|  |  |  |
| --- | --- | --- |
| **Shortcut for** | **Eclipse** | **Intellij** |
| Open resource | Ctrl+shift+R | Ctrl+Shift+N (Navigate to File) |
|  | Ctrl+Shift+T | Ctrl+N |
| Maximize the active window | Ctrl+M | Ctrl+Shift+F12 |
| Quick outline | Ctrl+O | Ctrl+F12 |
| Move line | Alt+Up/Down | Shift+Alt+Up/Down |
| Delete line | Ctrl+D | Ctrl+Y |
| Copy line | Ctrl+Alt+Down | Ctrl+D |
| Quick switch to editor | Ctrl+E | Ctrl+Tab |
| Hierarchy view | Ctrl+T | Ctrl+H (Class Hierarchy) |
| Ctrl+shift+H (Method Hierarchy) |
| Last edit location | Ctrl+Q | Ctrl+shift+Backspace |
| Correct indentation | Ctrl+I | Ctrl+Alt+I |
| Format code | Ctrl+Shift+F | Ctrl+Alt+L |
| Open declaration | F3 | Ctrl+B |
| Ctrl+shift+I (Quick definition) |
| References in workspace | Ctrl+Shift+G | Alt + F7(Find Usages) |
| Ctrl+Alt+F7 (Show Usages) |
| Open search dialog | Ctrl+H | Ctrl+Shift+F |
| Navigate to line | Ctrl+L | Ctrl+G |
| Rename a variable | Ctrl+Alt+R | Shift+F6 |
| Optimize imports | Ctrl+Shift+O | Ctrl+Alt+O |
| To compile the class |  | Ctrl+F9 |
| Code completion | Ctrl+Space | Ctrl+Space (basic completion) |
| Ctrl+shift+space (Smart completion) |
| Ctrl+shift+Enter (Statement Completion) |
| Quick fix | Ctrl+1 | Alt+Enter |
| Type hierarchy | F4 | Ctrl+H |
| Close Current Editor | Ctrl+W | Ctrl+F4 |
| Search text in file | Ctrl+K | Shift+F3 |
| Extend selection |  | Ctrl+W |

Ways to create Java8 Streams

* /\* Empty Stream \*/

Stream emptyStream = Stream.empty();

* /\* Of method \*/

Stream way1=Stream.*of*(**"Apple"**,**"Orange"**,**"Banana"**,**"Grapes"**);

* /\* From Collections \*/

List<String> collection = new ArrayList<>();

Stream streamFromCollection = collection.stream();

* */\* From Arrays \*/* String[] arrayFruits = {**"Apple"**,**"Orange"**,**"Banana"**,**"Grapes"**};  
   Stream streamArray = Arrays.*stream*(arrayFruits);
* /\* From Generate method \*/

Stream<String> streamGenerate = Stream.generate(()->"test").limit(10);

streamGenerate.forEach(System.out::println);

* /\* from iterate method \*/

Stream streamIterate = Stream.iterate(BigInteger.ZERO,n->n.add(BigInteger.ONE)).limit(10);  
streamIterate.forEach(System.out::println);

/\* from iterate method \*/

Stream streamIterate = Stream.iterate(BigInteger.ZERO,n->n.add(BigInteger.ONE)).limit(10); streamIterate.forEach(System.out::println);

Techinical words that reason for Micro-services

Domain-driver design, Continuous delivery, On-demand virtualization, Infrastructure Automation, Small Autonomous teams, Systems at Scale Key words for Microservices.

What are Micro-services:

Micro-Services are small autonomous services are work together.

Cohesion is the important word in Micro-Services world. Code grouping is important function when we consider the Micro-Services.

This is reinforced by Robert C. Martin’s definition of the Single

Responsibility Principle, Maintaining this wil help us to find the where functionality have to added and it will help the code bases are smaller and maintainable.

Often Micro-services are deployed as an Isolated services on a Platform as a Service (Paas). Sometimes it could be overhead but in future overhead pblms are mitigated.

The services should be changed independently and to be deployed without requiring customer change.

When too much sharing happened, every time we depends on customer change. It makes autonomy as more coupled with internal systems.

Without decoupling, everything discussed in this book very hard to achieve. Golden rule, Can you make a change to a service and deploy it by itself with out changing anything else. It the answer is no then have to think your application and try to achieve this.

Key Benefits:

**Technology Heterogeneity:** You can choose best fit for the services instead of having one-size-fits-all approach.

Ex: social network user interactions could be stored on Graph DB and Images/Pictures are stored on BLOB DB and posts are Document store.

In a monolithic system,changing one technology impacts all parts of the system.

**Resilience:**

If one component fails, the other components in the systems are still working. We can notify other components and degrade the functionality of the application.

**Scaling:**

Consider in our application, one component is constrained the performance, we have to scale entire components as a piece. If its micro services, major performance consumer in powerful hardware system and others in less.

**Ease of Deployment:**

One line change requires a entire application to be deployed in the monolithic. Its high risk.

With Micro-services, you can deploy a service indepently from each other components. If a pblm occurs its easier to isolate the service and rollback.

**Organizational alignment:**

Smaller teams working on smaller code bases are always productive.

**Composability:**

Major advantage of using micro-services are reuse of functionality. In Microservices, we should think and design our functionality consumed by different ways for different purposes.

**Optimizing for Replaceability:**

You get a chance to see legacy systems no one wants to touch. Why it has not been replaced? Its risky and too big job. With micro services, replacing the code with better approach and delete them altogether is easy.

Consider in a legacy can you delete 100 lines in a day and not worried too much. This can be possible in micro services.

With micro-services, rewriting and removing code is less risky in application.

**What about Service Oriented Architecture:**

Is a design approach where multiple services colloborately provide the end set of capabilities. Service calls are here happening through network instead of method calls in single boundary.

SOA emerged to combat the challenges of legacy monolithic applications.

SOA, at its heart having very sensible idea. People failing to understand the problem and provide the compelling alternative with narrow set of design principles. Due to this missing service granularity, opting for wrong software vendors and wrong place to split the system.

Many SOA pitfalls are happening on many unsaid things, Micro services are emerged from real world use and do SOA well.

Decompositional Techniques:

**Shared Libraries:**

Standard way breaking down the code base into libraries. This libraries are provided by third party vendors or some team in your organization.

Drawback:

Lose Technology Heterogeneity: Typically in your own language to understand your code.

2. unless you are using dynamically linked libraries, you have to deploy entire process to deploy a new library. What safety measures are with libraries to resilient

**Modules:**

Some language provide modular decomposition. You don’t need to dependent on any other modules and you can deploy any changes on your module with out affecting others.

**The Architect:**

Micro services gives lot of choices and lot of decisions. Ex: how many different technologies to use,different teams for different programing, should we split or merge a service ,

Creep in - gradualy start noticable

To work with Functional Style We just follow below

1. Be Declarative

2. Promote Immutability

3. Avoid Side Effects

4. Prefer Expressions over statements

5. Design with High order Functions

Declarative:

Ex: contains() method in collections is declarative style.

actual implementation of if.. else is imperative style

Promote Immutability:

When object changes its nature hard to understand the object

1. Declare fields and objects as final

2. prefer String class objects

3. create immutable collections such as Arrays.asList / unmodifiableList

Avoid Side effects:

consider fetching stack price and update the shared variable, consider many prices have to fetch

in that case , we have to avoid race condition & have to avoid threading synchronization issues

better to avoid such side effects by giving immutable state.

Prefer Expressions over statements:

Statements are stubborn and force mutability and expressions promote immutability

we used for statement to compute the discounted prices, this version of code promote the mutable

on variables and other objects.

when we move the code to map and sum functions we avoided mutation and able to chain the code

Design High Order functions:-

Java is not pure functional style language. Java allows to modify the variable at will.

Instead of relying on objects and classes for reuse we can rely on simple small focused and cohesive well written functions

In Object Oriented style,

1. create object in methods

2. pass objects to methods

3. return objects from methods

With high order functions we can do

1. pass functions to functions

2. create functions with in functions

3. return functions from functions

Example passing function to function

prices.stream()

.filter(price -> price.compareTo(BigDecimal.valueOf(20)) > 0)

.map(price -> price.multiply(BigDecimal.valueOf(0.9)))

.reduce(BigDecimal.ZERO, BigDecimal::add);

price -> price.multiply(BigDecimal.valueOf(0.9), the function created at the time of executing function map

Generally a function has name, parameter list, body and return type.

A function with out a is Anonymous function or lambda expressions

git config --global user.name "Your Name"

git config --global user.email "your.email@your-place.com"

Open or create the ~/.bash\_profile file and add the following line:

alias npp='notepad++ -multiInst -nosession'

Open Git Bash and issue the command:

git config core.editor "notepad++ -multiInst -nosession"

Then test it out by:

git config --global -e

git config --global --list

Install p4merge from perforce.com

Configure P4Merge as Diff Tool in Git:

git config --global diff.tool p4merge

git config --global difftool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"

git config --global difftool.prompt false

Configure P4Merge as Merge Tool in Git:

git config --global merge.tool p4merge

git config --global mergetool.p4merge.path "C:/Program Files/Perforce/p4merge.exe"

git config --global mergetool.prompt false

TO create a repository in local:-

git init demo

\* inside .git file will be there

Git States:-

Working Directory:- contains all files, not managed by git. bt git aware

staging area:- files are moved to working directory to staging area before commit

Git Repository:- Part of git history. Commited files are moved to here

Basic Git commands:-

1. git status

2. Git add file.txt

3. git commit -m "Hello"

4. git init . (TO initialize the git repos)

5. git add . (TO add all changed files are staged to commit)

6. git log -> to show all the git commits part of this repository

7. git show -> (displays the last commit and the changes)

8. git commit -a -> add the modified files to staging area and do the commit

Ex:- git commit -am "Commiting Modified files"

Changing the staged file to last known state:-

git reset HEAD <filename>

ex:- git reset HEAD readme.md --> this will unstage the file from staging area

to Reset the contents of the file to previous state:-

git checkout -- <filename>

ex:- git checkout -- readme.md

Renaming the GIt file:-

git mv <oldfile> <newfile>

ex:- git mv example.txt demo.txt

git rm <file>

ex:- git rm demo.txt

When we delete file using normal doss command Please do the following:-

1. git add -A (git add -u)

Types of Merge:

1. Fast-Forward

2. AutoMatic

3. Manual

4.

Fast-forward Merge:-

directly applying the changes of branch into Master branch.

No additional work to git on this merge

This can be disabled

Like no other branch exists and no modification on master

AutoMatic:-

Non conflicting changes on parent

Manual:-

have conflict with same files

Speacial Marker:- HEAD , points to the last commit of the current branch, This pointer can be moved

To checkout to a new branch:-

git branch -a -> to show the current branch

1. git checkout -b updates

2. once u done with commit, move to master branch by git checkout master

3. now use git merge updates, this apply updates branch changes into master branch

4. to remove the branch use git branch -d updates

Creating logs:-

\*. git log --oneline --graph --decorate --all

Tags:-

\*. git tag mytag (This will create the tag)

\*. git tag -d mytag (This will remove the tag)

use

git tag -a v1.0 -m "Release v1.0"

This will shows major milestone on git commit...

Git Stashing:-

1. git stash - Will clear the working directory

2. git stash list

3. git stash pop

once git stash pop is applied it will apply the last stashed items into the branch and drop the stash

git reset/ git reflog commands:-

Three types of Reset:-

1. soft

2. hard

3. Mixed

Ex: git reset <commitId> --soft

This will unstage the files from commit

Ex:- git reset <commitId> --mixed

similar to --soft

Ex:- git reset <commitId> --hard

this will remove the changes in the commit and working directory clean

Linking Local GIT Repository to GIT hub repository:-

1. git remote -v

2. git remote add <referencename> <remotegiturl>

Ex:- git remote add origin https://github.com/prabaharan/personal.git

origin is the conventional name given to the git remote url.

you can use any name for remote url...

In my local:-

1. git pull origin remote --allow-unrelated-directories

2. git push -u origin master

-u -> setup the tracking relation ship with local master and remote master

Why SSH is better than Https:- Saves Time!

Every time https connection requires username and password credentials

in SSH, once established you dont require to pass credentials everytime but once in setup.

to generate the SSH:-

in git bash type

ssh-keygen -t rsa -C "prababuddy@gmail.com"

.ssh directory needs to be created under user home directory.

once command executed successfully, it will create id\_rsa file and id\_rsa.pub file in .ssh folder

To Authenticate the local address,

ssh -T git@github.com

git clone https://<URI> -- (this will clone the repo to your local)

git clone https://<URI> <LOCALREPONAME> -- same as above but will create the folder name given

difference between fetch and pull:-

fetch is non destructive

pull is destructive. do two task atonce, 1. pull the latest changes 2. merge the changes into local (automatically merge)

Consider your GITHub is changed to my-website to website

How u update ur local repos to the change

git remote -v

display current unchange my-website url

git remote set-url origin <url til .git>

now git remote -v

To merge the other branch use

git merge otherbranchname

This will merge the otherbranch with ur local branch and removes the other branch

TO delete the local branch created locally, you can do this by

git branch -d branchname

To remove the dead references

git fetch -p

-- -p Look for any dead branches

git fetch look up any new branches created

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

create a local branch

git checkout -b newbranchname

to push the local repository

git push -u origin newbranchname

-u creates the tracking relationship with origin and ur branch

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To remove the remote branch

1. git pull --all

2. git merge <branchname>

3. git push origin: <branchname> -- this will delete the branch in remote and in local

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

git pull --rebase

1. rewind the local branch and pull the changes from git hub

2. apply ur local commits on top of it

Git Hub comparing includes

1. branches

2. Commits

3. Tags

4.

Compare changes based on date ranges

master@{3days}

master@{2018-06-06}

HEAD@{3} -- THis means 3rd highest commit in the branch

HEAD^

Two ways to refer the commits:

1. Absolute (Unique Identifier on every git commit (sha1 identifier)

You can navigate the commits by

Git checkout <sha1>

1. Relative (something through pointer )

The commit before the current one is referred by HEAD~1, HEAD~2

To move the last commit before the current one is

Git checkout HEAD~1

To back to our latest commit , git checkout <branchname>

Opposite of ~ is ^. ~ referred to child commits ^ referred to parent commit..

AngularJs-6:

Angular is a JavaScript framework which allows you to create reactive Single Page Applications.

One single HTML page, loads all the dynamic contents in it…

1. This is faster, Javascript retrieves the pages u want it from the server and loads them on the HTML page.
2. No refresh button on the browser. Every content is dynamically loaded.
3. If you see the page source, nothing in it.

Angular6, Angular5, Angular4, Angular2 and Angular1:-

Angular 6,5,4,2 referred as Angular frameworks

Angular1 is referred as AngularJs framework.

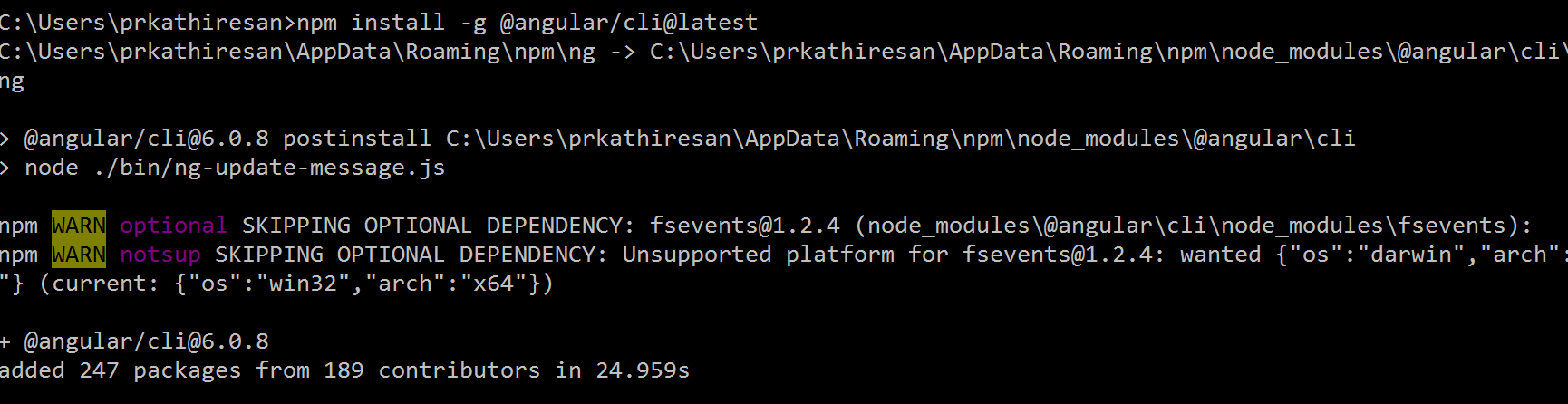
Angular2 complete rewrite of angular1.

Angular frameworks (2,4,5,6) have base on Angular2 and have some increamental features are added.

Angular 1 & Angular 2 is totally different.

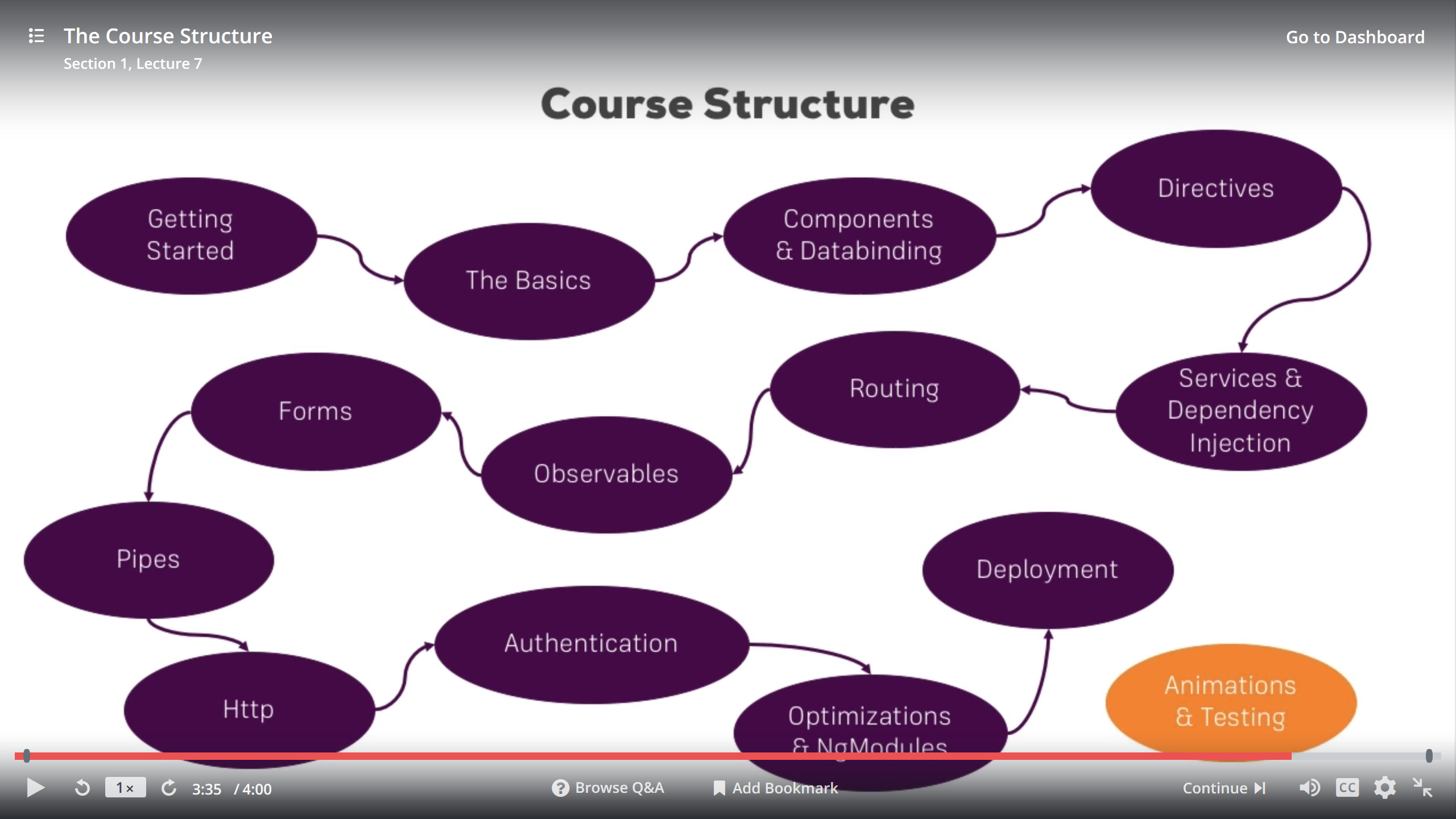
Things that to start:-

1. Download Node Js from nodeJs site. Install it in your machine
2. Verify nodejs by node -v command on your command prompt
3. Now, npm install -g @angular/cli@latest



1. ng new myfirstapp
2. ng serve

Angular features:-



What is Type Script ?

Super Set of Java Script offers more features like (Types, Classes , Interfaces ). More than Vannila Js

TypeScript can not be executed in browser directly.

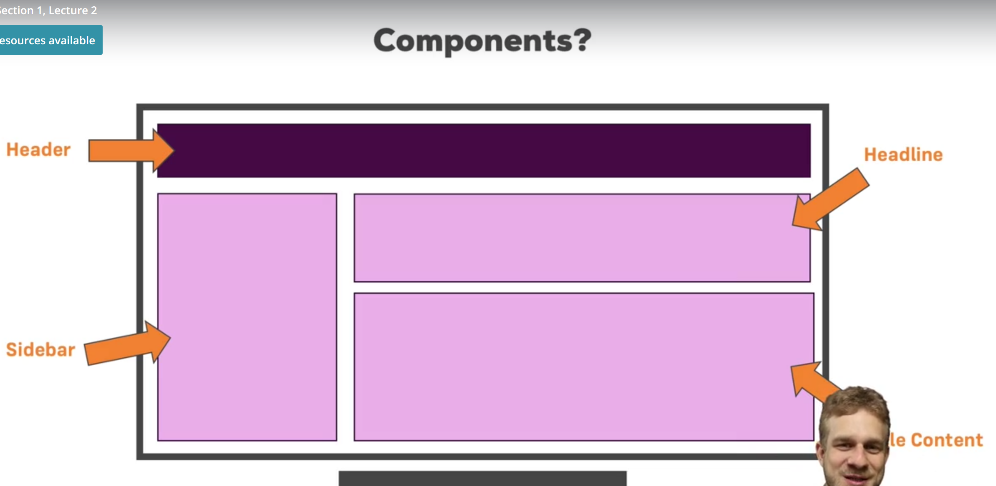
TypeScript 🡪 Java Script (conversion will be faster)

**React JS:-**

A java Script library for creating User interfaces…

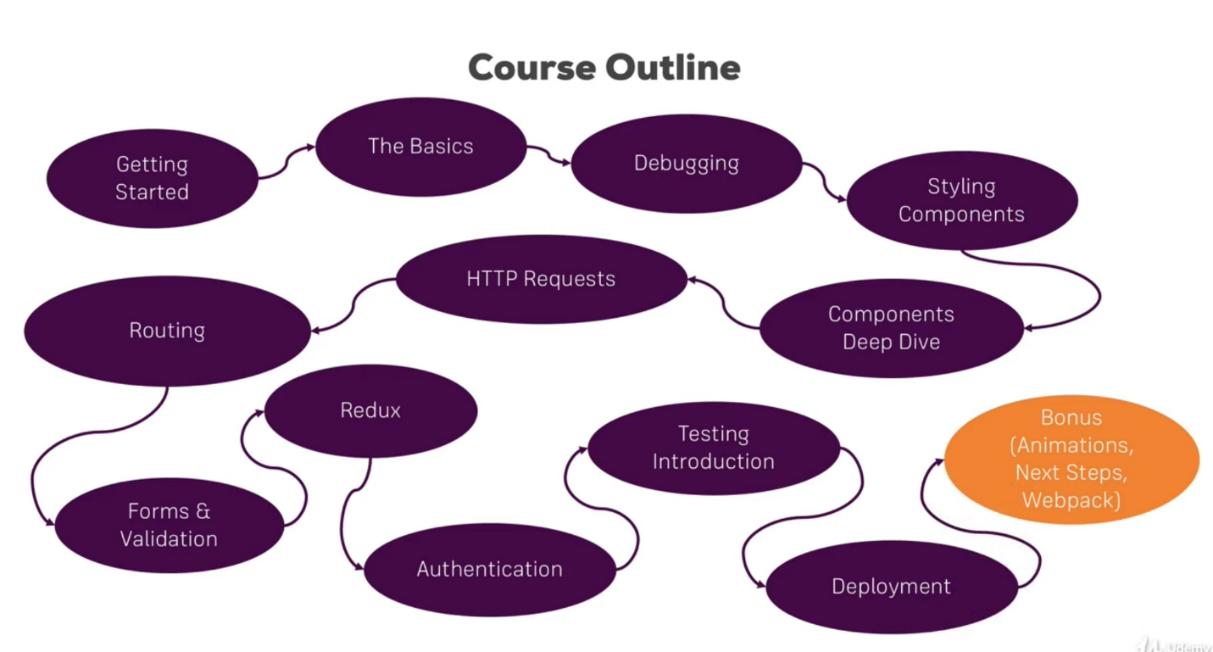
(User Interfaces are build upon Components)

What is components?



Header Component, Headling Component, Sidebar Component, Article component

Course Outline:-



Next generation JS functions are Arrow functions and classes.

<Quora>

ES6 is basically EcmaScript 6 / EcmaScript 2015. EcmaScript is not any scripting language instead a standard that Javascript is based upon. So, ES6 is a new version or new standard of Javascript. ES6 brings many new feature like concept of classes, template tags, arrow functions etc.

Almost all the modern browsers support ES6 but for the old browsers there are many transpilers e.g. Babel.js those we need to include at top of our code to transpile ES6 to ES5 (Javascript with old standards). All of the popular javascript libraries and frameworks like Node.js, ReactJS follow ES6.

</Quora>

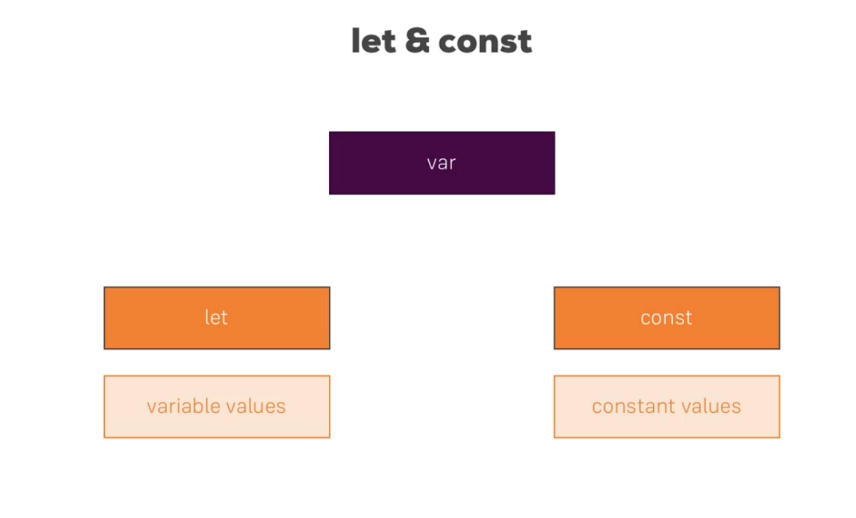
<Prabaharan>

ES6 is a new Java Script Standard, all the browsers started adopting the ES6 Java Script. Those who have not adopted to ES6 have to use transpilers like Babel.js.

</Prabaharan>

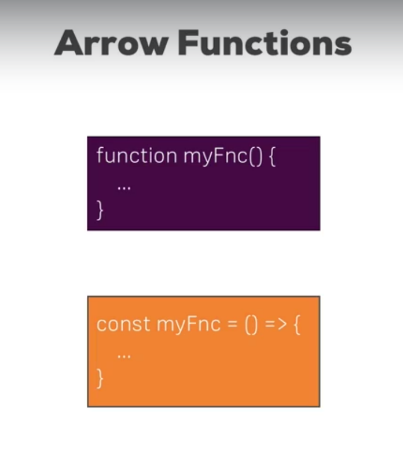
**Next Generation Java Script:-**

LET & CONST:-



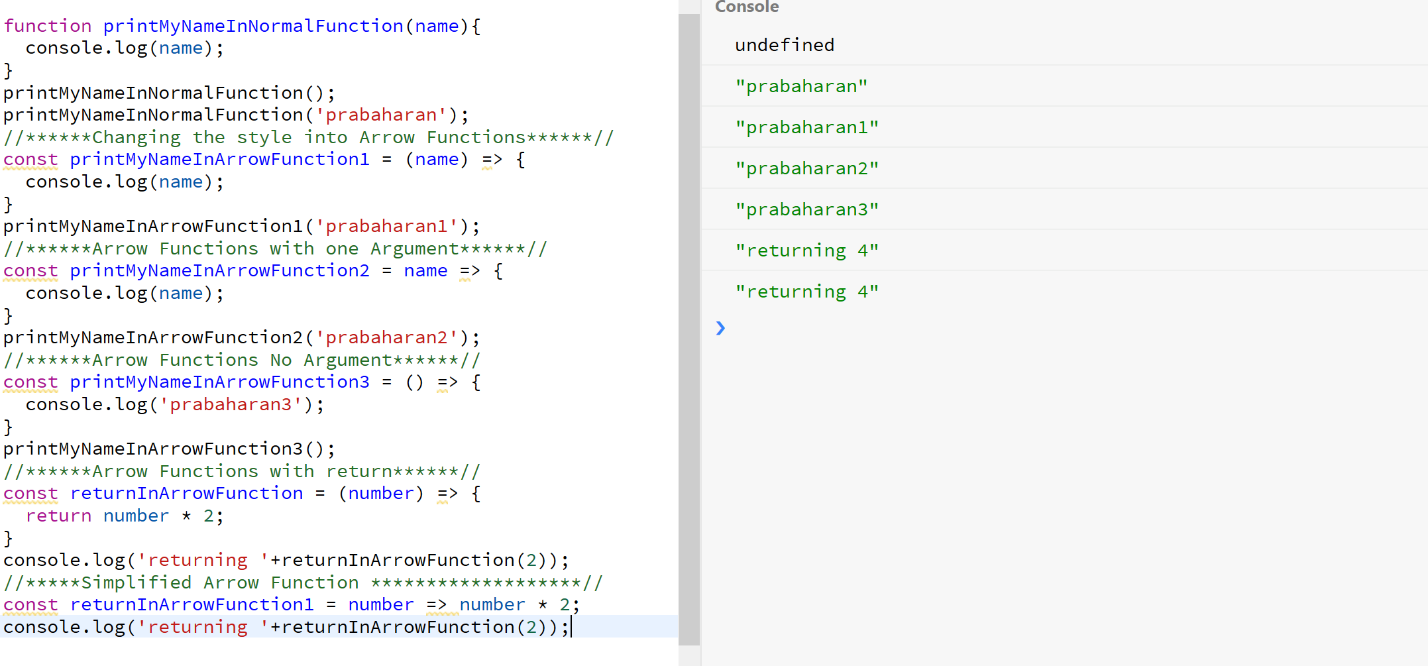
ARROW Functions:-

Syntax:-

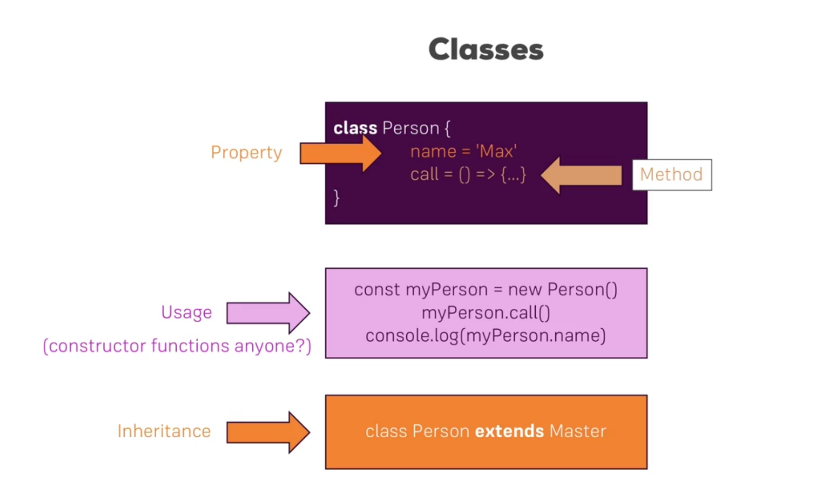


In Arrow Functions, no more issues with this keyword(this keyword have misunderstood in old).

Samples in Jsbin.com



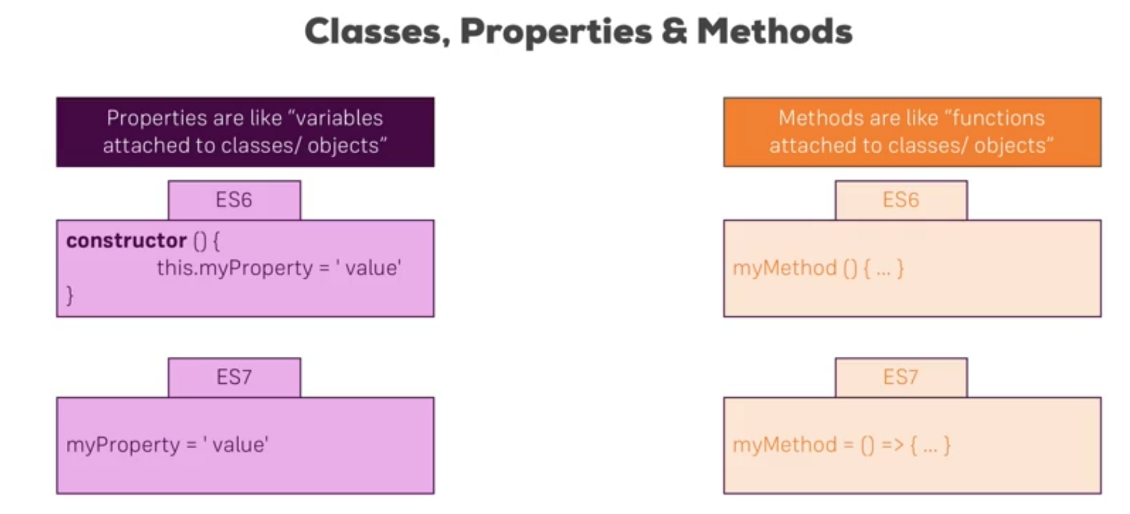
Classes:-



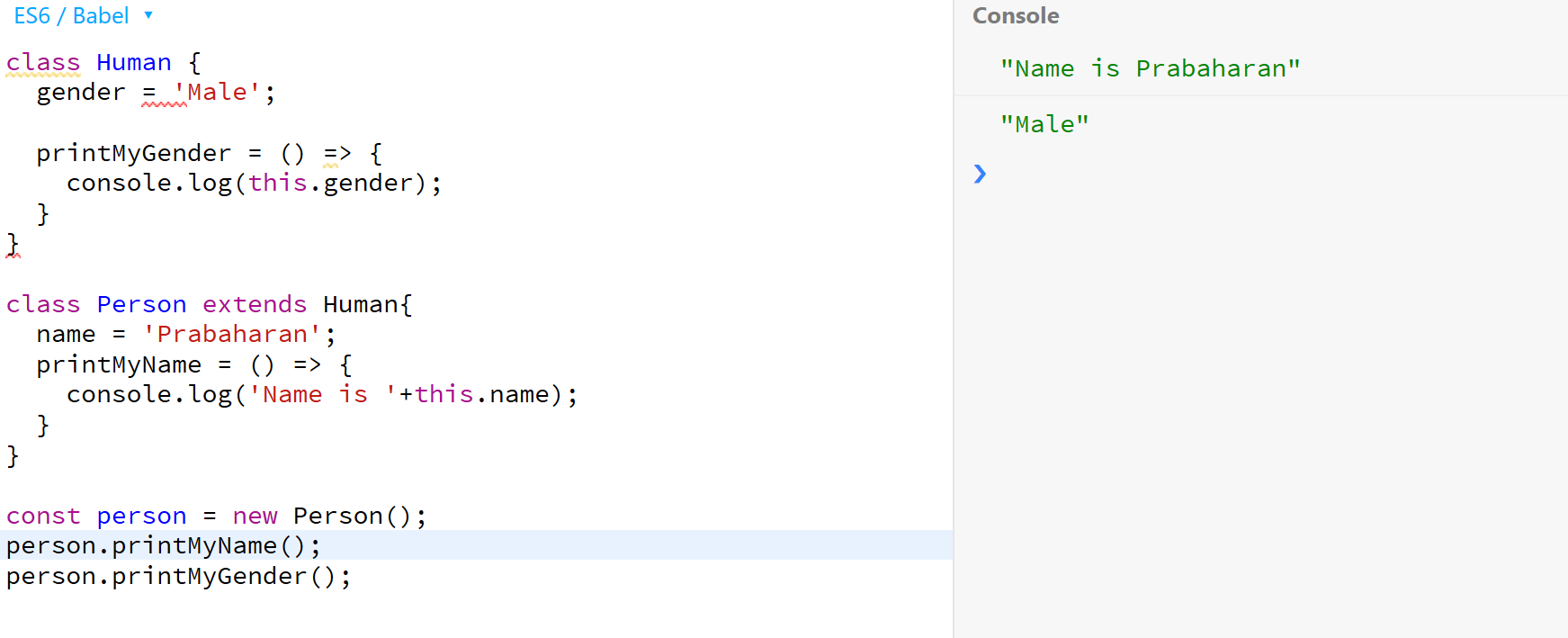
Classess in Action:-



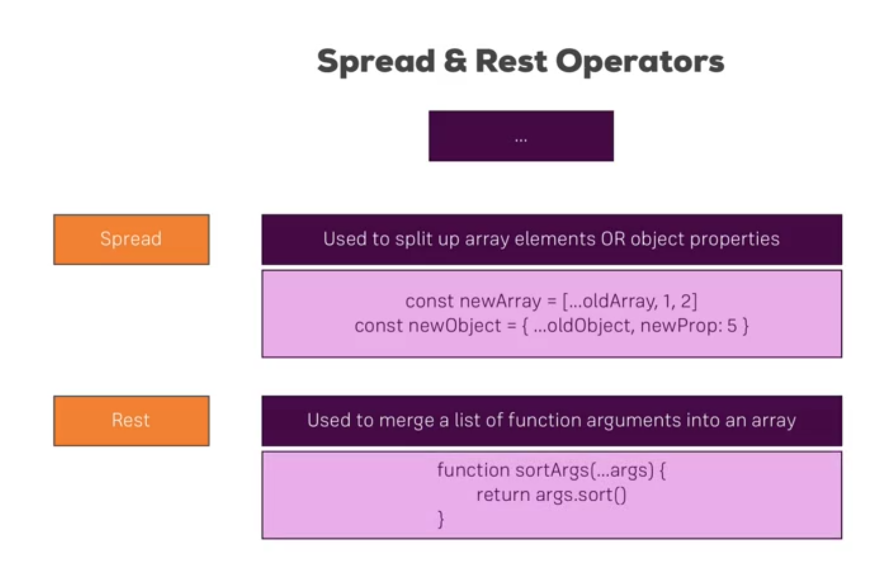
Classes, Properties and Methods in ES7:-



Still shows error on Java Script:- but can run on ES6/Babel

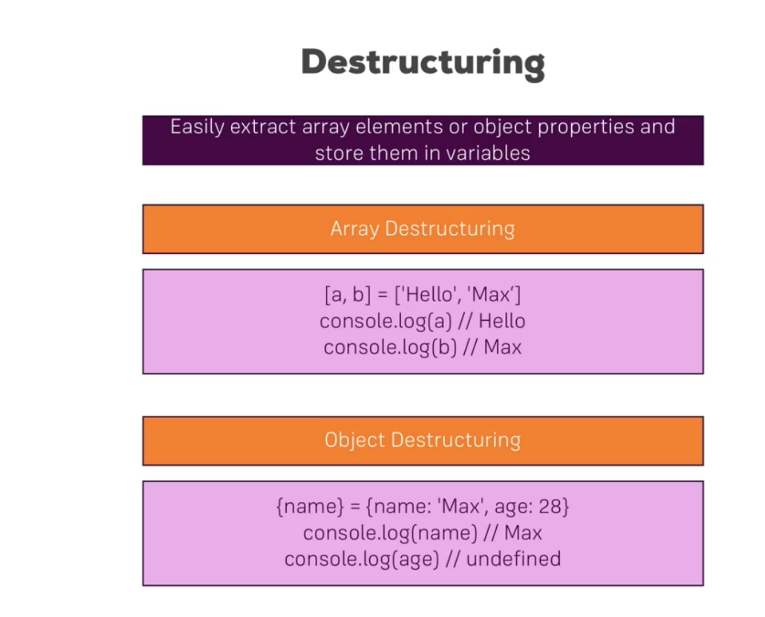


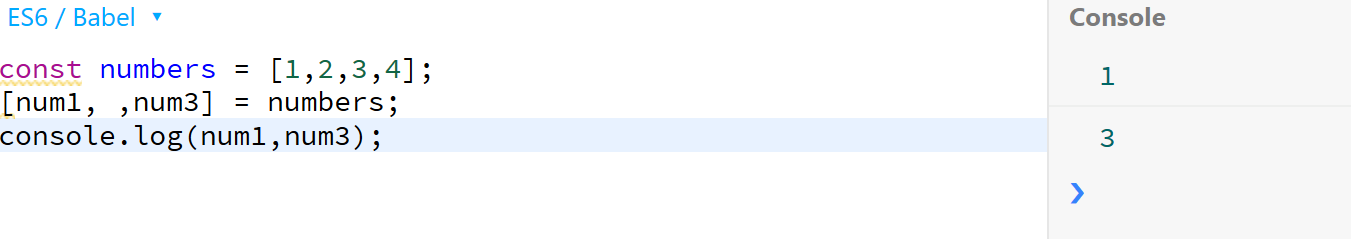
Spread & Rest Operators:-





DeStructuring:-





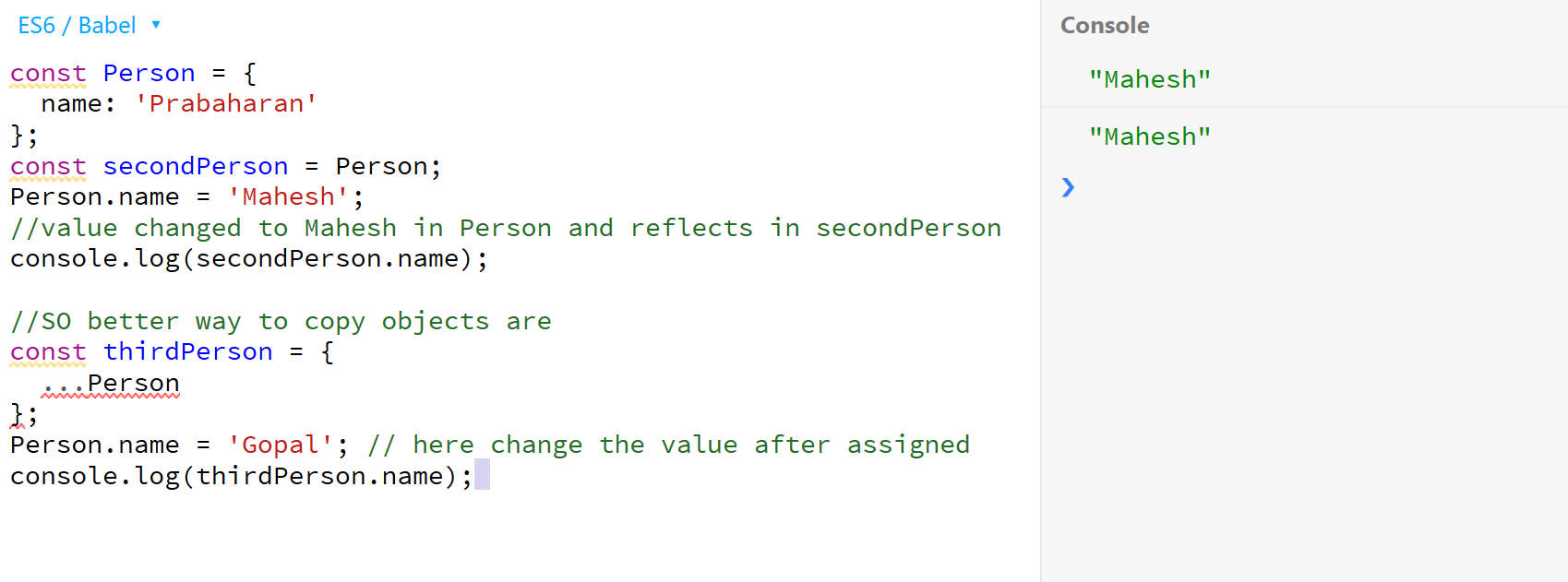
Primitives and reference Types:-

<Prabahran>

Primitives can be copied. Primitives are Strings, numbers, Boolean,..

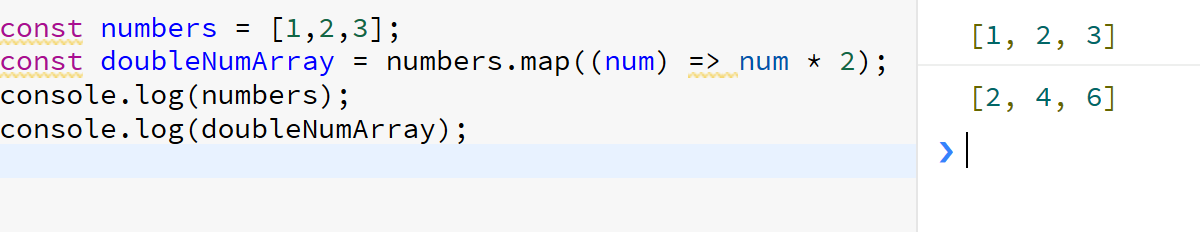
Reference types can not be copied, usually their reference only copied, Ex: Object and Array.

</Prabahran>

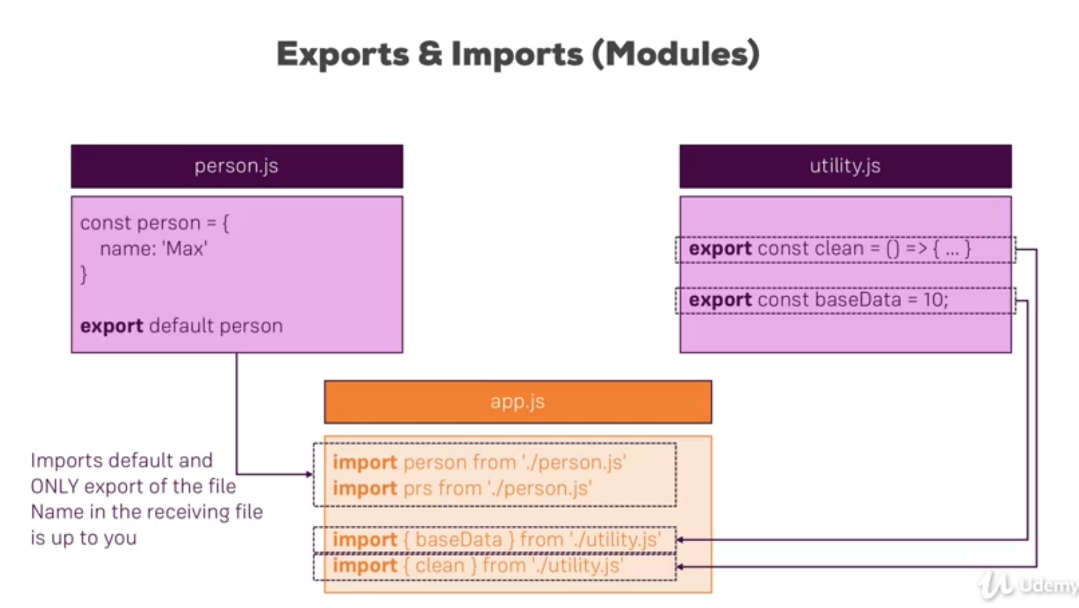


Array Functions:-

Map, Filter, Sort



Exports and Imports in Next Generation JS:-

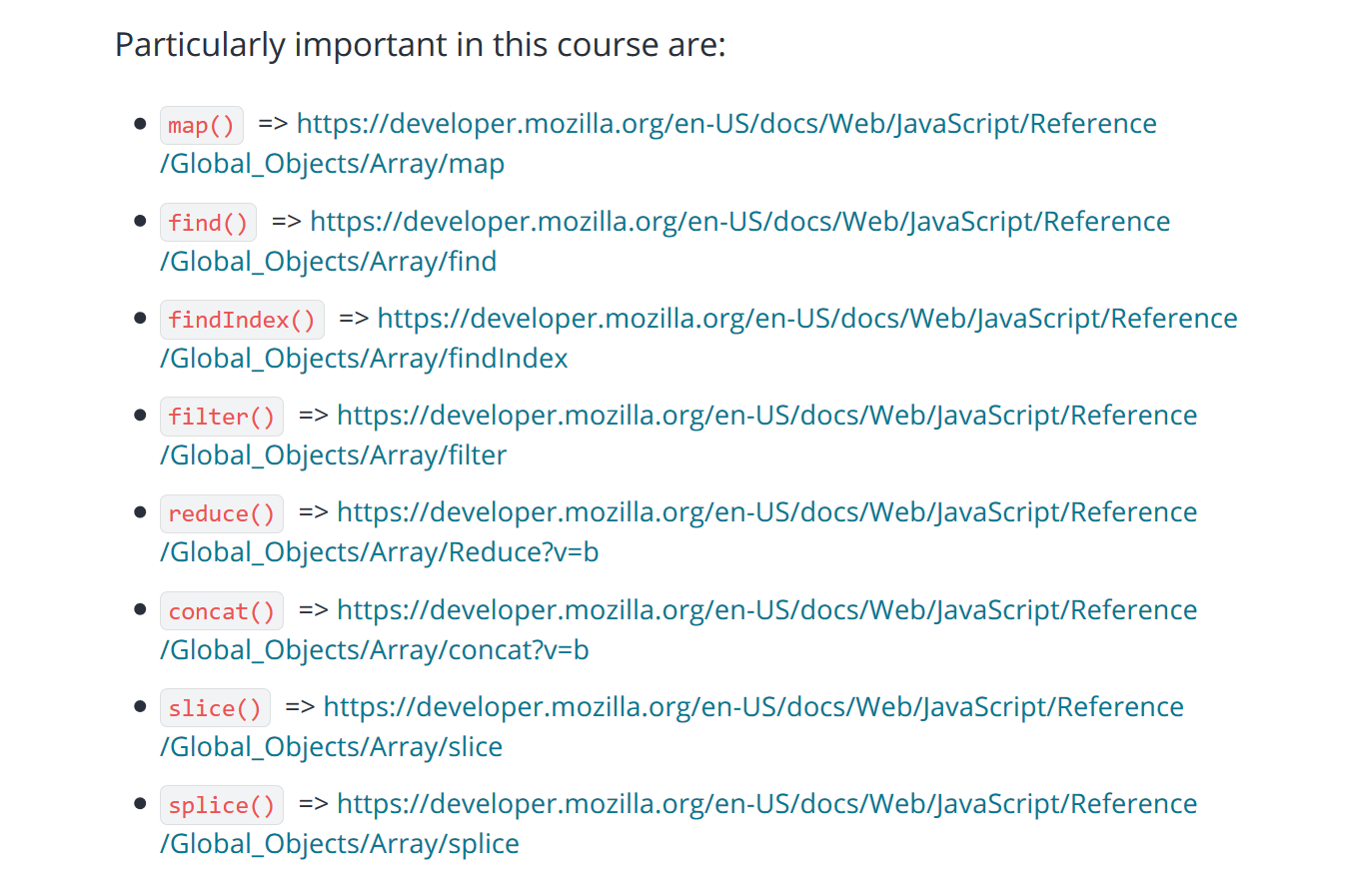


<Prabaharan>

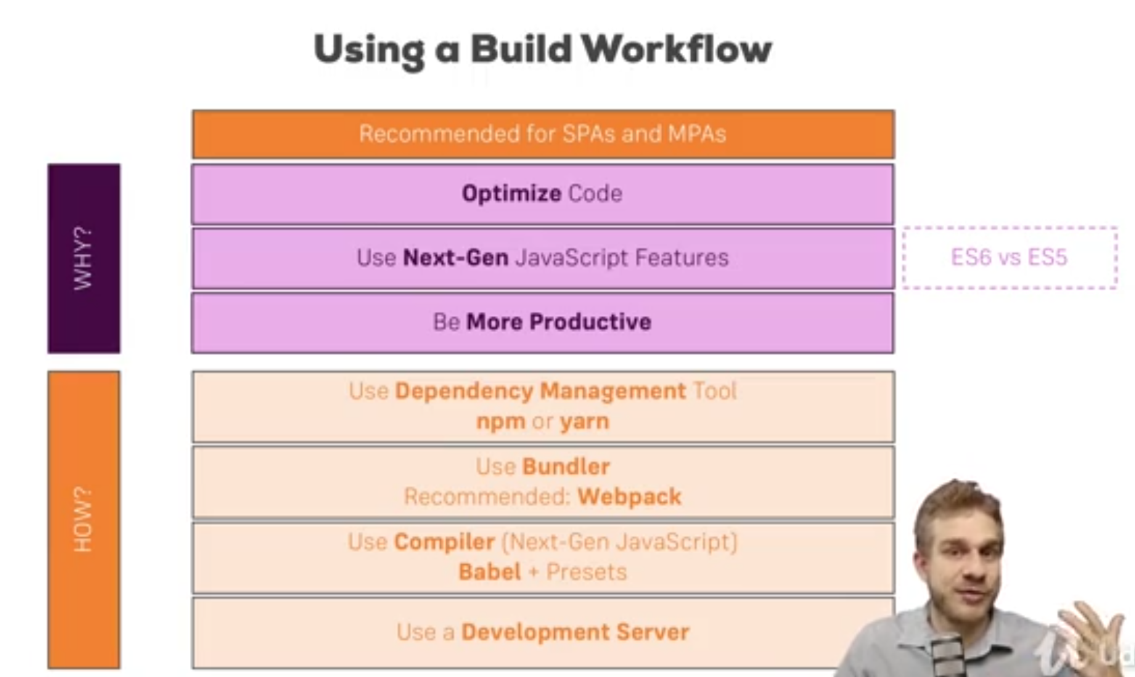
baseData & clean are the named Import

</Prabaharan>

Sample Java Array Functions:-



Build a Workflow:-



How to setup React Up:-

1. npm install create-react-app -g
2. create-react-app <appname>
3. npm start
4. in browser (http://localhost:3000/) type and see the first react application

Files description:-

1. Ignore lock.json file.
2. Package.json - dependencies are described in package.json.

3 dependencies- 1. React, 2. React-dom, 3. React-scripts by default

You can run the scripts by providing npm command

Npm start, npm run build, npm run test, npm run eject

1. Node modules contain all dependent packages by node
2. Public folder, index.html is important

<div id=root> </div>

1. Index.js is the root render component. Derived from app.js

Understanding the component Basics:-

<Prabaharan>

App component (App.js) is used in the Index.js file. In index.js file App component is rendered in place of root element in the index.html.

We can render the normal html tags in app component place.

App is component which extends react component. React component is derived from react. All the react components needs render method. Every react component should render or return HTML code to the DOM.

The html code available in the return method is not a html tag but a jsx. The Html tags are invented by react team, they used syntactical sugar for the tags.

</Prabaharan>

Understanding JSX:-

All the html tags in components will be compiled into JSX format in the end..

<div className='App'>

<h1> WoW!!!!! </h1>

</div>

JSX style;-

//return React.createElement('div',{className: "App"},React.createElement('h1',null,'WoW!!!'),React.createElement('h2',null,'WoW!!!'));

JSX restrictions:-

1. ex: css class is not available className
2. one parent element

<Prabaharan>

React custom components are start with Captital First Letter, so that react can understand this is a custom component.

</Prabaharan>

