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।

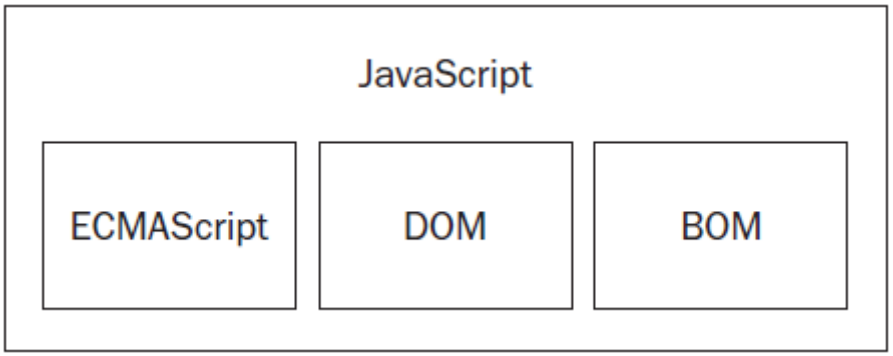
फल स्वरूप इस Committee ने अन्तिम रूप से **ECMAScript-262** नाम का एक Standard तैयार किया और JavaScript का नाम बदलकर **ECMAScript** हो गया। यानी आज की जो JavaScript है वह वास्तव में JavaScript नहीं बल्कि ECMAScript है।

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।

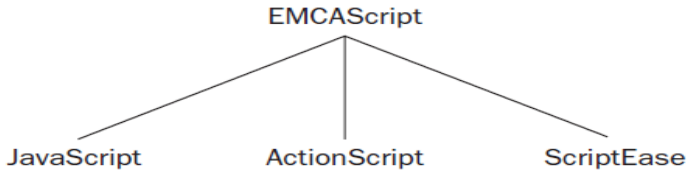
DOM यानी Document Object Model भी एक Extension ही होता है जो अपने Core के रूप में ECMAScript के Type व Syntax को Use करता है तथा Host Environment, जो कि Web Browser भी हो सकता है और कोई अन्य Software भी, Additional Functionality Provide करता है। सामान्यतः अन्य Host Environments के रूप में **ScriptEase**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



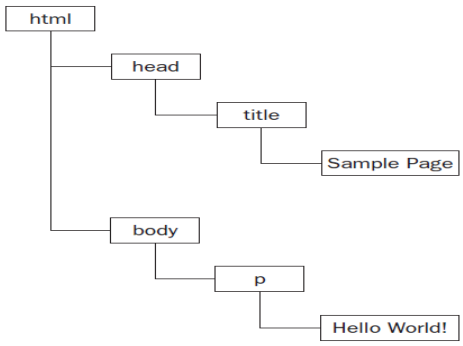
वर्तमान समय में ECMAScript का 5th Version आ चुका है, लेकिन इसे पूरी तरह से विभिन्न Web Browsers में Implement नहीं किया गया है। वर्तमान समय में Internet Explorer, FireFox, Safari, Chrome व Opera जो कि सबसे ज्यादा Use किए जाने वाले Web Browsers हैं, ने **ECMAScript3.1** Specification को पूरी तरह से Implement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

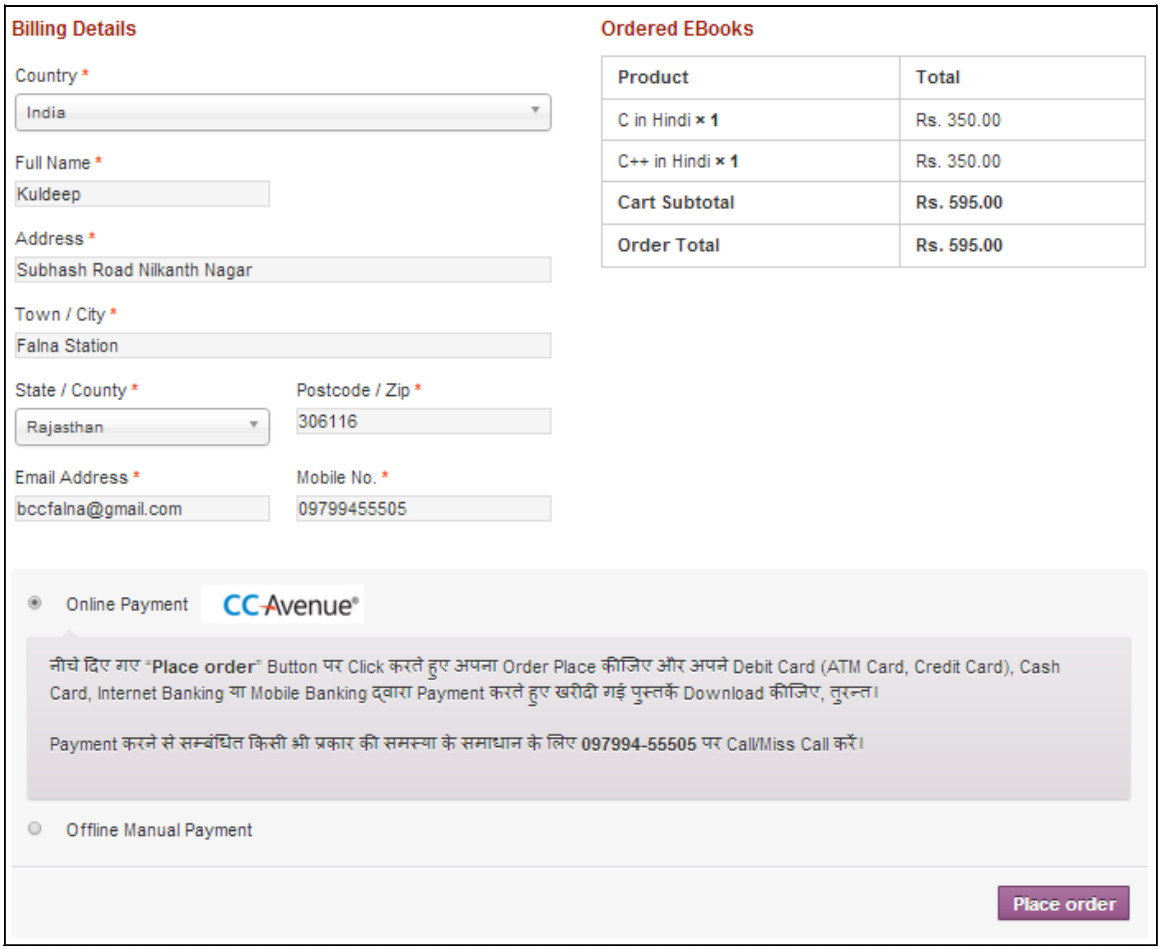
जबकि Backup के लिए केवल "**en**" Specified है, जो कि इस बात की Information है कि बिना किसी Geographical Locale की स्थिति में Default रूप से ये Web Browser English Language को Support करता है। Web Server इस Information को उस स्थिति में Ignore कर देता है, जब Web Browser द्वारा Requested Page केवल एक ही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

How to Buy from BccFalna.com

इस Website पर उपलब्ध सभी **Saleable Hindi EBooks** के साथ "**ADD TO CART**" नाम का एक Button Attached है। आप जो भी पुस्तक खरीदना चाहते हैं, उसके साथ Associated **ADD TO CART** Button को Click करते ही वह पुस्तक आपके **Shopping Cart** में Add हो जाती है:

यदि आप अपने Shopping Cart में कई पुस्तकें Add करते हैं, तो **Extra Discount** प्राप्त होता है, जो कि **Discount Amount** व **Discount Amount** घटाने के बाद सभी पुस्तकों के **Total Payable Amount** के रूप में इसी **My Shopping Cart** में उपरोक्त चित्रानुसार दिखाई देता है।

सभी वांछित पुस्तकें अपने **Shopping Cart** में Add करने के बाद अपना **Order Place** करने हेतु **Checkout** Button को Click करना होता है। परिणामस्वरूप निम्नानुसार **Checkout Page** Display होता है, जहां आपको अपनी **Billing Details** को Specify करके अपना **Payment Mode** Select करना होता है:

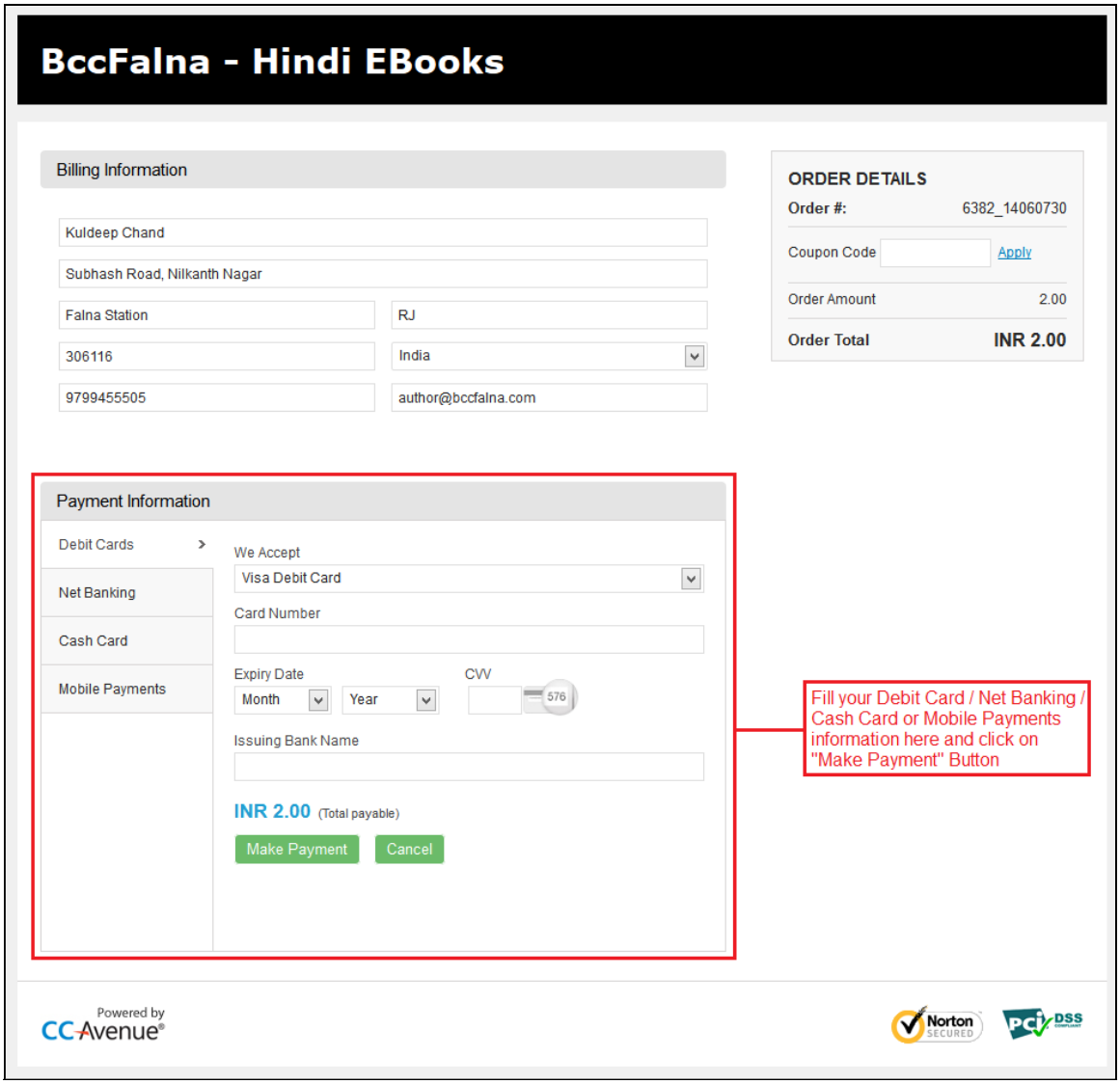


यदि आप इन पुस्तकों को खरीदने के लिए **Total Payable Amount** का भुगतान अपने Debit Card (*ATM, Credit Card), Cash Card* अथवा *Net Banking*) द्वारा घर बैठे करना चाहते हैं, तो आपको **Online Payment** Option Select करना होता है।

लेकिन यदि आपके पास किसी प्रकार का CCAvenue Supported **Debit Card** (ATM, Credit Card) या *Cash Card* नहीं है, न ही आपके पास किसी Bank की *Internet* या *Mobile Banking* सुविधा है, तो उस स्थिति में अपना Offline Order Place करने के लिए **Offline Manual Payment** Option को Select कर सकते हैं।

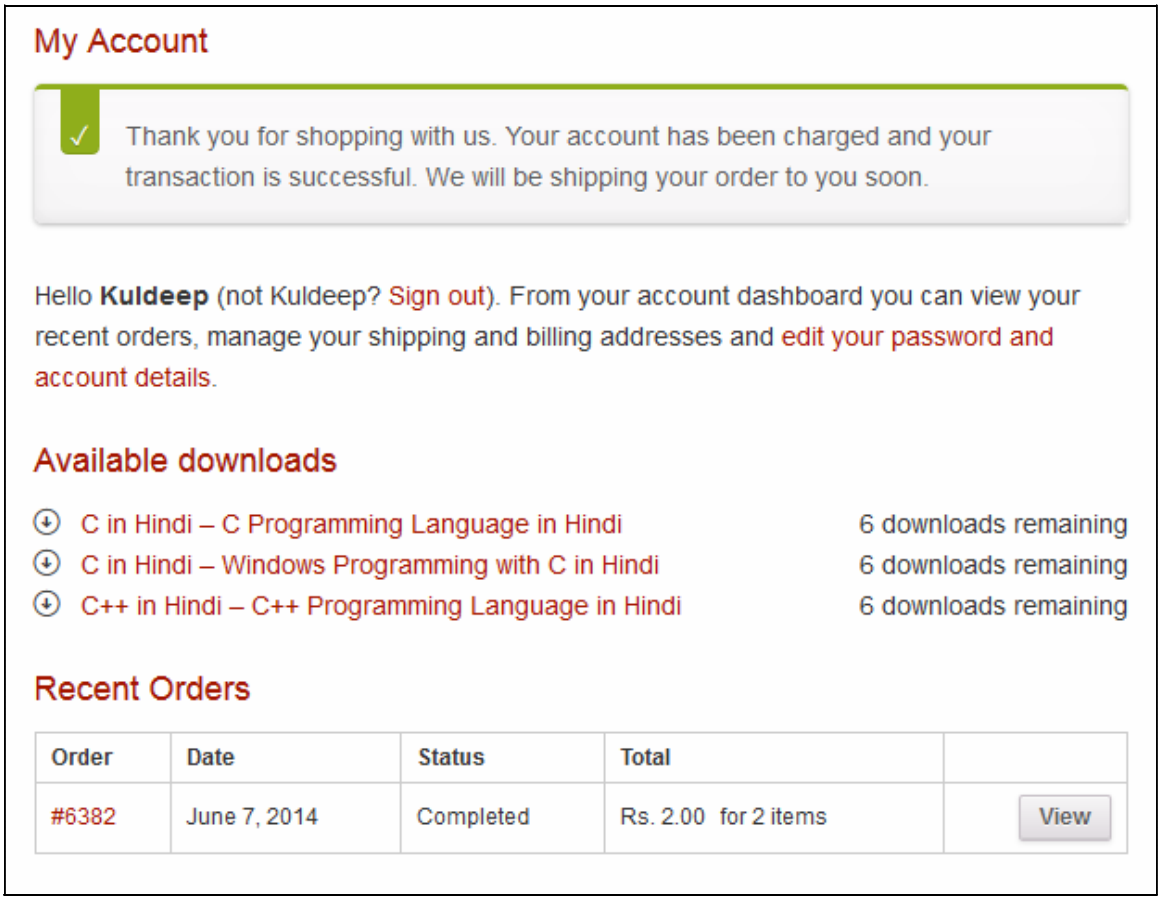
Online Payment using CCAvenue

जब आप **Online Payment** Option को Select करते हुए "**Place order**" Button पर Click करते हैं, तो आपके सामने निम्नानुसार Page Display होता है:



अपनी सुविधा अपने *Debit Card (ATM, Credit Card), Net Banking, Cash Card* या *Mobile Payments* Option को Select कीजिए और उपरोक्त चित्र में दर्शाए अनुसार सभी जरूरी Payment Information को Fill करके **Make Payment** Button पर Click कीजिए तथा आगे आने वाले Payment Security से सम्बंधित Step (Login/Password/Pin) Follow कीजिए, ताकि आपका Total Payable Amount आपके Bank A/c से हमारे Bank A/c में Transfer हो सके।

आपका Payment Transfer होते ही आप निम्न चित्रानुसार **My Account** Page पर पहुंच जाते हैं, जहां पर आप द्वारा Order की गई सभी पुस्तकों के Download Links होते हैं, साथ ही आपको Automatically एक Email भी Send कर दिया जाता है, जिसमें आप द्वारा खरीदी गई पुस्तकों के **Download Links** होते हैं:



My Account

Thank you for shopping with us. Your account has been charged and your transaction is successful. We will be shipping your order to you soon.

Hello **Kuldeep** (not Kuldeep? Sign out). From your account dashboard you can view your recent orders, manage your shipping and billing addresses and edit your password and account details.

Available downloads

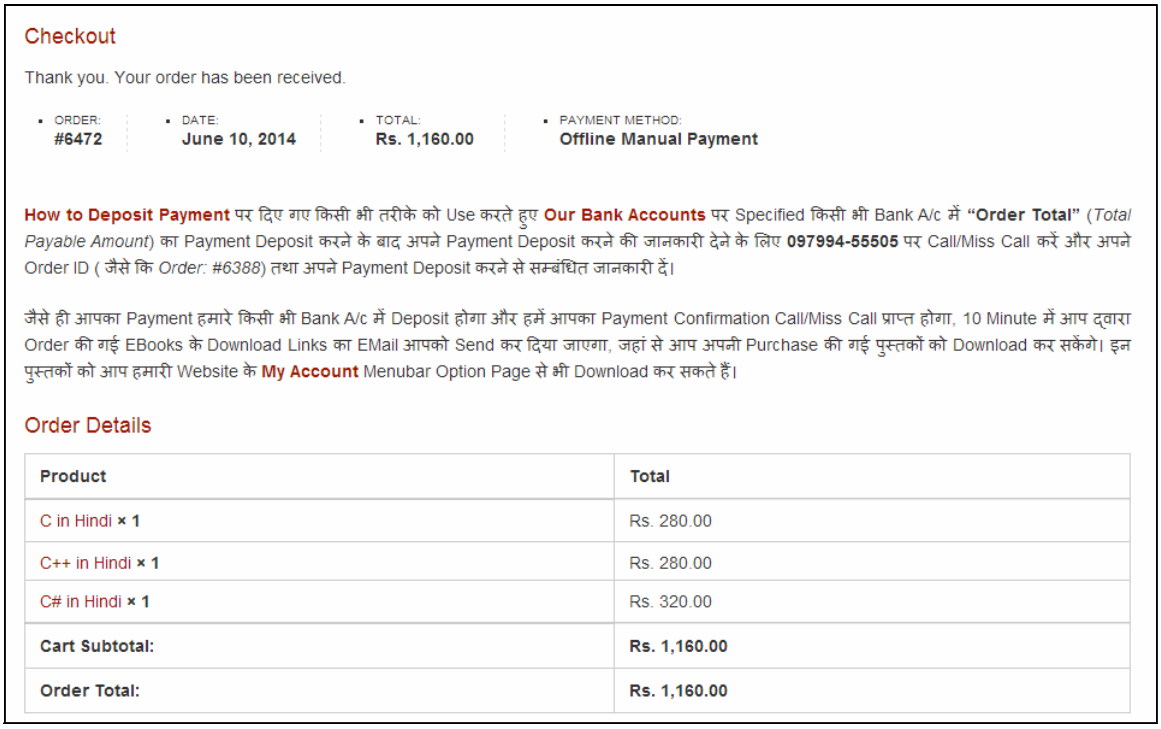
C in Hindi – C Programming Language in Hindi	6 downloads remaining
C in Hindi – Windows Programming with C in Hindi	6 downloads remaining
C++ in Hindi – C++ Programming Language in Hindi	6 downloads remaining

Recent Orders

Order	Date	Status	Total	
#6382	June 7, 2014	Completed	Rs. 2.00 for 2 items	View

Offline Payment using Manual Ways

जब आप **Offline Manual Payment** Option को Select करते हुए "**Place order**" Button पर Click करते हैं, तो Click करते ही आपका Order Place हो जाता है और आपके सामने निम्नानुसार Page Display होता है:



Checkout

Thank you. Your order has been received.

- ORDER: #6472
- DATE: June 10, 2014
- TOTAL: Rs. 1,160.00
- PAYMENT METHOD: Offline Manual Payment

How to Deposit Payment पर दिए गए किसी भी तरीके को Use करते हुए **Our Bank Accounts** पर Specified किसी भी Bank A/c में **"Order Total"** (*Total Payable Amount*) का Payment Deposit करने के बाद अपने Payment Deposit करने की जानकारी देने के लिए **097994-55505** पर Call/Miss Call करें और अपने Order ID (जैसे कि *Order: #6388*) तथा अपने Payment Deposit करने से सम्बंधित जानकारी दें।

जैसे ही आपका Payment हमारे किसी भी Bank A/c में Deposit होगा और हमें आपका Payment Confirmation Call/Miss Call प्राप्त होगा, 10 Minute में आप द्वारा Order की गई EBooks के Download Links का EMail आपको Send कर दिया जाएगा, जहां से आप अपनी Purchase की गई पुस्तकों को Download कर सकेंगे। इन पुस्तकों को आप हमारी Website के **My Account** Menubar Option Page से भी Download कर सकते हैं।

Order Details

Product	Total
C in Hindi × 1	Rs. 280.00
C++ in Hindi × 1	Rs. 280.00
C# in Hindi × 1	Rs. 320.00
Cart Subtotal:	**Rs. 1,160.00**
Order Total:	**Rs. 1,160.00**

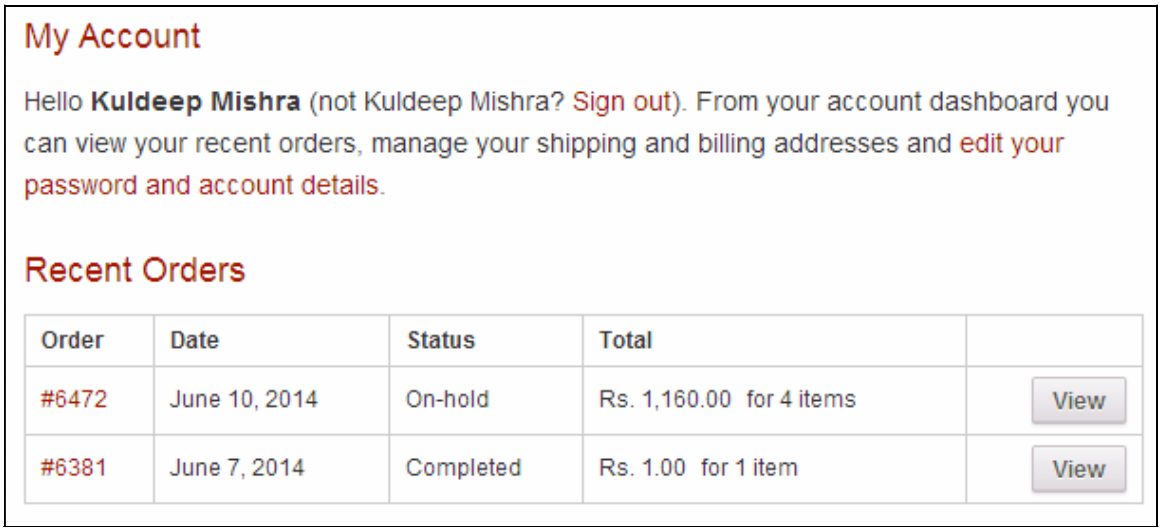
ये Webpage आप द्वारा Place किए गए Order की Information के साथ ही Order की गई पुस्तकों के **Download Links** प्राप्त करने के लिए Follow किए जाने वाले अगले Step की जानकारी भी देता है, साथ ही इस Page पर दिखाई देने वाली सारी Information आपको आपके Email पर भी Send कर दी जाती है, जिन्हें Follow करते हुए आप अपने **Total Payable Amount** का **Offline Manual Payment** करके अपनी Order की गई पुस्तकों के Download Links प्राप्त कर सकते हैं।

जब आप इस **Offline Manual Payment** Option को Select करते हुए Order Place करते हैं, तो आपका Order तब तक **On-Hold** Status में रहता है, जब तक कि आप **Offline Manual Payment** Page पर Specified किसी भी तरीके का प्रयोग करते हुए अपना **Total Payable Amount**, हमारे **Bank A/c** में Transfer/Deposit नहीं कर देते।

अपना *Total Payable Amount* हमारे **Bank A/c** में Transfer/Deposit करने के बाद आपको हमारे Mobile No.: **097994-55505** पर Call/Miss Call करके अपने Payment Transfer/Deposit करने से सम्बंधित जानकारी देनी होती है। जैसे ही आपका Call/Miss Call हमें प्राप्त होता है, हम अपना Bank A/c Check करते हैं और जैसे ही आपका Payment हमारे Bank A/c में Transfer/Deposit होता है, हम आप द्वारा Order की गई पुस्तकों का Download Link Manually Activate कर देते हैं।

परिणामस्वरूप आपको Automatically एक EMail प्राप्त होता है, जिसमें आप द्वारा Order की गई सभी पुस्तकों के Download Links होते हैं, जिन्हें आप अगले 48 घण्टों के दौरान Download कर सकते हैं। साथ ही आपके Download Links के Activate होने की Information हम आपको Call/SMS के माध्यम से भी देते हैं।

जबकि अपने Order की Current Status देखने के लिए आप Website के Menubar में दिखाई देने वाले **My Account** Menu Option पर Click कर सकते हैं, जहां आपके Order की Current Status Information निम्न चित्रानुसार दिखाई देती है:



My Account

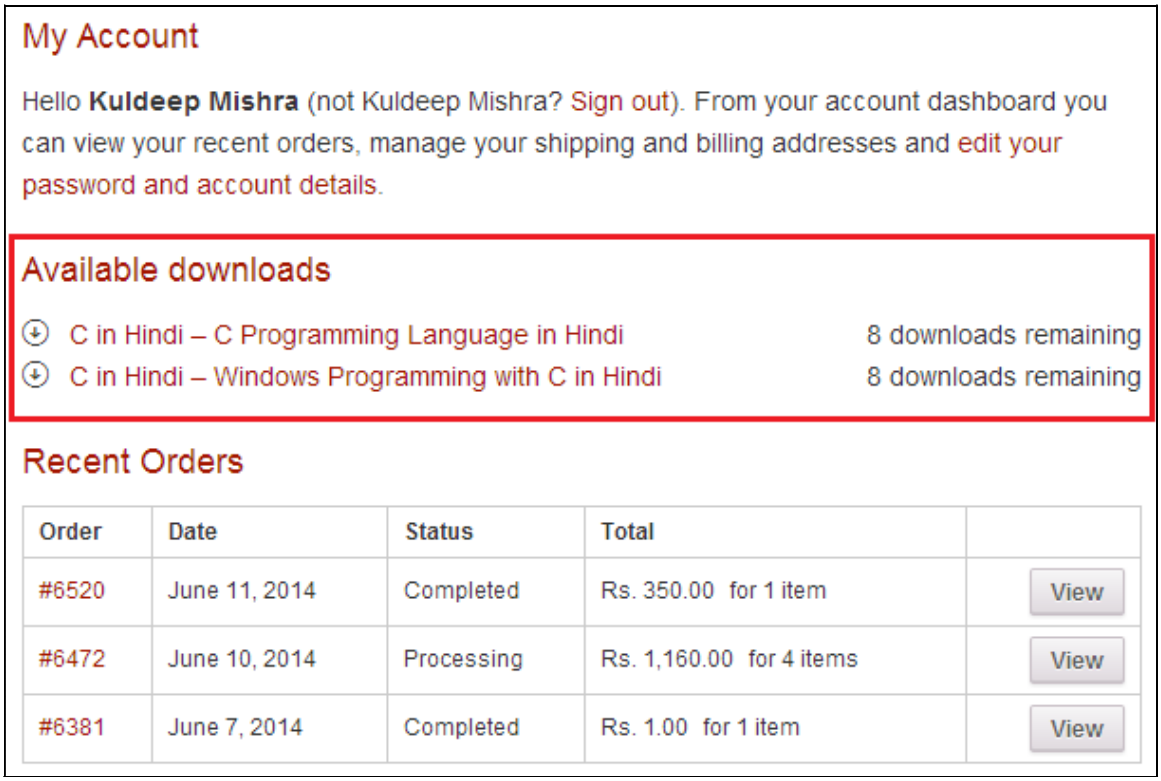
Hello **Kuldeep Mishra** (not Kuldeep Mishra? Sign out). From your account dashboard you can view your recent orders, manage your shipping and billing addresses and edit your password and account details.

Recent Orders

Order	Date	Status	Total	
#6472	June 10, 2014	On-hold	Rs. 1,160.00 for 4 items	View
#6381	June 7, 2014	Completed	Rs. 1.00 for 1 item	View

चूंकि ये सारा Process हमें व आपको Manually Follow करना होता है, इसलिए इस Offline Manual Payment द्वारा Order करने की स्थिति में पुस्तकों का Download Link प्राप्त होने में 5 से 10 मिनट का समय लगता है।

एक बार Download Link Activate हो जाने के बाद आप अपनी खरीदी गई पुस्तकों को अपने **My Account** Page से भी Download कर सकते हैं, जहां Download Link Activate होने के बाद आपको अपना **My Account** Page निम्न चित्रानुसार दिखाई देने लगता है:



My Account

Hello **Kuldeep Mishra** (not Kuldeep Mishra? Sign out). From your account dashboard you can view your recent orders, manage your shipping and billing addresses and edit your password and account details.

Available downloads

C in Hindi – C Programming Language in Hindi	8 downloads remaining
C in Hindi – Windows Programming with C in Hindi	8 downloads remaining

Recent Orders

Order	Date	Status	Total	
#6520	June 11, 2014	Completed	Rs. 350.00 for 1 item	View
#6472	June 10, 2014	Processing	Rs. 1,160.00 for 4 items	View
#6381	June 7, 2014	Completed	Rs. 1.00 for 1 item	View

इसके अलावा किसी Particular Order की Details प्राप्त करने के लिए आप इस **My Account** Page पर दिखाई देने वाले **View** Button को भी Click कर सकते हैं।

Offline Manual Methods to Pay "Total Payable Amount"

अपना **Total Payable Amount** Pay करने के लिए आप अपनी सुविधानुसार निम्न में से किसी भी तरीके को Use कर सकते हैं:

Fund Transfer Using ATM Machine

वर्तमान समय में लगभग सभी Banks अपनी ATM Machine द्वारा **Fund Transfer** करने की सुविधा Provide करते हैं, जहां आप अपने ATM Card द्वारा हमारे किसी भी Bank Account में अपनी पुस्तकों का **Total Payable Amount** Transfer कर सकते हैं। अतः यदि आपके पास निम्न में से किसी भी **State Bank** का **Debit Card** है:

- **SBI** *(State Bank of India)*
- **SBBJ** *(State Bank of Bikaner and Jaipur)*
- **SBH** *(State Bank of Bikaner and Hyderabad)*
- **SBP** *(State Bank of Bikaner and Patiala)*
- **SBM** *(State Bank of Mysore)*
- **SBT** *(State Bank of Travancore)*

अथवा PNB *(Punjab National Bank)* या BOB (*Bank of Baroda*) का **Debit Card** है, तो आप SBI/PNB/BOB के **ATM Machine** से भी अपना Payment हमारे SBI/PNB/BOB Bank A/c में Transfer कर सकते हैं।

यदि आप SBI ATM Machine से हमारे SBI Bank A/c में Payment Transfer करना चाहते हैं, तो आपको निम्न Steps को Follow करना होता है:

- SBI ATM Machine में अपना **Debit Card, Swipe** कीजिए।
- ATM Screen के **Bottom Right** Corner में दिखाई देने वाले **Transfer** नाम के Option को Select कीजिए।
- अपने **Debit Card** का **PIN Number** Enter कीजिए।
- अब **Card to Card Transfer** नाम के Option को Select कीजिए।
- अब हमारे **SBI Debit Card Number (6220180786800030243)** को Enter कीजिए।
- अब हमारे **SBI Debit Card Number (6220180786800030243)** को दोबारा Enter कीजिए।
- अब **Transferable Amount** के रूप में **Total Payable Amount** Specify कीजिए।
- अब अपने **Account Type** (Savings or Checking) को Select कीजिए।
- उपरोक्त सभी Steps सही तरीके से Follow होने की स्थिति में आपका **Transaction Complete** हो चुका है और **Total Payable Amount** लगभग तुरन्त हमारे SBI Bank A/c में जमा हो जाता है।

ठीक इसी तरह के Steps आपको उस समय भी Follow करने होते हैं, जब आप **PNB या BOB** के **ATM Machine** के माध्यम से हमारे **PNB Bank A/c** में **Total Payable Amount**, **Card to Card Transfer** करते हैं।

हालांकि आप **HDFC, IDB, ICICI** जैसे कई अन्य **ATM Machines** द्वारा भी **Card to Card Transfer** कर सकते हैं, लेकिन यदि दोनों **Debit Cards** समान Banks (PNB, SBI or BOB) के न हों, तो Transaction Perform होने में 24 से 96 घण्टे का समय लगता है।

इसलिए इस स्थिति में बेहतर यही होता है कि यदि आपके पास **Net Banking, Mobile Baking, AirTel Money** या **SBI/PNB/BOB Debit Card** किसी भी तरह की सुविधा न हो, तो आप **Total Payable Amount** का Payment करने के लिए Cash Deposit तरीके को ही Use करें अथवा Bank में जाकर **NEFT Transfer** भी कर सकते हैं, जिसमें आपका Payment अधिकतम 4 घण्टे के दरम्यान हमारे Bank A/c में Deposit हो जाता है।

Payment Transfer Using Net-Banking

यदि आपके पास **Net-Banking** की सुविधा है, तो आप Payment Transfer करने के लिए अपने Account में Login करके निम्न में से किसी भी Bank A/c में Payment Deposit कर सकते हैं:

भारतीय स्टेट बैंक
State Bank of India
With you - all the way

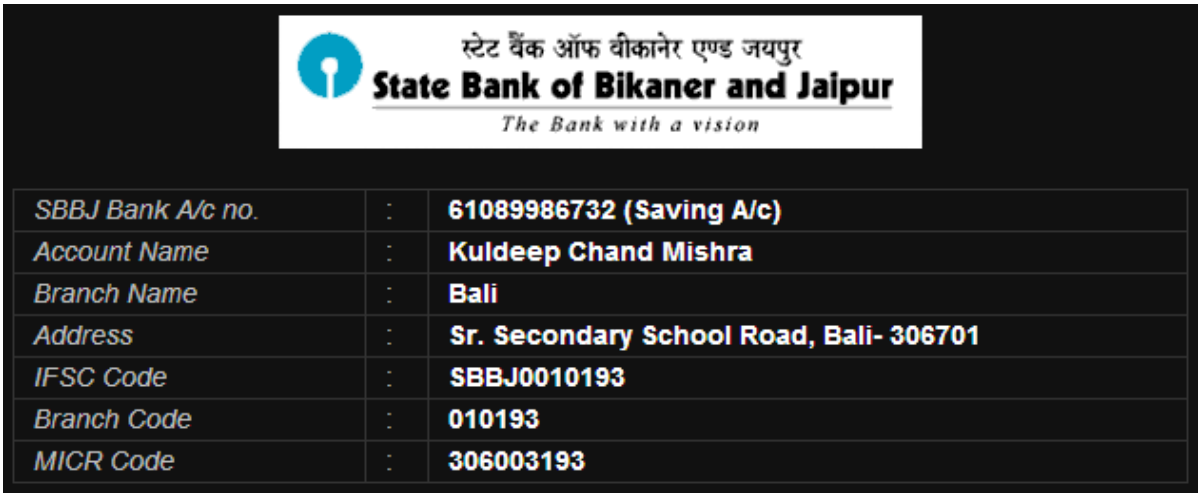
SBI Bank A/c no.	:	**31154882587 (Saving A/c)**
Account Name	:	**Namita Mishra**
Branch Name	:	**Falna**
Address	:	**Near Railway Crossing, Falna Station – 306116**
IFSC Code	:	**SBIN0007868**
Branch Code	:	**007868**
MICR Code	:	**306002100**
ATM Debit Card No.	:	**6220180786800030243 (Maestro)** *(For **Card to Card Transfer** using ATM Machine)*

punjab national bank
...The name you can BANK upon!

PNB Bank A/c no.	:	**4445000100034960 (Saving A/c)**
Account Name	:	**Kuldeep Chand**
Branch Name	:	**Falna**
Address	:	**College Road, Falna Station – 306116**
IFSC Code	:	**PUNB0444500**
Branch Code	:	**444500**
MICR Code	:	**306024200**
ATM Debit Card No.	:	**5126 5200 1286 3655 (MasterCard)** *(For **Card to Card Transfer** using ATM Machine)*

बैंक ऑफ़ बड़ौदा
Bank of Baroda
India's International Bank

BOB Bank A/c no.	:	**35260100003212 (Saving A/c)**
Account Name	:	**Namita Sharma**
Branch Name	:	**Falna**
Address	:	**Sanderao Road, Falna, Dist. Pali (Raj.)- Pin-306116**
IFSC Code	:	**BARB0FALNAX**
Branch Code	:	**FALNAX**
MICR Code	:	**NON-MICR**
ATM Debit Card No.	:	**4029850310081366 (VISACard)** *(For **Card to Card Transfer** using ATM Machine)*



SBBJ Bank A/c no.	:	61089986732 (Saving A/c)
Account Name	:	Kuldeep Chand Mishra
Branch Name	:	Bali
Address	:	Sr. Secondary School Road, Bali- 306701
IFSC Code	:	SBBJ0010193
Branch Code	:	010193
MICR Code	:	306003193

जब आप **Net-Banking** के माध्यम से Payment करना चाहते हैं, तो आपको लगभग **8** से **24** घण्टे पहले हमारे उस Account को **Beneficiary** के रूप में अपने Bank A/c से Link करना पडता है, जिसमें आप Payment **Transfer** करना चाहते हैं। जब एक बार हमारा Bank Account Beneficiary के रूप में Activate हो जाता है, उसके बाद आप उस Bank Account में अपना **Total Payable Amount** Transfer कर सकते हैं।

Pay with Mobile-Banking or AirTel Money

यदि आपने अपने **Mobile Number** पर **AirTel Money** नाम की Service को Activate किया हुआ है, तो आप अपने Mobile द्वारा AirTel Money Account के माध्यम से भी हमें Payment कर सकते हैं। जबकि यदि आपने अपने Bank से Mobile Banking की सुविधा को Activate करवाया हुआ है, तो आप अपने Mobile द्वारा हमें **Mobile-Banking** के माध्यम से भी Payment Transfer कर सकते हैं।

यदि आप अपना **Total Payable Amount** Pay करने के लिए **AirTel Money** या **Mobile Banking** Transfer सुविधा को Use करते हैं, तो आपका Payment तुरन्त हमारे Account में Transfer हो जाता है। इसलिए तुरन्त EBooks प्राप्त करने हेतु **Payment Transfer** करने का ये सबसे तेज तरीका है। जबकि इसके अलावा जो दूसरा सबसे तेज तरीका है, वह ATM Machine द्वारा **Fund Transfer** सुविधा का उपयोग करते हुए Payment Transfer करना है।

जबकि **Internet-Banking** उस स्थिति में काफी धीमा Process है, जब आप पहली बार Payment कर रहे होते हैं, क्योंकि First Time Payment करने से पहले आपको Beneficiary के रूप में हमारे किसी एक Bank Account को अपने Bank Account से Link करना पडता है और इस Process में Payment Transfer हेतु हमारा Bank Account **Activate** होने में कम से कम 8 से 24 घण्टे का समय लगता है। हालांकि एक बार Account Activate हो जाने के बाद आपका Transfer तुरन्त हो जाता है।

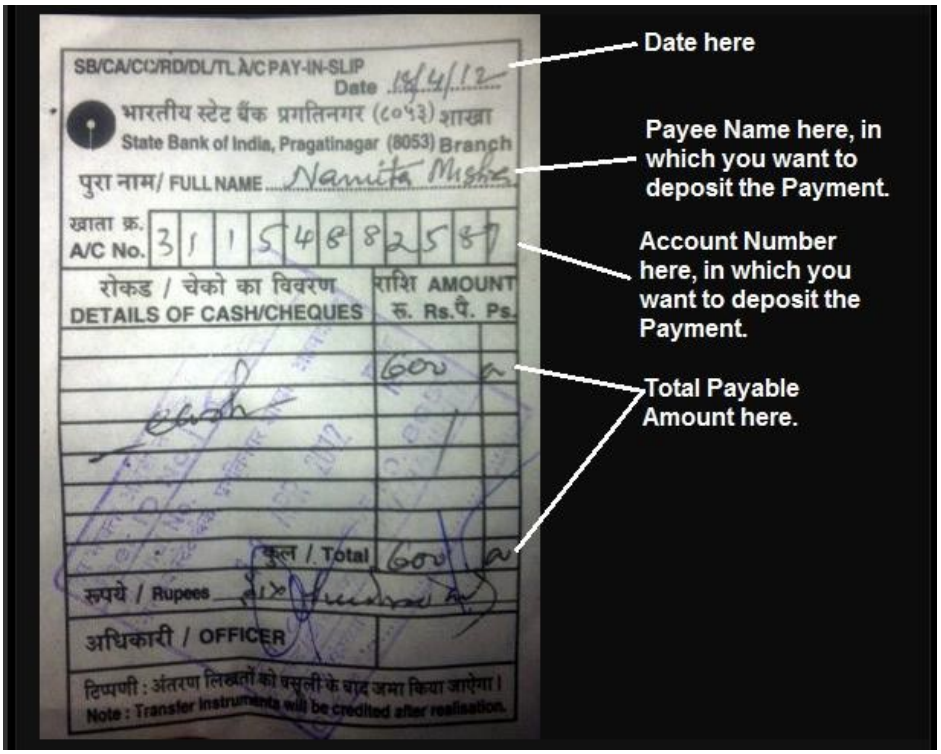
जबकि **Cash Deposit** का तरीका सबसे धीमा तरीका है, जहां आप Bank Holidays को Payment Deposit नहीं कर सकते और Business Days में भी Payment Deposit करने का एक निश्चित समय **10PM to 4AM** होता है।

Cash Deposit in Bank Brach

यदि आपके पास Net-Banking या Mobile-Banking की सुविधा नहीं है, तो आप हमारे किसी भी Bank A/c में *Total Payable Amount*, **Cash Deposit** भी कर सकते हैं अथवा आप Bank Branch में जाकर **NEFT Transfer** के माध्यम से भी Payment कर सकते हैं, जो कि Cash Deposit के समान ही होता है।

जब आप **Direct Deposit** करना चाहते हैं, तब आपको आपके किसी भी नजदीकी Bank Branch में जाकर एक *Payment Deposit Slip* Fill-Up करना होता है, जिसमें आपको हमारे किसी भी Bank A/c की Information को Fill करना होता है, जबकि **Payment Deposit** करवाने के लिए उसी **Bank** में आपका स्वयं का **Account** होना जरूरी नहीं है।

उदाहरण के लिए यदि आप हमारे SBI Bank A/c में अपनी Selected पुस्तकों का *Total Payable Amount* Pay करने के लिए Bank में जाकर Direct Deposit करना चाहते हैं, तो आप जो **Payment Deposit Slip** Fill-Up करेंगे, वह अगले चित्रानुसार करना होता है।



इस चित्र द्वारा आप समझ सकते हैं कि Payment, Direct Deposit करने के लिए आपको हमारे किसी Bank A/c की Information को *Payment Deposit Slip* में Specify करना होता है, इसलिए उस Bank में आपका स्वयं का Bank A/c होना जरूरी नहीं होता।

इसी तरह से यदि आप चाहें, तो हमारे किसी भी Bank A/c में Check द्वारा भी **Total Payable Amount** का **Check Deposit** कर सकते हैं।

यानी आप किसी भी तरीके से हमारे किसी भी Bank A/c में *Total Payable Amount* Deposit कर सकते हैं। लेकिन हम **Money-Order, Demand-Draft** या **Check** जैसे Manual माध्यमों से Payment Accept नहीं करते, क्योंकि इस तरह का Payment Clear होने में बहुत समय लगता है।

जबकि **ATM Fund Transfer, Cash Deposit, Mobile Banking** अथवा **Net-Banking** के माध्यम से तुरन्त Payment Transfer हो जाता है, जिससे हम आपको आपकी Purchased EBooks **10** से **30 Minute** के दरम्यान आपके Order में Specified **Email Address** पर Send कर देते हैं।

अपना Payment करने के लिए आप जिन **Offline Manual** तरीकों को उपयोग में ले सकते हैं, उनकी **Detailed Information** आप http://www.bccfalna.com/how-to-deposit-payment/ से भी प्राप्त कर सकते हैं, जहां आपको Payment करने से सम्बंधित किसी भी तरह का Latest Update प्राप्त होता है।

Pay with PayPal if you live Out Of India

यदि आप **India** में नहीं रहते लेकिन ये **Hindi EBooks** खरीदना चाहते हैं, तो आप अपनी वांछित पुस्तकों के **Total Payable Amount** का भुगतान हमें **PayPal** के माध्यम से *bccfalna@gmail.com* पर भी Send कर सकते हैं।

चूंकि **International Payment Processing** में विभिन्न प्रकार के **Extra Charges** Pay करने होते हैं, इसलिए PayPal के माध्यम से Payment करते समय आपको **Total Payable Amount (In USD) + $2** का Extra Payment Send करना जरूरी होता है।

अपना Payment Send करने के बाद आप अपने Payment की Information हमें SMS या Email के माध्यम से दे सकते हैं। जैसे ही आपका **Email/SMS** हमें प्राप्त होगा, आप द्वारा Order की गई पुस्तकों का Download Link आपके **Email Address** पर जितना जल्दी सम्भव होगा, उतना जल्दी Send कर दिया जाएगा।

Confirm the Payment

जब आप अपनी Order की गई पुस्तकों को खरीदने के लिए उपरोक्तानुसार किसी भी **Offline Manual** तरीके से *"Total Payable Amount"* हमारे किसी भी Bank A/c में Deposit/Transfer कर देते हैं, तो Payment Deposit/Transfer करते ही आपको हमें उसी **Mobile Number** से एक **Call/Miss Call/SMS** करना होता है, जिसे आपने Order Place करते समय "**Order Form**" में Specify किया था।

इसी Mobile Number के माध्यम से हमें पता चलता है कि आपने किन पुस्तकों के लिए कौनसा Order किया है और उनका *Total Payable Amount* कितना है। साथ ही हमें ये भी पता चल जाता है कि आप द्वारा Purchase की जा रही पुस्तकें किस **Email Address** पर Send करनी है।

आपके *Total Payable Amount* को हम Net-Banking के माध्यम से अपने Bank A/c में Check करते हैं और यदि आपका *Total Payable Amount* हमारे किसी भी Bank A/c में Deposit/Transfer हुआ होता है, तो हम आपको **10 Minute** के दरम्यान आपकी Order की गई EBooks आपके Email Address पर Send कर देते हैं, जिसे आप अगले 2 दिनों में कभी भी Download कर सकते हैं।

If you have any problem

यदि पुस्तकें खरीदने से सम्बंधित किसी भी प्रकार की कोई बात आपको ठीक से समझ में न आई हो या किसी भी तरह का Confusion हो, तो आप **097994-55505** पर **Call/Miss Call/SMS** कर सकते हैं। यथा सम्भव तुरन्त आपकी समस्या का समाधान किया जाएगा।

चूंकि ये सारी पुस्तकें **PDF Format Softcopy Ebooks** हैं इसलिए इन पुस्तकों का **Download Link** आपको आपके Email पर ही **Send** किया जाता है, जिन्हें Click करते ही ये पुस्तकें आपके Computer पर Download होना शुरू हो जाती हैं।

एक बार इन पुस्तकों को Download करने के बाद आप इन्हें किसी भी PDF Supported *Computer, Mobile, Smart Phone, Tablet PC, Net-Book, Notebook* या *Laptop* जैसी Device के माध्यम से पढ सकते हैं अथवा यदि आप चाहें, तो अपने Printer द्वारा इन पुस्तकों का Hard Copy Printout निकाल सकते हैं।

चूंकि इन पुस्तकों के Download Links आपको आपके Email Address पर ही प्राप्त होते हैं, इसलिए जरूरी है कि उपरोक्त "**Order Form**" पर आप अपना जो **Email Address** व **Mobile Number** Specify करते हैं, वह Working और एकदम सही हो। क्योंकि किसी भी तरह की परेशानी होने की स्थिति में हम आपको आपके **Mobile Number** अथवा **Email Address** द्वारा ही Contact करते हैं।