**JAVASCRIPT**  
  
ECMA Script- not a language

ECMA-Provides version updates for JS

1993-Mosaic Web Browser -1st GUI Browser

1994-Netscape Org-Read only web pages

Java-beta-emerging-Sun Microsystems

Netscape+sun micro systems- New Scripting Language

Brendan Eich - Mocha - Livescript - Javascript

Dynamic websites.

1996-Microsoft - IE - JScript -CSS

Best Viewed On. browser conflict.

1996-Netscape approached ECMA - European Computer Manufacturers Association (ECMA)

Javascript name cannot be standardised because Sun Microsystems owns trademark

ECMA... Brendan Eich disliked.. ECMAScript

1997-ECMAScript first release : support of microsoft IE and netscape

1998- ECMAScript2

1999- ECMAScript3

2000- Windows IE popular - ECMAScript contribution halted meanwhile JScript developed

Since Microsoft contributions were largest - ECMAScript Stopped.

After 4 years. JScript Updates Declined

2004- Mozilla Firefox

2005- Mozilla joined ECMA - to standardize ECMAScript 4

Adobe ActionScript...with lot more features ECMAScript 4 Failed.

ECMAScript - OpenSource

2005- Jesse James Garrett - AJAX Asynchronous Javascript and XML,

Network Request - Fetch Data and Update in Background in Runtime.

;;;;;;;;;;;;;jquery...frameworks...;;;;;;;;;;;

ECMAScript,ActionScript,JScript,Opensource

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

2008-Chrome- V8 JavaScript Engine - JIT Compilation (just-in-time)

all browsers outdated.

Oslo Conference, Norway: Google, ECMA, Microsoft and fellow contributors

2009- ECMAScript5 standardised

2015-ECMAScript6 ES6- (all features left out in ES4) - ES2015

ES2021 ... Now

BUT ACTUALLY THAT IS WHAT WE CALL JAVASCRIPT.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

basic things to cover before proceeding:

//let/const //Destructuring

//Objects //Spread

//tihs //classes

//Arrow Funcs //Modules

ES2021 ... //replaceAll

//Private Methods//Private Accessors

//Promise.any//Logical Assignmen Operators

//Numeric Seperators//WeakRef and Finalizers

JS: can announce and refer variables and objects that dont exist,

interpreted/translated by a browser, error is only caught in runtime by browser.

Dynamically Typed. Extra Transpilation: Can run directly on browser.

TYPESCRIPT: oops, static type checking. like a Compiled and interpreted language,

that compiles it into JS (tsc) supports all ECMAScript versions.

tsconfig. allows to change behaviour of compiler.

optional static typing:

intelliSense

static reading

generics

+node.js .. ruby on rails

explicit: annotate the type of data prefixed with a colon

:any type , opt out of typecheck/ loose type

if var and value declared without explicit declaration of type,

the type of data is declared implicitly.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TYPESCRIPT: Basarat Ali Syed: TypeScript Deep Dive  
/////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////

**DOM  
Simple Start**

**Case A:**The Onclick HTML attribute is used here to execute the javascript statement,

Document.getElementById(‘an\_id\_of\_an\_element’) 🡪 locates the element by its id.

Document.getElementById(‘an\_id\_of\_an\_element’).innerHTML 🡪The HTML content within the element is accessed.

Document.getElementById(‘an\_id\_of\_an\_element’).innerHTML = ‘any content’ 🡪 Assigning new values to the element’s attribute.

    <div *id*="div1"> <!--CHANGING CONTENT-->

        Playing with JavaScript

    </div>

    <button *onclick*="document.getElementById('div1').innerHTML='Hello World'">H</button>

    <button *onclick*="document.getElementById('div1').innerHTML='Playing with JavaScript'">P</button>

**Case B:**

Here the href attribute of the element is accessed and its value is changed upon onclick.

    <div> <!--CHANGING ATTRIBUTE :BUTTON WITH LINK-->

        <a *id*="a1" *href*=https://www.google.co.in> <button>Search Engine</button> </a>

    </div>

    <button *onclick*="document.getElementById('a1').href='https*:*//www.bing.com'">Bing</button>

    <button *onclick*="document.getElementById('a1').href='https*:*//www.google.co.in'">Google</button>

**Case C:**

Document.getElementById(‘an\_id\_of\_an\_element’).style 🡪 the style attribute of the element is accessed

Document.getElementById(‘an\_id\_of\_an\_element’).style

    <div *id*="div2" *style*="font-family: magneto;color: coral;"> <!--CONTENT WITH PROPERTIES-->

        GOBLIN

    </div>

    <button *onclick*="document.getElementById('div2').style='font-family: Ravie;color: green;font-size:200px'">Green</button>

    <button *onclick*="document.getElementById('div2').style='font-family: magneto;color: coral;'">Orange</button>

    <button *onclick*="document.getElementById('div2').style.color='yellow'">yellow</button>

    <button *onclick*="document.getElementById('div2').style.fontFamily='monospace'">text</button>

    <button *onclick*="document.getElementById('div2').style.fontSize='20px'">size</button>

**Case D:**

    <div> <!--IMAGE WITH HEIGHT AND WIDTH-->

        <img *id*="img1" *src*="D:\Dinesh\WEB\JS\800.png" *alt*="smiles" *height*="10%" *width*="10%">

    </div>

    <button *onclick*="document.getElementById('img1').src='801.png'">up</button>

    <button *onclick*="document.getElementById('img1').src='800.png'">down</button>

**Case E:**

    <button *onclick*="document.getElementById('div3').style.display='none'"> Hide</button>

**PASSING SCRIPTS:**Scripts can be passed inside the HTML or by attaching the Script page, below are a few examples.

Script inside the Head tag

<head>

    <meta *charset*="UTF-8">

    <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title *id*="title\_a">Script Tags</title>

    <script>

        document.getElementById('title\_a').innerHTML='Horses'

        console.log('The Script in my head')

    </script> <!--Passing Javascript in Script tag in head of the HTML-->

</head>

Script inside the Body tag

<body>

    <div *id*="div\_a">

    </div>

<br> <br>

    <div *id*="div\_b">

        <button *onclick*='print\_as\_pdf()'> ON/OFF </button>  <!--Calling a function that is defined below in a script tag-->

    </div>

<br> <br>

    <div *id*="div\_b">

        <input *type*="text" *style*="background-color: green;color: black;" *placeholder*="GREEN">

    </div>

<br> <br>

<script>

    document.getElementById('div\_a').innerHTML='Good Evening'; //innerHTML: Script to display data inside a tag

    document.write('Ohayou Gozaimasu Jscript San') //write method: directly displays an output in html,

        //calling write after HTML is loaded clears all HTML and calls write, dangerous used only for testing

    alert('This is a message box using Javascript') //Display pop notification, can be used with window.alert

    console.log('The Script in my body'); //Displays data in console, Debugging

    function print\_as\_pdf() //Calling this function prints the entire page

        { window.print() }

</script> <!--Passing script tag inside Body of the HTML-->

<script *src*="mojo.js">

</script> <!--Passing script with external js file-->

<!--------------------------------------------------------------------------------------------------------->

</body>

</html>