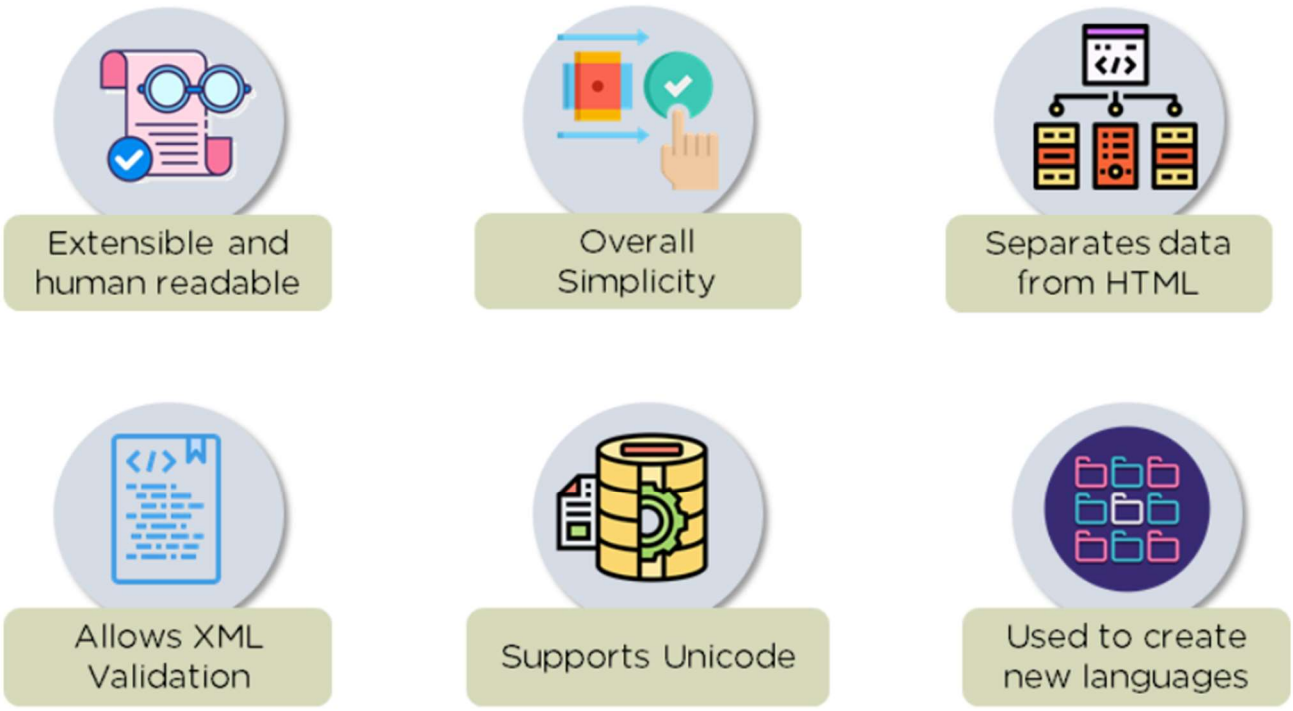
21 Explain the Characteristic and syntax of XML.

XML comes with a lot of features that make it stand out from other languages. Here is the list of come salient features of XML



- Extensible and Human Readable - Most XML applications will work as expected even if new data is added.
- Overall Simplicity - XML simplifies data sharing, data transport, platform changes, and data availability. XML makes it easier to extend or upgrade to new operating systems, new apps, or new browsers without losing data. Data can be made available to a variety of "reading machines," such as people, computers, voice machines, news feeds, and so on.
- Separates data from HTML - Data can be saved in different XML files using XML. This way, you can concentrate on using HTML/CSS for display and layout while knowing that changes to the underlying data will not require HTML changes.
- Allows XML Validation - A DTD or XML schema can be used to validate an XML document. This ensures that the XML document is syntactically valid and prevents any problems from arising from a faulty XML.
- XML supports Unicode - XML is Unicode-compatible, which means it can communicate practically any information in any written human language.
- Used to create new languages - XML has spawned a slew of new Internet languages.
    - For describing available web services, WSDL can be used

- As markup languages for portable devices, WAP and WML are used
- For news feeds, RSS languages are used
- RDF and OWL are used to describe resources and ontologies
- SMIL is a standard for specifying web-based multimedia

XML Syntax

Let us now look at the XML syntax and declaration.

The following is the basic XML syntax -

```
<?xml version = "1.0" encoding = "UTF-8" ?

<root>

   <child>

      <subchild>.....</subchild>

   </child>

</root>
```

22 Write note on XML document prolog section.

XML Prolog is the component added in the beginning of an XML document. Otherwise, we can say that whatever it appears before the document's root element can be considered as Prolog. XML Prolog includes XML declaration, DOCTYPE and comments, processing instructions too.

Example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

<?xml-stylesheet type="text/css" href="/style/design"?>

<!-- This is a comment -->

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<customer_list>
```

```
	<customer>
		<name> Sanjay</name>
		<location> Mumbai</location>
	</customer>
	<customer>
		<name> Micheal</name>
		<location> Washington</location>
	</customer>
</customer_list>
```

23 Write note on XML document element section

XML elements can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.

Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

Syntax

Following is the syntax to write an XML element –

```
<element-name attribute1 attribute2>
....content
</element-name>
```

where,

- **element-name** is the name of the element. The *name* its case in the start and end tags must match.
- **attribute1, attribute2** are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as –

```
name = "value"
```

name is followed by an = sign and a string *value* inside double(" ") or single(' ') quotes.

Empty Element

An empty element (element with no content) has following syntax –

```
<name attribute1 attribute2.../>
```

Following is an example of an XML document using various XML element –

```
<?xml version = "1.0"?>
<contact-info>
  <address category = "residence">
    <name>Tanmay Patil</name>
```

```
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
  </address>
</contact-info>
```

XML Elements Rules

Following rules are required to be followed for XML elements –

- An element *name* can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (_) and period (.).
- Names are case sensitive. For example, Address, address, and ADDRESS are different names.
- Start and end tags of an element must be identical.
- An element, which is a container, can contain text or elements as seen in the above example.

24 Write rules of XML declaration with example

Following syntax shows XML declaration –

```
<?xml
  version = "version_number"
  encoding = "encoding_declaration"
  standalone = "standalone_status"
?>
```

Each parameter consists of a parameter name, an equals sign (=), and parameter value inside a quote. Following table shows the above syntax in detail –

Parameter	Parameter_value	Parameter_description
Version	1.0	Specifies the version of the XML standard used.
Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	It defines the character encoding used in the document. UTF-8 is the default encoding used.
Standalone	yes or no	It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to *no*. Setting it to *yes* tells the processor there are no external declarations required for parsing the document.

Rules

An XML declaration should abide with the following rules –

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: *version, encoding and standalone.*
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. **</?xml>**

25 What is XML? Discuss the rules of XML declaration with example.

SAME AS QUESTION 24:

26 How JSON array of objects works? Explain in detail with example.

JSON Array

JSON array represents ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON array, values must be separated by comma.

The **[** (square bracket) represents JSON array.

JSON Array of Strings

Let's see an example of JSON arrays storing string values.

1. ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

JSON Array of Numbers

Let's see an example of JSON arrays storing number values.

1. [12, 34, 56, 43, 95]

JSON Array of Booleans

Let's see an example of JSON arrays storing boolean values.

1. [**true**, **true**, **false**, **false**, **true**]

JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

```
{"employees":[
    {"name":"Ram", "email":"ram@gmail.com", "age":23},
    {"name":"Shyam", "email":"shyam23@gmail.com", "age":28},
    {"name":"John", "email":"john@gmail.com", "age":33},
    {"name":"Bob", "email":"bob32@gmail.com", "age":41}
]}
```

JSON Multidimensional Array

We can store array inside JSON array, it is known as array of arrays or multidimensional array.

```
[
 [ "a", "b", "c" ],
 [ "m", "n", "o" ],
 [ "x", "y", "z" ]
]
```

28 Explain features of JSON in detail with example.

Features of JSON

- Simplicity
- Openness
- Self-Describing
- Internationalization
- Extensibility
- Interoperability

29 jQuery events with example.

SAME AS QUESTION 2:

30 Explain file system module in Node.js with proper example.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js helps developers to write JavaScript code to run on the server-side, to generate dynamic content and deliver to the web clients. The two features that make Node.js stand-out are:

- Event-driven
- Non-blocking I/O model

About Node.js file system: To handle file operations like creating, reading, deleting, etc., Node.js provides an inbuilt module called FS (File System). Node.js gives the

functionality of file I/O by providing wrappers around the standard POSIX functions. All file system operations can have synchronous and asynchronous forms depending upon user requirements. To use this File System module, use the require() method:

Common use for File System module:

- Read Files
- Write Files
- Append Files
- Close Files
- Delete Files
- Now let us create a js file named **main.js** with the following code:

- javascript

```
var fs = require("fs");

// Asynchronous read
fs.readFile('input.txt', function (err, data) {
   if (err) {
      return console.error(err);
   }
   console.log("Asynchronous read: " + data.toString());
});
```

31 Compare JSON with XML.

XML	JSON
Markup Language	Data Exchange Language
Larger file size, more verbose, slower	Smaller file size, less verbose, faster
Harder to parse through, requires XML parser	Easy to parse with simple JS function, ready to use as JS object
Use with SOAP and REST API's	Use with REST API's
Supports various encoding formats	Only supports UTF-8 and UTF-16 encoding
Data does not have type	Data has a type (string, boolean, number, array, object, null)
Supports schema, namespace, and metadata	Does not support schema, namespace and metadata
Tree (heirarchical) data structure	Map data structure (keys and value pairs)
Additional functionality through XPath, XQuery, XSLT, and Attributes	No additional functionality
Readily rendered by DOM (XML DOM)	Not readily rendered by DOM
Does not support arrrays	Supports arrays
BOTH	
Languages involved in the communication between servers and clients	
Widely used language	
Able to be parsed by a multitude of programming languages	
Can be fetched by XMLHTTPRequest	

32 Explain JSON array and Object

Object in JSON

In JSON an object is represented by a collection of Key-Value pairs. This collection of Key-Value pairs are grouped using { } (*opening and closing curly braces*}. Rules to writing an Object are

- *Key-Value pairs should be separated by a , (Comma)*
- *Each Object should **Start** with an **Opening {** (Opening Curly Brace)*
- Each Object should **End** with a ***Closing }** (Closing Curly Brace)*

An example here is the Person Object, discussed above. The Person object follows the rules mentioned for representing an Object

```
{
 "FirstName" : "Virender",
 "LastName"  : "Singh",
 "Age"       : 34,
 "Profession": "Engineer"
}
```

Array in JSON

Arrays are similar to Arrays that you know from any other programming language. In JSON an Array is collection of Values separated by Comma. Here are the rules to write an Array

- *An Array starts with an opening **[** (Bracket)*
- *An Array ends with a closing **]** (Bracket)*
- *Values in the Array are separated by **,** (Comma)*

To understand an Array let us add one more property to the ***Person Object***. Let us add hobby also, a Person can have multiple hobbies. This makes it suitable to represent hobbies as an Array. As shown in the *JSON* below

```
{
 "FirstName" : "Virender",
 "LastName"  : "Singh",
 "Age"       : 34,
 "Profession": "Engineer",
 "Hobbies"   : ["Videos games", "Computers", "Music"]
}
```

33 How will you write comment in JSON? Explain with example.

JSON is not intended and does not support comments by design, which means that comments in the form // ... or / * ... * / are not allowed in JSON files. But there is a workaround for adding comments to JSON files. To do this, you need to add an element to your JSON file, such as "_comment," which will contain your comment. The JSON API endpoint must ignore this particular JSON comment element. In this JSON comment example, we have included two comment elements in the JSON data, which you can see below.

JSON Comments: Complete Guide with ExamplesFormat

```
{
```

```
    "//first_comment": "The first comment.",
    "//second_comment": "The second comment.",
    "Customer": "Eric Sweet",
    "Id": 1890,
    "Price": 19.0,
    "Quantity": 1,
    "item": {
        "//first_comment": "First nested comment.",
        "//second_comment": "Second nested comment.",
        "Name": "Item"
    }
}
```