

به نام خدا

مجموعه سوال و جواب‌های جاوسکریپت

عیسی رضائی

مجموعه سوال و جواب‌های جاواسکریپتی

اگه از کتاب خوشتون اوMD به گیت‌هابمون مراجعه کنین و بهمون  بدین. اگه هم قصد مشارکت داشتید خیلی خوشحال می‌شیم 😊
<http://github.com/mariotek>

دانلود کتاب به فرمتهای PDF/Epub

می‌تونین خیلی راحت نسخه آنلاین کتاب استفاده کنین یا اگه به فایل کتاب می‌خواهین دسترسی داشته باشین، از بخش ریلیزهای گیت‌هاب به فرمتهای مختلف آخرین نسخه کتاب رو می‌تونین دریافت کنین.

فهرست

ردیف	سوال	صفحه
۱	روش‌های ایجاد objects توی جاواسکریپت چیا هستن؟	
۲	زنجیره prototype چیه؟	
۳	تفاوت‌های بین Bind، Apply و Call چیا هستن؟	
۴	فرمت JSON چیه و عملیات‌های معمول بر روی آن چیا هستن؟	
۵	هدف از متدهای slice و آرایه‌ها چیه؟	
۶	هدف از متدهای splice و آرایه‌ها چیه؟	
۷	تفاوت متدهای splice و slice چیا هستن؟	
۸	تفاوت‌های Map و Object چیا هستن؟	
۹	تفاوت‌های بین عملگرهای == و === چیا هستن؟	
۱۰	توابع arrow-function یا lambda چی هستن؟	

ردیف	سوال	صفحه
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 یعنی چی؟	
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 تعريف کنیم؟	
۲۱	Temporal-Dead-Zone چیه؟	
۲۲	(توابع بلا فاصله صدا زده شده) IIFE چی هستن؟	
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؟	

صفحه	سوال	ردیف
	توى service-worker چطوري مىشه DOM رو دستکاري کرد؟	۳۲
	چطوري مىتونيم بین ریست شدن های service-worker داده های مورد نظرمون رو مجدد استفاده کنیم؟	۳۳
	IndexedDB چیه؟	۳۴
	Web-storage چیه؟	۳۵
	Post-message چیه؟	۳۶
	Cookie چیه؟	۳۷
	چرا به cookie نیاز داریم؟	۳۸
	گزینه های قابل تنظیم توى cookie چیا هستن؟	۳۹
	چطوري مىشه به cookie رو حذف کرد؟	۴۰
	تفاوت های بین session-storage و cookie، local-storage چیا هستن؟	۴۱
	تفاوت های بین localStorage و sessionStorage چیا هستن؟	۴۲
	چطوري به web-storage دسترسی پیدا میکنی؟	۴۳
	چه متدهایی روی session-storage قابل استفاده هستن؟	۴۴
	رخداد storage چیه و چطوري ازش استفاده میکنیم؟	۴۵
	چرا به web-storage نیاز داریم؟	۴۶
	چطوري مىتونيم پشتيبانی از web-storage تو سط مرورگر رو بررسی کنیم؟	۴۷
	چطوري مىتونيم پشتيبانی از web-worker تو سط مرورگر رو بررسی کنیم؟	۴۸
	یه مثال از web-workerها میتونی بزنی؟	۴۹
	محدودیت های web-workerها روی DOM چیا هستن؟	۵۰
	Promise چیه؟	۵۱
	چرا به promise نیاز داریم؟	۵۲

ردیف	سوال	صفحه
۵۳	سه تا وضعیت ممکن برای یه promise چیا هستن؟	
۵۴	توابع callback چی هستن؟	
۵۵	چرا به توابع callback نیاز داریم؟	
۵۶	چرا جهنم توابع callback-hell یا 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 دارند؟	
۶۱	اصلی‌ترین قوانین promise‌ها چیا هستن؟	
۶۲	Callback چطوری رخ میده؟	
۶۳	زنجیره promise‌ها چیه؟	
۶۴	کاربرد متدهای 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 حالت strict را فعال کنیم؟	
۶۹	هدف از عملگر نقطی دوتابعی (!!) چیه؟	
۷۰	هدف از عملگر delete چیه؟	
۷۱	عملگر typeof چیکار می‌کنه؟	
۷۲	undefined چیه و چه زمانی undefined می‌گیریم؟	
۷۳	null چیه؟	

ردیف	سوال	صفحه
۷۴	تفاوت‌های بین null و undefined چیا هستن؟	
۷۵	eval چیه؟	
۷۶	تفاوت‌های بین window و document چیا هستن؟	
۷۷	توی جاواسکریپت چطوری می‌تونیم به history دسترسی داشته باشیم؟	
۷۸	انواع داده‌های جاواسکریپت کدوما هستن؟	
۷۹	NaN چیه و چیکار می‌کنه؟	
۸۰	تفاوت‌های بین undefined و undeclared چیا هستن؟	
۸۱	کدوم متغیرها عمومی هستن؟	
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 object و native، host و user چیا هستن؟	
۹۲	کدوم ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده می‌شون؟	
۹۳	مزایا و معایب استفاده از promise به جای callback چیا هستن؟	

صفحه	سوال	ردیف
	تفاوت‌های بین property و attribute روی DOM چیا هستن؟	۹۴
	سیاست same-origin چیه؟	۹۵
	هدف استفاده از void چیه؟	۹۶
	جاواسکریپت یه زبان تفسیری هست یا کامپایلری؟	۹۷
	آیا جاواسکریپت یه زبان حساس به بزرگی و کوچکی(case-sensitive) حروف است؟	۹۸
	ارتباطی بین JavaScript و Java وجود داره؟	۹۹
	Event‌ها چی هستن؟	۱۰۰
	کی جاواسکریپت رو ساخته؟	۱۰۱
	هدف از متد preventDefault چیه؟	۱۰۲
	کاربرد متد stopPropagation چیه؟	۱۰۳
	مراحلی که موقع استفاده از event-handler return false تویی یه رخ میده چیا هستن؟	۱۰۴
	BOM چیه؟	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
	Event-delegation چیه؟	۱۰۹
	ECMAScript چیه؟	۱۱۰
	JSON چیه؟	۱۱۱
	قوانين فرمت JSON کدوما هستن؟	۱۱۲
	هدف از متد JSON.stringify چیه؟	۱۱۳

صفحه	سوال	ردیف
	چطوری می‌تونیم یه رشته JSON (string) رو تجزیه کنیم؟	۱۱۴
	چرا به JSON نیاز داریم؟	۱۱۵
	PWA‌ها چی هستن؟	۱۱۶
	هدف از متدها setTimeout و clearInterval چیه؟	۱۱۷
	هدف از متدها setInterval و clearInterval چیه؟	۱۱۸
	توی جاوااسکریپت، چطوری می‌شه به یه صفحه جدید redirect انجام داد؟	۱۱۹
	چطوری بررسی می‌کنین که یه string شامل یه substring هست یا نه؟	۱۲۰
	توی جاوااسکریپت، چطوری مقدار یه آدرس email رو اعتبارسنجی می‌کنین؟	۱۲۱
	چطوری می‌تونیم مقدار آدرس url جاری رو بخونیم؟	۱۲۲
	ویژگی‌های مختلف url روی object history مربوط به کدوما هستن؟	۱۲۳
	توی جاوااسکریپت چطوری می‌تونیم مقدار یه query-string رو بخونیم؟	۱۲۴
	چطوری می‌تونیم بررسی کنیم که آیا یه پرآپرتی روی آبجکت وجود داره یا نه؟	۱۲۵
	ری روی یه object حلقه میزني؟	۱۲۶
	چطوری تست می‌کنی که یه object خالیه؟	۱۲۷
	arguments چیه؟	۱۲۸
	چطوری حرف اول یه رشته رو به حرف بزرگ تبدیل می‌کنی؟	۱۲۹
	مزایا و معایب حلقه for چیا هستن؟	۱۳۰
	چطوری تاریخ جاری رو توی جاوااسکریپت نشون میدی؟	۱۳۱
	چطوری دو تا date object رو با هم مقایسه می‌کنی؟	۱۳۲
	چطوری بررسی می‌کنی که یه رشته با یه رشته دیگه شروع می‌شه؟	۱۳۳
	چطوری یه رشته رو trim می‌کنی؟	۱۳۴

ردیف	سوال	صفحه
۱۳۵	توى جاواسكريپت چطوري مىتونيم يه زوج مرتب از key يه valueها بسازيم؟	
۱۳۶	آيا عبارت '!-' عملگر خاصی هست؟	
۱۳۷	چطوري مىتونيم به متغيرها مون مقادير اوليه بدیم؟	
۱۳۸	چطوري مىتونيم متن های چند خطی درست کنیم؟	
۱۳۹	مدل app-shell چیه؟	
۱۴۰	چطوري مىتونيم روی يه تابع property اضافه کنیم؟	
۱۴۱	چطوري مىتونيم تعداد پارامترهاي ورودی يه تابع رو به دست بیاریم؟	
۱۴۲	Polyfill چیه؟	
۱۴۳	عبارات Break و continue چی هستن؟	
۱۴۴	توى جاواسكريپت labelها چیكار مىکنن؟	
۱۴۵	مزایای declare کردن متغيرها در اوایل کد چیه؟	
۱۴۶	مزایای مقداردهی اوليه متغيرها چیه؟	
۱۴۷	روش توصیه شده برای ایجاد object چیه؟	
۱۴۸	چطوري مىتونيم آرایه JSON تعریف کنیم؟	
۱۴۹	چطوري مىتونيم اعداد تصادفي تولید کنیم؟	
۱۵۰	مىتونی يه تابع تولید اعداد تصادفي توى يه بازه مشخص بنویسی؟	
۱۵۱	Tree-shaking چیه؟	
۱۵۲	دلایل نیاز به tree-shaking کدوما هستن؟	
۱۵۳	آيا استفاده از eval توصیه مىشه؟	
۱۵۴	Regular-Expression چیه؟	
۱۵۵	متدهای رشته که روی Regular-expression مجاز هستن کدوماست؟	

ردیف	سوال	صفحه
۱۵۶	توى Regex بخش modifiers چیکار مى کنه؟	
۱۵۷	پترن های regular-expression چیه؟	
۱۵۸	آبجکت RegExp چیه؟	
۱۵۹	چطوری روی یه رشته دنبال یه پترن RegExp مى گرددی؟	
۱۶۰	هدف از متدها exec چیه؟	
۱۶۱	چطوری استایل های یه المنت HTML رو تغییر میدی؟	
۱۶۲	نتیجه عبارت $3' + 2 + 1$ چی می شه؟	
۱۶۳	عبارة debugger چیکار مى کنه؟	
۱۶۴	هدف از متدها breakpoint چیه؟	
۱۶۵	آیا می تونیم از عبارت های رزرو شده در تعریف identifier ها (اسم متغیر، کلاس و ...) استفاده کنیم؟	
۱۶۶	چطوری تشخیص بدیم که یه مرورگر mobile هست یا نه؟	
۱۶۷	چطوری بدون Regex تشخیص بدیم که یه مرورگر mobile هست یا نه؟	
۱۶۸	چطوری طول و عرض یه تصویر رو با جاواسکریپت به دست میاری؟	
۱۶۹	چطوری درخواست های synchronous HTTP بزنیم؟	
۱۷۰	چطوری درخواست های asynchronous HTTP بزنیم؟	
۱۷۱	چطوری یه تاریخ رو به یه تاریخ در timezone دیگه تبدیل کنیم؟	
۱۷۲	چه property هایی برای اندازه گیزی سایز window به کار میره؟	
۱۷۳	عملگر شرطی سه گانه توى جاواسکریپت چیه؟	
۱۷۴	آیا میشه روی عملگر شرطی زنجیره شرطها رو اعمال کرد؟	
۱۷۵	روش های اجرای جاواسکریپت بعد از لود شدن صفحه کدوما هستن؟	
۱۷۶	تفاوت های بین proto و prototype کدوما هستن؟	

ردیف	سوال	صفحه
۱۷۷	میتوانی یه مثال از زمانی که واقعا به سمیکولون(;) نیاز هست بزنی؟	
۱۷۸	متد freeze چیکار می‌کنه؟	
۱۷۹	هدف از متد freeze چیه؟	
۱۸۰	چرا به متد freeze نیاز داریم؟	
۱۸۱	چطوری می‌تونیم زبان ترجیحی یه مرورگر رو تشخیص بدیم؟	
۱۸۲	چطوری می‌تونیم حرف اول همه کلمات یه رشته رو به حرف بزرگ تبدیل کنیم؟	
۱۸۳	چطوری می‌شه تشخیص داد که جاواسکریپت به صفحه وب غیرفعال شده؟	
۱۸۴	عملگرهای پشتیبانی شده توسط جاواسکریپت کدوما هستن؟	
۱۸۵	پارامتر rest چیکار می‌کنه؟	
۱۸۶	اگه پارامتر rest رو به عنوان آخرین پارامتر استفاده نکنیم چی می‌شه؟	
۱۸۷	عملگرهای منطقی باینری توی جاواسکریپت کدوما هستن؟	
۱۸۸	عملگر spread چیکار می‌کنه؟	
۱۸۹	چطوری تشخیص میدی که یه آبجکت freeze شده یا نه؟	
۱۹۰	چطوری بررسی کنیم که دو تا مقدار(شامل آبجکت) با هم برابر یا نه؟	
۱۹۱	هدف از متد is روی object چیه؟	
۱۹۲	چطوری property های یه object رو به یه object دیگه کپی می‌کنی؟	
۱۹۳	کاربردهای ممتد assign چیه؟	
۱۹۴	آبجکت proxy چیه؟	
۱۹۵	هدف از متد seal چیه؟	
۱۹۶	کاربردهای متد seal چیه؟	
۱۹۷	تفاوت‌های بین متدهای seal و freeze چیا هست؟	

صفحه	سوال	ردیف
	چطوری تشخیص میدی که یه آبجکت seal شده یا نه؟	۱۹۸
	چطوری کلید و مقدارهای enumerable رو به دست میاری؟	۱۹۹
	تفاوت‌های بین متدهای Object.entries و Object.values چیا هست؟	۲۰۰
	چطوری لیست کلیدهای یه object رو بدست میاری؟	۲۰۱
	چطوری یه prototype با object درست می‌کنی؟	۲۰۲
	چیه؟ WeakSet	۲۰۳
	تفاوت‌های بین Set و WeakSet کدوما هستن؟	۲۰۴
	لیست متدهایی که رو WeakSet قابل استفاده هستن رو می‌تونی بگی؟	۲۰۵
	چیه؟ WeakMap	۲۰۶
	تفاوت‌های بین Map و WeakMap کدوما هستن؟	۲۰۷
	لیست متدهایی که رو WeakMap قابل استفاده هستن رو می‌تونی بگی؟	۲۰۸
	هدف از متدهای uneval چیه؟	۲۰۹
	چطوری یه encode URL رو می‌کنی؟	۲۱۰
	چطوری یه decode URL رو می‌کنی؟	۲۱۱
	چطوری محتوای یه صفحه رو پرینت می‌گیری؟	۲۱۲
	تفاوت‌های بین eval و uneval چیا هستن؟	۲۱۳
	تابع anonymous چیه؟	۲۱۴
	تفاوت تقدم بین متغیرهای local و global چطوریه؟	۲۱۵
	چیکار می‌کنن؟ accessor جاواسکریپت	۲۱۶
	چطوری روی Object یه constructor یه مقدار تعریف می‌کنی؟	۲۱۷
	تفاوت‌های بین get و defineProperty چیا هست؟	۲۱۸

ردیف	سوال	صفحه
۲۱۹	مزایای استفاده از Getter و Setter چیه؟	
۲۲۰	می‌توینیم getter و setter را با استفاده از متدهای defineProperty تعریف کنیم؟	
۲۲۱	هدف استفاده از switch-case چیه؟	
۲۲۲	چه قواعدی برای استفاده از switch-case باید رعایت بشه؟	
۲۲۳	نوع داده‌های primitive کدوماً هستن؟	
۲۲۴	روش‌های مختلف دسترسی به object‌های property کدوماً هستن؟	
۲۲۵	قوانين پارامترهای توابع کدوماً هستن؟	
۲۲۶	آبجکت error چیه؟	
۲۲۷	چه موقعی خطای syntax دریافت می‌کنیم؟	
۲۲۸	عنوان خطاهای مختلف که روی error-object برمیگردن کدوماً هستن؟	
۲۲۹	عبارات مختلف که در هنگام مدیریت error استفاده می‌شون کدوماً هستن؟	
۲۳۰	دو نوع مختلف حلقه‌ها در جاواسکریپت کدوماً هستن؟	
۲۳۱	آبجکت nodejs چیه؟	
۲۳۲	آبجکت Intl چیه؟	
۲۳۳	چطوری تاریخ و زمان رو بر اساس زبان جاری سیستم کاربر نمایش بدیم؟	
۲۳۴	آبجکت Iterator چیه؟	
۲۳۵	حلقه‌های synchronous(همزمان) چطوری کار می‌کنند؟	
۲۳۶	آبجکت Event-loop چیه؟	
۲۳۷	آبجکت Call-stack چیه؟	
۲۳۸	آبجکت Event-queue چیه؟	
۲۳۹	آبجکت Decorator چیه؟	

صفحه	سوال	ردیف
	مقادیر موجود روی آبجکت <code>Intl</code> کدوما هستند؟	۲۴۰
	عملگر <code>Unary</code> چیه؟	۲۴۱
	چطوری المنشاهی موجود تو یه آرایه رو مرتب می‌کنی؟	۲۴۲
	هدف از تابع مرتب‌سازی موقع استفاده از متدهای <code>sort</code> چیه؟	۲۴۳
	چطوری آیتم‌های یه آرایه رو معکوس مرتب کنیم؟	۲۴۴
	چطوری حداقل و حداقل‌تر مقدار یه آرایه رو بدست بیاریم؟	۲۴۵
	چطوری حداقل و حداقل‌تر مقدار یه آرایه رو بدون استفاده از متدهای <code>Math</code> بدست بیاریم؟	۲۴۶
	عبارت خالی چیه و هدف از استفاده ازش چیه؟	۲۴۷
	چطوری <code>meta data of a module</code> یه مازول رو بدست میاری؟	۲۴۸
	عملگر <code>comma</code> چیه و چیکار می‌کنه؟	۲۴۹
	مزایای استفاده از عملگر <code>comma</code> چیه؟	۲۵۰
	چیه؟ <code>TypeScript</code>	۲۵۱
	تفاوت‌های بین <code>typescript</code> و <code>javascript</code> کدوما هستن؟	۲۵۲
	مزایای <code>typescript</code> نسبت به <code>javascript</code> چیاست؟	۲۵۳
	چیه؟ <code>object-initializer</code>	۲۵۴
	متدهای <code>constructor</code> چیه؟	۲۵۵
	اگه متدهای <code>constructor</code> رو بیش از یه بار توی کلاس بنویسیم چی می‌شه؟	۲۵۶
	چطوری متدهای <code>constructor</code> کلاس والد رو صدا بزنیم؟	۲۵۷
	چطوری <code>Object</code> یه <code>prototype</code> رو به دست میاری؟	۲۵۸
	اگه به متدهای <code>getPrototypeOf</code> رشته پاس بدیم چی می‌شه؟	۲۵۹
	چطوری <code>Object</code> یه <code>prototype</code> روی یه <code>Object</code> دیگه ستد کنیم؟	۲۶۰

ردیف	سوال	صفحه
۲۶۱	چطوری بررسی می‌کنی که یه object قابل extend هست یا نه؟	
۲۶۲	چطوری جلوی object یه extend رو بگیریم؟	
۲۶۳	روش‌های مختلف برای تبدیل یه object به object غیرقابل extend چیه؟	
۲۶۴	چطوری property‌های متعددی رو روی یه object تعریف می‌کنی؟	
۲۶۵	منظور از MEAN توی جاواسکریپت چیه؟ javascript	
۲۶۶	توی جاواسکریپت javascript چیه و چیکار می‌کنه؟ Obfuscation	
۲۶۷	چه نیازی به Obfuscate کردن داریم؟	
۲۶۸	چیه؟ Minification	
۲۶۹	مزایای minification یا کم حجم‌سازی چیه؟	
۲۷۰	تفاوت‌های بین Encryption و Obfuscation چیه؟	
۲۷۱	ابزارهای مختلف برای minification کدوما هستن؟	
۲۷۲	چطوری اعتبارسنجی فرم رو با javascript انجام میدی؟	
۲۷۳	چطوری اعتبارسنجی فرم رو بدون javascript انجام میدی؟	
۲۷۴	متدهای موجود روی DOM برای اعتبارسنجی کدوما هستن؟	
۲۷۵	مقادیر موجود روی DOM برای اعتبارسنجی کدوما هستن؟	
۲۷۶	مقادیر موجود روی input برای اعتبارسنجی کدوما هستن؟	
۲۷۷	یه مثال از استفاده ویژگی rangeOverflow می‌تونی بزنی؟	
۲۷۸	جاواسکریپت قابلیت استفاده از enum رو پیش‌فرض توی خودش داره؟	
۲۷۹	چیه؟ enum	
۲۸۰	چطوری همه property‌های یه object رو به دست بیاریم؟	
۲۸۱	چطوری get property descriptors of an object	

ردیف	سوال	صفحه
۲۸۳	گزینه‌هایی که موقع تعریف ویژگی object با descriptor داریم کدوما هستن؟	
۲۸۴	چطوری کلاس‌ها رو extend می‌کنی؟	
۲۸۵	چطوری آدرس صفحه رو بدون رفرش صفحه عوض کنیم؟	
۲۸۶	چطوری بررسی می‌کنی که یه آرایه یه مقدار مشخص رو داره یا نه؟	
۲۸۷	چطوری آرایه‌های scalar رو با هم مقایسه می‌کنی؟	
۲۸۸	چطوری می‌شه پارامترهای صفحه رو از متدهای GET گرفت؟	
۲۸۹	چطوری اعداد رو می‌شه سه رقم سه رقم جدا کرد؟	
۲۹۰	تفاوت بین javascript و java چیه؟	
۲۹۱	آیا جاواسکریپت namespace رو پشتیبانی می‌کنه؟	
۲۹۲	چطوری namespace تعریف می‌کنی؟	
۲۹۳	چطوری می‌تونیم تکه کد جاواسکریپت داخل یه iframe رو از صفحه والد صدا بزنیم؟	
۲۹۴	چطوری می‌شه اختلاف timezone رو از آبجکت date بگیریم؟	
۲۹۵	چطوری فایل‌های CSS و JS رو به شکل داینامیک بارگذاری کنیم؟	
۲۹۶	روش‌های مختلف برای پیدا کردن element‌ها توی DOM کدوما هستن؟	
۲۹۷	چیه؟ jQuery	
۲۹۸	موتور V8 جاواسکریپت چیه؟	
۲۹۹	چرا ما جاواسکریپت رو به عنوان یه زبان داینامیک می‌شناسیم؟	
۳۰۰	عملگر void چیکار می‌کنه؟	
۳۰۱	چطوری می‌شه نمایشگر موس صفحه رو به درحال لود تغییر داد؟	
	چطوری می‌شه یه حلقه بی‌نهایت درست کرد؟	

صفحه	سوال	ردیف
	چرا باید در استفاده از عبارت with تجدیدنظر کرد؟	۳۰۲
	خروجی این حلقه‌ها چی می‌شه؟	۳۰۳
	می‌تونی یه سری از ویژگی‌های ES6 رو اسم ببری؟	۳۰۴
	آیا می‌تونیم متغیرهای تعریف شده با let و const رو مجددا declare کنیم؟	۳۰۵
	آیا استفاده از const برای تعریف متغیر اونا رو immutable می‌کنه؟	۳۰۶
	آیا پارامترهای پیشفرض چی هستن؟	۳۰۷
	چطوری رشته‌های چند خطی رو توی template-literal می‌نویسیم؟	۳۰۸
	چطوری رشته‌های تودرتو چی هستن؟	۳۰۹
	چطوری رشته‌های خام چی هستن؟	۳۱۰
	چطوری destructuring با assign چیه و چطوری انجام می‌شه؟	۳۱۱
	چطوری destructuring با assign چیه و چطوری اولیه تعریف می‌شه؟	۳۱۲
	رشته‌های خام چی هستن؟	۳۱۳
	چطوری destructuring با assign چیه و چطوری انجام می‌شه؟	۳۱۴
	چطوری destructuring با assign چیه و چطوری اولیه تعریف می‌شه؟	۳۱۵
	چطوری می‌تونیم مقدار یه آرایه رو با استفاده از destructuring assignment تعویض کنیم؟	۳۱۶
	چطوری Enhanced-object-literal چی هستن؟	۳۱۷
	چیه‌های داینامیک چی هستن؟	۳۱۸
	کاربرد import‌های داینامیک چیه؟	۳۱۹
	آرایه‌های نوع‌دار (typed-arrays) چیه؟	۳۲۰
	مزایای لودر مازول‌ها چیه؟	۳۲۱

صفحه	سوال	ردیف
	چیه؟ collation	۳۲۲
	عبارت for...of چیه؟	۳۲۳
	خروجی عملگر spread روی آرایه زیر چیه؟	۳۲۴
	آیا PostMessage امنه؟	۳۲۵
	مشکلات استفاده از origin postmessage با wildcard روی چیه؟	۳۲۶
	چطوری از دریافت postMessage های ناخواسته و نامن از طرف هکرهای جلوگیری کنیم؟	۳۲۷
	می تونیم کلا postMessage ها را غیرفعال کنیم؟	۳۲۸
	آیا postMessage ها به صورت synchronous و همزمان کار می کنند؟	۳۲۹
	پارادیم زبان جاواسکریپت چیه؟	۳۳۰
	تفاوت های بین جاواسکریپت داخلی و خارجی چیه؟	۳۳۱
	آیا جاواسکریپت سریعتر از اسکریپت های سمت سرور است؟	۳۳۲
	چطوری وضعیت چک بودن یه checkbox رو بدست بیاریم؟	۳۳۳
	هدف از عملگر double-tilde چیه؟	۳۳۴
	چطوری یه کاراکتر رو به کد ASCII تبدیل کنیم؟	۳۳۵
	چیه؟ ArrayBuffer	۳۳۶
	خروجی کد زیر چی خواهد بود؟	۳۳۷
	هدف از Error-object چیه؟	۳۳۸
	هدف از EvalError-object چیه؟	۳۳۹
	خطاهایی که در حالت strict-mode رخ میدن ولی در غیر اون وجود ندارن کدامها هستن؟	۳۴۰
	آیا همه object ها دارای prototype هستن؟	۳۴۱

صفحه	سؤال	ردیف
	تفاوت‌های بین parameter و argument چیه؟	۳۴۲
	هدف از متدهای some روی آرایه‌ها چیه؟	۳۴۳
	چطوری دو یا تعداد بیشتری از آرایه‌ها را با هم ترکیب کنیم؟	۳۴۴
	تفاوت‌های بین Shallow و Deep کپی چیه؟	۳۴۵
	چطوری می‌توانیم به یه تعداد مشخص از یه رشته کپی کنیم؟	۳۴۶
	چطوری همه string های match شده با یه regular-expression را برگردانیم؟	۳۴۷
	چطوری یه رشته رو از اول یا از آخر trim کنیم؟	۳۴۸
	خروجی کنسول زیر با عملگر unary چی می‌شه؟	۳۴۹
	آیا جاوا‌سکریپت از mixin‌ها استفاده می‌کنه؟	۳۵۰
	تابع thunk چیه و چیکار می‌کنه؟	۳۵۱
	چیکار می‌کنن؟ asynchronous های thunk چیکار می‌کنند؟	۳۵۲
	خروجی فراخوانی‌های توابع زیر چی می‌شه؟	۳۵۳
	چطوری همه خطوط جدید رو از یه رشته حذف کرد؟	۳۵۴
	تفاوت بین repaint و reflow چیه؟	۳۵۵
	اگه قبل از یه آرایه عملگر نفی «!» بزاریم چی می‌شه؟	۳۵۶
	اگه دو تا آرایه رو با هم جمع ببنديم چی می‌شه؟	۳۵۷
	اگه عملگر جمع «+» روی قبل از مقادیر falsy قرار بدیم چی می‌شه؟	۳۵۸
	چطوری با استفاده از آرایه‌ها و عملگرهای منطقی می‌توانیم رشته self را تولید کنیم؟	۳۵۹
	چطوری می‌توانیم مقادیر falsy رو از آرایه حذف کنیم؟	۳۶۰
	چطوری مقادیر تکراری رو از یه آرایه حذف کنیم؟	۳۶۱

صفحه	سوال	ردیف
	چطوری هم زمان با destructuring alias کار می کنیم؟	۳۶۲
	چطوری آیتم های یه آرایه رو بدون استفاده از متدها map پیمایش کنیم؟	۳۶۳
	چطوری یه آرایه رو خالی کنیم؟	۳۶۴
	چطوری اعداد رو با تعداد رقم اعشار مشخص رند می کنیم؟	۳۶۵
	ساده ترین روش برای تبدیل آرایه به object چیه؟	۳۶۶
	چطوری یه آرایه با یه سری داده درست کنیم؟	۳۶۷
	متغیرهای موجود روی آبجکت console کدوما هستن؟	۳۶۸
	میشه پیام های کنسول رو استایل دهی کرد؟	۳۶۹
	هدف از متدها dir و console چیه؟	۳۷۰
	آیا میشه المنت های HTML رو توی console دیباگ کرد؟	۳۷۱
	چطوری میشه داده ها رو به شکل جدولی توی console نمایش بدمیم؟	۳۷۲
	چطوری میشه بررسی کرد که یه پارامتر Number هست یا نه؟	۳۷۳
	چطوری یه متن رو میتونیم به clipboard کپی کنیم؟	۳۷۴
	چطوری میشه timestamp رو بدست آورد؟	۳۷۵
	چطوری یه آرایه چند سطحی رو تک سطحی کنیم؟	۳۷۶
	ساده ترین روش برای بررسی چند شرطی چیه؟	۳۷۷
	چطوری کلیک روی دکمه برگشت مرورگر رو متوجه بشیم؟	۳۷۸
	چطوری میتونیم کلیک راست رو غیر فعال کنیم؟	۳۷۹
	object ها چی هستن؟	۳۸۰
	AJAX چیه؟	۳۸۱
	روش های مختلف مدیریت یه کد Asynchronous چیه؟	۳۸۲

صفحه	سوال	ردیف
	چطوری یه درخواست fetch رو کنسل کنیم؟	۳۸۳
	چیه؟ Speech-API	۳۸۴
	حداقل timeout توی throttling چقدره؟	۳۸۵
	چطوری میشه یه timeout صفر توی مرورگر اجرا کرد؟	۳۸۶
	چی هستن؟ event-loop	۳۸۷
	چی هستن؟ microtask	۳۸۸
	چی هستن؟ event-loop های مختلف کدوما هستن؟	۳۸۹
	هدف از queueMicrotask چیه؟	۳۹۰
	چطوری میشه از کتابخونههای جاواسکریپت توی فایل typescript استفاده کرد؟	۳۹۱
	تفاوت‌های بین observable و promise ها کدوما هستن؟	۳۹۲
	چیه؟ heap	۳۹۳
	چیه؟ event-table	۳۹۴
	چیه؟ microTask	۳۹۵
	تفاوت بین shim و polyfill چیه؟	۳۹۶
	چطوری متوجه primitive یا غیر primitive بودن یه نوع داده میشیم؟	۳۹۷
	چیه؟ babel	۳۹۸
	آیا Node.js به شکل کامل تک thread کار می‌کنه؟	۳۹۹
	کاربردهای مرسوم observable ها کدوما هستن؟	۴۰۰
	چیه؟ RxJS	۴۰۱
	تفاوت بین function-declaration و Function-constructor چیه؟	۴۰۲
	شرط Short-circuit یا اتصال کوتاه چیه؟	۴۰۳

ردیف	سوال	صفحه
۱۰۴	ساده‌ترین روش برای تغییر سایز یه آرایه چیه؟	
۱۰۵	observable چیه؟	
۱۰۶	تفاوت‌های بین تابع و کلاس‌ها چیه؟	
۱۰۷	تابع async چیه؟	
۱۰۸	چطوری خطاهای ایجاد شده هنگام استفاده از promise را کنترل کنیم؟	
۱۰۹	Deno چیه؟	
۱۱۰	توی جاواسکریپت چطوری یه object قابل پیمایش درست کنیم؟	
۱۱۱	روش مناسب برای فراخوانی تابع بازگشتی چیه؟	
۱۱۲	چطوری بررسی کنیم که یه آبجکت promise هست یا نه؟	
۱۱۳	چطوری متوجه بشیم که یا تابع با تابع constructor صدا زده شده یا نه؟	
۱۱۴	تفاوت‌های بین آبجکت argument و پارامتر rest چیه؟	
۱۱۵	تفاوت‌های بین عملگر spread و پارامتر rest چیه؟	
۱۱۶	نوع‌های مختلف generatorها کدوما هستن؟	
۱۱۷	آبجکت‌های iterable کدوما هستن؟	
۱۱۸	تفاوت‌های بین حلقه for...in و for...of چیه؟	
۱۱۹	چطوری property‌های instance و غیر instance تعريف می‌کنی؟	
۱۲۰	تفاوت‌های بین Number.isNaN و NaN کدوما هستن؟	

پیشگفتار

در ابتداء، ممنونم از شما که با خرید این کتاب بهمون کمک کردین که بتونیم قدمی در راه کمک به افراد نیازمند برداریم و با درآمد حاصل از فروش این کتاب کمکی هر چند کوچیک در راه مسئولیت اجتماعی مون برداریم، به هم دیگه کمک کنیم، با هم مهربون تر باشیم و در کنار هم پیشرفت کنیم. تشکر گرم من رو، دورادور پذیرا باشین و امیدوارم این کتاب به جهت افزایش دانش‌تون و کمک به پیشرفت شغلی‌تون کمکی کرده باشه.

كتابي که پيش‌روي شمامست، حاصل تلاش نه فقط من، بلکه چندين نفر از بهترین و حرفة‌اي ترين دوستان بنده هم هست که در اينجا به ترتيب ميزان زحمتی که متقبل شدن اسمشونو قيد مي‌کنم و کمال تشکر رو ازشون دارم:

- جعفر رضائي
- مهسا مصبح

این عزيزان هر کدام با کمک‌هاشون برای ترجمه، ویراستاري‌هاشون و حتی دل‌گرمی‌هاشون باعث شدن این مجموعه به زبان فارسي آماده بشه و به شکل چاپی بتونه به دستان شما برسه.

ماريوتك

برادر من جعفر رضائي، پلتفرم ماريوتك رو با هدف آموزش اصولي و رايگان، تاسيس کرد و من هم اين کتاب رو از مجموعه ماريوتك منتشر ميکنم. ما ماريوتك رو متعلق به همه مي‌دونيم، پس اگه بعضی تایم‌های بیکاری داري که فکر مي‌کني می‌تونی باهامون توی اين مسیر همراه باشي حتما بهم ايميل بزن. ایده‌های ماريوتك برای افزایش آگاهی و دانش تا حد امکان رايگان خواهد بود و تا به اينجا هم، تنها هزينه‌های چاپ برداشته شده و مابقی به موسسات خيريه داده شدن.

مطلوب کتاب

مطلوب اين کتاب می‌تونن تا حد بسیار خوبی دانش شما رو توی مسائل کلیدی مربوط جاواسکریپت و کتابخونه‌های پیرامون اون افزایش بدن. سوالات چالشی و کلیدی مطرح شده توی کتاب اکثرا سوالاتی هستند که توی مصاحبه‌های استخدامی پرسیده می‌شن و مسلط بودن به اوونا می‌تونه شانس موفقیت شما برای موقعیت‌های شغلی که مد نظر دارین افزایش بده. مطالب اين کتاب به دلیل ترجمه بودن تا حد زیادی قابل دستکاری نبودن و سعی شده تا حد امکان حق گردآورنده محفوظ باشه و با نسخه اصلی سورس که توسط Sudheer Jonna جمع‌آوری شده تفاوت معنایی نداشته باشه. بخشی از مطالب کتاب اصلی به خاطر قدیمی بودن منقضی شده بودن و به عنوان مترجم بخش‌های زيادي از نمونه کدها و مطالب قدیمي تصحیح شدند. در آخر، امیدوارم همیشه شاد و خندان و خوشحال باشین. مخلصیم

۱. روش‌های ایجاد objects توی جاواسکریپت چیا هستن؟

۱. سازنده آبجکت:

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

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

۲. متد ایجاد آبجکت:

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

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

۳. :Object literal syntax

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

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

۴. سازنده تابع:

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

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

۵. سازنده تابع با نمونه اولیه:

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

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

این معادل نمونه‌ای هستش که با متد 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);
```

(۶)

```
// Create a new instance using function prototype.  
var newInstance = Object.create(func.prototype)  
  
// Call the function  
var result = func.call(newInstance, x, y, z),  
  
// If the result is a non-null object then use it otherwise  
// just use the new instance.  
console.log(result && typeof result === 'object' ? result :  
newInstance);
```

:ES6 Class syntax .۶

ویژگی‌های کلاس رو برای ایجاد آبجکت‌ها معرفی می‌کنند

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

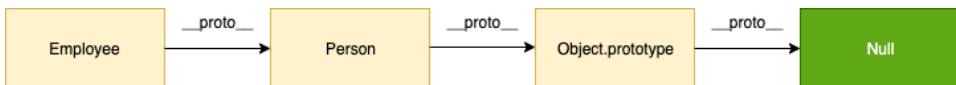
:Singleton .۷

آبجکت‌ایه 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";  
}
```

:زنجیره prototype چیه؟ .۸

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



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

متده `call`: 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?
  
```

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 می‌خوایم 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?
```

Apply و Call تقریباً قابل تعویض هستن. هر دو بلافاصله تابع فعلی رو اجرا می‌کنن. شما باید تصمیم بگیرید که ارسال آرایه در آرایه آسون تره یا فهرستی از آرگومان‌های جدا شده با کاما. می‌تونین به خاطر داشته باشیم که Call برای کاما (فهرست جدا شده) و Apply برای Array اس. در حالی که Bind یه تابع جدید ایجاد می‌کنه که «this» روی اولین پارامتر ارسال شده به «bind» تنظیم می‌شه.

۴. فرمت JSON چیه و عملیات‌های معمول بر روی آن چیا هستن؟

JSON یه قالب داده مبتنی بر متن هستش که از نحو آبجکت جاوا اسکریپت (javascript) objext syntax (Douglas Crockford) پیروی می‌کنه و توسط 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 MIME 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)
```

۵. هدف از متد slice روی آرایه‌ها چیه؟

متد **slice** عناصر انتخاب شده توى يه آرایه رو به عنوان يه آبجکت آرایه جديد برمي‌گردونه. اين عناصر رو از اولين آرگومان داده شده انتخاب مى‌کنه و با آرگومان پايانى و اختياری داده شده بدون در نظر گرفتن آخرین عنصر به پايان مى‌رسونه. اگه آرگومان دوم رو حذف کنيم تا آخر آرایه همه عناصر رو انتخاب مى‌کنه. چند تا مثال در مورد اين متد اينجا 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: متد Slice آرایه اصلی رو تغيير نمیده ولی يه زيرمجموعه به عنوان يه آرایه جديد برمي‌گردونه

۶. هدف از متد splice روی آرایه‌ها چیه؟

متد **splice** برای اضافه کردن به آرایه یا حذف از اون استفاده می‌شه و مورد حذف شده رو برمي‌گردونه. آرگومان اول موقعیت آرایه رو برای درج یا حذف مشخص می‌کنه در حالی که آرگومان اختياری دوم تعداد عناصر حذف شده رو مشخص می‌کنه. هر آرگومان اضافه‌ای به آرایه اضافه می‌شه. چند تا مثال در اين مورد اينجا 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: متد Splice آرایه اصلی رو اصلاح می‌کنه و آرایه حذف شده رو برمي‌گردونه.

۷. تفاوت متدهای splice و slice چیا هستن؟

Splice	Slice
آرایه اصلی را تغییر نمیده (تغییر پذیر)	آرایه اصلی را تغییر نمیده (تغییر ناپذیر)
عناصر حذف شده را به عنوان آرایه برمی گردونه	زیر مجموعه آرایه اصلی را برمی گردونه
برای درج عناصر به آرایه یا حذف از اون استفاده می شه	برای انتخاب عناصر از آرایه استفاده می شه

۸. تفاوت های Map و Object چیا هستن؟

آبجکت ها شبیه به نقشه ها (Maps) هستن از این جهت که هردو بهمون این امکتن رو میدن که کلیدها رو روی مقادیر تنظیم کنیم، مقادیر رو بازیابی کنیم، کلیدها رو حذف کنیم و ببینیم چیزی توی یه کلید ذخیره شده یا نه. به همین دلیل از آبجکت ها به عنوان نقشه ها (Maps) در طول تاریخ استفاده شده. اما تفاوت های مهمی وجود داره که استفاده از نقشه (map) رو توی موارد خاص ترجیح میده.

۱. کلیدهای یه آبجکت رشته ها و نمادها هستن، در حالی که برای نقشه مقادیر مختلفی میتونه وجود داشته باشه که شامل توابع، آبجکت ها و هر نوع اولیه دیگه ای می شه.
۲. کلیدهای نقشه (map) مرتب میشن در حالی که کلیدهای اضافه شده به آبجکت اینطوری نیستن. بنابراین موقع تکرار روی اون، object map کلیدها رو به ترتیب اضافه شدنشون برمی گردونه.
۳. اندازه map رو می تونیم به راحتی با ویژگی سایز بدست بیاریم، در حالی که تعداد خصوصیات یه آبجکت باید به صورت دستی حساب بشه.
۴. نقشه قابل تکراره و میتوانه مستقیما تکرار بشه، در حالی که تکرار روی یه آبجکت مستلزم بدست آوردن کلیدهای اون به روشنی خاص و تکرار روی اونهاست.
۵. آبجکت یه نمونه اولیه (prototype) داره، بنابراین کلیدهای پیش فرض توی map وجود داره که اگه دقت نکنیم ممکنه با کلیدهایمون برخورد کنه. از زمان ES5 می تونیم با استفاده از (map = Object.create(null))، این قضیه رو دور بزنیم ولی بهندرت این کار انجام می شه.

۶. نقشه ممکنه توی سناریوهای شامل جمع و حذف مکرر جفت کلیدها عملکرد بهتری داشته باشه

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

جاواسکریپت مقایسه برابری سخت (`=`) و تبدیل نوع (`==`, `!=`) رو فراهم می‌کنه. عملگرهای سختگیرانه نوع متغیر رو در نظر می‌گیرند، در حالی که عملگرهای غیر دقیق، اصلاح/تبدیل نوع رو بر اساس مقادیر متغیرها انجام می‌دهند. اپراتورهای سختگیر از شرایط زیر برای انواع مختلف پیروی می‌کنن.

۱. دو رشته زمانی کاملاً برابر هستن که توالی کاراکترهای یکسان، طول یکسان و کاراکترهای مشابه در موقعیت‌های متناظر داشته باشن.

۲. دو عدد زمانی که از نظر عددی مساوی باشن کاملاً برابر هستند. یعنی داشتن مقدار عددی یکسان.

دو مورد خاص در این مورد وجود داره،

۳. NaN با هیچ چیز از جمله NaN برابر نیست.

۴. صفرهای مثبت و منفی با هم برابرند.

۵. اگه هر دو درست یا نادرست باشن، دو عملوند بولین کاملاً برابر هستند.

۶. اگه دو شیء به یه شیء اشاره کنن کاملاً برابر هستند.

۷. انواع Null و Undefined با === برابر نیستن، بلکه با == برابر هستند. یعنی

`null undefined --> true` اما `null=undefined --> false`

بعضی از مثال‌هایی که موارد بالا رو پوشش میده،

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

۱۰. توابع arrow-function یا lambda چی هستن؟

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

۱۱. یه تابع first-class چجور تابعیه؟

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

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

۱۲. یه تابع first-order چجور تابعیه؟

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

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

۱۳. یه تابع higher-order چجور تابعیه؟

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

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

۱۴. یه تابع unary چجور تابعیه؟

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

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

۱۵. توابع 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 میگیم. Currying از نام یه ریاضی‌دان به اسم Haskell Curry گرفته شده. با استفاده از Currying در واقع یه تابع رو به یه تابع currying 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
```

تابع Curried برای بهبود قابلیت استفاده مجدد کد و ترکیب عملکردی عالی هستن.

۱۶. چه توابعی pure هستن؟

تابع خالص تابعیه که مقدار برگشتیش فقط توسط آرگومان‌هاش تعیین می‌شه بدون هیچ side effect یا عوارض جانبی. برای 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 توابع خالص و توابع ناخالص بزنیم.

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

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

بر اساس تکه کدهای بالا، تابع push با تغییر روی آرایه و برگرداندن شماره ایندکس `push` که مستقل از مقدار پارامتر هستش، یه تابع ناخالص به حساب می‌اد. در حالی که از یه طرف متدهای آرایه رو می‌گیره و اونو با یه آرایه دیگه ترکیب می‌کنه و یه آرایه کاملاً جدید و بدون هیچ عوارض جانبی تولید می‌کنه. همچنین مقدار برگشتی با آرایه قبلی ترکیب شده هستش.

به یاد داشته باشیم که توابع خالص مهم هستند چون اونا تست واحد رو بدون هیچ گونه عوارض جانبی و بدون نیاز به تزریق وابستگی ساده می‌کنن. اونا همچنین از اتصال محکم جلوگیری می‌کنن و با نداشتن عوارض جانبی، شکستن برنامه شما رو سخت تر می‌کنن. این اصول در کنار هم قرار می‌گیرند **تغییرناپذیری** مفهوم ES6 با ارجحیت به `const` نسبت به استفاده `let`.

۱۷. هدف از کلمه کلیدی let چیه؟

دستور `let` یه متغیر محلی **block scope** تعریف 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 تو ش 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)

```

۱۸. تفاوت‌های کلمات کلیدی `let` و `var` چیا هستن؟

تفاوت‌ها رو توی جدول زیر می‌بینیم
: var

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

:let

۱. به عنوان بخشی از ES6 معرفی شده
 ۲. محدود به scope یا دامنه هستش
 ۳. Hoist شده ولی مقدارههی اولیه نمی‌شه
- بیان 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)
}

```

۱۹. دلیل انتخاب کلمه کلیدی `let` چیه؟

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

۲۰. چطوری می‌تونیم توی بلوک مربوط به switch بدون دریافت خطا متغیر تعريف کنیم؟

اگه بخوایم متغیرها رو مجدداً توی یه بلاک switch تعريف کنیم، این کار باعث خطای شه چون در واقع فقط یه بلاک وجود داره. برای 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 خطای نحوی می‌توانیم یه بلاک تو در تو داخل case ایجاد کنیم و یه محیط واژگانی دارای محدوده بلاک جدید ایجاد کنیم.

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

۲۱. Temporal-Dead-Zone چیه؟

رفتاری توی جاواسکریپته که موقع تعریف متغیر با کلمات کلیدی let و const رخ میده، نه با کلمه کلیدی var. توی اکماسکریپت ۶، دستیابی به متغیر let قبل از تعریفش (توی scope خودش) باعث خطای reference میشه. فاصله زمانی ایجاد اون، بین ایجاد اتصال متغیر و تعریف اون، منطقه Temporal Dead Zone هستش. بیاین با یه مثال ببینیم،

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

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

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

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

دلیل اصلی استفاده از IIFE بست آوردن حریم خصوصی دادههاست، چون محیط خارجی به متغیرهایی که توی IIFE تعریف شده دسترسی نداره. برای مثال، اگه سعی کنیم با IIFE به متغیرها دسترسی پیدا کنیم این خطا رو میگیریم،

```
(function ()
{
    var message = "IIFE";
    console.log(message);
})
();

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

۲۳. مزایای استفاده از module‌ها چیه؟

استفاده از مازول‌ها مزایای زیادی دارد،

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

۲۴. Memoization چیه؟

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

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

۲۵. چیه؟ Hoisting

یه مکانیسم جاوااسکریپتیه 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 مقدارههی اولیه اونا رو. بیاین یه مثال ساده از hoist کردن متغیرها بزنیم،

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

ترجمه کد بالا اینطوری میشه

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

۲۶. ES6 چی هستن؟ Class

در ES6، کلاس‌های جاوااسکریپت عمدهاً قند نحوی بر وراثت مبتنی بر نمونه اولیه جاوااسکریپتیه.

برای مثال، وراثت مبتنی بر نمونه اولیه که در عبارت تابع به صورت زیر نوشته شده.

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

درحالیکه کلاس‌های ES6 به جای اونا و به شکل ساده‌تری تعریف یشن:

```

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

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

```

۲۷. Closure ها چیا هستن؟

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

۱. دامنه رو جایی تعریف میکنیم که متغیرها بین کلی براکت‌های اون تعریف شده باشه
 ۲. متغیرهای تابع بیرونی
 ۳. متغیرهای محلی
- بیاین یه مثال راجع به مفهوم کلاژور بزنیم

```

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

```

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

۲۸. Module ها چیا هستن؟

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

۲۹. چرا به module‌ها نیاز داریم؟

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

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

۳۰. توی جاواسکریپت scope چیه و چیکار می‌کنه؟

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

۳۱. چیه service-worker؟

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

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

مستقیماً نمی‌توان به DOM دسترسی پیدا کنه، اما می‌توانه با پاسخ به پیام‌های ارسالی از طریق رابط postMessage با صفحاتی که کنترل می‌کنه ارتباط برقرار

کنه و اين صفحات ميتوانن DOM رو دستکاري کنن.

۳۳. چطوری ميتوnim بین ریست شدن های service-worker داده های مورد نظرمون رو مجدد استفاده کنيم؟

مشکلی که توی service worker وجود داره اينه که در صورت عدم استفاده خاتمه پیدا میکنه و در صورت نياز بعدی دوباره راه اندازی میشه، بنابراین نميتوnim به state سراسری توی نگهدارنده های onmessage و onfetch یه service worker اعتماد کنيم. تو اين حالت service worker برای تداوم کار و استفاده مجدد موقع شروع مجدد، به indexedDB API دسترسی خواهد داشت.

۳۴.IndexedDB چیه؟

IndexedDB یه API سطح پايین برای ذخیره client-side یا سمت کاربر توی مقادير بيشتری از داده ساخت يافته شامل فایلها و حبابها هستش. اين API از index آها برای فعال کردن جستجوهای با کارايی بالا توی اين دادهها استفاده میکنه.

۳۵. Web-storage چیه؟

web storage یه API هستش که مکانيسمی رو فراهم میکنه که مرورگرها ميتوزن مقدار و کليد رو ب صورت محلی توی مرورگر کاربر ذخیره کنن، ب روشی کاملاً قابل درک نسبت به استفاده ا کوکیها. فضای ذخیره سازی وب دو مکانيزم ر برای ذخیره اطلاعات روی مشتری فراهم میکنه.

۱. **Local storage:** داده ها رو برای مبدا فعلی و بدون تاریخ انقضا ذخیره میکنه.

۲. **Session storage:** داده ها رو برای يه جلسه ذخیره میکنه و با بسته شدن

تب مرورگر داده ها از بين ميرن.

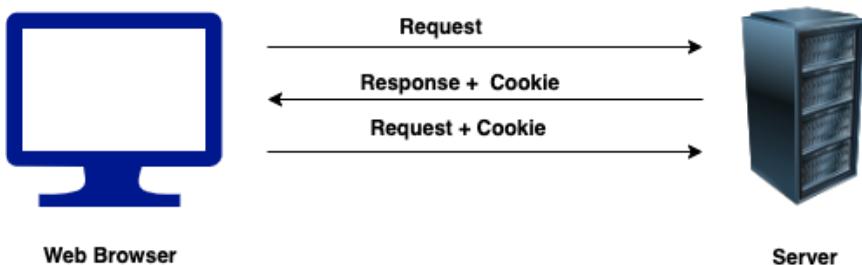
۳۶. Post-message چیه؟

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

۳۷. چیه؟ Cookie

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. برای مثال می‌تونیم یه کوکی با نام کاربری مثل زیر ایجاد کنیم،

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



۳۸. چرا به cookie نیاز داریم؟

از کوکی‌ها برای به خاطر سپردن اطلاعات مربوط به مشخصات کاربر (مانند نام کاربری) استفاده می‌شه. در اصل شامل دو مرحله هستش،
۱. وقتی که کاربر از یه صفحه وب بازدید می‌کنه، مشخصات کاربر می‌تونه توی یه کوکی ذخیره بشه.
۲. دفعه بعد که کاربر از صفحه بازدید کرد، کوکی مشخصات کاربر رو به خاطر می‌اره.

۳۹. گزینه‌های قابل تنظیم توی cookie چیا هستن؟

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

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

۲. به طور پیش فرض، کوکی به صفحه فعلی تعلق داره. اما با استفاده از پارامتر path
می‌تونیم به مرورگر بگیم که کوکی متعلق به چه مسیری هستش.

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

۳. چطوری می‌شه یه cookie رو حذف کرد؟

با تنظیم تاریخ انقضا به عنوان تاریخ گذشته می‌تونیم کوکی رو حذف کنیم. تو این حالت
نیازی به تعیین مقدار کوکی نیست.
برای مثال، می‌تونیم 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 مرورگرها تا زمانی که پارامتر مسیر رو تعیین نکنیم اجازه حذف کوکی رو نمیدن.

۴. تفاوت‌های بین session-storage و cookie، local-storage چیا هستن؟

تفاوت‌های بین کوکی، لوکال استوریج و سشن استوریج اینا هستن:

Session storage	Local storage	Cookie	ویژگی
فقط سرویس گیرنده	فقط سرویس گیرنده	هردو سرویس دهنده و سرویس گیرنده	قابل دسترسی از طرف سرویس گیرنده یا سرور

Session storage	Local storage	Cookie	ویژگی
تا وقتی که تب بسته نشده	تا وقتی که حذف باشے	پیکربندی شده با استفاده از گزینه اکسپایر	طول عمر
پشتیبانی نمی شه	پشتیبانی نمی شه	پشتیبانی می شه	پشتیبانی از SSL
5MB	5MB	4KB	حداکثر اندازه داده

۴۲. تفاوت های بین sessionStorage و localStorage چیا هستن؟

لوکال استوریج همون سشن استوریج هستش اما داده ها با بستن و دوباره باز کردن مرورگر همچنان حفظ می شه (تاریخ انقضا نداره) در حالی که سشن استوریج داده ها رو با بستن پنجره مرورگر پاک می کنه.

۴۳. چطوری به Web-storage دسترسی پیدا می کنی؟

شی window به ترتیب ویژگی های WindowLocalStorage و WindowSessionStorage را که دارای ویژگی های localStorage (window.localStorage) و sessionStorage (window.sessionStorage) هستن رو پشتیبانی می کنه. این خصوصیات نمونه ای از شی Storage رو ایجاد می کنه که از طریق اون می شه موارد داده رو برای یه دامنه خاص و نوع ذخیره سازی (session یا محلی) تنظیم، بازیابی و حذف کرد. برای مثال، می تونیم روی اشیای ذخیره سازی محلی مثل زیر بخونیم و بنویسیم

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

۴۴. چه متدهایی روی session-storage قابل استفاده هستن؟

متدهایی رو برای 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();
```

۴۵. رخداد storage چیه و چطوری ازش استفاده می‌کنیم؟

رویدادی storageEvent هستش که با تغییر مکان ذخیره‌سازی در متن سند دیگهای فعال می‌شه. در حالی که خاصیت ذخیره‌سازی یه EventHandler برای پردازش رویدادهای ذخیره‌سازی‌هه.

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

بیاین برای مثال استفاده از رویداد onstorage رو بینیم که کلید ذخیره و مقادیر او نو ثبت می‌کنه

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

۴۶. چرا به web-storage نیاز داریم؟

فضای ذخیره سازی وب از امنیت بیشتری برخورداره و مقدار زیادی داده می‌توان به صورت محلی ذخیره 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.

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

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

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

۱۴۸. 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..  
}
```

۱۴۹. یه مثال از **web-worker** 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 می‌شود.
۱. 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 رو از web worker دریافت کنیم

```

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

```

۱. به پایان رسوندن یه web Worker می‌توانیم این را stop کنیم. برای خاتمه دادن به گوش دادن به پیام‌ها می‌توانیم از دستور terminate استفاده کنیم.

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

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

```
w = undefined;
```

۵۰. محدودیت‌های web-worker روی DOM چیا هستن؟

WebWorker ها به اشیا جاوااسکریپت دسترسی ندارن چون توانی یه فایل خارجی تعریف شدن.

Window object .۱

Document object .۲

Parent object .۳

چیه Promise .۵۱

یه آبجکت ایه که ممکنه در آینده یه مقدار واحد رو با یه مقدار حل شده یا به دلیل حل نشدنش (مثلاً خطای شبکه) تولید کنه. تو یهی از ۳ حالت ممکن خواهد بود:

انجام شده، رد شده، یا در انتظار.

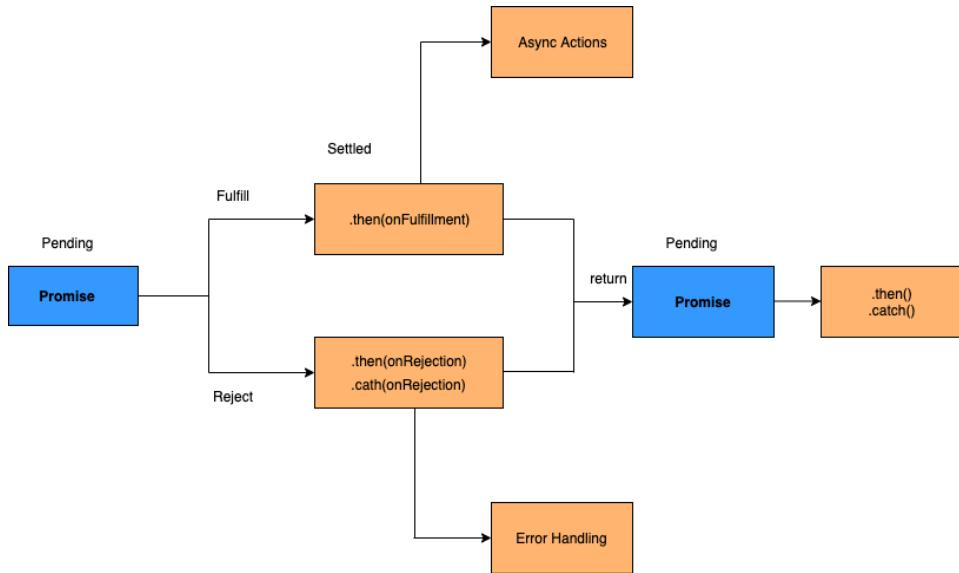
برای ساختن Promise ها از سینتکس زیر استفاده می‌شود

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

استفاده از پرمیس‌ها رو هم باهم ببینیم

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

جریان عمل یه پرمیس هم به این شکل انجام میشه:



۵۲. چرا به promise نیاز داریم؟

برای رسیدگی به عملیات ناهمزمان استفاده می‌شود. اونا با کاهش جهنم Promises و نوشتن کد پاک‌کننده، یه رویکرد جایگزین برای callback

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

پرمیس‌ها سه حالت دارند:

۱. این حالت اولیه Promise قبل از شروع عملیاته **Pending**.
۲. این حالت نشون میده که عملیات مشخص شده تکمیل شده. **Fulfilled**.
۳. این حالت نشون میده که عملیات کامل نشده. تو این حالت یه مقدار خطا داده خواهد شد. **Rejected**.

۵۴. توابع callback چی هستن؟

تابع callback تابعیه که به عنوان آرگومان به تابع دیگری منتقل می‌شود. این تابع در داخل تابع خارجی برای تکمیل یه عمل فراخوانی می‌شود.

بیاین یه مثال ساده از نحوه استفاده از تابع callback بزنیم

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

۵۵. چرا به توابع callback نیاز داریم؟

callback ها مورد نیاز هستن چون جاواسکریپت یه زبان ایونت محوره. این به این معنیه که به جای منتظر موندن برای جواب جاواسکریپت در حین گوش دادن به ایونت های دیگه اجرا می شه.

بیاین مثال رو با اولین تابع callback یه API (setTimeout) شبیه سازی شده توسط تابع بعدی که پیام رو ثبت می کنه، بیاریم.

```
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
```

همونطور که از خروجی می شه فهمید، جاواسکریپت منتظر جواب اولین تابع نبود و بلوک کد باقی مونده اجرا شد. بنابراین از callback ها به گونه ای استفاده می شه تا مطمئن شیم که کد خاصی تا موقعی که اجرای کد دیگه تموم نشده، اجرا نمی شه.

۵۶.Callback-hell یا جهنم توابع callback چیه؟

Callback Hell یه ضد الگو با چندین تماس تو در توعه که خوندن کد و دیباگ رو در موقع برخورد با منطق سخت میکنه. جهنم callback شبیه کد زیره

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

۵۷. (Server-sent-events(SSE) چیه؟

event ارسال شده توسط سرور (SSE) یه فناوری فشار سروه که به مرورگر امکان میده به روزرسانی های خودکار رو از طریق اتصال HTTP بدون استفاده از نظرسنجی دریافت کنه. اینا یه کanal ارتباطی یه طرفه هستن - event ها فقط از سروری به مشتری دیگر منتقل میشن این در به روزرسانی های فیسبوک / تویتر، به روزرسانی قیمت سهام، فیدهای خبری و غیره استفاده شده.

۵۸. چطوری میتونیم اعلان های server-sent-event رو دریافت کنیم؟

شی EventSource برای دریافت اعلان های ارسال شده از سرور استفاده میشه. برای مثال، میتونین پیام هایی رو از سرور مثل مثال زیر دریافت کنین.

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

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

می‌توانیم قبل از استفاده از event‌های ارسال شده توسط مرورگر مانند زیر، پشتیبانی مرورگر رو انجام بدین.

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

۶۰. کدام توابع روی SSE وجود دارند؟

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

Description	Event
موقعی که اتصال به سرور باز می‌شود استفاده می‌شود	onopen
این event زمانی استفاده می‌شود که پیامی دریافت شد	onmessage
زمانی اتفاق می‌افتد که خطایی رخ بده	onerror

۶۱. اصلی‌ترین قوانین promise چیا هستن؟

۱. پرمیس آبجکت‌ایه که متده «then» سازگار با استانداره رو ارائه می‌کنه
۲. یه پرمیس معلق ممکنه به حالت تحقق یافته یا رد شده تبدیل شه
۳. پرمیس تمام شده یا رد شده حل و فصل می‌شه و نباید به حالت دیگری تبدیل شه.
۴. پس از اتمام قول، ارزش اون نباید تغییر کنه.

۶۲. توى callback چطوری رخ میده؟

می‌توانیم یه پاسخ تماس رو در داخل یه تماس دیگر قرار بدم تا اقدامات رو به صورت متوالی یکی یکی انجام بدم. این به عنوان callbacks در callbacks شناخته می‌شه.

```
loadScript('/script1.js', function(script) {  
    console.log('first script is loaded');  
  
    loadScript('/script2.js', function(script) {  
  
        console.log('second script is loaded');  
  
        loadScript('/script3.js', function(script) {  
  
            console.log('third script is loaded');  
            // after all scripts are loaded  
        });  
    });  
});
```

۶۳. زنجیره promise‌ها چیه؟

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

```

new Promise(function(resolve, reject) {

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

}).then(function(result) {

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

}).then(function(result) {

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

}).then(function(result) {

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

});

```

در دسته‌های بالا، نتیجه به زنجیره‌های then handler با جریان کار زیر منتقل می‌شه.

۱. پرامیس اولیه در ۱ ثانیه حل می‌شه،
۲. پس از اون، «handler» با ثبت نتیجه (۱) فراخوانی می‌شه و سپس یه پرامیس با مقدار نتیجه * ۲ برمی‌گردونه.
۳. پس از اون مقدار به بعدی منتقل شد. سپس با ثبت نتیجه (۲) و برگرداندن یه پرامیس با نتیجه * ۳.
۴. در نهایت مقدار به آخرین . سپس با ثبت نتیجه (۶) و یه پرامیس با نتیجه * ۴ `handler`

۱۴. کاربرد متدهای promise.all چیه؟

یه پرامیسه که آرایه ای از پرامیس‌ها رو به عنوان ورودی می‌گیره (یک تکرار) و زمانی حل می‌شه که همه پرامیس‌ها حل شن یا یکی از اونها رد شه. برای مثال، نحو متدهای زیر اومده.

```

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

```

نکته: ترتیب پرامیس‌ها (خروجی نتیجه) طبق ترتیب ورودی حفظ می‌شود.

۶۵. هدف از متدهای promise race چیه؟

متدهای promise race نمونه‌ای از پرامیس‌رو که اول حل یا رد شده را بر می‌گردونند. بیاین مثالی از متدهای race را در نظر بگیرید که تو اون پرامیس ۲ اول حل می‌شود.

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

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

۶۶. حالت سخت strict توی جاواسکریپت چی کار می‌کنه؟

یه ویژگی جدید در ECMAScript 5 اس که به شما امکان میده یه برنامه یا تابع رو تو یه زمینه عملیاتی "سخت" قرار بدین. به این ترتیب از انجام بعضی اقدامات جلوگیری می‌کنه و استثناهای بیشتری رو ایجاد می‌کنه. عبارت تحت لفظی "استفاده از سخت"؛ به مرورگر دستور میده تا از کد جاواسکریپت در حالت Strict استفاده کنه.

۶۷. چرا به حالت strict نیاز داریم؟

حالت سخت گیرانه برای نوشتتن جاواسکریپت "امن" با اطلاع رسانی "بد نحوی" به خطاهای واقعی مفید است. برای مثال، ایجاد تصادفی یه متغیر گلوبال رو با پرتاب یه خطا حذف می‌کنه و همچنانی یه خطا برای انتساب به یه ویژگی غیرقابل نوشتتن، یه ویژگی فقط گیرنده، یه ویژگی غیرموجود، یه متغیر غیرموجود یا یه ویژگی غیر قابل نوشتتن پرتاب می‌کنه. شی موجود

۶۸. چطوری می‌توانیم حالت strict را فعال کنیم؟

حالت سخت با اضافه کردن `use strict` اعلام می‌شود. به ابتدای یه اسکریپت یا یه تابع. اگه در ابتدای یه اسکریپت اعلام شد، دامنه گلوبال داره.

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

و اگه در داخل یه تابع اعلام کنین محدوده محلی داره

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

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

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

علامت تعجب دوتایی یا نفی (!!) تضمین می‌کنه که نوع حاصل یه بولینه. اگه نادرست بود (برای مثال 0، تهی، تعریف نشده، و غیره)، نادرسته، در غیر این صورت، درسته. برای مثال، می‌توانی نسخه IE رو با استفاده از عبارت زیر آزمایش کنین.

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

اگه از این عبارت استفاده نکنین مقدار اصلی رو برمی‌گردونه.

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

نکته: بیان !! یه اپراتور نیست و فقط دوتا اپراتور ! هست.

۷۰. هدف از عملگر delete چیه؟

کلمه کلیدی `delete` برای حذف ویژگی و همچنین مقدار اون استفاده می‌شود.

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

۷۱. عملگر typeof چیکار می‌کنه؟

برای یافتن نوع متغیر جاوااسکریپت می‌توانیم از عملگر typeof JavaScript استفاده کنیم. نوع یه متغیر یا یه عبارت رو برمی‌گردونه.

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

۷۲. چیه و چه زمانی undefined می‌گیریم؟

ویژگی تعريف نشده نشون میده که به یه متغیر مقداری اختصاص داده نشده یا اصلاً اعلام نشده است. نوع مقدار تعريف نشده هم تعريف نشده.

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

هر متغیری رو می‌شه با تنظیم مقدار روی undefined خالی کرد.

```
user = undefined
```

۷۳. چیه null؟

مقدار null عدم وجود عمدى هر مقدار شى رو نشون میده. این یکی از مقادیر اولیه جاوااسکریپته. نوع مقدار null آبجکته. با قرار دادن مقدار null می‌توانیم متغیر رو خالی کنیم.

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

۷۴. تفاوت‌های بین null و undefined چیا هستن؟

تفاوت‌های اصلی بین null و undefined

Undefined	Null
این یه مقدار انتساب نیست که تو اون متغیری اعلام شده باشه اما هنوز مقداری به اون اختصاص داده نشده.	این یه مقدار انتسابه که نشون میده متغیر به هیچ شیئی اشاره نمی‌کنه.
تایپ undefined عه	تایپ null آبجکته
مقدار تعریف نشده یه مقدار اولیه اس used زمانی که به یه متغیر مقداری اختصاص داده نشده باشه.	مقدار null یه مقدار اولیه اس که نشون دهنده مرجع تهی، خالی یا غیر موجوده.
عدم وجود خود متغیر رو نشون میده	عدم وجود مقدار برای یه متغیر رو نشون میده
در حین انجام عملیات اولیه به NaN تبدیل می‌شه	در حین انجام عملیات اولیه به صفر (۰) تبدیل شد

۷۵. eval چیه؟

تابع eval کد جاواسکریپت رو که به صورت رشته نمایش داده شده رو ارزیابی می‌کنه. رشته میتوانه یه عبارت جاواسکریپت، متغیر، دستور یا دنباله ای از عبارات باشه.

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

۷۶. تفاوت‌های بین document و window چیا هستن؟

Document	Window
----------	--------

Document	Window
این فرزند مستقیم شی پنجره است. این همچنین به عنوان مدل شیء document (DOM) معرفی شده است.	این عنصر سطح ریشه در هر صفحه وب
هر دسترسی شما می‌تواند از طریق window.document یا document به اون دسترسی داشته باشید.	به طور پیش فرض شی پنجره به طور ضمنی در صفحه
متدهایی مانند getElementById، getElementByTagName، createElement وغیره رو فراهم می‌کنه	دارای متدهایی مانند alert، confirm و window.location

۷۷. توى جاواسکریپت چطوری مى‌تونیم به history دسترسی داشته باشیم؟

شی window.history حاوی تاریخچه مرورگر است. با استفاده از متدهای next و back می‌توانیم URLهای قبلی و بعدی رو در تاریخچه بارگذاری کنیم.

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

نکته: همچنین می‌توانیم بدون پیشوند پنجره به تاریخچه دسترسی داشته باشیم.

۷۸. انواع داده‌های جاواسکریپت کدوما هستند؟

- در زیر لیستی از انواع داده‌های جاواسکریپت موجود رو می‌بینیم
- ۱. Number
 - ۲. String
 - ۳. Boolean

Object .۴
Undefined .۵

۷۹. isNaN چیه و چیکار می‌کنه؟

تابع `isNaN` برای تعیین اینکه آیا یه مقدار یه عدد غیرقانونی (Not-a-Number) هست یا نه استفاده می‌شه. یعنی اگه مقدار برابر با `NaN` باشه، این تابع `true` برمی‌گردونه. در غیر این صورت `false` برمی‌گردد.

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

۸۰. تفاوت‌های بین `undefined` و `undeclared` چیا هستن؟

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

<code>undefined</code>	<code>undeclared</code>
تعریف نمی‌شن این متغیرها در برنامه اعلام شده اما هیچ مقدار	این متغیرها تو یه برنامه وجود ندارن و
اگه سعی کنین مقدار یه متغیر تعریف نشده رو بخوانید، یه مقدار تعریف نشده برگردانده می‌شه.	اگه سعی کنین مقدار یه متغیر اعلام نشده رو بخوانید، با خطای زمان اجرا مواجه می‌شوید

۸۱. کدوم متغیرها عمومی هستن؟

متغیرهای عمومی اونایی ان که در طول کد بدون هیچ محدوده ای در دسترسن. کلمه کلیدی `var` برای اعلام یه متغیر محلی استفاده می‌شه اما اگه اونو حذف کنین تبدیل به متغجهانی می‌شه.

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

۸۲. مشکلات متغیرهای عمومی چیا هستن؟

مشکل متغیرهای سراسری تضاد نام متغیرها با دامنه محلی و گلوباله. دیباگ و آزمایش کدی که به متغیرهای سراسری متکیه سخته.

۸۳. مقدار NaN چیه؟

ویژگی NaN یه ویژگی گلوباله که مقدار "Not-a-Number" رو نشون میده. یعنی نشون میده که یه مقدار یه عدد قانونی نیست. استفاده از NaN تو یه برنامه بسیار نادر است، اما میشه از اون به عنوان مقدار بازگشتی برای موارد کمی استفاده کرد

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

۸۴. هدف از تابع isFinite چیه؟

تابع isFinite برای تعیین اینکه آیا یه عدد محدود و قانونیه استفاده میشه. اگه مقدار infinity یا -infinity یا NaN (Not-a-Number) باشه false برمیگردونه، در غیر این صورت true رو برمیگردونه.

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

۸۵. یه event-flow چیه؟

ترتیبیه که event در صفحه وب دریافت میشه. وقتی روی عنصری کلیک میکنیم که در عنصرهای مختلف دیگه تودرتوشه. قبل از اینکه کلیکمون واقعاً به مقصد یا عنصر هدف برسه، باید event کلیک رو برای هر یه از عنصرهای والد خود ابتدا راهاندازی کنه و از بالا با شی پنجره گلوبال شروع شه. دو راه برای جریان event وجود داره

۱. از بالا به پایین(Event Capturing)
۲. از پایین به بالا(Event Capturing)

چیه؟ Event-bubbling .۸۶

نوعی انتشار event هس که تو اون event ابتدا روی درونی‌ترین عنصر Event-bubbling هدف راهاندازی می‌شه و سپس به طور متوالی روی اجداد (والد) عنصر هدف در همون سلسله مراتب تودرتو راهاندازی می‌شه تا زمانی که به بیرونی‌ترین عنصر DOM برسه.

چیه؟ Event-capturing .۸۷

نوعی انتشار event که تو اون event اول با بیرونی‌ترین عنصر ثبت می‌شه و سپس به طور متوالی بر روی children (children) عنصر هدف در همون سلسله مراتب تودرتو راه اندازی می‌شه تا زمانی که به درونی‌ترین عنصر DOM برسه.

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

می‌تونین با استفاده از جاوااسکریپت فرمی رو ارسال کنین use onsubmit event تمام اطلاعات ورودی فرم با استفاده از document.form[0].submit ارسال می‌شه handler

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

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

شی window.navigator حاوی اطلاعاتی درباره جزئیات سیستم عامل مرورگر بازدیدکننده است. بعضی از ویژگی‌های سیستم عامل تحت ویژگی پلتفرم در دسترس هستن،

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

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

رویداد `DOMContentLoaded` زمانی فعال می‌شده که سند اولیه HTML به‌طور کامل بارگیری و تجزیه شده باشد، بدون اینکه منتظر بمانید تا دارایی‌ها (سبک‌ها، تصاویر و فریم‌های فرعی) بارگیری تمام شده. در حالی که رویداد بارگیری زمانی فعال می‌شده که کل صفحه بارگیری شده، از جمله تمام منابع وابسته (شیوه‌ها، تصاویر).

۹۱. تفاوت‌های بین `object`، `host` و `native` چیا هستن؟

آبجکت‌هایی هستن که بخشی از زبان جاواسکریپت تعریف شده توسط مشخصات ECMAScript هستن. برای مثال، اشیاء اصلی رشته، ریاضی، `RegExp`، `Object`، `Function` و غیره که در مشخصات ECMAScript تعریف شدن. آبجکت‌هایی هستن که توسط مرورگر یا محیط زمان اجرا (Node) ارائه می‌شون. برای مثال، پنجره، `XMLHttpRequest`، گره‌های DOM و غیره به عنوان اشیاء میزبان در نظر گرفته می‌شون. آبجکت‌هایی هستن که در کد جاواسکریپت تعریف شدن. برای مثال، اشیاء کاربر ایجاد شده برای اطلاعات پروفایل.

۹۲. کدام ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده می‌شون؟

می‌توانیم از ابزارها یا تکنیک‌های زیر برای اشکال زدایی جاواسکریپت استفاده کنیم

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

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

مزایا و معایب Promise به جای callback

مزایا:

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

معایب:

۱. کد کمی پیچیده می‌سازد
۲. اگه ES6 پشتیبانی نمی‌شه، باید یه polyfill بارگذاری کنین

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

ویژگی‌ها در نشونه گذاری HTML تعریف می‌شن در حالی که ویژگی‌ها در DOM تعریف می‌شن برای مثال، عنصر HTML زیر دارای ۲ ویژگی نوع و مقدار هستش

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

می‌توانین مقدار ویژگی رو به صورت زیر بازیابی کنین

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

و بعد از اینکه مقدار فیلد متن رو به "Good evening" تغییر دادید مانند می‌شه

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

۹۵. سیاست same-origin چیه؟

خط مشی همون مبدأ خط مشیه که از درخواست جاوااسکریپت در سراسر مرزهای دامنه جلوگیری می‌کنه. مبدأ به عنوان ترکیبی از طرح URI، نام میزبان و شماره پورت تعریف

می‌شه. اگه این خطمشی رو فعال کنین از دسترسی یه اسکریپت مخرب تو یه صفحه به داده‌های حساس در صفحه وب دیگر با استفاده از (Document Object Model) (DOM) جلوگیری می‌کنه.

۹۶. هدف استفاده از `void 0` چیه؟

(Void) برای جلوگیری از به روز رسانی صفحه استفاده می‌شه. این برای از بین بردن عارضه جانبی ناخواسته مفید خواهد بود، چون مقدار اولیه تعریف نشده رو برمی‌گردونه. معمولاً برای اسناد HTML استفاده می‌شه که از `(0); href="JavaScript:Void(0)"` استفاده می‌کنن. تو یه عنصر `<a>` . یعنی وقتی روی یه پیوند کلیک می‌کنین مرورگر یه صفحه جدید رو بارگیری می‌کنه یا همون صفحه رو تازه می‌کنه. اما با استفاده از این عبارت از این رفتار جلوگیری می‌شه.

برای مثال، پیوند زیر پیام رو بدون بارگیری مجدد صفحه مطلع می‌کنه

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

۹۷. جاواسکریپت یه زبان تفسیری هست یا کامپایلری؟

جاواسکریپت یه زبان تفسیری‌هه، نه یه زبان کامپایل شده. یه مفسر در مرورگر کد جاواسکریپت رو می‌خواند، هر خط رو تفسیر می‌کنه و اونو اجرا می‌کنه. امروزه مرورگرهای مدرن از فناوری موسوم به کامپایل JIT (Just-In-Time) استفاده می‌کنن که جاواسکریپت رو در زمانی که در شرف اجراست به بایت کد اجرایی کامپایل می‌کنه.

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

بله، جاواسکریپت یه زبان حساس به حروف کوچک و بزرگه. کلمات کلیدی زبان، متغیرها، نام تابع و اشیا، و هر شناسه دیگر باید همی‌شه با حروف بزرگ تایپ شن.

۹۹. ارتباطی بین JavaScript و Java وجود داره؟

نه، اونا کاملاً دو زبان برنامه نویسی متفاوت هستن و هیچ ارتباطی با یکدیگر ندارن. اما هر دوی اونا زبان‌های برنامه نویسی شی گرا هستن و مانند بسیاری از زبان‌های دیگر، از نحو مشابهی برای ویژگی‌های اساسی (اگر، غیره، برای، سوئیچ، شکستن، ادامه و غیره) پیروی می‌کنن.

۱۰۰. Event‌ها چی هستن؟

رویدادها «چیزهایی» هستن که برای عناصر HTML اتفاق می‌افتد. موقعی که جاواسکریپت در صفحات HTML استفاده می‌شه، جاواسکریپت می‌تونه به این رویدادها واکنش نشون بده. بعضی از نمونه‌های رویدادهای HTML عبارتند از:

۱. بارگیری صفحه وب به پایان رسید
۲. فیلد ورودی تغییر کرد
۳. دکمه کلیک شد

بیاین رفتار رویداد کلیک رو برای عنصر دکمه شرح بدیم،

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

۱۰۱. کی جاواسکریپت رو ساخته؟

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

۱۰۲. هدف از متدها preventDefault چیه؟

متدها preventDefault اگه رویداد قابل لغو باشه، اونو لغو میکنه، به این معنی که عمل یا رفتار پیشفرض متعلق به رویداد رخ نمیده. برای مثال، جلوگیری از ارسال فرم موقع کلیک بر روی دکمه ارسال و جلوگیری از باز شدن URL صفحه موقع کلیک کردن بر روی لینک از موارد رایج استفاده شده.

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

.Remember that not all events are cancelable **نکته:**

۱۰۳. کاربرد متدها stopPropagation چیه؟

روش stopPropagation برای جلوگیری از حبابی شدن رویداد در زنجیره رویداد استفاده میشه. برای مثال، divهای تودرتو زیر با متدها stopPropagation از انتشار پیشفرض رویداد موقع کلیک بر روی Div1 (Div1) جلوگیری میکنه.

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

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

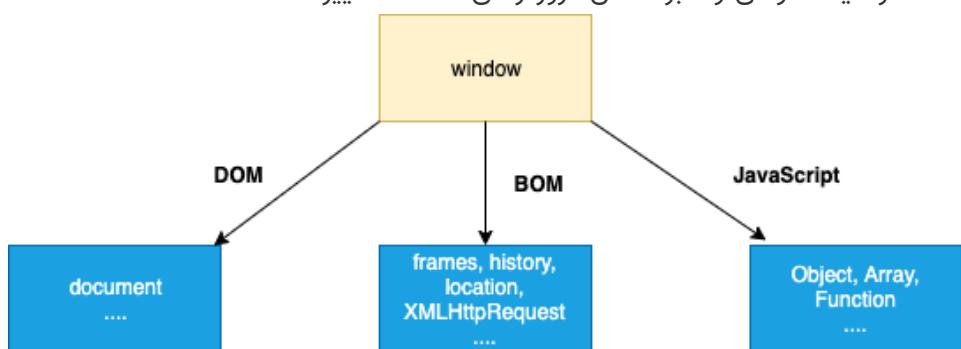
function secondFunc() {
  alert("DIV 2");
}
</script>
```

۱۰۴. مراحلی که موقع استفاده از return false توی یه توی یه رخ میده چیا هستن؟

- عبارت `return false` در کنترل کننده رویداد مراحل زیر را انجام میده:
۱. ابتدا عملکرد یا رفتار پیش فرض مرورگر را متوقف می‌کنه.
 ۲. این رویداد از انتشار **DOM** جلوگیری می‌کنه
 ۳. اجرای **callback** را متوقف می‌کنه و بلافاصله پس از فراخوانی برمی‌گردد.

۱۰۵. BOM چیه؟

مدل شیء مرورگر (BOM) به جاواسکریپت اجازه میده تا با مرورگر "صحبت کنه". این شامل ناوبر اشیاء، تاریخچه، صفحه، مکان و سنده که فرزندان پنجره هستن. مدل شیء مرورگر استانداره نیست و می‌توانه بر اساس مرورگرهای مختلف تغییر کنه.



۱۰۶. موارد استفاده از setTimeout کدوما هستن؟

متده `setTimeout` برای فراخوانی یه تابع یا ارزیابی یه عبارت پس از تعداد مشخصی از میلی ثانیه استفاده می‌شه. برای مثال، بیاین یه پیام رو پس از 2 ثانیه با استفاده از روش `setTimeout` ثبت کنیم.

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

۱۰۷. موارد استفاده از setInterval کدوما هستن؟

متده `setInterval` برای فراخوانی یه تابع یا ارزیابی یه عبارت در بازه‌های زمانی مشخص (بر حسب میلی ثانیه) استفاده می‌شه. برای مثال، اجازه بدین یه پیام رو پس از 2 ثانیه با استفاده از روش `setInterval` ثبت کنیم.

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

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

جاواسکریپت یه زبان تک Thread-ه. چون مشخصات زبان به برنامه نویس اجازه نمیده تا کدی بنویسه که مفسر بتونه بخش‌هایی از اونو به صورت موازی در چندین Thread-یا پردازش اجرا کنه. در حالی که زبان‌هایی مانند C، go، ++java، میتوانن برنامه‌های چند رشته‌ای و چند فرآیندی بسازند.

۱۰۹. چیه؟ Event-delegation

تفویض رویداد تکنیکی برای گوش دادن به رویدادهاست که تو اون یه عنصر والد رو به عنوان شنونده برای همه رویدادهایی که در داخلش اتفاق می‌افتد، تفویض می‌کنین. برای مثال، اگه می‌خواین تغییرات فیلد رو تو یه فرم خاص تشخیص بدین، می‌تونین از تکنیک انتقال رویداد استفاده کنین.

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

// Listen for changes to fields inside the form
form.addEventListener('input', function (event) {
    // Log the field that was changed
    console.log(event.target);

}, false);
```

۱۱۰. چیه؟ ECMAScript

زبان برنامه نویسیه که اساس جاواسکریپت رو تشکیل میده. ECMAScript توسط سازمان استانداره بین المللی ECMA در مشخصات ECMA-262 و ECMA-402 استانداره شده است. اولین نسخه ECMAScript در سال ۱۹۹۷ منتشر شد.

یه فرمت سبک وزن هس که برای تبادل داده‌ها استفاده می‌شه. این بر اساس زیرمجموعه‌ای از زبان جاواسکریپت که اشیا در جاواسکریپت ساخته می‌شن.

۱۱۲. قوانین فرمت JSON کدوما هستن؟

- در زیر لیستی از قوانین نحوی JSON آمده است
۱. داده‌ها به صورت جفت نام/مقدار هستن
 ۲. داده‌ها با کاما از هم جدا می‌شن
 ۳. برآکتها اجسام رو نگه می‌دارن
 ۴. برآکتها مربعی آرایه‌ها رو نگه می‌دارن

۱۱۳. هدف از متده JSON.stringify چیه؟

موقع ارسال داده‌ها به وب سرور، داده‌ها باید در قالب رشته‌ای باشند. شما می‌تونین با تبدیل شی `JSON` به رشته با استفاده از متده `stringify` هدف دست یابید.

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


** *[فهرست] (#فهرست)
```

## ۱۱۴. چطوری می‌تونیم یه رشته JSON رو تجزیه کنیم؟

موقع دریافت داده‌ها از یه وب سرور، داده‌ها همیشه در قالب رشته‌ای هستن. اما می‌تونین این مقدار رشته رو با استفاده از متده `parse` به یه شی جاواسکریپت تبدیل

کنین.

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

## ۱۱۴. چرا به JSON نیاز داریم؟

موقع تبادل داده بین مرورگر و سرور، داده‌ها فقط می‌توان متنی باشند. از اونجایی که JSON فقط متنی است، می‌توان اونو به راحتی به سرور ارسال کرد و از اون به عنوان قالب داده توسط هر زبان برنامه‌نویسی استفاده کرد.

## ۱۱۵. PWA‌ها چی هستند؟

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

## ۱۱۶. هدف از متدهای clearTimeout چیه؟

تابع clearTimeout در جاوااسکریپت برای پاک کردن بازه زمانی استفاده می‌شود که قبل از اون توسط تابع setTimeout تنظیم شده است. یعنی مقدار بازگشتی تابع setTimeout تویه متغیر ذخیره می‌شود و برای پاک کردن تایمر به تابع clearTimeout منتقل می‌شود.

برای مثال، از روش setTimeout زیر برای نمایش پیام پس از 3 ثانیه استفاده می‌شود. این مهلت زمانی رو می‌شود با روش clearTimeout پاک کرد.

```

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

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

```

## ۱۱۷. هدف از متدهای clearInterval چیه؟

تابع `clearInterval` تو جاوااسکریپت برای پاک کردن فاصله ای که توسط تابع `setInterval` تنظیم شده استفاده می‌شود. برای مثال، مقدار بازگشتی که توسط تابع `setInterval` برمی‌گردد تویه متغیر ذخیره می‌شود و برای پاک کردن فاصله به تابع `clearInterval` ارسال می‌شود. برای مثال، از روش `setInterval` زیر برای نمایش پیام در هر ۳ ثانیه استفاده می‌شود. این بازه رو می‌شه 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 جاواسکریپت، چطوری می‌شه به یه صفحه جدید redirect انجام داد؟

در `vanila` جاواسکریپت، می‌تونین با استفاده از ویژگی `location` شی پنجره به صفحه جدیدی هدایت بشین.

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

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

۳ روش ممکن برای بررسی اینکه آیا یه رشته دارای یه رشته فرعیه یا نه وجود دارد.  
۱. استفاده از متدهای `String.prototype.includes`: ES6 روش `includes` برای آزمایش یه رشته حاوی یه رشته فرعی ارائه کرد.

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

۲. استفاده از متدهای `String.prototype.indexOf`: تو یه محیط ES5 یا قدیمی‌تر، می‌تونین از «» استفاده کنین که ایندکس یه رشته فرعی را بر می‌گردونه. اگه مقدار شاخص برابر با ۱ نباشه، به این معنیه که رشته فرعی در رشته اصلی وجود دارد.

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

۳. استفاده از `RegEx`: راه حل پیشرفته از روش آزمون عبارت استفاده کننه، که امکان آزمایش در برابر عبارات منظم رو فراهم می‌کنه.

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

## ۱۲۰. توانی جاوااسکریپت، چطوری مقدار یه آدرس email رو اعتبارسنجی میکنین؟

میتوانین با استفاده از `Regex` ایمیل رو در جاوااسکریپت تأیید کنین. توصیه میشه به جای سمت کلاینت، اعتبارسنجی در سمت سرور انجام شه. چون جاوااسکریپت رو میشه 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 رو می پذیرد.

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

میتوانین از عبارت `window.location.href` برای دریافت مسیر آدرس فعلی استفاده کنین و میتوانین از همون عبارت برای بهروزرسانی URL هم استفاده کنین. همچنین میتوانین از `document.URL` برای اهداف فقط خواندنی استفاده کنین اما این راه حل مشکلاتی در FF داره.

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

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

برای دسترسی به اجزای URL صفحه میتوان از ویژگی‌های شی `location` زیر استفاده کرد.

۱. URL - ورودی `href`

۲. URL - پروتکل استفاده شده

۳. URL - هاست و پورت `host`

۴. URL - هاست `hostname`

۵. URL - شماره پورت `port`

- ۶. URL - مسیر pathname
- ۷. URL - قسمت جستجو search
- ۸. URL - محل جایگیری hash

## ۱۲۳. توى جاواسكريپت چطوري مىتونيم مقدار يه query-string رو بخونيم؟

مىتونين از `URLSearchParams` برای دریافت مقادير رشته پرس و جو در جاواسكريپت استفاده کنин. بياين مثالی برای دریافت مقدار کد مشتری از رشته پرس و جو `URL` ببینيم،

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

## ۱۲۴. چطوري مىتونيم بررسی کنيم که آيا يه پرآپرتی روی آبجکت وجود داره يا نه؟

۱. استفاده از **عملگرها**: شما مىتونيم از عملگر `in` استفاده کنин که آيا کلیدی تو يه شي وجود داره يا نه

```
"key" in obj
```

و اگه مىخواين بررسی کنин که آيا کلید وجود نداره، به ياد داشته باشيم که از پرانتز استفاده کنин

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

۲. استفاده از متده `hasOwnProperty`: مىتونين از `hasOwnProperty` برای آزمایش ویژگی های نمونه شی (و نه ویژگی های ارثی) استفاده کنин.

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

۳. استفاده از مقایسه `undefined`: اگر از يه شي به يه ویژگی غير موجود دسترسی پيدا کنيد، نتيجه `undefined` است. بياين ویژگي ها را با `undefined` مقایسه کنيم تا وجود ویژگي را مشخص کنيم.

```

const user = {
 name: 'John'
};

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

```

## ۱۲۵. چطوری روی یه object حلقه میزنی؟

میتونین از حلقه `for-in` برای حلقه زدن شی جاواسکریپت استفاده کنین. همچنین میتونین مطمئن شین که کلیدی که دریافت میکنین به ویژگی واقعی یه آبجکته و با استفاده از روش `hasOwnProperty` از نمونه اولیه نیست.

```

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

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

```

## ۱۲۶. چطوری تست میکنی که یه object خالیه؟

راه حل‌های مختلفی بر اساس نسخه‌های `ECMAScript` وجود داره

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

میتونیم از `Object.entries` استفاده کنیم و `length` اونا رو چک کنیم

```

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

```

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

می‌توانیم از `object.keys` استفاده کنیم و `length` اونو چک کنیم

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

## ۲. استفاده از `for-in` با متدهای `hasOwnProperty` (Pre-ECMA 5)

حلفه `for-in` استفاده کنیم و هر پارامتر رو با `hasOwnProperty` چک کنیم

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

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

## ۳. `arguments` چیه؟

شیء آرگومان‌ها یه شیء آرایه ماننده که در داخل توابع قابل دسترسیه و حاوی مقادیر آرگومان‌های ارسال شده به اون تابعه. برای مثال، بیاین ببینیم چگونه از شیء آرگومان‌ها در تابع `sum` استفاده کنیم

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

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

**نکته:** شما نمی‌توانید از متدهای ارایه برای این شیء آرگومان‌ها استفاده کنید اما می‌توانید به ارایه تبدیلش کنید

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

## ۱۲۹. چطوری حرف اول یه رشته رو به حرف بزرگ تبدیل می‌کنی؟

می‌تونیم با درست کردن یهتابع که با استفاده از زنجیره‌ای از متدهای استرینگ‌ها مثلا slice یه استرنیگ با حرف اول بزرگ ایجاد کرد

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

## ۱۳۰. مزایا و معایب حلقه for چیا هستن؟

حلقه for یه سینتکس تکراری رایج در جاواسکریپت‌ه که هم مزایا و هم معایب داره

### مزایا

۱. توی همه‌ی محیط‌ها env کار می‌کنه
۲. می‌تونیم از break و continue برای کنزل جریان داده استفاده کرد

### معایب

۱. پر هزینه
۲. ضروریت
۳. ممکنه با خطاهای یه به یه رو برو شین

## ۱۳۱. چطوری تاریخ جاری رو توی جاواسکریپت نشون میدی؟

شما می‌تونین از کلاس new Date استفاده کنین که یه آبجکت از زمان و تاریخ بهمون میده برمیم یه مثال ازش ببینیم

```
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);
```

## ۱۳۲. چطوری دو تا date object رو با هم مقایسه می‌کنی؟

برای این کار باید از متده استفاده کرد و نباید از اپراتورها استفاده کنیم

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

## ۱۳۳. چطوری بررسی می‌کنی که یه رشته با یه رشته دیگه شروع می‌شه؟

ما می‌تونیم از متده startWith که بر روی پرتوتایپ رشته‌ها وجود داره استفاده کرد که یه رشته رو می‌گیره و چک می‌کنه که با اون شروع می‌شه رشته مورد نظر یا نه برمی‌یه مثال بینیم در موردش

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

## ۱۳۴. چطوری یه رشته رو trim می‌کنی؟

جاواسکریپت یه متده به ما میده به اسم trim که روی استرینگ‌ها قرار داره با استفاده از این متده همه‌ی 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 NBS
 var rtrim = /^[^\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g;
 String.prototype.trim = function() {
 return this.replace(rtrim, '');
 };
 })();
}
```

## ۱۳۵. توى جاواسكريپت چطورى مىتونيم يه زوج مرتب از key يه valueها بسازيم؟

براي اضافه کردن key جديد به ابجکها دو روش وجود دارن

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

۱. اين روش زمانی موثر هستش که اسم پراپرتی رو

ميدونيم

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

۲. اين روش وقتی موثره که اسم پراپرتی رو

داینامیک باشه

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

## ۱۳۶. آيا عبارت '!--' عملگر خاصی هست؟

نه! اون يه اپراتور خاص نىست اما ترکيب شده دو تا اپراتور استانداره هستش يكى بعد اون يكى

۱. اپراتور نقبيض (!)

۲. کاهش كننده (-)

اول يه شماره از مقدار متغير به مثال کم مىشه بعد تست مىشه که مساوى صفر هستش يا نه که مشخص كننده درست يا غلط بودن شرط هستش

## ۱۳۷. چطورى مىتونيم به متغيرهامون مقادير اوليه بدیم؟

مىتونيم از عملگر يا اپراتور || استفاده تعريف يه مقدار پيشفرض استفاده کرد مانند مثال زير

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

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

## ۱۳۸. چطوری می‌تونیم متن‌های چند خطی درست کنیم؟

ما می‌توانیم از / برای تعریف کردن رشته‌های چند لایه استفاده کنیم برای مثال

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

اما اگه یه فاصله بعد / داشته باشیم، کد دقیقاً به همون حالتی که هست نشون داده می‌شه اما یه اور خطای نوشتاری کد قراره داشته باشیم

## ۱۳۹. مدل app-shell چیه؟

یکی از راه‌های ساخت یه PWA هس که به طور قابل اعتماد و فوری بر روی صفحه نمایش کاربران شما بارگیری می‌کنند، مشابه اونجه در برنامه‌های کاربردی بومی مشاهده می‌کنید. برای رسوندن سریع HTML اولیه به صفحه بدون شبکه مفیده.

## ۱۴۰. چطوری می‌توانیم روی یه تابع property اضافه کنیم؟

بله ما می‌توانیم برای توابع پراپرتی تعیین کنیم چون توابع اصولاً آبجکت هستن.

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

fn.name = "John";

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

## ۱۴۱. چطوری می‌تونیم تعداد پارامترهای ورودی یه تابع رو به دست بیاریم؟

با استفاده کردن از `function.length` ما می‌تونیم به تعداد پارامترهایی که یه تابع منتظر داره بگیره دسترسی داشته باشیم برمی‌یه مثال درموردش ببینیم

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

## ۱۴۲. چیه Polyfill؟

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

## ۱۴۳. عبارات `continue` و `Break` چی هستن؟

دستور `break` برای "پرش به بیرون" از یه حلقه استفاده می‌شه. یعنی حلقه رو می‌شکنه و اجرای کد رو بعد از حلقه ادامه نمیده.

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

دستور `continue` برای "پرش از روی" یه تکرار در حلقه استفاده می‌شه. یعنی یه تکرار (در حلقه) رو می‌شکنه، اگه شرایط مشخصی رخ بده، و با تکرار بعدی در حلقه ادامه می‌ده.

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

## ۱۴۴. توی جاوااسکریپت `label`ها چیکار می‌کنن؟

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

```
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"
```

## ۱۴۵. مزایای `declare` کردن متغیرها در اوایل کد چیه؟

- توصیه می شه تمام اعلان ها رو بالای هر اسکریپت یا تابع نگه دارین. مزیت این کار
۱. کد ما تمیز تر می شه
  ۲. این یه مکان واحد برای جستجوی متغیرهای محلی فراهم می کنه
  ۳. می شه راحت از استفاده متغیرهای ناخواسته جلوگیری کرد
  ۴. این کار محاسبات ناخواسته رو کمتر می کنه

## ۱۴۶. مزایای مقداردهی اولیه متغیرها چیه؟

- توضیه می شه که مقدار اولیه برای متغیرها تعیین بشه که دلایلشو چک می کنیم
۱. خروجیمون کد تمیز تری می شه
  ۲. این کار باعث می شه یه جا برای این متغیر رزرو بشه
  ۳. از برگشتمن خطای `undefined` جلوگیری می شه

## ۱۴۷. روش توصیه شده برای ایجاد object چیه؟

برای ساخت یه `object` با مقادیر پیشفرش مثالهای زیر روشهای ساخت مقادیر پیشفرض رو بررسی میکنیم

۱. استفاده از {} به جای `new Object`
۲. استفاده از "" به جای `new String`
۳. استفاده از ۰ به جای `new Number`
۴. استفاده از false به جای `new Boolean`
۵. استفاده از [] به جای `new Array`
۶. استفاده از ()/ به جای `new RegExp`
۷. استفاده از (){} به جای `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(){};
```

## ۱۴۸. چطوری میتونیم آرایه JSON تعریف کنیم؟

آرایههای JSON نوشته شده در داخل براکت‌ها و ارایههایی که دارای `Object` هستن بریم یه مثال درموردش ببینیم

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

## ۱۴۹. چطوری میتوانیم اعداد تصادفی تولید کنیم؟

ما میتوانیم از متدهای `Math.random` برای ساخت یه عدد رندوم بین ۰ تا ۱ یه و از متدهای `Math.floor` برای رند کردن اون عدد استفاده کنیم حالا حاصل عدد به دست اومده رو

ضربرد ر د کنیم عددی بین یه 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
```

نکته: یه عدد تصادفی بین ۰ (شامل) و ۱ (انحصاری) برمی‌گردونه.

## ۱۵۰. می‌تونی یهتابع تولید اعداد تصادفی توی یه بازه مشخص بنویسی؟

بله ما می‌تونیم این تابع رو داشته باشیم که مقادیر حداکثر و حداقل رو بگیره و برای ما عدد رندوم ایجاد کنه

```
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
```

## ۱۵۱. Tree-shaking چیه؟

نوعی حذف کد مرده هستش. این بد این معنیه که مازولهای استفاده نشده در طول فرآیند ساخت در بسته گنجونده نمی‌شن و برای اون بر ساختار استاتیک مازول ES2015 متکیه (یعنی import و export). در ابتدا این باندلر مازول "rollup" رایج شد.

## ۱۵۲. دلایل نیاز به tree-shaking کدوما هستن؟

می‌توانه اندازه کد رو در هر برنامه ای به میزان قابل توجهی کاهش بده. یعنی هرچی کد کمتری از طریق سیم بفرستیم برنامه کاربردی تره. به عنوان مثال، اگر فقط بخواهیم یه برنامه Hello World با استفاده از چارچوبهای SPA ایجاد کنیم، حدود چند

مگابایت طول می کشد، اما با tree-shaking می تونه اندازه رو به چند صد کیلوبایت کاهش بده. در باندلرهای Rollup و Webpack tree-shaking پیاده سازی شده.

## ۱۵۳. آیا استفاده از eval توصیه می شه؟

خیر، اجازه اجرای کد دلخواه رو میده که باعث ایجاد مشکل امنیتی می شه. همونطور که میدونیم از تابع eval برای اجرای متن به عنوان کد استفاده می شه. در بیشتر موارد استفاده از اون ضروری نیست.

## ۱۵۴. چیه؟ Regular-Expression

یا همون Regex یا regular expression یه توالیه که یه ساختار جستجو ایجاد می کنه با استفاده از این ساختاز ما می تونیم 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 می شه

/John/i

## ۱۵۵. متدهای رشته که روی Regular-expression مجاز هستن کداماست؟

دو تا متدهای Regular Expressions برای رشته ها داره : search و replace. متدهای search عبارت رو می گیره اونو جسنجو می کنه و محل اون عبارت رو برمی گردونه

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

متدهای replace برای برگرداندن رشته اصلاح شده در جایی که الگو جایگزین می شه استفاده می شه

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

## ۱۵۶. توی Regex بخش modifiers چیکار می‌کنه؟

Modifier میتوان زمانی استفاده بشن که به جستجوهای بدون حروف کوچک و بزرگ سراسری نیاز داریم بیان یه مثال درموردشون ببینیم

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

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
```

## ۱۵۷. پترن‌های regular-expression چیه؟

Regex یه گروهی از ساختارها برای این اماده کرده که با اونا کاراکترها رو چک کنیم اونا تو سه مدل طبقه بندی میشن

۱. براکت‌ها: این‌ها استفاده میشن تا یه رنجی از کاراکتر رو پیدا کنن

برای مثال پایین چن تا مورد استفاده لیست شدن

۲. [abc]: استفاده می‌شه تا هر کاراکتری بین این سه کاراکتر پیدا بشه

۳. [0-9]: استفاده می‌شه تا ارقام بین این دو عدد پیدا بشه

۴. (a|b): برای پیدا کردن هر یه از گزینه‌های جدا شده با | استفاده می‌شه

۵. کاراکتر برابر با: این عبارت‌ها کاراکترهایی با معنی خاص هستن

برای مثال پایین دو تا مورد که استفاده می‌شه ازشون رو ببینیم

۶. \d: استفاده برای پیدا کردن اعداد

۷. استفاده برای پیدا کردن فاصله‌ها
۸. استفاده برای پیدا کردن کاراکترهای همخوانی داشته با شروع شدن یا پایانشون
۹. **کمیت کنندگان:** این‌ها برای تعریف کمیت‌ها موثر هستند
- برای مثال پایین دو تا مورد استفاده برآشون اوردهیم
۱۰. +: برای پیدا کردن رشته همخوانی داشته با حداقل یه کاراکتر
۱۱. \*: برای پیدا کردن همخوانی هر رشته شامل صفر یا بیشتر
۱۲. ?: برای پیدا کردن هر رشته که شامل صفر یا یه کاراکتر می‌شه

## ۱۵۸. آبجکت RegExp چیه؟

یه عبارت معمولی با پرایپرتبه و متدهای تعریف شده از قبل هس. بریم یه مثال از نحوه استفاده‌شون ببینیم.

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

## ۱۵۹. چطوری روی یه رشته دنبال یه پترن RegExp می‌گردی؟

می‌تونین از متدهای `test` عبارت منظم برای جستجوی یه رشته برای الگو استفاده کنین و بسته به نتیجه، `true` یا `false` رو برگردونین.

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

## ۱۶۰. هدف از متدهای exec چیه؟

هدف متدهای `exec` شبیه به روش `test` است، اما جستجوی یه تطابق تو یه رشته مشخص رو انجام میده و یه آرایه نتیجه یا `null` رو به جای برگرداندن `true/false` برمی‌گردونه.

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

## ۱۶۱. چطوری استایل‌های یه المنت HTML رو تغییر میدی؟

شما می‌تونین سبک درون خطی یا اسم کلاس یه عنصر HTML رو با استفاده از جاواسکریپت تغییر بدین

۱. استفاده از پرایپری استایل: با استفاده از ویژگی style می‌تونین استایل درون خطی رو تغییر بدین

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

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

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

## ۱۶۲. نتیجه عبارت $1+2+3^3$ چی می‌شه؟

خروجی '33' می‌شه. از اونجایی که «1» و «2» مقادیر عددی هستن، نتیجه دو رقم اول یه مقدار عددی «3» خواهد بود. رقم بعدی یه مقدار نوع رشته اس چون افزودن مقدار عددی «3» و مقدار رشته «3» فقط یه مقدار الحاقی «33» می‌شه.

## ۱۶۳. عبارت debugger چیکار می‌کنه؟

دستور `debugger` هر گونه عملکرد اشکال زدایی موجود رو فراخوانی می‌کنه، مانند تعیین `breakpoint`. اگه هیچ عملکرد اشکال زدایی در دسترس نباشه، این عبارت تاثیری نداره. برای مثال، در تابع زیر یه دستور `debugger` درج شده. بنابراین اجرا در دستور `debugger` مانند یه `breakpoint` در منبع اسکریپت متوقف می‌شه.

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

## ۱۶۴. هدف از breakpoint‌ها توی debugging چیه؟

پس از اجرای دستور debugger و باز شدن پنجره دیبیاگر، می‌توانیم breakpoint‌ها را در کد جاوااسکریپت تنظیم کنیم. در هر breakpoint، جاوااسکریپت اجرا نمی‌شود و به شما اجازه میدهد مقادیر جاوااسکریپت را بررسی کنیم. پس از بررسی مقادیر، می‌توانیم با استفاده از دکمه پخش، اجرای کد را از سر بگیرید.

## ۱۶۵. آیا می‌توانیم از عبارت‌های رزرو شده در تعریف identifier‌ها (اسم متغیر، کلاس و ...) استفاده کنیم؟

نه، شما نمی‌توانید از کلمات رزرو شده به عنوان متغیر، برچسب، اسم آبجکت یا تابع استفاده کنید. بیاین یه مثال ساده را بینید،

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

## ۱۶۶. چطوری تشخیص بدیم که یه مرورگر mobile هست یا نه؟

ما می‌توانیم با استفاده از boolean که یه Regex که به ما برمی‌گردونه 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|blaz
|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|u
(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)|ap
m|r |s
)|avan|be(ck|ll|nq)|bi(lb|rd)|bl(ac|az)|br(e|v)w|bumb|bw\-
(n|u)|c55\/*|capi|ccwa|cdm\-*|cell|chtm|cldc|cmd\-
|co(mp|nd)|craw|da(it|ll|ng)|dbte|dc\-
s|devi|dica|dmob|do(c|p)o|ds(12|\-
d)|el(49|ai)|em(l2|ul)|er(ic|k0)|esl8|ez([4-
7]0|os|wa|ze)|fetc|fly(\-_)|g1 u|g560|gene|gf\-\-5|g\-
mo|go(\.\.w\od)|gr(ad|un)|haie|hcit|hd\-(m|p|t)|hei\-
|hi(pt|ta)|hp(i|ip)|hs\-\-c|ht(c(\-_
|_|a|g|p|s|t)|tp)|hu(aw|tc)|i\-\-(20|go|ma)|i230|iac(|\
|\/*)|ibro|idea|ig01|ikom|im1k|inno|ipaq|iris|ja(t|v)a|jbro|jemu|j.
|\/)|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(
|[1-
8]|c))|phil|pire|pl(ay|uc)|pn\-\-2|po(ck|rt|se)|prox|psio|pt\-
g|qa\-\-a|qc(07|12|21|32|60|\-\-[2-
7]|i\-\-)|qtek|r380|r600|raks|rim9|ro(ve|zo)|s55\/*|sa(ge|ma|mm|ms|n
|oo|p\-\-)|sdk\/*|se(c(\-\-0|1)|47|mc|nd|ri)|sg|h\-\-shar|sie(\-\
|m)|sk\-\-0|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\-\-9|up(\.b|g1|si)|utst|v400|v750|veri|vi(rg|te)|vk(40|5
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;
};

```

## ۱۶۷. چطوری بدون Regex تشخیص بدیم که یه مرورگر mobile هست یا نه؟

می‌تونیم مرورگرهای تلفن همراه رو با اجرای فهرستی از دستگاهها و بررسی اینکه آیا `navigator.userAgent` با چیزی مطابقت داره یا نه، شناسایی کنیم. این یه 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;
}
}
```

## ۱۶۸. چطوری طول و عرض یه تصویر رو با جاواسکریپت به دست میاری؟

ما می‌تونیم با استفاده از جاواسکریپت به صورت برنامه‌ریزی شده تصویر رو بدست بیاریم و بعد (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';
```

## ۱۶۹. چطوری درخواست‌های synchronous HTTP بزنیم؟

مرورگرها یه شی XMLHttpRequest ارائه می‌دان که می‌تونه برای ایجاد درخواست‌های HTTP همزمان از جاواسکریپت استفاده شه.

```

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

```

## ۱۷۰. چطوری درخواست‌های asynchronous HTTP بزنیم؟

مرورگرها یه شی XMLHttpRequest رو ارائه می‌دان که می‌توونن برای درخواست‌های HTTP ناهمزمان از جاوااسکریپت 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);
 }
 xmlhttpReq.open("GET", theUrl, true); // true for asynchronous
 xmlhttpReq.send(null);
}

```

## ۱۷۱. چطوری یه تاریخ رو به یه تاریخ در timezone دیگه تبدیل کنیم؟

می‌توونیم از متدها toLocaleString برای تبدیل تاریخ‌ها تو یه منطقه زمانی به منطقه زمانی دیگر استفاده کنیم. برایم یه مثال درموردش ببینیم.

```

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

```

## ۱۷۲. چه property‌هایی برای اندازه‌گذاری سایز window به کار میره؟

می‌تونیم از ویزگی‌های `innerWidth`, `innerHeight`, `clientWidth`, `clientHeight` ویندوز، عنصر `document` و اشیاء بدنه `document` برای یافتن اندازه `window` پنجره استفاده کنیم. بیاین از ترکیب اونا برای محاسبه اندازه `window` یا `document` استفاده کنیم.

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

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

## ۱۷۳. عملگر شرطی سه گانه توی جاواسکریپت چیه؟

عملگر شرطی (ternary) تنها عملگر جاواسکریپت هستش که سه عملوند رو 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
```

## ۱۷۴. آیا می‌شه روی عملگر شرطی زنجیره شرط‌ها رو اعمال کرد؟

بله، می‌تونیم زنجیره‌سازی رو 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; }
}

```

## ۱۷۵. روش‌های اجرای جاواسکریپت بعد از لود شدن صفحه کدوما هستن؟

:window.onload .۱

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

:document.onload .۲

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

:body onload .۳

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

## ۱۷۶. تفاوت‌های بین prototype و proto کدوما هستن؟

آبجکت‌ای `__proto__` آبجکت واقعیه که در زنجیره جستجو برای حل متدها و غیره استفاده می‌شود. در حالی که `prototype` آبجکت‌ایه که برای ساخت استفاده می‌شود زمانی که یه آبجکت با جدید ایجاد می‌کنیم.

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

## ۱۷۷. میتوانی یه مثال از زمانی که واقعاً به سمیکولون(;) نیاز هست بزنی؟

توصیه می‌شده بعد از هر عبارت در جاواسکریپت از نقطه ویرگول استفاده کنیم. برای مثال، در مورد زیر به دلیل از دست دادن نقطه ویرگول، خطای `is not a function` را در زمان اجرا میندازه.

```
// define a function
var fn = function () {
 //...
}

// semicolon missing at this line

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

و از اون بدست می‌داد

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

در این حالت، تابع دوم رو به عنوان آرگومان به تابع اول ارسال می‌کنیم و سعی می‌کنیم نتیجه فراخوانی تابع اول رو به عنوان تابع فراخوانی کنیم. از این رو، تابع دوم با خطای `is not a function` در زمان اجرا اور می‌گیریم.

## ۱۷۸. متدهای `freeze` چیکار می‌کنه؟

متدهای `freeze` برای فریز کردن یه آبجکت استفاده می‌شده. ثابت کردن یه آبجکت اجازه افزودن ویژگی‌های جدید به یه آبجکت رو نمی‌ده. از حذفش جلوگیری می‌کنه و از تغییر قابلیت شمارش پذیری، پیکربندی یا قابلیت نوشتن ویژگی‌های موجود جلوگیری می‌کنه. یعنی آبجکته ارسال شده رو برمی‌گردونه و کپی ثابتی ایجاد نمی‌کنه.

```

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 می‌ده که ارگومان داده شده `object` نیست

## ۱۷۹. هدف از متد `freeze` چیه؟

۱. برای فریز کردن آبجکت‌ها و آرایه‌ها
۲. برای `immutable` کردن آبجکت‌ها

## ۱۸۰. چرا به متد `freeze` نیاز داریم؟

در پارادایم شی گرا، یه API موجود حاوی عناصر خاصیه که قصد توسعه، اصلاح یا استفاده مجدد در خارج از زمینه فعلی خود رو ندارن. از این رو، این کلمه کلیدیه `final` که در زبان‌های مختلف استفاده می‌شه.

## ۱۸۱. چطوری می‌تونیم زبان ترجیحی یه مرورگر رو تشخیص بدیم؟

ما می‌تونیم از آبجکت `navigator` گه بر روری مرورگر 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);

```

## ۱۸۲. چطوری می‌تونیم حرف اول همه کلمات یه رشته رو به حرف بزرگ تبدیل کنیم؟

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

```
function toTitleCase(str) {
 return str.replace(
 /\w\S*/g,
 function(txt) {
 return txt.charAt(0).toUpperCase() +
 txt.substr(1).toLowerCase();
 }
);
}
toTitleCase("good morning john"); // Good Morning John
```

## ۱۸۳. چطوری می‌شه تشخیص داد که جاواسکریپت یه صفحه وب غیرفعال شده؟

برای تشخیص غیرفعال بودن یا نبودن جاواسکریپت می‌تونین از تگ `<noscript>` استفاده کنین. بلوک کد داخل `<noscript>` زمانی اجرا می‌شه که جاواسکریپت غیرفعال است، و معمولاً برای نمایش محتوای جایگزین زمانی که صفحه در جاواسکریپت تولید می‌شه، استفاده می‌شه.

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

## ۱۸۴. عملگرهای پشتیبانی شده توسط جاواسکریپت کدوما هستن؟

یه عملگر قادر به دستکاری (محاسبات ریاضی و منطقی) مقدار یا عملوند معینه. اپراتورهای مختلفی توسط جاواسکریپت پشتیبانی می‌شن این اپراتورها هستن

۱. **عملگرهای حسابی**: شامل + (اضافه), - (منها), \* (ضرب), / (تقسیم), % (درصد), + (اضافه کردن) و - (کم کردن)
۲. **عملگرهای مقایسه ای**: شامل == (برابر), != (غیر برابر), >= (برابر و تایپ برابر), < (بزرگتر مساوی), > (کوچکتر مساوی)
۳. **عملگرهای منطقی**: شامل && ("و" منطقی), || ("یا" منطقی), ! ("منطقی نه")
۴. **عملگرهای تعیین مقدار**: شامل = (اپراتور تعیین مقدار), += (اضافه کردن و تعیین مقدار), -= (منها کردن و تعیین مقدار), \*= (ضرب و تعیین مقدار), /= (تقسیم و تعیین مقدار), %= (باقي مانده و تعیین مقدار)
۵. **اپراتور سه تایی**: شامل اپراتورهای شرطی سه تایی
۶. **اپراتور تایپ**: از اون برای پیدا کردن تایپ متغیرها استفاده می‌شود به صورت `typeof variable`

## ۱۸۵. پارامتر rest چیکار می‌کنه؟

پارامتر `Rest` یه روش بهبود یافته برای مدیریت پارامترهای تابع هستش که به ما امکان میده تعداد نامحدودی از آرگومان‌ها رو به عنوان یه آرایه نمایش بدیم

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

برای مثال، بیاین یه مثال مجموع برای محاسبه تعداد پویا پارامترها در نظر بگیریم،

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

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

## ۱۸۶. اگه پارامتر rest رو به عنوان آخرین پارامتر استفاده نکنیم چی میشه؟

پارامتر Rest باید آخرین آرگومان باشه، چون وظیفه اش جمع آوری تمام آرگومان های باقی مونده تو یه آرایه اس. برای مثال، اگه تابعیو مثل زیر تعریف کنیم معنی نداره و یه خطا ایجاد می کنه.

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

## ۱۸۷. عملگرهای منطقی باینری توی جاوااسکریپت کدوما هستن؟

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

## ۱۸۸. عملگر spread چیکار میکنه؟

عملگر Spread به تکرارپذیرها (آرایهها / اشیاء / رشتهها) اجازه میده تا به آرگومان ها / عناصر منفرد گسترش پیدا کنن. 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
```

## ۱۸۹. چطوری تشخیص میدی که یه آبجکت freeze شده یا نه؟

- متد Object.isFrozen برای تعیین اینکه آیا یه آبجکت منجمد هس یا نه استفاده می‌شه.  
اگه همه شرایط زیر درست باشه، یه آبجکت منجمد می‌شه.
۱. اگه قابل توسعه نباشه.
  ۲. اگه تمام خصوصیاتش غیر قابل تنظیم باشن.
  ۳. اگه تمام خصوصیات داده اون غیر قابل نوشتن باشه.

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

## ۱۹۰. چطوری بررسی کنیم که دو تا مقدار(شامل آبجکت) با هم برابرن یا نه؟

متد Object.is تعیین می‌کنه که آیا دو مقدار یه مقدار هستن یا نه. برای مثال، استفاده با انواع مختلف مقادیر،

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

اگه یکی از موارد زیر برقرار باشه، دو مقدار یکسان هستن:

۱. هردو undefined

۲. هردو null

۳. هردو true یا هر دو false

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

۵. هر دو یه آبجکت (یعنی هر دو شی رفرنس یکسان دارن)

۶. هر دو عدد و

هر دو + ۰

هر دو - ۰

هر دو NaN

هر دو غیر صفر و هر دو NaN نیستن و هر دو دارای یه مقدار هستن.

## ۱۹۱. هدف از متدها `Object.assign` چیه؟

۱. برای مقایسه دو رشته استفاده می‌شود.
۲. برای مقایسه دو عدد استفاده می‌شود.
۳. برای مقایسه قطبیت دو عدد استفاده می‌شود.
۴. برای مقایسه دو آبجکت استفاده می‌شود.

## ۱۹۲. چطوری `Object` رو به یه `property` دیگه کپی می‌کنی؟

می‌توانیم از متدهای `Object.assign` استفاده کنیم که برای کپی کردن مقادیر و ویژگی‌ها از یه یا چند آبجکت منبع به یه آبجکت هدف استفاده می‌شود. آبجکت مورد نظر رو که دارای خواص و مقادیر کپی شده از آبجکت مورد نظر است، برمی‌گردونه.

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

بیاین با یه منبع و یه 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 }
```

همونطور که در کد بالا مشاهده شد، یه ویژگی مشترک ( `b` ) از منبع به مقصد وجود دارد، بنابراین مقداش بازنویسی شده.

## ۱۹۳. کاربردهای ممتدهای `assign` چیه؟

۱. برای شبیه سازی یه شی استفاده می‌شود.
۲. برای ادغام اشیاء با ویژگی‌های یکسان استفاده می‌شود.

## ۱۹۴. آبجکت proxy چیه؟

آبجکت Proxy برای تعریف رفتار سفارشی برای عملیات‌های اساسی مثل جستجوی ویژگی، تخصیص، شمارش، فراخوانی تابع و غیره استفاده می‌شه.

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

بیاین 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 کنترل‌کننده «get» استفاده می‌کنه که رفتار پراکسی رو موقع انجام عملیات روی اون تعریف می‌کنه.

## ۱۹۵. هدف از متد seal چیه؟

روش **Object.seal** برای مهر و موم کردن یه آبجکت با جلوگیری از اضافه شدن ویژگی‌های جدید بهش و علامت گذاری تمام ویژگی‌های موجود به عنوان غیر قابل تنظیم، استفاده می‌شه. اما مقادیر پراپرتی‌های فعلی تا زمانی که قابل نوشتمن باشن، قابل تغییر هستن. بیاین مثال زیرو 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

```

## ۱۹۶. کاربردهای متدهای seal چیه؟

۱. برای آب بندی آبجکت‌ها و آرایه‌ها استفاده می‌شود.
۲. برای غیرقابل تغییر کردن یه آبجکت استفاده می‌شود.

## ۱۹۷. تفاوت‌های بین متدهای seal و freeze چیا هست؟

اگه یه آبجکت با استفاده از متدهای Object.freeze منجمد شه، ویژگی‌هاش تغییرناپذیر می‌شون و هیچ تغییری در اونا نمیتوانیم ایجاد کنیم در حالی که اگه یه آبجکت با استفاده از متدهای Object.seal مهر و موم شده باشه، می‌توان تغییرات رو در ویژگی‌های موجود ایجاد کرد. از آبجکت

## ۱۹۸. چطوری تشخیص میدی که یه آبجکت seal شده یا نه؟

- متدهای Object.isSealed برای تعیین مهر و موم بودن یا نبودن یه آبجکت استفاده می‌شوند.
- اگه همه شرایط زیر درست باشه یه شی مهر و موم می‌شود:
۱. اگه قابل توسعه نباشه.
  ۲. اگه تمام خصوصیات اون غیر قابل تنظیم باشن.
  ۳. اگه قابل جابجایی نباشه (اما لزوماً غیرقابل نوشتن نیست).
- بیاین اونو 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

```

## ۱۹۹. چطوری کلید و مقدارهای enumerable رو به دست میاری؟

متد Object.entries برای برگرداندن آرایه‌ای از جفت‌های [key, value] دارای کلید رشته‌ای شمارش‌پذیر یه شی معین، به همان ترتیبی که توسط یه حلقه for...in ارائه می‌شود، استفاده می‌شود. بیان عملکرد متد object.entries را تو یه مثال بینیم، متد Object.entries برای برگرداندن آرایه‌ای از جفت‌های [key, value] دارای کلید رشته‌ای شمارش‌پذیر یه شی معین، به همون ترتیبی که توسط یه حلقه for...in ارائه می‌شه، استفاده می‌شه. بیان عملکرد متد object.entries رو تو یه 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 نمی‌شه.

## ۲۰۰. تفاوت‌های بین متدهای Object.entries و Object.values چیا هست؟

رفتار متد Object.entries مشابه روش Object.values هست اما به جای جفت 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
}

```

## ۱۰۰. چطوری لیست کلیدهای یه object رو بدست میاري؟

میتونین از متده استفاده کنین که برای برگرداندن آرایه‌اي از اسم ویژگی‌های يه شی معين استفاده می‌شه، به همون ترتیبی که با يه حلقة معمولی دریافت می‌کنیم.  
برای مثال:

```

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

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

```

## ۱۰۱. چطوری یه object با prototype درست می‌کنی؟

متده Object.create برای ایجاد یه object جدید با object نمونه اولیه و ویژگی‌های مشخص شده استفاده می‌شه. برای مثال، از يه object موجود به عنوان نمونه اولیه object جدید ایجاد شده استفاده می‌کنه. يه object رو با object نمونه اولیه و ویژگی‌های مشخص شده برمی‌گردونه.

```

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

```

## ۲۰۳. چیه؟ WeakSet

برای ذخیره مجموعه ای از اشیاء ضعیف (مرجع ضعیف) استفاده می شه.

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

بیاین مثال زیرو برای توضیح رفتارش 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

```

## ۲۰۴. تفاوت های بین Set و WeakSet کدوما هستن؟

تفاوت اصلی اينه که ارجاع به اشیاء تو Set قويه در حالی که ارجاع به اشیا تو WeakSet ضعیفه. برای مثال، يه شی تو WeakSet ميتوونه زباله جمع آوری شه اگه مرجع دیگری به اون وجود نداشته باشه.

تفاوت های دیگر عبارتند از

۱. مجموعه ها هر مقداری رو ذخیره کنن در حالی که WeakSets ميتوونه تنها مجموعه ای از اشیاء رو ذخیره کنه

برخلاف Set دارای ویژگی اندازه نیست.  
WeakSet متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی‌ها، forEach را  
نداره.  
قابل تکرار نیست.

## ۲۰۵. لیست متدهایی که رو WeakSet قابل استفاده هستن رو می‌توانی بگی؟

در زیر لیستی از روش‌های موجود در WeakSet آمده است،  
یه شی جدید با مقدار داده شده به مجموعه ضعیف اضافه می‌شه

مقدار رو از مجموعه WeakSet حذف می‌کنه.  
اگه مقدار در مجموعه WeakSet وجود داشته باشه رو برمنی‌گردونه در غیر این صورت false رو برمنی‌گردونه.  
طول ضعیف SetObject رو برمنی‌گردونه length.  
بیاین عملکرد تمام روش‌های بالا رو توی یه مثال بینیم،

```
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);
```

## ۲۰۶. WeakMap چیه؟

آبجکت WeakMap مجموعه‌ای از جفت‌های کلید/مقداره که تو اون کلیدها به صورت ضعیف ارجاع داده شدن. در این حالت، کلیدها باید اشیا باشن و مقادیر می‌توان مقادیر دلخواه باشن. برای مثال:

```
new WeakMap([iterable])
```

بیاین مثال زیرو برای توضیح رفتار اون بینیم،

```
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
```

## ۲۰۷. تفاوت‌های بین Map و WeakMap کدوما هستن؟

تفاوت اصلی اینه که ارجاعات به آبجکت‌ها کلیدی در نقشه قوی هستن در حالی که ارجاعات به اشیاء کلیدی در WeakMap ضعیف هستن. برای مثال، یه شی کلیدی در WeakMap در صورتی که هیچ مرجع دیگری بهش وجود نداشته باشه، میتونه زباله جمع آوری شه.

تفاوت‌های دیگر عبارتند از

۱. Map‌ها میتونن هر نوع کلیدی رو ذخیره کنن، در حالی که WeakMaps فقط

میتونه مجموعه ای از اشیاء کلیدی رو ذخیره کنه

۲. WeakMap برخلاف Map دارای ویژگی اندازه نیست

۳. WeakMap متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی‌ها، forEach رو نداره.

۴. WeakMap قابل تکرار نیست.

## ۲۰۸. لیست متدهایی که رو WeakMap قابل استفاده هستن رو می‌توانی بگی؟

۱. (set(key, value)): مقدار کلید رو در شی WeakMap تنظیم می‌کنه. شی WeakMap رو برمی‌گردونه.

۲. (delete(key)): هر مقدار مربوط به کلید رو حذف می‌کنه.

۳. (has(key)): یه Boolean رو برمی‌گردونه که نشون میده آیا مقداری به کلید در WeakMap مرتبط شده اس یا نه.

۴. (get(key)): مقدار مربوط به کلید رو برمی‌گردونه، یا اگه کلیدی وجود نداشته باشه، تعریف نشده.

بیاین عملکرد تمام روش‌های بالا رو تو یه 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);

```

## ۲۰۹. هدف از متده‌ی uneval چیه؟

uneval یه تابع داخلی‌هه که برای ایجاد نمایش رشته‌ای از کد منبع یه شی استفاده می‌شه. این یه تابع سطح بالاس و با هیچ آبجکت‌ای مرتبط نیست. بیاین مثال زیر رو درموردش ببینیم،

```

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

```

## ۲۱۰. چطوری یه URL رو encode می‌کنی؟

تابع encodeURI برای رمزگذاری URI کامل استفاده می‌شه که دارای کاراکترهای خاص به جز (, /, :, ?, @, #, \$, +, =, &) هست.

```

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

```

## ۲۱۱. چطوری یه URL رو decode می‌کنی؟

تابع decodeURI برای رمزگشایی یه شناسه منبع یکنواخت (URI) که قبلاً توسط encodeURI ایجاد شده اس استفاده می‌شه.

```

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);
}

```

## ۲۱۲. چطوری محتوای یه صفحه رو پرینت می‌گیری؟

شی window یه متده است print ارائه می‌کنه که برای چاپ محتویات پنجره فعلی استفاده می‌شه. یه کادر محاوره ای چاپ رو باز می‌کنه که به شما امکان میده بین گزینه‌های مختلف چاپ انتخاب کنین. بیاین استفاده از روش چاپ رو تو یه مثال بینیم،

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

نکته: در اکثر مرورگرها، زمانی که کادر گفتگوی چاپ باز است، مسدود می‌شه.

## ۲۱۳. تفاوت‌های بین eval و uneval چیا هستن؟

تابع 'uneval' منبع یه شی معین رو برمی‌گردونه. در حالی که تابع "eval" با ارزیابی اون کد منبع تو یه ناحیه حافظه متفاوت، برعکس عمل می‌کنه. بیاین مثالی رو برای روشن شدن تفاوت بینیم،

```

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

```

## ۲۱۴. تابع anonymous چیه؟

تابع ناشناس یه تابع بدون نام است! توابع ناشناس معمولاً به نام متغیر اختصاص داده می شن یا به عنوان یه تابع فراخوانی استفاده میشن بریم یه مثال در موردش ببینیم،

```
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
});
```

بیاین تابع ناشناس بالا رو تو یه مثال ببینیم،

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

## ۲۱۵. تفاوت تقدم بین متغیرهای global و local چطوریه؟

یه متغیر محلی بر یه متغیر سراسری با همین نام ارجحیت داره. بیاین این رفتار رو تو یه مثال ببینیم.

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

## ۲۱۶. accessor چیکار می کنن؟

ECMAScript 5 دسترسی های شی جاواسکریپت یا ویژگی های محاسبه شده رو از طریق گیرنده ها و تنظیم کننده ها معرفی کرد. Getters از کلمه کلیدی "get" استفاده می کنه در

حالی که از کلمه کلیدی "set" استفاده می‌کنی.

```
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
```

## ۲۱۷. چطوری روی Object یه مقدار تعریف می‌کنی؟

متده استاتیک Object.defineProperty برای تعریف یه ویژگی جدید به طور مستقیم بر روی یه آبجکت یا تغییر ویژگی موجود روی یه آبجکت استفاده می‌شه و آبجکت رو برمی‌گردونه. بیایین مثالی رو ببینیم تا بدونیم چجوری ویژگی رو 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
```

## ۲۱۸. تفاوت‌های بین defineProperty و get چیا هست؟

هر دو نتایج مشابهی دارن مگه اینکه از کلاس‌ها استفاده کنین. اگه از «`get`» استفاده می‌کنین ویژگی روی نمونه اولیه شی تعریف می‌شه، در حالی که با استفاده از «`Object.defineProperty`»، ویژگی روی نمونه‌ای که بهش اعمال می‌شه، تعریف می‌شه.

## ۲۱۹. مزایای استفاده از `Setter` و `Getter` چیه؟

۱. اونا نحو ساده تری ارائه می‌دهند
۲. اونا برای تعریف ویژگی‌های محاسبه شده یا دسترسی‌ها در JS استفاده می‌شن
۳. برای ارائه رابطه هم ارزی بین خواص و روش‌ها مفید است
۴. اونا می‌توانن کیفیت داده‌های بهتری رو ارائه دهند
۵. برای انجام کارها در پشت صحنه با منطق محصور شده مفید است.

## ۲۲۰. می‌تونیم `getter` و `setter` رو با استفاده از متدهای `defineProperty` کنیم؟

بله، می‌تونین از روش `Object.defineProperty` برای اضافه کردن `Setter` و `Getters` استفاده کنین. برای مثال، آبجکت شمارنده زیر از ویژگی‌های افزایش، کاهش، جمع و تفریق استفاده می‌کنه.

```

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

```

## ۲۲۱. هدف استفاده از switch-case چیه؟

عبارت switch case در جاواسکریپت برای اهداف تصمیم گیری استفاده می شه. در چند مورد، استفاده از دستور if-else راحتتر از switch case است. بریم یه مثال در موردش ببینیم،

```

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

```

دستور شعبه چند طرفه بالا یه راه آسان برای ارسال اجرا به قسمت‌های مختلف کد بر اساس مقدار عبارت ارائه میده.

## ۲۲۲. چه قواعدی برای استفاده از switch-case باید رعایت بشه؟

۱. عبارت میتوانه از نوع عددی یا رشته‌ای باشه.
۲. مقادیر تکراری برای عبارت مجاز نیستن.
۳. بیانیه پیش فرض اختیاری است. اگه عبارت ارسال شده به سوئیچ با هیچ مقدار case مطابقت نداشته باشه، دستور در حالت پیش فرض اجرا میشه.
۴. دستور break در داخل سوئیچ برای پایان دادن به دنباله دستور استفاده میشه.
۵. عبارت break اختیاری است. اما اگه حذف شه، اجرا در مورد بعدی ادامه پیدا میکنه

## ۲۲۳. نوع داده‌های primitive کدوما هستن؟

۵. یه نوع داده اولیه، داده‌ایه که دارای یه مقدار اولیه اس (که هیچ ویژگی یا روشی نداره).
۶. نوع نوع داده اولیه وجود داره.
۷. رشته‌ها

۲. اعداد

boolean.۳

null.۴

undefined.۵

## ۲۲۴. روش‌های مختلف دسترسی به object‌های property کدوما هستن؟

۱. از نقطه برای دسترسی به ویژگی‌ها استفاده می‌کنیم: **Dot notation**.

```
objectName.property
```

۲. از براکت‌های مربعی برای دسترسی به دیتا استفاده می‌کنیم: **Square brackets notation**.

استفاده می‌کنیم

```
objectName["property"]
```

۳. از عبارت در کروشه استفاده می‌کنیم: **Expression notation**.

```
[objectName[expression]
```

## ۲۲۵. قوانین پارامترهای توابع کدوما هستن؟

۱. تعاریف تابع انواع داده‌ها رو برای پارامترها مشخص نمی‌کنیم.

۲. بررسی نوع آرگومان‌های ارسال شده رو انجام ندیم.

۳. تعداد آرگومان‌های دریافتی رو بررسی نکنیم.

یعنی تابع زیر از قوانین بالا پیروی می‌کنیم،

```
function functionName(parameter1, parameter2, parameter3) {
 console.log(parameter1); // ۱
}
functionName(1);
```

## ۲۲۶. آبجکت error چیه؟

یه آبجکت خطای داخلیه که موقع بروز خطا، اطلاعات خطا رو ارائه میده.  
این دو ویژگی داره: نام و پیام. برای مثال، تابع زیر جزئیات خطا رو ثبت می‌کنه،

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

## ۲۲۷. چه موقعی خطای syntax دریافت می‌کنیم؟

اگه بخواهید کد رو با یه خطای نحوی ارزیابی کنین یه SyntaxError پرتاب می‌شه. برای مثال، نقل قول زیر برای پارامتر تابع یه خطای نحوی ایجاد می‌کنه

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

## ۲۲۸. عنوان خطاهای مختلف که روی error-object برمیگردن کدوما هستن؟

توضیحات	نام خطا
خطایی در تابع eval رخداده است	EvalError
خطایی با عدد "خارج از محدوده"	RangeError
خطا به دلیل ارجاع غیرقانونی	خطای مرجع
خطای ناشی از خطای نحو	SyntaxError
خطای ناشی از خطای نوع	TypeError

نام خطأ	توضيحات
URIError	يُظهر خطأ بدليل URI

## ۲۲۹. عبارات مختلف که در هنگام مدیریت error استفاده میشن کدوما هستن؟

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

## ۲۳۰. دو نوع مختلف حلقهها در جاواسکریپت کدوما هستن؟

۱. در این نوع حلقه، شرایط تست قبل از ورود به بدنه حلقه آزمایش میشه. برای مثال while Loop و For Loop در این دسته قرار میگیرن.
۲. در این نوع حلقه، شرایط تست در انتهای بدنه حلقه آزمایش یا ارزیابی میشه. یعنی بدنه حلقه حداقل یه بار بدون در نظر گرفتن شرایط تست true یا false اجرا میشه. برای مثال، حلقه do-while در این دسته قرار میگیره.

## ۲۳۱. چیه nodejs

Node.js یه پلتفرم سمت سروره که بر اساس زمان اجرا جاواسکریپت کروم برای ساخت آسان برنامههای شبکه سریع و مقیاس پذیر ساخته شده. این یه زمان اجرا ۱/۰ ناهمزمان مبتنی بر رویداد، غیر مسدود کننده اس که از موتور جاواسکریپت V8 گوگل و کتابخانه libuv استفاده میکنه.

## ۲۳۳۲. آبجکت Intl چیه؟

آبجکت Intl فضای نامی برای ECMAScript Internationalization API اس که مقایسه رشته‌های حساس زبان، قالب بندی اعداد و قالب بندی تاریخ و زمان رو ارائه میده. دسترسی به چندین سازنده و توابع حساس به زبان رو فراهم می‌کنه.

## ۲۳۳۳. چطوری تاریخ و زمان رو بر اساس زبان جاری سیستم کاربر نمایش بدیم؟

می‌توانیم از شی «Intl.DateTimeFormat» استفاده کنیم که سازنده آبجکت‌هاییه که قالب بندی تاریخ و زمان حساس به زبان رو فعال می‌کنه. بیاین این رفتار رو با یه 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
```

## ۲۳۴۰. Iterator چیه؟

آبجکت‌ایه که پس از خاتمه، یه توالی و یه مقدار بازگشتی رو تعریف می‌کنه. پروتکل Iterator رو با متده «next» پیاده‌سازی می‌کنه که یه شی رو با دو ویژگی برمی‌گردونه: «value» (مقدار بعدی در دنباله) و «done» (که اگه آخرین مقدار در دنباله مصرف شده باشه درسته.).

## ۲۳۴۵. حللهای (همزمان)synchronous چطوری کار می‌کنن؟

تکرار همزمان در ES6 معرفی شد و با مجموعه ای از اجزای زیر کار می‌کنه: **Iterable**: این یه آبجکت‌ایه که می‌توانه از طریق روشی که کلید اون `Symbol.iterator` هس تکرار شه.

**Iterator**: این یه آبجکت‌ایه که با فراخوانی «`[Symbol.iterator]`» بر روی یه تکرار برگردانده می‌شه. این آبجکت تکرار شونده هر عنصر تکرار شده رو تو یه آبجکت پیچیده می‌کنه و اونو

از طریق متده «next» یکی برمی‌گردونه.

**IteratorResult**: این یه آبجکت‌ایه که با متده «next» برگردانده می‌شه. آبجکت شامل دو ویژگی است. ویژگی "value" حاوی یه عنصر تکرار شده و ویژگی "done" تعیین می‌کنه که آیا عنصر آخرین عنصر هست یا نه.

بیاین تکرار همزمان رو با آرایه ای مانند زیر نشون بدیم،

```
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 }
```

## ۲۳۶. چیه؟ Event-loop

یه صف از توابع callback موقعی که یه تابع async اجرا می‌شه، تابع event loop در صف قرار می‌گیرد. موتور جاواسکریپت پردازش حلقه رویداد رو شروع نمی‌کنه تا زمانی که تابع async اجرای کد رو به پایان برساند.

**نکته:** این به Node.js اجازه میده تا عملیات ۰/۱ غیر مسدود کننده رو انجام بده حتی اگه جاواسکریپت تک رشته‌ای باشه.

## ۲۳۷. چیه؟ Call-stack

Call Stack یه ساختار داده برای مفسران جاواسکریپت‌ههای فراخوانی‌های تابع در برنامه رو پیگیری کنه. دو عمل عمده داره،

۱. هر زمان که یه تابع رو برای اجرای آن فراخوانی می‌کنین اوونو به پشته هل می‌بدین.

۲. هر زمان که اجرا به پایان برسد، تابع از پشته خارج می‌شه.  
بیاین مثالی بزنیم و نمایش حالت در قالب نمودار است

```

function hungry() {
 eatFruits();
}

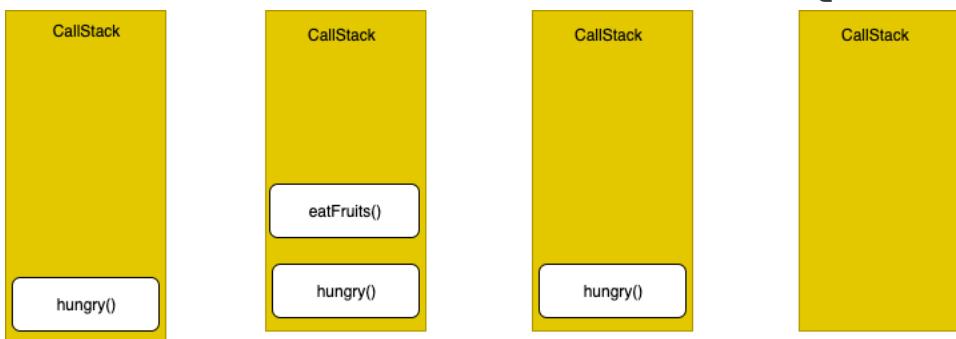
function eatFruits() {
 return "I'm eating fruits";
}

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

```

کد بالا تو یه پشته تماس به صورت زیر پردازش میشه.

۱. تابع 'hungry' رو به لیست پشته تماس اضافه کنین و کد رو اجرا کنین.
۲. تابع 'eatFruits' رو به لیست پشته تماس اضافه کنین و کد رو اجرا کنین.
۳. تابع 'eatFruits' رو از لیست پشته تماس ما حذف کنین.
۴. تابع 'eatFruits' رو از لیست پشته تماس ما حذف کنین.



## چیه؟ Event-queue .۲۳۸

## چیه؟ Decorator .۲۳۹

دکوراتور عبارتیه که یه تابع رو ارزیابی میکنه و هدف، نام و توصیف‌کننده تزئین رو به عنوان آرگومان می‌گیره. همچنین، به صورت اختیاری یه توصیفگر دکوراتور رو برای نصب بر روی آبچکت مورد نظر برمی‌گدونه. بیاین در زمان طراحی، دکوراتور ادمین رو 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

```

## ۲۴۰. مقادیر موجود روی آبجکت Intl کدوما هستن؟

۱. **Collator:** اینا آبجکت‌هایی هستن که مقایسه رشته‌های حساس به زبان رو امکان پذیر می‌کنن.
۲. **DateTimeFormat:** اینا آبجکت‌هایی هستن که قالب بندی تاریخ و زمان حساس به زبان رو فعال می‌کنن.
۳. **ListFormat\*\*:** اینا آبجکت‌هایی هستن که قالب بندی لیست حساس به زبان رو فعال می‌کنن.
۴. **NumberFormat:** آبجکت‌هایی که قالب بندی اعداد حساس به زبان رو فعال می‌کنن.
۵. **PluralRules\*\*:** آبجکت‌هایی که قالب بندی حساس به جمع و قوانین خاص زبان رو برای جمع فعال می‌کنن.
۶. **RelativeTimeFormat:** آبجکت‌هایی که قالب بندی زمان نسبی حساس به زبان رو فعال می‌کنن.

## ۲۴۱. عملگر Unary چیه؟

عملگر (+) برای تبدیل یه متغیر به عدد استفاده می‌شه. اگه متغیر قابل تبدیل نباشه، همچنان به عدد تبدیل می‌شه اما با مقدار NaN. بیاین این رفتار رو تو یه عمل ببینیم.

```
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
```

## ۲۴۲. چطوری المنتهای موجود تو یه آرایه رو مرتب می‌کنی؟

متدهای sort برای مرتب سازی عناصر یه آرایه در جای خود استفاده می‌شوند و آرایه مرتب شده را برمی‌گردونند. استفاده از مثال زیر خواهد بود،

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

## ۲۴۳. هدف از تابع مرتب‌سازی موقع استفاده از متدهای sort چیه؟

از compareFunction برای تعریف ترتیب مرتب سازی استفاده می‌شوند. اگه حذف شوند، عناصر آرایه به رشته تبدیل می‌شون، سپس بر اساس مقدار نقطه کد یونیکد هر کاراکتر مرتب می‌شون بیاین مثالی بزنیم تا کاربرد compareFunction رو ببینیم،

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

## ۲۴۴. چطوری آیتمهای یه آرایه رو معکوس مرتب کنیم؟

برای معکوس کردن عناصر یه آرایه می‌توانیم از متدهای reverse استفاده کنیم. این روش برای مرتب کردن یه آرایه به ترتیب نزولی مفید است. بیاین استفاده از متدهای reverse رو تو یه 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]
```

## ۲۴۵. چطوری حداقل و حداکثر مقدار یه آرایه رو بدست بیاریم؟

می‌تونین از روش‌های «Math.max» و «Math.min» روی متغیرهای آرایه برای یافتن حداقل و حداکثر عناصر تو یه آرایه استفاده کنین. بیاین دو تابع برای پیدا کردن مقدار min و max تو یه آرایه ایجاد کنیم.

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

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

## ۲۴۶. چطوری حداقل و حداکثر مقدار یه آرایه رو بدون استفاده از متدهای Math بدست بیاریم؟

ما می‌توانیم توابعی بنویسیم که تو یه آرایه حلقه می‌زنن و هر مقدار را با کمترین یا بالاترین مقدار مقایسه می‌کنن تا مقادیر حداقل و حداکثر رو پیدا کنن. بریم یه مثال درمودش ببینیم.

```

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));

```

## ۲۴۷. عبارت خالی چیه و هدف از استفاده ازش چیه؟

ویرگول (,) هس که نشون میده هیچ دستوری اجرا نخواهد شد، حتی اگه نحو جاوسکریپت به اون نیاز داشته باشه. از اونجایی که هیچ اقدامی با دستور خالی وجود نداره، ممکنه فکر کنین که استفاده از اون خیلی کمه اما دستور خالی گاهی اوقات موقعي مفیده که می خوابین حلقه ای ایجاد کنین که بدنش خالیه. برای مثال، می تونین یه آرایه با مقادیر صفر رو مانند زیر مقدارههی اولیه کنین.

```

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

```

## ۲۴۸. چطوری **meta data of a module** به ماژول رو بدست میاری؟

میتوانیم از آبجکت `import.meta` استفاده کنیم که یه ویژگی متاعه که متا داده های متنی خاص رو تو یه ماژول جاوا اسکریپت قرار می ده. این شامل اطلاعاتی در مورد ماژول فعلی، مانند URL ماژوله. در مرورگرها، ممکنه متا داده های متفاوتی نسبت به NodeJS دریافت کنیم.

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

## ۲۴۹. عملگر **comma** چیه و چیکار می کنه؟

عملگر کاما برای ارزیابی هر یه از عملوندهاش از چپ به راست استفاده می شه و مقدار آخرین عملوند رو برمی گردونه. این کاملاً با استفاده از کاما در آرایه ها، اشیاء و آرگومان ها و پارامتر های تابع متفاوته. بریم یه مثال در موردش بینیم.

```
var x = 1;
x = (x++, x);

console.log(x); // 2
```

## ۲۵۰. مزایای استفاده از عملگر **comma** چیه؟

معمولاً برای گنجوندن چندین عبارت در مکانی که یه عبارت واحد نیاز داره استفاده می شه. یکی از کاربردهای رایج این عملگر کاما، ارائه چندین پارامتر تو یه حلقه «`for`» است. برای مثال، حلقه `for` زیر از چند عبارت تو یه مکان واحد با استفاده از عملگر کاما استفاده می کنه.

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

همچنین می تونیم از عملگر کاما تو یه عبارت بازگشتی استفاده کنیم جایی که قبل از بازگشت پردازش می کنه.

```

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

```

## ۲۵۱. چیه؟ Typescript

TypeScript یه ابر مجموعه تایپ شده از جاوااسکریپت ه است که توسط مايكروسافت ایجاد شده که انواع اختیاری، کلاس ها، و بسیاری ویژگی های دیگر را اضافه می کنند و به جاوااسکریپت ساده کامپایل می کنند. Angular به طور کامل در TypeScript ساخته شده و به عنوان زبان اصلی استفاده می شود. شما می تونین اونو به صورت گلوبال نصب کنید

```
npm install -g typescript
```

بیان یه مثال ساده از استفاده از TypeScript رو بینیم،

```

function greeting(name: string): string {
 return "Hello, " + name;
}

let user = "Sudheer";

console.log(greeting(user));

```

متد greeting فقط نوع رشته رو به عنوان آرگومان مجاز می کند.

## ۲۵۲. تفاوت های بین javascript و typescript کدوما هستن؟

جاوااسکریپت	تایپ	ویژگی
زبان اسکریپت	زبان برنامه نویسی شی گرا	پارادایم زبان
دارای تایپ پویا	پشتیبانی از تایپ استاتیک	پشتیبانی از تایپ

جاواسکریپت	تاپ	ویژگی
پشتیبانی نمی‌شه	پشتیبانی شده	ماژول‌ها
از رابط‌ها پشتیبانی نمی‌کنه	دارای مفهوم رابط	رابط
عدم پشتیبانی از پارامترهای اختیاری اختیاری برای توابع	توابع از پارامترهای اختیاری پشتیبانی می‌کنن	پارامترهای اختیاری

## ۲۵۳. مزایای javascript نسبت به typescript چیاست؟

۱. میتوانه خطاهای زمان کامپایل رو فقط در زمان توسعه پیدا کنه و باعث می‌شه خطاهای زمان اجرا کمتر شه. در حالی که جاواسکریپت یه زبان تفسیر شده است.
۲. TypeScript به شدت تایپ می‌شه یا از تایپ استاتیک پشتیبانی می‌کنه که امکان بررسی صحت نوع رو در زمان کامپایل فراهم می‌کنه. این در جاواسکریپت در دسترس نیست.
۳. کامپایلر TypeScript برخلاف ویژگی‌های ES6 جاواسکریپت که ممکنه در بعضی از مرورگرها پشتیبانی نشه، میتوانه فایل‌های ts. رو در ES3، ES4، ES5 کامپایل کنه.

## ۲۵۴. object-initializer چیه؟

یه آبجکت اولیه عبارتیه که مقدار دهی اولیه یه آبجکت رو توصیف می‌کنه. نحو این عبارت به صورت فهرستی با کاما از صفر یا چند جفت نام ویژگی و مقادیر مرتبط یه آبجکت، محصور در پرانتزهای فرفری ({} ) نشون داده می‌شه. این همچنین به عنوان نماد تحت اللفظی شناخته می‌شه. یکی از راههای ایجاد یه آبجکته.

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

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

## ۲۵۵. متد constructor چیه؟

متد سازنده یه متد خاص برای ایجاد و مقدارههی اولیه یه آبجکت ایجاد شده تو یه کلاس است. اگه متد سازنده رو مشخص نکنین از سازنده پیش فرض استفاده میشه. بریم یه مثال در موردش ببینیم.

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

var employeeObject = new Employee();

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

## ۲۵۶. اگه متد constructor رو بیش از یه بار توی کلاس بنویسیم چی میشه؟

"سازنده" تو یه کلاس یه متد خاصه و باید فقط یه بار تو یه کلاس تعریف شه. برای مثال، اگه یه متد سازنده رو بیش از یه بار تو یه کلاس بنویسین، یه خطای «SyntaxError» ایجاد میکنه.

```
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);
```

## ۲۵۷. چطوری متد constructor کلاس والد رو صدا بزنیم؟

می‌توانیں از کلمه کلیدی `super` برای فراخوانی سازنده کلاس والد استفاده کنیں. به یاد داشته باشیم که «`super`» باید قبل از استفاده از مرجع «`this`» فراخوانی شه. در غیر این صورت باعث خطای مرجع می‌شه. بیاین از اون 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;
 }
}
```

## ۲۵۸. چطوری `object` یه `prototype` رو به دست میاری؟

می‌توانیں از روش `Object.getPrototypeOf(obj)` برای برگرداندن نمونه اولیه شی مشخص شده استفاده کنیں. یعنی مقدار ویژگی «نمونه اولیه» داخلی. اگه هیچ ویژگی ارشی وجود نداشته باشه، مقدار "null" برگردانده می‌شه.

```
;{} = const newPrototype
;const newObject = Object.create(newPrototype)

console.log(Object.getPrototypeOf(newObject) === newPrototype);
// true
```

## ۲۵۹. اگه به متده `getPrototypeOf` رشته پاس بدیم چی می‌شه؟

در ES5، اگه پارامتر `obj` یه شی نباشه، یه استثنای `TypeError` ایجاد می‌کنه. در حالی که در ES2015، پارامتر به یه «شیء» اجباری می‌شه.

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

## ۲۶۰. چطوری object روی یه object دیگه ست کنیم؟

میتوانیں از متده استفاده کنین که نمونه اولیه (یعنی ویژگی داخلی «Prototype») یه شی مشخص شده رو روی یه شی دیگر یا تهی تنظیم میکنید. برای مثال، اگه میخواین نمونه اولیه یه جسم مربع رو روی شی مستطیلی تنظیم کنین به صورت زیر است:

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

## ۲۶۱. چطوری بررسی میکنی که یه object قابل extend هست یا نه؟

متده `Object.isExtensible` برای تعیین اینکه یه شی قابل توسعه هس یا نه استفاده میشه. یعنی اینکه میتوانه ویژگیهای جدیدی به اون اضافه شه یا نه.

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

**نکته:** به طور پیش فرض، تمام اشیاء قابل گسترش هستن. برای مثال، ویژگیهای جدید رو میتوانیم اضافه یا تغییر بدیم.

## ۲۶۲. چطوری جلوی object رو بگیریم؟

متده «Object.preventExtensions» برای جلوگیری از افزودن ویژگیهای جدید به یه شی استفاده میشه. به عبارت دیگر، از پسوندهای بعدی به شی جلوگیری میکند. بیاین استفاده از این ویژگی رو ببینیم،

```

const newObject = {};
Object.preventExtensions(newObject); // NOT extendable

try {
 Object.defineProperty(newObject, 'newProperty', { // Adding
 new property
 value: 100
 });
} catch (e) {
 console.log(e); // TypeError: Cannot define property
 newProperty, object is not extensible
}

```

## ۲۶۳. روش‌های مختلف برای تبدیل یه object به غیرقابل extend چیه؟

شما می‌توانید یه شی غیر قابل گسترش رو به ۳ روش علامت گذاری کنید.

۱. Object.preventExtensions

۲. Object.seal

۳. Object.freeze

```

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

```

## ۲۶۴. چطوری property‌های متعددی رو روی یه object تعریف می‌کنی؟

متده «Object.defineProperties» برای تعریف یا اصلاح ویژگیهای موجود مستقیماً روی یه شی و برگرداندن شی استفاده میشه. بیاین چندین ویژگی رو روی یه شی خالی تعریف کنیم،

```
const newObject = {};

Object.defineProperties(newObject, {
 newProperty1: {
 value: 'John',
 writable: true
 },
 newProperty2: {}
});
```

## ۲۶۵. منظور از MEAN توی جاواسکریپت چیه؟

پشته نرم افزار جاواسکریپت منبع بازه که برای ساخت برنامه های وب پویا در دسترسه، جایی که می تونین نیمه های سمت سرور و سمت مشتری پروژه وب رو بنویسین. کاملا در جاواسکریپت

## ۲۶۶. منظور از Obfuscation توی جاواسکریپت چیه و چیکار می کنه؟

میهم سازی عمل عمدی ایجاد کد جاواسکریپت مبهم (یعنی کد منبع یا ماشین) هس که درک اون برای انسان سخته. این چیزی شبیه به رمزگذاریه، اما یه ماشین میتونه کد رو درک کنه و اونو اجرا کنه.

بیاین تابع زیر رو قبل از Obfuscation ببینیم،

```
function greeting() {
 console.log('Hello, welcome to JS world');
}
```

و بعد از کد Obfuscation به صورت زیر ظاهر می شه

```

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
ole|greeting|function|log|Hello|JS|to|welcom
e|world'.split('')),0
({})

```

## ۲۶۷. چه نیازی به Obfuscate کردن داریم؟

۱. اندازه کد کمتر میشه. بنابراین انتقال داده بین سرور و مشتری سریع تر انجام میشه.
۲. این منطق کسب و کار رو از دنیای خارج پنهان میکنه و از کد در برابر دیگران محافظت میکنه
۳. مهندسی معکوس کردن کد رو سخت میکنه
۴. زمان دانلود کاهش پیدا میکنه

## ۲۶۸. چیه؟ Minification

حذف تمام کاراکترهای غیر ضروریه (فضاهای خالی حذف میشن) و Minification متغیرها بدون تغییر در عملکر اون تغییر نام می دن همچنین نوعی مهمه.

## ۲۶۹. مزایای minification یا کم حجم سازی چیه؟

- به طور معمول توصیه میشه برای ترافیک سنگین و نیازهای فشرده منابع از Minification استفاده کنین. اندازه فایل رو با مزایای زیر کاهش میده
۱. زمان بارگذاری یه صفحه وب رو کاهش میده
  ۲. در مصرف پهنای باند صرفه جویی میکنه

## ۲۷۰. تفاوت‌های بین Encryption و Obfuscation چیه؟

رمزنگاری	مبهم سازی	ویژگی
تغییر فرم اطلاعات به فرمت ناخوانا با استفاده از کلید	تغییر فرم هر داده به هر شکل دیگر	تعریف
لازم	می‌شه اونو بدون هیچ کلید رمزگشایی کرد	کلیدی برای رمزگشایی
تبديل به فرمت ناخوانا	به فرم پیچیده تبدیل می‌شه	فرمت داده‌های هدف

## ۲۷۱. ابزارهای مختلف برای minification کدوما هستن؟

۱. کامپایلر بسته شدن گوگل UglifyJS2
۲. jsmin
۳. /javascript-minifier.com
۴. prettydiff.com
- ۵.

## ۲۷۲. چطوری اعتبارسنجی فرم رو با javascript انجام میدی؟

از جاواسکریپت می‌شه برای اعتبارسنجی فرم HTML استفاده کرد. برای مثال، اگه فیلد فرم خالی باشه، تابع باید اطلاع بده و false رو برگرداند تا از ارسال فرم جلوگیری شه.  
اجازه بدین ورود کاربر رو در فرم html انجام بدیم،

```
<form name="myForm" onsubmit="return validateForm()">
 method="post">
 User name: <input type="text" name="uname">
 <input type="submit" value="Submit">
</form>
```

و اعتبارسنجی 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;
 }
}
```

## ۳۷۳. چطوری اعتبارسنجی فرم رو بدون javascript انجام میدی؟

می‌تونین بدون استفاده از جاواسکریپت اعتبارسنجی فرم HTML رو به صورت خودکار انجام بدین. اعتبارسنجی با اعمال ویژگی «لازم» برای جلوگیری از ارسال فرم زمانی که ورودی خالیه فعال می‌شه.

```
<form method="post">
 <input type="text" name="uname" required>
 <input type="submit" value="Submit">
</form>
```

نکته: اعتبارسنجی فرم خودکار در اینترنت اکسپلورر 9 یا قبل از اون کار نمی‌کنه.

## ۳۷۴. متدهای موجود روی DOM برای اعتبارسنجی کدوما هستن؟

روش‌های DOM زیر برای اعتبارسنجی محدودیت در ورودی نامعتبر موجود است.  
۱. checkValidity: اگه یه عنصر ورودی حاوی داده‌های معتبر باشه، مقدار true رو برمی‌گردونه.

۲. setCustomValidity: برای تنظیم خاصیت validationMessage یه عنصر ورودی استفاده می‌شه.

بیاین یه فرم ورود کاربر با اعتبارسنجی 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";
 }
}

```

## ۲۷۵. مقادیر موجود روی DOM برای اعتبارسنجی کدوما هستن؟

در زیر لیستی از بعضی از ویژگی‌های DOM اعتبارسنجی محدودیت موجود است،  
۱. validity: فهرستی از ویژگی‌های بولین مربوط به اعتبار یه عنصر ورودی رو ارائه میده.

- ۲. validationMessage: زمانی که اعتبار نادرست باشه، پیام رو نمایش میده.
- ۳. willValidate: این نشون میده که آیا یه عنصر ورودی اعتبار سنجی می‌شه یا نه.

## ۲۷۶. مقادیر موجود روی input برای اعتبارسنجی کدوما هستن؟

ویژگی اعتبار یه عنصر ورودی مجموعه ای از ویژگی‌های مربوط به اعتبار داده‌ها رو ارائه میده.  
۱. customError: اگه یه پیام اعتبار سفارشی تنظیم شده باشه، true رو برمی‌گردونه.

- ۲. patternMismatch: اگه مقدار یه عنصر با ویژگی الگوی آن مطابقت نداشته باشه، مقدار true رو برمی‌گردونه.

۳. rangeOverflow: اگه مقدار یه عنصر از ویژگی max آن بیشتر باشه، مقدار true رو برمی‌گردونه.

- ۴. rangeUnderflow: اگه مقدار یه عنصر کمتر از ویژگی min باشه، مقدار true رو برمی‌گردونه.

۵. stepMismatch: اگه مقدار عنصر مطابق با ویژگی step نامعتبر باشه، مقدار true رو برمی‌گردونه.

.۶: اگه مقدار یه عنصر از ویژگی maxLength آن بیشتر شه، مقدار true رو برمی‌گردونه.

.۷: اگه مقدار یه عنصر بر اساس ویژگی نوع نامعتبر باشه، مقدار true رو برمی‌گردونه.

.۸: اگه عنصری با ویژگی مورد نیاز ارزش نداشته باشه، مقدار true رو برمی‌گردونه.

.۹: اگه مقدار یه عنصر معتبر باشه، مقدار true رو برمی‌گردونه.

## ۲۷۷. یه مثال از استفاده ویژگی rangeOverflow می‌تونی بزنی؟

اگه مقدار یه عنصر از ویژگی max آن بیشتر باشه، ویژگی rangeOverflow مقدار true رو برمی‌گردونه. برای مثال، فرم ارسالی زیر اگه مقدار آن بیش از 100 باشه، خطا میده.

```
<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");
 }
}
```

## ۲۷۸. جاواسکریپت قابلیت استفاده از enum رو پیش‌فرض توی خودش داره؟

نه، جاواسکریپت به صورت بومی از enumها پشتیبانی نمی‌کنه. اما انواع مختلفی از راه‌حل‌ها برای شبیه‌سازی اونا وجود داره، اگرچه ممکنه معادلهای دقیقی ارائه نکنن. برای مثال، می‌تونین از فریز یا مهر و موم روی شی استفاده کنین

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

enum نوعیه که متغیرها رو به یه مقدار از مجموعه ای از ثابت‌های از پیش تعریف شده محدود می‌کنه. جاواسکریپت هیچ enum نداره اما تایپ اسکریپت از enum داخلی پشتیبانی می‌کنه.

```
} enum Color
RED, GREEN, BLUE
{
```

## ۲۸۰. چطوری همه property‌های یه object رو به دست بیاریم؟

می‌تونین از متند «Object.getOwnPropertyNames» استفاده کنین که آرایه‌ای از تمام ویژگی‌هایی رو که مستقیماً تو یه شیء داده شده یافت می‌شه، برمنی‌گردونه. بیاین استفاده از اونو تو یه مثال بیان کنیم،

```
const newObject = {
 a: 1,
 b: 2,
 c: 3
};

console.log(Object.getOwnPropertyNames(newObject)); ["a", "b",
"c"]
```

## ۲۸۱. چطوری get property descriptors of an object

می‌تونین از متند «Object.getOwnPropertyDescriptors» استفاده کنین که تمام توصیف‌گرهای ویژگی یه شی معین رو برمنی‌گردونه. 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

```

## ۲۸۲. گزینه‌هایی که موقع تعریف ویژگی object با descriptor داریم کدوما هستن؟

۱. ارزش مرتبط با ملک: value
۲. تعیین می‌کند که آیا مقدار مرتبط با ویژگی قابل تغییر هس یا نه: writable
۳. اگه بتوان نوع توصیفگر این ویژگی رو تغییر داد و اگه ویژگی رو بتوان از شی مربوطه حذف کرد، مقدار true رو برمی‌گردونه: configurable
۴. تعیین می‌کند که آیا ویژگی در موقع شمارش خصوصیات روی شی مربوطه ظاهر می‌شه یا نه: enumerable
۵. تابعی که به عنوان تنظیم کننده برای ویژگی عمل می‌کند: set
۶. تابعی که به عنوان یه گیرنده برای ملک عمل می‌کند: get

## ۲۸۳. چطوری کلاس‌ها رو extend می‌کنی؟

کلمه کلیدی "extends" در اعلان‌ها/عبارات کلاس برای ایجاد کلاسی که فرزند کلاس دیگه ایه استفاده می‌شه. می‌شه از اون برای زیر کلاس بندی کلاس‌های سفارشی و همچنین اشیاء داخلی استفاده کرد. بریم یه مثال در موردش ببینیم،

```
class ChildClass extends ParentClass { ... }
```

بیاین یه نمونه از زیر کلاس مربع از کلاس والد Polygon رو 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;
 }
}

```

## ۲۸۴. چطوری آدرس صفحه رو بدون رفرش صفحه عوض کنیم؟

ویژگی «window.location.url» برای تغییر url مفید خواهد بود اما صفحه رو دوباره بارگیری می‌کنه. HTML5 متدهای «history.pushState» و «history.replaceState» را معرفی کرد که به شما اجازه میده به ترتیب ورودی‌های تاریخ رو اضافه و تغییر بدین. برای مثال، می‌توانیم از pushState مانند زیر استفاده کنیم.

```
window.history.pushState('page2', 'Title', '/page2.html');
```

## ۲۸۵. چطوری بررسی می‌کنی که یه آرایه یه مقدار مشخص رو داره یا نه؟

متد «Array#includes» برای تعیین اینکه آیا یه آرایه دارای مقدار خاصی در میان ورودی‌های خود با برگرداندن true یا false هس یا نه استفاده می‌شه. بیاین مثالی برای یافتن یه عنصر (عددی و رشته‌ای) تو یه آرایه ببینیم.

```

var numericArray = [1, 2, 3, 4];
console.log(numericArray.includes(3)); // true

var stringArray = ['green', 'yellow', 'blue'];
console.log(stringArray.includes('blue')) //true

```

## ۲۸۶. چطوری آرایه‌های scalar رو با هم مقایسه می‌کنی؟

می‌توانیں از طول و هر روش آرایه برای مقایسه دو آرایه اسکالار (مقایسه مستقیم با استفاده از `==`) استفاده کنین. ترکیب این عبارات می‌توانه نتیجه مورد انتظار رو به دست بده،

```
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
```

اگه می‌خواین آرایه‌ها رو بدون توجه به ترتیب مقایسه کنین باید اوナ رو قبل از مرتب سازی، مرتب کنین.

```
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
```

## ۲۸۷. چطوری می‌شه پارامترهای صفحه رو از متدهای GET گرفت؟

شیء «URL» رشته url رو می‌پذیرد و از ویژگی «searchParams» این شی می‌توان برای دسترسی به پارامترهای get استفاده کرد. به یاد داشته باشیم که ممکنه برای دسترسی به «window.location» در مرورگرهای قدیمی (از جمله IE) نیاز به استفاده از polyfill یا داشته باشیم.

```
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
```

## ۲۸۸. چطوری اعداد رو می‌شه سه رقم سه رقم جدا کرد؟

می‌توانیں از متدهای «Number.prototype.toLocaleString» استفاده کنین که رشته‌ای رو با نمایشی حساس به زبان مانند جداکننده هزار، ارز و غیره از این عدد برمی‌گردونه.

```

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

console.log(convertToThousandFormat(12345.6789));

```

## ۲۸۹. تفاوت بین javascript و java چیه؟

هر دو زبان برنامه نویسی کاملاً نامرتب هستن و هیچ ارتباطی بین اونا وجود نداره. جاوا بصورت ایستا تایپ می‌شه، کامپایل می‌شه، روی ماشین مجازی خود اجرا می‌شه. در حالی که جاواسکریپت به صورت پویا تایپ می‌شه، تفسیر می‌شه و در محیط‌های مرورگر و nodejs اجرا می‌شه. بیان تفاوت‌های عمدۀ رو در قالب جدولی ببینیم،

جاواسکریپت	جاوا	ویژگی
این یه زبان تایپ شده پویاسن	این یه زبان قوی تایپ شده اس	تایپ شده
برنامه نویسی مبتنی بر نمونه اولیه	برنامه نویسی شی گرا	پارادایم
محدوده عملکردی	محدوده بلوك	محدوده
مبتنی بر رویداد	بر اساس موضوع	همزمانی
از حافظه کمتری استفاده می‌کنه. از این رو برای صفحات وب استفاده خواهد شد	از حافظه بیشتر استفاده می‌کنه	حافظه

## ۲۹۰. آیا جاواسکریپت namespace را پشتیبانی می‌کنه؟

جاواسکریپت به طور پیش فرض از فضای نام پشتیبانی نمی‌کنه. بنابراین اگه هر عنصری (تابع، روش، شی، متغیر) ایجاد کنین گلوبال می‌شه و فضای نام گلوبال رو آلوده می‌کنه. بیان مثالی از تعریف دو تابع بدون فضای نام بزنیم،

```

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 رو فراخوانی می‌کنه. در این صورت فضای نام مشکل برخورد نام رو حل می‌کنه.

## ۲۹۱. چطوری namespace تعریف می‌کنی؟

حتی اگه جاواسکریپت فاقد فضاهای نام باشه، می‌تونیم از IIFE برای ایجاد فضاهای نام استفاده کنیم.

۱. **Using Object Literal Notation**: بیاین متغیرها و توابع رو درون یه Object literal ببیچیم که به عنوان فضای نام عمل می‌کنه. پس از اون می‌تونین با استفاده از نماد شیء به اونا دسترسی داشته باشیم

```

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

```

۲. **(Using IIFE (Immediately invoked function expression))**: جفت پرانتز

IIFE یه محدوده محلی برای تمام کدهای داخل آن ایجاد می‌کنه و تابع ناشناس رو به یه عبارت تابع تبدیل می‌کنه. به همین دلیل، می‌تونین یه تابع رو در دو عبارت تابع مختلف ایجاد کنین تا به عنوان فضای نام عمل کنه.

```

(function() {
 function fun1(){
 console.log("This is a first definition");
 } fun1();
}());

(function() {
 function fun1(){
 console.log("This is a second definition");
 } fun1();
}());

```

در ۶ ECMAScript، شما

می‌توانید به سادگی از یه بلوک و یه اعلان let برای محدود کردن دامنه یه متغیر به یه 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.

```

۲۹۲. چطوری می‌توانید تکه کد جاواسکریپت داخل یه iframe رو از صفحه والد صدا بزنید؟

در ابتدا باید iFrame با استفاده از «document.getElementById» یا «window.frames» قابل دسترسی باشید. پس از آن ویژگی «targetFunction» به contentWindow ای فریم دسترسی میدهید

```
document.getElementById('targetFrame').contentWindow.targetFunction();

window.frames[0].frameElement.contentWindow.targetFunction();
// Accessing iframe this way may not work in latest versions
chrome and firefox
```

## ۲۹۳. چطوری میشه اختلاف date رو از آبجکت timezone بگیریم؟

می‌تونیں از روش «getTimezoneOffset» شی تاریخ استفاده کنین. این روش اختلاف منطقه زمانی رو بر حسب دقیقه از محلی فعلی (تنظیمات سیستم میزبان) به UTC برمی‌گردونه

```
var offset = new Date().getTimezoneOffset();
console.log(offset); // -480
```

## ۲۹۴. چطوری فایل‌های CSS و JS رو به شکل داینامیک بارگذاری کنیم؟

شما می‌تونین هر دو عنصر پیوند و اسکریپت رو در DOM ایجاد کنین و اوها رو به عنوان فرزند به تگ head اضافه کنین. بیایین یه تابع برای اضافه کردن منابع اسکریپت و سبک مثل 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)
}
```

## ۲۹۵. روش‌های مختلف برای پیدا کردن element‌ها توی DOM کدوما هستن؟

اگه می‌خواین به هر عنصری در صفحه HTML دسترسی داشته باشیم، باید با دسترسی به شی سند شروع کنین. بعداً می‌تونین از یکی از روش‌های زیر برای یافتن عنصر HTML استفاده کنین.

۱. (document.getElementById(id)): یه عنصر رو با id پیدا می‌کنه

۲. (document.getElementsByTagName(name)): یه عنصر رو با نام تگ پیدا می‌کنه

۳. (document.getElementsByClassName(name)): یک عنصر رو با نام کلاس پیدا می‌کنه

## ۲۹۶. jQuery چیه؟

jQuery یه کتابخانه جاواسکریپت متقابل مرورگر محبوبه که با به حداقل رسوندن اختلاف بین مرورگرهای پیمایش مدل شی سند (DOM)، مدیریت رویداد، اینیمیشن‌ها و تعاملات AJAX رو فراهم می‌کنه. با فلسفه اشن «کمتر بنویس، بیشتر انجام بده» شهرت زیادی داره. برای مثال، می‌تونین پیام خوش آمدگویی رو در بارگذاری صفحه با استفاده از jQuery به صورت زیر نمایش بدین.

```
$(document).ready(function(){ // It selects the document and
 apply the function on page load
 alert('Welcome to jQuery world');
});
```

\*نکته: می‌تونین اونو از سایت رسمی jquery دانلود کنین یا از CDN‌ها مانند گوگل نصب کنین.

## ۲۹۷. موتور V8 چیه؟

V8 یه موتور جاواسکریپت با کارایی بالا منبع بازه که توسط مرورگر Google Chrome استفاده می‌شه و به زبان C++ نوشته شده است. همچنین در پروژه node.js استفاده می‌شه. ECMAScript و WebAssembly را پیاده‌سازی می‌کنه و روی ویندوز 7 یا بالاتر، macOS 10.12 و سیستم‌های لینوکس که از پردازنده‌های ARM x64، IA-32، MIPS +macOS پشتیبانی می‌کنند.

استفاده میکن اجرا میشه.

نکته: میتونه به صورت مستقل اجرا شه یا میتونه در هر برنامه C++ تعبیه شه.

## ۲۹۸. چرا ما جاواسکریپت رو به عنوان یه زبان داینامیک میشناسیم؟

جاواسکریپت یه زبان ساده تایپ شده یا یه زبان پویاue چون متغیرها در جاواسکریپت مستقیماً با هیچ نوع مقدار خاصی مرتبط نیستن و هر متغیری رو میشه 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
```

## ۲۹۹. عملگر void چیکار میکنه؟

عملگر `void` عبارت داده شده رو ارزیابی میکنه و بعد تعریف نشده (یعنی بدون بازگشت مقدار) رو برمیگردونه. بریم یه مثال در موردش ببینیم،

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

بیاین پیامی رو بدون هیچ گونه تغییر مسیر یا بارگیری مجدد نمایش بدیم

```
a href="javascript:void(alert('Welcome to JS world'))">Click>
<here to see a message
```

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

## ۳۰۰. چطوری میشه نمایشگر موس صفحه رو به درحال لود تغییر داد؟

مکان نما رو میشه برای انتظار در جاواسکریپت با استفاده از ویژگی `cursor` تنظیم کرد. بیاین این رفتار رو در بارگذاری صفحه با استفاده از تابع زیر انجام بدیم.

```
function myFunction() {
 window.document.body.style.cursor = "wait";
}
```

و این تابع در بارگذاری صفحه فراخوانی می‌شه

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

## ۱. چطوری می‌شه یه حلقه بی‌نهایت درست کرد؟

شما می‌تونین حلقه‌های بی‌نهایت با استفاده از حلقه‌های `for` و `while` بدون استفاده از هیچ عبارتی ایجاد کنین. ساختار یا نحو حلقه `for` از نظر ESLint و ابزارهای بهینه ساز کد، رویکرد بهتریه.

```
for (;;) {}

while(true) {
}
```

## ۲. چرا باید در استفاده از عبارت `with` تجدیدنظر کرد؟

دستور با جاواسکریپت در نظر گرفته شده بود که مختصراً برای نوشتن دسترسی‌های تکرارشونده به اشیا ارائه بده. بنابراین می‌توانه با کاهش نیاز به تکرار یه مرجع طولانی بدون جریمه عملکرد، به کاهش اندازه فایل کمک کنه. بیاین مثالی بزنیم که تو اون برای جلوگیری از افزونگی موقع چندین بار دسترسی به یه شی استفاده می‌شه.

```
a.b.c.greeting = 'welcome';
a.b.c.age = 32;
```

استفاده از "with" کد رو به این شکل تبدیل می‌کنه:

```
with(a.b.c) {
 greeting = "welcome";
 age = 32;
}
```

اما این عبارت `with` مشکلات عملکردی ایجاد می‌کنه، چون نمی‌شه پیش‌بینی کرد که آیا یه آرگومان به یه متغیر واقعی اشاره می‌کنه یا به یه ویژگی درون آرگومان `.with`.

## ۳۰۳. خروجی این حلقه‌ها چی می‌شه؟

```
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 حلقه‌های بالا ۴ ۴ ۴ ۴ و ۰ ۲ ۱ است

**توضیح:** با توجه به صف رویداد/حلقه جاواسکریپت، تابع 'setTimeout' پس از اجرای حلقه فراخوانی می‌شه. از اونجایی که متغیر `i` با کلمه کلیدی "var" اعلام می‌شه، به یه متغیر جهانی تبدیل می‌شه و با استفاده از تکرار زمانی که تابع `setTimeout` است. در حالی که در حلقه می‌شه، مقدار آن برابر با ۴. بنابراین، خروجی حلقه اول '۴ ۴ ۴ ۴' است. در حالی که در حلقه دوم، متغیر `i` به عنوان کلمه کلیدی «`let`» اعلام می‌شه، به متغیری با محدوده بلوک تبدیل می‌شه و یه مقدار جدید (۳, ۰, ۱, ۲) برای هر تکرار داره. بنابراین، خروجی حلقه اول «۰ ۱ ۲ ۳» است.

## ۳۰۴. می‌تونی یه سری از ویژگی‌های ES6 رو اسم ببری؟

در زیر لیستی از بعضی از ویژگی‌های جدید ES6 اومده.

۱. پشتیبانی از ثابت‌ها یا متغیرهای تغییرناپذیر

۲. پشتیبانی Block-scope برای متغیرها، ثابت‌ها و توابع

Arrow functions

۴. پارامترهای پیش فرض

۵. پارامترهای Rest and Spread

Template Literals

Multi-line Strings

Destructuring Assignment

## ۳۰۵. ES6 چیه؟

ES6 ششمین نسخه از زبان جاواسکریپت و در ژوئن 2015 منتشر شد. در ابتدا با نام ECMAScript 6 (ES6) شناخته شد و بعداً به 2015 تغییر نام داد. تقریباً همه مرورگرهای مدرن از ES6 پشتیبانی می‌کنند اما برای مرورگرهای قدیمی ترانسپایلرهای زیادی وجود داره، مانند Babel.js و غیره

## ۳۰۶. آیا می‌توانیم متغیرهای تعریف شده با let و const را مجدداً تعریف کنیم؟

نه، شما نمی‌توانید متغیرهای let و const را مجدداً اعلام کنید. اگه این کار رو انجام بدین، خطای زیر رو نشون میده

```
Uncaught SyntaxError: Identifier 'someVariable' has already been declared
```

**توضیح:** اعلان متغیر با کلمه کلیدی "var" به یه محدوده تابع اشاره داره و با متغیر به دلیل ویژگی بالا بردن به گونه ای رفتار می‌شه که گویی در بالای محدوده محصور اعلام شده است. بنابراین همه اعلان‌های چندگانه بدون هیچ خطایی در ایجاد یه متغیر hoisted مشترک نقش دارن. بیاین مثالی از اعلان مجدد متغیرها تو یه 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 بلوک، خطای نحوی ایجاد می‌کنه،

```
let name = 'John';
function myFunc() {
 let name = 'Nick';
 let name = 'Abraham'; // Uncaught SyntaxError: Identifier
'name' has already been declared
 alert(name);
}

myFunc();
alert(name);
```

### ۳۰۷. آیا استفاده از `const` برای تعریف متغیر او نا رو `immutable` می‌کنه؟

خیر، متغیر `const` مقدار را تغییرناپذیر نمی‌کنه. اما تخصیص‌های بعدی رو مجاز نمی‌داند (یعنی می‌توانین با تخصیص اعلام کنین اما بعداً نمی‌توانین مقدار دیگری رو اختصاص بدین)

```
const userList = [];
userList.push('John'); // Can mutate even though it can't re-
assign
console.log(userList); // ['John']
```

### ۳۰۸. پیش‌فرض چی هستن؟ parameter

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 میده تا پارامترها با مقادیر پیش‌فرض مقداردهی اولیه بشن، اگه مقداری یا تعریف‌نشده ارسال نشه. بیان رفتار رو با یه مثال مقایسه کنیم،

```
//ES5
var calculateArea = function(height, width) {
 height = height || 50;
 width = width || 60;

 return width * height;
}
console.log(calculateArea()); //300
```

پارامترهای پیش فرض، مقدارهای اولیه را ساده تر می‌کنه،

```
//ES6
var calculateArea = function(height = 50, width = 60) {
 return width * height;
}

console.log(calculateArea()); //300
```

## ۳۰۹. **چی هستن؟ template-literal**

حروف الفبای الگو یا رشته‌های الگو، حروف الفبای رشته‌ای هستن که امکان عبارات تعبیه شده رو میدن. اینا به جای گیومه‌های دوتایی یا تک با کاراکتر بک تیک (') محصور میشون

در ES6، این ویژگی استفاده از عبارات پویا رو به شرح زیر امکان پذیر می‌کنه.

```
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: می‌تونین از رشته‌های چند خطی و ویژگی‌های درون‌یابی رشته‌ای با الفبای الگو استفاده کنین.

## ۳۱۰. **چطوری رشته‌های چند خطی رو توی template-literal می‌نویسیم؟**

در ES5، برای بدست آوردن رشته‌های چند خطی، باید از کاراکترهای فرار از خط جدید (\n) و نمادهای الحاقی (+) استفاده کنین.

```
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`);
```

## ۳۱۱. تودرتو چی هستن؟ template-literal

الگوی تودرتو یه ویژگیه که در نحو تحت اللفظی الگو پشتیبانی میشه تا امکان بکتیکهای درونی تو یه مکان نمای \${ } رو در قالب فراهم کنه. برای مثال، الگوی تودرتو زیر برای نمایش نمادها بر اساس مجوزهای کاربر استفاده میشه، در حالی که الگوی بیرونی نوع پلت فرم رو بررسی میکنه.

```
const iconStyles = `icon ${ isMobilePlatform() ? '' :
`icon-${user.isAuthorized ? 'submit' : 'disabled'}}`;
```

میتونین مورد استفاده بالا رو بدون ویژگیهای الگوی تودرتو هم بنویسین. با این حال، ویژگی الگوی تودرتو فشردهتر و خواناتر است.

```
//Without nesting templates
const iconStyles = `icon ${ isMobilePlatform() ? '' :
(user.isAuthorized ? 'icon-submit' : 'icon-disabled')}`;
```

## ۳۱۲. چی هستن؟ tagged-template

الگوهای برچسب‌گذاری شده شکل پیشرفته‌ای از قالب‌ها هستن که تو اون برچسب‌ها به شما اجازه میدن تا کلمات قالب رو با یه تابع تجزیه کنین. تابع تگ اولین پارامتر رو به عنوان آرایه‌ای از رشته‌ها و پارامترهای باقی مانده رو به عنوان عبارات می‌پذیرد. این تابع همچنین میتوانه رشته‌های دستکاری شده رو بر اساس پارامترها برگرداند. بیاین نحوه استفاده از رفتار الگوی برچسب‌گذاری شده مجموعه مهارت‌های حرفه‌ای فناوری اطلاعات تو یه سازمان رو 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

```

## ۳۱۳. رشته‌های خام چی هستن؟

ES6 با استفاده از روش «String.raw» ویژگی رشته‌های خام را ارائه می‌کنه که برای دریافت شکل رشته خام رشته‌های الگو استفاده می‌شه. این ویژگی به شما این امکان را میدهد تا به رشته‌های خام همونطور که وارد شده‌اند، بدون پردازش دنباله‌های فرار دسترسی داشته باشیم. برایم یه مثال در موردش ببینیم،

```
var calculationString = String.raw `The sum of numbers is
\n${1+2+3+4}!`;
console.log(calculationString); // The sum of numbers is 10
```

اگه از رشته‌های خام استفاده نمی‌کنین دنباله کاراکترهای خط جدید با نمایش خروجی در چندین خط پرداش می‌شه.

```
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]);
}
```

## ۳۱۴. **assign** کردن با **destructuring** چیه و چطوری انجام می‌شه؟

تخصیص **destructuring** یه عبارت جاواسکریپت‌ه که امکان باز کردن مقادیر آرایه‌ها یا خصوصیات از اشیاء به متغیرهای مجزا رو فراهم می‌کنه.

بیاین مقادیر ماه رو از یه 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 می‌تونین ویژگی‌های کاربر یه آبجکت رو با استفاده از انتساب به **destructuring** دست بیارین،

```
var {name, age} = {name: 'John', age: 32};

console.log(name); // John
console.log(age); // 32
```

## ۳۱۵. موقع assign کردن با destructuring چطوری می‌شه مقدار اولیه تعریف کرد؟

زمانی که مقدار باز شده از آرایه یا شیء در طول تخصیص ساختارشکن تعریف نشده باشد، می‌شه به يه متغیر يه مقدار پیش فرض اختصاص داد. این کمک می‌کنه تا از تنظیم مقادیر پیش فرض جداگانه برای هر انتساب جلوگیری کنیں. بیاین 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
```

## ۳۱۶. چطوری می‌تونیم مقدار یه آرایه رو با استفاده از destructuring assignment تعویض کنیم؟

اگه از destructuring-assignment استفاده نمی‌کنیں تعویض دو مقدار به يه متغیر وقت نیاز داره. در حالی که با استفاده از يه ویژگی ساختارشکن، دو مقدار متغیر رو می‌شه تو يه عبارت ساختار شکن جایگزین کرد. بیاین دو متغیر عددی رو 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
```

## ۳۱۷ Enhanced-object-literal چی هستن؟

ایجاد سریع اجسام با ویژگی‌های درون برآکت رو آسان می‌کنه. برای مثال، نحو کوتاهتری رو برای تعریف ویژگی شی مشترک به شرح زیر ارائه می‌کنه.

```
//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}
```

## ۳۱۸ import چی هستن؟

واردات پویا با استفاده از نحو تابع «import» به ما اجازه میده تا ماثول‌ها رو در صورت تقاضا با استفاده از دستورات یا دستور async/await بارگذاری کنیم. در حال حاضر این ویژگی در پیشنهاد مرحله 4 [https://github.com/tc39/proposal-dynamic-import] (https://github.com/tc39/proposal-dynamic-import) هستش. مزیت اصلی واردات پویا کاهش اندازه بسته‌های ما، پاسخ حجم/بار بار درخواست‌های ما و بهبود کلی در تجربه کاربر است.

```
import('./Module').then(Module => Module.method());
```

## ۳۱۹ کاربرد import چیه؟

در زیر بعضی از موارد استفاده از واردات پویا نسبت به واردات استاتیک آورده شده است.  
۱. یه ماثول رو به صورت درخواستی یا مشروط وارد کنین. برای مثال، اگه می‌خواین یه polyfill رو در مرورگر قدیمی بارگذاری کنین

```
if (isLegacyBrowser()) {
 import('...').then(...);
}
```

۲. تعیین کننده ماثول رو در زمان اجرا محاسبه کنین. برای مثال می‌تونین از اون برای گلوبال سازی استفاده کنین.

```
import(`messages_${getLocale()}.js`).then(...);
```

۳. یه مازول رو از داخل یه اسکریپت معمولی به جای یه مازول وارد کنین.

## ۳۲۰. آرایه‌های نوع‌دار (typed-arrays) چیه؟

آرایه‌های تایپ شده آبجکت‌هایی آرایه مانند از ECMAScript 6 API برای مدیریت داده‌های باینری هستن. جاواسکریپت ۸ نوع آرایه تایپ شده رو ارائه میده،

- ۱. آرایه‌ای از اعداد صحیح امضا شده ۸ بیتی :Int8Array
- ۲. آرایه‌ای از اعداد صحیح امضا شده ۱۶ بیتی :Int16Array
- ۳. آرایه‌ای از اعداد صحیح امضا شده ۳۲ بیتی :Int32Array
- ۴. آرایه‌ای از اعداد صحیح بدون علامت ۸ بیتی :Uint8Array
- ۵. آرایه‌ای از اعداد صحیح بدون علامت ۱۶ بیتی :Uint16Array
- ۶. آرایه‌ای از اعداد صحیح بدون علامت ۳۲ بیتی :Uint32Array
- ۷. آرایه‌ای از اعداد ممیز شناور ۳۲ بیتی :Float32Array
- ۸. آرایه‌ای از اعداد ممیز شناور ۶۴ بیتی :Float64Array

برای مثال، شما می‌تونین یه آرایه از اعداد صحیح امضا شده ۸ بیتی مانند زیر ایجاد کنین

```
const a = new Int8Array();
// You can pre-allocate n bytes
const bytes = 1024
const a = new Int8Array(bytes)
```

## ۳۲۱. مزایای لودر مازول‌ها چیه؟

مازول لودر ویژگی‌های زیر رو ارائه میده

- ۱. Dynamic loading
- ۲. State isolation
- ۳. Global namespace isolation
- ۴. Compilation hooks
- ۵. Nested virtualization

## ۳۲۴. چیه؟ collation

برای مرتب سازی مجموعه ای از رشته ها و جستجو در مجموعه ای از رشته ها استفاده می شه. این پارامتر توسط محلی و از Unicode آگاه است. بیان ویژگی های مقایسه و مرتب سازی رو در نظر بگیریم،  
**:Comparison .۱**

```
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
```

## ۳۲۵. Sorting .۲

```
var list = ["ä", "a", "z"]; // In German, "ä" sorts with "a"
Whereas in Swedish, "ä" sorts after "z"
var l10nDE = new Intl.Collator("de");
var l10nSV = new Intl.Collator("sv");
console.log(list.sort(l10nDE.compare)) // ["a", "ä", "z"]
console.log(list.sort(l10nSV.compare)) // ["a", "z", "ä"]
```

## ۳۲۶. عبارت for...of چیه؟

دستور for...of یه حلقه تکرار بر روی اشیاء یا عناصر قابل تکرار مثل رشته داخلی، آرایه، اشیاء آرایه مانند (مثل آرگومان ها یا NodeList، TypedArray، Map، Set و تکرارهای Node)، تعريف شده توسط کاربر ایجاد می کنه.

```
let arrayIterable = [10, 20, 30, 40, 50];

for (let value of arrayIterable) {
 value ++;
 console.log(value); // 11 21 31 41 51
}
```

## ۳۲۷. خروجی عملگر spread روی آرایه زیر چیه؟

[... 'John Resig']

خروجی آرایه [ 'g', 'J', 's', 'e', 'R', 'n', 'h', 'o', 'i' ] است.

**توضیح:** رشته یه نوع تکرارپذیره و عملگر spread تو یه آرایه هر کاراکتر یه تکرارپذیر رو به یه عنصر نگاشت میکنه. از این رو، هر کاراکتر یه رشته به عنصری تو یه آرایه تبدیل میشه.

## ۳۲۵. آیا PostMessage امنه؟

بله، تا زمانی که برنامهنویس/توسعهدهنده مراقب منشأ و منبع پیام دریافتی باشد، postMessages رو میشه بسیار امن در نظر گرفت. اما اگه بخواین پیامی رو بدون تأیید منبع آن ارسال یا دریافت کنیں حملات اسکریپت بین سایتی ایجاد میشه.

## ۳۲۶. مشکلات استفاده از wildcard روی origin با postmessage چیه؟

آرگومان دوم متده postMessage مشخص میکنه که کدوم مبدأ مجاز به دریافت پیامه. اگه از علامت "\*" به عنوان آرگومان استفاده کنیں هر منبعی مجاز به دریافت پیامه. در این حالت، هیچ راهی برای پنجره فرستنده وجود نداره که بفهمه پنجره هدف در موقع ارسال پیام در مبدأ هدف قرار داره یا نه. اگه پنجره هدف به مبدأ دیگری هدایت شه، مبدأ دیگر دادهها رو دریافت میکنه. از این رو، این ممکنه منجر به آسیب پذیریهای XSS شه.

```
targetWindow.postMessage(message, '*');
```

## ۳۲۷. چطوری از دریافت postMessage های ناخواسته و نامن از طرف هکرهای جلوگیری کنیم؟

از اونجایی که شنونده به هر پیامی گوش میده، مهاجم میتونه برنامه رو با ارسال پیامی از مبدأ مهاجم فریب بده که این تصور رو ایجاد میکنه که گیرنده پیام رو از پنجره فرستنده واقعی دریافت کرده. شما میتونین با اعتبارسنجی مبدأ پیام در انتهای گیرنده با استفاده از ویژگی "message.origin" از این مشکل جلوگیری کنیم. برای مثال، اجازه بدین مبدأ فرستنده <http://www.some-sender.com> در سمت گیرنده [ receiver.com- (receiver.com) (www.some.com) ] رو 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);
 }
});
```

## ۳۲۸. می‌تونیم کلا postMessage را غیرفعال کنیم؟

شما نمی‌تونین به طور کامل (یا ۱۰۰٪) از postMessages استفاده نکنین. حتی اگه برنامه شما با توجه به خطرات از postMessage استفاده نمی‌کنه، بسیاری از اسکریپت‌های شخص ثالث از postMessage برای برقراری ارتباط با سرویس شخص ثالث استفاده می‌کنن. بنابراین ممکنه برنامه شما بدون اطلاع شما از postMessage استفاده کنه.

## ۳۲۹. آیا postMessage را به صورت synchronous و همزمان کار می‌کنن؟

در مرورگر IE8 همگام هستن اما در IE9 و سایر مرورگرهای مدرن دیگر (یعنی IE9+, Firefox, Chrome, Safari) ناهمزمان هستن. به دلیل این رفتار ناهمزمان، زمانی که postMessage برگردونده می‌شه، از مکانیزم callback استفاده می‌کنیم.

## ۳۳۰. پارادیم زبان جاوااسکریپت چیه؟

جاوااسکریپت یه زبان چند پارادایمه که از برنامه نویسی امری/روشی، برنامه نویسی شی گرا و برنامه نویسی تابعی پشتیبانی می‌کنه. جاوااسکریپت از برنامه نویسی شی گرا با وراثت اولیه پشتیبانی می‌کنه.

## ۳۳۱. تفاوت‌های بین جاوااسکریپت داخلی و خارجی چیه؟

جاوااسکریپت داخلی: کد منبع درون تگ اسکریپته.

جاوااسکریپت خارجی: کد منبع تو یه فایل خارجی (ذخیره شده با پسوند js). ذخیره میشه و در تگ ارجاع میشه.

## ۳۳۲. آیا جاوااسکریپت سریعتر از اسکریپتهاي سمت سرور است؟

بله، جاوااسکریپت سریعتر از اسکریپت سمت سروره. از اونجایی که جاوااسکریپت یه اسکریپت سمت کلایینته برای محاسبات یا محاسبات خودش به کمک سرور وب نیازی نداره. بنابراین جاوااسکریپت همیشه سریعتر از هر اسکریپت سمت سرور مانند ASP، PHP و غیره اس.

## ۳۳۳. چطوری وضعیت چک بودن یه checkbox رو بدست بیاریم؟

میتونین ویژگی checked رو در کادر انتخاب شده در DOM اعمال کنین. اگه مقدار "True" باشه به این معنیه که چک باکس علامت زده شده در غیر این صورت علامت اونو بردارین. برای مثال، عنصر چک باکس HTML زیر رو میشه با استفاده از جاوااسکریپت 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
```

## ۳۳۴. هدف از عملگر double-tilde چیه؟

عملگر دابل (~) به عنوان عملگر bitwise NOT (tilde) شناخته میشه. این عملگر قراره جایگزین سریع تری برای . Math.floor

## ۳۳۵. چطوری یه کاراکتر رو به کد ASCII تبدیل کنیم؟

برای تبدیل کاراکترهای رشته به اعداد اسکی می‌توانیم از متدهای `String.prototype.charCodeAt` استفاده کنیم. برای مثال، بیاین کد ASCII رو برای حرف اول رشته «ABC» پیدا کنیم،

```
"ABC".charCodeAt(0) // returns 65
```

در حالی که روش `String.fromCharCode` اعداد رو به کاراکترهای ASCII برابر تبدیل می‌کند.

```
String.fromCharCode(65,66,67); // returns 'ABC'
```

## ۳۳۶. **ArrayBuffer چیه؟**

یه شی `ArrayBuffer` برای نشون دادن یه بافر داده باینری خام عمومی با طول ثابت استفاده می‌شه. می‌توانیم اونو به صورت زیر ایجاد کنیم

```
let buffer = new ArrayBuffer(16); // create a buffer of length 16
alert(buffer.byteLength); // 16
```

برای دستکاری یه `ArrayBuffer`، باید از یه شی "view" استفاده کنیم.

```
//Create a DataView referring to the buffer
let view = new DataView(buffer);
```

## ۳۳۷. **خروجی کد زیر چی خواهد بود؟**

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

خروجی عبارت بالا "W" عه.

**توضیح:** نماد برآکت با شاخص خاص روی یه رشته کاراکتر رو تو یه مکان خاص برمی‌گردونه. از این رو، کاراکتر "W" رشته رو برمی‌گردونه. از اونجایی که این مورد در نسخه‌های IE7 و پایین‌تر پشتیبانی نمی‌شه، ممکنه لازم باشه از متدهای `charAt` برای به دست آوردن نتیجه دلخواه استفاده کنیم.

## ۳۳۸. هدف از Error-object چیه؟

سازنده Error یه شی خطا ایجاد می‌کنه و نمونه‌هایی از آبجکت‌های خطا موقع رخ دادن خطاهای زمان اجرا پرتاب می‌شون شی Error همچنین می‌توانه به عنوان یه شی پایه برای استثناهای تعریف شده توسط کاربر استفاده شه. برای مثال

```
new Error([message[, fileName[, lineNumber]]])
```

شما می‌تونین استثناهای خطاها یا خطاها تعریف شده توسط کاربر رو با استفاده از آبجکت Error در بلوک try...catch مانند زیر پرتاب کنین.

```
try {
 if(withdraw > balance)
 throw new Error("Oops! You don't have enough balance");
} catch (e) {
 console.log(e.name + ': ' + e.message);
}
```

## ۳۳۹. هدف از EvalError-object چیه؟

شی eval یه خطا در رابطه با تابع eval جهانی رو نشون میده. حتی اگه این استثنای دیگر توسط جاواسکریپت پرتاب نمی‌شه، شی EvalError برای سازگاری باقی می‌ماند. برای مثال

```
new EvalError([message[, fileName[, lineNumber]]])
```

می‌تونین EvalError رو با بلوک try...catch مانند زیر پرتاب کنین.

```
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"
}
```

## ۳۴۰. خطاهایی که در حالت strict-mode رخ میدن ولی در غیر اون وجود ندارن کدوما هستن؟

وقتی `use strict` رو اعمال میکنین، syntax، بعضی از موارد زیر قبل از اجرای اسکریپت یه `SyntaxError` ایجاد میکنن  
۱. وقتی از دستور `Octal` استفاده میکنین

```
var n = 022;
```

۲. استفاده از عبارت `.with`.

۳. وقتی از عملگر حذف روی نام متغیر استفاده میکنین

۴. استفاده از `eval` یا آرگومانها به عنوان متغیر یا نام آرگومان تابع

۵. موقعی که از کلمات کلیدی رزرو شده جدید استفاده میکنین

۶. موقعی که یه تابع رو تو یه بلوک اعلام میکنین

```
if (someCondition) { function f() {} }
```

از این رو، خطاهای موارد بالا برای جلوگیری از خطا در محیط‌های توسعه/تولید مفید هستن.

## ۳۴۱. آیا همه object‌ها دارای prototype هستن؟

خیر. همه object‌ها دارای نمونه اولیه هستن به جز `Object` پایه که توسط کاربر ایجاد میشه یا آبجکت‌ای که با استفاده از کلمه کلیدی `new` ایجاد میشه.

## ۳۴۲. تفاوت‌های بین argument و parameter چیه؟

پارامتر نام متغیر تعریف یه تابعه در حالی که یه آرگومان نشون دهنده مقدار داده شده به تابع موقع کال شدنشه. بیاین این رو با یه تابع ساده توضیح بدیم

```

function myFunction(parameter1, parameter2, parameter3) {
 console.log(arguments[0]) // "argument1"
 console.log(arguments[1]) // "argument2"
 console.log(arguments[2]) // "argument3"
}
myFunction("argument1", "argument2", "argument3")

```

## ۳۴۳. هدف از متد some روی آرایه‌ها چیه؟

متد `some` برای آزمایش اینکه آیا حداقل یه عنصر در آرایه از آزمون پیاده‌سازی شده توسط تابع ارائه شده عبور می‌کنه یا نه استفاده می‌شه. متد یه مقدار بولین برمی‌گردونه. بیاین مثالی بزنیم تا هر عنصر عجیب و غریب رو 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)

```

## ۳۴۴. چطوری دو یا تعداد بیشتری از آرایه‌ها رو با هم ترکیب کنیم؟

متد `concat` برای پیوستن دو یا چند آرایه با برگرداندن یه آرایه جدید حاوی تمام عناصر استفاده می‌شه. مثال:

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

بیاین مثالی از الحق آرایه با آرایه‌های سبزیجات و میوه‌ها رو 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

```

## ۳۴۵. تفاوت‌های بین Deep و Shallow کپی چیه؟

## کپی کم عمق:

کپی کم عمق یه کپی بیتی از یه شی است. یه شی جدید ایجاد میشه که یه کپی دقیق از مقادیر موجود در شی اصلی داره. اگه هر یه از فیلدهای شی ارجاع به آبجکتهاي دیگه باشه، فقط آدرسهاي مرجع کپی می شن یعنی فقط آدرس حافظه کپی میشه.

### مثال

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

برای ایجاد یه نسخه تکراری

```
var empDetailsShallowCopy = empDetails //Shallow copying!
```

اگه مقداری از ویژگی رو تو یه تکراری به این صورت تغییر بدیم:

```
empDetailsShallowCopy.name = "Johnson"
```

دستور بالا همچنین نام `empDetails` را تغییر میده، چون ما یه کپی کم عمق داریم. یعنی ما دادههای اصلی رو هم از دست می دیم.

### کپی عمیق(Deep)

یه کپی عمیق همه فیلدها رو کپی میکنه و از حافظه تخصیص یافته به صورت پویا که توسط فیلدها به آن اشاره میشه کپی میکنه. یه کپی عمیق زمانی اتفاق میوافته که یه شی همراه با آبجکتهايی که به اون اشاره داره کپی شه.

### Example

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

یه کپی عمیق با استفاده از خواص از ابجکت اصلی در متغیر جدید ایجاد کنین

```
var empDetailsDeepCopy = {
 name: empDetails.name,
 age: empDetails.age,
 expertise: empDetails.expertise
}
```

اگه `empDetailsDeepCopy` رو تغییر بدین، فقط `empDetailsDeepCopy.name` و نه `empDetailsDeepCopy` رو تغییر بدین، فقط `empDetailsDeepCopy.name` و نه `empDetails` رو تحت تأثیر قرار خواهد داد.

## ۳۴۶. چطوری می‌تونیم به یه تعداد مشخص از یه رشته کپی کنیم؟

متدهای `repeat` برای ساخت و برگرداندن یه رشته جدید استفاده می‌شه که حاوی تعداد مشخصی از کپی‌های رشته‌ای هس که روی آن فراخوانی شده و به هم پیوسته شدن. به یاد داشته باشیم که این روش به مشخصات ECMAScript 2015 اضافه شده است. بیاین یه مثال از یه `Hello` روش تکرار آن ۴ بار در نظر بگیریم.

```
'Hello'.repeat(4); // 'HelloHelloHelloHello'
```

## ۳۴۷. چطوری همه string های regular-expression شده با یه match رو برگردانیم؟

متدهای `matchAll` می‌تواند برای برگرداندن یه تکرارکننده از تمام نتایجی که یه رشته با یه عبارت منظم مطابقت دارن، استفاده شه. برای مثال، مثال زیر آرایه ای از نتایج رشته منطبق رو در برابر یه عبارت منظم برمی‌گردونه.

```
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
```

## ۳۴۸. چطوری یه رشته رو از اول یا از آخر trim کنیم؟

روش `trim` نمونه اولیه رشته برای برش دادن دو طرف یه رشته استفاده می‌شه. اما اگه می‌خواهید به خصوص در ابتدای انتها یا انتهای رشته رو برش بدین، می‌توانین از روش‌های `trimEnd/trimRight` و `trimStart/trimLeft` استفاده کنین. بیاین نمونه ای از این روش‌ها رو 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!"
```

## ۳۴۹. خروجی کنسول زیر با عملگر unary چی می‌شه؟

بیاین دستور کنسول رو با عملگر unary همونطور که در زیر نشون داده شده است، بگیریم.

```
console.log(+ 'Hello');
```

خروجی دستور log کنسول بالا NaN را برمی‌گردونه. از اونجا که عنصر توسط عملگر unary پیشونده و مفسر جاوااسکریپت سعی می‌کنه آن عنصر رو به یه نوع عدد تبدیل کنه. از اونجایی که تبدیل با شکست مواجه می‌شه، مقدار عبارت به مقدار NaN منجر می‌شه.

## ۳۵۰. آیا جاوااسکریپت از mixin‌ها استفاده می‌کنه؟

## ۳۵۱. تابع thunk چیه و چیکار می‌کنه؟

فقط تابعیه که ارزیابی مقدار رو به تاخیر میندازه. هیچ آرگومانی نمی‌گیره، اما هر زمان که thunk رو فراخوانی می‌کنین مقدار رو میده. برای مثال، از اون استفاده می‌شه که الان اجرا نشه، اما زمانی در آینده اجرا می‌شه. بیاین یه مثال همزمان بگیریم،

```

const add = (x,y) => x + y;

const thunk = () => add(2,3);

thunk() // 5
```

## ۳۵۲. چیکار می‌کن؟ asyntchunk های

های ناهمزمان برای ایجاد درخواست‌های شبکه مفید هستن. بیاین نمونه ای از درخواست‌های شبکه رو 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 نمی‌شه و تنها زمانی فراخوانی می‌شه که داده‌ها از نقطه پایانی API در دسترس باشن. تابع setTimeout هم برای ناهمزمان کردن کد ما استفاده می‌شه. بهترین مثال زمان واقعی، کتابخانه مدیریت حالت redux هس که از thunk ناهمزمان برای به تأخیر انداختن اعمال برای ارسال استفاده می‌کنه.

## ۳۵۳. خروجی فراخوانی‌های توابع زیر چی می‌شه؟

: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 باشیم که قطر یه تابع منظم است، در حالی که مقدار محیط یه تابع فلش است. کلمه کلیدی «this» یه تابع معمولی (یعنی قطر) به محدوده اطراف که یه کلاسه (یعنی شی شکل) اشاره دارد. در حالی که این کلمه کلیدی تابع محیطی به محدوده اطراف که یه آبجکت window هس اشاره دارد. از اونجایی که هیچ ویژگی شعاع در آبجکتهای window وجود نداره، یه مقدار تعریف نشده برمی‌گردونه و ضرب مقدار مقدار NaN رو برمی‌گردونه.

## ۳۵۴. چطوری همه خطوط جدید رو از به رشته حذف کرد؟

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 یه رشته خالی است.

```
function remove_linebreaks(var message) {
 return message.replace(/\r\n]+/gm, "");
}
```

در عبارت بالا g و m برای پرچم‌های سراسری و چند خطی هستن.

## ۳۵۵. تفاوت بین repaint و reflow چیه؟

رنگ‌آمیزی مجدد زمانی اتفاق می‌افتد که تغییراتی ایجاد می‌شه که روی دید یه عنصر تأثیر می‌گذارد، اما روی طرح آن تأثیر نمی‌گذارد. نمونه‌هایی از این موارد شامل طرح کلی، نمایان بودن یا رنگ پس زمینه است. یه reflow شامل تغییراتیه که بر طرح بندی بخشی از صفحه (یا کل صفحه) تأثیر می‌zarه. تغییر اندازه پنجره مرورگر، تغییر فونت، تغییر محتوا (مانند تایپ متن توسط کاربر)، استفاده از روش‌های جاوااسکریپت شامل سبک‌های محاسبه شده، افزودن یا حذف عناصر از DOM، و تغییر کلاس‌های یه عنصر چند مورد از مواردی هستن که می‌تونن جریان مجدد رو آغاز کنن. جریان مجدد یه عنصر باعث جریان مجدد بعدی همه عناصر فرزند و اجداد و همچنین هر عنصری که به دنبال آن در DOM است می‌شه.

## ۳۵۶. اگه قبل از یه آرایه عملگر نفی «!» بزاریم چی می‌شه؟

نفی یه آرایه با کاراکتر «!»، آرایه رو به یه بولین وادار می‌کنه. از اونجایی که آرایه‌ها صدق در نظر گرفته می‌شن، پس نفی اون `false` رو برمی‌گردونه.

```
console.log(![]); // false
```

## ۳۵۷. اگه دو تا آرایه رو با هم جمع ببندیم چی می‌شه؟

اگه دو آرایه رو با هم اضافه کنین هر دو اونا رو به رشته تبدیل می‌کنه و اونا رو به هم متصل می‌کنه. برای بریم یه مثال در موردش ببینیم.

```
console.log(['a'] + ['b']); // "ab"
console.log([] + []); // ""
console.log(![] + []); // "false", because ![] returns false.
```

## ۳۵۸. اگه عملگر جمع «+» روی قرار بدیم چی می‌شه؟

اگه عملگر (+) رو روی مقادیر نادرست (null، undefined، NaN، false) قرار بدین، مقدار falsy به مقدار عددی صفر تبدیل می‌شه. بیاین اونا رو در کنسول مرورگر به صورت زیر نمایش بدیم،

```
console.log(+null); // 0
console.log(+undefined); // NaN
console.log(+false); // 0
console.log(+NaN); // NaN
console.log(+"""); // 0
```

## ۳۵۹. چطوری با استفاده از آرایه‌ها و عملگرهای منطقی می‌تونیم رشته self رو تولید کنیم؟

رشته `self` رو می‌شه با ترکیب کاراکترهای `[]()` تشکیل داد. برای رسیدن به این الگو باید قراردادهای زیر رو به خاطر بسپارید.

۱. از اونجایی که آرایه‌ها مقادیر `true` هستن، با نفی آرایه‌ها `false` تولید می‌شه: !

```
false === []
```

۲. طبق قوانین اجباری جاواسکریپت، اضافه کردن آرایه‌ها به هم اونا رو به

رشته‌بندی تبدیل می‌کنه: `"" ==="" + []`

۳. Prepend یه آرایه با عملگر `+` به آرایه رو به نادرست تبدیل می‌کنه، انکار اونو

درست می‌کنه و در نهایت تبدیل نتیجه مقدار '`'` رو تولید می‌کنه: `(([]+)!) +`

`1 ===`

با اعمال قوانین بالا می‌توانیم شرایط زیر رو استخراج کنیم

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

اکنون الگوی کارکتر به صورت زیر ایجاد می‌شه.

<code>s</code>	<code>e</code>	<code>l</code>	<code>f</code>
<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [3] + ( ![] + [] ) [4] + ( ![] + [] ) [2] + ( ![] + [] ) [0]</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [+!+[ ]+!+[ ]+!+[ ] ] +</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [+!+[ ]+!+[ ]+!+[ ]+!+[ ] ] +</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [+!+[ ]+!+[ ] ] +</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [+!+[ ] ] +</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>
<code>( ![] + [] ) [+!+[ ]+!+[ ]+!+[ ] ] + ( ![] + [] ) [+!+[ ]+!+[ ]+!+[ ]+!+[ ] ] + ( ![] + [] ) [+!+[ ]+!+[ ] ] + ( ![] + [] ) [+!+[ ] ] + ( ![] + [] ) [ +[] ]</code>	<code>~~~~~</code>	<code>~~~~~</code>	<code>~~~~~</code>

## ۳۶۰. چطوری می‌توانیم مقادیر falsy رو از آرایه حذف کنیم؟

شما می‌توانین با وارد کردن Boolean به عنوان پارامتر، روش فیلتر رو روی آرایه اعمال کنین.

به این ترتیب تمام مقادیر نادرست (0، تعریف نشده، null، false و "") از آرایه حذف می‌شه.

```
const myArray = [false, null, 1, 5, undefined]
myArray.filter(Boolean); // [1, 5] // is same as
myArray.filter(x => x);
```

## ۳۶۱. چطوری مقادیر تکراری رو از یه آرایه حذف کنیم؟

شما می‌توانید مقادیر منحصر به فرد یه آرایه رو با ترکیب دستور "Set" و rest (...) دریافت کنین.

```
console.log([...new Set([1, 2, 4, 4, 3])]); // [1, 2, 4, 3]
```

## ۳۶۲. همراهی همزنان با destructuring چطوری کار می‌کنند؟

گاهی اوقات شما دوست دارید یه متغیر تخریب شده با نامی متفاوت از نام ویژگی داشته باشیم. در این صورت، از یه «newName» برای تعیین نام برای متغیر استفاده خواهید کرد. این فرآیند نام مستعار تخریب ساختاری نامیده می‌شه.

```
const obj = { x: 1 };
// Grabs obj.x as as { otherName }
const { x: otherName } = obj;
```

## ۳۶۳. چطوری آیتم‌های یه آرایه رو بدون استفاده از متدهای map پیمایش کنیم؟

می‌توانید مقادیر آرایه‌ها رو بدون استفاده از روش «نقشه» تنها با استفاده از روش «از» آرایه ترسیم کنین. بیان نام شهرها رو 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']
```

## ۳۶۴. چطوری یه آرایه رو خالی کنیم؟

با صفر کردن طول آرایه می‌توانیم به سرعت یه آرایه رو خالی کنیم.

```
let cities = ['Singapore', 'Delhi', 'London'];
cities.length = 0; // cities becomes []
```

## ۳۶۵. چطوری اعداد رو با تعداد رقم اعشار مشخص رند می‌کنی؟

می‌توانیم با استفاده از روش «toFixed» از جاواسکریپت، اعداد رو به تعداد معینی از اعشار گرد کنیم.

```
let pie = 3.141592653;
pie = pie.toFixed(3); // 3.142
```

## ۳۶۶. ساده‌ترین روش برای تبدیل آرایه به object چیه؟

شما می‌توانید با استفاده از عملگر (...) یه آرایه رو به یه شی با همون داده تبدیل کنید.

```
var fruits = ["banana", "apple", "orange", "watermelon"];
var fruitsObject = {...fruits};
console.log(fruitsObject); // {0: "banana", 1: "apple", 2: "orange", 3: "watermelon"}
```

## ۳۶۷. چطوری یه آرایه با یه سری داده درست کنیم؟

می‌توانید با استفاده از روش «fill» یه آرایه با مقداری داده یا یه آرایه با همون مقادیر ایجاد کنید.

```
var newArray = new Array(5).fill("0");
console.log(newArray); // ["0", "0", "0", "0", "0"]
```

## ۳۶۸. متغیرهای موجود روی آبجکت console کدوما هستن؟

۱. ۰% - یه شی رو می‌گیرد،

۲. ۵% - یه رشته می‌گیره،

۳. d% - برای اعشار یا عدد صحیح استفاده می‌شود

این متغیرها رو می‌شود در console.log به صورت زیر نشون داد

```
const user = { "name": "John", "id": 1, "city": "Delhi"};
console.log("Hello %s, your details %o are available in the
object form", "John", user); // Hello John, your details {name:
"John", id: 1, city: "Delhi"} are available in object
```

### ۳۶۹. می‌شه پیام‌های کنسول رو استایل دهی کرد؟

بله، می‌تونین سبک‌های CSS رو برای پیام‌های کنسول مشابه متن html در صفحه وب اعمال کنین.

```
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 می‌شه

> 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 سبک‌های CSS رو می‌شه برای پیام‌های کنسول اعمال کرد.

### ۳۷۰. هدف از متد dir روی آبجکت console چیه؟

برای نمایش یه لیست تعاملی از ویژگی‌های شی جاواسکریپت مشخص شده به عنوان JSON استفاده می‌شه.

```
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 [Object]
 name: "John"
 id: 1
 city: "Delhi"
 ► __proto__: Object
```

## ۳۷۱. آیا می‌شه المنتهای HTML رو توی console دیباگ کرد؟

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 رو در کنسول چاپ می‌کنه،

```
> const element = document.getElementsByTagName("body")[0];
< undefined
> console.log(element);
► <body class="question-page unified-theme">...</body>
< undefined
> |
```

## ۳۷۲. چطوری می‌شه داده‌ها رو به شکل جدولی توی console نمایش بدیم؟

برای نمایش داده‌ها در کنسول در قالب جدولی برای تجسم آرایه‌ها یا اشیاء پیچیده استفاده می‌شه.

```
const users = [{ "name": "John", "id": 1, "city": "Delhi"}, {
 "name": "Max", "id": 2, "city": "London"}, { "name": "Rod", "id": 3, "city": "Paris"}];
console.table(users);
```

داده‌هایی که در قالب جدول مشاهده می‌شن،

```

> 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)

نکته: یادتون باشه `console.table` توی مرورگر IE پشتیبانی نمیشه 😞

## ۳۷۳. چطوری میشه بررسی کرد که یه پارامتر Number هست یا نه؟

ترکیبی از روش‌های `isFinite` و `isNaN` برای تأیید عدد بودن یا نبودن آرگومان استفاده می‌شود.

```

function isNumber(n){

 return !isNaN(parseFloat(n)) && isFinite(n);

}

```

## ۳۷۴. چطوری یه متن رو می‌تونیم به clipboard کپی کنیم؟

شما باید محتوا (با استفاده از روش `select`) عنصر ورودی رو انتخاب کنین و دستور `copy` را با `execCommand('copy')` اجرا کنین (یعنی `execCommand('copy')`). شما همچنان می‌توانید سایر دستورات سیستم `cut` و `paste` رو اجرا کنین.

```

document.querySelector("#copy-button").onclick = function() {

 // Select the content

 document.querySelector("#copy-input").select();

 // Copy to the clipboard

 document.execCommand('copy');

};

```

## ۳۷۵. چطوری میشه timestamp رو بدست آورد؟

می‌توانید از «`Date.getTime`» برای دریافت مهر زمانی فعلی استفاده کنید. یه میانبر جایگزین برای دریافت مقدار وجود دارد.

```
console.log(+new Date());
console.log(Date.now());
```

## ۳۷۶. چطوری یه آرایه چندسطحی رو تک سطحی کنیم؟

مسطح کردن آرایه‌های 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 می‌توانید اونو با آرایه‌های چند بعدی با تماس‌های بازگشته کار کنید

```
function flattenMultiArray(arr) {
 const flattened = [].concat(...arr);
 return flattened.some(item => Array.isArray(item)) ?
 flattenMultiArray(flattened) : flattened;
}
const multiDimensionalArr = [11, [22, 33], [44, [55, 66, [77, [88]], 99]]];
const flatArr = flattenMultiArray(multiDimensionalArr); // [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

## ۳۷۷. ساده‌ترین روش برای بررسی چندشرطی چیه؟

می‌توانید از «`indexOf`» برای مقایسه ورودی با چندین مقدار به جای بررسی هر مقدار به عنوان یه 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();
}
```

## ۳۷۸. چطوری کلیک روی دکمه برگشت مرورگر رو متوجه بشیم؟

روش «window.onbeforeunload» برای ضبط رویدادهای دکمه بازگشت مرورگر استفاده می‌شود. این برای هشدار دادن به کاربران در مورد از دست دادن داده‌های فعلی مفید است.

```
window.onbeforeunload = function() {
 alert("You work will be lost");
};
```

## ۳۷۹. چطوری می‌توانیم کلیک راست رو غیرفعال کنیم؟

کلیک راست روی صفحه رو می‌شه با برگرداندن false از ویژگی «oncontextmenu» در عنصر بدن غیرفعال کرد.

```
<body oncontextmenu="return false;">
```

## ۳۸۰. object-wrapper چی هستن؟

مقادیر اولیه مانند رشته، عدد و بولین خواص و روشی ندارن، اما زمانی که می‌خواهید اقداماتی رو روی اونا انجام بدین، به طور موقت به یه شی (آبجکت Wrapper) تبدیل یا مجبور می‌شن. برای مثال، اگه متدهایUpperCase رو روی یه مقدار رشته اولیه اعمال کنین خطای ایجاد نمی‌کنه، اما حروف بزرگ رشته رو برمی‌گردونه.

```
let name = "john";

console.log(name.toUpperCase()); // Behind the scenes treated
as console.log(new String(name).toUpperCase());
```

یعنی هر اولیه به جز null و undefined دارای wrapper Object هس و لیست اشیاء BigInt و String، Number، Boolean، Symbol wrapper عبارتند از

## ۳۸۱. AJAX چیه؟

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

## ۳۸۲. روش‌های مختلف مدیریت به کد Asynchronous چیه؟

۱. callback‌ها
۲. پرامیس‌ها
۳. Async/await
۴. کتابخانه‌های شخص ثالث مانند async.js، bluebird و غیره

## ۳۸۳. چطوری یه درخواست fetch رو کنسل کنیم؟

تا چند روز پیش، یکی از کاستی‌های وعده‌های بومی راه مستقیمی برای لغو درخواست fetch نیست. اما «AbortController» جدید از مشخصات js به شما امکان میده از سیگنالی برای لغو یه یا چند تماس fetch استفاده کنین.

جریان اصلی لغو یه درخواست fetch اینجوری می‌شه.

۱. یه نمونه «AbortController» ایجاد کنین

۲. ویژگی سیگنال یه نمونه رو دریافت کنین و سیگنال رو به عنوان یه گزینه fetch برای سیگنال ارسال کنین

۳. برای لغو تمام fetch‌هایی که از اون سیگنال استفاده میکنن با ویژگی's abort کال AbortController رو کال کنید.

برای مثال، بیاین یه سیگنال رو به چندین تماس fetch ارسال کنیم، همه درخواست‌های با آن سیگنال لغو میشن.

```

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);

```

## چیه؟ Speech-API .۳۸۴

API گفتار وب برای فعال کردن مروگرهای مدرن برای شناسایی و ترکیب گفتار (یعنی داده‌های صوتی در برنامه‌های وب) استفاده می‌شود. این API توسط انجمن W3C در سال 2012 معرفی شد و دارای دو بخش اصلیه.

۱. تشخیص گفتار (تشخیص گفتار ناهمزمان یا گفتار به متن): این امکان رو فراهم می‌کند که زمینه صدا رو از ورودی صوتی تشخیص داده و به اون پاسخ بدین. این توسط رابط "SpeechRecognition" قابل دسترسیه. مثال زیر نحوه استفاده از این API برای دریافت متن از گفتار رو نشون میدهد.

```

window.SpeechRecognition = window.webkitSpeechRecognition ||

window.SpeechRecognition; // webkitSpeechRecognition for

Chrome and SpeechRecognition for FF

const recognition = new window.SpeechRecognition();

recognition.onresult = (event) => { // SpeechRecognitionEvent

type

 const speechToText = event.results[0][0].transcript;

 console.log(speechToText);

}

recognition.start();

```

در این API، مرورگر برای استفاده از میکروفون شما از شما اجازه می خواهد این امکان را فراهم می کنه تا زمینه صدا را از ورودی صوتی تشخیص داده و پاسخ بده. این توسط رابط "SpeechSynthesis" قابل دسترسیه.

برای مثال، کد زیر برای دریافت صدا/گفتار از متن استفاده می شه.

```

if('speechSynthesis' in window){

 var speech = new SpeechSynthesisUtterance('Hello World!');

 speech.lang = 'en-US';

 window.speechSynthesis.speak(speech);

}

```

نمونه های بالا را می شه روی کنسول برنامه نویس مرورگر کروم (+33) آزمایش کرد.  
**توجه:** این API هنوز یه پیش نویس فعله و فقط در مرورگرهای کروم و فایرفاکس وجود داره (البته کروم فقط مشخصات رو اجرا می کنه)

## حداقل timeout توی throttling چقدر؟ ۳۸۵

هم مرورگر و هم محیط های جاوا سکریپت NodeJS با حداقل تاخیری که بیشتر از 0 میلی ثانیه اس throttles رو انجام می دهند. این بدان معناست که حتی اگه تنظیم یه تاخیر 0ms به طور آنی اتفاق نیوفته.

**مرورگرها:** حداقل 4 میلی ثانیه تاخیر دارن. این throttles زمانی اتفاق میوفته که تماس های متوالی به دلیل تودرتوی Callback (عمق معین) یا پس از تعداد معینی فواصل متوالی آغاز شه.

**توجه:** مرورگرهای قدیمی حداقل 10 میلی ثانیه تاخیر دارن.

**Nodejs:** حداقل 1ms تاخیر دارن. این throttles زمانی اتفاق میوفته که تاخیر بزرگتر از

بهترین مثال برای توضیح این رفتار throttling وقفه، ترتیب قطعه کد.

```
function runMeFirst() {
 console.log('My script is initialized');
}
setTimeout(runMeFirst, 0);
console.log('Script loaded');
```

و خروجی

```
Script loaded
My script is initialized
```

اگه از «setTimeout» استفاده نمیکنین ترتیب گزارش‌ها این شکلی می‌شه.

```
function runMeFirst() {
 console.log('My script is initialized');
}
runMeFirst();
console.log('Script loaded');
```

و خروجی

```
My script is initialized
Script loaded
```

## ۳۸۶. چطوری می‌شه یه timeout صفر توی مرورگر اجرا کرد؟

به دلیل حداقل تاخیر بیش از ۰ میلی ثانیه، نمیتوانیں از setTimeout(fn, 0) برای اجرای فوری کد استفاده کنین. اما برای دستیابی به این رفتار میتوانید از window.postMessage استفاده کنین.

## ۳۸۷. task‌ها توی event-loop چی هستن؟

وظیفه هر کد/برنامه جاواسکریپتیه که قراره توسط مکانیسم‌های استانداره اجرا شه، مثل شروع اولیه اجرای یه برنامه، اجرای یه رویداد، callback یا یه بازه زمانی یا وقفه در حال

اجرا. همه این وظایف تویه صفت کار برنامه ریزی میشن در زیر لیستی از موارد استفاده برای افزودن وظایف به صفت کار اوردیم.

۱. موقعی که یه برنامه جاواسکریپت جدید مستقیماً از کنسول اجرا میشه یا توسط عنصر `<script>` اجرا میشه، وظیفه به صفت کار اضافه میشه.
۲. موقعی که یه رویداد شلیک میشه، پاسخ تماس رویداد به صفت کار اضافه میشه

۳. وقتی به یه `setInterval` یا `setTimeout` رسید، پاسخ تماس مربوطه به صفت کار اضافه میشه

## ۳۸۸. **چی هستن؟ microtask**

کد جاواسکریپته که باید بلافصله پس از تکمیل task/Microtask در حال اجرا اجرا شه. اونا در طبیعت به نوعی مسدود کننده هستن. یعنی تا زمانی که صفت microtask خالی نشه، رشته اصلی مسدود میشه.

منابع اصلی Microtask ها عبارتند از، `Promise.resolve`, `Promise.reject`, `MutationObservers`, `IntersectionObservers` و غیره.

**توجه:** همه این Microtask ها در همون چرخش event-loop پردازش میشن

## ۳۸۹. **event-loop کداما هستن؟**

- این حلقه اصلی یه برنامه اس. به طور معمول این در پایین صفحه اصلیه. خروج معمولاً نشان دهنده تمایل به بسته شدن برنامه اس. در هر برنامه فقط یکی از این موارد میتونه وجود داشته باشد. - این حلقه ایه که معمولاً در پایین رویه اصلی یه UI یافت میشه.

## ۳۹۰. **هدف از queueMicrotask چیه؟**

این به کد شما اجازه میده بدون تداخل با کد دیگه با اولویت بالاتری که در حالت تعلیقه اجرا بشه اما قبل از اینکه مرورگر کنترل روی زمینه اجرا رو دوباره به دست آورد، بسته به کاری که باید تکمیل کنین.

## ۳۹۱. چطوری می‌شه از کتابخونه‌های جاواسکریپت توی فایل typescript استفاده کرد؟

مشخصه که همهی کتابخونه‌ها یا چارچوب‌های جاواسکریپت دارای فایل‌های اعلان TypeScript نیستن. اما اگه هنوزم می‌خواین از کتابخانه‌ها یا فریمورک‌ها در فایل‌های TypeScript بدون دریافت خطاهای کامپایل استفاده کنین تنها راحل کلمه کلیدی declare به همراه یه اعلان متغیره. برای مثال، بیاین تصور کنیم که شما یه کتابخونه به نام "customLibrary" دارید که اعلان TypeScript نداره و فضای نامی به نام "customLibrary" در فضای نام گلوبال داره. می‌تونین از این کتابخانه در کد تایپ اسکریپت به صورت زیر استفاده کنین.

```
declare var customLibrary;
```

در زمان اجرا، تایپ اسکریپت نوع اونو به متغیر «کتابخانه سفارشی» به صورت «هرگونه» ارائه می‌کنه. جایگزین دیگه بدون استفاده از کلمه کلیدی declare رو تو مثال زیر ببینیم

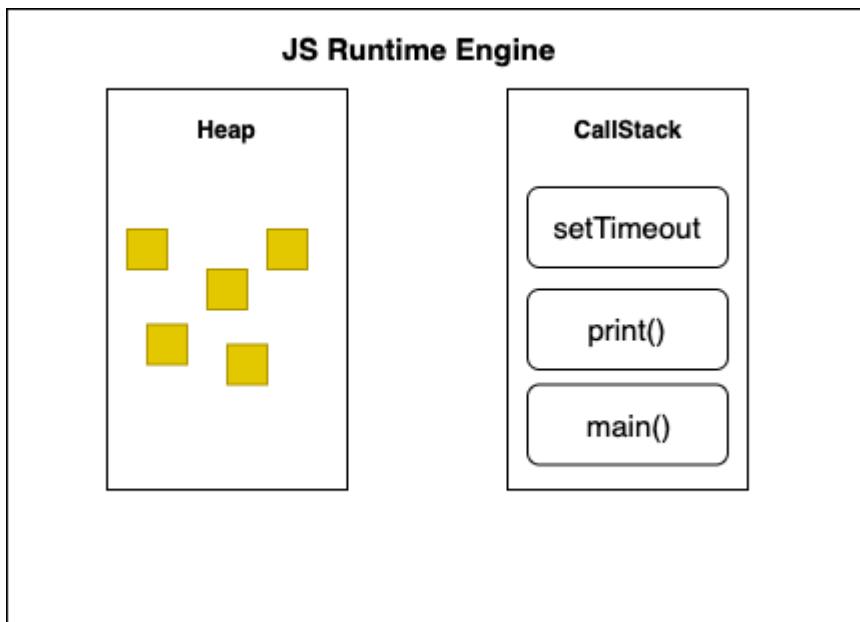
```
var customLibrary: any;
```

## ۳۹۲. تفاوت‌های بین observable‌ها و promise‌ها کداما هستن؟

observable‌ها	promise‌ها
چندین مقدار رو تو یه دوره زمانی منتشر می‌کنه (جریان مقادیری از ۰ تا چندگانه)	فقط یه مقدار رو تو یه زمان منتشر می‌کنه
اونا برای فراخوانی نیاز به اشتراک دارن	قراره فوراً فراخواتی شن
observable‌ها می‌تونن همزمان یا ناهمزمان	Promise همی‌شه ناهمزمانه حتی اگه بلافاصله حل شه
اپراتورهایی مانند map، forEach، filter، reduce، retryWhen و غیره رو ارائه میده.	هیچ اپراتور ارائه نمیده
با استفاده از روش unsubscribe لغو می‌شن	قابل لغو نیست

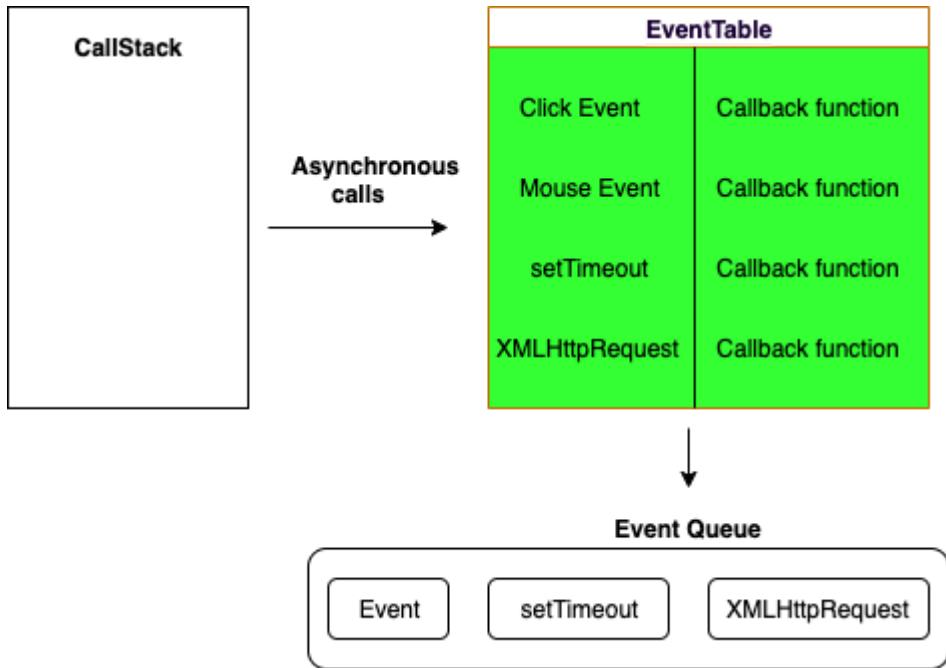
## ۳۹۳. heap چیه؟

Heap (یا پشته حافظه) محلیه که تو اون آبجکت‌ها در موقع تعریف متغیرها ذخیره می‌شون یعنی این محلیه که تمام تخصیص حافظه و عدم تخصیص تو اون انجام می‌شه. هر دو و در ظرف زمان اجرا JS call-stack و heap هستن. هر زمان که زمان اجرا با متغیرها و اعلان‌های تابع در کد مواجه می‌شه، او نا رو در Heap ذخیره می‌کنه.



## ۳۹۴. event-table چیه؟

Event Table یه ساختار داده‌ایه که تمام رویدادهایی رو که به صورت ناهمزمان اجرا می‌شن، مانند پس از مدتی فاصله زمانی یا پس از رفع بعضی از درخواست‌های API، ذخیره و ردیابی می‌کنه. یعنی هر زمان که یه تابع `setTimeout` رو فراخوانی کنین یا عملیات `async` رو فراخوانی کنین به جدول رویداد اضافه می‌شه. توابع رو به تنهایی اجرا نمی‌کنه. هدف اصلی جدول رویدادها پیگیری رویدادها و ارسال اونا به صف رویداد همونطور که در نمودار زیر نشون داده شده.



## ۳۹۵. صف microTask چیه؟

promise صف جدیدیه که تو اون تمام وظایف آغاز شده توسط اشیاء Microtask Queue قبل از صف برگشت پردازش میشن microtasks قبل از کارهای رندر و نقاشی بعدی پردازش میشه. اما اگه این ریزکارها برای مدت طولانی اجرا شن، منجر به تخریب بصری میشن.

## ۳۹۶. تفاوت بین shim و polyfill چیه؟

shim کتابخونه ایه که یه API جدید رو با استفاده از ابزارهای آن محیط به یه محیط قدیمی تر میاره. لزوماً محدود به یه برنامه وب نیست. برای مثال، es5-shim.js برای شبیه سازی ویژگی های ES5 در مرورگرهای قدیمی (عمدتاً قبل از IE9) استفاده میشه. در حالی که polyfill یه قطعه کد (یا افزونه) اس که فناوری رو ارائه میکنه که شما، توسعه دهنده، از مرورگر انتظار دارید که به صورت بومی ارائه کنه. تو یه جمله ساده، API A برای shim یه polyfill است.

## ۳۹۷. چطوری متوجه primitive یا غیر primitive بودن یه نوع داده میشیم؟

در جاواسکریپت، انواع ابتدایی عبارتند از ،boolean، string، number، BigInt، null و undefined. در حالی که انواع غیر ابتدایی شامل Objectها می‌شوند. اما با تابع زیر می‌توانید به راحتی اونا را شناسایی کنید:

```
var myPrimitive = 30;
var myNonPrimitive = {};
function isPrimitive(val) {
 return Object(val) !== val;
}

isPrimitive(myPrimitive);
isPrimitive(myNonPrimitive);
```

اگه مقدار یه نوع داده اولیه باشه، سازنده Object یه آبجکت پوشاننده جدید برای مقدار ایجاد می‌کنه. اما اگه مقدار یه نوع داده غیر ابتدایی (یک آبجکت) باشه، سازنده Object همون آبجکت رو میده.

## ۳۹۸. babel چیه؟

Babel یه ترانسپایلر جاواسکریپت برای تبدیل کد ECMAScript 2015 + به یه نسخه سازگار جاواسکریپت در مرورگرها یا محیط‌های فعلی و قدیمی تر است. بعضی از ویژگی‌های اصلی در زیر ذکر شده است،

۱. تبدیل نحو
۲. ویژگی‌های fill که در محیط هدف شما وجود نداره (با استفاده از (babel/polyfill@
۳. تبدیل کد منبع (یا کد مدر

## ۳۹۹. آیا Node.js به شکل کامل تک thread کار می‌کنه؟

Node یه رشته است، اما بعضی از توابع موجود در کتابخانه استانداره Node.js (برای مثال، `fs`) تک رشته‌ای نیستند. یعنی منطق اونا خارج از رشته Node.js اجرا می‌شوند تا سرعت و عملکرد یه برنامه رو بهبود بخشد.

## ۴۰۰. کاربردهای مرسوم observable کداما هستن؟

بعضی از رایج‌ترین موارد استفاده از موارد مشاهده شده عبارتند از سوکت‌های وب با اعلان‌های فشار، تغییرات ورودی کاربر، فواصل تکراری و غیره.

## ۴۰۱. RxJS چیه؟

(افزونه‌های واکنش‌گرا برای جاواسکریپت) کتابخانه‌ای برای پیاده‌سازی برنامه‌نویسی واکنش‌گرا با استفاده از مشاهده‌پذیره که نوشتمن کد ناهمزمان یا مبتنی بر تماس رو آسان‌تر می‌کنه. همچنین توابع کاربردی رو برای ایجاد و کار با مشاهده‌پذیرها فراهم می‌کنه.

## ۴۰۲. تفاوت بین function-declaration و Function-constructor چیه؟

توابعی که با "سازنده تابع" ایجاد می‌شن، برای زمینه‌های ایجاد خود بسته ایجاد نمی‌کن، اما همی‌شه در محدوده جهانی ایجاد می‌شون یعنی تابع فقط می‌توانه به متغیرهای محلی خود و متغیرهای دامنه جهانی دسترسی داشته باشد. در حالی که اعلان‌های تابع می‌توان به متغیرهای تابع بیرونی (بسته شدن) هم دسترسی داشته باشند.  
بیان این تفاوت رو با یه 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
```

## ۴۰۳. شرط Short-circuit یا اتصال کوتاه چیه؟

شرایط اتصال کوتاه برای روش فشرده نوشتگی دستورات if ساده در نظر گرفته شده است. بیاین سناریو رو با استفاده از یه مثال نشون بدیم. اگه می خواین وارد پورتالی با شرایط احراز هویت شوید، عبارت زیر خواهد بود:

```
if (authenticate) {
 loginToPorta();
}
```

از اونجایی که عملگرهای منطقی جاوا اسکریپت از چپ به راست ارزیابی می شن عبارت بالا رو می شه با استفاده از عملگر منطقی && ساده کرد.

```
authenticate && loginToPorta();
```

## ۴۰۴. ساده‌ترین روش برای تغییر سایز یه آرایه چیه؟

ویژگی length یه آرایه برای تغییر اندازه یا خالی کردن سریع آرایه اس. بیاین ویژگی رو روی آرایه اعداد اعمال کنیم تا تعداد عناصر رو از ۵ به ۲ تغییر بدیم.

```
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 رو هم می شه خالی کرد

```
var array = [1, 2, 3, 4, 5];
array.length = 0;
console.log(array.length); // 0
console.log(array); // []
```

## ۴۰۵. observable چیه؟

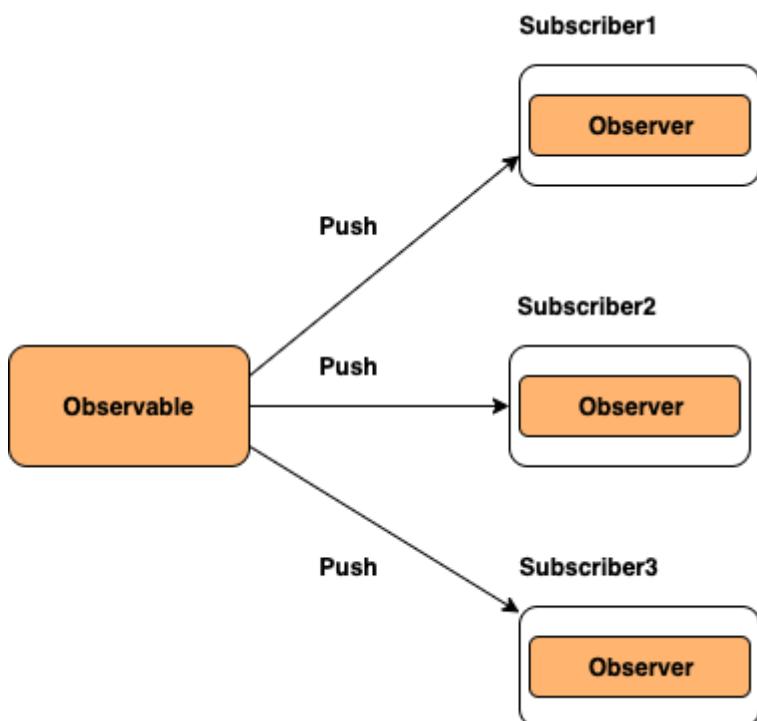
اساساً Observable تابعیه که میتوانه جریانی از مقادیر رو به صورت همزمان یا ناهمزمان به یه ناظر در طول زمان برگردونه. مصرف کننده میتوانه با فراخوانی متده «subscribe» مقدار رو دریافت کنه.

بیاین به یه مثال ساده از یه 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:** Observable‌ها هنوز بخشی از زبان جاواسکریپت نیستن اما پیشنهاد شده که به زبان اضافه بشن.

## ۴۰. تفاوت‌های بین توابع و کلاس‌ها چیه؟

تفاوت اصلی بین اعلان‌های تابع و اعلان‌های کلاس "بالا بردن" هست. اعلان‌های تابع بالا می‌رن اما اعلان‌های کلاس نه.

### :Classes

```
const user = new User(); // ReferenceError

class User {}
```

### :Constructor Function

```
const user = new User(); // No error

function User() {
}
```

## تابع `async` چیه؟

یه تابع `async` تابعیه که با کلمه کلیدی `async` اعلام شده که با اجتناب از زنجیره پرامیس رفتار ناهمزمان و مبتنی بر قول به سبک تمیزتری نوشته شه. این توابع میتوونن شامل صفر یا بیشتر عبارت `await` باشن.  
بیاین یه مثال تابع همگام زیر رو در نظر بگیریم،

```
async function logger() {

 let data = await fetch('http://someapi.com/users'); // pause
 until fetch returns
 console.log(data)
}
logger();
```

این اساساً خوبی نحوی بیش از پرامیس‌ها و توابع جنریتور ES2015 هست.

## چطوری خطاهای ایجاد شده هنگام استفاده از `promise` را کنترل کنیم؟

در حین استفاده از کد ناهمزمان، `promise`‌های ES6 جاواسکریپت میتوونن زندگی شما رو بدون داشتن هرم `callback`ها و مدیریت خطا در هر خط دوم بسیار آسان‌تر کنن. اما

Promise‌ها مشکلاتی دارن و بزرگترین اونا به طور پیش‌فرض بلعیدن خطاهاس. فرض کنین انتظار دارید برای تمام موارد زیر یه خط توى کنسول چاپ کنن.

```
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 جاوا‌اسکریپت مدرن وجود دارن که هیچ خطایی رو چاپ نمی‌کنن. شما می‌تونین این مشکل رو به روش‌های مختلف حل کنین

**Add catch block at the end of each chain:** You can add catch .  
block to the end of each of your promise chains

```
Promise.resolve('promised value').th (function() {
 throw new Error('error');
}).catch(function(error) {
 console.error(error.stack);
});
```

اما تایپ کردن برای هر زنجیره پرمیس‌ها و پرخاطب هم خیلی سخته.  
**۲. Add done method :** می‌تونین ابتدا راه حل‌ها رو جایگزین کنین و بعد با روش انجام شده بلوک‌ها رو بگیرید

```
Promise.resolve('promised value').don(function() {
 throw new Error('error');
});
```

فرض کنین می‌خواین داده‌ها رو با استفاده از fetch HTTP کنین و بعداً پردازش داده‌های حاصل رو به صورت ناهمزمان انجام بدین. می‌تونین بلوک «انجام شد» رو به صورت زیر بنویسین.

```
getDataFromHttp()
 .then(function(result) {
 return processDataAsync(result);
 })
 .done(function(processed) {
 displayData(processed);
 });
});
```

در آینده، اگه API کتابخانه پردازش به همگام تغییر کنه، می‌تونین بلوک «انجام شد» رو  
مانند زیر حذف کنین.

```
getDataFromHttp()
 .then(function(result) {
 return displayData(processDataAsyn(result));
 })
});
```

سپس فراموش کردین که بلوک «انجام شد» رو به بلوک then اضافه کنین که منجر به  
خطاهای خاموش می‌شه.

### :Extend ES6 Promises by Bluebird.<sup>۱۳</sup>

Bluebird API میاد های اکما اسکریپت رو گسترش میده تا در راه حل  
دوم مشکلی ایجاد نشه. این کتابخانه یه کنترل کننده "پیش فرض" در  
Promises هس که تمام خطاهای رو از Rejection رد شده به stderr چاپ  
می‌کنه. پس از نصب، می‌تونین ردهای کنترل نشده رو پردازش کنین

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

یه reject رو انجام بدین فقط با یه catch خالی اونو مدیریت کنین

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

## چیه؟ Deno.<sup>۱۴۰۹</sup>

یه ران تایم(run-time) ساده، مدرن و ایمن برای جاواسکریپت و تایپ اسکریپته که از  
موتور جاواسکریپت V8 و زبان برنامه نویسی Rust استفاده می‌کنه و توسط رایان دال، خالق  
نود جی اس استارت توسعه اش زده شده.

## ۱۴۰. توى جاواسكريپت چطورى يه object قابل پىمايش درست كنيم؟

به طور پيش فرض، اشیاء ساده قابل تكرار نىستن. اما مىتونىن با تعريف ويزگى «Symbol.iterator» روی اون شى رو قابل تكرار كنин. بىان اين رو با يه مثال نشون بديم،

```
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}
```

فرآيند بالا رو مىشه با استفاده از يه تابع مولد ساده كرد،

```

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}

```

## ۱۱. روش مناسب برای فراخوانی توابع بازگشتی چیه؟

ابتدا، قبل از صحبت در مورد "دوم خوانی مناسب" باید در مورد دم دم بدونیم. فراخوانی دنباله یه فراخوانی فرعی یا تابعیه که به عنوان آخرین عمل یه تابع فراخوانی انجام میشه. در حالی که **فراخوانی دنباله مناسب (PTC)** تکنیکیه که تو اون برنامه یا کد فریم‌های پشته ای اضافی برای بازگشت ایجاد نمیکنه، زمانی که فراخوانی تابع یه فراخوانی دنباله اس. برای مثال، بازگشت کلاسیک یا سر تابع فاکتوریل زیر به پشته برای هر مرحله بستگی داره. هر مرحله باید تا " $n * \text{فاکتوریل}(1 - n)$ " پردازش شه

```

function factorial(n) {
 if (n === 0) {
 return 1
 }
 return n * factorial(n - 1)
}

console.log(factorial(5)); //120

```

اما اگه از Tail callback استفاده میکنین اونا تمام دادههای لازم رو که به آن نیاز داره رو بدون تکیه بر پشته، در بازگشت به پایین منتقل میکن.

```

function factorial(n, acc = 1) {
 if (n === 0) {
 return acc
 }
 return factorial(n - 1, n * acc)
}
console.log(factorial(5)); //120

```

الگوی بالا همون خروجی مورد اول رو برمی‌گردونه. اما انباشت کننده کل رو به عنوان آرگومان بدون استفاده از حافظه پشته توی callBack رديابي می‌کنه.

## ۴۱۲. چطوری بررسی کنیم که یه آبجکت promise هست یا نه؟

اگه نمی‌دونین یه مقدار یه promise هس یا نه، مقدار رو به صورت `(Promise.resolve(value))` بپیچید که یه قول رو برمی‌گردونه.

```

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 اينه که نوع `handler.then` رو بررسی کنин

```

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

```

## ۴۱۳. چطوری متوجه بشیم که یا تابع با تابع constructor صدا زده شده یا نه؟

می‌توینیں از ویژگی شبیه «new.target» برای تشخیص اینکه آیا یه تابع به عنوان سازنده (با استفاده از عملگر جدید) فراخوانی شده یا به عنوان یه فراخوانی تابع معمولی استفاده کنین.

۱. اگه سازنده یا تابعی با استفاده از عملگر جدید فراخوانی شه، یه new.target مرجع به سازنده یا تابع برمی‌گردونه.
۲. برای فراخوانی تابع، new.target تعریف نشده اس.

```

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

```

## ۴۱۴. تفاوت‌های بین آبجکت rest و argument پارامتر چیه؟

۱. آبجکت آرگومان‌ها آرایه ماننده اما آرایه نیست. در حالی که بقیه پارامترها نمونه‌های آرایه هستن.

۲. آبجکت آرگومان‌ها از روش‌هایی مانند `sort`، `map`، `forEach` یا `pop` پشتیبانی نمی‌کنه. در حالی که این روش‌ها رو می‌شه در پارامترهای استراحت استفاده کرد.

۳. بقیه پارامترها فقط اونایی هستن که نام جدگانه ای به اوナ داده نشده در حالی که آبجکت آرگومان‌ها شامل تمام آرگومان‌های ارسال شده به تابعه.

## ۴۱۵. تفاوت‌های بین عملگر `rest` و پارامتر `spread` چیه؟

پارامتر Rest تمام عناصر باقی مانده رو در یه آرایه جمع آوری می‌کنه. در حالی که عملگر Spread به تکرارپذیرها (آرایه‌ها / اشیاء / رشته‌ها) اجازه میده تا به آرگومان‌ها / عناصر منفرد گسترش پیدا کنن. یعنی پارامتر Rest مخالف عملگر spread هس.

## ۴۱۶. نوع‌های مختلف `generator`‌ها کداما هستن؟

: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 .۱۴

```

class MyClass {
 * myGeneratorMethod() {
 yield 1;
 yield 2;
 yield 3;
 }
}
const myObject = new MyClass();
const genObj = myObject.myGeneratorMethod();

```

### :Generator as a computed property .۱۵

```

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

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

```

## ۱۷ . کدوما هستن؟ iterable built-in های

۱. آرایه‌ها و TypedArrays
۲. رشته‌ها: روی هر کاراکتر یا نقاط کد یونیکد تکرار کنین
۳. نقشه‌ها: روی جفت‌های کلید-مقدار آن تکرار شه
۴. مجموعه‌ها: روی عناصر خود تکرار می‌شه
۵. آرگومان‌ها: یه متغیر خاص آرایه مانند در توابع

## ۴۱۸. تفاوت‌های بین حلقه for...in و for...of چیه؟

هم برای...in و هم برای...of دستورات روی ساختارهای داده از تکرار می‌شن. تنها تفاوت در مورد چیزیه که اونا تکرار می‌کنند:

۱. روی تمام کلیدهای خصوصیت شمارش پذیر یه شی تکرار می‌شه
۲. بیش از مقادیر یه شی قابل تکرار.

بیان 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 می‌شه حلقه اول، ۱، ۲ و newProp رو در حین تکرار روی شی آرایه ثبت می‌کنه. حلقه of..for روی مقادیر یه ساختار داده arr تکرار می‌شه و a، b، c رو در کنسول ثبت می‌کنه.

## ۴۱۹. چطوری property های instance و غیر instance تعريف می‌کنی؟

خصوصیات Instance باید در داخل متدهای کلاس تعريف بشن. برای مثال، مشخصات نام و سن سازنده داخلی هم مثل مثال پایین تعريف میشن.

```
class Person {
 constructor(name, age) {
 this.name = name;
 this.age = age;
 }
}
```

اما خصوصیات داده Static(class) و نمونه اولیه باید خارج از اعلان ClassBody (Person) به صورت زیر اختصاص بدیم.

```
Person.staticAge = 30;
Person.prototype.prototypeAge = 40;
```

## ۴۲۰. تفاوت‌های بین Number.isNaN و isNaN هستن؟

۱. تابع سراسری «isNaN» آرگومان رو به عدد تبدیل می‌کنه و اگه مقدار NaN باشه، true رو برمی‌گردونه.
۲. این روش آرگومان رو تبدیل نمی‌کنه. اما زمانی که نوع یه عدد و مقدار NaN باشه مقدار true رو برمی‌گردونه.

بیاین تفاوت رو با یه مثال ببینیم،

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