Q1

Provide an overview and description of a standard source control process for a large project.

A1

For any software development project, the source code is one of the most important assets that the entire project team nurture over time. As the code base grows it becomes very critical to track what changes were made for which functionality and how new changes will affect current codebase. In a large project where many developers are working on a single codebase, it gets very challenging and critical to track the changes and maintain the working source code.

To manage all of the above challenges a source control or version control mechanism is required that can make sure the code integrity is maintained all the time. The software application that aids in an effective source control mechanism is called **Source Code Management (SCM) Application**.

There two main approaches for source control **Centralized** and **Distributed**. As the name suggests in the Centralized approach there is only one working copy of the source code and the version control is performed on that copy, whenever changes are required to be made on files, developers need to acquire exclusive rights to update the code in a file by checking out and then commit the changes using check-in. However this approach leads to inefficiencies and many other issues, since only one developer can make changes to a file at a time.

In a distributed source control system such as Git, developers acquire a copy of the codebase (source of truth) locally and then make and commit changes to the local copy. When the developer is satisfied with the newly created or updated functionality the changes are synced back to the main shared codebase (a new source of truth). Git being a fully flexible SCM tool, it doesn't enforce a specific way to collaborate however a process or an agreement called git workflow is required to make sure the codebase remains in good shape over a longer period of development life cycle. One of such git workflow that is mainly employed by opensource projects is described below.

1. **Main repo (upstream):** The source of truth or main repo is hosted on a remote. The final production code is derived from this repository.

2. **Fork:** Whenever a developer would like to contribute to a project, s/he forks the main repository into their remote repository. Then this forked repository is cloned by the developer to his/her local development environment.

3. **Add remote to upstream:** The forked repository by default has origin remote which points to the developer's forked repository. The developer also needs to add a remote URL to the main repository which is by convention called upstream.

4. **Topic Branch:** While working on fixing a specific issue or to add a new feature the developer creates a new branch on the cloned repository. This branch is categorized as a topic branch because it specifically created to work on a specific topic. The topic branch can be named appropriate to its purpose. The developer regularly commits the code to the topic branch and pushes to the origin repo.

5. **Merge upstream** When the developer is ready to push his/her changes back to the Main repository s/he needs to first merge the code with the current state of the main branch and resolve the conflicts. Now when the topic branch is stable the code can be pushed to the origin.

6. **Pull Request** Once the topic branch on the forked repository is ready with the latest code from the upstream main. The developer opens a pull request to merge his/her contribution to the main branch of the remote repository. Once the pull request is reviewed and approved by the owner of the main repository the changes from the developer are pushed to the main branch of the main repository.

References

1. https://www.atlassian.com/git/tutorials/what-is-version-control
2. https://confluence.atlassian.com/get-started-with-bitbucket/types-of-version-control-856845192.html
3. https://www.perforce.com/blog/vcs/what-source-control
4. https://lonewolfonline.net/source-control/
5. https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow
6. https://git-scm.com/book/en/v2/Git-Branching-Branching-Workflows

Q2

What are the most important aspects of quality software?

A2

There could be many aspects of quality software from technical and non-technical perspectives however from the end user's perspective following qualities would be at top of the list.

- User Experience

  The term "User experience (UX)" is used to define the user's experience while using the software. For example how well organized the UI is, how well the navigation is structured or even how quick the percieved speed of the application is. It collectively defines how good the user's experience is compared to using other similar applications.

  Generally user-friendly softwares that are easy to use provides better user experience which helps in achieving higher adaption of the application, increased productivity for users of the application, and better results for the business of the organization.

- Reliability

  Any software application is prone to failures which might be due to varied number of reasons. Reliability of application is defined as the number and amount significant failures the user experiences while using the application. The less the number and criticality of the failure the more reliable the application.

  Reliability of the application plays most important part if the application is being used for implementing realtime or business critical processes. For example a Banking Transaction software.

- Secure

  In connected world of today application security is becoming much more critical as more and more computers are interconnected through internet. It is important that the software application handles the user's credentials and data securely to make sure that the data handled by the application can be used only by its intended users and the users who should have access to the information are not deined the access by the application.

  In current competitive world information security plays a big part in success of the entities using the software application.

- Portable

  Portability of application from end user perspective is the different kind of environments the applicatio can run correctly. The environment can be varied number operating systems supported by the application or different kind of browsers in which the application can be run even the different kind of devices on which the application can run.

  Portablity gives users flexibility to use the application in different situations for example on desktop while working in office or home, along with at least limited critical functionality on mobile device while on the move.

  Portablity gives agility to the software when better and/or new hardware devices are created.

- Efficient

  The software applications must be efficient enough to be able to make optimum use of available resources to provide better user experience without interfering with functionality of other applications running in the same environment.

  Efficiency results improving end user productivity as well as cost reduction.

- Flexible

  Flexibility is the measure of how easily changes can be introduced in the application to respond to changes of needs of the users. Flexible applications not only helps in reducing overall cost of the software development but also it helps in providing consistent user experience since the users can still use other features of the application in the same as they have always used.

References

1. https://www.softwaresuggest.com/blog/five-characteristics-make-excellent-software/
2. https://courses.cs.vt.edu/csonline/SE/Lessons/Qualities/index.html
3. http://www.idc-online.com/technical_references/pdfs/data_communications/Characteristics_of_good_software.pdf
4. https://uxplanet.org/how-user-experience-shapes-custom-software-development-78490943b0fc
5. https://www.quora.com/What-are-some-characteristics-of-good-software

Q3

Outline a standard high level structure for a MERN stack application and explain the components

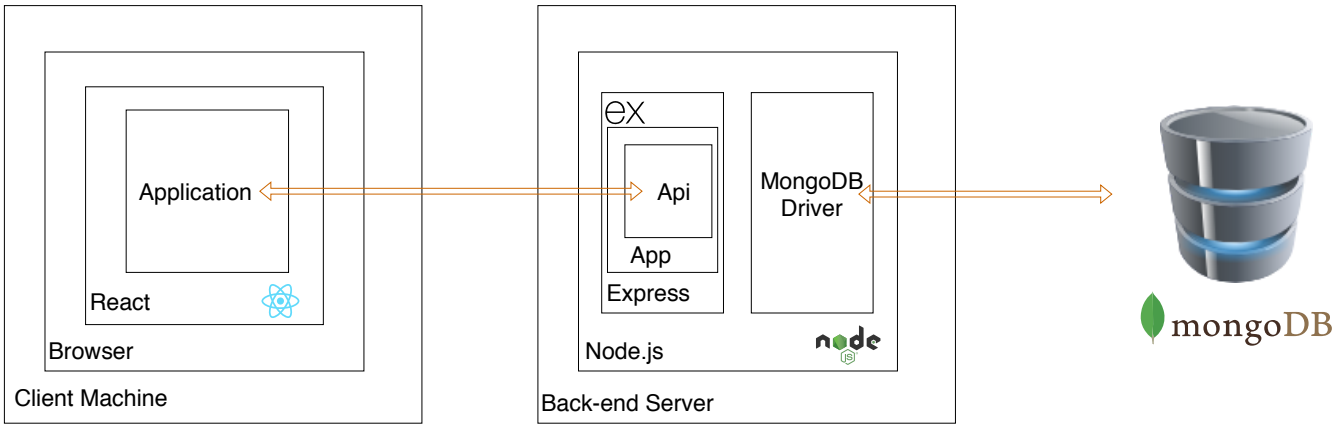
A3

In context of software applications the term stack describes the technologies that are used to build the application functionality. For example **LAMP** stack which refers group of technologies Linux, Apache, MySQL and PHP that can be used to build web applications.

In past few years JavaScript has matured to the level that it can not only be used to develop frontend but backend as well, collectively called fullstack applications. However the web applications built using javascript still needs to use a group of technologies that perform distinct functions. MERN stack is one of such javascript based technology stack which can be used for building fullstack web applications.

The acryonym MERN refers to four components that could be used to build fullstack web applications.

Following modified version of the original diagram from the blog "The Modern Application Stack - Part 1: Introducing The MEAN Stack" illustrates the location of each component in a fullstack app.



1. **MongoDB**

   One of the first component of the MERN stack in the acrynym is the MongoDB. As the name suggests it is a database management system. Though MongoDB not built using JavaScript but its query language and the form of datastorage is JavaScript based.

   MongoDB is a no-sql database which means its not a relational database and hence it does not store the data in form of tables of rows and columns. Instead the data is stored as a document that is nothing but a BSON object. BSON objects are similar to JSON objects of javascript but the distinction is that they are Binary objects unlike JSON.

   The data from MongoDB can be queried using JavaScript, that is interpreted by MongoDB using Shell using JavaScript Run time and then the data is returned as a JSON object which eliminates the need to peform translate the data into object at web application level.

2. **Node.js**

   Node.js is server side JavaScript runtime which uses Google Chrome's V8 engine to parse and execute the javascript code at server side. It also bundles modules to interact with server resources such as file system, networks etc...

Until advent of Node.js JavaScript code could only be executed by the Browser at the frontend of the application by refering the javascript code in the HTML. However with Node.js the JavaScript can not only be used for frontend but also for backend of the application. Due to efficiency of the Google's V8 engine the JavaScript code executes much faster using Node.js compared to other traditional server side languages. It is also more easier for application developers to master one language and deliver fullstack functionality.

Node.js functionality can be extended by building custom modules, that is one of the most popular feature of Node.js besides other popular features.

3. **Express**

   Express is Node.js webserver framework module that enables Node.js to respond to http requests. It can be used to serve HTML documents, rest apis, as well as other types of media using http protocol. Like any webserver it handles routing, request url, header & body parsing and it can also be used to send http responses, set response codes & cookies, custom headers etc...

   Express can also be extended using Node.js modules called middleware. Middleware can be used to perform many of the common tasks performed by web servers such as request body parsing, logging, authentication etc...

   With combination of Node.js, Express and MongoDB one can easily create apis that can respond to request to retrieve and manipulate data in MongoDB over http that is commonly known as backend of the application.

4. **React**

   React is an open source javascript library built and maintained by Facebook. It can be used to create **Single Page Applications (SPAs)**. One of the notorious characteristics of a SPA is that the page transitions are performed asynchronously without reloading of the entire page. This helps in building applications with superior user experience as the page transitions are faster compared to the web applications that does server side rendering. React takes advantage of a concept called "Virtual DOM" which is React equivalent of DOM of an HTML page. React uses optimized alogrithms to update UI aka HTML DOM using Virtual DOM.

   The developers create frontend of the application using React library which contains logic to render the initial UI for application, logic to fetch data from backend based on user interaction with application, and the logic to update UI using the data returned by the api.

References

1. https://www.apress.com/gp/blog/all-blog-posts/why-mern/12056000
2. https://www.geeksforgeeks.org/mern-stack/
3. https://medium.com/nybles/switching-to-the-modern-day-mern-stack-574bb478fc64
4. https://www.iamtimsmith.com/blog/what-is-the-mern-stack-and-how-does-it-work/
5. https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack

Q4

A team is about to engage in a project, developing a website for a small business. What knowledge and skills would they need in order to develop the project?

A4

For any project there are a diverse range of skills required. Similarly, in case of web application development different kind of technical and non-technical skills are essential to successfully sail through the conception to production deployment phase.

Following is a list of essential skills that are required to implement a website for a small business.

1. Requirement Analysis

   Most of the project whether big or small starts with Requirements Analysis in some form to understand what is required to be built.

   Having the skills to analyse the requirements helps to figure out

   - What are the requirements and the major usecases?
   - How the new application will help the customer to solve a problem?
   - Set the boundries terms what needs to be delivered?

2. Project Management and Planning

   Management and Planning are essential for any kind of project to make sure the project is delivered within optimal time frame and the issues are addressed timely. Project management is required to make sure the resources are utilized optimally, the team sticks to the defined timelines, all the goals of the project are achieved. It is one of major skill that helps to make sure the project is completed successfully. Project management skills requires knowledge of at least one of the project management framework such as Agile Project Management.

3. UI Design and Wireframing

   UI Design involves the process of creating intituitive and easy to follow user interfaces. UI designers uses wireframing to sketch and communicate various visual components that might be availalbe to the user to interact with the software application. It helps in communicating and deciding flow of the interaction and what componentes will be present on actual user interface when its ready. For customers it also helps in figuring out if the application can provide all the desired functionality without building the entire application.

4. Programming

   Programming skills are the collective technical skills required to implement UI design, Business Logic, persistence of data in a storage medium, deployment etc... It involves knowledge of frontend technologies, programming languages, database management applications, server deployment and many other relevant technologies that is required to design and implement software applications.

5. Testing

Testing is process of verifying if the web application works correctly for all the use cases that are defined during the requirement gathering. Investigative testing skills important to deliver a good quality website and to make sure that all the required standards of performance are maintained.

References

1. https://onextrapixel.com/12-skills-you-need-to-develop-a-website/
2. https://en.wikipedia.org/wiki/Requirements_analysis
3. https://www.wrike.com/project-management-guide/faq/what-are-project-management-skills/
4. https://balsamiq.com/learn/courses/intro-to-ui-design/what-is-ui-design/
5. https://www.guru99.com/web-application-testing.html

Q5

With reference to one of your own projects, discuss what knowledge or skills were required to complete your project, and to overcome challenges.

A5

Recently I have worked on a Two Sided Marketplace app using Ruby on Rails. The app was a two sided marketplace for Photographers to borrow and lend photography gears in exchange for some money.

I needed following skills to build my app,

- **Search skills**

  I feel my ability to Search Internet using right keywords helped me a lot to

  - Figure out different ideas for the app,
  - Find what kind of functionalityl generally an app the of kinds requires
  - Understand how to implement certain features using Rails
  - Fix various errors

- **Project Management**

  The app was required to be built within fairly short timeline and it was very important to be on track since the project was required to be completed individually. It was very critical to have project and time management skills to breakdown the project into list of tasks, estimate time required to finish each task, track the progress and adjust the milestones based on the progress.

- **UI Design and Wireframing**

  The wireframes for the app was built using Balsamiq Mockups 3. Knowledge of building mockups using Balsamiq helped a lot to figureout various screens that were required to be built to implement various use cases as well how the placements of various components on large and small screens.

  I had build the UI using Twitter Bootstrap 4. Knowledge of laying out various components on large and small screen using Bootstrap to achieve responsive web design worked really well.

- **Programming, Testing and Deployment**

As mentioned earlier the app was built using Ruby on Rails. Knowledge of various Ruby concepts and how it is used by Rails to implement an MVC style app was a big booster to quickly architecht the app, build the backend on PostgreSQL, query the database using inbuilt ORM capabilities provided by Rails and implement the UI by componentising various UI elements.

Fair knowledge of Testing and Deployment techniques was also required to ultimately test the app to make sure all the use cases were implemented correctly and to make the app live on internet by deploying it on a hosting platform.

Q6

With reference to one of your own projects, evaluate how effective your knowledge and skills were for this project, and suggest changes or improvements for future projects of a similar nature.

A6

As per my experience to successfully deliver any project weather small or big requires a lot of skills as well as persistence. When I look back at effectiveness of my skills for two sided market place application, I think following worked effectively

1. Technical know how Rails, Ruby, Database, web development skills etc...
2. Knowledge of Wireframing and principles of responsive design.
3. Effective use of internet to design and fix issues.

I think I still need to improve my self on following

1. More effectively plan the MVP for the application.
2. More effectively plan and manage the tasks from start to its completion.
3. It is also important to effectively scope each functionality to make sure the application can be implemented within defined timelines.

Q7

Explain control flow, using an example from the JavaScript programming language

A7

In a computer application normally all the statements execute sequentially from top to bottom. The Control flow statements are the instructions or statements that alter sequence or in other words control flow of execution of instructions based on specified conditions. If we consider an example of an application that presents user's personal dashboard. Given that the application has access to a service to authenticate current user and a service to show the autheticated user's personal dashboard. In such a situation control statements can be used to show user's dashboard if the current user is signed in or else show login screen.

As evident from above example control statements help the computer or more precisely the CPU to make decision to which path of code to execute based on outcome of a true or false condition commonly known as logical or boolean expression.

Control flow statements provide two kind of functionality decision making and repeating certain statements when certain criteria is true. Below text explains more about both:

1. Conditional

   Conditional statements are mainly used for decision making based on outcome of testing of the condition. There are various kind of conditional statements supported by JavaScript as mentioned below:

   - if

     If statements are used for executing blocks of statements if given condition evaluates to true.

     For example: To check if the user is logged in then show the dashboard following JavaScript statement can be used.

     ```
     if(userLoggedIn) {
         showDashboard(userName);
     }
     ```

   - else

     Else statement is used in conjuction with if statement. It can be used to execute the statements when condition in if statement fails.

     For example: To show login screen when user is not logged in following JavaScript statements can be used.

     ```
     if(userLoggedIn) {
         showDashboard(userName);
     } else {
         showLoginScreen();
     }
     ```

   - else if

     Else If statement can also be used in conjuction with if statement to evaluate another condition when condition in previous if statement returns falsy value.

     For example: While perfming comparison of two number the outcome would be

     - the first number is smaller than the second number or
     - the first number is larger than the second number or
     - both the numbers are equal

     Above conditions can be checked using JavaScript as follows:

     ```
     ```

```
if(num1 < numb2){
    return "smaller";
} else if(num1 > numb2) {
    return "larger";
} else {
    return "equal";
}
```

- Switch If, Else, Else If statements work very well when ther are limited outcomes while evaluating the expression. However if there are multiple possible outcomes of evaluation of the expression then Switch statement is much more convinient to use

  For example: for calling a specific function based on the route chosen by the user follwing Switch statement can be useful

```
switch (route) {
    case 'route1':
        function1();
        break;

    case 'route2':
        function2();
        break;

    case 'route3'
        function3();
        break;

    default:
        defaultRoute();
}
```

2. Loop

   The loop statements are mainly useful for performing certain repititive tasks. For example traverse through each element of an array list.

   There are various loop statements supprted by JavaScript as listed below

   - for
   - for of
   - for in
   - while

References

1. https://developer.mozilla.org/en-US/docs/Glossary/Control_flow
2. https://exploringjs.com/impatient-js/ch_control-flow.html

3. https://medium.com/javascript-in-plain-english/introduction-to-javascript-control-flow-6272f92b75fa
4. https://dev.to/mugas/control-flow-in-javascript-246l
5. https://www.oreilly.com/library/view/learning-javascript-3rd/9781491914892/ch04.html
6. https://www.codecademy.com/learn/introduction-to-javascript/modules/learn-javascript-control-flow
7. https://www.techotopia.com/index.php/JavaScript_Flow_Control_and_Looping

Q8

Explain type coercion, using examples from the JavaScript programming language

A8

Type coercion is a feature of a lot of programming languages which offer implicit automatic type conversion when an operation is done between operands of different kind of types.

For example: In JavaScript 1 + '5' results in a string '15' This is because while performing addition operation using + operator JavaScript attempts to convert the operand to string if one of the operand is string too.

This type of conversion is permitted and even required because JavaScript does not force string type rules. So in order to continue execution of the application even when two different kindo of types involved in an opearation JavaScript first attempts to perform automatic type conversion. If it fails then only the JavaScript engine throws an error.

Though it's a useful feature if not understood correctly it may also result in some unintended and complicated bugs in any JavaScript application.

for example:

While comparing two variables say a = '1' and b = 'abc' result below operation would be a boolean vaue 'true'

```
a < b === true
```

However if untentionally the datatype of say variable 'a' is changed to number 1 the above code would evaluate to a false value. This kind of mistakes can result in bugs wich are complex and not so easy to spot.

If used correctly Type coersion can be used by the programmers to their example. For example to convert any string value to a number one case use - opeartor as shorthand instead of using any other methods

```
    let x = '42' - 0 //would result in conversion of the string '42' to a
number value 42.
```

References

1. https://hackernoon.com/understanding-js-coercion-ff5684475bfc
2. https://www.freecodecamp.org/news/js-type-coercion-explained-27ba3d9a2839/
3. https://dorey.github.io/JavaScript-Equality-Table/
4. https://developer.mozilla.org/en-US/docs/Glossary/Type_coercion
5. https://2ality.com/2019/10/type-coercion.html
6. https://stackoverflow.com/questions/19915688/what-exactly-is-type-coercion-in-javascript
7. https://exploringjs.com/deep-js/ch_type-coercion.html
8. https://medium.com/codezillas/let-me-coerce-you-into-liking-javascripts-dynamic-typing-system-3cd22c19cb64

Q9

Explain data types, using examples from the JavaScript programming language

A9

Every programming language supports concept of data type to classify the type of values that is stored in variables. There are various kinds of datatypes supported by different kind of programming languages based on it's application. A programming language may support primitive data type which are basic data types such as string, numbers, booleans etc... The other types are composite types which are combination of primitive data types that are based on some kind of data structure.

JavaScript mainly supports string, boolean and number primitive data type. It also supports Array & Object which are the composite types and it also supports few special types null & undefined.

A **Number** data type is used for storing number values which can be a positive or a negative number with or without decimal point value.

A **String** data type is sequence of alphanumeric characters enclosed within single or double quotes.

A **Boolean** data type is used to indicate one of two binary value using keyword true (to indicate postive or yes) and false (to indicate a negative or no).

An **Array** data type is a linear collection or list of values of allowed data types in JavaScript.

An **Object** data type is a collection of key-value pairs like Hashes in Ruby programming language. The key can be used to retrieve the value within the object. Unlike ruby keys are always of type string in JavaScript Objects.

null and **undefined** is used to indicate absence or emptyness of the value in a variable.

Below code are examples of assigning different kind of data types to variables.

```
let num = -1.23 // creates a num variable and assigns value of number type
let s = "Hello world" // creates a variable called s and assigns a string
value to it
```

```
let b = true // creates a variable b and assigs it a boolean value true
let a = [1, "two", [true, false], {name: "hiren"}]
let o = {lang: "JavaScript"}
let nothing = null;
let empty = undefined;
```

Datatypes are mainly useful for defining what kind of operations can be performed on values stored in the variables. For example addition operation using + symbol on number values would result in numeric addition however the same addition operation on two string values would result in concatenation operation.

References

1. https://techterms.com/definition/datatype
2. https://en.wikipedia.org/wiki/Data_type
3. https://searchapparchitecture.techtarget.com/definition/data-type
4. https://hackr.io/blog/c-data-types
5. https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php

Q10

Explain how arrays can be manipulated in JavaScript, using examples from the JavaScript programming language

A10

An array is a datastructure that allows storage of list or sequence values in a variable.

In JavaScript an array can be declared as follows,

```
let array = [1, 2, 3]; // [1, 2, 3]
```

The elements of array can be accessed using 0 based index value. For example: The value 2 from above array can be retrieved as follows,

```
let value = array[1] // value = 2
```

To update any value in the array we can use the index. For example: To replace the last value 3 in above array we can write following code,

```
array[2] = "three"; // [1, 2, "three"]
```

In JavaScript the arrays can also expand or shrink at runtime. For example to add and element at the end of list we can use Push method as follows

```
array.push(4); // will expand the array and will add 4 at end to make array look like [1, 2, "three", 4]
```

To retrieve and remove the last element from array we can use following code

```
let lastItem = array.pop(); // array = [1, 2, "three"], lastItem = 4
```

Similarly we can also use unshift and shift methods respectively to add and remove an element to and from start of an array.

To traverse through all elements of an array we use following loops,

- for
- for of
- forEach

For example, below code uses for of loop to retrieve each element sequentially from the above array and print the value to the console.

```
for(let item of array) {
    console.log(item);
}
```

JavaScript provides many other built-in methods on array objects to perform various kinds of operation. Below table lists some of the most common operations and the methods provided by JavaScript.

Operation	Method
Transform	map
search	find,filter, indexOf, lastIndexOf, findIndexOf, includes
Sorting	sort

References

1. https://www.elated.com/manipulating-javascript-arrays/
2. https://www.w3schools.com/js/js_array_methods.asp
3. https://www.freecodecamp.org/news/manipulating-arrays-in-javascript/
4. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
5. https://www.sitepoint.com/quick-tip-create-manipulate-arrays-in-javascript/
6. https://dev.to/thomasaudo/advanced-array-manipulation-in-javascript--fhi
7. https://www.w3schools.com/js/js_arrays.asp

Q11

Explain how objects can be manipulated in JavaScript, using examples from the JavaScript programming language

A11

An Object in JavaScript is a collection of key-value pairs. There are two types of keys called Properties and Methods. Properties hold value of any kind of data types in permitted in JavaScript where as methods are used to hold only functions which can access all the properties within context of the object.

For example below code creates an object called person which has name property and a greet method which returns a greeting using value of name property.

```
let person = {
  name: 'Alex',
  greet() {
    return `Hello ${this.name}`;
  }
}

console.log(person.greet()); // prints 'Hello Alex'
```

As illustrated in above example the methods of an object can be accessed using dot operator and the methods can be invoked in the same fashion as normal functions. Values of properties stored in object could also be retrieved using dot operator.

For example in person object described above the value of name property can be retrieved as follows

```
console.log(person.name) // prints 'Alex'
```

The dot operator can also be used to update values of properties of an object. For example to update value of name property following code can be used.

```
person.name = 'Kanye' // overwrites value of 'Alex' in name property to 'Kanye'
```

There are many useful functions on Object class that can be used to manipulate objects in JavaScript.

1. Object.keys():

Object.keys() method can be used to retrieve an array of all the keys of pased object.

For example below code returns an array of keys of person object

```
console.log(Object.keys(person)) // prints [ 'name', 'greet' ]
```

2. Object.values():

Similar to Object.keys() method Object.values() method is used to retrieve an array of the values associated with all of the keys in passed object.

For example: Below code prints values associated with all the keys in person objct

```
console.log(Object.values(person)) // prints [ 'Kanye', [Function: greet] ]
```

3. Object.assign():

Object.assign combines keys and values of two or more objects and assigns it to the target object.

For example: Below code can be used to create a new object from person object.

```
let personCopy = Object.assign({}, person);
console.log(personCopy == person); // prints false because Object.assign
copies the keys and values to first empty object and then returns the new
object
console.log(personCopy) // prints { name: 'Kanye', greet: [Function:
greet] }
```

References

1. https://www.geeksforgeeks.org/objects-in-javascript/
2. https://www.w3schools.com/js/js_objects.asp
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object
4. https://medium.com/infancyit/javascript-object-manipulation-5d1145cf06ef
5. https://medium.com/@TK_CodeBear/manipulating-objects-in-javascript-59fefeb6a738
6. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects
7. https://codeburst.io/useful-javascript-array-and-object-methods-6c7971d93230

Q12

Explain how JSON can be manipulated in JavaScript, using examples from the JavaScript programming language.

A12

JSON is acronym for JavaScript Object Notation. It is a text based format which can be easily serialized and deserialized.

It is similar to JavaScript object literals however unlike object literals the keys needs to be enclosed within double quotes. The JSON object supports values of following datatypes only and it does not support methods.

1. string: should be enclosed within double quotes only
2. numbers: should be number values without quotes
3. objects: nested JSON object only
4. arrays: array cotaining values supported by JSON
5. boolean: true or false value without quotes
6. null: null without quotes

Follow example shows depicts a JSON object,

```
let person = {
    "name": "Kanye West",
    "age": 42,
    "home-town": {
        "city": "Chicago",
        "state": "Illinois",
        "country": "US"
```

```
    },
    "childeren": ["North", "Saint", "Chicago", "Psalm"],
    "married": true,
    "personal-debt": null
}
```

".json" extension is used to indicate the file contains an JSON object. JSON is very much popular not only as data storage format but also for storing configuration values, as an intermediate data formate while transfering data between systems.

Similar to object literal, the values of a JSON object can be retrieved using dot notation.

For example, retrieve name and update following JavaScript code can be used

```
let prevName = person.name;
person.name = "Ye"
```

To convert the JSON into JavaScript provides stringify method on JSON object. For example to print the person object as a string on console following code can be used

```
let personString = JSON.stringify(person)
console.log(personString) // prints '{"name":"Kanye West","age":42,"home-
town":{"city":"Chicago","state":"Illinois","country":"US"},"childeren":
["North","Saint","Chicago","Psalm"],"married":true,"personal-debt":null}'
```

To parse a string version of JSON into JSON object JavaScript provide parse method on JSON object. For example: following JavaScript code can be used to convert personSting into JSON object as follows

```
let personCopy = JSON.parse(personString);
```

One of the common use of JSON.stringify() and JSON.parse() is to create deep copy of an object. This makes sure that changes to nested JSON object or array in copy does not mutate the original object.

References

1. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON
2. https://www.w3schools.com/js/js_json.asp
3. https://www.digitalocean.com/community/tutorials/how-to-work-with-json-in-javascript
4. https://www.tutorialrepublic.com/javascript-tutorial/javascript-json-parsing.php
5. https://medium.com/@timothyrobards/understanding-json-in-javascript-5098876d0915

Q13

For the code snippet provided below, write comments for each line of code to explain its functionality. In your comments you must demonstrates your ability to recognise and identify functions, ranges and classes

Q13 Code Snippet

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it was made in ' + this.model;
  }
}

let makes = ["Ford", "Holden", "Toyota"]
let models = Array.from(new Array(40), (x,i) => i + 1980)

function randomIntFromInterval(min,max) { // min and max included
    return Math.floor(Math.random()*(max-min+1)+min);
}

for (model of models) {

  make = makes[randomIntFromInterval(0,makes.length-1)]
  model = models[randomIntFromInterval(0,makes.length-1)]

  mycar = new Model(make, model);
  console.log(mycar.show())
}
```

A13

```
// declares a class Car
class Car {
  // defines constructor with an argument brand.
  // The constuctor executes each time a new instace of car is executed.
  constructor(brand) {
    // creates a property carname in scope of instance of Car class
    // and assigns value of brand to carname property
```

```
    this.carname = brand;
  }
  // defines a method present which can be called on the instance of the
Car class
  present() {
    // returns a string 'I have a ' concatenated with value of carname for
the instance of car class
    return 'I have a ' + this.carname;
  }
}

// Declares class Model which inherits all the properties and methods of
car class
class Model extends Car {
  // defines constructor, with two parameters brand and mod, for the Model
class
  constructor(brand, mod) {
    // passes value of brand parameter to the constructor of Car class
    super(brand);
    // assigns value of mod parameter to the model property of the
instance of the Model class
    this.model = mod;
  }
  // defines a method show which can be executed on any instance of Model
class
  show() {
    // returs value of present method which was defined in the Car class
    // concatenated with ', it was made in ' and value of model property
for the instance of the Model class
    return this.present() + ', it was made in ' + this.model;
  }
}

// create an array of string value and assign it to the makes variable
let makes = ["Ford", "Holden", "Toyota"]

// create an Array with 40 elements whose values are in the range 1980 to
2019
// and assign the array to models variable
let models = Array.from(new Array(40), (x,i) => i + 1980)

// Declare a function randomIntFromInterval which accepts two parameters
min and max
function randomIntFromInterval(min,max) { // min and max included
    // generate and return a random number in the range between min and
max numbers
    return Math.floor(Math.random()*(max-min+1)+min);
}

// iterate through each element in models array and and assign it to the
model variable in parent scope
// if the model variable does not exist then create it in the global scope
for (model of models) {
  // create a make variable in global scope and assign it a random element
```

```
from makes array
  make = makes[randomIntFromInterval(0,makes.length-1)]
  // create a model variable in global scope and assign it a random
element from models array
  model = models[randomIntFromInterval(0,makes.length-1)]
  // create a new instance of Model class pass it the value of make and
model variables
  // and then assign the new instance to mycar variable in global scope
  mycar = new Model(make, model);
  // log the value returned by show method on the instance of mycar
variable to the console
  console.log(mycar.show())
}
```