**Web Storage or Cookie vs local storage or session storage**

**localStorage object:**  
stores the data without an expiry date   
data will not be deleted when the browser window is closed  
Data storage 5MB  
**sessionStorage object:**stores the data only for one session.  
data is deleted when the browser window closes  
Data is never transferred to the server.  
Storage limit is larger than a cookie (at least 5MB).  
**Cookie**Stores data that has to be sent back to the server with subsequent requests. Its expiration varies based on the type and the expiration duration can be set from either server-side or client-side (normally from server-side).  
Cookies are primarily for server-side reading (can also be read on client-side).  
localStorage and sessionStorage can only be read on client-side.  
Size must be less than 4KB.

**Difference between undefined and null**Both represent empty values   
Undefined ==> when we define a variable it would return undefined because we aren’t assign any value it would take automatically undefined   
Null ==> we can manually empty a variable by using null. if we assign null means the variable will get empty value   
  
typeof(undefined) => undefined  
  
typeof(null) => objects  
 **Difference between let and var**Both are used to declare a variable/method  
Var is function scope   
Let is block scope   
 **delete()**The delete keyword deletes a property from an object  
  
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};  
delete person.age; // or delete person["age"];  **What is the difference between “==” and “===” ? Explain with example**Both are comparison operators  
== compares value   
=== compares type   
 **Console.log inside condition like(5=== 6)**console.log(5=== 6)  
OUTPUT:  
false **How to define two dimensional array.**You can just declare a regular array like so:  
  
var arry = [];  
Then when you have a pair of values to add to the array, all you need to do is:  
  
arry.push([value\_1, value2]);  
Ex:  
> var arry = [];  
undefined  
> arry.push([1,2]);  
1  
> arry  
[ [ 1, 2 ] ]  
> arry.push([2,3]);  
2  
> arry  
[ [ 1, 2 ], [ 2, 3 ] ]  
 **How to check given object is array ?**isArray():  
var a={  
"name":'sakthi',  
"age":23  
}  
Array.isArray(a);  
OUTPUT:  
false

**Method 2: Using the instanceof operator**The instanceof operator is used to test whether the prototype property of a constructor appears anywhere in the prototype chain of an object. This can be used to evaluate if an the given variable has a prototype of ‘Array’.

Syntax:

**variable instanceof Array**

The operator returns a true boolean value if the variable is same as what is specified (here an Array) and false if it is not. This is shown in the example below.

 let str = 'This is a string';

            let num = 25;

            let arr = [10, 20, 30, 40];

            ans = str instanceof Array;

console.log(ans);

            ans = num instanceof Array;

            console.log(ans);

            ans = arr instanceof Array;

console.log(ans);

OUTPUT:

False;

False;

True;

**Method 3: Checking the constructor property of the variable**

Another method to check a variable is an array is by checking it’s constructor with Array.

Syntax:

**variable.constructor === Array**

This becomes true if the variable is same as what is specified (here an Array) and false if it is not. This is shown in the example below.

 let str = 'This is a string';

            let num = 25;

            let arr = [10, 20, 30, 40];

            ans = str.constructor === Array ;

console.log(ans);

            ans = num.constructor === Array;

            console.log(ans);

            ans = arr.constructor === Array;

console.log(ans);

OUTPUT:

False;

False;

True;

**What is the importance of doctype ?   
<!DOCTYPE>**It is an instruction to the web browser about what version of HTML the page is written in.  
Don’t have an end tag   
Not case sensitive   
The doctype declaration must be the very first thing in HTML5 document **Is JavaScript case sensitive?**JavaScript is a case-sensitive language.   
This means that language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.   
The while keyword, for example, must be typed “while”, not “While” or “WHILE”. **What is the result of “20” + 20;**20+20=40  
"20"+"20"="2020"  
"20"+20="2020"  
20+"20"="2020"  
  
"20"+"20"+"20"="202020"  
"20"+"20"+20="202020"  
"20"+20+"20"="202020"  
"20"+20+20="202020"  
  
20+"20"+"20"="202020"  
20+"20"+20="202020"  
20+20+"20"="4020"  
20+20+20=60  
 **How does Delete and unshift works in arrays?  
pop(): Remove an item from the end of an array**let cats = ['Bob', 'Willy', 'Mini'];  
  
cats.pop(); // ['Bob', 'Willy'] **push(): Add items to the end of an array**let cats = ['Bob'];  
  
cats.push('Willy'); // ['Bob', 'Willy']  
  
cats.push('Puff', 'George'); // ['Bob', 'Willy', 'Puff', 'George'] **shift(): Remove an item from the beginning of an array**let cats = ['Bob', 'Willy', 'Mini'];  
  
cats.shift(); // ['Willy', 'Mini'] **unshift(): Add items to the beginning of an array**let cats = ['Bob'];  
  
cats.unshift('Willy'); // ['Willy', 'Bob']  
  
cats.unshift('Puff', 'George'); // ['Puff', 'George', 'Willy', 'Bob'] **How to convert the entire array to a string  
join()**var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var a=fruits.join();  
console.log(a)  
OUTPUT:  
Banana,Orange,Apple,Mango  
 **toString()**var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var a=fruits.toString();  
console.log(a);  
OUTPUT:  
Banana,Orange,Apple,Mango  
 **If you try to use the undeclared variable, what will happen?  
Solution 1:** It would declare as a global variable and assign value **Solution 2:** In strict mode it would throw error  
function x() {  
y = 1; // Throws a ReferenceError in strict mode  
var z = 2;  
}  
  
x();  
  
console.log(y); // logs "1"   
console.log(z); // Throws a ReferenceError: z is not defined outside x **How to implement stack and queue using arrays. Explain.  
// Stack class**class Stack {   
 **// Array is used to implement stack**constructor()   
{   
this.items = [];   
}   
  
push(element)   
{  **// push element into the items**this.items.push(element);   
}   
pop()   
{  **// return top most element in the stack   
// and removes it from the stack   
// Underflow if stack is empty**if (this.items.length == 0)   
return "Underflow";   
return this.items.pop();   
}   
peek()   
{ **// return the top most element from the stack   
// but does'nt delete it.**return this.items[this.items.length - 1];   
}   
isEmpty()   
{  **// return true if stack is empty**return this.items.length == 0;   
}   
printStack()   
{   
var str = "";   
for (var i = 0; i < this.items.length; i++)   
str += this.items[i] + " ";   
return str;   
}   
  
}   
 **// creating object for stack class**var stack = new Stack();   
 **// testing isEmpty and pop on an empty stack   
  
// returns false**console.log(stack.isEmpty());   
 **// returns Underflow**console.log(stack.pop());   
 **// Adding element to the stack**stack.push(10);   
stack.push(20);   
stack.push(30);   
 **// Printing the stack element   
// prints [10, 20, 30]**console.log(stack.printStack());   
 **// returns 30**console.log(stack.peek());   
 **// returns 30 and remove it from stack**console.log(stack.pop());  **// returns [10, 20]**console.log(stack.printStack());  **Queue  
// Queue class**class Queue   
{  **// Array is used to implement a Queue**constructor()   
{   
this.items = [];   
}  **// Functions to be implemented   
// enqueue function**enqueue(element)   
{   
**// adding element to the queue**this.items.push(element);   
}   
**// dequeue function**dequeue()   
{   
**// removing element from the queue   
// returns underflow when called   
// on empty queue**if(this.isEmpty())   
return "Underflow";   
return this.items.shift();   
}   
**// front function**front()   
{   
**// returns the Front element of   
// the queue without removing it.**if(this.isEmpty())   
return "No elements in Queue";   
return this.items[0];   
}   
**// isEmpty function**isEmpty()   
{   
**// return true if the queue is empty.**return this.items.length == 0;   
}   
**// printQueue function**printQueue()   
{   
var str = "";   
for(var i = 0; i < this.items.length; i++)   
str += this.items[i] +" ";   
return str;   
}   
}   
  
**// creating object for queue class**var queue = new Queue();   
  
 **// Testing dequeue and pop on an empty queue   
// returns Underflow**console.log(queue.dequeue());   
 **// returns true**console.log(queue.isEmpty());   
 **// Adding elements to the queue   
// queue contains [10, 20, 30, 40, 50]**queue.enqueue(10);   
queue.enqueue(20);   
queue.enqueue(30);   
queue.enqueue(40);   
queue.enqueue(50);   
queue.enqueue(60);   
  
**// returns 10**console.log(queue.front());   
 **// removes 10 from the queue   
// queue contains [20, 30, 40, 50, 60]**console.log(queue.dequeue());  **// returns 20**console.log(queue.front());  **// removes 20   
// queue contains [30, 40, 50, 60]**console.log(queue.dequeue());  **// printing the elements of the queue   
// prints [30, 40, 50, 60]**console.log(queue.printQueue());  **Prototype   
Every function has a property called prototype   
This property by default it is empty   
And you can add properties and methods into it  
  
Ex:  
Var x=function (j){  
this.i=0;  
this.j=j  
  
this.getJ=function(){  
return this.j  
}  
}  
  
Var x1= new x(1);  
Var x2= new x(2);  
  
console.log(x1.getJ())  
console.log(x2.getJ())  
  
OUTPUT   
1  
2  
  
x1,x2 is the instance of the x  
It inherited all the properties and methods   
  
x is technically parent class   
In each object I would get the getJ() method  
If I great 100 objects from x.each object would have it is own property and methods   
  
So the solution is prototype   
  
Ex  
  
  
Var x=function (j){  
this.i=0;  
this.j=j  
}  
  
x.prototype.getJ=function(){  
return this.j  
}  
  
Var x1= new x(1);  
Var x2= new x(2);  
  
console.log(x1.getJ())  
console.log(x2.getJ())  
  
OUTPUT  
1  
2  
  
Now exactly same output but x1,x2 doesn’t have the own method getJ()  
Instead of it use the parent method getJ()  
  
What are the different ways to create object in Javascript ? Explain each one with example ?   
And which is best ?**