**JavaScript Assignment**

**Submitted By:**

**Pranjal Suyal**

**Theoretical:**

**Q1. What is js ? Why it is so popular for web development ?**

**Ans:** Javascript (JS) is a scripting language, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled. JavaScript renders web pages in an interactive and dynamic manner. This allows the page to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user’s browser, etc.

JavaScript is a flexible and powerful programming language that is implemented consistently by various web browsers.There are a number of factors responsible for it:

* JavaScript is standardized, so it’s frequently updated with new versions.
* JavaScript can be used in both the front-end and back-end web development.
* JavaScript works with the document object model, or the DOM, to respond to user interactions. The DOM is the structure in the browser that displays a web pages.
* JavaScript allows websites to have interactivity like scroll transitions and object movement. Modern browsers still compete to process JavaScript the fastest for the best user experiences. Chrome, the most used Internet browser in 2017, has been so successful in part because of its ability to process JavaScript quickly.
* JavaScript offers a wide range of frameworks and libraries that help developers create complex applications with low overhead. Programmers can import libraries and frameworks in their code to augment their application’s functionality.

**Q2. What are data type in js ? Differences between let, const, var ?**

**Ans:** Programming languages all have built-in data structures, but these often differ from one language to another.JavaScript is a loosely typed or a dynamic language. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types.

There are 7 basic data types in JavaScript.

* **number** for numbers of any kind: integer or floating-point.
* **string** for strings. A string may have one or more characters, there’s no separate single-character type.
* **boolean** for true/false.
* **null** for unknown values – a standalone type that has a single value null.
* **undefined** for unassigned values – a standalone type that has a single value undefined.
* **object** for more complex data structures.
* **symbol** for unique identifiers.

**Differences between let, const, var:**

1. **let** and **const** have a block scope but **var** has function scope.
2. **var** and **let** can change their value and **const** cannot change its value
3. **var** can be accessible anywhere in function but **let** and **const** can only be accessible inside the block where they are declared.
4. **const** cannot be declared only, you need to initialize it with declaration
5. **let** and **const** hoist(where variables and function declarations are moved to the top of their scope before code execution) but you cannot access them before the actual declaration is evaluated at runtime. So in case of **let** and **const** you cannot access them before declaration.

**Q3. Explain logic operators in js with proper examples ?**

**Ans:** There are three logical operators in JavaScript: || (OR), && (AND), ! (NOT).

Although they are called “logical”, they can be applied to values of any type, not only boolean. Their result can also be of any type.

The OR || operator does the following:

* Evaluates operands from left to right.
* For each operand, converts it to boolean. If the result is true, stops and returns the original value of that operand.
* If all operands have been evaluated (i.e. all were false), returns the last operand.

alert( 1 || 0 ); // 1 (1 is truthy)

alert( true || 'no matter what' ); // (true is truthy)

alert( null || 1 ); // 1 (1 is the first truthy value)

alert( null || 0 || 1 ); // 1 (the first truthy value)

alert( undefined || null || 0 ); // 0 (all falsy, returns the last value)

The AND && operator does the following:

* Evaluates operands from left to right.
* For each operand, converts it to a boolean. If the result is false, stops and returns the original value of that operand.
* If all operands have been evaluated (i.e. all were truthy), returns the last operand.

// if the first operand is truthy,

// AND returns the second operand:

alert( 1 && 0 ); // 0

alert( 1 && 5 ); // 5

// if the first operand is falsy,

// AND returns it. The second operand is ignored

alert( null && 5 ); // null

alert( 0 && "no matter what" ); // 0

The boolean NOT operator is represented with an exclamation sign !.

The operator accepts a single argument and does the following:

1. Converts the operand to boolean type: true/false.
2. Returns the inverse value.

For instance:

alert( !true ); // false

alert( !0 ); // true

A double NOT !! is sometimes used for converting a value to boolean type:

**Q4. Difference between switch and if else statement.**

**Ans:** Key Differences Between if-else and switch

1. Expression inside if statement decide whether to execute the statements inside if block or under else block. On the other hand, expressions inside switch statement decide which case to execute.
2. You can have multiple if statements for multiple choice of statements. In switch you only have one expression for the multiple choices.
3. If-else statement checks for equality as well as for logical expression . On the other hand, switch checks only for equality.
4. The if statement evaluates integer, character, pointer or floating-point type or boolean type. On the other hand, switch statement evaluates only character or an integer datatype.
5. Sequence of execution is like either statement under if block will execute or statements under else block statement will execute. On the other hand the expression in switch statement decide which case to execute and if you do not apply a break statement after each case it will execute till the end of switch statement.
6. If expression inside if turn outs to be false, statement inside the else block will be executed. If expression inside switch statement turn out to be false then default statements is executed.
7. It is difficult to edit if-else statements as it is tedious to trace where the correction is required. On the other hand it is easy to edit switch statements as they are easy to trace.

**Q5.What are looping? Explain all ways to loop an array with examples? List them with their differences.**

**Ans:** Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.

There are multiple ways one can iterate over an array in Javascript. The most useful ones are mentioned below.

Using for loop.

This is similar to for loops in other languages like C/C++, Java, etc. for starts at a given index and continues to loop through elements depending upon the condition. For loop checks condition and then executes the code.

|  |
| --- |
| <script>  array = [ 1, 2, 3, 4, 5, 6 ];  for (index = 0; index < array.length; index++) {  console.log(array[index]);  }  </script> |

Output:

1

2

3

4

5

6

Using while loop.

This is again similar to other languages. While loops through elements until condition is proven false, i.e. it checks conditions after execution. So even if the condition is false, the code will execute at least once.

|  |
| --- |
| <script>  index = 0;  array = [ 1, 2, 3, 4, 5, 6 ];    while (index < array.length) {  console.log(array[index]);  index++;  }</script> |

Output:

1

2

3

4

5

6

Using every method.

The every() method checks if all elements in an array pass a test (provided as a function).

|  |
| --- |
| <script>  index = 0;  array = [ 1, 2, 3, 4, 5, 6 ];    const under\_five = x => x < 5;    if (array.every(under\_five)) {  console.log('All are less than 5');  }  else {  console.log('At least one element is not less than 5');  }  </script> |

Output:

At least one element is not less than 5.

using forEach method.

The forEach method calls the provided function once for every array element in the order.

|  |
| --- |
| <script>  index = 0;  array = [ 1, 2, 3, 4, 5, 6 ];    array.forEach(myFunction);  function myFunction(item, index)  {  console.log(item);  }</script> |

Output:

1

2

3

4

5

6

Using map.

A map applies a function over every element and then returns the new array.

map() VS forEach()

Just about anything you can do with forEach() you can do with map(), and vise versa.

map() allocates memory and stores return values. forEach() throws away return values and always returns undefined.

forEach() will allow a callback function to mutate the current array. map() will instead return a new array.

|  |
| --- |
| <script>  index = 0;  array = [ 1, 2, 3, 4, 5, 6 ];    square = x => Math.pow(x, 2);  squares = array.map(square);  console.log(array);  console.log(squares);</script> |

Output:

1 2 3 4 5 6

1 4 9 16 25 36

**Q6. What is an object ? Explain different ways to initialize and update it.**

**Ans.** Objects in JavaScript, is it’s most important data-type and forms the building blocks for modern JavaScript. These objects are quite different from JavaScript’s primitive data-types(Number, String, Boolean, null, undefined and symbol) in the sense that while these primitive data-types all store a single value each (depending on their types).

* Objects are more complex and each object may contain any combination of these primitive data-types as well as reference data-types.
* An object, is a reference data type. Variables that are assigned a reference value are given a reference or pointer to that value. That reference or pointer points to the location in memory where the object is stored. The variables don’t actually store the value.
* objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

Ways to initialize a js objects are :

Using functions as class:

One of the easiest way to instantiate an object in JavaScript. We define a classical JavaScript function and create an object of the function using new keyword. The properties and methods of function are created using the this keyword.

Using object literals:

Literals are smaller and simpler ways to define objects.This method works same along the line of the previous one but instead of bundling the parameters ( name and age ) and the method ( printInfo ) inside of a function, we bundle them in the object itself, initialize the object and simply use the methods.

Singleton using a function:

The third way presented is a combination of the other two that we already saw. We can use a function to define a singleton object.This is a combination of the previous two methods, we bundle the methods and parameters inside a function but don’t declare a separate function for it. Instead we simply use the function structure to declare an object.

**Q7. What is object destructuring ? Explain it with proper use case example.**

**Ans.** Destructuring simply implies breaking down a complex structure into simpler parts. In JavaScript, this complex structure is usually an object or an array. With the destructuring syntax, you can extract smaller fragments from arrays and objects. Destructuring syntax can be used for variable declaration or variable assignment. You can also handle nested structures by using nested destructuring syntax.

Basically, you use an object literal on the left-hand-side of an assignment expression for object destructuring.

Here we used object destructuring syntax to assign values to three variables: firstname, lastname and country using the values from their corresponding keys on the student object. This is the most basic form of object destructuring.

const student = {

firstname: 'Glad',

lastname: 'Chinda',

country: 'Nigeria

},

// Object Destructuring

const { firstname, lastname, country } = student;

console.log(firstname, lastname, country);

**Q8. What is date objects ? Explain all its functions (getting month / year etc..).**

**Ans.**The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ) as shown below.

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

The ECMAScript standard requires the Date object to be able to represent any date and time, to millisecond precision, within 100 million days before or after 1/1/1970. This is a range of plus or minus 273,785 years, so JavaScript can represent date and time till the year 275755.

1 Date()

Returns today's date and time

2 getDate()

Returns the day of the month for the specified date according to local time.

3 getDay()

Returns the day of the week for the specified date according to local time.

4 getFullYear()

Returns the year of the specified date according to local time.

5 getHours()

Returns the hour in the specified date according to local time.

6 getMilliseconds()

Returns the milliseconds in the specified date according to local time.

7 getMinutes()

Returns the minutes in the specified date according to local time.

8 getMonth()

Returns the month in the specified date according to local time.

9 getSeconds()

Returns the seconds in the specified date according to local time.

10 getTime()

Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.

11 getTimezoneOffset()

Returns the time-zone offset in minutes for the current locale.

12 getUTCDate()

Returns the day (date) of the month in the specified date according to universal time.

13 getUTCDay()

Returns the day of the week in the specified date according to universal time.

14 getUTCFullYear()

Returns the year in the specified date according to universal time.

15 getUTCHours()

Returns the hours in the specified date according to universal time.

**Q9. Explain split(), replace(), slice(), splice(), reduce(), filter() with examples in respect to both string & array (In which Applicable).**

**Ans.**

split():

The split() method is used to split a string into an array of substrings, and returns the new array.

If an empty string ("") is used as the separator, the string is split between each character.

The split() method does not change the original string.

Ex:

var str = 'It iS great Day.'

var array = str.split(" ");

print(array);

Output:

[It,iS,a,great,Day.]

replace():

The replace() method searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced.

If you are replacing a value (and not a regular expression), only the first instance of the value will be replaced. To replace all occurrences of a specified value, use the global (g) modifier.

This method does not change the original string.

Ex:

<script>

// Assigning a string

var string = 'Hi my name is Pranjal Suyal’';

// Calling replace() function

var newstring = string.replace(/Pranjal/, 'Gunjan’');

// Printing replaced string

document.write(newstring);

</script>

output: ‘Hi my name is Gunjan Suyal’

slice():

The slice() method returns the selected elements in an array, as a new array object.

The slice() method selects the elements starting at the given start argument, and ends at, but does not include, the given end argument.

The original array will not be changed.

Ex:

var arr = [23,56,87,32,75,13];

var new\_arr = arr.slice();

print(arr);

print(new\_arr);

Output:

[23,56,87,32,75,13]

[23,56,87,32,75,13]

splice():

splice() method is an inbuilt method in JavaScript which is used to modify the contents of an array by removing the existing elements and/or by adding new elements. Parameter: This method accepts many parameters some of them are described below: index: It is a required parameter.

Ex:

const months = ['Jan', 'March', 'April', 'June'];

months.splice(1, 0, 'Feb');

// inserts at index 1

console.log(months);

// expected output: Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');

// replaces 1 element at index 4

console.log(months);

// expected output: Array ["Jan", "Feb", "March", "April", "May"]

reduce():

The reduce () method reduces the array to a single value. The reduce() method executes a provided function for each value of the array (from left-to-right). The return value of the function is stored in an accumulator (result/total).

Ex:

const array1 = [1, 2, 3, 4];

const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4

console.log(array1.reduce(reducer));

// expected output: 10

// 5 + 1 + 2 + 3 + 4

console.log(array1.reduce(reducer, 5));

// expected output: 15

filter():

The filter() method creates an array filled with all array elements that pass a test (provided as a function). filter() does not execute the function for array elements without values. filter() does not change the original array.

Ex:

const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

const result = words.filter(word => word.length > 6);

console.log(result);

// expected output: Array ["exuberant", "destruction", "present"

**Q10. What is XML & JSON? Which one is better & why?**

**Ans.**

**XML**

XML stands for Extensible Markup Language. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.

**JSON**

JavaScript Object Notation is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value). It is a very common data format, with a diverse range of applications, such as serving as replacement for XML in AJAX systems.

JSON is a language-independent data format. It was derived from JavaScript, but many modern programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/json. JSON filename use the extension .json.

JSON is better because :

1. Less verbose- XML uses more words than necessary
2. JSON is faster- Parsing XML software is slow and cumbersome. Many of these DOM manipulation libraries can lead to your applications using large amounts of memory due to the verbosity and cost of parsing large XML files.
3. XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.
4. JSON
   * doesn't use end tag
   * JSON is shorter
   * JSON is quicker to read and write
   * JSON can use arrays.

**Q11. What are cookies ? How to create cookie and delete cookie ?**

**Ans:** Cookies are small items of data, each consisting of a name and a value, stored on behalf of a website by visitors’ web browsers. In JavaScript, cookies can be accessed through the document.cookie object, but the interface provided by this object is very primitive. Cookies.js is a JavaScript object that allows cookies to be created, retrieved, and deleted through a simple and intuitive interface.

Cookies can be created using the set function, which takes the cookie name and value as parameters:

// create a cookie

Cookies.set('theme', 'green');

A cookie can be deleted using the clear function, which takes the cookie name as a parameter:

// delete the theme cookie

Cookies.clear('theme');

**Q12. What are the common properties of HTML forms? List them all.**

**Ans.** The <form> tag in HTML is used to create form for user input. There are many elements which are used within form tag.

The <form> element can contain one or more of the following form elements:

<input> - It is used to specify the input field for user.

<textarea> - It is used to specify for multi-line text input field for user.

<button> - It is used to perform an operation in a form by the user.

<select> - It is used to create a drop-down list.

<option> - It defines an option in a select list.

<optgroup> - It is used to group related options in a drop-down list.

<fieldset> - It is used to group related elements in a form.

<label> - It is used to give label to any tag like button, input etc.

<output> - It represents the result of a calculation (like one performed by a script).

**Q13. What is local storage? What are their methods ? Whether the data persist there after closing browser?**

**Ans:** Web storage objects localStorage and sessionStorage allow to save key/value pairs in the browser.Unlike cookies, web storage objects are not sent to server with each request. Because of that, we can store much more. Most browsers allow at least 2 megabytes of data (or more) and have settings to configure that.Also unlike cookies, the server can’t manipulate storage objects via HTTP headers. Everything’s done in JavaScript.The storage is bound to the origin (domain/protocol/port triplet). That is, different protocols or subdomains infer different storage objects, they can’t access data from each other.

Both storage objects provide same methods and properties:

* setItem(key, value) – store key/value pair.
* getItem(key) – get the value by key.
* removeItem(key) – remove the key with its value.
* clear() – delete everything.
* key(index) – get the key on a given position.
* length – the number of stored items.

The data does not expire. It remains after the browser restart and even OS reboot.

**Q14. Try to set all types of data in local storage. Explain your experience of results.**

**Ans:** To store data you use the setItem() function. This function takes two parameters, the item key and a value.

For String:

localStorage.setItem("lastname", "Suyal");

For Array:

localStorage only supports strings. Use JSON.stringify() and JSON.parse() for converting data to string and then store

The same happens with other data types too, therefore we convert our data first to string and then store it.

For object:

Use stringify to convert your object before storing it, and later parse it when you retrieve it:

**Q15. What is AJAX? Why is it useful ? Explain with examples. (Syntax is not required.)**

**Ans.** AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Javascript.

* Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
* Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
* With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
* XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.

**Usefulness of AJAX**

AJAX is a technique for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Example:

// Creating the XMLHttpRequest object

var request = new XMLHttpRequest();

// Instantiating the request object

request.open("GET", "greet.php?fname=John&lname=Clark");

// Defining event listener for readystatechange event

request.onreadystatechange = function() {

// Check if the request is compete and was successful

if(this.readyState === 4 && this.status === 200) {

// Inserting the response from server into an HTML element

document.getElementById("result").innerHTML = this.responseText;

}

};

// Sending the request to the server

request.send();

The readyState property holds the status of the XMLHttpRequest.

0: request not initialized

1: server connection established

2: request received

3: processing request

4: request finished and response is ready

The onreadystatechange property defines a function to be executed when the readyState changes.

The status property and the statusText property holds the status of the XMLHttpRequest object.

200: "OK"

403: "Forbidden"

404: "Page not found"