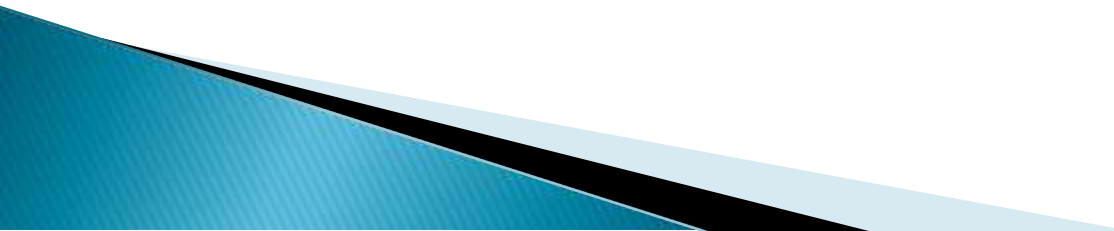
JavaScript in `<head>` or `<body>`

- ▶ You can place any number of scripts in an HTML document.
- ▶ Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.

JavaScript in <head>

- ▶ In this example, a JavaScript function is placed in the <head> section of an HTML page.
- ▶ Example of Javascript in head

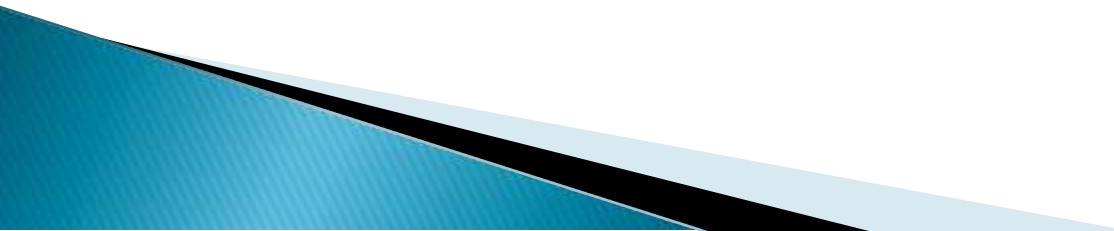
JavaScript in `<body>`

- ▶ In this example, a JavaScript function is placed in the `<body>` section of an HTML page.
 - ▶ Example
- 

External JavaScript

- ▶ Scripts can also be placed in external files: `myScript.js`
- ▶ To use an external script, put the name of the script file in the `src` (source) attribute of a `<script>` tag:
- ▶ `<script src="myScript.js"></script>`

JavaScript Display Possibilities

- ▶ Writing into the HTML output using **document.write()**.
 - ▶ Writing into an HTML element, using **innerHTML**.
 - ▶ Writing into an alert box, using **window.alert()**.
- 

JavaScript Functions

- ▶ A JavaScript function is a block of code designed to perform a particular task.

```
function myFunction()  
{  
    //Statements  
}
```

Declaring (Creating) JavaScript Variables

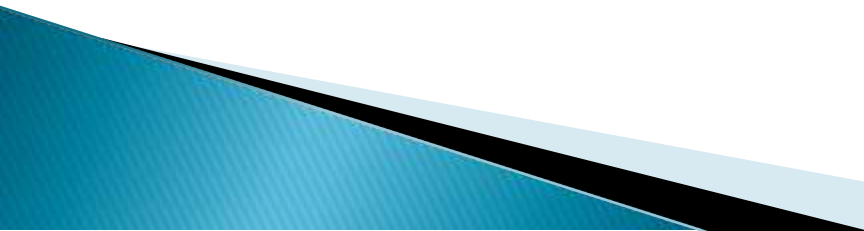
- ▶ Creating a variable in JavaScript is called "declaring" a variable.
- ▶ You declare a JavaScript variable with the **var** keyword:

```
var carName;
```
- ▶ After the declaration, the variable has no value. (Technically it has the value of **undefined**)

HTML Events

- ▶ An HTML event can be something the browser does, or something a user does.
- ▶ Here are some examples of HTML events:
 - An HTML web page has finished loading
 - An HTML input field was changed
 - An HTML button was clicked

Common HTML Events

- ▶ **onclick**
 - The user clicks an HTML element
 - ▶ **onmouseover**
 - The user moves the mouse over an HTML element
 - ▶ **onmouseout**
 - The user moves the mouse away from an HTML element
 - ▶ **onkeydown**
 - The user pushes a keyboard key
 - ▶ **onload**
 - The browser has finished loading the page
- 

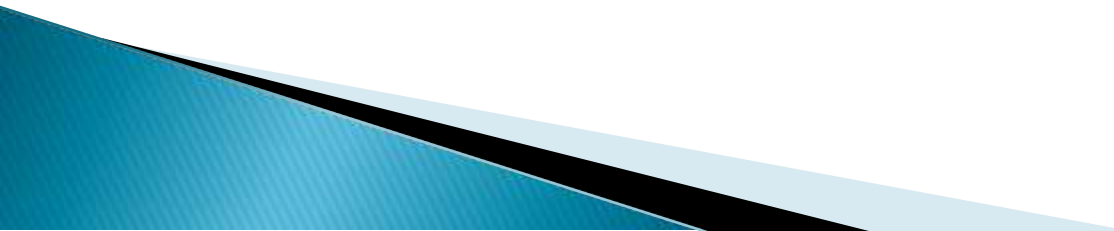
String Properties in JS

- ▶ String Properties
- ▶ length – The **length** property returns the length of a string
- ▶ Methods:
 - **toLowerCase()**
 - **toUpperCase()**
 - **charAt(x)**
 - **indexOf(substr, [start])**
 - **lastIndexOf(substr, [start])**
 - **substr(start, [length])**
 - **concat(v1, v2,...)**

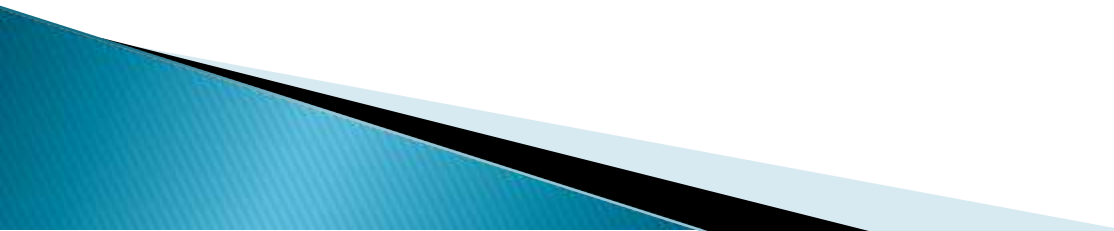
Operators of JS

- ▶ Arithmetic Operators : +, -, *, /, %, ++, --
- ▶ Logical Operators : &&, ||, !
- ▶ Comparison Operators : ==, !=, <, >, <=, >=

CONDITIONAL STATEMENTS

- Conditional statements are used to perform different actions based on different conditions.
 - The conditional statement will either return TRUE or FALSE.
- 

CONDITIONAL STATEMENTS

- JavaScript supports two conditional statements:
 - If...Else Statement
 - Switch Statement.
- 

If...Else STATEMENTS

- The **if statement** executes a statement if a specified condition is true.
- If the condition is false, else part can be executed.
- Syntax

```
If(condition){
```

```
    block of code to be executed if the condition is true
```

```
}
```

```
else{
```


```
    block of code to be executed if the condition is false
```

```
}
```



If...Else STATEMENTS

```
If(condition1){  
    statement1  
}  
else if(condition2) {  
    statement2  
}  
else if(condition n) {  
    statement3  
}  
else  
    statement4  
}
```



If...Else STATEMENTS

<html>

<head> <title> If - else if - else in JS </title> </head>

<body>

<p> An example of nested if else</p>

<script>

```
var d = new Date();
var d1 = ["Sun", "Mon", "Tues", "wed", "thurs", "fri", "sat"];
var d2 = d1[d.getDay()];
document.write(d2);
document.write("<br><br>");
if (d2 == "Mon") {
    document.write("Week has just started! Keep High Energy!");
}
else if (d2 == "Tues") {
    document.write("2nd day of Week! Keep High Energy!");
}
else if (d2 == "wed") {
    document.write("3rd day of Week! Take good food.. still 2 days to weekend");
}
else if (d2 == "thurs") {
    document.write("A fast day! May Goddess Saraswati and Lord Vishnu always bless
us!!");
}
else if (d2 == "fri") {
    document.write("Weekend is soon to begin.. :) :) Enjoy and work with high spirit!")
}
else if(d2 == "sat") {
    document.write("Weekend has started! A party on saturday evening!")
}
else {
    document.write("Sunday is a Funday! Rock on beats!")
}
```

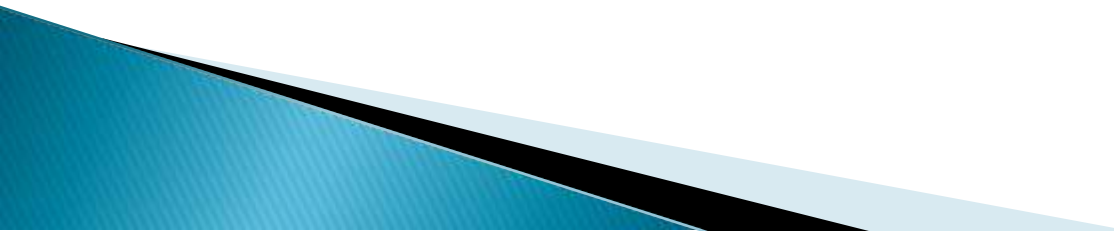
</script>

</body>

</html>

Switch Case

```
switch(expression) {  
  case a:  
    // code block  
    break;  
  case b:  
    // code block  
    break;  
  default:  
    // code block  
}
```



Switch Case

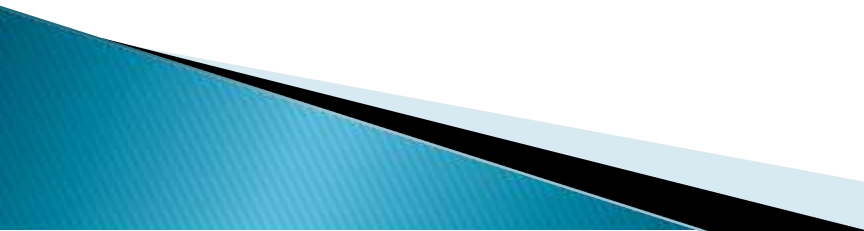
<script>

```
var d = new Date();
var day = ["Sun", "Mon", "Tues", "wed", "thurs", "fri", "sat"];
var day1 = day[d.getDay()];
switch(day1)
{
    case "Mon":
        document.writeln("Its a Monday");
        break;
    case "Tues":
        document.writeln("Its a Tuesday");
        break;
    case "wed":
        document.writeln("Its a Wednesday");
        break;
    case "thurs":
        document.writeln("Its a Thursday");
        break;
    case "fri":
        document.writeln("Its a Friday");
        break;
    case "sat":
        document.writeln("Its a Saturday");
        break;
    case "Sun":
        document.writeln("Its a Sunday");
        break;
}
```

</script>



Conditional Loops

- ▶ Very often when you write code, you want the same block of code to run a number of times.
 - ▶ You can use looping statements in your code to do this.
 - ▶ In JavaScript we have the following looping statements:
 - **while** - loops through a block of code while a condition is true
 - **do...while** - loops through a block of code once, and then repeats the loop while a condition is true
 - **for** - run statements a specified number of times
- 

While Condition

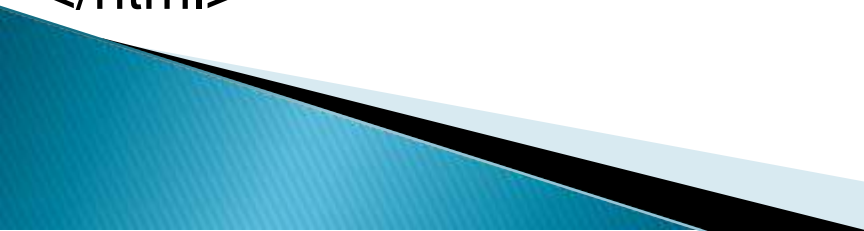
- ▶ The while statement will execute a block of code while a condition is true..

- ▶ Syntax:

```
while (condition)
{
    code to be executed
}
```

While Condition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript While Loop</title>
</head>
<body>
  <script>
    var i = 1;
    while(i <= 5) {
      document.write("<p>The number is " + i + "</p>");
      i++;
    }
  </script>
</body>
</html>
```



Do ... While Condition

- ▶ The do...while statement will execute a block of code once, and then it will repeat the loop while a condition is true

- ▶ Syntax:

```
do
{
    code to be executed
}
while (condition)
```

Do ... While Condition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Do-While Loop</title>
</head>
<body>
  <script>
    var i = 1;
    do {
      document.write("<p>The number is " + i + "</p>");
      i++;
    }
    while(i <= 5);
  </script>
</body>
</html>
```

For Condition

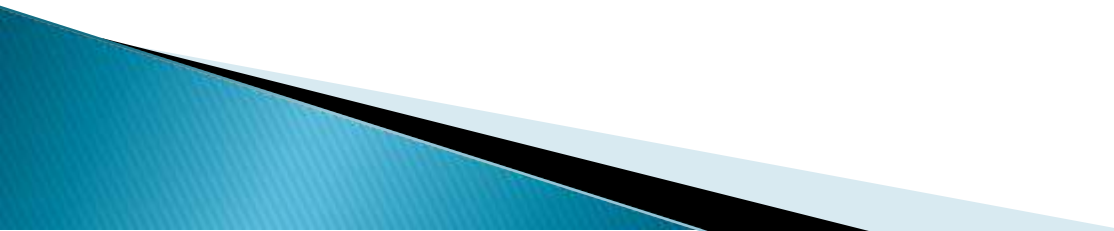
- ▶ The for statement will execute a block of code a specified number of times

- ▶ Syntax:

```
for (initialization; condition; increment)
{
    code to be executed
}
```

For Condition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript For Loop</title>
</head>
<body>
  <script>
    for(var i=1; i<=5; i++) {
      document.write("<p>The number is " + i + "</p>");
    }
  </script>
</body>
</html>
```



For Condition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Loop through an Array Using For-In Loop</title>
</head>
<body>
  <script>
    // An array with some elements
    var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];

    // Loop through all the elements in the array
    for(var i=0; i<fruits.length; i++) {
      document.write("<p>" + fruits[i] + "</p>");
    }
  </script>
</body>
</html>
```


For Condition

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>JavaScript Iterate Over an Array Using For Loop</title>
</head>
<body>
  <script>
    // An object with some properties
    var person = {"name": "Clark", "surname": "Kent", "age": "36"};

    // Loop through all the properties in the object
    for(var prop in person) {
      document.write("<p>" + prop + " = " + person[prop] + "</p>");
    }
  </script>
</body>
</html>
```