

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 دوتایی (!! چیه؟
۷۰	هدف از عملگر delete چیه؟
۷۱	عملگر typeof چیکار می‌کنه؟
۷۲	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 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 switch 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 پيشفرض 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	٣٧٤
What is the shortcut to get timestamp	٣٧٥
How do you flattening multi dimensional arrays	٣٧٦
What is the easiest multi condition checking	٣٧٧
How do you capture browser back button	٣٧٨
How do you disable right click in the web page	٣٧٩
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:

۲. 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();
```

۳. 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 .۴

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

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

۵. 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");
```

۶. 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 نمونه‌ای 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);
```

(OR)

```
// 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 .۷

ویژگی‌های کلاس رو برای ایجاد آبجکت‌ها معرفی میکنه ES6

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

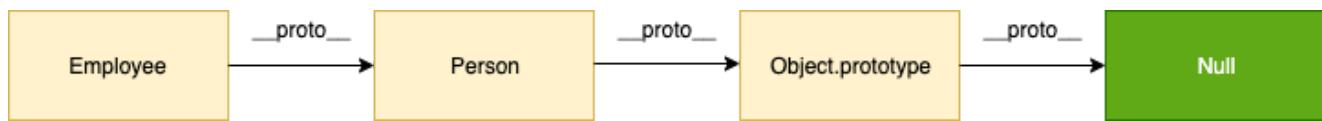
۸. الگوی Singleton

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";
}
```

۹. 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.



۱۰. 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 and apply are pretty interchangeable. Both execute the current function immediately. You need to decide whether it's easier to send in an array or a comma separated list of arguments. You can remember by treating Call is for comma (separated list) and Apply is for Array. Whereas Bind creates a new function .() that will have `this` set to the first parameter passed to bind

۱۱. فرمت JSON چیه و عملیات‌های 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 داده‌ها رو از طریق شبکه انتقال بدیم و اساساً یه فایل متنی با پسوند json و نوع application/json از تجزیه (Parsing): تبدیل یه رشته به یه آبجکت محلی (native Object)

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

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

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

۱۲. هدف از متدهای slice چیه؟

متدهای slice عناصر انتخاب شده توی یه آرایه رو به عنوان یه آبجکت آرایه جدید برمی‌گردونه. 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 متدهای 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: متدهای slice اصلی رو تغییر نمیده ولی یه زیرمجموعه به عنوان یه آرایه جدید برمی‌گردونه

۱۳. هدف از متدهای splice و آرایه‌ها چیه؟

متدهای **splice** برای اضافه کردن به آرایه یا حذف از آن استفاده می‌شوند. آرگومان اول موقعیت آرایه را برای درج یا حذف مشخص می‌کند که آرگومان اختیاری دوم تعداد عناصر حذف شده را مشخص می‌کند. هر آرگومان اضافه‌ای به آرایه اضافه می‌شود. 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** آرایه اصلی را اصلاح می‌کنند و آرایه حذف شده را برگردانند.

۱۴. تفاوت متدهای slice و splice چیا هستن؟

بعضی از تفاوت‌های عمده تویی به جدول

| Slice | Splice |

----- | ----- |

آرایه اصلی را تغییر نمیده (تغییرنایپذیر)	آرایه اصلی را تغییر میده (تغییر پذیر)
عناصر حذف شده را به عنوان آرایه برگردانه	عنوان آرایه برگردانه
برای انتخاب عناصر از آرایه استفاده می‌شود	برای درج عناصر به آرایه یا حذف از آن استفاده می‌شود

۱۵. تفاوت‌های Map و Object چیا هستن؟

آبجکت‌ها شبیه به نقشه‌ها (**Maps**) هستند 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 مهم وجود داره که استفاده از نقشه (**map**) رو توی موارد خاص ترجیح میده.

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

۱۷. کلیدهای نقشه (**map**) مرتب می‌شن در حالی که کلیدهای اضافه شده به آبجکت اینپطوری نیستند. بنابراین موقع تکرار روی اون، **object map** کلیدها رو به ترتیب اضافه شدنشون برگردانه.

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

۱۹. نقشه قابل تکراره و می‌توونه مستقیما تکرار بشه، در حالی که تکرار روی یه آبجکت مستلزم بدست آوردن کلیدهای اون به روشن خاص و تکرار روی اونهاست.

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

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

JavaScript provides both strict(=, !=) and type-converting(==, !=) equality comparison. The strict operators take type of variable in consideration, while non-strict operators make type correction/conversion based upon values of variables. The strict operators follow the below conditions for different types
Two strings are strictly equal when they have the same sequence of characters, same length, and same characters in corresponding positions.

. Two numbers are strictly equal when they are numerically equal. i.e, Having the same number value.
There are two special cases in this

. NaN is not equal to anything, including NaN.

. Positive and negative zeros are equal to one another.

. Two Boolean operands are strictly equal if both are true or both are false.

. Two objects are strictly equal if they refer to the same Object.

, Null and Undefined types are not equal with =, but equal with . i.e.
null=undefined --> false but nullundefined --> true

, Some of the example which covers the above cases

```
<"span dir="ltr" align="left">

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

<span/>

\*\*[فهرست] (#فهرست)\*\*

## ۹. توابع arrow-function یا lambda چی هستن؟

arrow function ها به صورت ساده‌تر و کوتاه‌تر تعریف می‌شون و **new.target** یا **this**, **arguments**, **super** ندارن. این توابع بدون متدهستند و به عنوان سازنده یا constructor استفاده نمی‌شون.

## ۱۰. یه تابع first-class چجور تابعیه؟

توبی جاوااسکریپت، 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، توبی 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 توبی کد زیر، توابع نگهدارنده یا **handler** به یه شنونده یا **listener** اختصاص داده شده.

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

```

[فهرست](#فهرست)

۱۱. 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!');
```

۱۲. 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);
```

۱۳. یه تابع unary چجور تابعیه؟

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

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

۱۴. توابع currying یعنی 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 ریاضی دان به اسم Haskell Curry گرفته شده. با استفاده از Currying در واقع 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 هستن.

۱۵. چه توابعی pure هستن؟

تابع خالص تابعیه که مقدار برگشتیش فقط توسط آرگومان‌هاش تعیین میشه 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 میشه. بیاین یه مثال از تفاوت بین تابع خالص و تابع 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 میاد. 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. Remember that Pure functions are important as they simplify unit testing without any side effects and no need for dependency injection. They also avoid tight coupling and make it harder to break your application by not having any side effects. These principles are coming together with **Immutability** concept of ES6 by giving preference to **const** over **let** usage

۱۶. هدف از کلمه کلیدی let چیه؟

دستور **let** یه **متغیر محلی block scope** تعریف میکنه. از این رو متغیرهایی که با کلمه کلیدی **let** تعریف میشن محدود به همون اسکوپی که تو ش تعریف شدن، دستورها و عبارت‌های توی همون اسکوپ میشن. 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 توشن تعریف شده، استفاده میشه. 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)
```

۱۷. 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 میشنه

:let

۱. به عنوان بخشی از ES6 معرفی شده

۲. محدود به scope یا دامنه هستش

۳. Hoist شده ولی مقداردهی اولیه نمیشه

بیاین با یه مثال تفاوتش رو بهتر بینیم

```
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)
}
```

۱۸. دلیل انتخاب کلمه کلیدی let چیه؟

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.

۱۹. چطوری می‌تونیم توی بلوک مربوط به switch بدون دریافت خطا متغیر تعریف کنیم؟

اگه بخوایم متغیرها رو مجددا توی یه switch block تعریف کنیم، این کار باعث خطا 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;
  }
}

```

۲۰. 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;
}

```

۲۱. (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  
}  
)  
();
```

دلیل اصلی استفاده از IIFE بdst آوردن حریم خصوصی داده‌هاست، چون محیط خارجی به متغیرهایی که توی IIFE تعریف شده دسترسی نداره. به عنوان مثال، اگه سعی کنیم با IIFE به متغیرها دسترسی پیدا کنیم این خط را می‌گیریم،

```
(function ()  
{  
    var message = "IIFE";  
    console.log(message);  
}  
)();  
console.log(message); //Error: message is not defined
```

۲۲. مزایای استفاده از module‌ها چیه؟

استفاده از مازول‌ها مزایای زیادی داره،

۱. قابلیت نگهداری
۲. قابلیت استفاده مجدد
۳. نامگذاری

۲۳. Memoization چیه؟

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 نوشتمن 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 starts with
      the number which is not a valid JavaScript identifier. Hence, can only be accessed using
      the square bracket notation.
    }
    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

```

۲۴. 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';

```

۲۵. چی هستن؟ ES6 Class ها توی

In ES6, Javascript classes are primarily syntactic sugar over JavaScript's existing prototype-based inheritance.

,For example, the prototype based inheritance written in function expression as below

```

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

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

```

Whereas ES6 classes can be defined as an alternative

```

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

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

```

.۲۶ 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 کرلی برآکت‌های 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.

.۲۷ 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.

۲۸. چرا به module‌ها نیاز داریم؟

- لیستی از مزایای استفاده از ماژول‌ها در اکوسیستم جاوااسکریپت اینجا گفته شده
- ۲۹. قابلیت نگهداری
- ۳۰. قابلیت استفاده مجدد
- ۳۱. نامگذاری

[فهرست] (#فهرست)

۲۹. 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.

۳۰. service-worker چیه؟

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 عمدۀ 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.

۳۱. توی service-worker چطوری میشه DOM رو دستکاری کرد؟

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

۳۲. 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 نمیتوونیم به state های سراسری توی نگهدارنده‌های onmessage و onfetch یه indexedDB API worker اعتماد کنیم. تو این حالت service worker برای تداوم کار و استفاده مجدد موقع شروع مجدد، به دسترسی خواهد داشت.

چیه؟ IndexedDB .۳۳

یه API سطح پایین برای ذخیره 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.

چیه؟ Web-storage .۳۴

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

.۳۵ : داده‌ها رو برای مبدا فعلی و بدون تاریخ انقضا ذخیره میکنه.

.۳۶ : داده‌ها رو برای یه جلسه ذخیره میکنه و با بسته شدن تب مرورگر داده‌ها از بین میرن.

** [فهرست] (#فهرست)

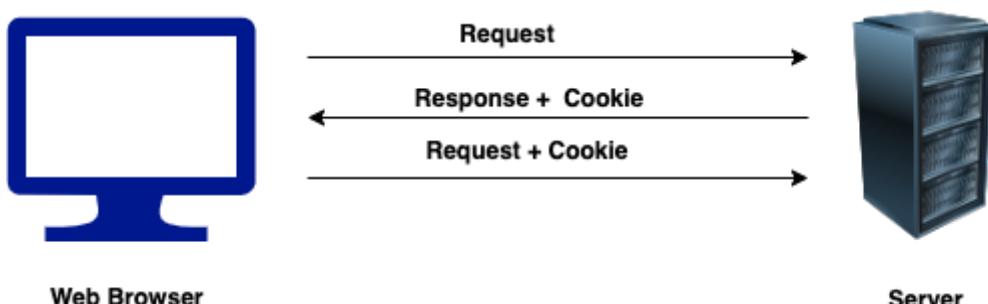
چیه؟ Post-message .۳۵

روشی 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 به دسترسی به همدیگه هستن، تنها در صورتی که صفحات از خط مشی یکسانی تعییت کنن. (یعنی صفحات از پروتکل، شماره پورت و میزبان یکسانی برخوردار هستن)

چیه؟ Cookie .۳۶

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 میتوونیم یه کوکی با نام کاربری مثل زیر ایجاد کنیم،

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



چرا به cookie نیاز داریم؟ .۳۷

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.

۳۸. 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";
```

۳۹. 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.

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

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

ویژگی | کوکی | لوکال استوریج | سشن استوریج

قابل دسترسی از طرف سرویس گیرنده یا سرور | هردو سرویس دهنده و سرویس گیرنده | فقط سرویس گیرنده | فقط سرویس گیرنده |

طول عمر | پیکربندی شده با استفاده از گزینه اکسپایر | تا زمانی که حذف بشه | تا زمانی که پنجره مرورگر بسته بشه
پشتیبانی از SSL | پشتیبانی میشه | پشتیبانی نمیشه | پشتیبانی نمیشه
حداکثر اندازه داده | ۴ کیلوبایت | ۵ مگابایت

۴۱. تفاوت های بین sessionStorage و localStorage چیا هستن؟

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

۴۲. 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) هستند رو پشتیبانی می‌کنند. این خصوصیات نمونه‌ای از شی Storage را ایجاد می‌کنند که از طریق اون می‌شوند 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');
```

۴۳. چه متدهایی روی session-storage قابل استفاده هستند؟

متدهایی روی خواندن، نوشتن و پاکسازی داده‌های session storage ارائه میدهند.

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

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

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

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

۴۴. رخداد storage چیه و چطوری ازش استفاده می‌کنیم؟

رویدادی هستش که با تغییر مکان ذخیره سازی در متن سند دیگه‌ای فعال می‌شود. 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 رو بینیم که کلید ذخیره و مقادیر اونو ثبت می‌کند

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

۴۵. چرا به web-storage نیاز داریم؟

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

۴۶. چطوری می‌تونیم پشتیبانی از web-storage توسط مرورگر رو بررسی کنیم؟

قبل از استفاده از فضای ذخیره‌سازی وب، باید پشتیبانی مرورگر رو برای sessionStorage و localStorage بررسی کنیم.

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

۴۷. چطوری می‌تونیم پشتیبانی از web-worker توسط مرورگر رو بررسی کنیم؟

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

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

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

برای شروع استفاده از web worker ها برای مثال شمارنده باید مراحل زیر رو دنبال کنیم

۱. ساخت یه فایل Web Worker: برای افزایش مقدار شمارش، باید یه اسکریپت بنویسیم. بیاین اسمشو counter.js بذاریم

```
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 میشه

۲. ایجاد شی Web Worker: با بررسی پشتیبانی مرورگر میتوانیم یه شی Web Worker ایجاد کنیم. بیاین اسم این فایل رو web_worker_example.js بذاریم.

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

و ما میتوانیم پیام‌ها را از web worker دریافت کنیم

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

۱. به پایان رساندن یه web Worker 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();
```

۲. استفاده مجدد از worker را : اگه متغیر undefined یا تعریف نشده تنظیم کنیم، میتوانیم از کد استفاده مجدد کنیم.

```
w = undefined;
```

۴۹. محدودیت‌های web-worker روی DOM چیا هستن؟

- ۱. WebWorker ها به اشیا جاوا اسکریپت دسترسی ندارن چون توی یه فایل خارجی تعریف شدن.
- ۲. Window object .۵۰
- ۳. Document object .۵۱
- ۴. Parent object .۵۲

** [فهرست] (#فهرست)

۵۰. چیه؟ Promise ؟

A promise is an object that may produce a single value some time in the future with either a resolved value or a reason that it's not resolved(for example, network error). It will be in one of the 3 possible states:
.fulfilled, rejected, or pending
,The syntax of Promise creation looks like below

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

,The usage of a promise would be as below

```

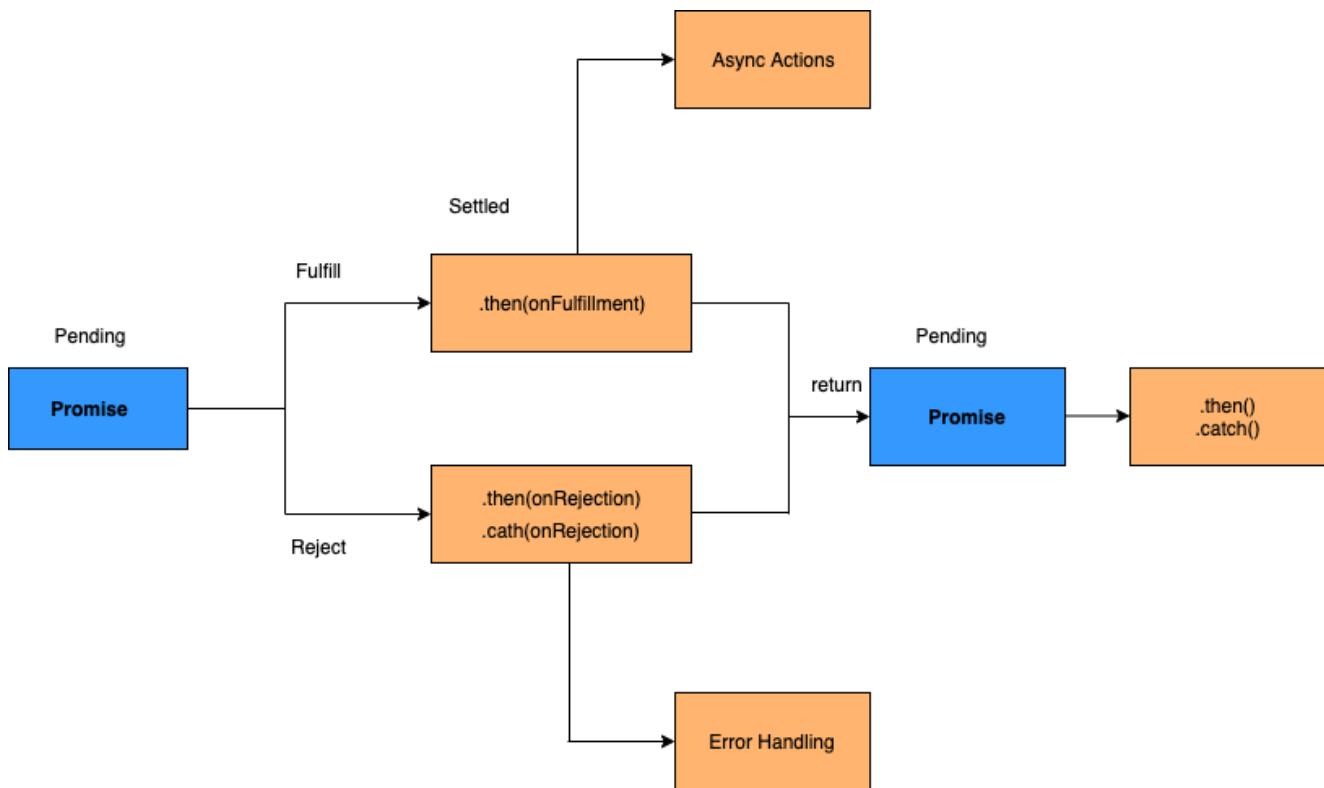
const promise = new Promise(resolve => {
  setTimeout(() => {
    resolve("I'm a Promise!");
  }, 5000);
}, reject => {

});

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

```

,The action flow of a promise will be as below



۵۱. چرا به promise نیاز داریم؟

Promises are used to handle asynchronous operations. They provide an alternative approach for callbacks by reducing the callback hell and writing the cleaner code

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

:Promises have three states

Pending: This is an initial state of the Promise before an operation begins .۵۳

Fulfilled: This state indicates that the specified operation was completed .۵۴

Rejected: This state indicates that the operation did not complete. In this case an error value will be .۵۵ thrown

۵۳. توابع callback چی هستن؟

A callback function is a function passed into another function as an argument. This function is invoked .inside the outer function to complete an action
Let's take a simple example of how to use callback function

```
function callbackFunction(name) {
  console.log('Hello ' + name);
}

function outerFunction(callback) {
  let name = prompt('Please enter your name.');
  callback(name);
}

outerFunction(callbackFunction);
```

۵۴. چرا به توابع callback نیاز داریم؟

The callbacks are needed because javascript is an event driven language. That means instead of waiting .for a response javascript will keep executing while listening for other events
Let's take an example with the first function invoking an API call(simulated by setTimeout) and the next .function which logs the message

```
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
```

As observed from the output, javascript didn't wait for the response of the first function and the remaining code block got executed. So callbacks are used in a way to make sure that certain code doesn't execute .until the other code finishes execution

۵۵.Callback-hell یا جهنم توابع callback چیه؟

Callback Hell is an anti-pattern with multiple nested callbacks which makes code hard to read and debug ,when dealing with asynchronous logic. The callback hell looks like below

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

۵۶. چیه؟ (Server-sent-events(SSE)

Server-sent events (SSE) is a server push technology enabling a browser to receive automatic updates from a server via HTTP connection without resorting to polling. These are a one way communications channel - events flow from server to client only. This has been used in Facebook/Twitter updates, stock .price updates, news feeds etc

۵۷. چطوری می‌تونیم اعلان‌های server-sent-event را دریافت کنیم؟

The EventSource object is used to receive server-sent event notifications. For example, you can receive ,messages from server as below

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

۵۸. چطوری می‌تونیم پشتیبانی مرورگر برای SSE را بررسی کنیم؟

,You can perform browser support for server-sent events before using it as below

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

۵۹. کدام توابع روی SSE وجود دارند؟

Below are the list of events available for server sent events

Description	Event
It is used when a connection to the server is opened	onopen
This event is used when a message is received	onmessage
It happens when an error occurs	onerror

۶۰. اصلی‌ترین قوانین **promise** ها چیا هستن؟

,A promise must follow a specific set of rules

۶۱ A promise is an object that supplies a standard-compliant .then() method

۶۲ A pending promise may transition into either fulfilled or rejected state

۶۳ .A fulfilled or rejected promise is settled and it must not transition into any other state

۶۴ .Once a promise is settled, the value must not change

فهرست [#]

۶۱. توى چطورى رخ مىدە؟ **Callback**

You can nest one callback inside in another callback to execute the actions sequentially one by one. This
.is known as callbacks in callbacks

```
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  
        });  
  
    })  
});
```

۶۲. زنجيره **promise** ها چيە؟

The process of executing a sequence of asynchronous tasks one after another using promises is known ,as Promise chaining. Let's take an example of promise chaining for calculating the final result

```

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;

});

```

,In the above handlers, the result is passed to the chain of .then() handlers with the below work flow

,The initial promise resolves in 1 second .۶۳

After that .then handler is called by logging the result(1) and then return a promise with the value of .۶۴

.result * 2

After that the value passed to the next .then handler by logging the result(2) and return a promise with .۶۵

.result * 3

Finally the value passed to the last .then handler by logging the result(6) and return a promise with .۶۶

.result * 4

[فهرست](#فهرست)

۶۳. کاربرد متدهای promise.all

Promise.all is a promise that takes an array of promises as an input (an iterable), and it gets resolved when all the promises get resolved or any one of them gets rejected. For example, the syntax of ,promise.all method is below

```

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

```

Note: Remember that the order of the promises(output the result) is maintained as per input order

۶۴. هدف از متدهای promise.race

Promise.race() method will return the promise instance which is firstly resolved or rejected. Let's take an example of race() method where promise2 is resolved first

```

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
});

```

٦٥. حالت strict توی جاواسکریپت چی کار میکنه؟

Strict Mode is a new feature in ECMAScript 5 that allows you to place a program, or a function, in a “strict” operating context. This way it prevents certain actions from being taken and throws more exceptions. The `.literal expression "use strict";` instructs the browser to use the javascript code in the Strict mode

٦٦. چرا به حالت strict نیاز داریم؟

Strict mode is useful to write “secure” JavaScript by notifying “bad syntax” into real errors. For example, it eliminates accidentally creating a global variable by throwing an error and also throws an error for assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing `.variable`, or a non-existing object

٦٧. چطوری میتوانیم حالت strict رو فعال کنیم؟

.The strict mode is declared by adding “use strict”; to the beginning of a script or a function
.If declared at the beginning of a script, it has global scope

```

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

```

and if you declare inside a function, it has local scope

```

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

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

```

۶۸. هدف از عملگر نقیض دو تایی (!!) چیه؟

The double exclamation or negation (!!) ensures the resulting type is a boolean. If it was falsey (e.g. 0, null, undefined, etc.), it will be false, otherwise, true
For example, you can test IE version using this expression as below

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

.If you don't use this expression then it returns the original value

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

Note: The expression !! is not an operator, but it is just twice of ! operator

۶۹. هدف از عملگر delete چیه؟

.The delete keyword is used to delete the property as well as its value

```
var user= {name: "John", age:20};  
delete user.age;  
  
console.log(user); // {name: "John"}
```

۷۰. عملگر typeof چیکار می کنه؟

You can use the JavaScript typeof operator to find the type of a JavaScript variable. It returns the type of a variable or an expression

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

۷۱. چیه و چه زمانی undefined می گیریم؟

The undefined property indicates that a variable has not been assigned a value, or not declared at all. The type of undefined value is undefined too

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

.Any variable can be emptied by setting the value to undefined

```
user = undefined
```

The value null represents the intentional absence of any object value. It is one of JavaScript's primitive values. The type of null value is object. You can empty the variable by setting the value to null

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

۷۳. تفاوت‌های بین null و undefined چیا هستن؟

,Below are the main differences between null and undefined
Null	Undefined

It is an assignment value which indicates that variable points to no object.	It is not an assignment value
.where a variable has been declared but has not yet been assigned a value	
Type of null is object Type of undefined is undefined	
The null value is a primitive value that represents the null, empty, or non-existent reference.	The
.undefined value is a primitive value used when a variable has not been assigned a value	
Indicates the absence of a value for a variable Indicates absence of variable itself	
Converted to zero (0) while performing primitive operations	Converted to NaN while performing
	primitive operations

۷۴. چیه؟ eval

The eval() function evaluates JavaScript code represented as a string. The string can be a JavaScript expression, variable, statement, or sequence of statements

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

۷۵. تفاوت‌های بین document و window چیا هستن؟

,Below are the main differences between window and document
Window	Document

It is the root level element in any web page	It is the direct child of the window object. This is also known
	(as Document Object Model(DOM
By default window object is available implicitly in the page	You can access it via window.document or
	.document

It has methods like alert(), confirm() and properties like document, location | It provides methods like | getElementById, getElementByTagName, createElement etc

۷۶. توی جاواسکریپت چطوری می‌تونیم به history دسترسی داشته باشیم؟

The window.history object contains the browser's history. You can load previous and next URLs in the .history using back() and next() methods

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

.**Note:** You can also access history without window prefix

۷۷. انواع داده‌های جاواسکریپت کدوما هستن؟

Below are the list of javascript data types available

- Number .۷۸
- String .۷۹
- Boolean .۸۰
- Object .۸۱
- Undefined .۸۲

[فهرست] (#فهرست)

۷۸. چیه و چیکار می‌کنه isNaN؟

The isNaN() function is used to determine whether a value is an illegal number (Not-a-Number) or not. i.e, .This function returns true if the value equates to NaN. Otherwise it returns false

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

۷۹. تفاوت‌های بین undefined و undeclared چیا هستن؟

,Below are the major differences between undeclared and undefined variables

undeclared	undefined
----- ---	

These variables do not exist in a program and are not declared | These variables declared in the program |
| but have not assigned any value

If you try to read the value of an undeclared variable, then a runtime error is encountered | If you try to |
| .read the value of an undefined variable, an undefined value is returned

۸۰. کدام متغیرها عمومی هستن؟

Global variables are those that are available throughout the length of the code without any scope. The var keyword is used to declare a local variable but if you omit it then it will become global variable

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

۸۱. مشکلات متغیرهای عمومی چیا هستن؟

The problem with global variables is the conflict of variable names of local and global scope. It is also difficult to debug and test the code that relies on global variables

۸۲. مقدار NaN چیه؟

The NaN property is a global property that represents "Not-a-Number" value. i.e, It indicates that a value is not a legal number. It is very rare to use NaN in a program but it can be used as return value for few cases

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

۸۳. هدف از تابع isFinite چیه؟

The isFinite() function is used to determine whether a number is a finite, legal number. It returns false if .the value is +infinity, -infinity, or NaN (Not-a-Number), otherwise it returns true

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

۸۴. یه event-flow چیه؟

Event flow is the order in which event is received on the web page. When you click an element that is nested in various other elements, before your click actually reaches its destination, or target element, it must trigger the click event for each of its parent elements first, starting at the top with the global window object

There are two ways of event flow
(Top to Bottom(Event Capturing .۸۵
(Bottom to Top (Event Bubbling .۸۶

[فهرست] (#فهرست)

چیه؟ Event-bubbling .۸۵

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

چیه؟ Event-capturing .۸۶

Event capturing is a type of event propagation where the event is first captured by the outermost element, and then successively triggers on the descendants (children) of the target element in the same nesting hierarchy till it reaches the innermost DOM element

چطوری میشه یه فرم رو با استفاده از جاوااسکریپت ثبت کرد؟ .۸۷

You can submit a form using JavaScript use document.form[0].submit(). All the form input's information is submitted using onsubmit event handler

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

چطوری میشه به اطلاعات مربوط به سیستم کاربر دسترسی داشت؟ .۸۸

The window.navigator object contains information about the visitor's browser OS details. Some of the OS properties are available under platform property

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

۸۹. تفاوت‌های بین رخدادهای DOMContentLoaded و document-load چیا هستن؟

The `DOMContentLoaded` event is fired when the initial HTML document has been completely loaded and parsed, without waiting for assets(stylesheets, images, and subframes) to finish loading. Whereas The `load` event is fired when the whole page has loaded, including all dependent resources(stylesheets, images).

۹۰. تفاوت‌های بین `user` و `native`, `host` چیا هستن؟

`Native objects` are objects that are part of the JavaScript language defined by the ECMAScript specification. For example, `String`, `Math`, `RegExp`, `Object`, `Function` etc core objects defined in the ECMAScript spec.

`Host objects` are objects provided by the browser or runtime environment (Node). For example, `window`, `XMLHttpRequest`, `DOM nodes` etc are considered as host objects.

`User objects` are objects defined in the javascript code. For example, User objects created for profile information.

۹۱. کدوم ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده میشند؟

You can use below tools or techniques for debugging javascript

Chrome Devtools .۹۲

debugger statement .۹۳

Good old console.log statement .۹۴

:[فهرست] (#فهرست)

۹۲. مزایا و معایب استفاده از `promise` به جای `callback` چیا هستن؟

,Below are the list of pros and cons of promises over callbacks

:Pros

It avoids callback hell which is unreadable .۹۳

()Easy to write sequential asynchronous code with `.then` .۹۴

()Easy to write parallel asynchronous code with `Promise.all` .۹۵

Solves some of the common problems of callbacks(call the callback too late, too early, many times and .۹۶
(swallow errors/exceptions

:Cons

It makes little complex code .۱

You need to load a polyfill if ES6 is not supported .۲

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

Attributes are defined on the HTML markup whereas properties are defined on the DOM. For example, the ,below HTML element has 2 attributes type and value

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

, You can retrieve the attribute value as below

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

And after you change the value of the text field to "Good evening", it becomes like

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

۹۴. سیاست same-origin چیه؟

The same-origin policy is a policy that prevents JavaScript from making requests across domain boundaries. An origin is defined as a combination of URI scheme, hostname, and port number. If you enable this policy then it prevents a malicious script on one page from obtaining access to sensitive data .(on another web page using Document Object Model(DOM

۹۵. هدف استفاده از void 0 چیه؟

Void(0) is used to prevent the page from refreshing. This will be helpful to eliminate the unwanted side-effect, because it will return the undefined primitive value. It is commonly used for HTML documents that use href="JavaScript:Void(0);" within an element. i.e, when you click a link, the browser loads a new .page or refreshes the same page. But this behavior will be prevented using this expression
For example, the below link notify the message without reloading the page

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

۹۶. جاواسکریپت یه زبان تفسیری هست یا کامپایلری؟

JavaScript is an interpreted language, not a compiled language. An interpreter in the browser reads over the JavaScript code, interprets each line, and runs it. Nowadays modern browsers use a technology

known as Just-In-Time (JIT) compilation, which compiles JavaScript to executable bytecode just as it is .about to run

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

Yes, JavaScript is a case sensitive language. The language keywords, variables, function & object names, .and any other identifiers must always be typed with a consistent capitalization of letters

۹۸. ارتباطی بین Java و JavaScript وجود داره؟

No, they are entirely two different programming languages and have nothing to do with each other. But both of them are Object Oriented Programming languages and like many other languages, they follow .(similar syntax for basic features(if, else, for, switch, break, continue etc

۹۹. Event چی هستن؟

Events are "things" that happen to HTML elements. When JavaScript is used in HTML pages, JavaScript ,can react on these events. Some of the examples of HTML events are
Web page has finished loading .۱
Input field was changed .۲
Button was clicked .۳

,Let's describe the behavior of click event for button element

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

۱۰۰. کی جاواسکریپت رو ساخته؟

جاوا اسکریپت توسط برندان ایچ در سال 1995 و در نت اسکریپ ارتباطات ایجاد شد. در ابتدا با نام Mocha توسعه یافت، اما بعداً زمانی که برای اولین بار در نسخه های بتا نت اسکریپ عرضه شد، این زبان به طور رسمی LiveScript نامیده

۱۰۱. هدف از متد preventDefault چیه؟

متد () preventDefault اگر رویداد قابل لغو باشد، آن را لغو می‌کند، به این معنی که عمل یا رفتار پیش‌فرض 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();
});
```

Note: Remember that not all events are cancelable

۱۰۲. کاربرد متد stopPropagation چیه؟

روش stopPropagation برای جلوگیری از حبابی شدن رویداد در زنجیره رویداد استفاده می‌شود. به عنوان مثال، div های تودرتو زیر با متد stopPropagation از انتشار پیش‌فرض رویداد هنگام کلیک بر روی (Div1) (Div1) جلوگیری می‌کنند.

```
<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>
```

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

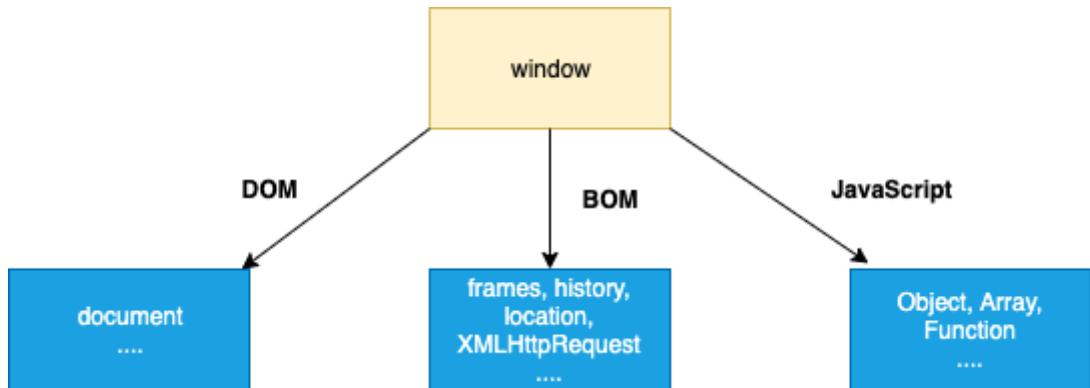
عبارت return false در کنترل کننده رویداد مراحل زیر را انجام می‌دهد:

۱۰۴. ابتدا عملکرد یا رفتار پیش‌فرض مرورگر را متوقف می‌کنند.

۱۰۵. این رویداد از انتشار DOM جلوگیری می‌کنند

۱۰۶. اجرای callback را متوقف می‌کنند و بلافاصله پس از فراخوانی برمی‌گردند.

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



۱۰۸. موارد استفاده از setTimeout کدوما هستن؟

متدهای setTimeout() برای فراخوانی یک تابع یا ارزیابی یک عبارت پس از تعداد مشخصی از میلی ثانیه استفاده میشند. 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);
```

۱۰۹. موارد استفاده از setInterval کدوما هستن؟

متدهای setInterval() برای فراخوانی یک تابع یا ارزیابی یک عبارت در بازه های زمانی مشخص (بر حسب میلی ثانیه) استفاده میشند. 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);
```

۱۱۰. چرا جاوا اسکریپت 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، ++java، go، C++ می توانند برنامه های چند رشته ای و چند فرآیندی بسازند.

۱۱۱. Event-delegation چیه؟

تفویض رویداد تکنیکی برای گوش دادن به رویدادهاست که در آن یک عنصر والد را به عنوان شنوونده 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);
```

۱۱۲. ECMAScript چیه؟

ECMAScript زبان برنامه نویسی است که اساس جاوا اسکریپت را تشکیل می دهد. توسط سازمان استاندارد بین المللی ECMA در مشخصات ECMA-262 و ECMA-402 استاندارد شده است. اولین نسخه ECMAScript در سال ۱۹۹۷ منتشر شد.

۱۱۳. JSON چیه؟

JSON (JavaScript Object Notation) یک فرمت سبک وزن است که برای تبادل داده ها استفاده می شه. این بر اساس زیرمجموعه ای از زبان جاوا اسکریپت است که اشیا در جاوا اسکریپت ساخته میشن.

۱۱۴. قوانین فرمت JSON کدوما هستن؟

- ۱۱۵. در زیر لیستی از قوانین نحوی JSON آمده است
- ۱۱۶. داده ها به صورت جفت نام/مقدار هستند
- ۱۱۷. داده ها با کاما از هم جدا میشن
- ۱۱۸. براکت ها اجسام را نگه می دارند
- ۱۱۹. براکت های مربعی آرایه ها را نگه می دارند

۱۱۹. هدف از متده JSON.stringify چیه؟

هنگام ارسال داده ها به وب سرور، داده ها باید در قالب رشته ای باشند. شما می توانید با تبدیل شی JSON` به رشته با استفاده از متدهای `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)
```

## ۱۱۳. چطوری می تونیم یه رشته JSON رو تجزیه کنیم؟

هنگام دریافت داده ها از یک وب سرور، داده ها همیشه در قالب رشته ای هستند. اما میتوانیم این مقدار رشته را با استفاده از متدهای `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}
```

## ۱۱۴. چرا به JSON نیاز داریم؟

هنگام تبادل داده بین مرورگر و سرور، داده ها فقط می توانند 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 برنامه نویسی استفاده کرد.

## ۱۱۵. PWA ها چی هستن؟

نوعی از برنامه های تلفن همراه هستند که از طریق وب ارائه میشون، و با استفاده از فناوری های رایج وب از جمله HTML، CSS و جاوا اسکریپت ساخته میشون. این PWA ها در سرورها مستقر میشون، از طریق URL ها قابل دسترسی هستند و توسط موتورهای جستجو فهرست بندی میشون.

## ۱۱۶. هدف از متدهای clearTimeout 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 تایмер 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 می توان 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 متد clearInterval چیه؟

تابع `clearInterval()` در جاوا اسکریپت برای پاک کردن فاصله‌ای که توسط تابع `setInterval()` تنظیم شده است استفاده می‌شود. به عنوان مثال، مقدار بازگشتی که توسط تابع `setInterval()` برمی‌گردد در یک متغیر ذخیره می‌شود و برای پاک کردن فاصله به تابع `clearInterval()` ارسال می‌شود. به عنوان مثال، از روش `setInterval()` زیر برای نمایش پیام در هر ۳ ثانیه استفاده می‌شود. 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. توی جاوااسکریپت، چطوری می‌شه به یه صفحه جدید redirect انجام داد؟

در `vanila` جاوا اسکریپت، می‌توانی با استفاده از ویژگی `location` شی پنجره به صفحه جدیدی هدایت بشین.

```

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

```

## ۱۱۹. چطوری بررسی می‌کنیم که یه string شامل یه substring هست یا نه؟

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

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

۱۲۱. استفاده از متده 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 ۱ نباشد، به این معنی است که رشته فرعی در رشته اصلی وجود دارد.

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

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

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

## ۱۲۳. توی جاوااسکریپت، چطوری مقدار یه آدرس email رو اعتبارسنجی می‌کنیم؟

می‌توانید با استفاده از Regex ایمیل را در جاوا اسکریپت تأیید کنید. توصیه می‌شود به جای سمت کلاینت، اعتبارسنجی در سمت سرور انجام شود. زیرا جاوا اسکریپت را می‌توان 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 کاراکترهای یونیکد را می‌پذیرد.

## ۱۲۴. چطوری می‌توانیم مقدار آدرس url جاری رو بخونیم؟

می‌توانید از عبارت window.location.href برای دریافت مسیر آدرس فعلی استفاده کنید و می‌توانید از همان عبارت برای بهروزرسانی URL نیز استفاده کنید. همچنین می‌توانید از document.URL برای اهداف فقط خواندنی استفاده کنید، اما این راه حل مشکلاتی در FF دارد.

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

## ۱۲۵. ویژگی‌های مختلف url روی object history مربوط به کدوما هستن؟

- برای دسترسی به اجزای URL صفحه می‌توان از ویژگی‌های شی location زیر استفاده کرد.
- ۱۲۶. URL - href
  - ۱۲۷. URL - protocol
  - ۱۲۸. URL - host
  - ۱۲۹. URL - hostname
  - ۱۳۰. URL - port
  - ۱۳۱. URL - pathname
  - ۱۳۲. URL - search
  - ۱۳۳. URL - hash

## ۱۳۴. توی جاوااسکریپت چطوری می‌تونیم مقدار یه query-string رو بخونیم؟

می‌توانید از URLSearchParams برای دریافت مقادیر رشته پرس و جو در جاوا اسکریپت استفاده کنید. بیایید مثالی برای دریافت مقدار کد مشتری از رشته پرس و جو URL بینیم،

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

## ۱۳۵. چطوری می‌تونیم بررسی کنیم که آیا یه پرایپرتوی 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.

۱۳۶. 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)
```

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

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

۱۳۸. 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

```

## How do you loop through or enumerate javascript object .۱۳۹

می توانید از حلقه `for-in` برای حلقه زدن شی جاوا اسکریپت استفاده کنید. همچنین می توانید مطمئن شوید که کلیدی که دریافت می کنید 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 ...
 }
}

```

## How do you test for an empty object .۱۴۰

راه حل های مختلفی بر اساس نسخه های ECMAScript وجود دارد  
:(+Object entries(ECMA 7 .۱۴۱ استفاده

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

```

:(+Object keys(ECMA 5 .۱۴۲ استفاده از

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

```

. استفاده از `for-in` با متدهای `hasOwnProperty`(Pre-ECMA 5 .۱۴۳ استفاده از حلقه `for-in` میتوانیم از `hasOwnProperty` و هر پارامتر رو با چک کنیم

```

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

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

```

## What is an arguments object .۱۴۴

شیء آرگومان ها یک شیء آرایه مانند که در داخل توابع قابل دسترسیه و حاوی مقادیر آرگومان های ارسال شده به اون تابعه. به عنوان مثال، بباید بینیم 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

```

**Note:** شما نمیتوانین از متدهای ارایه برای این شیء آرگومان ها استفاده کنین اما میتوانین به ارایه تبدیلش کنین

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

## How do you make first letter of the string in an uppercase .۱۴۵

میتوانیم با درست کردن یه فانکشن 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 .۱۴۶

حلقه `for` یک سینتکس تکراری رایج در جاوا اسکریپته که هم مزايا و هم معایب داره  
مزايا #####

.۱۴۷ توی همهی محیط ها env کار میکنه

.۱۴۸ میتوانیم از `break` و `continue` برای کنسل جریان داده استفاده کرد

## How do you display the current date in javascript .۱۵۲

شما میتوانید از کلاس new Date() استفاده کنید که یه اجکت 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 .۱۵۳

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 .۱۵۴

ما میتوانید از متده startsWith() که بر روی پرتوتاپ 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 موردش

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

## How do you trim a string in javascript .۱۵۵

جاوا اسکریپت یه متده 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 . ۱۵۶

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

```

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

```

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

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

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

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

## Is the !-- notation represents a special operator . ۱۵۹

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

۱۶۰ . اپراتور نقیض (!)

۱۶۱ . 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 . ۱۶۲

میتونیم 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 undefined یا false باشد

## How do you define multiline strings . ۱۶۳

ما میتوانیم از / برای تعریف 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 نوشتاری کد قراره داشته باشیم

## What is an app shell model . ۱۶۴

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 . ۱۶۵

بله ما میتوانیم برای توابع پرداختی 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 . ۱۶۶

با استفاده کردن از `function.length` ما میتوانیم به تعداد پارامتر هایی که یه تابع انتظار داره بگیره دسترسی داشته باشیم  
بریم 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 . ۱۶۷

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

## What are break and continue statements . ۱۶۸

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 . ۱۶۹

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 . ۱۷۰

- توصیه می شه تمام اعلان ها رو بالای هر اسکریپت یا تابع نگه دارین. مزیت این کار
- ۱۷۱. کد ما تمیز تر میشه
  - ۱۷۲. این یک مکان واحد برای جستجوی متغیرهای محلی فراهم می کنه
  - ۱۷۳. میشه راحت از استفاده متغیر های ناخواسته جلوگیری کرد
  - ۱۷۴. این کار محاسبات ناخواسته رو کمتر میکنه

## What are the benefits of initializing variables . ۱۷۵

- توضیه میشه که مقدار اولیه برای متغیرها تعیین بشه که دلایلشو چک میکنیم
- ۱۷۶. خروجیمون کد تمیز تری میشه
  - ۱۷۷. این کار باعث میشه یه جا برای این متغیر رزرو بشه
  - ۱۷۸. از برگشتن خطای undefined جلوگیری میشه

## What are the recommendations to create new object . ۱۷۹

- برای ساخت یه object با مقادیر پیشفرض مثال های زیر روش های ساخت مقادیر پیشفرض رو بررسی میکنیم
- ۱۸۰. استفاده از {} به جای ()new Object
  - ۱۸۱. استفاده از "" به جای ()new String
  - ۱۸۲. استفاده از 0 به جای ()new Number
  - ۱۸۳. استفاده از false به جای ()new Boolean
  - ۱۸۴. استفاده از [] به جای ()new Array
  - ۱۸۵. استفاده از /() / به جای ()new RegExp
  - ۱۸۶. استفاده از (){} به جای ()new Function
- بریم یه چن تا مثال ببینیم

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

## How do you define JSON arrays . ۱۸۷

آرایه های JSON نوشته شده در داخل برآکت ها و ارایه هایی که دارای object هستند بریم یه مثال درموردش ببینیم

```

"users": [
 {"firstName": "John", "lastName": "Abrahm"},
 {"firstName": "Anna", "lastName": "Smith"},
 {"firstName": "Shane", "lastName": "Warn"}
]

```

## How do you generate random integers .۱۸۸

ما میتوانیم از متدهای Math.floor() برای ساخت یک عدد رندوم بین ۰ تا یک و از متدهای 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

```

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

## Can you write a random integers function to print integers with in a range .۱۸۹

بله ما میتوانیم این تابع رو داشته باشیم که مقادیر حداقل و حداقل رو بگیره و برای ما عدد رندوم ایجاد کنه بریم ببینیم 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 .۱۹۰

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 متنکی است (یعنی واردات و صادرات). در ابتدا این باندل مازول 'rollup ES2015' رایج شد.

## What is the need of tree shaking .۱۹۱

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

## Is it recommended to use eval . ۱۹۲

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

## What is a Regular Expression . ۱۹۳

regular expression یا همون Regex یه توالیه که یه ساختار جستجو ایجاد میکنه با استفاده از این ساختاز ما میتوانیم دیتامون رو بر اساس یه ساختار که مینویسم جستجو کنیم و به قولی دیتامون رو اعتبارسنجی کنیم /pattern/modifiers;

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

## What are the string methods available in Regular expression . ۱۹۴

دو تا متده برای رشتةها داره : search() و replace(). متده برای جستجو اونو جسنجو میکنه و محل اون عبارت رو برمیگردونه

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

متده replace() برای برگرداندن رشتة اصلاح شده در جایی که الگو جایگزین میشه استفاده میشه

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

## What are modifiers in regular expression . ۱۹۵

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

| Modifier | Description                                             |
|----------|---------------------------------------------------------|
| i        | Perform case-insensitive matching                       |
| g        | Perform a global match rather than stops at first match |
| m        | Perform multiline matching                              |
|          | 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 .۱۹۶

- Regex یه گروهی از ساختارها برای این ها استفاده میشون اما ممکن است کاراکترها را چک کنیم اونا تو سه مدل طبقه بندی میشون
- ۱۹۷. **براكت ها:** اینها برای مثال پایین چن تا مورد استفاده میشون تا اینکه از کاراکتر را پیدا کنند
  - ۱۹۸. [abc]: استفاده میشون تا هر کاراکتری بین این سه کاراکتر پیدا بشون
  - ۱۹۹. [۰-۹]: استفاده میشون تا ارقام بین این دو عدد پیدا بشون
  - ۲۰۰. (a|b): برای پیدا کردن هر یک از گزینه های جدا شده با | استفاده میشون
  - ۲۰۱. **كاراكتر برابر با:** این عبارتها کاراکترهایی با معنی خاص هستند
  - ۲۰۲. برای مثال پایین دو تا مورد که استفاده میشون را ببینید
  - ۲۰۳. \d: استفاده برای پیدا کردن اعداد
  - ۲۰۴. \s: استفاده برای پیدا کردن فاصله ها
  - ۲۰۵. \b: استفاده برای پیدا کردن کاراکترهای همخوانی داشته با شروع شدن یا پایانشون
  - ۲۰۶. **كميت گتنده ها:** اینها برای تعریف کمیت ها موثر هستند
  - ۲۰۷. برای مثال پایین دو تا مورد استفاده برآشون اوردیم
  - ۲۰۸. +n: برای پیدا کردن رشته همخوانی داشته با حداقل یک کاراکتر
  - ۲۰۹. \*n: برای پیدا کردن همخوانی هر رشته شامل صفر یا بیشتر
  - ۲۱۰. ?n: برای پیدا کردن هر رشته که شامل صفر یا یک کاراکتر میشون

## What is a RegExp object .۲۰۹

های object های Regex یه عبارت معمولی با پرایپری 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 .۲۱۰

میتوانید از متدهای 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 .۲۱۱

هدف متدهای exec شبهی به روش تست 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 .۲۱۲

شما می توانید سبک درون خطی یا نام کلاس یک عنصر HTML را با استفاده از جاوا اسکریپت تغییر دهید  
۲۱۳. استفاده از پرایپری استایل: با استفاده از ویژگی style می توانید استایل درون خطی را تغییر دهید

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

۲۱۴. استفاده از پرایپری className: تغییر کلاس عنصر با استفاده از ویژگی آسان است

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

## 'What would be the result of 1+2+'3 .۲۱۵

خروجی '3' 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 .۲۱۶

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 .۲۱۷

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.

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 .۲۱۹

ما می توانیم با استفاده از Regex که یک boolean به ما برمیگردانه بفهمیم که مرورگری 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)|amoi|an(ex|ny|yw)|aptu|ar(ch|go)|as(te|us)|ati
 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\-*|g\-
 mo|go(\.w\od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-*|hi(pt|ta)|hp(i|ip)|hs\-*|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)|pdwg|pg(13)\-([1-
 8]|c))|phil|pire|pl(ay|uc)|pn\-*|po(ck|rt|se)|prox|psio|pt\-*|qa\-*|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 .۲۲۰

می توانیم مرورگرهای تلفن همراه رو با اجرای فهرستی از دستگاهها و بررسی اینکه آیا userAgent با چیزی مطابقت دارد یا خیر، شناسایی کنید. 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 .۲۲۱

ما می‌توانیم با استفاده از جاوا اسکریپت می‌توانید به صورت برنامه ریزی 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 .۲۲۲

مرورگرها یک شی XMLHttpRequest ارائه می‌دان که می‌توانه برای ایجاد درخواست های 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 .۲۲۳

مرورگرها یک شی XMLHttpRequest را ارائه می‌دان که می‌توان برای درخواست های HTTP ناهمzman 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 .۲۲۴

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

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

## What are the properties used to get size of window .۲۲۵

می‌توانیم از ویژگی‌های `innerWidth`, `innerHeight`, `clientWidth`, `clientHeight` ویندوز، عنصر سند و 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 .۲۶

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 .۲۲۷

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 .۲۲۸

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

:window.onload .۲۲۹

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

:document.onload .۲۳۰

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

:body onload .۲۳۱

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

## What is the difference between proto and prototype .۲۳۲

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 .۲۳۳

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 .۲۳۴

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 .۲۳۵

مزیت استفاده کردن از متده **freeze()** رو ببینیم  
 ۲۳۶. برای فریز کردن ابجکت ها و آرایه ها  
 ۲۳۷. برای **immutable** کردن ابجکت ها

## Why do I need to use freeze method .۲۳۸

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 .۲۳۹

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 .۲۴۰

این کار باعث میشه که حرف اول یه رشته به صورت بزرگ(زبان انگلیسی) نشون داده بشه که ما میتوانیم با فانکشن 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 .۲۴۱

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 .۲۴۲

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

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

. ۲۴۴ . عملگر های مقایسه ای: شامل = (برابر), != (غیر برابر), == (برابر و تایپ برابر), < (بزرگتر), > (کوچکتر), >= (کوچکتر مساوی)

. ۲۴۵ . عملگر های منطقی: شامل && ("و" منطقی), || ("یا" منطقی), ! ("منطقی نه")

. ۲۴۶ . 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)

. ۲۴۷ . اپراتور سه تایی: شامل اپراتور های شرطی سه تایی

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

## What is a rest parameter .۲۴۹

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، بیایید 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
```

**Note:** Rest parameter is added in ES2015 or ES6

## What happens if you do not use rest parameter as a last argument .۲۵۰

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 .۲۵۱

- در زیر لیستی از عملگرهای منطقی بیتی مورد استفاده در جاوا اسکریپت آمده است
- ۲۵۲. به صورت بیتی AND (&)
  - ۲۵۳. به صورت بیتی OR (|)
  - ۲۵۴. به صورت بیتی XOR (^)
  - ۲۵۵. به صورت بیتی NOT (~)
  - ۲۵۶. تغییر مکان به چپ (">>)
  - ۲۵۷. علامت در حال انتشار به سمت راست (<>)
  - ۲۵۸. صفر پر کردن Shift راست (<<>)

## What is a spread operator .۲۵۹

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 .۲۶۰

- متدهای 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.
- ۲۶۱. اگر قابل توسعه نباشد.
  - ۲۶۲. اگر تمام خصوصیات آن غیر قابل تنظیم باشند.
  - ۲۶۳. 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 .۲۶۴

متد `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:  
۲۶۵. هردو undefined

۲۶۶. هردو null

۲۶۷. هردو true یا هر دو false

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

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

۲۷۰. هر دو عدد و

۰+ هر دو

۰- هر دو

NaN هر دو

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 .۲۷۱

- برخی از کاربردهای متد «Object.is» به شرح زیر است،  
۲۷۲. برای مقایسه دو رشته استفاده می شود.
- ۲۷۳. برای مقایسه دو عدد استفاده می شود.
- ۲۷۴. برای مقایسه قطبیت دو عدد استفاده می شود.
- ۲۷۵. برای مقایسه دو شی استفاده می شود.

## How do you copy properties from one object to other .۲۷۶

می توانید از متد `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. نحو به 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 .۲۷۷

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

۱. برای شبیه سازی یک شی استفاده می شود.
۲. برای ادغام اشیاء با ویژگی های یکسان استفاده می شود.

## What is a proxy object .۳

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

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

بیایید مثالی از شیء پروکسی بزنیم،

```
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
```

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

## What is the purpose of seal method .۴

روش **Object.seal()** برای مهر و موم کردن یک شی، با جلوگیری از اضافه شدن ویژگی‌های جدید به آن و علامت گذاری تمام ویژگی‌های موجود به عنوان غیر قابل تنظیم، استفاده می‌شود. اما مقادیر ویژگی‌های فعلی تا زمانی که قابل نوشتمن باشند همچنان قابل تغییر هستند. باید 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 .۵

- ۱. در زیر کاربردهای اصلی متدهای **Object.seal()** آورده شده است.
- ۲. برای آب بندی اشیا و آرایه‌ها استفاده می‌شود.
- ۳. برای غیرقابل تغییر کردن یک جسم استفاده می‌شود.

## What are the differences between freeze and seal methods .۶

اگر یک شی با استفاده از متدهای **Object.freeze()** منجمد شود، ویژگی‌های آن تغییرناپذیر می‌شوند و هیچ تغییری در آنها نمی‌توان ایجاد کرد، در حالی که اگر یک شی با استفاده از متدهای **Object.seal()** مهر و موم شده باشد، می‌توان تغییرات را در ویژگی‌های موجود ایجاد کرد. از شی

## How do you determine if an object is sealed or not .۷

- ۸. متدهای **Object.isFrozen()** و **Object.isSealed()** برای تعیین مهر و موم بودن یا نبودن یک شی استفاده می‌شود. اگر همه شرایط زیر درست باشد یک شی مهر و موم می‌شود.
  - ۹. اگر قابل توسعه نباشد.
  - ۱۰. اگر تمام خصوصیات آن غیر قابل تنظیم باشند.
  - ۱۱. اگر قابل جابجایی نباشد (اما لزوماً غیرقابل نوشتمن نیست).
  - ۱۲. باید 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 .۱۳

متده 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 متده 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

رفتار متده Object.entries() مشابه روش 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

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 اشیاء در WeakSet ضعیف است. به عنوان مثال، یک شی در WeakSet می تواند زباله جمع آوری شود اگر مرجع دیگری به آن وجود نداشته باشد. تفاوت های دیگر عبارتند از

- مجموعه ها می توانند هر مقداری را ذخیره کنند در حالی که WeakSets می توانند تنها مجموعه ای از اشیاء را ذخیره کند

- برخلاف Set دارای ویژگی اندازه نیست WeakSet

- WeakSet متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی ها، forEach را ندارد. قابل تکرار نیست. WeakSet.4

## List down the collection of methods available on WeakSet

- در زیر لیستی از روش های موجود در WeakSet آمده است،

- (add(value): یک شی جدید با مقدار داده شده به مجموعه ضعیف اضافه می شود

- (delete(value): مقدار را از مجموعه WeakSet حذف می کند.

- (has(value): اگر مقدار در مجموعه WeakSet وجود داشته باشد true را برمی گرداند، در غیر این صورت false را برمی گرداند.

- ()length: طول ضعیف SetObject را برمی گرداند بیایید عملکرد تمام روش های 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 مجموعه ای از جفت های کلید/مقدار است که در آن کلیدها به صورت ضعیف ارجاع داده شده اند. در این حالت، کلیدها باید اشیا باشند و مقادیر می توانند مقادیر دلخواه باشند. نحوه شکل زیر است،

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

باید 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، می تواند زباله جمع آوری شود.

تفاوت های دیگر عبارتند از  
1. نقشه ها می توانند هر نوع کلیدی را ذخیره کنند، در حالی که WeakMaps فقط می توانند مجموعه ای از اشیاء کلیدی را ذخیره کند

برخلاف Map دارای ویژگی اندازه نیست

متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی ها، forEach را ندارد.

قابل تکرار نیست.

## List down the collection of methods available on WeakMap

در زیر لیستی از روش های موجود در WeakMap آمده است،

: مقدار کلید را در شی WeakMap تنظیم می کند. شی WeakMap را برمی گرداند.

: هر مقدار مربوط به کلید را حذف می کند.

: یک Boolean را برمی گرداند که نشان می دهد آیا مقداری به کلید در شی WeakMap مرتبط شده است یا خیر.

(key) get() : مقدار مربوط به کلید را برمی‌گرداند، یا اگر کلیدی وجود نداشته باشد، تعریف نشده است.  
بیایید عملکرد تمام روش‌های بالا را در یک مثال بینیم،

```
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

) یک تابع داخلی است که برای ایجاد نمایش رشته‌ای از کد منبع یک شی استفاده می‌شود. این یک تابع سطح بالا است و با هیچ شیئی مرتبط نیست. بیایید مثال زیر را بینیم 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 می‌شود 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 قبلاً توسط encodeURI ایجاد شده است استفاده می‌شود.

```

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 متد print() ارائه می‌کند که برای چاپ محتویات پنجره فعلی استفاده می‌شود. یک کادر محاوره ای چاپ را باز می‌کند که به شما امکان می‌دهد بین گزینه‌های 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 می‌شود.

## What is the difference between uneval and eval

تابع 'uneval' منبع یک شی معین را برمی‌گرداند. در حالی که تابع "eval" با ارزیابی آن کد منبع در یک ناحیه حافظه متفاوت، برعکس عمل می‌کند. بباید 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 می‌شوند یا به عنوان یک تابع فراخوانی استفاده می‌شوند. نحو به 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
});

```

بیایید تابع ناشناس بالا را در یک مثال بینیم،

```

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. بیایید این رفتار را در یک مثال بینیم.

```

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

```

•

## What are javascript accessors

ECMAScript 5 دسترسی‌های شی جاوا اسکریپت یا ویژگی‌های محاسبه‌شده را از طریق گیرنده‌ها و تنظیم‌کننده‌ها معرفی کرد. از Getters 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 می شود و شی را برمی گرداند. 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.

- آنها نحو ساده تری ارائه می دهند
- آنها برای تعریف ویژگی های محاسبه شده یا دسترسی ها در JS استفاده می شوند.
- برای ارائه رابطه هم ارزی بین خواص و روش ها مفید است
- آنها می توانند کیفیت داده های بهتری را ارائه دهند
- 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 Getters و Setter 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 if-else از دستور case 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،

- عبارت می تواند از نوع عددی یا رشته ای باشد.
- مقادیر تکراری برای عبارت مجاز نیستند.
- 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.
- دستور break در داخل سوئیچ برای پایان دادن به دنباله دستور استفاده می شود.
- 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). 5 نوع نوع داده اولیه وجود دارد.



اعداد



boolean



null



undefined



## What are the different ways to access object properties

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

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

`objectName.property`

۲. از براکت های مربع برای دسترسی به دیتا استفاده می کند

`objectName["property"]`

۳. از عبارت در کروشه استفاده می کند

`[objectName[expression]`

## What are the function parameter rules.

توابع جawa 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،

```
function functionName(parameter1, parameter2, parameter3) {
 console.log(parameter1); // 1
}
functionName(1);
```

## What is an error object.<sup>۲۲۵</sup>

یک شیء خطای 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،

```


javascript````
} try
;("greeting("Welcome
{
} (catch(err
;(console.log(err.name + "
" + err.message
{
```  
  
<span/>  
  
**[فهرست] (#فهرست)
```

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}

6 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 به دلیل encodeURI() | خطای URIE |

What are the various statements in error handling .۲۲۹

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

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

What are the two types of loops in javascript .۵

۶. **For**: در این نوع حلقه، شرایط تست قبل از ورود به بدنه حلقه آزمایش می شود. به عنوان مثال `for` در این دسته قرار می گیرند.

۷. **Exit Controlled Loops**: 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 .۸

`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 I/O ناهمزمان مبتنی بر رویداد، غیر مسدود کننده است که از موتور جاوا اسکریپت V8 گوگل و کتابخانه `libuv` استفاده می کند.

[فهرست] (#فهرست)

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 ببینیم،

```
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
```

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

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()` و `done()` یکی برمی گرداند.
- **IteratorResult**: این یک شی است که با متدهای `value` و `done` تعریف شده است و 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،

```
<span dir="ltr" align="left">
```

```
javascript```
;['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
```

```

```

```

\*\*[فهرست](#فهرست)\*\*

## 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. **Note**: این به Node.js اجازه می دهد تا عملیات `I/O` غیر مسدود کننده را انجام دهد حتی اگر جاوا اسکریپت تک رشته ای باشد.

## What is call stack .۲۳۷

یک ساختار داده برای مفسران جاوا اسکریپت است تا فراخوانی های تابع در برنامه را پیگیری کند. دو عمل عمدی دارد، هر زمان که یک تابع را برای اجرای آن فراخوانی می کنید، آن را به پشته هل می دهید.

.۲۳۹ 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
```

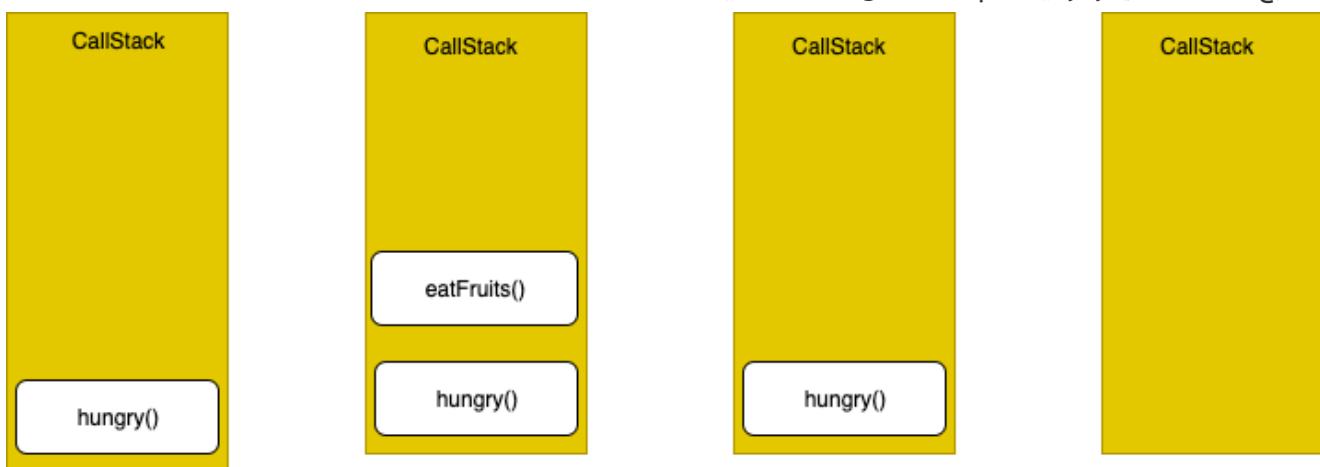
کد بالا در یک پشته تماس به صورت زیر پردازش می شود.

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

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

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

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



## What is an event queue .۲۴۲

## What is a decorator .۲۴۳

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 .۲۴۴

در زیر لیستی از ویژگی های موجود در شیء Intl آمده است،

۱. اینها اشیایی هستند که مقایسه رشته های حساس به زبان را امکان پذیر می کنند. **Collator**.
۲. اینها اشیایی هستند که قالب بندی تاریخ و زمان حساس به زبان را فعال می کنند. **DateTimeFormat**.
۳. اینها اشیایی هستند که قالب بندی لیست حساس به زبان را فعال می کنند. **ListFormat\*\***.
۴. اشیایی که قالب بندی اعداد حساس به زبان را فعال می کنند. **NumberFormat**.
۵. اشیایی که قالب بندی حساس به جمع و قوانین خاص زبان را برای جمع فعال می کنند. **PluralRules\*\***.
۶. اشیایی که قالب بندی زمان نسبی حساس به زبان را فعال می کنند. **RelativeTimeFormat**.

## What is an Unary operator .۷

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 .۸

متدهای sort() برای مرتب سازی عناصر یک آرایه در جای خود استفاده می‌شود و آرایه مرتب شده را برمی‌گرداند. 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 .۹

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

```
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 .۱۰

برای معکوس کردن عناصر یک آرایه می‌توانید از متدهای reverse() استفاده کنید. این روش برای مرتب کردن یک آرایه به ترتیب نزولی مفید است. بیایید استفاده از متدهای reverse() را در یک مثال ببینیم،

```
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 .۱۱

می‌توانید از روش‌های آرایه برای یافتن حداقل و حداکثر 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(...arr);
}
function findMax(arr) {
 return Math.max(...arr);
}

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

## How do you find min and max values without Math functions .۱۲

دستور خالی یک نقطه ویرگول (逗號) 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 .۱۳

می توانید از شی «import.meta» استفاده کنید که متأ در این داده های متنی خاص را در یک مازول جاوا اسکریپت قرار می دهد. این شامل اطلاعاتی در مورد مازول فعلی، مانند URL مازول است. در مرورگرهای ممکن است متأ داده های 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 .۱۴

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 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 .۱۵

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 .۱۶

معمولأ 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 .۱۷

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

```

متد خوشنامدگویی فقط نوع رشته را به عنوان آرگومان مجاز می کند.

## What are the differences between javascript and typescript .۱۸

در زیر لیستی از تفاوت های بین جاوا اسکریپت و تایپ اسکریپت آمده است.  
ویژگی	تایپ	جاوا اسکریپت
پارادایم زبان	زبان برنامه نویسی شی گرا	زبان اسکریپت
پشتیبانی از تایپ	پشتیبانی از تایپ استاتیک	دارای تایپ پویا
ماژول ها	پشتیبانی شده	پشتیبانی نمی شود
رابط	دارای مفهوم رابط	از رابط ها پشتیبانی نمی کند
پارامترهای اختیاری	توابع از پارامترهای اختیاری پشتیبانی از پارامترهای اختیاری برای توابع	

## What are the advantages of typescript over javascript .۱۹

در زیر برخی از مزایای تایپ اسکریپت نسبت به جاوا اسکریپت آورده شده است.  
 ۲۰. 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.  
 ۲۱. TypeScript به شدت تایپ می شود یا از تایپ پشتیبانی می کند که امکان بررسی صحت نوع را در زمان کامپایل فراهم می کند. این در جاوا اسکریپت در دسترس نیست.  
 ۲۲. کامپایلر TypeScript برخلاف ویژگی های ES6 جاوا اسکریپت که ممکن است در برخی از مروگرها پشتیبانی نشود، می تواند فایل های ts و ES3، ES4 را در ES5 کامپایل کند.

## What is an object initializer .۲۳

یک شیء اولیه عبارتی است که مقدار دهی 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 .۲۴

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 .۲۵

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

```
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 .۲۶

می توانید 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 می شود. باید 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 .۲۷

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 ارشی 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 .۲۸

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 .۲۹

می توانید از متد `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 .۳۰

متدهایی که برای تعیین اینکه آیا یک شی قابل توسعه است یا نه استفاده می‌شود. یعنی اینکه آیا می‌تواند ویژگی‌های جدیدی به آن اضافه شود یا خیر.

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

**نکته:** به طور پیش‌فرض، تمام اشیاء قابل گسترش هستند. به عنوان مثال، ویژگی‌های جدید را می‌توان اضافه یا تغییر داد.

## How do you prevent an object to extend .۳۱

متدهایی که برای جلوگیری از افزودن ویژگی‌های جدید به یک شی استفاده می‌شود. به عبارت دیگر، از پسوندهای بعدی به شی جلوگیری می‌کند. بباید استفاده از این ویژگی را بینیم،

```
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 .۳۲

شما می‌توانید یک شی غیرقابل گسترش را به ۳ روش علامت گذاری کنید.

Object.preventExtensions .۳۳

Object.seal .۳۴

Object.freeze .۳۵

```
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 .<sup>۳۶</sup>

متد «Object.defineProperties()» برای تعریف یا اصلاح ویژگی‌های موجود مستقیماً روی یک شی و برگرداندن شی استفاده می‌شود. باید 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 .<sup>۳۷</sup>

پشته (Node.js و MongoDB، Express، AngularJS) 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 .<sup>۳۸</sup>

مبهم سازی عمل عمده ایجاد کد جاوا اسکریپت مبهم (یعنی کد منبع یا ماشین) است که درک آن برای انسان دشوار است. این چیزی شبیه به رمزگذاری است، اما یک ماشین می‌تواند کد را درک کرده و آن را اجرا کند. باید تابع زیر را قبل از **Obfuscation** ببینیم،

```
<"span dir="ltr" align="left">
javascript``
} ()function greeting
;('console.log('Hello, welcome to JS world
{
```
```

```
<span/>
```

و بعد از کد **Obfuscation** به صورت زیر ظاهر می‌شود

```
<"span dir="ltr" align="left">  
javascript``  
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  
```
```

```

```

\*\*[فهرست] (#فهرست)\*\*

## Why do you need Obfuscation .۲۶۷

در زیر چند دلیل برای مبهم سازی آمده است،

۱. اندازه کاهش می‌یابد. بنابراین انتقال داده بین سرور و مشتری سریع خواهد بود.
۲. این منطق کسب و کار را از دنیای خارج پنهان می‌کند و از کد در برابر دیگران محافظت می‌کند
۳. مهندسی معکوس بسیار دشوار است
۴. زمان دانلود کاهش می‌یابد

## What is Minification .۵

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 .۶

به طور معمول توصیه می‌شود برای ترافیک سنگین و نیازهای فشرده منابع از **Minification** استفاده کنید. اندازه فایل را با مزایای زیر کاهش می‌دهد

٧. زمان بارگذاری یک صفحه وب را کاهش می دهد
٨. در مصرف پنهانی باند صرفه جویی می کند

## What are the differences between Obfuscation and Encryption .٩

در زیر تفاوت های اصلی بین مبهم سازی و رمزگذاری وجود دارد.

	ویژگی   مبهم سازی   رمزگذاری	-----   -----   -----
	تعريف   تغییر فرم هر داده به هر شکل دیگر	اطلاعات به فرم ناخوانا با استفاده از کلید
	کلیدی برای رمزگشایی   می توان آن را بدون هیچ کلید رمزگشایی کرد	لازم است
	فرم داده های هدف   به فرم پیچیده تبدیل می شود	تبدیل به فرم ناخوانا

## What are the common tools used for minification .١٠

ابزارهای آنلاین/آفلاین زیادی برای کوچک کردن فایل های جاوا اسکریپت وجود دارد.

١١. کامپایلر بسته شدن گوگل UglifyJS2 .١٢
١٣. jsmin .١٤
١٥. /javascript-minifier.com .١٤
١٦. prettydiff.com .١٥

## How do you perform form validation using javascript .١٦

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 .۱۷

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 .۱۸

روش های DOM زیر برای اعتبار سنجی محدودیت در ورودی نامعتبر موجود است.  
۱۹. checkValidity(): اگر یک عنصر ورودی حاوی داده های معتبر باشد، مقدار true را برمی گرداند.  
۲۰. setCustomValidity(): برای تنظیم خاصیت validationMessage یک عنصر ورودی استفاده می شود.  
بایاید 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 .۲۱

در زیر لیستی از برخی از ویژگی های DOM اعتبار سنجی محدودیت موجود است،  
۲۲. validity: فهرستی از ویژگی های بولی مربوط به اعتبار یک عنصر ورودی را ارائه می دهد.  
۲۳. validationMessage: زمانی که اعتبار نادرست باشد، پیام را نمایش می دهد.  
۲۴. willValidate: این نشان می دهد که آیا یک عنصر ورودی اعتبار سنجی می شود یا خیر.

## What are the list of validity properties .۲۵

ویژگی اعتبار یک عنصر ورودی مجموعه ای از ویژگی های مربوط به اعتبار داده ها را ارائه می دهد.  
۲۶. customError: اگر یک پیام اعتبار سفارشی تنظیم شده باشد، true را برمی گرداند.  
۲۷. patternMismatch: اگر مقدار یک عنصر با ویژگی الگوی آن مطابقت نداشته باشد، مقدار true را برمی گرداند.  
۲۸. rangeOverflow: اگر مقدار یک عنصر از ویژگی max آن بیشتر باشد، مقدار true را برمی گرداند.  
۲۹. rangeUnderflow: اگر مقدار یک عنصر کمتر از ویژگی min باشد، مقدار true را برمی گرداند.

- .۳۰. اگر مقدار عنصر مطابق با ویژگی stepMismatch نامعتبر باشد، مقدار true را برمی گرداند.
- .۳۱. اگر مقدار یک عنصر از ویژگی maxLength آن بیشتر شود، مقدار true را برمی گرداند.
- .۳۲. اگر مقدار یک عنصر بر اساس ویژگی نوع نامعتبر باشد، مقدار true را برمی گرداند.
- .۳۳. اگر عنصری با ویژگی مورد نیاز ارزش نداشته باشد، مقدار true را برمی گرداند.
- .۳۴. اگر مقدار یک عنصر معتبر باشد، مقدار true را برمی گرداند.

## Give an example usage of rangeOverflow property .۳۵

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 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 .۳۶

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 معادله های دقیقی ارائه نکنند. به عنوان مثال، می توانید از فریز یا فریز یا مهر و موم روی شی استفاده کنید

```
var DaysEnum = Object.freeze({ "monday":1, "tuesday":2, "wednesday":3, ... })
```

## What is an enum .۳۷

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 .۳۸

می توانید از متد Object.getOwnPropertyNames() استفاده کنید که آرایه ای از تمام ویژگی هایی را که مستقیماً در یک شیء داده شده یافت می شود، برمی گرداند. باید 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 .۳۹

می‌توانید از متده استفاده کنید که تمام توصیف‌گرهای ویژگی یک شی معین را بر می‌گرداند. 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 .۴۰

توصیفگر ویژگی رکوردي است که دارای ویژگی های زیر است  
۴۱. value: ارزش مرتبط با ملک

۴۲. writable: تعیین می کند که آیا مقدار مرتبط با ویژگی قابل تغییر است یا خیر

۴۳. configurable: اگر بتوان نوع توصیفگر این ویژگی را تغییر داد و اگر ویژگی را بتوان از شی مربوطه حذف کرد، مقدار true را بر می‌گرداند.

۴۴. enumerable: تعیین می کند که آیا ویژگی در هنگام شمارش خصوصیات روی شی مربوطه ظاهر می شود یا خیر.

۴۵. set: تابعی که به عنوان تنظیم کننده برای ویژگی عمل می کند

۴۶. get: تابعی که به عنوان یک گیرنده برای ملک عمل می کند

## How do you extend classes .۴۷

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 بندی 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 .۴۸

ویژگی «window.location.url» برای تغییر url مفید خواهد بود اما صفحه را دوباره بارگیری می کند. HTML5 متدهای «()history.replaceState» و «()history.pushState» 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 .۴۹

متده «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')) //true

```

## How do you compare scalar arrays .۵۰

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 .۵۱

شیء «URL» رشته url را می‌پزیرد و از ویژگی «searchParams» این شی می‌توان 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 .۵۲

می‌توانید از متد «Number.prototype.toLocaleString()» استفاده کنید که رشته‌ای را با نمایشی حساس به زبان مانند جداکننده هزار، ارز و غیره از این عدد برمی‌گرداند.

```

function convertToThousandFormat(x){
 return x.toLocaleString(); // 12,345.679
}

console.log(convertToThousandFormat(12345.6789));

```

## What is the difference between java and javascript .۵۳

هر دو زبان برنامه نویسی کاملاً نامرتب هستند و هیچ ارتباطی بین آنها وجود ندارد. جاوا بصورت ایستا تایپ می‌شود، کامپایل می‌شود، روی ماشین مجازی خود اجرا می‌شود. در حالی که جاوا اسکریپت به صورت پویا تایپ می‌شود، تفسیر می‌شود و در محیط‌های مرورگر و nodejs اجرا می‌شود. بیایید تفاوت‌های عمدۀ را در قالب جدولی ببینیم،

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
محفوذه عملکردی	محفوذه بلوك	محفوذه
مبتنی بر رویداد	بر اساس موضوع	همزنمانی

جاوا اسکریپت	جاوا	ویژگی
از حافظه کمتری استفاده می کند. از این رو برای صفحات وب استفاده خواهد شد	از حافظه بیشتر استفاده می کند	حافظه

## Is javascript supports namespace .۵۴

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 آلوده می کند. بباید 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.

## How do you declare namespace .۵۵

حتی اگر جاوا اسکریپت قادر فضاهای نام باشد، می توانیم از IIFE برای ایجاد فضاهای نام استفاده کنیم. **Using Object Literal Notation** بباید متغیرها و توابع را درون یک Object literal بپیچیم 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
```

**(Using IIFE (Immediately invoked function expression .۵۷)**: جفت پرانتز بیرونی IIFE یک محدوده محلی 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();
}());

```

در 6 ECMAScript، شما می توانید به سادگی از یک بلوک و یک اعلان برای محدود کردن دامنه یک متغیر به یک بلوک استفاده کنید.

```

{
 let myFunction= function fun1(){
 console.log("This is a first definition");
 }
 myFunction();
}
//myFunction(): ReferenceError: myFunction is not defined.

{
 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 .۵۹

در ابتدا باید 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 .۶۰

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 .٦١

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 .٦٢

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.  
٦٣. (document.getElementById(id)): یک عنصر را با Id پیدا می کند  
٦٤. (document.getElementsByTagName(name)): یک عنصر را با نام تگ پیدا می کند  
٦٥. (document.getElementsByClassName(name)): یک عنصر را با نام کلاس پیدا می کند

## What is jQuery .٦٦

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 .٦٧

یک موتور جاوا اسکریپت با کارایی بالا منبع باز است که توسط مرورگر Google Chrome استفاده می‌شود و به زبان C++ نوشته شده است. همچنین در پروژه node.js استفاده می‌شود. WebAssembly و ECMAScript را پیاده‌سازی می‌کند و روی ویندوز 7 یا بالاتر، macOS 10.12+ و سیستم‌های لینوکس که از پردازنده‌های IA-32، ARM، MIPS x64 یا استفاده می‌کنند، اجرا می‌شود.

**نکته:** می‌تواند به صورت مستقل اجرا شود یا می‌تواند در هر برنامه C++ تعبیه شود.

## Why do we call javascript as dynamic language .۶۸

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 می‌توان 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 .۶۹

عملگر «void» عبارت داده شده را ارزیابی می‌کند و سپس تعریف نشده (یعنی بدون بازگشت مقدار) را برمی‌گرداند. نحوه صورت زیر خواهد بود،

```
void (expression)
void expression
```

بیایید پیامی را بدون هیچ گونه تغییر مسیر یا بارگیری مجدد نمایش دهیم

```
Click here to see a message
```

**نکته:** این عملگر اغلب برای بدست آوردن مقدار اولیه تعریف نشده با استفاده از "void(0)" استفاده می‌شود.

## How to set the cursor to wait .۷۰

مکان نما می‌توان برای انتظار در جاوا اسکریپت با استفاده از ویژگی "cursor" تنظیم کرد. بیایید 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 می‌شود

```
<body onload="myFunction()">
```

## How do you create an infinite loop .٧١

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 .٧٢

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 .٧٣

```
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 ۰ ۱ ۲ ۳ ۴ ۴ ۴ ۴ ۴ ۰ ۴ ۴ ۴ 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 '۰ ۱ ۲ ۳' 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 «۰ ۱ ۲ ۳» است).

## List down some of the features of ES6. ۷۴

- در زیر لیستی از برخی از ویژگی‌های جدید ES6 آمده است.
- ۷۵. پشتیبانی از ثابت‌ها یا متغیرهای تغییرناپذیر
  - ۷۶. پشتیبانی Block-scope برای متغیرها، ثابت‌ها و توابع
  - ۷۷. Arrow functions
  - ۷۸. پارامترهای پیش‌فرض
  - ۷۹. Rest and Spread
  - ۸۰. Template Literals
  - ۸۱. Multi-line Strings
  - ۸۲. Destructuring Assignment
  - ۸۳. Enhanced Object Literals
  - ۸۴. Promises
  - ۸۵. Classes
  - ۸۶. Modules

\*\*[فهرست] (#فهرست)

## 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 2015 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 مازولها 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.  
۳۲۰ یک مازول 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(...);
}
```

۱۳۲. 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(...);
```

۱۳۳. یک ماژول را از داخل یک اسکریپت معمولی به جای یک ماژول وارد کنید.

## What are typed arrays. ۱۳۴

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،

۱۳۴. آرایه ای از اعداد صحیح امضا شده 8 بیتی

۱۳۵. آرایه ای از اعداد صحیح امضا شده 16 بیتی

۱۳۶. آرایه ای از اعداد صحیح امضا شده 32 بیتی

۱۳۷. آرایه ای از اعداد صحیح بدون علامت 8 بیتی

۱۳۸. آرایه ای از اعداد صحیح بدون علامت 16 بیتی

۱۳۹. آرایه ای از اعداد صحیح بدون علامت 32 بیتی

۱۴۰. آرایه ای از اعداد ممیز شناور 32 بیتی

۱۴۱. آرایه ای از اعداد ممیز شناور 64 بیتی

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. ۱۴۲

ماژول لودر ویژگی های زیر را ارائه می دهد

Dynamic loading. ۱۴۳

State isolation. ۱۴۴

Global namespace isolation. ۱۴۵

Compilation hooks. ۱۴۶

Nested virtualization. ۱۴۷

## What is collation. ۱۴۸

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 آگاه است. بیایید ویژگی های مقایسه و مرتب سازی را در نظر بگیریم، **:Comparison**.<sup>۳۴۹</sup>

```
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**.<sup>۳۴۰</sup>

```
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.<sup>۳۴۱</sup>

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 توسعه 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.<sup>۳۴۲</sup>

```
[..., '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.<sup>۳۴۳</sup>

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 .<sup>۳۴۴</sup>

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 .<sup>۳۴۵</sup>

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)) را بررسی کنیم -

```
//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 .<sup>۳۴۶</sup>

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 .<sup>۳۴۷</sup>

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 .۳۴۸

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 .۳۴۹

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 .۳۵۰

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 .۳۵۱

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

```

```
console.log(document.getElementById('checkboxname').checked); // true or false
```

## What is the purpose of double tilde operator .۳۵۲

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 .۳۵۳

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 .۳۵۴

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 .۳۵۵

```
console.log("Welcome to JS world"[0])
```

خروجی عبارت فوق "W" است.

\*توضیح:\*\* نماد برآکت با شاخه خام 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 متدها (charAt) برای به دست آوردن نتیجه دلخواه استفاده کنید.

\*\*[فهرست] (#فهرست)

## What is the purpose of Error object .۳۵۸

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 خطاها یا خطاهای 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».

۳. 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

```
<"span dir="ltr" align="left>
 javascript```
 { {} ()if (someCondition) { function f
  ```

<span/>
```

از این رو، خطاهای موارد فوق برای جلوگیری از خطا در محیط های توسعه/تولید مفید هستند.

**[فهرست] (#فهرست)

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 یکتابع است در حالی که یک آرگومان نشان دهنده مقدار داده شده به یکتابع در هنگام فراخوانی آن است. باید این را با یکتابع 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

متدهای some() برای آزمایش اینکه آیا حداقل یک عنصر در آرایه از آزمون پیادهسازی شده توسط تابع ارائه شده عبور می‌کند یا خیر استفاده می‌شود. متدهای مقدار بولی بر می‌گردانند. باید 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

متدهای concat() برای پیوستن دو یا چند آرایه با برگرداندن یک آرایه جدید حاوی تمام عناصر استفاده می‌شود. نحوه صورت زیر خواهد بود،

```
array1.concat(array2, array3, ..., arrayX)
```

باید مثالی از الحاق 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.

```
var empDetails = {
  name: "John", age: 25, expertise: "Software Developer"
}
```

برای ایجاد یک نسخه تکراری

```
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 «empDetailsDeepCopy» را تحت تأثیر قرار خواهد داد.

How do you create specific number of copies of a string .۳۴۶

متدهای «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()» میتواند 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 برمی‌گرداند.

```
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 می‌شود. اما اگر می‌خواهید بهخصوص در ابتدای انتهای رشته را برش دهید، می‌توانید از روش‌های «trimEnd/trimRight» و «trimStart/trimLeft» استفاده کنید. باید نمونه ای از این روش‌ها را 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

باید 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 برمی‌گرداند. از آنجا که عنصر توسط عملگر unary پیشوند است و مفسر جاوا اسکریپت سعی می‌کند آن عنصر را به یک نوع عدد تبدیل کند. از آنجایی که تبدیل با شکست مواجه می‌شود، مقدار عبارت به مقدار NaN منجر می‌شود.

Does javascript uses mixins .۳۵۰

What is a thunk function .۳۵۱

فقط تابعی است که ارزیابی مقدار را به تأخیر می‌اندازد. هیچ آرگومانی نمی‌گیرد، اما هر زمان که thunk می‌کنید، مقدار را می‌دهد. به عنوان مثال، از آن استفاده می‌شود که اکنون اجرا نشود، اما زمانی در آینده خواهد بود. بباید یک مثال همزمان بگیریم،

```
const add = (x,y) => x + y;  
  
const thunk = () => add(2,3);  
  
thunk() // 5
```

What are asynchronous thunks .۳۵۲

Thunk های ناهمزمان برای ایجاد درخواست های شبکه مفید هستند. بباید 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 نمی‌شود، اما تنها زمانی فراخوانی می‌شود که داده‌ها از نقطه پایانی API در دسترس باشند. redux نیز برای ناهمزمان کردن کد ما استفاده می‌شود. بهترین مثال زمان واقعی، کتابخانه مدیریت حالت است که از thunk های ناهمزمان برای به تأخیر انداختن اعمال برای ارسال استفاده می‌کند.

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 شعاد در اشیاء پنجره وجود ندارد، یک مقدار تعریف نشده برمی‌گرداند و مضرب مقدار 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 می‌کنیم 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 پرچم‌های سراسری و چند خطی هستند.

What is the difference between reflow and repaint. ۳۵۵

رنگ‌آمیزی مجدد زمانی اتفاق می‌افتد که تغییراتی ایجاد می‌شود که روی دید یک عنصر تأثیر می‌گذارد، اما روی طرح آن تأثیر نمی‌گذارد. نمونه‌هایی 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 می‌گذارد. تغییر اندازه پنجره مرورگر، تغییر فونت، تغییر محتوا (مانند تایپ متن توسط کاربر)، استفاده از روش‌های جاوا اسکریپت شامل سبک‌های محاسبه شده، افزودن یا حذف عناصر از DOM، و تغییر کلاس‌های یک عنصر چند مورد از مواردی هستند که می‌توانند 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 می‌شود.

What happens with negating an array. ۳۵۶

نفي یک آرایه با کاراکتر «!»، آرایه را به یک بولی وادر می‌کند. از آنجایی که آرایه‌ها صدق در نظر گرفته می‌شوند، پس نفي آن "نادرست" را برمی‌گرداند.

```
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 می‌کند و آنها را به هم متصل می‌کند. به عنوان مثال، نتیجه اضافه کردن آرایه‌ها 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 دهید، مقدار falsy به 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

رشته self را می توان با ترکیب کاراکترهای [() ! + تشکیل داد. برای رسیدن به این الگو باید قراردادهای زیر را به خاطر بسپارید.

- ۳۶۰. از آنجایی که آرایه ها مقادیر واقعی هستند، با نفی آرایه ها false تولید می شود: [] != false ===""
 - ۳۶۱. طبق قوانین اجباری جاوا اسکریپت، اضافه کردن آرایه ها به هم آنها را به رشته بندی تبدیل می کند: [] == "" == + []
 - ۳۶۲. 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 ۱' را تولید می کند: +(![])) == 1
- با اعمال قوانین فوق می توانیم شرایط زیر را استخراج کنیم

```
<"span dir="ltr" align="left">
```

```
javascript```
"false" === [] + []
1 === []+!+
```
```

```

```

اکنون الگوی کاراکتر به صورت زیر ایجاد می شود.

```
<"span dir="ltr" align="left">
```

```
javascript```
s e l f
~~~~~       ~~~~~       ~~~~~       ~~~~~
```

```
[0]([ ] + [ ]) + [2]([ ] + [ ]) + [4]([ ] + [ ]) + [3]([ ] + [ ])
~~~~~       ~~~~~       ~~~~~       ~~~~~
+ [[]+!+[]+!+[]+!+]([] + [])
+ [[]+!+[]+!+[]+!+[]+!+]([] + [])
+ [[]+!+[]+!+]([] + [])
[[]+]([] + [])
~~~~~       ~~~~~
[[ ]+]( [ ]+[ ]) + [[ ]+!+[ ]+!+]([ ]+[ ]) + [[ ]+!+[ ]+!+[ ]+!+]([ ]+[ ]) + [[ ]+!+[ ]+!+[ ]+!+]([ ]+[ ])
```
```

```
<span/>
```

\*\*[فهرست](#فهرست)\*\*

## 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. بیایید 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،

۰% - یک شی را می گیرد،

۵% - یک رشته می گیرد،

۱۰% - برای اعشار یا عدد صحیح استفاده می شود

این متغیرها را می توان در log به صورت زیر نشان داد

```
<"span dir="ltr" align="left">

javascript```
;{"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
```

<span/>

**[فهرست](#فهرست)**
```

## 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

«console.dir()» 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);
```

داده هایی که در قالب جدول مشاهده می شوند،

```

>     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 = [].concat(...biDimensionalArr); // [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

اما شما می توانید آن را با آرایه های چند بعدی با تماس های بازگشتی 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

می توانید 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 می شود. این برای هشدار دادن به کاربران در مورد از دست داده های فعلی مفید است.

```
window.onbeforeunload = function() {
  alert("Your 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 بدنه 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 می‌خواهید اقداماتی را روی آن‌ها انجام دهید، به طور موقت به یک شی (اجکت Wrapper) تبدیل یا مجبور می‌شوند. برای مثال، اگر متدهایUpperCase() را روی یک مقدار رشته اولیه اعمال کنید، خطایی ایجاد نمی‌کند، اما حروف بزرگ رشته را برمی‌گرداند.

```
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 است و لیست اشیاء wrapper عبارتند از .BigInt، String، Number، Boolean، Symbol

\*\*[فهرست] (#فهرست)

## What is AJAX .۳۸۱

AJAX مخفف Asynchronous JavaScript و XML است و گروهی از فناوری‌های مرتبط با HTML، CSS، JavaScript، XMLHttpRequest API (و غیره) است که برای نمایش داده‌ها به صورت ناهمزمان استفاده می‌شود. یعنی ما می‌توانیم داده‌ها را به سرور ارسال کنیم و بدون بارگیری مجدد صفحه وب، داده‌ها را از سرور دریافت کنیم.

## What are the different ways to deal with Asynchronous Code .۳۸۲

در زیر لیستی از روش‌های مختلف برای مقابله با کدهای ناهمزمان آورده شده است.  
callback.1

۳۸۳. پرامیس ها

Async/await.3

۳۸۴. کتابخانه‌های شخص ثالث مانند bluebird، async.js، و غیره

\*\*[فهرست] (#فهرست)

## How to cancel a fetch request .۳۸۳

تا چند روز پیش، یکی از کاستی‌های وعده‌های بومی راه مستقیمی برای لغو درخواست واکشی نیست. اما «AbortController» جدید از مشخصات js به شما امکان می‌دهد از سیگنالی برای لغو یک یا چند تماس واکشی استفاده کنید. جریان اصلی لغو یک درخواست واکشی به صورت زیر خواهد بود.

۳۸۴. یک نمونه «AbortController» ایجاد کنید

۳۸۵. ویژگی سیگنال یک نمونه را دریافت کنید و سیگنال را به عنوان یک گزینه واکشی برای سیگنال ارسال کنید

۳۸۶. برای لغو تمام واکشی‌هایی که از آن سیگنال استفاده می‌کنند، با ویژگی abort's AbortController تماس بگیرید.

به عنوان مثال، بباید یک سیگنال را به چندین تماس واکشی ارسال کنیم، همه درخواست‌های با آن سیگنال لغو می‌شوند.

```

<"span dir="ltr" align="left>

javascript```
;()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
```

```

<span/>

\*\*[فهرست] (#فهرست)

## 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. **تشخیص گفتار (تشخیص گفتار ناهمزمان 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. مثال زیر نحوه استفاده از این API برای دریافت متن از گفتار را نشان می‌دهد.

```

<"span dir="ltr" align="left>

javascript```
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
```

```

<span/>

در این API، مرورگر برای استفاده از میکروفون شما از شما اجازه می‌خواهد

**(SpeechSynthesis (Text-to-Speech .)** این امکان را فراهم می کند 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 می شود.

```
<"span dir="ltr" align="left">  
  
javascript``  
}{('speechSynthesis' in window  
;(!var speech = new SpeechSynthesisUtterance('Hello World  
;'speech.lang = 'en-US  
;(window.speechSynthesis.speak(speech  
{  
```  
  
<span/>
```

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)  
\*\*[فهرست] (#فهرست)\*\*

## What is minimum timeout throttling .۳۸۵

هم مرورگر و هم محیط های جاوا اسکریپت NodeJS با حداقل تاخیری که بیشتر از 0 میلی ثانیه است دریچه گاز را انجام می دهند. این بدان معناست که حتی اگر تنظیم یک تاخیر 0ms به طور آنی اتفاق نیفتد.

**مرورگرها:** حداقل 4 میلی ثانیه تاخیر دارند. این دریچه گاز زمانی اتفاق می افتد که تماس های متوالی به دلیل تودرتوی برگشت به تماس (عمق معین) یا پس از تعداد معین فواصل متوالی آغاز شود.

توجه: مرورگرهای قدیمی حداقل 10 میلی ثانیه تاخیر دارند.

**Nodejs:** حداقل 1ms تاخیر دارد. این دریچه گاز زمانی اتفاق می افتد که تاخیر بزرگتر از 2147483647 یا کمتر از 1 باشد.

```
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 .<sup>۳۸۶</sup>

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 .<sup>۳۸۷</sup>

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.

۳۸۸. 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.

۳۸۹. هنگامی که یک رویداد شلیک می شود، پاسخ تماس رویداد به صف کار اضافه می شود

۳۹۰. وقتی به یک `setInterval` یا `setTimeout` یا `requestAnimationFrame` پاسخ تماس مربوطه به صف کار اضافه می شود

\*\*[فهرست] (#فهرست)\*\*

## What is microtask .<sup>۳۸۸</sup>

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.all`.

غیره.

**توجه:** همه این ریزکارها در همان چرخش حلقه رویداد پردازش می شوند.

## What are different event loops .<sup>۳۸۹</sup>

## What is the purpose of queueMicrotask .<sup>۳۹۰</sup>

## How do you use javascript libraries in typescript file .<sup>۳۹۱</sup>

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، بیایید 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. می‌توانید از این کتابخانه در کد تایپ اسکریپت به صورت زیر استفاده کنید.

```
<span dir="ltr" align="left">
```

```
javascript```
declare var customLibrary
```
```

```
<span/>
```

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 زیر آمده است

```
<span dir="ltr" align="left">
```

```
javascript```
;var customLibrary: any
```
```

```
<span/>
```

```
**[فهرست] (#فهرست)
```

## What are the differences between promises and observables. ۳۹۲

برخی از تفاوت‌های عمدۀ در شکل جدولی  
| وعده‌ها | قابل مشاهده |

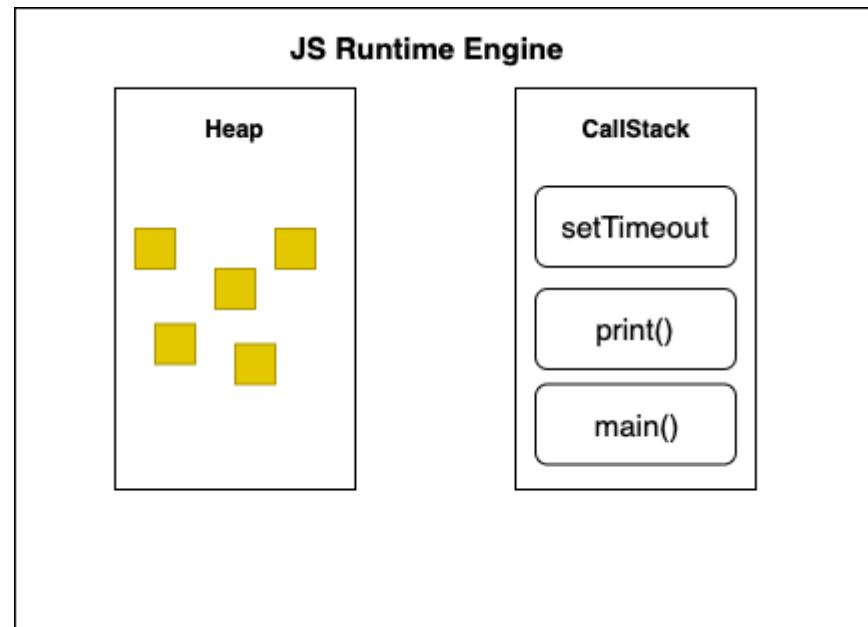
----- | -----

| فقط یک مقدار را در یک زمان منتشر می‌کند | چندین مقدار را در یک دوره زمانی منتشر می‌کند (جريان مقاديری از 0 تا چندگانه) |

مشتق در طبیعت؛ قرار است فوراً فراخوانی شوند	تبل در طبیعت؛ آنها برای فراخوانی نیاز به اشتراک دارند
همیشه ناهمzman است حتی اگر بلافصله حل شود	قابل مشاهده می‌تواند همزمان یا ناهمzman
هیچ اپراتور ارائه نمی‌دهد	اپراتورهایی مانند retryWhen و map، forEach، filter، reduce
قابل لغو نیست	با استفاده از روش unsubscribe() لغو شد

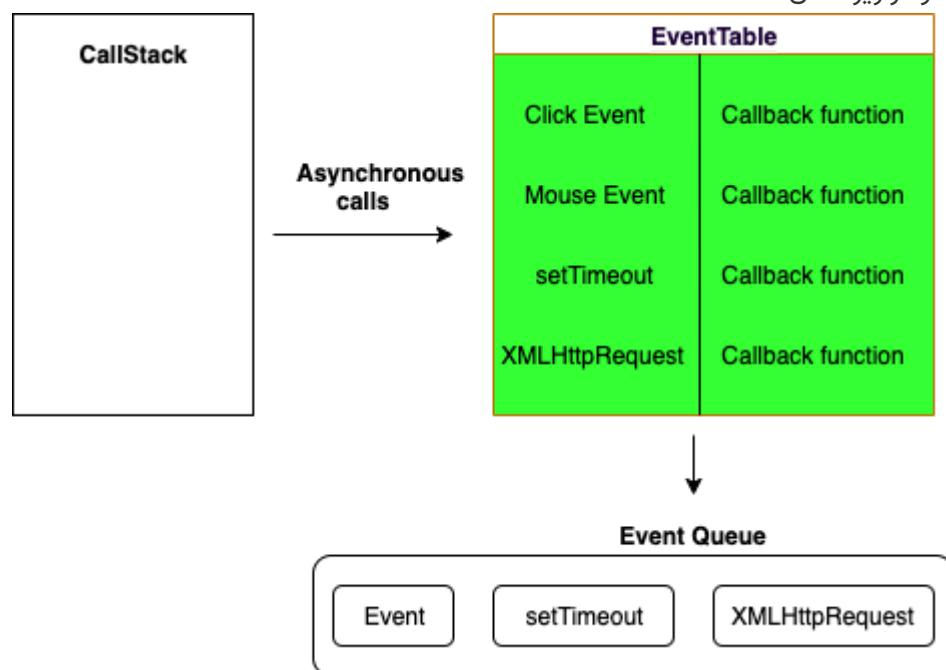
## What is heap. ۳۹۳

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 **call-stack** و **heap** 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]
```

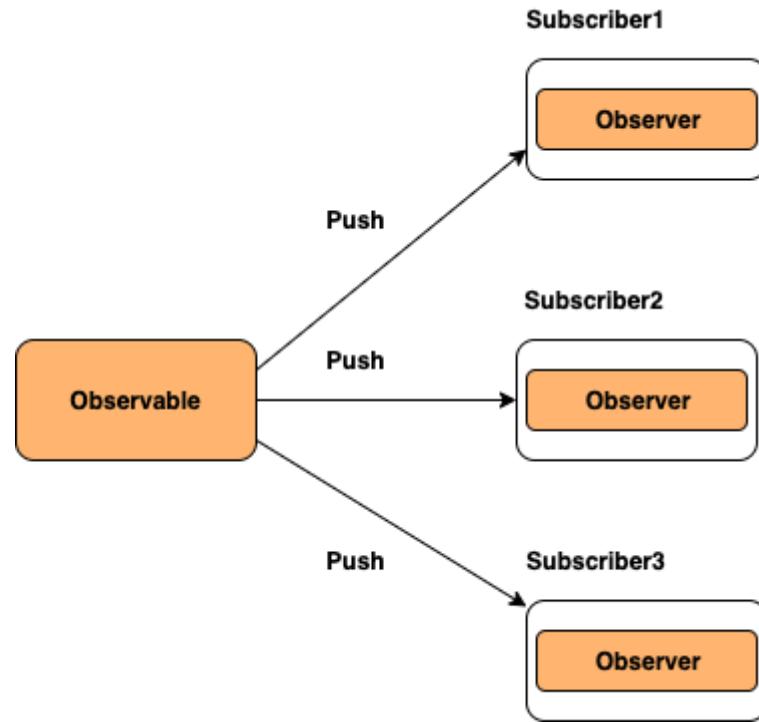
و آرایه را نیز می توان خالی کرد

```
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 .۱۰۸

در حین استفاده از کد ناهمزمان، وعده‌های ES6 جاوا اسکریپت می‌توانند زندگی شما را بدون داشتن هرم برگشت تماس و مدیریت خطا در هر خط دوم بسیار آسان‌تر کند. اما Promises مشکلاتی دارد و بزرگ‌ترین آنها به‌طور پیش‌فرض 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 محیط‌های جاوا اسکریپت مدرن وجود دارند که هیچ خطایی را چاپ نمی‌کنند. شما می‌توانید این مشکل را به روش‌های 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 .۱۰۹ chains

```

<"span dir="ltr" align="left>

javascript```
} ()Promise.resolve('promised value').then(function
;('throw new Error('error
} (catch(function(error.{{
;(console.error(error.stack
;({{
```
<span/>

```

اما تایپ کردن برای هر زنجیره پرامیس ها و پرخاطب نیز بسیار دشوار است.  
می‌توانید ابتدا راه حل ها را جایگزین کنید و سپس با روش انجام شده بلوک ها را بگیرید .۲

```
Promise.resolve('promised value').done(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، می توانید بلوک «انجام شد» را مانند زیر حذف کنید.

```
javascript```
()getDataFromHttp
} (then(function(result.
;((return displayData(processDataAsync(result
(
````
```

و سپس فراموش کرده اید که بلوک «انجام شد» را به بلوک «سپس» اضافه کنید که منجر به خطاهای خاموش می شود.

#### :Extend ES6 Promises by Bluebird .۳

کننده "پیش فرض" در Rejection است که تمام خطاهای را از stderr چاپ می کند. پس از نصب، می توانید ردهای کنترل نشده را پردازش کنید

```
```javascript
Promise.onPossiblyUnhandledRejection(function(error){
    throw error;
});
````
```

و یک رد را دور بیندازید، فقط با یک شکار خالی آن را مدیریت کنید

```
<"span dir="ltr" align="left">

javascript```
;({} ()Promise.reject('error value').catch(function
````

<span/>
```

یک زمان اجرا ساده، مدرن و ایمن برای جاوا اسکریپت و تایپ اسکریپت است که از موتور جاوا اسکریپت V8 و زبان Deno برنامه نویسی Rust استفاده می کند.

## How do you make an object iterable in javascript .۱۴۱۰

به طور پیش فرض، اشیاء ساده قابل تکرار نیستند. اما می توانید 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}
```

ابتدا، قبل از صحبت در مورد "دوم خوانی مناسب" باید در مورد دم بدایم. 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 می کنید، آنها تمام داده های لازم را که به آن نیاز دارد را بدون تکیه بر پشته، در بازگشت به پایین منتقل می کنند.

```
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 برمی گرداند. اما اນباشت کننده کل را به عنوان آرگومان بدون استفاده از حافظه پشته در تماس های بازگشته رديابی می کند.

## How do you check an object is a promise or not .۴۱۲

اگر نمی دانید یک مقدار یک وعده است یا نه، مقدار را به صورت «Promise.resolve(value)» بپیچید که یک قول را برمی گرداند.

```

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 .۱۳

می‌توانید از ویژگی شبه «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.  
۱۴. اگر سازنده یا تابعی با استفاده از عملگر جدید فراخوانی شود، new.target یک مرجع به سازنده یا تابع برمی‌گرداند.  
۱۵. 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

```

\*\*[فهرست] (#فهرست)\*\*

## What are the differences between arguments object and rest parameter .۱۶

- سه تفاوت اصلی بین پارامترهای شیء آرگومان و پارامترهای استراحت وجود دارد.  
 ۱۵. شیء آرگومان ها آرایه مانند است اما آرایه نیست. در حالی که بقیه پارامترها نمونه های آرایه هستند.  
 ۱۶. شیء آرگومان ها از روش هایی مانند pop، sort، forEach یا map، پشتیبانی نمی کند. در حالی که این روش ها را می توان در پارامترهای استراحت استفاده کرد.  
 ۱۷. بقیه پارامترها فقط آنها بی 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.

\*\*[فهرست] (#فهرست)\*\*

## What are the differences between spread operator and rest parameter .۱۵

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 .۱۶

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 .۴۱۶

```

const SomeObj = {
  *[Symbol.iterator] () {
    yield 1;
    yield 2;
    yield 3;
  }
}

console.log(Array.from(SomeObj)); // [ 1, 2, 3 ]

```

## What are the built-in iterables .۴۱۷

- 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

## What are the differences between for...of and for...in statements .۴۱۸

- هم برای...in و هم برای...از دستورات روی ساختارهای داده s 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 می‌شود حلقه اول، ۰، ۱، ۲ و newProp را در حین تکرار روی شی آرایه ثبت می‌کنند. حلقه for..of روی مقادیر یک ساختار داده arr تکرار می‌شود و a، b، c را در کنسول ثبت می‌کنند.

## How do you define instance and non-instance properties .۴۱۹

خصوصیات Instance باید در داخل متدهای کلاس تعریف بشون. به عنوان مثال، مشخصات نام و سن سازنده داخلی هم مثل مثال پایین تعریف می‌شون.

```

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 بشون. بیان 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 نمی‌کند. اما زمانی که نوع یک عدد و مقدار NaN باشد مقدار true را برمی‌گرداند. بیایید تفاوت را با یک مثال ببینیم،

```

isNaN('hello'); // true
Number.isNaN('hello'); // false

```

