**Dart:**

**Variable declaration:**

void main() {

//const and final are used to have fixed values which can’t be changed

const name = 'Mike'; // compile time, must be initialized at time of declaration

final myname; // can have separate initialization and declaration

myName = returnName();

var surname = 'Tyson';

print('My name length is ${name.length}');

print('My name is ${myName + ' ' + surname}'); - //String interpolation

}

String returnName() { //type should be defined for return

return 'Mike';

}

**Classes**

void main() {

var user = new User();

user.firstName = 'Mike'; //this does not follow oops concept of encapsulation

user.lastName = 'Tyson';

user.printCompleteName();

//correct way with Constructor

var user = new User('Mike', 'Tyson');

user.printCompleteName();

}

class User {

var firstName; //declaring type is better for unit testing

String lastName;

//Constructor and initialization

User(this.firstName, this.lastName);

//can't have two constructors with same name OR one class can have only one constructor

//User () {

//

// }

printFirstName() {

print(firstName);//this.firstName can also be used

}

printCompleteName() {

print('${this.firstName + ' ' + lastName}');

}

}

**Classes and List:**

void main() {

var deck = new Deck();

print(deck.cards.length);//52

//add toString method in the class to get result in String format

print(deck.cards);

print(deck.cards.first);//Ace of Spade

//randomizes the order

deck.cards.shuffle();

print(deck.cards.first);

//filtering the data

//=> is called implicit return

//print( deck.cards.where((card) => card.suit == 'Spade'));

//The method 'findAllCardsOfSuit' isn't defined for the class 'Deck'.

//- 'Deck' is from 'package:dartpad\_sample/main.dart' ('lib/main.dart').print(deck.findAllCardsOfSuit('Spade'));

//Error was due to menthod being inside constructor

print(deck.findAllCardsOfSuit('Spade'));

//using sublist

print(deck.getRandomNumberOfCards(2));

deck.removeCard('Heart', 'Queen');

print(deck.getCard().length);//49

}

class Deck{

//initialization is required

List<Card> cards = [];

List<Card> getCard(){

return cards;

}

Deck(){

var ranks = ['Ace', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven',

'Eight','Nine','Ten','Jack', 'Queen', 'King'];

var suits = ['Spade', 'Heart', 'Diamond', 'Club'];

//create deck

for (var rank in ranks){

for(var suit in suits){

cards.add(new Card(rank, suit));

//named parameters to avoid confusion which value to pass first

// cards.add(new Card(rank: rank, suit: suit));

}

}

}

//filtering the list

findAllCardsOfSuit(String suit){

//entire implementation in method i.e. after 'where'((card{}))

return cards.where((card){

return card.suit == suit;

});

}

getRandomNumberOfCards(int count){

var randomCards = cards.sublist(0,count);

cards = cards.sublist(count);

return randomCards;

}

removeCard(String suit, String rank){

cards.removeWhere((card) {

return card.suit == suit && card.rank == rank;

});

}

toString(){

return cards.toString();

}

}

class Card{

String suit;

String rank;

Card(this.rank, this.suit);

//some initializations error

//Card({this.suit, this.rank});

String toString(){

return ('${'\n' + rank +' of ' + suit}');

}

}

**Generic Classes**

void main() {

var circleSlot = new Slot<Circle>();

var squareSlot = new Slot<Square>();

circleSlot.insert(new Circle());

squareSlot.insert(new Square());

//following class will not work cause circleSlot is expecting a circle object

//circleSlot.insert(new Triangle());

}

class Triangle{}

class Circle{}

class Square{}

//generic class, it can accept different type of classes

class Slot<T>{

insert(T shape){

}

}

**Json Parsing**

import 'dart:convert';

void main() {

//json creation

var rawJson = '{"url":"www.google.com", "id":1}';

//parsing

var parsedJson = json.decode(rawJson);

print(parsedJson['url']); //www.google.com

//parsing to a model class through json

var imageModel = new ImageModel.fromJson(parsedJson);

//parsing to a model class through picking variables

var imageModel2 = new ImageModel(parsedJson['id'], parsedJson['url']);

print(imageModel.url);//1

print(imageModel2);//id: 1 {linebreak} url: www.google.com

}

class ImageModel{

//requires to be nullable for jsonparsing in model class[1]

int? id;

String? url;

ImageModel.fromJson(jsonData){//[1]

id = jsonData['id'];

url = jsonData['url'];

}

ImageModel(this.id, this.url);

String toString(){

return ('${'\n' +'id: ' + '$id' + '\nurl: ' + '$url'}');

}

}

**StreamController:**

import 'dart:async'; //it has all stream related stuff

void main(){

//StreamController has two objects

//sink add new data to stream

//it is easy to do time related tasks using stream

final controller = new StreamController();

// final student = new Student('Naoya',true);

final student = new Student('Casimiro',false);

final teacher = new StreamTransformer.fromHandlers(

handleData: (attendance, sink){

if(attendance == true){

sink.add(new ClassRegister());

} else {

sink.addError('${student.name} is absent');

}});

//adds a new value to be processed by stream

controller.sink.add(student);

//map function looks at value and processes it before returning

controller.stream

.map((student) => student.present)

.transform(teacher)

.listen((classRegister) =>

print('${student.name} is present'),

onError:(err)=>print(err));

}

class ClassRegister{}

class Student{

bool present;

String name;

Student(this.name,this.present);

}

import 'dart:async'; //it has all stream related stuff

import 'dart:html';

void main(){

final Element? btn = querySelector('button');

btn?.onClick.timeout(

new Duration(seconds:2),

onTimeout: (sink)=> sink.addError('failed'))

.listen((event)=>print('clicked'),

onError:(err)=>print(err));

}