Best Practices for Localization of Mobile web applications in Indian Languages

Version 2.6

August 2014

Government of India

Ministry of Communication and Information Technology

Department of Electronics and Information Technology

Metadata of the Document

S. No.	Data elements	Values
1.	**Title**	Best Practices For Localization of Mobile web applications in Indian Languages.
2.	**Title Alternative**	L10N-MobiWeb
3.	**Document Identifier** *(To be allocated at the time of release of final document)*	
4.	**Document Version, month, year of release** *(To be allocated at the time of release of final document)*	2.6, August, 2014
5.	**Present Status**	
6.	**Publisher**	Ministry of Communication and Information Technology, Department of Electronics and Information Technology
7.	**Date of Publishing**	
8.	**Type of Standard Document** *(Policy / Technical Specification/ Best Practice /Guideline/ Process)*	*Best Practice*
9.	**Enforcement Category** *(Mandatory/ Recommended)*	*Recommended*
10.	**Creator** *(An entity primarily responsible for making the resource)*	Ministry of Communication and Information Technology, Department of Electronics and Information Technology
11.	**Contributor** *(An entity responsible for making contributions to the resource)*	Centre for Development of Advanced Computing (CDAC), GIST, Pune, India

12.	**Brief Description**	This document discussed various guidelines to help developers to localize their software products / services on Mobiles. This document will benefit new software developers in using the existing technologies together with various initiatives and standards, which will help them, understand the complications involved with various scripts and platforms. A further benefit is better integration and interoperability of products. The document brings to limelight some national as well as international standards currently used in the Localization industry. Various important topics have been touched in this document like Inputting, storage and rendering Indian language data, Unicode migration from legacy data, Usage of Common Locale Data Repository (CLDR), Characters encoding for proper representation on various platforms, Unicode, directionality issues with right to left scripts such as Urdu and the problem of using Cascading Style Sheets (CSS) in context Indian languages. Frequently Used Entries for Localization (FUEL) for term consistency as an open source initiative, in context of Indian language has been cited. Various ISO 639 Language Codes have been provided at the end of the document. However, some of the guidelines are generic in nature and may or may not be applicable to mobile platforms.
13.	**Target Audience** *(Who would be referring / using the document)*	This document is intended for: Software Designers / Engineers, Testing and QA Engineers , Mobile App Developers and VAS Providers and Policy makers
14.	**Owner of approved standard**	Ministry of Communication and Information Technology, Department of Electronics and Information Technology
15.	**Subject**	Localisation

	(Major Area of Standardization)	
16.	**Subject. Category** *(Sub Area within major area)*	Mobile Web
17.	**Coverage. Spatial**	India
18.	**Format**	PDF
19.	**Language** *(To be translated in other Indian languages later)*	English *(To be translated in other Indian languages later)*
20.	**Copyrights**	Ministry of Communication and Information Technology, Department of Electronics and Information Technology
21.	**Source** ***(****Reference to the resource from which present resource is derived)*	Adapted from and partly based on the references provided in section 19: References
22.	**Relation** *(Relation with other e-Governance standards notified by DeitY)*	

Amendments Log

Old Version No, Month and Year	Briefing about change request and action taken	New Version No, Month and Year	The sections , which have been revised or New section added
1.1 October 2012	Draft Version	1.2 November 2012	Draft Prepared
1.2 November 2012	Based on feedback	1.3 November 2012	Section on Complexities in Indian languages Added
1.3 November 2012	Based on feedback	1.4 November 2012	3GPP Standard Added
1.4 November 2012	Based on feedback	1.5 December 2012	Internet Browsing section added
1.5 December 2012	Based on feedback	1.6 December 2012	CSS section modified
1.6 December 2012	Based on feedback	1.7 December 2012	Added Acronyms & Abbreviations
1.7 December 2012	Based on feedback	1.8 December 2012	Added Annexure I
1.8 December 2012	Based on feedback	1.9 January 2012	Modified Device Specific guidelines section
1.9 January 2014	Based on feedback	2.0 January 2014	Modified Application Specification guidelines section
2.0 January 2014	Based on feedback	2.1 January 2014	Modified Application Specification guidelines section
2.1 January 2014	Based on feedback	2.2 January 2014	Modified Application Specification guidelines section
2.2 January 2014	Based on feedback	2.3 February 2014	Restructuring done based on feedback
2.3 February 2014	Based on feedback	2.4 February 2014	Restructuring done based on feedback
2.4 February 2014	Based on feedback	2.5 March 2014	Template Added
2.5 March 2014	Based on Feedback by E-Gov Localization Committee	2.6 July 2014	Restructured based on the feedback

Important information

Many legacy and existing e-Governance applications are based on proprietary tools, technologies and database formats. Few of these also form part of some of the Mission Mode Projects envisaged under the NeGP. Therefore, to cater to the requirements of such applications the scope of this document has been kept broad, also to include specific sections on proprietary software. Under National e-Governance Plan (NeGP) for all software applications being currently planned and developed for delivering e-Governance services and solutions, the Government of India however strongly recommends adherence to Open Standards while taking decisions on the use of tools, technologies and database formats .All stakeholders are thus exhorted to contribute to the cause of open standards to help ensure seamless integration of services and sharing of data across different e-Governance applications and services."

Contents

1 Introduction

1.1 Outline

The means to access the internet are changing from a desktop to mobile by the rapid change in the technology. This situation not only opens interesting new avenues to content publishers to reach as many users as possible on mobile devices, but also brings with it many challenges to deal with when we consider language support, especially the complex Indian scripts on these devices. From the beginning of the development, the Standard bodies have seen the need of mobile web standards in order to optimize the web user experience on small and constricted mobile devices. However, they have not been able to keep pace with the drastic changes on the hardware front, having many standards that do not reflect and exploit the new hardware situation. This lack of governing mobile standard results in a paradigm shift on mobile devices; a very interactive and rich Web 2.0 website on the desktop web, where a browser-based experience is common, very often does not provide the same functionality through mobile web.

The intent of this document is to discuss localization in Indian language perspective for mobile devices and the concepts involved in localization like: inputting, storage, display, communication protocols, CLDR (Common Locale Data Repository), Collation Order, Hyphenation rules, Cursor movement rules, addition deletion rules, bidi (Bi-directional) algorithm and many more. It also provides a set of guidelines which is broadly classified under the heading “Application Specific Guidelines”.

The document also summarizes the basic components of Indian Language (IL) solution and the role of the embedded device manufacturer to integrate IL solution into the system. It has been observed that for enabling IL on various platforms across the world, the required core functionality to enable IL is highly proprietary in nature, very complex, tightly coupled with the performance of device and is not generic at all.

1.2 Purpose of this Document

This document is intended for application developers, quality assurance, and also handset/device manufacturers and all the stakeholders for seamless interoperability. This document can also be used as a guideline by various Government of India departments like e-Governance etc. for ensuring that their various schemes/initiatives are seamlessly provided on mobile/tablet platforms in Indian languages.

1.3 Scope

The scope of this document is to lay down guidelines or best practices to be followed for mobile application development esp. in view of Indian Languages. The scope of this document is to create awareness among the software developers and the designers to create localized applications/products. There are some coding snippets / examples which are used to illustrate the concept; however they are not intended as a guide to implementation. Some of the guidelines are generic in nature and may be followed as per the requirement.

The document focuses only on HTML 5 based forms for mobile applications. The other standard forms are not under the scope of this document.

1.4 Background

India is a multilingual country with 22 scheduled languages and 12 different scripts. The scheduled languages of India are Assamese, Bangla, Bodo (Boro), Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Odia (Oriya), Panjabi (Punjabi), Sanskrit, Santali, Sindhi, Tamil, Telugu and Urdu. The scripts which are widely used are:

Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Meitei Mayek, Ol-Chiki, Oriya, Perso-Arabic, Tamil, and Telugu. To support all these languages on a mobile device is a major challenge.

The complexity of Indian scripts in terms of inputting, storage and display has been the major challenge while giving support of Indian languages on mobiles. Traditionally when mobiles started arriving in the market, the main focus was on roman like scripts and more complex scripts were always a afterthought. The entire ecosystem of the mobile handset including hardware, operating system, applications, communication protocols such as SMS etc, were designed keeping the focus on roman like scripts. The limited 12 keys on the physical keypad have always posed a daunting challenge to input Indian languages. In attempt to overcome this challenge and come with quick solution, many vendors/handset manufacturers have implemented their own version of keyboard layout. User is finding it difficult to get accustom to so many proprietary keyboards. It is even more difficult for the users to shift their inputting style if they change the mobile handset of different vendor. Hence, there is an urgent need to standardize the keyboards of all vendors/manufacturers.

The storage and display of Indian languages are greatly linked with inputting. In fact, all three components are inter-dependant and cannot be considered in isolation. There have been many proprietary storage formats used by the vendors in the past; however Unicode has become

accepted standard. The vendors are rather slow in adapting to revisions of Unicode for Indian scripts. This creates the compatibility issue when data is migrated to-and-from the mobile to desktop/internet and vice versa. Display of Indian languages is mostly governed by the applications. The applications like browsers have their own complex script renderers where as other applications depend on system wide implementation of Indian scripts. This has created a rather difficult situation to the users, wherein on the same mobile handset, user may have to use two-or-more different implementations of Indian languages; with one being system implementation provided by the mobile vendor/manufacturer and other is application specific implementation like browser etc. This is confusing to the users and sometimes may lead to users turning away from Indian language usage on mobiles.

1.5 Sophistication of Indian Scripts

Alphabets for Indian languages are highly developed and have a highly evolved system of organization, coding of sounds, and the system of writing.

The alphabet is organized scientifically based on position of tongue and aspiration of breath as you speak a sound. It is phonetic- where you speak what you write directly correlated to individual aksharas. As a result, deviations between speaking and writing, if any, which have crept in different Indian languages, are more predictable and systematic than say with Roman alphabet and English.

The script (the actual writing of the characters in a sequence) evolved from the Brahmi script and is highly sophisticated. It is not just a placement of the characters one after another like in Roman script. The graphemes for vowels following the graphemes for consonants are systematically merged into an akshara (consisting of graphemes for maatraas and consonant clusters).

This sophistication in writing became a problem, unfortunately, because the initial rudimentary computer technology was unable to handle it.

The display technology evolved with True Type Font (TTF) technology to incorporate variable width fonts richer than the typewriter. Though ISCII (Indian Standard Code for Information Interchange) the BIS standard was already in place in the year 1988 itself, it was never made mandatory for all the Operating Systems / Applications to follow it. TTF technology although good in display of shapes, was incapable of rendering the sophisticated Indian scripts. As a result, developers using this insufficient technology were forced to adopt work-around which led to storage of data in their respective proprietary font code (TTF). As a result, for a dark period lasting about 27 years, OS / applications including graphic display software technology was unable to handle Indian scripts properly for want of software implementation, even when it was

technologically possible as demonstrated through GIST technology. Unfortunately, the work-around solution did violence to our scripts rendering text keyed-in by this method non-displayable across schemes, as well as it led to the proliferation of non-standard representation in memory and keyboard layouts. Indian languages are still recovering from this damage, due to proliferation of non-standards during this dark period.

With the Open Type Font technology which was developed specifically to handle Indian and other sophisticated scripts, the earlier technology lacuna was finally overcome. However, the proliferation of inputting using different keyboards still remains a problem as different people got used to different types of keyboards.

In the Indian scripts, a sequence of consonants is not just placed left to right but combines to form a consonant cluster. It is more compact and takes less space to display. It requires a longer learning time, but is faster to read. Vowels following a consonant cluster are put as matra (or diacritic) on the cluster which again shows the sophistication of the scripts. Such a unit is called an akshara and is like a syllable. The aksharas are placed left to right one after another. Thus, the script's basic unit or atom is the character (vowel or consonant) like in Roman, and is used for inputting as well as pronunciation. However, these atoms are put together into molecules or aksharas (or syllables) by a complex process. Once an akshara is formed, it is displayed after the previous akshara from left to right in a simple manner.

The formation of consonant clusters has a visually appealing and an intuitive process, but capturing such language processes in rules has always been a challenging task. This has been achieved in OTF technology by making provision for rules as well as pre-stored forms in tables.

The inputting proceeds by typing of characters. Each character has a well defined sound which it stands for. The characters are typed in the order in which the sounds occur. The display proceeds by composing each akshara by complex rules, and the akshara themselves are placed from left to right one after another.

While characters are being input, an akshara is being formed dynamically for displaying. The correspondence between the characters input and the form being displayed is not always one to one. Therefore, one needs to learn not just the individual characters but also how the aksharas look. Backspace deletes the entire last akshara and not just the last character.

The guidelines here are to make it easier and aesthetically more pleasing to process and work with Indian scripts.

1.6 Indian languages on mobiles:

Inputting of Indian language text on mobiles also requires a special keyboard driver to interpret Indian language text entry. Such keyboard driver can allow direct Indian language inputting, like Enhanced INSCRIPT (http://pune.cdac.in/html/gist/down/inscript_d.asp), phonetic like inputting (where IL is entered by using English keypad e.g. ghar ja raha tha), Multitap, CDAC's two key mechanism.

Some existing standards like Enhanced INSCRIPT should be referred for mobile based inputting. Also IL alphabets are required to be should be engraved on the keys of the mobile device. For mobile devices having touch screen interfaces only and where the screen size is small, 12-Key Virtual Keyboard inputting mechanism should be preferred. For smart phones where the screen size is more than 3 inches (measured diagonally) INSCRIPT based Virtual Keyboard should be preferred.

This document discusses guidelines for mobile devices along with the basic needs required to localize a piece of text i.e. input, storage, display and communication.

2 Target Audience

This document is intended for:

Software Designers / Engineers: To understand and evaluate Localization areas related to product design.

Testing and QA Engineers: To define test plans and test cases keeping Localization and Internationalization in mind.

Mobile App Developers and VAS Providers: To ensure seam less Indian languages support.

Policy Makers: To ensure policy level requirements for Indian Language support for Mobile Ecosystem

3 Type of Standard Document & Enforcement Standard form

This document, as the name suggests, provides norms and recommendations termed as Best Practices for L10N. L10N for Mobile web requires a common minimum requirement for all stakeholders and this is the major aim of this document. Govt. Bodies or third party developers would use this document to create L10N in all the constitutionally recognized languages to ensure that that they get the widest outreach possible. This document is more in the form of recommendations and provides norms for the purpose of L10N in mobile Applications.

4 Definition and Acronyms

For definitions and acronyms please refer Annexure 3.

5 Guidelines

The guidelines mainly focuses on mobile application development under the heading “Application Specific Guidelines”. However to complete the device requirements, the Related aspects of device level issues are covered in the Annexure – I

6 Application Specific Guidelines

Application specific guidelines could be divided into two categories viz.,

i) W3C Mobile web best practices (MWBP) & other related W3C guidelines. The details are given in the next section

ii) Inherent Indian languages support.

6.1 Mobile Web Best Practices

The Mobile Web Best Practices, a set of recommendations released by the W3C in 2008, are not just about technical issues like the use of standards for the mark-up or style sheets, but also to improve the user experience in accessing the web from mobile devices. The goals of the Best Practices are a pleasant and functional presentation of the content, simplification in the input procedure, efficiency in bandwidth and costs, achieve user goals, ensure an effective space for advertising and work towards a "One Web" experience, where the same information is available across all devices. W3C has developed a number of Web technologies that explicitly take into account the specificities of mobile devices:

Visit http://www.w3.org/TR/2008/REC-mobile-bp-20080729/ for more details.

The generalized Web Architecture for representation various mobile and wireless devices is as depicted below:

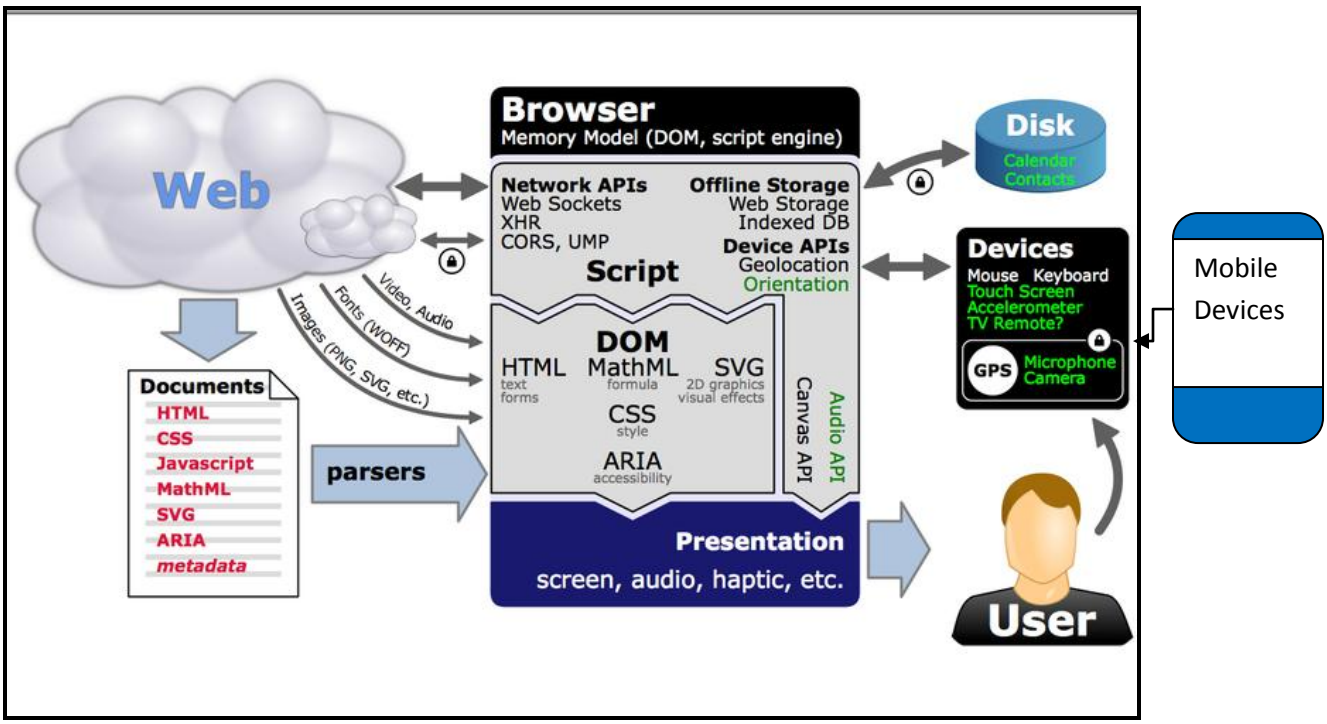


Fig1. Generalized Mobile Web Architecture

Standards that are specifically adhered for seamless mobile applications development can be divided in the following categories

1. Graphics
2. Multimedia
3. Device Adaptation
4. Forms
5. User interactions
6. Data storage
7. Sensors and hardware integration
8. Network

The W3C specifications / recommendations that may be adopted for mobile web development corresponding to each of the above categories are listed in the table:

Sl No.	Category	W3C Specification	References
1.	Graphics	(1) Scalable Vector Graphics (SVG) 1.1 **(2)** CSS Backgrounds and Borders	(1) http://www.w3.org/TR/SVG11/ (2) http://www.w3.org/TR/css3-background/
2.	Multimedia (Video and Audio Playback)	(1) Video playback (2) Audio Playback	(1) http://www.w3.org/TR/html5/embedded-content-0.html#the-video-element (2) http://www.w3.org/TR/html5/embedded-content-0.html#the-audio-element
3.	Device Adaptation	(1) Device Description Repository Simple API	(1) http://www.w3.org/TR/DDR-Simple-API/
4.	Forms	(1) Forms Control	(1) http://www.w3.org/TR/html5/forms.html
5.	User Interactions	(1) Touch Event Control (2) Input Method Editor API	(1) http://www.w3.org/TR/touch-events-extensions/ (2) http://www.w3.org/TR/ime-api/
6.	Storage	(1) Web Storage	(1) http://www.w3.org/TR/webstorage/
7.	Sensors and Hardware Integration	(1) Geolocation	http://www.w3.org/TR/geolocation-API/
8.	Network Information	**(1)** WebRTC 1.0: Real-time Communication Between Browsers	http://dev.w3.org/2011/webrtc/editor/webrtc.html

6.2 Default Delivery Context

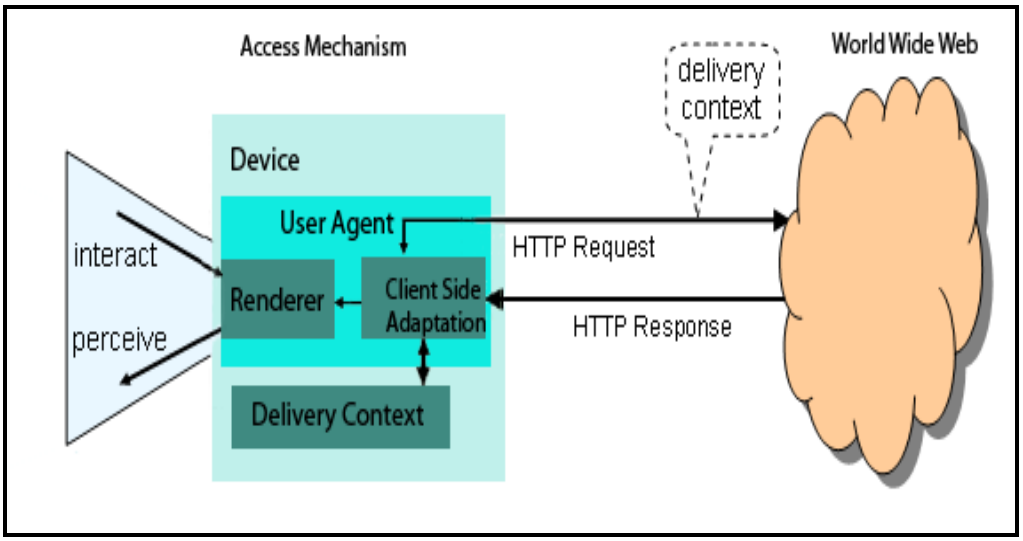


Fig.2 Delivery Context for Mobile Applications

In order to allow content providers to share a consistent view of a default mobile experience the BPWG has defined the Default Delivery Context. This allows providers to create appropriate experiences in the absence of adaptation and provides a baseline experience where adaptation is used. The Default Delivery Context is the minimum delivery context specification necessary for a reasonable experience of the Web on Mobile platform. This specification is made against the background of demographic, cultural and economic assumptions. The Default Delivery Context specification as follows:

- Usable Screen Width - 120 pixels, minimum.
- Markup Language Support - XHTML Basic 1.1 [XHTML-Basic] delivered with content type application/xhtml+xml.
- Character Encoding - UTF-8
- Maximum Total Page Weight – 40 KB (International recommendation is 20 KB however due to multiple byte requirement of Indian languages. It has been investigated that these will require 40 KB)
- Image Format Support – JPEG 2000 (Although GIF 89 A is allowed as per MWPB but to avoid network load only JPEG is recommended.

- Colors - 256 Colors, minimum.
- Style Sheet Support - CSS 2.1
- HTTP 1.0/ 1.1
- Script - no client side scripting is recommended due to device limitations.

6.3 Implementation notes regarding localization of web based forms

A form is a component of a Web page that has form controls, such as text fields, buttons, checkboxes, range controls, or color pickers. A user can interact with such a form, providing data that can then be sent to the server for further processing (e.g. returning the results of a search or calculation). No client-side scripting is needed in many cases, though an API is available so that scripts can augment the user experience or use forms for purposes other than submitting data to a server.

Browsers are encouraged to use user interfaces that present dates, times, and numbers according to the conventions of either the locale implied by the input element's language or the user's preferred locale. Using the page's locale will ensure consistency with page-provided data.

For example, it would be confusing to users if an American English page claimed that a Cirque De Soleil show was going to be showing on 02/03, but their browser, configured to use the British English locale, only showed the date 03/02 in the ticket purchase date picker. Using the page's locale would at least ensure that the date was presented in the same format everywhere. (There's still a risk that the user would end up arriving a month late, of course, but there's only so much that can be done about such cultural differences to review for deletion.

Refer: HTML5-A vocabulary and associated APIs for HTML and XHTML - http://www.w3.org/TR/html5/forms.html#input-impl-notes

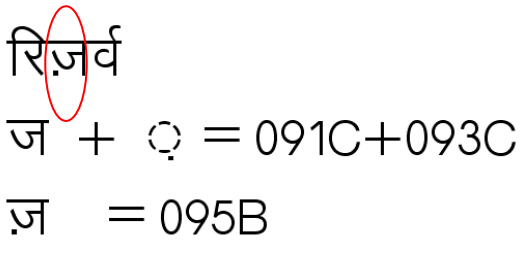
6.4 Normalization in Unicode

The Unicode data requires normalization. There are cases in Unicode which allow dual representation of some of the conceptual characters. For the purpose of text processing, it is highly desirable that such dual representations be brought in a single unified form. This process is known as Normalization.

This equivalence of characters is in two forms known as canonical and compatibility equivalence. In canonical equivalence, the two representations look exactly similar and are just different values

of the same thing. Compatibility equivalence on the other hand depends on the context in which the characters are present and may or Indian Script Codes for Information Interchange & Enhanced INSCRIPT Keyboard Layouts. Visit http://unicode.org/reports/tr15/ for more details

e.g.



The text normalization issues may be addressed in the implementations.

6.5 User Input

This section contains statements relating to user input. This is typically more restrictive on mobile devices than on desktop computers (and often a lot more restrictive). For example, mobile devices may lack pointing devices and often do not have a standard keyboard for text entry. Given the typical input limitations of a mobile device, the interface must as far as possible minimize user input. Where possible, use selection lists, radio buttons and other controls that do not require typing. Some mark-up languages allow the specification of an input mode, which is particularly useful in cases where user input is to be restricted, for example to numeric only. It is anticipated that XHTML-Basic [XHTML-Basic] will support this functionality in the future.

Input

- [MINIMIZE_KEYSTROKES] Keep the number of keystrokes to a minimum.
- [AVOID_FREE_TEXT] Avoid free text entry where possible.
- [PROVIDE_DEFAULTS] Provide pre-selected default values where possible.
- [DEFAULT_INPUT_MODE] Specify a default text entry mode, language and/or input format, if the device is known to support it.

6.6 Mobile SVG

The mission of SVG 1.0 specifically addresses small devices as a target area for vector graphics display. Each mobile device has different characteristics in terms of CPU speed, memory size, and colour support. To address the range of different device families, two profiles are defined. The

first low-level profile, SVG Tiny (SVGT) is suitable for highly restricted mobile devices; while the second profile, SVG Basic (SVGB) is targeted for higher level mobile devices.

Visit http://www.w3.org/TR/2003/REC-SVGMobile-20030114/ for more details.

6.7 Mobile CSS

This specification defines in general a subset of CSS 2.1 [CSS21] that is to be considered a baseline for interoperability between implementations of CSS on constrained devices (e.g. mobile phones). Its intent is not to produce a profile of CSS incompatible with the complete specification, but rather to ensure that implementations that due to platform limitations cannot support the entire specification implement a common subset that is interoperable not only amongst constrained implementations but also with complete ones. Visit http://www.w3.org/TR/2008/CR-css-mobile-20081210/ for more details.

In the WAP 2.0 specification there is also a part dedicated to the Wireless Cascading Style Sheets (WCSS). It is a subset of the CSS 2 and aims to bring base functionality of CSS to mobile devices. Functionality like inheritance, cascading, selectors are provided and properties for the layout model, fonts, text, visual effects, colours and lists, too. Additionally, OMA defined some extensions to CSS named "marquee" to create simple animation effects, "input" to specify the format of user input in a form and "accesskey" to specify an optional way to select elements in a document.

Also the W3C, following the intent to align the work of both W3C and OMA, has specified a CSS profile, trying to be as similar as possible to WCSS. The CSS Mobile Profile (CSS-MP) is a subset completely compatible with a full or constrained implementation of the CSS 2.1 specification by browsers. It also implements some new features from CSS 3, like the marquee effects, and an optional feature set, designed for most advanced browsers. Because of this optional set, once implemented, CSS-MP goes much further than Wireless CSS and it does not differ so much from CSS 2.1. A very interesting standard for graphics on mobile devices is the Scalable Vector Graphics (SVG). It is an XML document to describe vector graphics, static or animated. The most interesting issue for mobile devices is the scalability of vector graphics, Francesco Luminati 41 making it possible to have the same graphic on large or small screens without a loss of quality. W3C defined two versions of SVG for mobile use, the SVG Tiny, without scripting and styling support, and SVG Basic, with scripting and styling support, to use with more advanced devices. For multimedia presentation there is a standard specified by the W3C called Synchronized Multimedia Integration Language (SMIL). SMIL is an XML based language that allows to describe properties like timing, animations, visual transition, media embedding and so on. It is possible to integrate it in other

XML languages like SVG for animation purpose. The most important implementation of SMIL is in the Multimedia Messaging Services, better known as MMS.

6.8 Listing in CSS

Number schemes/ bulleting needs to be supported in Indian languages as well. Some standards however need to be provided to those developing CSS so that by default user could have the facility to use bulleting in his own Indic languages. The word processors are sometimes used by the user to develop pages for the web. Therefore standards are must. In most application Devanagari order is followed for languages sharing the script, which unfortunately is not the correct thing while deciding on sorting/collation for Indic languages. Number schemes to be supported in Indian languages also.

The bulleting of Indian Languages is proposed with Hindi as an example. This recommendation from India has already accepted by W3C.

The proposed bulleting for Hindi language is shown in the table given below.

Character	Code Points
क	**0915**
ख	**0916**
.....	**0938**
ह	**0939**
कक	**0915+0915**
कख	**0915+0916**
कग	**0915+0917**
.....	**.....**
कह	**0915+0939**
खक	**0916+0915**
खख	**0916+0916**
.....	**.....**
खह	**0916+0939**
.....	**.....**
हह	**0939+0939**
ककक	**0915+0915+0915**

ककख	**0915+0915+0916**
……	**……**
ककह	**0915+0915+0939**
खखक	**0916+0916+0915**
खखख	**0916+0916+0916**
……	**……**
खखह	**0916+0916+0939**
……	**……**
हहह	**0939+0939+0939**

6.9 CSS rendering Issues in Indic Script

There are many rendering issues associated with the application of CSS on an Indic script. The proposed solution is to use Indic syllable.

ABNF definition in the context of the various problems listed in this section

ABNF definition of Indic Syllable

V[m] |{CH}C[v][m]|CH

The Linguistic definition of Indic syllable has been mapped to ABNF(Augmented Backus–Naur Form) for the purpose of text segmentation, Line breaking , Drop letter, letter spacing in horizontal text and vertical text representation. The definition has been elaborated taking Hindi as an example.

The definition is combination of 3 rules :

Rule 1 : V[m]

Rule 2 : {CH}C[v][m]

Rule 3 : CH (This rule is applicable only at the end of the word)

V (Upper case) is complete vowel

M is modifier (Anusvara/Visarga/Chandrabindu)

C is Consonant as per Unicode definition which may or may not include nukta

V (lower case) is any dependent vowel or vowel sign (mātrā)

H is halant / virama

| is a rule seperator

[ ] - The enclosed item is optional under this bracket

{} - The enclosed item/items occurs once or repeated multiple times

Examples:

Rule 1 : V[m]

Sl. No.	Examples	Definition
1.	अ, ई, उ	V (Vowel) is a syllable
2.	अं, उँ, आः	V+ Modifier is a syllable

Rule 2 : {CH}C[v][m]

Sl. No.	Examples	Definition
1.	र, क, ज, ल, म	Consonant is a syllable
2.	प्प,क्ख,च्त, ज्ज्व, त्क्ल, त्स्न	Zero or more Consonant + Virama sequences followed by consonant is a syllable
3.	र्त, र्त्स, र्त्स्न, र्त्स्न्य, फ़्क़	Zero or more Consonant (Nukta) +Virama followed by consonant is a syllable
4.	र्ता, र्त्स्न्र्या, फ़्जी, क्या	Zero or more consonant+ (Nukta)+ virāma sequences followed by a consonant (+Nukta) followed by a vowel sign is a syllable
5.	तः,स्तं, स्त्रँ, स्तः, फ़्ज़ँ	zero or more consonant+ (Nukta)+ virāma sequences followed by a consonant (+Nukta) followed by modifier is a syllable

6.	त्स्न्यां: त्स्न्युं, त्स्न्युँ, फ़्ज़ें,हिं	zero or more consonant+ (Nukta)+ virāma sequences followed by a consonant (+Nukta) followed by a vowel sign and modifier is a syllable
7.	स्थि,ज्जि,ख्वा	Zero or more Consonant +halant sequences followed by a consonant followed by vowel sign is a syllable

Rule 3 : CH

त् , व् , म् , भ् etc usually only these syllable occur at the end of the word (in a alphabetical order)

Examples of combination of the rules:

1. स्वागतम् - CHCv + C + C + CH has following syllables:

स्वा	**CHCv**
ग	**C**
त	**C**
म्	**CH**

2. भरतनाट्यम- C + C + C + Cv + CHC + C

भ	**C**
र	**C**
त	**C**

ना	Cv
ट्य	CHC
म	C

3. सदबुद्धि - C + CHCv + CHCv

स	C
दबु	CHCv
द्धि	CHCv

(i) Styling of first letter pseudo-element Issues in Indian script – Hindi

The first-letter pseudo-element represents the first letter of The first line of a block, if it is not preceded by any other content (such as images or inline tables) on its line. It allows that first letter to be styled individually, without mark-up. It may be used for "initial caps" and "drop caps“

For more details: www.w3.org/TR/CCS2/Selector.html

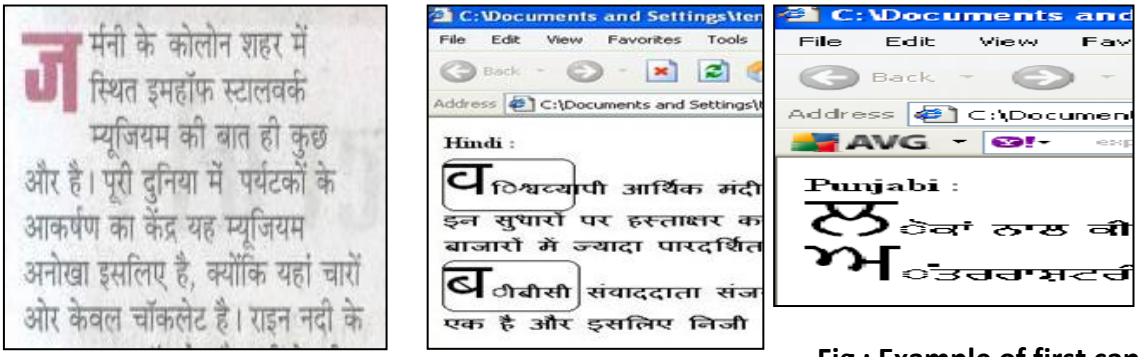


Fig : Example of first cap in Indian languages

This example is showing a possible rendering of an initial cap. In this example the computed height or width of the first letter is different, that is why it is not showing first character properly. (IE, Opera).

(ii) Line Breaking

It is recommended that line should break preferably at word boundary .If unavoidable the guiding principles are as below:

Rule 1: New line cannot begin with following symbols/Punctuation marks. Also these should be retain with the associated text

- Closing brackets
- Devanagari Danda /Purnaviram
- Commas
- Visarga
- Decimal symbols
- Semicolon
- Repetition of punctuation marks such as semicolon with closing brackets, Semicolon with single/ Double quotes , Closing brackets with commas/Semicolon etc
- Mathematical operators

Rule 2: The definition of Indic syllable may be used to break the line and a hyphen should be at the breaking point so that word can be read intuitively

Rule 3: The hyphenated words can be broken after the hyphen e.g.:

नर-नारी should be treated as:

नर- on the first line and नारी on the next line

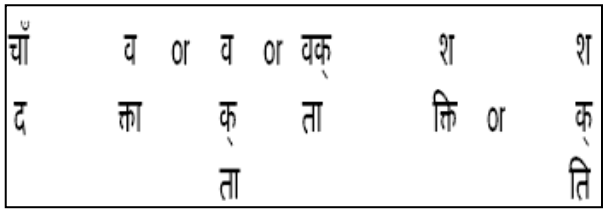
Rule 4: Expression with mathematical symbol should be treated as single unit so that at the end of the line expression shouldn't breaks at operator level

Rule 5: Breaking should not be allowed at numerical values such as currency values, year etc. e.g.

"100.00" or "10,000", nor in "12:59"

(iii) Vertical arrangements of characters

Presentation / Styling issues: A String if written in vertical style have multiple representations as demonstrate through Hindi. Using Indic syllable definition, the first option is the right way to represent.



(iv) Letter Spacing

In case of horizontal alignment of characters, the space is given between the every character in case of English. But in case of Indian language like Bangla, Assamese etc the space may give not in every character but after some portion of the character sequence.

Horizontal justification in HTML page in Indic languages, does not work on few browsers. Hence Indic syllable rule may be followed.

(v) Underlining of the Text

There are some examples of Indian languages in which Matras are not readable. When the text is underlined e.g.

Hindi - अन्य भाषाओं में भी अनुवाद

Punjabi ਗੁਰੂ

Bengali: তাই পুরোনো আর্কাইভ একটু ওলট পালট।

Guajarati - સરદાર ગુજરી

Marathi- मराठी मुला मुलींची नावे

Tamil- நீரிற்குமிழி யிளமை நிறைசெல்வம்

நீரிற் சுருட்டு நெடுந்திரைகள் - நீரில்

எழுத்தாகும் யாக்கை நமரங்கா ளென்னே
வழுத்தாத தெம்பிரான் மன்று

Telugu - శ్రీశైలం ప్రాజెక్ట్‌పై TV9 ప్రోగ్రాం " డ్యామ్ ఇన్ డేంజర్ " పార్ట్ - 2

This may be kept in mind while designing the fonts may be tested in various browsers for proper display of baseline allowing proper rendering of matras below the text.

6.10 XHTML for Mobile

The XHTML Basic document type includes the minimal set of modules required to be an XHTML host language document type, and in addition it includes images, forms, basic tables, and object support. It is designed for Web clients that do not support the full set of XHTML features; for example, Web clients such as mobile phones, PDAs, pagers, and set top boxes. The document type is rich enough for content authoring.

XHTML Basic is designed as a common base that may be extended. The goal of XHTML Basic is to serve as a common language supported by various kinds of user agents.

Visit http://www.w3.org/TR/2010/REC-xhtml-basic-20101123/ for more details

For Difference between HTML5 and HTML 4 please visit http://www.w3.org/TR/2008/WD-html5-diff-20080122/ for more details

6.11 Inherent Indian languages support

In order to handle complex scripts like Indian scripts the mobile devices and the associated mobile operating system should inherently support Indian languages. The device should support, Unicode, Virtual Keyboard, Locale data and fonts. The Indian languages requirement for mobile devices are detailed in Annexure –I.

6.12 Internationalized domain names (IDN):

For Internationalised domain names (IDN) the browser should clearly and legibly display the domain name. If by virtue of user settings, browser displays the PUNYCODE instead of actual IDN,

it may flag so appropriately to the user. Auto fill for Country code top level domain (ccTLD) .BHARAT in Indian languages should be made available.

7 Mobile OK Checker/ Validator

This checker performs various tests on a Web Page to determine its level of mobile-friendliness. The tests are defined in the mobileOK Basic Tests 1.0 specification. A Web Page is mobileOK when it passes all the tests. For further details please refer: http://validator.w3.org/mobile/

8 ISO 639 Language Codes

Any application which has multiple language support in some way, e.g. language-wise interfaces, language-specific validation checks or simply language selection by a user, need to refer to language names in some form. ISO 639 is a standard which provides codes which can be used to refer language names. Compliance with this can ensure seamless exchange of language information.

A tabular representation of ISO 639 language code is given below:

639-3	639-2/639-5	639-1	Language Name
asm	asm	as	Assamese
ben	ben	bn	Bengali
brx			Boro (India)
dgo			Dogri (individual language)
guj	guj	gu	Gujarati
hin	hin	hi	Hindi
kan	kan	kn	Kannada
kas	kas	ks	Kashmiri
kok	kok		Konkani (macro language)
mai	ma		Maithili
mal	mal	ml	Malayalam
mni	mni		Manipuri
mar	mar	mr	Marathi
nep	nep	ne	Nepali (macro language)
ori	ori	or	Oriya (macro language)
pan	pan	pa	Panjabi
sat	sat		Santali
san	san	sa	Sanskrit
snd	snd	sd	Sindhi
tam	tam	ta	Tamil
tel	tel	te	Telugu
urd	urd	ur	Urdu

For further details please refer: http://sil.org/iso639-3/codes.asp

Four letters Script code is available at the following url: http://unicode.org/iso15924/iso15924-codes.html

9 Localization

Schäler (2007) with emphasis on digital content defines Localization as:

> *"Localization is the linguistic and cultural adaptation of digital content to the requirements and locale of a foreign market, and the provision of services and technologies for the management of multilingualism across the digital global information flow."*

Localisation is the process of adaptation of digital content to a target audience; digital content is not only text in digital form, but also multimedia. There are different conceptualisations of people and images across different languages and cultures throughout the world, and localisation tries to capture these different aspects and transform them from one into another.

LISA defined Localization as follows:

> *"Localization refers to the actual adaptation of the product for a specific market. It includes translation, adaptation of graphics, adoption of local currencies, use of proper forms for dates, addresses, and phone numbers, and many other details, including physical structures of products in some cases".*

In order to meet the criteria for bare minimum localisation support, operating systems within handsets should support FUEL, CLDR, syllable based cursor movement, syllable based addition and deletion rules, Bidi algorithm, Collation order and Hyphenation rules. Each of these terms shall be described briefly in the following sections.

9.1 Frequently Used Entries for Localisation (FUEL)

FUEL is an open source initiative to standardize terms for open source software programs. The GIST Group of CDAC is actively participating in the same and has initiated FUEL for "Web", "Standalone Applications" and "Mobile" platforms. It aims at resolving the problem of term inconsistency and lack of standardization in Computer software translation, across various platforms. It also works to provide a standard and consistent terminology for a language. Support for following Indian Languages has been added in this initiative.

- Assamese
- Bengali
- Gujarati
- Hindi
- Kannada

- Maithili
- Malayalam
- Marathi
- Punjabi
- Odiya
- Tamil
- Telugu
- Urdu

For further details please refer: http://fuelproject.org/home/index

9.2 Common Locale Data Repository (CLDR)

The CLDR provides key building blocks for software to support the world's languages. The data in the repository is used by companies for their software internationalization and localization: adapting software to the conventions of different languages for such tasks as formatting of dates, times, time zones, numbers, and currency values; sorting text; choosing languages or countries by name; and many others. CLDR'S provide useful information as to the locale and are therefore crucial from the perspective of localization. Mobile based CLDRs should be made and used to enhance the localisation across different cultures and locales. CLDR mostly comprises of

- Calendars
- Numeric formats,
- Date and Time formats
- Currencies

Sample Extract of Hindi CLDR

</monthWidth>

<monthWidth type="wide">

<month type="1">जनवरी </month>

<month type="2"> फरवरी </month>

<month type="3">मार्च</month>

<month type="4">एप्रैल </month>

<month type="5">मेई </month>

<month type="6">जून </month>

<month type="7">जूलै </month>

<month type="8">अगस्त </month>

<month type="9">सितंबर </month>

<month type="10">अक्तूबर </month>

<month type="11">नवंबर </month>

<month type="12">दिसंबर </month>

For further details please refer: http://*cldr.unicode.org/*

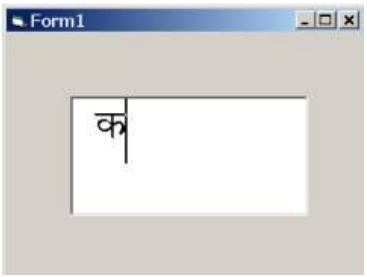
9.3 Indic Based Cursor Movement, addition and deletion

Cursor movement and deletion of characters should be based on ABNF definition of Indic syllable.

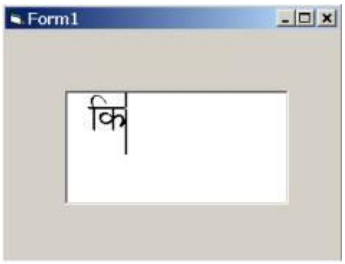
While editing text in Indian Languages cursor position should be changed as per the syllable instead of individual character. While deleting the characters from the entered text, a syllable wise deletion should happen so that it will reduce the burden of processing and redisplaying the half syllable. More details can be visualised in the following images.

eg: **1 :Text- कि (small) Typing sequence in INSCRIPT - kf**

On pressing k with script key on



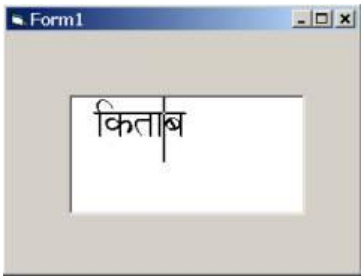
On pressing f with script key on



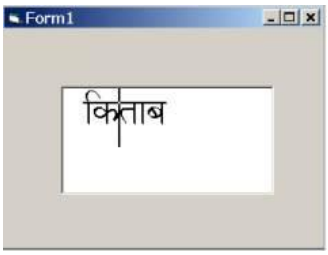
- Cursor movement is syllable based. Selection of Words happens across syllables.

Eg : 1 :Text- किताब

With cursor location: between "ता and ब"



Key Stroke: left arrow



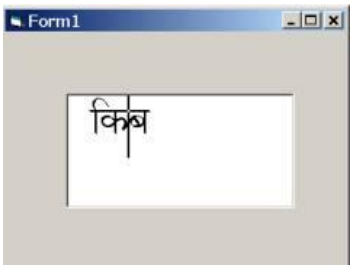
- Syllable level editing

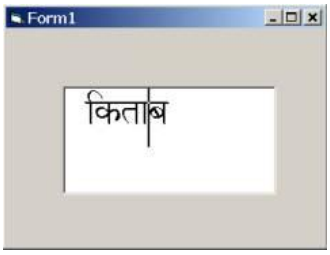
Eg: **1: Text - "किताब"with cursor location: between "ता and ब"**

b) Key Stroke: **Delete**

Result text: "किब"

Result cursor location: between "की and ब"

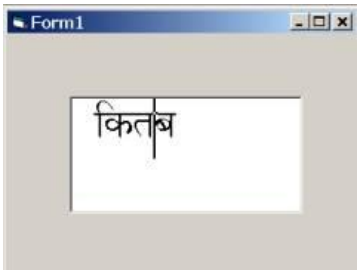




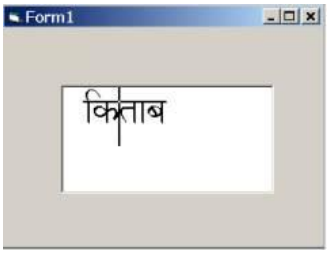
c) Key Stroke: **Backspace**

Result text: "कितब"

Result cursor location: between "त and ब"



2 : Text - "किताब"with cursor location : between "कि and ता"



9.4 Directionality

Using the bi-Directionality algorithm while writing a software application, one can switch between right to left and left to right scripts.

Three languages in India viz., Urdu, Sindhi and Kashmiri in Perso-Arabic script are written in Right to Left direction.

The direction can be set using the following “dir” attributes.

dir = LTR | RTL

e.g. In HTML :

Mohan said ") "السلام عليكمas-salaam alaykum] to me.

Other useful terminologies which can also be used for the same purposes are listed below:

LRE: A short name for the Unicode character U+202A LEFT-TO-RIGHT EMBEDDING. This invisible control character is used to begin a range of text with an embedded base direction of left-to-right.

LRO: A short name for the Unicode character U+202E LEFT-TO-RIGHT OVERRIDE. This invisible control character is used to begin a range of text that ignores the Unicode bidirectional algorithm and arranges characters from left to right.

PDF: A short name for the Unicode character U+202C POP DIRECTIONAL FORMATTING. This invisible control character is used to signal the end of a range of text that was started with one of the RLE, LRE, RLO or LRO characters.

RLE: A short name for the Unicode character U+202B RIGHT-TO-LEFT EMBEDDING. This invisible control character is used to begin a range of text with an embedded base direction of right-to-left.

RLO: A short name for the Unicode character U+202E RIGHT-TO-LEFT OVERRIDE. This invisible control character is used to begin a range of text that ignores the Unicode bidirectional algorithm and arranges characters from right to left.

9.5 Collation Order

The collation order of languages should be as per Unicode CLDR.

Example for Hindi: **अ आ इ ई उ ऊ ऋ ऌ ऍ ए ऐ ऑ ओ औ क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ द ध न प फ ब भ म य र ल ळ व श ष स ह s ा ि ी \u0941 \u0942 \u0943 \u0944 \u0945 \u0947 \u0948 ॉ ो ौ \u094D**

For more details please visit: *http://cldr.unicode.org/index/cldr-spec/collation-guidelines*

9.6 Script Behaviour for Hindi

The term script behaviour refers to the behaviour pattern of the writing system of a given language. Languages which have written representations do not use a haphazard manner of storing the information within the system, but use a coherent pattern which is similar to the linguistic grammar of a given language. With the help of specialists (not necessarily linguist) who work in the area of the written representation of the language, the manner in which the shapes of the characters of the language and the representation of the conjunct forms is provided. In other words it deals with the surface structure of the language and tries to provide the best possible "fit" for shapes and their representation. An example from the Devanagari script will make the above clear. Although Marathi and Hindi share the same script Devanagari, not only do they not share the same character inventory but in addition the representation of certain characters is different. Thus the Hindi /la/ is different from the Marathi /la/ in so far as the placement of the stem is concerned Hindi ल Marathi ळ. This ensures that the Script Behaviour guidelines conform to the language in question and provide the character shapes acceptable to a given user community. It should be noted that this does not mean monotony. The Marathi and Hindi /la/ can have a variety of forms once the intrinsic structure of the character is determined.

Script behaviour may be taken into consideration while developing applications and contents. Devanagari Script behaviour for Hindi is available at :

< http://tdil-dc.in/index.php?option=com_vertical&parentid=79&lang=en>

10 Other Generic Guidelines for mobile Apps

Some generic Guidelines could be:

1. Use Appropriate Client-Side Storage Technologies for Local Data
2. Replicate Local Data
3. Do not Execute Unescaped or Untrusted JSON data
4. Ensure the User is Informed About Use of Personal and Device Information
5. Offer Automatic Sign-in
6. Use Transfer Compression
7. Minimize Application and Data Size : mobile has limited storage
8. Avoid Redirects
9. Optimize Network Requests
10. Minimize External Resources
11. Aggregate Static Images into a Single Composite Resource (Sprites)
12. Include Background Images Inline in CSS Style Sheets
13. Cache Resources By Fingerprinting Resource References
14. Cache AJAX Data
15. Do not Send Cookie Information Unnecessarily
16. Keep DOM Size Reasonable
17. Optimize For Application Start-up Time
18. Minimize Perceived Latency
19. Design for Multiple Interaction Methods
20. Preserve Focus on Dynamic Page Updates
21. Use Fragment IDs to Drive Application View
22. Make Telephone Numbers "Click-to-Call"
23. Ensure Paragraph Text Flows
24. Ensure Consistency Of State Between Devices

25. Consider Mobile Specific Technologies for Initiating Web Applications
26. Use Meta Viewport Element To Identify Desired Screen Size
27. Prefer Server-side Detection Where Possible
28. Use Client-side Detection When Necessary
29. Use Device Classification to Simplify Content Adaptation
30. Support a non-JavaScript Variant if Appropriate
31. Offer Users a Choice of Interfaces

1 Device Specific guidelines

Now-a-days smart phones have made life easier by their rich navigation capabilities. Also the communication technology has considerably evolved in the last few years, allowing a fast mobile data communication. Device specific guidelines are based on the fundamental issues which have to be addressed by various mobile handsets manufacturers with varied hardware support. These guidelines will enable seamless communication and enhance interoperability among different mobile handsets.

For any mobile device to support Indian language there are some basic requirements. Considering those basic requirements the device specific guidelines are categorized into three types viz.

i) Inputting ii) Storage iii) Display

1.1 Inputting

The Inputting of Indian language text onto mobile handset can be done in the following ways:

a. For Mobiles with Physical keypad
    1 12-key keypad
    2 A QWERTY keypad
    3 A touch-screen with 12-Key Keypad
b. For mobiles with Touch screen only (soft in Mobile OS specific section)
    1 A touch-screen with QWERTY Virtual keypad

For implementing IL inputting, some of the existing work like Enhanced INSCRIPT can be referred. Also IL alphabets should be engraved on the keys of the mobile device. For mobile devices having touch screen interfaces only and where the screen size is small, 12-Key Virtual Keyboard inputting mechanism should be preferred. For smart phones where the screen size is more than 3 inches (measured diagonally) INSCRIPT based Virtual Keyboard should be preferred.

1.2 Storage

For mobiles, Indian language data should be stored in Unicode. Unicode is a de-facto standard which is supported by almost all the programming languages, operating systems and applications. Unicode comes with its own forms of encoding in the form of UTF-8/UTF-16. UTF-8 is a widely recognized form of Unicode encoding over the web.

1.3 Unicode

Unicode consortium defines Unicode as:

> *"Unicode is the universal character encoding, maintained by the Unicode consortium. This encoding standard provides the basis for processing, storage and interchange of text data in any language in all modern software and information technology protocols."*

It is the superset of characters within all the languages in the world which also includes punctuation, special characters (shapes), currency symbols, mathematical symbols etc. Using Unicode, more than 65000 different characters can be represented. Unicode comprises of many code pages.

The Unicode code charts can be referred at: http://www.unicode.org/charts/.

Various editors / applications need to understand how to read in the given Unicode data and interpret the same. Various encoding schemes to represent Unicode are UTF-8, UTF-16, UTF-32 with a combination of endian-ness. These encoding schemes are referred below.

Also given below are some of the normalization rules which are required to be followed for data compatibility between various applications / underlying environment. Non adherence to some of these may lead to wrong interpretation of data and will also pose problems in searches as well.

Please refer to www.unicode.org for further details.

1.4 Display

The key component in IL support is the ability of the mobile device to display the IL text by maintaining the intelligibility of Indian languages with smart aesthetic sense. For IL display on a mobile device the following components are required:

- A good aesthetic font, fine-tuned to the particular screen size and resolution.

- A layout engine to manage the complexities of IL scripts such as Akshar boundary identification, reordering of text, cursor movements, scrolling, word-wrapping, character(s) insertion /deletion and hyphenation rules
- A Rasterizer to render the font on screen. (Rasterisation is a process of converting the given vector font into a raster image (pixels or dots) for output on a screen)

1.5 Fonts

A font is a collection of glyphs (shapes) and its associated information such as glyph ID, glyph names, matrix, etc. Usually glyphs that share certain common aesthetic & other design similarities are part of single font. A font family is a set of fonts that represents some design idea i.e. features by which a character's design is recognized. "DV Yogesh" is a font family, even called typeface or simply face e.g. "DV Yogesh Italics "is a font name. Fonts are said to belong to one font family, wherein the font(s) share common design but have different attributes such as width, weight, etc. Devanagari Yogesh family of font may have Normal, Italic, Bold & Bold Italic fonts of style Yogesh.

In order to render the font onto the display or any other medium requires mapping from character encoding to font encoding. Due to complexity of the Indian languages, various glyphs can be mapped to show a single ligature" and some more content if needed.

Fonts are of various types: 8-bit, 16-bit, etc. Most commonly 8 bit fonts are the True Type fonts. There is a native support of True Type Rasteriser in most of the common operating systems. Open Font Format (OFF) are Unicode compliant 16 bit fonts, Wherein one can have the large group of glyph set and typographic rules to truly represent the language(s). In 8-bit fonts one has only 256 code points available. Out of these first 127 are reserved for system requirements, Roman characters including punctuation marks. While certain additional code points are required by various operating systems especially UNIX, Linux, etc. Out of these 256 code points approx. 96 code points are available for accommodating various glyphs for the given Indian language in order to truly represent the same. There are 3 specific types of fonts

(i) Bit map fonts (used by low resolution / low cost handset).

(ii) True type fonts (used by high resolution / handsets)

(iii) Open type fonts (used by high resolution / Smartphone and are currently in wider use)

The bit map fonts are not scalable and hence one may require various sizes of bit map fonts for different application purposes. Different size fonts for menu items, normal text, bold text, italic text, etc.

True Type Font is a scalable vector graphic font where one requires to have rasterization engine to convert the mathematical expression to bitmaps on the fly.

Open Type is a cross-platform font file format developed jointly by Adobe and Microsoft. This font format has an additional facility of positioning the justification and substitution mechanisms where it is more suitable for complex Indian scripts. This is also a scalable vector graphic font.

Low end handsets are currently using bitmap fonts, however, the implementation is proprietary in nature. Because of limited resources (processing power, memory, display size, etc) availability in the low end handsets bitmap fonts are preferred. In case bitmap fonts are used it is advised that the standards for Inputting, storage and communications to be followed.

Most popular rendering engines in use are FreeType, Uniscribe, Pango, ICU, Harfbuzz

Uniscribe: Uniscribe is a Microsoft shaping and rendering engine to ensure proper representation of multilingual content on windows platform.

Pango: Pango is a library for laying out and rendering of text, with an emphasis on internationalization. Pango can be used where text layout is needed, though most of the work on Pango so far has been done in the context of the GTK+ widget toolkit. Pango forms the core of text and font handling for GTK+-2.x.

ICU: ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software.

Harfbuzz: HarfBuzz is an OpenType text shaping engine.

FreeType: is a open source software development library for rendering vector fonts to bitmaps and supports other font-related operations. It supports number of font formats, including TrueType, Type 1, and OpenType. The FreeType library is small, efficient, highly customizable, and portable and is suitable especially for mobile devices

FreeType 1:

FreeType 1 had support for only the TrueType font format but it included an extension to support OpenType text layout features.

FreeType 2 :

In FreeType 2 functionality of FreeType 1 was removed

For further details please refer the following URLs:

ICU : http://site.icu-project.org/

PANGO : http://www.pango.org/

ICU : http://site.icu-project.org/

Harfbuzz : http://www.freedesktop.org/wiki/Software/HarfBuzz

FreeType : www.freetype.org

1.6 WOFF (Web Open Font Format)

Mobile devices are in general distributed with only a limited set of fonts. WOFF (Web Open Font Format) are automatically downloaded through style sheets, while keeping the size of the downloaded fonts limited to what is actually needed to render the interface.

This format was designed to provide lightweight, easy-to-implement compression of the font data, suitable for use in conjunction with the @font-face CSS declaration. Any TrueType/Open Type/Open Font Format file can be lossless converted to WOFF for Web use (subject to licensing of the font data); once decoded by a user agent, the WOFF font will display identically to the original desktop font from which it was created.

The WOFF format also allows additional metadata to be attached to the file; this can be used by font designers to include licensing or other information, beyond that present in the original font. Such metadata does not affect the rendering of the font in any way, but may be displayed to the user on request.

The WOFF format is not expected to replace other formats such as TrueType/Open Type/Open Font Format or SVG fonts, but provides an alternative solution for use cases where these formats may be less performant, or where licensing considerations make their use less acceptable.

Why Use WOFF:

WOFF provides a lot of advantages over other font choices. The standard fonts you can use end up making a web page. WOFF fonts have the following benefits:

- They are more accessible than fonts as images. WOFF styles plain HTML text with CSS.
- WOFF files include typographical information like contextual forms and old style figures. This gives your web pages better typography because you're using the correct characters, not just approximations.
- WOFF fonts can help with internationalization because you can embed fonts with characters from other languages.

- Like all CSS styled text, fonts styled with WOFF are more search engine friendly.
- WOFF fonts are compressed, so they are smaller than other types of font files

1.7 SVG (Scalable Vector Graphics) Fonts:

XML-based mark-up language to describe two-dimension vector graphics. The graphics are described as a set of geometric shapes and can be scaled which makes them well-suited to work on mobile devices.

1.8 How to Use @font-face

To Use @font-face requires a few lines of CSS and then declare the font-family just like you would any other font on the web given below. The following code snippet is taken from: http://boldperspective.com/2011/how-to-use-css-font-face/

```
body { font-family: web-font, fallback-fonts; }
strong { font-family: web-font-bold; }
em { font-family: web-font-italic; }
@font-face {
font-family: 'web-font';
src: url('web-font.eot?') format('eot'),
url('web-font.woff') format('woff'),
url('web-font.ttf') format('truetype'),
url('web-font.svg') format('svg');
font-weight: normal;
font-style: normal;
}
@font-face {
font-family: 'web-font-bold';
src: url('web-font-italic.eot?') format('eot'),
url('web-font-italic.woff') format('woff'),
url('web-font-italic.ttf') format('truetype'),
url('web-font-italic.svg') format('svg');
```

```
font-weight: bold;
font-style: normal;
}
@font-face {
font-family: 'web-font-italic';
src: url('web-font-bold.eot?') format('eot'),
url('web-font-bold.woff') format('woff'),
url('web-font-bold.ttf') format('truetype'),
url('web-font-bold.svg') format('svg');
font-weight: normal;
font-style: italic;
}
```

Annexure II: Acronyms and Abbreviations

1. Acronyms & Abbreviations

CLDR	Common Locale Data Repository
EOT	Embedded Open Type
FUEL	Frequently Used Entries for Localization
IL	Indian Language
ISO	International Organization for Standardization
ISCII	Indian Script Code for Information Interchange
I18N	Internationalization
L10N	Localization
OASIS	Organization for the Advancement of Structured Information Standards
PASCII	Perso-Arabic Script Code for Information Interchange
PFR	Portable Font Resource
SRX	Segmentation Rules eXchange
TBX	Term Base Exchange
TMX	Translation Memory eXchange
WOFF	Web Open Font Format
W3C	World Wide Web Consortium
XLIFF	XML Localization Interchange File Format
3GPP	3rd Generation Partnership Project

Annexure III: References

1. Localisation : http://www.wordesign.com/Localization /index.htm#_Toc469910877 [Accessed: 12 April 2012]
2. Unicode: http://unicode.org [Accessed: 20 April 2012]
3. CDAC: http://cdac.in/downloads [Accessed: 20 April 2012]
4. Lingobit Best Practices: http://www.lingobit.com/solutions/bestpractices.html [Accessed: 30 April 2012]
5. Localisation: http://www.wordesign.com/Localization/index.htm#_Toc469910877 [Accessed: 30 April 2012]
6. I18Guy Localisation: http://www.i18nguy.com/unicode/c-unicode.html [Accessed: 25 April 2012]
7. ISO 639 code for languages: http://sil.org/iso639-3/codes.asp[Accessed: 30 June 2012]
8. 4 Letter Script Code: http://unicode.org/iso15924/iso15924-codes.html[Accessed: 30 June 2012]
9. CLDR: http://cldr.unicode.org/[Accessed: 30 June 2012]
10. Language Tag & Test Direction: http://www.w3.org/TR/html4/struct/dirlang.html[Accessed: 30 June 2012]
11. R. Gupta and S. Unde 'Towards evolution of localisation standards in Indian scenario', Translation, Technology And Globalization in Multilingual Context 3rd International Conference, June 23-26, 2012, New Delhi, [URL: India http://www.itaindia.org/ITAINDIA_conference_2012.pdf]
12. IBM – Globalisation Guidelines: http://www-01.ibm.com/software/globalization/[Accessed: 30 June 2012]
13. Indic Scripts and Unicode http://www.Unicode.org/standard/WhatIsUnicode.html [Accessed on 21 May 2012]
14. ISO/IEC Guide 2:2004, Standardization and related activities – General vocabulary.
15. TDIL: url: tdil.mit.gov.in [Accessed on 10-April-2012]
16. PASCII: url: http://parc.cdac.in/PASCii.htm [Accessed on: 29 May 2012]

17. Esselink, B. (2000). A Practical Guide to Localisation, Amsterdam: Benjamins.
18. Sikes, R. (2009) 'localisation: the global pyramid capstone', Multilingual, April, pp.3-5.
19. ICU: http://site.icu-project.org/ [Accessed on 29 May 2012]
20. PANGO: http://www.pango.org/ [Accessed on 29 May 2012]
21. Harfbuzz: http://www.freedesktop.org/wiki/Software/HarfBuzz [Accessed on 29 May 2012]
22. FUEL: https://fedorahosted.org/fuel/wiki/fuel-hindi [Accessed on 20 July 2012]
23. Font face: http://boldperspective.com/2011/how-to-use-css-font-face/ [Accessed on 25 July 2012]
24. D. Anastasiou and R. Gupta, 'Comparison of crowdsourcing translation with Machine Translation', Journal of Information Science, vol. 37, no. 6, pp. 637-659, 2011.
25. http://www.w3.org/TR/2009/NOTE-di-testing-20090512/
26. Mobile Web Generic Guidelines: http://www.w3.org/TR/mobile-bp/
27. http://www.w3.org/Mobile/mobile-web-app-state/
28. http://www.w3.org/standards/webdesign/mobilweb
29. http://mobilehtml5.org/
30. N.S. Nikolov, R. Gupta and M. O'Haodha, Crowdsourcing and pedagogy: some reflections on the museum as collaborative learning space, 5th QQML conference [URL: http://www.isast.org/images/Book_of_ABSTRACTS_2013.pdf]
31. http://html5center.sourceforge.net/blog/What-to-Expect-from-HTML5-in-2013
32. https://en.wikipedia.org/wiki/XHTML_Mobile_Profile
33. https://en.wikipedia.org/wiki/XHTML_Basic
34. http://www.w3.org/TR/WOFF/
35. http://caniuse.com/woff;
36. https://en.wikipedia.org/wiki/Web_Open_Font_Format
37. CLDR: http://cldr.unicode.org/[Accessed: 30 June 2012]