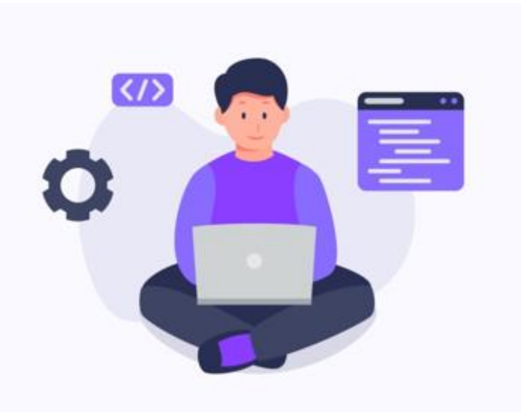

Session 8
JSON in Dart

Session Overview

- Identify JSON in Dart.
- Elaborate on the structure of a JSON document.
- Explain encoding and decoding of JSON objects.

JSON

JavaScript Object Notation (JSON) is a text-based open standard designed and developed for data interchange that is human-readable.

JSON files have `.json` as extension.

Applications use JSON parsing to fetch data from the Internet.

JSON Parsing

- Write JSON parsing codes manually
  - Encoding and decoding JSON
  - Parsing JSON to Dart code using a factory constructor
  - Serializing back to JSON
  - Data validation
  - Defining type-safe model class
- Automate the process of code generation
  - `json_serializable`
  - Freezed

Structure of JSON Document

File `product_data.json`

Code Snippet 1:

```
{
"productName": "Suit",
"brand": "Armani",
"reviews": [
{
"rating": 4.5,
"review": "Superb brand!"
},
{
"rating": 5.0,
"review": "Amazing fabric and look!"
}
]
}
```

Data Types in JSON

Data Types	Description
`String`	Surrounded by quotation marks (" ")
`Number`	Integer or any numeric data type
`Float`	Double (decimal number)
`Array`	JSON array
`Object`	JSON object (can be nested)
`Boolean`	True or false
`Empty`	Null

Encoding of JSON in Dart

`main.dart` is a JSON data encoded as a `string`.

Code Snippet 2: `main.dart`

```
final json = '{"productName": "Suit",
"brand": "Armani", "reviews": [{"rating":
4.5,"review": "Superb brand!"},{"rating":
5.0,"review": "Amazing fabric and look
!"}]}';
```

Decoding JSON with `dart:convert`

`main.dart` **decoding JSON with** `dart:convert`

Code Snippet 3: `main.dart`

```
 import 'dart:convert';

main(){
  final json = '{"productName": "Suit", "brand": "Armani"}';
  final parsedJson = jsonDecode(json);
  print('${parsedJson.runtimeType} : $parsedJson');
}
```

Parsing JSON to Model Class

Following code shows a JSON file and a converted Dart file.

Code Snippet 4: `product_data.json`

```
{
    "productName": "Suit",
    "brand": "Armani"
}
```

Code Snippet 5: `main.dart`

```
class Products
{
  Products({this.productName, this.brand});
  final String productName;
  final String brand;
}
```

Code Snippet 6: `main.dart`

```
products.productName
products.brand
```

Factory Constructor [1-2]

A Factory constructor will return an already created object instead of creating a new instance.

This improves memory and performance.

JSON parsing is done using Factory constructor.

Code Snippet 7: `main.dart`

```
factory Products.fromJson(Map<String, dynamic> data) {
final productName = data['productName'] as String;
final brand = data['brand'] as String;
return Products(productName: productName, brand: brand);
}
```

Factory Constructor [2-2]

Code Snippet 8: main.dart

```
import 'dart:convert';
class Products {
  Products({this.productName, this.brand});
  final String productName;
  final String brand;
  factory Products.fromJson(Map<String, dynamic> data) {
    final productName = data['productName'] as String;
    final brand = data['brand'] as String;
    return Products(productName: productName, brand: brand);
  }
}
main(){
  final json = '{"productName": "Suit", "brand": "Armani"}';
  final parsedJson = jsonDecode(json);
  final products = Products.fromJson(parsedJson);
  print(products.productName);
  print(products.brand);
 }
```

Serialization with `toJson()`

The `toJson()` method is used to convert a model object back to JSON data.

Code Snippet 9: `main.dart`

```
import 'dart:convert';
class Products {
  Products({this.productName, this.brand});
  final String productName;
  final String brand;
  Map<String, dynamic> toJson() {
     return {
    'productName': productName,
    'brand': brand,
     };
  } }
```

```
main(){
  final products = Products(productName: "Suit", brand: "Versache");
  final jsonMap = products.toJson();
  final encodedJson = jsonEncode(jsonMap);
  print(encodedJson);
}
```

Summary

- JSON also called 'JavaScript Object Notation', is a lightweight text-based open standard that is designed and developed for human-readable data interchange.
- JSON has been derived from JavaScript and it is a language-independent data format. Support for it is included in many modern programming languages such as Dart, to generate and parse JSON-format data.
- JSON data can contain both maps of key-value pairs using ({ }) and lists using ([ ]) separated by a colon (:). Each key-value pair is separated by a comma (,).
- JSON data must be encoded or serialized before sending it over the network. This process of manipulating data structure into a string is called encoding.
- When JSON data is received as a string from the network, it must be decoded or deserialized. This process of deserializing the data is called decoding.
- JSON parsing is done using a **Factory** constructor and it allows users to create variables and perform validations before the result is returned.