1. What are the primitive data types in JS?

A Primitive data types is a data that is not an object and has no methods or properties.

There are 7 primitive data types in java:

String, number, Boolean, undefined, symbol, null , bigint.  
  
2. What's the difference between a variable that is: null, undefined or undeclared?

Null is pointing to nothing in memory. Undefined is a variable that has not been assigned any value. Lastly, undeclared is a variable that has not been properly declared using const, var, or let.  
  
3. What is the difference between while and do-while loops in JavaScript?

 A while loop checks the condition, then executes the loop. A Do/While executes the loop and then checks the conditions.  
  
4. What language constructions do you use for iterating over object properties and array items?  
for loop, for..in, for each..in, map, reduce etc.

5. What are the promises and how do they work?

A promise is an object that may produce a single value some time in the future: either a resolved value, or a reason that it’s not resolved (e.g., a network error occurred). A promise may be in one of 3 possible states: resolve, rejected, or pending. Promise users can attach callbacks to handle the fulfilled value or the reason for rejection.

A promise is an object which can be returned synchronously from an asynchronous function. It will be in one of 3 possible states:

* **Resolve:** onFulfilled() will be called (e.g., resolve() was called)
* **Rejected:** onRejected() will be called (e.g., reject() was called)
* **Pending:** not yet fulfilled or rejected

6. What are IIFEs and explain with an example where they can be used?

An **Immediately-invoked Function Expression** (IIFE for friends) is a way to execute functions immediately, as soon as they are create. IIFEs are very useful because **they don’t pollute the global object**, and they are a simple way to **isolate variables declarations**.

7. Explain event delegation.

Event delegation refers to **the process of using event propagation (bubbling) to handle events at a higher level in the DOM than the element on which the event originated**. It allows us to attach a single event listener for elements that exist now or in the future.  
  
8. Explain how *this* works in JavaScript.  
a.Can you give an example of one of the ways that working with *this* has changed in ES6?

ES6 allows you to use arrow functions which uses the enclosing lexical scope.  
  
9. Explain how prototypal inheritance works.

The Prototypal Inheritance is a feature in javascript used to add methods and properties in objects. It is **a method by which an object can inherit the properties and methods of another object**. Traditionally, in order to get and set the [[Prototype]] of an object, we use Object. getPrototypeOf and Object.  
  
10. What is a closure, and how/why would you use one? Closures are inner functions inside of an outer function. They have their own local scope and has access to outer function’s scope, parameters (but NOT arguments object), and they also have access to global variables. From what I understand, Closures is a neat way to deal with scope issues. Reasons we use Closures is because Javascript is a *function-level scope* rather than as with other languages, *block-level scope* and Javascript is an *asynchronous/event driven language*. Example that Closure is frequently used is jQuery (ex. click()).

**This is how Closures work.**  
1. After its outer function has been executed and has returned a value, closures can still run.  
2. Closures store references to the outer function’s variable, hence, we will always have access to the updated values of outer function’s variables.  
3. Since we have access to the updated values of outer function’s variables. We will have issue/bugs when a variable changes via for loop, but this can be fixed by using IIFE, Immediately Invoked Function Expression.

11. Can you describe the main difference between the Array.forEach() loop and Array.map() methods and why you would pick one versus the other?

The main difference between map and forEach is that **the map method returns a new array by applying the callback function on each element of an array, while the forEach method doesn't return anything**. You can use the forEach method to mutate the source array, but this isn't really the way it's meant to be used.  
  
12. What's a typical use case for anonymous functions?

An anonymous function is a function that does not have any name associated with it. Normally we use the *function*keyword before the function name to define a function in JavaScript, however, in anonymous functions in JavaScript, we use only the *function*keyword without the function name. An anonymous function is not accessible after its initial creation, it can only be accessed by a variable it is stored in as a *function as a value*. An anonymous function can also have multiple arguments, but only one expression.  
  
13. What's the difference between host objects and native objects? **Host Objects** are objects supplied by a certain environment. They are not always the same because each environment differs and contains host objects that accommodates execution of ECMAScript. Example, browser environment supplies objects such as window. While a node.js/server environment supplies objects such as NodeList. **Native Objects or Built-in Objects**are standard built-in objects provided by Javascript. Native objects is sometimes referred to as ‘Global Objects’ since they are objects Javascript has provided natively available for use.  
  
14. Explain the difference between: function Person(){}, var person = Person(), and var person = new Person()?

function Person(){} is just a normal function declaration. The convention is to use PascalCase for functions that are intended to be used as constructors.

function Person(name) {

this.name = name

}

var person = Person() invokes the Person as a function, and not as a constructor. Invoking as such is a common mistake if the function is intended to be used as a constructor. Typically, the constructor does not return anything, hence invoking the constructor like a normal function will return undefined and that gets assigned to the variable intended as the instance.

var person = Person('John')

console.log(person) // undefined

console.log(person.name) // Uncaught TypeError: Cannot read property 'name' of undefined

var person = new Person() creates an instance of the Person object using the new operator, which inherits from Person.prototype. An alternative would be to use Object.create, such as: Object.create(Person.prototype).

var person = new Person('John')

console.log(person) // Person { name: "John" }

console.log(person.name) // "john"

15. Explain the differences on the usage of foo between function foo() {} and var foo = function() {}

**function foo() {}** is a Normal Function or Function Declaration and **var foo = function() {}** is a Anonymous Function or Expression Function

16. Can you explain what Function.call and Function.apply do? What's the notable difference between the two?

**The call() method takes arguments separately.** The apply() method takes arguments as an array. The apply() method is very handy if you want to use an array instead of an argument list.  
  
17. Explain Function.prototype.bind.

**bind** is a method on the prototype of all functions in JavaScript. It allows you to create a new function from an existing function, change the new function’s this context, and provide any arguments you want the new function to be called with. The arguments provided to bind will precede any arguments that are passed to the new function when it is called.

18. What's the difference between feature detection, feature inference, and using the UA string?

**Feature Detection**Feature detection is just a way of determining if a feature exists in certain browsers. A good example is a modern HTML5 feature ‘Location’.

*if (navigator.geolocation) {  
// detect users location here B-) and do something awesome  
}*

**Feature Inference**Feature Inference is when you have determined a feature exists and assumed the next web technology feature you are implementing unto your app exists as well. Its usually bad practice to assume, so its better to explicitly specify features you want to detect and plan a fallback action.

**UA String**UA String or User Agent String is a string text of data that each browsers send and can be access via navigator.userAgent. These “string text of data” contains information of the browser environment you are targeting.  
If you open your console and run

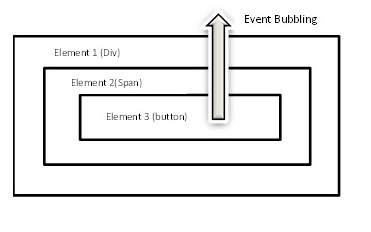
*navigator.userAgent*

19. Explain "hoisting".

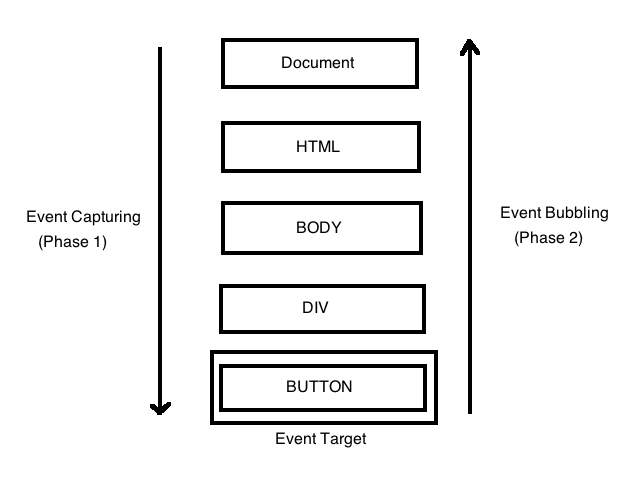
Hosting (also known as Web site hosting, Web hosting, and Webhosting) is **the business of housing, serving, and maintaining files for one or more Web sites**. More important than the computer space that is provided for Web site files is the fast connection to the Internet.

20. Describe event bubbling.

Event bubbling is a type of event propagation where the event first triggers on the innermost target element, and then successively triggers on the ancestors (parents) of the target element in the same nesting hierarchy till it reaches the outermost DOM element or document object (Provided the handler is initialized).

  
  
21. Describe event capturing.

Event capturing is **one of two ways to do event propagation in the HTML DOM**. In event capturing, an event propagates from the outermost element to the target element. It is the opposite of event bubbling, where events propagate outwards from the target to the outer elements. Capturing happens before bubbling.

  
  
22. What's the difference between an "attribute" and a "property"?  
**Attributes are additional information which we can put in the HTML to initialize certain DOM properties.** **Properties are formed when the browser parses the HTML and generates the DOM**. Each of the elements in the DOM have their own set of properties which are all set by the browser.

23. What are the pros and cons of extending built-in JavaScript objects?

The main argument against doing this is: if, in future, a browser decides to implement its own version of your method, your method might get overridden (silently) and the browser's implementation (which is probably different from yours) would take over. So not extending in the first place is future proofing your code.  
  
On the flip side, if you decide to overwrite the browsers definition, any future developer working on your code won't know about the change. They'll have a harder time getting up to speed.  
  
24. What is the difference between == and ===?

== is used for comparison between two variables irrespective of the datatype of variable. === is used for comparision between two variables but this will check strict type, which means it will check datatype and compare two values.  
  
25. Explain the same-origin policy with regards to JavaScript.  
The same origin policy states that a web browser permits a script contained in one page (or frame) to access data in another page (or frame) only if both pages have the same origin. (Same protocol, port, and host)

26. Why is it called a Ternary operator, what does the word "Ternary" indicate?

ternary means operand with three parameters. This is a one-line shorthand for an if-then statement. It is called a ternary operator or a conditional operator.  
  
27. What is strict mode? What are some of the advantages/disadvantages of using it?

Strict mode **prohibits function statements that are not at the top level of a script or function**. In normal mode in browsers, function statements are permitted "everywhere". Strict mode **throws more errors and disables some features in an effort to make your code more robust, readable, and accurate.**  
  
28. What are some of the advantages/disadvantages of writing JavaScript code in a language that compiles to JavaScript?

Advantages of JavaScript

* Speed. Client-side JavaScript is very fast because it can be run immediately within the client-side browser. Unless outside resources are required, JavaScript is unhindered by network calls to a backend server.
* Simplicity. JavaScript is relatively simple to learn and implement.
* Popularity. JavaScript is used everywhere on the web.
* Interoperability. JavaScript plays nicely with other languages and can be used in a huge variety of applications.
* Server Load. Being client-side reduces the demand on the website server.
* Gives the ability to create rich interfaces.

Disadvantages of JavaScript

* Client-Side Security. Because the code executes on the users’ computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable Javascript.
* Browser Support. JavaScript is sometimes interpreted differently by different browsers. This makes it somewhat difficult to write cross-browser code.

29. What tools and techniques do you use debugging JavaScript code?

Web/Browser console using console.log or any developer tools   
  
30. Explain the difference between mutable and immutable objects.

**A mutable object is an object whose state can be modified after it is created.** **Immutables are the objects whose state cannot be changed once the object is created**. Strings and Numbers are Immutable.

a. What is an example of an immutable object in JavaScript?

**String** is an example of an immutable type. A String object always represents the same string.

b. What are the pros and cons of immutability?

c. How can you achieve immutability in your own code?  
**Mutable objects are those whose state is allowed to change over time**. An immutable value is the exact opposite — after it has been created, it can never change. Strings and Numbers are inherently immutable in JavaScript.

31. Explain the difference between synchronous and asynchronous functions.  
The differences between asynchronous and synchronous include: Async is multi-thread, which means operations or programs can run in parallel. Sync is single-thread, so only one operation or program will run at a time. Async is non-blocking, which means it will send multiple requests to a server.

32. What is an event loop?

 An event loop is **something that pulls stuff out of the queue and places it onto the function execution stack whenever the function stack becomes empty**.

a. What is the difference between call stack and task queue?

CallStack is a data structure which keeps track of function calls in our program. When ever we call a function for its execution, we are pushing it to the stack. It is popped out of the stack when the execution is completed.

Task Queue (Micro task Queue) is a JavaScript runtime messaging queue which handles task that is allocated by different Web Apis. This queue is dedicated to handle the Web Apis callbacks. The message are processed once the call stack is clear.

Event Loop has pretty specific work. It has responsibility to see weather the call-stack is empty and does the task queue contains pending task to process. If the call-stack is empty, it will push the task to the call-stack from the queue and the task gets processed.

33. What are the differences between variables created using let, var or const?

**var:**Global scoped or function scoped. The scope of the*var* keyword is the global or function scope. It means variables defined outside the function can be accessed globally, and variables defined inside a particular function can be accessed within the function.

**let:** The scope of a *let*variable is only block scoped. It can’t be accessible outside the particular block ({block}).

**const:** When users declare a *const* variable, they need to initialize it, otherwise, it returns an error. The user cannot update the *const*variable once it is declared.   
  
34. What are the differences between ES6 class and ES5 function constructors?  
**ES6 Class constructors:** ES6 class constructors work quite the same as class constructors in other object-oriented languages. They are used to create new objects.

**ES5 function constructor:** ES5 function constructors are also used to create objects. The above example can be modified as following through the use of function constructors.

35. Can you offer a use case for the new arrow => function syntax? How does this new syntax differ from other functions?  
  
36. What advantage is there for using the arrow syntax for a method in a constructor?

The main advantage of using an arrow function as a method inside a constructor is that the value of this gets set at the time of the function creation and can't change after that. So, when the constructor is used to create a new object, this will always refer to that object.

37. What is the definition of a higher-order function?

A higher order function is **a function that takes a function as an argument, or returns a function**. Higher order function is in contrast to first order functions, which don't take a function as an argument or return a function as output.  
  
38. Can you give an example for destructuring an object or an array?  
  
39. Can you give an example of generating a string with ES6 Template Literals?

Template literals are a new feature introduced in ECMAScript 2015/ ES6. It **provides an easy way to create multiline strings and perform string interpolation**. Template literals are the string literals and allow embedded expressions. Before ES6, template literals were called as template strings.

40. Can you give an example of a curry function and why this syntax offers an advantage?

41. What are the benefits of using spread syntax and how is it different from rest syntax?  
Rest syntax looks exactly like spread syntax. In a way, rest syntax is the opposite of spread syntax. **Spread syntax "expands" an array into its elements, while rest syntax collects multiple elements and "condenses" them into a single element**

42. How can you share code between files?

To share code between components, create an ES6 module in a service component and export the variables or functions that you want to share using standard JavaScript syntax.

An [ES6 module](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules) is a JavaScript file that explicitly exports variables or functions that other modules can use. Modules make it easier to structure your code.

LWC has two patterns for sharing code:

* Create JavaScript files that export code in the same folder as the component importing the code. Import the code using a relative path. Other components can’t import this file directly. This approach supports structuring code within a component rather than sharing code with other components.
* Create a service component (library), which is a component folder that contains one or more JavaScript files that export code. To import the code, other components use c/*componentName* syntax. Components can import code only from the main JavaScript file, which has the same name as the folder name. To share code from supplemental JavaScript files in the library, export the code from those files, then re-export it from the main JavaScript file

43. Why you might want to create static class members?

**Static methods are often utility functions, such as functions to create or clone objects**, whereas static properties are useful for caches, fixed-configuration, or any other data you don't need to be replicated across instances.