

به نام خدا

مجموعه سوال و جواب‌های جاوسکریپت

عیسی رضائی

مجموعه سوال و جواب‌های جاواسکریپت

اگه خوشتون اومد به گیت‌هابمون مراجعه کنین و بهمون ★ بدین. اگر هم قصد مشارکت داشتید خیلی خوشحال می‌شیم 😊

دانلود کتاب به فرمتهای PDF/Epub

می‌تونید از بخش ریلیزهای گیت‌هاب دانلود کنین (این لینک).

پیش گفتار

پرسش و پاسخهای پیش رو، حاصل گردآوری سوالات پر تکراری است که در جلسات مصاحبه برای موقعیت‌های کاری مرتبط با جاوا اسکریپت پرسیده می‌شود.
امیدواریم که با مطالعه این سوالات توی مصاحبه‌های شغلی‌تون موفق باشین و به دانش‌تون اضافه بشه.

فهرست

ردیف	سوال
۱	روش‌های ایجاد objects توی جاوا اسکریپت چیا هستن؟
۲	زنجیره prototype چیه؟
۳	تفاوت‌های بین Call، Apply و Bind چیا هستن؟
۴	فرمت JSON چیه و عملیات‌های معمول بر روی آن چیا هستند؟
۵	هدف از متده slice روی آرایه‌ها چیه؟
۶	هدف از متده splice روی آرایه‌ها چیه؟
۷	تفاوت متدهای slice و splice چیا هستن؟
۸	تفاوت‌های Map و Object چیا هستن؟
۹	تفاوت‌های بین عملگرهای == و === چیا هستن؟
۱۰	توابع arrow-function یا lambda چی هستن؟
۱۱	یه تابع first-class چجور تابعیه؟
۱۲	یه تابع first-order چجور تابعیه؟
۱۳	یه تابع higher-order چجور تابعیه؟
۱۴	یه تابع unary چجور تابعیه؟
۱۵	توابع currying چیه؟
۱۶	چه توابعی pure هستن؟
۱۷	هدف از کلمه کلیدی let چیه؟
۱۸	تفاوت‌های کلمات کلیدی let و var چیا هستن؟
۱۹	دلیل انتخاب کلمه کلیدی let چیه؟
۲۰	چطوری می‌تونیم توی بلوک مربوط به switch بدون دریافت خطا متغیر تعریف کنیم؟
۲۱	Temporal-Dead-Zone چیه؟
۲۲	(توابع بلا فاصله صدا زده شده) چی هستن؟

ردیف	سوال
۲۳	مزایای استفاده از module‌ها چیه؟
۲۴	Memoization چیه؟
۲۵	Hoisting چیه؟
۲۶	Class‌ها توی ES6 چی هستن؟
۲۷	Closure‌ها چیا هستن؟
۲۸	Module‌ها چیا هستن؟
۲۹	چرا به module‌ها نیاز داریم؟
۳۰	توی جاواسکریپت scope چیه و چیکار می‌کنه؟
۳۱	service-worker چیه؟
۳۲	توی service-worker چطوری میشه DOM رو دستکاری کرد؟
۳۳	چطوری می‌تونیم بین ریست شدن‌های service-worker داده‌های مورد نظرمون رو مجدد استفاده کنیم؟
۳۴	IndexedDB چیه؟
۳۵	Web-storage چیه؟
۳۶	Post-message چیه؟
۳۷	Cookie چیه؟
۳۸	چرا به cookie نیاز داریم؟
۳۹	گزینه‌های قابل تنظیم توی cookie چیا هستن؟
۴۰	چطوری میشه ye cookie رو حذف کرد؟
۴۱	تفاوت‌های بین session-storage و cookie، local-storage چیا هستن؟
۴۲	تفاوت‌های بین sessionStorage و localStorage چیا هستن؟
۴۳	چطوری به web-storage دسترسی پیدا می‌کنی؟
۴۴	چه متدهایی روی sessionStorage قابل استفاده هستن؟
۴۵	رخداد storage چیه و چطوری ازش استفاده می‌کنیم؟
۴۶	چرا به web-storage نیاز داریم؟
۴۷	چطوری می‌تونیم پشتیبانی از web-storage توسط مرورگر رو بررسی کنیم؟
۴۸	چطوری می‌تونیم پشتیبانی از web-worker توسط مرورگر رو بررسی کنیم؟
۴۹	یه مثال از web-workerها می‌تونی بزنی؟

ردیف	سوال
۵۰	محدودیت‌های web-worker‌ها روی DOM چیا هستن؟
۵۱	چیه؟ Promise
۵۲	چرا به promise نیاز داریم؟
۵۳	سه تا وضعیت ممکن برای یه promise چیا هستن؟
۵۴	توابع callback چی هستن؟
۵۵	چرا به توابع callback نیاز داریم؟
۵۶	توابع callback یا جهنم Callback-hell چیه؟
۵۷	(Server-sent-events(SSE) چیه؟
۵۸	چطوری می‌تونیم اعلان‌های server-sent-event را دریافت کنیم؟
۵۹	چطوری می‌تونیم پشتیبانی مرورگر برای SSE را بررسی کنیم؟
۶۰	کدوم توابع روی SSE وجود دارن؟
۶۱	اصلی‌ترین قوانین promise‌ها چیا هستن؟
۶۲	توبی callback چطوری رخ میده؟
۶۳	زنجیره promise‌ها چیه؟
۶۴	کاربرد متدها promise.all چیه؟
۶۵	هدف از متدهای promise race چیه؟
۶۶	حالات strict توی جاوااسکریپت چی کار میکنه؟
۶۷	چرا به حالات strict نیاز داریم؟
۶۸	چطوری می‌تونیم حالات strict را فعال کنیم؟
۶۹	هدف از عملگر نقطی دو تایی (!! چیه؟
۷۰	هدف از عملگر delete چیه؟
۷۱	عملگر typeof چیکار می‌کنه؟
۷۲	عملگر undefined چیه و چه زمانی undefined می‌گیریم؟
۷۳	چیه؟ null
۷۴	تفاوت‌های بین null و undefined چیا هستن؟
۷۵	چیه؟ eval
۷۶	تفاوت‌های بین window و document چیا هستن؟

ردیف	سوال
۷۷	توی جاواسکریپت چطوری می‌تونیم به history دسترسی داشته باشیم؟
۷۸	انواع داده‌های جاواسکریپت کدام‌ها هستند؟
۷۹	isNaN چیه و چیکار می‌کنه؟
۸۰	تفاوت‌های بین undefined و undeclared چیا هستن؟
۸۱	کدام متغیرها عمومی هستند؟
۸۲	مشکلات متغیرهای عمومی چیا هستن؟
۸۳	مقدار NaN چیه؟
۸۴	هدف از تابع isFinite چیه؟
۸۵	یه event-flow چیه؟
۸۶	Event-bubbling چیه؟
۸۷	Event-capturing چیه؟
۸۸	چطوری می‌شه یه فرم رو با استفاده از جاواسکریپت ثبت کرد؟
۸۹	چطوری می‌شه به اطلاعات مربوط به سیستم عامل کاربر دسترسی داشت؟
۹۰	تفاوت‌های بین رخدادهای DOMContentLoaded و document-load چیا هستن؟
۹۱	تفاوت‌های بین user و native، host و object چیا هستن؟
۹۲	کدام ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده می‌شن؟
۹۳	مزایا و معایب استفاده از promise به جای callback چیا هستن؟
۹۴	تفاوت‌های بین attribute و property روی DOM چیا هستن؟
۹۵	سیاست same-origin چیه؟
۹۶	هدف استفاده از void چیه؟
۹۷	جاواسکریپت یه زبان تفسیری هست یا کامپایلری؟
۹۸	آیا جاواسکریپت یه زبان حساس به بزرگی و کوچکی(case-sensitive) حروف است؟
۹۹	ارتباطی بین Java و JavaScript وجود داره؟
۱۰۰	Event‌ها چی هستن؟
۱۰۱	کی جاواسکریپت رو ساخته؟
۱۰۲	هدف از متد preventDefault چیه؟
۱۰۳	کاربرد متد stopPropagation چیه؟

ردیف	سوال
۱۰۴	مراحلی که هنگام استفاده از event-handler return false توی یه رخ میده چیا هستن؟
۱۰۵	موارد استفاده از setTimeout کدوما هستن؟
۱۰۶	موارد استفاده از setInterval کدوما هستن؟
۱۰۷	چرا جاواسکریپت رو به عنوان یه زبان تک thread می‌شناسن؟
۱۰۸	چیه؟ Event-delegation
۱۰۹	چیه؟ ECMAScript
۱۱۰	چیه؟ JSON
۱۱۱	قوانین فرمت JSON کدوما هستن؟
۱۱۲	هدف از متدهای JSON.stringify چیه؟
۱۱۳	چطوری می‌تونیم یه رشته JSON(string) رو تجزیه کنیم؟
۱۱۴	چرا به JSON نیاز داریم؟
۱۱۵	PWAها چی هستن؟
۱۱۶	هدف از متدهای clearTimeout چیه؟
۱۱۷	هدف از متدهای clearInterval چیه؟
۱۱۸	توی جاواسکریپت، چطوری میشه به یه صفحه جدید redirect انجام داد؟
۱۱۹	چطوری بررسی می‌کنیم که یه string شامل یه substring هست یا نه؟
۱۲۰	توی جاواسکریپت، چطوری مقدار یه آدرس email رو اعتبارسنجی می‌کنیم؟
۱۲۱	چطوری می‌تونیم مقدار آدرس url جاری رو بخونیم؟
۱۲۲	ویژگی‌های مختلف url روی object history مربوط به کدوما هستن؟
۱۲۳	توی جاواسکریپت چطوری می‌تونیم مقدار یه query-string رو بخونیم؟
۱۲۴	چطوری می‌تونیم بررسی کنیم که آیا یه پرآپرتی روی آبجکت وجود داره یا نه؟
۱۲۵	How do you loop through or enumerate javascript object
۱۲۶	How do you test for an empty object
۱۲۷	What is an arguments object
۱۲۸	How do you make first letter of the string in an uppercase
۱۲۹	What are the pros and cons of for loop
۱۳۰	

سؤال	رديف
How do you display the current date in javascript	١٣١
How do you compare two date objects	١٣٢
How do you check if a string starts with another string	١٣٣
How do you trim a string in javascript	١٣٤
How do you add a key value pair in javascript	١٣٥
Is the '!--' notation represents a special operator	١٣٦
How do you assign default values to variables	١٣٧
How do you define multiline strings	١٣٨
What is an app shell model	١٣٩
Can we define properties for functions	١٤٠
What is the way to find the number of parameters expected by a function	١٤١
What is a polyfill	١٤٢
What are break and continue statements	١٤٣
What are js labels	١٤٤
What are the benefits of keeping declarations at the top	١٤٥
What are the benefits of initializing variables	١٤٦
What are the recommendations to create new object	١٤٧
How do you define JSON arrays	١٤٨
How do you generate random integers	١٤٩
Can you write a random integers function to print integers with in a range	١٥٠
What is tree shaking	١٥١
What is the need of tree shaking	١٥٢
Is it recommended to use eval	١٥٣
What is a Regular Expression	١٥٤
What are the string methods available in Regular expression	١٥٥
What are modifiers in regular expression	١٥٦
What are regular expression patterns	١٥٧

سؤال	ردیف
What is a RegExp object	۱۵۸
How do you search a string for a pattern	۱۵۹
What is the purpose of exec method	۱۶۰
How do you change style of a HTML element	۱۶۱
'What would be the result of 1+2+'3	۱۶۲
What is a debugger statement	۱۶۳
What is the purpose of breakpoints in debugging	۱۶۴
Can I use reserved words as identifiers	۱۶۵
How do you detect a mobile browser	۱۶۶
How do you detect a mobile browser without regexp	۱۶۷
How do you get the image width and height using JS	۱۶۸
How do you make synchronous HTTP request	۱۶۹
How do you make asynchronous HTTP request	۱۷۰
How do you convert date to another timezone in javascript	۱۷۱
What are the properties used to get size of window	۱۷۲
What is a conditional operator in javascript	۱۷۳
Can you apply chaining on conditional operator	۱۷۴
What are the ways to execute javascript after page load	۱۷۵
What is the difference between proto and prototype	۱۷۶
Give an example where do you really need semicolon	۱۷۷
What is a freeze method	۱۷۸
What is the purpose of freeze method	۱۷۹
Why do I need to use freeze method	۱۸۰
How do you detect a browser language preference	۱۸۱
How to convert string to title case with javascript	۱۸۲
How do you detect javascript disabled in the page	۱۸۳
What are various operators supported by javascript	۱۸۴

ردیف	سوال
۱۸۴	What is a rest parameter
۱۸۵	What happens if you do not use rest parameter as a last argument
۱۸۶	What are the bitwise operators available in javascript
۱۸۷	What is a spread operator
۱۸۸	How do you determine whether object is frozen or not
۱۸۹	How do you determine two values same or not using object
۱۹۰	What is the purpose of using object is method
۱۹۱	How do you copy properties from one object to other
۱۹۲	What are the applications of assign method
۱۹۳	What is a proxy object
۱۹۴	What is the purpose of seal method
۱۹۵	What are the applications of seal method
۱۹۶	What are the differences between freeze and seal methods
۱۹۷	How do you determine if an object is sealed or not
۱۹۸	How do you get enumerable key and value pairs
۱۹۹	What is the main difference between Object.values and Object.entries method
۲۰۰	How can you get the list of keys of any object
۲۰۱	How do you create an object with prototype
۲۰۲	What is a WeakSet
۲۰۳	What are the differences between WeakSet and Set
۲۰۴	List down the collection of methods available on WeakSet
۲۰۵	What is a WeakMap
۲۰۶	What are the differences between WeakMap and Map
۲۰۷	List down the collection of methods available on WeakMap
۲۰۸	What is the purpose of uneval
۲۰۹	How do you encode an URL
۲۱۰	How do you decode an URL
۲۱۱	

سؤال	ردیف
How do you print the contents of web page	۲۱۲
What is the difference between uneval and eval	۲۱۳
What is an anonymous function	۲۱۴
What is the precedence order between local and global variables	۲۱۵
What are javascript accessors	۲۱۶
How do you define property on Object constructor	۲۱۷
What is the difference between get and defineProperty	۲۱۸
What are the advantages of Getters and Setters	۲۱۹
Can I add getters and setters using defineProperty method	۲۲۰
What is the purpose of switch-case	۲۲۱
What are the conventions to be followed for the usage of swtich case	۲۲۲
What are primitive data types	۲۲۳
What are the different ways to access object properties	۲۲۴
What are the function parameter rules	۲۲۵
What is an error object	۲۲۶
When you get a syntax error	۲۲۷
What are the different error names from error object	۲۲۸
What are the various statements in error handling	۲۲۹
What are the two types of loops in javascript	۲۳۰
چیه؟ nodejs	۲۳۱
What is an Intl object	۲۳۲
How do you perform language specific date and time formatting	۲۳۳
What is an Iterator	۲۳۴
How does synchronous iteration works	۲۳۵
What is an event loop	۲۳۶
What is call stack	۲۳۷
What is an event queue	۲۳۸

ردیف	سوال
۲۳۹	What is a decorator
۲۴۰	What are the properties of Intl object
۲۴۱	What is an Unary operator
۲۴۲	How do you sort elements in an array
۲۴۳	What is the purpose of compareFunction while sorting arrays
۲۴۴	How do you reversing an array
۲۴۵	How do you find min and max value in an array
۲۴۶	How do you find min and max values without Math functions
۲۴۷	What is an empty statement and purpose of it
۲۴۸	How do you get meta data of a module
۲۴۹	What is a comma operator
۲۵۰	What is the advantage of a comma operator
۲۵۱	What is typescript
۲۵۲	What are the differences between javascript and typescript
۲۵۳	What are the advantages of typescript over javascript
۲۵۴	What is an object initializer
۲۵۵	What is a constructor method
۲۵۶	What happens if you write constructor more than once in a class
۲۵۷	How do you call the constructor of a parent class
۲۵۸	How do you get the prototype of an object
۲۵۹	What happens If I pass string type for getPrototypeOf method
۲۶۰	How do you set prototype of one object to another
۲۶۱	How do you check whether an object can be extendable or not
۲۶۲	How do you prevent an object to extend
۲۶۳	What are the different ways to make an object non-extensible
۲۶۴	How do you define multiple properties on an object
۲۶۵	منظور از MEAN توی جاواسکریپت چیه؟ javascript

ردیف	سوال
۲۶۶	توى جاواسکریپت چىه و چىكار مىكنه؟ javascript Obfuscation
۲۶۷	چە نيازى به Obfuscate كردن داريم؟
۲۶۸	چىه؟ Minification
۲۶۹	What are the advantages of minification
۲۷۰	What are the differences between Obfuscation and Encryption
۲۷۱	What are the common tools used for minification
۲۷۲	How do you perform form validation using javascript
۲۷۳	How do you perform form validation without javascript
۲۷۴	What are the DOM methods available for constraint validation
۲۷۵	What are the available constraint validation DOM properties
۲۷۶	What are the list of validity properties
۲۷۷	Give an example usage of rangeOverflow property
۲۷۸	جاواسکریپت قابليت استفاده از enum رو پيش فرض توى خودش داره؟
۲۷۹	چىه؟ enum
۲۸۰	How do you list all properties of an object
۲۸۱	How do you get property descriptors of an object
۲۸۲	What are the attributes provided by a property descriptor
۲۸۳	How do you extend classes
۲۸۴	How do I modify the url without reloading the page
۲۸۵	How do you check whether an array includes a particular value or not
۲۸۶	How do you compare scalar arrays
۲۸۷	How to get the value from get parameters
۲۸۸	How do you print numbers with commas as thousand separators
۲۸۹	What is the difference between java and javascript
۲۹۰	Is javascript supports namespace
۲۹۱	How do you declare namespace
۲۹۲	How do you invoke javascript code in an iframe from parent page

ردیف	سوال
۲۹۳	How do get the timezone offset from date
۲۹۴	How do you load CSS and JS files dynamically
۲۹۵	What are the different methods to find HTML elements in DOM
۲۹۶	What is jQuery
۲۹۷	What is V8 JavaScript engine
۲۹۸	Why do we call javascript as dynamic language
۲۹۹	What is a void operator
۳۰۰	How to set the cursor to wait
۳۰۱	How do you create an infinite loop
۳۰۲	Why do you need to avoid with statement
۳۰۳	What is the output of below for loops
۳۰۴	List down some of the features of ES6
۳۰۵	What is ES6
۳۰۶	Can I redeclare let and const variables
۳۰۷	Is const variable makes the value immutable
۳۰۸	What are default parameters
۳۰۹	What are template literals
۳۱۰	How do you write multi-line strings in template literals
۳۱۱	What are nesting templates
۳۱۲	What are tagged templates
۳۱۳	What are raw strings
۳۱۴	What is destructuring assignment
۳۱۵	What are default values in destructuring assignment
۳۱۶	How do you swap variables in destructuring assignment
۳۱۷	What are enhanced object literals
۳۱۸	What are dynamic imports
۳۱۹	What are the use cases for dynamic imports

ردیف	سوال
۳۲۰	What are typed arrays
۳۲۱	What are the advantages of module loaders
۳۲۲	What is collation
۳۲۳	What is for...of statement
۳۲۴	What is the output of below spread operator array
۳۲۵	Is PostMessage secure
۳۲۶	What are the problems with postmessage target origin as wildcard
۳۲۷	How do you avoid receiving postMessages from attackers
۳۲۸	Can I avoid using postMessages completely
۳۲۹	Is postMessages synchronous
۳۳۰	What paradigm is Javascript
۳۳۱	What is the difference between internal and external javascript
۳۳۲	Is JavaScript faster than server side script
۳۳۳	How do you get the status of a checkbox
۳۳۴	What is the purpose of double tilde operator
۳۳۵	How do you convert character to ASCII code
۳۳۶	What is ArrayBuffer
۳۳۷	What is the output of below string expression
۳۳۸	What is the purpose of Error object
۳۳۹	What is the purpose of EvalError object
۳۴۰	What are the list of cases error thrown from non-strict mode to strict mode
۳۴۱	Is all objects have prototypes
۳۴۲	What is the difference between a parameter and an argument
۳۴۳	What is the purpose of some method in arrays
۳۴۴	How do you combine two or more arrays
۳۴۵	What is the difference between Shallow and Deep copy
۳۴۶	How do you create specific number of copies of a string

سؤال	رديف
How do you return all matching strings against a regular expression	٣٤٧
How do you trim a string at the beginning or ending	٣٤٨
What is the output of below console statement with unary operator	٣٤٩
Does javascript uses mixins	٣٥٠
What is a thunk function	٣٥١
What are asynchronous thunks	٣٥٢
What is the output of below function calls	٣٥٣
How to remove all line breaks from a string	٣٥٤
What is the difference between reflow and repaint	٣٥٥
What happens with negating an array	٣٥٦
What happens if we add two arrays	٣٥٧
What is the output of prepend additive operator on falsy values	٣٥٨
How do you create self string using special characters	٣٥٩
How do you remove falsy values from an array	٣٦٠
How do you get unique values of an array	٣٦١
What is destructuring aliases	٣٦٢
How do you map the array values without using map method	٣٦٣
How do you empty an array	٣٦٤
How do you rounding numbers to certain decimals	٣٦٥
What is the easiest way to convert an array to an object	٣٦٦
How do you create an array with some data	٣٦٧
What are the placeholders from console object	٣٦٨
Is it possible to add CSS to console messages	٣٦٩
What is the purpose of dir method of console object	٣٧٠
Is it possible to debug HTML elements in console	٣٧١
How do you display data in a tabular format using console object	٣٧٢
How do you verify that an argument is a Number or not	٣٧٣

سؤال	رديف
How do you create copy to clipboard button	٣٧٣
What is the shortcut to get timestamp	٣٧٤
How do you flattening multi dimensional arrays	٣٧٥
What is the easiest multi condition checking	٣٧٦
How do you capture browser back button	٣٧٧
How do you disable right click in the web page	٣٧٨
What are wrapper objects	٣٨٠
What is AJAX	٣٨١
What are the different ways to deal with Asynchronous Code	٣٨٢
How to cancel a fetch request	٣٨٣
What is web speech API	٣٨٤
What is minimum timeout throttling	٣٨٥
How do you implement zero timeout in modern browsers	٣٨٦
What are tasks in event loop	٣٨٧
What are microtasks	٣٨٨
What are different event loops	٣٨٩
What is the purpose of queueMicrotask	٣٩٠
How do you use javascript libraries in typescript file	٣٩١
What are the differences between promises and observables	٣٩٢
What is heap	٣٩٣
What is an event table	٣٩٤
What is a microTask queue	٣٩٥
What is the difference between shim and polyfill	٣٩٦
How do you detect primitive or non primitive value type	٣٩٧
What is babel	٣٩٨
Is Node.js completely single threaded	٣٩٩
What are the common use cases of observables	٤٠٠

ردیف	سوال
۴۰۱	What is RxJS
۴۰۲	What is the difference between Function constructor and function declaration
۴۰۳	What is a Short circuit condition
۴۰۴	What is the easiest way to resize an array
۴۰۵	What is an observable
۴۰۶	What is the difference between function and class declarations
۴۰۷	What is an async function
۴۰۸	How do you prevent promises swallowing errors
۴۰۹	What is deno
۴۱۰	How do you make an object iterable in javascript
۴۱۱	What is a Proper Tail Call
۴۱۲	How do you check an object is a promise or not
۴۱۳	How to detect if a function is called as constructor
۴۱۴	What are the differences between arguments object and rest parameter
۴۱۵	What are the differences between spread operator and rest parameter
۴۱۶	What are the different kinds of generators
۴۱۷	What are the built-in iterables
۴۱۸	What are the differences between for...of and for...in statements
۴۱۹	How do you define instance and non-instance properties
۴۲۰	تفاوت‌های بین Number.isNaN و NaN کدام است؟

۱. روش‌های ایجاد objects توی جاوااسکریپت چیا هستن؟

روش‌های زیادی برای ایجاد آبجکت‌ها توی جاوااسکریپت وجود داره:

۱. سازنده آبجکت:

ساده‌ترین راه برای ایجاد یه آبجکت خالی استفاده از سازنده آبجکت هستش. در حال حاضر این روش توصیه نمیشه

```
var object = new Object();
```

۲. متد ایجاد آبجکت:

متد ایجاد آبجکت با انتقال نمونه اولیه آبجکت به عنوان یه پارامتر، یه آبجکت جدید ایجاد میکنه

```
var object = Object.create(null);
```

:Object literal syntax .۳

The object literal syntax is equivalent to create method when it passes null as parameter

```
var object = {};
```

۴. سازنده تابع:

هر تابعی که بخوایم رو ایجاد میکنیم و از طریق عملگر new یه نمونه آبجکت جدید میسازیم.

```
function Person(name){  
    var object = {};  
    object.name=name;  
    object.age=21;  
    return object;  
}  
var object = new Person("Sudheer");
```

۵. سازنده تابع با نمونه اولیه:

شبیه سازنده تابع هستش اما از نمونه اولیه برای متدها و خصوصیاتشون استفاده میکنه

```
function Person(){}
Person.prototype.name = "Sudheer";
var object = new Person();
```

این معادل نمونه ای هستش که با متدهای ایجاد آبجکت با نمونه اولیه تابع ایجاد شده و تابع رو با یه نمونه و پارامترهاش به عنوان آرگومان فراخوانی میکنه.

```
function func {};
```

```
new func(x, y, z);
```

(یا)

```
// Create a new instance using function prototype.
var newInstance = Object.create(func.prototype)
```

```
// Call the function
var result = func.call(newInstance, x, y, z),
```

```
// If the result is a non-null object then use it otherwise just use the new instance.
console.log(result && typeof result === 'object' ? result : newInstance);
```

:ES6 Class syntax .۶

ES6 ویژگی های کلاس رو برای ایجاد آبجکت ها معرفی میکنه

```
class Person {  
    constructor(name) {  
        this.name = name;  
    }  
}  
  
var object = new Person("Sudheer");
```

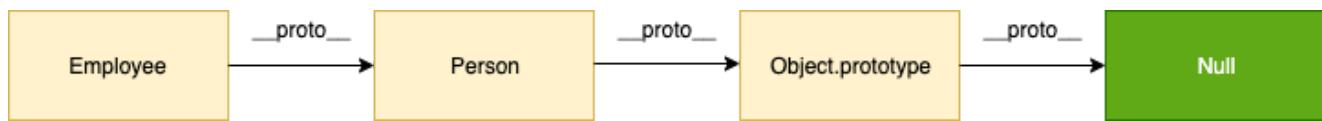
۷. الگوی Singleton

آبجکتیه که فقط یه بار قابل نمونه‌گیری هستش. فراخوانی‌های پی در پی با سازندهش همون نمونه رو برمی‌گردونه و اینطوری میشه مطمئن شد که به طور تصادفی نمونه‌های مختلف ایجاد نمیشه.

```
var object = new function(){
    this.name = "Sudheer";
}
```

۲. زنجیره prototype چیه؟

زنگیره prototype برای ساخت انواع جدیدی از آبجکت‌ها براساس موارد موجود استفاده میشه. این کار شبیه ارث بری توی یه زبان مبتنی بر کلاس هستش. روی نمونه آبجکت از طریق ویژگی `Object.getPrototypeOf(object)` یا `proto` در دسترسه در حالی که نمونه اولیه توی عملکرد سازنده‌ها از طریق `object.prototype` در دسترسه.



۳. تفاوت‌های بین Bind، Call، Apply چیا هستن؟

تفاوت بین `Bind`، `Call` و `Apply` توی مثالهای زیر توضیح داده شده
:**Call**: متده `call()` یه تابع با یه مقدار `this` و آرگومان‌های ارائه شده رو دونه فراخوانی میکنه

```
var employee1 = {firstName: 'John', lastName: 'Rodson'};
var employee2 = {firstName: 'Jimmy', lastName: 'Baily'};

function invite(greeting1, greeting2) {
    console.log(greeting1 + ' ' + this.firstName + ' ' + this.lastName+ ', ' + greeting2);
}

invite.call(employee1, 'Hello', 'How are you?'); // Hello John Rodson, How are you?
invite.call(employee2, 'Hello', 'How are you?'); // Hello Jimmy Baily, How are you?
```

Apply: تابع رو فراخوانی میکنه و بهمون اجازه میده تا آرگومان‌ها رو به عنوان یه آرایه منتقل کنیم

```
var employee1 = {firstName: 'John', lastName: 'Rodson'};
var employee2 = {firstName: 'Jimmy', lastName: 'Baily'};

function invite(greeting1, greeting2) {
    console.log(greeting1 + ' ' + this.firstName + ' ' + this.lastName+ ', ' + greeting2);
}

invite.apply(employee1, ['Hello', 'How are you?']); // Hello John Rodson, How are you?
invite.apply(employee2, ['Hello', 'How are you?']); // Hello Jimmy Baily, How are you?
```

bind: یه تابع جدید برمی‌گردونه، در حالی که بهمون اجازه میده هر تعداد آرگومانی که می خوایم رو توی یه آرایه منتقل کنیم

```

var employee1 = {firstName: 'John', lastName: 'Rodson'};
var employee2 = {firstName: 'Jimmy', lastName: 'Baily'};

function invite(greeting1, greeting2) {
    console.log(greeting1 + ' ' + this.firstName + ' ' + this.lastName + ', ' + greeting2);
}

var inviteEmployee1 = invite.bind(employee1);
var inviteEmployee2 = invite.bind(employee2);
inviteEmployee1('Hello', 'How are you?'); // Hello John Rodson, How are you?
inviteEmployee2('Hello', 'How are you?'); // Hello Jimmy Baily, How are you?

```

Call و Apply تقریباً قابل تعویض هستند. هر دو بلافاصله تابع فعلی را اجرا می کنند. شما باید تصمیم بگیرید که ارسال آرایه در آرایه آسان تر است یا فهرستی از آرگومان های جدا شده با کاما. می توانید به خاطر داشته باشید که Call برای کاما (فهرست جدا شده) و Apply برای Array است. در حالی که یک تابع جدید ایجاد می کند که «this» روی اولین پارامتر ارسال شده به «()» تنظیم می شود.

۴. فرمت JSON چیه و عملیات های معمول بر روی آن چیا هستند؟

JSON یه قالب داده مبتنی بر متن هستش که از نحو آبجکت جاوا اسکریپت (Javascript object syntax) پیروی میکنه و توسط Douglas Crockford رایج شد. کاربردش زمانیه که بخوایم داده ها رو از طریق شبکه انتقال بدیم و اساسا یه فایل متنی با پسوند json و نوع از application/json (native Object): تبدیل یه رشته به یه آبجکت محلی (Parsing)

```
JSON.parse(text)
```

رشته سازی: تبدیل یه آبجکت محلی به یه رشته تا بتونه از طریق شبکه منتقل بشه

```
JSON.stringify(object)
```

۵. هدف از متده slice روی آرایه ها چیه؟

متده slice عناصر انتخاب شده توی یه آرایه رو به عنوان یه آبجکت آرایه جدید برمی گردونه. این عناصر رو از اولین آرگومان داده شده انتخاب میکنه و با آرگومان پایانی و اختیاری داده شده بدون در نظر گرفتن آخرین عنصر به پایان میرسونه. اگه آرگومان دوم رو حذف کنیم تا آخر آرایه همه عناصر رو انتخاب میکنه. چند تا مثال در مورد این متده اینجا نوشته شده

```

let arrayIntegers = [1, 2, 3, 4, 5];
let arrayIntegers1 = arrayIntegers.slice(0,2); // returns [1,2]
let arrayIntegers2 = arrayIntegers.slice(2,3); // returns [3]
let arrayIntegers3 = arrayIntegers.slice(4); //returns [5]

```

نکته: متده slice آرایه اصلی رو تغییر نمیده ولی یه زیرمجموعه به عنوان یه آرایه جدید برمی گردونه

۶. هدف از متدهای splice و slice چیه؟

متدهای **splice** برای اضافه کردن به آرایه یا حذف از اون استفاده میشه و مورد حذف شده رو برمیگردونه. آرگومان اول موقعیت آرایه رو برای درج یا حذف مشخص میکنه در حالی که آرگومان اختیاری دوم تعداد عناصر حذف شده رو مشخص میکنه. هر آرگومان اضافهای به آرایه اضافه میشه. چند تا مثال در این مورد اینجا نوشته شده.

```
let arrayIntegersOriginal1 = [1, 2, 3, 4, 5];
let arrayIntegersOriginal2 = [1, 2, 3, 4, 5];
let arrayIntegersOriginal3 = [1, 2, 3, 4, 5];

let arrayIntegers1 = arrayIntegersOriginal1.splice(0,2); // returns [1, 2]; original array: [3, 4, 5]
let arrayIntegers2 = arrayIntegersOriginal2.splice(3); // returns [4, 5]; original array: [1, 2, 3]
let arrayIntegers3 = arrayIntegersOriginal3.splice(3, 1, "a", "b", "c"); //returns [4];
original array: [1, 2, 3, "a", "b", "c", 5]
```

نکته: متدهای **slice** و **splice** اصلی را اصلاح میکنند و آرایه حذف شده رو برمیگردونند.

۷. تفاوت متدهای splice و slice چیا هستن؟

بعضی از تفاوت‌های عمده تویی به جدول

Splice	Slice
آرایه اصلی رو تغییر میده (تغییر پذیر)	آرایه اصلی رو تغییر نمیده (تغییرنپذیر)
عناصر حذف شده رو به عنوان آرایه برمیگردونه	زیرمجموعه آرایه اصلی رو برمیگردونه
برای درج عناصر به آرایه یا حذف از اون استفاده میشه	برای انتخاب عناصر از آرایه استفاده میشه

۸. تفاوت‌های Map و Object چیا هستن؟

آبجکت‌ها شبیه به نقشه‌ها (**Maps**) هستن از این جهت که هردو بهمون این امکتن رو میدن که کلیدها رو روی مقادیر تنظیم کنیم، مقادیر رو بازیابی کنیم، کلیدها رو حذف کنیم و بینیم چیزی تویی به کلید ذخیره شده یا نه. به همین دلیل از آبجکت‌ها به عنوان نقشه‌ها (**Maps**) در طول تاریخ استفاده شده. اما تفاوت‌های مهمی وجود داره که استفاده از نقشه (**map**) رو تویی موارد خاص ترجیح میده.

۱. کلیدهایی به آبجکت رشته‌ها و نمادها هستن، در حالی که برای نقشه مقادیر مختلفی میتونه وجود داشته باشه که شامل توابع، آبجکت‌ها و هر نوع اولیه دیگه‌ای میشه.

۲. کلیدهای نقشه (**map**) مرتب میشن در حالی که کلیدهای اضافه شده به آبجکت اینطوری نیستن. بنابراین موقع تکرار روی اون، کلیدها رو به ترتیب اضافه شدنشون برمیگردونه.

۳. اندازه **map** رو میتوانیم به راحتی با ویژگی سایز بدست بیاریم، در حالی که تعداد خصوصیات یه آبجکت باید به صورت دستی حساب بشه.

۴. نقشه قابل تکراره و میتوانه مستقیما تکرار بشه، در حالی که آبجکت مستلزم بدست آوردن کلیدهای اون به روشنی خاص و تکرار روی اونهاست.

۵. آبجکت یه نمونه اولیه (prototype) داره، بنابراین کلیدهای پیشفرض توی map وجود داره که اگه دقت نکنیم ممکنه با کلیدهایی بروخورد کنه. از زمان ES5 میتوانیم با استفاده از null (map = Object.create(null)), این قضیه رو دور بزنیم ولی بهندرت این کار انجام میشه.
۶. نقشه ممکنه توی سناریوهای شامل جمع و حذف مکرر جفت کلیدها عملکرد بهتری داشته باشه

۹. تفاوت‌های بین عملگرهای == و === چیا هستن؟

- جاوا اسکریپت مقایسه برابری سخت (=, !) و تبدیل نوع (==, !=) را فراهم می‌کند. عملگرهای سختگیرانه نوع متغیر را در نظر می‌گیرند، در حالی که عملگرهای غیر دقیق، اصلاح/تبدیل نوع را بر اساس مقادیر متغیرها انجام می‌دهند. اپراتورهای سختگیر از شرایط زیر برای انواع مختلف پیروی می‌کنند.
۱. دو رشته زمانی کاملاً برابر هستند که توالی کاراکترهای یکسان، طول یکسان و کاراکترهای مشابه در موقعیت‌های متناظر داشته باشند.
 ۲. دو عدد زمانی که از نظر عددی مساوی باشند کاملاً برابر هستند. یعنی داشتن مقدار عددی یکسان. دو مورد خاص در این مورد وجود دارد،
 ۳. NaN با هیچ چیز از جمله NaN برابر نیست.
 ۴. صفرهای مثبت و منفی با هم برابرند.
 ۵. اگر هر دو درست یا نادرست باشند، دو عملوند بولی کاملاً برابر هستند.
 ۶. اگر دو شیء به یک شیء اشاره کنند کاملاً برابر هستند.
 ۷. انواع Null و Undefined با === برابر نیستند، بلکه با == برابر هستند. یعنی null undefined --> true اما null undefined --> false
- برخی از مثال‌هایی که موارد فوق را پوشش می‌دهد،

```
0 == false    // true
0 === false   // false
1 == "1"      // true
1 === "1"     // false
null == undefined // true
null === undefined // false
'0' == false // true
'0' === false // false
[] == [] or [] === [] // false, refer different objects in memory
{} == {} or {} === {} // false, refer different objects in memory
```

۱۰. توابع arrow-function یا lambda چی هستن؟

- های arrow function به صورت ساده‌تر و کوتاه‌تر تعریف می‌شن و new.target یا this، arguments، super بدون متدهستند و به عنوان سازنده یا constructor استفاده نمی‌شن.

۱۱. یه تابع first-class چجور تابعیه؟

توى جاوااسكريپت، توابع آبجکت‌های کلاس اول یا first class هستن. توابع کلاس اول زمانی معنی میدن که توابع توی اون زبان باهاشون مثل بقیه متغیرها رفتار بشه.

به عنوان مثال، توی چنین زبانی، یه تابع میتونه به عنوان آرگومان به یه تابع دیگه انتقال داده بشه، میتونه به عنوان مقدار نهایی یه تابع دیگه برگشت داده بشه و میتونه به یه متغیر دیگه به عنوان مقدار اختصاص داده بشه. به عنوان مثال توی کد زیر، توابع نگهدارنده یا handler به یه شنونده یا listener اختصاص داده شده.

```
const handler = () => console.log ('This is a click handler function');
document.addEventListener ('click', handler);
```

۱۲. یه تابع first-order چجور تابعیه؟

تابع مرتبه اول یا first-order تابعیه که هیچ تابع دیگه‌ای رو به عنوان آرگومان قبول نمیکنه و هیچ تابعی رو هم به عنوان مقدار برگشتی یا return value برنمی‌گردونه.

```
const firstOrder = () => console.log ('I am a first order function!');
```

۱۳. یه تابع higher-order چجور تابعیه؟

تابع مرتبه بالا توابعی هستن که یه تابع رو به عنوان پارامتر ورودی دریافت و یا به عنوان خروجی ارسال میکنن.

```
const firstOrderFunc = () => console.log ('Hello I am a First order function');
const higherOrder = ReturnFirstOrderFunc => ReturnFirstOrderFunc ();
higherOrder (firstOrderFunc);
```

۱۴. یه تابع unary چجور تابعیه؟

تابع unary تابعیه که فقط یه آرگومان ورودی دریافت میکنه. بیاین یه مثال از توابع unary بزنیم.

```
// Add 10 to the given argument and display the value
const unaryFunction = a => {
  console.log (a + 10);
};
```

۱۵. توابع يعني currying چی؟

به فرایندی که در اون یه تابع با چندین آرگومان رو به مجموعه‌ای از توابع که فقط یه آرگومان دریافت میکنن، تبدیل کنیم Currying از نام یه ریاضی‌دان به اسم Haskell Curry گرفته شده. با استفاده از Currying در واقع یه تابع رو به یه تابع unary تبدیل میکنیم. بیاین یه مثال از یه تابع با چندین آرگومان و تبدیلش به تابع currying بزنیم.

```

const multiArgFunction = (a, b, c) => a + b + c;
const curryUnaryFunction = a => b => c => a + b + c;
curryUnaryFunction (1); // returns a function: b => c => 1 + b + c
curryUnaryFunction (1) (2); // returns a function: c => 3 + c
curryUnaryFunction (1) (2) (3); // returns the number 6

```

توابع Curried برای بهبود قابلیت استفاده مجدد کد و ترکیب عملکردی عالی هستند.

۱۶. چه توابع pure هستند؟

تابع خالص تابعی که مقدار برگشتیش فقط توسط آرگومان‌هاش تعیین می‌شود بدون هیچ side effect یا عوارض جانبی. به عنوان مثال اگه ما یه تابع رو `n` بار در `n` جای مختلف برنامه فراخوانی کنیم همیشه یه مقدار مشخص برگشت داده می‌شود. بیاین یه مثال از تفاوت بین توابع خالص و توابع ناخالص بزنیم.

```

//Impure
let numberArray = [];
const impureAddNumber = number => numberArray.push (number);
//Pure
const pureAddNumber = number => argNumberArray =>
    argNumberArray.concat ([number]);

//Display the results
console.log (impureAddNumber (6)); // returns 1
console.log (numberArray); // returns [6]
console.log (pureAddNumber (7) (numberArray)); // returns [6, 7]
console.log (numberArray); // returns [6]

```

بر اساس تکه کدهای بالا، تابع `push` با تغییر روی آرایه و برگرداندن شماره ایندکس `push` که مستقل از مقدار پارامتر هستش، یه تابع ناخالص به حساب می‌آید. در حالی که از یه طرف متده `concat` آرایه رو می‌گیره و اونو با یه آرایه دیگه ترکیب می‌کنه و یه آرایه کاملاً جدید و بدون هیچ عوارض جانبی تولید می‌کنه. همچنین مقدار برگشتی با آرایه قبلی ترکیب شده هستش. به یاد داشته باشید که توابع خالص مهم هستند زیرا آنها تست واحد را بدون هیچ گونه عوارض جانبی و بدون نیاز به تزریق وابستگی ساده می‌کنند. آنها همچنین از اتصال محکم جلوگیری می‌کنند و با نداشتن عوارض جانبی، شکستن برنامه شما را سخت تر می‌کنند. این اصول در کنار هم قرار می‌گیرند **تغییرناپذیری** مفهوم ES6 با ارجحیت به `const` نسبت به استفاده `let`.

۱۷. هدف از کلمه کلیدی let چیه؟

دستور `let` یه **متغیر محلی** **block scope** تعریف می‌کند. از این رو متغیرهایی که با کلمه کلیدی `let` تعریف می‌شون محدود به همون اسکوپی که تو شعری شدن، دستورها و عبارتهاي توی همون اسکوپ می‌شون. درحالی که متغیرهای تعریف شده با کلمه کلیدی `var` برای تعریف یه متغیر توی سطح `global` یا محلی برای استفاده در کل توابع بدون در نظر گرفتن اسکوپی که تو شعری شده، استفاده می‌شون. بیاین برای نشون دادن کاربردش یه مثال بزنیم.

```

let counter = 30;
if (counter === 30) {
    let counter = 31;
    console.log(counter); // 31
}
console.log(counter); // 30 (because if block variable won't exist here)

```

۱۸. تفاوت‌های کلمات کلیدی let و var چیا هستن؟

تفاوت‌ها رو توی جدول زیر می‌بینیم

: var

۱. از ابتدای جاوااسکریپت در دسترس هستش
۲. دامنه تابع داره
۳. متغیرها Hoist می‌شن

:let

۱. به عنوان بخشی از ES6 معرفی شده

۲. محدود به scope یا دامنه هستش

۳. شده ولی مقداردهی اولیه نمی‌شه

بیاین با یه مثال تفاوت‌ش رو بهتر ببینیم

```

function userDetails(username) {
    if(username) {
        console.log(salary); // undefined(due to hoisting)
        console.log(age); // error: age is not defined
        let age = 30;
        var salary = 10000;
    }
    console.log(salary); //10000 (accessible to due function scope)
    console.log(age); //error: age is not defined(due to block scope)
}

```

۱۹. دلیل انتخاب کلمه کلیدی let چیه؟

Let یه عنوان ریاضی هستش که توسط زبان‌های برنامه‌نویسی اولیه مثل Basic و Scheme پذیرفته شده. این زبان از دهها زبان دیگه گرفته شده که از let به عنوان یه کلمه کلیدی سنتی تا حد ممکن نزدیک به var استفاده می‌کنه.

۲۰. چطوری می‌تونیم توی بلوک مربوط به switch بدون دریافت خطای متغیر تعریف کنیم؟

اگه بخوایم متغیرها رو مجدداً توی یه switch block تعریف کنیم، این کار باعث خطای می‌شه چون در واقع فقط یه بلاک وجود داره. به عنوان مثال توی کد زیر یه خطای نحوی ایجاد می‌شه

```

let counter = 1;
switch(x) {
    case 0:
        let name;
        break;

    case 1:
        let name; // SyntaxError for redeclaration.
        break;
}

```

برای جلوگیری از این خطأ، میتوانیم یه بلاک تو داخل case ایجاد کنیم و یه محیط واژگانی دارای محدوده بلاک جدید ایجاد کنیم.

```

let counter = 1;
switch(x) {
    case 0: {
        let name;
        break;
    }
    case 1: {
        let name; // No SyntaxError for redeclaration.
        break;
    }
}

```

۲۱. چیه؟ Temporal-Dead-Zone

رفتاری توی جاوااسکریپت که هنگام تعریف متغیر با کلمات کلیدی let و const رخ میده، نه با کلمه کلیدی var. توی اکمااسکریپت ۶، دستیابی به متغیر let و const قبل از تعریفش (توی scope خودش) باعث خطا میشه. فاصله زمانی ایجاد اون، بین ایجاد اتصال متغیر و تعریف اون، منطقه refrence هستش. بیاین با یه مثال ببینیم،

```

function somemethod() {
    console.log(counter1); // undefined
    console.log(counter2); // ReferenceError
    var counter1 = 1;
    let counter2 = 2;
}

```

۲۲. (تابع بلافاصله صدا زده شده) چی هستن؟ IIFE

IIFE (فراخوانی عملکرد بلافاصله) یه تابع جاوااسکریپت که به محض تعریف اجرا میشه. امضای اون به این صورته،

```
(function ()
{
// logic here
}
());
()

console.log(message); //Error: message is not defined
```

۲۳. مزایای استفاده از module‌ها چیه؟

- استفاده از مازول‌ها مزایای زیادی دارد،
۱. قابلیت نگهداری
 ۲. قابلیت استفاده مجدد
 ۳. نامگذاری

۲۴. Memoization چیه؟

Memoization یه روش برنامه‌نویسی هست که سعی داره با ذخیره نتایج قبلی یه تابع عملکرد اون تابع رو افزایش بده. هر بار که یه تابع Memoize شده فراخوانی میشه، پارامترهای اون cache میشه یعنی توی حافظه پنهان ذخیره میشه. اگه داده وجود داشته باشه، بدون اجرای کل تابع میشه اونو برگرداند در غیر این صورت تابع اجرا میشه و بعدش نتیجه توی حافظه پنهان ذخیره میشه.

بیاین یه مثال از نوشتن یه تابع با Memoization بزنیم،

```

const memoizAddition = () => {
  let cache = {};
  return (value) => {
    if (value in cache) {
      console.log('Fetching from cache');
      return cache[value]; // Here, cache.value cannot be used as property name
    }
    else {
      console.log('Calculating result');
      let result = value + 20;
      cache[value] = result;
      return result;
    }
  }
}

// returned function from memoizAddition
const addition = memoizAddition();
console.log(addition(20)); //output: 40 calculated
console.log(addition(20)); //output: 40 cached

```

۲۵. چیه؟ Hoisting

Hoisting یه مکانیسم جاوااسکریپت که متغیرها و تعاریف توابع رو به بالای scope یا دامنه خودشون انتقال میده. یادمون باشه که جاوااسکریپت فقط تعریف متغیرها و توابع رو Hoist میکنه، نه مقداردهی اولیه اوتا رو. بیاین یه مثال ساده از hoist کردن متغیرها بزنیم،

```

console.log(message); //output : undefined
var message = 'The variable Has been hoisted';

```

ترجمه کد بالا اینطوری میشه،

```

var message;
console.log(message);
message = 'The variable Has been hoisted';

```

۲۶. چی هستن؟ ES6 Class ها توی

در ES6، کلاس‌های جاوا اسکریپت عمدتاً قند نحوی بر وراثت مبتنی بر نمونه اولیه جاوا اسکریپت است. به عنوان مثال، وراثت مبتنی بر نمونه اولیه که در عبارت تابع به صورت زیر نوشته شده است.

```

function Bike(model,color) {
    this.model = model;
    this.color = color;
}

Bike.prototype.getDetails = function() {
    return this.model + ' bike has' + this.color + ' color';
};

```

در حالیکه کلاس‌های ES6 به جای اونا و به شکل ساده‌تری تعریف یشن:

```

class Bike{
    constructor(color, model) {
        this.color= color;
        this.model= model;
    }

    getDetails() {
        return this.model + ' bike has' + this.color + ' color';
    }
}

```

.۲۷ Closure چیا هستن؟

کلاژور ترکیبی از یه تابع و محیط واژگانی هستش که تابع در اون تعریف شده. به عنوان مثال این یه تابع داخلیه که به متغیرهای تابع خارجی دسترسی داره. کلاژور دارای سه زنجیره دامنه هستش

۱. دامنه رو جایی تعریف می کنیم که متغیرها بین کرلي براكت‌های اون تعریف شده باشه

۲. متغیرهای تابع بیرونی

۳. متغیرهای محلی

بیاین یه مثال راجع به مفهوم کلاژور بزنیم

```

function Welcome(name){
    var greetingInfo = function(message){
        console.log(message+' '+name);
    }
    return greetingInfo;
}
var myFunction = Welcome('John');
myFunction('Welcome '); //Output: Welcome John
myFunction('Hello Mr.') //output: Hello Mr.John

```

مطابق کد بالا، تابع داخلی (greetingInfo) حتی بعد از بازگشت تابع خارجی به متغیرهای محدوده تابع خارجی (welcome) دسترسی داره.

.۲۸ Module چیا هستن؟

ماژول‌ها به واحدهای کوچیکی از کد مستقل و قابل استفاده مجدد اشاره می‌کنند و همچنین به عنوان پایه بسیاری از الگوهای طراحی javascript عمل می‌کنند. خروجی ماژول‌های javascript یه شی، یه تابع یا constructor هستش.

۲۹. چرا به module‌ها نیاز داریم؟

لیستی از مزایای استفاده از ماژول‌ها در اکوسیستم جاوااسکریپت اینجا گفته شده

۱. قابلیت نگهداری

۲. قابلیت استفاده مجدد

۳. نامگذاری

۳۰. توی جاوااسکریپت scope چیه و چیکار می‌کنه؟

scope یا محدوده، دسترسی متغیرها، توابع و اشیاء در بعضی از قسمت‌های کدمون در زمان اجرا هستش. به عبارت دیگه، دامنه قابلیت دیده شدن متغیرها و بقیه منابع رو تو قسمت‌هایی از کدمون تعیین می‌کنه.

۳۱. چیه service-worker؟

اساساً یه اسکریپت هستش که جدا از یه صفحه وب توی پس‌زمینه اجرا می‌شه و ویژگی‌هایی رو فراهم می‌کنه که نیازی به صفحه وب یا تعامل کاربر نداره. بعضی از ویژگی‌های عمدہ service worker ها عبارتند از: تجارب غنی آفلاین (اولین برنامه آفلاین وب)، همگام‌سازی دوره‌ای پس‌زمینه، push notification، رهگیری و رسیدگی به درخواست‌های شبکه و مدیریت برنامه‌ای cache response ها.

۳۲. توی چطوری می‌شه service-worker رو دستکاری کرد؟

مستقیماً نمی‌توونه به DOM دسترسی پیدا کنه، اما می‌توانه با پاسخ به پیام‌های ارسالی از طریق رابط postMessage با صفحاتی که کنترل می‌کنه ارتباط برقرار کنه و این صفحات می‌توانن DOM رو دستکاری کنن.

۳۳. چطوری می‌تونیم بین ریست شدن‌های service-worker داده‌های مورد نظرمون رو مجدد استفاده کنیم؟

مشکلی که توی service worker وجود داره اینه که در صورت عدم استفاده خاتمه پیدا می‌کنه و در صورت نیاز بعدی دوباره راه‌اندازی می‌شه، بنابراین نمی‌توانیم به state های سراسری توی نگهدارنده‌های onmessage و onfetch یه indexedDB API اعتماد کنیم. تو این حالت service worker برای تداوم کار و استفاده مجدد موقع شروع مجدد، به دسترسی خواهد داشت.

۳۴.IndexedDB چیه؟

یه API سطح پایین برای ذخیره client-side یا سمت کاربر توی مقادیر بیشتری از داده ساخت یافته شامل فایل‌ها و جباب‌ها هستش. این API از index‌ها برای فعال کردن جستجوهای با کارایی بالا توی این داده‌ها استفاده میکنه.

۳۵. Web-storage چیه؟

یه API هستش که مکانیسمی رو فراهم میکنه که مرورگرها میتونن مقدار و کلید رو ب صورت محلی توی مرورگر کاربر ذخیره کنن، ب روشنی کاملا قابل درک نسبت به استفاده ا کوکی‌ها. فضای ذخیره‌سازی وب دو مکانیزم را برای ذخیره اطلاعات روی مشتری فراهم میکنه.

۱. **Local storage**: داده‌ها رو برای مبداء فعلی و بدون تاریخ انقضا ذخیره میکنه.

۲. **Session storage**: داده‌ها رو برای یه جلسه ذخیره میکنه و با بسته شدن تب مرورگر داده‌ها از بین میرن.

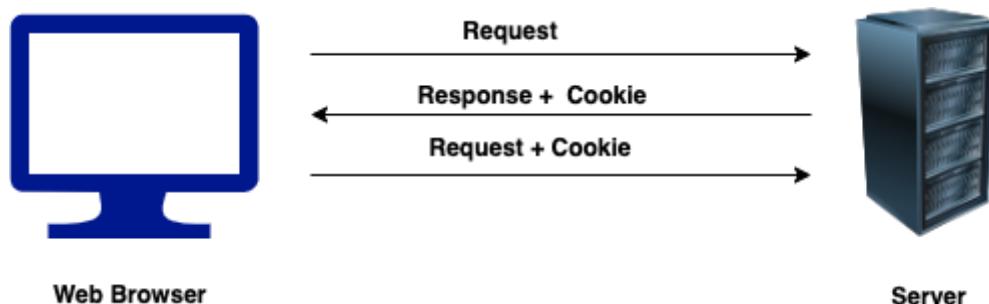
۳۶. Post-message چیه؟

روشنی هست که امکان ایجاد ارتباط متقابل بین آبجکت‌های window رو فراهم میکنه (به عنوان مثال، بین یه صفحه و یه پنجره بازشو که باعث ایجاد اون شده، یا بین یه صفحه و یه iframe جاسازی شده در اون) به طور کل اسکریپت‌های موجود در صفحات مختلف مجاز ب دسترسی به هم‌دیگه هستن، تنها در صورتی که صفحات از خط مشی یکسانی تبعیت کنن. (یعنی صفحات از پروتکل، شماره پورت و میزبان یکسانی برخوردار هستن)

۳۷. Cookie چیه؟

کوکی قطعه‌ای از داده هستش که توی کامپیوتراون ذخیره میشه تا مرورگر به اون دسترسی داشته باشه. کوکی‌ها به عنوان جفت‌های کلید و مقدار ذخیره میشن.
به عنوان مثال میتونیم یه کوکی با نام کاربری مثل زیر ایجاد کنیم،

```
document.cookie = "username=John";
```



۳۸. چرا به cookie نیاز داریم؟

از کوکی ها برای به خاطر سپردن اطلاعات مربوط به مشخصات کاربر (مانند نام کاربری) استفاده میشه. در اصل شامل دو مرحله هستش،

۱. وقتی که کاربر از یه صفحه وب بازدید میکنه ، مشخصات کاربر میتونه توی یه کوکی ذخیره بشه.
۲. دفعه بعد که کاربر از صفحه بازدید کرد ، کوکی مشخصات کاربر رو به خاطر میاره.

۳۹. گزینه های قابل تنظیم توی cookie چیا هستن؟

گزینه های زیر برای کوکی موجوده ،

۱. به طور پیش فرض ، کوکی موقع بسته شدن مرورگر حذف میشه اما با تنظیم تاریخ انقضا (به وقت UTC) می تونیم این رفتار رو تغییر بدیم.

```
document.cookie = "username=John; expires=Sat, 8 Jun 2019 12:00:00 UTC";
```

۱. به طور پیش فرض ، کوکی به صفحه فعلی تعلق دارد. اما با استفاده از پارامتر path می تونیم به مرورگر بگیم که کوکی متعلق به چه مسیری هستش.

```
document.cookie = "username=John; path=/services";
```

۴۰. چطوری میشه یه cookie رو حذف کرد؟

با تنظیم تاریخ انقضا به عنوان تاریخ گذشته می تونیم کوکی رو حذف کنیم. تو این حالت نیازی به تعیین مقدار کوکی نیست. به عنوان مثال ، می تونیم کوکی نام کاربری رو توی صفحه فعلی به صورت زیر حذف کنیم.

```
document.cookie = "username=; expires=Fri, 07 Jun 2019 00:00:00 UTC; path=/";
```

نکته برای اطمینان از نحوه درست پاک کردن کوکی باید گزینه مسیر کوکی رو تعیین کنیم. بعضی از مرورگرهای زمانی که پارامتر مسیر رو تعیین نکنیم اجازه حذف کوکی رو نمیدن.

۴۱. تفاوت های بین session-storage و cookie، local-storage چیا هستن؟

تفاوت های بین کوکی، لوکال استوریج و سشن استوریج اینها هستند:

Session storage	Local storage	Cookie	ویژگی
فقط سرویس گیرنده	فقط سرویس گیرنده	هردو سرویس دهنده و سرویس گیرنده	قابل دسترسی از طرف سرویس گیرنده یا سرور
تا وقتی که تب بسته نشده	تا وقتی که حذف بشه	پیکربندی شده با استفاده از گزینه اکسپلایر	طول عمر
پشتیبانی نمیشه	پشتیبانی نمیشه	پشتیبانی می شود	پشتیبانی از SSL

Session storage	Local storage	Cookie	ویژگی
5MB	5MB	4KB	حداکثر اندازه داده

۴۲. تفاوت‌های بین sessionStorage و localStorage چیا هستن؟

لوکال استوریج همان سشن استوریج هستش اما داده‌ها با بستن و دوباره باز کردن مرورگر همچنان حفظ می‌شه (تاریخ انقضا نداره) در حالی که سشن استوریج داده‌ها رو با بستن پنجره مرورگر پاک می‌کنه.

۴۳. چطوری به web-storage دسترسی پیدا می‌کنی؟

شی window به ترتیب ویژگی‌های WindowSessionStorage و WindowLocalStorage را که دارای ویژگی‌های sessionStorage (window.sessionStorage) و localStorage (window.localStorage) هستن رو پشتیبانی می‌کنه. این خصوصیات نمونه‌ای از شی Storage رو ایجاد می‌کنه که از طریق اون می‌شه موارد داده رو برای یه دامنه خاص و نوع ذخیره سازی (session یا محلی) تنظیم، بازیابی و حذف کرد.
به عنوان مثال، می‌توانیم روی اشیای ذخیره سازی محلی مثل زیر بخونیم و بنویسیم

```
localStorage.setItem('logo', document.getElementById('logo').value);
localStorage.getItem('logo');
```

۴۴. چه متدهایی روی sessionStorage قابل استفاده هستن؟

متدهایی رو برای خواندن، نوشتن و پاکسازی داده‌های session storage ارائه میده.

```
// Save data to sessionStorage
localStorage.setItem('key', 'value');

// Get saved data from sessionStorage
let data = sessionStorage.getItem('key');

// Remove saved data from sessionStorage
localStorage.removeItem('key');

// Remove all saved data from sessionStorage
localStorage.clear();
```

۴۵. رخداد storage چیه و چطوری ازش استفاده می‌کنیم؟

رویدادی هستش که با تغییر مکان ذخیره سازی در متن سند دیگه‌ای فعال می‌شه. در حالی که خاصیت ذخیره سازی یک EventHandler برای پردازش رویدادهای ذخیره سازی است.

```
window.onstorage = functionRef;
```

بیاین به عنوان مثال استفاده از رویداد onstorage را بینیم که کلید ذخیره و مقادیر او را ثبت می‌کند

```
window.onstorage = function(e) {
    console.log('The ' + e.key +
    ' key has been changed from ' + e.oldValue +
    ' to ' + e.newValue + '.');
};
```

۴۶. چرا به web-storage نیاز داریم؟

فضای ذخیره سازی وب از امنیت بیشتری برخورداره و مقدار زیادی داده می‌توان به صورت محلی ذخیره کنن، بدون اینکه روی عملکرد وب سایت تأثیر بذارن. همچنین، اطلاعات هرگز به سرور منتقل نمی‌شون. به همین دلیل این روش نسبت به کوکی‌ها بیشتر توصیه می‌شود.

۴۷. چطوری می‌توانیم پشتیبانی از web-storage توسط مرورگر را بررسی کنیم؟

قبل از استفاده از فضای ذخیره سازی وب، باید پشتیبانی مرورگر را برای sessionStorage و localStorage بررسی کنیم.

```
if (typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

۴۸. چطوری می‌توانیم پشتیبانی از web-worker توسط مرورگر را بررسی کنیم؟

قبل از استفاده، باید پشتیبانی مرورگر را برای web worker ها بررسی کنیم

```
if (typeof(Worker) !== "undefined") {
    // code for Web worker support.
} else {
    // Sorry! No Web Worker support..
}
```

۴۹. یه مثال از web-worker ها می‌تونی بزنی؟

برای شروع استفاده از web worker ها برای مثال شمارنده باید مراحل زیر را در نظر بگیریم:
۱. ساخت یه فایل Web Worker: برای افزایش مقدار شمارش، باید یه اسکریپت بنویسیم. بیاین اسمشو counter.js باشد.

```

let i = 0;

function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()", 500);
}

timedCount();

```

اینجا از روش `postMessage()` برای ارسال پیام به صفحه HTML استفاده میشے
۱. ایجاد شی Web Worker: با بررسی پشتیبانی مرورگر میتوانیم یه شی Web Worker ایجاد کنیم. بیاین اسم این فایل رو
بذرایم.

```

if (typeof(w) == "undefined") {
    w = new Worker("counter.js");
}

```

و ما میتوانیم پیامها رو از web worker دریافت کنیم

```

w.onmessage = function(event){
    document.getElementById("message").innerHTML = event.data;
};

```

۱. به پایان رساندن یه web Worker
۲. ها تا زمان خاتمه یافتن پیامها (حتی بعد از اتمام یه اسکریپت خارجی) به گوش دادن ادامه میدن. برای خاتمه
دادن به گوش دادن به پیامها میتوانیم از دستور `terminate()` استفاده کنیم.

```
w.terminate();
```

۳. استفاده مجدد از worker رو : اگه متغیر `worker` رو `undefined` یا تعریف نشده تنظیم کنیم، میتوانیم از کد استفاده
مجدد کنیم.

```
w = undefined;
```

۵۰. محدودیتهای web-worker روی DOM چیا هستن؟

WebWorker ها به اشیا جاوااسکریپت دسترسی ندارن چون توی یه فایل خارجی تعریف شدن.
۱. Window object.
۲. Document object.
۳. Parent object.

۵۱. Promise چیه؟

یک promise آبجکتی است که ممکن است در آینده یک مقدار واحد را با یک مقدار حل شده یا به دلیل حل نشدن آن (مثلاً خطای شبکه) تولید کند. در یکی از ۳ حالت ممکن خواهد بود: انجام شده، رد شده، یا در انتظار. برای ساختن Promise ها از سینتکس زیر استفاده میشه

```
const promise = new Promise(function(resolve, reject) {
  // promise description
})
```

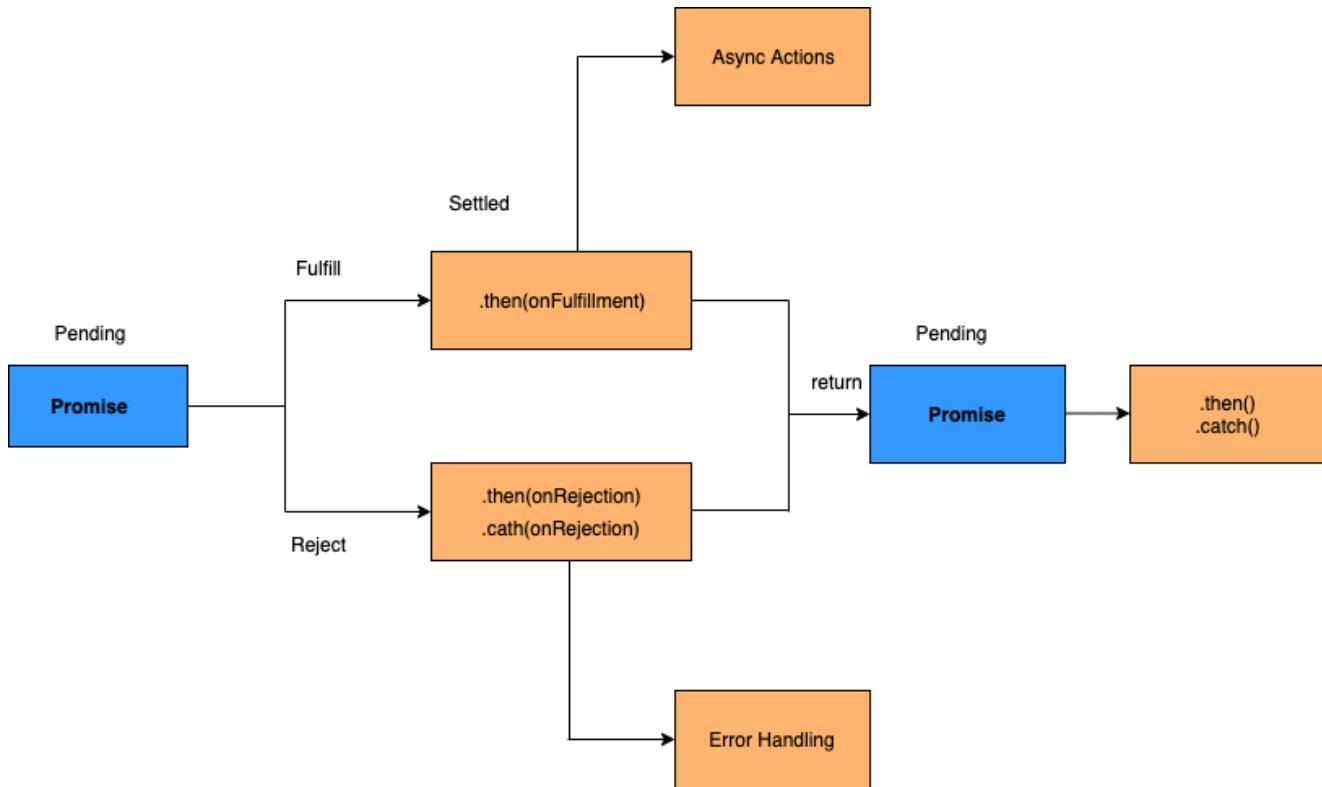
استفاده از پرمیس ها رو هم ببینیم

```
const promise = new Promise(resolve => {
  setTimeout(() => {
    resolve("I'm a Promise!");
  }, 5000);
}, reject => {

});

promise.then(value => console.log(value));
```

جريان عمل یک پرمیس به صورت زیر خواهد بود:



۵۲. چرا به promise نیاز داریم؟

Promises برای رسیدگی به عملیات ناهمزمان استفاده می شود. آنها با کاهش جهنم برگشت تماس و نوشتن کد پاک کننده، یک رویکرد جایگزین برای تماس های برگشتی ارائه می کنند.

۵۳. سه تا وضعیت ممکن برای یه promise چیا هستن؟

پرامیس ها ها سه حالت دارند:

۱. **Pending**: این حالت اولیه Promise قبل از شروع عملیات است

۲. **Fulfilled**: این حالت نشان می دهد که عملیات مشخص شده تکمیل شده است.

۳. **Rejected**: این حالت نشان می دهد که عملیات کامل نشده است. در این حالت یک مقدار خطأ داده خواهد شد.

۵۴. توابع چی هستن؟

تابع callback تابعی است که به عنوان آرگومان به تابع دیگری منتقل می شود. این تابع در داخل تابع خارجی برای تکمیل یک عمل فراخوانی می شود.

بیایید یک مثال ساده از نحوه استفاده از تابع callback بزنیم

```
function callbackFunction(name) {  
    console.log('Hello ' + name);  
}  
  
function outerFunction(callback) {  
    let name = prompt('Please enter your name.');//  
    callback(name);  
}  
  
outerFunction(callbackFunction);
```

۵۵. چرا به توابع callback نیاز داریم؟

callback ها مورد نیاز هستند زیرا جاوا اسکریپت یک زبان رویداد محور است. این بدان معناست که به جای منتظر ماندن برای پاسخ، جاوا اسکریپت در حین گوش دادن به رویدادهای دیگر اجرا می شود.

بیایید مثالی را با اولین تابع callback یک API (شبیه‌سازی شده توسط setTimeout) و تابع بعدی که پیام را ثبت می‌کند، بیاوریم.

```
function firstFunction(){  
    // Simulate a code delay  
    setTimeout( function(){  
        console.log('First function called');  
    }, 1000 );  
}  
function secondFunction(){  
    console.log('Second function called');  
}  
firstFunction();  
secondFunction();  
  
// Output :  
// Second function called  
// First function called
```

همانطور که از خروجی مشاهده می شود، جاوا اسکریپت منتظر پاسخ اولین تابع نبود و بلوک کد باقی مانده اجرا شد. بنابراین از callback ها به گونه ای استفاده می شود تا مطمئن شویم که کد خاصی تا زمانی که اجرای کد دیگر به پایان نرسد اجرا نمی شود.

۵۶.Callback-hell یا جهنم توابع چیه؟

یک ضد الگو با چندین تماس تو در تو است که خواندن کد و اشکال زدایی را در هنگام برخورد با منطق ناهمزمان سخت می کند. جهنم برگشت به تماس شبیه زیر است،

```
async1(function(){
  async2(function(){
    async3(function(){
      async4(function(){
        ....
      });
    });
  });
});
```

۵۷.(SSE)Server-sent-events چیه؟

رویدادهای ارسال شده توسط سرور (SSE) یک فناوری فشار سرور است که به مرورگر امکان می دهد به روزرسانی های خودکار را از طریق اتصال HTTP بدون استفاده از نظرسنجی دریافت کند. اینها یک کanal ارتباطی یک طرفه هستند - رویدادها فقط از سروری به مشتری دیگر منتقل می شوند. این در به روزرسانی های فیسبوک / توییتر، به روزرسانی قیمت سهام، فیدهای خبری و غیره استفاده شده است.

۵۸.چطوری می تونیم اعلان های server-sent-event را دریافت کنیم؟

شی EventSource برای دریافت اعلان های رویداد ارسال شده از سرور استفاده می شود. به عنوان مثال، می توانید پیام هایی را از سرور به صورت زیر دریافت کنید.

```
if(typeof(EventSource) !== "undefined") {
  var source = new EventSource("sse_generator.js");
  source.onmessage = function(event) {
    document.getElementById("output").innerHTML += event.data + "<br>";
  };
}
```

۵۹.چطوری می تونیم پشتیبانی مرورگر برای SSE را بررسی کنیم؟

می توانید قبل از استفاده از رویدادهای ارسال شده توسط مرورگر مانند زیر، پشتیبانی مرورگر را انجام دهید.

```

if(typeof(EventSource) !== "undefined") {
    // Server-sent events supported. Let's have some code here!
} else {
    // No server-sent events supported
}

```

۶۰. کدوم توابع روی SSE وجود دارن؟

در زیر لیستی از رویدادهای موجود برای رویدادهای ارسال شده توسط سرور آمده است

Description	Event
هنگامی که اتصال به سرور باز می شود استفاده می شود	onopen
این رویداد زمانی استفاده می شود که پیامی دریافت شود	onmessage
زمانی اتفاق می افتد که خطای رخ دهد	onerror

۶۱. اصلی‌ترین قوانین promise‌ها چیا هستن؟

یک پرمیس باید از یک سری قوانین خاص پیروی کند،

۱. پرمیس آبجکتی است که متد «.then()» سازگار با استاندارد را ارائه می کند

۲. یک پرمیس معلق ممکن است به حالت تحقق یافته یا رد شده تبدیل شود

۳. پرمیس تمام شده یا رد شده حل و فصل می شود و نباید به حالت دیگری تبدیل شود.

۴. پس از اتمام قول، ارزش آن نباید تغییر کند.

۶۲. توى callback چطوری رخ میده؟

می‌توانید یک پاسخ تماس را در داخل یک تماس دیگر قرار دهید تا اقدامات را به صورت متوالی یکی انجام دهید. این به عنوان callback شناخته می شود.

```

loadScript('/script1.js', function(script) {
    console.log('first script is loaded');

    loadScript('/script2.js', function(script) {
        console.log('second script is loaded');

        loadScript('/script3.js', function(script) {
            console.log('third script is loaded');
            // after all scripts are loaded
        });
    });
});

```

۶۳. زنجیره promise‌ها چیه؟

فرآیند اجرای دنباله‌ای از وظایف ناهمزمان یکی پس از دیگری با استفاده از پرمیس‌ها به عنوان Promise chaining شناخته می‌شود. باید برای محاسبه نتیجه نهایی مثالی از زنجیره قولی بزنیم.

```

new Promise(function(resolve, reject) {

    setTimeout(() => resolve(1), 1000);

}).then(function(result) {

    console.log(result); // 1
    return result * 2;

}).then(function(result) {

    console.log(result); // 2
    return result * 3;

}).then(function(result) {

    console.log(result); // 6
    return result * 4;

});

```

در دسته‌های فوق، نتیجه به زنجیره‌های then() handler با جریان کار زیر منتقل می‌شود.

۱. پرمیس اولیه در ۱ ثانیه حل می‌شود،
۲. پس از آن، «.then» با ثبت نتیجه (1) فراخوانی می‌شود و سپس یک پرمیس با مقدار نتیجه * 2 برمی‌گرداند.
۳. پس از آن مقدار به بعدی منتقل شد. سپس با ثبت نتیجه (2) و بازگرداندن یک پرمیس با نتیجه * 3.
۴. در نهایت مقدار به آخرین . سپس با ثبت نتیجه (6) و یک پرمیس با نتیجه * 4، `handler`.

۶۴. کاربرد متدهای promise.all چیه؟

یک پرمیس است که آرایه ای از پرمیس ها را به عنوان ورودی می‌گیرد (یک تکرار) و زمانی حل می‌شود که همه پرمیس ها حل شوند یا یکی از آنها رد شود. به عنوان مثال، نحوه متدهای `Promise.all` در زیر آمده است.

```
Promise.all([Promise1, Promise2, Promise3]).then(result => { console.log(result) })
  .catch(error => console.log(`Error in promises ${error}`))
```

نکته: به یاد داشته باشید که ترتیب پرمیس ها (خروجی نتیجه) طبق ترتیب ورودی حفظ می‌شود.

۶۵. هدف از متدهای promise.race چیه؟

متدهای `Promise.race` نمونه ای از پرمیس را که ابتدا حل یا رد شده است بر می‌گردانند. باید مثالی از متدهای `race()` را در نظر بگیریم که در آن پرمیس 2 ابتدا حل می‌شود

```
var promise1 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 500, 'one');
});
var promise2 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 100, 'two');
});

Promise.race([promise1, promise2]).then(function(value) {
  console.log(value); // "two" // Both promises will resolve, but promise2 is faster
});
```

۶۶. حالت سخت جاوا اسکریپت چی کار میکنه؟

یک ویژگی جدید در ECMAScript 5 است که به شما امکان می‌دهد یک برنامه یا یک تابع را در یک زمینه عملیاتی "سخت" قرار دهید. به این ترتیب از انجام برخی اقدامات جلوگیری می‌کند و استثناهای بیشتری را ایجاد می‌کند. عبارت تحت لفظی "استفاده از سخت"؛ به مرورگر دستور می‌دهد تا از کد جاوا اسکریپت در حالت Strict استفاده کند.

۶۷. چرا به حالت strict نیاز داریم؟

حالت سخت گیرانه برای نوشتن جاوا اسکریپت "امن" با اطلاع رسانی "بد نحوی" به خطاهای واقعی مفید است. به عنوان مثال، ایجاد تصادفی یک متغیر سراسری را با پرتاب یک خطا حذف می‌کند و همچنین یک خطا برای انتساب به یک ویژگی غیرقابل نوشتن، یک ویژگی فقط گیرنده، یک ویژگی غیرموجود، یک متغیر غیرموجود یا یک ویژگی غیرقابل نوشتن پرتاب می‌کند. شی موجود

۶۸. چطوری می‌توانیم حالت strict را فعال کنیم؟

حالت سخت با اضافه کردن "استفاده از سخت" اعلام می شود. به ابتدای یک اسکریپت یا یک تابع.
اگر در ابتدای یک اسکریپت اعلام شود، دامنه جهانی دارد.

```
"use strict";
x = 3.14; // This will cause an error because x is not declared
```

و اگر در داخل یک تابع اعلام کنید، محدوده محلی دارد

```
x = 3.14;           // This will not cause an error.
myFunction();

function myFunction() {
  "use strict";
  y = 3.14;   // This will cause an error
}
```

۶۹. هدف از عملگر نقطی دوتایی (!!) چیه؟

تعجب مضاعف یا نفی (!!) تضمین می کند که نوع حاصل یک بولی است. اگر نادرست بود (به عنوان مثال 0، تهی، تعریف نشده، وغیره)، نادرست خواهد بود، در غیر این صورت، درست است.
به عنوان مثال، می توانید نسخه IE را با استفاده از عبارت زیر آزمایش کنید.

```
let isIE8 = false;
isIE8 = !! navigator.userAgent.match(/MSIE 8.0/);
console.log(isIE8); // returns true or false
```

اگر از این عبارت استفاده نکنید، مقدار اصلی را برمی گرداند.

```
console.log(navigator.userAgent.match(/MSIE 8.0/)); // returns either an Array or null
```

نکته: بیان !! یک اپراتور نیست، اما فقط دو برابر است! اپراتور.

۷۰. هدف از عملگر delete چیه؟

کلمه کلیدی delete برای حذف ویژگی و همچنین مقدار آن استفاده می شود.

```
var user= {name: "John", age:20};
delete user.age;

console.log(user); // {name: "John"}
```

۷۱. عملگر typeof چیکار می کنه؟

برای یافتن نوع متغیر جاوا اسکریپت می توانید از عملگر typeof JavaScript استفاده کنید. نوع یک متغیر یا یک عبارت را برمی گرداند.

```
typeof "John Abraham"      // Returns "string"
typeof (1 + 2)            // Returns "number"
```

۷۲. چیه و چه زمانی undefined می‌گیریم؟

ویژگی تعریف نشده نشان می‌دهد که به یک متغیر مقداری اختصاص داده نشده است یا اصلاً اعلام نشده است. نوع مقدار تعریف نشده نیز تعریف نشده است.

```
var user;      // Value is undefined, type is undefined
console.log(typeof(user)) //undefined
```

هر متغیری را می‌توان با تنظیم مقدار روی undefined خالی کرد.

```
user = undefined
```

۷۳. چیه null؟

مقدار null عدم وجود عمده هر مقدار شی را نشان می‌دهد. این یکی از مقادیر اولیه جاوا اسکریپت است. نوع مقدار null شی است.

با قرار دادن مقدار null می‌توانید متغیر را خالی کنید.

```
var user = null;
console.log(typeof(user)) //object
```

۷۴. تفاوت‌های بین null و undefined چیا هستن؟

در زیر تفاوت‌های اصلی بین null و undefined آورده شده است.

Undefined	Null
این یک مقدار انتساب نیست که در آن متغیری اعلام شده باشد اما هنوز مقداری به آن اختصاص داده نشده باشد.	این یک مقدار انتساب است که نشان می‌دهد متغیر به هیچ شیئی اشاره نمی‌کند.
تایپ undefined است	تایپ null آبجکت است
مقدار تعریف نشده یک مقدار اولیه است used زمانی که به یک متغیر مقداری اختصاص داده نشده باشد.	مقدار null یک مقدار اولیه است که نشان دهنده مرجع تهی، خالی یا غیر موجود است.
عدم وجود خود متغیر را نشان می‌دهد	عدم وجود مقدار برای یک متغیر را نشان می‌دهد
در حین انجام عملیات اولیه به NaN تبدیل می‌شود	در حین انجام عملیات اولیه به صفر (۰) تبدیل شد

تابع eval() کد جاوا اسکریپت را که به صورت رشته نمایش داده شده است ارزیابی می‌کند. رشته می‌تواند یک عبارت جاوا اسکریپت، متغیر، دستور یا دنباله‌ای از عبارات باشد.

```
console.log(eval('1 + 2')); // 3
```

۷۶. تفاوت‌های بین document و window چیا هستن؟

در زیر تفاوت‌های اصلی بین پنجره و سند وجود دارد.

Document	Window
این فرزند مستقیم شی پنجره است. این همچنین به عنوان مدل شیء سند (DOM) معرفی شده است.	این عنصر سطح ریشه در هر صفحه وب است
موجود است شما می‌توانید از طریق window.document یا سند به آن دسترسی داشته باشید.	به طور پیش فرض شی پنجره به طور ضمنی در صفحه
متدهایی مانند getElementById، getElementByTagName، createElement و غیره را فراهم می‌کند	دارای متدهایی مانند alert()، confirm() و ویژگی‌هایی مانند document، location

۷۷. توی جاوا اسکریپت چطوری می‌تونیم به history دسترسی داشته باشیم؟

شی window.history حاوی تاریخچه مرورگر است. با استفاده از متدهای back() و next() می‌توانید URL‌های قبلی و بعدی را در تاریخچه بارگذاری کنید.

```
function goBack() {
    window.history.back()
}
function goForward() {
    window.history.forward()
}
```

نکته: همچنین می‌توانید بدون پیشوند پنجره به تاریخچه دسترسی داشته باشید.

۷۸. انواع داده‌های جاوا اسکریپت کدوما هستن؟

- در زیر لیستی از انواع داده‌های جاوا اسکریپت موجود است
۱. Number
 ۲. String
 ۳. Boolean

Object.۴

Undefined.۵

۷۹. isNaN چیه و چیکار می‌کنه؟

تابع `isNaN()` برای تعیین اینکه آیا یک مقدار یک عدد غیرقانونی (Not-a-Number) است یا خیر استفاده می‌شود. یعنی اگر مقدار برابر با `NAN` باشد، این تابع `true` برمی‌گرداند. در غیر این صورت `false` برمیگردد.

```
isNaN('Hello') //true  
isNaN('100') //false
```

۸۰. تفاوت‌های بین undefined و undeclared چیا هستن؟

در زیر تفاوت عمدۀ بین متغیرهای اعلام نشده و تعریف نشده آورده شده است.

undefined	undeclared
تعریف نمی‌شوند این متغیرها در برنامه اعلام شده اما هیچ مقدار	این متغیرها در یک برنامه وجود ندارند و
اگر سعی کنید مقدار یک متغیر تعریف نشده را بخوانید، یک مقدار تعریف نشده برگردانده می‌شود.	اگر سعی کنید مقدار یک متغیر اعلام نشده را بخوانید، با خطای زمان اجرا مواجه می‌شوید

۸۱. کدوم متغیرها عمومی هستن؟

متغیرهای سراسری آنهایی هستند که در طول کد بدون هیچ محدوده‌ای در دسترس هستند. کلمه کلیدی `var` برای اعلام یک متغیر محلی استفاده می‌شود، اما اگر آن را حذف کنید، تبدیل به متغیر جهانی می‌شود.

```
msg = "Hello" // var is missing, it becomes global variable
```

۸۲. مشکلات متغیرهای عمومی چیا هستن؟

مشکل متغیرهای سراسری تضاد نام متغیرها با دامنه محلی و جهانی است. همچنین اشکال زدایی و آزمایش کدی که به متغیرهای سراسری متعلق است دشوار است.

۸۳. مقدار NaN چیه؟

ویژگی NaN یک ویژگی جهانی است که مقدار "Not-a-Number" را نشان می دهد. یعنی نشان می دهد که یک مقدار یک عدد قانونی نیست. استفاده از NaN در یک برنامه بسیار نادر است، اما می توان از آن به عنوان مقدار بازگشتی برای موارد کمی استفاده کرد

```
Math.sqrt(-1)  
parseInt("Hello")
```

۸۴. هدف از تابع isFinite چیه؟

تابع isFinite() برای تعیین اینکه آیا یک عدد یک عدد محدود و قانونی است استفاده می شود. اگر مقدار +infinity، -infinity یا NaN (Not-a-Number) باشد false برمی‌گرداند، در غیر این صورت true را برمی‌گرداند.

```
isFinite(Infinity); // false  
isFinite(NaN); // false  
isFinite(-Infinity); // false  
  
isFinite(100); // true
```

۸۵. یه event-flow چیه؟

جريان رویداد ترتیبی است که رویداد در صفحه وب دریافت می شود. وقتی روی عنصری کلیک می کنید که در عناصر مختلف دیگر تودرتو است، قبل از اینکه کلیک شما واقعاً به مقصد یا عنصر هدف برسد، باید رویداد کلیک را برای هر یک از عناصر والد خود ابتدا راه اندازی کند و از بالا با شی پنجره جهانی شروع شود.

- دو راه برای جريان رویداد وجود دارد
- ۱. از بالا به پایین(Event Capturing)
- ۲. از پایین به بالا(Event Bubbling)

۸۶. Event-bubbling چیه؟

حباب کردن رویداد نوعی انتشار رویداد است که در آن رویداد ابتدا روی درونی ترین عنصر هدف راه اندازی می شود، و سپس به طور متوالی روی اجداد (والد) عنصر هدف در همان سلسله مراتب تودرتو راه اندازی می شود تا زمانی که به بیرونی ترین عنصر DOM برسد.

۸۷. Event-capturing چیه؟

ثبت رویداد نوعی انتشار رویداد است که در آن رویداد ابتدا توسط بیرونی ترین عنصر ثبت می شود و سپس به طور متوالی بر روی فرزندان (فرزنده) عنصر هدف در همان سلسله مراتب تودرتو راه اندازی می شود تا زمانی که به درونی ترین عنصر DOM برسد.

۸۸. چطوری میشه یه فرم رو با استفاده از جاواسکریپت ثبت کرد؟

می توانید با استفاده از جاوا اسکریپت فرمی را ارسال کنید ()use document.form[0].submit. تمام اطلاعات ورودی فرم با استفاده از onsubmit event handler ارسال می شود

```
function submit() {  
    document.form[0].submit();  
}
```

۸۹. چطوری میشه به اطلاعات مربوط به سیستم عامل کاربر دسترسی داشت؟

شی window.navigator حاوی اطلاعاتی درباره جزئیات سیستم عامل مرورگر بازدیدکننده است. برخی از ویژگی های سیستم عامل تحت ویژگی پلتفرم در دسترس هستند،

```
console.log(navigator.platform);
```

۹۰. تفاوت های بین رخدادهای DOMContentLoaded و document-load چیا هستن؟

رویداد «DOMContentLoaded» زمانی فعال می شود که سند اولیه HTML به طور کامل بارگیری و تجزیه شده باشد، بدون اینکه منتظر بمانید تا دارایی ها (سبکها، تصاویر و فریم های فرعی) بارگیری تمام شود. در حالی که رویداد بارگیری زمانی فعال می شود که کل صفحه بارگیری شود، از جمله تمام منابع وابسته (شیوه ها، تصاویر).

۹۱. تفاوت های بین object های user و native، host چیا هستن؟

"اشیاء بومی" اشیایی هستند که بخشی از زبان جاوا اسکریپت تعریف شده توسط مشخصات ECMAScript هستند. به عنوان مثال، اشیاء اصلی رشته، ریاضی، Object، Function، RegExp و غیره که در مشخصات ECMAScript تعریف شده اند. "اشیاء میزبان" اشیایی هستند که توسط مرورگر یا محیط زمان اجرا (Node) ارائه می شوند. به عنوان مثال، پنجره، XMLHttpRequest، گره های DOM و غیره به عنوان اشیاء میزبان در نظر گرفته می شوند. "اشیاء کاربر" اشیایی هستند که در کد جاوا اسکریپت تعریف شده اند. به عنوان مثال، اشیاء کاربر ایجاد شده برای اطلاعات پروفایل.

۹۲. کدام ابزار و تکنیک ها برای دیباگ کدن برنامه جاواسکریپتی استفاده میشن؟

می توانید از ابزارها یا تکنیک های زیر برای اشکال زدایی جاوا اسکریپت استفاده کنید

۱. Chrome Devtools
۲. debugger statement
۳. console.log statement

۹۳. مزایا و معایب استفاده از promise ها به جای callback چیا هستن؟

در زیر لیستی از مزایا و معایب وعده‌های مربوط به تماس‌های تلفنی آورده شده است.

مزایا:

۱. از جهنم callback که قابل خواندن نیست جلوگیری می‌کند
۲. نوشتن کدهای ناهمزمان متوالی با .then() آسان است.
۳. نوشتن کد ناهمزمان موازی آسان با Promise.all()
۴. برخی از مشکلات رایج callback های برگشتی را حل می‌کند (بسیار دیر، خیلی زود، بارها callback و خطاهای استثنایها را بپذیرید)

معایب:

۱. کد کمی پیچیده می‌سازد
۲. اگر ES6 پشتیبانی نمی‌شود، باید یک polyfill بارگذاری کنید

۹۴. تفاوت‌های بین DOM روی property و attribute چیا هستن؟

ویژگی‌ها در نشانه گذاری HTML تعریف می‌شوند در حالی که ویژگی‌ها در DOM تعریف می‌شوند. به عنوان مثال، عنصر HTML زیر دارای ۲ ویژگی نوع و مقدار است،

```
<input type="text" value="Name:>
```

می‌توانید مقدار ویژگی را به صورت زیر بازیابی کنید

```
const input = document.querySelector('input');
console.log(input.getAttribute('value')); // Good morning
console.log(input.value); // Good morning
```

و بعد از اینکه مقدار فیلد متن را به "Good evening" تغییر دادید مانند می‌شود

```
console.log(input.getAttribute('value')); // Good morning
console.log(input.value); // Good evening
```

۹۵. سیاست same-origin چیه؟

خط مشی همان مبدأ خط مشی است که از درخواست جاوا اسکریپت در سراسر مرزهای دامنه جلوگیری می‌کند. مبدأ به عنوان ترکیبی از طرح URI، نام میزبان و شماره پورت تعریف می‌شود. اگر این خطمشی را فعال کنید، از دسترسی یک اسکریپت مخرب در یک صفحه به داده‌های حساس در صفحه وب دیگر با استفاده از Document Object Model (DOM) جلوگیری می‌کند.

۹۶. هدف استفاده از void 0 چیه؟

(Void) برای جلوگیری از به روز رسانی صفحه استفاده می شود. این برای از بین بردن عارضه جانبی ناخواسته مفید خواهد بود، زیرا مقدار اولیه تعریف نشده را برمی گرداند. معمولاً برای اسناد HTML استفاده می شود که از (void(0); استفاده می کنند. در یک عنصر <a> . یعنی وقتی روی یک پیوند کلیک می کنید، مرورگر یک صفحه جدید را بارگیری می کند یا همان صفحه را تازه می کند. اما با استفاده از این عبارت از این رفتار جلوگیری می شود. به عنوان مثال، پیوند زیر پیام را بدون بارگیری مجدد صفحه مطلع می کند

```
<a href="JavaScript:void(0);" onclick="alert('Well done!')>Click Me!</a>
```

۹۷. جاوااسکریپت یه زبان تفسیری هست یا کامپایلری؟

جاوا اسکریپت یک زبان تفسیر شده است، نه یک زبان کامپایل شده. یک مفسر در مرورگر کد جاوا اسکریپت را می خواند، هر خط را تفسیر می کند و آن را اجرا می کند. امروزه مرورگرهای مدرن از فناوری موسوم به کامپایل JIT (Just-In-Time) استفاده می کنند که جاوا اسکریپت را در زمانی که در شرف اجراست به بایت کد اجرایی کامپایل می کند.

۹۸. آیا جاوااسکریپت یه زبان حساس به بزرگی و کوچکی(case-sensitive) حروف است؟

بله، جاوا اسکریپت یک زبان حساس به حروف کوچک و بزرگ است. کلمات کلیدی زبان، متغیرها، نام تابع و اشیا، و هر شناسه دیگر باید همیشه با حروف بزرگ تایپ شوند.

۹۹. ارتباطی بین JavaScript و Java وجود داره؟

نه، آنها کاملاً دو زبان برنامه نویسی متفاوت هستند و هیچ ارتباطی با یکدیگر ندارند. اما هر دوی آنها زبان های برنامه نویسی شی گرا هستند و مانند بسیاری از زبان های دیگر، از نحو مشابهی برای ویژگی های اساسی (اگر، غیره، برای، سوئیچ، شکستن، ادامه و غیره) پیروی می کنند.

۱۰۰. Event ها چی هستن؟

رویدادها «چیزهایی» هستند که برای عناصر HTML اتفاق می افتد. هنگامی که جاوا اسکریپت در صفحات HTML استفاده می شود، جاوا اسکریپت می تواند به این رویدادها واکنش نشان دهد. برخی از نمونه های رویدادهای HTML عبارتند از:

۱. بارگیری صفحه وب به پایان رسید
 ۲. فیلد ورودی تغییر کرد
 ۳. دکمه کلیک شد
- بیایید رفتار رویداد کلیک را برای عنصر دکمه شرح دهیم،

```

<!doctype html>
<html>
<head>
<script>
  function greeting() {
    alert('Hello! Good morning');
  }
</script>
</head>
<body>
  <button type="button" onclick="greeting()">Click me</button>
</body>
</html>

```

۱۰۱. کی جاواسکریپت رو ساخته؟

جاوا اسکریپت توسط برندان ایچ در سال ۱۹۹۵ و در زمان او در نت اسکیپ ارتباطات ایجاد شد. در ابتدا با نام Mocha توسعه یافت، اما بعداً زمانی که برای اولین بار در نسخه های بتا نت اسکیپ عرضه شد، این زبان به طور رسمی LiveScript نامیده شد.

۱۰۲. هدف از متد preventDefault چیه؟

متد () preventDefault اگر رویداد قابل لغو باشد، آن را لغو می کند، به این معنی که عمل یا رفتار پیش فرض متعلق به رویداد رخ نخواهد داد. به عنوان مثال، جلوگیری از ارسال فرم هنگام کلیک بر روی دکمه ارسال و جلوگیری از باز شدن URL صفحه هنگام کلیک کردن بر روی لینک از موارد رایج استفاده است.

```

document.getElementById("link").addEventListener("click", function(event){
  event.preventDefault();
});

```

نکته: Remember that not all events are cancelable

۱۰۳. کاربرد متد stopPropagation چیه؟

روش stopPropagation برای جلوگیری از حبابی شدن رویداد در زنجیره رویداد استفاده می شه. به عنوان مثال، div های تودرتو زیر با متد stopPropagation از انتشار پیش فرض رویداد هنگام کلیک بر روی Div1 (Div1) جلوگیری می کنه.

```

<p>Click DIV1 Element</p>
<div onclick="secondFunc()">DIV 2
  <div onclick="firstFunc(event)">DIV 1</div>
</div>

<script>
function firstFunc(event) {
  alert("DIV 1");
  event.stopPropagation();
}

function secondFunc() {
  alert("DIV 2");
}
</script>

```

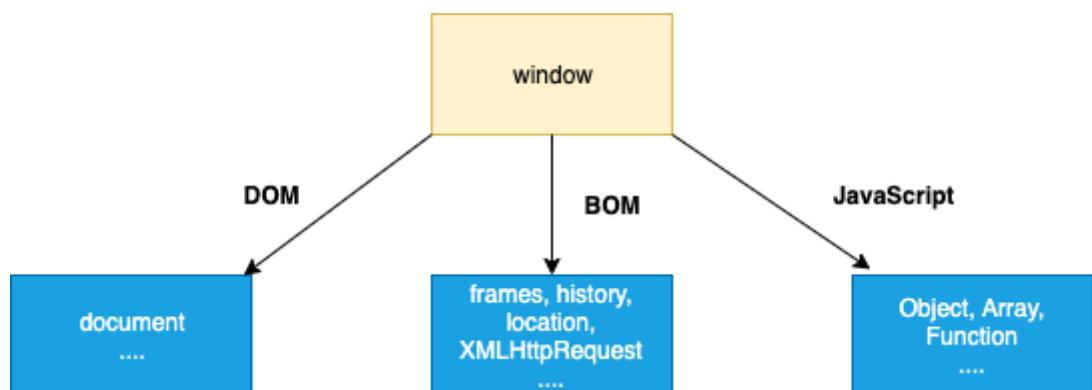
۱۰۴. مراحلی که هنگام استفاده از event-handler رخ میده چیا هستن؟

عبارت return false در کنترل کننده رویداد مراحل زیر را انجام می‌دهد:

۱. ابتدا عملکرد یا رفتار پیش فرض مرورگر را متوقف می‌کنه.
۲. این رویداد از انتشار DOM جلوگیری می‌کنه
۳. اجرای callback را متوقف می‌کنه و بلافاصله پس از فراخوانی برموی گردد.

۱۰۵. BOM چیه؟

مدل شیء مرورگر (BOM) به جاوا اسکریپت اجازه می‌دهد تا با مرورگر "صabit کند". این شامل ناوبر اشیاء، تاریخچه، صفحه، مکان و سند است که فرزندان پنجره هستند. مدل شیء مرورگر استاندارد نیست و می‌توانه بر اساس مرورگرهای مختلف تغییر کند.



۱۰۶. موارد استفاده از setTimeout کدوما هستن؟

متدهای setTimeout() برای فراخوانی یکتابع یا ارزیابی یک عبارت پس از تعداد مشخصی از میلی ثانیه استفاده می‌شوند. به عنوان مثال، باید یک پیام را پس از 2 ثانیه با استفاده از روش setTimeout ثبت کنیم.

```
setTimeout(function(){ console.log("Good morning"); }, 2000);
```

۱۰۷. موارد استفاده از **setInterval** کداما هستن؟

متده `setInterval()` برای فراخوانی یک تابع یا ارزیابی یک عبارت در بازه های زمانی مشخص (بر حسب میلی ثانیه) استفاده میشے. به عنوان مثال، اجازه دهید یک پیام را پس از 2 ثانیه با استفاده از روش `setInterval` ثبت کنیم.

```
setInterval(function(){ console.log("Good morning"); }, 2000);
```

۱۰۸. چرا جاواسکریپت رو به عنوان یه زبان تک **thread** میشناسن؟

جاوا اسکریپت یک زبان تک رشته ای است. زیرا مشخصات زبان به برنامه نویس اجازه نمی دهد تا کد بنویسد تا مفسر بتواند بخش هایی از آن را به صورت موازی در چندین رشته یا پردازش اجرا کند. در حالی که زبان هایی مانند C، go، ++java، C++ می توانند برنامه های چند رشته ای و چند فرآیندی بسازند.

۱۰۹. **Event-delegation** چیه؟

تفویض رویداد تکنیکی برای گوش دادن به رویدادهاست که در آن یک عنصر والد را به عنوان شنونده برای همه رویدادهایی که در داخل آن اتفاق میافتد، تفویض میکنید.
به عنوان مثال، اگر می خواهید تغییرات فیلد را در یک فرم خاص تشخیص دهید، می توانید از تکنیک انتقال رویداد استفاده کنید.

```
var form = document.querySelector('#registration-form');

// Listen for changes to fields inside the form
form.addEventListener('input', function (event) {

  // Log the field that was changed
  console.log(event.target);

}, false);
```

۱۱۰. **ECMAScript** چیه؟

زبان برنامه نویسی است که اساس جاواسکریپت را تشکیل می دهد. ECMAScript توسط سازمان استاندارد بین المللی ECMA در مشخصات ECMA-262 و ECMA-402 استاندارد شده است. اولین نسخه ECMAScript در سال 1997 منتشر شد.

یک فرمت سبک وزن است که برای تبادل داده‌ها استفاده می‌شود. این بر اساس زیرمجموعه‌ای از زبان جاوا اسکریپت است که اشیا در جاوا اسکریپت ساخته می‌شون.

۱۱۲. قوانین فرمت JSON کدام هستند؟

- در زیر لیستی از قوانین نحوی JSON آمده است
- ۱. داده‌ها به صورت جفت نام/مقدار هستند
- ۲. داده‌ها با کاما از هم جدا می‌شون
- ۳. براکت‌ها اجسام را نگه می‌دارند
- ۴. براکت‌های مربعی آرایه‌ها را نگه می‌دارند

۱۱۳. هدف از متد JSON.stringify چیه؟

هنگام ارسال داده‌ها به وب سرور، داده‌ها باید در قالب رشته ای باشند. شما می‌توانید با تبدیل شی JSON` به رشته با استفاده از متد `stringify` () به این هدف دست یابید.

```
<span dir="ltr" align="left>
javascript```
{var userJSON = {'name': 'John', age: 31
;(var userString = JSON.stringify(user
"{console.log(userString); //{"name":"John","age":31
```


**[فهرست] (#[فهرست])
```

## ۱۱۴. چطوری می‌توانیم یه رشته JSON (string) رو تجزیه کنیم؟

هنگام دریافت داده‌ها از یک وب سرور، داده‌ها همیشه در قالب رشته ای هستند. اما می‌توانیم این مقدار رشته را با استفاده از متد `parse` () به یک شی جاوا اسکریپت تبدیل کنیم.

```
var userString = '{"name":"John","age":31}';
var userJSON = JSON.parse(userString);
console.log(userJSON); // {name: "John", age: 31}
```

## ۱۱۵. چرا به JSON نیاز داریم؟

هنگام تبادل داده بین مرورگر و سرور، داده‌ها فقط می‌توانند متنی باشند. از آنجایی که JSON فقط متنی است، می‌توان آن را به راحتی به سرور ارسال کرد و از آن به عنوان قالب داده توسط هر زبان برنامه‌نویسی استفاده کرد.

## ۱۱۵. PWA‌ها چی هستن؟

نوعی از برنامه‌های تلفن همراه هستند که از طریق وب ارائه میشون، و با استفاده از فناوری‌های رایج وب از جمله HTML، CSS و جاوا اسکریپت ساخته میشون. این PWA‌ها در سرورها مستقر میشون، از طریق URL‌ها قابل دسترسی هستند و توسط موتورهای جستجو فهرست بندی میشون.

## ۱۱۶. هدف از متدهای clearTimeout و clearInterval چیه؟

تابع `clearTimeout()` در جاوا اسکریپت برای پاک کردن بازه زمانی استفاده میشے که قبل از آن توسط تابع `setTimeout()` تنظیم شده است. یعنی مقدار بازگشتی تابع `setTimeout()` در یک متغیر ذخیره میشے و برای پاک کردن تایمر به تابع `clearTimeout()` منتقل میشے.  
به عنوان مثال، از روش `setTimeout()` زیر برای نمایش پیام پس از 3 ثانیه استفاده میشے. این مهلت زمانی را می‌توان با روش `clearTimeout()` پاک کرد.

```
<script>
var msg;
function greeting() {
 alert('Good morning');
}
function start() {
 msg = setTimeout(greeting, 3000);
}

function stop() {
 clearTimeout(msg);
}
</script>
```

## ۱۱۷. هدف از متدهای clearInterval و clearInterval چیه؟

تابع `clearInterval()` در جاوا اسکریپت برای پاک کردن فاصله‌ای که توسط تابع `setInterval()` تنظیم شده است استفاده میشے. به عنوان مثال، مقدار بازگشتی که توسط تابع `setInterval()` برمی‌گردد در یک متغیر ذخیره میشے و برای پاک کردن فاصله به تابع `clearInterval()` ارسال میشے.  
به عنوان مثال، از روش `setInterval()` زیر برای نمایش پیام در هر 3 ثانیه استفاده میشے. این بازه را می‌توان با روش `clearInterval()` پاک کرد.

```

<script>
var msg;
function greeting() {
 alert('Good morning');
}
function start() {
 msg = setInterval(greeting, 3000);
}

function stop() {
 clearInterval(msg);
}
</script>

```

## ۱۱۸. توى جاواسکریپت، چطوري مىشە به يه صفحە جديد redirect انجام داد؟

در Java اسکریپت، مىتونين با استفاده از ویژگی location شى پنجره به صفحە جديدي هدايت بشين.

```

function redirect() {
 window.location.href = 'newPage.html';
}

```

## ۱۱۹. چطوري بررسى مىكニن که يه string شامل يه substring هست يا نه؟

روش ممکن برای بررسی اینکه آیا یک رشته دارای یک رشته فرعی است یا خیر وجود دارد.  
استفاده از متد ES6 String.prototype.includes روش آزمایش یک رشته حاوی یک رشته فرعی ارائه کرد.

```

var mainString = "hello", subString = "hell";
mainString.includes(subString)

```

۱. استفاده از متد indexOf(): در یک محیط ES5 یا قدیمی‌تر، می‌توانید از «String.prototype.indexOf» استفاده کنید که ایندکس یک رشته فرعی را برمی‌گرداند. اگر مقدار شاخص برابر با 1 نباشد، به این معنی است که رشته فرعی در رشته اصلی وجود دارد.

```

var mainString = "hello", subString = "hell";
mainString.indexOf(subString) !== -1

```

۲. استفاده از Regex: راه حل پیشرفتی از روش آزمون عبارت ('Regular') استفاده می‌کند، که امکان آزمایش در برابر عبارات منظم را فراهم می‌کند.

```

var mainString = "hello", regex = "/hell/";
regex.test(mainString)

```

## ۱۲۰. توى جاواسکریپت، چطوری مقدار يه آدرس email رو اعتبارسنجی مى‌کنин؟

مى‌توانيد با استفاده از Regex ایمیل را در جاوا اسکریپت تأیید کنيد. توصیه مى‌شود به جای سمت کلاینت، اعتبارسنجی در سمت سرور انجام شود. زیرا جاوا اسکریپت را مى‌توان در سمت کلاینت غیرفعال کرد.

```
function validateEmail(email) {
 var re = /^[^<>()[]\.,;:\s@"]+(\.\[^<>()[]\.,;:\s@"]+)*|(.+@)([\[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.])|(([a-zA-Z\-_0-9]+\.)+[a-zA-Z]{2,})$/;
 return re.test(String(email).toLowerCase());
}
```

بالا کarakترهای یونیکد را می‌پذیرد.

## ۱۲۱. چطوری مى‌تونيم مقدار آدرس url جاري رو بخونيم؟

مى‌توانيد از عبارت window.location.href برای دریافت مسیر آدرس فعلی استفاده کنيد و مى‌توانيد از همان عبارت برای بهروزرسانی URL نیز استفاده کنيد. همچنین مى‌توانيد از document.URL برای اهداف فقط خواندنی استفاده کنيد، اما اين راه حل مشکلاتي در FF دارد.

```
console.log('location.href', window.location.href); // Returns full URL
```

## ۱۲۲. ويژگی‌های مختلف url روی history object مربوط به کدوما هستن؟

برای دسترسی به اجزای URL صفحه مى‌توان از ويژگی‌های شی location زیر استفاده کرد.

۱. href - ورودی URL
۲. protocol - پروتکل استفاده شده URL
۳. host - هاست و پورت URL
۴. hostname - URL هاست
۵. port - شماره پورت URL
۶. pathname - مسیر URL
۷. search - قسمت جستجو URL
۸. hash - محل جایگيری URL

## ۱۲۳. توى جاواسکریپت چطوری مى‌تونيم مقدار يه query-string رو بخونيم؟

مى‌توانيد از URLSearchParams برای دریافت مقادير رشته پرس و جو در جاوا اسکریپت استفاده کنيد. بيايد مثالی برای دریافت مقدار کد مشتری از رشته پرس و جو URL ببینيم،

```
const urlParams = new URLSearchParams(window.location.search);
const clientCode = urlParams.get('clientCode');
```

## ۱۲۴. چطوری می‌تونیم بررسی کنیم که آیا یه پرایپرتو روی آبجکت وجود داره یا نه؟

با استفاده از سه رویکرد می‌توانیم بررسی کنید که آیا یک کلید در یک شی وجود دارد یا خیر.

۱. استفاده از عملگرها: شما می‌توانیم از عملگر `in` استفاده کنید که آیا کلیدی در یک شی وجود دارد یا خیر

```
"key" in obj
```

و اگر می‌خواهید بررسی کنید که آیا کلید وجود ندارد، به یاد داشته باشید که از پرانتز استفاده کنید،

```
!("key" in obj)
```

۲. استفاده از متدهای `hasOwnProperty`: می‌توانید از `hasOwnProperty` برای آزمایش ویژگی‌های نمونه شی (و نه ویژگی‌های ارثی) استفاده کنید.

```
obj.hasOwnProperty("key") // true
```

۳. استفاده از مقایسه `undefined`: If you access a non-existing property from an object, the result is `undefined`. Let's compare the properties against `undefined` to determine the existence of the property.

```
const user = {
 name: 'John'
};

console.log(user.name !== undefined); // true
console.log(user.nickName !== undefined); // false
```

## ۱۲۵. How do you loop through or enumerate javascript object

می‌توانید از حلقه `for-in` برای حلقه زدن شی جاوا اسکریپت استفاده کنید. همچنین می‌توانید مطمئن شوید که کلیدی که دریافت می‌کنید یک ویژگی واقعی یک شی است و با استفاده از روش `hasOwnProperty` از نمونه اولیه نیست.

```
var object = {
 "k1": "value1",
 "k2": "value2",
 "k3": "value3"
};

for (var key in object) {
 if (object.hasOwnProperty(key)) {
 console.log(key + " -> " + object[key]); // k1 -> value1 ...
 }
}
```

## ۱۲۶. How do you test for an empty object

راه حل‌های مختلفی بر اساس نسخه‌های ECMAScript وجود دارد

## .۱۲۷ استفاده (+Object entries(ECMA 7

میتوانیم از object entries استفاده کنیم و length اونا رو چک کنیم

```
Object.entries(obj).length === 0 && obj.constructor === Object // Since date object
length is 0, you need to check constructor check as well
```

## .۱. استفاده از (+Object keys(ECMA 5

میتوانیم از object keys استفاده کنیم و length اونو چک کنیم

```
Object.keys(obj).length === 0 && obj.constructor === Object // Since date object length
is 0, you need to check constructor check as well
```

## .۱. استفاده از for-in با متدهای hasOwnProperty (Pre-ECMA 5

پارامتر رو با hasOwnProperty چک کنیم

```
function isEmpty(obj) {
 for(var prop in obj) {
 if(obj.hasOwnProperty(prop)) {
 return false;
 }
 }

 return JSON.stringify(obj) === JSON.stringify({});
}
```

## .۱۲۸ What is an arguments object.

شیء آرگومان ها یک شیء آرایه ماننده که در داخل توابع قابل دسترسی و حاوی مقادیر آرگومان های ارسال شده به اون تابعه.  
به عنوان مثال، ببینیم چگونه از شیء آرگومان ها در تابع sum استفاده کنیم

```
function sum() {
 var total = 0;
 for (var i = 0, len = arguments.length; i < len; ++i) {
 total += arguments[i];
 }
 return total;
}

sum(1, 2, 3) // returns 6
```

**نکته:** شما نمیتوانید از متدهای ارایه برای این شیء آرگومان ها استفاده کنید اما میتوانید به ارایه تبدیل کنید

```
var argsArray = Array.prototype.slice.call(arguments);
```

## .۱۲۹ How do you make first letter of the string in an uppercase.

میتوانیم با درست کردن یه تابع که با استفاده از زنجیره ای از متدهای استرینگ ها مثل `charAt` و `toUpperCase` یه استرینگ با حرف اول بزرگ ایجاد کرد `slice`

```

function capitalizeFirstLetter(string) {
 return string.charAt(0).toUpperCase() + string.slice(1);
}

```

## What are the pros and cons of for loop .۱۳۰

حلقه for یک سینتکس تکراری رایج در جاوا اسکریپت که هم مزایا و هم معایب دارد  
مزایا####

۱. توی همهی محیط ها env کار میکنه
۲. میتوانیم از break و continue برای کنول جریان داده استفاده کرد
۳. ممکنه با خطاهای یک به یک رو برو شین
۴. پر هزینه
۵. ضروریت

## How do you display the current date in javascript .۱۳۱

شما میتوانید از کلاس new Date() استفاده کنید که یه ابجکت از زمان و تاریخ بهمون میده بریم یه مثال ازش ببینید

```

var today = new Date();
var dd = String(today.getDate()).padStart(2, '0');
var mm = String(today.getMonth() + 1).padStart(2, '0'); //January is 0!
var yyyy = today.getFullYear();

today = mm + '/' + dd + '/' + yyyy;
document.write(today);

```

## How do you compare two date objects .۱۳۲

برای این کار باید از متده date().getTime() که بر روی قرار دارد استفاده کرد و نباید از اپراتور ها استفاده کنید

```

var d1 = new Date();
var d2 = new Date(d1);
console.log(d1.getTime() === d2.getTime()); //True
console.log(d1 === d2); // False

```

## How do you check if a string starts with another string .۱۳۳

ما میتوانید از متده startWith() که بر روی پرتوتایپ رشته ها وجود دارد استفاده کرد که یک رشته رو میگیره و چک میکنه که با اون شروع میشه رشته مورد نظر یا نه بریم یه مثال ببینید در موردن

```
"Good morning".startsWith("Good"); // true
"Good morning".startsWith("morning"); // false
```

## How do you trim a string in javascript . ۱۳۴

جاوا اسکریپت یه متده به ما میده به اسم trim() که روی استرینگ ها قرار داره با استفاده از این متده همه ی فضای خالی بین اون استرینگ برداشته میشه

```
" Hello World ".trim(); //Hello World
```

.If your browser(<IE9) doesn't support this method then you can use below polyfill

```
if (!String.prototype.trim) {
 (function() {
 // Make sure we trim BOM and NBSP
 var rtrim = /^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g;
 String.prototype.trim = function() {
 return this.replace(rtrim, '');
 };
 })();
}
```

## How do you add a key value pair in javascript . ۱۳۵

برای اضافه کردن key جدید به ابجک ها دو روش وجود دارن

```
var object = {
 key1: value1,
 key2: value2
};
```

: این روش زمانی موثر هستش که اسم پراپرتی رو میدونیم **Using dot notation** .

```
object.key3 = "value3";
```

: این روش وقتی موثره که اسم پراپرتی داینامیک باشه **Using square bracket notation** .

```
obj["key3"] = "value3";
```

## Is the !-- notation represents a special operator . ۱۳۶

نه! اون یه اپراتور خاص نیست اما ترکیب شده دو تا اپراتور استاندارد هستش یکی بعد اون یکی

۱. اپراتور نقیض (!)
۲. کاهش کننده (-)

اول یک شماره از مقدار متغیر به مثال کم میشه بعد تست میشه که مساوی صفر هستش یا نه که مشخص کننده درست یا غلط بودن شرط هستش

## How do you assign default values to variables . ۱۳۷

میتوانیم از عملگر یا اپراتور || استفاده تعريف یک مقدار دیفالت استفاده کرد مانند مثال زیر

```
var a = b || c;
```

مثال تعريف شده بالا مقدار متغیر a زمانی برابر مقدار متغیر c خواهد شد که b خالی یا undefined باشد

## How do you define multiline strings . ۱۳۸

ما میتوانیم از / برای تعريف کردن رشته های چند لایه استفاده کنیم برای مثال

```
var str = "This is a \
very lengthy \
sentence!";
```

اما اگه یه فاصله بعد / داشته باشیم, کد دقیقا به همون حالتی که هست نشون داده میشه اما یه ارور خطای نوشتری کد قراره داشته باشیم

## What is an app shell model . ۱۳۹

An application shell (or app shell) architecture is one way to build a Progressive Web App that reliably and instantly loads on your users' screens, similar to what you see in native applications. It is useful for getting some initial HTML to the screen fast without a network

## Can we define properties for functions . ۱۴۰

بله ما میتوانیم برای توابع پراپرتی تعیین کنیم چون توابع اصولا ابجکت هستن.

```
fn = function(x) {
 //Function code goes here
}

fn.name = "John";

fn.profile = function(y) {
 //Profile code goes here
}
```

## What is the way to find the number of parameters expected by a function . ۱۴۱

با استفاده کردن از `function.length` ما میتوانیم به تعداد پارامتر هایی که یه تابع انتظار داره بگیره دسترسی داشته باشیم  
برایم یه مثال درموردش ببینیم

```
function sum(num1, num2, num3, num4){
 return num1 + num2 + num3 + num4;
}
sum.length // 4 is the number of parameters expected.
```

## What is a polyfill . ۱۴۲

یه `polyfill` یه قسمت از کد جاوااسکریپتیه که با استفاده از اون ما میتوانیم توابع پیشرفته رو روی مروگرهایی که به طور طبیعی پشتیبانی نمیکنن استفاده کنیم. پلاگین `Silverlight` که برای تقلید کردن توابع بر روی `canvas` یا مروگ `IE7` استفاده کرد

## What are break and continue statements . ۱۴۳

دستور `break` برای "پرش به بیرون" از یک حلقه استفاده می شه. یعنی حلقه رو می شکنه و اجرای کد رو بعد از حلقه ادامه میده.

```
for (i = 0; i < 10; i++) {
 if (i === 5) { break; }
 text += "Number: " + i + "
";
}
```

دستور `continue` برای "پرش از روی" یک تکرار در حلقه استفاده می شه. یعنی یک تکرار (در حلقه) را می شکنه، اگر شرایط مشخصی رخ بده، و با تکرار بعدی در حلقه ادامه می ده.

```
for (i = 0; i < 10; i++) {
 if (i === 5) { continue; }
 text += "Number: " + i + "
";
}
```

## What are js labels . ۱۴۴

دستور `label` به ما اجازه می ده تا حلقه ها و بلوک ها را در جاوا اسکریپت نام گذاری کنیم. سپس می تونیم از این برچسبها برای مراجعه به کد بعداً استفاده کنیم. برای مثال، کد زیر با برچسب ها از چاپ اعداد در صورت یکسان بودن آنها جلوگیری می کند.

```

var i, j;

loop1:
for (i = 0; i < 3; i++) {
 loop2:
 for (j = 0; j < 3; j++) {
 if (i === j) {
 continue loop1;
 }
 console.log('i = ' + i + ', j = ' + j);
 }
}

// Output is:
// "i = 1, j = 0"
// "i = 2, j = 0"
// "i = 2, j = 1"

```

## What are the benefits of keeping declarations at the top .۱۴۵

توصیه می شه تمام اعلان ها رو بالای هر اسکریپت یا تابع نگه دارین. مزیت این کار

۱. کد ما تمیز تر میشه
۲. این یک مکان واحد برای جستجوی متغیرهای محلی فراهم می کنه
۳. میشه راحت از استفاده متغیر های ناخواسته جلوگیری کرد
۴. این کار محاسبات ناخواسته رو کمتر میکنه

## What are the benefits of initializing variables .۱۴۶

توضیه میشه که مقدار اولیه برای متغیرها تعیین بشه که دلایلشو چک میکنیم

۱. خروجیمون کد تمیز تری میشه
۲. این کار باعث میشه یه جا برای این متغیر رزرو بشه
۳. از برگشتن خطای undefined جلوگیری میشه

## What are the recommendations to create new object .۱۴۷

برای ساخت یه object با مقادیر پیشفرش مثال های زیر روش های ساخت مقادیر پیشفرض رو بررسی می کنیم

۱. استفاده از {} به جای new Object()
۲. استفاده از "" به جای new String()
۳. استفاده از 0 به جای new Number()
۴. استفاده از false به جای new Boolean()
۵. استفاده از [] به جای new Array()
۶. استفاده از ()/() به جای new RegExp()

.۷ استفاده از () به جای {}

بریم یه چن تا مثال بینیم

```
var v1 = {};
var v2 = "";
var v3 = 0;
var v4 = false;
var v5 = [];
var v6 = /(/;
var v7 = function(){};
```

## How do you define JSON arrays .۱۴۸

آرایه‌های JSON نوشته شده در داخل برآکت ها و ارایه‌هایی که دارای object هستند بریم یه مثال درموردش بینیم

```
"users": [
 {"firstName": "John", "lastName": "Abrahm"},

 {"firstName": "Anna", "lastName": "Smith"},

 {"firstName": "Shane", "lastName": "Warn"}]
```

## How do you generate random integers .۱۴۹

ما میتوانیم از متدهای Math.floor و Math.random برای رند کردن اون عدد استفاده کنیم حالا حاصل عدد به دست اومده رو ضربدر ده کنیم عددی بین یک تا ده خواهیم داشت

```
Math.floor(Math.random() * 10) + 1; // returns a random integer from 1 to 10
Math.floor(Math.random() * 100) + 1; // returns a random integer from 1 to 100
```

(Math.random() returns a random number between 0 (inclusive), and 1 (exclusive) نکته:

## Can you write a random integers function to print integers with in a range .۱۵۰

بله ما میتوانیم این تابع رو داشته باشیم که مقادیر حدکثر و حداقل رو بگیره و برای ما عدد رندوم ایجاد کنه بریم بینیم چطور همچین تابعی رو بنویسیم

```
function randomInteger(min, max) {
 return Math.floor(Math.random() * (max - min + 1)) + min;
}
randomInteger(1, 100); // returns a random integer from 1 to 100
randomInteger(1, 1000); // returns a random integer from 1 to 1000
```

## What is tree shaking .۱۵۱

تکان دادن درخت نوعی حذف کد مرده است. این بدان معناست که مازولهای استفاده نشده در طول فرآیند ساخت در بسته گنجانده نمی‌شوند و برای این کار بر ساختار استاتیک مازول ES2015 متکی است (یعنی واردات و صادرات). در ابتدا این باندل مازول 'ES2015' rollup رایج شد.

## What is the need of tree shaking . ۱۵۲

نوعی حذف کد مرده هستش. این بد این معنیه که مازولهای استفاده نشده در طول فرآیند ساخت در بسته گنجونده نمی‌شن و برای آن بر ساختار استاتیک مازول ES2015 متکی است (یعنی واردات و صادرات). در ابتدا این باندلر مازول 'ES2015' rollup رایج شد.

## Is it recommended to use eval . ۱۵۳

خیر، اجازه اجرای کد دلخواه را می‌دهد که باعث ایجاد مشکل امنیتی می‌شود. همانطور که می‌دانیم از تابع eval() برای اجرای متن به عنوان کد استفاده می‌شود. در بیشتر موارد استفاده از آن ضروری نیست.

## What is a Regular Expression . ۱۵۴

یا همون Regex یه تواليه که يه ساختار جستجو ايجاد ميکنه با استفاده از اين ساختاز ما ميتوينيم ديتامون رو بر اساس يه ساختار که مينويسم جستجو کنيم و به قولی ديتامون رو اعتبارسنجي کنيم /pattern/modifiers;

برای مثال Regex حساس به حروف کوچک و بزرگ زبان انگلیسی به صورت رير نوشته ميشه /John/i

## What are the string methods available in Regular expression . ۱۵۵

دو تا متده برای رشتةها داره : search() و replace(). متده search() عبارت رو میگیره اونو جسنجو ميکنه و محل اون عبارت رو برميگردونه

```
var msg = "Hello John";
var n = msg.search(/John/i); // 6
```

متده replace() برای برگرداندن رشتة اصلاح شده در جایی که الگو جايگزين ميشه استفاده ميشه

```
var msg = "Hello John";
var n = msg.replace(/John/i, "Buttler"); // Hello Buttler
```

## What are modifiers in regular expression . ۱۵۶

Modifier میتوان زمانی استفاده بشن که به جستجو های بدون حروف کوچک و بزرگ سراسری نیاز داریم بیاین یه مثال درموردشون ببینیم

| توضیح                                | اصلاح کننده |
|--------------------------------------|-------------|
| تطبيق حساس به حروف                   | i           |
| تطبيق کلی به جای توقف در اولین تشابه | g           |
| تطبيق چندخطی                         | m           |

بریم یه مثال از modifier گلوبال ببینیم

```
var text = "Learn JS one by one";
var pattern = /one/g;
var result = text.match(pattern); // one,one
```

## What are regular expression patterns . ۱۵۷

Regex یه گروهی از ساختار ها برای اینکه مدل طبیعی میشن

۱. **براکت ها:** این ها استفاده میشن تا یه رنجی از کاراکتر را پیدا کنن  
برای مثال پایین چن تا مورد استفاده لیست شدنهاي
۲. [abc]: استفاده میشه تا هر کاراکتری بین این سه کاراکتر پیدا بشه
۳. [0-9]: استفاده میشه تا ارقام بین این دو عدد پیدا بشه
۴. (a|b): برای پیدا کردن هر یک از گزینه های جدا شده با | استفاده میشه
۵. کاراکتر برابر با: این عبارت ها کاراکتر هایی با معنی خاص هستند  
برای مثال پایین دو تا مورد که استفاده میشه ازشون رو ببینیم
۶. \d: استفاده برای پیدا کردن اعداد
۷. \s: استفاده برای پیدا کردن فاصله ها
۸. \b: استفاده برای پیدا کردن کاراکتر های همخوانی داشته با شروع شدن یا پایانشون
۹. **کمیت کننده ها:** این ها برای تعریف کمیت ها موثر هستن  
برای مثال پایین دو تا مورد استفاده برآشون اوردیم
۱۰. +n: برای پیدا کردن رشته همخوانی داشته با حداقل یک کاراکتر
۱۱. n\*: برای پیدا کردن همخوانی هر رشته شامل صفر یا بیشتر
۱۲. n?: برای پیدا کردن هر رشته که شامل صفر یا یک کاراکتر میشه

## What is a RegExp object . ۱۵۸

object های Regex یه عبارت معمولی با پراپرتی ها و متدهای تعریف شده از قبله بریم یه مثال از نحوه استفادشون ببینیم.

```
var regexp = new RegExp('\\w+');
console.log(regexp);
// expected output: /\w+/
```

## How do you search a string for a pattern . ۱۵۹

می‌توانید از متد `test()` عبارت منظم برای جستجوی یک رشته برای الگو استفاده کنید و بسته به نتیجه، `true` یا `false` را برگردانید.

```
var pattern = /you/;
console.log(pattern.test("How are you?")); //true
```

## What is the purpose of exec method . ۱۶۰

هدف متد `exec` شبیه به روش `test` است، اما جستجوی یک تطابق در یک رشته مشخص را انجام می‌دهد و یک آرایه نتیجه `null` را به جای برگرداندن `true/false` برمی‌گرداند.

```
var pattern = /you/;
console.log(pattern.exec("How are you?")); //["you", index: 8, input: "How are you?", groups: undefined]
```

## How do you change the style of a HTML element . ۱۶۱

شما می‌توانید سبک درون خطی یا نام کلاس یک عنصر HTML را با استفاده از جاوا اسکریپت تغییر دهید  
۱. استفاده از پرپرتبی استایل: با استفاده از ویژگی `style` می‌توانید استایل درون خطی را تغییر دهید

```
document.getElementById("title").style.fontSize = "30px";
```

۲. استفاده از پرپرتبی `className`: تغییر کلاس عنصر با استفاده از ویژگی `className` آسان است

```
document.getElementById("title").style.className = "custom-title";
```

## 'What would be the result of 1+2+'3 . ۱۶۲

خروجی '33' خواهد بود. از آنجایی که «1» و «2» مقادیر عددی هستند، نتیجه دو رقم اول یک مقدار عددی «3» خواهد بود.  
رقم بعدی یک مقدار نوع رشته است، زیرا افزودن مقدار عددی «3» و مقدار رشته «3» فقط یک مقدار الحاقی «33» خواهد بود.

## What is a debugger statement . ۱۶۳

دستور دیباگر هر گونه عملکرد اشکال زدایی موجود را فراخوانی می کند، مانند تعیین نقطه شکست. اگر هیچ عملکرد اشکال زدایی در دسترس نباشد، این عبارت تاثیری ندارد.  
به عنوان مثال، در تابع زیر یک دستور دیباگر درج شده است. بنابراین اجرا در دستور دیباگر مانند یک نقطه شکست در منبع اسکریپت متوقف می شود.

```
function getProfile() {
// code goes here
debugger;
// code goes here
}
```

## What is the purpose of breakpoints in debugging . ۱۶۴

پس از اجرای دستور دیباگر و باز شدن پنجره دیباگر، می توانید نقاط شکست را در کد جاوا اسکریپت تنظیم کنید. در هر نقطه شکست، جاوا اسکریپت اجرا نمی شود و به شما اجازه می دهد مقادیر جاوا اسکریپت را بررسی کنید. پس از بررسی مقادیر، می توانید با استفاده از دکمه پخش، اجرای کد را از سر بگیرید.

## Can I use reserved words as identifiers . ۱۶۵

نه، شما نمی توانید از کلمات رزرو شده به عنوان متغیر، برچسب، نام شی یا تابع استفاده کنید. بیایید یک مثال ساده را ببینیم،

```
var else = "hello"; // Uncaught SyntaxError: Unexpected token else
```

## How do you detect a mobile browser . ۱۶۶

ما می توانیم با استفاده از Regex که یک boolean به ما برمیگردانه بفهمیم که کاربر داره ازش استفاده میکنه چیه.

```

window.mobilecheck = function() {
 var mobileCheck = false;
 (function(a)
 {if((/android|bb\d+|meego).+mobile|avantgo|bada\/*blackberry|blazer|compal|elaine|fennec|h|
 |maemo|midp|mmp|mobile.+firefox|netfront|opera m(ob|in)i|palm(os)?|
 |phone|p(ixi|re)\/*plucker|pocket|psp|series(4|6)0|symbian|treo|up\.
 (browser|link)|vodafone|wap|windows
 ce|xda|xiino/i.test(a) |||1207|6310|6590|3gso|4thp|50[1-6]i|770s|802s|a
 wa|abac|ac(er|oo|s\-)|ai(ko|rn)|al(av|ca|co)|amo(i|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|at
 m|r|s)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-
 (n|u)|c55\/*capi|ccwa|cdm\-*cell|chtm|cldc|cmd\-*co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-
 s|devi|dica|dmob|do(c|p)o|ds(12|\-d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-
 7]0|os|wa|ze)|fetc|fly(\-|_)|g1 u|g560|gene|gf\-5|g\-
 mo|go(\.w\od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-(hi(pt|ta)|hp(i|ip)|hs\|-c|ht(c(\-|
 |_a|g|p|s|t)|tp)|hu(aw|tc)|i\-(20|go|ma)|i230|iac(\-
 |\|)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|jigs|kddi|keji|kgt(
 |\|)|klon|kpt |kwc\-*kyo(c|k)|le(no|xi)|lg(g|\/(k|l|u)|50|54|\-[a-w])|libw|lynx|m1\-
 w|m3ga|m50\/*ma(te|ui|xo)|mc(01|21|ca)|m\-
 cr|me(rc|ri)|mi(o8|oa|ts)|mmef|mo(01|02|bi|de|do|t(\-|_o|v)|zz)|mt(50|p1|v
)|mwbp|mywa|n10[0-2]|n20[2-3]|n30(0|2)|n50(0|2|5)|n7(0|0|1)|10)|ne((c|m)\-
 |on|tf|wf|wg|wt)|nok(6|i)|nzph|o2im|op(ti|wv)|oran|owg1|p800|pan(a|d|t)|pdxg|pg(13|\-([1-
 8]|c))|phil|pire|pl(ay|uc)|pn\-2|po(ck|rt|se)|prox|psio|pt\-*g|qa\-*a|qc(07|12|21|32|60|\-
 [2-7]|i\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/*sa(ge|ma|mm|ms|ny|va)|sc(01|h\-
 |oo|p\-)|sdk\/*se(c(\-|0|1)|47|mc|nd|ri)|sgh\-*shar|sie(\-
 |m)|sk\-*sl(45|id)|sm(al|ar|b3|it|t5)|so(ft|ny)|sp(01|h\-*v\-*v
)|sy(01|mb)|t2(18|50)|t6(00|10|18)|ta(gt|lk)|tcl\-*tdg\-*tel(i|m)|tim\-*t\-
 mo|to(pl|sh)|ts(70|m\-
 |m3|m5)|tx\-*up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5[0-3])\-
 v)|vm40|voda|vulc|vx(52|53|60|61|70|80|81|83|85|98)|w3c(\-|)|webc|whit|wi(g
 |nc|nw)|wmlb|wonu|x700|yas\-*your|zeto|zte\-*i.test(a.substr(0,4))) mobileCheck = true;})
(navigator.userAgent||navigator.vendor||window.opera);
 return mobileCheck;
};

```

## How do you detect a mobile browser without regexp .۱۶۷

می‌توانیم مرورگرهای تلفن همراه را با اجرای فهرستی از دستگاه‌ها و بررسی اینکه آیا `useragent` با چیزی مطابقت دارد یا خیر، شناسایی کنیم. این یک راه حل جایگزین برای استفاده از RegExp هستش

```

function detectmob() {
 if(navigator.userAgent.match(/Android/i)
 || navigator.userAgent.match(/webOS/i)
 || navigator.userAgent.match(/iPhone/i)
 || navigator.userAgent.match(/iPad/i)
 || navigator.userAgent.match(/iPod/i)
 || navigator.userAgent.match(/BlackBerry/i)
 || navigator.userAgent.match(/Windows Phone/i)
) {
 return true;
}
else {
 return false;
}
}

```

## How do you get the image width and height using JS .۱۶۸

ما می‌توانیم با استفاده از جاوا اسکریپت می‌توانید به صورت برنامه‌ریزی شده تصویر را بدست بیاریم و ابعاد (عرض و ارتفاع) رو بررسی کنیم.

```

var img = new Image();
img.onload = function() {
 console.log(this.width + 'x' + this.height);
}
img.src = 'http://www.google.com/intl/en_ALL/images/logo.gif';

```

## How do you make synchronous HTTP request .۱۶۹

مرورگرها یک شی XMLHttpRequest ارائه می‌دان که می‌توانه برای ایجاد درخواست‌های HTTP همزمان از جاوا اسکریپت استفاده شه.

```

function httpGet(theUrl) {
 var xmlhttpReq = new XMLHttpRequest();
 xmlhttpReq.open("GET", theUrl, false); // false for synchronous request
 xmlhttpReq.send(null);
 return xmlhttpReq.responseText;
}

```

## How do you make asynchronous HTTP request .۱۷۰

مرورگرها یک شی XMLHttpRequest را ارائه می‌دان که می‌توان برای درخواست‌های HTTP ناهمزمان از جاوا اسکریپت با ارسال پارامتر سوم به عنوان true استفاده کنن.

```

function httpGetAsync(theUrl, callback)
{
 var xmlhttpReq = new XMLHttpRequest();
 xmlhttpReq.onreadystatechange = function() {
 if (xmlhttpReq.readyState == 4 && xmlhttpReq.status == 200)
 callback(xmlhttpReq.responseText);
 }
 xmlhttp.open("GET", theUrl, true); // true for asynchronous
 xmlhttp.send(null);
}

```

## How do you convert date to another timezone in javascript .۱۷۱

می‌توانیم از متدهای `toLocaleString()` برای تبدیل تاریخ‌ها در یک منطقه زمانی به منطقه زمانی دیگر استفاده کنیم. برایم به مثال در مرورگر بینیم.

```
console.log(event.toLocaleString('en-GB', { timeZone: 'UTC' })); //29/06/2019, 09:56:00
```

## What are the properties used to get size of window .۱۷۲

می‌توانیم از ویژگی‌های `innerWidth`, `innerHeight`, `clientWidth`, `clientHeight` و `width` و `height` اشیاء بدنی سند برای یافتن اندازه یک پنجره استفاده کنیم. باید از ترکیب آنها برای محاسبه اندازه یک پنجره یا سند استفاده کنیم.

```

var width = window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;

var height = window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;

```

## What is a conditional operator in javascript .۱۷۳

عملگر شرطی (ternary) تنها عملگر جاوا اسکریپت هستش که سه عملوند را می‌گیره که به عنوان میانبر برای دستور `if` عمل می‌کنه.

```

var isAuthenticated = false;
console.log(isAuthenticated ? 'Hello, welcome' : 'Sorry, you are not authenticated');
//Sorry, you are not authenticated

```

## Can you apply chaining on conditional operator .۱۷۴

بله، می‌توانید زنجیره‌سازی را روی عملگرهای شرطی مشابه if ... else if ... else if... other chain اعمال کنیم.

```
function traceValue(someParam) {
 return condition1 ? value1
 : condition2 ? value2
 : condition3 ? value3
 : value4;
}

// The above conditional operator is equivalent to:
```

```
function traceValue(someParam) {
 if (condition1) { return value1; }
 else if (condition2) { return value2; }
 else if (condition3) { return value3; }
 else { return value4; }
}
```

## What are the ways to execute javascript after page load . ۱۷۵

ما می‌توانیم جاوا اسکریپت را پس از بارگذاری صفحه به روش‌های مختلف اجرا کنیم.  
:window.onload .

```
window.onload = function ...
```

:document.onload .

```
document.onload = function ...
```

:body onload .

```
<body onload="script();">
```

## What is the difference between proto and prototype . ۱۷۶

شی \_\_proto\_\_ شی واقعیه که در زنجیره جستجو برای حل متدها و غیره استفاده می‌شود. در حالی که prototype شیئی است که برای ساخت \_\_proto\_\_ استفاده می‌شود زمانی که یک شی با جدید ایجاد می‌کنید.

```
(new Employee).__proto__ === Employee.prototype;
(new Employee).prototype === undefined;
```

## Give an example where do you really need semicolon . ۱۷۷

توصیه می‌شود بعد از هر عبارت در جاوا اسکریپت از نقطه ویرگول استفاده کنیم. به عنوان مثال، در مورد زیر به دلیل از دست دادن نقطه ویرگول، خطای is not a function .. را در زمان اجرا میندازد.

```
// define a function
var fn = function () {
 //...
} // semicolon missing at this line

// then execute some code inside a closure
(function () {
 //...
})();
```

و از آن تعبیر خواهد شد

```
var fn = function () {
 //...
}(function () {
 //...
})();
```

در این حالت، تابع دوم رو به عنوان آرگومان به تابع اول ارسال می کنیم و سعی می کنیم نتیجه فراخوانی تابع اول را به عنوان تابع فراخوانی کنیم. از این رو، تابع دوم با خطای "... یک تابع نیست" در زمان اجرا ارور می گیریم.

## What is a freeze method .۱۷۸

متده **freeze()** برای فریز کردن یک شی استفاده می شه. ثابت کردن یک شی اجازه افزودن ویژگی های جدید به یک شی رو نمی ده. از حذف آن جلوگیری می کنه و از تغییر قابلیت شمارش پذیری، پیکربندی یا قابلیت نوشتن ویژگی های موجود جلوگیری می کنه. یعنی شیء ارسال شده را برمی گردونه و کپی ثابتی ایجاد نمی کنه.

```
const obj = {
 prop: 100
};

Object.freeze(obj);
obj.prop = 200; // Throws an error in strict mode

console.log(obj.prop); //100
```

**نکته:** یه تایپ ارور بهمون می ده که ارگومان داده شده object نیست

## What is the purpose of freeze method .۱۷۹

مزیت استفاده کردن از متده **frefreeze** رو ببینیم

۱. برای فریز کردن ابجکت ها و آرایه ها
۲. برای **immutable** کردن ابجکت ها

## Why do I need to use freeze method .۱۸۰

در پارادایم شی گرا، یک API موجود حاوی عناصر خاصی است که قصد توسعه، اصلاح یا استفاده مجدد در خارج از زمینه فعلی خود را ندارند. از این رو، این کلمه کلیدی final است که در زبان های مختلف استفاده می شود.

## How do you detect a browser language preference .۱۸۱

ما میتوانیم از ابجکت navigator گه بر روری بروز و وجود داره این کارو انجام بدیم

```
var language = navigator.languages && navigator.languages[0] || // Chrome / Firefox
 navigator.language || // All browsers
 navigator.userLanguage; // IE <= 10

console.log(language);
```

## How to convert string to title case with javascript .۱۸۲

این کار باعث میشه که حرف اول یه رشته به صورت بزرگ(زبان انگلیسی) نشون داده بشه که ما میتوانیم با تابع زیر این کارو انجام بدیم

```
function toTitleCase(str) {
 return str.replace(
 /\w\S*/g,
 function(txt) {
 return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCase();
 }
);
}

toTitleCase("good morning john"); // Good Morning John
```

## How do you detect javascript disabled in the page .۱۸۳

برای تشخیص غیرفعال بودن یا نبودن جawa اسکریپت می توانید از تگ <noscript> استفاده کنید. بلوک کد داخل <noscript> زمانی اجرا می شود که جawa اسکریپت غیرفعال است، و معمولاً برای نمایش محتوای جایگزین زمانی که صفحه در جawa اسکریپت تولید می شود، استفاده میشه.

```
<script type="javascript">
 // JS related code goes here
</script>
<noscript>
 JavaScript is disabled in the page. Please click
Next Page
</noscript>
```

## What are various operators supported by javascript . ۱۸۴

یک عملگر قادر به دستکاری (محاسبات ریاضی و منطقی) مقدار یا عملوند معینیه. اپراتورهای مختلفی توسط جاوا اسکریپت پشتیبانی میشن این اپراتور ها هستن

۱. **عملگر های حسابی:** شامل + (اضافه)، - (منها)، \* (ضرب)، / (تقسیم)، % (درصد)، + (اضافه کردن) و - (کم کردن)

۲. **عملگر های مقایسه ای:** شامل == (برابر)، != (غیر برابر)، < (بزرگتر)، > (بزرگتر مساوی)، <= (کوچکتر)، >= (کوچکتر مساوی)

۳. **عملگر های منطقی:** شامل && ("و" منطقی)، || ("یا" منطقی)، !(منطقی "نه")

۴. **عملگر های تعیین مقدار:** شامل = (اپراتور تعیین مقدار)، += (اضافه کردن و تعیین مقدار)، -= (منها کردن و تعیین مقدار)، \*= (ضرب و تعیین مقدار)، /= (تقسیم و تعیین مقدار)، %= (باقي مانده و تعیین مقدار)

۵. **اپراتور سه تایی:** شامل اپراتور های شرطی سه تایی

۶. **اپراتور تایپ:** از اون برای پیدا کردن تایپ متغیر ها استفاده میشه به صورت `typeof variable`

## What is a rest parameter . ۱۸۵

پارامتر Rest یک روش بهبود یافته برای مدیریت پارامترهای تابع هستش که به ما امکان میده تعداد نامحدودی از آرگومان ها رو به عنوان یک آرایه نمایش میدیم

```
function f(a, b, ...theArgs) {
 // ...
}
```

به عنوان مثال، بباید یک مثال مجموع برای محاسبه تعداد پویا پارامترها در نظر بگیریم،

```
function total(...args){
 let sum = 0;
 for(let i of args){
 sum+=i;
 }
 return sum;
}
console.log(fun(1,2)); //3
console.log(fun(1,2,3)); //6
console.log(fun(1,2,3,4)); //13
console.log(fun(1,2,3,4,5)); //15
```

نکته: Rest parameter is added in ES2015 or ES6

## What happens if you do not use rest parameter as a last argument . ۱۸۶

پارامتر Rest باید آخرین آرگومان باشد، زیرا وظیفه آن جمع آوری تمام آرگومان های باقی مانده در یک آرایه هست. به عنوان مثال، اگر تابعیو مثل زیر تعریف کنیم معنی ندارد و یک خطا ایجاد می کنه.

```
function someFunc(a,...b,c){
 //Your code goes here
 return;
}
```

## What are the bitwise operators available in javascript . ۱۸۷

در زیر لیستی از عملگرهای منطقی بیتی مورد استفاده در جاوا اسکریپت آمده است

۱. به صورت بیتی AND (&)
۲. به صورت بیتی OR (|)
۳. به صورت بیتی XOR (^)
۴. به صورت بیتی NOT (~)
۵. تغییر مکان به چپ («>)
۶. علامت در حال انتشار به سمت راست («><) )
۷. صفر پر کردن Shift راست («<<) )

## What is a spread operator . ۱۸۸

عملگر Spread به تکرارپذیرها (آرایه ها / اشیاء / رشته ها) اجازه می دهد تا به آرگومان ها / عناصر منفرد گسترش یابند. برای مشاهده این رفتار مثالی بزنیم،

```
function calculateSum(x, y, z) {
 return x + y + z;
}

const numbers = [1, 2, 3];

console.log(calculateSum(...numbers)); // 6
```

## How do you determine whether object is frozen or not . ۱۸۹

متده Object.isFrozen() برای تعیین اینکه آیا یک شی منجمد است یا خیر استفاده می شود. اگر همه شرایط زیر درست باشد، یک شی منجمد می شود.

۱. اگر قابل توسعه نباشد.
۲. اگر تمام خصوصیات آن غیر قابل تنظیم باشند.
۳. اگر تمام خصوصیات داده آن غیر قابل نوشتمن باشد.

استفاده به صورت زیر خواهد بود

```

const object = {
 property: 'Welcome JS world'
};
Object.freeze(object);
console.log(Object.isFrozen(object));

```

## How do you determine two values same or not using object .۱۹۰

متده است `Object.is()` تعیین می کند که آیا دو مقدار یک مقدار هستند یا خیر. به عنوان مثال، استفاده با انواع مختلف مقادیر،

```

Object.is('hello', 'hello'); // true
Object.is(window, window); // true
Object.is([], []); // false

```

اگر یکی از موارد زیر برقرار باشد، دو مقدار یکسان هستند:

۱. هردو `undefined`

۲. هردو `null`

۳. هردو `true` یا هر دو `false`

۴. هر دو رشته با طول یکسان با کاراکترهای مشابه به ترتیب یکسان

۵. هر دو یک شی (یعنی هر دو شی مرجع یکسان دارند)

۶. هر دو عدد و

۷. هر دو `+0`

۸. هر دو `-0`

۹. هر دو `NaN`

هر دو غیر صفر و هر دو `NaN` نیستند و هر دو دارای یک مقدار هستند.

## What is the purpose of using object is method .۱۹۱

برخی از کاربردهای متده است `Object.is()` به شرح زیر است،

۱. برای مقایسه دو رشته استفاده می شود.

۲. برای مقایسه دو عدد استفاده می شود.

۳. برای مقایسه قطبیت دو عدد استفاده می شود.

۴. برای مقایسه دو شی استفاده می شود.

## How do you copy properties from one object to other .۱۹۲

می توانید از متده است `Object.assign()` استفاده کنید که برای کپی کردن مقادیر و ویژگی ها از یک یا چند شی منبع به یک شی هدف استفاده می شود. شی مورد نظر را که دارای خواص و مقادیر کپی شده از شی مورد نظر است، برمی گرداند. نحوه صورت زیر خواهد بود،

```
Object.assign(target, ...sources)
```

بیایید با یک منبع و یک شی هدف مثال بزنیم،

```
const target = { a: 1, b: 2 };
const source = { b: 3, c: 4 };

const returnedTarget = Object.assign(target, source);

console.log(target); // { a: 1, b: 3, c: 4 }

console.log(returnedTarget); // { a: 1, b: 3, c: 4 }
```

همانطور که در کد بالا مشاهده شد، یک ویژگی مشترک ( b ) از منبع به مقصد وجود دارد، بنابراین مقدار آن بازنویسی شده است.

## What are the applications of assign method . ۱۹۳

در زیر تعدادی از کاربردهای اصلی متد ()Object.assign آورده شده است.

- 1. برای شبیه سازی یک شی استفاده می شود .
  - 2. برای ادغام اشیاء با ویژگی های یکسان استفاده می شود .
- \*\*[فهرست] (#فهرست)

## What is a proxy object . ۱۹۴

شیء Proxy برای تعریف رفتار سفارشی برای عملیات های اساسی مانند جستجوی ویژگی، تخصیص، شمارش، فراخوانی تابع و غیره استفاده می شود. نحوه شرح زیر است:

```
var p = new Proxy(target, handler);

var handler = {
 get: function(obj, prop) {
 return prop in obj ?
 obj[prop] :
 100;
 }
};

var p = new Proxy({}, handler);
p.a = 10;
p.b = null;

console.log(p.a, p.b); // 10, null
console.log('c' in p, p.c); // false, 100
```

بیایید مثالی از شیء پروکسی بزنیم،

در کد بالا، از کنترل کننده «get» استفاده می کند که رفتار پراکسی را هنگام انجام عملیات روی آن تعریف می کند.

## What is the purpose of seal method . ۱۹۵

روش **Object.seal()** برای مهر و موم کردن یک شی، با جلوگیری از اضافه شدن ویژگی های جدید به آن و علامت گذاری تمام ویژگی های موجود به عنوان غیر قابل تنظیم، استفاده می شود. اما مقادیر ویژگی های فعلی تا زمانی که قابل نوشتن باشند همچنان قابل تغییر هستند. بیایید مثال زیر را برای درک بیشتر در مورد روش **seal()** ببینیم

```
const object = {
 property: 'Welcome JS world'
};
Object.seal(object);
object.property = 'Welcome to object world';
console.log(Object.isSealed(object)); // true
delete object.property; // You cannot delete when sealed
console.log(object.property); //Welcome to object world
```

## What are the applications of seal method . ۱۹۶

در زیر کاربردهای اصلی متده **Object.seal()** آورده شده است.

۱. برای آب بندی اشیا و آرایه ها استفاده می شود.
۲. برای غیرقابل تغییر کردن یک جسم استفاده می شود.

## What are the differences between freeze and seal methods . ۱۹۷

اگر یک شی با استفاده از متده **Object.freeze()** منجمد شود، ویژگی های آن تغییرناپذیر می شوند و هیچ تغییری در آنها نمی توان ایجاد کرد، در حالی که اگر یک شی با استفاده از متده **Object.seal()** مهر و موم شده باشد، می توان تغییرات را در ویژگی های موجود ایجاد کرد. از شی

## How do you determine if an object is sealed or not . ۱۹۸

متده **Object.isSealed()** برای تعیین مهر و موم بودن یا نبودن یک شی استفاده می شود. اگر همه شرایط زیر درست باشد یک شی مهر و موم می شود

۱. اگر قابل توسعه نباشد.
۲. اگر تمام خصوصیات آن غیر قابل تنظیم باشند.
۳. اگر قابل جابجایی نباشد (اما لزوماً غیرقابل نوشتن نیست).

بیایید آن را در عمل ببینیم

```

const object = {
 property: 'Hello, Good morning'
};

Object.seal(object); // Using seal() method to seal the object

console.log(Object.isSealed(object)); // checking whether the object is sealed or not

```

## How do you get enumerable key and value pairs .۱۹۹

متدهای Object.entries() برای برگرداندن آرایه‌ای از جفت‌های [key, value] دارای کلید رشته‌ای شمارش‌پذیر یک شی معین، به همان ترتیبی که توسط یک حلقه for...in ارائه می‌شود، استفاده می‌شود. بیایید عملکرد متدهای Object.entries() را در یک مثال بینیم،

```

} = const object
,'a: 'Good morning
b: 100
;{

} ((for (let [key, value] of Object.entries(object
 ;(`{console.log(`${key}: ${value
'a: 'Good morning //
b: 100 //
{

```

**نکته:** سفارش به عنوان شی تعریف شده تضمین نمی‌شود.

## What is the main difference between Object.values and Object.entries method

روش Object.entries() مشابه روش Object.values() است اما به جای جفت [key,value] آرایه‌ای از مقادیر را برمی‌گرداند.

```

const object = {
 a: 'Good morning',
 b: 100
};

for (let value of Object.values(object)) {
 console.log(`{$value}`); // 'Good morning' 100
}

```

## How can you get the list of keys of any object

می‌توانید از متده استفاده کنید که برای برگرداندن آرایه‌ای از نام ویژگی‌های یک شی معین استفاده می‌شود، به همان ترتیبی که با یک حلقه معمولی دریافت می‌کنیم. به عنوان مثال، می‌توانید کلیدهای یک شی کاربر را دریافت کنید،

```
const user = {
 name: 'John',
 gender: 'male',
 age: 40
};

console.log(Object.keys(user)); // ['name', 'gender', 'age']
```

## How do you create an object with prototype

متده Object.create() برای ایجاد یک شی جدید با شی نمونه اولیه و ویژگی‌های مشخص شده استفاده می‌شود. به عنوان مثال، از یک شی موجود به عنوان نمونه اولیه شی جدید ایجاد شده استفاده می‌کند. یک شی جدید را با شی نمونه اولیه و ویژگی‌های مشخص شده برمی‌گرداند.

```
const user = {
 name: 'John',
 printInfo: function () {
 console.log(`My name is ${this.name}.`);
 }
};

const admin = Object.create(user);

admin.name = "Nick"; // Remember that "name" is a property set on "admin" but not on "user" object

admin.printInfo(); // My name is Nick
```

## What is a WeakSet

برای ذخیره مجموعه‌ای از اشیاء ضعیف (مرجع ضعیف) استفاده می‌شود. نحوه به صورت زیر خواهد بود

```
new WeakSet([iterable]);
```

باید مثال زیر را برای توضیح رفتار آن ببینیم،

```
var ws = new WeakSet();
var user = {};
ws.add(user);
ws.has(user); // true
ws.delete(user); // removes user from the set
ws.has(user); // false, user has been removed
```

## What are the differences between WeakSet and Set

تفاوت اصلی این است که ارجاع به اشیاء در Set قوی است در حالی که ارجاع به اشیا در WeakSet ضعیف است. به عنوان مثال، یک شی در WeakSet می‌تواند زباله جمع آوری شود اگر مرجع دیگری به آن وجود نداشته باشد.

تفاوت‌های دیگر عبارتند از

۱. مجموعه‌ها می‌توانند هر مقداری را ذخیره کنند در حالی که WeakSets می‌تواند تنها مجموعه‌ای از اشیاء را ذخیره کند
۲. WeakSet برخلاف Set دارای ویژگی اندازه نیست
۳. WeakSet متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی‌ها، forEach را ندارد.
۴. قابل تکرار نیست.

## List down the collection of methods available on WeakSet

در زیر لیستی از روش‌های موجود در WeakSet آمده است،

۱. add(value): یک شی جدید با مقدار داده شده به مجموعه ضعیف اضافه می‌شود
۲. delete(value): مقدار را از مجموعه WeakSet حذف می‌کند.
۳. has(value): اگر مقدار در مجموعه WeakSet وجود داشته باشد true را برمی‌گرداند، در غیر این صورت false را برمی‌گرداند.
۴. length(): طول ضعیف SetObject را برمی‌گرداند

بیایید عملکرد تمام روش‌های بالا را در یک مثال ببینیم،

```
var weakSetObject = new WeakSet();
var firstObject = {};
var secondObject = {};
// add(value)
weakSetObject.add(firstObject);
weakSetObject.add(secondObject);
console.log(weakSetObject.has(firstObject)); //true
console.log(weakSetObject.length()); //2
weakSetObject.delete(secondObject);
```

## What is a WeakMap

شی WeakMap مجموعه‌ای از جفت‌های کلید/مقدار است که در آن کلیدها به صورت ضعیف ارجاع داده شده اند. در این حالت، کلیدها باید اشیا باشند و مقادیر می‌توانند مقادیر دلخواه باشند. نحوه شکل زیر است،

```
([new WeakMap([iterable
```

بیایید مثال زیر را برای توضیح رفتار آن ببینیم،

```

var ws = new WeakMap();
var user = {};
ws.set(user);
ws.has(user); // true
ws.delete(user); // removes user from the map
ws.has(user); // false, user has been removed

```

## What are the differences between WeakMap and Map

تفاوت اصلی این است که ارجاعات به اشیاء کلیدی در نقشه قوی هستند در حالی که ارجاعات به اشیاء کلیدی در WeakMap ضعیف هستند. به عنوان مثال، یک شی کلیدی در WeakMap در صورتی که هیچ مرجع دیگری به آن وجود نداشته باشد، می‌تواند زباله جمع آوری شود.

تفاوت‌های دیگر عبارتند از

۱. نقشه‌ها می‌توانند هر نوع کلیدی را ذخیره کنند، در حالی که WeakMaps فقط می‌توانند مجموعه‌ای از اشیاء کلیدی را ذخیره کنند.
۲. برخلاف Map دارای ویژگی اندازه نیست WeakMap.
۳. متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی‌ها، forEach را ندارد.
۴. قابل تکرار نیست WeakMap.

## List down the collection of methods available on WeakMap

- در زیر لیستی از روش‌های موجود در WeakMap آمده است،
۱. مقدار کلید را در شی WeakMap تنظیم می‌کند. شی WeakMap را برمی‌گرداند. (set)(key, value).
  ۲. هر مقدار مربوط به کلید را حذف می‌کند. (delete)(key).
  ۳. یک Boolean را برمی‌گرداند که نشان می‌دهد آیا مقداری به کلید در شی WeakMap مرتبط شده است یا خیر. (has)(key).
  ۴. مقدار مربوط به کلید را برمی‌گرداند، یا اگر کلیدی وجود نداشته باشد، تعریف نشده است. (get)(key).
- بیایید عملکرد تمام روش‌های بالا را در یک مثال ببینیم،

```

var weakMapObject = new WeakMap();
var firstObject = {};
var secondObject = {};
// set(key, value)
weakMapObject.set(firstObject, 'John');
weakMapObject.set(secondObject, 100);
console.log(weakMapObject.has(firstObject)); //true
console.log(weakMapObject.get(firstObject)); // John
weakMapObject.delete(secondObject);

```

## What is the purpose of uneval

(uneval) یک تابع داخلی است که برای ایجاد نمایش رشته ای از کد منبع یک شی استفاده می شود. این یک تابع سطح بالا است و با هیچ شیئی مرتبط نیست. بباید مثال زیر را ببینیم تا در مورد عملکرد آن بیشتر بدانید،

```
var a = 1;
uneval(a); // returns a String containing 1
uneval(function user() {}); // returns "(function user(){})"
```

## How do you encode an URL

تابع () encodeURI برای رمزگذاری URI کامل استفاده می شود که دارای کarakترهای خاص به جز (# \$ + = & @ ? : / ) است.

```
var uri = 'https://mozilla.org/?x=шеллы';
var encoded = encodeURI(uri);
console.log(encoded); // https://mozilla.org/?x=%D1%88%D0%B5%D0%BB%D0%BB%D1%8B
```

## How do you decode an URL

تابع () decodeURI برای رمزگشایی یک شناسه منبع یکنواخت (URI) که قبل از تابع encodeURI ایجاد شده است استفاده می شود.

```
var uri = 'https://mozilla.org/?x=шеллы';
var encoded = encodeURI(uri);
console.log(encoded); // https://mozilla.org/?x=%D1%88%D0%B5%D0%BB%D0%BB%D1%8B
try {
 console.log(decodeURI(encoded)); // "https://mozilla.org/?x=шеллы"
} catch(e) { // catches a malformed URI
 console.error(e);
}
```

## How do you print the contents of web page

شی window یک متده است (print) که برای چاپ محتویات پنجره فعلی استفاده می شود. یک کادر محاوره ای چاپ را باز می کند که به شما امکان می دهد بین گزینه های مختلف چاپ انتخاب کنید. بباید استفاده از روش چاپ را در یک مثال ببینیم،

```
<input type="button" value="Print" onclick="window.print()" />
```

**نکته:** در اکثر مرورگرها، زمانی که کادر گفتگوی چاپ باز است، مسدود می شود.

## What is the difference between uneval and eval

تابع "uneval" منبع یک شی معین را برمی گرداند. در حالی که تابع "eval" با ارزیابی آن کد منبع در یک ناحیه حافظه متفاوت، برعکس عمل می کند. باید مثالی را برای روشن شدن تفاوت ببینیم،

```
var msg = uneval(function greeting() { return 'Hello, Good morning'; });
var greeting = eval(msg);
greeting(); // returns "Hello, Good morning"
```

## What is an anonymous function

تابع ناشناس یک تابع بدون نام است! توابع ناشناس معمولاً به یک نام متغیر اختصاص داده می شوند یا به عنوان یک تابع فراخوانی استفاده می شوند. نحوه صورت زیر خواهد بود،

```
function (optionalParameters) {
 //do something
}

const myFunction = function(){ //Anonymous function assigned to a variable
 //do something
};

[1, 2, 3].map(function(element){ //Anonymous function used as a callback function
 //do something
});
```

باید تابع ناشناس بالا را در یک مثال ببینیم،

```
var x = function (a, b) {return a * b};
var z = x(5, 10);
console.log(z); // 50
```

## What is the precedence order between local and global variables

یک متغیر محلی بر یک متغیر سراسری با همین نام ارجحیت دارد. باید این رفتار را در یک مثال ببینیم.

```
var msg = "Good morning";
function greeting() {
 msg = "Good Evening";
 console.log(msg);
}
greeting();
```

## What are javascript accessors

5 ECMAScript دسترسی‌های شی جاوا اسکریپت یا ویژگی‌های محاسبه‌شده را از طریق گیرنده‌ها و تنظیم‌کننده‌ها معرفی کرد. Getters از کلمه کلیدی "get" استفاده می‌کند در حالی که Setters از کلمه کلیدی "set" استفاده می‌کند.

```
var user = {
 firstName: "John",
 lastName : "Abraham",
 language : "en",
 get lang() {
 return this.language;
 }
 set lang(lang) {
 this.language = lang;
 }
};
console.log(user.lang); // getter access lang as en
user.lang = 'fr';
console.log(user.lang); // setter used to set lang as fr
```

## How do you define property on Object constructor

متده استاتیک Object.defineProperty() برای تعریف یک ویژگی جدید به طور مستقیم بر روی یک شی یا تغییر ویژگی موجود روی یک شی استفاده می‌شود و شی را برمی‌گرداند. باید مثالی را ببینیم تا بدانیم چگونه ویژگی را تعریف کنیم،

```
const newObject = {};

Object.defineProperty(newObject, 'newProperty', {
 value: 100,
 writable: false
});

console.log(newObject.newProperty); // 100

newObject.newProperty = 200; // It throws an error in strict mode due to writable setting
```

## What is the difference between get and defineProperty

هر دو نتایج مشابهی دارند مگر اینکه از کلاس ها استفاده کنید. اگر از «get» استفاده می کنید، ویژگی روی نمونه اولیه شی تعریف می شود، در حالی که با استفاده از «Object.defineProperty()»، ویژگی روی نمونه ای که به آن اعمال می شود، تعریف می شود.

## What are the advantages of Getters and Setters

- در زیر لیستی از مزایای Getters و Setter آمده است.
۱. آنها نحو ساده تری ارائه می دهند
  ۲. آنها برای تعریف ویژگی های محاسبه شده یا دسترسی ها در JS استفاده می شوند.
  ۳. برای ارائه رابطه هم ارزی بین خواص و روش ها مفید است
  ۴. آنها می توانند کیفیت داده های بهتری را ارائه دهند
  ۵. برای انجام کارها در پشت صحنه با منطق محصور شده مفید است.

## Can I add getters and setters using defineProperty method

بله، می توانید از روش «Object.defineProperty()» برای اضافه کردن Setter و Getters استفاده کنید. به عنوان مثال، شی شمارنده زیر از ویژگی های افزایش، کاهش، جمع و تفریق استفاده می کند.

```
var obj = {counter : 0};

// Define getters
Object.defineProperty(obj, "increment", {
 get : function () {this.counter++;}
});
Object.defineProperty(obj, "decrement", {
 get : function () {this.counter--;}
});

// Define setters
Object.defineProperty(obj, "add", {
 set : function (value) {this.counter += value;}
});
Object.defineProperty(obj, "subtract", {
 set : function (value) {this.counter -= value;}
});

obj.add = 10;
obj.subtract = 5;
console.log(obj.increment); //6
console.log(obj.decrement); //5
```

## What is the purpose of switch-case

عبارت switch case در جاوا اسکریپت برای اهداف تصمیم گیری استفاده می شود. در چند مورد، استفاده از دستور switch case راحت‌تر از if-else است. نحو به صورت زیر خواهد بود،

```
switch (expression)
{
 case value1:
 statement1;
 break;
 case value2:
 statement2;
 break;
 .
 .
 case valueN:
 statementN;
 break;
 default:
 statementDefault;
}
```

دستور شعبه چند طرفه بالا یک راه آسان برای ارسال اجرا به قسمت های مختلف کد بر اساس مقدار عبارت ارائه می دهد.

## What are the conventions to be followed for the usage of switch case

در زیر لیستی از کنوانسیون ها وجود دارد که باید مراقب آنها بود،

۱. عبارت می تواند از نوع عددی یا رشته ای باشد.
۲. مقادیر تکراری برای عبارت مجاز نیستند.
۳. بیانیه پیش فرض اختیاری است. اگر عبارت ارسال شده به سوئیچ با هیچ مقدار case مطابقت نداشته باشد، دستور در حالت پیش فرض اجرا خواهد شد.
۴. دستور break در داخل سوئیچ برای پایان دادن به دنباله دستور استفاده می شود.
۵. عبارت break اختیاری است. اما اگر حذف شود، اجرا در مورد بعدی ادامه می یابد.

## What are primitive data types

یک نوع داده اولیه، داده ای است که دارای یک مقدار اولیه است (که هیچ ویژگی یا روشی ندارد). ۵ نوع نوع داده اولیه وجود دارد.

۱. رشته ها
۲. اعداد
۳. boolean

null .۴

undefined .۵

## What are the different ways to access object properties

3 راه ممکن برای دسترسی به ویژگی یک شی وجود دارد.

۱. از نقطه برای دسترسی به ویژگی ها استفاده می کند

objectName.property

۱. از براکت های مربع برای دسترسی به دیتا استفاده می کند :**Square brackets notation**

objectName["property"]

۱. از عبارت در کروشه استفاده می کند :**Expression notation**

[objectName[expression

## What are the function parameter rules

توابع جاوا اسکریپت از قوانین زیر برای پارامترها پیروی می کنند.

۱. تعاریف تابع انواع داده ها را برای پارامترها مشخص نمی کند.

۲. بررسی نوع آرگومان های ارسال شده را انجام ندهید.

۳. تعداد آرگومان های دریافتی را بررسی نکنید.

یعنی تابع زیر از قوانین بالا پیروی می کند،

```
<"span dir="ltr" align="left">
javascript````
} (function functionName(parameter1, parameter2, parameter3
console.log(parameter1); // 1
{
;(functionName(1
```  
  
<span/>  
**[فهرست] (#فهرست)**
```

What is an error object .۲۲۶

یک شیء خطای داخلی است که هنگام بروز خطای اطلاعات خطای را ارائه می‌دهد. این دو ویژگی دارد: نام و پیام. به عنوان مثال، تابع زیر جزئیات خطای را ثبت می‌کند،

```
try {
    greeting("Welcome");
}
catch(err) {
    console.log(err.name + "<br>" + err.message);
}
```

When you get a syntax error .۲۲۷

اگر بخواهید کد را با یک خطای نحوی ارزیابی کنید، یک SyntaxError پرتاب می‌شود. به عنوان مثال، نقل قول زیر برای پارامتر تابع یک خطای نحوی ایجاد می‌کند

```
} try
eval("greeting('welcome')"); // Missing ' will produce an error
{
} (catch(err
;(console.log(err.name
{
```

What are the different error names from error object .۲۲۸

۶ نوع مختلف نام خطای وجود دارد که از شی خطای بازگردانده می‌شود.

توضیحات	نام خطای
خطایی در تابع eval() رخداده است	EvalError
خطایی با عدد "خارج از محدوده"	RangeError
خطایی با دلیل ارجاع غیرقانونی	خطای مرجع
خطای ناشی از خطای نحو	SyntaxError
خطای ناشی از خطای نوع	TypeError
یک خطای دلیل encodeURI()	خطای URIE

What are the various statements in error handling .۲۲۹

در زیر لیستی از عبارات استفاده شده در مدیریت خطای آمده است.

۱. این عبارت برای آزمایش یک بلوک کد برای خطاهای استفاده می شود **try**.
۲. این عبارت برای رسیدگی به خطاهای استفاده می شود **catch**.
۳. این عبارت برای ایجاد خطاهای سفارشی استفاده می شود **throw****.
۴. این عبارت برای اجرای کد پس از تلاش و گرفتن بدون توجه به نتیجه استفاده می شود **:finally****.

What are the two types of loops in javascript .۲۳۰

۱. در این نوع حلقه، شرایط تست قبل از ورود به بدنه حلقه آزمایش می شود. به عنوان مثال **while** و **For Loop** در این دسته قرار می گیرند.
۲. در این نوع حلقه، شرایط تست در انتهای بدنه حلقه آزمایش یا ارزیابی می شود. یعنی بدنه حلقه حداقل یک بار بدون در نظر گرفتن شرایط تست **true** یا **false** اجرا می شود. به عنوان مثال، حلقه **do-while** در این دسته قرار می گیرد.

What is nodejs .۲۳۱

Node.js یک پلتفرم سمت سرور است که بر اساس زمان اجرا جاوا اسکریپت کروم برای ساخت آسان برنامه های شبکه سریع و مقیاس پذیر ساخته شده است. این یک زمان اجرا / API ناهمزنمان مبتنی بر رویداد، غیر مسدود کننده است که از موتور جاوا اسکریپت V8 گوگل و کتابخانه libuv استفاده می کند.

What is an Intl object .۲۳۲

شی **Intl** فضای نامی برای ECMAScript Internationalization API است که مقایسه رشته های حساس زبان، قالب بندی اعداد و قالب بندی تاریخ و زمان را ارائه می دهد. دسترسی به چندین سازنده و توابع حساس به زبان را فراهم می کند.

How do you perform language specific date and time formatting .۲۳۳

می توانید از شی **Intl.DateTimeFormat** استفاده کنید که سازنده اشیایی است که قالب بندی تاریخ و زمان حساس به زبان را فعال می کند. بیایید این رفتار را با یک مثال بینیم،

```
var date = new Date(Date.UTC(2019, 07, 07, 3, 0, 0));
console.log(new Intl.DateTimeFormat('en-GB').format(date)); // 07/08/2019
console.log(new Intl.DateTimeFormat('en-AU').format(date)); // 07/08/2019
```

What is an Iterator .۲۳۴

تکرار کننده شی ای است که پس از خاتمه، یک توالی و یک مقدار بازگشتی را تعریف می کند. پروتکل **Iterator** را با متدهای **next()** پیاده سازی می کند که یک شی را با دو ویژگی برمی گرداند: «**value**» (مقدار بعدی در دنباله) و «**done**» (که اگر آخرین

How does synchronous iteration works .۲۳۵

تکرار همزمان در ES6 معرفی شد و با مجموعه ای از اجزای زیر کار می کند:

- Iterable**: این یک شی است که می تواند از طریق روشی که کلید آن `Symbol.iterator` است، تکرار شود.
- Iterator**: این یک شی است که با فراخوانی «`Symbol.iterator()`» بر روی یک تکرار برگردانده می شود. این شی تکرار شونده هر عنصر تکرار شده را در یک شی پیچیده می کند و آن را از طریق متدهای `next()` یکی یکی برمی گرداند.
- IteratorResult**: این یک شی است که با متدهای `next()` برگردانده می شود. شی شامل دو ویژگی است. ویژگی "value" حاوی یک عنصر تکرار شده است و ویژگی "done" تعیین می کند که آیا عنصر آخرین عنصر است یا خیر.

بیایید تکرار همزمان را با آرایه ای مانند زیر نشان دهیم،

```
const iterable = ['one', 'two', 'three'];
const iterator = iterable[Symbol.iterator]();
console.log(iterator.next()); // { value: 'one', done: false }
console.log(iterator.next()); // { value: 'two', done: false }
console.log(iterator.next()); // { value: 'three', done: false }
console.log(iterator.next()); // { value: 'undefined', done: true }
```

What is an event loop .۲۳۶

حلقه رویداد یک صفت از توابع برگشت به تماس است. هنگامی که یک تابع `async` اجرا می شود، تابع `callback` در صف قرار می گیرد. موتور جاوا اسکریپت پردازش حلقه رویداد را شروع نمی کند تا زمانی که تابع `async` اجرای کد را به پایان برساند.

نکته: این به Node.js اجازه می دهد تا عملیات ۰/۱ غیر مسدود کننده را انجام دهد حتی اگر جاوا اسکریپت تک رشته ای باشد.

What is call stack .۲۳۷

Call Stack یک ساختار داده برای مفسران جاوا اسکریپت است تا فراخوانی های تابع در برنامه را پیگیری کند. دو عمل عمدۀ دارد،

۱. هر زمان که یک تابع را برای اجرای آن فراخوانی می کنید، آن را به پشته هل می دهید.
 ۲. هر زمان که اجرا به پایان برسد، تابع از پشته خارج می شود.
- بیایید مثالی بزنیم و نمایش حالت در قالب نمودار است

```
} ()function hungry
;()eatFruits
{
} ()function eatFruits
;"return "I'm eating fruits
{

Invoke the `hungry` function //
;()hungry
```

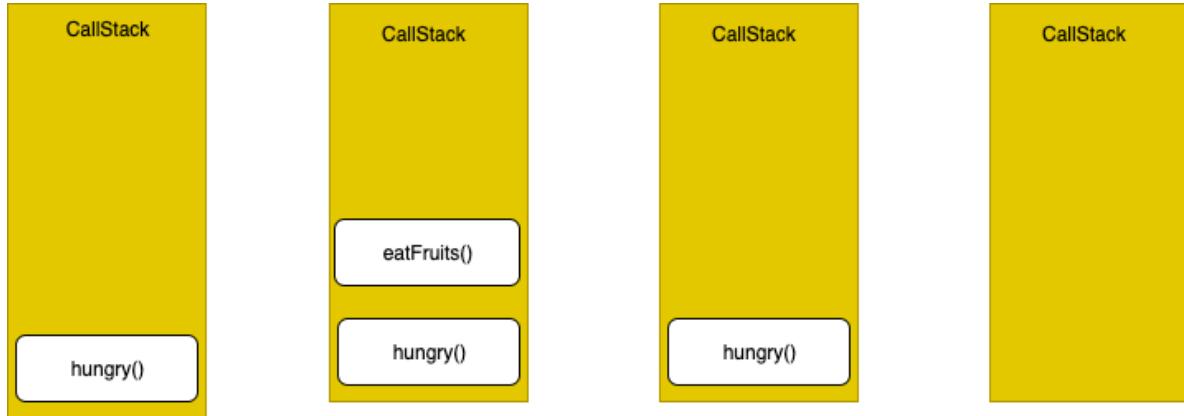
کد بالا در یک پشته تماس به صورت زیر پردازش می شود.

۱. تابع 'hungry()' را به لیست پشته تماس اضافه کنید و کد را اجرا کنید.

۲. تابع 'eatFruits()' را به لیست پشته تماس اضافه کنید و کد را اجرا کنید.

۳. تابع 'eatFruits()' را از لیست پشته تماس ما حذف کنید.

۴. تابع 'eatFruits()' را از لیست پشته تماس ما حذف کنید.



What is an event queue .۲۳۸

What is a decorator .۲۳۹

دکوراتور عبارتی است که یک تابع را ارزیابی می کند و هدف، نام و توصیف کننده تزئین را به عنوان آرگومان می گیرد. همچنین، به صورت اختیاری یک توصیفگر دکوراتور را برای نصب بر روی شی مورد نظر برمی گرداند. بیایید در زمان طراحی، دکوراتور ادمین را برای کلاس کاربر تعریف کنیم،

```
function admin(isAdmin) {
  return function(target) {
    target.isAdmin = isAdmin;
  }
}

@admin(true)
class User() {
}
console.log(User.isAdmin); //true

@admin(false)
class User() {
}
console.log(User.isAdmin); //false
```

What are the properties of Intl object .۲۴۰

- در زیر لیستی از ویژگی های موجود در شیء `Intl` آمده است.
1. **Collator**: اینها اشیایی هستند که مقایسه رشته های حساس به زبان را امکان پذیر می کنند.
 2. **DateTimeFormat**: اینها اشیایی هستند که قالب بندی تاریخ و زمان حساس به زبان را فعال می کنند.
 3. **ListFormat**: اینها اشیایی هستند که قالب بندی لیست حساس به زبان را فعال می کنند.
 4. **NumberFormat**: اشیایی که قالب بندی اعداد حساس به زبان را فعال می کنند.
 5. **PluralRules**: اشیایی که قالب بندی حساس به جمع و قوانین خاص زبان را برای جمع فعال می کنند.
 6. **RelativeTimeFormat**: اشیایی که قالب بندی زمان نسی حساس به زبان را فعال می کنند.

**[فهرست] (#فهرست)

What is an Unary operator .۲۴۱

عملگر (+) برای تبدیل یک متغیر به عدد استفاده می شود. اگر متغیر قابل تبدیل نباشد، همچنان به عدد تبدیل می شود اما با مقدار NaN. باید این رفتار را در یک عمل بینیم.

```
var x = "100";
var y = + x;
console.log(typeof x, typeof y); // string, number

var a = "Hello";
var b = + a;
console.log(typeof a, typeof b, b); // string, number, NaN
```

How do you sort elements in an array .۲۴۲

متده sort() برای مرتب سازی عناصر یک آرایه در جای خود استفاده می شود و آرایه مرتب شده را برمی گرداند. استفاده از مثال زیر خواهد بود،

```
var months = ["Aug", "Sep", "Jan", "June"];
months.sort();
console.log(months); // ["Aug", "Jan", "June", "Sep"]
```

What is the purpose of compareFunction while sorting arrays .۲۴۳

از compareFunction برای تعریف ترتیب مرتب سازی استفاده می شود. اگر حذف شود، عناصر آرایه به رشته تبدیل می شوند، سپس بر اساس مقدار نقطه کد یونیکد هر کاراکتر مرتب می شوند. باید مثالی بزنیم تا کاربرد compareFunction را بینیم،

```
let numbers = [1, 2, 5, 3, 4];
numbers.sort((a, b) => b - a);
console.log(numbers); // [5, 4, 3, 2, 1]
```

How do you reversing an array .۲۴۴

برای معکوس کردن عناصر یک آرایه می توانید از متده استفاده کنید. این روش برای مرتب کردن یک آرایه به ترتیب نزولی مفید است. بیایید استفاده از متده reverse() را در یک مثال بینیم،

```
let numbers = [1, 2, 5, 3, 4];
numbers.sort((a, b) => b - a);
numbers.reverse();
console.log(numbers); // [1, 2, 3, 4, 5]
```

How do you find min and max value in an array .۲۴۵

می توانید از روش های آرایه برای یافتن حداقل و حداقل عناصر در یک آرایه استفاده کنید. بیایید دو تابع برای پیدا کردن مقدار min و max در یک آرایه ایجاد کنیم.

```
var marks = [50, 20, 70, 60, 45, 30];
function findMin(arr) {
    return Math.min.apply(null, arr);
}
function findMax(arr) {
    return Math.max.apply(null, arr);
}

console.log(findMin(marks));
console.log(findMax(marks));
```

How do you find min and max values without Math functions .۲۴۶

دستور خالی یک نقطه ویرگول (⋮) است که نشان می دهد هیچ دستوری اجرا نخواهد شد، حتی اگر نحو جاوا اسکریپت به آن نیاز داشته باشد. از آنجایی که هیچ اقدامی با دستور خالی وجود ندارد، ممکن است فکر کنید که استفاده از آن بسیار کمتر است، اما دستور خالی گاهی اوقات زمانی مفید است که می خواهید حلقه ای ایجاد کنید که بدنه آن خالی است. به عنوان مثال، می توانید یک آرایه با مقادیر صفر را مانند زیر مقداردهی اولیه کنید.

```

var marks = [50, 20, 70, 60, 45, 30];
function findMin(arr) {
    var length = arr.length
    var min = Infinity;
    while (length--) {
        if (arr[length] < min) {
            min = arr[length];
        }
    }
    return min;
}

function findMax(arr) {
    var length = arr.length
    var max = -Infinity;
    while (length--) {
        if (arr[length] > max) {
            max = arr[length];
        }
    }
    return max;
}

console.log(findMin(marks));
console.log(findMax(marks));

```

What is an empty statement and purpose of it .۲۴۷

می‌توانید از شی «import.meta» استفاده کنید که یک ویژگی متن است که متن داده‌های متنی خاص را در یک مازول جاوا اسکریپت قرار می‌دهد. این شامل اطلاعاتی در مورد مازول فعلی، مانند URL مازول است. در مرورگرهای ممکن است متن داده‌های متفاوتی نسبت به NodeJS دریافت کنید.

```

// Initialize an array a
for(int i=0; i < a.length; a[i++] = 0) ;

```

How do you get metadata of a module .۲۴۸

عملگر کاما برای ارزیابی هر یک از عملوندهای آن از چپ به راست استفاده می‌شود و مقدار آخرین عملوند را برمی‌گرداند. این کاملاً با استفاده از کاما در آرایه‌ها، اشیاء و آرگومان‌ها و پارامترهای تابع متفاوت است. به عنوان مثال، استفاده از عبارات عددی به صورت زیر خواهد بود.

```

<script type="module" src="welcome-module.js"></script>
console.log(import.meta); // { url: "file:///home/user/welcome-module.js" }

```

What is a comma operator .۲۴۹

عملگر کاما برای ارزیابی هر یک از عملوندهای آن از چپ به راست استفاده می شود و مقدار آخرین عملوند را برمی گرداند. این کاملاً با استفاده از کاما در آرایه ها، اشیاء و آرگومان ها و پارامترهای تابع متفاوت است. به عنوان مثال، استفاده از عبارت عددی به صورت زیر خواهد بود.

```
var x = 1;  
x = (x++, x);  
  
console.log(x); // 2
```

What is the advantage of a comma operator .۲۵۰

ممدولاً برای گنجاندن چندین عبارت در مکانی که به یک عبارت واحد نیاز دارد استفاده می شود. یکی از کاربردهای رایج این عملگر کاما، ارائه چندین پارامتر در یک حلقه «for» است. به عنوان مثال، حلقه for زیر از چند عبارت در یک مکان واحد با استفاده از عملگر کاما استفاده می کند.

```
for (var a = 0, b = 10; a <= 10; a++, b--)
```

همچنین می توانید از عملگر کاما در یک عبارت بازگشتی استفاده کنید، جایی که قبل از بازگشت پردازش می کند.

```
function myFunction() {  
    var a = 1;  
    return (a += 10, a); // 11  
}
```

What is typescript .۲۵۱

یک ابر مجموعه تایپ شده از جاوا اسکریپت است که توسط مایکروسافت ایجاد شده است که انواع اختیاری، کلاس ها، و بسیاری ویژگی های دیگر را اضافه می کند و به جاوا اسکریپت ساده کامپایل می کند. Angular به طور کامل در TypeScript ساخته شده و به عنوان زبان اصلی استفاده می شود. شما می توانید آن را به صورت جهانی نصب کنید

```
npm install -g typescript
```

بیایید یک مثال ساده از استفاده از TypeScript را ببینیم،

```
function greeting(name: string): string {  
    return "Hello, " + name;  
}  
  
let user = "Sudheer";  
  
console.log(greeting(user));
```

متدهای خواهد گویی فقط نوع رشته را به عنوان آرگومان مجاز می کند.

What are the differences between javascript and typescript .۲۵۲

در زیر لیستی از تفاوت های بین جاوا اسکریپت و تایپ اسکریپت آمده است.

جاوا اسکریپت	تایپ	ویژگی
زبان اسکریپت	زبان برنامه نویسی شی گرا	پارادایم زبان
دارای تایپ پویا	پشتیبانی از تایپ استاتیک	پشتیبانی از تایپ
پشتیبانی نمی شود	پشتیبانی شده	ماژول ها
از رابط ها پشتیبانی نمی کند	دارای مفهوم رابط	رابط
عدم پشتیبانی از پارامترهای اختیاری برای توابع	توابع از پارامترهای اختیاری پشتیبانی می کنند	پارامترهای اختیاری

What are the advantages of typescript over javascript .۲۵۳

در زیر برخی از مزایای تایپ اسکریپت نسبت به جاوا اسکریپت آورده شده است.

۱. TypeScript می تواند خطاهای زمان کامپایل را فقط در زمان توسعه پیدا کند و باعث می شود خطاهای زمان اجرا کمتر شود. در حالی که جاوا اسکریپت یک زبان تفسیر شده است.

۲. TypeScript به شدت تایپ می شود یا از تایپ استاتیک پشتیبانی می کند که امکان بررسی صحت نوع را در زمان کامپایل فراهم می کند. این در جاوا اسکریپت در دسترس نیست.

۳. کامپایلر TypeScript برخلاف ویژگی های ES6 جاوا اسکریپت که ممکن است در برخی از مرورگرها پشتیبانی نشود، می تواند فایل های ts را در ES4، ES3 و ES5 کامپایل کند.

What is an object initializer .۲۵۴

یک شیء اولیه عبارتی است که مقدار دهنده اولیه یک شی را توصیف می کند. نحو این عبارت به صورت فهرستی با کاما از صفر یا چند جفت نام ویژگی و مقادیر مرتبط یک شی، محصور در پرانتزهای فرفری ({}) نشان داده می شود. این همچنین به عنوان نماد تحت اللفظی شناخته می شود. یکی از راه های ایجاد یک شی است.

```
var initObject = {a: 'John', b: 50, c: {}};

console.log(initObject.a); // John
```

What is a constructor method .۲۵۵

متد سازنده یک متد خاص برای ایجاد و مقداردهی اولیه یک شی ایجاد شده در یک کلاس است. اگر متد سازنده را مشخص نکنید، از سازنده پیش فرض استفاده می شود. مثال استفاده از سازنده به صورت زیر خواهد بود.

```

class Employee {
  constructor() {
    this.name = "John";
  }
}

var employeeObject = new Employee();

console.log(employeeObject.name); // John

```

What happens if you write constructor more than once in a class .۲۵۶

"سازنده" در یک کلاس یک متده خاص است و باید فقط یک بار در یک کلاس تعریف شود. به عنوان مثال، اگر یک متده سازنده را بیش از یک بار در یک کلاس بنویسید، یک خطای «SyntaxError» ایجاد می‌کند.

```

class Employee {
  constructor() {
    this.name = "John";
  }
  constructor() { // Uncaught SyntaxError: A class may only have one constructor
    this.age = 30;
  }
}

var employeeObject = new Employee();

console.log(employeeObject.name);

```

How do you call the constructor of a parent class .۲۵۷

می‌توانید از کلمه کلیدی `super` برای فراخوانی سازنده کلاس والد استفاده کنید. به یاد داشته باشید که «`super()`» باید قبل از استفاده از مرجع «`this`» فراخوانی شود. در غیر این صورت باعث خطای مرجع می‌شود. بیایید از آن استفاده کنیم،

```

class Square extends Rectangle {
  constructor(length) {
    super(length, length);
    this.name = 'Square';
  }

  get area() {
    return this.width * this.height;
  }

  set area(value) {
    this.area = value;
  }
}

```

How do you get the prototype of an object .۲۵۸

می توانید از روش `Object.getPrototypeOf(obj)` برای برگرداندن نمونه اولیه شی مشخص شده استفاده کنید. یعنی مقدار ویژگی «نمونه اولیه» داخلی. اگر هیچ ویژگی ارثی وجود نداشته باشد، مقدار "null" برگردانده می شود.

```
;{} = const newPrototype  
;(const newObject = Object.create(newPrototype)  
  
console.log(Object.getPrototypeOf(newObject) === newPrototype); // true
```

What happens If I pass string type for getPrototypeOf method .۲۵۹

در ES5، اگر پارامتر `obj` یک شی نباشد، یک استثنای `TypeError` ایجاد می کند. در حالی که در ES2015، پارامتر به یک «شیء» اجباری می شود.

```
// ES5  
Object.getPrototypeOf('James'); // TypeError: "James" is not an object  
// ES2015  
Object.getPrototypeOf('James'); // String.prototype
```

How do you set prototype of one object to another .۲۶۰

می توانید از متده `Object.setPrototypeOf()` استفاده کنید که نمونه اولیه (یعنی ویژگی داخلی «Prototype») یک شی مشخص شده را روی یک شی دیگر یا تهی تنظیم می کند. به عنوان مثال، اگر می خواهید نمونه اولیه یک جسم مربع را روی شی مستطیلی تنظیم کنید، به صورت زیر است:

```
Object.setPrototypeOf(Square.prototype, Rectangle.prototype);  
Object.setPrototypeOf({}, null);
```

How do you check whether an object can be extendable or not .۲۶۱

متده `Object.isExtensible()` برای تعیین اینکه آیا یک شی قابل توسعه است یا نه استفاده می شود. یعنی اینکه آیا می تواند ویژگی های جدیدی به آن اضافه شود یا خیر.

```
const newObject = {};  
console.log(Object.isExtensible(newObject)); //true
```

نکته: به طور پیش فرض، تمام اشیاء قابل گسترش هستند. به عنوان مثال، ویژگی های جدید را می توان اضافه یا تغییر داد.

How do you prevent an object to extend .۲۶۲

متده «Object.preventExtensions()» برای جلوگیری از افزودن ویژگی های جدید به یک شی استفاده می شود. به عبارت دیگر، از پسوندهای بعدی به شی جلوگیری می کند. بباید استفاده از این ویژگی را ببینیم،

```
const newObject = {};
Object.preventExtensions(newObject); // NOT extendable

try {
  Object.defineProperty(newObject, 'newProperty', { // Adding new property
    value: 100
  });
} catch (e) {
  console.log(e); // TypeError: Cannot define property newProperty, object is not extensible
}
```

What are the different ways to make an object non-extensible .۲۶۳

شما می توانید یک شی غیرقابل گسترش را به ۳ روش علامت گذاری کنید.

Object.preventExtensions .۱

Object.seal .۲

Object.freeze .۳

```
var newObject = {};

Object.preventExtensions(newObject); // Prevent objects are non-extensible
Object.isExtensible(newObject); // false

var sealedObject = Object.seal({}); // Sealed objects are non-extensible
Object.isExtensible(sealedObject); // false

var frozenObject = Object.freeze({}); // Frozen objects are non-extensible
Object.isExtensible(frozenObject); // false
```

How do you define multiple properties on an object .۲۶۴

متده «Object.defineProperties()» برای تعریف یا اصلاح ویژگی های موجود مستقیماً روی یک شی و برگرداندن شی استفاده می شود. بباید چندین ویژگی را روی یک شی خالی تعریف کنیم،

```
const newObject = {};

Object.defineProperties(newObject, {
  newProperty1: {
    value: 'John',
    writable: true
  },
  newProperty2: {}
});
```

What is MEAN in javascript .۲۶۵

پشته MEAN (MongoDB، Express، AngularJS و Node.js) محبوب‌ترین پشته فناوری نرم‌افزار جاوا اسکریپت منبع باز است که برای ساخت برنامه‌های وب پویا در دسترس است، جایی که می‌توانید نیمه‌های سمت سرور و سمت مشتری پروژه وب را بنویسید. کاملاً در جاوا اسکریپت

What Is Obfuscation in javascript .۲۶۶

مبهم سازی عمل عمدی ایجاد کد جاوا اسکریپت مبهم (یعنی کد منبع یا ماشین) است که در آن برای انسان دشوار است. این چیزی شبیه به رمزگذاری است، اما یک ماشین می‌تواند کد را درک کرده و آن را اجرا کند. بباید تابع زیر را قبل از Obfuscation ببینیم،

```
function greeting() {  
    console.log('Hello, welcome to JS world');  
}
```

و بعد از کد Obfuscation به صورت زیر ظاهر می‌شود

```
eval(function(p,a,c,k,e,d){e=function(c){return c};if(!''.replace(/\^/,String)){while(c--)  
{d[c]=k[c]||c}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1;while(c--)  
{if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c])}}return p}('2 1(){0.3(\\"4, 7  
,6 5 8\\")}',9,9,'console|greeting|function|log|Hello|JS|to|welcome|world'.split('|'),0  
(({}))
```

Why do you need Obfuscation .۲۶۷

- ۱. در زیر چند دلیل برای مبهم سازی آمده است،
- ۲. اندازه کد کاهش می‌یابد. بنابراین انتقال داده بین سرور و مشتری سریع خواهد بود.
- ۳. این منطق کسب و کار را از دنیای خارج پنهان می‌کند و از کد در برابر دیگران محافظت می‌کند
- ۴. مهندسی معکوس بسیار دشوار است
- ۵. زمان دانلود کاهش می‌یابد

What is Minification .۲۶۸

کوچک سازی فرآیند حذف تمام کاراکترهای غیر ضروری است (فضاهای خالی حذف می‌شوند) و متغیرها بدون تغییر در عملکرد آن تغییر نام می‌دهند. همچنین نوعی مبهم است.

What are the advantages of minification .۲۶۹

به طور معمول توصیه می شود برای ترافیک سنگین و نیازهای فشرده منابع از Minification استفاده کنید. اندازه فایل را با مزایای زیر کاهش می دهد

۱. زمان بارگذاری یک صفحه وب را کاهش می دهد

۲. در مصرف پهنای باند صرفه جویی می کند

What are the differences between Obfuscation and Encryption .۲۷۰

در زیر تفاوت های اصلی بین مبهم سازی و رمزگذاری وجود دارد.

رمزگذاری	مبهم سازی	ویژگی
تغییر فرم اطلاعات به فرمت ناخوانا با استفاده از کلید	تغییر فرم هر داده به هر شکل دیگر	تعریف
لازم است	می توان آن را بدون هیچ کلید رمزگشایی کرد	کلیدی برای رمزگشایی
تبديل به فرمت ناخوانا	به فرم پیچیده تبدیل می شود	فرمت داده های هدف

What are the common tools used for minification .۲۷۱

ابزارهای آنلاین/آفلاین زیادی برای کوچک کردن فایل های جاوا اسکریپت وجود دارد.

۱. کامپایلر بسته شدن گوگل

UglifyJS2.۲

jsmin.۳

/javascript-minifier.com.۴

prettydiff.com.۵

How do you perform form validation using javascript .۲۷۲

از جاوا اسکریپت می توان برای اعتبار سنجی فرم HTML استفاده کرد. به عنوان مثال، اگر فیلد فرم خالی باشد، تابع باید اطلاع دهد و false را برگرداند تا از ارسال فرم جلوگیری شود.
اجازه دهید ورود کاربر را در فرم html انجام دهیم،

```
<form name="myForm" onsubmit="return validateForm()" method="post">
User name: <input type="text" name="uname">
<input type="submit" value="Submit">
</form>
```

و اعتبار سنجی ورود کاربر در زیر است،

```

function validateForm() {
  var x = document.forms["myForm"]["uname"].value;
  if (x == "") {
    alert("The username shouldn't be empty");
    return false;
  }
}

```

How do you perform form validation without javascript .۲۷۳

می توانید بدون استفاده از جاوا اسکریپت اعتبار سنجی فرم HTML را به صورت خودکار انجام دهید. اعتبار سنجی با اعمال ویژگی «لازم» برای جلوگیری از ارسال فرم زمانی که ورودی خالی است فعال می شود.

```

<form method="post">
  <input type="text" name="uname" required>
  <input type="submit" value="Submit">
</form>

```

نکته: اعتبار سنجی فرم خودکار در اینترنت اکسپلورر ۹ یا قبل از آن کار نمی کند.

What are the DOM methods available for constraint validation .۲۷۴

روش های DOM زیر برای اعتبار سنجی محدودیت در ورودی نامعتبر موجود است.
 ۱. (checkValidity): اگر یک عنصر ورودی حاوی داده های معتبر باشد، مقدار true را برمی گرداند.
 ۲. (setCustomValidity): برای تنظیم خاصیت validationMessage یک عنصر ورودی استفاده می شود.
 باید یک فرم ورود کاربر با اعتبار سنجی DOM بگیریم

```

function myFunction() {
  var userName = document.getElementById("uname");
  if (!userName.checkValidity()) {
    document.getElementById("message").innerHTML = userName.validationMessage;
  } else {
    document.getElementById("message").innerHTML = "Entered a valid username";
  }
}

```

What are the available constraint validation DOM properties .۲۷۵

در زیر لیستی از برخی از ویژگی های DOM اعتبار سنجی محدودیت موجود است،
 ۱. validity: فهرستی از ویژگی های بولی مربوط به اعتبار یک عنصر ورودی را ارائه می دهد.
 ۲. validationMessage: زمانی که اعتبار نادرست باشد، پیام را نمایش می دهد.
 ۳. willValidate: این نشان می دهد که آیا یک عنصر ورودی اعتبار سنجی می شود یا خیر.

What are the list of validity properties .۲۷۶

- ویژگی اعتبار یک عنصر ورودی مجموعه ای از ویژگی های مربوط به اعتبار داده ها را ارائه می دهد.
- ۱. customError: اگر یک پیام اعتبار سفارشی تنظیم شده باشد، true را برمی گرداند.
 - ۲. patternMismatch: اگر مقدار یک عنصر با ویژگی آن مطابقت نداشته باشد، مقدار true را برمی گرداند.
 - ۳. rangeOverflow: اگر مقدار یک عنصر از ویژگی آن بیشتر باشد، مقدار true را برمی گرداند.
 - ۴. rangeUnderflow: اگر مقدار یک عنصر کمتر از ویژگی min باشد، مقدار true را برمی گرداند.
 - ۵. stepMismatch: اگر مقدار عنصر مطابق با ویژگی step نامعتبر باشد، مقدار true را برمی گرداند.
 - ۶. tooLong: اگر مقدار یک عنصر از ویژگی maxLength آن بیشتر شود، مقدار true را برمی گرداند.
 - ۷. typeMismatch: اگر مقدار یک عنصر بر اساس ویژگی نوع نامعتبر باشد، مقدار true را برمی گرداند.
 - ۸. valueMissing: اگر عنصری با ویژگی مورد نیاز ارزش نداشته باشد، مقدار true را برمی گرداند.
 - ۹. valid: اگر مقدار یک عنصر معتبر باشد، مقدار true را برمی گرداند.

Give an example usage of rangeOverflow property .۲۷۷

اگر مقدار یک عنصر از ویژگی max آن بیشتر باشد، ویژگی rangeOverflow مقدار true را برمی گرداند. به عنوان مثال، فرم ارسالی زیر اگر مقدار آن بیش از 100 باشد، خطای خطای می دهد.

```
<input id="age" type="number" max="100">
<button onclick="myOverflowFunction()">OK</button>
```

```
function myOverflowFunction() {
  if (document.getElementById("age").validity.rangeOverflow) {
    alert("The mentioned age is not allowed");
  }
}
```

Is enums feature available in javascript .۲۷۸

نه، جاوا اسکریپت به صورت بومی از enum ها پشتیبانی نمی کند. اما انواع مختلفی از راه حل ها برای شبیه سازی آنها وجود دارد، اگرچه ممکن است معادل های دقیقی ارائه نکنند. به عنوان مثال، می توانید از فریز یا مهر و موم روی شی استفاده کنید

```
var DaysEnum = Object.freeze({ "monday":1, "tuesday":2, "wednesday":3, ... })
```

What is an enum .۲۷۹

نوعی است که متغیرها را به یک مقدار از مجموعه ای از ثابت های از پیش تعریف شده محدود می کند. جاوا اسکریپت هیچ enum ندارد اما تایپ اسکریپت از enum داخلی پشتیبانی می کند.

```
} enum Color  
RED, GREEN, BLUE  
{
```

How do you list all properties of an object .۲۸۰

می‌توانید از متد ()Object.getOwnPropertyNames استفاده کنید که آرایه‌ای از تمام ویژگی‌هایی را که مستقیماً در یک شیء داده شده یافت می‌شود، برمی‌گرداند. بیایید استفاده از آن را در یک مثال بیان کنیم،

```
const newObject = {  
  a: 1,  
  b: 2,  
  c: 3  
};  
  
console.log(Object.getOwnPropertyNames(newObject));  ["a", "b", "c"]
```

How do you get property descriptors of an object .۲۸۱

می‌توانید از متد ()Object.getOwnPropertyDescriptors استفاده کنید که تمام توصیف‌گرهای ویژگی یک شی معین را برمی‌گرداند. مثال استفاده از این روش در زیر آمده است

```
const newObject = {  
  a: 1,  
  b: 2,  
  c: 3  
};  
const descriptorsObject = Object.getOwnPropertyDescriptors(newObject);  
console.log(descriptorsObject.a.writable); //true  
console.log(descriptorsObject.a.configurable); //true  
console.log(descriptorsObject.a.enumerable); //true  
console.log(descriptorsObject.a.value); // 1
```

What are the attributes provided by a property descriptor .۲۸۲

توصیفگر ویژگی رکورדי است که دارای ویژگی‌های زیر است
۱. value: ارزش مرتبط با ملک

۲. writable: تعیین می‌کند که آیا مقدار مرتبط با ویژگی قابل تغییر است یا خیر

۳. configurable: اگر بتوان نوع توصیفگر این ویژگی را تغییر داد و اگر ویژگی را بتوان از شی مربوطه حذف کرد، مقدار true را برمی‌گرداند.

۴. enumerable: تعیین می‌کند که آیا ویژگی در هنگام شمارش خصوصیات روی شی مربوطه ظاهر می‌شود یا خیر.

۵. set: تابعی که به عنوان تنظیم کننده برای ویژگی عمل می‌کند

۶. get: تابعی که به عنوان یک گیرنده برای ملک عمل می‌کند

How do you extend classes .۲۸۳

کلمه کلیدی "extends" در اعلان ها/عبارات کلاس برای ایجاد کلاسی که فرزند کلاس دیگری است استفاده می شود. می توان از آن برای زیر کلاس بنده کلاس های سفارشی و همچنین اشیاء داخلی استفاده کرد. نحو به صورت زیر خواهد بود،

```
class ChildClass extends ParentClass { ... }
```

بیایید یک نمونه از زیر کلاس مربع از کلاس والد Polygon را مثال بزنیم،

```
class Square extends Rectangle {
    constructor(length) {
        super(length, length);
        this.name = 'Square';
    }

    get area() {
        return this.width * this.height;
    }

    set area(value) {
        this.area = value;
    }
}
```

How do I modify the url without reloading the page .۲۸۴

ویژگی «window.location.url» برای تغییر url مفید خواهد بود اما صفحه را دوباره بارگیری می کند. HTML5 متدهای «history.pushState» و «history.replaceState» را معرفی کرد که به شما اجازه می دهد به ترتیب ورودی های تاریخ را اضافه و تغییر دهید. به عنوان مثال، می توانید از pushState مانند زیر استفاده کنید.

```
window.history.pushState('page2', 'Title', '/page2.html');
```

How do you check whether an array includes a particular value or not .۲۸۵

متدهای «Array.includes()» برای تعیین اینکه آیا یک آرایه دارای مقدار خاصی در میان ورودی های خود با برگرداندن true یا false است یا خیر استفاده می شود. بیایید مثالی برای یافتن یک عنصر (عددی و رشته ای) در یک آرایه بینیم.

```
var numericArray = [1, 2, 3, 4];
console.log(numericArray.includes(3)); // true

var stringArray = ['green', 'yellow', 'blue'];
console.log(stringArray.includes('blue'));
```

How do you compare scalar arrays .۲۸۶

می توانید از طول و هر روش آرایه برای مقایسه دو آرایه اسکالر (مقایسه مستقیم با استفاده از `==`) استفاده کنید. ترکیب این عبارات می تواند نتیجه مورد انتظار را به دست دهد،

```
const arrayFirst = [1,2,3,4,5];
const arraySecond = [1,2,3,4,5];
console.log(arrayFirst.length === arraySecond.length && arrayFirst.every((value, index)
=> value === arraySecond[index])); // true
```

اگر می خواهید آرایه ها را بدون توجه به ترتیب مقایسه کنید، باید آنها را قبل از مرتب سازی، مرتب کنید.

```
const arrayFirst = [2,3,1,4,5];
const arraySecond = [1,2,3,4,5];
console.log(arrayFirst.length === arraySecond.length && arrayFirst.sort().every((value,
index) => value === arraySecond[index])); //true
```

How to get the value from get parameters .۲۸۷

شیء «URL» رشته url را می پزیرد و از ویژگی «searchParams» این شی می توان برای دسترسی به پارامترهای get استفاده کرد. به یاد داشته باشید که ممکن است برای دسترسی به URL در مرورگرهای قدیمی (از جمله IE) نیاز به استفاده از `polyfill` یا `window.location` داشته باشید.

```
let urlString = "http://www.some-domain.com/about.html?x=1&y=2&z=3";
//window.location.href
let url = new URL(urlString);
let parameterZ = url.searchParams.get("z");
console.log(parameterZ); // 3
```

How do you print numbers with commas as thousand separators .۲۸۸

می توانید از متد «Number.prototype.toLocaleString()» استفاده کنید که رشته ای را با نمایشی حساس به زبان مانند جداکننده هزار، ارز و غیره از این عدد برمی گرداند.

```
function convertToThousandFormat(x){
  return x.toLocaleString(); // 12,345.679
}

console.log(convertToThousandFormat(12345.6789));
```

What is the difference between java and javascript .۲۸۹

هر دو زبان برنامه نویسی کاملاً نامرتب هستند و هیچ ارتباطی بین آنها وجود ندارد. جاوا بصورت ایستتا تایپ می شود، کامپایل می شود، روی ماشین مجازی خود اجرا می شود. در حالی که جاوا اسکریپت به صورت پویا تایپ می شود، تفسیر می شود و در محیط های مرورگر و nodejs اجرا می شود. باید تفاوت های عمدی را در قالب جدولی بینیم،

جاوا اسکریپت	جاوا	ویژگی
این یک زبان تایپ شده پویا است	این یک زبان قوی تایپ شده است	تایپ شده
برنامه نویسی مبتنی بر نمونه اولیه	برنامه نویسی شی گرا	پارادایم
محدوده عملکردی	محدوده بلوک	محدوده
مبتنی بر رویداد	بر اساس موضوع	همزمانی
از حافظه کمتری استفاده می کند. از این روش برای صفحات وب استفاده خواهد شد	از حافظه بیشتر استفاده می کند	حافظه

۲۹۰ Is javascript supports namespace

جاوا اسکریپت به طور پیش فرض از فضای نام پشتیبانی نمی کند. بنابراین اگر هر عنصری (تابع، روش، شی، متغیر) ایجاد کنید، جهانی می شود و فضای نام جهانی را آلووده می کند. بیایید مثالی از تعریف دو تابع بدون فضای نام بیاوریم،

```
function func1() {
    console.log("This is a first definition");

}

function func1() {
    console.log("This is a second definition");
}
func1(); // This is a second definition
```

همیشه تعریف تابع دوم را فراخوانی می کند. در این صورت فضای نام مشکل برخورد نام را حل می کند.

۲۹۱ How do you declare namespace

حتی اگر جاوا اسکریپت قادر فضاهای نام باشد، می توانیم از Objects، IIFE برای ایجاد فضاهای نام استفاده کنیم.
Using Object Literal Notation . بیایید متغیرها و توابع را درون یک Object literal پیچیم که به عنوان فضای نام عمل می کند. پس از آن می توانید با استفاده از نماد شیء به آنها دسترسی داشته باشید

```
var namespaceOne = {
    function func1() {
        console.log("This is a first definition");
    }
}

var namespaceTwo = {
    function func1() {
        console.log("This is a second definition");
    }
}

namespaceOne.func1(); // This is a first definition
namespaceTwo.func1(); // This is a second definition
```

برای تمام کدهای داخل آن ایجاد می کند و تابع ناشناس را به یک عبارت تابع تبدیل می کند. به همین دلیل، می توانید یک تابع را در دو عبارت تابع مختلف ایجاد کنید تا به عنوان فضای نام عمل کند.

```
(function() {
    function fun1(){
        console.log("This is a first definition");
    } fun1();
}());

(function() {
    function fun1(){
        console.log("This is a second definition");
    } fun1();
}());
```

شما می توانید به سادگی از یک بلوک و **Using a block and a let/const declaration** در 6 ECMAScript برای محدود کردن دامنه یک متغیر به یک بلوک استفاده کنید.

```
{
let myFunction= function fun1(){
    console.log("This is a first definition");
}
myFunction();
}

{
let myFunction= function fun1(){
    console.log("This is a second definition");
}
myFunction();
}

//myFunction(): ReferenceError: myFunction is not defined.
```

How do you invoke javascript code in an iframe from parent page .۲۹۲

در ابتدا باید iFrame با استفاده از «window.frames» یا «document.getElementById('targetFrame')» قابل دسترسی باشد. پس از آن targetFunction به contentWindow از iFrame دسترسی می دهد ویژگی «contentWindow» iFrame

```
document.getElementById('targetFrame').contentWindow.targetFunction();
window.frames[0].frameElement.contentWindow.targetFunction(); // Accessing iframe this way may not work in latest versions chrome and firefox
```

How do get the timezone offset from date .۲۹۳

می توانید از روش «getTimezoneOffset» شی تاریخ استفاده کنید. این روش اختلاف منطقه زمانی را بر حسب دقیقه از محلی فعلی (تنظیمات سیستم میزبان) به UTC برمی گرداند

```
var offset = new Date().getTimezoneOffset();
console.log(offset); // -480
```

How do you load CSS and JS files dynamically .۲۹۴

شما می توانید هر دو عنصر پیوند و اسکریپت را در DOM ایجاد کنید و آنها را به عنوان فرزند به تگ head اضافه کنید. باید یک تابع برای اضافه کردن منابع اسکریپت و سبک مانند زیر ایجاد کنیم.

```
function loadAssets(filename, filetype) {
  if (filetype == "css") { // External CSS file
    var fileReference = document.createElement("link");
    fileReference.setAttribute("rel", "stylesheet");
    fileReference.setAttribute("type", "text/css");
    fileReference.setAttribute("href", filename);
  } else if (filetype == "js") { // External JavaScript file
    var fileReference = document.createElement('script');
    fileReference.setAttribute("type", "text/javascript");
    fileReference.setAttribute("src", filename);
  }
  if (typeof fileReference != "undefined")
    document.getElementsByTagName("head")[0].appendChild(fileReference)
}
```

What are the different methods to find HTML elements in DOM .۲۹۵

اگر می خواهید به هر عنصری در صفحه HTML دسترسی داشته باشید، باید با دسترسی به شی سند شروع کنید. بعداً می توانید از یکی از روش های زیر برای یافتن عنصر HTML استفاده کنید.
۱. (document.getElementById(id)): یک عنصر را با Id پیدا می کند
۲. (document.getElementsByTagName(name)): یک عنصر را با نام تگ پیدا می کند
۳. (document.getElementsByClassName(name)): یک عنصر را با نام کلاس پیدا می کند

What is jQuery .۲۹۶

jQuery یک کتابخانه جاوا اسکریپت متقابل مروگر محبوب است که با به حداقل رساندن اختلاف بین مروگرها، پیمایش مدل شی سند (DOM)، مدیریت رویداد، اینیمیشن ها و تعاملات AJAX را فراهم می کند. با فلسفه اش «کمتر بنویس، بیشتر انجام بده» شهرت زیادی دارد. به عنوان مثال، می توانید پیام خوش آمدگویی را در بارگذاری صفحه با استفاده از jQuery به صورت زیر نمایش دهید.

```
$(document).ready(function(){ // It selects the document and apply the function on page
load
  alert('Welcome to jQuery world');
});
```

نکته: می توانید آن را از سایت رسمی jquery دانلود کنید یا از CDN ها مانند گوگل نصب کنید.

What is V8 JavaScript engine .۲۹۷

یک موتور جاوا اسکریپت با کارایی بالا منبع باز است که توسط مرورگر Google Chrome استفاده می شود و به زبان C++ نوشته شده است. همچنین در پروژه node.js استفاده می شود. WebAssembly و ECMAScript را پیاده سازی می کند و روی ویندوز 7 یا بالاتر، macOS 10.12، ARM 32، ARM 64، MIPS یا x64 استفاده می کند، اجرا می شود.

نکته: می تواند به صورت مستقل اجرا شود یا می تواند در هر برنامه C++ تعبیه شود.

Why do we call javascript as dynamic language .۲۹۸

جاوا اسکریپت یک زبان ساده تایپ شده یا یک زبان پویا است زیرا متغیرها در جاوا اسکریپت مستقیماً با هیچ نوع مقدار خاصی مرتبط نیستند و هر متغیری را می توان با مقادیری از همه نوع تخصیص/تخصیص مجدد داد.

```
let age = 50;      // age is a number now
age = 'old'; // age is a string now
age = true; // age is a boolean
```

What is a void operator .۲۹۹

عملگر «void» عبارت داده شده را ارزیابی می کند و سپس تعریف نشده (یعنی بدون بازگشت مقدار) را برمی گرداند. نحوه صورت زیر خواهد بود،

```
void (expression)
void expression
```

باید پیامی را بدون هیچ گونه تغییر مسیر یا بارگیری مجدد نمایش دهیم

```
<a href="javascript:void(alert('Welcome to JS world'))">Click here to see a message</a>
```

نکته: این عملگر اغلب برای بدست آوردن مقدار اولیه تعریف نشده با استفاده از "void(0)" استفاده می شود.

How to set the cursor to wait .۳۰۰

مکان نما می توان برای انتظار در جاوا اسکریپت با استفاده از ویژگی "cursor" تنظیم کرد. باید این رفتار را در بارگذاری صفحه با استفاده از تابع زیر انجام دهیم.

```
function myFunction() {
window.document.body.style.cursor = "wait";
}
```

و این تابع در بارگذاری صفحه فراخوانی می شود

```
<body onload="myFunction()">
```

How do you create an infinite loop .۳۰۱

شما می توانید حلقه های بی نهایت با استفاده از حلقه های `for` و `while` بدون استفاده از هیچ عبارتی ایجاد کنید. ساختار یا نحو حلقه `for` از نظر ESLint و ابزارهای بهینه ساز کد، رویکرد بهتری است.

```
for (;;) {}  
while(true) {  
}
```

Why do you need to avoid with statement .۳۰۲

دستور با جاوا اسکریپت در نظر گرفته شده بود که مختصری برای نوشتن دسترسی های تکرارشونده به اشیا ارائه دهد. بنابراین می تواند با کاهش نیاز به تکرار یک مرجع طولانی بدون جریمه عملکرد، به کاهش اندازه فایل کمک کند. بیایید مثالی بزنیم که در آن برای جلوگیری از افزونگی هنگام چندین بار دسترسی به یک شی استفاده می شود.

```
a.b.c.greeting = 'welcome';  
a.b.c.age = 32;
```

:Using `with` it turns this into

```
with(a.b.c) {  
    greeting = "welcome";  
    age = 32;  
}
```

اما این عبارت «`with`» مشکلات عملکردی ایجاد می کند، زیرا نمی توان پیش بینی کرد که آیا یک آرگومان به یک متغیر واقعی اشاره می کند یا به یک ویژگی درون آرگومان `with`.

What is the output of below for loops .۳۰۳

```
for (var i = 0; i < 4; i++) { // global scope  
    setTimeout(() => console.log(i));  
}  
  
for (let i = 0; i < 4; i++) { // block scope  
    setTimeout(() => console.log(i));  
}
```

خروجی حلقه های بالا ۴ ۴ ۴ ۴ و ۰ ۱ ۲ ۳ است

توضیح: با توجه به صفت رویداد / حلقه جاوا اسکریپت، تابع 'setTimeout' پس از اجرای حلقه فراخوانی می شود. از آنجایی که متغیر `i` با کلمه کلیدی "var" اعلام می شود، به یک متغیر جهانی تبدیل می شود و با استفاده از تکرار زمانی که تابع `time` `setTimeout` فراخوانی می شود، مقدار آن برابر با ۴ است. بنابراین، خروجی حلقه اول '4 4 4 4' است. در حالی که در حلقه دوم، متغیر `i` به عنوان کلمه کلیدی «`let`» اعلام می شود، به متغیری با محدوده بلوك تبدیل می شود و یک مقدار جدید ۰, ۱, ۲, ۳ (برای هر تکرار دارد. بنابراین، خروجی حلقه اول «3 2 1 0» است).

List down some of the features of ES6. ۳۰۴

در زیر لیستی از برخی از ویژگی‌های جدید ES6 آمده است.

۱. پشتیبانی از ثابت‌ها یا متغیرهای تغییرناپذیر

۲. پشتیبانی Block-scope برای متغیرها، ثابت‌ها و توابع

۳. Arrow functions

۴. پارامترهای پیش فرض

۵. پارامترهای Rest and Spread

۶. Template Literals

۷. Multi-line Strings

۸. Destructuring Assignment

۹. Enhanced Object Literals

۱۰. Promises

۱۱. Classes

۱۲. Modules

What is ES6. ۳۰۵

ES6 ششمین نسخه از زبان جاوا اسکریپت است و در ژوئن 2015 منتشر شد. در ابتدا با نام (ES6) (ECMAScript 6) شناخته شد و بعداً به ECMAScript تغییر نام داد. تقریباً همه مروگرهای مدرن از ES6 پشتیبانی می‌کنند اما برای مروگرهای قدیمی ترانسپایلرهای زیادی وجود دارد، مانند `Babel.js` و غیره

Can I redeclare let and const variables. ۳۰۶

نه، شما نمی‌توانید متغیرهای `let` و `const` را مجددًا اعلام کنید. اگر این کار را انجام دهید، خطای زیر را نشان می‌دهد

`Uncaught SyntaxError: Identifier 'someVariable' has already been declared`

توضیح: اعلان متغیر با کلمه کلیدی "var" به یک محدوده تابع اشاره دارد و با متغیر به دلیل ویژگی بالا بردن به گونه‌ای رفتار می‌شود که گویی در بالای محدوده محصور اعلام شده است. بنابراین همه اعلان‌های چندگانه بدون هیچ خطایی در ایجاد یک متغیر `hoisted` مشترک نقش دارند. بیایید مثالی از اعلان مجدد متغیرها در یک محدوده برای متغیرهای `var` و `let/const` بزنیم.

```
var name = 'John';
function myFunc() {
    var name = 'Nick';
    var name = 'Abraham'; // Re-assigned in the same function block
    alert(name); // Abraham
}
myFunc();
alert(name); // John
```

اعلان چندگانه با محدوده بلوك، خطای نحوی ایجاد می‌کند،

```

let name = 'John';
function myFunc() {
    let name = 'Nick';
    let name = 'Abraham'; // Uncaught SyntaxError: Identifier 'name' has already been
declared
    alert(name);
}

myFunc();
alert(name);

```

Is const variable makes the value immutable .۳۰۷

خیر، متغیر const مقدار را تغییرناپذیر نمی کند. اما تخصیص های بعدی را مجاز نمی داند (یعنی می توانید با تخصیص اعلام کنید اما بعداً نمی توانید مقدار دیگری را اختصاص دهید)

```

const userList = [];
userList.push('John'); // Can mutate even though it can't re-assign
console.log(userList); // ['John']

```

What are default parameters .۳۰۸

در ES5، برای مدیریت مقادیر پیشفرض پارامترهای تابع، باید به عملگرهای OR منطقی وابسته باشیم. در حالی که در ES6، ویژگی پارامترهای تابع پیشفرض اجازه می دهد تا پارامترها با مقادیر پیشفرض مقداردهی اولیه شوند، اگر مقداری یا تعریف نشده ارسال نشود. بیایید رفتار را با یک مثال مقایسه کنیم،

```

//ES5
var calculateArea = function(height, width) {
    height = height || 50;
    width = width || 60;

    return width * height;
}
console.log(calculateArea()); //300

```

پارامترهای پیشفرض، مقداردهی اولیه را ساده تر می کند،

```

//ES6
var calculateArea = function(height = 50, width = 60) {
    return width * height;
}

console.log(calculateArea()); //300

```

What are template literals .۳۰۹

حروف الفبای الگو یا رشته‌های الگو، حروف الفبای رشته‌ای هستند که امکان عبارات تعبیه‌شده را می‌دهند. اینها به جای گیومه‌های دوتایی یا تکی با کاراکتر بک تیک (`) محصور می‌شوند.
در ES6، این ویژگی استفاده از عبارات پویا را به شرح زیر امکان پذیر می‌کند.

```
var greeting = `Welcome to JS World, Mr. ${firstName} ${lastName}.`
```

In ES5, you need break string like below

```
var greeting = 'Welcome to JS World, Mr. ' + firstName + ' ' + lastName.'
```

نکته: می‌توانید از رشته‌های چند خطی و ویژگی‌های درون‌یابی رشته‌ای با الفبای الگو استفاده کنید.

How do you write multi-line strings in template literals. ۳۱۰

در ES5، برای بدست آوردن رشته‌های چند خطی، باید از کاراکترهای فرار از خط جدید (`\n`) و نمادهای الحاقی (+) استفاده کنید.

```
console.log('This is string sentence 1\n' +
'This is string sentence 2');
```

در حالی که در ES6، نیازی به ذکر کاراکتر دنباله خط جدید نیست،

```
console.log(`This is string sentence
'This is string sentence 2');
```

What are nesting templates. ۳۱۱

الگوی تودرتو یک ویژگی است که در نحو تحت لفظی الگو پشتیبانی می‌شود تا امکان بکتیک‌های درونی در یک مکان‌نمای \${} را در قالب فراهم کند. برای مثال، الگوی تودرتو زیر برای نمایش نمادها بر اساس مجوزهای کاربر استفاده می‌شود، در حالی که الگوی بیرونی نوع پلت فرم را بررسی می‌کند.

```
const iconStyles = `icon ${ isMobilePlatform() ? '' :
`icon-${user.isAuthorized ? 'submit' : 'disabled'}` }`;
```

می‌توانید مورد استفاده بالا را بدون ویژگی‌های الگوی تودرتو نیز بنویسید. با این حال، ویژگی الگوی تودرتو فشرده‌تر و خواناتر است.

```
//Without nesting templates
const iconStyles = `icon ${ isMobilePlatform() ? '' :
(user.isAuthorized ? 'icon-submit' : 'icon-disabled') }`;
```

What are tagged templates. ۳۱۲

الگوهای برچسب‌گذاری شده شکل پیشرفته‌ای از قالب‌ها هستند که در آن برچسب‌ها به شما اجازه می‌دهند تا کلمات قالب را با یک تابع تجزیه کنید. تابع تگ اولین پارامتر را به عنوان آرایه‌ای از رشته‌ها و پارامترهای باقی مانده را به عنوان عبارات می‌پذیرد. این تابع همچنین می‌تواند رشته‌های دستکاری شده را بر اساس پارامترها برگرداند. بیایید نحوه استفاده از رفتار الگوی برچسب‌گذاری شده مجموعه مهارت‌های حرفه‌ای فناوری اطلاعات در یک سازمان را ببینیم.

```

var user1 = 'John';
var skill1 = 'JavaScript';
var experience1 = 15;

var user2 = 'Kane';
var skill2 = 'JavaScript';
var experience2 = 5;

function myInfoTag(strings, userExp, experienceExp, skillExp) {
  var str0 = strings[0]; // "Mr/Ms. "
  var str1 = strings[1]; // " is a/an "
  var str2 = strings[2]; // "in"

  var expertiseStr;
  if (experienceExp > 10){
    expertiseStr = 'expert developer';
  } else if(skillExp > 5 && skillExp <= 10) {
    expertiseStr = 'senior developer';
  } else {
    expertiseStr = 'junior developer';
  }

  return ${str0}${userExp}${str1}${expertiseStr}${str2}${skillExp};
}

var output1 = myInfoTag`Mr/Ms. ${ user1 } is a/an ${ experience1 } in ${skill1}`;
var output2 = myInfoTag`Mr/Ms. ${ user2 } is a/an ${ experience2 } in ${skill2}`;

console.log(output1); // Mr/Ms. John is a/an expert developer in JavaScript
console.log(output2); // Mr/Ms. Kane is a/an junior developer in JavaScript

```

What are raw strings .۳۱۳

ES6 با استفاده از روش «String.raw»() ویژگی رشته‌های خام را ارائه می‌کند که برای دریافت شکل رشته خام رشته‌های الگو استفاده می‌شود. این ویژگی به شما این امکان را می‌دهد تا به رشته‌های خام همانطور که وارد شده‌اند، بدون پردازش دنباله‌های فرار دسترسی داشته باشید. به عنوان مثال، استفاده به صورت زیر خواهد بود،

```

var calculationString = String.raw `The sum of numbers is \n${1+2+3+4}!`;
console.log(calculationString); // The sum of numbers is 10

```

اگر از رشته‌های خام استفاده نمی‌کنید، دنباله کاراکترهای خط جدید با نمایش خروجی در چندین خط پردازش می‌شود.

```

var calculationString = `The sum of numbers is \n${1+2+3+4}!`;
console.log(calculationString);
// The sum of numbers is
// 10

```

همچنین، ویژگی خام در اولین آرگومان تابع تگ موجود است

```
function tag(strings) {  
  console.log(strings.raw[0]);  
}
```

What is destructuring assignment .۳۱۴

تخصیص تخریب ساختار یک عبارت جاوا اسکریپت است که امکان باز کردن مقادیر آرایه ها یا خصوصیات از اشیاء به متغیرهای مجزا را فراهم می کند.

باید مقادیر ما را از یک آرایه با استفاده از تخصیص ساختارشکن به دست آوریم

```
var [one, two, three] = ['JAN', 'FEB', 'MARCH'];  
  
console.log(one); // "JAN"  
console.log(two); // "FEB"  
console.log(three); // "MARCH"
```

و شما می توانید ویژگی های کاربر یک شی را با استفاده از انتساب تخریب، به دست آورید،

```
var {name, age} = {name: 'John', age: 32};  
  
console.log(name); // John  
console.log(age); // 32
```

What are default values in destructuring assignment .۳۱۵

زمانی که مقدار باز شده از آرایه یا شیء در طول تخصیص ساختارشکن تعریف نشده باشد، می توان به یک متغیر یک مقدار پیش فرض اختصاص داد. این کمک می کند تا از تنظیم مقادیر پیش فرض جداگانه برای هر انتساب جلوگیری کنید. باید برای هر دو آرایه و موارد استفاده از شی مثالی بزنیم،

:Arrays destructuring

```
var x, y, z;  
  
[x=2, y=4, z=6] = [10];  
console.log(x); // 10  
console.log(y); // 4  
console.log(z); // 6
```

:Objects destructuring

```
var {x=2, y=4, z=6} = {x: 10};  
  
console.log(x); // 10  
console.log(y); // 4  
console.log(z); // 6
```

How do you swap variables in destructuring assignment .۳۱۶

اگر از تخصیص تخریب ساختار استفاده نمی کنید، تعویض دو مقدار به یک متغیر وقت نیاز دارد. در حالی که با استفاده از یک ویژگی ساختارشکن، دو مقدار متغیر را می توان در یک عبارت ساختار شکن جایگزین کرد. باید دو متغیر عددی را در انتساب ساختارزدایی آرایه با هم عوض کنیم،

```
var x = 10, y = 20;  
  
[x, y] = [y, x];  
console.log(x); // 20  
console.log(y); // 10
```

What are enhanced object literals .۳۱۷

حروف الفبای شی، ایجاد سریع اجسام با ویژگی های درون بریس های فرفری را آسان می کند. برای مثال، نحو کوتاهتری را برای تعریف ویژگی شی مشترک به شرح زیر ارائه می کند.

```
//ES6  
var x = 10, y = 20  
obj = { x, y }  
console.log(obj); // {x: 10, y:20}  
//ES5  
var x = 10, y = 20  
obj = { x : x, y : y }  
console.log(obj); // {x: 10, y:20}
```

What are dynamic imports .۳۱۸

وارادات پویا با استفاده از نحو تابع «import()» به ما اجازه می دهد تا ماثولوها را در صورت تقاضا با استفاده از دستورات یا دستور `await` بارگذاری کنیم. در حال حاضر این ویژگی در [پیشنهاد مرحله 4](<https://github.com/tc39/proposal-async/await>) است. مزیت اصلی واردات پویا کاهش اندازه بسته های ما، پاسخ حجم/بار بار درخواست های ما و بهبود کلی در تجربه کاربر است. نحو واردات پویا به صورت زیر خواهد بود.

```
import('./Module').then(Module => Module.method());
```

What are the use cases for dynamic imports .۳۱۹

در زیر برخی از موارد استفاده از واردات پویا نسبت به واردات استاتیک آورده شده است.
۱. یک ماثول را به صورت درخواستی یا مشروط وارد کنید. به عنوان مثال، اگر می خواهید یک polyfill را در مرورگر قدیمی بارگذاری کنید

```
if (isLegacyBrowser()) {  
    import(...)  
    .then(...);  
}
```

۲. تعیین کننده ماژول را در زمان اجرا محاسبه کنید. به عنوان مثال می توانید از آن برای بین المللی سازی استفاده کنید.

```
import(`messages_${getLocale()}.js`).then(...);
```

۳. یک ماژول را از داخل یک اسکریپت معمولی به جای یک ماژول وارد کنید.

What are typed arrays. ۳۲۰

آرایه های تایپ شده اشیایی آرایه مانند از ECMAScript 6 API برای مدیریت داده های باینری هستند. جاوا اسکریپت ۸ نوع آرایه تایپ شده را ارائه می دهد،

۱. آرایه ای از اعداد صحیح امضا شده 8 بیتی :Int8Array
۲. آرایه ای از اعداد صحیح امضا شده 16 بیتی :Int16Array
۳. آرایه ای از اعداد صحیح امضا شده 32 بیتی :Int32Array
۴. آرایه ای از اعداد صحیح بدون علامت 8 بیتی :Uint8Array
۵. آرایه ای از اعداد صحیح بدون علامت 16 بیتی :Uint16Array
۶. آرایه ای از اعداد صحیح بدون علامت 32 بیتی :Uint32Array
۷. آرایه ای از اعداد ممیز شناور 32 بیتی :Float32Array
۸. آرایه ای از اعداد ممیز شناور 64 بیتی :Float64Array

به عنوان مثال، شما می توانید یک آرایه از اعداد صحیح امضا شده 8 بیتی مانند زیر ایجاد کنید

```
const a = new Int8Array();  
// You can pre-allocate n bytes  
const bytes = 1024  
const a = new Int8Array(bytes)
```

What are the advantages of module loaders. ۳۲۱

- ماژول لودر ویژگی های زیر را ارائه می دهد
۱. Dynamic loading
 ۲. State isolation
 ۳. Global namespace isolation
 ۴. Compilation hooks
 ۵. Nested virtualization

برای مرتب سازی مجموعه ای از رشته ها و جستجو در مجموعه ای از رشته ها استفاده می شود. این پارامتر توسط محلی و از Unicode آگاه است. باید ویژگی های مقایسه و مرتب سازی را در نظر بگیریم، **:Comparison** .

```
var list = [ "ä", "a", "z" ]; // In German, "ä" sorts with "a" Whereas in Swedish, "ä" sorts after "z"
var l10nDE = new Intl.Collator("de");
var l10nSV = new Intl.Collator("sv");
console.log(l10nDE.compare("ä", "z") === -1); // true
console.log(l10nSV.compare("ä", "z") === +1); // true
```

:Sorting .۲

```
var list = [ "ä", "a", "z" ]; // In German, "ä" sorts with "a" Whereas in Swedish, "ä" sorts after "z"
var l10nDE = new Intl.Collator("de");
var l10nSV = new Intl.Collator("sv");
console.log(list.sort(l10nDE.compare)) // [ "a", "ä", "z" ]
console.log(list.sort(l10nSV.compare)) // [ "a", "z", "ä" ]
```

What is for...of statement .۳۲۳

دستور **for...of** یک حلقه تکرار بر روی اشیاء یا عناصر قابل تکرار مانند رشته داخلی، آرایه، اشیاء آرایه مانند (مانند آرگومان ها یا NodeList، TypedArray، Map، Set و تکرارهای تعریف شده توسط کاربر ایجاد می کند. کاربرد اصلی عبارت for...of در آرایه ها به صورت زیر خواهد بود.

```
let arrayIterable = [10, 20, 30, 40, 50];

for (let value of arrayIterable) {
  value++;
  console.log(value); // 11 21 31 41 51
}
```

What is the output of below spread operator array .۳۲۴

```
[... 'John Resig']
```

خروجی آرایه [J, 'o', 'h', 'n', 'R', 'e', 's', 'i', 'g'] است.

توضیح: رشته یک نوع تکرارپذیر است و عملگر **spread** در یک آرایه هر کarakتر یک تکرارپذیر را به یک عنصر نگاشت می کند. از این رو، هر کarakتر یک رشته به عنصری در یک آرایه تبدیل می شود.

Is PostMessage secure .۳۲۵

بله، تا زمانی که برنامه نویس توسعه دهنده مراقب منشأ و منبع پیام دریافتی باشد، postMessages را می‌توان بسیار امن در نظر گرفت. اما اگر بخواهید پیام را بدون تأیید منبع آن ارسال یا دریافت کنید، حملات اسکریپت بین سایتی ایجاد می‌شود.

What are the problems with postmessage target origin as wildcard .۳۲۶

آرگومان دوم متده postMessage مشخص می‌کند که کدام مبدأ مجاز به دریافت پیام است. اگر از علامت "*" به عنوان آرگومان استفاده کنید، هر منبعی مجاز به دریافت پیام است. در این حالت، هیچ راهی برای پنجره فرستنده وجود ندارد که بفهمد پنجره هدف در هنگام ارسال پیام در مبدأ هدف قرار دارد یا خیر. اگر پنجره هدف به مبدأ دیگری هدایت شود، مبدأ دیگر داده‌ها را دریافت می‌کند. از این رو، این ممکن است منجر به آسیب‌پذیری‌های XSS شود.

```
targetWindow.postMessage(message, '*');
```

How do you avoid receiving postMessages from attackers .۳۲۷

از آنجایی که شنونده به هر پیامی گوش می‌دهد، مهاجم می‌تواند برنامه را با ارسال پیامی از مبدأ مهاجم فریب دهد، که این تصور را ایجاد می‌کند که گیرنده پیام را از پنجره فرستنده واقعی دریافت کرده است. شما می‌توانید با اعتبارسنجی مبدأ پیام در انتهای گیرنده با استفاده از ویژگی "message.origin" از این مشکل جلوگیری کنید. برای مثال، اجازه دهید مبدأ فرستنده در سمت گیرنده [www.some-receiver.com] (www.some[www.some-sender.com](http://www.some-sender.com(receiver.com) -

```
//Listener on http://www.some-receiver.com/
window.addEventListener("message", function(message){
  if(/^http://www\.some-sender\.com$/.test(message.origin)){
    console.log('You received the data from valid sender', message.data);
  }
});
```

Can I avoid using postMessages completely .۳۲۸

شما نمی‌توانید به طور کامل (یا 100٪) از postMessages استفاده نکنید. حتی اگر برنامه شما با توجه به خطرات از postMessage استفاده نمی‌کند، بسیاری از اسکریپت‌های شخص ثالث از postMessage برای برقراری ارتباط با سرویس شخص ثالث استفاده می‌کنند. بنابراین ممکن است برنامه شما بدون اطلاع شما از postMessage استفاده کند.

Is postMessages synchronous .۳۲۹

postMessages در مرورگر IE8 همگام هستند اما در IE9+، Firefox، Chrome، Safari (یعنی، Safari ناهمزن هستند. به دلیل این رفتار ناهمزن، زمانی که postMessage برگردانده می‌شود، از مکانیزم برگشت تماس استفاده می‌کنیم).

What paradigm is Javascript .۳۳۰

جاوا اسکریپت یک زبان چند پارادایم است که از برنامه نویسی امری/روشی، برنامه نویسی شی گرا و برنامه نویسی تابعی پشتیبانی می کند. جاوا اسکریپت از برنامه نویسی شی گرا با وراثت اولیه پشتیبانی می کند.

What is the difference between internal and external javascript .۳۳۱

جاوا اسکریپت داخلی: کد منبع درون تگ اسکریپت است.
جاوا اسکریپت خارجی: کد منبع در یک فایل خارجی (ذخیره شده با پسوند js) ذخیره می شود و در تگ ارجاع می شود.

Is JavaScript faster than server side script .۳۳۲

بله، جاوا اسکریپت سریعتر از اسکریپت سمت سرور است. از آنجایی که جاوا اسکریپت یک اسکریپت سمت کلاینت است، برای محاسبات یا محاسبات خود به کمک سرور وب نیازی ندارد. بنابراین جاوا اسکریپت همیشه سریعتر از هر اسکریپت سمت سرور مانند ASP، PHP و غیره است.

How do you get the status of a checkbox .۳۳۳

می توانید ویژگی «checked» را در کادر انتخاب شده در DOM اعمال کنید. اگر مقدار "True" باشد به این معنی است که چک باکس علامت زده شده است در غیر این صورت علامت آن را بردارید. به عنوان مثال، عنصر چک باکس HTML زیر را می توان با استفاده از جاوا اسکریپت به صورت زیر در دسترس قرار داد.

```
<input type="checkbox" name="checkboxname" value="Agree"> Agree the conditions<br>
```

```
console.log(document.getElementById('checkboxname').checked); // true or false
```

What is the purpose of double tilde operator .۳۳۴

عملگر دابل (~~) به عنوان عملگر bitwise double NOT (~~)tilde شناخته می شود. این عملگر قرار است جایگزین سریع تری برای .()Math.floor

How do you convert character to ASCII code .۳۳۵

برای تبدیل کاراکترهای رشته به اعداد اسکی می توانید از متدهای String.prototype.charCodeAt() استفاده کنید. برای مثال، باید کد ASCII را برای حرف اول رشته «ABC» پیدا کنیم،

```
"ABC".charCodeAt(0) // returns 65
```

در حالی که روش «String.fromCharCode» اعداد را به کاراکترهای ASCII برابر تبدیل می کند.

```
String.fromCharCode(65, 66, 67); // returns 'ABC'
```

What is ArrayBuffer .۳۳۶

یک شی ArrayBuffer برای نشان دادن یک بافر داده با پاینری خام عمومی با طول ثابت استفاده می شود. می توانید آن را به صورت زیر ایجاد کنید،

```
let buffer = new ArrayBuffer(16); // create a buffer of length 16
alert(buffer.byteLength); // 16
```

برای دستکاری یک ArrayBuffer، باید از یک شی "view" استفاده کنیم.

```
//Create a DataView referring to the buffer
let view = new DataView(buffer);
```

What is the output of below string expression .۳۳۷

```
console.log("Welcome to JS world"[0])
```

خروجی عبارت فوق "W" است.

توضیح: نماد براکت با شاخص خاص روی یک رشته کاراکتر را در یک مکان خاص برمی گرداند. از این رو، کاراکتر "W" رشته را برمی گرداند. از آنجایی که این مورد در نسخه های IE7 و پایین تر پشتیبانی نمی شود، ممکن است لازم باشد از متدهای charAt() برای به دست آوردن نتیجه دلخواه استفاده کنید.

What is the purpose of Error object .۳۳۸

سازنده Error یک شی خطا ایجاد می کند و نمونه هایی از اشیاء خطا هنگام رخدادن خطاهای زمان اجرا پرتاب می شوند. شی Error همچنین می تواند به عنوان یک شی پایه برای استثناهای تعریف شده توسط کاربر استفاده شود. نحو شیء خطا به صورت زیر خواهد بود.

```
new Error([message[, fileName[, lineNumber]]])
```

شما می توانید استثناهای خطا های تعریف شده توسط کاربر را با استفاده از شی Error در بلوک try...catch مانند زیر پرتاب کنید.

```
try {
  if(withdraw > balance)
    throw new Error("Oops! You don't have enough balance");
} catch (e) {
  console.log(e.name + ': ' + e.message);
}
```

What is the purpose of EvalError object .۳۳۹

شی EvalError یک خطا در رابطه با تابع "eval()" جهانی را نشان می دهد. حتی اگر این استثنای دیگر توسط جاوا اسکریپت پرتاب نمی شود، شی EvalError برای سازگاری باقی می ماند. نحو این عبارت به صورت زیر خواهد بود

```
new EvalError([message[, fileName[, lineNumber]]])
```

می توانید EvalError را با بلوک try...catch مانند زیر پرتاب کنید.

```
try {
  throw new EvalError('Eval function error', 'someFile.js', 100);
} catch (e) {
  console.log(e.message, e.name, e.fileName); // "Eval function error",
  "EvalError", "someFile.js"
}
```

What are the list of cases error thrown from non-strict mode to strict mode .۳۴۰

وقتی «استفاده سخت» را اعمال می کنیں. syntax، برخی از موارد زیر قبل از اجرای اسکریپت یک SyntaxError ایجاد می کنند
۱. وقتی از دستور Octal استفاده می کنید

```
var n = 022;
```

۲. استفاده از عبارت «with». وقتی از عملگر حذف روی نام متغیر استفاده می کنید ۴. استفاده از eval یا آرگومان ها به عنوان متغیر یا نام آرگومان تابع ۵. هنگامی که از کلمات کلیدی رزرو شده جدید استفاده می کنید ۶. هنگامی که یک تابع را در یک بلوک اعلام می کنید

```
if (someCondition) { function f() {} }
```

از این رو، خطاهای فوق برای جلوگیری از خطا در محیط های توسعه/تولید مفید هستند.

Is all objects have prototypes .۳۴۱

خیر. همه اشیا دارای نمونه اولیه هستند به جز شی پایه که توسط کاربر ایجاد می شود یا شیئی که با استفاده از کلمه کلیدی جدید ایجاد می شود.

What is the difference between a parameter and an argument .۳۴۲

پارامتر نام متغیر تعریف یک تابع است در حالی که یک آرگومان نشان دهنده مقدار داده شده به یک تابع در هنگام فراخوانی آن است. باید این را با یک تابع ساده توضیح دهیم

```

function myFunction(parameter1, parameter2, parameter3) {
    console.log(arguments[0]) // "argument1"
    console.log(arguments[1]) // "argument2"
    console.log(arguments[2]) // "argument3"
}
myFunction("argument1", "argument2", "argument3")

```

What is the purpose of some method in arrays .۳۴۳

متد some() برای آزمایش اینکه آیا حداقل یک عنصر در آرایه از آزمون پیاده‌سازی شده توسط تابع ارائه شده عبور می‌کند یا خیر استفاده می‌شود. متد یک مقدار بولی بر می‌گرداند. بیایید مثالی بزنیم تا هر عنصر عجیب و غریب را آزمایش کنیم،

```

var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

var odd = element ==> element % 2 !== 0;

console.log(array.some(odd)); // true (the odd element exists)

```

How do you combine two or more arrays .۳۴۴

متد concat() برای پیوستن دو یا چند آرایه با برگرداندن یک آرایه جدید حاوی تمام عناصر استفاده می‌شود. نحو به صورت زیر خواهد بود،

```
array1.concat(array2, array3, ..., arrayX)
```

بیایید مثالی از الحق آرایه با آرایه های سبزیجات و میوه ها را مثال بزنیم.

```

var veggies = ["Tomato", "Carrot", "Cabbage"];
var fruits = ["Apple", "Orange", "Pears"];
var veggiesAndFruits = veggies.concat(fruits);
console.log(veggiesAndFruits); // Tomato, Carrot, Cabbage, Apple, Orange, Pears

```

What is the difference between Shallow and Deep copy .۳۴۵

دو راه برای کپی کردن یک شی وجود دارد،
کپی کم عمق:

کپی کم عمق یک کپی بیتی از یک شی است. یک شی جدید ایجاد می‌شود که یک کپی دقیق از مقادیر موجود در شی اصلی دارد. اگر هر یک از فیلدهای شی ارجاع به اشیاء دیگر باشد، فقط آدرس های مرجع کپی می‌شوند، یعنی فقط آدرس حافظه کپی می‌شود.

مثال

```

var empDetails = {
  name: "John", age: 25, expertise: "Software Developer"
}

```

```
var empDetailsShallowCopy = empDetails //Shallow copying!
```

اگر مقداری از ویژگی را در یک تکراری به این صورت تغییر دهیم:

```
empDetailsShallowCopy.name = "Johnson"
```

دستور بالا همچنین نام "empDetails" را تغییر می دهد، زیرا ما یک کپی کم عمق داریم. این بدان معناست که ما داده های اصلی را نیز از دست می دهیم.

:Deep copy

یک کپی عمیق همه فیلدها را کپی می کند و از حافظه تخصیص یافته به صورت پویا که توسط فیلدها به آن اشاره می شود کپی می کند. یک کپی عمیق زمانی اتفاق می افتد که یک شی همراه با اشیایی که به آن اشاره دارد کپی شود.

Example

```
var empDetails = {  
    name: "John", age: 25, expertise: "Software Developer"  
}
```

یک کپی عمیق با استفاده از خواص از شی اصلی در متغیر جدید ایجاد کنید

```
var empDetailsDeepCopy = {  
    name: empDetails.name,  
    age: empDetails.age,  
    expertise: empDetails.expertise  
}
```

اکنون اگر «empDetailsDeepCopy» را تغییر دهید، فقط «empDetailsDeepCopy.name» و نه «empDetails.name» را تحت تأثیر قرار خواهد داد.

How do you create specific number of copies of a string .۳۴۶

متده «repeat()» برای ساخت و برگرداندن یک رشته جدید استفاده می شود که حاوی تعداد مشخصی از کپی های رشته ای است که روی آن فراخوانی شده و به هم پیوسته شده اند. به یاد داشته باشید که این روش به مشخصات ECMAScript 2015 اضافه شده است. بیایید یک مثال از رشته Hello را برای تکرار آن 4 بار در نظر بگیریم.

```
'Hello'.repeat(4); // 'HelloHelloHelloHello'
```

How do you return all matching strings against a regular expression .۳۴۷

متده «matchAll()» می تواند برای برگرداندن یک تکرارکننده از تمام نتایجی که یک رشته با یک عبارت منظم مطابقت دارد، استفاده شود. به عنوان مثال، مثال زیر آرایه ای از نتایج رشته منطبق را در برابر یک عبارت منظم برمنی گرداند.

```

let regexp = /Hello(\d?)/g;
let greeting = 'Hello1Hello2Hello3';

let greetingList = [...greeting.matchAll(regexp)];

console.log(greetingList[0]); //Hello1
console.log(greetingList[1]); //Hello2
console.log(greetingList[2]); //Hello3

```

How do you trim a string at the beginning or ending .۳۴۸

روش «تریم» نمونه اولیه رشته برای برش دادن دو طرف یک رشته استفاده می شود. اما اگر می خواهید بهخصوص در ابتدا یا انتهای رشته را برش دهید، می توانید از روش های «trimEnd/trimRight» و «trimStart/trimLeft» استفاده کنید. بیایید نمونه ای از این روش ها در پیام تبریک ببینیم،

```

var greeting = '    Hello, Goodmorning!    ';

console.log(greeting); // "    Hello, Goodmorning!    "
console.log(greeting.trimStart()); // "Hello, Goodmorning!    "
console.log(greeting.trimLeft()); // "Hello, Goodmorning!    "

console.log(greeting.trimEnd()); // "    Hello, Goodmorning!"
console.log(greeting.trimRight()); // "    Hello, Goodmorning!"

```

What is the output of below console statement with unary operator .۳۴۹

بیایید دستور کنسول را با عملگر unary همانطور که در زیر نشان داده شده است، بگیریم.

```
console.log(+ 'Hello');
```

خروجی دستور log کنسول فوق NaN را برمی گرداند. از آنجا که عنصر توسط عملگر unary پیشوند است و مفسر جاوا اسکریپت سعی می کند آن عنصر را به یک نوع عدد تبدیل کند. از آنجایی که تبدیل با شکست مواجه می شود، مقدار عبارت به مقدار NaN منجر می شود.

Does javascript uses mixins .۳۵۰

What is a thunk function .۳۵۱

فقط تابعی است که ارزیابی مقدار را به تأخیر می اندازد. هیچ آرگومانی نمی گیرد، اما هر زمان که thunk را فراخوانی می کنید، مقدار را می دهد. به عنوان مثال، از آن استفاده می شود که اکنون اجرا نشود، اما زمانی در آینده خواهد بود. بیایید یک مثال همزمان بگیریم،

```

const add = (x,y) => x + y;

const thunk = () => add(2,3);

thunk() // 5

```

What are asynchronous thunks .۳۵۲

های ناهمزمان برای ایجاد درخواست های شبکه مفید هستند. بیایید نمونه ای از درخواست های شبکه را ببینیم،

```

function fetchData(fn){
  fetch('https://jsonplaceholder.typicode.com/todos/1')
    .then(response => response.json())
    .then(json => fn(json))
}

const asyncThunk = function (){
  return fetchData(function getData(data){
    console.log(data)
  })
}

asyncThunk()

```

تابع "getdata" فوراً فراخوانی نمی شود، اما تنها زمانی فراخوانی می شود که داده ها از نقطه پایانی API در دسترس باشند. تابع setTimeout نیز برای ناهمزمان کردن کد ما استفاده می شود. بهترین مثال زمان واقعی، کتابخانه مدیریت حالت redux است که از های ناهمزمان برای به تأخیر انداختن اعمال برای ارسال استفاده می کند.

What is the output of below function calls .۳۵۳

:Code snippet

```

const circle = {
  radius: 20,
  diameter() {
    return this.radius * 2;
  },
  perimeter: () => 2 * Math.PI * this.radius
};

console.log(circle.diameter());
console.log(circle.perimeter());

```

:Output

خروجی 40 و NaN است. به یاد داشته باشید که قطر یک تابع منظم است، در حالی که مقدار محیط یک تابع فلش است. کلمه کلیدی «this» یک تابع معمولی (یعنی قطر) به محدوده اطراف که یک کلاس است (یعنی شی شکل) اشاره دارد. در حالی که این کلمه کلیدی تابع محیطی به محدوده اطراف که یک شی پنجره است اشاره دارد. از آنجایی که هیچ ویژگی شعاع در اشیاء پنجره وجود ندارد، یک مقدار تعریف نشده برمی‌گرداند و مضرب مقدار NaN را برمی‌گرداند.

How to remove all line breaks from a string .۳۵۴

ساده ترین روش استفاده از عبارات منظم برای شناسایی و جایگزینی خطوط جدید در رشته است. در این حالت از تابع تعویض به همراه رشته برای جایگزینی استفاده می کنیم که در مورد ما یک رشته خالی است.

```
function remove_linebreaks( var message ) {  
    return message.replace( /\r\n/gm, "" );  
}
```

در عبارت فوق `g` و `m` برای پرچم های سراسری و چند خطی هستند.

What is the difference between reflow and repaint .۳۵۵

زنگآمیزی مجدد زمانی اتفاق می افتد که تغییراتی ایجاد می شود که روی دید یک عنصر تأثیر می گذارد، اما روی طرح آن تأثیر نمی گذارد. نمونه هایی از این موارد شامل طرح کلی، نمایان بودن یا رنگ پس زمینه است. یک `reflow` شامل تغییراتی است که بر طرح بندی بخشی از صفحه (یا کل صفحه) تأثیر می گذارد. تغییر اندازه پنجره مرورگر، تغییر فونت، تغییر محتوا (مانند تایپ متن توسط کاربر)، استفاده از روش های جاوا اسکریپت شامل سبک های محاسبه شده، افزودن یا حذف عناصر از DOM، و تغییر کلاس های یک عنصر چند مورد از مواردی هستند که می توانند جریان مجدد را آغاز کنند. جریان مجدد یک عنصر باعث جریان مجدد بعدی همه عناصر فرزند و اجداد و همچنین هر عنصری که به دنبال آن در DOM است می شود.

What happens with negating an array .۳۵۶

نفی یک آرایه با کاراکتر «!»، آرایه را به یک بولی وادر می کند. از آنجایی که آرایه ها صدق در نظر گرفته می شوند، پس نفی آن "نادرست" را برمی گرداند.

```
console.log(![]); // false
```

What happens if we add two arrays .۳۵۷

اگر دو آرایه را با هم اضافه کنید، هر دو آنها را به رشته تبدیل می کند و آنها را به هم متصل می کند. به عنوان مثال، نتیجه اضافه کردن آرایه ها به صورت زیر خواهد بود.

```
console.log(['a'] + ['b']); // "ab"  
console.log([] + []); // ""  
console.log(![] + []); // "false", because ![] returns false.
```

What is the output of prepend additive operator on falsy values .۳۵۸

اگر عملگر `(+)` (additive) را روی مقادیر نادرست (`null`, `undefined`, `NaN`, `false`, "") قرار دهیم، مقدار عددی صفر تبدیل می شود. بیایید آنها را در کنسول مرورگر به صورت زیر نمایش دهیم،

```

console.log(+null); // 0
console.log(+undefined); // NaN
console.log(+false); // 0
console.log(+NaN); // NaN
console.log(+"""); // 0

```

How do you create self string using special characters .۳۵۹

شته self را می توان با ترکیب کاراکترهای [() ! + تشکیل داد. برای رسیدن به این الگو باید قراردادهای زیر را به خاطر بسپارید.

۱. از آنجایی که آرایه ها مقادیر واقعی هستند، با نفی آرایه ها false ==="" تولید می شود: ![]
 ۲. طبق قوانین اجباری جاوا اسکریپت، اضافه کردن آرایه ها به هم آنها را به رشته بندی تبدیل می کند: [] + ""
 ۳. Prepend یک آرایه با عملگر + یک آرایه را به نادرست تبدیل می کند، انکار آن را درست می کند و در نهایت تبدیل نتیجه مقدار '1' را تولید می کند: +(![]+()) ==="" 1
- با اعمال قوانین فوق می توانیم شرایط زیر را استخراج کنیم

```

![] + [] === "false"
+!+[] === 1

```

اکنون الگوی کاراکتر به صورت زیر ایجاد می شود.

```

s          e          l          f
~~~~~     ~~~~~     ~~~~~     ~~~~~
(![] + [])[3] + (![] + [])[4] + (![] + [])[2] + (![] + [])[0]
~~~~~     ~~~~~     ~~~~~     ~~~~~
(![] + []) [+!+[]]+!+[ ]+!+[ ]+ +
(![] + []) [+!+[]]+!+[ ]+!+[ ]+ +
(![] + []) [+!+[]]+!+[ ]+ +
(![] + []) [+[]]
~~~~~     ~~~~~     ~~~~~     ~~~~~
(![]+[])[+!+[]]+!+[ ]+!+[ ]+(![]+[])[+!+[]]+!+[ ]+!+[ ]+!+[ ]+(![]+[])+!+[ ]+!+[ ]+!+[ ]+!+[ ]

```

How do you remove falsy values from an array .۳۶۰

شما می توانید با وارد کردن Boolean به عنوان پارامتر، روش فیلتر را روی آرایه اعمال کنید. به این ترتیب تمام مقادیر نادرست (0، تعریف نشده، null، false و "") از آرایه حذف می شود.

```

const myArray = [false, null, 1, 5, undefined]
myArray.filter(Boolean); // [1, 5] // is same as myArray.filter(x => x);

```

How do you get unique values of an array .۳۶۱

شما می توانید مقادیر منحصر به فرد یک آرایه را با ترکیب دستور "Set" و "rest expression/spread" دریافت کنید.

```
console.log([...new Set([1, 2, 4, 4, 3])]); // [1, 2, 4, 3]
```

What is destructuring aliases .۳۶۲

گاهی اوقات شما دوست دارید یک متغیر تخریب شده با نام متفاوت از نام ویژگی داشته باشید. در این صورت، از یک «newName» برای تعیین نام برای متغیر استفاده خواهید کرد. این فرآیند نام مستعار تخریب ساختاری نامیده می شود.

```
const obj = { x: 1 };
// Grabs obj.x as as { otherName }
const { x: otherName } = obj;
```

How do you map the array values without using map method .۳۶۳

می توانید مقادیر آرایه ها را بدون استفاده از روش «نقشه» تنها با استفاده از روش «از» آرایه ترسیم کنید. باید نام شهرها را از آرایه کشورها ترسیم کنیم،

```
const countries = [
  { name: 'India', capital: 'Delhi' },
  { name: 'US', capital: 'Washington' },
  { name: 'Russia', capital: 'Moscow' },
  { name: 'Singapore', capital: 'Singapore' },
  { name: 'China', capital: 'Beijing' },
  { name: 'France', capital: 'Paris' },
];
const cityNames = Array.from(countries, ({ capital }) => capital);
console.log(cityNames); // ['Delhi', 'Washington', 'Moscow', 'Singapore', 'Beijing', 'Paris']
```

How do you empty an array .۳۶۴

با صفر کردن طول آرایه می توانید به سرعت یک آرایه را خالی کنید.

```
let cities = ['Singapore', 'Delhi', 'London'];
cities.length = 0; // cities becomes []
```

How do you rounding numbers to certain decimals .۳۶۵

می توانید با استفاده از روش «toFixed» از جاوا اسکریپت، اعداد را به تعداد معینی از اعشار گرد کنید.

```
let pie = 3.141592653;
pie = pie.toFixed(3); // 3.142
```

What is the easiest way to convert an array to an object .۳۶۶

شما می توانید با استفاده از عملگر spread (...) یک آرایه را به یک شی با همان داده تبدیل کنید.

```
var fruits = ["banana", "apple", "orange", "watermelon"];
var fruitsObject = {...fruits};
console.log(fruitsObject); // {0: "banana", 1: "apple", 2: "orange", 3: "watermelon"}
```

How do you create an array with some data .۳۶۷

می توانید با استفاده از روش «fill» یک آرایه با مقداری داده یا یک آرایه با همان مقادیر ایجاد کنید.

```
var newArray = new Array(5).fill("0");
console.log(newArray); // ["0", "0", "0", "0", "0"]
```

What are the placeholders from console object .۳۶۸

در زیر لیستی از متغیرهای موجود از شی کنسول وجود دارد،

۱. ۰% - یک شی را می گیرد،

۲. ۵% - یک رشته می گیرد،

۳. d% - برای اعشار یا عدد صحیح استفاده می شود

این متغیرها را می توان در console.log به صورت زیر نشان داد

```
const user = { "name": "John", "id": 1, "city": "Delhi"};
console.log("Hello %s, your details %o are available in the object form", "John", user);
// Hello John, your details {name: "John", id: 1, city: "Delhi"} are available in object
```

Is it possible to add CSS to console messages .۳۶۹

بله، می توانید سبکهای CSS را برای پیامهای کنسول مشابه متن html در صفحه وب اعمال کنید.

```
console.log('%c The text has blue color, with large font and red background', 'color: blue; font-size: x-large; background: red');
```

متن به صورت زیر نمایش داده می شود

```
> console.log('%c Color of the text', 'color: blue; font-size: x-large; background: red');
```

Color of the text

vendors~main.51281d83.chunk.js:

نکته: تمام سبک های CSS را می توان برای پیام های کنسول اعمال کرد.

What is the purpose of dir method of console object .۳۷۰

)» برای نمایش یک لیست تعاملی از ویژگی های شی جاوا اسکریپت مشخص شده به عنوان JSON استفاده می شود.

```
const user = { "name": "John", "id": 1, "city": "Delhi"};
console.dir(user);
```

شی کاربر نمایش داده شده در نمایش JSON

```
>     const user = { "name": "John", "id": 1, "city": "Delhi"};
      console.dir(user);

▼ Object ⓘ
  name: "John"
  id: 1
  city: "Delhi"
▶ __proto__: Object
```

Is it possible to debug HTML elements in console .۳۷۱

بله، دریافت و اشکال زدایی عناصر HTML در کنسول، درست مانند بازرسی عناصر، امکان پذیر است.

```
const element = document.getElementsByTagName("body") [0];
console.log(element);
```

این عنصر HTML را در کنسول چاپ می کند،

```
> const element = document.getElementsByTagName("body") [0];
< undefined
> console.log(element);
▶ <body class="question-page unified-theme">...</body>
< undefined
> |
```

How do you display data in a tabular format using console object .۳۷۲

برای نمایش داده‌ها در کنسول در قالب جدولی برای تجسم آرایه‌ها یا اشیاء پیچیده استفاده می‌شود.

```
const users = [{ "name": "John", "id": 1, "city": "Delhi"}, { "name": "Max", "id": 2, "city": "London"}, { "name": "Rod", "id": 3, "city": "Paris"} ];  
console.table(users);
```

داده‌هایی که در قالب جدول مشاهده می‌شوند،

```
> const users = [{ "name": "John", "id": 1, "city": "Delhi"},  
    { "name": "Max", "id": 2, "city": "London"},  
    { "name": "Rod", "id": 3, "city": "Paris"}];  
< undefined  
> console.table(users);  
VM92:1  


| (index) | name   | id | city     |
|---------|--------|----|----------|
| 0       | "John" | 1  | "Delhi"  |
| 1       | "Max"  | 2  | "London" |
| 2       | "Rod"  | 3  | "Paris"  |



▶ Array(3)


```

.Not: Remember that `console.table()` is not supported in IE

How do you verify that an argument is a Number or not .۳۷۳

ترکیبی از روش‌های `isFinite` و `isNaN` برای تأیید عدد بودن یا نبودن آرگومان استفاده می‌شود.

```
function isNumber(n){  
    return !isNaN(parseFloat(n)) && isFinite(n);  
}
```

How do you create copy to clipboard button .۳۷۴

شما باید محتوا (با استفاده از روش `select()`) عنصر ورودی را انتخاب کنید و دستور `copy` را با `execCommand` اجرا کنید (یعنی `execCommand('copy')`). شما همچنین می‌توانید سایر دستورات سیستم مانند `cut` و `paste` را اجرا کنید.

```
document.querySelector("#copy-button").onclick = function() {  
    // Select the content  
    document.querySelector("#copy-input").select();  
    // Copy to the clipboard  
    document.execCommand('copy');  
};
```

What is the shortcut to get timestamp .۳۷۵

می‌توانید از «تاریخ جدید()» برای دریافت مهر زمانی فعلی استفاده کنید. یک میانبر جایگزین برای دریافت مقدار وجود دارد.

```
console.log(+new Date());
console.log(Date.now());
```

How do you flattening multi dimensional arrays .۳۷۶

مسطح کردن آرایه های دو بعدی با عملگر Spread بی اهمیت است.

```
const biDimensionalArr = [11, [22, 33], [44, 55], [66, 77], 88, 99];
const flattenArr = [...biDimensionalArr]; // [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

اما شما می‌توانید آن را با آرایه های چند بعدی با تماس های بازگشته کار کنید،

```
function flattenMultiArray(arr) {
  const flattened = [].concat(...arr);
  return flattened.some(item => Array.isArray(item)) ? flattenMultiArray(flattened) :
flattened;
}
const multiDimensionalArr = [11, [22, 33], [44, [55, 66, [77, [88]], 99]]];
const flatArr = flattenMultiArray(multiDimensionalArr); // [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

What is the easiest multi condition checking .۳۷۷

می‌توانید از «indexOf» برای مقایسه ورودی با چندین مقدار به جای بررسی هر مقدار به عنوان یک شرط استفاده کنید.

```
// Verbose approach
if (input === 'first' || input === 1 || input === 'second' || input === 2) {
  someFunction();
}
// Shortcut
if (['first', 1, 'second', 2].indexOf(input) !== -1) {
  someFunction();
}
```

How do you capture browser back button .۳۷۸

روش «window.onbeforeunload» برای ضبط رویدادهای دکمه بازگشت مرورگر استفاده می‌شود. این برای هشدار دادن به کاربران در مورد از دست داده های فعلی مفید است.

```
window.onbeforeunload = function() {
  alert("You work will be lost");
};
```

How do you disable right click in the web page .۳۷۹

کلیک راست روی صفحه را می توان با برگرداندن `false` از ویژگی «oncontextmenu» در عنصر بدنے غیرفعال کرد.

```
<body oncontextmenu="return false;">
```

What are wrapper objects .۳۸۰

مقادیر اولیه مانند رشته، عدد و بولی خواص و روشی ندارند، اما زمانی که می خواهید اقداماتی را روی آنها انجام دهید، به طور موقت به یک شی (اجکت Wrapper) تبدیل یا مجبور می شوند. برای مثال، اگر متدهایUpperCase() را روی یک مقدار رشته اولیه اعمال کنید، خطایی ایجاد نمی کند، اما حروف بزرگ رشته را برمی گرداند.

```
let name = "john";  
  
console.log(name.toUpperCase()); // Behind the scenes treated as console.log(new  
String(name).toUpperCase());
```

یعنی هر اولیه به جز `null` و `undefined` دارای wrapper Object است و لیست اشیاء wrapper عبارتند از، `.BigInt` و `Boolean`، `Symbol`

What is AJAX .۳۸۱

AJAX مخفف XMLHttpRequest API (XML، CSS، JavaScript و XML است و گروهی از فناوری های مرتبط با HTTP است) و غیره) است که برای نمایش داده ها به صورت ناهمزمان استفاده می شود. یعنی ما می توانیم داده ها را به سرور ارسال کنیم و بدون بارگیری مجدد صفحه وب، داده ها را از سرور دریافت کنیم.

What are the different ways to deal with Asynchronous Code .۳۸۲

۱. در زیر لیستی از روش های مختلف برای مقابله با کدهای ناهمزمان آورده شده است.
۱. `callback` ها
۲. پaramیس ها
۳. `Async/await`
۴. کتابخانه های شخص ثالث مانند `bluebird` و `async.js` و غیره

How to cancel a fetch request .۳۸۳

تا چند روز پیش، یکی از کاستی های وعده های بومی راه مستقیمی برای لغو درخواست واکشی نیست. اما «AbortController» جدید از مشخصات `js` به شما امکان می دهد از سیگنالی برای لغو یک یا چند تماس واکشی استفاده کنید. جریان اصلی لغو یک درخواست واکشی به صورت زیر خواهد بود.

۱. یک نمونه «AbortController» ایجاد کنید
 ۲. ویژگی سیگنال یک نمونه را دریافت کنید و سیگنال را به عنوان یک گزینه واکشی برای سیگنال ارسال کنید
 ۳. برای لغو تمام واکشی‌هایی که از آن سیگنال استفاده می‌کنند، با ویژگی AbortController's abort می‌توان تماس بگیرید.
- به عنوان مثال، باید یک سیگنال را به چندین تماس واکشی ارسال کنیم، همه درخواست‌های با آن سیگنال لغو می‌شوند.

```

const controller = new AbortController();
const { signal } = controller;

fetch("http://localhost:8000", { signal }).then(response => {
  console.log(`Request 1 is complete!`);
}).catch(e => {
  if(e.name === "AbortError") {
    // We know it's been canceled!
  }
});

fetch("http://localhost:8000", { signal }).then(response => {
  console.log(`Request 2 is complete!`);
}).catch(e => {
  if(e.name === "AbortError") {
    // We know it's been canceled!
  }
});

// Wait 2 seconds to abort both requests
setTimeout(() => controller.abort(), 2000);

```

What is web speech API .۳۸۴

API گفتار وب برای فعال کردن مرورگرهای مدرن برای شناسایی و ترکیب گفتار (یعنی داده‌های صوتی در برنامه‌های وب) استفاده می‌شود. این API توسط انجمن W3C در سال 2012 معرفی شده است. دارای دو بخش اصلی است.

۱. **تشخیص گفتار (تشخیص گفتار ناهمزن یا گفتار به متن):** این امکان را فراهم می‌کند که زمینه صدا را از ورودی صوتی تشخیص داده و به آن پاسخ دهد. این توسط رابط "SpeechRecognition" قابل دسترسی است.
۲. مثال زیر نحوه استفاده از این API برای دریافت متن از گفتار را نشان می‌دهد.

```

window.SpeechRecognition = window.webkitSpeechRecognition || window.SpeechRecognition;
// webkitSpeechRecognition for Chrome and SpeechRecognition for FF
const recognition = new window.SpeechRecognition();
recognition.onresult = (event) => { // SpeechRecognitionEvent type
  const speechToText = event.results[0][0].transcript;
  console.log(speechToText);
}
recognition.start();

```

در این API، مرورگر برای استفاده از میکروفون شما از شما اجازه می‌خواهد (SpeechSynthesis (Text-to-Speech)): این امکان را فراهم می‌کند تا زمینه صدا را از ورودی صوتی تشخیص داده و پاسخ دهد. این توسط رابط "SpeechSynthesis" قابل دسترسی است. به عنوان مثال، کد زیر برای دریافت صدا/گفتار از متن استفاده می‌شود.

```

if('speechSynthesis' in window){
  var speech = new SpeechSynthesisUtterance('Hello World!');
  speech.lang = 'en-US';
  window.speechSynthesis.speak(speech);
}

```

نمونه های بالا را می توان روی کنسول برنامه نویس مرورگر کروم (+33) آزمایش کرد.

توجه: این API هنوز یک پیش نویس فعال است و فقط در مرورگرهای کروم و فایرفاکس موجود است (البته کروم فقط مشخصات را اجرا کرده است)

What is minimum timeout throttling .۳۸۵

هم مرورگر و هم محیط های جاوا اسکریپت NodeJS با حداقل تأخیری که بیشتر از 0 میلی ثانیه است دریچه گاز را انجام می دهند. این بدان معناست که حتی اگر تنظیم یک تأخیر 0ms به طور آنی اتفاق نیفتد.

مرورگرها: حداقل 4 میلی ثانیه تأخیر دارند. این دریچه گاز زمانی اتفاق می افتد که تماس های متوالی به دلیل تودرتوی برگشت به تماس (عمق معین) یا پس از تعداد معینی فواصل متوالی آغاز شود.

توجه: مرورگرهای قدیمی حداقل 10 میلی ثانیه تأخیر دارند.

Nodejs: حداقل 1ms تأخیر دارند. این دریچه گاز زمانی اتفاق می افتد که تأخیر بزرگتر از 2147483647 یا کمتر از 1 باشد. بهترین مثال برای توضیح این رفتار throttling وقفه، ترتیب قطعه کد زیر است.

```

function runMeFirst() {
  console.log('My script is initialized');
}
setTimeout(runMeFirst, 0);
console.log('Script loaded');

```

و خروجی در خواهد بود

```

Script loaded
My script is initialized

```

اگر از «setTimeout» استفاده نمی کنید، ترتیب گزارش ها به ترتیب خواهد بود.

```

function runMeFirst() {
  console.log('My script is initialized');
}
runMeFirst();
console.log('Script loaded');

```

و خروجی این است،

```

My script is initialized
Script loaded

```

How do you implement zero timeout in modern browsers .۳۸۶

به دلیل حداقل تاخیر بیش از ۰ میلی ثانیه، نمی توانید از `setTimeout(fn, 0)` برای اجرای فوری کد استفاده کنید. اما برای دستیابی به این رفتار می توانید از `window.postMessage()` استفاده کنید.

What are tasks in event loop .۳۸۷

وظیفه هر کد/برنامه جاوا اسکریپتی است که قرار است توسط مکانیسمهای استاندارد اجرا شود، مانند شروع اولیه اجرای یک برنامه، اجرای یک فرآخوان رویداد، یا یک بازه زمانی یا وقفه در حال اجرا. همه این وظایف در یک صف کار برنامه ریزی می شوند.

در زیر لیستی از موارد استفاده برای افزودن وظایف به صف کار آمده است.

۱. هنگامی که یک برنامه جاوا اسکریپت جدید مستقیماً از کنسول اجرا می شود یا توسط عنصر `<script>` اجرا می شود، وظیفه به صف کار اضافه می شود.

۲. هنگامی که یک رویداد شلیک می شود، پاسخ تماس رویداد به صف کار اضافه می شود

۳. وقتی به یک `setInterval` یا `setTimeout` رسید، پاسخ تماس مربوطه به صف کار اضافه می شود

What is microtask .۳۸۸

Microtask کد جاوا اسکریپت است که باید بلافاصله پس از تکمیل وظیفه/میکرووظیفه در حال اجرا اجرا شود. آنها در طبیعت به نوعی مسدود کننده هستند. یعنی تا زمانی که صف microtask خالی نشود، رشته اصلی مسدود خواهد شد.

منابع اصلی ریزکارها عبارتند از `Promise.resolve`, `Promise.reject`, `MutationObservers`, `IntersectionObservers` و `Promise`.

غیره.

توجه: همه این ریزکارها در همان چرخش حلقه رویداد پردازش می شوند.

What are different event loops .۳۸۹

What is the purpose of queueMicrotask .۳۹۰

How do you use javascript libraries in typescript file .۳۹۱

مشخص است که همه کتابخانه‌ها یا چارچوب‌های جاوا اسکریپت دارای فایل‌های اعلان TypeScript نیستند. اما اگر همچنان می‌خواهید از کتابخانه‌ها یا فریمورک‌ها در فایل‌های TypeScript خود بدون دریافت خطاهای کامپایل استفاده کنید، تنها راه حل کلمه کلیدی «اعلان» به همراه یک اعلان متغیر است. به عنوان مثال، بباید تصور کنیم که شما یک کتابخانه به نام "customLibrary" دارید که اعلان TypeScript ندارد و فضای نامی به نام "customLibrary" در فضای نام جهانی دارد. می توانید از این کتابخانه در کد تایپ اسکریپت به صورت زیر استفاده کنید.

```
declare var customLibrary;
```

در زمان اجرا، تایپ اسکریپت نوع آن را به متغیر «کتابخانه سفارشی» به صورت «هرگونه» ارائه می‌کند. جایگزین دیگر بدون استفاده از کلمه کلیدی declare در زیرآمده است

```
var customLibrary: any;
```

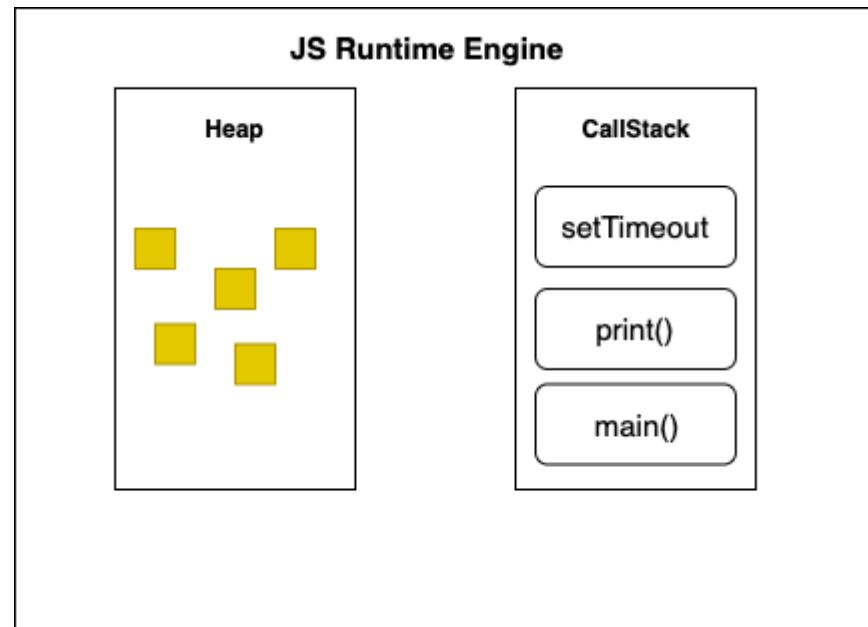
What are the differences between promises and observables .۳۹۲

برخی از تفاوت‌های عمدۀ در شکل جدولی

قابل مشاهده	وعده ها
چندین مقدار را در یک دوره زمانی منتشر می‌کند (جريان مقاديری از ۰ تا چندگانه)	فقط یک مقدار را در یک زمان منتشر می‌کند
تنبل در طبیعت؛ آنها برای فراخوانی نیاز به اشتراک دارند	مشتاق در طبیعت؛ قرار است فوراً فراخوانی شوند
قابل مشاهده می‌تواند همزمان یا ناهمزمان	Promise همیشه ناهمزمان است حتی اگر بلافاصله حل شود
retryWhen و map، forEach، filter، reduce، retry و غیره را ارائه می‌دهد.	هیچ اپراتور ارائه نمی‌دهد
با استفاده از روش ()unsubscribe لغو شد	قابل لغو نیست

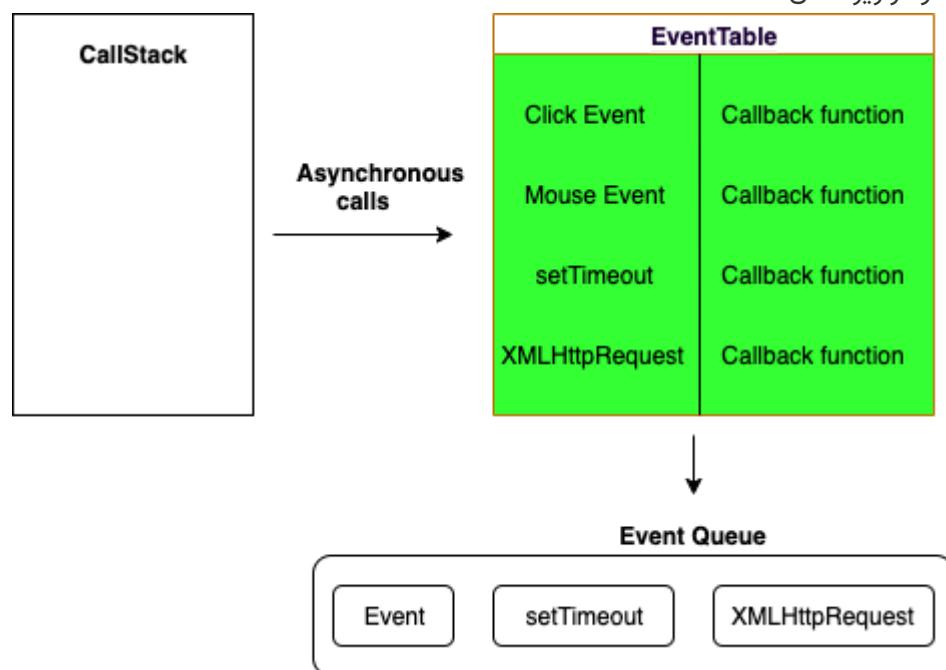
What is heap .۳۹۳

Heap (یا پشته حافظه) مکانی است که در آن اشیا در هنگام تعریف متغیرها ذخیره می‌شوند. یعنی این مکانی است که تمام تخصیص حافظه و عدم تخصیص در آن انجام می‌شود. هر دو heap و call-stack دو ظرف زمان اجرا JS هستند. هر زمان که زمان اجرا با متغیرها و اعلان‌های تابع در کد مواجه می‌شود، آنها را در Heap ذخیره می‌کند.



What is an event table .۳۹۴

یک ساختار داده ای است که تمام رویدادهایی را که به صورت ناهمزمان اجرا می شوند، مانند پس از مدتی فاصله زمانی یا پس از رفع برخی از درخواست های API، ذخیره و رديابی می کند. یعنی هر زمان که یک تابع `setTimeout` را فراخوانی کنید یا عملیات `async` را فراخوانی کنید، به جدول رویداد اضافه می شود. توابع را به تنها یی اجرا نمی کند. هدف اصلی جدول رویدادها پیگیری رویدادها و ارسال آنها به صف رویداد همانطور که در نمودار زیر نشان داده شده است.



What is a microTask queue .۳۹۵

صف جدیدی است که در آن تمام وظایف آغاز شده توسط اشیاء وعده قبل از صفحه برگشت پردازش می شوند.

صف microtasks قبل از کارهای رندر و نقاشی بعدی پردازش می شود. اما اگر این ریزکارها برای مدت طولانی اجرا شوند، منجر به تخریب بصری می شود.

What is the difference between shim and polyfill .۳۹۶

شیم کتابخانه ای است که یک API جدید را با استفاده از ابزارهای آن محیط به یک محیط قدیمی تر می آورد. لزوماً محدود به یک برنامه وب نیست. به عنوان مثال، es5-shim.js برای شبیه سازی ویژگی های ES5 در مرورگرهای قدیمی (عمدتاً قبل از IE9) استفاده می شود.

در حالی که polyfill یک قطعه کد (یا افزونه) است که فناوری را ارائه می کند که شما، توسعه دهنده، از مرورگر انتظار دارید که به صورت بومی ارائه کند. در یک جمله ساده، polyfill یک شیم برای API مرورگر است.

How do you detect primitive or non primitive value type .۳۹۷

در جاوا اسکریپت، انواع ابتدایی عبارتند از undefined، boolean، string، number، BigInt، null، Symbol و Object. در حالی که انواع غیر ابتدایی شامل Object ها می شود. اما با تابع زیر می توانید به راحتی آنها را شناسایی کنید

```
var myPrimitive = 30;
var myNonPrimitive = {};
function isPrimitive(val) {
    return Object(val) !== val;
}

isPrimitive(myPrimitive);
isPrimitive(myNonPrimitive);
```

اگر مقدار یک نوع داده اولیه باشد، سازنده Object یک شیء پوشاننده جدید برای مقدار ایجاد می کند. اما اگر مقدار یک نوع داده غیر ابتدایی (یک شی) باشد، سازنده Object همان شی را می دهد.

What is babel .۳۹۸

یک ترانسپایلر جاوا اسکریپت برای تبدیل کد ECMAScript 2015+ به یک نسخه سازگار جاوا اسکریپت در مرورگرها یا محیط های فعلی و قدیمی تر است. برخی از ویژگی های اصلی در زیر ذکر شده است،

۱. تبدیل نحو

۲. ویژگی های Polyfill که در محیط هدف شما وجود ندارد (با استفاده از @babel/polyfill)

۳. تبدیل کد منبع (یا کد مد)

Is Node.js completely single threaded .۳۹۹

یک رشته است، اما برخی از توابع موجود در کتابخانه استاندارد Node.js (به عنوان مثال، توابع ماژول fs) تک رشته ای نیستند. یعنی منطق آنها خارج از رشته Node.js اجرا می شود تا سرعت و عملکرد یک برنامه را بهبود بخشد.

What are the common use cases of observables .۴۰۰

برخی از رایج‌ترین موارد استفاده از موارد مشاهده‌شده عبارتند از سوکت‌های وب با اعلان‌های فشار، تغییرات ورودی کاربر، فواصل تکراری و غیره.

What is RxJS .۴۰۱

RxJS (افزونه‌های واکنش‌گرا برای جاوا اسکریپت) کتابخانه‌ای برای پیاده‌سازی برنامه‌نویسی واکنش‌گرا با استفاده از مشاهده‌پذیر است که نوشتمن کد ناهمزنان یا مبتنی بر تماس را آسان‌تر می‌کند. همچنین تابع کاربردی را برای ایجاد و کار با مشاهده‌پذیرها فراهم می‌کند.

What is the difference between Function constructor and function declaration .۴۰۲

توابعی که با "سازنده تابع" ایجاد می‌شوند، برای زمینه‌های ایجاد خود بسته ایجاد نمی‌کنند، اما همیشه در محدوده جهانی ایجاد می‌شوند. یعنی تابع فقط می‌تواند به متغیرهای محلی خود و متغیرهای دامنه جهانی دسترسی داشته باشد. در حالی که اعلان‌های تابع می‌توانند به متغیرهای تابع بیرونی (بسته شدن) نیز دسترسی داشته باشند.

بیایید این تفاوت را با یک مثال بینیم،

:Function Constructor

```
var a = 100;
function createFunction() {
  var a = 200;
  return new Function('return a;');
}
console.log(createFunction()); // 100
```

:Function declaration

```
var a = 100;
function createFunction() {
  var a = 200;
  return function func() {
    return a;
  }
}
console.log(createFunction()); // 200
```

What is a Short circuit condition .٤٠٣

شرایط اتصال کوتاه برای روش فشرده نوشتن دستورات if ساده در نظر گرفته شده است. بیا بید سناریو را با استفاده از یک مثال نشان دهیم. اگر می خواهیم وارد پورتالی با شرایط احراز هویت شوید، عبارت زیر خواهد بود:

```
if (authenticate) {  
    loginToPorta();  
}
```

از آنجایی که عملگرهای منطقی جاوا اسکریپت از چپ به راست ارزیابی می شوند، عبارت فوق را می توان با استفاده از عملگر منطقی `&&` ساده کرد.

```
authenticate && loginToPorta();
```

What is the easiest way to resize an array .٤٠٤

ویژگی `length` یک آرایه برای تغییر اندازه یا خالی کردن سریع آرایه مفید است. بیا بید ویژگی `length` را روی آرایه اعداد اعمال کنیم تا تعداد عناصر را از 5 به 2 تغییر دهیم.

```
var array = [1, 2, 3, 4, 5];  
console.log(array.length); // 5  
  
array.length = 2;  
console.log(array.length); // 2  
console.log(array); // [1,2]
```

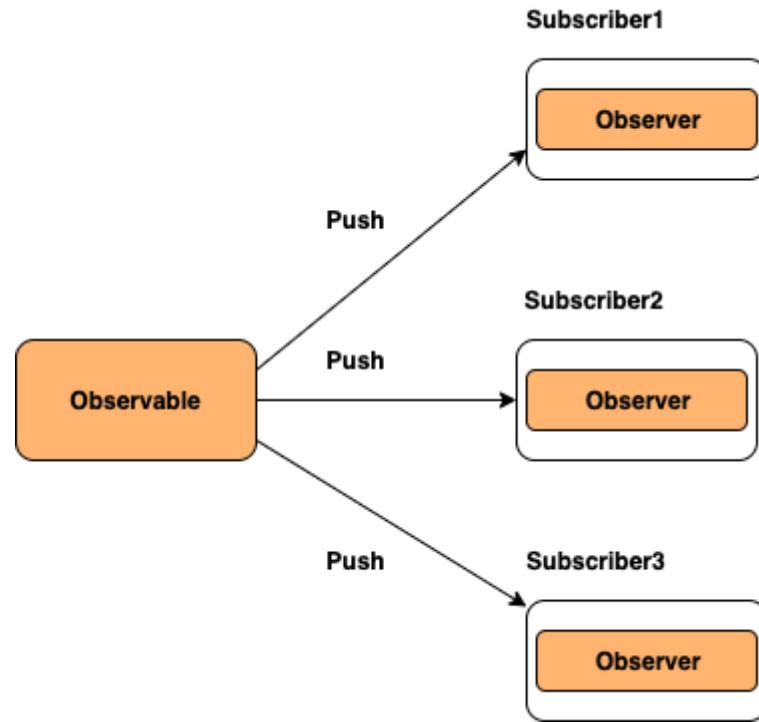
و آرایه را نیز می توان خالی کرد

```
var array = [1, 2, 3, 4, 5];  
array.length = 0;  
console.log(array.length); // 0  
console.log(array); // []
```

What is an observable .٤٠٥

اساساً Observable تابعی است که می تواند جریانی از مقادیر را به صورت همزمان یا ناهمزمان به یک ناظر در طول زمان برگرداند. مصرف کننده می تواند با فراخوانی متند «`subscribe()`» مقدار را دریافت کند. بیا بید به یک مثال ساده از یک Observable نگاه کنیم

```
import { Observable } from 'rxjs';  
  
const observable = new Observable(observer => {  
    setTimeout(() => {  
        observer.next('Message from a Observable!');  
    }, 3000);  
});  
  
observable.subscribe(value => console.log(value));
```



توجه: مشاهده پذیرها هنوز بخشی از زبان جاوا اسکریپت نیستند اما پیشنهاد شده است که به زبان اضافه شوند.

What is the difference between function and class declarations .۴۰۶

تفاوت اصلی بین اعلان های تابع و اعلان های کلاس "بالا بردن" است. اعلان های تابع بالا می روند اما اعلان های کلاس نیستند.

:**Classes**

```
const user = new User(); // ReferenceError
class User {}
```

:**Constructor Function**

```
const user = new User(); // No error
function User() {}
```

What is an async function .۴۰۷

یک تابع همگام، تابعی است که با کلمه کلیدی «ناهمگام» اعلام شده است که با اجتناب از زنجیره وعده، رفتار ناهمزمان و مبتنی بر قول را قادر می سازد به سبک تمیزتری نوشته شود. این توابع می توانند شامل صفر یا بیشتر عبارت «انتظار» باشند. بیایید یک مثال تابع همگام زیر را در نظر بگیریم،

```

async function logger() {

  let data = await fetch('http://someapi.com/users'); // pause until fetch returns
  console.log(data)
}

logger();

```

این اساساً قند نحوی بیش از پرامیس ها و توابع جنریتور ES2015 است.

How do you prevent promises swallowing errors .۱۴۰۸

در حین استفاده از کد ناهمزمان، وعده های ES6 جاوا اسکریپت می توانند زندگی شما را بدون داشتن هرم برگشت تماس و مدیریت خطا در هر خط دوم بسیار آسان تر کند. اما Promises مشکلاتی دارد و بزرگ ترین آنها به طور پیش فرض بلعیدن خطاهای است.

فرض کنید انتظار دارید برای تمام موارد زیر یک خطا در کنسول چاپ کنید.

```

Promise.resolve('promised value').then(function() {
  throw new Error('error');
});

Promise.reject('error value').catch(function() {
  throw new Error('error');
});

new Promise(function(resolve, reject) {
  throw new Error('error');
});

```

اما بسیاری از محیط های جاوا اسکریپت مدرن وجود دارند که هیچ خطایی را چاپ نمی کنند. شما می توانید این مشکل را به روش های مختلف حل کنید،

Add catch block at the end of each chain: You can add catch block to the end of each of your promise chains

```

Promise.resolve('promised value').th      (function() {
  throw new Error('error');
}).catch(function(error) {
  console.error(error.stack);
});

```

اما تایپ کردن برای هر زنجیره پرامیس ها و پر مخاطب نیز بسیار دشوار است.
می توانید ابتدا راه حل ها را جایگزین کنید و سپس با روش انجام شده بلوک ها را بگیرید

```

Promise.resolve('promised value').don(function() {
  throw new Error('error');
});

```

فرض کنید می خواهید داده ها را با استفاده از HTTP واکشی کنید و بعداً پردازش داده های حاصل را به صورت ناهمزمان انجام دهید. می توانید بلوک «انجام شد» را به صورت زیر بنویسید.

```
getDataFromHttp()
  .then(function(result) {
    return processDataAsync(result);
  })
  .done(function(processed) {
    displayData(processed);
  });
}
```

در آینده، اگر API کتابخانه پردازش به همگام تغییر کند، می‌توانید بلوک «انجام شد» را مانند زیر حذف کنید.

```
getDataFromHttp()
  .then(function(result) {
    return displayData(processDataAsyn(result));
  })
}
```

سپس فراموش کرده اید که بلوک «انجام شد» را به بلوک «سپس» اضافه کنید که منجر به خطاهای خاموش می‌شود.
:Extend ES6 Promises by Bluebird.^۳

یک کنترل کننده "پیش فرض" در Rejection است که تمام خطاهای را از Promises رد شده به stderr چاپ می‌کند. پس از نصب، می‌توانید ردهای کنترل نشده را پردازش کنید

```
Promise.onPossiblyUnhandledRejection(function(error){
  throw error;
});
```

یک رد را دور بیندازید، فقط با یک شکار خالی آن را مدیریت کنید

```
Promise.reject('error value').catch(function() {});
```

What is deno .۴۰۹

یک زمان اجرا ساده، مدرن و ایمن برای جاوا اسکریپت و تایپ اسکریپت است که از موتور جاوا اسکریپت V8 و زبان برنامه نویسی Rust استفاده می‌کند.

How do you make an object iterable in javascript .۴۱۰

به طور پیش فرض، اشیاء ساده قابل تکرار نیستند. اما می‌توانید با تعریف ویژگی «Symbol.iterator» روی آن، شی را قابل تکرار کنید.
بیایید این را با یک مثال نشان دهیم،

```

const collection = {
  one: 1,
  two: 2,
  three: 3,
  [Symbol.iterator]() {
    const values = Object.keys(this);
    let i = 0;
    return {
      next: () => {
        return {
          value: this[values[i++]],
          done: i > values.length
        }
      }
    };
  }
};

const iterator = collection[Symbol.iterator]();

console.log(iterator.next());    // → {value: 1, done: false}
console.log(iterator.next());    // → {value: 2, done: false}
console.log(iterator.next());    // → {value: 3, done: false}
console.log(iterator.next());    // → {value: undefined, done: true}

```

فرآیند فوق را می توان با استفاده از یک تابع مولد ساده کرد،

```

const collection = {
  one: 1,
  two: 2,
  three: 3,
  [Symbol.iterator]: function * () {
    for (let key in this) {
      yield this[key];
    }
  }
};

const iterator = collection[Symbol.iterator]();
console.log(iterator.next());    // {value: 1, done: false}
console.log(iterator.next());    // {value: 2, done: false}
console.log(iterator.next());    // {value: 3, done: false}
console.log(iterator.next());    // {value: undefined, done: true}

```

What is a Proper Tail Call .۴۱۱

ابتدا، قبل از صحبت در مورد "دوم خوانی مناسب" باید در مورد دم دم بدانیم. فراخوانی دنباله یک فراخوانی فرعی یا تابعی است که به عنوان آخرین عمل یک تابع فراخوانی انجام می شود. در حالی که **فراخوانی دنباله مناسب (PTC)** تکنیکی است که در آن برنامه یا کد فریم های پشته ای اضافی برای بازگشت ایجاد نمی کند، زمانی که فراخوانی تابع یک فراخوانی دنباله است.

برای مثال، بازگشت کلاسیک یا سر تابع فاکتوریل زیر به پشته برای هر مرحله بستگی دارد. هر مرحله باید تا " n^* فاکتوریل(- n)" پردازش شود

```

function factorial(n) {
  if (n === 0) {
    return 1
  }
  return n * factorial(n - 1)
}
console.log(factorial(5)); //120

```

اما اگر از توابع بازگشته Tail استفاده می‌کنید، آن‌ها تمام داده‌های لازم را که به آن نیاز دارد را بدون تکیه بر پشته، در بازگشت به پایین منتقل می‌کنند.

```

function factorial(n, acc = 1) {
  if (n === 0) {
    return acc
  }
  return factorial(n - 1, n * acc)
}
console.log(factorial(5)); //120

```

الگوی بالا همان خروجی مورد اول را برمی‌گرداند. اما ابیاشت کننده کل را به عنوان آرگومان بدون استفاده از حافظه پشته در تماس‌های بازگشته ردیابی می‌کند.

How do you check an object is a promise or not .۱۴

اگر نمی‌دانید یک مقدار یک وعده است یا نه، مقدار را به صورت «Promise.resolve(value)» بپیچید که یک قول را برمی‌گرداند.

```

function isPromise(object){
  if(Promise && Promise.resolve){
    return Promise.resolve(object) == object;
  }else{
    throw "Promise not supported in your environment"
  }
}

var i = 1;
var promise = new Promise(function(resolve,reject){
  resolve()
});

console.log(isPromise(i)); // false
console.log(isPromise(p)); // true

```

راه دیگر این است که نوع «.then()» را بررسی کنید

```

function isPromise(value) {
    return Boolean(value && typeof value.then === 'function');
}
var i = 1;
var promise = new Promise(function(resolve,reject){
    resolve()
});

console.log(isPromise(i)) // false
console.log(isPromise(promise)); // true

```

How to detect if a function is called as constructor .۴۱۳

می‌توانید از ویژگی شبیه «new.target» برای تشخیص اینکه آیا یک تابع به عنوان سازنده (با استفاده از عملگر جدید) فراخوانی شده است یا به عنوان یک فراخوانی تابع معمولی استفاده کنید.

۱. اگر سازنده یا تابعی با استفاده از عملگر جدید فراخوانی شود، new.target یک مرجع به سازنده یا تابع برمی‌گرداند.

۲. برای فراخوانی تابع، new.target تعریف نشده است.

```

function Myfunc() {
    if (new.target) {
        console.log('called with new');
    } else {
        console.log('not called with new');
    }
}

new Myfunc(); // called with new
Myfunc(); // not called with new
Myfunc.call({}); not called with new

```

What are the differences between arguments object and rest parameter .۴۱۴

سه تفاوت اصلی بین پارامترهای شیء آرگومان و پارامترهای استراحت وجود دارد

۱. شیء آرگومان ها آرایه مانند است اما آرایه نیست. در حالی که بقیه پارامترها نمونه های آرایه هستند.

۲. شیء آرگومان ها از روش هایی مانند pop، sort، map، forEach یا پشتیبانی نمی کند. در حالی که این روش ها را می توان در پارامترهای استراحت استفاده کرد.

۳. بقیه پارامترها فقط آنهایی هستند که نام جداگانه ای به آنها داده نشده است، در حالی که شیء آرگومان ها شامل تمام آرگومان های ارسال شده به تابع است.

What are the differences between spread operator and rest parameter .۴۱۵

پارامتر Rest تمام عناصر باقی مانده را در یک آرایه جمع آوری می کند. در حالی که عملگر Spread به تکرار پذیرها (آرایه ها / اشیاء / رشته ها) اجازه می دهد تا به آرگومان ها / عناصر منفرد گسترش یابند. یعنی پارامتر Rest مخالف عملگر spread است.

What are the different kinds of generators .۴۱۶

پنج نوع تابع ژنراتور وجود دارد،

:Generator function declaration .۱

```
function* myGenFunc() {  
    yield 1;  
    yield 2;  
    yield 3;  
}  
const genObj = myGenFunc();
```

:Generator function expressions .۲

```
const myGenFunc = function* () {  
    yield 1;  
    yield 2;  
    yield 3;  
};  
const genObj = myGenFunc();
```

:Generator method definitions in object literals .۳

```
const myObj = {  
    * myGeneratorMethod() {  
        yield 1;  
        yield 2;  
        yield 3;  
    }  
};  
const genObj = myObj.myGeneratorMethod();
```

:Generator method definitions in class .۴

```
class MyClass {  
    * myGeneratorMethod() {  
        yield 1;  
        yield 2;  
        yield 3;  
    }  
}  
const myObject = new MyClass();  
const genObj = myObject.myGeneratorMethod();
```

:Generator as a computed property .۵

```

const SomeObj = {
  *[Symbol.iterator] () {
    yield 1;
    yield 2;
    yield 3;
  }
}

console.log(Array.from(SomeObj)); // [ 1, 2, 3 ]

```

What are the built-in iterables .۱۷

- در زیر لیستی از تکرارهای داخلی در جاوا اسکریپت آمده است.
۱. آرایه ها و TypedArrays
 ۲. رشته ها: روی هر کاراکتر یا نقاط کد یونیکد تکرار کنید
 ۳. نقشه ها: روی جفت های کلید-مقدار آن تکرار شود
 ۴. مجموعه ها: روی عناصر خود تکرار می شود
 ۵. آرگومان ها: یک متغیر خاص آرایه مانند در توابع NodeList DOM مانند
 ۶. مجموعه DOM

What are the differences between for...of and for...in statements .۱۸

هم برای...in و هم برای...az دستورات روی ساختارهای داده `js` تکرار می شن. تنها تفاوت در مورد چیزیه که او نا تکرار میکننه:

۱. `for..in` روی تمام کلیدهای خصوصیت شمارش پذیر یک شی تکرار می شود
۲. `for..of` بیش از مقادیر یک شی قابل تکرار.

بیاین این تفاوت رو توی یه مثال ببینیم،

```

let arr = ['a', 'b', 'c'];

arr.newProp = 'newValue';

// key are the property keys
for (let key in arr) {
  console.log(key);
}

// value are the property values
for (let value of arr) {
  console.log(value);
}

```

از اونجا که حلقه `for..in` روی کلیدهای شی تکرار می شه حلقه اول ۰، ۱، ۲ و `newProp` را در حین تکرار روی شی آرایه ثبت می کنه. حلقه `for..of` روی مقادیر یک ساختار داده `arr` تکرار می شه و a، b، c را در کنسول ثبت می کنه.

How do you define instance and non-instance properties .۴۱۹

خصوصیات Instance باید در داخل متدهای کلاس تعریف بشن. به عنوان مثال، مشخصات نام و سن سازنده داخلی هم مثل مثال پایین تعریف میشن.

```
class Person {  
    constructor(name, age) {  
        this.name = name;  
        this.age = age;  
    }  
}
```

اما خصوصیات داده Static(class) و نمونه اولیه باید خارج از اعلان ClassBody تعریف بشن. بیاین مقدار سن را برای کلاس Person به صورت زیر اختصاص بدیم.

```
Person.staticAge = 30;  
Person.prototype.prototypeAge = 40;
```

?What is the difference between isNaN and Number.isNaN .۴۲۰

۱. **isNaN**: تابع سراسری «isNaN» آرگومان را به عدد تبدیل میکند و اگر مقدار حاصل NaN باشد، true را برمیگرداند.

۲. **Number.isNaN**: این روش آرگومان را تبدیل نمی کند. اما زمانی که نوع یک عدد و مقدار NaN باشد مقدار true را برمی گرداند.
بیایید تفاوت را با یک مثال ببینیم،

```
isNaN('hello'); // true  
Number.isNaN('hello'); // false
```