Introduction to HTML

Web Technology

Dr. Navanath Saharia
Indian Institute of Information Technology Senapati, Manipur
nsaharia@iiitmanipur.ac.in

Syllabus

- Unit 1: HTML5, CSS3 and XML: Introduction to markup language, elements of Html5, controlling of Form elements, Dynamic graphics (canvas, SVG, etc.), controlling of audio and video elements; Introduction to CSS, type, elements and their attributes, layout, controlling of motion and colours; Introduction to XML, Defining XML tags, their attributes and values, Document type definition, XML Schemas, Document Object model, XHTML, Parsing XML Data (DOM and SAX parsers), UI framework: Bootstrap 4

Syllabus

- Unit 1: HTML5, CSS3 and XML: Introduction to markup language, elements of Html5, controlling of Form elements, Dynamic graphics (canvas, SVG, etc.), controlling of audio and video elements; Introduction to CSS, type, elements and their attributes, layout, controlling of motion and colours; Introduction to XML, Defining XML tags, their attributes and values, Document type definition, XML Schemas, Document Object model, XHTML, Parsing XML Data (DOM and SAX parsers), UI framework: Bootstrap 4
- Unit 2: Client Side Scripting: Introduction to JavaScript, declaring variables, scope of variables functions, event handlers, Document Object Model, Form validations.

Syllabus

- Unit 1: HTML5, CSS3 and XML: Introduction to markup language, elements of Html5, controlling of Form elements, Dynamic graphics (canvas, SVG, etc.), controlling of audio and video elements; Introduction to CSS, type, elements and their attributes, layout, controlling of motion and colours; Introduction to XML, Defining XML tags, their attributes and values, Document type definition, XML Schemas, Document Object model, XHTML, Parsing XML Data (DOM and SAX parsers), UI framework: Bootstrap 4
- Unit 2: Client Side Scripting: Introduction to JavaScript, declaring variables, scope of variables functions, event handlers, Document Object Model, Form validations.
- Unit 3: JavaScript Frameworks: ReactJS, AngularJS, VueJS, architectures, Model-view-controller, virtual DOM. Server Side Scripting using *Stack: Introduction to Node.JS, ExpressJS, MongoDB, Data Flow in MEAN and MERN stack, architectures, example application;

Syllabus

- Unit 1: HTML5, CSS3 and XML: Introduction to markup language, elements of Html5, controlling of Form elements, Dynamic graphics (canvas, SVG, etc.), controlling of audio and video elements; Introduction to CSS, type, elements and their attributes, layout, controlling of motion and colours; Introduction to XML, Defining XML tags, their attributes and values, Document type definition, XML Schemas, Document Object model, XHTML, Parsing XML Data (DOM and SAX parsers), UI framework: Bootstrap 4
- Unit 2: Client Side Scripting: Introduction to JavaScript, declaring variables, scope of variables functions, event handlers, Document Object Model, Form validations.
- Unit 3: JavaScript Frameworks: ReactJS, AngularJS, VueJS, architectures, Model-view-controller, virtual DOM. Server Side Scripting using *Stack: Introduction to Node.JS, ExpressJS, MongoDB, Data Flow in MEAN and MERN stack, architectures, example application;
- Unit 4: Server Side Scripting using PHP: Introduction to PHP, Declaring variables, data types, arrays, strings, operations, expressions, control structures, functions, Reading data from web form controls like Text Boxes, radio buttons, lists etc., Handling File Uploads, Connecting to database (MySQL/MariaDB as reference), executing simple queries, handling results, Handling sessions and cookies; File operations like opening, closing, reading, writing, appending, deleting etc. on text and binary files, listing directories.

- Evaluation
  - Assessment 1 : 25 Marks → Unit 1
  - Assessment 2 : 25 Marks → Unit 2
  - Assessment 3 : 25 Marks → Unit 3 OR (Quiz, Mini project and/or Assignment)
  - Assessment 4: End Term : 100 Marks → [30 % from Unit 1 and Unit 2] + [70% from Uni 3 and Unit 4]

- Evaluation
  - Assessment 1 : 25 Marks → Unit 1
  - Assessment 2 : 25 Marks → Unit 2
  - Assessment 3 : 25 Marks → Unit 3 OR (Quiz, Mini project and/or Assignment)
  - Assessment 4: End Term : 100 Marks → [30 % from Unit 1 and Unit 2] + [70% from Uni 3 and Unit 4]
  - Grading will be based on the total scaled down score (to 50) from Assessment 1, Assessment 2 and Assessment 3 and total scaled down score (to 50) from End term.

- Evaluation
  - Assessment 1 : 25 Marks → Unit 1
  - Assessment 2 : 25 Marks → Unit 2
  - Assessment 3 : 25 Marks → Unit 3 OR (Quiz, Mini project and/or Assignment)
  - Assessment 4: End Term : 100 Marks → [30 % from Unit 1 and Unit 2] + [70% from Uni 3 and Unit 4]
  - Grading will be based on the total scaled down score (to 50) from Assessment 1, Assessment 2 and Assessment 3 and total scaled down score (to 50) from End term.
- 100% attendance is mandatory. Regarding medical and other emergency leave need to address to academic section.

Text book and References

- Matthew MacDonald, Creating a Website - The Missing Manual, 4th ed, 2015, O'Reilly.
- Programming world wide web, R.W. Sebesta. Fourth Edition, Pearson.
- Internet and World Wide Web – How to program, Dietel and Nieto, Pearson
- Greg Lim, Beginning MERN Stack: Build and Deploy a Full Stack MongoDB, Express, React, Node.js App
- Cris Bates, Web Programming: Building Internet Applications, 3ed, Wiley
- HTML5, CSS3, JavaScript, PHP Tutorials http://www.w3schools.com
- jQuery Tutorial https://learn.jquery.com
- MongoDB Tutorial and Certifications https://university.mongodb.com
- Express https://expressjs.com/en/starter/installing.html
- React Tutorial https://reactjs.org/tutorial/tutorial.html
- Node https://nodeschool.io

What is HTML?

- HTML → HyperText Markup Language → Language used to create Web pages
  - Language → Communication medium
  - Markup language → Sequence of characters within a text or word processing file to define print or display properties and document's logical structure
  - HyperText → System of writing that allows users to navigate between different pieces of content by clicking on hyperlinks. It is a key feature of the World Wide Web and allows users to easily access information on a variety of topics by clicking on links embedded in web pages.

What is HTML?

- HTML → HyperText Markup Language → Language used to create Web pages
  - Language → Communication medium
  - Markup language → Sequence of characters within a text or word processing file to define print or display properties and document's logical structure
  - HyperText → System of writing that allows users to navigate between different pieces of content by clicking on hyperlinks. It is a key feature of the World Wide Web and allows users to easily access information on a variety of topics by clicking on links embedded in web pages.
  - Example of Markup language → HTML, XML, MathML, Open Document Format (ODF), MusicXML, (La)TeX
  - Types of mark-up languages: Presentation markup (ODF), Procedural markup (TeX) and descriptive (logical or conceptual) markup (HTML, XML)
- It is a specialized form of SGML → Standard Generalized Markup Language

What is SGML?

- SGML → Standardized system for organizing and annotating electronic documents. A Meta-language for defining document markup vocabularies.
  - Meta-language → Framework that provides a set of rules for defining other languages, which can be used to create specific markup languages for particular domains or applications.
  - Developed in the 1970s as a way to standardize the exchange of documents between different computer systems, and it became an ISO standard in 1986.
  - Designed to be flexible and extensible, so that it could be used to remove dependency for a wide range of platform, system, vendor and version-dependent documents.
- In context of W3 Supporting full SGML on the Web was too difficult so HTML made some simplifications
  - not extensible
  - limited structure
  - not content oriented
  - cannot be validated

What is Markdown?

A lightweight markup language for creating formatted text using a plain-text editor (RFC 7763 introduced MIME type `text/markdown`). John Gruber and Aaron Swartz created Markdown in 2004.

What is Markdown?

A lightweight markup language for creating formatted text using a plain-text editor (RFC 7763 introduced MIME type `text/markdown`). John Gruber and Aaron Swartz created Markdown in 2004.

- Lightweight markup language → designed to be simple and easy to use, with a minimal syntax and a focus on readability

What is Markdown?

A lightweight markup language for creating formatted text using a plain-text editor (RFC 7763 introduced MIME type `text/markdown`). John Gruber and Aaron Swartz created Markdown in 2004.

- Lightweight markup language → designed to be simple and easy to use, with a minimal syntax and a focus on readability
- Widely used in blogging, instant messaging, online forums, collaborative software, documentation pages, and readme files.
- Example of Websites: GitHub, Bitbucket, Reddit, Stack Exchange, OpenStreetMap and SourceForge

What is Markdown?

A lightweight markup language for creating formatted text using a plain-text editor (RFC 7763 introduced MIME type `text/markdown`). John Gruber and Aaron Swartz created Markdown in 2004.

- Lightweight markup language → designed to be simple and easy to use, with a minimal syntax and a focus on readability
- Widely used in blogging, instant messaging, online forums, collaborative software, documentation pages, and readme files.
- Example of Websites: GitHub, Bitbucket, Reddit, Stack Exchange, OpenStreetMap and SourceForge

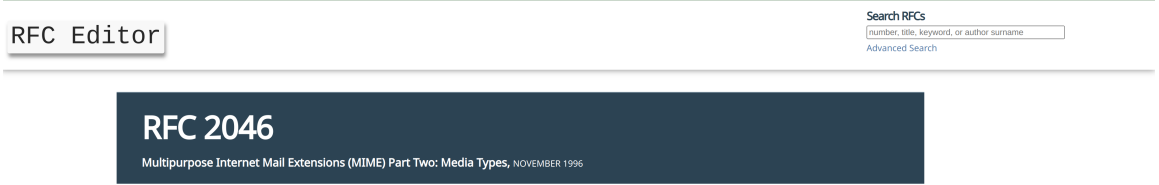
Text using Markdown syntax	Corresponding HTML produced by a Markdown processor	Text viewed in a browser
`Heading` `=======` `Sub-heading` `----------` `# Alternative heading #` `## Alternative sub-heading ##` `Paragraphs are separated` `by a blank line.` `Two spaces at the end of a line` `produce a line break.`	`<h1>Heading</h1>` `<h2>Sub-heading</h2>` `<h1>Alternative heading</h1>` `<h2>Alternative sub-heading</h2>` `<p>Paragraphs are separated` `by a blank line.</p>` `<p>Two spaces at the end of a line ` `produce a line break.</p>`	Heading Sub-heading Alternative heading Alternative sub-heading Paragraphs are separated by a blank line. Two spaces at the end of a line produce a line break.

RFC → Request for Comments

- A series of documents published by the Internet Engineering Task Force, an organization that develops and maintains technical standards for the Internet.

RFC → Request for Comments

- A series of documents published by the Internet Engineering Task Force, an organization that develops and maintains technical standards for the Internet.
- It is authored by individuals or groups of engineers and computer scientists in the form of a memorandum describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems for peer review or to convey new concepts or information
- RFCs are submitted as plain ASCII text and is published in that form and are static. For any changes it is submitted again and assigned a new RFC number. IETF adopts some of the proposals published as RFC as Internet Standards
- Official source: https://www.rfc-editor.org/rfc.html



Media type or MIME → Multipurpose Internet Mail Extensions

- A Standard, used to identify files or content types on the Internet while exchanging files according to their nature and format.

Media type or MIME → Multipurpose Internet Mail Extensions

- A Standard, used to identify files or content types on the Internet while exchanging files according to their nature and format. For example, a file with the extension .txt is typically identified as a plain text file, while a file with the extension .jpg is typically identified as a JPEG (Joint Photographic Experts Group) image. This allows the receiving system to properly handle the content, whether it is a simple text message or a more complex multimedia file.

Media type or MIME → Multipurpose Internet Mail Extensions

- A Standard, used to identify files or content types on the Internet while exchanging files according to their nature and format. For example, a file with the extension .txt is typically identified as a plain text file, while a file with the extension .jpg is typically identified as a JPEG (Joint Photographic Experts Group) image. This allows the receiving system to properly handle the content, whether it is a simple text message or a more complex multimedia file.
- `MIME-Version: 1.0` declares that the message adheres to the MIME format, allowing email clients and servers to:
  - Interpret various content types beyond plain text.
  - Handle attachments, multiple character sets, and non-English text.

Media type or MIME → Multipurpose Internet Mail Extensions

- A Standard, used to identify files or content types on the Internet while exchanging files according to their nature and format. For example, a file with the extension .txt is typically identified as a plain text file, while a file with the extension .jpg is typically identified as a JPEG (Joint Photographic Experts Group) image. This allows the receiving system to properly handle the content, whether it is a simple text message or a more complex multimedia file.
- `MIME-Version: 1.0` declares that the message adheres to the MIME format, allowing email clients and servers to:
  - Interpret various content types beyond plain text.
  - Handle attachments, multiple character sets, and non-English text.
- Media-types comprises of `header` and `body` section.
- `Header` contains information regarding the actual content of `body` section and comprises of `content-type`, `Content-disposition`, and `Content-transfer-encoding`
  - General Syntax of `content-type`: type/[tree .] subtype [+suffix]* [; parameter]
  - Example: text/html, text/css, application/pdf, image/jpeg

Media type: Example

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="........"

Content-Type: multipart/related; boundary=".........."; type="text/html"

Content-Type: multipart/alternative; boundary=".........."
Content-Transfer-Encoding: quoted-printable  ......

Content-Type: text/html; charset="UTF-8"
Content-Range: bytes 100-200

Content-Type: image/png; name=logo.png
Content-Disposition: inline; filename=logo.png

Content-Type: image/png; name=abc.png
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=abc.png
```

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

- `Content-type` consists of a `type` and a `subtype` with optional `suffix` and `parameter`.
  - `type` defines the broad use of the media types such as application, audio, example, image, font, message, multipart, model, text, and video

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

- `Content-type` consists of a `type` and a `subtype` with optional `suffix` and `parameter`.
  - `type` defines the broad use of the media types such as application, audio, example, image, font, message, multipart, model, text, and video
  - `subtype` deals with media format registered through IANA (Internet Assigned Numbers Authority), which are maintained in the form of a tree. For example `vnd` prefix for vendor, `prs` prefix for personal, `x` prefix for unregistered

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

- `Content-type` consists of a `type` and a `subtype` with optional `suffix` and `parameter`.
  - `type` defines the broad use of the media types such as application, audio, example, image, font, message, multipart, model, text, and video
  - `subtype` deals with media format registered through IANA (Internet Assigned Numbers Authority), which are maintained in the form of a tree. For example `vnd` prefix for vendor, `prs` prefix for personal, `x` prefix for unregistered
- Suffix is an augmentation to the media type definition to additionally specify the underlying structure of that media type, allowing for generic processing based on that structure and independent of the exact type's particular semantics. For example, `application/epub+zip`, `model/x3d+binary`, `image/svg+xml`, and `application/ld+json`
- The optional last field Parameter is used to specify encoding schemes used in content, language of the content etc.

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

- `Content-type` consists of a `type` and a `subtype` with optional `suffix` and `parameter`.
  - `type` defines the broad use of the media types such as application, audio, example, image, font, message, multipart, model, text, and video
  - `subtype` deals with media format registered through IANA (Internet Assigned Numbers Authority), which are maintained in the form of a tree. For example `vnd` prefix for vendor, `prs` prefix for personal, `x` prefix for unregistered
- Suffix is an augmentation to the media type definition to additionally specify the underlying structure of that media type, allowing for generic processing based on that structure and independent of the exact type's particular semantics. For example, `application/epub+zip`, `model/x3d+binary`, `image/svg+xml`, and `application/ld+json`
- The optional last field Parameter is used to specify encoding schemes used in content, language of the content etc.
- Types, subtypes, and parameter names are case-insensitive while `Parameter values are not`

Header >> Content-type

```
type "/" [tree "."] subtype ["+" suffix]* [";" parameter];
```

- `Content-type` consists of a `type` and a `subtype` with optional `suffix` and `parameter`.
  - `type` defines the broad use of the media types such as application, audio, example, image, font, message, multipart, model, text, and video
  - `subtype` deals with media format registered through IANA (Internet Assigned Numbers Authority), which are maintained in the form of a tree. For example `vnd` prefix for vendor, `prs` prefix for personal, `x` prefix for unregistered
- Suffix is an augmentation to the media type definition to additionally specify the underlying structure of that media type, allowing for generic processing based on that structure and independent of the exact type's particular semantics. For example, `application/epub+zip`, `model/x3d+binary`, `image/svg+xml`, and `application/ld+json`
- The optional last field Parameter is used to specify encoding schemes used in content, language of the content etc.
- Types, subtypes, and parameter names are case-insensitive while `Parameter values are not`
- Registrations in the standards tree must either be associated with IETF specifications or registered by IANA. Details https://www.rfc-editor.org/rfc/rfc6838.html

Header >> Content-disposition

- `Content-disposition` indicates how the content should be processed. Primary types are:
  - Inline → Indicates that the content should be automatically displayed when the message is displayed
  - Attachment → Indicates that some form of action is required from the user to open it. Content should not display automatically.
- In addition to the presentation style, the `Content-disposition` field also provides parameters for specifying the name of the file, date of creation/modification etc.

```
Content-Disposition: attachment; filename="...."; modification-date="...."

Content-Disposition: inline; filename=logo.png
```

Header >> Content-transfer-encoding

- It indicates whether or not a `binary-to-text encoding scheme` has been used on the top of the original encoding as specified within the `Content-Type`. It deals `hop-to-hop` encoding to transmit data not the `end-to-end` encoding

Header >> Content-transfer-encoding

- It indicates whether or not a `binary-to-text encoding scheme` has been used on the top of the original encoding as specified within the `Content-Type`. It deals `hop-to-hop` encoding to transmit data not the `end-to-end` encoding
- In other words this field is used to indicate the type of transformation that has been used in order to represent the body in an acceptable manner for transport.
- It may have `BASE64`, `QUOTED-PRINTABLE`, `8BIT`, `7BIT`, `BINARY`, and `x-token` as value, which are case insensitive. In absence of this field 7BIT is considered as default.

Header >> Content-transfer-encoding

- It indicates whether or not a `binary-to-text encoding scheme` has been used on the top of the original encoding as specified within the `Content-Type`. It deals `hop-to-hop` encoding to transmit data not the `end-to-end` encoding
- In other words this field is used to indicate the type of transformation that has been used in order to represent the body in an acceptable manner for transport.
- It may have `BASE64`, `QUOTED-PRINTABLE`, `8BIT`, `7BIT`, `BINARY`, and `x-token` as value, which are case insensitive. In absence of this field 7BIT is considered as default.
  - The difference between `8bit` and the `binary` is that `binary` does not require adherence to any limits on line length or to the SMTP CRLF semantics, while the bit-width tokens do require such adherence.
  - If the body contains data in any bit-width other than 7-bit, the appropriate bit-width Content-Transfer-Encoding token must be used (e.g., `8bit` for unencoded 8 bit wide data). If the body contains binary data, the `binary` Content-Transfer-Encoding token must be used.

Header >> Content-transfer-encoding

- It indicates whether or not a `binary-to-text encoding scheme` has been used on the top of the original encoding as specified within the `Content-Type`. It deals `hop-to-hop` encoding to transmit data not the `end-to-end` encoding
- In other words this field is used to indicate the type of transformation that has been used in order to represent the body in an acceptable manner for transport.
- It may have `BASE64`, `QUOTED-PRINTABLE`, `8BIT`, `7BIT`, `BINARY`, and `x-token` as value, which are case insensitive. In absence of this field 7BIT is considered as default.
  - The difference between `8bit` and the `binary` is that `binary` does not require adherence to any limits on line length or to the SMTP CRLF semantics, while the bit-width tokens do require such adherence.
  - If the body contains data in any bit-width other than 7-bit, the appropriate bit-width Content-Transfer-Encoding token must be used (e.g., `8bit` for unencoded 8 bit wide data). If the body contains binary data, the `binary` Content-Transfer-Encoding token must be used.
- However, majority of mail user agents did not follow this rule

Working strategy of base64 encoding scheme

- A Binary-to-text encoding scheme.
- Other examples: Base45 (https://www.rfc-editor.org/rfc/rfc9285), Base122 (https://blog.kevinalbs.com/base122), uuencoding, yEnc (http://www.yenc.org)

Working strategy of base64 encoding scheme

- A Binary-to-text encoding scheme.
- Other examples: Base45 (https://www.rfc-editor.org/rfc/rfc9285), Base122 (https://blog.kevinalbs.com/base122), uuencoding, yEnc (http://www.yenc.org)
- Used 65 characters : 26 uppercase letters (0-25), 26 lowercase letters (26-51), 10 digits (52-61) 2 symbols (+, /, 62-63) and one symbol (=) for padding. In case of URL and Filename, instead of plus (+) and and slash(/), the minus (-) and underscore (_) characters are used.

Working strategy of base64 encoding scheme

- A Binary-to-text encoding scheme.
- Other examples: Base45 (https://www.rfc-editor.org/rfc/rfc9285), Base122 (https://blog.kevinalbs.com/base122), uuencoding, yEnc (http://www.yenc.org)
- Used 65 characters : 26 uppercase letters (0-25), 26 lowercase letters (26-51), 10 digits (52-61) 2 symbols (+, /, 62-63) and one symbol (=) for padding. In case of URL and Filename, instead of plus (+) and and slash(/), the minus (-) and underscore (_) characters are used.
- Working strategy:
  - `binStr` = convert the string to ASCII
  - club each 6 bits of `binStr` into one chunk
  - for each chunk in `binStr`
    - convert each chunk to decimal
    - translate each decimal value to character using `base64 Index table`
  - Print the characters

Working strategy of base64 encoding scheme

- A Binary-to-text encoding scheme.
- Other examples: Base45 (https://www.rfc-editor.org/rfc/rfc9285), Base122 (https://blog.kevinalbs.com/base122), uuencoding, yEnc (http://www.yenc.org)
- Used 65 characters : 26 uppercase letters (0-25), 26 lowercase letters (26-51), 10 digits (52-61) 2 symbols (+, /, 62-63) and one symbol (=) for padding. In case of URL and Filename, instead of plus (+) and and slash(/), the minus (-) and underscore (_) characters are used.
- Working strategy:
  - `binStr` = convert the string to ASCII
  - club each 6 bits of `binStr` into one chunk
  - for each chunk in `binStr`
    - convert each chunk to decimal
    - translate each decimal value to character using `base64 Index table`
  - Print the characters

```
$base64 <<< abc
YWJjCg==

$base64 path/to/file > abc
```

```
$echo -n abc | base64
YWJj

$base64 -d abc
```

base64 Index table

Value	Encoding		Value	Encoding		Value	Encoding		Value	Encoding
0	A		17	R		34	i		51	z
1	B		18	S		35	j		52	0
2	C		19	T		36	k		53	1
3	D		20	U		37	l		54	2
4	E		21	V		38	m		55	3
5	F		22	W		39	n		56	4
6	G		23	X		40	o		57	5
7	H		24	Y		41	p		58	6
8	I		25	Z		42	q		59	7
9	J		26	a		43	r		60	8
10	K		27	b		44	s		61	9
11	L		28	c		45	t		62	+
12	M		29	d		46	u		63	/
13	N		30	e		47	v			
14	O		31	f		48	w		(pad)	=
15	P		32	g		49	x			
16	Q		33	h		50	y			

Source: https://datatracker.ietf.org/doc/html/rfc4648#section-4

MIME Type: multipart

- `Multipart MIME-types` comprises of discrete MIME-types against each component separated by a unique string of characters that acts as a delimiter to separate different parts termed as boundary.
- Different types: `multipart/form-data`, `multipart/alternative`, `multipart/related`, `multipart/byteranges`, `multipart/digest`, `multipart/mixed`, `multipart/parallel`
- Content-Type: multipart/related vs multipart/alternative
  - `Content-Type: multipart/related` combines related parts that need to be presented together to render the complete message correctly. Let us consider an email with embedded images. The HTML part references the images using Content-ID headers, and the "related" type ensures all parts are delivered and understood as a single document.
  - `Content-Type: multipart/alternative` implies multiple versions of the same content, where, only one part is displayed at a time, depending on the user's preferences or device capabilities. Let us consider an email (which is a combination of plain text and HTML versions). The recipient's email client chooses the most appropriate version based on their settings and capabilities.
- MIME sniffing: A mechanism, where, browsers may perform a test to identify the correct MIME type by looking at the bytes of the resource, in the absence of MIME types

Back to *NIX

Location where all the media-types are stored

```
$sudo cat /etc/mime.types
 application/vnd.oasis.opendocument.text   odt
 application/epub+zip                      epub
 application/x-debian-package              deb
 audio/mpeg                                mpga mpega mp2 mp3 m4a
 font/otf                                  ttf otf
 image/png                                 png
 message/rfc822                            eml
```

Back to *NIX

Location where all the media-types are stored

```
$sudo cat /etc/mime.types
 application/vnd.oasis.opendocument.text   odt
 application/epub+zip                      epub
 application/x-debian-package              deb
 audio/mpeg                                mpga mpega mp2 mp3 m4a
 font/otf                                  ttf otf
 image/png                                 png
 message/rfc822                            eml
```

Extracting media-type by using `file` command

```
$file --mime-type base64.png
base64.png: image/png
```

Back to *NIX

Location where all the media-types are stored

```
$sudo cat /etc/mime.types
 application/vnd.oasis.opendocument.text   odt
 application/epub+zip                      epub
 application/x-debian-package              deb
 audio/mpeg                                mpga mpega mp2 mp3 m4a
 font/otf                                  ttf otf
 image/png                                 png
 message/rfc822                            eml
```

Extracting media-type by using `file` command

```
$file --mime-type base64.png
base64.png: image/png
```

Extracting media-type by using `xdg-mime` utility

```
$xdg-mime query filetype base64.png
image/png
```

Standard organization

IETF → Internet Engineering Task Force → International standards organization for the Internet and is responsible for the technical standards that make up the Internet protocol suite. It was established in 1986.

Standard organization

IETF → Internet Engineering Task Force → International standards organization for the Internet and is responsible for the technical standards that make up the Internet protocol suite. It was established in 1986.

W3C → World Wide Web Consortium → International standard organization for the World Wide Web. Founded in 1994 and led by Tim Berners-Lee at the CSAIL Lab MIT with support from the ERCIM, and ARPANET.

Standard organization

IETF → Internet Engineering Task Force → International standards organization for the Internet and is responsible for the technical standards that make up the Internet protocol suite. It was established in 1986.

W3C → World Wide Web Consortium → International standard organization for the World Wide Web. Founded in 1994 and led by Tim Berners-Lee at the CSAIL Lab MIT with support from the ERCIM, and ARPANET.


```
World Wide Web is a platform that runs on top of the Internet, which is a
global network of interconnected computer networks. The Web allows users to
access and share information, while the Internet enables the communication
and connection of computers and devices around the world.
```

XHTML

- In 2000, W3C announced that XHTML will be the future of Web

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML?

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML? A metalanguage

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML? A metalanguage
- Extensible HyperText Markup Language is an application belongs to XML family, that visualizes HTML as an XML document, so that HTML documents can readily be viewed, edited, and validated with standard XML tools.

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML? A metalanguage
- Extensible HyperText Markup Language is an application belongs to XML family, that visualizes HTML as an XML document, so that HTML documents can readily be viewed, edited, and validated with standard XML tools.
- What is XML?

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML? A metalanguage
- Extensible HyperText Markup Language is an application belongs to XML family, that visualizes HTML as an XML document, so that HTML documents can readily be viewed, edited, and validated with standard XML tools.
- What is XML? Extensible Markup Language is a flexible, text-based, platform/device-independent and self-describing SGML with strict validation.

XHTML

- In 2000, W3C announced that XHTML will be the future of Web
- What is XHTML? A metalanguage
- Extensible HyperText Markup Language is an application belongs to XML family, that visualizes HTML as an XML document, so that HTML documents can readily be viewed, edited, and validated with standard XML tools.
- What is XML? Extensible Markup Language is a flexible, text-based, platform/device-independent and self-describing SGML with strict validation.
- Application of XML includes: data-exchange over layer/application/framework, management of system/software configuration details, content management

Conflict between Browser vendors and W3C

Web Hypertext Application Technology Working Group → Formed after various members of the W3C became agitated by the direction being taken with XHTML. They preferred a different, less drastic approach, where the existing HTML was extended.

Conflict between Browser vendors and W3C

Web Hypertext Application Technology Working Group → Formed after various members of the W3C became agitated by the direction being taken with XHTML. They preferred a different, less drastic approach, where the existing HTML was extended.

The WHATWG Community is interested in evolving HTML and related technologies. It was founded by individuals from Apple Inc., the Mozilla Foundation and Opera Software, leading Web browser vendors in 2004. The community lead by Ian Hickson and David Hyatt released the first version of HTML5 in January 2008.

Conflict between Browser vendors and W3C

Web Hypertext Application Technology Working Group → Formed after various members of the W3C became agitated by the direction being taken with XHTML. They preferred a different, less drastic approach, where the existing HTML was extended.

The WHATWG Community is interested in evolving HTML and related technologies. It was founded by individuals from Apple Inc., the Mozilla Foundation and Opera Software, leading Web browser vendors in 2004. The community lead by Ian Hickson and David Hyatt released the first version of HTML5 in January 2008.


```
The Web is, and should be, driven by technical merit, not consensus. The
W3C pretends otherwise, and wastes a lot of time for it. The WHATWG does
not. - Ian Hickson
As of today, W3C and WHATWG are working together (MoU
https://www.w3.org/2019/04/WHATWG-W3C-MOU.html) for the development of
HTML 5
```

WHATWG

- The WHATWG works on a number of technologies that are fundamental parts of the web platform.
- HTML5 is widely used as a buzzword to refer to modern web technologies, many of which are developed at the WHATWG.
  - Compatibility (@compatstandard): Describes a collection of non-standard and often vendor-prefixed CSS properties
  - Console (@consolelog): Defines APIs for console debugging facilities.
  - DOM (@thedomstandard): Defines the core infrastructure used to define the web.
  - Encoding (@encodings): Defines how character encodings work on the web.
  - Fetch (@fetchstandard): Defines the networking model for resource retrieval on the web.
  - File System (@whatfilesystem): Defines infrastructure and an API for file systems.
  - Fullscreen API (@fullscreenapi): Defines how web pages can take over a user's entire screen (at the user's request), e.g., for gaming or to watch a video.
  - HTML (@htmlstandard): Defines the core markup language for the web, HTML, as well as numerous APIs
  - Infra (@infrastandard): Define the fundamental concepts upon which standards are built.

WHATWG

- MIME Sniffing (@mimesniff): Defines algorithms used to determine the type of resources.
- Notifications API (@notifyapi): Provides an API to display notifications to alert users outside the context of a web page.
- Quirks Mode (@quirksstandard): Describes behaviours in CSS and Selectors that are not yet defined in the relevant specifications but that are nonetheless widely implemented.
- Storage (@storagestandard): Defines an API for persistent storage and quota estimates, as well as the platform storage architecture.
- Streams (@streamsstandard): Provides APIs for creating, composing, and consuming streams of data that map efficiently to low-level I/O primitives.
- Test Utils (@testutils): Defines internal APIs for automating testing of web platform features implemented in web browsers.
- URL (@urlstandard): Defines the infrastructure around URLs on the web.
- URL Pattern (@urlpatterns): Provides a web platform primitive for matching URLs based on a convenient pattern syntax.
- Web IDL (@webidl): Defines an interface definition language, Web IDL, that can be used to describe interfaces that are intended to be implemented in web browsers.
- WebSockets (@whatsockets): Provides APIs to enable web applications to maintain bidirectional

Markup Indicator

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

Markup Indicator

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

Markup Indicator

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

The purpose of the markup language is to relieve the content provider from worrying about the actual appearance of the document. The author merely indicates via markup tags the semantic meaning of the words and sentences (such as paragraph, heading, emphasis, and strong), and leave it to the browser to interpret the markups and render the document for display on the screen.

Markup Indicator

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

The purpose of the markup language is to relieve the content provider from worrying about the actual appearance of the document. The author merely indicates via markup tags the semantic meaning of the words and sentences (such as paragraph, heading, emphasis, and strong), and leave it to the browser to interpret the markups and render the document for display on the screen.

In other words, it allows the separation of content and presentation. The content provider focuses on the document contents, while the designer concentrates on the view and presentation. Now a days, CSS along with HTML is used for presentation.

Markup Indicator in SGML

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

Markup Indicator in SGML

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

Markup Indicator in SGML

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

The purpose of the markup language is to relieve the content provider from worrying about the actual appearance of the document. The author merely indicates via markup tags the semantic meaning of the words and sentences (such as paragraph, heading, emphasis, and strong), and leave it to the browser to interpret the markups and render the document for display on the screen.

Markup Indicator in SGML

For content and presentation Markup languages require markup indicator. The markup indicator in HTML is called tag. They are keywords often enclosed by angle brackets such as <p> (for paragraph), <img> (for image), <a> (for hyperlink).

HTML tags are used to perform the following operations

- Setting-up the layout the documents
- Proving links to other documents for navigation
- Embedding multimedia

The purpose of the markup language is to relieve the content provider from worrying about the actual appearance of the document. The author merely indicates via markup tags the semantic meaning of the words and sentences (such as paragraph, heading, emphasis, and strong), and leave it to the browser to interpret the markups and render the document for display on the screen.

In other words, it allows the separation of content and presentation. The content provider focuses on the document contents, while the designer concentrates on the view and presentation. Now a days, CSS along with HTML is used for presentation.

Separation of concerns

- Separation of concerns allows the document to be presented by different user agents according to their purposes and abilities.
- For example, a user agent can select an appropriate style sheet to present a document by displaying on a monitor, printing on paper, or to determine speech characteristics in an audio-only user agent. The structural and semantic functions of the markup remain identical in each case.

Separation of concerns

- Separation of concerns allows the document to be presented by different user agents according to their purposes and abilities.
- For example, a user agent can select an appropriate style sheet to present a document by displaying on a monitor, printing on paper, or to determine speech characteristics in an audio-only user agent. The structural and semantic functions of the markup remain identical in each case.
- Content vs. presentation vs. behavior
  - HTML is used to represent the structure or content of a document `<html>` ..... `</html>`
  - Presentation remains the sole responsibility of CSS. `<style>` ..... `</style>`
  - Behavior (interactivity) is handled by scripts. `<script>` ..... `</script>`

Structure of HTML5 document

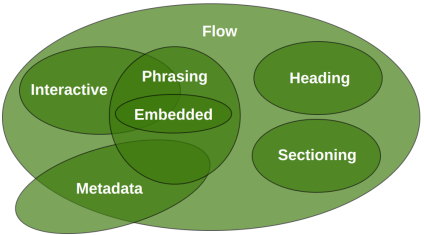
- `<!DOCTYPE html>`
  - SGML → A document type declaration to check what type of document it is.
  - HTML5 → A mechanism to stick with its standards. Its sole purpose is to prevent a browser from switching into `quirks mode` when rendering a document
  - `DOCTYPE` ensures that the browser makes a best-effort to follow the relevant specifications, rather than using a different rendering mode that is incompatible with some specifications.

Structure of HTML5 document

- `<!DOCTYPE html>`
  - SGML → A document type declaration to check what type of document it is.
  - HTML5 → A mechanism to stick with its standards. Its sole purpose is to prevent a browser from switching into `quirks mode` when rendering a document
  - `DOCTYPE` ensures that the browser makes a best-effort to follow the relevant specifications, rather than using a different rendering mode that is incompatible with some specifications.
- `Elements`: Component of an document, which can be nested and has a defined content model
  - `Content model`: A description of the element's expected contents.
  - Each element falls into zero/more categories that group elements with similar characteristics

Structure of HTML5 document

- `<!DOCTYPE html>`
  - SGML → A document type declaration to check what type of document it is.
  - HTML5 → A mechanism to stick with its standards. Its sole purpose is to prevent a browser from switching into `quirks mode` when rendering a document
  - `DOCTYPE` ensures that the browser makes a best-effort to follow the relevant specifications, rather than using a different rendering mode that is incompatible with some specifications.
- `Elements`: Component of an document, which can be nested and has a defined content model
  - `Content model`: A description of the element's expected contents.
  - Each element falls into zero/more categories that group elements with similar characteristics



Primary componant

Enjoying pleasant weather of Manipur

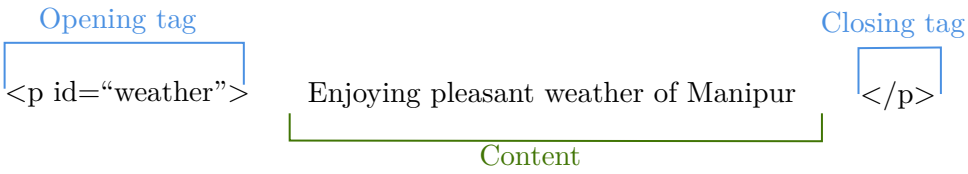
Primary componant

 Enjoying pleasant weather of Manipur

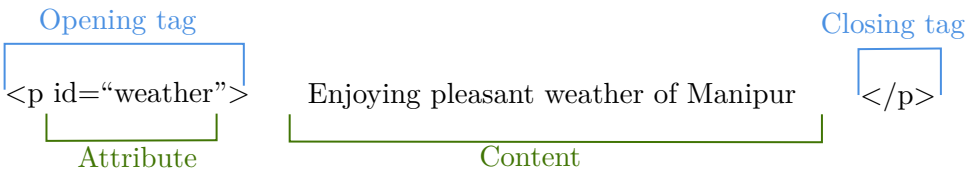
Primary componant

Primary componant

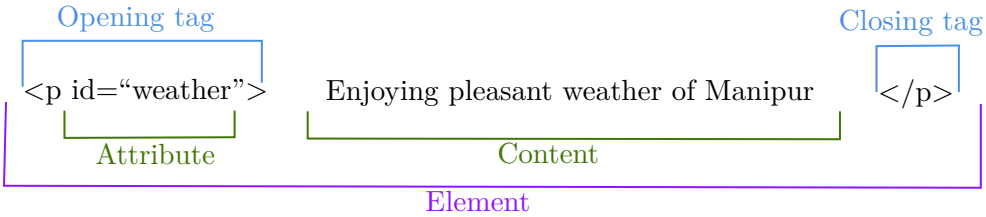
Primary componant



Primary componant



Primary componant



Content Categories I

- Metadata content
  - Deals with presentation, behaviour of the rest of the content, and relationship with other docs.
  - Example: `base, link, meta, noscript, script, style, template, title`
- Flow content
  - Encompasses most elements including heading, sectioning, phrasing, embedding, interactive, and form-related elements
  - Example: `a, abbr, kbd, nav, sub, sup, div, output`
- Sectioning content
  - Used to create section in the underlining document, defining the scope of `header`, `footer` and `body content`
  - Example: `article, aside, nav, section, footer, blockquote, fieldset`
- Heading content
  - Defines the heading of a section
  - Example: `h1, h2, h3, ..., h6`

Content Categories II

- Phrasing content
  - Associated with formatting of nothing or text. Primarily used to marked-up texts
  - Example: `abbr`, `label`, `span`, `strong`, `cite`, `code`, `data`, `ins`, `del`
- Embedded content
  - Used to imports another resource into the document. Elements that are from namespaces other than the HTML namespace
  - Emphasis is in content not in the metadata
  - Example: `audio`, `canvas`, `embed`, `img`, `math`, `video`
- Interactive content
  - Objective is to smoothen the user interaction
  - Example: `button`, `textarea`, `select`, `details`, `embed`, `iframe`

Secondary Content Categories

- `nothing` content model: Elements that are designed to have no content/children

```
<meta charset="UTF-8">
<br>
<hr>
```

Secondary Content Categories

- **nothing** content model: Elements that are designed to have no content/children

```
<meta charset="UTF-8">
<br>
<hr>
```

- **Transparent** content model: Refers to a set of rules that define how elements can inherit the content model of their parent elements. Elements with a transparent content model are placeholders for other elements, and they do not introduce a new nesting level in the HTML document.

```
<p>This is the <a href="#">link</a> to my homepage</p>
```

Secondary Content Categories

- `nothing` content model: Elements that are designed to have no content/children

```
<meta charset="UTF-8">
<br>
<hr>
```

- `Transparent` content model: Refers to a set of rules that define how elements can inherit the content model of their parent elements. Elements with a transparent content model are placeholders for other elements, and they do not introduce a new nesting level in the HTML document.

```
<p>This is the <a href="#">link</a> to my homepage</p>
```

- `Palpable` content model: Elements whose content are visible or palpable in the rendered document. Palpable content makes an element non-empty by providing non-empty descendant. Unlike `transparent` or `nothing` content models, the `palpable` content model involves elements that have a clear, tangible presence in the rendered document

Secondary Content Categories

- `nothing` content model: Elements that are designed to have no content/children

```
<meta charset="UTF-8">
<br>
<hr>
```

- `Transparent` content model: Refers to a set of rules that define how elements can inherit the content model of their parent elements. Elements with a transparent content model are placeholders for other elements, and they do not introduce a new nesting level in the HTML document.

```
<p>This is the <a href="#">link</a> to my homepage</p>
```

- `Palpable` content model: Elements whose content are visible or palpable in the rendered document. Palpable content makes an element non-empty by providing non-empty descendant. Unlike `transparent` or `nothing` content models, the `palpable` content model involves elements that have a clear, tangible presence in the rendered document
- Script-supporting elements: Elements which are used support scripts, either by containing or specifying script code directly, or by specifying data that will be used by scripts.
  - ✍ `<script>` and `<template>`

Element type

- Six types based on semantics
  - Void element→ `<area>`, `<base>`, `<br>`, `<col>`, `<embed>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`, `<source>`, `<track>`, `<wbr>`
  - Template elements → `<template>`
  - Raw text elements → `<script>`, `<style>`
  - Escapable raw text elements → `<textarea>`, `<title>`
  - Foreign elements→ Elements from the MathML namespace and the SVG namespace.
  - Normal elements→ All other html elements.

Element type

- Six types based on semantics
  - Void element→ `<area>`, `<base>`, `<br>`, `<col>`, `<embed>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`, `<source>`, `<track>`, `<wbr>`
  - Template elements → `<template>`
  - Raw text elements → `<script>`, `<style>`
  - Escapable raw text elements → `<textarea>`, `<title>`
  - Foreign elements→ Elements from the MathML namespace and the SVG namespace.
  - Normal elements→ All other html elements.
- Two types based on default display value
  - Block (level) element→ `<address>`, `<article>`, `<blockquote>`, `<canvas>`, `<div>`, `<footer>`, `<header>`, `<section>`, `<table>`, `<video>`
  - Inline elements → `<a>`, `<abbr>`, `<acronym>`, `<cite>`, `<dfn>`, `<em>`, `<span>` etc...

Element vs Tag

- What is tag?

Element vs Tag

- What is tag? Mark-up indicator used to separate content, presentation and behaviour.
- Tags are used to delimit the start and end of elements in the markup.

Element vs Tag

- What is tag? Mark-up indicator used to separate content, presentation and behaviour.
- Tags are used to delimit the start and end of elements in the markup.

- How element is different from tag?
  - Tags are building blocks of HTML Page that comprises of `starting angular bracket` character sequence `closing angular bracket`.
  - However, elements comprises of starting-tag, content and ending/closing-tag.
  - HTML elements can be nested, however tags are not.

Element vs Tag

- What is tag? Mark-up indicator used to separate content, presentation and behaviour.
- Tags are used to delimit the start and end of elements in the markup.

- How element is different from tag?
  - Tags are building blocks of HTML Page that comprises of `starting angular bracket` character sequence `closing angular bracket`.
  - However, elements comprises of starting-tag, content and ending/closing-tag.
  - HTML elements can be nested, however tags are not.

- Deprecated tags in HTML5: `<font>`, `<strike>`, `<applet>`, `<acronym>`, `<center>`, `<noframe>`, `<frame>`, `<frameset>`, `<u>`, `<tt>`, `< s>`, `<marquee>`

Attribute

- `Attribute`: Property of the element/tag. Additional information about an HTML element and is usually defined with the opening tag of the element
- Attributes are composed of a name and value pair, separated by an equals sign (=). The value is often enclosed in double or single quotes.
- Attributes have a name and a value. Attribute names must consist of one or more characters other than controls, U+0020 (space), U+0022 (”), U+0027 (’), U+003E (>), U+002F (/), U+003D (=), and noncharacters.
- An attributes can be applied to various elements, and different elements support different attributes.

```
<a href="https://www.abc.in" target="_self">Link to my homepage</a>
```

- Can be specified in four different ways:
  - Empty attribute <input disabled>
  - Unquoted attribute value <input value=yes>
  - Single-quoted attribute value <input type='checkbox'>
  - Double-quoted attribute value <input name="abc">

Type of attribute

- Boolean attribute
  - Attribute that does not require a value that are used to indicate a true or false condition. If a Boolean attribute is present, its value is considered to be true, and if it is absent, its value is considered to be false
  - Values of the Boolean attribute can either be omitted, set to an empty string, or be the name of the attribute. All values, including true, false, and 'xyz', while invalid, will resolve to true.
  - Example: `autofocus, inert, checked, disabled, required, reversed, allowfullscreen, default, loop, autoplay, controls, muted, readonly, multiple, and selected`
- Enumerated attribute
  - Attribute that can only take one of a predefined set of values.
  - The state for such an attribute is derived by combining the attribute's value, a set of keyword/state mappings given in the specification of each attribute
  - Example: `type` attribute of `<input>` element, `target` attribute of `<a>` element
- Global attribute
  - Attributes that can be set on any HTML element.
  - While these can all, in theory, be added to any HTML element, some global attributes have no effect when set on some elements; for example, setting hidden on a <meta> as meta content is not displayed

Global attribute

- Attributes that can be used on any HTML element regardless of type.
  - class → Specifies one or more class names for an element defined in CSS or JavaScript environment.
  - id → Unique identifier for an element within the document
  - style → Defines inline CSS styles for an element
  - title → Used to display additional information about an element lie tooltip
  - accesskey → Defines a keyboard shortcut to activate or focus on the element to improve accessibility
  - tabindex → Specifies the order in which an element should receive focus when navigating through the document using the keyboard
  - Example global attribute: autocapitalize, autofocus, contenteditable, dir, draggable, enterkeyhint, hidden, inert, inputmode, is, itemid, itemprop, itemref, itemscope, itemtype, lang, nonce, popover, spellcheck, translate

My First Program

- Open Gedit Text Editor using (terminal `$gedit` or `show applications` menu)
- Type the following lines in the Gedit Text Editor

```
<!DOCTYPE html>
<html>
   <head>
      <title>Homepage</title>
   </head>
   <body>
      <h1>Welcome to my Homepage</h1>
   </body>
</html>
```

- Save and run the `HTML` program
- Open in a Web Browser and see the output

Can we ommit tags?

- Certain tags can be omitted.
  - An html element's start tag may be omitted if the first thing inside the html element is not a comment.
  - An html element's end tag may be omitted if the html element is not immediately followed by a comment.
  - A head element's start tag may be omitted if the element is empty, or if the first thing inside the head element is an element.

```
<!DOCTYPE html>
<html>
<head>
      <title>Hello</title>
</head>
<body>
      <p>Welcome to this example.</p>
</body>
</html>
```

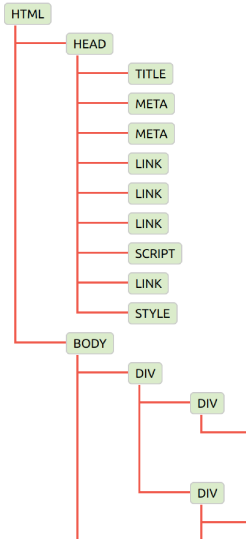
```
<!DOCTYPE html>
<title>Hello</title>
<p>Welcome to this example.</p>
```

Parsing an HTML document

- HTML Parsing is a mechanism to represent an HTML document to translator program (such as, web browser, artificial agent and other similar tools such as crawler, scraper) understandable format.
- It involves breaking down the raw HTML code into its constituent parts, such as element, tag, name-value pair for attribute, and content to build a hierarchical model of the document known as the Document Object Model and the structure is known as DOM tree.
- A DOM tree is an in-memory representation of an HTML document with node, property and relationship type.
- Tokenization and tree construction are the two stages of DOM Tree creation.
  - Tokenization → Mechanism to breaking down the HTML code into smaller, meaningful units called tokens such as start/end tag, name-value pair of attribute and content.
  - Tree construction → Involves mapping HTML elements into node of tree, and establishing hierarchy/parent-child relationship based on nesting pattern.

Parsing an HTML document

- HTML Parsing is a mechanism to represent an HTML document to translator program (such as, web browser, artificial agent and other similar tools such as crawler, scraper) understandable format.
- It involves breaking down the raw HTML code into its constituent parts, such as element, tag, name-value pair for attribute, and content to build a hierarchical model of the document known as the Document Object Model and the structure is known as DOM tree.
- A DOM tree is an in-memory representation of an HTML document with node, property and relationship type.
- Tokenization and tree construction are the two stages of DOM Tree creation.
  - Tokenization → Mechanism to breaking down the HTML code into smaller, meaningful units called tokens such as start/end tag, name-value pair of attribute and content.
  - Tree construction → Involves mapping HTML elements into node of tree, and establishing hierarchy/parent-child relationship based on nesting pattern.



Parse errors

- Parse errors are only errors with the syntax of HTML. In addition to checking for parse errors, conformance checkers will also verify that the document obeys all the other conformance requirements described in the specification.
  - duplicate-attribute → If the parser encounters an attribute in a tag that already has an attribute with the same name
  - eof-in-tag → This error occurs if the parser encounters the end of the input stream in a start tag or an end tag. For example: `<div id=`). Such a tag is ignored.
  - end-tag-with-attributes → This error occurs if the parser encounters an end tag with attributes. Attributes in end tags are ignored and do not make their way into the DOM.
  - missing-attribute-value → This error occurs if the parser encounters a U+003E (>) code point where an attribute value is expected (e.g., `<div id=>`). The parser treats the attribute as having an empty value.
  - unexpected-character-in-attribute-name → This error occurs if the parser encounters a U+0022 (”), U+0027 (’), or U+003C (<) code point in an attribute name. For example: `<div abc <div>` or `<div id'abc'>`