Survery No. 146 (Part), Village Kaman, Kaman Bhiwandi Road, (Maharashtra) State Highway, Taluka Vasai, Palghar - 401208.
Mob.: 8007478659 / 8007045755

Assignment ~~Experiment~~ No. 1

1) Difference between Web 1.0, 2.0 & 3.0

Ans.

| | Web 1.0 | Web 2.0 | Web 3.0 |
|---|---|---|---|
| 1 | Typically read-only | Strongly read-write | Read write interact |
| 2 | Owned content | Shared Content | Consolidated Content |
| 3 | The web | The Social Web | The Semantic web |
| 4 | Connect information | Connect people | Connect Knowledge. |
| 5 | Information Sharing | Interaction | immersion |
| 6 | 1996 - 2004 | 2004 - 2016 | 2016 and so on |
| 7 | One Directional | Bi - Directional | Multi-user Virtual environment |
| 8 | Focus on Companies | Focus on Communities | Focus on individual |
| 9 | Personal Websites | Blog & Social profile | Semi Blog, Haystack |
| 10 | Static Content | Dynamic Content | It is curiously undefined AI & 3D. The web learning. |
| 11 | Millions of users | Billions of users | Trillions of users |
| 12 | Echo System | participation and interaction | understanding self. |

2) Explain Steps involved in web analytics process.

Ans

① Setting Goals :- This first Steps in the web analytics process is for businesses to determine goals and the end results they are trying to achieve. These goals can include increased Sales, customer Satisfaction and brand awareness. Business goals can be both quantitative and qualitative.

② Collecting data :- The Second Step in web analytics is the collection and Storage of data. Businesses can collect data directly from a website or web analytics tool, Such as Google Analytics. The data mainly comes from Hypertext Transfer Protocol requests -- including data at the network and application levels -- and can be combined with external data to interpret web usage. For example,

③ Processing data :- The next Stage of the web analytics funnel involves businesses processing the collected data into actionable information.

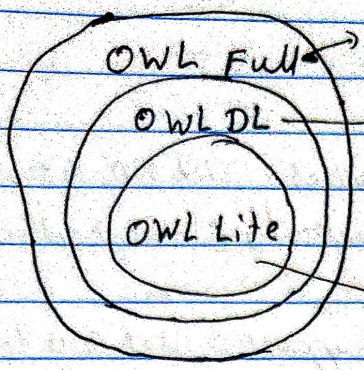
④ Identifying Key performance indicators (KPIs) :- In web analytics, a KPI is a quantifiable measure to monitor and analyze user behavior on a website. Examples include bounce rates, unique users, user Sessions and on-Site Search queries.

⑤ Developing a Strategy :- This Stage involves implementing insights to formulate Strategies that align with an organization's goals. For example, Search queries conducted on Site can help an organization develop a content Strategy based on what users are Searching for on its website.

⑥ Experimenting and testing :- Businesses need to experiment with different Strategies in order to find the one that yields the best results. For example, A/B testing is a Simple Strategy to help learn how an audience responds to different content. The process involves creating two or more versions of content and then displaying it to different audience Segments to reveal which version of the content performs better.

3) Explain web ontology language in detail.

Ans




→ Very expressive but no reasoning.
→ Based on DL - broad expressive power Reasoning capabilities
Widespread over the web.
→ poor expressive power
Not so much exploited.

Web ontology Language is a Semantic web language that is designed to process and integrate information over the web, making Sense of it in a manner similar to human reasoning.
Web ontology Language are built upon a Standard of the World wide web Consortium called Resource Description Framework (RDF).

(1) OWL Lite :- Supports those users primarily needing a classification hierarchy and simple constraints. It only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives and OWL Lite provides a quick migration path for thesauri and other taxonomies.

(2) OWL DL :- Supports those users who want the maximum expressiveness while retaining computational completeness and decidability. OWL DL includes all ~~over~~ OWL language constructs, but they can be used only under certain restrictions. OWL DL is so named due to its correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL.

(3) OWL Full :- It is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment

Vidya Vikas Education Trust
Universal College of Engineering
Survery No. 146 (Part), Village Kaman, Kaman Bhiwandi Road, (Maharashtra) State Highway, Taluka Vasai, Palghar - 401208.
Mob.: 8007478659 / 8007045755

Page No.
Date:

its meaning of the pre-defined vocabulary.

4) Characteristics of Semantic Web.

Ans.

(1) Use of Blockchain and Cryptocurrencies :- Web 3.0 integrates blockchain technology and cryptocurrencies to enable decentralized transactions, content monetization and smart contracts.

(2) Ubiquitous Connectivity and IOT Support :- Web 3.0 leverages ubiquitous connectivity and supports the Internet of Things (IOT), allowing seamless data exchange and interaction across devices and applications.

(3) Foundations in Semantic Web Infrastructure :- Semantic web infrastructure enhances data interoperability, search and analysis, enabling machines to understand and process information more effectively.

(4) Reliance on Artificial Intelligence (AI) :- AI plays a significant role in web 3.0, enabling proactive predictions, natural language processing (NLP), and ~~intuiti~~ intuitive human computer interaction.

(5) Decentralization :- Web 3.0 emphasizes decentralized networks, enabling peer to peer transactions, data ownership and privacy without reliance on intermediaries.

(6) 3D and Spatial-Experienced User Experience :- Web 3.0 incorporates 3D graphics and spatial experiences, enhancing user engagement and immersion in virtual environments.

(7) Supporting the Metaverse :- Web 3.0 facilitates the development of the metaverse, integrating 3D environments, virtual reality, and IOT devices to create immersive online experiences.

(8) Redefined Data ownership :- Web 3.0 empowers users with control over their data, allowing them to market and manage it on decentralized networks, reducing reliance on centralized platforms.

(9) Integration with Edge Computing Infrastructure :- Web 3.0 integrates with edge computing infrastructure, reducing latency and enabling distributed processing closer to data sources, enhancing efficiency and user control.

5) State the features of Typescript. Also State the advantages of Typescript.

Ans Features of Typescript

① Static Typing :- Typescript introduces static typing, allowing developers to define types for variables, function parameters, return types and more. This helps catch type related errors during development and provides better tooling support for code editors.

② Type Inference :- Typescript type inference system can often infer the types of variables and expressions based on their usage, reducing the need for explicit type annotations while providing type safety.

③ Interfaces :- Typescript supports interfaces, which define the shape of objects and provides a way to enforce contracts within the codebase. Interfaces enable better code organization and facilitate ~~interper~~ interoperability between different parts of the code.

④ Enums :- Typescript introduces enums, allowing developers to define named constant values with a set of associated numeric or string values. Enums improve code readability and maintainability by providing meaningful names for values.

⑤ Generics :- Typescript supports generics, enabling the creation of reusable and type safe functions, classes and data structures. Generics allow developers to write flexible and efficient code while maintaining type safety.

⑥ Union Types and Intersection Types :- TypeScript supports union types and intersection types, allowing developers to combine multiple types into a single types. Union types enable flexibility in function parameters and return types, while intersection types allow combining properties from multiple types.

⑦ Decorators :- Typescript supports decorators a form of metadata annotation applied to classes, methods and properties. Decorators enables developers to add behavior or modify the structure of code at runtime, facilitating aspects like dependency injection, logging and validation.

(8) Optional chaining and Nullish coalesing :- Typescript introduces features like optional chaining ('?.') and nullish coalescing ('??') operators, which help streamline code and handle potentially undefined underfined or null values more safely and efficiently.

Advantages of TypeScript :-

(1) ~~Type Safety~~ :- Improved Code Quality :- TypeScript static typing and other features can help to improve the quality of your code by catching errors early on and making your code more organized and maintainable.

(2) Increased Developer productivity :- Typescript tooling support. Such as code completion and refactoring, can help you to write code more quickly and efficiently.

(3) Reduced Bugs :- TypeScript static typing and other features can help to reduce the number of bugs in your code.

(4) Better Collaboration :- TypeScript type annotations can help to make your code more readable and understandable to other developers, which can improve collaboration.

(5) Scalability :- TypeScript features can help to make your code scalable which can be important for large projects.

6) Different between TypeScript and JavaScript.

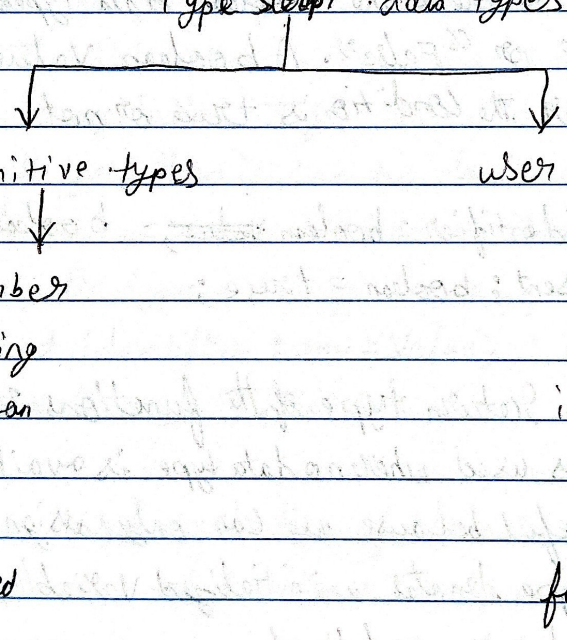
Ans.

| TypeScript | JavaScript |
|---|---|
| (1) It is a compiled language. | It is a Interpreted language. |
| (2) It is an object oriented programming language. | It it a Scripting language. |
| (3) It ~~is~~ Supports Statics typing | It ~~Doesn't~~ Support Dynamic typing. |
| (4) Supports. interfaces | Doesn't Support interfaces. |

| | | |
|---|---|---|
| (5) | Supports optional parameters | Doesn't Support optional parameters. |
| (6) | Supports ~~Rest~~ REST parameters. | Doesn't Support REST parameters. |
| (7) | Supports Generics | Doesn't Support Generics |
| (8) | Supports Modules | Doesn't Support Modules |
| (9) | extensions are .ts and .tsx | extensions are .js and .jsx |
| (10) | need to be compiled. | doesn't need to be compiled. |
| (11) | annotated required Constantly. | annotated not required |

7) List out and explain Typescript data types.

Ans:-




* Primitive types :-

(1) Number :- like JavaScript, all its numbers in TypeScript are stored as floating point values. These numeric values are treated like a number data type. The numeric (2) datatype can be used to represent both integers and fractions.
TypeScript also Supports Binary (Base 2), Octal (Base 8), Decimal (Base 10), and Hexadecimal (Base 16) literals.
~~Syntax~~ Syntax :- let identifier : number = Value;
eg :- let ~~da~~ first : number = 14.0;

(2) String :- We use the String data type to represents the text in Typescript. String type work with textual data. We include String literals in our Scripts by enclosing them in Single or double quotation marks. It also represents a Sequence of unicode characters. It embeds the expressions in its form of ${expr}.

Syntax :- let identifier : String = "";
or
let identifier : String = '';

eg :- let empName : String = "Sachin";

(3) Boolean :- The String and numeric data types can have an unlimited number of different values, whereas the boolean data types can have only two values. They are "true" or "False". A boolean value is a truth value which specifies whether the condition is true or not.

Syntax :-
let identifier : boolean ~~value~~ ; = boolean Value ;

eg :- let isPresent : boolean = true ;

(4) Void :- A void is a Return type of the functions which do not return any type of value. It is used where no data type is available. A variable of type voide is not useful because we can only assign undefined or null to them. An undefined data type denotes uninitialized variable, whereas null represents a variable whose value is undefined.

Syntax :- let identifier : void = undefined ;
eg :- let tempNum : void = undefined ;

(5) Null :- Null Represents a variable whose value is undefined. Much like the void, it is not extremely useful on its own. The Null accepts the only one value, which is null. The Null Keyword is used to defined the Null type in Typescript, but it is not useful because we can only assign a null value to it.

Syntax :- let identifier : ~~number~~ datatype = null;
eg :- let num: number = null;

(7) Undefined :- The undefined primitive type denotes all uninitialized variables in Typescript and Javascript. It has only one value, which is undefined. The undefined keyword defines the undefined type in Typescript, but it is not useful because we can only assign an undefined value to it.
Syntax :- let identifier : datatype = undefined;
eg :- let num: number = undefined;

(8) Any :- It is the "super type" of all data type in Typescript. It is used to represents any Javascript value. It allows us to opt-in and opt-out of type-checking during compilation. If a variable cannot be represented in any of the basic data types, then it can be declared using "Any" data type. Any type is useful when we do not known about the type of value, and we want to skip the type checking on compile time.
Syntax :- let identifier : any = value;
eg :- let val: any = 'Hello';

* user-Defined :-

(9) Array :- An array is a collection of elements of the same data type. like Javascript, Typescript also allows us to work with arrays of values. An array can be written in two ways.
(1) followed by [] to denote an array
Syntax :- let identifier : string [] = [' ', ' ', ' '];
eg :- let fruits : string [] = ['Apple', 'Mango', 'Kiwi'];
(2) ..generic array type Array<element Type>.
Syntax :- let identifier : Array<String> = [' ', ' ', ' '];
eg :- let fruits : Array<String> = ['Apple', 'Mango', 'Kiwi'];

(2) Class :- classes are used to create reusable components and acts as a template for creating objects. It is a logical entity which store variables and functions to perform operations. TypeScript gets support for classes from ES6. It is different from the interface which has an implementation inside it, whereas an interface doesnot have any implementation inside it.

Syntax :- class classname { }

eg :- class Employee { }

(3) Tuple :- The tuple is a data type which includes two sets of values of different data types. It allows us to express an array where the type of a fixed number of elements is known, but they are not the same.
It is a pair of string and number ~~or numb~~

Syntax :- let identifier : [ datatype, datatype ]; a = [ "", ];

eg :- let a : [ string, number ];
a = [ "Hii", 1 ];
console.log (a)

(4) Interface :- An Interface is a structure which acts as a contract in our application. It defines the syntax for classes to follow, means a class which implements an interface is bound to implement all its members. It cannot be instantiated but can be referenced by the class which implements it. The TypeScript compiler uses interfaces for type checking that is also known as "duck typing" or "structural subtyping".

Syntax :- interface ~~IPerso~~ identifier { }

eg :- interface IPerson { }

(5) Enums :- Enums define a set of named constant. Typescript provides both string based and numeric based enums. By default, enums begins numbering their element starting from 0, but we can also change this by manually setting the value to one of its elements. TypeScripts gets support for enums from ES6. Syntax :- enum identifier { }

eg :- enum PrintMedia { }

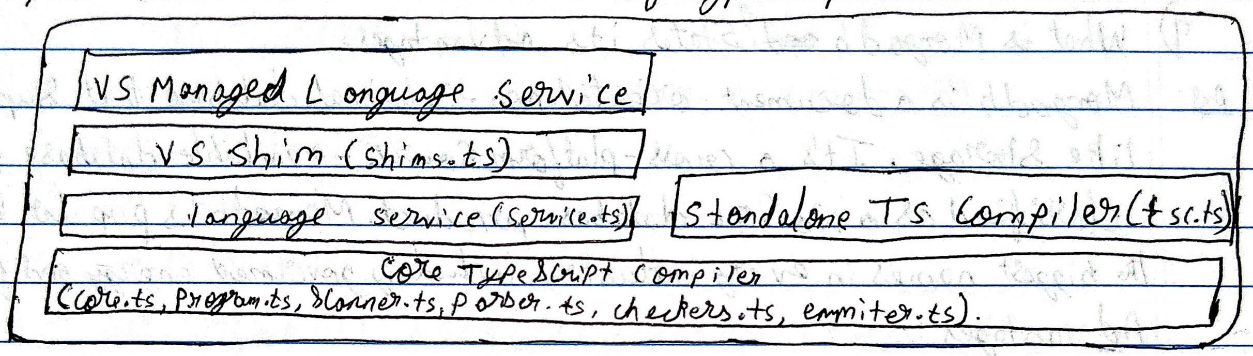
(6) Function :- A function is the logical blocks of code to organize the program. like Javascript, Typescript can also be used to create functions either as a named function or as an anonymous function. Functions ensure that our program is readable, maintainable and reusable. A function declaration has a functions name, return type, and parameters.

Syntax :- function functionname ( ) { }

eg :- function emp ( ) { }

8) Explain the internal architecture of typescript.

Ans :-




(1) Language :- It features the Typescript language elements. It consists of Syntax, Keywords and type annotations.

(2) The Typescript Compiler :- The Typescript compiler (tsc) changes the instructions written in Typescript to its Javascript equivalent. Browser doesn't support the execution of Typescript Code directly. So the program written in Typescript must be rewritten in Javascript equivalent code which supports the execution of code in the browser directly. To perform this, Typescript comes with Typescript compiler named "tsc". The current version of Typescript compiler supports ES 6, by default. It compiles the source code in any module like ES6, SystemJS, AMD, etc. We can install the Typescript compiler by locally, globally, or both with any npm package. Once installation completes, we can compile the Typescript file by running "tsc" command on the command line.

When we compile the Typescript source code, it gives an option to generate a declaration file with the extension .d.ts. This file works as an interface to the components in the compiled Javascript. If a file has an

extension .d.ts, then each most level definition must have the declare Keyword prefixed to it. It makes clear that there will be no code emitted by Typescript, which ensures that the declared item will exist at runtime. The declaration file provides Intellisense for Javascript libraries like jQuery.

(3) The Typescript Language Service :- The "Language Service" reveals an extra layer around the core compiler pipeline that are editor-like application The language service assists the common set of typical editor operations like statement completion, signature help, code formatting and outlining, colorization, etc.

9) What is Mongodb ~~and~~? States its advantages.

Ans: Mongodb is a document-oriented, non-relational database that supports JSON like storage. It's a cross-platform, source-available database program that classified as a NoSQL database product. Mongodb is popular with some of the biggest names in every industry, including governed, energy and financial services.

Advantages :-

(1) Flexible document Schemas.
(2) change-friendly design
(3) powerful querying and analytics.
(4) Easy horizontal Scale-out with Sharding
(5) Simple installation
(6) Cost-effective
(7) Easy Scalability
(8) High performance.

10) What is Collection in Mongodb? Explain with example.

Ans:- In Mongodb, a collection is a group of documents. It's the equivalent of a table in a relational database management System (RDBMS). Collections are Schema-less, meaning documents within a collection don't need to have the same Structure or fields.

Survery No. 146 (Part), Village Kaman, Kaman Bhiwandi Road, (Maharashtra) State Highway, Taluka Vasai, Palghar - 401208.
Mob.: 8007478659 / 8007045755

(1) Creating a Mongodb collection:-
db.createCollection("Student",)

(2) Insert entry in collection:-
db.Student.insertMany([{
...name:"ABC", age:18, Country:India},{
name:"PQR", age:17, Country:"India"}])

(3) Display the collected data by using find():-
db.RecordDB.find()
[{"_id":ObjectId("601c5a123454c8a8f1454827d"),
"name":"ABC",
"age":18,
"Country":"India"},
{"_id":ObjectId("65c4cfd3261eb5953789d5923")
"name":"PQR",
"age":17
"Country":"India"}
]

11) Explain Mongodb CRUD Operations with an example.

Ans:-

(1) Create (or insert) operations add new documents to a collection. There are two ways to add new documents to a collection:-
db.collection.insertOne()
db.collection.insertMany()
insertOne() operation allows us to create individual documents in a collection, while the insertMany() operation is used to create multiple documents in a single operation.
eg:- db.cars.insertOne({name:"Bugatti", Model:"2005"})

(4) Explain Mongodb CRUD Operations with an example.
db.cars.insertMany([{ name: "Aston martin", model: "2018" }, { name: "Ferrari", model: "2013" }])

(2) Read :- Read operations retrieve documents from a collection. Here is the method in Mongodb to retrieve information :-
db.collection.find()
find() operation will return everything from a collection if you call it without any parameters. On the other hand, we can specify any filter or criteria to retrieve information from a collection using :-
db.collection.find(query)
eg:- db.cars.find()
[{ "_id": ObjectId("1"), "name": "Bugatti", "model": "2005 }, { "_id": ObjectId("2"), "name": "Aston Martin", "model": 2018 }, { "_id": ObjectId("3"), "name": "Ferrari", "model": 2013 }]

(3) Update :- update operations modify existing documents in a collection. There are three ways to update documents of a collection :
db.collection.updateOne()
updating a field will not remove the old field instead a new field will be added to the document.
db.collection.updateMany()
updating all field in the document where the given criteria or filter meets the condition.
db.collection.replaceOne()
Replace the entire document. It will replace the old fields and value with new one.
eg:- db.cars.replaceOne({ "_id": ObjectId("1") }, { "new-model": 2020 })

(4) Delete :- Delete operations delete documents from a collection. There are two methods to delete documents of a collection:

db.Collection.deleteOne()
db.collection.deleteMany()

deleteOne() method removes only the first document matched by the query filter document, and deleteMany() deletes multiple objects at once.

db.cars.deleteOne({"model":2013})
db.cars.deleteMany({"model":2005})

12) Difference between Mongodb & MySQL database.

Ans

| | Mongodb | MySQL |
|---|---|---|
| (1) | It is a document-oriented No SQL database. | It is a relational database. |
| (2) | Supports unstructured data | Supports structured data with schemas. |
| (3) | Non-relational Data Model | Relational Data model |
| (4) | Support horizontal Scaling | Support Vertical Scaling |
| (5) | It is not ACID complaint | It is ACID compliant. |
| (6) | Dynamic Schemas | Static Schemas |
| (7) | Rich Data Model is required | No Rich Data Model is required |
| (8) | Auto sharding is required | No Auto Sharding is required |
| (9) | Data Locality is required | No Data Locality is required |
| (10) | Data type is Document Oriented. | Datatype is RDBMS. |
| (11) | No complex transactions is required | Complex transactions is required. |
| (12) | Represents data as Jsondocs | Represents data as tables & rows. |

(13) Explain REST API in detail?

Ans:- REST (Representational State Transfer) API is an architectural style for designing networked applications, particularly web services, that follows a set of constraints to ensure simplicity, scalability and flexibility.

- REST API provides a way to access web services in a simple and flexible manner without extensive processing.
- It enables communication between systems over the internet by using standard HTTP methods like GET, POST, PUT, DELETE, etc.
- REST APIs are preffered over SOAP (Simple Object Access Protocol) due to their simplicity, bandwidth efficient, and flexibility.
- ~~GET~~: A client sends a request to a server using HTTP methods like GET, POST, PUT, or DELETE.
- The server processes the request and sends back a response, typically in the form of a resource such as HTML, XML, image or JSon.
- JSON (Javascript Object Notation) is commonly used for representing Resources in REST APIs due to its simplicity and ease of use.
- GET :- used to retrieve a representation of a resource. Returns HTTP Status 200 for success and 404 or 400 for errors.
- POST :- Creates a new resource, often subordinate to some other resource. Returns HTTP Status 201 for success.
- PUT :- Updates existing resources. Can also create a resource if the ID is chosen by the client. Returns HTTP Status 200 or 201 for success.
- PATCH :- Modifies existing resources with partial updates. Returns HTTP Status 200 for success.
- DELETE :- Deletes a resource identified by a URI. Returns HTTP Status 200 for success.

14) What are the datatypes in Mongodb?

Ans:-

① Integer :- In MongoDB, the integer data type is used to store an integer value. We can store integer data type in two forms 32-bit Signed integer and 64 bit Signed integer.

② String :- This is the most commonly used data type in MongoDB to store data, BSON Strings are of UTF-8. So, the drivers for each programming language convert from the string format of the language to UTF-8 while Serializing and de-Serializing BSON. The String Must be a Valid UTF-8.

③ Double :- The double data type is used to store the floating point values.

④ Boolean :- The boolean data type is used to store either true or false.

⑤ Null :- The Null data type is used to store the null value.

⑥ Array :- The Array is the set of values. It can store the same or different data types values in it. In MongoDB, the array is created using Square brackets ([]).

⑦ Object :- Object data types Stores Embedded documents. Embedded documents are also known as nested documents. Embedded document or nested documents are those types of documents which contain a document inside another document.

⑧ Object Id :- Whenever we create a new document in the collection Mongodb automatically creates a unique objectid for that document. There is an _id field in MongoDB for each document. The data which is stored in id is of hexadecimal format and the length of the Id is 12 bytes which consist :-

- 4-bytes for Timestamp Value
- 5-bytes for Random Value i.e. 3-bytes for machine id and 2-bytes for process id.
- 3-bytes for counter.

You can also create your own id field, but make Sure that the value of that id field must be unique.

(9) Undefined :- This data type stores the undefined values.

(10) Binary Data :- This datatype is used to store binary data.

(11) Date :- Date data type stores date. It is a 64-bit integer which represents the number of milliseconds. BSON data type generally supports UTC datetime and it is signed. If the value of the ~~data type~~ date data type is negative then it represents the dates before 1970. There various methods to return date, it can be returned either as a string or as a date object.

- Date() :- It returns the current date in string format.
- new Date() :- Returns a date object. uses the ISODate() wrapper.
- new ISODate() :- It also returns a date object. uses the ISODate() wrapper.

(12) Min & Max Key :- Min Key compares the value of the lowest BSON element and Max Key compares the value against the highest BSON element. Both are internal data type.

(13) Symbol :- This data type similar to the string data type. It is generally not supported by a mongo shell, but if the shell gets a symbol from the database, then it converts this type into a string type.

(14) Regular Expression :- This data type is used to store regular expressions.

(15) JavaScript :- This data type is used to store JavaScript code into the document without scope.

(16) JavaScript with Scope :- This MongoDB data type store JavaScript data with a scope. This data types is deprecated in MongoDB 4.4.

(17) Timestamp :- In MongoDB, this data type is used to store a timestamp. It is useful when we modify our data to keep a record and the value of this data type is 64-bit. The value of the timestamp data type is always unique.

(18) Decimal :- This MongoDB data type store 128 bit decimal based floating point value. This data type was introduced in MongoDB version 3.4.

15) Explain typescript Modules with Suitable Example?

Ans:- The files created in typescript have global access, which means that variables declared in one file are easily accessed in another file. The global nature can cause code conflicts and can cause issues with execution at run time. We have export and import module functionality which can be used to avoid global variable, function conflicts. This features is available in javascript with ES6 release and also supported in typescript.

Typescript modules are divided into two parts:-

(1) Internal module.
(2) External module.

(1) Internal module:- Internal module is used to logically group variables, function, classes, and interfaces in one unit and then export it in the other module. Typescript gives the name of this logical grouping as a namespace in the latest version. In the old module concept namespace is not present, instead namespace we use the traditional way to use modules in old versions.

eg:- message.js

```
Const message = () => {
    Const name = "Naruto";
    return name + " Uzumaki";
};
export default message;
```

- index.html

```
<!DOCType html>
<html>
<body> <p id = "demo"></p>
<Script type = "module">
    import message from "./message.js";
    document.getElementById("demo").innerHTML = message();
</Script> </body> </html>
```

o/p → Naruto Uzumaki.

(2) External Module :- External modules in TypeScript are used to specify and load dependencies between multiple external js file. If there is only one js file used, then in this case external modules are not used. In that case we use traditional dependency management between Javascript files by using browser script tags (<script></script>).

eg:-

- Script.ts

```
export function multiple (x: number, y: number): number { log (`${x} * ${y}`);
  return x+y;
} function log (message : string ): void {
  console.log ("Numbers", message); }
```

- index.js

```
import { multiple } from "./script";
console.log("value of x*y is:", multiple (6, 2));
```

run tsc Script.ts
run node Index.js

o/p :- Number 6 * 2
value of x*y is: 8