# Chapter 1 – Basics

## Doctype

* It instructs the browser which type of document is ours. Eg. HTML4, HTML5, XHTML etc.
* It is case-insensitive.
* If the doctype is present, the browser will render the page in standard mode.
* If absent, the browser will render the page in quirks mode.
* Eg. <!DOCTYPE html> - is for standard HTML5 page.
* Quirks mode is browser specific mode, meaning it will be different for all the browsers.
* Doctype also ensures compatibility across browsers.

## Head section tags

* <title>, <meta>, <link>, <script>, <noscript>, <style>, <base> etc..
* <head> contains machine-readable information (metadata) and not human readable information.
* Minimum tags to be present in <head> are:
* <meta charset="utf-8">
* <meta name="viewport" content="width=device-width, initial-scale=1">
* *<!--*
* *The above 2 meta tags \*must\* come first in the <head>*
* *to consistently ensure proper document rendering.*
* *Any other head element should come \*after\* these tags.*
* *-->*
* <title>Page Title</title>

## Body section tags

* <a>, <div>, <p>, <header>, <nav>, <h1> to <h6>, <image> etc..
* It contains the main content of the page.
* There can be only one body element in the document.
* Attributes of the tags are the properties written within the tag.
* An empty tag is the tag which does not have a closing tag. Eg. <link rel=”stylesheet” href=”/styles.css” type=”text/css”/>
* Inline elements are those elements which only occupy the space bounded by the tag.
* Block level elements are those which occupy entire width of the browser. They always start with a line break.

## How browsers read HTML

* Browsers store the HTML page in DOM tree format in its memory.
* DOM tree is an object-oriented representation of the document and it is stored as nodes and objects.
* Each HTML tag is an object in DOM.
* DOM tree can be modified using JavaScript.

## Ways to include CSS

* Internal CSS – which is written using the <style> tag
* External CSS – which is written in a separate file and linked using <link> tag in head section.
* Inline CSS – which is written within the tag using “style” attribute.

## Where should JS be included and why?

* JS should be placed at bottom of the document.
* Browser will load and parse the script before proceeding with rendering the page. User will be kept on waiting.
* We can also use async and differ to load the script asynchronously without blocking the browser.

## Why is external CSS better than internal?

* Readability
* Reusability
* Size of the page increases in case of internal CSS
* Caching – Browsers cache static resources such as CSS, JS, images etc..

## Online practice editors

* JSFiddle - <http://jsfiddle.net/>
* Codepen - <https://codepen.io/>
* JSBin - <https://jsbin.com/>

## Editors to write code

* Atom
* VS Code
* Sublime
* Notepad++

## Developer’s Toolbar

* Elements – Displays the HTML of the page
* Console – Displays any errors and log messages. Allows us to write JavaScript also.
* Sources – All the static files present in the production environment. It is used to debug JavaScript.
* Network – Time taken to load each resource. It shows the request and response time for each resource. This panel is important for site optimization.

## Question and Answers

1. Can we use <style> tag inside of <body>?
2. We can include <style> tag inside <head> or <body>. But it is recommended to include the <style> tag inside <head> for organizational purposes. Its better to separate out presentation and content of your document. The best way would be to include an external stylesheet.
3. In Trello, after we modify, delete or add any card, the information is persisted how?
4. Trello might be making AJAX (Asynchronous JavaScript) requests to save the changes. These kind of requests do not block the users and refreshes only a particular section of the page.

# Chapter 2 - CSS fundamentals

## CSS Selectors

* **Universal Selector** - *\*, ns|\*, \*|\**

It matched all the elements of the document

* **Type Selector** – *elementname*

It matches all the elements of the document with the given node name.

* **Class Selector** - *.classname*

Selects all the elements of the document with the given class name.

* **ID Selector** - *#idname*

Selects all the elements of the document with the given ID name.

* **Attribute Selector** – [attr], [attr=value], [attr~=value], [attr|=value], [attr^=value], [attr$=value], [attr\*=value]

Example [type] – it will select all the elements with the “type” attribute

which is set to any value.

## Grouping Selectors

* “,” will select all the matching nodes.

Eg. – div, span will select all the div and span elements of the document.

* “ “ (space) descendants combinator selects nodes which are descendants of the first element.

Eg. – div span will select all span elements that are inside a div element.

* “>” child combinator selects the direct children of the first element.

Eg. – ul > li will select all the <li> elements that are direct children of <ul>

* “+” adjacent sibling combinator will select adjacent siblings of the first element.

Eg. – h2 + p will select all <p> elements which are adjacent to all h2.

## Pseudo

* Pseudo Classes – : pseudo allows the selection of elements based on a state which is not defined in DOM

Eg. - a:visited

* Pseudo Elements - :: pseudo represent entities that are not included in HTML.

Eg. – p::first-line

## Cascade

* A rule or ruleset in CSS means the entire block.

.example(selector) {

Color: red; (declaration)

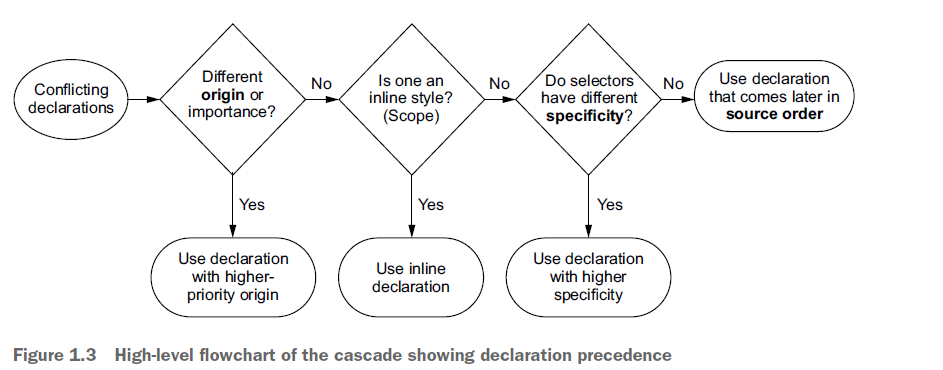
Font-family: sans-serif; (declaration) \*\*

}

\*\* - Yellow portion is called as declaration block

* Predicting how rule behaves requires an understanding of a cascade.
* When two or more rules have conflicting declarations, browser follows a set of rules to predict the result. These set of rules is called as Cascade.
* It tells us how conflicts are resolved.
* Cascade has the following 3 things to resolve a conflict.

1. **Stylesheet origin** – Where the styles come from. It is custom styles or the browser specific styles (user-agent stylesheet)
2. **Selector specificity** – Which selectors have precedence over which
3. **Source order** – Order in which the styles are declared in the stylesheet



## Understanding Stylesheet Origin

* Where the styles are coming from.
* User-agent styles are browser default styles for an element.
* Author styles are our styles.
* Author styles are given higher precedence over user-agent styles.
* Author important styles (!important) is given highest precedence.

1. Author important styles
2. Author styles
3. User-agent styles

## Understanding Specificity

* If the conflicting declarations cannot be resolved using origin, browser considers the specificity.

1. Styles applied inline
2. Styles applied using a selector

* Inline styles are applied using <style> attribute for the HTML element.
* Inline styles override any declarations present in your stylesheet or <style> tag.
* Inline styles are also called as “scoped” declarations.
* Selector specificity is determined by following rules:

1. If the selector has more IDs, it wins
2. If the results end in a tie, the selector with most classes wins
3. If the results end in a tie, the selector with most tags (type selector) wins

* Pseudo class selectors and attribute selectors have same precedence like a class selector.
* Universal selector and combinators (>,+,~) has no effect on specificity.
* Notation of specificity – (IDs, classes, tags)
* Three ways you can fix the specificity

1. Using !important
2. Increasing the specificity for the styles you want apply
3. Decreasing the specificity for the other selectors

* Try to avoid !important as much as possible.
* Try to keep the selector specificity as low as possible so that when you want to override, it becomes simple.

## Understanding Source Order

* When the selectors have same specificity, then it is resolved by the source order.
* The styles at the end wins in this case. OR when the stylesheet is linked at later parts of your document.
* When styling problem comes, consider these:

1. Think about which declarations will make a fix
2. Structure the selectors and choose the best which fits the needs

* Link styles should be in certain order

a:link

a:visited

a:hover

a:active

* The declaration that wins the cascade is called as cascaded value. It is the value given to the property applied to an element as a result of the cascade.

# Chapter 3 – Git Fundamentals and Hands-on

## Git fundamentals

* Git is a distributed version control software.
* This means many developers can easily work on a particular feature.
* Branches are light weight and hence can be created many.
* Branches are for features and not for people.
* There is a repository which contains all your branches. When you create a new repository, it automatically creates a new branch by name “main”.
* “main” branch is often called as master and it holds the production code.
* All the other branches are replica of the “main” branch.

## Git Commands

1. git clone – To clone entire repository
2. git branch – To check the current branch you are in
3. git branch -a – To check all the branches in local and remote
4. git checkout -b <branchname> - To create a new branch and pointing to it
5. git checkout <branchname> - To switch the branch to other
6. git status – To check all the changes made
7. git add – Adds the new or modified files in the staging area
8. git commit -m “meaningful message” – Commits the changes to locally running Git.
9. git push origin development – Pushes the changes from local to remote branch.
10. git log – To new all the commits happened