Advance JavaScript

BccFalna.com
097994-55505

Kuldeep Chand

In this EBook, I have not only covered Simple Client Side Programming Concepts of JavaScript and Web Development but also various Advance Concepts like **Anonymous Functions, JavaScript OOPS, JSON, AJAX, Clousers,** etc…

After learning JavaScript, you can very easily move to various JavaScript Frameworks like **jQuery, Prototype**, etc… for fast and easy Client Side Development.

If you really want to be a Programmer as a Professional Developer, you will sure need to learn JavaScript because now each and everything is being developed on the basics of JavaScript.

Like HTML5, which is the latest technology for web development, have been divided in various parts for various kinds of tasks to fulfill and for fulfilling various kinds of requirements, we need to use HTML5 API like **Geo Location**, and that is available only in JavaScript API Format.

So for learning JavaScript Properly in easy to understand HINDI Language with hundreds of Example Programs, this is the only EBook for you. Just read and learn by fun.

Advance JavaScript In Hindi

■■■
Kuldeep Chand

BetaLab Computer Center
Falna

Advance JavaScript in Hindi

Lead Editors: **Kuldeep Chand**

Distributed to the book trade worldwide by BetaLab Computer Center, Behind of Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

e-mail *bccfalna@gmail.com*

or

visit *http://www.bccfalna.com*

For information on translations, please contact BetaLab Computer Center, Behind of Vidhya Jyoti School, Falna Station Dist. Pali (Raj.) Pin 306116

Phone **097994-55505**

This book is dedicated to those

who really wants to be

a

PROFESSIONAL DEVELOPER

INDEX OF CONTENTS

Table of Contents

JAVASCRIPT INTRODUCTION

JAVASCRIPT INTRODUCTION

किसी भी प्रकार की Programming Language में Program या Software Develop करते समय कई Basic Steps Follow करने होते हैं। लेकिन हमेंशा सबसे पहले हमें किसी Text Editor में अपनी Language से संबंधित Codes लिखकर कोई Program Create करना होता है। इस प्रकार के Codes को हम जिस File में लिखते हैं, उस File को **Source File** कहा जाता है, क्योंकि Program से संबंधित मूल Codes इसी Source File में होते हैं और यदि हमें हमारे Program में कोई Modification करना हो, तो हम वह Modification इसी Source File में करते हैं।

Source File केवल एक **Plain Text File** ही होती है, जिसमें हम हमारे समझने योग्य English Language में Programming Language से संबंधित Codes लिखते हैं। लेकिन Computer एक Electronic Machine मात्र है, जो हिन्दी, अंग्रेजी, Chinese जैसी उन भाषाओं को नहीं समझता जिन्हें हम Human Beings Real Life में समझते हैं, बल्कि वह केवल Binary Language या अन्य शब्दों में कहें तो Machine Language को ही समझता है। जबकि परेशानी ये है कि हम Human Beings Computer की Machine Language को आसानी से नहीं समझ सकते।

इस स्थिति में एक ऐसे Inter-Mediator की जरूरत होती है, जो हमारी English जैसी भाषा में लिखे गए Codes को Computer के समझने योग्य Machine Language में Convert कर सके और Computer द्वारा हमारे Program के आधार पर Generate होने वाले Output या Result को हमारे समझने योग्य English जैसी भाषा में Convert कर सके। इस प्रकार के Inter-mediator को Computer की भाषा में **Compiler** या **Interpreter** कहते हैं।

Compiler व Interpreter दोनों ही एक प्रकार के **Software** मात्र होते हैं, लेकिन इनका मूल काम हमारे Program के Codes को Computer के समझने योग्य मशीनी भाषा में और मशीनी भाषा में Generate होने वाले Results को हमारे समझने योग्य English जैसी भाषा में Convert करना होता है। इस प्रकार से Programming की दुनियां में मूल रूप से दो प्रकार की Programming Languages हैं:

1 पहले प्रकार की Programming Languages को Compiler Based Programming Languages कहते हैं, जिसके अन्तर्गत “C”, “C++” जैसी Languages आती हैं। इस प्रकार की Languages की मूल विशेषता ये है कि इस प्रकार की Programming Languages में हम जो Program Create करते हैं, उन्हें Compile करने पर वे Program पूरी तरह से **Machine Codes** में Convert हो जाते हैं, जिन्हें हमारा Computer Directly Run करता है।

   Compiler Based Programming Languages की मूल विशेषता ये होती है कि जब हम हमारे किसी Program को उसके Compiler द्वारा Compile कर लेते हैं, तो एक नई Executable File बनती है, जिसमें केवल Computer के समझने योग्य Machine Codes होते हैं और इस File को Run करने के लिए अब हमें हमारी Source File की जरूरत नहीं रहती।

   ये Executable File पूरी तरह से Current Computer Architecture व Operating System पर आधारित होती है। यानी यदि हम किसी Program को उस Computer पर Compile करें जिस पर **Windows** Operating System Run हो रहा हो, और Generate होने वाली Executable File को हम किसी दूसरे ऐसे Computer पर Run करने की कोशिश करें, जिस पर **Linux** Operating System हो, तो हमारा Program Linux Operating System पर Run नहीं होगा, क्योंकि Compiler Based Programming Language के Compiler द्वारा Generate होने वाली File हमेंशा अपने Operating System व Computer Architecture पर Depend होती है इसलिए पूरी तरह से Portable नहीं होती।

लेकिन चूंकि Compiler Based Programming Language में Program को Compile करने पर एक नई Executable File बन जाती है, जो कि पूरी तरह से Current Operating System व Computer Architecture पर आधारित होती है, इसलिए इस Executable File को अब उसके Source File की जरूरत नहीं रहती।

यानी एक बार किसी Program को Compile करके उसकी Executable File प्राप्त कर लेने के बाद अब यदि हम उसकी Source File को Delete भी कर दें, तब भी उसकी Executable File के आधार पर Computer हमारे Program को Run करेगा।

लेकिन यदि हमें हमारे Program में कोई Modification करना हो, तो हमें फिर से उस Program की Source File की जरूरत होगी, जिसे हमने Compile किया था और Modification करने के बाद हमें फिर से अपनी Source File को Compile करके एक नई Executable File Create करनी होगी, तभी हमारा Computer हमारे Modified Program को समझ सकेगा।

यानी Compiler Based Programming Languages को अपने Source Program की जरूरत केवल एक बार उस समय होती है, जब Source Program को Compile करके Executable File Create किया जाता है।

2 जबकि दूसरी प्रकार की Programming Languages को **Interpreter** Based Programming Language कहते हैं और इस प्रकार की Programming Languages की मुख्य विशेषता ये होती है कि Interpreter Based Programming Languages कभी भी Machine Depended Executable Files Create नहीं करते, इसलिए हमेंशा अपनी Source File पर Depend होते हैं।

यानी हालांकि Compiler व Interpreter दोनों ही हमारे Program को Machine Codes में Convert करते हैं, ताकि हमारा Computer उसे समझ सके, लेकिन Compiler Based Programming Language अपने Computer Architecture व Operating System पर Dependent एक नई Executable File Create करता है, इसलिए उसे अपनी Source File की जरूरत नहीं रहती। जबकि Interpreter Based Programming Language किसी भी तरह की नई Executable File Create नहीं करता। परिणामस्वरूप Interpreter Based Programming Language को हमेंशा अपनी Source File की जरूरत रहती है और यदि हम Source File को Delete कर दें, तो हमारा Program भी हमेंशा के लिए खत्म हो जाता है।

चूंकि Interpreter Based Programming Languages की कोई Executable Create नहीं होती, इसलिए इनमें बने हुए Programs को Run होने के लिए हमेंशा किसी न किसी Host Environment की जरूरत होती है, जिनमें Interpreter Based Languages के Programs Run होते हैं।

इसी वजह से किसी भी Interpreter Based Programming Language में यदि किसी प्रकार का परिवर्तन करना हो, तो उसकी Source File को ही Modify करना होता है और जब हम उस Modified Source File को फिर से Interpret करते हैं, हमें उसका Modification तुरन्त Reflect हो जाता है, जबकि Compiler Based Languages में हमें Source Filed में Modification करने के बाद उसे फिर से Compile करना जरूरी होता है, अन्यथा Modification का कोई Effect हमें Executable Program में दिखाई नहीं देता।

Interpreter व Compiler दोनों ही प्रकार की Programming Languages की एक विशेषता व एक कमी है। चूंकि Compiler Based Programs की हमेंशा एक Executable File बनती है, जो कि पूरी तरह से Current Computer Architecture व Operating System पर Depend होती है, इसलिए Compiler Based Programs की Speed हमेंशा Interpreter Based Programs की तुलना में Fast होती है, क्योंकि Interpreter Based Programs की तरह इन्हें बार–बार Machine Codes में Convert नहीं होना पडता।

लेकिन Interpreter Based Program किसी भी Computer Architecture व Operating System पर बिना Recompile किए हुए ज्यों के त्यों बार–बार Run हो सकते हैं। यानी ये Portable होते हैं क्योंकि ये हमेंशा अपने **Host Environment** में Current Computer Architecture व Operating System के आधार पर बार–बार हर बार Interpret होते हैं यानी Machine Codes में Covert होते हैं और Program Run होने के बाद इनके Machine Codes समाप्त हो जाते हैं।

"C", "C++" जैसी Programming Languages, Compiler Based Programming Languages हैं, जबकि HTML, CSS, XML, JavaScript, ASP आदि Interpreter Based Markup व Client Side Scripting Languages हैं, जो हमेंशा किसी Host Environment में Run होते हैं। यानी इनका अलग से कोई Inter-Mediator Software नहीं होता बल्कि इनका Interpreter इनके Host Environment के अन्दर ही होता है।

Host Environment वह Software होता है, जिनमें विभिन्न Interpreter Based Programming Languages के Programs Run होते हैं। उदाहरण के लिए Web Browser वह Host Environment होता है, जहां HTML, XML, CSS, JavaScript आदि के Programs Run होते हैं और हमें इनका Output एक Rendered Web Page के रूप में दिखाई देता है।

जैसाकि हमने पहले भी कहा कि JavaScript एक Client Side में Run होने वाली Interpreter Based Scripting Language है और Interpreter Based होने की वजह से JavaScript का अलग से कोई Interpreter Software नहीं होता, बल्कि JavaScript Programs जिस Software में Run होते हैं, उन Software में ही JavaScript के Engine को Build किया गया होता है।

सामान्यतः Web Browsers ही JavaScript का Host Environment होते हैं, लेकिन इसका मतलब ये नहीं है कि JavaScript के Programs केवल Web Browser में ही Run हो सकते हैं। वास्तव में सच्चाई ये है कि जिस किसी भी Software में JavaScript Engine Embedded होता है, हर उस Software में JavaScript के Programs Run हो सकते हैं।

इसीलिए JavaScript केवल Web Browser में ही Use नहीं किया जाता बल्कि JavaScript Engine को कई अन्य Platforms में भी Embed किया गया है, जहां JavaScript के Programs Run हो सकते हैं।

उदाहरण के लिए Adobe Flash एक प्रकार का Animation Software है, जहां Programming Language के रूप में **ActionScript** को Use किया जाता है। ये भी एक प्रकार की JavaScript Language ही है। इसी तरह से Adobe PDF Reader में भी JavaScript Supported है।

वर्तमान समय में विभिन्न प्रकार के Web Development IDEs उपलब्ध हैं, जैसेकि Adobe DreamWeaver, Eclipse, NetBeans आदि, इनमें भी JavaScript Engine Embedded है, इसलिए ये भी JavaScript के Host Environments हैं।

यानी हम जिस Software को Use कर रहे हैं, यदि उसमें **ECMAScript Standard** आधारित कोई भी Scripting Language Supported है, तो वह एक प्रकार से JavaScript का भी **Host Environment** है।

चूंकि JavaScript का सबसे ज्यादा प्रयोग Web Pages व Web Applications को **Interactive** (User Interaction Supported) बनाने के लिए किया जाता है, इसलिए इस पुस्तक में हमारे लिए Web Browsers ही JavaScript का **Host Environment** है।

History of JavaScript

JavaScript को सबसे पहले 1995 में Netscape Navigator के Developers ने अपने Web Browser में Client Side Validation के लिए Develop किया था। Netscape तो Market से पूरी तरह से जा चुका है, लेकिन उसकी Develop की गई JavaScript Language अभी भी Market में है और आगे भी लम्बे समय तक रहने वाली है क्योंकि अब ये Language न केवल Client Side Validation के लिए उपयोगी है, बल्कि कई जगहों पर इसे Server Side Scripting Language के रूप में भी Use किया जाता है।

1992 के आसपास **Nombas** नाम की एक Company ने जिसे बाद में **Openware** नाम की Company ने खरीद दिया, एक Scripting Language Develop करना शुरू किया, जिसका नाम C-Minus-Minus रखा गया था। CMM इसलिए, क्योंकि ये लगभग पूरी तरह से C व C++ Language पर आधारित थी, लेकिन आसानी से Web Browsers में Client Side Requirements को पूरा कर सकती थी और Developers इसे आसानी से सीख सकते थे।

कुछ समय बाद Nombas ने इस Language का नाम CMM से बदलकर **ScriptEase** रख दिया। जब Netscape Navigator Market में Popular होने लगा, तो Nombas ने इसी Language का एक नया Version Develop किया जो कि Web Page में Embed हो सकता था। शुरूआत में इस Embedding Process को Espresso Pages कहा जाता था और यही World Wide Web का पहला Client Side Scripting Language बना।

Internet पर लोगों का रूझान बढने की वजह से Web Page की Size भी बढने लगी जिससे Network का Traffic भी बढने लगा क्योंकि ज्यादातर Validation व Interactivity के कामों को पूरा करने के लिए बार–बार Web Browser को Web Server से Request करनी पडती थी। इसलिए Netscape ने महसूस किया कि Web Server का Interaction कम करने के लिए एक ऐसी Scripting Language की जरूरत है जो Web Browser में ही ज्यादातर Validation के कामों को पूरा कर दे।

इस जरूरत को ध्यान में रखते हुए **Brendan Eich** जो कि Netscape Navigator को Develop कर रहे थे, ने **LiveScript** नाम की एक Client Side Scripting Language को अपने Web Browser में Include किया। उसी समय **Sun Microsystems** अपनी Programming Language "**Java**" को Develop कर रहा था और लोगों में Java बहुत Popular हो रही थी, इसलिए Netscape Navigator ने Official Release के बाद LiveScript का नाम बदल कर **JavaScript** कर दिया, ताकि लोग ये समझकर इस Language पर भी ध्यान दें कि JavaScript, Java से संबंधित ही कोई Language है ताकि JavaScript भी Popular हो जाए और हुआ भी ऐसा ही।

Netscape व उसके **JavaScript** की सफलता के साथ ही Microsoft ने भी Web Browser Technology में कदम रखा और अपनी स्वयं की JavaScript जैसी Scripting Language बनाई जिसका नाम **JScript** रखा गया।

इस समय तक वास्तव में **JavaScript, JScript** व **ScriptEase** तीन Client Side Scripting Languages हो गई थीं, जो कि किसी भी तरह से एक Unique Standard को Follow नहीं कर रही थीं।

चूंकि इन Client Side Scripting Language की Popularity बहुत कम समय में बहुत ज्यादा हो गई थी, इसलिए इस Language को भी Standardized करने की जरूरत महसूस की गई, ताकि Scripting Language Develop करने वाली सभी Companies उन Standards के आधार पर ही अपनी Scripting Language को Develop करें व Web Developers को अलग–अलग Web Browsers के लिए अलग–अलग तरह की Scripting Languages न सीखनी पडे।

इसलिए 1997 में को **European Computer Manufactures Association (ECMA)** को JavaScript 1.1 को Standardized करने का एक Proposal भेजा गया और इस Association ने **Netscape, Sun, Microsoft, Borland** व अन्य Companies, जो कि Client Side Scripting Language Develop करने में Interested थीं, के सदस्यों की एक Technical Committee गठित की ताकि JavaScript को **Cross Platform, Vendor Neutral** Scripting Language बनाने के लिए उसके Syntax व Semantics को Standardize किया जा सके।

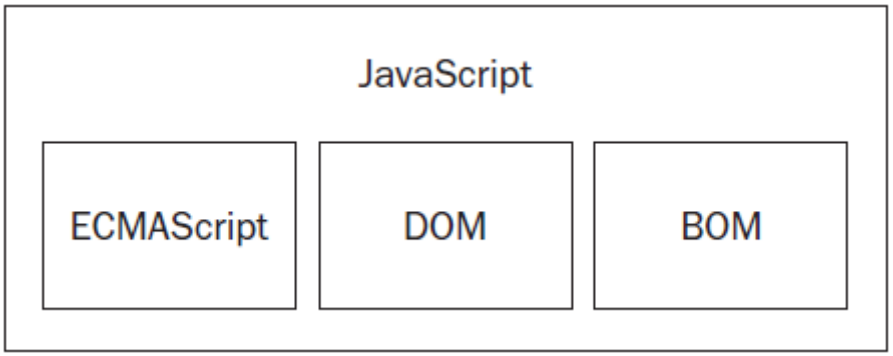
फल स्वरूप इस Committee ने अन्तिम रूप से **ECMAScript-262** नाम का एक Standard तैयार किया और JavaScript का नाम बदलकर **ECMAScript** हो गया। यानी आज की जो JavaScript है वह वास्तव में JavaScript नहीं बल्कि ECMAScript है।

आगे आने वाले कुछ सालों में *International Organization for Standardization and International Electotechnical Commission* (**ISO/IEC**) ने भी ECMAScript को एक Standard की तरह Accept कर लिया और फिर बनने वाले सभी Web Browsers में JavaScript के Implementation के लिए ECMAScript को आधार के रूप में उपयोग में लिया जाने लगा।

JavaScript Implementation

चूंकि सामान्यतः ECMAScript व JavaScript दोनों को एक ही समझा जाता है, जबकि JavaScript, ECMS-262 से कुछ ज्यादा है। एक Complete JavaScript Implementation के तीन हिस्से होते हैः

1. **The Core (ECMAScript)**
2. **The Document Object Model (DOM)**
3. **The Browser Object Model (BOM)**



ECMAScript

ECMA-262 में Define किया गया ECMAScript किसी Web Browser से Tied नहीं होता। वास्तव में इस Language में Input Output के लिए कोई Method नहीं है। ये Standard केवल एक Specification है जो विभिन्न Companies को एक आधार देता है कि उन्हें JavaScript को किस प्रकार से Implement करना चाहिए, ताकि वह विभिन्न अन्य Web Browsers के Standard के समरूप रहे। Web Browsers केवल वह **Host Environment** होते हैं, जिसमें **ECMAScript Implementation** Exist होता है।

एक **Host Environment** ECMAScript के Implementation का आधार होता है और ये Host हमेंशा कोई Web Browser ही हो, ऐसा जरूरी नहीं है। इसीलिए Adobe Company ने इस Specification के आधार पर अपनी Scripting Language Develop की है जिसका नाम **ActionScript** है और इस Scripting Language के Codes का प्रयोग करके ही Adobe Flash में **Cross-Browser Animation** Create किया जाता है। यानी **ActionScript** Scripting Language का भी आधार ECMAScript ही है।

इसीलिए यदि आप इस पुस्तक को अच्छी तरह से समझते हैं तो आप बडी ही आसानी से ActionScript Programming को भी सीख सकते हैं और Adobe Flash में ऐसे Applications Create कर सकते हैं जिनमें Animation का प्रयोग किया जाता है।

ECMAScript के Implementation के साथ ही विभिन्न Web Browsers अपने स्वयं के भी कुछ Extensions Develop करते हैं, ताकि Web Browsers को Users ज्यादा बेहतर तरीके से Web Browsing के लिए Use कर सकें।

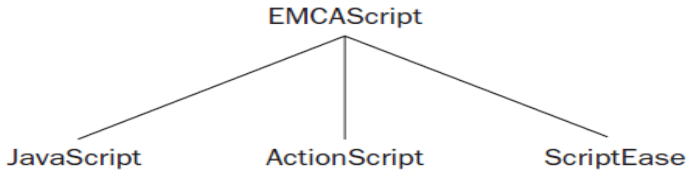
DOM यानी Document Object Model भी एक Extension ही होता है जो अपने Core के रूप में ECMAScript के Type व Syntax को Use करता है तथा Host Environment, जो कि Web Browser भी हो सकता है और कोई अन्य Software भी, Additional Functionality Provide करता है। सामान्यतः अन्य Host Environments के रूप में **ScriptEase** व **Adobe Flash** को समझा जा सकता है।

ECMA-262 वास्तव में किसी Web Browser को Reference नहीं करता बल्कि इसका Specification किसी भी Scripting Language के निम्न Parts को Describe करता है, जिसे हम **Core JavaScript** भी कह सकते हैं:

1. **Syntax**
2. **Types**

3 **Statements**
4 **Keywords**
5 **Reserved Words**
6 **Operators**
7 **Objects**

ECMAScript केवल किसी Language के Implementation का Description मात्र है, इसलिए JavaScript वास्तव में ECMAScript को Implement करता है, ECMAScript स्वयं कोई Programming Language नहीं है बल्कि इसके आधार पर अन्य Scripting Language Develop की गई हैं, जिनमें से कुछ Most Poplar Implementations निम्नानुसार हैं:



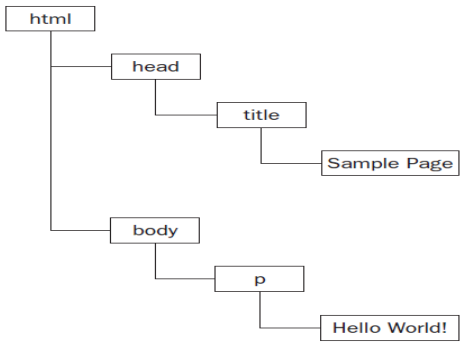
वर्तमान समय में ECMAScript का 5th Version आ चुका है, लेकिन इसे पूरी तरह से विभिन्न Web Browsers में Implement नहीं किया गया है। वर्तमान समय में Internet Explorer, FireFox, Safari, Chrome व Opera जो कि सबसे ज्यादा Use किए जाने वाले Web Browsers हैं, ने **ECMAScript3.1** Specification को पूरी तरह से Implement किया है।

Document Object Model(DOM)

DOM एक *Application Programming Interface* **(API)** है, जिसे XML के लिए Define किया गया था ताकि HTML Documents को Extend किया जा सके। DOM किसी भी Document को Memory में **Nodes की एक Hierarchy** के रूप में Model करता है। HTML या XML Document का हर Element या Tag, Attribute व Text आदि DOM के Nodes को Represent करते हैं। उदाहरण के लिए निम्न HTML Code देखिए:

```
<html>
   <head>
      <title>Sample page</title>
   </head>
   <body>
      <p>Hello World! </p>
   </body>
</html>
```

जब ये HTML Code Web Browser की Memory में Load होता है, तब निम्नानुसार Form में विभिन्न HTML Elements की एक Hierarchy बन जाती है:



किसी Document के विभिन्न Elements के Memory में इस तरह से Model होने की व्यवस्था को ही DOM या **Document Object Model** कहा जाता है, जिसमें Document के विभिन्न Elements DOM के एक **Node** को Represent करते हैं और हर Node एक **Object** की तरह व्यवहार करता है, जिसकी स्वयं की कुछ ***Properties*** व ***Behaviors*** होते हैं।

Document के विभिन्न Contents की एक Tree बनाकर DOM, किसी Web Developmer को अपने Document पर पूरी तरह से Control करने की सुविधा प्रदान करता है क्योंकि JavaScript जैसी किसी Scripting Language का प्रयोग करके Web Developer अपने Document के किसी Node को Remove कर सकता है, DOM में नया Node Add कर सकता है, किसी अवांछित Node को Replace कर सकता है अथवा DOM API का प्रयोग करते हुए किसी Node को Modify कर सकता है।

चूंकि Web Browser में Document Render होने से पहले उस Document का DOM Tree Create होता है, जो कि उस Document का **In-Memory Model** होता है और Web Browser के Window में वही दिखाई देता है, जो DOM Tree में होता है, इसलिए DOM में किए जाने वाले परिवर्तनों का Effect तुरन्त Web Browser में Reflect होता है।

इसलिए DOM Tree किसी भी Client Side Scripting Language के लिए एक मुख्य Source होता है, जिस पर वह Scripting Language विभिन्न प्रकार के Operations Perform करके Document को ज्यादा Interactive बनाने में सक्षम हो पाता है।

चूंकि DOM को विभिन्न Companies ने अपने–अपने Web Browsers में अपनी सुविधानुसार अलग–अलग तरीकों से Develop किया था, इसलिए Web को Cross Platform यानी Platform Independent बनाए रखने के लिए व सभी Web Browsers में किसी Document को एक जैसा दिखाने के लिए फिर से एक Standard तरीके की जरूरत को महसूस किया गया।

फलस्वरूप एक नया Organization अस्तित्व में आया जिसका नाम World Wide Web Consortium (**W3C**) था। ये Organization विभिन्न प्रकार के Web Related Standards Develop करने का काम करता है। इस Organization में विभिन्न बडी कम्पनियों जैसे कि Microsoft, Google, Yahoo, AOL आदि के Members Participate करते हैं और Web किस दिशा में आगे बढेगा इस बात का निर्णय लेकर Standards Create करते हैं।

DOM के आज तक में कुल तीन Levels W3C द्वारा Define किए गए हैं। DOM Level 1 सबसे पहले October 1998 में Recommend किया गया था। इस DOM के दो हिस्से **DOM Core** व **DOM HTML** थे।

DOM Core किसी XML Based Document को Structure करने की सुविधा प्रदान करता है ताकि Developers किसी XML Document के विभिन्न हिस्सों को आसानी से Access कर सकें तथा **DOM HTML** वास्तव में DOM Core का ही एक Extension है, जिसमें HTML के साथ कुछ Specific Objects व Methods को Add करके HTML को Extend किया गया है।

DOM JavaScript नहीं है और **ECMAScript** की तरह ही इसे भी कई अन्य Programming Languages में Implement किया गया है। हालांकि Web Browsers में DOM को ECMAScript का प्रयोग करके Implement किया गया है और अब ये DOM JavaScript Language का एक सबसे बडा व सबसे महत्वपूर्ण हिस्सा है।

DOM भी ECMAScript की तरह ही केवल एक Specification है। जिस तरह से ECMAScript के आधार पर विभिन्न प्रकार की Scripting Languages को Develop किया गया है, उसी तरह से DOM के आधार पर विभिन्न प्रकार की Programming Languages में किसी Document को Access व Manipulate करने के तरीकों को Develop किया जाता है ताकि एक Programming Language में Develop किया गया Document किसी दूसरी Programming Language में भी आसानी से उपयोग में लिया जा सके।

हालांकि DOM Level 1 का मूल उद्देश्य किसी Document को Structure करना था, ताकि Developers JavaScript जैसी Client Side Scripting Language द्वारा Document के विभिन्न हिस्सों को आसानी से Access व Manipulate कर सकें जबकि DOM Level 2 को Develop करने का मूल उद्देश्य DOM के साथ Mouse व User Interface Events, Ranges, Traversals, तथा Cascading Style Sheets को Support करवाना था, ताकि Document को न केवल बेहतर तरीके से Structure किया जा सके बल्कि उसे आसानी से Style भी किया जा सके। साथ ही उसे Interactive भी बनाया जा सके। इसलिए DOM Level 1 के Core को **XML Namespaces** को Support करने के लिए Extend किया गया। DOM Level 2 में निम्न नए Modules को Extend किया गया थाः

1 **DOM Views**
2 **DOM Events**
3 **DOM Styles**
4 **DOM Traversal and Range**

Document की Styling करने से पहले व Styling करने के बाद एक ही Document के कई Views हो जाते हैं। इन Views को Handle करने के लिए **DOM Views** का Concept Describe किया गया।

Document को User के लिए ज्यादा Interactive बनाने के लिए विभिन्न प्रकार के Events व Event Handlers को **DOM Events** के रूप में Describe किया गया।

Document की Styling को Control करने व Document के Structure से अलग रखने के लिए **DOM Styles** को Describe किया गया ताकि Document की Styling को Control, Manage व Handle करना आसान हो सके।

DOM Traversal and Range को Describe करके DOM को Access, Manipulate व Traverse करने के लिए नए Descriptions को Define किया गया।

वर्तमान समय में **DOM Level 3** को Describe किया जा रहा है, जिसमें ऐसे Methods को Support किया जा रहा है ताकि Web Browser या Host Environment के Document को

Local Device पर Save किया जा सके व Local Device से Host Environment में Load किया जा सके।

एक तरह से देखा जाए, तो अब Web Technology पूरी तरह से Desktop Technology के समकक्ष आने वाली है। क्योंकि DOM Level 2 तक किसी भी Document को Local Device में Save नहीं किया जा सकता था, इसीलिए कोई भी User केवल वही Document देख सकता था, या वैसे ही किसी Document को Access कर सकता था, जैसा Developer ने उसे अधिकृत किया था।

लेकिन DOM Level 3 के पूर्ण Implementation के बाद ये बात पूरी तरह से बदल जाएगी। क्योंकि उस स्थिति में User अपनी इच्छानुसार किसी Document को Modify कर सकेगा और अपने Personal Device पर Save कर सकेगा। जिससे एक ही Document को अलग–अलग Users अपनी इच्छानुसार अलग–अलग तरीके से Access व Manipulate कर सकेंगे।

DOM Level 3 का Implementation धीरे–धीरे होने लगा है और **HTML5** DOM Level 3 व **CSS3** का ही एक Implementation है, जो कि धीरे–धीरे विभिन्न Web Browsers में Support किया जाने लगा है।

इन मूल DOMs के अलावा कुछ अन्य DOMs भी हैं, जिन्हें अलग प्रकार की जरूरतों को पूरा करने के लिए Define किया गया है। उदाहरण के लिए **SVG 1.0** व **MathML 1.0** का अपना DOM है। SVG Host Environment में Graphics Develop करने से संबंधित Standards को Handle करता है, जबकि MathML Mathematics से संबंधित Functions, Formulas आदि को Handle करता है। इसी तरह से **SMIL** के लिए Document में Multimedia Integration से संबंधित DOM को Specify किया गया है।

इनके अलावा अन्य Languages ने अपनी जरूरत के अनुसार अपना स्वयं का DOM Develop किया है। उदाहरण के लिए Mozilla ने **XML** का प्रयोग करके **XUL (XML User Interface Language)** विकसित किया है, जिसका प्रयोग Mozilla व Firefox Web Browsers के Front End को Develop करने के लिए किया गया है। लेकिन इस Language व ऐसी ही कई और Languages को W3C ने Standard के रूप में Accept नहीं किया है, जिन्हें अलग–अलग Companies ने XML के आधार पर अपनी Specific जरूरतों को पूरा करने के लिए Develop किया है।

Browser Object Model (BOM)

Web Browsers के शुरूआती दिनों में Standards बनने से पहले विभिन्न Web Browsers बनाने वाली Companies ने अपने–अपने Web Browsers में एक Specific तरह का **Browser Object Model** बनाया था, जो Web Browser को Access व Manipulate करने की सुविधा देता था। BOM का प्रयोग करके Web Developers अपने Web Page से अपने Web Browser को Access करने की क्षमता प्राप्त करते थे।

चूंकि विभिन्न Web Browser बनाने वाली Companies अपने Web Browsers को अपनी इच्छानुसार बनाती हैं, इसलिए यही एक ऐसा हिस्सा है जहां विभिन्न Companies के Web Browsers में JavaScript Implementation का कोई Standard नहीं है।

प्राथमिक रूप से BOM Web Browser **Window** व **Frames** के साथ Deal करता है लेकिन सामान्यतः Browser Specific Extensions को JavaScript में Develop किया जाता है जो कि

BOM के एक हिस्से की तरह काम करता है। कुछ Extensions निम्नानुसार हैं, जो लगभग सभी Web Browsers में Common हैं हालांकि उनको अलग–अलग तरीके से Implement किया गया है:

1. नया Window Popup करने की Capability
2. Web Browser Window को *Move, Resize* या *Close* करने की Capability
3. **navigator** Object जो कि Web Browser से संबंधित Detailed जानकारी देता है।
4. **location** Object जो कि Web Browser में Loaded Web Page की Detained जानकारी देता है।
5. **screen** Object हो कि User के Computer के Screen Resolution की Detailed जानकारी देता है।
6. Cookies का Support भी एक Extension के रूप में Web Browser के BOM का हिस्सा होता है।
7. **XMLHttpRequest** तथा Internet Explorer का ActiveXObject भी Web Browser के BOM की Capabilities का ही एक हिस्सा है।

चूंकि BOM के लिए कोई Standard नहीं है, इसलिए सभी Web Browsers में BOM का Implementation पूरी तरह से Web Browser बनाने वाली Company की नीतियों पर आधारित होता है। फिर भी सभी Web Browsers में **window** व **navigator** Object जरूर होता है लेकिन इन Objects की Properties व Methods को अलग–अलग Web Browsers अपनी इच्छानुसार तय करते हैं।

अलग–अलग Standards के साथ JavaScript के भी कई Versions विभिन्न Web Browsers में Implement किए गए हैं। वर्तमान समय में लगभग सभी Web Browsers JavaScript 2.0 Version को Support कर रहे हैं।

JavaScript के Version बढने के साथ उसके Features जैसे कि Keywords, Syntaxes, Features आदि भी Change होते हैं। JavaScript 2.0 वास्तव में **ECMAScript 3.1** Proposal का ही Implementation है।

चूंकि ECMAScript का 5th Version भी आ चुका है, तो जाहिर सी बात है कि जैसे–जैसे Web Browsers, ECMAScript के इस 5th Version को Support करने लगेंगे, JavaScript का एक और नया Version भी आएगा।

Web Browsers

चूंकि **JavaScript**, वास्तव में **BOM** (*Browser Object Model*), **Core ECMAScript** व **DOM** (*Document Object Model*) तीनों का Combination है, इसलिए JavaScript को समझने के लिए हमें इन तीनों को Best तरीके से समझना होगा और जैसाकि हमने पहले भी कहा है कि इस पुस्तक में Web Browser ही हमारा Host Environment है, इसलिए Web Browser को अच्छी तरह से समझे बिना हम JavaScript को उसकी पूरी ताकत के साथ उपयोग में नहीं ले सकते।

Web Browser एक ऐसा माध्यम होता है जो किसी Web Application या Web Document को Download करता है, Render करता है व Execute करता है। Web Browsers दो तरह के होते हैं। पहले प्रकार के Web Browsers केवल Text Browser होते हैं जो केवल Text Content को ही Render करते हैं। **lynx** एक ऐसा ही Web Browser है जो कि http://lynx.isc.org/ Website पर Free Available है।

जबकि दूसरे प्रकार के Web Browsers Text के अलावा विभिन्न प्रकार के Multimedia जैसे कि Sound, Audio, Video, Images, Animations आदि को भी Render करने में सक्षम होते हैं। *Google Chrome, Mozilla Firefox, Apple Safri, Internet Explorer, Opera* आदि सबसे ज्यादा Use होने वाले इस Group के Modern Web Browsers के Examples हैं।

Engines

चूंकि एक Web Browser विभिन्न प्रकार के Resources जैसेकि HTML Document, CSS Stylesheets, Multimedia Plugins, आदि को आपस में व्यवस्थित तरीके से Organize करके User के सामने Present करता है, इसलिए इन विभिन्न प्रकार के Resources को Process करने के लिए एक Web Browser में विभिन्न प्रकार के Resource Processors होते हैं, जिन्हें **Engines** कहा जाता है।

ये Engines ही किसी CSS Style को किसी HTML Element पर Apply करते हैं अथवा किसी Element पर Click करने पर Trigger होने वाले Event को Response करते हैं। यानी ये Engines ही Internally विभिन्न प्रकार के HTML, CSS, JavaScript, XML आदि Codes को Process करते हैं और हमारे सामने एक Well Organized Web Page **Render** करते हैं। Engines की कार्यप्रणाली को हम एक Car के उदाहरण द्वारा बेहतर तरीके से समझ सकते हैं।

जिस प्रकार से किसी Car की Body उसका बाहरी ढांचा मात्र होता है और उस Car की Body के Good Looking होने का मतलब ये नहीं होता कि वह Car वास्तव में Efficient व Powerful है बल्कि उस Car में जो Engine होता है, वह Engine ही उस Car की Efficiency व Power तय करता है।

ठीक इसी प्रकार से कोई Web Browser कितना अच्छा दिखाई दे रहा है अथवा Web Browser का User Interface कितना अच्छा है, इस बात से हम Web Browser की Inner Working व Power का पता नहीं लगा सकते, बल्कि Web Browser की Efficiency व Power पूरी तरह से उसमें Use किए गए **Engines** पर निर्भर करती है, जो कि किसी भी Web Page के विभिन्न Resources (HTML, XML, CSS, JavaScript Codes) को Process करके Render करने का काम करते हैं।

किसी Web Page का पूरी तरह से Process होकर Web Browser में पूरी तरह से Load होने की प्रक्रिया को Web Page का **Render** होना कहते हैं।

किसी भी Web Browser में मूल रूप से हमेंशा दो प्रकार के **Engines** होते हैं:

1. **Rendering Engine -** इसे सामान्यतः Layout Engine भी कहते हैं जो कि HTML व CSS Codes को Process करके एक Page को Screen पर व्यवस्थित तरीके से Organize करके Visible या Show करता है।
2. **JavaScript Engine -** ये Engine, JavaScript Codes को समझकर Process व Execute करता है, जिसका Effect Web Page व Web Browser के **Chrome** पर Reflect करता है।

Web Browser का वह हिस्सा जिससे User Interact करता है, Web Browser का **Chrome** कहलाता है। किसी Web Browser का Menubar, Bookmark Toolbar, Web Browser का Frame, Web Browser का Title Bar, Standard Toolbar आदि Web Browser के Chrome का हिस्सा होते हैं।

Web Browsers के ये Engines, User Inter से पूरी तरह से अलग होते हैं। यानी कोई भी User Interface Element जैसेकि Menubar, Standard Toolbar या Navigation Bar इन Engines से Directly Connected नहीं होता।

विभिन्न प्रकार के **Rendering** व **JavaScript** Engines को अलग–अलग प्रकार की Companies, Organizations या Individuals ने Develop किया है और उन्होंने ही ये तय किया है कि कोई Web Page उनके Web Browser में किस प्रकार का दिखाई देगा। इसलिए यदि हम एक ही Web Page जैसे कि http://www.google.com के Home Page को अलग–अलग Web Browsers में Open करें, तो समान Home Page भी अलग–अलग Web Browsers में Exactly समान दिखाई नहीं देता।

चूंकि Web Browsers के Engines, Web Browser के User Interface से पूरी तरह से अलग रहते हैं इसलिए Technically ऐसा सम्भव है कि एक ही **Rendering** या **JavaScript Engine** को Use करते हुए दो बिल्कुल अलग Web Browsers या Software (**Host Environment**) Create किए जा सकते हैं, जो कि एक दूसरे से बिल्कुल भिन्न दिखाई देते हों जबकि विभिन्न Web Browsers के User Interface को हम JavaScript Engines के Container की तरह समझ सकते हैं।

यानी JavaScript Engine किसी Web Browser में ठीक उसी तरह से Exist होता है, जिस तरह से किसी Car में उसका Engine Exist होता है और Web Browser का User Interface उस JavaScript Engine के Skin या Body की तरह होता है और जिस तरह से समान प्रकार का Engine Use करते हुए अलग–अलग प्रकार की Body की Car बनाई जा सकती है, उसी तरह से समान प्रकार का JavaScript व Rendering Engine Use करते हुए, अलग–अलग प्रकार के Web Browser User Interface बनाए जा सकते हैं।

वर्तमान समय में बहुत ज्यादा उपयोग में लिए जाने वाले विभिन्न Web Browsers के **Rendering Engines** को हम निम्न सारणी अनुसार समझ सकते हैं:

Rendering Engine	**Web Browser**
Trident	Microsoft Internet Explorer
Gecko	Mozilla Firefox
Presto	Opera browser
WebKit	Apple Safari (including iPhone), Google Chrome, Nokia (for mobile devices)

इसी तरह से वर्तमान समय में बहुत ज्यादा उपयोग में लिए जाने वाले विभिन्न Web Browsers के **JavaScript Engines** को हम निम्न सारणी अनुसार समझ सकते हैं:

JavaScript Engine	**Web Browser**
Jscript	Microsoft Internet Explorer
SpiderMonkey	Mozilla Firefox (up to and including version 3.5)
TraceMonkey	Mozilla Firefox (version 3.6)
JavaScriptCore	Apple Safari (up to and including version 3.2)
Nitro	Apple Safari (version 4)
V8	Google Chrome
Futhark	Opera

जैसाकि उपरोक्त सारणी द्वारा हम समझ सकते हैं कि एक ही Web Browser में हम Rendering Engine व JavaScript Engines के अलग–अलग Combinations को Use कर सकते हैं।

उदाहरण के लिए Mozilla Firefox ने अपने Firefox 3.5 Version तक ***SpiderMonkey*** नाम के JavaScript Engine को Use किया है जबकि बाद के Versions में ***TraceMonkey*** नाम के Version को Use करना शुरू कर दिया है।

विभिन्न **JavaScript Engine** Develop करने वाले Developers का मूल उद्देश्य यही है कि उनका Engine ज्यादा से ज्यादा तेज गति से JavaScript Codes को Process करे, ताकि Web Browsers Based Web Applications, जो कि JavaScript पर निर्भर हों, उसी Speed से Run हो सकें, जिस Speed से Compiler Based Executables Run होते हैं। इसलिए कई मायनों में Web Browser एक प्रकार से नया Operating System बनते जा रहे हैं।

इससे पहले कि हम आगे बढें, Web Browser की कार्यप्रणाली को भी थोडा बेहतर तरीके से समझना उपयोगी रहेगा, क्योंकि Web Browser के **Request** व **Response** Message से संबंधित कई प्रकार के Web Browser Related Objects होते हैं, जिन्हें JavaScript द्वारा Access व Manipulate करने की जरूरत पडती है।

Web Page – Request and Response

HTTP वह Protocol या Software Piece है, जो Web Browser के Addressbar में Specify किए जाने वाले Web Address के Resource को Web Browser में Load करने का काम करता है।

यानी Web Server व Web Browser के बीच जो Data Transfer होता है, उसे Handle करने का काम **HTTP** (*Hyper Text Transfer Protocol*) करता है। इस Protocol के अन्तर्गत **Web Browser** एक **Client** होता है, जो किसी Web Resource के लिए Request करता है जबकि **Web Host** वह **Server** होता है, जो Web Browser से आने वाली Request को पूरा करते हुए उसे उसका Required Web Resource Available करवाता है।

Web पर उपलब्ध किसी भी File (HTML, XML, CSS, JavaScript, Image, Sound, Video etc…) को **Web Resource** कहा जाता है।

जब Web Browser के Address Bar में कोई URL (Uniform Resource Locator) जैसे कि http://www.bccfalna.com Specify किया जाता है या किसी Web Page पर Specified किसी Hyperlink को Click किया जाता है, तो Web Browser एक **Request Message** Create करके उसे Web Server पर भेज देता है। जिसके बदले में Web Server उस Resource को Web पर Search करता है और एक **Response Message** के साथ वह Resource Web Browser को Available करवाता है। इस प्रकार से Client व Server के बीच HTTP के माध्यम से Resources का Transfer होता रहता है।

HTTP Request Message

जब Web Browser किसी URL के लिए कोई Request करता है, तो Request के रूप में एक Plain Text HTTP Request Message Create होता है, जिसे Web Server पर Send किया जाता है। इस Request Message में उस Resource की Information होती है, जिसे Web Server से प्राप्त करके Current Web Browser में Load किया जाना होता है।

उदाहरण के लिए यदि हम Web Browser के Address Bar में http://wwww.google.com Type करके Enter Key Press करते हैं, तो Web Browser निम्नानुसार HTTP Request Message Create करता है:

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:19.0) Gecko/20100101 Firefox/19.0
Accept: text/html,application/xhtml+xml,application/xml;
Accept-Language: en-gb,en;
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;
Keep-Alive: 300
Connection: keep-alive
Cookie: PREF=ID=980395a10a8f6655:U=c31bdc3844339937:…
```

इस Request में हर Line का Code एक प्रकार का Header Message है और हर Line Web Server को Request किए गए Resource से संबंधित विभिन्न प्रकार की जरूरी जानकारियां देता है। चलिए, इस Request Message को थोडा समझने की कोशिश करते हैं।

इस Header या Request Message में सबसे पहले वह Action या Method Define होता है, जिसका प्रयोग करते हुए Request Message को Web Server पर Send किया जाना है।

HTTP में हम मूल रूप से 8 प्रकार के Actions या Methods को उपयोग में लेते हुए Web Server से किसी Resource की Request कर सकते हैं। लेकिन सामान्यतः जब हम Web Browser द्वारा किसी Resource की Request करते हैं, तब वह Request **GET** या **POST** Method द्वारा की जाती है। फिर भी विभिन्न प्रकार के Request Methods की Details निम्नानुसार हैं:

GET Method

किसी भी Webpage के हमेंशा दो हिस्से होते हैं, जिन्हें **Head Part** व **Body Part** के नाम से जाना जाता है। Head Part में हमेंशा Meta Information होते हैं, जो कि Basically Search Engines व Web Browser के Chrome से संबंधित होते हैं, जबकि Body Part में Web Page के Actual Contents होते हैं।

इस Method को Use करने पर Specified URL पर स्थित Page के Content का HTML Format Body Return होता है।

POST Method

इस Method का प्रयोग सामान्यतः HTML Form में किया जाता है, जिसमें किसी Data को फिर से Process होने के लिए Web Server पर भेजना होता है।

HEAD Method

ये Method **GET** Method के समान ही काम करता है। दोनों में मूल अन्तर केवल इतना है कि GET Method Use करने पर Requested HTML Page की Body भी Return होती है, जबकि

HEAD Method Use करने पर Requested HTML Page का केवल Head Part ही Return होता है, जिसमें Web Browser से संबंधित Metadata Information होती है।

इस Method का प्रयोग हम तब करते हैं, जब हमें केवल Response के साथ आने वाले Metadata को ही प्राप्त करना होता है अथवा इस बात का पता लगाना होता है कि Specified URL Actually Exist है या नहीं।

PUT Method

इस Method को Use करके हम किसी Web Server पर स्थित किसी Resource को Update कर सकते हैं। ये सामान्यतः POST Method के समान काम करता है, लेकिन ये केवल उसी स्थिति में Server के किसी Resource को Modify कर सकता है, जबकि Server इस बात की Permission देता हो।

DELETE Method

इस Method को Use करके हम किसी Web Server पर स्थित किसी Resource को Delete कर सकते हैं, लेकिन ये केवल उसी स्थिति में Server के किसी Resource को Delete कर सकता है, जबकि Server इस बात की Permission देता हो।

TRACE Method

ये Method, Web Server पर Sender द्वारा आने वाली Request को फिर से उसी Sender को भेज देता है। इस Method का प्रयोग करके हम इस बात का पता लगा सकते हैं कि Request के दौरान कौन–कौन से Servers, Services आदि Client व Server के बीच बनने वाले Connection के Chain में Involve हो रहे हैं।

OPTIONS Method

इस Method को Use करके हम किसी Particular URL पर Available विभिन्न Actions या Methods का पता लगा सकते हैं, जिसे वह URL Support करता है। यदि हम URL को एक Wildcard Character (*) की तरह Specify करते हैं, तो Web Server हमें उस Resource पर Perform हो सकने वाले सभी Actions (Methods) की List Response के रूप में Return करता है।

अब यदि हम हमारे उपरोक्त उदाहरण के Request Message की पहली Line को देखें, जो कि निम्नानुसार हैः

```
GET / HTTP/1.1
```

तो हम समझ सकते हैं कि ये Line Web Server को इस बात की Information देगा कि Web Browser को Request किए जाने वाले Page का HTML Markup यानी Body Part चाहिए। जबकि Line में दिखाई देने वाला "/" Character इस बात को Specify कर रहा है कि Web Browser को Specified Domain के Root Page या Home Page की जरूरत है और इस जरूरत को HTTP/1.1 यानी HTTP Protocol के 1.1 Version के Rules को Use करते हुए पूरा करना है।

Host: www.google.com

Request Message की ये Line Web Server को बताता है कि Web Browser जिस Host से Resource या Home Page की Request कर रहा है, वह Host www.google.com है।

User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:19.0) Gecko/20100101 Firefox/19.0

Request Message की ये Line उस Web Browser की Information दे रहा है, जिससे Request Perform की गई है। सामान्यतः User-Agent Header में Current Web Browser की Information होती है।

ये Header Line इस बात को Specify कर रहा है कि Perform होने वाली Request **Mozilla/5.0** Web Browser से Perform की गई है, जो कि **Windows NT 6.2** Operating System यानी Windows-8 पर Installed है, जबकि **WOW64** इस बात को Specify कर रहा है कि Installed Windows Operating System **64bit** का है।

Web Browser के Operating System की Information के बाद **Gecko/20100101** इस बात को Specify कर रहा है कि Current Web Browser Gecko Based Web Browser है, जिसका नाम Firefox है और Version 19.0 है।

Accept: text/html,application/xhtml+xml,application/xml;

Request Message की ये Header Line इस बात को Specify कर रहा है कि Current Web Browser किस–किस प्रकार के Document को Support करता है। यानी Current Web Browser किन File Types या **MIME Types** (Multipurpose Internet Mail Extensions) को हमारे समझने योग्य Format में Convert करके Render कर सकता है।

उपरोक्त Header इस बात को Specify कर रहा है कि Current Web Browser HTML, XHTML व XML Types के Documents को इस तरह से Render करने में सक्षम है, जिस तरह से वह हमें यानी Human Beings को समझ में आता है।

Accept-Language: en-gb,en;

ये Header Line Web Server को इस बात की जानकारी देता है कि Current Web Browser किस Locale व Languages के लिए Currently Configured है। ये Line इस बात को Specify कर रहा है कि Current Web Browser English Language व UK Locale के लिए Configured है क्योंकि "**gb**" UK Locale को Represent करता है।

जबकि Backup के लिए केवल "**en**" Specified है, जो कि इस बात की Information है कि बिना किसी Geographical Locale की स्थिति में Default रूप से ये Web Browser English Language को Support करता है। Web Server इस Information को उस स्थिति में Ignore कर देता है, जब Web Browser द्वारा Requested Page केवल एक ही Language Version में Available हो।

Accept-Encoding: gzip,deflate

ये Header Web Server को इस बात की जानकारी दे रहा है कि Current Web Browser किस प्रकार के Encoding के Content को Accept कर रहा है। यदि Web Browser द्वारा Specified Encoding Type को Web Server Support न करता हो, तो Web Server स्वयं Standard Encoding Use कर लेता है। लेकिन यदि Web Server, Web Browser Accepted Encoding को समझता है, तो वह Response Content को उसी Format में Compress करके Send करता है, ताकि Response Content की Size कम हो जाए व Content ज्यादा तेजी से Web Server तक पहुंच सके।

उपरोक्त Header में **gzip** व **deflate** Compression Format को Current Web Browser Support करता है। जिसका मतलब ये है कि यदि Web Server इन Compression Formats को Support करता है, तो वह Requested Resource यानी HTML, CSS, JavaScript आदि Files को इनमें से किसी Format में Compress करके Web Browser को Send कर सकता है, जिससे Documents के Web Browser में Download होने की Speed काफी तेज हो जाती है।

Accept-Charset: ISO-8859-1,utf-8;

ये Header Message Web Server को इस बात की जानकारी देता है कि Current Web Browser ISO-8859-1 व utf-8 Character Sets की Encoding को Accept करता है, जिनमें लगभग दुनियां कि किसी भी भाषा के अक्षरों व Symbols के लिए Specific Code समाहित हैं। यानी Web Browser दुनियां की किसी भी भाषा के अक्षरों व Symbols को Web Browser में Render करने में सक्षम है।

Keep-Alive: 300

ये Header Message Web Server को इस बात की जानकारी देता है कि Web Browser व Web Server के बीच Data Transfer के लिए जो Connection बनेगा, वह Connection 300 Seconds यानी 5 Minutes तक Available रहेगा। परिणामस्वरूप यदि 300 Seconds की अवधि में Current Web Browser से फिर से उसी Web Server पर कोई Request Send की जाती है, तो Web Server पर फिर से नया Connection Open करने की जरूरत नहीं रहेगी। लेकिन यदि Request 300 Seconds के बाद की जाती है, तो Client व Server के बीच का Connection Lost हो जाएगा और Web Browser व Server के बीच फिर से एक नया Connection Open होगा।

Connection: keep-alive

ये Connection Header Information इस बात को Specify करता है कि Client व Server के बीच किस प्रकार का Connection बनेगा। सामान्यतः **HTTP/1.1** Protocol के साथ **keep-alive** सबसे Common रूप से Use होने वाला Connection Type होता है।

Cookie: PREF=ID=980395a10a8f6655:U=c31bdc3844339937:…

HTTP Cookie Client Computer पर Locally Store होने वाले Text Based Data होते हैं, जिनका प्रयोग सामान्यतः Web Server द्वारा किसी Client Computer को Uniquely Identify करने अथवा Session Create करने के लिए किया जाता है। Cookies को सामान्यतः उस Web Site द्वारा Client Computer पर Place किया जाता है, जिसकी Request Web Browser करता है, ताकि Web Site अपने हर Viewer को Personalized Information दे सके।

चूंकि HTTP एक **Connectionless Protocol** है, यानी Client द्वारा एक बार Request करने और Server द्वारा उस Request को पूरा कर दिए जाने के बाद Client व Server दोनों एक दूसरे से पूरी तरह से अनजान हो जाते हैं, इसलिए उस स्थिति में Cookies का प्रयोग करके इस बात की जानकारी को Maintain किया जाता है कि किस Client ने Web Server पर किस Resource की Request की है। यानी एक बार एक Request पूरी हो जाने के बाद Cookies ही Client व Server के बीच एक दूसरे को फिर से Identify करने का माध्यम होते हैं।

HTTP Response Message

जब एक बार किसी Web Browser से किसी Resource की Request की जाती है, तो उस Request को पूरा करने के लिए पिछले Session में Discuss किए अनुसार एक HTTP Request Message बनता है जिसमें विभिन्न प्रकार की Header Information होती हैं। ये Request Message ही Web Server पर पहुंचता है, जिसे Web Server Receive करके इसके Data को Process करता है और बदले में एक HTTP Response Message Create करता है।

यदि हम उपरोक्त उदाहरण को ही आगे बढाऐं, तो Request के बदले में Create होने वाला Response Message निम्नानुसार हो सकता हैः

```
HTTP/1.x 200 OK
Cache-Control: private, max-age=0
Date: Fri, 29 Mar 2013 12:42:14 GMT
Expires: -1
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 2520

<html><head>
... rest of HTML for Google's home page ...
</body></html>
```

चलिए, जिस तरह से हमने Request Message के विभिन्न Headers को One by One समझा, उसी तरह से हम इस Response Message के भी सभी Headers को One by One समझने की कोशिश करते हैं।

```
HTTP/1.x 200 OK
```

Response Message का ये पहला Header Request करने वाले Web Browser को इस बात की जानकारी देता है कि Request को पूरा करने के लिए किस HTTP Protocol का प्रयोग हुआ है।

Version की Information के बाद एक Short Description Information Web Browser को Return होता है, जो कि Web Browser को इस बात की जानकारी देता है कि उसकी Request की Processing का क्या परिणाम रहा।

उपरोक्त Header में ये Status Code **200** व Description **OK** है, जो इस बात को Indicate कर रहा है, कि Web Browser द्वारा Perform की गई Request सही तरीके से Process हो गई है और Web Server ने Request किया गया Resource Return कर दिया है।

Web Server द्वारा अलग–अलग परिस्थितियों में अलग–अलग प्रकार के **Status Code** व **Description** Return होते हैं, जिनके बारे में हम आगे Detail से जानेंगे।

Cache-Control: private, max-age=0

हर Web Browser में एक Local File Cache होता है, जो Recently Requested Files को Store करके रखता है, ताकि यदि फिर से उन्हीं Files की Request हो, तो Web Browser उन Files को फिर से Download न करके अपने Cache से ही उन्हें प्राप्त कर ले, ताकि Request ज्यादा तेजी से Perform हो जाए और Response ज्यादा तेजी से प्राप्त हो जाए।

Cache-Control Header Web Browser को Currently Requested Resource से संबंधित कुछ Parameters देता है, जो Web Browser को इस बात की जानकारी देता है कि Web Browser कितने समय तक Web Server से आने वाले Resources को Cache करके रख सकता है।

हमारे उदाहरण में **Cache-Control** Header के साथ "**private**" Specified है, जो Web Browser के लिए इस बात का Indication है कि Current Resources केवल Current User के लिए ही है और यदि Current User किसी LAN यानी Local Area Network पर है, तो LAN पर उपलब्ध अन्य Users के लिए वह Resource Available नहीं होगा।

परिणामस्वरूप यदि LAN पर काम करने वाले अन्य Users उसी Web Page को Open करेंगे, तो उनके लिए हर Resource फिर से Download होगा और उनके लिए अलग से Cache होगा, फिर भले ही सभी Users समान Web Browser Program को ही Share क्यों न कर रहे हों।

साथ ही "**max-age**" Property Web Browser को इस बात की जानकारी देता है कि Web Browser कितने Seconds तक Requested Page को Cache करके रख सकता है। हमारे उदाहरण में "**max-age**" का मान 0 है, जो इस बात का Indication है कि Web Browser Google के Homepage को 0 Seconds के लिए ही Cache करेगा या दूसरे शब्दों में कहें तो Cache नहीं करेगा।

यदि हम किसी Web Page को Web Browser में Cache करवाना नहीं चाहते, तो हम **Cache-Control** Header में Value के रूप में "**no-cache**" मान Specify करके ऐसा कर सकते हैं।

Date: Fri, 29 Mar 2013 12:42:14 GMT

Response Message के इस Header में उस समय की Information होती है, जब HTTP Response Message, Web Server से Sent किया गया होता है।

Expires: -1

ये Header Web Browser को इस बात की Information देता है कि Requested Page किस Date-Time पर Old हो जाएगा और यदि फिर से उस Page को Load किया जाएगा, तो वह Web Browser के Cache से Load न होकर फिर से Download होगा। यानी Requested Web Page कब Expire हो जाएगा, इस बात की जानकारी इस Header में होती है।

Current Header में हमें -1 दिखाई दे रहा है, जो इस बात का Indication है कि Requested Page पहले से ही काफी पुराना हो चुका है और अगली बार यदि इस Page का Request किया जाएगा, तो Web Browser इस Page को Cache से Load करने के बजाय Web Server से Page की एक नई Copy Download करेगा।

Content-Type: text/html; charset=UTF-8

ये Header Requested Document के MIME Type को Represent करता है, जो Current Example में इस बात को Specify कर रहा है कि Currently Requested Page एक Plain Text HTML Page है जो कि एक **UTF-8** Character Set Supported Page है।

Content-Encoding: gzip

ये Header Web Browser को ये बात Specify कर रहा है कि Web Server से Return होने वाला Document **gzip** Format में Compressed है, जिसे Web Browser Extract करके Render करेगा।

Web Server उसी स्थिति में किसी Resource को Compress करता है, जबकि Web Browser उस Compression Mode को Accept करता है। चूंकि Request Message में हमने देखा था कि Web Browser **gzip** व **deflate** Compression Mode को Support करता है, इसलिए Web Server ने Requested Document **gzip** Format में Return किया है।

यदि Request Message में **Accept-Encoding** Header में Specified Compression Mode को Web Server Support नहीं करता, तो Web Server Return होने वाले Resource को Compress करके Send नहीं करता बल्कि Normal Content की तरह ही Send करता, जिससे Web Page या Resource के Web Browser में Load होने की Speed कुछ धीमी हो जाती।

Server: gws

ये Header Web Browser को इस बात की जानकारी दे रहा है कि Return होने वाला Resource किस Web Server से Return हो रहा है। उपरोक्त उदाहरण में ये **gws** है जो कि Google का स्वयं का Web Server है।

Content-Length: 2520

ये Header Return होने वाले Resource की Length या Size को Bytes के रूप में Return करता है। यानी ये Header Web Brower को इस बात की जानकारी देता है कि Web Server से आने वाले Resource की Size किया है।

इन सभी Response Headers के बाद अन्त में Requested Resource का Actual Content होता है, जो कि Current Example में एक HTML Document है। यदि Requested Resource कोई Image File होता, तो यहां पर Text Representation के रूप में वह Image Download होना शुरू होता, जो Download होने के बाद Current Web Browser व Operating System द्वारा Image Format में Convert होकर Web Browser में Display होता है।

अब आप समझ सकते हैं कि जब आप Web Browser के Address Bar में कोई URL Specify करके Keyboard पर Enter Key Press करते हैं या Web Page पर दिखाई देने वाले किसी Hyperlink को Click करते हैं, तब उपरोक्तानुसार दोनों Request व Response Messages में कितनी सारी Header Information Send व Receive होती है और इसी Header Information के आधार पर Web Browser व Web Server के बीच Information का Transfer होता है।

HTTP Status Codes

सभी HTTP Status Codes 3-Digit Numbers होते हैं, जो किसी Request के बदले में Web Server द्वारा किए गए Response को Represent करते हैं और इस बात को Indicate करते हैं कि Request सही तरीके से Fulfill हुई या Client द्वारा किसी अन्य Action की जरूरत है, ताकि Requested Data को उपयुक्त तरीके से Successfully Locate किया जा सके। यहां हम कुछ Common Status Codes के बारे में Discuss कर रहे हैं, जबकि HTTP के सभी Status Codes की जानकारी *http://en.wikipedia.org/wiki/List_of_HTTP_status_codes* पर प्राप्त कर सकते हैं।

200+ (Success)

200 से 299 के बीच के सभी Status Codes इस बात को Indicate करते हैं कि Web Server द्वारा Request Message को ठीक से Receive करके Process कर लिया गया है तथा Web Browser को किसी प्रकार का Content Return कर दिया गया है। इस Range में सामान्य रूप से उपयोग में आने वाले Status Codes का Description निम्नानुसार हैः

200 OK

ये Status Code इस बात को Represent करता है कि Request Successful रहा तथा Response Message में Requested Data Exist है।

201 Created

ये Status Code इस बात को Represent करता है कि POST या PUT Method के अनुसार Server पर नया Resource Create हो गया है।

204 No Content

ये Status Code इस बात को Represent करता है कि Request Successful रहा लेकिन Requested URL पर Response Message में Return करने के लिए कोई Data Exist नहीं है।

206 Partial Content

ये Status Code इस बात को Represent करता है कि Request Successful रहा लेकिन Requested Data ज्यादा होने के कारण अथवा Network Failure या User Cancellation के कारण Content Web Browser में पूरी तरह से Download नहीं हो सका और पूरा Data प्राप्त करने के लिए Web Browser को फिर से Request करना होगा अथवा Download को Resume करना होगा।

300+ (Redirection)

300 से 399 के बीच के सभी Status Codes इस बात को Indicate करते हैं कि Web Server द्वारा Request Message को ठीक से पूरा करने के लिए Client को एक Extra Step लेना होगा व किसी अन्य URL पर Redirect करना होगा अन्यथा कोई Content Return नहीं होगा। इस Range में सामान्य रूप से उपयोग में आने वाले Status Codes का Description निम्नानुसार हैः

301 Moved Permanently

ये Status Code इस बात को Represent करता है कि Requested URL किसी अन्य Location पर Permanently Move कर दिया गया है। यानी Requested Resource Current URL पर Available नहीं है बल्कि किसी अन्य URL Location पर उपलब्ध है।

302 Found

ये Status Code इस बात को Represent करता है कि Requested URL किसी अन्य Location पर Temporarily Move कर दिया गया है। यानी Requested Resource Current URL पर Currently Available नहीं है लेकिन भविष्य में फिर से इस URL पर वह Resource Available हो सकता है।

304 Not Modified

ये Status Code इस बात को Represent करता है कि Requested URL को पहले भी Request किया गया है और तब से अब तक उसमें किसी तरह का कोई Modification नहीं किया गया है। इसलिए Client को Web Browser के Cache में Stored Resource को Locally ेम करना चाहिए।

400+ (Client Error)

400 से 499 के बीच के सभी Status Codes इस बात को Indicate करते हैं कि Web Browser द्वारा भेजे गए Request Message में किसी तरह की Error थी इसलिए Web Server Requested Resource को Return नहीं कर पाया। ये Codes इस बात को Indicate करते हैं कि वांछित Response प्राप्त न हो पाने का Fault *Client Side* में है न कि Server Side में। इस Range में सामान्य रूप से उपयोग में आने वाले Status Codes का Description निम्नानुसार हैः

400 Bad Request

ये Status Code इस बात को Represent करता है कि Request Message उपयुक्त Format में न होने की वजह से Web Server उसे ठीक से समझ ही नहीं पाया।

401 Unauthorized

ये Status Code इस बात को Represent करता है कि Request Message के साथ Web Server को उपयुक्त **Username** व **Password** भी चाहिए, क्योंकि Requested Content एक Restricted Content है।

403 Forbidden

ये Status Code इस बात को Represent करता है कि Web Server ने Client की Request को Refuse कर दिया है। ऐसा तब हो सकता है, जब Web Server पर उस **IP Address** को Block या Blacklisted कर दिया गया हो, जिस पर Installed Web Browser से Request Perform किया गया है।

404 Not Found

ये Status Code इस बात को Represent करता है कि Requested URL Current Location पर Available नहीं है, लेकिन भविष्य में इस Location पर कोई Content हो सकता है, इसलिए Web Browser भविष्य में फिर से इस URL की Request कर सकता है।

405 Method Not Allowed

ये Status Code इस बात को Represent करता है कि Request Message जिस तरह का Interaction Current URL के साथ करना चाहता है, उस प्रकार का Interaction Specified URL पर Allowed नहीं है। ये Code तब Generate हो सकता है, जब User Google के Homepage को DELETE Method द्वारा Delete करने की कोशिश करे।

410 Gone

ये Status Code "**404 Not Found**" Status Code की तरह ही काम करता है। अन्तर केवल इतना है कि ये Status Code इस बात को Represent करता है कि Specified URL को फिर से Try नहीं करना चाहिए।

सामान्यतः ये Status Code, Search Engine Spider के लिए उपयोगी होती है, क्योंकि यदि Search Engine Spiders को ये Status Code प्राप्त होता है, तो Search Engine Spiders Specified URL को अपने Index से हमेंशा के लिए Remove कर सकते हैं, ताकि वे फिर से इस URL पर न आऐं।

413 Request Entity Too Large

ये Status Code इस बात को Represent करता है कि Request Message इतना बडा है कि Web Server उसे Process नहीं कर सकता। ये Status Code तब Return हो सकता है, जब कोई HTML Form अपनी Limit से ज्यादा Data Web Server पर Process होने के लिए Submit कर देता है।

414 Request URL Too Long

ये Status Code इस बात को Represent करता है कि Request Message में Specified URL Acceptable Size से ज्यादा बडा है।

500+ (Server Error)

500 से 599 के बीच के सभी Status Codes इस बात को Indicate करते हैं कि Web Browser द्वारा भेजा गया Request Message पूरी तरह से ठीक था लेकिन Server की किसी समस्या के कारण Request पूरी नहीं हो सकी। ये Codes इस बात को Indicate करते हैं कि वांछित Response प्राप्त न हो पाने का Fault *Server Side* में है न कि Client Side में। इस Range में सामान्य रूप से उपयोग में आने वाले Status Codes का Description निम्नानुसार हैः

500 Internal Server Error

ये Status Code सामान्यतः तब Return होता है, जब Server Side में कोई Script Run हो रही होती है और उस Script में किसी तरह का Error Trigger हो जाता है।

501 Not Implemented

ये Status Code सामान्यतः तब Return होता है, जब Server HTTP Method को ठीक से समझ नहीं पाता या Support नहीं करता।

502 Bad Gateway

ये Status Code सामान्यतः **Proxy Server** द्वारा तब Return होता है, जब Client व Server के बीच Data Transfer ठीक से नहीं हो पाता। जिसका मतलब ये है कि Web Server या तो Request Message को ठीक से समझ नहीं पाता अथवा इस बात के लिए Sure नहीं होता कि Web Server द्वारा Return होने वाला Response Data Web Client तक पहुंचेगा या नहीं

503 Service Unavailable

ये Status Code सामान्यतः तब Return होता है, जब या तो Web Server **Overload** हो जाता है या फिर **Scheduled Maintenance Period** में होता है।

504 Gateway Timeout

ये Status Code सामान्यतः तब Return होता है, जब Web Client व Web Server के बीच स्थित Proxy Server, Client व Destination के बीच Messages को ठीक तरह से Forward नहीं कर पा रहा होता है।

Status Codes व **Request/Response** से सम्बंधित उपरोक्त Discussion में बताए गए Concepts हमारे लिए तब उपयोगी होते हैं, जब हम AJAX Technology को Use करते हैं और AJAX वर्तमान समय में एक बहुत ही उपयोगी तकनीक है, जिसका प्रयोग करके हम Powerful ***Dynamic Websites*** व ***Web Applications*** Create कर सकते हैं।

Development Environment Setup

आप कोई भी नई Programming Language सीखना चाहते हों, सीखने का सबसे बेहतर तरीका यही है कि उस Language से संबंधित Basics व Fundamentals को छोटे–छोटे Programs बनाते हुए सीखा जाए और Program बनाने के लिए हमें हमेंशा किसी न किसी Text Editor या IDE की जरूरत होती है।

JavaScript भी एक प्रकार की Programming Language या ज्यादा बेहतर शब्दों में कहें, तो एक प्रकार की **Client Side Scripting Language** है, इसलिए इससे पहले कि हम इस Language को समझें, हमें JavaScript Programs को Develop करने से सम्बंधित Basic Environment Setup करने की जरूरत है, ताकि पुस्तक में आगे आने वाले Program Codes की Working को आसानी से समझा जा सके।

अन्य सभी Programming, Scripting व Markup Languages की तरह ही JavaScript Programs को भी हम एक Simple Text Editor में लिख सकते हैं, लेकिन चूंकि JavaScript एक Interpreter Based Programming Language है और JavaScript का Interpreter सामान्यतः

Web Browser के अन्दर ही In-Built होता है, इसलिए सामान्यतः JavaScript को Web Pages को Interactive बनाने के लिए Use किया जाता है।

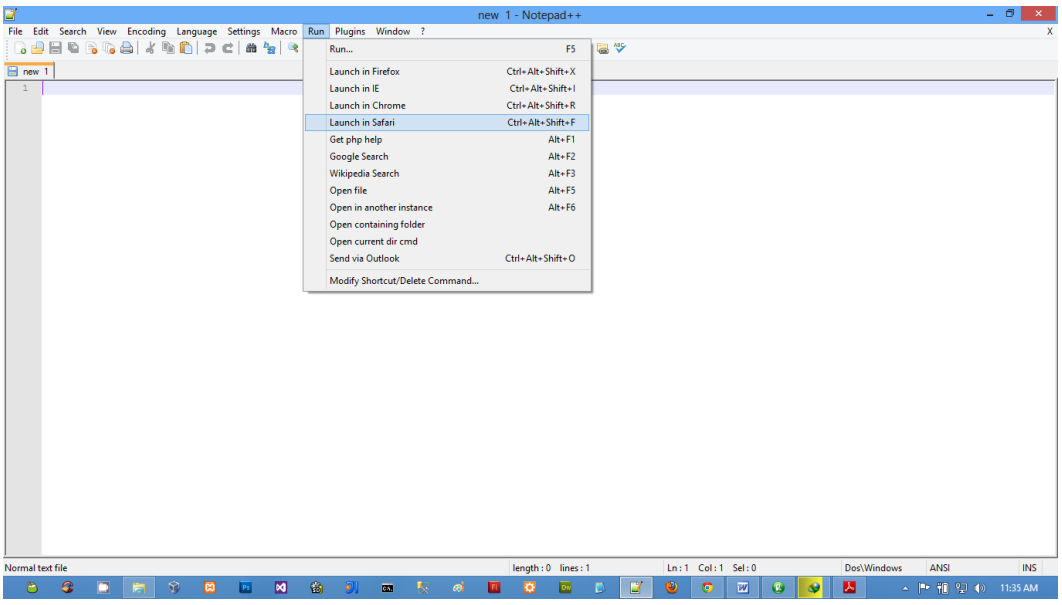
परिणामस्वरूप JavaScript Codes मूल रूप से Web Pages के लिए ही उपयोगी होते हैं और Web Pages Create करने के लिए जितने भी **IDE** (*Integrated Development Environment*) वर्तमान में उपलब्ध हैं, उन सभी को JavaScript Codes को लिखने के लिए Use किया जा सकता है। जैसे *Adobe DreamWeaver, Microsoft Visual Studio, NetBeans, Eclipse* आदि।

चूंकि किसी भी Program को Develop करने में कई Steps Involved होते हैं, जैसे कि Source Codes लिखना, उन्हें Compile या Interpret करना, Bugs को Identify करना, उन्हें Debug करना, Maintain करना, Test करना व Deploy करना। इन सभी कामों को एक ही स्थान पर पूरा करने के लिए यदि कोई Software बना लिया जाए, तो उस Software को **IDE** (Integrated Development Environment) कहते हैं।

हालांकि IDE किसी भी Program को Develop करने में काफी मदद करते हैं, लेकिन फिर भी यदि हम कोई नई Language सीखने के लिहाज से देखें, तो IDE फायदा करने के स्थान पर नुकसान करते हैं।

यानी यदि JavaScript आपके लिए बिल्कुल नई Language है, तो किसी IDE को Use करने के स्थान पर Simple Text Editor का प्रयोग करते हुए JavaScript Codes लिखना आपके लिए ज्यादा फायदेमन्द रहेगा और **Notepad++** किसी भी नई Programming Language को सीखने के लिए मेरा Favorite Text Editor है, जबकि Client Side Web Technologies (HTML, CSS, JavaScript, etc...) IDE के रूप में मुझे Eclipse आधारित **Aptana Studio** पसन्द है।

तो सबसे पहले **Notepad++** Text Editor को *http://www.notepad-plus-plus.org /download/* Website से Download करके अपने Computer पर Install कीजिए। ये Text Editor Free Available है। Install करके Open करने पर ये कुछ निम्नानुसार दिखाई देता है

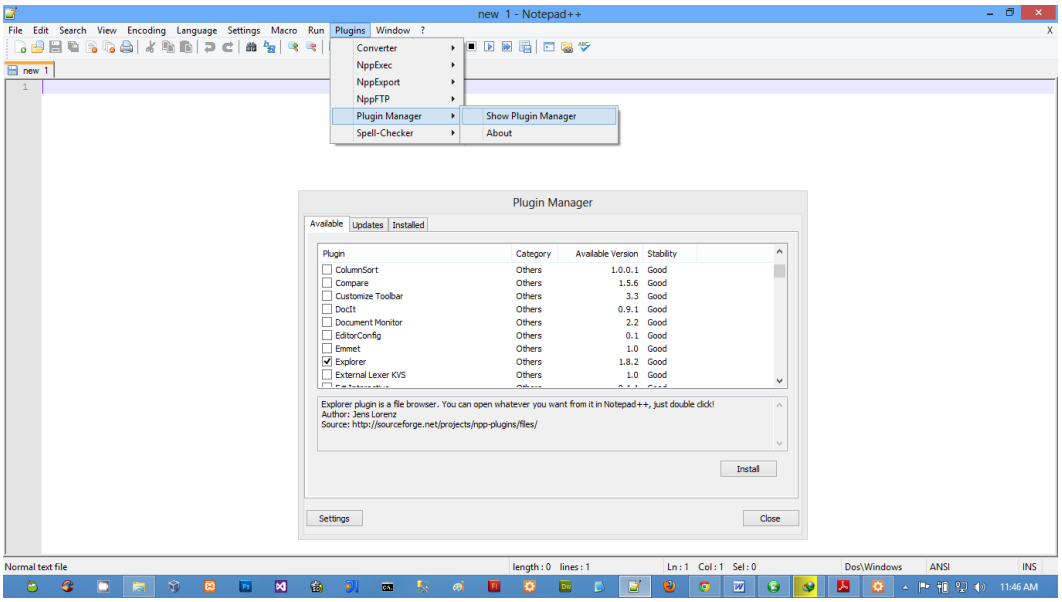


चूंकि HTML, CSS व JavaScript जैसी Scripting Languages, Web Browser में ही Interpret होते हैं और इनका Server Side से कोई Direct Connection होना जरूरी नहीं होता,

इसलिए JavaScript Codes के Effects को समझने के लिए हमें हमारे Local Computer पर किसी Local Web Server को Install करने की जरूरत नहीं है, बल्कि JavaScript Codes हमेंशा किसी न किसी HTML Web Page से Link या HTML Web Page में Embed होते हैं, इसलिए जैसे ही हम Web Page को किसी Web Browser में Open करते हैं, JavaScript Interpret होने लगता है।

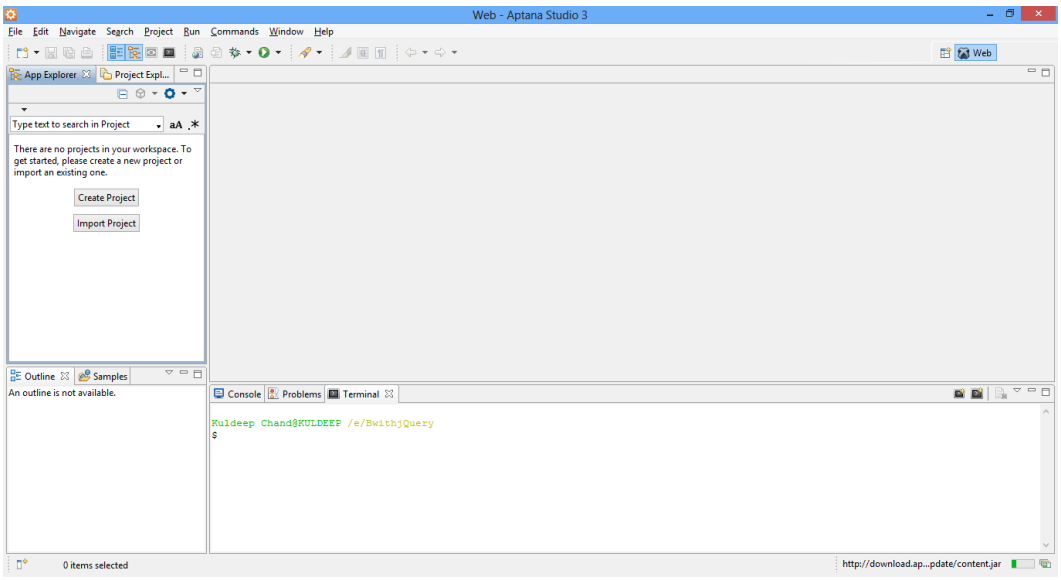
यानी JavaScript Programming सीखने के लिए हमें किसी External Software की जरूरत नहीं है। हमें केवल एक Text Editor की जरूरत है, जहां हम अपने HTML, CSS व JavaScript Codes को लिख सकें व एक Web Browser की जरूरत है, जहां हम हमारे JavaScript Codes के Output को देख सकें।

Notepad की Capabilities को Extend करने के लिए हम इसमें अपनी जरूरत के अनुसार विभिन्न प्रकार के Plug-ins को भी Install कर सकते हैं। विभिन्न प्रकार के Plug-ins Install करने के लिए हमें Notepad++ के **Plug-in** Menu के "**Plugin Manager**" Sub-Menu के "**Show Plugin Manager**" Option को Click करना होता है और हमारे सामने निम्नानुसार एक Dialog Box Open होता है, जिसमें हम उन Plug-ins को Select करके Install कर सकते हैं, जिन्हें हम हमारे Notepad++ Text Editor में Include करना चाहते हैं जबकि Install होने के बाद उस Plugin को उपयोग में लेने के लिए के लिए भी हमें इसी "**Plug-in**" Menu में ही जाना होता है।



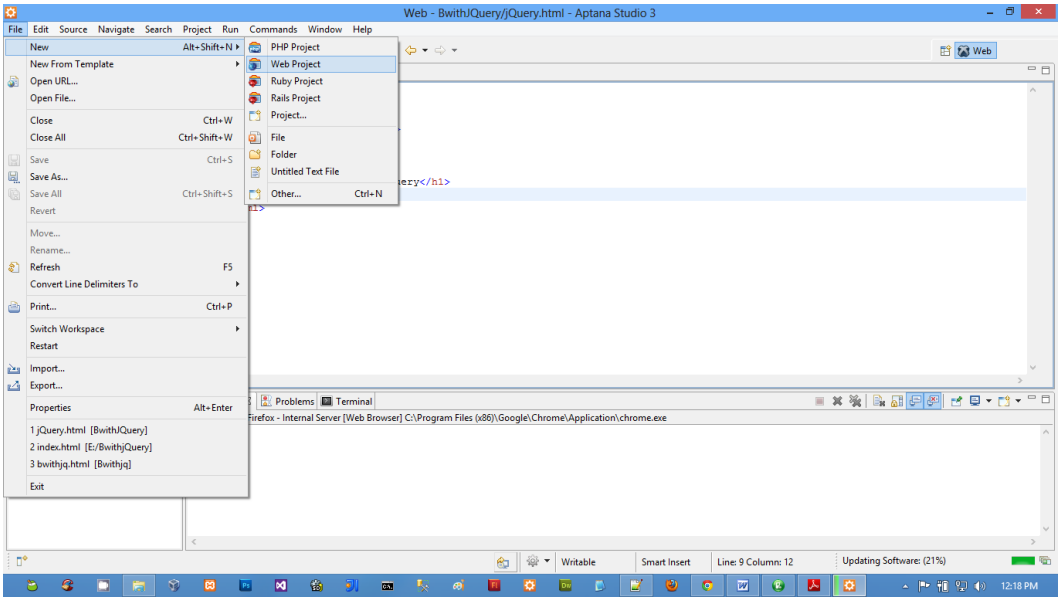
इसके अलावा यदि आप कोई IDE Use करना चाहते हैं, तो आप मेरा Favorite IDE **Aptana Studio** Use कर सकते हैं। ये एक Advance IDE है, इसलिए इसे इसकी पूरी क्षमता के साथ Use करने के लिए आपको कुछ Configuration करने की जरूरत पड सकती है।

ये एक ऐसा IDE है, जिसे Use करने पर आप अपना सारा Code एक ही स्थान पर लिख सकते हैं और उसे इसी Studio में उपलब्ध Internal Web Browser में Run करके उसका Output भी इसी Browser में देख सकते हैं। इस IDE को आप http://www.aptana.com/products/studio3/download Website से Download कर सकते हैं और ये भी पूरी तरह से Free है। Install करके Open करने पर ये IDE कुछ निम्नानुसार दिखाई देता है:

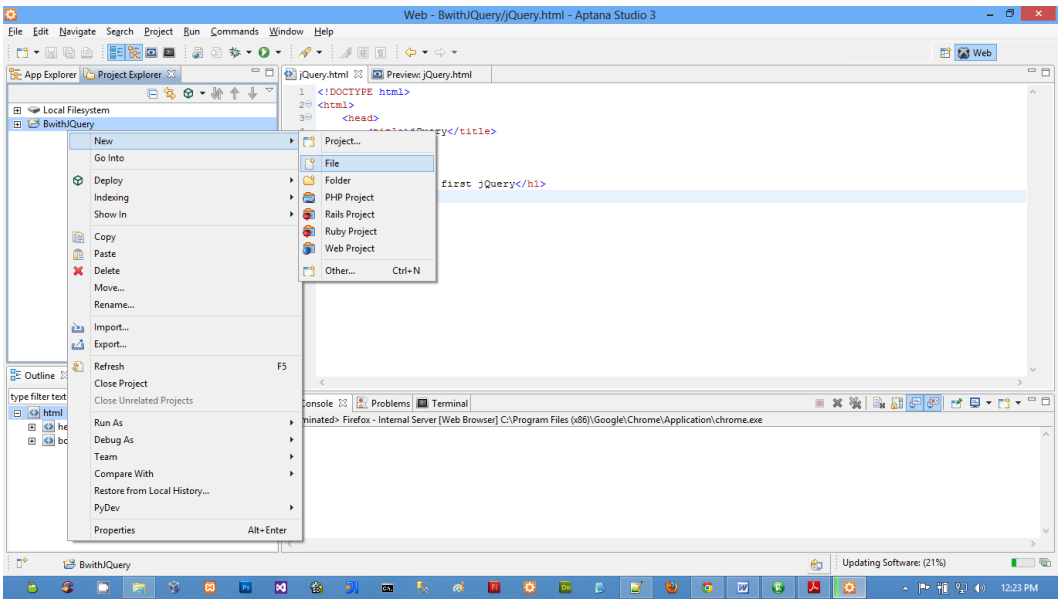


इस IDE की विशेषता ये है कि इस IDE में ही एक Local Web Server व Internal Web Browser भी है। जिसकी वजह से हमें हमारे Program को Test या Debug करने के लिए ***Text Editor*** व ***Web Browser*** के बीच Switch नहीं करना पडता।

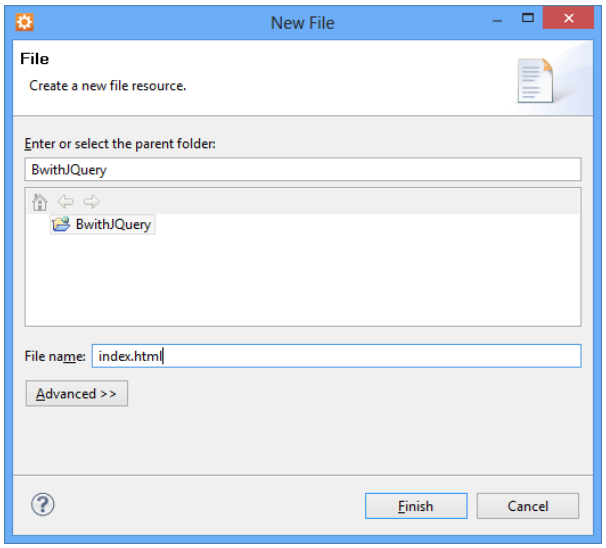
इस IDE को Use करने के लिए सबसे पहले हमें निम्न चित्रानुसार Option को Click करके एक नया **Web Project** Create करना होता है:



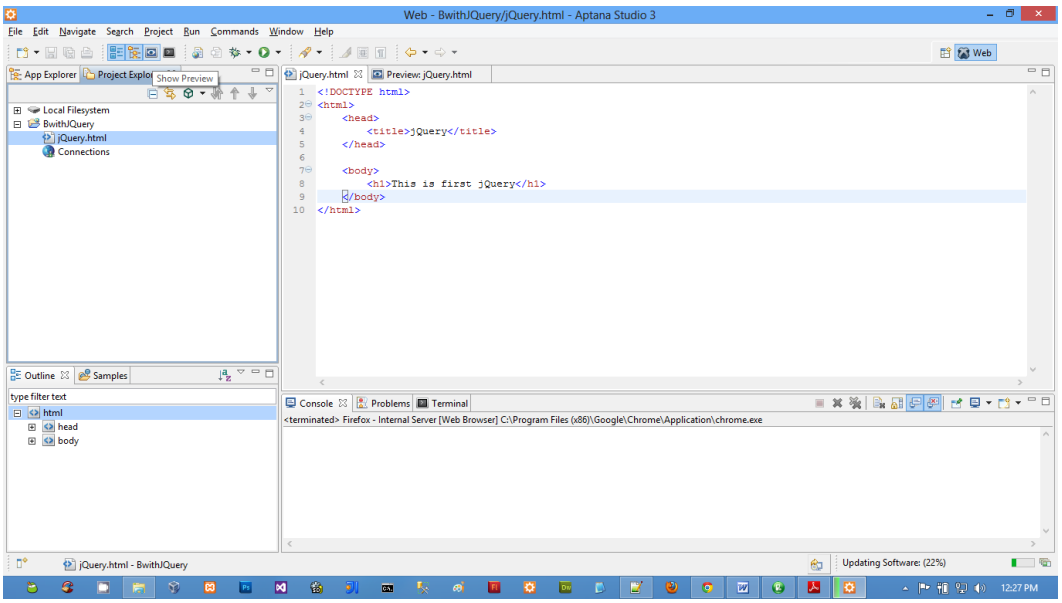
एक Dialog Box Display होता है, जहां हमें हमारे Project का नाम Specify करके Next Button पर नहीं बल्कि **Finish** Button पर Click करना होता है। ऐसा करते ही एक नया Project Create हो जाता है, जिसे हम IDE के Left Side में दिखाई देने वाले **"Project Explorer"** Tab में देख सकते हैं। फिर निम्न चित्रानुसार नई File Create करना होता है:



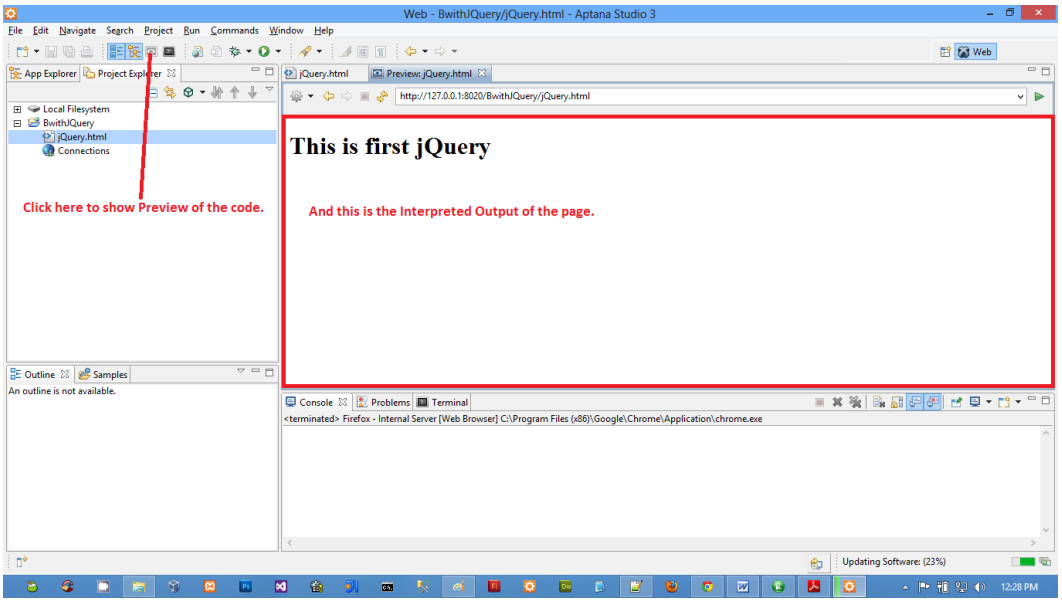
चित्र में दिखाए अनुसार "New => File" पर Click करते ही हमारे सामने एक Dialog Box आता है, जिसमें हमें हमारी File का नाम जैसे कि "**index.html**" Specify करके "**Finish**" Button पर Click करना होता है:



इस प्रकार से हमारे Project में एक नई File Add हो जाती है, जिसमें हम निम्नानुसार HTML, CSS या JavaScript Code लिख सकते हैं:



इस Web Page में HTML, CSS, JavaScript Codes लिखने के बाद उसका Output देखने के लिए हमें अगले चित्र में दिखाए अनुसार IDE के Standard Toolbar में दिए गए "**Show Preview**" Icon को Click करना होता है और हमें हमारे Page का Output निम्न चित्रानुसार दिखाई देने लगता है:



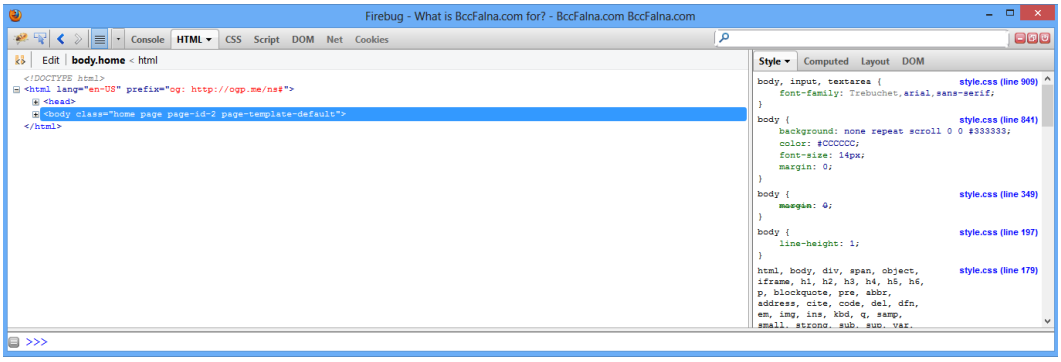
इस IDE के अलावा भी कई और IDEs हैं, जिनका प्रयोग Web Pages Create करने के लिए किया जा सकता है। उदाहरण के लिए आप NetBeans भी Use कर सकते हैं, जो कि मूल रूप से Java Development के लिए Oracle Company द्वारा Provide किया गया IDE है, लेकिन हम इसे Web Development के लिए भी Use कर सकते हैं और ये IDE भी काफी Powerful व Free Available हैं।

जबकि आप Adobe DreamWeaver या Microsoft Visual Studio IDE का भी प्रयोग Web Development के लिए कर सकते हैं, लेकिन ये IDE Free नहीं हैं बल्कि काफी महंगे हैं।

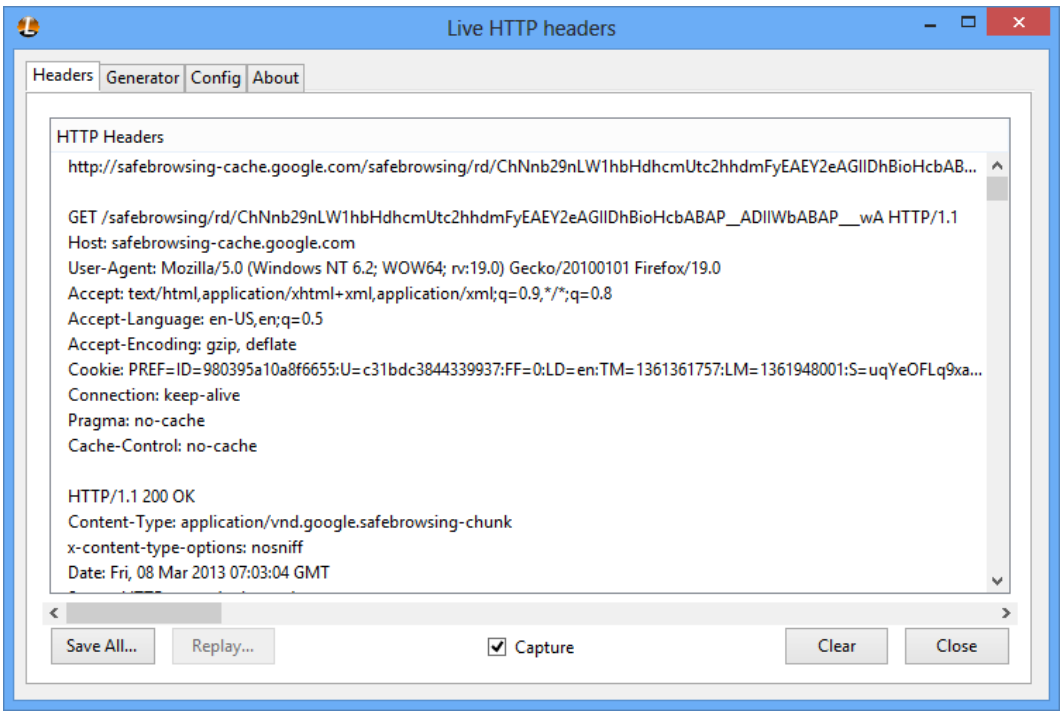
हालांकि आप **Notepad++** या **Aptana Studio** का प्रयोग करके Web Pages Create कर सकते हैं, जिनमें JavaScript Codes लिखकर उनका Effect समझ सकते हैं, लेकिन फिर भी Development के समय विशेष रूप से Codes की Debugging करते समय व Language की Internal Working को बेहतर तरीके से समझने के लिए हमें कुछ और Special प्रकार के Tools की अक्सर जरूरत पडती है और सामान्य रूप से ये **Tools,** Web Browser बनाने वाली Companies ने Default रूप से अपने Web Browser में दे रखा होता है, जिसे "**Developer Tools**" कहते हैं, और विभिन्न Web Browsers में सामान्यतः इन्हें **F12** Function Key Press करके **On/Off** किया जा सकता है।

फिर भी Developer Tools के मामले में Mozilla Firefox Web Browser सबसे Powerful Developer Tools Plugin के रूप में Install करने की सुविधा देता है और ये Tool भी **F12** Function Key द्वारा Enable/Disable कर सकते हैं। यानी यदि आप Mozilla Firefox Web Browser Use कर रहे हैं, तो निम्न Tools को अपने Web Browser में Extension के रूप में जरूर Install करें:

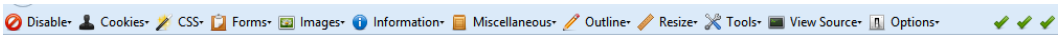
1ण http://www.getfirebug.com/
ये Tool वास्तव में सभी Web Developers के लिए एक बहुत ही उपयोगी Tool है, क्योंकि ये Tool Web Page Development व Debugging से संबंधित लगभग जरूरी Tools का एक Collection है।



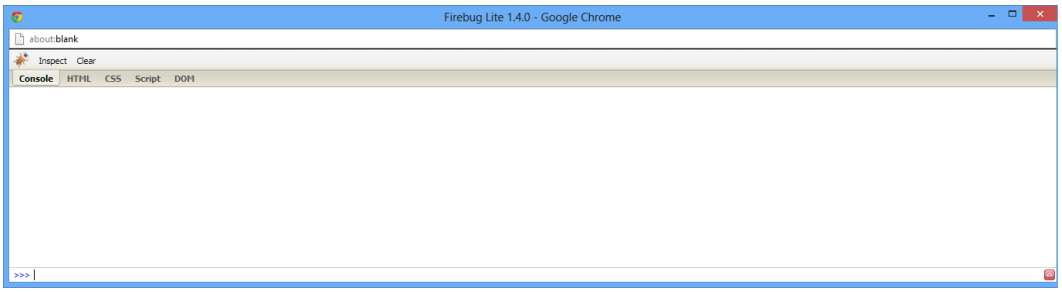
2ण http://livehttpheaders.mozdev.org/
इस Tool का प्रयोग करके हम Web Browser व Web Server के बीच Transfer होने वाले Message की Details प्राप्त कर सकते हैं।



3ण http://chrispederick.com/work/web-developer/
ये Tool Current Web Page से सम्बंधित लगभग सभी Elements की जानकारी व उन्हें Handle व Control करने की सुविधा देता है।

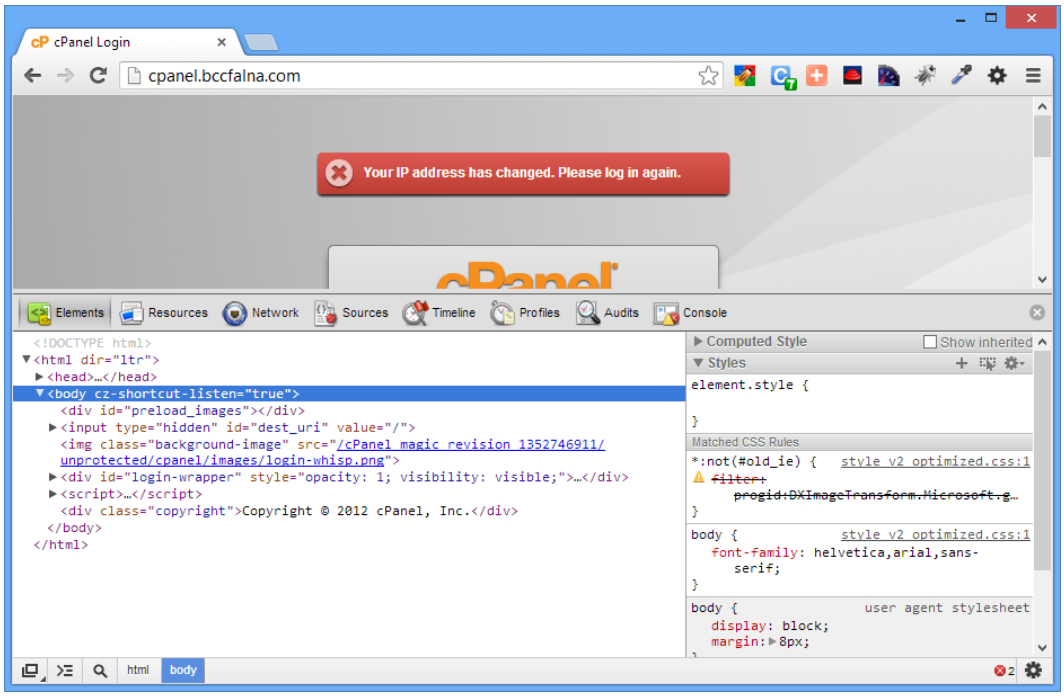


यदि आप Firefox Web Browser को ज्यादा उपयोग में लेते हैं, तो इन तीनों Tools के साथ कुछ समय व्यतीत करना आपके लिए काफी फायदेमन्द रहेगा। लेकिन यदि आप Google Chrome Web Browser को ज्यादा उपयोग में लेते हैं, तो उपरोक्त सभी Tools के Lite या Alternative Versions, Google Chrome Web Browser के लिए भी Plug-in के रूप में Available हैं, जो कि **F12** Function Key Press करने पर कुछ निम्नानुसार दिखाई देते हैं:

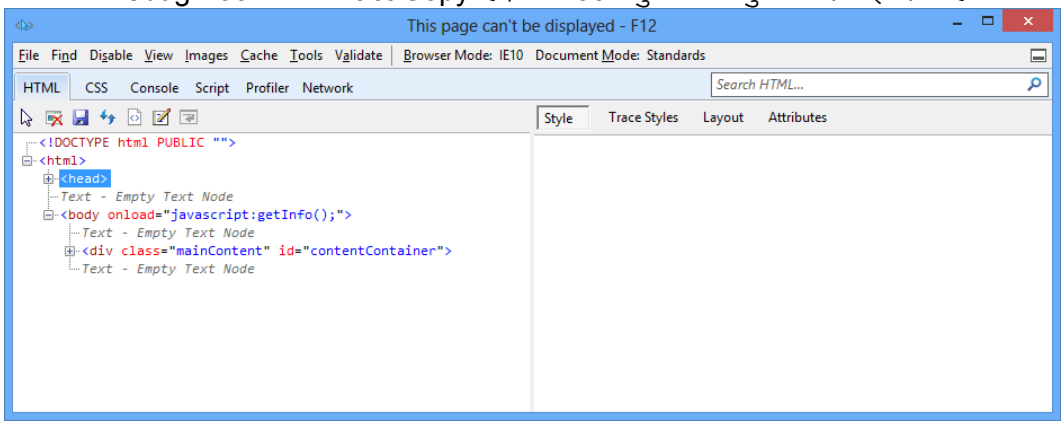


Google Chrome के लिए Firebug Tool का ये एक Lite Version है। इसके अलावा Google Chrome का स्वयं का भी एक Developer Tool है, जिस को उस स्थिति में **F12** Function Key द्वारा Activate किया जा सकता है, जबकि आपने Google Chrome में "**Firebug Lite**" Version को Install न किया हो। लेकिन यदि आपने "Firbug" के Lite Version Extension को Install

किया है, तो इस Default **Developer Tools** को Open करने के लिए आपको Google Chrome के **Tools** Menu में जाकर "**Developer Tools**" Option को Click करना होगा। ये Tool कुछ निम्नानुसार दिखाई देता है।



इसके अलावा Microsoft ने अपने Latest Web Browser के साथ भी अपना एक Developer Tool Provide किया है और वह भी **F12** Function Key द्वारा ही Activate होता है, जो कि लगभग Firebug Took की Exact Copy है। ये Tool कुछ निम्नानुसार दिखाई देता है:



इनके अलावा Apple Safari व Opera Web Browsers का भी अपना Develop Tool है। उपरोक्त सभी Tools देखकर आप समझ ही गए होंगे कि ये सभी Tools लगभग एक समान ही हैं। इसलिए आप चाहे जो Web Browser Use कर रहे हों, आपको इन Tools को अच्छी तरह से Use करना आना ही चाहिए।

वैसे भी यदि आप Web Developer बनना चाहते हैं, तो आपके Computer में सभी Modern Web Browsers Installed होने चाहिऐं और आपको अपने Web Page को सभी Modern Web

Browsers में Test करना चाहिए, ताकि आपको पता चल सके कि एक ही Web Page अलग–अलग Web Browsers में कितना अलग दिखाई दे सकता है।

इसके अलावा हालांकि हमने कुछ Tools के बारे में Discuss किया, लेकिन विभिन्न प्रकार की Requirements को पूरा करने हेतु विभिन्न Web Browsers के बहुत सारे Tools Plug-in के रूप में Available हैं, जिन्हें सुविधानुसार जरूर उपयोग में लेना सीखना चाहिए।

साथ ही इन अलग–अलग Web Browsers के "**Developer Tools**" में भी कुछ Special Types के अलग–अलग Options हैं, जो किसी दूसरे Web Browser के Developer Tool में उपलब्ध नहीं हैं। इसलिए सभी Web Browsers के Developer Tools को अच्छी तरह से समझना आपके लिए उपयोगी रहेगा

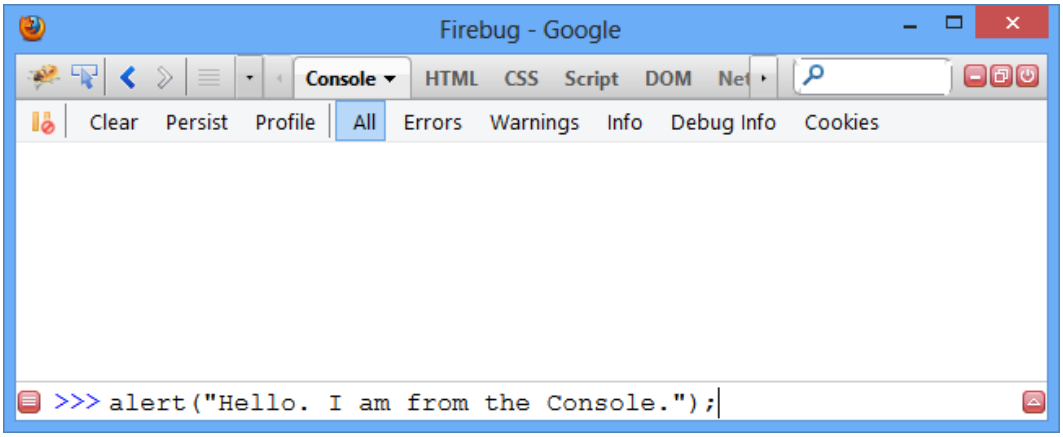
उदाहरण के लिए Internet Explorer का Developer Tools Use करके यदि हम Web Page के किसी Code में Change करते हैं, तो हम उस Code को Hard Disk पर एक अलग File के रूप में Save करके रख सकते है, जबकि ये सुविधा किसी भी अन्य Web Browser के Developer Tools में नहीं है।

यानी यदि आप कोई IDE या Text Editor Use न करें, तो आप सीधे ही Internet Explorer के Developer Tools को एक IDE की तरह Use करते हुए भी JavaScript Codes को Interpret कर सकते हैं, नया Web Page Create कर सकते हैं, उसकी Stylesheet बना सकते हैं।

यानी हर Web Browser के Developer Tools की अपनी विशेषता है इसलिए आपको सभी Web Browsers के Developer Tools के बारे में ज्यादा से ज्यादा जानना चाहिए ताकि आपको पता रहे कि किसी Specific Type की Requirement को पूरा करने के लिए आपको कौनसे Web Browser के Developer Tools की जरूरत है।

Developer Tools Console

जब आप विभिन्न Web Browsers के Developer Tools या Firebug Tool को Inspect करेंगे, तो आप देखेंगे कि उन सभी Tools में "**Console**" नाम का एक Tab है। ये वह स्थान है, जहां पर आप Directly JavaScript Codes लिखकर सीधे ही Web Browser में Code की Functionality का प्रभाव देख सकते हैं। यही नहीं, Web Browser में Loaded किसी भी Web Page को इस Console में JavaScript Code लिखकर उस Page पर JavaScript Code के Effect को देखा जा सकता है।



How to Get Complete PDF EBook

आप **Online Order** करके **Online** या **Offline** Payment करते हुए इस Complete EBook को तुरन्त Download कर सकते हैं।

Order करने और पुस्तक को Online/Offline Payment करते हुए खरीदने की पूरी प्रक्रिया की विस्तृत जानकारी प्राप्त करने के लिए आप **BccFalna.com** के निम्न Menu Options को Check Visit कर सकते हैं।

How to Make Order

How to Order?

How to Buy Online

How to Pay Online using PayUMoney

How to Pay Online using Instamojo

How to Pay Online using CCAvenue

How to Buy Offline

How to Pay Offline

Bank A/c Details

जबिक हमारे Old Buyers के **Reviews** भी देख सकते हैं ताकि आप इस बात का निर्णय ले सकें कि हमारे Buyers हमारे PDF EBooks से कितने Satisfied हैं और यदि आप एक से अधिक EBooks खरीदते हैं, तो **Extra Discount** की Details भी Menubar से प्राप्त कर सकते हैं।