# JavaScript

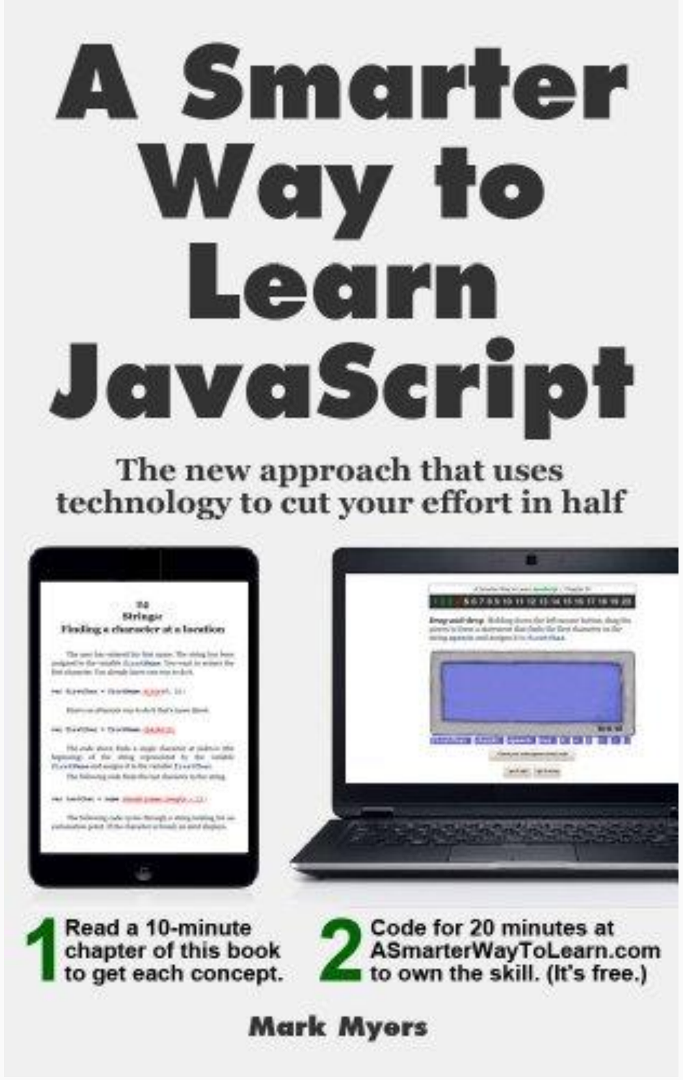
## Zeeshan Hanif

Director/CTO Panacloud
PIAIC

github.com/zeeshanhanif

linkedin.com/in/zeeshanhanif

# Book we will follow


A Smarter Way to Learn JavaScript
The new approach that uses technology to cut your effort in half
1 Read a 10-minute chapter of this book to get each concept.
2 Code for 20 minutes at ASmarterWayToLearn.com to own the skill. (It's free.)
Mark Myers

## Online Exercises

Below link provide you online exercises

http://asmarterwaytolearn.com/js/index-of-exercises.html

## Why we use computers

1. Computers have become part of our lives. We all have different reasons for wanting or needing to use computers.
2. Computers can make our jobs become easier. They can be used for communication purposes (internet), to store and calculate data and to write up massive documents multiple times while only needing to write it up once

## Why we use computers

1. We can summarize the reasons for using computers in four words
   a. Efficiency
   b. Reliability
   c. Accuracy
   d. Communication

## Why we use computers
### Efficiency

Computers save time, labor and resources

1. Time
    a. Computers and computer-controlled machines work more quickly than people can.
2. Labor
    a. Computers have reduced the labor involved in mentally intensive tasks.

## Why we use computers
## Efficiency (cont..)

Computers save time, labor and resources

1. Resources
    a. Resources include everything from electrical power and production of materials to something as simple as paper used in an office.

## Why we use computers
### Accuracy

1. If the software is correct, the computer can perform the same tasks over and over with 100% accuracy.
2. Accuracy that is repeatable is essential in mass production on assembly lines

# Why we use computers
## Reliability

1. Computers can be relied on to do tasks accurately, without tiring or getting bored, complaining or asking for money.
2. People are using computer to do their jobs because human can get affected by many factors and cause them can't achieved their job .

## Why we use computers
### Communication (now)

1. The global reach and speed of computer-based communications, mean that distance is now only relevant when transferring physical goods. Communication is no longer restricted by time or distance.
2. You can share files, use the same desktop and even work on the same document simultaneously
3. Use VoIP software like Skype to make overseas calls at a fraction of the cost of a traditional telephone call

## Why we develop Software Application

## Why we develop Software Application

1. To solve our real world problem by converting it into computer program
2. We convert our real world scenario/situation into software application
3. So that it can be done more efficiently, reliably and accurately

|  |  |
|---|---|
| Real world Problem/Situation | Software Solution |
| Letter | Email |

| Real world Problem/Situation | Software Solution |
|---|---|
| Accounting Books | Excel/QuickBook |

| Real world Problem/Situation | Software Solution |
|---|---|
| Sorting/Arranging information | Few clicks in Excel or database |

| Real world Problem/Situation | Software Solution |
|---|---|
| Driving | Driverless Car |

| Real world Problem/Situation | Software Solution |
|---|---|
| Networking | Facebook |

## How do we communicate with computers

## How do we communicate with computer

1. Computer acts like our servent
2. It will do whatever we ask computer to do.
3. But the problem is computer don’t understand what we say
4. It does not understand our plain English language
5. Computer understands only 0s and 1s

## Programing Language

## We need Programming Language

1. We need a way to give instructions to computer
2. A programming language is a formal language, which comprises a set of instructions that produce various kinds of output.
3. We need a programming language in computer programming for the same reason we need a natural language in our everyday life: to communicate and simplify things.

## We need Programming Language

4. Imagine how you can write all of the instructions to solve real problem using just binary (0s and 1s)
5. Programming Language acts like bridge or translator between humans and computers
6. Programming Language is vocabulary and a collection of rules that command a computer, devices, applications to work according to the written codes.

## Types of Programming languages

Programming languages are broadly classified into two types

1. Low-level languages
2. High-level languages

## Low-level Languages

Easily understood by machines but not by humans

## High-level Languages

Easily interpreted by programmers but not machines

## Low-level Languages

Closer to the native language of a computer (binary), making them harder for programmers to understand.

## High-level Languages

Written in a form that is close to our human language, enabling to programmer to just focus on the problem being solved.

## Low-level Languages

Low level language are the language which are machine dependent

## High-level Languages

High level language are the language which are machine independent

## Low-level Languages

Execute with high speed

## High-level Languages

Execute slower than lower level languages because they require a translator program

## Low-level Languages

Difficult to write, read, modify, debug and understand

## High-level Languages

Easy to write, read, modify, debug and understand

## Low-level Languages

Program written in low-level language is called a machine code

## High-level Languages

Program written in high-level language is called a source code

## Low-level Languages

Example

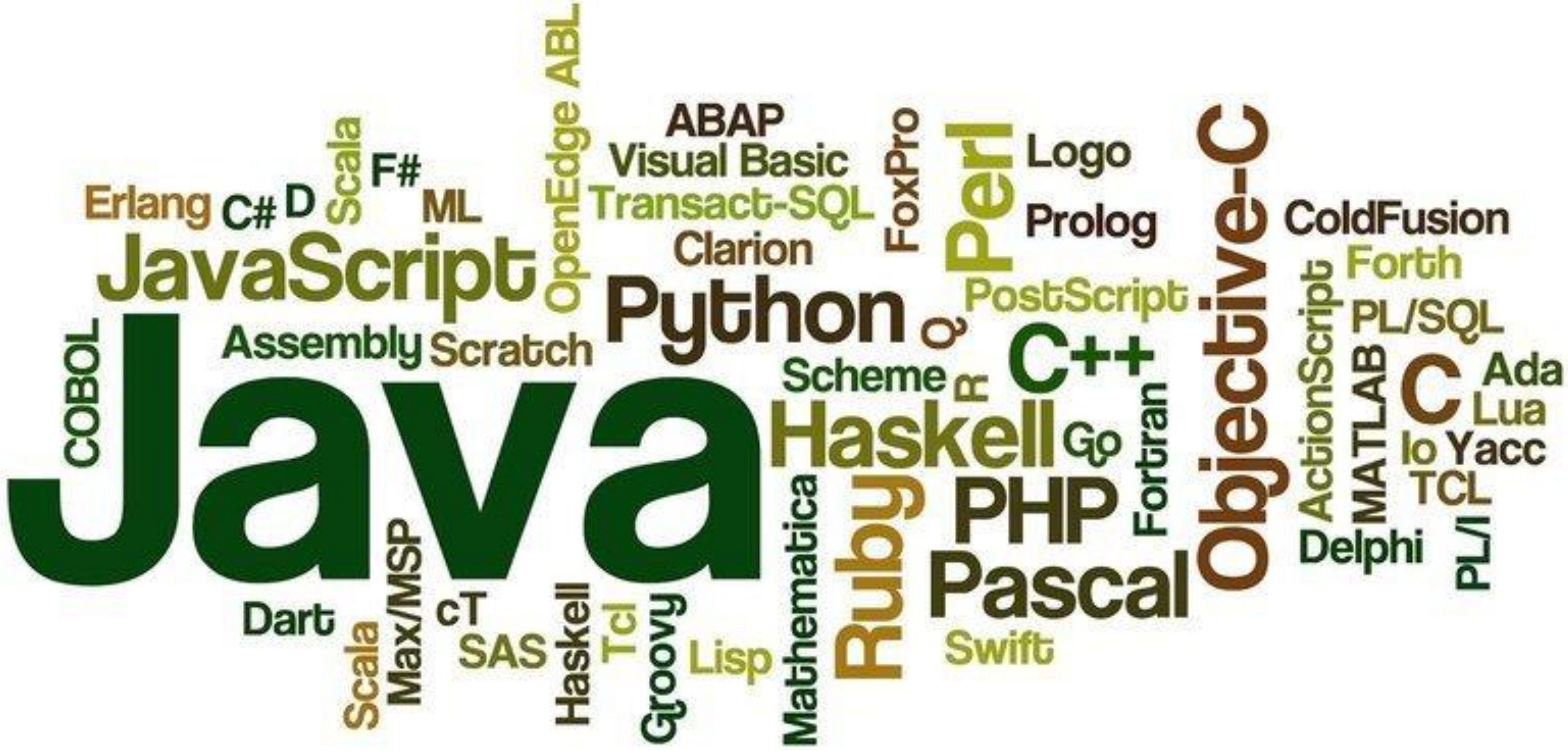
Assembly Language

Machine Code

## High-level Languages

Example

C++, Java, Pascal, Python, Visual Basic, Javascript


Java
JavaScript
Python
Haskell
Objective-C
PHP
Pascal
Ruby
Perl
C++
C
Erlang
C#
D
Scala
F#
ML
OpenEdge ABL
ABAP
Visual Basic
Transact-SQL
Clarion
FoxPro
Logo
Prolog
ColdFusion
Forth
PostScript
PL/SQL
Ada
Lua
Io
Yacc
TCL
Delphi
PL/I
MATLAB
ActionScript
Fortran
Go
Q
R
Scheme
COBOL
Assembly
Scratch
Dart
Scala
Max/MSP
cT
SAS
Haskell
Tcl
Groovy
Lisp
Mathematica
Swift

High level languages will be converted into a machine readable form by a compiler or interpreter

## Compiler vs Interpreter

Compiler and interpreter are the types of language translator.

Compiler and interpreter are programs that converts program written in high-level language into machine code understood by the computer.

## Compiler

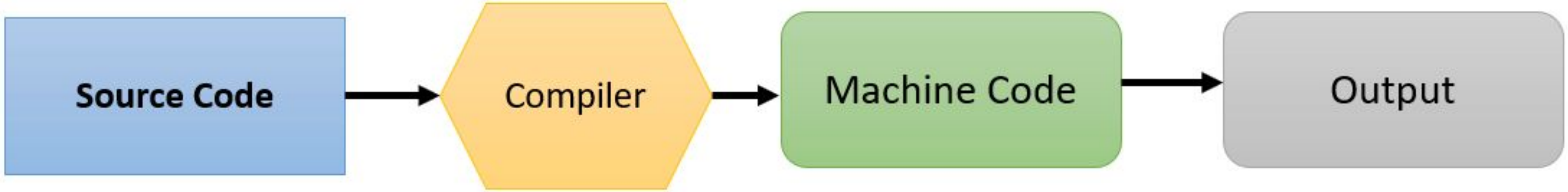
1. Compiler converts the whole program in one go
2. It generates intermediate object code.
3. It generates the error message only after scanning the whole program.
4. Program execution is separate from the compilation. It performed only after the entire output program is compiled.
5. Execution is comparatively faster than interpreter.

# Interpreter

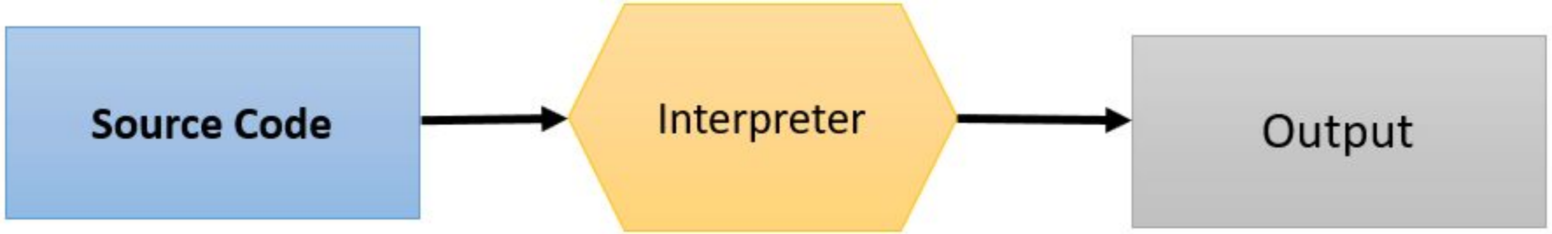
1. Interpreter converts the program by taking a single line at a time
2. It does not produce any intermediate object code.
3. Continues translating the program until the first error is met.
4. Program execution is a part of interpretation process, so it is performed line by line.
5. Execution is comparatively slower than compiler

## Compiler vs Interpreter Syntax Checking

## How Compiler Works

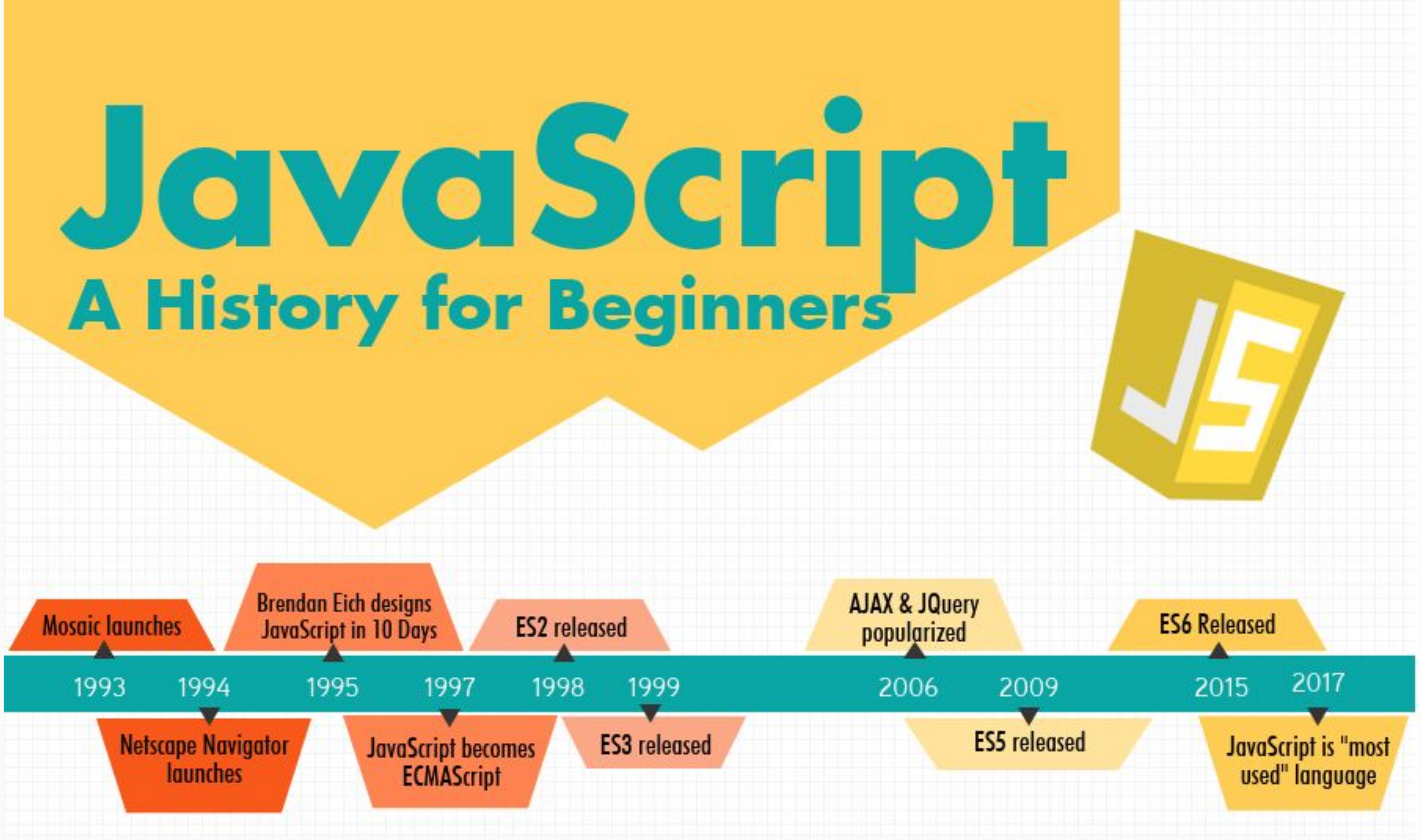


## How Interpreter Works



## History of JavaScript

## It All Began in the 90s


JavaScript
A History for Beginners
JS
1993 Mosaic launches
1994 Netscape Navigator launches
1995 Brendan Eich designs JavaScript in 10 Days
1997 JavaScript becomes ECMAScript
1998 ES2 released
1999 ES3 released
2006 AJAX & JQuery popularized
2009 ES5 released
2015 ES6 Released
2017 JavaScript is "most used" language

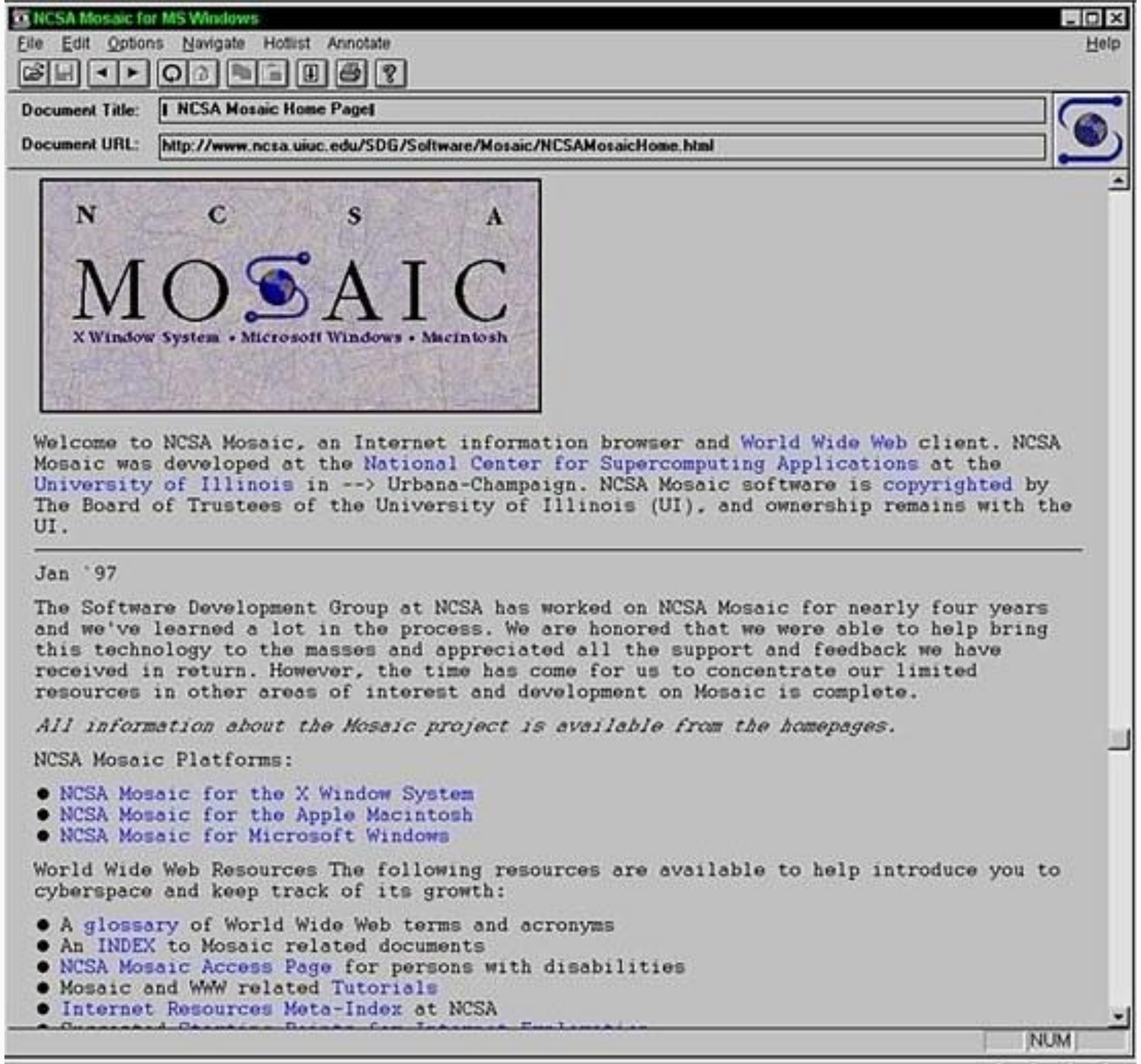
Source: https://www.coursereport.com/blog/history-of-javascript

JavaScript is arguably one of the most important languages today.

It is the de facto language of the internet

# 1993: NCSA Mosaic

1. In 1993, National Center for Supercomputing Applications at the University of Illinois to launch Mosaic
2. It is first popular user-friendly, graphical web browser.
3. The Internet was evolving from being a text-based to a multimedia universe for mainstream computer users.


NCSA Mosaic for MS Windows
File Edit Options Navigate Hotlist Annotate Help
Document Title: NCSA Mosaic Home Page
Document URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
N C S A
MOSAIC
X Window System • Microsoft Windows • Macintosh
NUM

Welcome to NCSA Mosaic, an Internet information browser and World Wide Web client. NCSA Mosaic was developed at the National Center for Supercomputing Applications at the University of Illinois in --> Urbana-Champaign. NCSA Mosaic software is copyrighted by The Board of Trustees of the University of Illinois (UI), and ownership remains with the UI.

---

Jan '97

The Software Development Group at NCSA has worked on NCSA Mosaic for nearly four years and we've learned a lot in the process. We are honored that we were able to help bring this technology to the masses and appreciated all the support and feedback we have received in return. However, the time has come for us to concentrate our limited resources in other areas of interest and development on Mosaic is complete.

*All information about the Mosaic project is available from the homepages.*

NCSA Mosaic Platforms:

- NCSA Mosaic for the X Window System
- NCSA Mosaic for the Apple Macintosh
- NCSA Mosaic for Microsoft Windows

World Wide Web Resources The following resources are available to help introduce you to cyberspace and keep track of its growth:

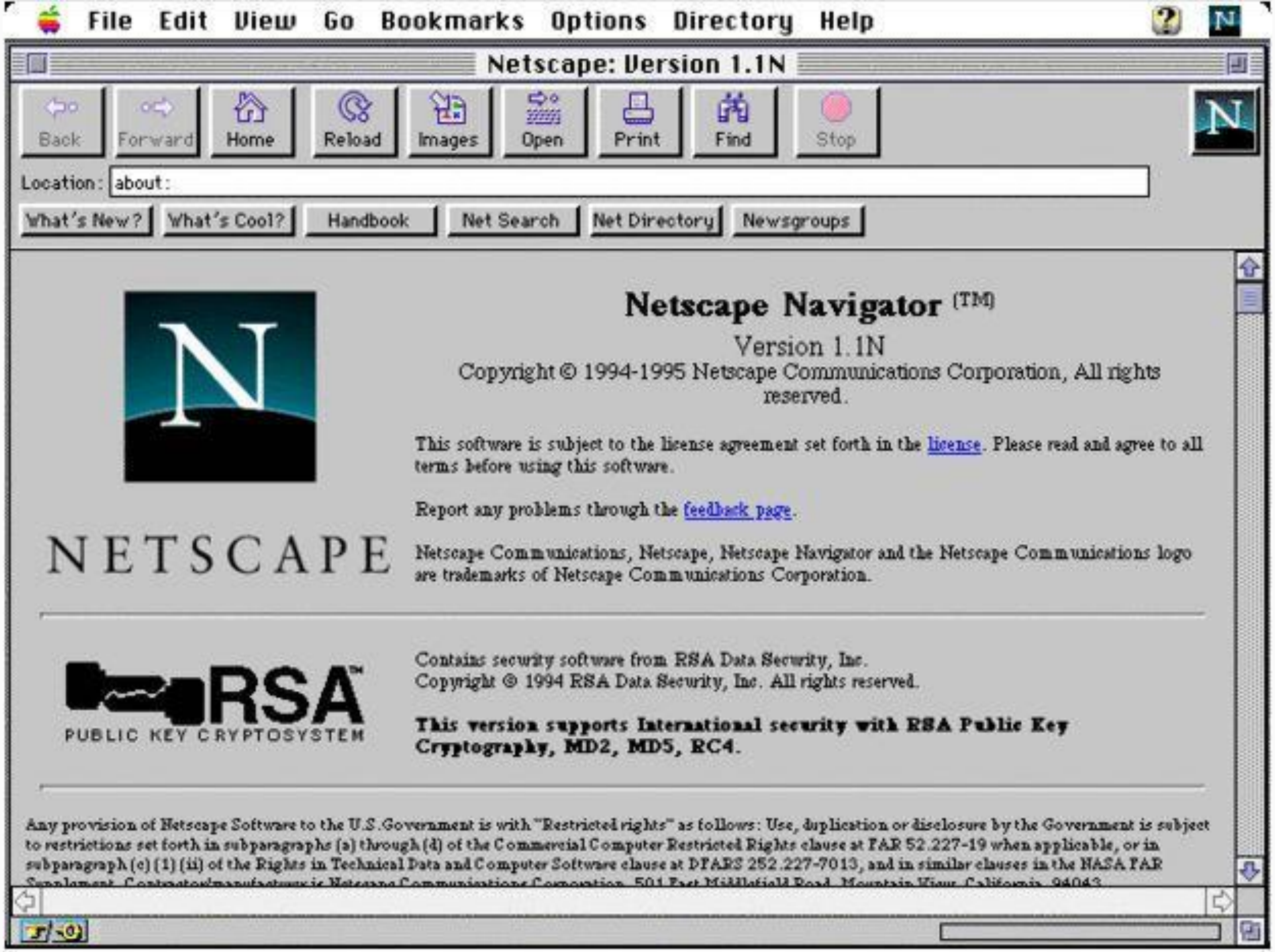
- A glossary of World Wide Web terms and acronyms
- An INDEX to Mosaic related documents
- NCSA Mosaic Access Page for persons with disabilities
- Mosaic and WWW related Tutorials
- Internet Resources Meta-Index at NCSA

## 1994: Netscape

1. On April 4, 1994 Mosaic Communications Corporation founded by Jim Clark who and Marc Andreessen
2. On October 13, 1994, it released web browser called Mosaic Netscape 0.9
3. It taken over the market and became the main browser for Internet users

# 1994: Netscape

1. To avoid trademark ownership problems with the NCSA, the browser was renamed to Netscape Navigator
2. And Company renamed to Netscape Communications Corporation



**Netscape Navigator** (TM)

Version 1.1N

Copyright © 1994-1995 Netscape Communications Corporation, All rights reserved.

This software is subject to the license agreement set forth in the license. Please read and agree to all terms before using this software.

Report any problems through the feedback page.

Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.

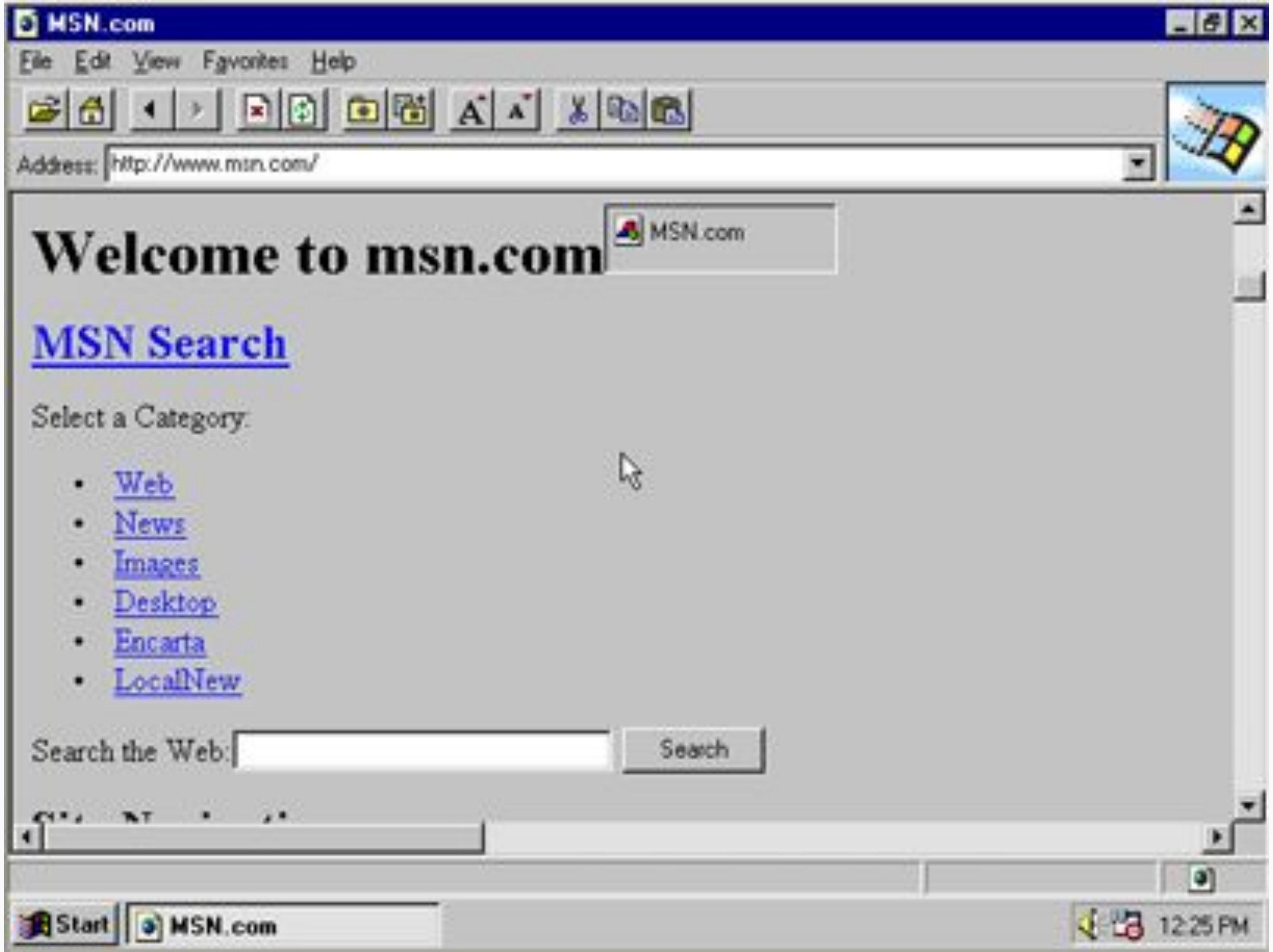
Contains security software from RSA Data Security, Inc.
Copyright © 1994 RSA Data Security, Inc. All rights reserved.

**This version supports International security with RSA Public Key Cryptography, MD2, MD5, RC4.**

Any provision of Netscape Software to the U.S.Government is with "Restricted rights" as follows: Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer Restricted Rights clause at FAR 52.227-19 when applicable, or in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Netscape Communications Corporation, 501 East Middlefield Road, Mountain View, California, 94043

## 1995: Internet Explorer

1. In August 1995 Microsoft released its graphical web browser Internet Explorer
2. And first Browser war begins
3. Later Microsoft won that first browser war



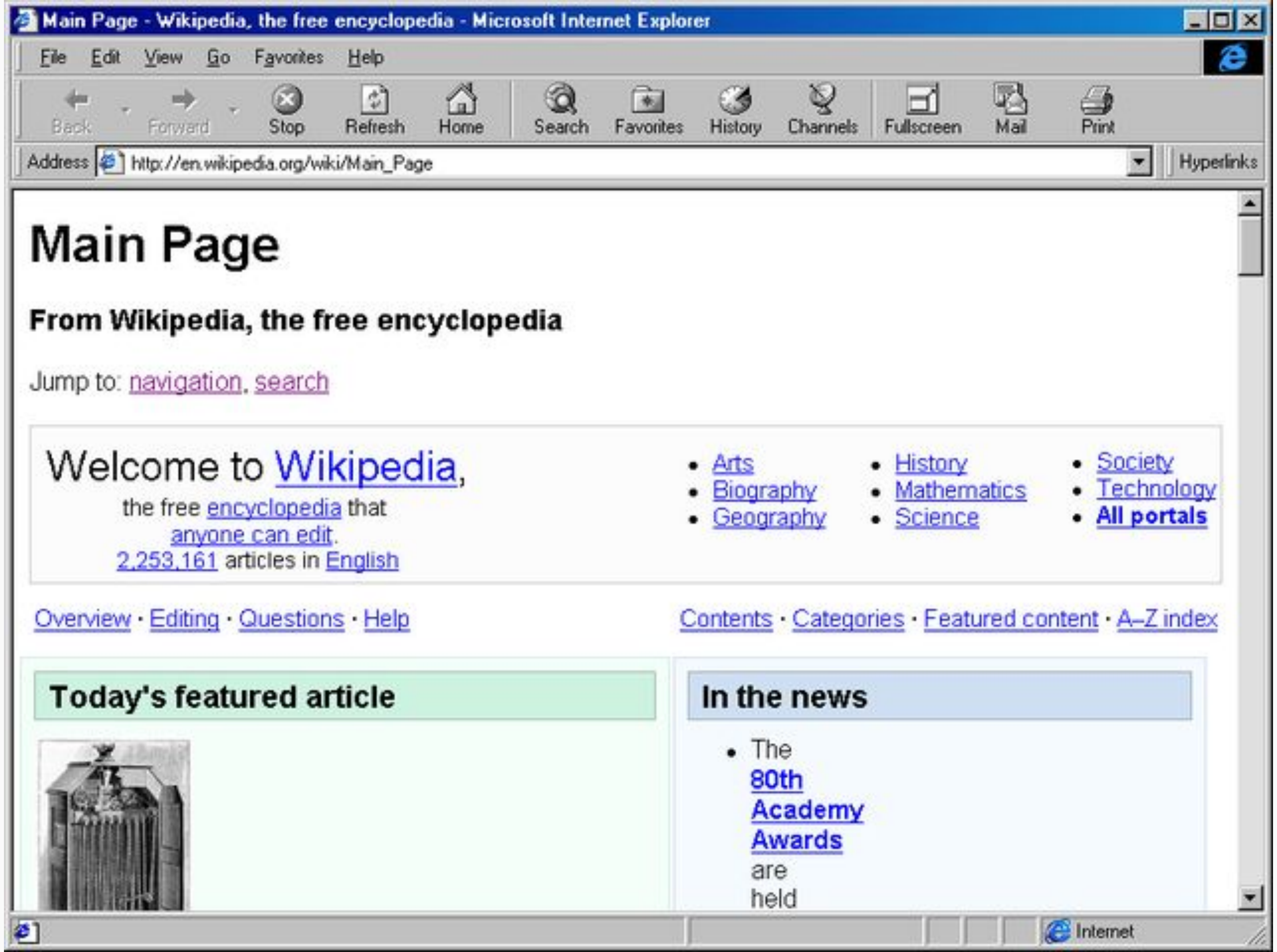
Welcome to msn.com

MSN Search

Select a Category:

- Web
- News
- Images
- Desktop
- Encarta
- LocalNew

Search the Web: Search


Main Page - Wikipedia, the free encyclopedia - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://en.wikipedia.org/wiki/Main_Page
Hyperlinks
Internet

Main Page

From Wikipedia, the free encyclopedia

Jump to: navigation, search

Welcome to Wikipedia,
the free encyclopedia that
anyone can edit.
2,253,161 articles in English

- Arts
- Biography
- Geography
- History
- Mathematics
- Science
- Society
- Technology
- **All portals**

Overview · Editing · Questions · Help

Contents · Categories · Featured content · A–Z index

**Today's featured article**

**In the news**

- The **80th Academy Awards** are held

## Marc Andreessen
Cofounder Netscape

1. He had the vision that the web needed a way to become more dynamic. Animations, interaction and other forms of small automation should be part of the web of the future
2. Web browsers needed to move beyond displaying static documents, and run truly interactive software

Web needed a small scripting language that could interact with the DOM

## Brendan Eich
Father of JavaScript

1. Marc Andreessen recruited Brendan Eich in 1995
2. He was given extreme short deadline to develop scripting language for web browser
3. HTML was still young and simple enough for non-developers to pick up.
4. So we needed language that should simple for non-programer

JavaScript was created by Brendan Eich in 1995 during his time at Netscape Communications. It was inspired by Java, Scheme and Self.

## 1995: Mocha To JavaScript

1. Eich was tasked to develop a "Scheme for the browser"
2. Meanwhile Netscape collaborated with Sun Microsystems to include Sun's programming language, Java, in Netscape Navigator to compete with Microsoft for user adoption of Web technologies and platforms.
3. Netscape decided that scripting language the wanted to create would complement Java and should have similar syntax.

## 1995: Mocha To JavaScript

4. To defend the idea of JavaScript and against the competing proposal, the company needed a prototype.
5. There was a lot of pressure to come up with a working prototype as soon as possible.
6. Eich wrote one in 10 days in May 1995

# 1995: Mocha To JavaScript

7. Eich had to work fast.
8. He had advantage: freedom to pick the right set of features
9. Unfortunately, he also had a big disadvantage: no time.
10. Lots of important decisions had to be made and very little time was available to make them.

# 1995: Mocha To JavaScript

11. Mocha was born in this context.
12. In a matter of weeks a working prototype was functional, and so it was integrated into Netscape Communicator.
13. In short time, it was renamed to **LiveScript** when it first shipped in beta releases of Netscape Navigator 2.0

## 1995: Mocha To JavaScript

15. In December 1995, Netscape Communications and Sun closed the deal:
    a. **Mocha/LiveScript** would be **renamed JavaScript**.
    b. It would be presented as a scripting language for small client-side tasks in the browser.
    c. While Java would be promoted as a bigger, professional tool to develop rich web components.

# 1995: Mocha To JavaScript

17. The choice of name was marketing ploy to take advantage of Java’s name for new Web Programming Language

## Different Implementations

## Different Implementations

1. From the very first days, JavaScript made such a considerable difference in user experience that competing browsers had no choice but to come up with a working solution, a working implementation of JavaScript.

## Different Implementations

3. Microsoft was compelled to reverse-engineer JavaScript support into Internet Explorer as "JScript" in 1996
4. Implementation of JScript was noticeably different from that found in Netscape Navigator at the time.
5. The features implemented by both, not always compatible, would define what would become of the web in the following years.

## Different Implementations

6. Differences in implementations made it difficult for designers and programmers to make a single website work well in both browsers, leading to the use of "best viewed in Netscape" and "best viewed in Internet Explorer" logos that characterized these early years of the browser wars.

# ECMAScript

## Standardization -- ECMAScript

1. Netscape realized that for an interactive, dynamic web to succeed, JavaScript would have to be consistent across browsers
2. In November 1996, Netscape submitted JavaScript to ECMA International to carve out a standard specification, which other browser vendors could then implement based on the work done at Netscape.

## Standardization -- ECMAScript

1. ECMA International given the ECMA-262 identification number for standard
2. Objective was to create standard specification which can be followed by all web browsers
3. It opened up JavaScript to a wider audience, and gave other potential implementers voice in the evolution of the language.

## Standardization -- ECMAScript

1. For trademark reasons, the ECMA committee was not able to use JavaScript as the name. So it was decided that the language described by the standard would be called ECMAScript.
2. It is generally referred as ES

## 1997: ECMAScript 1

1. In June 1997 first ECMAScript standard was release
2. And JavaScript was its well know implementation
3. Microsoft also followed the standard in its implementation of JScript

## 1998: ECMAScript 2

1. In June 1998 second version of the standard, ECMAScript 2, was released to fix inconsistencies between ECMA and the ISO standard for JavaScript (ISO/IEC 16262), so no changes to the language were part of it.

## 1999: ECMAScript 3

1. In December 1999 ECMAScript 3 was released
2. It was the first big change to the language
3. These changes shifted JavaScript from mere “scripting glue” to full-fledged language for applications
4. It was supported by all major web browsers at that time
5. ECMAScript 3 last longer and baseline for modern-day JavaScript

## 1999: ECMAScript 3 -- Some Features

1. Some of the features that were included in ECMAScript 3 are:
   a. Regular expressions
   b. The do-while block
   c. Exceptions and the try/catch blocks
   d. More built-in functions for strings and arrays
   e. Formatting for numeric output
   f. The in and instanceof operators
   g. Much better error handling

## 1999: The birth of AJAX

1. In 90s most of the Websites were based on complete HTML pages.
2. Each user action required that a complete new page be loaded from the server.
3. This process was inefficient, as reflected by the user experience: all page content disappeared, then the new page appeared. This placed additional load on the server and made bandwidth a limiting factor on performance.

## 1999: The birth of AJAX

4. In 1998, the Microsoft Outlook Web Access team developed the concept behind the XMLHttpRequest scripting object which shipped with Internet Explorer 5.0 in March 1999.
5. This function allowed a browser to perform an asynchronous (in background) HTTP request against a server, thus allowing pages to be updated on-the-fly.

## 1999: The birth of AJAX

6. This functionality was later implemented by all major browsers including Mozilla, Safari, Opera
7. This techniques enable JavaScript-powered websites to feel more like fast native apps and made possible to build rich applications like Gmail, Google Map etc and
8. The term Ajax was publicly used on 18 February 2005 by Jesse James Garrett in an article titled Ajax: A New Approach to Web Applications

## 1999-2008: ECMAScript 4 - Not Released

1. ECMAScript 4 was not able to come out of door.
2. As soon as work started on ES4, strong differences in the committee started to appear
3. There was a group of people that thought JavaScript needed features to become a stronger language for large-scale application development. This group proposed many features that were big in scope and in changes.

## 1999-2008: ECMAScript 4 - Not Released

4. Another group thought this was not the appropriate course for JavaScript.
5. The lack of consensus, and the complexity of some of the proposed features, pushed the release of ECMAScript 4 further and further away.
6. And in 2003 work on ECMAScript 4 stopped

# 1999-2008: ECMAScript 4 - Not Released

7. In 2005, the impact of AJAX and XMLHttpRequest sparked again the interest in a new version of JavaScript and committee resumed work.
8. Again concerns started to raise
9. Microsoft and Doug Crockford from Yahoo strongly oppose the ES4
10. With lots of fight and debets ES4 ended in 2008

## 2009: ES3.1, Later renamed ECMAScript 5

1. A working group led Microsoft and Yahoo started working ECMAScript 3.1
2. Objective was to implement simpler and reduced set of features with no syntax change only practical improvement
3. ES4 and ES3.1 group work side-by-side for sometime but finally everyone comes to a common ground ES3.1

## 2009: ES3.1 Later renamed ECMAScript 5

4. In 2008 ES4 come to an end and joint effort from all parties started on ES3.1
5. It was called “ECMAScript Harmony” Project
6. In 2009 ES3.1 completed and committee decided to rename it to ECMAScript 5 to avoid confusion
7. ECMAScript 5 was a modest improvement that helped JavaScript become a more usable language, for both small scripts, and bigger projects.

## 2009: ES3.1 Later renamed ECMAScript 5

8. ECMAScript 5 became one of the most supported versions of JavaScript
9. This long turmoil causes delay for Web to become common platform

## 2011: ECMAScript 5.1

1. In 2011 ECMAScript 5 saw another iteration in the form of ECMAScript 5.1
2. This release clarified some ambiguous points in the standard but didn't provide any new features.

## 2015: ES6 or ECMAScript 2015

1. The ECMAScript Harmony proposal became a hub for future improvements to JavaScript.
2. Many ideas from ECMAScript 4 were cancelled for good, but others were rehashed with a new mindset.
3. Classes, let, const, promises, generators, iterators, modules, etc. All these features meant to take JavaScript to a bigger audience

## 2015: ES6 or ECMAScript 2015

1. ECMAScript 6 was released in 2015
2. Later it was renamed to ECMAScript 2015

# ES2016 to ES2019

1. A new organized and streamline process, simplified the release process. Proposal which are ready can be included in each upcoming release
2. This also made it easy to release new features each year
3. ES7 or ES2016 released in year 2016
4. ES8 or ES2017 released in year 2017
5. ES9 or ES2018 released in year 2018
6. ES10 or ES2019 released in year 2019

## JavaScript Engines

## JavaScript Engines

1. A JavaScript engine is a computer program that executes JavaScript (JS) code.
2. The first JavaScript engines were mere interpreters, but all relevant modern engines utilize just-in-time compilation for improved performance.
3. The job of the JavaScript engine is to take your human language and turn it into something the machine understands.

# JavaScript Engines

4. JavaScript engines are typically developed by web browser vendors, and every major browser has one.
5. In a browser, the JavaScript engine runs in concert with the rendering engine via the Document Object Model.
6. After Chrome V8 engine, JavaScript engines is not limited to browsers. V8 engine is core component of the popular Node.js runtime system.

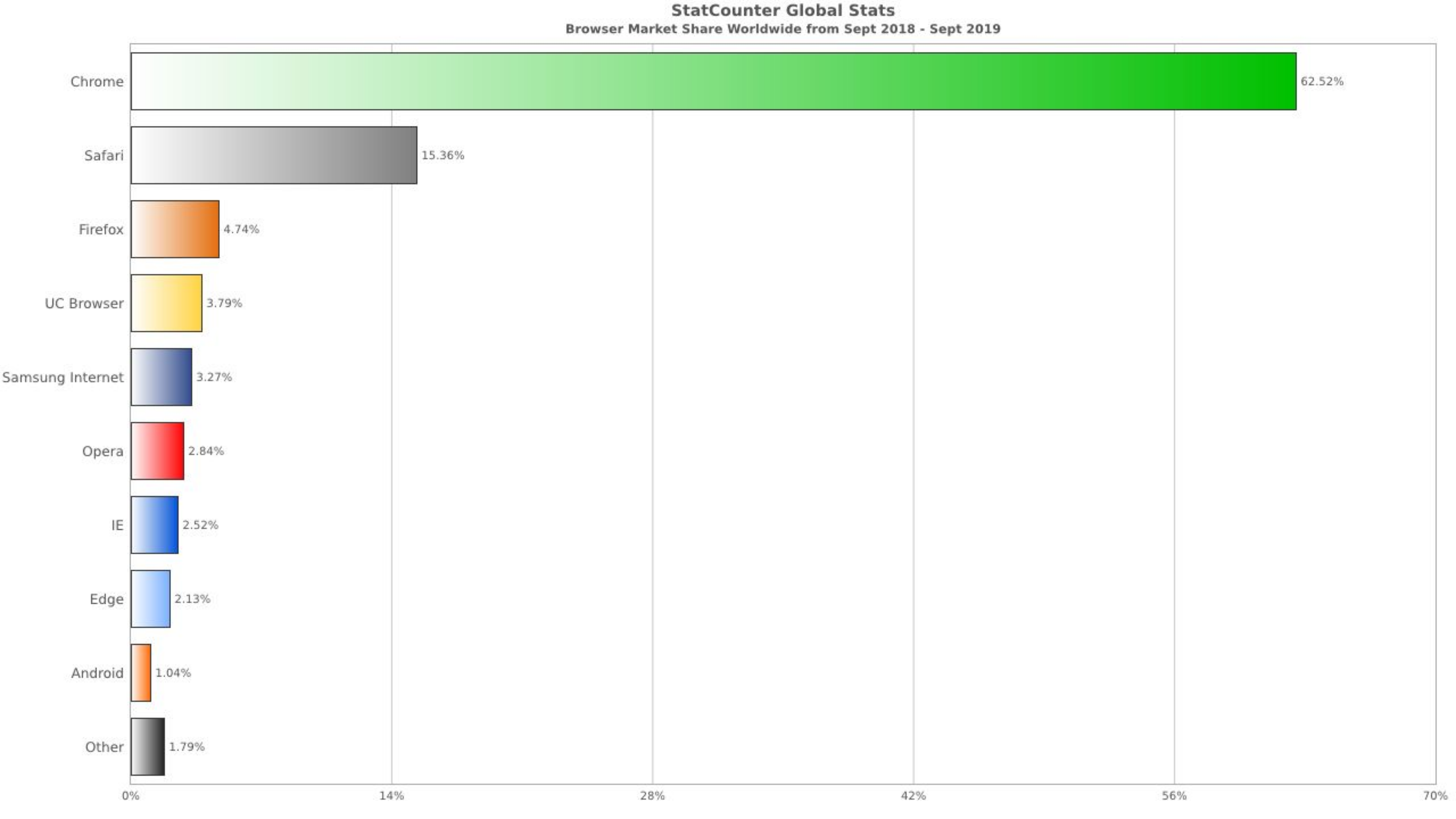
## JavaScript Engines

7. Since ECMAScript (ES) is the standardized specification of JavaScript, ECMAScript engine is another name for these engines.

# Few JavaScript Engines

| | |
|---|---|
| Chrome | Google Chrome uses V8 engine |
| Safari | Safari uses JavascriptCore also know as Nitro |
| Firefox | Firefox uses SpiderMonkey |
| Opera | Opera uses V8 engine |
| Edge | Edge uses Chakra engine and now moving to V8 |

## Browsers Market Share


StatCounter Global Stats
Browser Market Share Worldwide from Sept 2018 - Sept 2019
Chrome
62.52%
Safari
15.36%
Firefox
4.74%
UC Browser
3.79%
Samsung Internet
3.27%
Opera
2.84%
IE
2.52%
Edge
2.13%
Android
1.04%
Other
1.79%
0%
14%
28%
42%
56%
70%

## JS Libraries and Frameworks

prototype

jQuery

dōjō toolkit

mootools

ANGULAR

React

Vue.js

BACKBONE.JS

Knockout.

ember®

## 2005 to 2009: JS Libraries and Frameworks

1. In 2004, Google implemented a standardized version of AJAX technology on their Gmail and Google Maps products.
2. After Ajax JavaScript scene exploded, with new powerful libraries like jQuery which abstracted away browser inconsistencies and made it easier to apply design patterns.

## 2005 to 2009: JS Libraries and Frameworks

3. As ECMAScript standardization process was going through turmoil
4. So each browser was implementing its own set of features to give better experience, there were lot of inconsistencies in browsers
5. Even IE5, IE6 IE7 and IE8 had incompatibilities and developers had to apply checks for specific browser

## 2005 to 2009: JS Libraries and Frameworks

6. At that time there were lots of effort done by libraries like:
    a. Prototype
    b. jQuery
    c. Dojo
    d. Mootools
    e. And many others
7. jQuery outshine here and was the most popular JavaScript library for web interfaces

## 2005 to 2009: JS Libraries and Frameworks

8. **jQuery** is cross-platform JavaScript library designed to simplify the client-side scripting. It is free, open-source software.
9. jQuery is a lightweight, "write less, do more", JavaScript library.
10. 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.

## 2005 to 2009: JS Libraries and Frameworks

8. jQuery made is easier to write JavaScript code once and it will run well on all browsers and version.
9. jQuery handle that internally
10. jQuery became fantastically popular due to its ease of use
11. Ease of use does not necessarily result ease of maintenance.

## 2005 to 2009: JS Libraries and Frameworks

8. jQuery is unmaintainable, code can easily become spaghetti and grows in a monster-sized js file
9. Using jQuery it is very difficult to manage large and rich application, e.g Apps like Facebook, Gmail or YouTube are too complex to create
10. It lacked facilities for handling data consistently across shared views

2010 onwards

## Rise of Single Page Applications

## Single Page Applications

1. Because of Ajax and client side libraries most of the business logic shifted to client side, therefore there was need for library or framework that can manage client code in better way
2. Several frameworks were built to tackle the problem of data handling in shared views and code maintainability so that large application can be managed better

## Single Page Applications

3. Some of the earliest such JavaScript frameworks that quickly gained popularity
   a. Backbone
   b. Knockout
   c. Ember
4. AngularJS came into the market in October 2010. It quickly became the most popular JavaScript MVC framework.

## Single Page Applications

5. AngularJS offered two-way data binding, dependency injection, routing package and much more.
6. In May 2013 React was introduced and filled the gap created by AnguarJS, it was based on component based architecture
7. In Feb 2014 Vue.js was launched and has similarities to both Angular and React

## Single Page Applications

8. Then Angular 2 came out in Sep 2016 with complete redesign and based on component architecture
9. These libraries and framework with many others made client side application development rapid and efficient
10. It also gave life to JavaScript and made it popular language.

## Node.js

## 2008: Google Chrome

1. 2008 was another milestone year in the history of web.
2. In 2008, Google release:
   a. Open-source web browser Chromium
   b. Proprietary web browser Chrome
   c. Open-source V8 JavaScript engine

## 2008: Google Chrome

1. Most of Chrome's source code comes Chromium
2. Google made Chromium so that other can use it.
3. Google itself developed Chrome based on Chromium and then added additional features in chrome
4. V8 engine is also developed by Chromium Project for Google Chrome and Chromium browsers

## 2008: Google Chrome

1. V8 engine improved the performance of JavaScript and execution become very fast
2. V8 gets its speed from just-in-time (JIT) compilation of JavaScript to native machine code, just before executing it.
3. V8 engine is very fast and efficient and this made Chrome first choice for everyone.

## 2009: Node.js

1. The cool thing is that the JavaScript engine is independent by the browser in which it's hosted.
2. This key feature enabled the rise of Node.js.
3. V8 was chosen to be the engine that powered Node.js back in 2009.
4. As the popularity of Node.js exploded, V8 became the engine that now powers an incredible amount of server-side code written in JavaScript.

## 2009: Node.js

4. Node.js was written initially by Ryan Dahl in 2009, about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web.
5. The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent.

## 2009: Node.js

6. Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser.
7. Node.js lets developers use JavaScript to write command line tools and for server-side scripting
8. The Node.js ecosystem is huge and V8 also powers desktop apps, with projects like Electron.

## 2009: Node.js

9. This made JavaScript universal language that can be used for client-side as well as server-side development.
10. Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server- and client-side scripts.

Today JavaScript is most used language and it has advantage to learn single language that can be used on both client and server

## Moving on to Learn JavaScript

## Installation

1. To work with JavaScript you don’t need any software
2. You just need to have Browser installed on your machine and you can use any text editor to write code.
3. To manage projects and organize files we use IDE
4. We will use Microsoft’s Visual Studio Code
5. To run JavaScript on server side we will use Node.js

## Installation

1. Download and install Visual Studio code
    a. https://code.visualstudio.com

2. Download install Node.js
    a. https://nodejs.org/en/

3. Use default setting while installing both tools

Demo

How to use Visual Studio Code

## Initial Code and Setup

1. Open Visual Studio code and create two files
    a. index.html
    b. Index.js

# index.html file

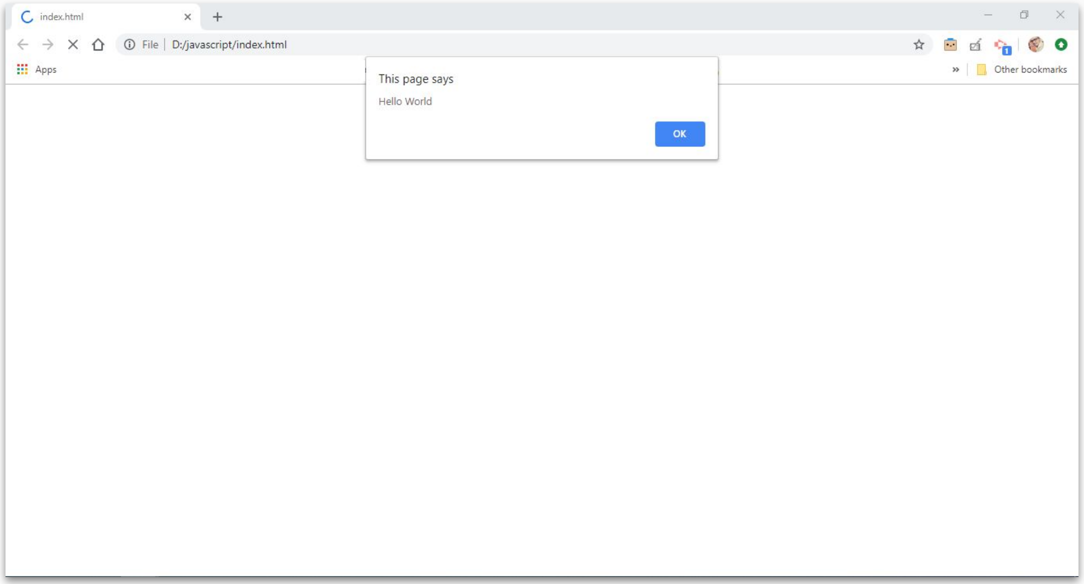
```
<html>
   <head>
       <script src="./index.js"></script>
   </head>
   <body>
       Hello World
   </body>
</html>
```

## index.js file

```
alert("Hello World");
```

# Initial Code and Setup

1. Open file in browser and you will see alert in browser window



This page says
Hello World

## Demo
## Create First JavaScript Example

## Demo
## How to run JavaScript in Node.js

## Alerts

1. An alert is a box that pops up to give the user a message.
2. Here's code for an alert that displays the message "Thanks for your input!"

```
alert("Hello World");
```

3. alert is a keyword
4. The quoted text "Thanks for your input!" is called a text string or simply a string.

## Alerts

1. ‘window.alert’ and ‘alert’ are the same
2. window is object in browser which contains many other objects and properties, and alert is one of them.
3. alert is a function that take any input and display it in popup to user
4. Alerts are not available when you are working with server side JavaScript in Node.js

## Console.log

1. console.log is function that write message on console/terminal
2. Objective of console.log is to create logs for debugging
3. Instead of displaying text to user it shows output in browser’s developer tool
4. Also when working with server side javascript, we can use console.log for logging and output will be in terminal

## Console.log

```
console.log("Hello World");
console.log(2+8);
```

# document.write

1. For testing purpose you can use document.write to display message or text in browser window

```
document.write("Hello Word");

document.write(2+8);
```

This will be displayed in browser window

## Variables

1. Variables are containers for storing data values.
2. Variables are used to store information to be referenced and manipulated in a computer program
3. Variable is a term that refers to a particular value
4. A variable changes over time or based on user input.
5. A variable is assigned a value in one place and then used repetitively.

## Variables

Keyword to create variable

Value

```
var nationality = "Pakistani";
```

Variable name/identifier

## Variables

```
var nationality = "Pakistani";

var age = 25;

var isFeePaid = true;

var weight = 60.55;
```

## Declaration and Initialization

1. You can declare and initialize in single line or you can do that in two line

```
var age = 25;   OR

var age;     -- Declaration
age = 25;    -- Initialization
```

Only declaration will leave variable *undefined*

# Data Types

1. Variables contains data
2. Each data has type, that’s why we call it data type
3. Variables can hold numbers like 100 and text values like "John Doe"
4. In programming, text values are called text strings.

## Data Types

1. Six types are considered to be primitives.
    a. Number -- integers, floats etc
    b. String -- an array of characters
    c. Boolean -- true or false
    d. Null -- No Value
    e. Undefined -- a declared variable but hasn't been given a value
    f. Symbol -- a unique value that's not equal to any other value

## Data Types

1. Complex Types
    a. Objects
    b. Functions

Function is callable object that executes a block of code, it lies under the object type

## Variable for String

1. To store text in variable we use string

```
var name = "Mark";
name = "John";
```

## String - Single quotes and double quotes

1. Text strings are always enclosed in quotes
2. You can use single or double quotes

```
var name = "Mark";
var nationality = 'US';
var message = "What is your father's number";
```

## Variable for Numbers

1. To store whole numbers or decimal values we use number data type

```
var age = 25;
var weight = 150.5;
var newWeight = weight + 20;
var originalNumber = 12 + 5;
```

## Variable for Boolean

1. To store true/false boolean data type

```
var isFeePaid = true;

var examPassed = false;
```

## Undefined

1. The undefined data type can only have one value-the special value *undefined*
2. If a variable has been declared, but has not been assigned a value, has the value undefined

```
var name;
var age;
```

## Undefined

1. You can be empty a variable by setting the value to undefined. The type will also be undefined

```
var name = "Mark";
name = undefined;
```

2. Now this variable contains no value, and it will cause error if we try to apply some operation on it

## Null

1. This is another special data type that can have only one value-the null value.
2. A null value means that there is no value.
3. It is not equivalent to an empty string ("") or 0, it is simply nothing

```
var name = null;
var nationality = "Mark";
nationality = null;
```

## Difference between null and undefined

1. undefined means a variable has been declared but has not yet been assigned a value.
2. null is an assignment value. It can be assigned to a variable as a representation of no value
3. undefined and null are two distinct types: undefined is a type itself (undefined) while null is an object.
4. Unassigned variables are initialized by JavaScript with a default value of undefined. JavaScript never sets a value to null. That must be done programmatically.

## JavaScript Data Types are Dynamic

1. JavaScript has dynamic types. This means that the same variable can be used to hold different data types
2. You can assign any type of value in variable any time and data type of variable will be changed

```
var name = "Mark";  // It's String
name = 25;  // Now changed to Number
name = true;  // Now changed to Boolean
```

## typeof Operator

1. You can use the JavaScript typeof operator to find the type of a JavaScript variable.
2. The typeof operator returns the type of a variable or an expression

```
typeof "Hello"          // Returns "string"
var a = 45;
typeof a;               // Returns "number"
typeof true;            // Returns "boolean"
```

## Statements

1. A computer program is a list of "instructions" to be "executed" by a computer.
2. In a programming language, these programming instructions are called statements.
3. A JavaScript program is a list of programming statements.
4. A statement can set a variable equal to a value.
5. A statement can also be a function call, i.e. document.write().

# Statements

1. Statements define what the script will do and how it will be done.
2. Most JavaScript programs contain many JavaScript statements.
3. The statements are executed, one by one, in the same order as they are written.

## Statements

1. Each line below is a statement

```
var a = 4;            // Statement 1
var b = 2;            // Statement 2
var c = 0;            // Statement 3
c = a + b;            // Statement 4
alert(a);             // Statement 4
console.log(c);       // Statement 4
```

# End of Statement with semicolon ;

1. To end statement add semicolon at the end of each executable statement
2. Semicolons separate JavaScript statements.

   ```
   var a = 4;
   ```

3. When separated by semicolons, multiple statements on one line are allowed

   ```
   i = 3; j = 5; k = i + j;
   alert(i);console.log(k);
   ```

## End of Statement without semicolon ;

1. Typically we end statements with semicolon but JavaScript semicolon is optional, but is highly recommended to end with semicolon
2. The end of a statement is most often marked by pressing enter and starting a new line.

# End of Statement without semicolon ;

```
var a = 5             // New line will end statement
a * 4
alert(a)
console.log(a)


var a = 5a * 4                // Error, Will Not work
alert(a)console.log(a)    // Error, Will not work
```

## Expressions

1. An expression is a combination of values, variables, function calls and operators, which computes to a value.
2. The computation is called an evaluation.

```
a + b;                       // expression
4 / 2;                       // expression
var a = 5;
a * 4;                       // expression
"John" + " " + "Doe";        // expression
```

## Comments

1. Comments are for the human, not the machine.
2. They help you and others understand your code when it comes time to revise, they make code more readable.
3. You can also use commenting to comment out portions of your code for testing and debugging.
4. They will prevent execution of code
5. There are two ways mark text as a comment, single line comments and multi-line comments

## Single line comments

1. Single line comments start with //.
2. Any text between // and the end of the line will be ignored (will not be executed).

```
// Declare and initialize variable
var a = 6; //This is comment


//This below code will not execute
//var b = 8;
```

## Multi-line comments

1. Multi-line comments start with /* and end with */.
2. Any text between /* and */ will be ignored.

```
/*
This code declared and initialize
variable a and show on screen
*/
var a = 6;
alert(a);
```

## Variable Names Legal and Illegal

1. A variable name can't contain any spaces
2. A variable name can contain only letters, numbers, dollar signs, and underscores.
3. The first character must be a letter, or an underscore (_), or a dollar sign ($).
4. Subsequent characters may be letters, digits, underscores, or dollar signs.
5. Numbers are not allowed as the first character of variable.

## Variable Names Legal and Illegal

1. Legal names:

```
var hello = 56;
var _xyz = 44;
var $work = 90;
var user2 = 56;
var i_info = 99;
var my$work = 77;
```

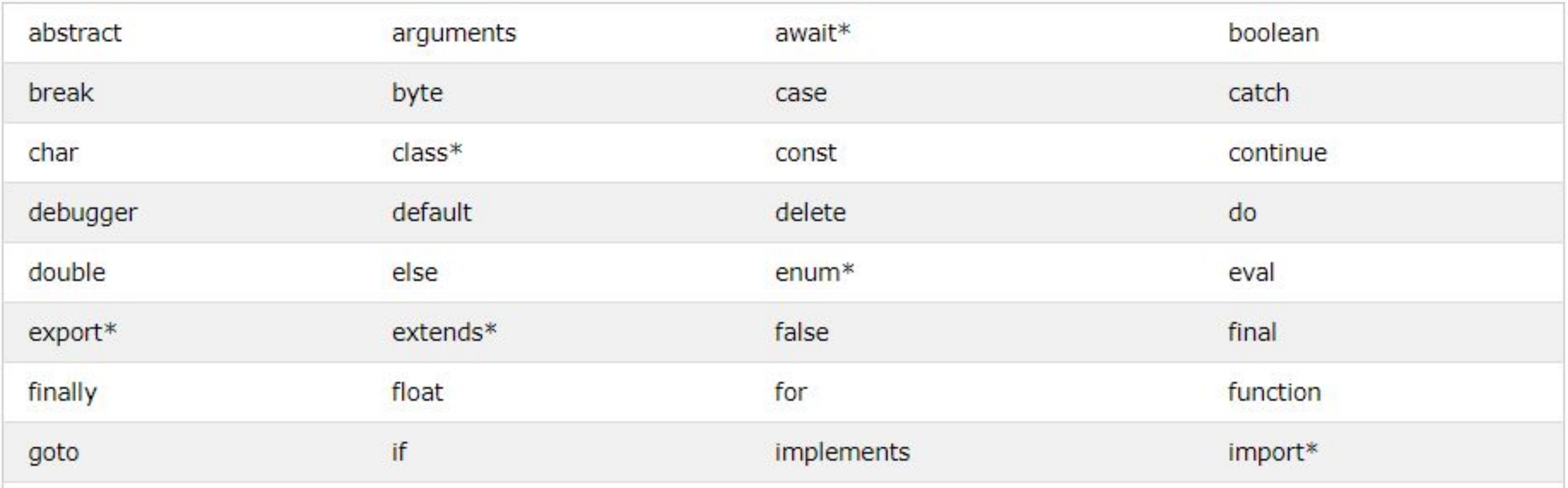
## Variable Names Legal and Illegal

1. Illegal names:

```
var 2user = 12;   // Can't start with number
var my user = 23; // Can't contains space
var hello#world = 34;
var my-info = 44;
var my?info = 45;
var my*info = 45;
```

# Reserved Keywords

1. Reserved Keywords cannot be used as variable name
2. Here are few reserved keywords



| | | | |
|---|---|---|---|
| abstract | arguments | await* | boolean |
| break | byte | case | catch |
| char | class* | const | continue |
| debugger | default | delete | do |
| double | else | enum* | eval |
| export* | extends* | false | final |
| finally | float | for | function |
| goto | if | implements | import* |

| | | | |
|---|---|---|---|
| abstract | arguments | await* | boolean |
| break | byte | case | catch |
| char | class* | const | continue |
| debugger | default | delete | do |
| double | else | enum* | eval |
| export* | extends* | false | final |
| finally | float | for | function |
| goto | if | implements | import* |
| in | instanceof | int | interface |
| let* | long | native | new |
| null | package | private | protected |
| public | return | short | static |
| super* | switch | synchronized | this |
| throw | throws | transient | true |
| try | typeof | var | void |
| volatile | while | with | yield |

## Case Sensitive

1. Variable names are case sensitive.
2. So **`rose`** and **`Rose`** are two different variables

```
var rose = "Hello";
var Rose = "Hello";
alert(rose);
alert(Rose);
alert(ROSE); // Error
```

## Camel Case

1. If there's more than one word in the variable name, then it is recommended to use camel case
2. A camelCase name begins in lower case. If there's more than one word in the name, each subsequent word gets an initial cap, creating a hump.

## Camel Case

```
var userResponse
var userResponseTime
var userResponseTimeLimit
var response
```

This Style of naming a variable is called camel case

## Arithmetic Operators

```
var a = 5;
var b = 3;
var c = a + b;         // Addition, result 8
var d = a - b;         // Subtraction, result 2
var e = a * b;         // Multiplication, result 15
var f = a / b;         // Division, result 1.66
var g = a % b;         // Modulus, result 2
var h = a ** b;        // Exponentiation, result 125
```

## Assignment Operators

1. Assignment operator assign value to variables
2. When you need to apply arithmetic operation and assign value to same variable then you can also use them

```
var a = 5; // equals = is assignment operator
a = a + 2; // Assign 7 in variable a
OR
var a = 5;
a+=2;       // Assign 7 in variable a
```

## Assignment Operators

| Example | Same as |
|---|---|
| `a += 5;` | `a = a + 5;` |
| `a -= 5;` | `a = a - 5;` |
| `a *= 5;` | `a = a * 5;` |
| `a /= 5;` | `a = a / 5;` |
| `a %= 5;` | `a = a % 5;` |
| `a **= 5;` | `a = a ** 5;` |

## Eliminating ambiguity -- BODMAS

1. Complex arithmetic expressions can pose a problem when there are multiple operators in single expression

```
var a = 5 + 2 * 3 - 2 / 2;  // result 10
```

2. The evaluation of above expression is depends on BODMAS rule
3.

## Eliminating ambiguity -- BODMAS

| | |
|---|---|
| B | Brackets first |
| O | Orders (i.e. Powers and Square Roots, etc.) |
| DM | Division and Multiplication (left-to-right) |
| AS | Addition and Subtraction (left-to-right) |

## Eliminating ambiguity -- BODMAS

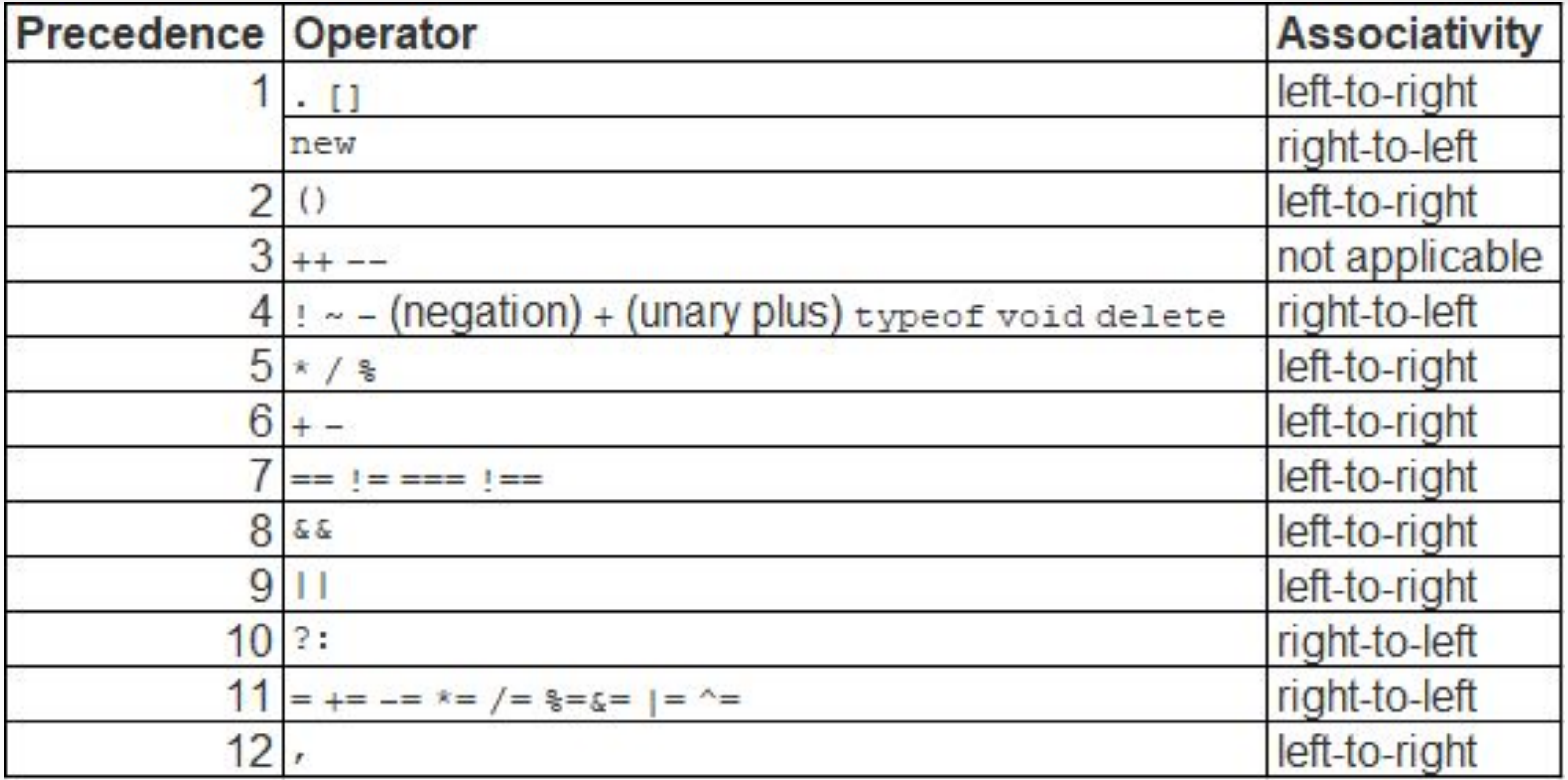
```
var a = 5 + 2 * (3 - 2) / 2; // result 6
```

1. 3 - 2 with brackets will be evaluated first, result 1
2. 2 * result of (3 - 2) so 2 * 1, result 2
3. Result of 2 * (3 -2) divide by 2 so 2 / 2, result 1
4. 5 + result of 2 * (3 - 2 ) / 2, so 5 + 1, result 6

## Eliminating ambiguity -- BODMAS

```
var a = 3 + 5 * 2;                    // result 13
var b = 8 / 2 - 1;                    // result 3
var c = 3 % 2 + 4 - 1;                // result 4
var d  = a + 5 * c - b / (3 + b);// result 32.5
```

# Operator Precedence -- Few of them



| Precedence | Operator | Associativity |
|---|---|---|
| 1 | `.` `[]` | left-to-right |
| | `new` | right-to-left |
| 2 | `()` | left-to-right |
| 3 | `++` `--` | not applicable |
| 4 | `!` `~` `-` (negation) `+` (unary plus) `typeof` `void` `delete` | right-to-left |
| 5 | `*` `/` `%` | left-to-right |
| 6 | `+` `-` | left-to-right |
| 7 | `==` `!=` `===` `!==` | left-to-right |
| 8 | `&&` | left-to-right |
| 9 | `\|\|` | left-to-right |
| 10 | `?:` | right-to-left |
| 11 | `=` `+=` `-=` `*=` `/=` `%=&=` `\|=` `^=` | right-to-left |
| 12 | `,` | left-to-right |

## Increment and Decrement Operator

1. While working on application you will frequently required to increase variable by 1 or decrease variable by 1
2. For this situation you can use increment and decrement operator
3. ++ increment operator
4. -- decrement operator
5. These operators can be used as prefix and postfix

## Increment and Decrement Operator

1. We can increase or decrease value using existing addition and subtraction operators

```
var a = 12;
a = a + 1;              // 13
a = a - 1;              // 12
var b = 12;
b += 1;                  // 13
b -= 1;                  // 13
```

## Increment and Decrement Operator

Prefix Increment and Decrement

```
var age = 12;
++age;
alert(age); //Result 13
--age;
alert(age); //Result 12
```

## Increment and Decrement Operator

Postfix Increment and Decrement

```
var age = 12;
age++;
alert(age); //Result 13, same as prefix
age--;
alert(age); //Result 12, same as postfix
```

## Increment and Decrement Operator

1. What is the difference between Prefix and Postfix
2. You will NOT find any difference if you will not assign result of prefix and postfix to any other variable
3. Prefix operator first increase/decrease the value in variable and then assign result to other variable
4. Postfix operator first assign the value in other value then increase/decrease value in actual variable

# Increment and Decrement Operator

Prefix Increment

```
var age = 12;
var newAge = ++age;
alert(age);        //Result 13
alert(newAge);     //Result 13
```

## Increment and Decrement Operator

Postfix Increment

```
var age = 12;
var newAge = age++;
alert(age);        //Result 13
alert(newAge);     //Result 12 , see the difference
```

## Increment and Decrement Operator

Prefix Decrement

```
var age = 12;
var newAge = --age;
alert(age);        //Result 11
alert(newAge);     //Result 11
```

## Increment and Decrement Operator

Postfix Decrement

```
var age = 12;
var newAge = age--;
alert(age);       //Result 11
alert(newAge);    //Result 12 , see the difference
```

## Increment and Decrement Operator

Example 1

```
var a = 4;
var b = 2;
var c = a++ - b; // first value of a placed here
which is 4 then increase 1 in a so will become 5
alert(a);           // 5
alert(b);           // 2
alert(c);           // 2
```

## Increment and Decrement Operator

Example 2

```
var a = 4;
var b = 2;
var c = ++a + b; // first value of a increased so
will become 5 then value of a will be placed here, 5
alert(a);          // 5
alert(b);          // 2
alert(c);          // 7
```

# Increment and Decrement Operator

Example 3

```
var a = 4;
var b = 3;
var c = a++ + --b - --a;
alert(a);          // 4
alert(b);          // 2
alert(c);          // 2
```

## Increment and Decrement Operator

Example 4

```
var a = 4;
var b = 3;
var c = ++a + b++ - a + --b;
alert(a);          // 5
alert(b);          // 3
alert(c);          // 6
```

## Concatenating Text strings

1. The + operator can also be used for concatenating strings
2. E.g:

```
var firstName = "Abraham";
var lastName = "Lincoln";
//concatenate firstName, space character and lastName
var fullName = firstName + " " + lastName;
alert(fullName);
```

## Concatenating strings and numbers

1. Adding two numbers, will return the sum, but adding a number and a string will return a string:

```
var a = "6" + 2;                // "62"
var b = 3 + "6";                // "36"
var c = "Hello " + 2;           // "Hello 2"
var d = 2 + "Hello ";           // "2 Hello"
var e = "Hello "+ 3 + 4;        // "Hello 34"
var f = 3 + 4 + "Hello ";       // "7Hello"
var g = "Hello "+ (3 + 4);      // "Hello 7"
```