

به نام خدا

مجموعه سوال و جواب‌های جاوسکریپت

عیسی رضائی



Mariotek platform

Mariotek.ir

مجموعه سوال و جواب‌های جاواسکریپتی

اگه از کتاب خوشتون اوmd به گیت‌هاب‌مون مراجعه کنین و بهمون ★ بدین. اگه هم قصد مشارکت داشتید همونجا می‌تونین شروع کنین و ما هم خیلی خوشحال می‌شیم 😊

<https://github.com/mariotek>

لینک گیت‌هاب ما برای مشارکت برای تولید کتاب‌ها:

نحوه 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 object‌ها توى جاواسکریپت کدوما هستن؟	۲۹
۲	زنجیره prototype چیه؟	۳۱
۳	تفاوت‌های بین bind و apply کدوما هستن؟	۳۱
۴	فرمت JSON چیه و عملیات‌های معمول روی اون چیا هستن؟	۳۳
۵	هدف از متدهای slice روى آرایه‌ها چیه؟	۳۳
۶	هدف از متدهای splice روى آرایه‌ها چیه؟	۳۴
۷	تفاوت متدهای slice و splice چیا هستن؟	۳۴
۸	تفاوت‌های Object و Map چیا هستن؟	۳۵
۹	تفاوت‌های بین عملگرهای == و === چیا هستن؟	۳۵
۱۰	تابع arrow-function یا lambda چی هستن؟	۳۶
۱۱	یه تابع first-class چجور تابعیه؟	۳۶
۱۲	یه تابع first-order چجور تابعیه؟	۳۷
۱۳	یه تابع higher-order چجور تابعیه؟	۳۷
۱۴	یه تابع unary چجور تابعیه؟	۳۸
۱۵	توابع currying چی؟	۳۸
۱۶	چه توابعی pure هستن؟	۳۸
۱۷	هدف از کلمه کلیدی let چیه؟	۳۹
۱۸	تفاوت‌های کلمات کلیدی let و var چیا هستن؟	۴۰

ردیف	سوال	صفحه
۱۹	دلیل انتخاب کلمه کلیدی let چیه؟	۴۱
۲۰	چطوری می‌تونیم توی بلوک مربوط به switch بدون دریافت خطا متغیر تعریف کنیم؟	۴۱
۲۱	Temporal-Dead-Zone چیه؟	۴۲
۲۲	(توابع بلا فاصله صدا زده شده) IIFE چی هستن؟	۴۲
۲۳	مزایای استفاده از module چیه؟	۴۳
۲۴	Memoization چیه؟	۴۳
۲۵	Hoisting چیه؟	۴۴
۲۶	Class چیکار می‌کنن؟	۴۵
۲۷	Closure چیا هستن؟	۴۵
۲۸	Module چیا هستن؟	۴۶
۲۹	چرا به module ها نیاز داریم؟	۴۶
۳۰	توی جاوا سکریپت scope چیه و چیکار می‌کننه؟	۴۷
۳۱	service-worker چیه؟	۴۷
۳۲	توی service-worker چطوری می‌شه DOM رو دستکاری کرد؟	۴۷
۳۳	چطوری می‌تونیم بین ریست شدن‌های service-worker داده‌های مورد نظرمون رو مجدد استفاده کنیم؟	۴۷
۳۴	IndexedDB چیه؟	۴۸
۳۵	Web-storage چیه؟	۴۸
۳۶	Post-message چیه؟	۴۸
۳۷	Cookie چیه؟	۴۸
۳۸	چرا به cookie نیاز داریم؟	۴۹

صفحه	سوال	ردیف
۴۹	گزینه‌های قابل تنظیم توی cookie چیا هستن؟	۳۹
۵۰	چطوری میشه یه cookie رو حذف کرد؟	۴۰
۵۰	تفاوت‌های بین session-storage و local-storage و cookie چیا هستن؟	۴۱
۵۰	تفاوت‌های بین sessionStorage و localStorage چیا هستن؟	۴۲
۵۱	چطوری به web-storage دسترسی پیدا میکنی؟	۴۳
۵۱	چه متدهایی روی session-storage قابل استفاده هستن؟	۴۴
۵۲	رخداد storage چیه و چطوری ازش استفاده میکنیم؟	۴۵
۵۲	چرا به web-storage نیاز داریم؟	۴۶
۵۲	چطوری میتونیم پشتیبانی از web-storage توسط مرورگر رو بررسی کنیم؟	۴۷
۵۳	چطوری میتونیم پشتیبانی از web-worker توسط مرورگر رو بررسی کنیم؟	۴۸
۵۳	یه مثال از web-workerها میتونی بزنی؟	۴۹
۵۴	محدودیت‌های web-workerها روی DOM چیا هستن؟	۵۰
۵۵	چیه Promise	۵۱
۵۶	چرا به Promise نیاز داریم؟	۵۲
۵۶	سه تا وضعیت ممکن برای یه Promise چیا هستن؟	۵۳
۵۷	توابع callback چی هستن؟	۵۴
۵۷	چرا به توابع callback نیاز داریم؟	۵۵
۵۸	یا جهنم توابع callback چیه؟	۵۶
۵۹	یا همون Server-sent-events چیه؟	۵۷
۵۹	چطوری میتونیم پیام‌های server-sent-event رو دریافت کنیم؟	۵۸
۵۹	چطوری میتونیم پشتیبانی مرورگر برای SSE رو بررسی کنیم؟	۵۹

ردیف	سوال	صفحه
۶۰	کدوم توابع روی SSE وجود دارن؟	۶۰
۶۱	اصلی‌ترین قوانین Promise‌ها چیا هستن؟	۶۱
۶۰	توى callback چطوری رخ میده؟	۶۲
۶۱	زنجیره Promise‌ها چیه؟	۶۳
۶۲	کاربرد متدهای promise.all چیه؟	۶۴
۶۳	هدف از متدهای race چیه؟	۶۵
۶۳	حالت strict توى جاوااسکریپت چی کار میکنه؟	۶۶
۶۳	چرا به حالت strict نیاز داریم؟	۶۷
۶۴	چطوری می‌توانیم حالت strict را فعال کنیم؟	۶۸
۶۴	هدف از عملگر نقطی دوتا بی(!!) چیه؟	۶۹
۶۵	هدف از عملگر delete چیه؟	۷۰
۶۵	عملگر typeof چیکار میکنه؟	۷۱
۶۵	undefined چیه و چه زمانی می‌گیریم؟	۷۲
۶۵	null چیه؟	۷۳
۶۶	تفاوت‌های بین null و undefined چیا هستن؟	۷۴
۶۶	متدهای eval چیه؟	۷۵
۶۷	تفاوت‌های بین window و document چیا هستن؟	۷۶
۶۷	توى جاوااسکریپت چطوری می‌توانیم به history دسترسی داشته باشیم؟	۷۷
۶۸	انواع داده‌های جاوااسکریپت کدامها هستند؟	۷۸
۶۸	NaN چیه و چیکار میکنه؟	۷۹
۶۸	تفاوت‌های بین undeclared و undefined چیا هستن؟	۸۰

صفحه	سوال	ردیف
۶۹	کدوم متغیرها عمومی هستن؟	۸۱
۶۹	مشکلات متغیرهای عمومی چیا هستن؟	۸۲
۶۹	مقدار NaN چیه؟	۸۳
۶۹	هدف از تابع isFinite چیه؟	۸۴
۷۰	یه event-flow چیه؟	۸۵
۷۰	Event-bubbling چیه؟	۸۶
۷۰	Event-capturing چیه؟	۸۷
۷۰	چطوری میشه یه فرم رو با استفاده از جاواسکریپت ثبت کرد؟	۸۸
۷۱	چطوری میشه به اطلاعات مربوط به سیستم عامل کاربر دسترسی داشت؟	۸۹
۷۱	تفاوت‌های بین رخدادهای DOMContentLoaded و document-load چیا هستن؟	۹۰
۷۱	تفاوت‌های بین object‌های host ، native و user چیا هستن؟	۹۱
۷۲	کدوم ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده میشون؟	۹۲
۷۲	مزایا و معایب استفاده از Promise‌ها به جای callback چیا هستن؟	۹۳
۷۲	تفاوت‌های بین property و attribute روی DOM چیا هستن؟	۹۴
۷۳	سیاست same-origin چیه؟	۹۵
۷۳	هدف استفاده از void چیه؟	۹۶
۷۴	جاواسکریپت یه زبان تفسیری هست یا کامپایلری؟	۹۷
۷۴	آیا جاواسکریپت یه زبان حساس به بزرگی و کوچکی(case-sensitive) و کوچکی حروف است؟	۹۸
۷۴	ارتباطی بین Java و JavaScript وجود داره؟	۹۹
۷۴	چی هستن؟ Event‌ها	۱۰۰

ردیف	سوال	صفحه
۱۰۱	کی جاواسکریپت رو ساخته؟	۷۵
۱۰۲	هدف از متد preventDefault چیه؟	۷۵
۱۰۳	کاربرد متد stopPropagation چیه؟	۷۶
۱۰۴	مراحلی که موقع استفاده از event-handler توی یه return false رخ میده چیا هستن؟	۷۶
۱۰۵	BOM چیه؟	۷۶
۱۰۶	موارد استفاده از setTimeout کدوما هستن؟	۷۷
۱۰۷	موارد استفاده از setInterval کدوما هستن؟	۷۷
۱۰۸	چرا جاواسکریپت رو به عنوان یه زبان تک thread میشناسن؟	۷۸
۱۰۹	Event-delegation چیه؟	۷۸
۱۱۰	ECMAScript چیه؟	۷۸
۱۱۱	JSON چیه؟	۷۹
۱۱۲	قوانين فرمت JSON کدوما هستن؟	۷۹
۱۱۳	هدف از متد JSON.stringify چیه؟	۷۹
۱۱۴	چطوری میتونیم یه رشته JSON رو تجزیه کنیم؟	۷۹
۱۱۵	چرا به JSON نیاز داریم؟	۸۰
۱۱۶	PWAها چی هستن؟	۸۰
۱۱۷	هدف از متد clearTimeout چیه؟	۸۰
۱۱۸	هدف از متد clearInterval چیه؟	۸۱
۱۱۹	توی جاواسکریپت، چطوری میشه به یه صفحه جدید redirect انجام داد؟	۸۱
۱۲۰	چطوری بررسی میکنیم که یه string شامل یه substring هست یا نه؟	۸۲
۱۲۱	توی جاواسکریپت، چطوری مقدار یه آدرس email را اعتبارسنجی میکنیم؟	۸۲

صفحه	سوال	ردیف
۸۳	چطوری می‌تونیم مقدار آدرس url جاری رو بخونیم؟	۱۲۲
۸۳	ویژگی‌های مختلف url روی object مربوط به history کدوماً هستن؟	۱۲۳
۸۴	توی جاواسکریپت چطوری می‌تونیم مقدار یه query-string رو بخونیم؟	۱۲۴
۸۴	چطوری می‌تونیم بررسی کنیم که آیا یه پرایپری روی آبجکت وجود داره یا نه؟	۱۲۵
۸۵	چطوری روی یه object حلقه میزندی؟	۱۲۶
۸۵	چطوری تست می‌کنی که یه object خالیه؟	۱۲۷
۸۶	چطوری arguments object چیه؟	۱۲۸
۸۶	چطوری حرف اول یه رشته رو به حرف بزرگ تبدیل می‌کنی؟	۱۲۹
۸۷	مزایا و معایب حلقه for چیا هستن؟	۱۳۰
۸۷	چطوری تاریخ جاری رو توی جاواسکریپت نشون میدی؟	۱۳۱
۸۷	چطوری دو تا date object رو با هم مقایسه می‌کنی؟	۱۳۲
۸۸	چطوری بررسی می‌کنی که یه رشته با یه رشته دیگه شروع می‌شه؟	۱۳۳
۸۸	چطوری یه رشته رو trim می‌کنی؟	۱۳۴
۸۸	توی جاواسکریپت چطوری می‌تونیم یه زوج مرتب از key یه value بسازیم؟	۱۳۵
۸۹	آیا عبارت '!-' عملگر خاصی هست؟	۱۳۶
۸۹	چطوری می‌تونیم به متغیرهایمان مقادیر اولیه بدیم؟	۱۳۷
۸۹	چطوری می‌تونیم متن‌های چند خطی درست کنیم؟	۱۳۸
۹۰	مدل app-shell چیه؟	۱۳۹
۹۰	چطوری می‌تونیم روی یه تابع property اضافه کنیم؟	۱۴۰
۹۱	چطوری می‌تونیم تعداد پارامترهای ورودی یه تابع رو به دست بیاریم؟	۱۴۱
۹۱	Polyfill چیه؟	۱۴۲

ردیف	سوال	صفحه
۱۴۳	عبارات Break و continue چی هستن؟	۹۲
۱۴۴	توى جاواسکریپت labelها چیکار می‌کنن؟	۹۲
۱۴۵	مزایای declare کردن متغیرها در اوایل کد چیه؟	۹۲
۱۴۶	مزایای مقداردهی اولیه متغیرها چیه؟	۹۳
۱۴۷	روش توصیه شده برای ایجاد object چیه؟	۹۳
۱۴۸	چطوری می‌تونیم آرایه JSON تعریف کنیم؟	۹۳
۱۴۹	چطوری می‌تونیم اعداد تصادفی تولید کنیم؟	۹۴
۱۵۰	می‌تونی یه تابع تولید اعداد تصادفی توى یه بازه مشخص بنویسی؟	۹۴
۱۵۱	Tree-shaking چیه؟	۹۵
۱۵۲	دلایل نیاز به tree-shaking کدوما هستن؟	۹۵
۱۵۳	آیا استفاده از eval توصیه می‌شه؟	۹۵
۱۵۴	Regular-Expression چیه؟	۹۵
۱۵۵	متدهای رشته که روی Regular-expression مجاز هستن کدوماست؟	۹۶
۱۵۶	توی Regex بخش modifiers چیکار میکنه؟	۹۶
۱۵۷	پترن‌های regular-expression چیه؟	۹۷
۱۵۸	آبجکت RegExp چیه؟	۹۷
۱۵۹	چطوری روی یه رشته دنبال یه پترن RegExp می‌گرددی؟	۹۸
۱۶۰	هدف از متدهای exec چیه؟	۹۸
۱۶۱	چطوری استایل‌های یه المنت HTML رو تغییر میدی؟	۹۸
۱۶۲	نتیجه عبارت $3' + 2 + 1$ چی می‌شه؟	۹۹
۱۶۳	عبارة debugger چیکار میکنه؟	۹۹

صفحه	سؤال	ردیف
۹۹	هدف از breakpoint ها توی debugging چیه؟	۱۶۴
۹۹	آیا می‌تونیم از عبارت‌های رزرو شده در تعریف identifierها(اسم متغیر، کلاس و ...) استفاده کنیم؟	۱۶۵
۱۰۰	چطوری تشخیص بدیم که یه مرورگر mobile هست یا نه؟	۱۶۶
۱۰۲	چطوری بدون Regex تشخیص بدیم که یه مرورگر mobile هست یا نه؟	۱۶۷
۱۰۲	چطوری طول و عرض یه تصویر رو با جاواسکریپت به دست میاری؟	۱۶۸
۱۰۳	چطوری درخواست‌های synchronous HTTP بزنیم؟	۱۶۹
۱۰۳	چطوری درخواست‌های asynchronous HTTP بزنیم؟	۱۷۰
۱۰۳	چطوری یه تاریخ رو به یه تاریخ در timezone دیگه تبدیل کنیم؟	۱۷۱
۱۰۴	چه هایی برای اندازه‌گیری سایز window property به کار میره؟	۱۷۲
۱۰۴	عملگر شرطی سه گانه توی جاواسکریپت چیه؟	۱۷۳
۱۰۴	آیا می‌شه روی عملگر شرطی زنجیره شرط‌ها رو اعمال کرد؟	۱۷۴
۱۰۵	روش‌های اجرای جاواسکریپت بعد از لود شدن صفحه کدوما هستن؟	۱۷۵
۱۰۵	تفاوت‌های بین proto و prototype کدوما هستن؟	۱۷۶
۱۰۶	میتوانی یه مثال از زمانی که واقعا به سمیکولون(;) نیاز هست بزنی؟	۱۷۷
۱۰۶	متدهای freeze چیکار میکنه؟	۱۷۸
۱۰۷	هدف از متدهای freeze چیه؟	۱۷۹
۱۰۷	چرا به متدهای freeze نیاز داریم؟	۱۸۰
۱۰۷	چطوری می‌تونیم زبان ترجیحی یه مرورگر رو تشخیص بدیم؟	۱۸۱
۱۰۸	چطوری می‌تونیم حرف اول همه کلمات یه رشته رو به حرف بزرگ تبدیل کنیم؟	۱۸۲
۱۰۸	چطوری می‌شه تشخیص داد که جاواسکریپت یه صفحه وب غیرفعال شده؟	۱۸۳

صفحه	سوال	ردیف
۱۰۹	عملگرهای پشتیبانی شده توسط جاواسکریپت کدوما هستن؟	۱۸۴
۱۰۹	پارامتر rest چیکار میکنه؟	۱۸۵
۱۱۰	اگه پارامتر rest رو به عنوان آخرین پارامتر استفاده نکنیم چی میشه؟	۱۸۶
۱۱۰	عملگرهای منطقی باینری توی جاواسکریپت کدوما هستن؟	۱۸۷
۱۱۱	عملگر spread چیکار میکنه؟	۱۸۸
۱۱۱	چطوری تشخیص میدی که یه آبجکت freeze شده یا نه؟	۱۸۹
۱۱۱	چطوری بررسی کنیم که دو تا مقدار(شامل آبجکت) با هم برابرن یا نه؟	۱۹۰
۱۱۲	هدف از متدهای object چیه؟	۱۹۱
۱۱۲	چطوری object‌هایی که property رو به یه object دیگه کپی میکنی؟	۱۹۲
۱۱۳	کاربردهای متدهای assign چیه؟	۱۹۳
۱۱۳	آبجکت proxy چیه؟	۱۹۴
۱۱۴	هدف از متدهای seal چیه؟	۱۹۵
۱۱۴	کاربردهای متدهای seal چیه؟	۱۹۶
۱۱۵	تفاوت‌های بین متدهای seal و freeze چیا هست؟	۱۹۷
۱۱۵	چطوری تشخیص میدی که یه آبجکت seal شده یا نه؟	۱۹۸
۱۱۵	چطوری کلید و مقدارهای enumerable رو به دست میاری؟	۱۹۹
۱۱۶	تفاوت‌های بین متدهای Object.entries و Object.values چیا هست؟	۲۰۰
۱۱۶	چطوری لیست کلیدهای یه object رو بدست میاری؟	۲۰۱
۱۱۷	چطوری یه object با prototype درست میکنی؟	۲۰۲
۱۱۷	WeakSet چیه؟	۲۰۳
۱۱۸	تفاوت‌های بین Set و WeakSet کدوما هستن؟	۲۰۴

صفحه	سوال	ردیف
۱۱۸	لیست متدهایی که رو WeakSet قابل استفاده هستن رو می‌تونی بگی؟	۲۰۵
۱۱۹	WeakMap چیه؟	۲۰۶
۱۱۹	تفاوت‌های بین Map و WeakMap کدوما هستن؟	۲۰۷
۱۲۰	لیست متدهایی که رو WeakMap قابل استفاده هستن رو می‌تونی بگی؟	۲۰۸
۱۲۰	هدف از متدهای uneval چیه؟	۲۰۹
۱۲۱	چطوری یه URL رو encode می‌کنی؟	۲۱۰
۱۲۱	چطوری یه URL رو decode می‌کنی؟	۲۱۱
۱۲۱	چطوری محتوای یه صفحه رو پرینت می‌کیری؟	۲۱۲
۱۲۲	تفاوت‌های بین eval و uneval چیا هستن؟	۲۱۳
۱۲۲	تابع anonymous چیه؟	۲۱۴
۱۲۳	تفاوت تقدم بین متغیرهای local و global چطوریه؟	۲۱۵
۱۲۳	accessorهای جاواسکریپت چیکار می‌کنن؟	۲۱۶
۱۲۴	چطوری روی Object یه مقدار تعریف می‌کنی؟	۲۱۷
۱۲۴	تفاوت‌های بین get و defineProperty چیا هست؟	۲۱۸
۱۲۴	مزایای استفاده از Setter و Getter چیه؟	۲۱۹
۱۲۵	می‌توnim getter و setter رو با استفاده از متدهای defineProperty تعریف کنیم؟	۲۲۰
۱۲۵	هدف استفاده از switch-case چیه؟	۲۲۱
۱۲۶	چه قواعدی برای استفاده از switch-case باید رعایت بشه؟	۲۲۲
۱۲۶	نوع داده‌های primitive کدوما هستن؟	۲۲۳
۱۲۷	روش‌های مختلف دسترسی به propertyهای object کدوما هستن؟	۲۲۴
۱۲۷	قوانين پارامترهای توابع کدوما هستن؟	۲۲۵

صفحه	سوال	ردیف
۱۲۷	آبجکت error چیه؟	۲۲۶
۱۲۸	چه موقعی خطای syntax دریافت می‌کنیم؟	۲۲۷
۱۲۸	عنوان خطاهای مختلف که روی error-object برمیگردن کدوما هستن؟	۲۲۸
۱۲۹	عبارات مختلف که در هنگام مدیریت error استفاده میشن کدوما هستن؟	۲۲۹
۱۲۹	دو نوع مختلف حلقه‌ها تو جاواسکریپت کدوما هستن؟	۲۳۰
۱۲۹	آبجکت nodejs چیه؟	۲۳۱
۱۳۰	آبجکت Intl چیه؟	۲۳۲
۱۳۰	چطوری تاریخ و زمان رو بر اساس زبان جاری سیستم کاربر نمایش بدیم؟	۲۳۳
۱۳۰	آبجکت Iterator چیه؟	۲۳۴
		۱۳۱
۱۳۱	آبجکت Event-loop چیه؟	۲۳۶
۱۳۱	آبجکت Call-stack چیه؟	۲۳۷
۱۳۲	آبجکت Event-queue چیه؟	۲۳۸
۱۳۲	آبجکت Decorator چیه؟	۲۳۹
۱۳۳	مقادیر موجود روی آبجکت Intl کدوما هستن؟	۲۴۰
۱۳۴	عملگر Unary چیه؟	۲۴۱
۱۳۴	چطوری المنتهای موجود تو یه آرایه رو مرتب می‌کنی؟	۲۴۲
۱۳۴	هدف از تابع مرتب‌سازی موقع استفاده از متده sort چیه؟	۲۴۳
۱۳۴	چطوری آیتم‌های یه آرایه رو معکوس مرتب کنیم؟	۲۴۴
۱۳۵	چطوری حداقل و حداقل‌تر مقدار یه آرایه رو بدست بیاریم؟	۲۴۵
۱۳۵	چطوری حداقل و حداقل‌تر مقدار یه آرایه رو بدون استفاده از متدهای Math بدست بیاریم؟	۲۴۶

صفحه	سوال	ردیف
۱۳۶	عبارت خالی چیه و هدف از استفاده از ارش چیه؟	۲۴۷
۱۳۶	چطوری metadata یه مازول رو بدست میاری؟	۲۴۸
۱۳۷	عملگر comma چیه و چیکار میکنه؟	۲۴۹
۱۳۷	مزایای استفاده از عملگر comma چیه؟	۲۵۰
۱۳۸	چیه؟ Typescript	۲۵۱
۱۳۸	تفاوت‌های بین javascript و typescript کدوما هستن؟	۲۵۲
۱۳۹	مزایای typescript نسبت به javascript چیاست؟	۲۵۳
۱۳۹	چیه؟ object-initializer	۲۵۴
۱۴۰	متد constructor چیه؟	۲۵۵
۱۴۰	اگه متد constructor رو بیش از یه بار توی کلاس بنویسیم چی میشه؟	۲۵۶
۱۴۱	چطوری متد constructor کلاس والد رو صدا برزنیم؟	۲۵۷
۱۴۱	چطوری object یه prototype رو به دست میاری؟	۲۵۸
۱۴۱	اگه به متد getPrototype رشته پاس بدیم چی میشه؟	۲۵۹
۱۴۲	چطوری object یه prototype روی یه object دیگه ست کنیم؟	۲۶۰
۱۴۲	چطوری بررسی میکنی که یه object extend قابل هست یا نه؟	۲۶۱
۱۴۲	چطوری جلوی object extend یه object رو بگیریم؟	۲۶۲
۱۴۳	روش‌های مختلف برای تبدیل یه object به object غیرقابل extend چیه؟	۲۶۳
۱۴۳	چطوری property‌های متعددی رو روی یه object تعریف میکنی؟	۲۶۴
۱۴۴	منظور از MEAN توی جاواسکریپت چیه؟	۲۶۵
۱۴۴	توی جاواسکریپت Obfuscation چیه و چیکار میکنه؟	۲۶۶
۱۴۵	چه نیازی به Obfuscate کردن داریم؟	۲۶۷

صفحه	سوال	ردیف
۱۴۵	چیه؟ Minification	۲۶۸
۱۴۵	مزایای minification یا کم حجم سازی چیه؟	۲۶۹
۱۴۶	تفاوت های بین Encryption و Obfuscation چیه؟	۲۷۰
۱۴۶	ابزارهای مختلف برای minification کدوما هستن؟	۲۷۱
۱۴۶	چطوری اعتبارسنجی فرم رو با javascript انجام میدی؟	۲۷۲
۱۴۷	چطوری اعتبارسنجی فرم رو بدون javascript انجام میدی؟	۲۷۳
۱۴۷	متدهای موجود روی DOM برای اعتبارسنجی کدوما هستن؟	۲۷۴
۱۴۸	مقادیر موجود روی DOM برای اعتبارسنجی کدوما هستن؟	۲۷۵
۱۴۸	مقادیر موجود روی input برای اعتبارسنجی کدوما هستن؟	۲۷۶
۱۴۹	یه مثال از استفاده ویژگی rangeOverflow میتونی بزنی؟	۲۷۷
۱۴۹	جاواسکریپت قابلیت استفاده از enum رو پیش فرض توی خودش داره؟	۲۷۸
۱۵۰	چیه؟ enum	۲۷۹
۱۵۰	چطوری همه property هایی به object رو به دست بیاریم؟	۲۸۰
۱۵۰	چطوری property-descriptor آجکت رو بدست بیاریم؟	۲۸۱
۱۵۱	گزینه هایی که موقع تعریف ویژگی object descriptor با داریم کدوما هستن؟	۲۸۲
۱۵۱	چطوری کلاس ها رو extend می کنی؟	۲۸۳
۱۵۲	چطوری آدرس صفحه رو بدون رفرش صفحه عوض کنیم؟	۲۸۴
۱۵۲	چطوری بررسی می کنی که یه آرایه یه مقدار مشخص رو داره یا نه؟	۲۸۵
۱۵۳	چطوری آرایه های scalar رو با هم مقایسه می کنی؟	۲۸۶
۱۵۳	چطوری میشه پارامترهای صفحه رو از متدهای GET گرفت؟	۲۸۷
۱۵۳	چطوری اعداد رو میشه سه رقم سه رقم جدا کرد؟	۲۸۸

صفحه	سوال	ردیف
۱۵۴	تفاوت بین javascript و java چیه؟	۲۸۹
۱۵۴	آیا جاواسکریپت namespace را پشتیبانی میکنه؟	۲۹۰
۱۵۵	چطوری namespace تعریف میکنی؟	۲۹۱
۱۵۶	چطوری میتونیم تکه کد جاواسکریپت داخل یه iframe رو از صفحه والد صدا بزنیم؟	۲۹۲
۱۵۷	چطوری میشه اختلاف timezone رو از آبجکت date بگیریم؟	۲۹۳
۱۵۷	چطوری فایل‌های CSS و JS رو به شکل داینامیک بارگذاری کنیم؟	۲۹۴
۱۵۸	روش‌های مختلف برای پیدا کردن element‌ها توی DOM کدوما هستن؟	۲۹۵
۱۵۸	جیوهjQuery؟	۲۹۶
۱۵۸	موتور V8 جاواسکریپت چیه؟	۲۹۷
۱۵۹	چرا ما جاواسکریپت رو به عنوان یه زبان داینامیک میشناسیم؟	۲۹۸
۱۵۹	عملگر void چیکار میکنه؟	۲۹۹
۱۵۹	چطوری میشه نمایشگر موس صفحه رو به درحال لود تغییر داد؟	۳۰۰
۱۶۰	چطوری میشه یه حلقه بی‌نهایت درست کرد؟	۳۰۱
۱۶۰	چرا باید در استفاده از عبارت with تجدیدنظر کرد؟	۳۰۲
۱۶۱	خروجی این حلقه‌ها چی می‌شه؟	۳۰۳
۱۶۱	میتونی یه سری از ویژگی‌های ES6 رو اسم ببری؟	۳۰۴
۱۶۲	آیا میتونیم متغیرهای تعریف شده با let و const رو مجددا declare کنیم؟	۳۰۵
۱۶۳	آیا استفاده از const برای تعریف متغیر اونا رو immutable میکنه؟	۳۰۷
۱۶۳	آیا پیش‌فرض parameter های پیش‌فرض چی هستن؟	۳۰۸

ردیف	سوال	صفحه
۳۰۹	template-literal چی هستن؟	۱۶۴
۳۱۰	چطوری رشته‌های چند خطی رو توی template-literal می‌نویسیم؟	۱۶۵
۳۱۱	رشته‌های تودرتو چی هستن؟	۱۶۵
۳۱۲	tagged-template چی هستن؟	۱۶۵
۳۱۳	رشته‌های خام چی هستن؟	۱۶۶
۳۱۴	assign کردن با destructuring چیه و چطوری انجام می‌شه؟	۱۶۷
۳۱۵	موقع assign کردن با destructuring چطوری می‌شه مقدار اولیه تعريف کرد؟	۱۶۸
۳۱۶	destructuring می‌تونیم مقدار یه آرایه رو با استفاده از Enhanced-object-literal چیه؟	۱۶۸
۳۱۷	assignment تعویض کنیم؟	
۳۱۸	import دینامیک چی هستن؟	۱۶۹
۳۱۹	کاربرد import دینامیک چیه؟	۱۶۹
۳۲۰	آرایه‌های نوع دار (typed-arrays) چیه؟	۱۷۰
۳۲۱	مزایای لودر ماژول‌ها چیه؟	۱۷۱
۳۲۲	collation چیه؟	۱۷۱
۳۲۳	عبارت for...of چیه؟	۱۷۱
۳۲۴	spread روی آرایه زیر چیه؟	۱۷۲
۳۲۵	آیا PostMessage امنه؟	۱۷۲
۳۲۶	wildcard روی origin با postmessage مشکلات استفاده از چیه؟	۱۷۲
۳۲۷	چطوری از دریافت postMessage های ناخواسته و نامن از طرف هکرها جلوگیری کنیم؟	۱۷۳
۳۲۸	می‌توانیم کلا postMessage ها رو غیرفعال کنیم؟	۱۷۳

صفحه	سوال	ردیف
۱۷۳	آیا <code>postMessage</code> و همزمان کار می‌کنند؟	۳۲۹
۱۷۴	پارادیم زبان جاواسکریپت چیه؟	۳۳۰
۱۷۴	تفاوت‌های بین جاواسکریپت داخلی و خارجی چیه؟	۳۳۱
۱۷۴	آیا جاواسکریپت سریعتر از اسکریپتهاست سمت سرور است؟	۳۳۲
۱۷۴	چطوری وضعیت چک بودن یه <code>checkbox</code> رو بدست بیاریم؟	۳۳۳
۱۷۵	هدف از عملگر <code>double-tilde</code> چیه؟	۳۳۴
۱۷۵	چطوری یه کاراکتر رو به کد ASCII تبدیل کنیم؟	۳۳۵
۱۷۵	چیه؟ <code>ArrayBuffer</code>	۳۳۶
۱۷۶	خروجی کد زیر چی خواهد بود؟	۳۳۷
۱۷۶	هدف از <code>Error-object</code> چیه؟	۳۳۸
۱۷۷	هدف از <code>EvalError-object</code> چیه؟	۳۳۹
۱۷۷	خطاهایی که در حالت strict-mode رخ میدن ولی در غیر اون وجود ندارن کداما هستن؟	۳۴۰
۱۷۸	آیا همه <code>object</code> دارای <code>prototype</code> هستن؟	۳۴۱
۱۷۸	تفاوت‌های بین <code>argument</code> و <code>parameter</code> چیه؟	۳۴۲
۱۷۸	هدف از متدهای <code>some</code> روی آرایه‌ها چیه؟	۳۴۳
۱۷۹	چطوری دو یا تعداد بیشتری از آرایه‌ها رو با هم ترکیب کنیم؟	۳۴۴
۱۷۹	تفاوت‌های بین <code>Shallow</code> و <code>Deep</code> کپی چیه؟	۳۴۵
۱۸۰	چطوری می‌توانیم به یه تعداد مشخص از یه رشته کپی کنیم؟	۳۴۶
۱۸۱	چطوری همه <code>string</code> های <code>match</code> شده با یه <code>regular-expression</code> رو برگردانیم؟	۳۴۷
۱۸۱	چطوری یه رشته رو از اول یا از آخر <code>trim</code> کنیم؟	۳۴۸

صفحه	سوال	ردیف
۱۸۱	خروجی کنسول زیر با عملگر unary چی می‌شه؟	۳۴۹
۱۸۲	آیا جاواسکریپت از mixin‌ها استفاده می‌کنه؟	۳۵۰
۱۸۲	تابع thunk چیه و چیکار می‌کنه؟	۳۵۱
۱۸۲	چیکار می‌کنن؟ asynchronous چیکار thunk	۳۵۲
۱۸۳	خروجی فراخوانی‌های توابع زیر چی می‌شه؟	۳۵۳
۱۸۴	چطوری همه خطوط جدید رو از یه رشته حذف کرد؟	۳۵۴
۱۸۴	تفاوت بین repaint و reflow چیه؟	۳۵۵
۱۸۴	اگه قبل از یه آرایه عملگر نفی «!» بزاریم چی می‌شه؟	۳۵۶
۱۸۵	اگه دو تا آرایه رو با هم جمع ببنديم چی می‌شه؟	۳۵۷
۱۸۵	اگه عملگر جمع «+» روی قبل از مقادير falsy قرار بدیم چی می‌شه؟	۳۵۸
۱۸۵	چطوری با استفاده از آرایه‌ها و عملگرهای منطقی می‌تونیم رشته self رو تولید کنیم؟	۳۵۹
۱۸۶	چطوری می‌تونیم مقادير falsy رو از آرایه حذف کنیم؟	۳۶۰
۱۸۶	چطوری مقادير تکراری رو از یه آرایه حذف کنیم؟	۳۶۱
۱۸۷	همزمان با alias چطوری کار می‌کنن؟ destructuring	۳۶۲
۱۸۷	چطوری آیتم‌های یه آرایه رو بدون استفاده از متدهای map پیمایش کنیم؟	۳۶۳
۱۸۷	چطوری یه آرایه رو خالی کنیم؟	۳۶۴
۱۸۸	چطوری اعداد رو با تعداد رقم اعشار مشخص رند می‌کنی؟	۳۶۵
۱۸۸	ساده‌ترین روش برای تبدیل آرایه به object چیه؟	۳۶۶
۱۸۸	چطوری یه آرایه با یه سری داده درست کنیم؟	۳۶۷
۱۸۸	متغیرهای موجود روی آبجکت console کدوماً هستن؟	۳۶۸
۱۸۹	می‌شه پیام‌های کنسول رو استایلدهی کرد؟	۳۶۹

ردیف	سوال	صفحه
۳۷۰	هدف از متد dir روی آبجکت console چیه؟	۱۸۹
۳۷۱	آیا میشه المنتهای HTML رو توی console دیباگ کرد؟	۱۹۰
۳۷۲	چطوری میشه دادهها رو به شکل جدولی توی console نمایش بدم؟	۱۹۰
۳۷۳	چطوری میشه بررسی کرد که یه پارامتر Number هست یا نه؟	۱۹۱
۳۷۴	چطوری یه متن رو میتونیم به clipboard کپی کنیم؟	۱۹۱
۳۷۵	چطوری میشه timestamp رو بدست آورد؟	۱۹۱
۳۷۶	چطوری یه آرایه چندسطحی رو تک سطحی کنیم؟	۱۹۲
۳۷۷	ساده‌ترین روش برای بررسی چندشرطی چیه؟	۱۹۲
۳۷۸	چطوری کلیک روی دکمه برگشت مرورگر رو متوجه بشیم؟	۱۹۳
۳۷۹	چطوری میتونیم کلیک راست رو غیرفعال کنیم؟	۱۹۳
۳۸۰	چطوری object‌ها چی هستن؟	۱۹۳
۳۸۱	AJAX چیه؟	۱۹۴
۳۸۲	روش‌های مختلف مدیریت یه کد Asynchronous چیه؟	۱۹۴
۳۸۳	چطوری یه درخواست fetch رو کنسل کنیم؟	۱۹۴
۳۸۴	چیه؟ Speech-API	۱۹۵
۳۸۵	حداقل timeout توی throttling چقدره؟	۱۹۶
۳۸۶	چطوری میشه یه timeout صفر توی مرورگر اجرا کرد؟	۱۹۷
۳۸۷	چی هستن؟ event-loop	۱۹۷
۳۸۸	چی هستن؟ microtask	۱۹۸
۳۸۹	چی هستن؟ event-loop	۱۹۸
۳۹۰	هدف از queueMicrotask چیه؟	۱۹۸

ردیف	سوال	صفحه
۳۹۱	چطوری می‌شه از کتابخونه‌های جاواسکریپت توی فایل typescript استفاده کرد؟	۱۹۹
۳۹۲	تفاوت‌های بین Promise‌ها و observable کدوما هستن؟	۱۹۹
۳۹۳	چیه؟ heap	۲۰۰
۳۹۴	چیه؟ event-table	۲۰۰
۳۹۵	چیه؟ microTask	۲۰۱
۳۹۶	تفاوت بین shim و polyfill چیه؟	۲۰۱
۳۹۷	چطوری متوجه primitive یا غیر primitive بودن یه نوع داده میشیم؟	۲۰۲
۳۹۸	چیه؟ babel	۲۰۲
۳۹۹	آیا Node.js به شکل کامل تک thread کار میکنه؟	۲۰۲
۴۰۰	کاربردهای مرسوم observable کدوما هستن؟	۲۰۳
۴۰۱	چیه؟ RxJS	۲۰۳
۴۰۲	تفاوت بین function-declaration و Function-constructor چیه؟	۲۰۳
۴۰۳	شرط Short-circuit یا اتصال کوتاه چیه؟	۲۰۴
۴۰۴	ساده‌ترین روش برای تغییر سایز یه آرایه چیه؟	۲۰۴
۴۰۵	چیه؟ observable	۲۰۵
۴۰۶	تفاوت‌های بین توابع و کلاس‌ها چیه؟	۲۰۶
۴۰۷	تابع async چیه؟	۲۰۷
۴۰۸	چطوری خطاهای ایجاد شده هنگام استفاده از Promise‌ها رو کنترل کنیم؟	۲۰۷
۴۰۹	چیه؟ Deno	۲۰۹
۴۱۰	توی جاواسکریپت چطوری یه object قابل پیمایش درست کنیم؟	۲۰۹
۴۱۱	روش مناسب برای فراخوانی تابع بازگشتی چیه؟	۲۱۱

ردیف	سوال	صفحه
۴۱۲	چطوری بررسی کنیم که یه آبجکت Promise هست یا نه؟	۲۱۱
۴۱۳	چطوری متوجه بشیم که یا تابع با تابع constructor صدا زده شده یا نه؟	۲۱۲
۴۱۴	تفاوت‌های بین آبجکت rest و پارامتر argument چیه؟	۲۱۳
۴۱۵	تفاوت‌های بین عملگر spread و پارامتر rest چیه؟	۲۱۳
۴۱۶	نوع‌های مختلف generatorها کدوما هستن؟	۲۱۳
۴۱۷	تفاوت‌های بین iterable کدوما هستن؟	۲۱۵
۴۱۸	تفاوت‌های بین حلقه for...of و for...in چیه؟	۲۱۵
۴۱۹	چطوری property‌های instance و غیر instance تعریف می‌کنی؟	۲۱۶
۴۲۰	تفاوت‌های بین NaN و Number.isNaN کدوما هستن؟	۲۱۶

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 شماست، 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 می‌کنم و کمال تشکر رو ازشون دارم:

- جعفر رضائي
- مهسا مصباح

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

ماريوتك

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

مطلوب کتاب

مطلوب اين كتاب 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 كتاب اکثرا سوالاتی هستند که توی مصاحبه‌های استخدامی پرسیده می‌شن و مسلط بودن به اونا می‌تونه شانس موفقیت شما برای موقعیت‌های شغلی که مدنظر دارین افزایش بده. مطالب اين كتاب به دليل ترجمه بودن تا حد زیادي قابل دستکاري نبودن و سعی شده تا حد امكان حق گرددآورنده محفوظ باشه و با نسخه اصلی سورس که توسط Sudheer Jonna جمع‌آوري شده تفاوت معناني نداشته باشه. بخشی از مطالب كتاب 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 هميشه شاد و خندان و خوشحال باشين. مخلصيم

۱. روش‌های ایجاد object‌ها تا جوا اسکریپت کدوما هستن؟

۱. سازنده آبجکت: ساده‌ترین راه برای ایجاد یه آبجکت خالی استفاده از کلاس Object. در حال حاضر این روش توصیه نمی‌شود.

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

۲. متده استاتیک Object.create: متده استاتیک create روی Object با انتقال آبجکت به عنوان پارامتر، یه آبجکت جدید ایجاد می‌کند.

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

۳. استفاده از syntax آبجکت: با مقداردهی یه متغیر توسط یه آبجکت ساده داخل syntax آبجکت.

```
const object = {  
    name: "Ali Karimi",  
    age: 30,  
};
```

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

```
function Person(name) {  
    const object = {};  
    object.name = name;  
    object.age = 30;  
  
    return object;  
}  
  
const object = new Person("Ali Karimi");
```

۵. تابع به همراه prototype شیوه سازنده تابع هستش اما از برای متدها و خصوصیاتشون استفاده می‌کنیم.

```
function Person() {}  
Person.prototype.name = "Ali Karimi";  
  
const object = new Person();
```

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

```
function func() {}

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

(ب)

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

// Call the function
const 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);
```

۶. استفاده از کلاس‌های **ES6**: توی ES6 کلمه کلیدی class رو برای ایجاد آبجکت‌ها معرفی کردن.

```
class Person {
  constructor(name) {
    this.name = name;
  }
}

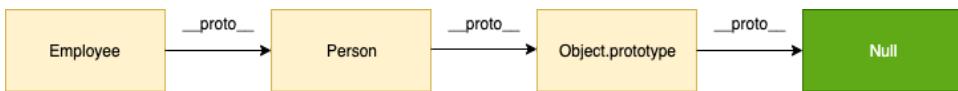
const object = new Person("Ali Karimi");
```

۷. الگو یا پترن **Singleton**: آبجکت‌ایه که فقط یه بار قابل نمونه‌گیری هستش و فراخوانی‌های بعدی روی سازندهش همون نمونه اولی رو برمی‌گردونه و اینطوری می‌شه مطمئن شد که به طور تصادفی نمونه‌های مختلف ایجاد نمی‌شه.

```
const object = new (function () {
  this.name = "Ali Karimi";
})();
```

۲. زنجیره prototype چیه؟

زنجیره **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 ویژگی **Object.getPrototypeOf(object)** یا ****proto**** در دسترسه در حالی که `prototype` توی عملکرد سازنده‌ها از طریق `object.prototype` در دسترسه.



۳. تفاوت‌های بین bind، apply و call چیا هستن؟

متد **call**: متد `call` یه تابع با یه مقدار `this` و آرگومان‌های ارائه شده رو دونه دونه فراخوانی میکنه.

```
const employee1 = {  
  firstName: "John",  
  lastName: "Rodson",  
};  
const 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?
```

متد **apply**: تابع رو فراخوانی میکنه و بهمون اجازه میده تا آرگومان‌ها رو به عنوان یه آرایه منتقل کنیم.

```

const employee1 = { firstName: "John", lastName: "Rodson" };
const 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?

```

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

```

const employee1 = { firstName: "John", lastName: "Rodson" };
const employee2 = { firstName: "Jimmy", lastName: "Baily" };

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

const inviteEmployee1 = invite.bind(employee1);
const 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?

```

متدهای call و apply تقریباً قابل تعویض هستن. هر دو بلافارسله تابع فعلی رو اجرا میکنن . شم ا باید تصمیم بگیرین که ارسال آرایه د رآرایه آسونتره یا فهرستی از آرگومانهای جدا شده با کاما. باید یادمون باشه که متدهای call برای کاما (فهرست جدا شده) و apply برای حالت آرایهایه. در حالی که bind یه تابع جدید ایجاد میکنه و میتونه با دریافت «this» روی اولین پارامتر ارسال شده کانتکست آبجکت جدید ساخته رو تنظیم کنه.

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

JSON یه قالب داده مبتنی بر متنه که از syntax آبجکت جاواسکریپت (Javascript) پیروی میکنه و توسط Douglas Crockford رایج شد. کاربردش زمانیه که بخواهیم داده‌ها رو از طریق شبکه انتقال بدیم و اساساً یه فایل متنی با پسوند json و نوع MIME از application/json داشته باشیم. اکثراً دو عملیات زیر روی JSON انجام میشه:

تجزیه (Parsing): تبدیل یه رشته به یه آبجکت جاواسکریپت (native Object).

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

رشته‌سازی (stringify): تبدیل یه آبجکت جاواسکریپت به یه رشته تا بتونه از طریق شبکه منتقل بشه یا یه جایی ذخیره بشه.

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

۵. هدف از متده slice روی آرایه‌ها چیه؟

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

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

نکته: متده slice آرایه اصلی رو تغییر نمیده ولی یه زیرمجموعه به عنوان یه آرایه جدید برمی‌گردونه.

۶. هدف از متدهای slice و splice چیه؟

متدهای **splice** برای اضافه کردن یا حذف از اون استفاده می‌شوند و مورد یا موارد حذف شده را برمی‌گردونند. آرگومان اول موقعیت آرایه را برای درج یا حذف مشخص می‌کنند در حالی که آرگومان اختیاری دوم تعداد عناصر برای حذف را مشخص می‌کنند و مابقی هر آرگومان اضافه‌ای که به این متدها ارسال بشوند، به آرایه اضافه می‌شوند. چند نمونه مثال در مورد نحوه کارکردش ببینید:

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

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

نکته: متدهای slice و splice اصلی را اصلاح می‌کنند یعنی متغیر اصلی را تحت تاثیر قرار میدهند و آرایه حذف شده را برمی‌گردند.

۷. تفاوت متدهای slice و splice چیا هستن؟

splice	slice
آرایه اصلی را تغییر میدهد (mutable) یا تغییرپذیر	آرایه اصلی را تغییر نمیدهد (immutable) یا تغییرنپذیر
عناصر حذف شده را به عنوان آرایه برمی‌گردند	زیر مجموعه آرایه اصلی را برمی‌گردند
برای درج عناصر به آرایه یا حذف از اون استفاده می‌شوند	برای انتخاب عناصر از آرایه استفاده می‌شوند

۸. تفاوت‌های Map و Object چیا هستن؟

آبجکت‌ها شبیه به **Map** هستن (البته خود **Map** هم آبجکته و اینجا منظور شاعر خود شخص **Object** را میگه) از این جهت که هردو بهمن این امکان رو میدن که مقادیر رو روی کلیدهای مشخصی تعریف کنیم، مقادیر رو بازیابی کنیم، کلیدها رو حذف کنیم و بینیم چیزی توی یه کلید ذخیره شده یا نه. به همین دلیل در طول تاریخ از آبجکت‌ها به عنوان **HashMap** و **Map** استفاده‌های زیادی شده. اما تفاوت‌های مهمی وجود داره که استفاده از **Map** رو توی موارد خاص ارجعیت میده.

۱. کلیدهای یه آبجکت رشته‌ها و **Symbol**‌ها هستن، در حالی که برای **Map** مقادیر مختلفی میتونه وجود داشته باشه که شامل توابع، آبجکت‌ها و هر نوع اولیه دیگه‌ای میشه.

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

۳. اندازه **Map** رو میتونیم به راحتی با ویژگی سایز بدست بیاریم، در حالی که تعداد خصوصیات یه آبجکت باید به صورت دستی و با ساخت یه آرایه از روی کلیدها و یا مقادیرش حساب بشه.

۴. **Map** قابل تکراره و میتونه مستقیما تکرار بشه، در حالی که تکرار روی یه آبجکت مستلزم بدست آوردن کلیدهای اون به روشنی خاص و تکرار روی او نهاست.

۵. آبجکت یه **prototype** داره، بنابراین کلیدهای پیش‌فرض توی **Object** وجود داره که که اگه دقت نکنیم ممکنه با کلیدهای موجود برخورد کنه. از زمان ES5 میتوانیم با استفاده از (**Map** = **Object.create(null)**)، این قضیه رو دور بزنیم ولی بهندرت این کار انجام میشه.

۶. **Map** 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 (==!) و 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 میدن. 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 دارد:

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

۲. صفرهای مثبت و منفی با هم برابرند.

۳. اگه هر دو درست یا نادرست باشند، دو عملوند بولین کاملاً برابر هستند.

۴. اگه دو شیء به یه شیء اشاره کنند کاملاً برابر هستند.

۵. انواع Null و Undefined با === برابر نیستند، بلکه با == برابر هستند. یعنی

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

یه چندتا مثال که موارد بالا رو پوشش میدن:

```
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 هستند؟

توابع arrow function ها به صورت ساده‌تر و کوتاه‌تر تعریف می‌شون و شیء this ، متغیر جادویی new.target یا super ، متدهای argumants سازنده یا constructor استفاده نمی‌شون.

۱۱. یه تابع first-class چجور تابعی‌هی؟

توبی جاواسکریپت، توابع آبجکت‌های کلاس اول یا first class هستند. توابع کلاس اول توبی هر زبان برنامه‌نویسی، زمانی معنی میدن که توابع توبی اون زبان باهاشون مثل بقیه متغیرها رفتار بشه.

برای مثال، توبی جاواسکریپت، یه تابع می‌تونه به عنوان آرگومان به یه تابع دیگه پاس 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 توبی کد زیر، تابع handler به عنوان callback به یه تابع listener پاس داده شده.

```
const handler = () => console.log("This is a click handler function");

/** 
 * Usage of `handler` method
 */
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. مثل:

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

۱۳. یه تابع higher-order چجور تابعیه؟

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

```
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. مثل:

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

۱۵. توابع یعنی چی؟ currying

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. Curry از نام یه ریاضی‌دان به اسم Haskell Curry گرفته شده. با استفاده از Curry در واقع یه تابع رو به یه تابع unary تبدیل می‌کنیم. بیاین یه مثال از یه تابع با چندین آرگومان و تبدیلش به تابع currying بزنیم.

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

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

۱۶. چه توابعی pure هستن؟

تابع pure 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 pure و توابع غیرpure بزنیم.

```
// Impure
const 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 که مستقل از مقدار پارامتر هستش، یه تابع ناخالص یا Impure به حساب می‌آید. در حالی

که از یه طرف متده concat آرایه رو میگیره و اونو با یه آرایه دیگه ترکیب میکنه و به آرایه کاملاً جدید و بدون هیچ side-effect تولید میکنه. همچنین مقدار برگشتی با آرایه قبلی ترکیب شده هستش.

یادتون باشه که توابع خالص یا pure مهمان چون اونا unite-test رو بدون هیچ-side effect و بدون نیاز به dependency-injection ساده میکنن. اونا همچنین از اتصال محکم بین بخش‌های مختلف برنامه جلوگیری میکنن و با نداشتن side-effect، احتمال برو زخطات و برنامه ر و کمکت ر میکنن . این اصول کتا ره م جم جم میشن و کنار نکته: دقت کنیم که چاپ یه مقدار روی کنسول و یا درخواست api-call هم محسوب میشه.

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، میشن و فقط دستورها و عبارت‌های توی همون اسکوپ بهش دسترسی دارن. درحالی که متغیرهای تعریف شده با کلمه کلیدی `var` برای تعریف یه متغیر توی سطح `global` و یا به شکل محلی و برای استفاده در کل تابع بدون در نظر گرفتن اسکوپی که تو ش تعریف شده، استفاده میشه. بیاین برای نشون دادن کاربردش یه مثال بزنیم.

```
const counter = 30;
if (counter === 30) {
  const 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؟

: var

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

:let

۱. به عنوان بخشی از ES6 معرفی شده
۲. محدود به scope یا دامنه هستش
۳. شده ولی قابل استفاده نیست Hoist

بیاین با یه مثال تفاوتش رو بهتر بینیم:

```
function userDetails(username) {
  if (username) {
    console.log(salary); // undefined(due to hoisting)
    console.log(age); // error: age is not defined
  (ReferenceError)
    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؟

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 تعریف کنیم؟

اگه بخوایم متغیرها رو مجددا توی یه بلوک `switch` تعریف کنیم، این کار باعث خطا می‌شه چون در واقع فقط یه بلوک 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 خطا، می‌توانیم یه بلوک تودرتو داخل case ایجاد کنیم و یه محیط واژگانی دارای محدوده بلوک جدید ایجاد کنیم.

```

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

```

۲۱. چیه؟ Temporal-Dead-Zone

رفتاری توی جاواسکریپت که موقع تعریف متغیر با کلمات کلیدی let و const رخ میده، نه با کلمه کلیدی var. توی اکماسکریپت 6، دستیابی به متغیر let و قبل از تعریفش (توی scope خودش) باعث خطای refrence 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 بلا فاصله صدا زده شده) چی هستن؟ IIFE

IIFE (فراخوانی عملکرد بلا فاصله) یه تابع جاوا اسکریپتیه که به محض تعریف اجرا میشه. تعریف اون به این صورته:

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

دلیل اصلی استفاده از IIFE بدبست آوردن حریم خصوصی داده هاست، 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 خط را رو میگیریم:

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

۲۳. مزایای استفاده از module‌ها چیه؟

استفاده از مازول‌ها مزایای زیادی داره، مثلا:

۱. قابلیت نگهداری بالایی دارن
۲. قابلیت استفاده مجدد دارن
۳. نامگذاری بخش‌های مختلف کد رو راحت‌تر میکنن

۲۴. Memoization چیه؟

Memoization یه روش برنامه‌نویسیه که سعی میکنه با ذخیره نتایج قبلی یه تابع عملکرد (پروفورمنس performance) اون تابع رو افزایش بده. هر بار که یه تابع Memoize شده فراخوانی میشه، پارامترهای اون به همراه نتیجه بدست او مده cache میشه (یعنی توی حافظه ذخیره میشه). موقع اجرای بعدی قبل از انجام محاسبه بررسی میشه که داده

قبل پردازش شده یا نه و اگه داده وجود داشته باشه، بدون اجرای کل تابع و از روی مقدار cache شده نتیجه برگشت داده میشه، در غیر این صورت تابع به شکل عادی اجرا میشه و بعدش نتیجه بدست اومده توی حافظه ذخیره میشه.

بیاین یه مثال از نوشتمن یه تابع با Memoization بزنیم،

```
const memoizAddition = () => {
  const 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.
    }
    console.log("Calculating result");
    const 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
```

چیه؟ Hoisting .۲۵

یه مکانیسم جاواسکریپت هه که متغیرها و تعاریف توابع رو به بالای scope یا دامنه خودشون انتقال میده. یادمون باشه که جاواسکریپت فقط تعریف متغیرها و توابع رو Hoist میکنه، نه مقدار دهی اولیه اونا رو.

بیاین یه مثال ساده از hoist کردن متغیرها بیینیم:

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

ترجمه کد بالا اینطوری میشه:

```
let message;
console.log(message);
message = "The variable Has been hoisted";
```

۲۶. Class‌ها توی ES6 چیکار می‌کنن؟

در ES6، کلاس‌های جاوااسکریپت اصطلاحاً فقط یه sugar-syntax روی وراثت مبتنی بر جاوااسکریپت هستن. prototype برای مثال، وراثت مبتنی بر prototype که به شکل 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`;
};
```

حالا همین کد با کلاس‌های ES6 و به شکل ساده‌تری به صورت زیر قابل نوشتن‌هه:

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

  getDetails() {
    return `${this.model} bike has ${this.color} color`;
  }
}
```

۲۷. Closure‌ها چیا هستن؟

کلاژور ترکیبی از یه تابع و یه scope هستش که تابع توی اون تعریف شده. برای مثال این یه تابع داخلی‌هه که به متغیرهای تابع خارجی دسترسی داره.

کلاژور دارای سه زنجیره scope هستش:
۱. scope داخلی و جایی که متغیرهای محلی بین برآکت‌های اون تعریف شده باشن

۲. متغیرهای تابع بیرونی
۳. متغیرهای عمومی و general
- بیانیه مثال راجع به مفهوم کلازور بزنیم

```
function welcome(name) {  
  const greetingInfo = function (message) {  
    console.log(` ${message} ${name}`);  
  };  
  return greetingInfo;  
}  
  
const 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 scope تابع خارجی (welcome) دسترسی دارد.

۲۸. **Module ها چیا هستن؟**

ماژول‌ها به واحدهای کوچکی از کد مستقل و قابل استفاده مجدد گفته می‌شون و به عنوان پایه بسیاری از دیزاین پترن‌های جاواسکریپت عمل می‌کنند. خروجی ماژول‌های جاواسکریپت یه شی، یه تابع یا constructor هستند.

۲۹. **چرا به module ها نیاز داریم؟**

دلایل زیادی برای استفاده کردن از ماژول‌ها وجود داره که یه تعداد ازاونا رو این زیر می‌اریم:

۱. قابلیت نگهداری
۲. قابلیت استفاده مجدد
۳. نام‌گذاری
۴. جدا بودن بخش‌های مختلف برنامه

۳۰. توی جاواسکریپت scope چیه و چیکار میکنه؟

scope یا محدوده، به نحوه دسترسی متغیرها، توابع و اشیاء توی بخش‌های مختلف کدمون در زمان اجرا گفته میشه. به عبارت دیگه، دامنه قابلیت دیده شدن متغیرها و بقیه منابع رو تو قسمت‌هایی از کدمون تعیین میکنه.

۳۱. service-worker چیه؟

اساسا یه اسکریپت هستش که جدا از یه صفحه وب توی پس‌زمینه اجرا می‌شه و ویژگی‌هایی رو فراهم میکنه که نیازی به صفحه وب یا تعامل کاربر نداره. بعضی از ویژگی‌های عمدۀ service worker عبارتند از: تجربه کار با برنامه بدون نیاز به اینترنت (آفلاین وب)، به روزرسانی داده‌ها به شکل متناسب در پس‌زمینه، push notification، رهگیری و رسیدگی به درخواست‌های شبکه و مدیریت درخواست‌های cache شده.

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

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

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

مشکلی که توی service worker وجود داره اینه که در صورت عدم استفاده برای یه مدت، قطع میشه و در صورت نیاز بعدی دوباره راهاندازی می‌شه، به همین دلیل نمی‌تونیم به service worker سراسری توی `onmessage` و `onfetch` event handlers اعتماد کنیم. تو این حالت service worker برای تداوم کار و استفاده مجدد موقع شروع مجدد، به indexedDB دسترسی خواهد داشت و میشه از اون استفاده کرد.

۳۴.IndexedDB چیه؟

IndexedDB یه API سطح پایین برای ذخیره client-side یا سمت کاربر و برای مقدار زیادی از داده‌های ساخت یافته (structured) شامل فایل‌ها و blob‌ها هستش. این API از index‌ها برای فراهم‌سازی جستجو با کارایی بالا توی این داده‌ها استفاده میکنه.

۳۵. Web-storage چیه؟

web storage یه API هستش که مکانیزمی رو فراهم میکنه که مرورگرها می‌تونن یه مقدار رو به همراه یه کلید و بصورت محلی توی مرورگر کاربر ذخیره کنن، که روشی بسیار مدرن‌تر و عملی‌تر نسبت به کوکی‌ها محسوب میشه. فضای ذخیره‌سازی وب دو مکانیزم برای ذخیره اطلاعات روی client فراهم میکنه.

۱. **Local storage**: داده‌ها رو برای مسیر (origin) فعلی و بدون تاریخ انقضا ذخیره میکنه.

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

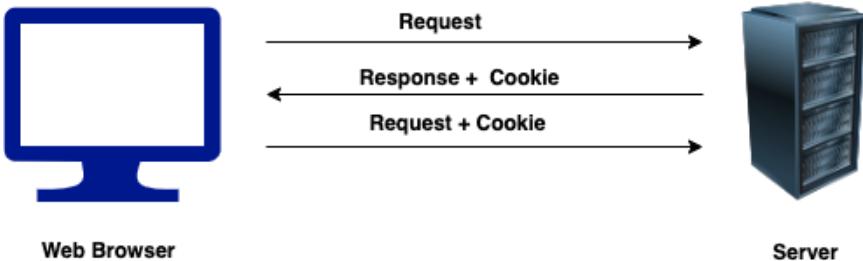
۳۶. Post-message چیه؟

روشی هست که امکان ایجاد ارتباط متقابل بین آجکت‌های window رو فراهم میکنه (برای مثال، بین یه صفحه و یه پنجره بازشو که باعث ایجاد اون شده، یا بین یه صفحه و یه iframe قرارداده شده توی اون) به طور کلی اسکریپت‌های موجود در صفحات مختلف اگه که صفحاتشون از خط و مشی یکسانی تبعیت کنن مجاز به دسترسی به همدیگه هستن. (یعنی صفحات از پروتکل، شماره پورت و host یکسانی برخوردار باشن).

۳۷. Cookie چیه؟

کوکی قطعه‌ای از داده هستش که توی کامپیوترمون ذخیره می‌شه تا مرورگر به اون دسترسی داشته باشه. کوکی‌ها به عنوان جفت‌های کلید و مقدار ذخیره میشن. برای مثال می‌تونیم یه کوکی با عنوان 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 می‌شود. 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 می‌شود اما با تنظیم تاریخ انقضا (به وقت UTC) می‌توانیم این رفتار را 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 دارد. اما با استفاده از پارامتر path می‌توانیم به مرورگر بگیم که کوکی متعلق به چه مسیریه.

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

۴۰. 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.

۱۴. تفاوت‌های بین **session-storage** و **local-storage** و **cookie** چیا هستن؟

Session storage	Local storage	Cookie	ویژگی
client فقط	client فقط	client و server هردو	قابل دسترسی از طرف client یا server
تا وقتی که تب پسته نشده	تا وقتی که حذف بشه	پیکربندی شده با استفاده از گزینه اکسپایر	طول عمر
پشتیبانی نمی‌شه	پشتیبانی نمی‌شه	پشتیبانی می‌شه	پشتیبانی از SSL
۵MB	5MB	4KB	حداکثر اندازه داده

۴۲. تفاوت‌های بین sessionStorage و localStorage چیا هستن؟

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

۴۳. چطوری به web-storage دسترسی پیدا می‌کنی؟

آجکت window ویژگی‌های WindowSessionStorage و WindowLocalStorage را که دارای ویژگی‌های sessionStorage و localStorage هستن رو پشتیبانی می‌کنه. این خصوصیات نمونه‌ای از شئ Storage رو ایجاد می‌کنه که از طریق اون می‌شه موارد داده رو برای یه دامنه خاص و نوع ذخیره سازی (session یا محلی) تنظیم، بازیابی و حذف کرد.

برای مثال، می‌تونیم 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 میده. مثلا:

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

// Get saved data from sessionStorage
const data = sessionStorage.getItem("key");

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

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

۴۵. رخداد storage چیه و چطوری ازش استفاده می‌کنیم؟

رویدادی هستش که با همزمان با تغییر یه محل ذخیره‌سازی تو یه context صفحه دیگه‌ای فراخوانی می‌شه. این امکان بهمون قابلیت پردازش تغییرات مقادیر ذخیره‌سازی شده توسط یه EventHandler رو میده. ساختار کد اون یه چیزی مث کد زیر میشه:

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

برای مثال استفاده از رویداد onstorage رو ببینیم که کلید ذخیره و مقادیر اونو 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؟

با استفاده از این قابلیت میشه گفت که فضای ذخیره‌سازی وب از امنیت بیشتری برخورداره و مقدار زیادی داده می‌تونن به صورت محلی ذخیره بشن، بدون اینکه روی عملکرد وب سایت تأثیر بذارن. همچنین، اطلاعات هرگز به سرور منتقل نمیشون و به همین دلیل این روش نسبت به کوکی‌ها بیشتر توصیه می‌شه.

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

قبل از استفاده از فضای ذخیره‌سازی وب، می‌تونیم پشتیبانی مرورگر رو برای sessionStorage و localStorage کنیم. البته مرورگرهای مدرن امروزی کاملا

پشتیبانی web-storage را ارائه میدن.

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

.۱۴۸ چطوری می‌توnim پشتیبانی از web-worker توسط مرورگر را بررسی کنیم؟

می‌توnim برای داشتن یه کد cross-functional قبل از استفاده از وب ورکرهای، پشتیبانی مرورگر را بررسی کنیم، انجام این کار هم ساده اس و با یه شرط ساده محقق میشه:

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

.۱۴۹ یه مثال از web-workerها می‌توانی بزنی؟

یه مثال ساده برای شروع استفاده از web worker میتونه مثال شمارنده باشه که برای اجرای اون باید مراحل زیر رو دنبال کنیم:

۱. ساخت یه فایل **Web Worker**: برای افزایش مقدار شمارشی، باید یه اسکریپت بنویسیم که این کار رو انجام میده. بیاین اسمشو counter.js بذاریم

```
let i = 0;  
  
function timedCount() {  
    i += 1;  
    postMessage(i);  
    setTimeout("timedCount()", 500);  
}  
  
timedCount();
```

اینجا از روش postMessage برای ارسال پیام به صفحه HTML استفاده می‌شه.

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

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

روی همین شئ ما می‌توانیم پیام‌ها رو از web worker دریافت کنیم:

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

۳. پایان دادن به web Worker: می‌دونیم که Web Worker‌ها تا زمان خاتمه یافتن پیام‌ها (حتی بعد از اتمام یه اسکریپت خارجی) به گوش دادن ادامه میدن. برای قطع کردن گوش دادن به پیام‌ها می‌توانیم از دستور terminate استفاده کنیم.

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

۴. استفاده مجدد از web worker: اگه متغیر worker رو undefined مقداردهی کنیم، می‌توانیم از کد نوشته شده مجدد استفاده کنیم.

```
w = undefined;
```

۵۰. محدودیت‌های web-worker روی DOM چیا هستن؟

WebWorker‌ها به اشیا جاواسکریپت دسترسی ندارن چون توی یه فایل خارجی تعریف شدن. پس به این آبجکت‌ها دسترسی نداریم:

۱. آبجکت Window

۲. آبجکت Document

۳. آبجکت Parent

یه آبجکت‌ایه که یه ممکنه یه callback برای موفق بودن یا رو با یه موفق نبودن(مثلاً خطای شبکه) فرآیند تولید کنه. پس کلا حالت‌های ممکن برای یه Promise یکی از این سه حالت خواهد بود: انجام شده، رد شده، یا در انتظار. برای ساختن Promise‌ها از سینتکس زیر استفاده می‌شه

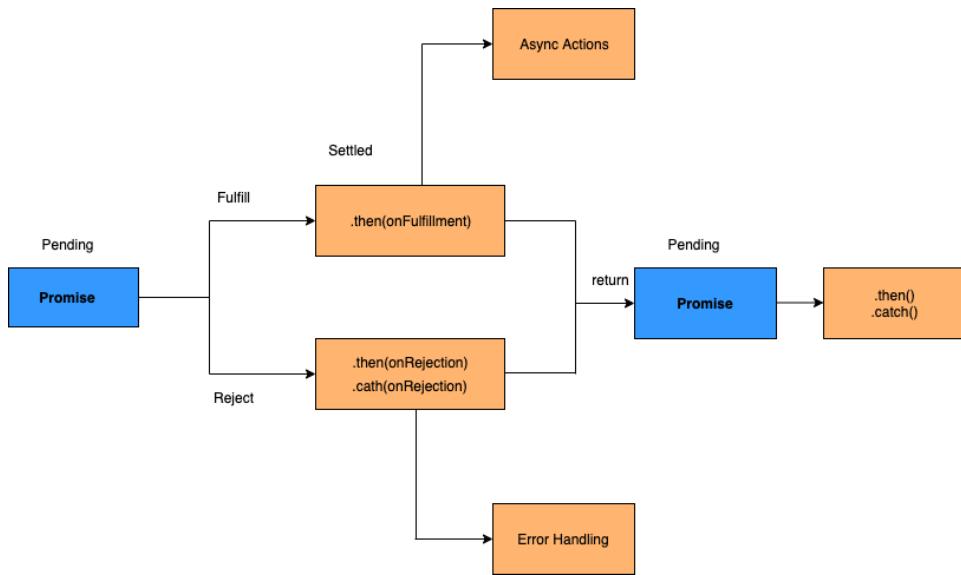
```
const promise = new Promise((resolve, reject) => {
  // promise description
});
```

استفاده از Promise‌ها رو هم ببینیم

```
const promise = new Promise(
  (resolve) => {
    setTimeout(() => {
      resolve("I'm a Promise!");
    }, 5000);
  },
  (reject) => {
    // if we want to reject, we can call `reject()` cb method
  }
);

/**
 * We can call promise and add some success callback
 functionality to run after resolve
 */
promise.then((value) => console.log(value)); // here after 5s we
will have a log in our console
```

چرخه فرآیند یه Promise به این شکل انجام می‌شه:



۵۲. چرا به Promise نیاز داریم؟

Promise‌ها برای رسیدگی به عملیات ناهمزمان (async) استفاده می‌شون. اونا با جلوگیری از وقوع callback hell و نوشتن کد clean و تمیزتر، یه رویکرد جایگزین برای callback‌ها ارائه می‌کنن.

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

Promise‌ها سه حالت دارن:

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 خطای داده خواهد شد. **Rejected**.

۵۴. توابع callback چی هستن؟

تابع callback تابعیه که به عنوان آرگومان به تابع دیگری منتقل می‌شه. این تابع در داخل تابع خارجی برای تکمیل یه عملیات فراخوانی می‌شه.
بیاین یه مثال ساده از نحوه استفاده از تابع callback ببینیم:

```
function callbackFunction(name) {  
    console.log(`Hello ${name}`);  
}  
  
function outerFunction(callback) {  
    const name = prompt("Please enter your name.");  
    callback(name);  
}  
  
outerFunction(callbackFunction);
```

۵۵. چرا به توابع callback نیاز داریم؟

چون جاواسکریپت یه زبان ایونت محوره به callback ها نیاز داریم. این به این معنیه که جاواسکریپت به جای منظر موندن برای جواب یه عملیات فراخوانی شده، در حین گوش دادن به ایونت‌های دیگه به اجرا شدن ادامه میده.

بیاین یه مثال از دو تابع که پشت سرهم اجرا میشن ولی یکیشون callback میزنه به API (شبیه‌سازی شده با setTimeout) و اون یکی یه لاغ کنسول ساده می‌ندازه رو ببینیم:

```

function firstFunction() {
  // Simulate a code delay
  setTimeout(() => {
    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 می‌شه فهمید، جاواسکریپت منتظر جواب تابع اول نمونده و بلوک کد باقی مونده رو اجرا کرده. بنابراین از callbackها می‌توانیم به شکلی استفاده کنیم که مطمئن شیم یه کد تا زمانی که اجرای کد دیگه تموم نشده، اجرا نمی‌شه.

۵۶.Callback hell یا جهنم توابعcallback چیه؟

Callback Hell اصطلاحاً یه آنتی‌پترن‌ه که با چندین callback تودرتو ایجاد می‌شه، همیشه هم خوندن کد و رفع خطاهرا رو سخت‌تر می‌کنه، مخصوصاً زمان‌هایی که با async کار می‌کنیم. یه جهنم callback با یه کد مثل کد پایین ایجاد می‌شه:

```

async1(() => {
  async2(() => {
    async3(() => {
      async4(() => {
        // ....
      });
    });
  });
});

```

۵۷. SSE یا همون Server-sent-events چیه؟

امکان رو میده که به روزرسانی‌های خودکار رو از طریق درخواست HTTP و بدون استفاده از polling (درخواست‌های پیوسته و منظم) دریافت کنه. این روش، یه کانال ارتباطی به طرفهست و event‌ها فقط از سرور به client ارسال میشن. این روزا از قابلیت‌های روی به روزرسانی‌های فیسبوک/تویتر، به روزرسانی قیمت سهام، فیدهای خبری و غیره استفاده میشه.

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

کلاس EventSource برای دریافت پیام‌های event ارسال شده از سرور استفاده می‌شه. برای مثال، می‌تونیم پیام‌هایی که می‌خواهیم رو از سرور مثل مثال زیر بگیرین.

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

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

می‌تونیم قبل از استفاده از SSE مانند زیر، پشتیبانی مرورگر رو با یه شرط شبیه کد زیر بررسی کنیم:

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

۶. کدوم توابع روی SSE وجود دارند؟

این پایین یه لیست از event های موجود برای SSE رو ذکر می کنیم:

توضیحات	ایونت
موقعی که اتصال به سرور باز می شده استفاده می شده	onopen
این event زمانی استفاده می شده که پیامی دریافت شده	onmessage
زمانی اتفاق می وفته که خطایی رخ بدید	onerror

۶.۱. اصلی ترین قوانین Promise ها چیا هستن؟

میشه گفت اصلی ترین قانون های Promise ها ایناست:

۱. آبجکت ایه که متده «then» رو برای منتظر پاسخ درخواست موندن ارائه میکنه.

۲. یه Promise معلق ممکنه به حالت تحقق یافته یا رد شده تبدیل بشه.

۳. Promise تمام شده یا رد شده حل و فصل می شده و نباید به حالت دیگری تبدیل شه.

۴. پس از اتمام Promise مقدار اون نباید تغییر کنه.

۶.۲. توی callback چطوری رخ میده؟

میتونیم یه پاسخ api-call رو داخل یه دیگر قرار بدین تا عملیاتها رو به شکل متوالی و یکی یکی انجام بدین. این به عنوان callbacks در callback شناخته می شده.

```

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

    loadScript('/script2.js', function(script) {
        console.log('second 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 ها چیه؟

فرآیند اجرای دنباله‌ای از عملیات‌های ناهمزمان (async) به طوریکه هر کدام بعد از تاموم شدن قبلی اجرا بشن با استفاده از Promise‌ها به عنوان Promise chaining شناخته می‌شوند. بیانی برای محاسبه نتیجه نهایی یه مقدار، یه مثال از زنجیره ها رو ببینیم:

```

new Promise((resolve, reject) => {
    setTimeout(() => resolve(1), 1000);
})
.then((result) => {
    console.log(result); // 1
    return result * 2;
})
.then((result) => {
    console.log(result); // 2
    return result * 3;
})
.then((result) => {
    console.log(result); // 6
    return result * 4;
});

```

توی بلوک‌های کد بالا، نتیجه هر Promise به زنجیره‌های then بعدی منتقل می‌شه، فرآیندشون به شکل زیر انجام می‌شه:
۱. اول توی ۱ ثانیه حل می‌شه،

۲. پس از اون با چاپ مقدار `1` توی کنسول، فراخوانی میشه و یه `Promise` با مقدار `result` ضربدر `2` برمیگردونه.
۳. پس از اون `result` به `callback` بعدی منتقل شده و با چاپ مقدار `2` و `3` برگرداندن یه `Promise` با `result` ضربدر `3`.
۴. در نهایت مقدار به آخرین `callback` رسیده و با چاپ مقدار `6` توی کنسول، یه `result` با `Promise` اول توی `1` ثانیه حل میشه.
۵. پس از اون «`then`» با ثبت نتیجه `(1)` فراخوانی میشه و سپس یه `Promise` با مقدار نتیجه `*2` برمیگردونه.
۶. پس از اون مقدار به بعدی منتقل شد. سپس با ثبت نتیجه `(2)` و برگرداندن یه `Promise` با نتیجه `*3`*
۷. در نهایت مقدار به آخرین . سپس با ثبت نتیجه `(6)` و یه `Promise` با نتیجه `*4` .`handler`.

۶۴. کاربرد متده `promise.all` چیه؟

متده `promise.all` یه آرایه‌ای از `Promise`ها رو به عنوان ورودی میگیره(یک تکرار) و زمانی `resolve` میشه که همه `Promise`ها `resolve` بشن، و اگه یکی از اونها `reject` شه میشه. برای مثال، تیکه کد زیر رو ببینیم:

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

```

نکته: ترتیب خروجی `Promise`ها طبق همون ترتیب آرایه ورودی ایجاد میشه.

۶۵. هدف از متده `race` روی `Promise` چیه؟

متده `promise.race` یه آرایه از `Promise`ها رو میگیره و نتیجه اولین ای `Promise` را `reject` رو برمیگردونه. بیاین مثالی از متده `race` رو در نظر بگیریم که تو اون `Promise2` اول `Promise` میشه:

```

const promise1 = new Promise((resolve, reject) => {
  setTimeout(resolve, 500, "one");
});
const promise2 = new Promise((resolve, reject) => {
  setTimeout(resolve, 100, "two");
});

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

```

۶۶. حالت توی جاواسکریپت چی کار میکنه؟

یکی از قابلیتهای ارائه شده توی ES5 هست که به ما این امکان رو میده که یه برنامه یا یه تابع رو تو یه حالت اجرایی "سختتر" قرار بدیم. به این ترتیب از انجام بعضی اقدامات جلوگیری میکنه و استثنای (Exception)های بیشتری رو ایجاد میکنه. عبارت تحت اللفظی "usestrict" به مرورگر دستور میده تا از کد جاواسکریپت در حالت Strict استفاده کنه.

۶۷. چرا به حالت strict نیاز داریم؟

حالت سخت گیرانه (strict mode) برای نوشتن جاواسکریپت "امن" و حصول اطمینان از اطلاع از "syntax" بد. برای جلوگیری از خطاهای واقعی استفاده میشه. برای مثال، ایجاد تصادفی یه متغیر گلوبال رو با ایجاد یه Exception حذف میکنه و یا یه خطا برای انتساب به یه ویژگی غیرقابل نوشتن، یه ویژگی فقط گیرند، یه ویژگی غیرم وجود د، یه متغیر غیرموجود یا یه ویژگی غیرقابل نوشتن ایجاد میکنه.

۶۸. چطوری میتوانیم حالت strict رو فعال کنیم؟

حالت سخت با اضافه کردن رشته `use strict` به ابتدای scope اعلام میشه. پس میتوانیم اونو به ابتدای یه اسکریپت یا یه تابع اضافه کنیم. اگه در ابتدای یه اسکریپت اعلام شه، دامنه عمومی داره:

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

و اگه در داخل یه تابع اعلام کنیم 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 نقطی دوتایی یا نفی (!!) اینه که تضمین میکنه که نوع حاصل از عملیات یه مقدار true یا false و تایپش بولینه. اگه بود (برای مثال 0، null، undefined و غیره)، میشه و در غیر این صورت، درسته و نتیجه true خواهد بود.

برای مثال، میتونیم نسخه IE را با استفاده از عبارت زیر آزمایش کنیم.

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

اگه از این عبارت استفاده نکنیم مقدار اصلی رو برمی‌گردونه.

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

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

نکته: استفاده از !! معادل استفاده از Boolean(var) هست ولی !! سریعتره.

۷۰. هدف از عملگر delete چیه؟

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

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

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

۷۱. عملگر `typeof` چیکار میکنه؟

برای بدست آوردن نوع متغیر جاوااسکریپت می‌توانیم از عملگر `typeof` استفاده کنیم و نوع یه متغیر یا یه عبارت رو به صورت یه رشته برمی‌گردونه.

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

۷۲. چیه و چه زمانی `undefined` می‌گیریم؟

ویژگی `undefined` می‌گیریم که به یه متغیر مقداری اختصاص داده نشده یا اصلاً تعریف نشده‌ست. هم نوع مقدار تعریف نشده هم تعریف نشده.

```
let 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 عمدی هر مقدار شی رو نشون میده. `null` یکی از مقادیر اولیه جاوااسکریپت‌ه و حواسمن باشد باشه که نوع مقدار `null` آبجکته. با قرار دادن مقدار `null` هم می‌توانیم متغیر رو خالی کنیم.

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

۷۴. تفاوت‌های بین null و undefined چیا هستن؟

تفاوت‌های اصلی بین null و undefined :

Undefined	Null
یه مقدار انتساب نیست که تو اون متغیری اعلام شده باشه اما هنوز مقداری به اون اختصاص داده نشده.	یه مقدار انتسابه که نشون میده متغیر به هیچ شیئی اشاره نمیکنه.
تایپ undefined همون تعریف نشده و undefined هستش	تایپ null آبجکته
مقدار undefined یه مقدار اولیه اس و زمانی استفاده میشه 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 میشه

۷۵. متد eval چیه؟

تابع eval کد جاواسکریپت‌ای رو که به صورت رشته بهش پاس داده شده رو اجرا میکنه. رشته میتوانه یه عبارت جاواسکریپت، متغیر، دستور یا دنباله‌ای از عبارات باشه.

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

۷۶. تفاوت‌های بین document و window چیا هستن؟

Document	Window
فرزنده مستقیم شی window هستش و همچنین به عنوان مدل شی (DOM)	عنصر ریشه در هر صفحه وبه

Document	Window
میتوانیم از طریق window.document یا document به اون دسترسی داشته باشیم.	به طور پیش فرض شی window به طور ضمنی در هر صفحه قرار دارد
متدهایی مانند getElementById، getElementByTagName، createElement وغیره را فراهم میکنند	دارای متدهایی مانند alert، confirm و ویژگی‌هایی مانند document، location

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

شی window.history حاوی تاریخچه مرورگره. با استفاده از متدهای back و next میتوانیم URLهای قبلی و بعدی را در تاریخچه بارگذاری کنیم. مثلا:

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

نکته: همچنانیم میتوانیم بدون پیشوند window history دسترسی داشته باشیم.

۷۸. انواع داده‌های جاواسکریپت کدوما هستند؟

- ۱. Number
- ۲. String
- ۳. Boolean
- ۴. Object
- ۵. Undefined

۷۹. isNaN چیه و چیکار میکنه؟

تابع `isNaN` (Not-a-Number) برای تعیین اینه که آیا یه مقدار یه عدد واقعیه یا نه 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؟

<code>undefined</code>	<code>undeclared</code>
این متغیرها در برنامه هست، اما هیچ مقداری نداره	این متغیرها تو یه برنامه وجود ندارن و تعریف نشدن
اگه سعی کنیم مقدار یه متغیر تعریف نشده رو بخوئیم، یه مقدار تعریف نشده برگردانده میشه.	اگه سعی کنیم مقدار یه متغیر undeclared رو بخوئیم، با خطای زمان اجرا مواجه میشیم

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 اوно حذف کنیم تبدیل به متغیر عمومی 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 متغیر واقعاً عددی نیست. استفاده از NaN تو برنامه‌ها خیلی نداره، اما می‌شه از اون به عنوان مقدار بازگشتی یه سری توابع و برای یه سری موارد کم استفاده کرد:

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

۸۴. هدف از تابع isFinite چیه؟

تابع isFinite برای تعیین اینکه آیا یه عدد یه عدد محدود و قانونیه استفاده می‌شه. اگه مقدار NaN (Not-a-Number) یا infinity، -infinity+ یا false باشه 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 یه المنت در صفحه وب کلیک می‌کنیم و این المنت به شکل تودرتو داخل المنتهای مختلف استفاده شده، قبل از اینکه کلیکمون واقعاً به مقصد یا المنت هدف برسه، باید event کلیک رو برای هر کدوم از عنصرهای والد خودش بفرسته و از بالا با شی پنجره گلوبال شروع شه. به شکل کلی دو راه برای جریان event وجود داره:

۱. از بالا به پایین(Event Capturing)

۲. از پایین به بالا(Event Capturing)

۳. از بالا به پایین(Event Capturing)

۴. از پایین به بالا(Event Capturing)

چیه؟ Event-bubbling .۸۶

نوعی انتشار event که تو اون event ابتدا روی درونی ترین عنصر هدف فراخوانی می‌شود و سپس به طور متوالی روی اجداد (والد) عنصر هدف تو همون سلسله مراتب تودرتو راه‌اندازی می‌شود تا زمانی که به بیرونی ترین عنصر DOM برسد.

چیه؟ Event-capturing .۸۷

نوعی انتشار event که تو اون event اول با بیرونی ترین عنصر ثبت می‌شود و سپس به طور متوالی بر روی children عنصر هدف تو همون سلسله مراتب تودرتو راه‌اندازی می‌شود تا زمانی که به درونی ترین عنصر DOM برسد.

چطوری می‌شود که فرم رو با استفاده از جاوااسکریپت ثبت کرد؟ .۸۸

می‌توانیم با استفاده از جاوااسکریپت فرم مورد نظرمون رو با استفاده از کد ارسال document.form[0].submit() مطلاعات ورودی فرم با استفاده از ارسال می‌شود onsubmit event handler

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

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

شی window.navigator حاوی اطلاعاتی درباره جزئیات سیستم و این مربوط رگر بازدیدکننده است. بعضی از ویژگی‌های سیستم عامل روی ویژگی پلتفرم در دسترس

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

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

رویداد `DOMContentLoaded` زمانی فراخوانی می‌شه که سند اولیه HTML بهطور کامل بارگیری و تجزیه شده باشه، بدون اینکه منتظر بمونیم تا بارگیری asset‌ها (استایل‌ها، تصاویر و فریم‌های فرعی) تمام شه. در حالی که رویداد `load` روی داکیومنت زمانی فراخوانی می‌شه که کل صفحه بارگیری شه، شامل همه استایل‌ها، تصاویر و

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

Native object‌ها آبجکت‌هایی هستن که به عنوان بخشی از زبان جاواسکریپت تعریف شدن و به عنوان بخشی از مشخصات ECMAScript هستن. برای مثال، اشیاء اصلی رشته، ریاضی، `Object`، `Function` و غیره که در مشخصات ECMAScript تعریف شدن.

Host objects آبجکت‌هایی هستن که توسط مرورگر یا محیط زمان اجرا (Node) ارائه می‌شون. برای مثال، پنجره، `XMLHttpRequest` نودهای DOM و غیره به عنوان اشیاء میزبان در نظر گرفته می‌شن.

User objects آبجکت‌هایی هستن که تو کد جاواسکریپت تعریف شدن. برای مثال، آبجکت‌های ایجاد شده توسط برای اطلاعات پروفایل.

۹۲. کدوم ابزار و تکنیک‌ها برای دیباگ کردن برنامه جاواسکریپتی استفاده می‌شن؟

۱. Chrome Devtools
۲. عبارت `debugger`
۳. متدهای `console.log`

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

مزایا:

مزایا:

۱. از جهنم callback که قابل خواندن نیست جلوگیری میکنه.
۲. نوشتن کدهای ناهمزمان متوالی با then آسون تره.
۳. نوشتن کد ناهمزمان موازی با all آسون تره.
۴. بعضی از مشکلات رایج callback‌های بازگشتی رو حل میکنه (مشکل فراخوانی بسیار دیر، خیلی زود، یا چندبار فراخوانی callback و استثناهای رو مدیریتیش را راحت‌تر میکنه).

معایب:

۱. کد یه کمی پیچیده میشه.
۲. اگه ES6 پشتیبانی نشد باید یه polyfill بارگذاری بشه.
۳. کد کمی پیچیده می‌سازد
۴. اگه ES6 پشتیبانی نمی‌شه، باید یه polyfill بارگذاری کنیم

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

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

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

می‌تونیم مقدار یه ویژگی رو به صورت زیر بدست بیاریم:

```
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 به "Good evening" تغییر داده شد، نتیجه مثل زیر می‌شه:

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

۹۵. سیاست same-origin چیه؟

سیاست یا خط مشی same-origin خط مشی که از درخواست جاوااسکریپت در روی کل domain جلوگیری میکنے. مبدأ به عنوان ترکیبی از شمای URI، نام میزبان(hostname) و شماره پورت(port) تعریف میشے. اگه این خطمشی رو فعال کنیم، مرورگر از دسترسی یه اسکریپت مخرب تو یه صفحه به دادههای حساس توی صفحه وب دیگه با استفاده از DOM(Document Object Model) جلوگیری میکنے.

۹۶. هدف استفاده از void چیه؟

Void(0) برای جلوگیری از به روزرسانی صفحه استفاده میشە. این متده برای از بین بردن ساید افکتهای ناخواسته مفیده، چون مقدار اولیه تعریف نشده رو برمیگردونه. معمولاً برای اسناد HTML استفاده میشە که از "href="JavaScript:Void(0)" رو تو یه عنصر <a> استفاده میکنن. یعنی وقتی روی یه لینک کلیک میکنیم مرورگر یه صفحه جدید رو بارگیری میکنە یا همون صفحه رو تازهسازی(reload) میکنە. ولی با استفاده از این عبارت میتونیم از این رفتار جلوگیری کنیم.

برای مثال، لینک زیر پیام رو بدون بارگیری مجدد صفحه مطلع مطلع میکنە

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

۹۷. جاوااسکریپت یه زبان تفسیری هست یا کامپایلری؟

جاوااسکریپت یه زبان تفسیریه و نه یه زبان کامپایلری. یه مفسر توی مرورگر کد جاوااسکریپت رو می خونه، هر خط رو تفسیر میکنە و اونو اجرا میکنە. امروزه مرورگرهای مدرن از فناوری موسوم به کامپایل Just-In-Time(JIT) استفاده میکنن که جاوااسکریپت رو موقعی که در شرف اجراست به بایت کد اجرایی کامپایل میکنە.

۹۸. آیا جاوااسکریپت یه زبان حساس به بزرگ و کوچک(case-sensitive) حرروف است؟

بله، جاوااسکریپت یه زبان حساس به حرروف کوچک و بزرگه. کلمات کلیدی استفاده شده توی زبان، متغیرها، توابع و اشیا، و هر شناسه دیگر باید همیشه با حرروف بزرگ تایپ شن.

۱۹۹. ارتباطی بین JavaScript و Java وجود داره؟

نه، این دو زبان برنامه نویسی کاملاً متفاوت هستن و هیچ ارتباطی با همدیگه ندارن. اما هر دوی اونا زبان‌های برنامه نویسی شی گرا هستن و مثل خیلی از زبان‌های دیگه، از syntax مشابهی برای ویژگی‌های اساسی (if، else، for، switch، break، continue) غیره) پیروی می‌کنن.

۱۰۰. Event‌ها چی هستن؟

رویدادها «چیزهایی» هستن که روی عناصر HTML و برای اونا اتفاق می‌افتن. موقعی که جاواسکریپت توی صفحات HTML استفاده می‌شه، می‌تونه به این رویدادها واکنش نشون بده و ما با استفاده از این رخدادها می‌تونیم رفتار خاصی رو موقع رخداد خاص تعریف کنیم. بعضی از نمونه‌های رویدادهای HTML عبارتند از:

۱. بارگذاری صفحه وب تموم شه
۲. فیلد ورودی تغییر کنه
۳. روی یه دکمه کلیک شه

بیاین رفتار رویداد کلیک رو برای یه button ببینیم:

```
<!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 Brendan Eich تو سال ۱۹۹۵ و موقع فعالیت ایشون توی نت اسکیپ (Netscape Communication) ایجاد شد و با نام Mocha توسعه پیدا کرد، اما

بعدها زمانی که برای اولین بار در نسخه‌های بتا نت اسکریپت 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 اتفاق نمی‌افته. برای مثال، جلوگیری از ارسال فرم موقع کلیک بر روی دکمه ارسال و جلوگیری از باز شدن **URL** صفحه موقع کلیک کردن روی لینک از موارد رایج استفاده شده.

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

نکته: همه رویدادها قابل لغو نیستن.

۱۰۳. کاربرد متدها stopPropagation چیه؟

روش **stopPropagation** برای جلوگیری از **event bubbling** توی یه زنجیره از رویدادها استفاده می‌شه. برای مثال، **div**‌های تودرتو زیر با متدها **stopPropagation** از انتشار پیش‌فرض رویداد موقع کلیک روی **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 event-handler رخ میده چیا هستن؟

عبارت `return false` تو event-handler مراحل زیر رو انجام میده:

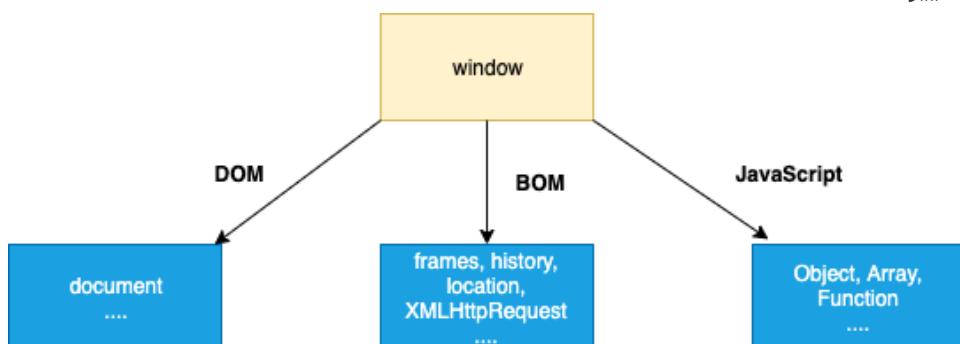
۱. ابتدا عملکرد یا رفتار پیش فرض مرورگر رو متوقف میکنه.

۲. رویداد از انتشار DOM جلوگیری میکنه.

۳. اجرای callback رو متوقف میکنه و بلافاصله پس از فراخونی بر می‌گردد.

۱۰۵. BOM چیه؟

مدل آبجکتی مرورگر (BOM) به جاواسکریپت اجازه میده تا با مرورگر صحبت کنه. این مدل شامل `window`, `document`, `navigation`, `history`, `page`, `location` که فرزندان `document` هستن. BOM مدل استاندارد همه مرورگرها نیست و می‌توانه بر اساس مرورگرهای مختلف تغییر کنه.



۱۰۶. موارد استفاده از `setTimeout` کدوما هستن؟

متدهای `setTimeout` برای فراخونی یه تابع یا ارزیابی یه عبارت پس از میزان مشخصی از زمان(میلی ثانیه) استفاده می‌شه. برای مثال، بیاین یه پیام رو پس از 2 ثانیه با استفاده از متدهای `console.log` چاپ کنیم:

```
setTimeout(() => {  
  console.log("Good morning");  
, 2000);
```

۱۰۷. موارد استفاده از setInterval کدوما هستن؟

متدهای `setInterval` برای فراخوانی یه تابع یا ارزیابی یه عبارت در بازه‌های زمانی مشخص (بر حسب میلی ثانیه) استفاده می‌شوند. برای مثال، بیاین یه پیام رو هر 2 ثانیه با استفاده از متدهای `setInterval` چاپ کنیم:

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

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

چون مشخصات زبان به برنامه نویس اجازه نمیده تا کدی بنویسه که مفسر بتوانه بخش‌هایی از اوно به صورت موازی در چندین Thread یا پردازش اجرا کنه. در حالی که زبان‌هایی مانند C++، java، go، می‌توانن برنامه‌های چند رشته‌ای و چند فرآیندی بسازن.

۱۰۹. Event-delegation چیه؟

تکنیکی برای گوش دادن به رویدادهایی که تو اون یه عنصر والد رو به عنوان شنونده برای همه رویدادهایی که در داخلش اتفاق می‌افتن، تفویض می‌کنیم. برای مثال، اگه می‌خواین تغییرات فیلد رو تو یه فرم خاص تشخیص بدین، می‌تونیم از تکنیک `Event-delegation` استفاده کنیم.

```
const form = document.querySelector("#registration-form");

// Listen for changes to fields inside the form
form.addEventListener(
  "input",
  (event) => {
    // Log the field that was changed
    console.log(event.target);
  },
  false
);
```

۱۱۰. ECMAScript چیه؟

زبان برنامه نویسی که اساس جاواسکریپت رو تشکیل میده. ECMAScript توسط سازمان استانداره بین المللی ECMA در مشخصات ۲۶۲ و ECMAScript استانداره شده است. اولین نسخه ECMAScript در سال ۱۹۹۷ منتشر شد. ECMAScript-402

۱۱۱. JSON چیه؟

JSON (JavaScript Object Notation) یه فرمت سبک هستش که برای تبادل داده‌ها استفاده می‌شه. اصول اولیه جیسون بر اساس زیرمجموعه‌ای از زبان جاواسکریپت و مشابه آبجکت اشیاییه که توی جاواسکریپت ساخته میشن.

۱۱۲. قوانین فرمت JSON کدوما هستن؟

۱. داده‌ها به صورت جفت نام/مقدار هستن
۲. داده‌ها با کاما از هم جدا میشن
۳. برآکتها اجسام رو نگه می‌دارن
۴. کروشه‌ها آرایه‌ها رو نگه می‌دارن

۱۱۳. هدف از متدهای JSON.stringify چیه؟

موقع ارسال داده‌ها به وب سرور، داده‌ها باید در قالب رشته‌ای باشن. می‌تونیم با استفاده از متدهای JSON آبجکت رو به رشته متنی تبدیل کنیم. مثل:

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

۱۱۴. چطوری می‌توnim یه رشته JSON رو تجزیه کنیم؟

موقع دریافت داده‌ها از یه وب سرور، داده‌ها همیشه در قالب رشته متنی هستن. اما می‌توnim این مقدار رشته رو با استفاده از متدها `parse` به یه آجکت جاواسکریپت تبدیل کنیم.

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

۱۱۵. چرا به JSON نیاز داریم؟

موقع تبادل داده بین مرورگر و سرور، داده‌ها به دلیل تبادل شدن با استفاده از پروتکل HTTP فقط می‌توونن متنی باشن. از اونجایی که JSON فقط متنیه میشه اونو به راحتی به سرور ارسال کرد و از اون به عنوان قالب داده برای هر زبان برنامه‌نویسی استفاده کرد.

۱۱۶. PWAها چی هستن؟

نوعی از برنامه‌های تلفن همراه هستن `Progressive web applications (PWAs)` که از طریق وب ارائه میشن، و با استفاده از فناوری‌های رایج وب از جمله `HTML`، `CSS` و `Javascript` ساخته میشن. PWAs در سرورها قرار می‌گیرن و از طریق آدرس صفحه وب قابل دسترسی و نصب هستن.

۱۱۷. هدف از متدها `clearTimeout` چیه؟

تابع `clearTimeout` در جاواسکریپت برای پاک کردن بازه زمانی استفاده می‌شه که قبل از اون توسط تابع `setTimeout` تنظیم شده. یعنی مقدار بازگشتی تابع `setTimeout` تو یه متغیر ذخیره می‌شه و برای پاک کردن تایمر به تابع `clearTimeout` پاس داده می‌شه.

برای مثال، از تابع `setTimeout` موجود توی کد پایین برای نمایش پیام بعد از 3 ثانیه استفاده می‌شه. این مهلت زمانی رو می‌شه با تابع `clearTimeout` پاک کرد.

```

<script>
  var msg;

  function greeting() {
    alert("Good morning");
  }

  function start() {
    msg = setTimeout(greeting, 3000);
  }

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

```

۱۱۸. هدف از متدهای clearInterval چیه؟

تابع `clearInterval` تو جاواسکریپت برای پاک کردن `interval` 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 `interval` به تابع `clearInterval` ارسال می‌شود.

برای مثال، از تابع `setInterval` توی کد پایینی برای نمایش پیام در هر ۳ ثانیه استفاده می‌شود. این بازه رو می‌شه با تابع `clearInterval` پاک کرد.

```

<script>
  var msg;

  function greeting() {
    alert("Good morning");
  }

  function start() {
    msg = setInterval(greeting, 3000);
  }

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

```

۱۱۹. توى جاواسكريپت، چطوري مىشە بە يە صفحە جديد redirect انجام داد؟

در `vanila` جاواسكريپت خام یا خالص هم ميگن)، مىتونيم با استفاده از ويزگي `location` آبجكت گلوبال `window` به صفحە جديدي هدایت بشين.

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

۱۲۰. چطوري بررسى مىكним كە يە string شامل يە substring هست يَا نە؟

سە روش براي بررسى اينكه يە رشته داراي يە رشته فرعىيە يَا نە، وجود داره.
۱. استفاده از متده `String.prototype.includes`: ES6 روش `includes` برای آزمایيش يە رشته حاوي يە رشته فرعى ارائه كرد.

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

۲. استفاده از متده `indexof`: توى محيط ES5 يە اقدمى تر ، مىتونىم از استفاده كنيم كە `indexof` يە رشته فرعى رو برمىگردونه. اگه مقدار برابر با 1 نباشه، يعنى رشته فرعى توى رشته اصلى وجود داره.

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

۳. استفاده از `Regex`: راه حل پيشرفتە از روش `test` عبارت استفاده ميکند، كە امك ان آزما يش در برابر عبارات منظم رو فراهم ميكنه.

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

۱۲۱. توی جاواسکریپت، چطوری مقدار یه آدرس email رو اعتبارسنجی می‌کنیم؟

می‌تونیم با استفاده از `Regex` ایمیل رو توی جاواسکریپت تأیید کنیم. توصیه می‌شه به جای سمت کلاینت، اعتبارسنجی سمت سرور انجام شه. چون جاواسکریپت رو می‌شه سمت کلاینت غیرفعال کرد.

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

بالا کarakترهای یونیک رو می‌پذیرد `Regex`

۱۲۲. چطوری می‌تونیم مقدار آدرس url جاری رو بخونیم؟

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

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

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

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

۱. URL - href
۲. URL - protocol
۳. URL - host
۴. URL - hostname

- ۵ - شماره پورت URL .port
- ۶ - مسیر URL .pathname
- ۷ - قسمت جستجو URL .search
- ۸ - محل جایگیری URL .hash

۱۲۴. توی جاواسکریپت چطوری میتونیم مقدار یه query-string رو بخونیم؟

میتونیم از `URLSearchParams` برای دریافت مقادیر رشته پرس و جو توی جاواسکریپت استفاده کنیم. بیاین مثالی برای دریافت مقدار کد مشتری از رشته پرس و جو `URL` ببینیم:

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

۱۲۵. چطوری میتونیم بررسی کنیم که آیا یه پراپرتی روی آبجکت وجود داره یا نه؟

۱. استفاده از عملگرها: میتونیم از عملگر `in` استفاده کنیم که آیا کلیدی توی آبجکت وجود داره یا نه

```
"key" in obj;
```

برای بررسی وجود یا عدم وجود کلید، باید از پرانتز استفاده کنیم

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

۲. استفاده از متدهای `hasOwnProperty`: میتوانیم از `hasOwnProperty` برای آزمایش ویژگیهای نمونه آبجکت (و نه ویژگیهای ارثی) استفاده کنیم.

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

۳. استفاده از مقایسه `undifiend`: اگه بخوایم از یه آبجکت به یه ویژگی غیر موجود دسترسی پیدا کنیم، بهمون `undifiend` برمیگردونه. بیاین ویژگیها را با مقایسه کنیم تا وجود ویژگی را مشخص کنیم

```

const user = {
  name: "John",
};

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

```

۱۲۶. چطوری روی یه object حلقه میزنی؟

میتونیم از حلقه `for-in` برای حلقه زدن آبجکت جاوااسکریپت استفاده کنیم. همچنین میتونیم مطمئن شیم که کلیدی که دریافت میکنیم ویژگی واقعی یه آبجکته و با استفاده از روش `Object.prototype.hasOwnProperty()` نیست.

```

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

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

```

۱۲۷. چطوری تست میکنی که یه object خالیه؟

راه حل‌های مختلفی بر اساس نسخه‌های ECMAScript وجود داره

۱. استفاده `:Object.entries(ECMA ۷+)`

میتوانیم از `Object.entries` استفاده کنیم و `length` او را چک کنیم

```

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

```

۲. استفاده از `:Object.keys(ECMA ۵+)`

میتوانیم از `Object.keys` استفاده کنیم و `length` او را چک کنیم

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

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

می‌توانیم از حلقه `for-in` استفاده کنیم و هر پارامتر رو با چک کنیم

```
function isEmpty(obj) {  
  for (const prop in obj) {  
    if (obj.hasOwnProperty(prop)) {  
      return false;  
    }  
  }  
  
  return JSON.stringify(obj) === JSON.stringify({});  
}
```

۱۲۸. `arguments` چیه؟

آبجکت `arguments` یه آبجکت آرایه ماننده که داخل توابع قابل دسترسیه و حاوی مقادیر آرگومان‌های ارسال شده به اون تابعه. برای مثال، بیاین ببینیم چطوری از آبجکت `arguments` تابع `sum` استفاده کنیم:

```
function sum() {  
  let total = 0;  
  for (let i = 0, len = arguments.length; i < len; ++i) {  
    total += arguments[i];  
  }  
  return total;  
}  
  
sum(1, 2, 3); // returns 6
```

نکته: ما نمی‌تونیم از متدهای ارایه برای آبجکت `arguments` استفاده کنیم اما می‌توانیم به ارایه تبدیلش کنیم

```
const argsArray = [...arguments];
```

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

می‌تونیم با درست کردن یه تابع و با استفاده از زنجیره ای از متدهای string مثلا slice وtoUpperCase وcharAt یه string اول بزرگ ایجاد کنیم

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

۱۳۰. مزایا و معایب حلقه for چیا هستن؟

حلقه for یه syntax تکرار رایجه که از مزایا و معایبیش میشه به موارد زیر اشاره کرد:

مزایا

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

معایب

۱. از لحاظ نوشتاری کد بیشتری باید نوشته بشه
۲. Imperative هست
۳. ممکنه با خطاهای one-by-off روبرو بشیم

۱۳۱. چطوری تاریخ جاری رو توی جاواسکریپت نشون میدی؟

می‌تونیم از کلاس new Date استفاده کنیم و با اجرای متدهLocaleString() تاریخ و زمان جدا شده توسط کاما رو بدست بیاریم :

```
const today = new Date().toLocaleString();  
console.log(today.split(",")[0]); // 5/19/2022
```

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

برای این مقایسه نباید از اپراتورها استفاده کنیم. میتوانیم مثل کد زیر از متدهای `getTime()` و `toString()` بر روی آبجکت `Date` قرار داره استفاده کنیم

```
const d1 = new Date();
const 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؟

میتوانیم از متدهای `startsWith()` و `endsWith()` برای بررسی این اتفاق استفاده کنیم که یه رشته رو می‌گیره و چک میکنه که رشته مورد نظر با اون رشته شروع میشه یا نه، برای مثال کد زیر رو براش مینویسیم:

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

۱۳۴. چطوری یه رشته رو trim می‌کنی؟

جاواسکریپت یه متدهایی به اسم `trim()` داشته باشند که روی رشته‌ها قرار داره با استفاده از این متدهایی فضاهای خالی بین اون رشته برداشته می‌شه

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

۱۳۵. توى جاواسکریپت چطوری می‌توانیم یه زوج مرتب از key یه value ها بسازیم؟

برای اضافه کردن key جدید به آبجکتها دو روش وجود داره

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

۱. استفاده از **dot**: این روش زمانی موثر هستش که اسم پرایپری رو میدونیم

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

۱. استفاده از **کروشه[]**: این روش زمانی موثر هستش که اسم پرایپری داینامیک باشد

```
obj.key3 = "value3";
```

۱۳۶. آیا عبارت '--!؟ عملگر خاصی هست؟

نه! اپراتور خاص نیست اما ترکیب دو تا اپراتور استاندار هستش یکی بعد اون یکی

۱. اپراتور نقییض (!)

۲. کاهش کننده (-)

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

۱۳۷. چطوری می‌تونیم به متغیرهای مقادیر اولیه بدم؟

می‌تونیم از عملگر یا اپراتور || برای تعریف یه مقدار پیش‌فرض استفاده کرد:

```
const a = b || c;
```

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

۱۳۸. چطوری می‌تونیم متن‌های چند خطی درست کنیم؟

می‌تونیم از / برای تعریف کردن رشته‌های چند خطی استفاده کنیم برای مثال:

```
const 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 ارور خطای نوشتاری کد قراره داشته باشیم

روش بعدی استفاده کردن از `backtick` هست که وقتی موقع تعریف رشته به جای کوتیشن مارک ازش استفاده بشه میتوانیم راحت یه رشته چند خطی تعریف کنیم. برای مثال میشه کد زیر رو براش نوشت:

```
const str = `This is a  
very lengthy  
sentence!`;
```

۱۳۹. مدل app-shell چیه؟

یکی از راههای ساخت PWA هستش که به طور قابل اعتماد و فوری بر روی صفحه نمایش کاربران شما بارگیری می‌شه، مشابه اونی که توی برنامه‌های کاربردی native به کاربر نشون داده میشه. برای رسوندن سریع HTML اولیه به صفحه بدون نیاز به شبکه مفیده.

۱۴۰. چطوری می‌تونیم روی یه تابع `property` اضافه کنیم؟

می‌تونیم برای توابع پر اپرتی تعیین کنیم چون توابع اصولاً آبجکت هستن.

```
const fn = function (x) {  
    // Function code goes here  
};  
  
fn.userName = "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` می‌تونیم به تعداد پارامترهایی که یه تابع انتظار داره بگیره دسترسی داشته باشیم.

بریم یه مثال درموردش بینیم:

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

۱۴۲. چیه؟ Polyfill

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

۱۴۳. عبارات break و continue 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 میده.

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

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}<br>`;  
}
```

۱۴۴. توی جاواسکریپت label چیکار می‌کنن؟

دستور `label` به ما اجازه میده تا حلقه‌ها و بلوک‌ها رو توی جاواسکریپت نام گذاری کنیم. بعد می‌تونیم از این برچسب‌ها برای مراجعه به کد استفاده کنیم. برای مثال، کد زیر با استفاده از برچسب‌ها از چاپ اعداد وقتی که یکسان هستن، جلوگیری می‌کنه.

```
loop1: for (let i = 0; i < 3; i++) {
  for (let 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 می‌شه که تمام تعریف متغیرها رو بالای هر اسکریپت یا تابع انجام بدیم. مزیت این کار:

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

۱۴۶. مزایای مقداردهی اولیه متغیرها چیه؟

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

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

۱۴۷. روش توصیه شده برای ایجاد object چیه؟

برای ساخت یه `object` با مقادیر پیشفرض میتوانیم به صورت زیر عمل کنیم

۱. استفاده از `{}` به جای `new Object`

۲. استفاده از `""` به جای `new String`

۳. استفاده از `0` به جای `new Number`

۴. استفاده از `false` به جای `new Boolean`

۵. استفاده از `[]` به جای `new Array`

۶. استفاده از `/()` به جای `new RegExp`

۷. استفاده از `()` به جای `new Function`

بریم چند تا مثال ببینیم:

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

۱۴۸. چطوری می‌تونیم آرایه JSON تعریف کنیم؟

برای تعریف آرایه‌های JSON از براکت استفاده میکنیم و هر تعداد که آبجکت خواستیم

داخلش تعریف میکنیم.

بیاین یه مثال در موردش ببینیم

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

۱۴۹. چطوری می‌تونیم اعداد تصادفی تولید کنیم؟

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

رو ضربدر ده کنیم عددی بین یک تا ده خواهیم داشت

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

نکته: `Math.random` یه عدد تصادفی بین 0 تا 1 ایجاد میکنه، یادمون باشه که استفاده از `Math.floor` قبل ضرب کردن عدد رندوم به دست او مده در ده به عنوان کمترین مقدار، باعث میشه که خروجیمون همیشه به صفر رند بشه.

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

بله ما می‌تونیم کد زیر رو برای این تابع 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 و به این معنیه که ماثولهای استفاده نشده در طول فرآیند ساخت در بسته گنجونده نمی‌شن و برای اون بر ساختار استاتیک ماثول ES2015 'rollup' متكیه (یعنی `import` و `export`). توی باندلر ماثول 'ES2015' از این عملکرد استفاده شده.

۱۵۲. دلایل نیاز به tree-shaking کدوما هستن؟

می‌تونه 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، اگه فقط بخواهیم یه برنامه Hello World با استفاده از چارچوبهای SPA ایجاد کنیم، حدود چند مگابایت حافظه رو اشغال میکنه، اما tree-shaking می‌تونه اندازه رو به چند صد

کیلوبایت کاهش بده. tree-shaking تو باندلرهای Webpack و Rollup پیاده سازی شد.

۱۵۳. آیا استفاده از eval توصیه می‌شود؟

نه، eval اجازه اجرای کد دلخواه را میدهد که باعث ایجاد مشکل امنیتی می‌شود. همونطور که میدونیم از تابع eval برای اجرای متن به عنوان کد استفاده می‌شود. در بیشتر موارد استفاده از اون ضروری نیست.

۱۵۴. چیه؟ Regular-Expression؟

با استفاده از این ساختار ما می‌توانیم دیتامون رو 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 می‌شود:

```
/John/i;
```

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

متدهای Regular Expressions دو تا متدهای رشته‌ها داره: replace و search. متدهای replace و search می‌گیره اونو جستجو می‌کنند و محل اون عبارت رو برمی‌گردونه:

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

متدهای replace برگرداندن رشته اصلاح شده تو جایی که الگو جایگزین می‌شود، استفاده می‌شوند:

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

۱۵۶. توی Regex بخش modifiers چیکار میکنه؟

Modifiers ها میتوانن زمانی استفاده بشن که به جستجوهای بدون حروف کوچک و بزرگ سراسری نیاز داریم بیاین 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:

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

۱۵۷. پترن‌های regular-expression چیه؟

Regex یه گروهی از ساختارها برای اینکه با اونا کاراکترها رو چک کنیم اونا تو سه مدل طبفه بندی میشن.

۱. **براکتها:** برای پیدا کردن رنگی از کاراکتر استفاده میشن

برای مثال پایین چن تا مورد استفاده لیست شدن

۱. [abc]: برای پیدا کردن هر کاراکتری بین این سه کاراکتر استفاده میشه

۲. [0-9]: برای پیدا کردن ارقام بین این دو عدد استفاده میشه

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

۲. **کاراکتر برابر با:** این عبارت‌ها کاراکترهایی با معنی خاص هستن

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

۱. d: برای پیدا کردن اعداد استفاده میشه

- ۱.۲: برای پیدا کردن فاصله‌ها استفاده می‌شود
 ۱.۳: برای پیدا کردن کاراکترهای همخوانی داشته با شروع شدن یا پایانشون استفاده می‌شود

۳. کمیت کنندگان: این‌ها برای تعریف کمیت‌ها موثر هستند
 برای مثال پایین دو تا مورد استفاده برآشون اورده‌یم
۱. n+: برای پیدا کردن رشته همخوانی داشته با حداقل یه کاراکتر
 ۲. n*: برای پیدا کردن همخوانی هر رشته شامل صفر یا بیشتر
 ۳. n?: برای پیدا کردن هر رشته که شامل صفر یا یه کاراکتر می‌شود

۱۵۸. آبجکت RegExp چیه؟

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

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

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

می‌تونیم از متدهای test منظم برای جستجوی یه رشته برای الگو استفاده کنیم که بسته به نتیجه، true یا false را برگرداند.

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

۱۶۰. هدف از متدهای exec چیه؟

هدف متدهای exec شبهه به روش تسته اما جستجوی یه تطابق تو یه رشته مشخص رو انجام میده و یه آرایه نتیجه یا null رو به جای برگرداندن true/false برگرداند.

```
const 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؟

می‌تونیم سبک درون خطی یا اسم کلاس یه عنصر HTML رو با استفاده از جاوااسکریپت تغییر بدین

۱. استفاده از پرایپرتو **style**: با استفاده از ویژگی **style** می‌تونیم استایل درون خطی رو تغییر بدین

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

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

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

۱۶۲. نتیجه عبارت $1'3'+2+1'3'$ 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 تو دستور **debugger** مثل 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 دیباگر، می‌توانیم breakpoint‌ها را در کد جاواسکریپت تنظیم کنیم. در هر breakpoint، جاواسکریپت اجرا نمی‌شود و به ما اجازه میدهد مقادیر جاواسکریپت رو بررسی کنیم. پس از بررسی مقادیر، می‌توانیم با استفاده از دکمه پخش، اجرای کد رو ادامه بدیم.

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

نه، ما نمی‌توانیم از کلمات رزرو شده به عنوان متغیر، برچسب، اسم آبجکت یا تابع استفاده کنیم. بیان یه مثال ساده رو بینیم:

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

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

ما می‌توانیم با استفاده از Regex که یه boolean بفهمیم که مرورگری که کاربر داره ازش استفاده میکنه موبایل هست یا نه، کد تشخیص به این صورت نوشته میشه:

```

window.mobilecheck = function () {
    let mobileCheck = false;
    (function (a) {
        if (
            /(android|bb\d+|meego).+mobile|avantgo|bada\/|blackberry|blazer|co
            |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 با چیزی مطابقت داره یا نه، شناسایی کنیم. این یه راه حل جایگزین برای استفاده از RegExp هستش

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

  return false;
}
```

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

می‌تونیم با استفاده از جاواسکریپت به صورت برنامه‌ریزی شده تصویر رو بدست بیاریم و ابعاد (عرض و ارتفاع) رو بررسی کنیم، برای مثال syntax بررسی می‌تونه طبق مثال زیر انجام بشه:

```
const 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؟

مروگرها یه کلاس XMLHttpRequest ارائه می‌دان که می‌تونه برای ایجاد درخواستهای HTTP همزمان از جاوااسکریپت استفاده شه.

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

۱۷۰. چطوری درخواستهای asynchronous HTTP بزنیم؟

مروگرها یه کلاس XMLHttpRequest رو ارائه می‌دان که می‌تونن برای درخواستهای HTTP ناهمزمان از جاوااسکریپت با ارسال پارامتر سوم به عنوان true استفاده کنن.

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

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

می‌تونیم از متدهای toLocaleString برای تبدیل تاریخ‌ها تو یه منطقه زمانی به منطقه زمانی دیگه استفاده کنیم. برایم یه مثال درموردش بینیم.

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

۱۷۲. چه property هایی برای اندازه‌گیری سایز window به کار میره؟

متونیم از ویژگی‌های `innerWidth`, `innerHeight`, `clientWidth`, `clientHeight` ویندوز، عنصر `body` document و آجکت `document` برای پیدا کردن اندازه یه پنجره استفاده کنیم. بیانی از ترکیب اونا برای محاسبه اندازه یه `window` یا `document` استفاده کنیم.

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

const 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 متغیره که به عنوان میانبر برای دستور `if` عمل میکنه.

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

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

بله، متونیم روی عملگرهای زنجیره‌سازی مشابه اعمال کنیم `if ... else if ... else if... other chain`.

```

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 جاوااسکریپت 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؟

آبجکت واقعیه که در زنجیره جستجو برای حل متدها و غیره استفاده می‌شود. در حالی که `prototype` آبجکت‌ایه که برای ساخت `Object` استفاده می‌شود زمانی که یه آبجکت با `new Object()` ایجاد می‌کنیم.

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

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

توصیه می‌شود که بعد از هر عبارت در جاوااسکریپت از سیمیکالن استفاده کنیم. برای مثال، توی مثال زیر نذاشتن سیمیکالن، خطای `is not a function` را در زمان اجرا ایجاد می‌کند:

```
// define a function  
const fn = (function () {  
    // ...  
})() => {  
    // ...  
}();
```

از مثل بالا جاوااسکریپت اینطور برداشت می‌کند

```
const fn = (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. با خاطر همین برای تابع دوم خطای `is not a function` رو موقع اجرا می‌گیریم.

۱۷۸. متدهای `freeze` می‌کار می‌کنه؟

متدهای `freeze` برای فریز کردن یه آبجکت استفاده می‌شود. ثابت کردن یه آبجکت اجازه افزودن ویژگی‌های جدید به یه آبجکت رو نمی‌دهد. از حذفش جلوگیری می‌کنند و از تغییر

قابلیت شمارش پذیری، پیکربندی یا قابلیت نوشتن ویژگی‌های موجود جلوگیری میکنه. یعنی آبجکت‌ء 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 متده `freeze` چیه؟

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

۱۸۰. چرا به متده `freeze` نیاز داریم؟

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

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

ما می‌تونیم از آبجکت `navigator` که بر روری مرورگر وجود داره این کارو انجام بدیم

```
const 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؟

ما می‌تونیم با تابع زیر این کارو انجام بدیم:

```
function toTitleCase(str) {  
    return str.replace(  
        /\w\S*/g,  
        (txt) => 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 می‌تونیم از تگ استفاده کنیم. بلوک کد داخل `<noscript>` زمانی اجرا می‌شه که جاواسکریپت غیرفعال 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>  
    <a href="next_page.html?noJS=true"  
        >JavaScript is disabled in the page. Please click Next  
    Page</a>  
    >  
</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)، <= (بزرگتر مساوی)، > (کوچکتر)، >= (کوچکتر مساوی)
۳. **عملگرهای منطقی**: شامل && ("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 دریافت کنیم:

```
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: پارامتر `Rest` در ES6 به استاندارد جاوا اسکریپت اضافه شد

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

پارامتر Rest چون وظیفه اش جمع آوری تمام آرگومان های باقی مونده تو یه آرایه اس پس باید همیشه آخرین پارامتر باشه. برای مثال، اگه تابعیو مثل کد زیر تعریف کنیم معنی نداره و یه خطای ایجاد میکنه:

```
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 راست (<<<)

۱۸۸. عملگر spread چیکار میکنه؟

عملگر Spread به تکرار پذیرها (آرایه ها / اشیاء / رشته ها) 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؟

- متند `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؟

- متند `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 در نظر گرفته میشه:

۱. هردو `undefined`
۲. هردو `null`
۳. هردو `false` یا هر دو `true`
۴. هر دو رشته با طول یکسان با کاراکترهای مشابه به ترتیب یکسان
۵. هر دو یه آبجکت (یعنی هر دو شی رفنس یکسان دارن)
۶. هر دو عدد و
 ۱. هر دو `+`
 ۲. هر دو `-`
 ۳. 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؟

برای مقایسه دو رشته یا عدد و یا آبجکت با هم دیگه و یا پیدا کردن قطبیت دو عدد استفاده میشه:

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

میتونیم از متدها `Object.assign` استفاده کنیم که برای کپی کردن مقادیر و ویژگیها از یه یا چند آبجکت منبع به یه آبجکت هدف استفاده میشه. آبجکت مورد نظر رو که دارای خواص و مقادیر کپی شده از آبجکت اولیه اس رو 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 داره، بنابراین مقداش بازنویسی شده.

۱۹۳. کاربردهای متدها `assign` چیه؟

۱. برای شبیه سازی یه آبجکت .
۲. برای ادغام آبجکت‌ها با ویژگی‌های یکسان .

۱۹۴. آبجکت proxy چیه؟

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

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

بیان مثالی از شیء پروکسی بزنیم،

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

const 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 می‌کنند که رفتار پراکسی رو موقع انجام عملیات روی اون تعریف می‌کنند.

۱۹۵. هدف از متده 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 می‌شود. اما مقادیر پرپرتبهای فعلی تا زمانی که قابل نوشتمن باشند، قابل تغییر هستند. بیان 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 `freeze` و `seal` 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، می‌شه تغییرات روی ویژگی‌های موجود ایجاد کرد.

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 می‌شون. اگه همه شرایط زیر درست باشه یه شی مهر و موم می‌شه

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 همون ترتیبی که توسط یه حلقه `for...in` ارائه می‌شود، استفاده می‌شود. بیاین عملکرد متد `Object.entries` رو تو یه مثال بینیم،

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

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

نکته: ترتیب به عنوان آبجکت تعریف شده تضمین نمی‌شود.

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 بر می‌گردانه. [key,value]

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

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

۲۰۱. چطوری لیست کلیدهای یه `object` رو بدست میاری؟

می‌تونیم از متد `Object.keys` استفاده کنیم که برای برگرداندن آرایه‌ای از اسم پیشگی‌های یه آبجکت معین استفاده می‌شود، به همون ترتیبی که با یه حلقه معمولی دریافت می‌کنیم. برای مثال:

```

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

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

```

۲۰۴. چطوری یه object با prototype درست می‌کنی؟

متدهای `Object.create` برای ایجاد یه object جدید با `object prototype` و ویژگی‌های مشخص شده استفاده می‌شه. برای مثال، از یه object موجود به عنوان `object prototype` جدید ایجاد شده استفاده می‌کنه. یه object جدید رو با ویژگی‌های مشخص شده برمی‌گردونه.

```

const user = {
  name: "John",
  printInfo() {
    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

```

۲۰۵. چیه WeakSet؟

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

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

بیاین مثال زیرو برای توضیح رفتارش ببینیم،

```

const ws = new WeakSet();
const 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. تفاوت‌های بین Set و WeakSet کدوما هستن؟

تفاوت اصلی اینه که ارجاع به اشیاء تو Set قویه در حالی که ارجاع به اشیا تو WeakSet ضعیفه. برای مثال، یه شی تو WeakSet میتونه زباله جمع آوری شه اگه مرجع دیگری به اون وجود نداشته باشه.

تفاوت‌های دیگر عبارتند از

۱. مجموعه‌ها میتونن هر مقداری رو ذخیره کنن در حالی که WeakSets میتونه تنها مجموعه‌ای از اشیاء رو ذخیره کنه
۲. WeakSet برخلاف Set دارای ویژگی اندازه نیست
۳. WeakSet متدهایی مانند پاک کردن، کلیدها، مقادیر، ورودی‌ها، forEach را نداره.
۴. WeakSet قابل تکرار نیست.

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

۱. add(value): یه شی جدید با مقدار داده شده به مجموعه ضعیف اضافه می‌شه
 ۲. delete(value): مقدار رو از مجموعه WeakSet حذف میکنه.
 ۳. has(value): اگه مقدار در مجموعه WeakSet وجود داشته باشه true را برمی‌گردونه در غیر این صورت false را برمی‌گردونه.
 ۴. length: طول ضعیف SetObject رو برمی‌گردونه
- بیاین عملکرد تمام روش‌های بالا رو توی یه مثال ببینیم،

```

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

```

۲۰۶. چیه WeakMap؟

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 میتوون مقادیر دلخواه باشن. برای مثال:

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

بیاین مثال زیرو برای توضیح رفتار اون ببینیم،

```

const ws = new WeakMap();
const 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. WeakMaps ها میتونن هر نوع کلیدی رو ذخیره کنن، در حالی که Map میتونه مجموعه‌ای از اشیاء کلیدی رو ذخیره کنه
۲. WeakMap برخلاف Map دارای ویژگی size نیست
۳. WeakMap متدهایی مثل clear, keys, values, entries forEach رو نداره.

۳۰۸. لیست متدهایی که رو WeakMap قابل استفاده هستن رو می‌تونی بگی؟

۱. set(key, value): مقدار کلید رو در آبجکت WeakMap تنظیم میکنه. آبجکت 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 آبجکت 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 ببینیم،

```
const weakMapObject = new WeakMap();
const firstObject = {};
const 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);
```

۳۰۹. هدف از متد uneval چیه؟

uneval یه تابع داخلیه 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 درموردنش ببینیم:

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

۲۱۰. چطوری یه URL رو encode می‌کنی؟

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

```
const uri = "https://mozilla.org/?x=шеллы";
const 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 می‌شود.

```
const uri = "https://mozilla.org/?x=шеллы";
const 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؟

آجکت `window` یه متده `print` ارائه میکنه که برای چاپ محتویات پنجره فعلی استفاده می‌شود. یه قادر محاوره ای چاپ رو باز میکنه که بهمون این امکان رو میده که بین گزینه‌های مختلف برای چاپ انتخاب کنیم. بیاین استفاده از روش چاپ رو تو یه مثال ببینیم،

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

نکته: بیشتر مرورگرها، زمانی که قادر چاپ بازه صفحه قفل می‌شوند.

۲.۱۳. 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، بر عکس عمل میکنه. بیاین مثالی رو برای روشن شدن تفاوت ببینیم،

```
const msg = uneval(() => "Hello, Good morning");
const greeting = eval(msg);
greeting(); // returns "Hello, Good morning"
```

۲.۱۴. تابع anonymous چیه؟

تابع ناشناس یه تابع بدون اسمه! توابع ناشناس معمولاً به نام متغیر اختصاص داده میشن یا به عنوان یه تابع 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،

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

۲۱۵. تفاوت تقدم بین متغیرهای local و global چطوریه؟

یه متغیر local بر یه متغیر global با همون اسم ارجاعیت داره. بیاین این رفتار رو تو یه مثال ببینیم.

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

۲۱۶. accessor جاوااسکریپت چیکار می‌کنن؟

ECMAScript 5 accessor های آبجکت جاوااسکریپت یا ویژگی‌های محاسبه شده رو از طریق گیرنده‌ها و تنظیم‌کننده‌ها معرفی کرد. Getters از کلمه کلیدی "get" استفاده می‌کنه در حالی که از کلمه کلیدی "set" استفاده می‌کنه.

```
const 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 constructor مقدار تعريف می‌کنی؟

متد استاتیک Object.defineProperty برای تعريف یه ویژگی جدید به طور مستقیم بر روی یه آبجکت یا تغییر ویژگی موجود روی یه آبجکت استفاده می‌شه و آبجکت رو برمی‌گردونه. بیاین مثالی رو ببینیم تا بدونیم چجوری ویژگی رو تعريف 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 کنیم. اگه از «`get`» استفاده می‌کنیم ویژگی روی `prototype` شی تعریف می‌شده، در حالی که با استفاده از `Object.defineProperty`، ویژگی روی نمونه‌ای که بهش اعمال می‌شده، تعریف می‌شود.

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

۱. اونا `syntax` ساده‌تری ارائه میدن
۲. اونا برای تعریف ویژگی‌های محاسبه شده یا دسترسی‌ها در JS استفاده می‌شن
۳. برای ارائه رابطه هم ارزی بین خواص و روش‌ها مفیدن
۴. اونا می‌توانن کیفیت داده‌های بهتری رو ارائه بدن
۵. برای انجام کارها در پشت صحنه با منطق محصور شده مفیدن.

۲۲۰. می‌توانیم `defineProperty` و `getter` را با استفاده از متدهای `setter` و `getter` تعريف کنیم؟

بله، می‌توانیم از روش `Object.defineProperty` برای اضافه کردن `Getters` و `Setter` استفاده کنیم. برای مثال، آبجکت شمارنده زیر از ویژگی‌های افزایش، کاهش، جمع و تفریق استفاده می‌کنند:

```

const obj = { counter: 0 };

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

// Define setters
Object.defineProperty(obj, "add", {
  set(value) {
    this.counter += value;
  },
});
Object.defineProperty(obj, "subtract", {
  set(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` تو جاوااسکریپت برای اهداف تصمیم گیری استفاده می‌شه. توى چند مورد، استفاده از دستور `switch case` راحتتر از `if-else` هست. بریم یه مثال در موردش بینیم:

```

switch (expression)
{
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;

    .
    .
    .

    case valueN:
        statementN;
        break;
    default:
        statementDefault;
}

```

دستور چند حالته بالا یه راه آسون برای انجام عملیات مختلف بهمون میده که بر اساس مقدار مبنا میشه عملکردهای مختلفی رو در نظر گرفت.

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

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

۲۲۳. نوع داده‌های primitive کدوما هستن؟

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

۱. string
۲. number
۳. boolean
۴. null

۳۲۴. روش‌های مختلف دسترسی به property‌های object کدوماً هستن؟

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

```
objectName.property;
```

۲. **دسترسی با کروشه:** از کروشه برای دسترسی به دیتا استفاده می‌کنیم

```
objectName.property;
```

۳. **دسترسی عبارتی:** از عبارت توى کروشه استفاده می‌کنیم

```
objectName[expression];
```

۳۲۵. قوانین پارامترهای توابع کدوماً هستن؟

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

۲. بررسی نوع آرگومان‌های ارسال شده رو انجام ندین.

۳. تعداد آرگومان‌های دریافتی رو بررسی نکنیم.

تابع زیر از قوانین بالا پیروی می‌کنند،

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

۳۲۶. آبجکت error چیه؟

یه آبجکت خطای داخلیه که موقع بروز خطا، اطلاعات خطا رو ارائه میده و این دو ویژگی رو داخلش داره: name و message. برای مثال، تابع زیر جزئیات خطا رو ثبت می‌کنند:

```

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

```

۲۲۷. چه موقعی خطای syntax دریافت می‌کنیم؟

اگه بخواییم کد رو با یه خطای syntax ارزیابی کنیم یه SyntaxError ارسال می‌شه. برای مثال، کد زیر برای پارامتر تابع یه خطای syntax 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 تو تابع eval رخ داده	EvalError
خطایی با عدد "خارج از محدوده"	RangeError
خطا به دلیل ارجاع غیرقانونی	خطای مرجع
خطای ناشی از خطای syntax	SyntaxError
خطای ناشی از خطای type	TypeError
یه خطا به دلیل encodeURI	خطای URI

۲۲۹. عبارات مختلف که در هنگام مدیریت error استفاده می‌شون کدوما

هستن؟

۱. این عبارت برای آزمایش یه بلوک کد برای خطاهای استفاده می‌شه
۲. این عبارت برای رسیدگی به خطای استفاده می‌شه
۳. این عبارت برای ایجاد خطاهای سفارشی استفاده می‌شه.
۴. این عبارت برای اجرای کد پس از تلاش و گرفتن بدون توجه به نتیجه استفاده می‌شه.

۲۳۰. دو نوع مختلف حلقه‌ها تو جاواسکریپت کدوما هستن؟

۱. حلقه‌های کنترل شده توسط ورودی: تو این نوع حلقه، شرایط تست قبل از ورود به بدنه حلقه آزمایش می‌شه. برای مثال For Loop و While Loop تو این دسته قرار می‌گیرن.
۲. حلقه‌های کنترل شده توسط خروجی: 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 false یا true 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 به زمان اجرا I/O ناهمزمان مبتنی بر رویداد، غیر مسدود کننده اس که از موتور جاواسکریپت V8 گوگل و کتابخونه libuv استفاده میکنه.

۲۳۲. 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.

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

می‌توانیم از کلاس `Intl.DateTimeFormat` استفاده کنیم که سازنده آبجکت‌هایی که قالب بندی تاریخ و زمان حساس به زبان رو فعال می‌کنند. 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
```

۲۳۴. چیه؟ Iterator

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

۲۳۵. حلقه‌های (همزمان) چطوری کار می‌کنن؟

تکرار همزمان در ES6 معرفی شد و با مجموعه‌ای از اجزای زیر کار می‌کنند: `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 می‌کنند و اونو از طریق متدهای «`next`» یکی یکی برمنی‌گردونه.

`IteratorResult`: این یه آبجکت‌ایه که با متدهای «`next`» برگردانده می‌شه. آبجکت شامل دو ویژگی است. ویژگی "value" حاوی یه عنصر تکرار شده و ویژگی "done" تعیین می‌کند 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 Event-loop چیه؟

یه صف از توابع event loop موقعي که یه تابع callback اجرا میشه، تابع در صف قرار میگیرد. موتور جاواسکریپت پردازش حلقه رویداد رو شروع نمیکنه تا زمانی که تابع callback اجرای کد رو تموم کنه.

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

۲۳۷ Call-stack چیه؟

Call Stack یه ساختار داده برای مفسران جاواسکریپت‌هه تا فراخونی‌های تابع تو برنامه رو پیگیری کنه و دو عمل عمده داره،

۱. هر زمان که یه تابع رو برای اجرای آن فراخوانی میکنیم اونو به stack هدایت میشه.

۲. هر زمان که اجرا کد تموم شه، تابع از stack خارج میشه.

بیاین یه مثال و توضیح خلاصه اون در قالب نمودار رو باهم ببینیم:

```

function hungry() {
  eatFruits();
}

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

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

```

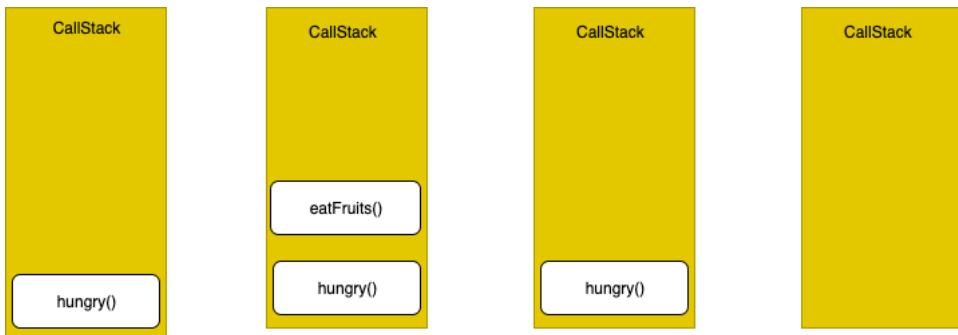
کد بالا تو یه call-stack به صورت زیر پردازش میشه.

۱. تابع 'hungry' رو به لیست call-stack اضافه میشه و کد رو اجرا میشه.

۲. تابع 'eatFruits' رو به لیست call-stack اضافه میشه و کد رو اجرا میشه.

۳. تابع 'eatFruits' از لیست call-stack ما حذف میشه.

۴. تابع 'eatFruits' از لیست call-stack ما حذف میشه.



۲۳۸. چیه؟ Event-queue

مسئول ارسال توابع جدید به stack برای پردازش. ساختارش به صورت صف داده اس تا توالی درستی رو نگه داره و همه عملیات باید برای اجرا ارسال شن.

۲۳۹. چیه؟ Decorator

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 آبجکت مورد نظر برمی‌گردونه. 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. آبجکت‌هایی هستن که مقایسه رشته‌های حساس به زبان رو امکان پذیر می‌کنن.
۲. آبجکت‌هایی هستن که قالب بندی تاریخ و زمان حساس به زبان رو فعال می‌کنن.
۳. آبجکت‌هایی هستن که قالب بندی لیست حساس به زبان رو فعال می‌کنن.
۴. آبجکت‌هایی که قالب بندی اعداد حساس به زبان رو فعال می‌کنن.
۵. آبجکت‌هایی که قالب بندی حساس به جمع و قوانین خاص زبان رو برای جمع فعال می‌کنن.
۶. آبجکت‌هایی که قالب بندی زمان نسبی حساس به زبان رو فعال می‌کنن.

۲۴۱. عملگر Unary چیه؟

عملگر unary (+) 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 بینیم:

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

const a = "Hello";
const 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 می‌شه و آرایه مرتب شده رو برمی‌گردونه. برای مثال

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

۲۴۳. هدف از تابع مرتب‌سازی موقع استفاده از متدها sort چیه؟

برای تعريف ترتیب مرتب سازی استفاده می‌شود. اگه حذف شه، عناصر آرایه به رشته تبدیل می‌شون، سپس بر اساس مقدار نقطه کد یونیک هر کاراکتر مرتب می‌شون بیاین مثالی بزنیم تا کاربرد compareFunction رو ببینیم،

```
const 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 می‌تونیم از متدها reverse استفاده کنیم. این روش برای مرتب کردن یه آرایه به ترتیب نزولی مفید است. بیاین استفاده از متدها reverse رو تو یه مثال ببینیم،

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

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

می‌تونیم از روش‌های آرایه برای یافتن حداقل و حداکثر عناصر تو یه آرایه استفاده کنیم. بیاین دو تابع برای پیدا کردن مقدار min و max تو یه آرایه ایجاد کنیم:

```

const marks = [50, 20, 70, 60, 45, 30];
function findMin(arr) {
    return Math.min(...arr)
}
function findMax(arr) {
    return Math.max(...arr));
}
console.log(findMin(marks));
console.log(findMax(marks));

```

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 میزن و هر مقدار را با کمترین یا بالاترین مقدار مقایسه میکن تا مقادیر حداقل و حداکثر رو پیدا کن. برایم یه مثال درمودش ببینیم:

```

const marks = [50, 20, 70, 60, 45, 30];
function findMin(arr) {
    let min = arr[0];
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
    return min;
}

function findMax(arr) {
    let max = arr[0];
    for (let i = 0; i < arr.length; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

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

```

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

سیمیکالن ; هس که نشون میده هیچ دستوری اجرا نمیشه، حتی اگه syntax جاواسکریپت به اون نیاز داشته باشه. از اونجایی که هیچ اقدامی با دستور خالی وجود نداره، ممکنه فکر کنیم که استفاده از اون خیلی کمه اما دستور خالی موقعی مفیده که میخواین یه حلقه ایجاد کنیم که بدنهاش خالیه. برای مثال، میتونیم یه آرایه با مقادیر صفر رو مثل کد زیر مقداردهی اولیه کنیم.

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

۲۴۸. چطوری metadata یه ماژول رو بدست میاري؟

میتونیم از آبجکت import.meta استفاده کنیم که یه ویژگی متعاهد که متا دادههای متنی خاص رو تو یه ماژول جاوا اسکریپت قرار می‌ده. این شامل اطلاعاتی در مورد ماژول فعلی، مانند URL ماژوله. در مرورگرها، ممکنه متا دادههای متفاوتی نسبت به NodeJS دریافت کنیم.

```
<script type="module" src="welcome-module.js"></script>

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

۲۴۹. عملگر comma چیه و چیکار میکنه؟

عملگر کاما برای ارزیابی هر یه از عملوندهاش از چپ به راست استفاده می‌شه و مقدار آخرین عملوند رو برمی‌گدونه. این کاملاً با استفاده از کاما در آرایه‌ها، اشیاء و آرگومان‌ها و پارامترهای تابع متفاوته. بریم به مثال در موردش ببینیم.

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

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

۲۵۰. مزایای استفاده از عملگر comma چیه؟

ممکن است که برای گنجوندن چن تا عبارت تو جایی که به یه عبارت واحد نیاز داره استفاده می‌شه. یکی از کاربردهای رایج این عملگر کاما، ارائه چندین پارامتر تو یه حلقه «for» اس. برای مثال، حلقه for زیر از چند عبارت تو یه مکان واحد با استفاده از عملگر کاما استفاده میکنه.

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

همچنین می‌توnim از عملگر کاما تو یه عبارت بازگشتی استفاده کنیم جایی که قبل از بازگشت پردازش میکنه.

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

۲۵۱. Typescript چیه؟

TypeScript یه ابر مجموعه تایپ شده از جاواسکریپت که توسط مایکروسافت ایجاد شده که انواع اختیاری، کلاس‌ها، async/wait و بسیاری ویژگی‌های دیگر رو اضافه میکنه و به جاواسکریپت ساده کامپایل میکنه. Angular به طور کامل در TypeScript ساخته شده و به عنوان زبان اصلی استفاده می‌شه. می‌توnim اونو به صورت گلوبال نصب کنیم:

```
npm install -g typescript
```

بیاین یه مثال ساده از استفاده از TypeScript رو ببینیم،

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

متده greeting فقط نوع رشته رو به عنوان آرگومان مجاز میکنه.

۲۵۲. تفاوت‌های بین javascript و typescript کدوما هستن؟

javascript	typescript	ویژگی
زبان اسکریپت	زبان برنامه نویسی شی گرا	پارادایم زبان
دارای تایپ پویا	پشتیبانی از تایپ استاتیک	پشتیبانی از تایپ
پشتیبانی نمی‌شده	پشتیبانی شده	ماژول‌ها
از رابطه‌ها پشتیبانی نمی‌کند	دارای مفهوم رابط	رابط
عدم پشتیبانی از پارامترهای اختیاری اختیاری برای توابع	توابع از پارامترهای اختیاری پشتیبانی می‌کنند	پارامترهای اختیاری

۲۵۳. مزایای javascript نسبت به typescript چیاست؟

۱. TypeScript 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 صحبت 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 ES4، ES3، ES5 کامپایل کنه.

۲۵۴. object-initializer چیه؟

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

```
const initObject = { a: "John", b: 50, c: {} };

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

۲۵۵. متد constructor چیه؟

متد `constructor` یه متد خاص برای ایجاد و مقداردهی اولیه یه آبجکت ایجاد شده تو یه کلاسه. اگه متد `constructor` رو مشخص نکنیم از `constructor` پیش فرض استفاده میشه. برمیم یه مثال در موردش ببینیم:

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

const employeeObject = new Employee();

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

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

تو یه کلاس یه متد خاصه و باید فقط یه بار تو یه کلاس تعریف شه. اگه متد سازنده رو بیش از یه بار تو یه کلاس بنویسیم، یه خطای `SyntaxError` ایجاد میشه.

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

const employeeObject = new Employee();

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

۲۵۷. چطوری متدها کلاس والد رو صدا بزنیم؟

می‌تونیم از کلمه کلیدی `super` برای فراخوانی constructor کلاس والد استفاده کنیم. یادمون باشه که `super` باید قبل از استفاده از مرجع `this` فراخوانی شه. در غیر این صورت باعث خطای Reference error 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؟

می‌تونیم از روش `Object.getPrototypeOf(obj)` برای برگرداندن آبجکت مشخص شده استفاده کنیم. یعنی مقدار ویژگی `prototype` داخلی. اگه هیچ ویژگی ارث وجود نداشته باشه، مقدار `null` برگردانده می‌شه.

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

۲۵۹. اگه به متدها پاس بدیم چی می‌شه؟

در ES5، اگه پارامتر `obj` یه آبجکت نباشه، یه استثنای `TypeError` ایجاد میکنه. در حالی که در ES2015، پارامتر به یه آبجکت اجباری تبدیل 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؟

می‌تونیم از متدهای `Object.setPrototypeOf` استفاده کنیم (یعنی ویژگی داخلی «Prototype») یه آبجکت مشخص شده رو روی یه آبجکت دیگه یا تهی تنظیم می‌کنیم. برای مثال، اگه بخوایم `Rectangle` رو روی آبجکت `Square` داشته باشیم تنظیم کنیم این شکلی میشه این کارو انجام داد:

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

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

متدهای `Object.isExtensible` برای تعیین اینکه یه آبجکت قابل توسعه هس یا نه (یعنی اینکه میتوانه ویژگی‌های جدیدی به اون اضافه شه یا نه) استفاده میشه.

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

نکته: به طور پیش فرض، همه ی آبجکت‌ها قابل گسترش هستن. برای مثال، ویژگی‌های جدید رو می‌تونیم اضافه یا تغییر بدیم.

۲۶۲. چطوری جلوی object رو بگیریم؟

متدهای `Object.preventExtensions` برای جلوگیری از افرودن ویژگی‌های جدید به یه آبجکت استفاده می‌شه. به عبارت دیگر، از پسوندهای بعدی به آبجکت جلوگیری می‌کنیم. بیان 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 غیرقابل extend چیه؟

می‌تونیم یه آبجکت غیر قابل گسترش رو به ۳ روش علامت گذاری کنیم.

۱. Object.preventExtensions .

۲. Object.seal .

۳. Object.freeze .

```

let newObject = {};

Object.preventExtensions(newObject); // Prevent objects are non-
extensible
Object.isExtensible(newObject); // false

let sealedObject = Object.seal({}); // Sealed objects are non-
extensible
Object.isExtensible(sealedObject); // false

let frozenObject = Object.freeze({}); // Frozen objects are non-
extensible
Object.isExtensible(frozenObject); // false

```

۲۶۴. چطوری های متعددی رو روی یه object تعریف می‌کنی؟

متدهای `Object.defineProperties` برای تعریف یا اصلاح ویژگی‌های موجود مستقیماً روی یه آبجکت و برگرداندن آبجکت استفاده می‌شه. بیاین چندین ویژگی رو روی یه آبجکت خالی تعریف کنیم:

```

const newObject = {};

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

```

۲۶۵. منظور از MEAN توی جاواسکریپت چیه؟

دسته MEAN (Node.js و MongoDB، Express، AngularJS) محبوب‌ترین دسته‌بندی فناوری نرم‌افزار جاواسکریپت منبع بازه که برای ساخت برنامه‌های وب پویا در دسترسه، جایی که می‌توانیم کدهای سمت سرور و سمت کلاینت پروژه وب رو کاملاً با جاواسکریپت بنویسیم.

۲۶۶. توی جاواسکریپت چیه و چیکار می‌کنه؟ Obfuscation

عمل عمده 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 اوно 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 می‌شه

```

eval(
  (function (p, a, c, k, e, d) {
    e = function (c) {
      return c;
    };
    if (!"".replace(/\^/, String)) {
      while (c--) {
        d[c] = k[c] || c;
      }
      k = [
        function (e) {
          return d[e];
        },
      ];
      e = function () {
        return "\w+";
      };
      c = 1;
    }
    while (c--) {
      if (k[c]) {
        p = p.replace(new RegExp(`\\b${e(c)}\\b`, "g"), k[c]);
      }
    }
    return p;
  ))(
    "2 1(){0.3('4, 7 6 5 8')}",
    9,
    9,
    "console|greeting|function|log|Hello|JS|to|welcome|world".split(""),
    0,
    {}
  );
);

```

۲۶۷. چه نیازی به Obfuscate کردن داریم؟

۱. اندازه کد کمتر میشه. که باعث انتقال سریع تر داده بین سرور و کلاینت 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. Minification چیه؟

حذف تمام کاراکترهای غیر ضروریه (فضاهای خالی حذف می‌شن) و Minification متغیرها بدون تغییر در عملکرد اون تغییر نام میدن همچنین برای مبهم سازی کد هم استفاده میشه.

۲۶۹. مزایای minification یا کم حجم سازی چیه؟

به طور معمول توصیه می‌شه برای ترافیک سنگین و نیازهای فشرده منابع از استفاده کنیم. 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؟

Encryption	Obfuscation	ویژگی
تغییر فرم اطلاعات به فرمت ناخوانا با استفاده از کلید	تغییر فرم هر داده به هر شکل دیگر	تعریف
برای رمزگشایی کلید لازمه	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
۲. UglifyJS2
۳. jsmin
۴. [/javascript-minifier.com](http://javascript-minifier.com)

۳۷۲. چطوری اعتبارسنجی فرم رو با javascript انجام میدی؟

میشه از جاوااسکریپت برای اعتبارسنجی فرم HTML استفاده کرد. برای مثال، اگه فیلد فرم خالی باشه، تابع باید اطلاع بده و false رو برگردونه تا از ارسال فرم جلوگیری شه. بریم 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 به این شکل هست.

```
function validateForm() {
    const 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؟

میتونیم بدون استفاده از جاوااسکریپت اعتبارسنجی فرم HTML رو به صورت خودکار انجام بدین. اعتبارسنجی با اعمال ویژگی required 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 برای IE9 یا ورژن‌های قبل از اون کار نمیکنه.

۳۷۴. متدهای موجود روی DOM برای اعتبارسنجی کدوما هستن؟

۱. اگه یه عنصر ورودی حاوی داده‌های معتبر باشه، مقدار true را برمنگردونه.
۲. برای تنظیم خاصیت validationMessage: عنصر ورودی setCustomValidity را استفاده می‌شه.

بیاین یه فرم ورود کاربر با اعتبارسنجی DOM بگیریم

```
function myFunction() {  
    let userName = document.getElementById("uname");  
    if (!userName.checkValidity()) {  
        document.getElementById("message").innerHTML =  
            userName.validationMessage;  
    } else {  
        document.getElementById("message").innerHTML =  
            "Entered a valid username";  
    }  
}
```

۳۷۵. مقادیر موجود روی DOM برای اعتبارسنجی کدوما هستن؟

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.

۳۷۶. مقادیر موجود روی input برای اعتبارسنجی کدوما هستن؟

۱. ویژگی validity: یه عنصر ورودی مجموعه ای از ویژگی‌های مربوط به اعتبارسنجی داده‌ها رو ارائه میده.
۲. customError: اگه یه پیام اعتبار سفارشی تنظیم شده باشه، true رو برمنگردونه.
۳. patternMismatch: اگه مقدار یه عنصر با ویژگی الگوی آن مطابقت نداشته باشه، مقدار true رو برمنگردونه.
۴. rangeOverflow: اگه مقدار یه عنصر از ویژگی max آن بیشتر باشه، مقدار true رو برمنگردونه.

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

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

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

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

۳۷۷. یه مثال از استفاده ویژگی rangeOverflow می‌تونی بزنی؟

اگه مقدار یه عنصر از ویژگی max اون بیشتر باشه، ویژگی rangeOverflow مقدار true رو برمی‌گردونه. برای مثال، توی فرم پایین اگه مقدار ورودی بیش از 100 باشه، خطای 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");
  }
}
```

۳۷۸. جاواسکریپت قابلیت استفاده از enum را پیش‌فرض توی خودش داره؟

نه، جاواسکریپت به صورت بومی از enum ها پشتیبانی نمیکنه. اما انواع مختلفی از راه حلها برای شبیه‌سازی اونا وجود داره، اگرچه ممکنه معادلهای دقیقی ارائه نکتن. برای مثال، می‌تونیم از seal یا freeze روی آبجکت استفاده کنیم:

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

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

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

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

میتونیم از متد Object.getOwnPropertyNames استفاده کنیم که آرایه‌ای از تمام ویژگی‌هایی رو که مستقیماً تو آبجکت داده شده یافت می‌شه رو برمی‌گردونه. بیاین استفاده از اونو تو یه مثال بینیم:

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

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

چطوری property-descriptor‌های یه آبجکت رو بدست بیاریم؟ .۲۸۱

میتونیم از متد Object.getOwnPropertyDescriptors استفاده کنیم که تمام توصیف‌گرهای ویژگی یه آبجکت معین رو برمی‌گردونه. بیاین استفاده از اونو توی یه مثال بینیم:

```

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 پرپرتبتی: value
۲. تعیین میکنه که آیا مقدار مرتبط با ویژگی قابل تغییر هس یا نه: writable
۳. اگه بتونیم نوع descriptor این ویژگی رو تغییر بدیم و اگه بتونیم ویژگی رو از آبجکت مربوطه حذف کنیم، مقدار true رو برمی‌گردونه: configurable
۴. تعیین میکنه که ویژگی در موقع شمارش خصوصیات روی آبجکت مربوطه ظاهر می‌شه یا نه: enumerable
۵. تابعی که به عنوان تنظیم کننده برای ویژگی عمل میکنه: set
۶. تابعی که به عنوان یه گیرنده برای ملک عمل میکنه: 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؟

برای تغییر url میشه از `window.location.url` استفاده کرد اما این کار باعث بارگیری دوباره صفحه میشه. 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، میتونیم از `pushState` مثل کد 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 مقدار خاصی رو بین ورودیهای خودش داره یا نه با برگرداندن `false` یا `true` استفاده میشه. بیاین مثالی برای پیدا کردن یه عنصر (عددی و رشته‌ای) تو یه آرایه ببینیم:

```

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

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

```

۲۸۶. چطوری آرایه‌های scalar را با هم مقایسه می‌کنی؟

می‌توانیم از `length` و هر روش آرایه برای مقایسه دو آرایه اسکالار (مقایسه مستقیم با استفاده از `==`) استفاده کنیم. ترکیب این دو تا روش می‌توانه نتیجه مورد نیازمان را به ما 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 بخوایم آرایه‌ها را بدون توجه به ترتیب مقایسه کنیم باید اونا را قبل از مقایسه، مرتب کنیم.

```
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 را از متدهای GET گرفت؟

کلاس `URL` رشته `url` را می‌پذیرد و از ویژگی `searchParams` این آبجکت می‌توانیم برای دسترسی به پارامترهای `get` استفاده کنیم. ممکنه برای دسترسی به `URL` در مرورگرهای قدیمی (از جمله IE) نیاز به استفاده از `polyfill` یا `window.location` داشته باشیم.

```
const urlString = "http://www.some-domain.com/about.html?
x=1&y=2&z=3"; // window.location.href
const url = new URL(urlString);
const parameterZ = url.searchParams.get("z");
console.log(parameterZ); // 3
```

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

می‌توانیم از متدهای `Number.prototype.toLocaleString()` استفاده کنیم که رشته‌ای رو با نمایشی حساس به زبان مثل جداول 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 نامرتبه هستند و هیچ ارتباطی بین اونا وجود ندارد. جاوا بصورت ایستاد تایپ می‌شه، کامپایل می‌شه، روی ماشین مجازی خودش اجرا می‌شه. در حالی که جاواسکریپت به صورت پویا تایپ می‌شه، تفسیر می‌شه و در محیط‌های مرورگر و nodejs اجرا می‌شه. بیاین تفاوت‌های عمدی رو در قالب جدولی ببینیم:

جاواسکریپت	جاوا	ویژگی
این یه زبان تایپ شده پویاس	یه زبان حساس به لحاظ تایپ دار بودنه	تایپ شده
برنامه نویسی مبتنی بر <code>prototype</code>	برنامه نویسی شی گرا	پارادایم
حدودده عملکردى	حدودده بلوک	حدودده
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 رو پشتیبانی میکنه؟

جاواسکریپت به طور پیش فرض از namespace پشتیبانی نمیکند. بنابراین اگه هر عنصری (تابع، متدهای، آبجکت، متغیر) ایجاد کنیم گلوبال می‌شه و namespace گلوبال رو آلووده میکنه. بیاین مثالی از تعریف دو تابع بدون namespace 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 namespace مشکل برخورد نام رو حل میکنه.

۲۹۱ چطوری namespace تعریف میکنی؟

حتی اگه جاواسکریپت فاقد namespace باشه، میتونیم از Objects، IIFE برای ایجاد namespace استفاده کنیم.

۱. استفاده از **Object literal**: بیاین متغیرها و توابع رو درون یه Object literal بپیچیم که به عنوان namespace عمل میکنه. پس از اون میتونیم با استفاده از نماد آبجکت به اونا دسترسی داشته باشیم

```

const namespaceOne = {
    function func1() {
        console.log("This is a first definition");
    }
}
const 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. استفاده از IIFE (**توابع بیانی بلافصله صدازده شده**): جفت پرانتز بیرونی IIFE محدوده محلی برای تمام کدهای داخلش ایجاد میکنه و تابع ناشناس رو به یه عبارت تابع تبدیل میکنه. به همین دلیل، میتونیم یه تابع رو در تو دو عبارت تابع مختلف ایجاد کنیم تا به عنوان namespace عمل کنه.

```

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

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

```

۳. استفاده کردن از بلوک و تعریف‌کننده let و const: در ES6، می‌توانیم از یه بلوک و یه اعلان let برای محدود کردن دامنه یه متغیر به یه بلوک استفاده کنیم.

```

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

// myFunction(): ReferenceError: myFunction is not defined.

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

// myFunction(): ReferenceError: myFunction is not defined.

```

۴۹۲. چطوری می‌توانیم تکه کد جاوا‌اسکریپت داخل یه iframe رو از صفحه والد صدا بزنیم؟

در ابتدا باید iFrame با استفاده از document.getElementById یا window.frames یا contentWindow به قابل دسترسی باشه. پس از اون ویژگی targetFunction دسترسی میده

```
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 timezone` رو از آجکت `date` بگیریم؟

می‌تونیم از روش `gettimezoneOffset` کلاس `Date` استفاده کنیم. این روش اختلاف منطقه زمانی رو بر حسب دقیقه از محلی فعلی (تنظیمات سیستم میزبان) به UTC برمی‌گرددونه.

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

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

می‌تونیم هر دو عنصر پیوند و اسکریپت رو توی DOM ایجاد کنیم و اوṇا رو به عنوان child به تگ `head` اضافه کنیم. بیاین یهتابع برای اضافه کردن منابع اسکریپت و سبک مثل زیر ایجاد کنیم:

```
function loadAssets(filename, filetype) {
  if (filetype == "css") {
    // External CSS file
    const fileReference = document.createElement("link");
    fileReference.setAttribute("rel", "stylesheet");
    fileReference.setAttribute("type", "text/css");
    fileReference.setAttribute("href", filename);
  } else if (filetype == "js") {
    // External JavaScript file
    const fileReference = document.createElement("script");
    fileReference.setAttribute("type", "text/javascript");
    fileReference.setAttribute("src", filename);
  }
  if (typeof fileReference !== "undefined")
    document.getElementsByTagName("head")
[0].appendChild(fileReference);
}
```

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

اگه بخوایم به هر عنصری در صفحه HTML دسترسی داشته باشیم، باید با دسترسی به آبجکت document شروع کنیم. بعداً می‌تونیم از یکی از روش‌های زیر برای پیدا کردن عنصر HTML استفاده کنیم.

۱. document.getElementById(id) : یه عنصر رو با id پیدا میکنه
۲. document.getElementsByTagName(name) : یه عنصر رو با اسم تگ پیدا میکنه
۳. document.getElementsByClassName(name) : یک عنصر رو با اسم کلاس پیدا میکنه

۱۹۶.jQuery چیه؟

jQuery یه کتابخونه جاواسکریپت متقابل مرورگر محبوبه که با به حداقل رسوندن اختلاف بین مرورگرها، پیمایش مدل آبجکت (DOM)، مدیریت رویداد، انیمیشن‌ها و تعاملات AJAX رو فراهم میکنه. با فلسفه اش «کمتر بنویس، بیشتر انجام بد» شهرت زیادی دارد. برای مثال، می‌توانیم پیام خوش آمدگویی رو موقع بارگذاری صفحه با استفاده از jQuery به صورت زیر نمایش بدین.

```
$(document).ready(() => {
  // It selects the document and apply the function on page load
  alert("Welcome to jQuery world");
});
```

نکته: می‌توانیم اونو از سایت رسمی jquery دانلود کنیم یا از CDN‌ها مثل گوگل نصب کنیم.

۱۹۷. موتور V8 جاواسکریپت چیه؟

V8 یه موتور جاواسکریپت با کارایی بالا منبع بازه که توسط مرورگر Google Chrome استفاده می‌شه و به زبان C++ نوشته شده است. توی پروژه node.js استفاده می‌شه. macOS و WebAssembly و ECMAScript رو پیاده‌سازی میکنه و روی ویندوز 7 یا بالاتر، +10.12 و سیستم‌های لینوکس که از پردازنده‌های ARM، IA-32، x64 یا MIPS استفاده

میکنن اجرا می‌شه.

نکته: میتونه به صورت مستقل اجرا شه یا میتونه در هر برنامه C++ تعییه شه.

۳۹۸. 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
```

۳۹۹. عملگر void چیکار میکنه؟

عملگر `void` عبارت داده شده رو ارزیابی میکنه و بعد تعریف نشده (یعنی بدون برگشتن 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  
</a>
```

نکته: این عملگر بیشتر برای بدست آوردن مقدار اولیه تعریف نشده با استفاده از `"void(0)"` استفاده می‌شه.

۴۰۰. 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 می‌شود

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

۳۰. چطوری می‌شه بی‌حلقه بی‌نهایت درست کرد؟

می‌توانیم حلقه‌های بی‌نهایت با استفاده از حلقه‌های `for` و `while` بدون استفاده از هیچ عبارتی ایجاد کنیم. ساختار یا `syntax` حلقه `for` از نظر ESLint و ابزارهای بهینه ساز کد، رویکرد بهتری برای این کار هست.

```
for (;;) {}
```

```
while (true) {}
```

۳۱. چرا باید در استفاده از عبارت `with` تجدیدنظر کرد؟

دستور `with` جاوااسکریپت در نظر گرفته شده بود که مختصراً برای نوشتن دسترسی‌های تکرارشونده به آبجکت 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 آبجکت 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 `with` به یه متغیر واقعی اشاره می‌کنه یا به یه ویژگی توی آرگومان.

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 ۰ و ۳ ۲ ۱ است

توضیح: با توجه به صفت رویداد/حلقه جاوااسکریپت، تابع 'setTimeout' بعد از اجرای حلقه فراخونی می‌شه. از اونجایی که متغیر `i` با کلمه کلیدی `var` تعریف می‌شه، به یه متغیر گلوبال تبدیل می‌شه و با استفاده از تکرار زمانی که تابع `setTimeout` فراخوانی می‌شه، مقدارش برابر با ۴. بنابراین، خروجی حلقه اول '۴ ۴ ۴' می‌شه. حالا توی حلقه دوم، متغیر `i` به عنوان کلمه کلیدی `let` تعریف می‌شه، به متغیری با محدوده بلوک تبدیل می‌شه و یه مقدار جدید (۳, ۰, ۱, ۲) برای هر تکرار داره. بنابراین، خروجی حلقه دوم «۰ ۱ ۲ ۳» می‌شه.

۳۰۴. می‌تونی یه سری از ویژگی‌های ES6 رو اسم ببری؟

۱. پشتیبانی از ثابت‌ها یا متغیرهای تغییرناپذیر
۲. پشتیبانی Block-scope برای متغیرها، ثابت‌ها و توابع
۳. Arrow functions
۴. پارامترهای پیش فرض
۵. پارامترهای Rest and Spread
۶. Template Literals
۷. Multi-line Strings
۸. Destructuring Assignment
۹. Enhanced Object Literals
۱۰. Promises
۱۱. Classes
۱۲. Modules

ششمین نسخه از زبان جاواسکریپت و در ژوئن 2015 منتشر شد. در ابتدا با نام ECMAScript 6 (ES6) شناخته شد و بعداً به ECMAScript 2015 تغییر نام داد. تقریباً همه مرورگرهای مدرن از ES6 پشتیبانی می‌کنند اما برای مرورگرهای قدیمی ترانسپایلرهای زیادی وجود داره، مثل Babel.js و غیره.

آیا می‌توانیم متغیرهای تعریف شده با `let` و `const` را مجدداً `declare` کنیم؟

نه، ما نمی‌توانیم متغیرهای `let` و `const` را مجدداً تعریف کنیم. اگه این کار رو انجام بدیم، یه خطای syntax دریافت می‌کنیم:

```
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 hoisting به گونه ای رفتار می‌شه که مثلاً در بالای محدوده محصور تعریف شده. پس همه اعلان‌های چندگانه بدون هیچ خطایی تو ایجاد یه متغیر hoisted مشترک نقش دارن. بیاین مثالی از اعلان مجدد متغیرها تو یه محدوده برای متغیرهای `var` و `const` بزنیم.

```
const 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 syntax 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 اونا رو میکنه؟

نه، متغیر **const** مقدار را تغییرنایذیر نمیکنه. اما تخصیص‌های بعدی را مجاز نمی‌دونه (یعنی می‌توانیم با ایجاد متغیر اعلام کنیم اما بعداً نمی‌توانیم مقدار دیگه ای را یا نوع اون متغیر را عوض کنیم)

```

const userList = [];
userList.push("John"); // Can mutate even though it can't re-
assign
console.log(userList); // ['John']
uerList = 123; // throws an error

```

۳۰۸. پیش‌فرض چی هستن؟ parameter

در ES5، برای مدیریت مقادیر پیش‌فرض پارامترهای تابع، باید به عملگرهای OR منطقی وابسته باشیم. ولی در ES6، ویژگی پارامترهای تابع پیش‌فرض اجازه میده تا پارامترها مقادیر پیش‌فرض مقداردهی اولیه بشن، اگه مقداری یا تعریف‌نشده ارسال نشه. بیاین رفتار رو با یه مثال مقایسه کنیم:

```

// ES5
const calculateArea = function (height, width) {
  height = height || 50;
  width = width || 60;

  return width * height;
};
console.log(calculateArea()); // 300

```

پارامترهای پیش‌فرض، مقداردهی اولیه رو ساده‌تر میکنه،

```
// ES6
const 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 تکی با کاراکتر بک تیک (`) محصور میشن در E6، این ویژگی استفاده از عبارات پویا رو به شرح زیر امکان پذیر میکنه.

```
const greeting = `Welcome to JS World, Mr. ${firstName}
${lastName}.`;
```

توی ES5 لازمه که ما کدمون رو این شکلی بنویسیم.

```
var greeting = 'Welcome to JS World, Mr. ' + firstName + ' ' +
lastName.`
```

نکته: میتونیم از رشته‌های چند خطی و ویژگی‌های درونیابی رشته‌ای با الفبای الگو استفاده کنیم.

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 کنیم.

```
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 به ویژگیه که در نحو تحت اللفظی الگو پشتیبانی می‌شده تا امکان بکتیک‌های درونی تو یه مکان‌نمای \${ } را در قالب فراهم کنه. برای مثال، الگوی تودرتو زیر برای نمایش نمادها بر اساس مجوزهای کاربر استفاده می‌شده، در حالی که الگوی بیرونی نوع پلت فرم رو بررسی می‌کنه.

```
const iconStyles = `icon ${  
  isMobilePlatform()  
  ? ""  
  : `icon-${user.isAuthorized ? "submit" : "disabled"}`  
};
```

می‌تونیم 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 بهمون اجازه میدن تا کلمات قالب رو با یه تابع تجزیه کنیم. تابع تگ اولین پارامتر رو به عنوان آرایه‌ای از رشته‌ها و پارامترهای باقی مانده رو به عنوان strings می‌گیره. این تابع همچنین می‌تونه رشته‌های دستکاری شده رو بر اساس پارامترها برگردانه.

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

```

۳.۱۳. رشته‌های خام چی هستن؟

ES6 با استفاده از روش «String.raw» ویژگی رشته‌های خام رو ارائه میکنه که برای دریافت شکل رشته خام رشته‌های الگو استفاده میشه. این ویژگی به ما این امکان رو میده تا به رشته‌های خام همونطور که وارد شده‌اند، بدون پردازش دنباله‌های فرار دسترسی داشته باشیم. بریم یه مثال در موردش ببینیم،

```
const 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 نمی‌کنیم دنباله کاراکترهای خط جدید با نمایش خروجی در چندین خط پردازش می‌شه.

```
const 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 کردن با **assign** و **destructuring** چیه و چطوری انجام می‌شه؟

تخصیص **destructuring** به عبارت جاواسکریپتیه 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

```
const [one, two, three] = ["JAN", "FEB", "MARCH"];  
  
console.log(one); // "JAN"  
console.log(two); // "FEB"  
console.log(three); // "MARCH"
```

و می‌تونیم ویژگی‌های کاربر یه آبجکت رو با استفاده از انتساب **destructuring** به دست بیارین،

```
const { name, age } = { name: "John", age: 32 };  
  
console.log(name); // John  
console.log(age); // 32
```

۳۱۵ موقع assign چطوری می‌شه destructuring کردن با 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 کنیم. بیاین برای هر دو آرایه و موارد استفاده از شی مثالی بزنیم،

:Arrays destructuring

```
let x;  
let y;  
let z;  
  
[x = 2, y = 4, z = 6] = [10];  
console.log(x); // 10  
console.log(y); // 4  
console.log(z); // 6
```

:Objects destructuring

```
const { 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 نمی‌کنیم تعویض دو مقدار به يه متغیر وقت نیاز دارد. در حالی که با استفاده از يه ویژگی ساختارشکن، دو مقدار متغیر رو می‌شه تو یه عبارت ساختار شکن جایگزین کرد. بیاین دو متغیر عددی رو در انتساب ساختارزدایی آرایه با هم عوض کنیم:

```
let x = 10;  
let y = 20;  
  
[x, y] = [y, x];  
console.log(x); // 20  
console.log(y); // 10
```

۳۱۷ Enhanced-object-literal ها 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;
var y = 20;
obj = { x, y };
console.log(obj); // {x: 10, y:20}
// ES5
var x = 10;
var y = 20;
obj = { x, y };
console.log(obj); // {x: 10, y:20}
```

۳۱۸ import های داینامیک چی هستن؟

واردات پویا با استفاده از نحو تابع «import» به ما اجازه میده تا مازول‌ها 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>) هستش. مزیت اصلی واردات پویا کاهش اندازه بسته‌ها‌ی ما، پاسخ حجم‌با ربار درخواست‌های ما و بهبود کلی در تجربه کاربر است.

```
import("./Module").then((Module) => Module.method());
```

۳۱۹ کاربرد import های داینامیک چیه؟

در زیر بعضی از موارد استفاده از واردات پویا نسبت به واردات استاتیک آورده شده است.
۱. یه مازول رو به صورت درخواستی یا مشروط وارد کنیم. برای مثال، اگه می‌خواین یه polyfill رو در مرورگر قدیمی بارگذاری کنیم

```
if (isLegacyBrowser()) {
  import(...)
  .then(...);
}
```

۲. تعیین کننده مازول رو در زمان اجرا محاسبه کنیم. برای مثال می‌توانیم از اون برای گلوبال سازی استفاده کنیم.

```
import(`messages_${getLocale()}.js`).then(...);
```

۳. یه مازول رو از داخل یه اسکریپت معمولی به جای یه مازول وارد کنیم.

۳۲۰. آرایه‌های نوع دار (typed-arrays) چیه؟

آرایه‌های تایپ شده آبجکت‌هایی آرایه مانند از API 6 ECMAScript برای مدیریت داده‌های باینری هستن. جاواسکریپت ۸ نوع آرایه تایپ شده رو ارائه میده،

۱. Int8Array: آرایه ای از اعداد صحیح امضا شده ۸ بیتی

۲. Int16Array: آرایه ای از اعداد صحیح امضا شده ۱۶ بیتی

۳. Int32Array: آرایه ای از اعداد صحیح امضا شده ۳۲ بیتی

۴. Uint8Array: آرایه ای از اعداد صحیح بدون علامت ۸ بیتی

۵. Uint16Array: آرایه ای از اعداد صحیح بدون علامت ۱۶ بیتی

۶. Uint32Array: آرایه ای از اعداد صحیح بدون علامت ۳۲ بیتی

۷. Float32Array: آرایه ای از اعداد ممیز شناور ۳۲ بیتی

۸. Float64Array: آرایه ای از اعداد ممیز شناور ۶۴ بیتی

برای مثال، می‌توانیم یه آرایه از اعداد صحیح امضا شده ۸ بیتی مانند زیر ایجاد کنیم

```
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

۳۲۲ چیه؟ collation

برای مرتب سازی مجموعه‌ی رشته‌ها و جستجو در مجموعه‌ای از رشته‌ها استفاده می‌شود. این پارامتر توسط محلی و از Unicode آگاهه. بیان ویژگی‌های مقایسه و مرتب سازی را در نظر بگیرید.

:Comparison .۱

```
const list = ["ä", "a", "z"]; // In German, "ä" sorts with "a"  
Whereas in Swedish, "ä" sorts after "z"  
const l10nDE = new Intl.Collator("de");  
const l10nSV = new Intl.Collator("sv");  
console.log(l10nDE.compare("ä", "z") === -1); // true  
console.log(l10nSV.compare("ä", "z") === +1); // true
```

:Sorting .۲

```
const list = ["ä", "a", "z"]; // In German, "ä" sorts with "a"  
Whereas in Swedish, "ä" sorts after "z"  
const l10nDE = new Intl.Collator("de");  
const l10nSV = new Intl.Collator("sv");  
console.log(list.sort(l10nDE.compare)); // [ "a", "ä", "z" ]  
console.log(list.sort(l10nSV.compare)); // [ "a", "z", "ä" ]
```

۳۲۳ چیه؟ for...of عبارت

دستور for...of یه حلقه تکرار بر روی اشیاء یا عناصر قابل تکرار مثل رشته داخلی، آرایه، اشیاء آرایه مانند (مثل آرگومان‌ها یا NodeList، Map، Set و TypedArray) و تکرارهای تعریف شده توسط کاربر ایجاد میکنه.

```
const arrayIterable = [10, 20, 30, 40, 50];  
  
for (let value of arrayIterable) {  
    value++;  
    console.log(value); // 11 21 31 41 51  
}
```

۳۲۴. خروجی spread عملگر روی آرایه زیر چیه؟

```
[... "John Resig"];
```

خروجی آرایه ['g', 'J'] است.

توضیح: رشته یه نوع تکرارپذیره و عملگر spread تو یه آرایه هر کاراکتر به 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 باشه، postMessages رو میشه بسیار امن در نظر گرفت. اما اگه بخواهیم بیام رو بدون تأیید منبع آن ارسال یا دریافت کنیم حملات اسکریپت بین سایتی ایجاد میشه.

۳۲۶. مشکلات استفاده از wildcard روی origin با postmessage چیه؟

آرگومان دوم متده postMessage مشخص میکنه که کدوم مبدا مجاز به دریافت پیامه. اگه از علامت "*" به عنوان آرگومان استفاده کنیم 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، مبدا 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 هکرها 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. میتونیم با اعتبارسنجی مبدأ پیام در انتهای گیرنده با استفاده از ویژگی "message.origin" از این مشکل جلوگیری کنیم. برای مثال، اجازه بدین مبدأ

فرستنده www.some-] در سمت گیرنده http://www.some-sender.com (receiver.com-، www.some) [receiver.com را بررسی کنیم

```
//Listener on http://www.some-receiver.com/
window.addEventListener("message", function(message){
  if(/^http://www\.some-sender\.com$/.test(message.origin)){
    console.log('You received the data from valid sender',
    message.data);
  }
});
```

۳۲۸. می‌توانیم کلا postMessage را غیرفعال کنیم؟

نمی‌توانیم به طور کامل (یا ۱۰۰٪) از postMessages استفاده نکنیم. حتی اگه برنامه‌مون با توجه به خطرات نمی‌کنه، بسیاری از اسکریپت‌های شخص ثالث از postMessage برای برقراری ارتباط با سرویس شخص ثالث استفاده می‌کنن. بنابراین ممکنه برنامه بدون اطلاع شما از postMessage استفاده کنه.

۳۲۹. آیا synchronous postMessage را به صورت همزمان کار می‌کنن؟

در مرورگر IE8 هستن اما در IE9 و سایر مرورگرهای مدرن دیگر (یعنی IE9+, Firefox, Chrome, Safari) synchronous postMessages برگردانده می‌شه، از مکانیزم asynchronous callback این رفتار رفته 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.

۱. تفاوت‌های بین جاواسکریپت داخلی و خارجی چیه؟

جاواسکریپت داخلی: کد منبع درون تگ اسکریپته.

جاواسکریپت خارجی: کد منبع تو یه فایل خارجی (ذخیره شده با پسوند js) ذخیره می‌شه و در تگ ارجاع می‌شه.

۲. آیا جاواسکریپت سریعتر از اسکریپت‌های سمت سرور است؟

بله، جاواسکریپت سریعتر از اسکریپت سمت سروره. از اونجایی که جاواسکریپت یه اسکریپت سمت کلاینته برای محاسبات یا محاسبات خودش به کمک سرور وب نیازی نداره. بنابراین جاواسکریپت همیشه سریعتر از هر اسکریپت سمت سرور مانند ASP، PHP و غیره اس.

۳. چطوری وضعیت چک بودن یه checkbox رو بدست بیاریم؟

می‌تونیم ویژگی checked رو در کادر انتخاب شده در DOM اعمال کنیم. 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 جاواسکریپت به صورت زیر در دسترس قرار داد:

```
<input type="checkbox" name="checkboxname" value="Agree" />
Agree the
conditions<br />
```

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

۴. هدف از عملگر double-tilde چیه؟

عملگر دابل tilde (~) به عنوان عملگر bitwise double NOT 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 می‌تونیم از متدهای استفاده کنیم. برای مثال، بیاین کد ASCII رو برای `String.prototype.charCodeAt` حرف اول رشته «ABC» پیدا کنیم:

```
"ABC".charCodeAt(0); // returns 65
```

در حالی که روش `String.fromCharCode` اعداد رو به کاراکترهای ASCII برابر تبدیل می‌کنه.

```
String.fromCharCode(65, 66, 67); // returns 'ABC'
```

۳۳۶. `ArrayBuffer` چیه؟

یه گلاس `ArrayBuffer` برای نشون دادن یه بافر داده باینری خام عمومی با طول ثابت استفاده می‌شه. می‌تونیم اونو به صورت زیر ایجاد کنیم:

```
const buffer = new ArrayBuffer(16); // create a buffer of length 16
alert(buffer.byteLength); // 16
```

برای دستکاری یه `ArrayBuffer`، باید از یه کلاس `DataView` استفاده کنیم.

```
// Create a DataView referring to the buffer
const view = new DataView(buffer);
```

۳۳۷. خروجی کد زیرچی خواهد بود؟

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

خروجی عبارت بالا "W" هه.

توضیح: نماد براکت با شاخص خاص روی یه رشته کاراکتر رو تو یه مکان خاص برمی‌گردونه. از این رو، کاراکتر "W" رشته ر و برمی‌گردونه. از اونجایی که این مورد در

نسخه‌های IE7 و پایین‌تر پشتیبانی نمی‌شود، ممکنه لازم باشه از متده `charAt` برای به دست آوردن نتیجه دلخواه استفاده کنیم.

۳۳۸. هدف از Error-object چیه؟

کلاس Error یه آبجکت error ایجاد میکنه و نمونه‌هایی از آبجکت‌های خطا موقع رخ دادن خطاهای زمان اجرا ارسال میشون، آبجکت Error همچنین میتوانه به عنوان یه آبجکت پایه برای استثناهای تعریف شده توسط کاربر استفاده شه. برای مثال

```
new Error([message[, fileName[, lineNumber]]])
```

می‌تونیم استثناهای تعریف شده توسط کاربر رو با استفاده از آبجکت Error در بلوک try...catch مثل کد زیر ارسال کنیم.

```
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؟

آبجکت EvalError یه خطا راجع به استفاده از تابع eval رو نشون میده. البته دیگه این استثنای توسط جاوااسکریپت ایجاد نمی‌شود و آبجکت EvalError فقط برای سازگاری با نسخه‌های باقی مونده. برای مثال

```
new EvalError([message[, fileName[, lineNumber]]])
```

می‌تونیم EvalError رو با بلوک try...catch مثل کد زیر ارسال کنیم.

```
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 میکنیم، syntax، بعضی از موارد زیر قبل از اجرای اسکریپت یه `SyntaxError` ایجاد میکنن
۱. وقتی از دستور `Octal` استفاده میکنیم

```
const n = 022;
```

۲. استفاده از عبارت `with`.

۳. وقتی از عملگر حذف روی نام متغیر استفاده میکنیم

۴. استفاده از `eval` یا آرگومانها به عنوان متغیر یا نام آرگومان تابع

۵. موقعی که از کلمات کلیدی رزرو شده جدید استفاده میکنیم

۶. موقعی که یه تابع رو تو یه بلوک اعلام میکنیم

```
if (someCondition) {  
  function f() {}  
}
```

از این رو، خطاهای موارد بالا برای جلوگیری از خطا در محیطهای توسعه/تولید مفید هستن.

۳۴۱. آیا همه `prototype` دارای `object` هستن؟

خیر. همه ها دارای `prototype` 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؟

نام متغیر تعریف یه تابعه در حالی که یه `parameter` نشون دهنده مقدار داده شده به تابع موقع فراخونای 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 متدهای some را آرایه‌ها چیه؟

متدهای `some` برای تست این که حداقل یه عنصر در آرایه، از تست پیاده‌سازی شده توسط تابع ارائه شده، عبور میکنند یا نه استفاده می‌شوند. متدهای `some` مقدار بولین برمی‌گردونند. بیاین مثالی بزنیم تا هر عنصر رو بر اساس داشتن عدد زوج آزمایش کنیم:

```
const array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  
  
const 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؟

متدهای `concat` برای ترکیب دو یا چند آرایه با برگرداندن یه آرایه جدید حاوی تمام عناصر استفاده می‌شوند. مثال:

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

بیاین مثالی از الحاق آرایه با آرایه‌های `fruits` و `veggies` رو مثال بزنیم.

```
const veggies = ["Tomato", "Carrot", "Cabbage"];  
const fruits = ["Apple", "Orange", "Pears"];  
const veggiesAndFruits = veggies.concat(fruits);  
console.log(veggiesAndFruits); // Tomato, Carrot, Cabbage,  
Apple, Orange, Pears
```

۳۴۵. تفاوت‌های بین Deep و Shallow کپی چیه؟

کپی سطحی:

کپی کم عمق یه کپی بیتی از یه آبجکته. یه آبجکت جدید ایجاد می‌شه که یه کپی دقیق از مقادیر موجود در آبجکت اصلی رو داره. اگه هر یه از فیلدهای آبجکت ارجاع به آبجکت‌های دیگه باشه، فقط آدرس‌های مرجع کپی می‌شن یعنی فقط آدرس توی حافظه کپی می‌شه.

مثال

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

برای ایجاد یه نسخه تکراری

```
const 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` رو تغییر میده، چون ما یه کپی کم عمق داریم. یعنی ما داده‌های اصلی رو هم از دست می‌دیم.

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 می‌یافته که یه آبجکت همراه با پرایپری‌هایی که به اون اشاره داره کپی شه.

Example

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

یه کپی عمیق با استفاده از خواص از آبجکت اصلی در متغیر جدید ایجاد کنیم:

```
const empDetailsDeepCopy = {
  name: empDetails.name,
  age: empDetails.age,
  expertise: empDetails.expertise,
};
```

اگه و نه `empDetailsDeepCopy` رو تغییر بدین، فقط `empDetailsDeepCopy.name` را تحت تأثیر قرار میده.
`empDetails`

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 اون فراخوانی شده و به هم پیوسته شدن. یادتون باشه که این روش به مشخصات ECMAScript 2015 اضافه شده است. بیاین یه مثال از رشته `Hello` رو برای تکرارش ۴ بار در نظر بگیریم:

```
'Hello'.repeat(4); // 'HelloHelloHelloHello'
```

۳۴۷. چطوری همه string های regular-expression شده با یه match بگردانیم؟

متدهای `matchAll` می‌تواند برای برگرداندن یه تکرارکننده از تمام نتایجی که یه رشته با یه عبارت منظم مطابقت دارن، استفاده شه. 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 prototype` رشته برای برش دادن دو طرف یه رشته استفاده می‌شه. اما اگه بخوایم بهخصوص در ابتدا یا انتهای رشته رو برش بدیم، می‌تونیم از روش‌های

استفاده کنیم. بیاین نمونه ای از `trimEnd/trimRight` و `trimStart/trimLeft` این روش‌ها رو در پیام greeting ببینیم:

```
const 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؟

```
console.log(+ "Hello");
```

خروجی دستور `log` کنسول بالا NaN را برمی‌گردونه. از اونجا که عنصر توسط عملگر unary پیشونده و مفسر جاواسکریپت سعی می‌کنده اون 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 جاواسکریپت از mixin‌ها استفاده می‌کنه؟

- یک اصطلاح برنامه نویسی شی گرا عمومیه: کلاسی که شامل متدهایی برای کلاس‌های دیگه اس. برخی از زبان‌های دیگر به ارث بردن چندگانه اجازه میدن. جاواسکریپت از وراثت چندگانه پشتیبانی نمی‌کنه، اما mixin‌ها رو می‌شه با کپی کردن متدها در نمونه اولیه پیاده سازی کرد.

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 می‌کنیم مقدار رو میده. برای مثال، از اون استفاده می‌شه که الان اجرا نشه، اما زمانی در آینده اجرا می‌شه. بیاین یه مثال sync بگیریم:

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

۳۵۲. **چیکار می‌کنن؟ thunk های asynchronous**

برای ایجاد درخواست‌های شبکه مفید هستن. بیان نمونه **Thunk asynchronous** ای از درخواست‌های شبکه رو بینیم:

```
function fetchData(fn) {  
  fetch("https://jsonplaceholder.typicode.com/todos/1")  
    .then((response) => response.json())  
    .then((json) => fn(json));  
}  
  
const asyncThunk = function () {  
  return fetchData((data) => {  
    console.log(data);  
  });  
};  
  
asyncThunk();
```

تابع **getData** فوراً فراخونی نمی‌شه و تنها زمانی فراخونی می‌شه که داده‌ها از نقطه پایانی API در دسترس باشن. تابع **setTimeout** هم برای ناهمزمان کردن کد ما استفاده می‌شه. بهترین مثال زمان واقعی، کتابخونه مدیریت حالت **redux** هس که از **thunk** ناهمزمان برای به تاخیر انداختن اعمال برای ارسال استفاده می‌کنه.

۳۵۳. **خروجی فراخوانی‌های توابع زیر چی می‌شه؟**

قسمت کد:

```

const circle = {
  radius: 20,
  diameter() {
    return this.radius * 2;
  },
  perimeter: () => 2 * Math.PI * this.radius,
};

console.log(circle.diameter());
console.log(circle.perimeter());

```

خروجی:

خروجی 40 و NaN هس. یادتون باشه که diameter یه تابع منظمه در حالی که مقدار arrow function یه `this` یه تابع معمولی (یعنی `diameter` به محدوده اطراف که یه کلاسه (یعنی آبجکت `circle`) اشاره داره. در حالی که این کلمه کلیدی تابع `perimeter` به محدوده اطراف که یه آبجکت `window` هس اشاره داره. از اونجایی که هیچ ویژگی `radius` در آبجکتهاي `window` وجود نداره ، یه مقدار `undefined` برمی‌گدونه و ضرب مقدار، مقدار `NaN` رو برمی‌گدونه.

۳۵۴. چطوری همه خطوط جدید رو از یه رشته حذف کرد؟

ساده ترین روش استفاده از `regex` برای شناسایی و جایگزینی خطوط جدید توی رشته اس. در این حالت از تابع `replace` به همراه رشته برای جایگزینی استفاده می‌کنیم که در این مثال یه رشته خالیه:

```

function remove_linebreaks( var message ) {
  return message.replace( /[\r\n]+/gm, "" );
}

```

تو عبارت بالا `g` و `m` برای flag‌های سراسری و چند خطی هستن.

۳۵۵. تفاوت بین `repaint` و `reflow` چیه؟

repaint زمانی اتفاق می‌افته که تغییراتی ایجاد می‌شه که روی دید یه عنصر تأثیر می‌داره، اما روی طرح اون تأثیر نمی‌داره. نمونه‌هایی از این موارد شامل طرح کلی، نمایان بودن یا رنگ پس زمینه اس. یه `reflow` شامل تغییراتیه که بر طرح بندی بخشی از صفحه (یا کل صفحه) تأثیر می‌زاره. تغییر اندازه پنجره مرورگر، تغییر فونت، تغییر محتوا (مانند تایپ متن

توسط کاربر)، استفاده از روش‌های جاواسکریپت شامل سیک‌های محاسبه شده، اضافه کردن 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 اون توی DOM هس می‌شه.

۳۵۶. اگه قبل از یه آرایه عملگر نفی «!» بزاریم چی می‌شه؟

نفی یه آرایه با کاراکتر «!»، آرایه رو به یه بولین تبدیل میکنه. از اونجایی که آرایه‌ها 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 کنیم 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 قبل از مقادیر falsy قرار بدیم چی می‌شه؟

اگه عملگر (+) روی مقادیر نادرست (null, undefined, NaN, false) additive (+) قرار بدیم، مقدار عددی صفر تبدیل می‌شه. بیاین اونا رو توی 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 می‌توانیم رشته self را تولید کنیم؟

روشته self رو می‌شه با ترکیب کاراکترهای [() ! +] تشکیل داد. برای رسیدن به این الگو باید موارد زیر را بدونیم:

۱. از اونچایی که آرایه‌ها مقادیر true هستن، با نفی آرایه‌ها false تولید می‌شه: ![]

false ===

۲. طبق قوانین اجباری جواوسکریپت، اضافه کردن آرایه‌ها به هم اونا رو به رشته‌بندی تبدیل می‌کنی: `"" == [] + []`

۳. Prepend یه آرایه با عملگر + یه آرایه رو به نادرست تبدیل میکنه، انکار اونو درست میکنه و در نهایت تبدیل نتیجه مقدار '1' رو تولید میکنه: +(!)(+) == 1 اعمال قوانین بالا می‌توانیم شرایط زیر رو استخراج کنیم:

```
( ! [ ] + [ ] === `false${!+[]}` ) === 1;
```

اکنون الگوی کاراکتر به صورت زیر ایجاد می‌شود:

```
s          e          l          f
~~~~~  
~~~~~  
~~~~~  
~~~~~  
  
(![] + []) [3] + (![] + []) [4] + (![] + []) [2] + (![] + []) [0]  
~~~~~  
~~~~~  
~~~~~  
~~~~~  
  
(![] + []) [+!+[ ]+!+[ ]+!+[ ]]+  
(![] + []) [+!+[ ]+!+[ ]+!+[ ]+!+[ ]]+  
(![] + []) [+!+[ ]+!+[ ]]+  
(![] + []) [+[]]  
~~~~~  
~~~~~  
  
(![]+[ ]) [+!+[ ]+!+[ ]+!+[ ]) + (![]+[ ]) [+!+[ ]+!+[ ]+!+[ ]+!+[ ]) + (![]+[ ]) +  
[!+[ ]+!+[ ]]+(![]+[ ]) [+!+[ ]]
```

.۳۶۰. چطوری می‌توانیم مقادیر falsy را از آرایه حذف کنیم؟

می‌توانیم با وارد کردن Boolean به عنوان پارامتر، روش فیلتر روی آرایه اعمال کنیم. به این ترتیب تمام مقادیر falsy (0, undefined, null, false) از آرایه حذف می‌شوند.

```
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؟

می‌تونیم مقادیر یونیک بـه آرایه رو با ترکیب دستور "Set" و (...)rest expression/spread دریافت کنیم.

```
console.log([...new Set([1, 2, 4, 4, 3])]); // [1, 2, 4, 3]
```

۳۶۲. چطوری همزمان با destructuring aliasهای کار می‌کنیم؟

کاهی اوقات لازم داریم یه متغیر `destructure` شده با نام متفاوت از نام ویژگی داشته باشیم. در این صورت، از یه `newName` برای تعیین اسم برای متغیر استفاده می‌کنیم. این فرآیند `destructuring alias` 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 متد map پیمایش کنیم؟

می‌تونیم مقادیر آرایه‌ها رو بدون استفاده از متد `map` تنها با استفاده از روش آرایه `from` ترسیم کنیم. 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 `length` آرایه می‌تونیم به سرعت یه آرایه رو خالی کنیم:

```
const cities = ["Singapore", "Delhi", "London"];
cities.length = 0; // cities becomes []
```

۳۶۵. چطوری اعداد رو با تعداد رقم اعشار مشخص رند می‌کنی؟

می‌تونیم با استفاده از روش `toFixed` از جاوااسکریپت، 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؟

می‌تونیم با استفاده از عملگر `(...)` یه آرایه رو به یه آبجکت با همون داده تبدیل کنیم.

```
const fruits = ["banana", "apple", "orange", "watermelon"];
const fruitsObject = { ...fruits };
console.log(fruitsObject); // {0: "banana", 1: "apple", 2: "orange", 3: "watermelon"}
```

۳۶۷. چطوری یه آرایه با یه سری داده درست کنیم؟

می‌تونیم با استفاده از روش `fill` یه آرایه با مقداری داده یا یه آرایه با همون مقدار ایجاد کنیم.

```
const 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 آبجکت رو می‌گیرد،
۲. ۵% - یه رشته می‌گیره،
۳. ۵% - برای اعشار یا عدد صحیح استفاده می‌شه
این متغیرها رو می‌شه توی `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، می‌تونیم سبک‌های CSS رو برای پیام‌های کنسول مشابه متن HTML در صفحه وب اعمال کنیم.

```
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 آبجکت جاواسکریپت `console.dir` مشخص شده به عنوان JSON استفاده می‌شه.

```
const user = { name: "John", id: 1, city: "Delhi" };
console.dir(user);
```

آبجکت user نمایش داده شده توی حالت JSON

```
>     const user = { "name": "John", "id": 1, "city": "Delhi" };
      console.dir(user);
▼ Object i
  name: "John"
  id: 1
  city: "Delhi"
► __proto__: Object
```

۳۷۱. آیا میشه المنهای HTML رو توی console دیباگ کرد؟

بله، دریافت و اشکال زدایی عناصر HTML توی کنسول، درست مثل بررسی عناصر، امکان پذیره.

```
const element = document.getElementsByTagName("body")[0];
console.log(element);
```

این عنصر HTML رو توی 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 توی کنسول توی یه قالب جدولی برای تجسم آرایه‌ها یا آبجکت‌های 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 می‌شون،

```

>     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 `isNaN` و `isFinite` برای تأیید عدد بودن یا نبودن آرگومان استفاده می‌شون.

```

function isNumber(n) {
  return !isNaN(parseFloat(n)) && isFinite(n);
}

```

۳۷۴. چطوری یه متن رو می‌تونیم به clipboard کپی کنیم؟

ما باید محتوا (با استفاده از روش `select.`) عنصر ورودی رو انتخاب کنیم و دستور `copy` رو با `execCommand('copy')` اجرا کنیم (یعنی `execCommand('copy')`). همچنین می‌توانیم سایر دستورات سیستم مثل `cut` و `paste` رو اجرا کنیم.

```

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؟

می‌تونیم از `Date.getTime` برای دریافت مهر زمانی فعلی استفاده کنیم. یه میانبر جایگزین برای دریافت مقدار وجود داره.

```
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 کاربرد نداره.

```
const biDimensionalArr = [11, [22, 33], [44, 55], [66, 77], 88, 99];  
const flattenArr = [...biDimensionalArr]; // [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

اما ما می‌تونیم اونو با آرایه‌های چند بعدی با فراخوانی‌های `recursive` کال کنیم:

```
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؟

می‌تونیم از تابع `indexOf` « برای مقایسه ورودی با چندین مقدار به جای بررسی هر مقدار به عنوان یه شرط استفاده کنیم.

```
// 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 کاربرها در مورد از دست دادن داده‌های فعلی می‌تواند استفاده شود. `syntax` تعریف کردنش به صورت زیر است:

```
window.onbeforeunload = function () {
  alert("Your 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 تگ `body` غیرفعال کرد.

```
<body oncontextmenu="return false;"></body>
```

۳۸۰. چی هستن `object-wrapper`؟

مقادیر اولیه مانند رشته، عدد و بولین پرایپری و متدهای ندارن، اما زمانی که می‌خوایم کارهایی رو روی اونا انجام بدیم، به طور موقت به یه آبجکت (Wrapper) تبدیل

می‌شن. برای مثال، اگه متده `UpperCase` رو روی یه مقدار رشته اولیه اعمال کنیم خطایی ایجاد نمیکنه، اما حروف بزرگ رشته رو برمی‌گردونه.

```
const 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 `Asynchronous JavaScript XML` هس و گروهی از فناوری‌های مرتبط با `HTML`, `CSS`, `JavaScript`, `XMLHttpRequest API` (و غیره) که برای نمایش داده‌ها به صورت ناهمزمان استفاده می‌شه. یعنی ما می‌توانیم داده‌ها رو به سرور ارسال کنیم و بدون بارگیری مجدد صفحه وب، داده‌ها رو از سرور دریافت کنیم.

۳۸۲. روش‌های مختلف مدیریت یه کد `Asynchronous` چیه؟

۱. `callback` ها
۲. `Promise` ها
۳. `Async/await`
۴. کتابخونه‌های شخص ثالث مانند `async.js`, `bluebird` و غیره

۳۸۳. چطوری یه درخواست `fetch` رو کنسل کنیم؟

یکی از ضعف `Promise` ها اینه که `native` راه مستقیمی برای لغو درخواست `fetch` نیست. اما `AbortController` جدید از مشخصات `js` به ما امکان میده که از یه سیگنال واسه لغو یک یا چند درخواست `fetch` استفاده کنیم. جریان اصلی لغو یه درخواست `fetch` اینجوری می‌شه.
۱. یه کلاس `AbortController` ایجاد کنیم

۲. ویژگی سیگنال اون آجکت ساخته شده رو دریافت کنیم و سیگنال رو به عنوان

یه متده است fetch ارسال کنیم

۳. برای لغو تمام هایی که از اون سیگنال استفاده میکنند با ویژگی abort از

AbortController رو فرآخوانی کنیم.

برای مثال، بیایین یه سیگنال رو به چندین درخواست fetch ارسال کنیم، همه

درخواستها با اون 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);
```

۳۸۴. چیه؟ Speech-API

API گفتار وب برای فعال کردن مرورگرهای مدرن برای شناسایی و ترکیب گفتار (یعنی داده‌های صوتی در برنامه‌های وب) استفاده می‌شود. این API توسط انجمن W3C در سال 2012 معرفی شد و دارای دو بخش اصلیه است.

۱. تشخیص گفتار (تشخیص گفتار ناهمزن یا گفتار به متن): این امکان را فراهم می‌کند که زمینه صدا را از ورودی صوتی تشخیص داده و به اون پاسخ بدین. این توسط رابط "SpeechRecognition" قابل دسترسیه.

مثال زیر نحوه استفاده از این API برای دریافت متن از گفتار رو نشون میده.

```

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 کاربر ازش اجازه می‌خواهد. این امکان را فراهم می‌کنند تا یه متن را به صدا تبدیل کنیم و به وسیله "SpeechSynthesisAPI" قابل دسترسی‌یه. برای مثال، کد زیر برای دریافت صدا/گفتار از متن استفاده می‌شه.

```

if ("speechSynthesis" in window) {
  const speech = new SpeechSynthesisUtterance("Hello World!");
  speech.lang = "en-US";
  window.speechSynthesis.speak(speech);
}

```

نمونه‌های بالا رو می‌شه روی کنسول برنامه‌نویسی مرورگر کروم (+33) تست کرد.
توجه: این API هنوز یه پیش‌نویس فعله و فقط روی مرورگرهای کروم و فایرفاکس وجود داره (البته کروم فقط مشخصات رو اجرا می‌کنند)

۳۸۵- حداقل timeout توی throttling چقدر؟

هم مرورگر و هم محیط‌های جاواسکریپت NodeJS با حداقل تاخیری که بیشتر از 0 میلی ثانیه اس throttles رو انجام میدن. یعنی حتی اگه تنظیم یه تاخیر 0ms به طور آنی اتفاق نیوفته.

مرورگرها: حداقل 4 میلی ثانیه تاخیر دارن. این throttles زمانی اتفاق می‌فونه که تماس‌های 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 می‌فونه که تاخیر بزرگتر از 2147483647 یا کمتر از 1 باشه. 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 نمی‌کنیم 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، نمی‌توانیم از setTimeout(fn, 0) برای اجرای فوری کد استفاده کنیم، اما ابرای دستیابی به این رفتار می‌توانیم از window.postMessage() استفاده کنیم.

۳۸۷. task 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 صف کار برنامه ریزی می‌شن. موارد استفاده برای افزودن وظایف به صف کار اینا هستن.

- موقعی که یه برنامه جاواسکریپت جدید مستقیماً از کنسول اجرا می‌شه یا توسط عنصر <script> اجرا می‌شه، وظیفه به صف کار اضافه می‌شه.

۲. موقعی که یه رویداد فراخونی میشه، callback رویداد به صف کار اضافه میشه
۳. وقتی به یه setInterval یا setTimeout رسید، callback مربوطه به صف کار اضافه میشه

۳۸۸. **چی هستن؟ microtask ها**

کد جاواسکریپتیه که باید بلافضلله پس از تکمیل task/Microtask در حال اجرا اجرا شه. اونا در واقع به نوعی مسدود کننده هستن. یعنی تا زمانی که صف microtask خالی نشه، رشته اصلی مسدود میشه.
Promise.resolve، Promise.reject، منابع اصلی Microtask ها عبارتند از MutationObservers، IntersectionObservers و غیره.
توجه: همه این Microtask ها توی همون چرخش event-loop پردازش میشن.

۳۸۹. **event-loop کوچکی های مختلف کدام هستن؟**

- این حلقه اصلی یه برنامه اس. به طور معمول این تو پایین صفحه اصلیه. خروج معمولاً نشون دهنده تمایل به بسته شدن برنامه اس. توی هر برنامه فقط یکی از این موارد میتونه وجود داشته باشه.
- این حلقه ایه که معمولاً تو پایین رویه اصلی یه UI یافت میشه.

۳۹۰. **هدف از queueMicrotask چیه؟**

صف میکروتسک به کد اجازه میده بدون تداخل با کدهای دیگه که در حالت تعليق هستن، با اولویت بالاتری اجرا بشه.

۳۹۱. **چطوری میشه از کتابخونههای جاواسکریپت توی فایل استفاده کرد؟ typescript**

مشخصه که همهی کتابخونهها یا چارچوبهای جاواسکریپت دارای فایل‌های اعلان TypeScript نیستن. اما اگه هنوزم میخواین از کتابخونهها یا فریمورکها تو فایل‌های TypeScript بدون دریافت خطاهای کامپایل استفاده کنیم تنها راه حل کلمه کلیدی

به همراه یه اعلان متغیره. برای مثال، بیاین تصور کنیم که یه کتابخونه به نام "customLibrary" دارید که اعلان TypeScript نداره و فضای نامی به نام customLibrary توی فضای نام گلوبال داره. میتونیم از این کتابخونه توی کد تایپاسکریپت به صورت زیر استفاده کنیم.

```
declare var customLibrary;
```

در زمان اجرا، تایپاسکریپت نوع اونو به متغیر customLibrary به صورت any 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؟

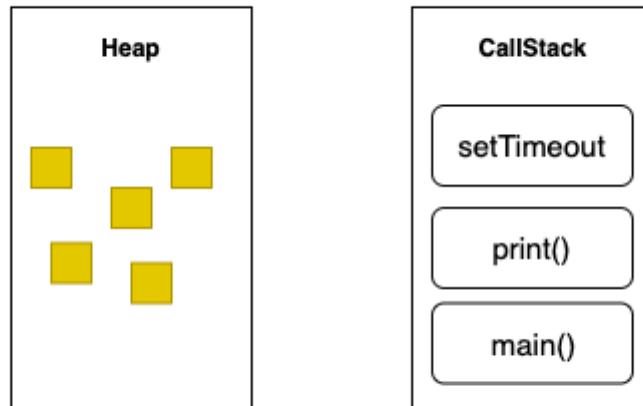
observable‌ها	Promise‌ها
چندین مقدار رو تو یه دوره زمانی منتشر میکنده (جريان مقاديری از ه تا چندگانه)	فقط یه مقدار رو تو یه زمان منتشر میکنه
اونا برای فراخوانی نیاز به اشتراک دارن	قراره فوراً فراخوانی شن
observable‌ها میتوون همزمان یا ناهمزمان	همیشه ناهمزمانه Promise حتی اگه بلافصله حل شه
map, forEach, filter, reduce, retryWhen و retry	هیچ اپراتور ارائه نمیده
با استفاده از روش unsubscribe لغو میشن	قابل لغو نیست

۳۹۳. چیه؟ heap

یا (memory heap) محلیه که تو اون آجکت‌ها موقع تعریف متغیرها ذخیره میشن یعنی این محلیه که تمام تخصیص حافظه و عدم تخصیص تو اون انجام میشه. هر دو call-stack و heap دو ظرف زمان اجرا JS هستن.

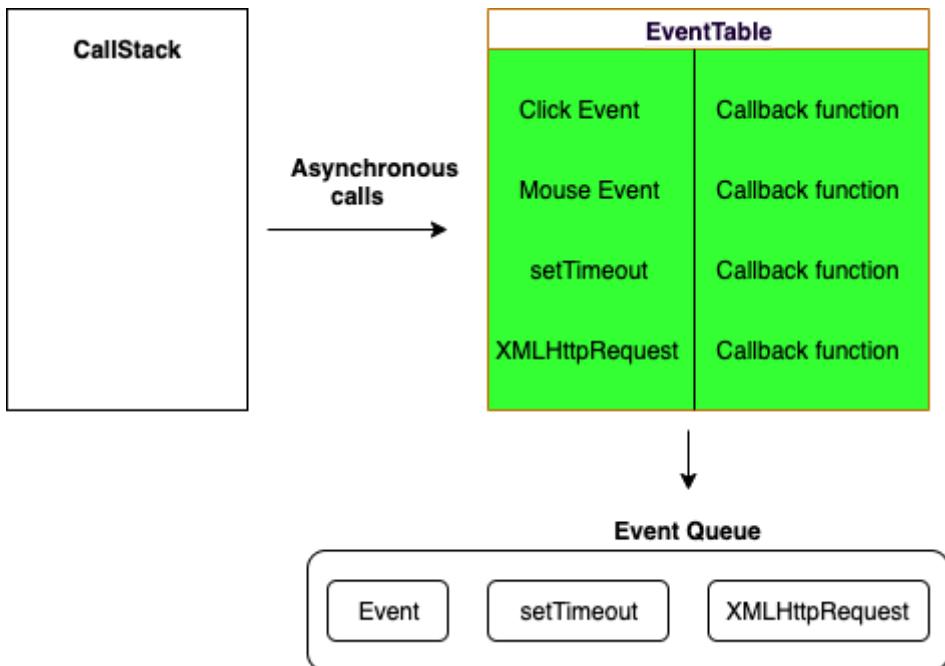
هر زمان که زمان اجرا با متغیرها و اعلان‌های تابع در کد مواجه میشه، اونا رو توی Heap ذخیره میکنه.

JS Runtime Engine



چیه؟ event-table .۳۹۴

Event-Table می‌شون، مثل بعد 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 کنیم یا عملیات `async` رو فراخوانی کنیم به جدول رویداد اضافه می‌شه. توابع رو به تنهایی اجرا نمی‌کنه. هدف اصلی جدول رویدادها پیگیری رویدادها و فرستادن‌شون به صف رویداد همونطور که توی نمودار زیر می‌بینیم.



۳۹۵. صف microTask چیه؟

صف جدیدیه که تو اون تمام وظایف آغاز شده توسط آبجکت Promise قبل از صف برگشت پردازش میشن صف microtask قبل از کارهای زندر و نقاشی بعدی پردازش میشه. اما اگه این ریزکارها برای مدت طولانی اجرا شن، منجر به مشکل بصری میشن.

۳۹۶. تفاوت بین polyfill و shim چیه؟

کتابخونهایه که یه API جدید رو با استفاده از ابزارهای اون 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 توی 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، shim یه polyfill برای API های مرورگر.

۳۹۷. چطوری متوجه primitive یا غیر primitive بودن یه نوع داده میشیم؟

در جاواسکریپت، دیتای primitive عبارتند از `boolean`, `string`, `number`, `primitive`. در حالی که انواع غیر primitive `undefined` و `BigInt`, `null`, `Symbol` ها می‌شوند. اما با تابع زیر می‌توانیم به راحتی اونا رو شناسایی کنیم `Object` ها.

```
const myPrimitive = 30;
const myNonPrimitive = {};
function isPrimitive(val) {
  return Object(val) !== val;
}

isPrimitive(myPrimitive);
isPrimitive(myNonPrimitive);
```

اگه مقدار یه نوع داده اولیه باشه، سازنده `Object` یه آبجکت wrapper جدید برای مقدار ایجاد میکنه. اما اگه مقدار یه نوع داده non-primitive (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 تو مرورگرها یا محیطهای فعلی و قدیمی تره. که میشه موارد زیر رو درموردش نوشت:

۱. تبدیل syntax
۲. ویژگی‌های Polyfill که در محیط هدف شما وجود نداره (با استفاده از `(babel/polyfill@`)
۳. تبدیل کد منبع

۳۹۹. آیا Node.js به شکل کامل تک thread کار میکنه؟

Node یه رشته است، اما بعضی از توابع موجود توی کتابخونه استانداره Node.js (برای `fs`) تک رشته‌ای نیستن. 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 اش، سوکت‌های وب با اعلان‌ها، تغییرات ورودی کاربر، فواصل تکراری و غیره اس.

۴۰۱. RxJS چیه؟

(افزونه‌های واکنش‌گرا برای جاوا‌اسکریپت) کتابخونه‌ای برای پیاده‌سازی برنامه‌نویسی observable با استفاده از که نوشتن کد ناهمزنمان یا مبتنی بر تماس رو آسان‌تر می‌کنه. همچنین توابع کاربردی رو برای ایجاد و کار با مشاهده پذیرها فراهم می‌کنه.

۴۰۲. تفاوت بین function-declaration و Function-constructor چیه؟

توابعی که با Function-constructor ایجاد می‌شن، برای زمینه‌های ایجاد خود بسته ایجاد نمی‌کنن، اما همیشه در محدوده جهانی ایجاد می‌شن یعنی تابع فقط می‌توانه به متغیرهای محلی خود و متغیرهای دامنه جهانی دسترسی داشته باشه. در حالی که اعلان‌های تابع می‌توانن به متغیرهای تابع بیرونی (بسته شدن) هم دسترسی داشته باشن. بیان این تفاوت رو با یه مثال بینیم،

:Function Constructor

```
const a = 100;
function createFunction() {
  const a = 200;
  return new Function("return a;");
}
console.log(createFunction()); // 100
```

:Function declaration

```
const a = 100;
function createFunction() {
  const a = 200;
  return function func() {
    return a;
  };
}
console.log(createFunction()); // 200
```

۴۰۳. شرط Short-circuit یا اتصال کوتاه چیه؟

شرایط اتصال کوتاه برای روش خلاصه شده نوشتمن دستورات if ساده استفاده میشه. بیاین سناریو رو با استفاده از يه مثال نشون بدیم. اگه بخوايم وارد پورتالی با شرایط احراز هویت بشیم، کد جاوا اسکریپتمون به این شکل نوشته میشه:

```
if (authenticate) {  
    loginToPorta();  
}
```

از اونجایی که عملگرهای منطقی جاوا اسکریپت از چپ به راست ارزیابی میشن عبارت بالا رو میشه 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 5 به 2 تغییر بدیم:

```
const 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 میشه خالی کرد:

```
const array = [1, 2, 3, 4, 5];  
array.length = 0;  
console.log(array.length); // 0  
console.log(array); // []
```

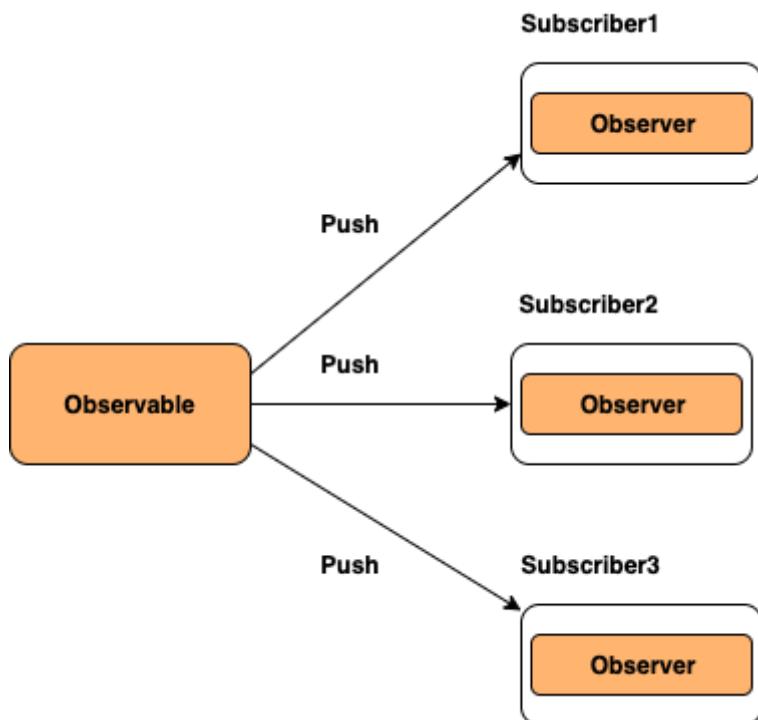
اساساً Observable تابعیه که میتوانه جریانی از مقادیر رو به صورت همزمان یا ناهمزمان subscribe به یه ناظر در طول زمان برگردونه. مصرف کننده میتوانه با فراخوانی متده مقدار رو دریافت کنه.

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.

۴۰۶. تفاوت‌های بین توابع و کلاس‌ها چیه؟

تفاوت اصلی بین اعلان‌های تابع و اعلان‌های کلاس **hoisting** هست. اعلان‌های تابع میشون **hoist** اما اعلان‌های کلاس نه.

:Classes

```
const user = new User(); // ReferenceError  
  
class User {}
```

:Constructor Function

```
const user = new User(); // No error  
  
function User() {}
```

۴۰۷. 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 میتونن شامل صفر یا بیشتر عبارت **await** باشن.
بیاین یه مثال تابع همگام زیر رو در نظر بگیریم،

```
async function logger() {  
  const data = await fetch("http://someapi.com/users"); // pause  
  until fetch returns  
  console.log(data);  
}  
logger();
```

این اساساً خوبی بیش از پرامیس‌ها و توابع جنریتور ES2015 هست.

۴۰۸. چطوری خطاهای ایجاد شده هنگام استفاده از **Promise**‌ها رو کنترل کنیم؟

موقع استفاده از کدهای **async** و ناهمزمان، **Promise**‌های ES6 میتونن زندگی برنامه‌نویس رو بدون داشتن ترس از **callback**‌ها و مدیریت خط در هر خط آسان‌تر

می‌کنن. اماPromise ها مشکلاتی دارن و بزرگ‌ترین اونا به‌طور پیش‌فرض مخفی کردن خطاهاست.

فرض کنیم انتظار دارین برای تمام موارد زیر یه خطأ توی کنسول چاپ بشه.

```
Promise.resolve("promised value").then(() => {
  throw new Error("error");
});

Promise.reject("error value").catch(() => {
  throw new Error("error");
});

new Promise((resolve, reject) => {
  throw new Error("error");
});
```

اما خیلی از محیط‌های جاواسکریپت مدرن وجود دارن که هیچ خطایی رو چاپ نمی‌کنن. می‌تونیم این مشکل رو به روش‌های مختلف حل کنیم.

۱. اضافه کردن یه بلوک **catch** به اخر هر زنجیره: ما می‌توانیم یه بلوک **catch** به آخر زنجیره پرمیس‌هایمان اضافه گنیم

```
Promise.resolve("promised value")
  .th(() => {
    throw new Error("error");
})
  .catch((error) => {
    console.error(error.stack);
});
```

اما تایپ کردن برای هر زنجیره پرمیس‌ها و پرمتاپ هم خیلی سخته. ۲. اضافه کردن متدهای **done**: می‌توانیم ابتدا راه حل‌ها رو جایگزین کنیم و بعد با روش انجام شده بلوک‌ها رو بگیریم

```
Promise.resolve("promised value").done(() => {
  throw new Error("error");
});
```

فرض کنیم می‌خواییم داده‌ها رو با استفاده از fetch HTTP کنیم و بعداً پردازش داده‌های حاصل رو به صورت ناهمزمان انجام بدیم. می‌توانیم بلوک **done** رو به صورت زیر بنویسیم.

```
getDataFromHttp()
  .then((result) => processDataAsync(result))
  .done((processed) => {
    displayData(processed);
  });
});
```

در آینده، اگه API کتابخونه پردازش به همگام تغییر کنه، میتونیم بلوک done رو حذف کنیم.

```
getDataFromHttp().then((result) =>
displayData(processDataAsyn(result)));
```

سپس فراموش کردین که بلوک «انجام شد» رو به بلوک then اضافه کنیم که منجر به خطاهای خاموش میشه.

۳. Extend ES6 Promises by Bluebird.

میاد Promise های اسکریپت رو گسترش میده تا در راه حل دو مشکل ایجا د نشه. این کتابخونه یه کنترل کنند ھ "پیش فرض" در Promises هس که تمام خطاهای رو از Rejection چاپ stderr رد شده به میکنه. پس از نصب، میتونیم ردهای کنترل نشده رو پردازش کنیم

```
Promise.onPossiblyUnhandledRejection((error) => {
  throw error;
});
```

یه reject رو انجام بدین فقط با یه catch خالی اونو مدیریت کنیم

```
Promise.reject("error value").catch(() => {});
```

چیه؟ Deno .۱۴۰۹

یه ران تایم (run-time) ساده، مدرن و ایمن برای جاواسکریپت و تایپ اسکریپت 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. توی جاواسکریپت چطوری یه object قابل پیمایش درست کنیم؟

به طور پیش فرض، آبجکت‌ها argument ساده قابل تکرار نیستن. اما می‌توانیم با تعریف ویژگی «Symbol.iterator» روی اون شی رو قابل تکرار کنیم. بیاین این رو با یه مثال نشون بدیم،

```
const collection = {
  one: 1,
  two: 2,
  three: 3,
  [Symbol.iterator]() {
    const values = Object.keys(this);
    let i = 0;
    return {
      next: () => ({
        value: this[values[i++]],
        done: i > values.length,
      }),
    };
  },
};

const iterator = collection[Symbol.iterator]();

console.log(iterator.next()); // → {value: 1, done: false}
console.log(iterator.next()); // → {value: 2, done: false}
console.log(iterator.next()); // → {value: 3, done: false}
console.log(iterator.next()); // → {value: undefined, done: true}
```

فرآیند بالا رو می‌شه با استفاده از یه تابع مولد ساده کرد:

```

const collection = {
  one: 1,
  two: 2,
  three: 3,
  *[Symbol.iterator]() {
    for (const 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}

```

۱۱۰. روش مناسب برای فراخوانی توابع بازگشته‌ی چیه؟

فراخوانی دنباله‌ی 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 پشته‌ای (stack) اضافی برای بازگشت ایجاد نمی‌کنه، زمانی که فراخوانی تابع یه فراخوانی دنباله اس.

برای مثال، بازگشت کلاسیک یا سر تابع فاکتوریل پایین به پشته (stack) 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 می‌کنیم اونا تمام داده‌های لازم رو که بهش نیاز داره رو بدون تکیه بر پشته (stack)، موقع برگشت به پایین منتقل می‌کنن.

```

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 ردیابی میکنه.

۱۴. چطوری بررسی کنیم که یه آبجکت Promise هست یا نه؟

اگه نمیدونیم یه مقدار یه Promise هس یا نه، مقدار رو به صورت بپیچید که یه قول رو برمی‌گردونه.

```

function isPromise(object) {
  if (Promise && Promise.resolve) {
    return Promise.resolve(object) === object;
  }
  throw "Promise not supported in your environment";
}

const i = 1;
const promise = new Promise((resolve, reject) => {
  resolve();
});

console.log(isPromise(i)); // false
console.log(isPromise(p)); // true

```

راه دیگر اینه که نوع handler.then رو بررسی کنیم

```

function isPromise(value) {
  return Boolean(value && typeof value.then === "function");
}
const i = 1;
const promise = new Promise((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؟

من تونیم 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 کنیم.

۱. اگه سازنده یا تابعی با استفاده از عملگر جدید فراخوانی شه، `new.target` یه 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 پارامتر argument چیه؟

۱. آبجکت `argument` آرایه ماننده اما آرایه نیست. در حالی که توی Rest پارامترها آرایه هستن.
۲. آبجکت `argument` از روش‌هایی مانند `pop`, `sort`, `map`, `forEach` یا `push` پشتیبانی نمیکنه. در حالی که این روش‌ها رو می‌شه رو پارامترهای Rest استفاده کرد.
۳. توی Rest پارامترها فقط اونایی هستن که نام جدایانه ای به اونا داده نشده در حالی که آبجکت `argument` شامل تمام آرگومان‌های ارسال شده به تابعه.

۴۱۵. تفاوت‌های بین عملگر spread و پارامتر rest چیه؟

پارامتر Rest تمام عناصر باقی مانده رو تو یه آرایه جمع آوری میکنه. در حالی که عملگر Spread به تکرار پذیرها (آرایه‌ها / اشیاء / رشته‌ها) اجازه میده تا به آرگومان‌ها / عناصر

منفرد گسترش پیدا کنن. یعنی پارامتر Rest مخالف عملگر spread هس.

۱۴. نوع‌های مختلف generator‌ها کداما هستند؟

۱. تعریف بیانی تابع :generator

```
function* myGenFunc() {  
    yield 1;  
    yield 2;  
    yield 3;  
}  
const genObj = myGenFunc();
```

۲. تعریف عبارتی تابع :generator

```
const myGenFunc = function* () {  
    yield 1;  
    yield 2;  
    yield 3;  
};  
const genObj = myGenFunc();
```

۳. تعریف بیانی تابع generator در آبجکت به صورت متد:

```
const myObj = {  
    *myGeneratorMethod() {  
        yield 1;  
        yield 2;  
        yield 3;  
    },  
};  
const genObj = myObj.myGeneratorMethod();
```

۴. تعریف تابع generator در کلاس‌ها:

```
class MyClass {  
    *myGeneratorMethod() {  
        yield 1;  
        yield 2;  
        yield 3;  
    }  
}  
const myObject = new MyClass();  
const genObj = myObject.myGeneratorMethod();
```

۵. تعریف تابع generator به عنوان یک ویژگی محاسبه شده:

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

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

۴۱۷. کدام built-in های iterable می باشد؟

۱. آرایه‌ها و TypedArrays

۲. رشته‌ها: روی هر کاراکتر یا نقاط تکرار یونیکد کنیم

۳. Map: روی جفت‌های کلید-مقدار آن تکرار شه

۴. مجموعه‌ها: روی عناصر خود تکرار می‌شه

۵. آرگومان‌ها: یه متغیر خاص آرایه مانند در توابع

۶. مجموعه NodeList DOM مانند

۴۱۸. تفاوت‌های بین حلقه for...in و for...of چیه؟

از for...of برای پیمایش بر روی داده‌ها با خاصیت تکرار (مثل آرایه‌ها)، استفاده می‌شه. تنها تفاوت در مورد چیزیه که اونا تکرار میکنن:

۱. روی تمام کلیدهای خصوصیت شمارش پذیر یه شی تکرار می‌شه

۲. بیش از مقادیر یه شی قابل تکرار.

بیاین این تفاوت رو توی یه مثال ببینیم،

```

const arr = ["a", "b", "c"];

arr.newProp = "newValue";

// key are the property keys
for (const key in arr) {
  console.log(key);
}

// value are the property values
for (const value of arr) {
  console.log(value);
}

```

از اونجا که حلقه for..in روی کلیدهای شی تکرار می‌شه حلقه اول ۰، ۱، ۲ و newProp را در حین تکرار روی شی آرایه ثبت می‌کنه. حلقه for..of روی مقادیر یه ساختار داده arr تکرار می‌شه و a، b، c را تو کنسول ثبت می‌کنه.

۴۱۹ چطوری property های instance و غیر instance می‌کنی؟

خصوصیات Instance باید در داخل متدهای کلاس تعریف بشن. برای مثال، مشخصات نام و سن سازنده داخلی هم مثل مثال پایین تعریف می‌شن.

```

class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
}

```

اما خصوصیات داده Static(class) و prototype از اعلان ClassBody تعریف بشن. بیاین مقدار سن رو برای کلاس Person به صورت زیر اختصاص بدیم.

```

Person.staticAge = 30;
Person.prototype.prototypeAge = 40;

```

۴۲۰. تفاوت‌های بین **Number.isNaN** و **isNaN** کدوما هستن؟

۱. **isNaN**: تابع سراسری «**isNaN**» آرگومان را به عدد تبدیل میکنε و اگهه مقدار حاصل **NaN** باشه، **true** رو برمی‌گردونε.

۲. **Number.isNaN**: این روش آرگومان را تبدیل نمیکنε. اما زمانی که نوع يه عدد و مقدار **NaN** باشه مقدار **true** رو برمی‌گردونε.

بیاین تفاوت رو با يه مثال ببینیم:

```
isNaN("hello"); // true  
Number.isNaN("hello"); // false
```