### آموزش وب دیزاین

فهرست

[آموزش وب دیزاین 1](#_Toc211851979)

[مفهوم Frontend (فرانت‌اند) 11](#_Toc211851980)

[بررسی مفهوم Backend 14](#_Toc211851981)

[معرفی HTML: 16](#_Toc211851982)

[معرفی Css: 20](#_Toc211851983)

[معرفی JavaScript 24](#_Toc211851984)

[بررسی مفهوم SEO 25](#_Toc211851985)

[بررسی مفهوم Progressive Enhancement بهبود تدریجی 28](#_Toc211851986)

[بررسی مفهوم Responsive Web Design 32](#_Toc211851987)

[بررسی مفهوم Internet 37](#_Toc211851988)

[بررسی مفهوم Protocol 43](#_Toc211851989)

[1. زبان مشترک 44](#_Toc211851990)

[بررسی مفهوم WEB 50](#_Toc211851991)

[تاریخچه وب به زبان ساده 52](#_Toc211851992)

[بررسی مفهوم HTTP 59](#_Toc211851993)

[HTTP چیست: 66](#_Toc211851994)

[HTTP چیست نیست: 66](#_Toc211851995)

[بررسی مفهوم Server 67](#_Toc211851996)

[بررسی مفهوم IP 75](#_Toc211851997)

[بررسی مفهوم Domain 78](#_Toc211851998)

[بررسی مفهوم DNS 81](#_Toc211851999)

[بررسی مفهوم Interanet 86](#_Toc211852000)

[بررسی مفهوم Extranet 89](#_Toc211852001)

[بررسی موتورهای رندرگیری مرورگرها 92](#_Toc211852002)

[بررسی URL 93](#_Toc211852003)

[بررسی مفهوم HTTP: 97](#_Toc211852004)

[بررسی مفهوم HTTPS 100](#_Toc211852005)

[بررسی سایت‌های Static و Dynamic 102](#_Toc211852006)

[بررسی request و Response های یک وب سایت 104](#_Toc211852007)

[بررسی نحوه رندرگیری کدها توسط مرورگر 107](#_Toc211852008)

[آموزش HTML 109](#_Toc211852009)

[کامنت ها در HTML 110](#_Toc211852010)

[معرفی tag ها در Html و انواع آن (singel & pairs) 113](#_Toc211852011)

[ساختار تگ ها در HTML 114](#_Toc211852012)

[معرفی VS Code و پلاگین های آن 115](#_Toc211852013)

[بررسی ساختار یک سند Html 117](#_Toc211852014)

[معرفی BOM و DOM 123](#_Toc211852015)

[DOM - Document Object Model (مدل شیءگرای سند) 124](#_Toc211852016)

[نمایش درخت Html 125](#_Toc211852017)

[بررسی مهفوم Semantic 129](#_Toc211852018)

[تگهای HTML 130](#_Toc211852019)

[معرفی تگ p و display آن 131](#_Toc211852020)

[معرفی تگ های Heading 131](#_Toc211852021)

[معرفی تگ hr و نحوه Display آن 133](#_Toc211852022)

[معرفی لیست‌های ترتیبی و غیر ترتیبی و display آنها 134](#_Toc211852023)

[آموزش EMMET 136](#_Toc211852024)

[معرفی attribute های تگ های HTML 140](#_Toc211852025)

[مفهوم Attribute ها 140](#_Toc211852026)

[انواع Attribute ها 140](#_Toc211852027)

[1. Attribute های جهانی (Global Attributes) 141](#_Toc211852028)

[2. Attribute های خاص تگ‌ها 141](#_Toc211852029)

[آموزش لیست‌های Nested 144](#_Toc211852030)

[معرفی پلاگین Live server در Vs code 144](#_Toc211852031)

[معرفی تگ blockqoute و dislayآن 146](#_Toc211852032)

[تگ Pre و Display آن 147](#_Toc211852033)

[معرفی Webstorm و نحوه نصب و کرک آن 150](#_Toc211852034)

[بررسی تفاوت‌های HTML4 و HTML5 151](#_Toc211852035)

[معرفی Doctype 155](#_Toc211852036)

[Semantic 157](#_Toc211852037)

[معرفی تگ main و Display آن 157](#_Toc211852038)

[تگ Header و display آن 160](#_Toc211852039)

[تگ Footer و Display آن 163](#_Toc211852040)

[تگ Section و Display آن 168](#_Toc211852041)

[تگ article و Display آن 172](#_Toc211852042)

[معرفی تگ aside و Display آن 180](#_Toc211852043)

[معرفی تگ nav و Display آن 181](#_Toc211852044)

[معرفی تگ address و display آن 187](#_Toc211852045)

[معرفی تگ em و display آن 196](#_Toc211852046)

[بررسی تفاوت‌های display های Inline و block 198](#_Toc211852047)

[معرفی تگ strong و display آن 202](#_Toc211852048)

[معرفی تگ b و display آن 206](#_Toc211852049)

[تگ i و Display آن 210](#_Toc211852050)

[معرفی تگ S و dispaly آن 213](#_Toc211852051)

[معرفی تگ u و display آن 217](#_Toc211852052)

[معرفی تگ small و display آن 217](#_Toc211852053)

[معرفی تگ q و display آن 217](#_Toc211852054)

[معرفی تگ abbr و display آن 218](#_Toc211852055)

[بررسی حالات nesting در element ها 218](#_Toc211852056)

[معرفی تگ sub و display آن 219](#_Toc211852057)

[معرفی تگ sup و display آن 219](#_Toc211852058)

[معرفی تگ mark و display آن 219](#_Toc211852059)

[معرفی تگ ins و display آن 219](#_Toc211852060)

[معرفی تگ del و display آن 220](#_Toc211852061)

[معرفی تگ br و display آن 220](#_Toc211852062)

[معرفی تگ div و display آن 220](#_Toc211852063)

[معرفی تگ span و display آن 221](#_Toc211852064)

[معرفی id و class و بررسی تفاوت آن ها 221](#_Toc211852065)

[معرفی global attributes 222](#_Toc211852066)

[بررسی character escapes 223](#_Toc211852067)

[آموزش دستورات پایه CMD 224](#_Toc211852068)

[معرفی node js 226](#_Toc211852069)

[معرفی npm 240](#_Toc211852070)

[معرفی live-server و دستورات آن 241](#_Toc211852071)

[معرفی تگ a و display آن 242](#_Toc211852072)

[بررسی mail links 246](#_Toc211852073)

[بررسی tel links 247](#_Toc211852074)

[معرفی تگ img و display آن 250](#_Toc211852075)

[بررسی تفاوت display های block و inline و inline-block 254](#_Toc211852076)

[معرفی تگ table و display آن 257](#_Toc211852077)

[بررسی چگونگی عمکرد form ها 272](#_Toc211852078)

[معرفی تگ iframe 286](#_Toc211852079)

[معرفی تگ video و display آن 289](#_Toc211852080)

[Display پیش‌فرض: `inline` 290](#_Toc211852081)

[معرفی تگ audio و display آن 293](#_Toc211852082)

[معرفی تگ picture 298](#_Toc211852083)

[آموزش CSS 299](#_Toc211852084)

[معرفی declaration 299](#_Toc211852085)

[معرفی selector و property و value 299](#_Toc211852086)

[معرفی element selector 300](#_Toc211852087)

[معرفی id selector(local & global) 300](#_Toc211852088)

[معرفی class selector(local & global) 301](#_Toc211852089)

[بررسی تفاوت Style های inline , internal ,external 301](#_Toc211852090)

[بررسی اولویت های style دهی 302](#_Toc211852091)

[Complete 303](#_Toc211852092)

[کامنت ها در CSS 303](#_Toc211852093)

[معرفی !important و اولویت آن 303](#_Toc211852094)

[معرفی group selector 304](#_Toc211852095)

[معرفی CSS Units 305](#_Toc211852096)

[معرفی Font-family 307](#_Toc211852097)

[Color 319](#_Toc211852098)

[Selectors 321](#_Toc211852099)

[استایل متن 326](#_Toc211852100)

[Width 339](#_Toc211852101)

[Height 340](#_Toc211852102)

[Opacity 341](#_Toc211852103)

[معرفی attribute selector ها و regex 342](#_Toc211852104)

[Background 344](#_Toc211852105)

[بررسی سایت css gradient generator 369](#_Toc211852106)

[ساخت کامپوننت parallax 369](#_Toc211852107)

[@import 370](#_Toc211852108)

[Padding 371](#_Toc211852109)

[border 373](#_Toc211852110)

[outline 385](#_Toc211852111)

[Margin 392](#_Toc211852112)

[بررسی مفهوم box model 398](#_Toc211852113)

[Overflow 400](#_Toc211852114)

[Box-sizing 403](#_Toc211852115)

[Root 403](#_Toc211852116)

[calc() 404](#_Toc211852117)

[Display 405](#_Toc211852118)

[Box-shadow 408](#_Toc211852119)

[لیست ها 409](#_Toc211852120)

[فونت آیکون ها: 411](#_Toc211852121)

[Font Awesome 413](#_Toc211852122)

[Google Material Icons 413](#_Toc211852123)

[Bootstrap Icons 414](#_Toc211852124)

[Phosphor Icons 414](#_Toc211852125)

[Tabler Icons 414](#_Toc211852126)

[Feather Icons 414](#_Toc211852127)

[Heroicons 415](#_Toc211852128)

[Remix Icon 416](#_Toc211852129)

[pseudo 416](#_Toc211852130)

[transition 419](#_Toc211852131)

[ساخت یک پروژه ساده 446](#_Toc211852132)

[Float 454](#_Toc211852133)

[Shape-outside 458](#_Toc211852134)

[Shape-margin 460](#_Toc211852135)

[Position(static,relative,absolute,fixed,sticky) 465](#_Toc211852136)

[Top , left , right , bottom 467](#_Toc211852137)

[Z index 467](#_Toc211852138)

[ساخت منو nested 471](#_Toc211852139)

[ساخت mega menu 477](#_Toc211852140)

[ساخت منو همبرگری 486](#_Toc211852141)

[ساخت منو دیجی کالا 508](#_Toc211852142)

[Transform 528](#_Toc211852143)

[Perspective 533](#_Toc211852144)

[Perspective-origin 534](#_Toc211852145)

[Filter 536](#_Toc211852146)

[animation 543](#_Toc211852147)

[Resize 556](#_Toc211852148)

[Table 559](#_Toc211852149)

[معرفی swiper js 562](#_Toc211852150)

[Flex 574](#_Toc211852151)

[آموزش وسط چین کردن تگ ها 587](#_Toc211852152)

[media query 598](#_Toc211852153)

[معرفی srcset و x-descriptor و w-descriptor 604](#_Toc211852154)

[معرفی owl carousel 605](#_Toc211852155)

[پروژه تبدیل UI به صفحات وب 609](#_Toc211852156)

[نحوه خرید دامنه و هاست 609](#_Toc211852157)

[معرفی grid css 609](#_Toc211852158)

[آموزش جاوا اسکریپت Java script 628](#_Toc211852159)

[معرفی زبان JavaScript 628](#_Toc211852160)

[معرفی کاربرد های زبان JavaScript 629](#_Toc211852161)

[معرفی فریمورک ها و کتابخانه های این زبان 630](#_Toc211852162)

[بررسی تفاوت های زبان های مفسری و کامپایلری 632](#_Toc211852163)

[معرفی تگ script 634](#_Toc211852164)

[بررسی قرار دادن تگ script در جای های مختلف 634](#_Toc211852165)

[معرفی اتریبیوت های defer و async و تفاوت آن ها 637](#_Toc211852166)

[نحوه نوشتن JavaScript به صورت external 638](#_Toc211852167)

[چاپ کردن دیتا با java script 639](#_Toc211852168)

[نحوه کامنت گذاری 644](#_Toc211852169)

[معرفی متغیر ها 645](#_Toc211852170)

[معرفی var و let و const 648](#_Toc211852171)

[معرفی انواع داده 649](#_Toc211852172)

[معرفی typeof 651](#_Toc211852173)

[بررسی concat در js 653](#_Toc211852174)

[بررسی بک تیک و template string 654](#_Toc211852175)

[نحوه ورودی گرفتن از کاربر 655](#_Toc211852176)

[آموزش casting در js 656](#_Toc211852177)

[معرفی block scope ها و متغیر های globalو local 657](#_Toc211852178)

[معرفی arithmetic operator ها 658](#_Toc211852179)

[معرفی assignment operator ها 658](#_Toc211852180)

[معرفی comparison operator ها 659](#_Toc211852181)

[معرفی ternary operator 659](#_Toc211852182)

[معرفی logical operator ها 660](#_Toc211852183)

[معرفی nan و تابع isNan 661](#_Toc211852184)

[معرفی توابع در js 667](#_Toc211852185)

[نحوه ورودی گرفتن توابع 669](#_Toc211852186)

[بررسی تفاوت توابع void و غیره 669](#_Toc211852187)

[بررسی توابع بازگشتی 670](#_Toc211852188)

[معرفی حلقه ها 673](#_Toc211852189)

[معرفی آرایه ها 677](#_Toc211852190)

[معرفی تابع isArray 679](#_Toc211852191)

[معرفی associative array ها 680](#_Toc211852192)

[معرفی تابع toString 681](#_Toc211852193)

[معرفی تابع join 681](#_Toc211852194)

[معرفی تابع delete 681](#_Toc211852195)

[معرفی تابع Splice 682](#_Toc211852196)

[معرفی تابع concat 682](#_Toc211852197)

[معرفی تابع Slice 683](#_Toc211852198)

[معرفی تابع sortNumeric 684](#_Toc211852199)

[معرفی متد foreach 685](#_Toc211852200)

[معرفی متد map 685](#_Toc211852201)

[بررسی تافوت foreach و map 686](#_Toc211852202)

[معرفی متد filter 686](#_Toc211852203)

[معرفی متد indexOf و lastIndexOf 687](#_Toc211852204)

[معرفی object 688](#_Toc211852205)

[نحوه دسترسی به عناصر object 689](#_Toc211852206)

[آموزش event ها 692](#_Toc211852207)

[نحوه css دادن با js 703](#_Toc211852208)

[ساخت ماشین حساب با js 714](#_Toc211852209)

[معرفی متد های string 724](#_Toc211852210)

[نحوه دسترسی به کارکتر های string 733](#_Toc211852211)

[معرفی متد split 736](#_Toc211852212)

[آموزش ساخت modal 741](#_Toc211852213)

[معرفی Math در js 744](#_Toc211852214)

[ساخت captcha 750](#_Toc211852215)

[معرفی arrow function ها 751](#_Toc211852216)

[معرفی dom و bom 751](#_Toc211852217)

[دسترسی به dom با استفاده از id و classname و tagname 754](#_Toc211852218)

[استفاده از queryselector 756](#_Toc211852219)

[معرفی innerhtml و innertext 760](#_Toc211852220)

[معرفی .attribute 761](#_Toc211852221)

[معرفی setattribute 762](#_Toc211852222)

[معرفی .clasename 767](#_Toc211852223)

[معرفی .href 769](#_Toc211852224)

[معرفی .src 771](#_Toc211852225)

[معرفی regular expression 775](#_Toc211852226)

[استفاده از متد های search و replace 795](#_Toc211852227)

[معرفی modifier ها در search 797](#_Toc211852228)

[استفاده از parentnode و childrennode و firstchild و lastchild و nextsibiling و previoussibiling 799](#_Toc211852229)

[معرفی createElement و append و insertBefore و remove 801](#_Toc211852230)

[معرفی bom 806](#_Toc211852231)

[معرفی setTimeOut و setInterval 811](#_Toc211852232)

[ساخت ساعت دیجیتال 815](#_Toc211852233)

[معرفی json 822](#_Toc211852234)

[بررسی تفاوت sync و async 830](#_Toc211852235)

[استفاده از callback در async 832](#_Toc211852236)

[استفاده از promise در async 834](#_Toc211852237)

[استفاده از await در async 840](#_Toc211852238)

[معرفی ajax و api 843](#_Toc211852239)

[بررسی تفاوت soap و Rest api 844](#_Toc211852240)

[معرفی get در api 846](#_Toc211852241)

[XmlHttpRequest 851](#_Toc211852242)

[Promise 853](#_Toc211852243)

[Fetch 854](#_Toc211852244)

[Axios 856](#_Toc211852245)

[معرفی ecmascript 863](#_Toc211852246)

[نحوه array destructing 865](#_Toc211852247)

[نحوه object destructing 867](#_Toc211852248)

[نحوه import و export 871](#_Toc211852249)

[استفاده از allias و \* در import و export 881](#_Toc211852250)

[معرفی مفاهیم شی گرایی و کلاس ها 882](#_Toc211852251)

[ساخت کلاس در js 882](#_Toc211852252)

[معرفی متغیر ها و توابع public و private 883](#_Toc211852253)

[معرفی setter و getter 884](#_Toc211852254)

[معرفی constructor 886](#_Toc211852255)

[معرفی وراثت در شی گرایی 887](#_Toc211852256)

[معرفی کلاس های static 891](#_Toc211852257)

[معرفی es-toolkit.dev 896](#_Toc211852258)

[چندپارادایمی: 898](#_Toc211852259)

[1- برنامه نویسی تابعی یا رویه‌ایی: 898](#_Toc211852260)

[2- برنامه نویسی شیئ گرا (ObjectOriented Programming) 899](#_Toc211852261)

[3- برنامه‌نویسی تابعی (Functional Programming) 900](#_Toc211852262)

[آموزش استفاده از Map در جاوا اسکریپت 903](#_Toc211852263)

[1- مقدمه‌ای بر Map 903](#_Toc211852264)

[نکات پیشرفته 913](#_Toc211852265)

[تمرینات اضافه 915](#_Toc211852266)

[آموزش کار با createDocumentFragment() 921](#_Toc211852267)

[آموزش استفاده از filter 922](#_Toc211852268)

[آموزش جامع استفاده از متد filter() در جاوااسکریپت 922](#_Toc211852269)

[1- مقدمه‌ای بر متد filter() 922](#_Toc211852270)

[2- سینتکس و پارامترها 922](#_Toc211852271)

[3- موارد استفاده و مثال‌های کاربردی 923](#_Toc211852272)

[4- تفاوت filter با سایر متدهای آرایه 924](#_Toc211852273)

[5- بهترین practices و نکات مهم 925](#_Toc211852274)

[خلاصه مزایای استفاده از filter(): 928](#_Toc211852275)

[آموزش استفاده از reduce 929](#_Toc211852276)

[مثال‌های کاربردی 929](#_Toc211852277)

[تمرین‌ : 931](#_Toc211852278)

[آموزش استفاده از Rest parameters آموزش استفاده از Spread Operator: 933](#_Toc211852279)

[1. Rest Parameters (پارامترهای rest) 933](#_Toc211852280)

[2. Spread Operator (عملگر spread) 934](#_Toc211852281)

[آموزش کار با تابع join() 936](#_Toc211852282)

[آموزش Destructuring Assignment 937](#_Toc211852283)

[Destructuring آبجکت‌ها (Object Destructuring) 938](#_Toc211852284)

[کاربردهای عملی Destructuring 940](#_Toc211852285)

[Async در جاوا اسکریپت 945](#_Toc211852286)

[Defer در جاوا اسکریپت 945](#_Toc211852287)

بررسی مفهمUI:

البته! در ادامه به بررسی مفهوم UI (رابط کاربری) به زبان فارسی میپردازیم.

UI (رابط کاربری) چیست؟

UI یا User Interface (رابط کاربری) به تمام بخش‌های بصری و گرافیکی یک محصول دیجیتال (مانند وب‌سایت، اپلیکیشن موبایل، نرم‌افزار) گفته می‌شود که کاربر با آن تعامل مستقیم دارد. هدف UI ایجاد یک رابط زیبا، intuitive (قابل درک و شهودی) و کارآمد است تا کاربر بتواند به راحتی با محصول ارتباط برقرار کند.

به زبان ساده، UI ظاهر و حس یک محصول است. هر چیزی که می‌بینید، کلیک می‌کنید، یا با آن تعامل دارید، بخشی از UI است.

اجزای اصلی UI (رابط کاربری)

رابط کاربری از عناصر مختلفی تشکیل شده است که با هم ترکیب می‌شوند تا یک تجربه بصری منسجم را ایجاد کنند. این اجزا شامل موارد زیر هستند:

|  |  |  |
| --- | --- | --- |
| دسته‌بندی | توضیح | مثال‌ها |
| ۱. عناصر بصری (Visual Elements) | اینها اجزایی هستند که ظاهر و حس کلی طراحی را شکل می‌دهند. | رنگ‌ها: پالت رنگی که برای طراحی انتخاب می‌شود  تایپوگرافی: فونت‌ها، سایز متن‌ها و فواصل بین حروف  آیکون‌ها: نمادهای کوچک برای انتقال مفاهیم.  عکس‌ها و تصاویر: محتوای بصری جذاب.  فضای خالی (White Space): فضای بین عناصر برای خوانایی بهتر. |
| ۲. عناصر تعاملی (Interactive Elements) | اینها المان‌هایی هستند که کاربر با آنها تعامل فیزیکی دارد. | دکمه‌ها (Buttons): برای انجام اقدامات (ثبت، ذخیره، ارسال).  فیلدهای فرم (Form Fields): برای وارد کردن اطلاعات (متون، ایمیل، رمز).  منوها (Menus): برای پیمایش در سایت یا اپلیکیشن  اسلایدرها (Sliders) و کاروسل‌ها (Carousels): برای نمایش محتوا به صورت متغیر  چک‌باکس‌ها (Checkboxes) و Radio Buttons: برای انتخاب گزینه‌ها. |
| ۳. چیدمان و ساختار (Layout & Structure) | این بخش به چگونگی سازماندهی و قرارگیری عناصر روی صفحه اشاره دارد. | Grid System: سیستم شبکه‌ای برای مرتب کردن عناصر.  هدر (Header) و فوتر (Footer): بخش‌های ثابت بالا و پایین صفحه.  ناوبری (Navigation): منوها و مسیرهای حرکت کاربر در محصول.  کارد (Card) و لیست‌ها: روش‌های نمایش اطلاعات |

تفاوت UI و UX (یک تفاوت کلیدی)

این دو مفهوم بسیار به هم مرتبط هستند، اما یکی نیستند :

UI (رابط کاربری): به ظاهر و المان‌های گرافیکی محصول می‌پردازد. (زیبایی‌شناسی، رنگ‌ها، تایپوگرافی)

UX (تجربه کاربری): به حس و تجربه کلی کاربر از تعامل با محصول می‌پردازد. (آسانی استفاده، کارایی، رضایت)

یک مقایس ساده:

تصور کنید یک خودرو دارید:

UI شامل: فرمان، صفحه کیلومترشمار، دکمه‌های رادیو، رنگ و طراحی داخلی خودرو است.

UX شامل: این است که رانندگی با این خودرو چقدر راحت است، آیا فرمان به خوبی پاسخ می‌دهد، و آیا از رانندگی با آن لذت می‌برید یا نه.

یک UI زیبا ولی با UX ضعیف، مانند یک ماشین بسیار زیبا ولی با فرمان سخت و ناراحت است. برعکس، یک UX عالی با UI ضعیف، مانند یک ماشین بسیار راحت ولی با ظاهر زشت است. محصول موفق ترکیبی از هر دو است.

اصول طراحی UI خوب

یک طراح UI برای ایجاد یک رابط موثر، این اصول را رعایت می‌کند:

1. وضوح و سادگی (Clarity & Simplicity): طراحی باید ساده و بدون عناصر غیرضروری باشد تا کاربر به راحتی بتواند هدف را پیدا کند.

2. ثبات و یکنواختی (Consistency): استفاده از المان‌های یکسان در کل محصول (مثلاً دکمه‌های همشکل، رنگ‌های ثابت). این کار باعث می‌شود کاربر احساس آشنا بودن کند.

3. بازخورد (Feedback): رابط باید به کاربر بازخورد دهد. مثلاً وقتی دکمه‌ای کلیک می‌شود، تغییر رنگ دهد یا یک پیام "عملیات موفق" نمایش داده شود.

4. سلسله مراتب بصری (Visual Hierarchy): چیدمان عناصر به گونه‌ای باشد که مهم‌ترین اطلاعات یا اقدامات، توجه کاربر را جلب کنند.

5. دسترسی‌پذیری (Accessibility): طراحی باید به گونه‌ای باشد که افراد با disabilities (معلولیت‌ها) مانند کوررنگی یا مشکلات بینایی نیز بتوانند از آن استفاده کنند (مثلاً کنتراست رنگی کافی داشته باشد).

نتیجه‌گیری

UI یا رابط کاربری، جلوه بصری و لمسی یک محصول دیجیتال است که اولین impression (تاثیر) را روی کاربر می‌گذارد. یک UI خوب، نه تنها زیبا است، بلکه کاربردی، intuitive و لذت‌بخش است و به کاربر کمک می‌کند تا به هدف خود به راحتی برسد.

طراحی UI یک تخصص فنی خلاقانه است که در کنار طراحی UX، پایه و اساس ساخت محصولات دیجیتال موفق را تشکیل می‌دهد.

# مفهوم Frontend (فرانت‌اند)

Frontend (فرانت‌اند) چیست؟

Frontend یا سمت کلاینت (Client Side) به تمام بخش‌هایی از یک وب‌سایت یا اپلیکیشن گفته می‌شود که مستقیماً توسط کاربر دیده و با آن تعامل می‌شود . هر چیزی که هنگام باز کردن یک صفحه وب در مرورگر خود مشاهده می‌کنید—از طرحبندی و رنگ‌ها گرفته تا منوها، دکمه‌ها، تصاویر و انیمیشن‌ها—نتیجه کدهایی است که توسط توسعه‌دهنده فرانت‌اند نوشته شده است.

به بیان ساده، فرانت‌اند ظاهر، حس و تعامل یک وب‌سایت است.

اجزای اصلی توسعه فرانت‌اند (Three Pillars)

توسعه فرانت‌اند بر سه technology (فناوری) اصلی استوار است:

|  |  |  |
| --- | --- | --- |
| فناوری | نقش و مسئولیت | توضیح |
| HTML | اسکلت و ساختار | (HyperText Markup Language)  مسئول ایجاد ساختار و محتوای صفحه است. تمامی متن‌ها، تصاویر، لینک‌ها و عناصر اصلی صفحه توسط HTML تعریف می‌شوند. |
| CSS | ظاهر و استایل | (Cascading Style Sheets) مسئول زیباسازی و چیدمان عناصر HTML است. رنگ‌ها، فونت‌ها، انیمیشن‌ها، layout (چیدمان) responsive (واکنش‌گرا) و هر آنچه مربوط به ظاهر سایت است با CSS پیاده‌سازی می‌شود. |
| JavaScript | رفتار و تعامل | مسئول هوشمندسازی و تعاملی کردن صفحه است. هرگونه (رفتار پویا)—مانند کلیک بر روی دکمه‌ها، ارسال فرم‌ها، بارگذاری داده بدون رفرش صفحه (AJAX) و ایجاد انیمیشن‌های پیچیده—توسط JavaScript کنترل می‌شود. |

تفاوت Frontend با Backend (یک مقایسه کلیدی)

برای درک کامل فرانت‌اند، مقایسه آن با backend (بک‌اند) ضروری است:

|  |  |  |
| --- | --- | --- |
| معیار | Frontend (فرانت‌اند) | Backend (بک‌اند) |
| محل اجرا | مرورگر کاربر (Client Side) | سرور (Server Side) |
| کاربری که می‌بیند | کاربر نهایی | توسعه‌دهندگان و صاحبان سایت |
| مسئولیت | چگونگی نمایش و تعامل با داده‌ها | پردازش، ذخیره‌سازی و مدیریت داده‌ها |
| مثال | ظاهر یک فرم ثبت‌نام | بررسی اطلاعات ورودی، ذخیره آن در دیتابیس و ارسال ایمیل تایید |
| تکنولوژی‌های رایج | HTML, CSS, JS, React, Angular, Vue | Python, PHP, Java, Node.js, Ruby دیتابیس‌هایی مثل MySQL |

یک مقایسه ساده:   
یک رستوران را در نظر بگیرید:

فرانت‌اند: فضای داخلی زیبای رستوران، میزها، صندلی‌ها، منوی غذایی که مشتری آن را می‌بیند و با آن تعامل دارد.

بک‌اند: آشپزخانه، محل ذخیره مواد غذایی، سیستم مدیریت سفارشات و آشپزها که در پشت صحنه کار می‌کنند.

هر دوی این بخش‌ها برای ارائه یک تجربه عالی به مشتری (کاربر) ضروری هستند.

مهارت‌های مورد نیاز یک توسعه‌دهنده فرانت‌اند

یک توسعه‌دهنده فرانت‌اند مدرن، علاوه بر تسلط بر سه پایه اصلی، به مهارت‌های دیگری نیز نیاز دارد:

1. فریم‌ورک‌ها و کتابخانه‌های JavaScript: تسلط بر حداقل یکی از فریم‌ورک‌های محبوب مانند React ، Angular یا Vue.js امروزه ضروری هستند. این ابزارها توسعه اپلیکیشن‌های پیچیده و تعاملی را بسیار سریع‌تر و آسان‌تر می‌کنند.

2. Responsive Design (طراحی واکنش‌گرا): توانایی طراحی سایت‌هایی که ظاهر و کارایی خود را روی تمام دستگاه‌ها (موبایل، تبلت، دسکتاپ) به optimal (بهینه)ترین شکل حفظ می‌کنند. این کار با استفاده از CSS Media Queries و layoutهای انعطافپذیر مانند Flexbox و CSS Grid انجام می‌شود.

3. Version Control (کنترل نسخه): تسلط بر سیستم‌هایی مانند Git (و پلتفرم‌هایی مثل GitHub یا GitLab) برای مدیریت کدها و همکاری با دیگر توسعه‌دهندگان.

4. ابزارهای Build و Module Bundlers: آشنایی با ابزارهایی مانند Webpack ، Vite ، Babel که کدهای نوشته شده را برای production (استفاده نهایی) بهینه و bundle می‌کنند.

5. APIها: توانایی برقراری ارتباط و دریافت داده از REST API یا GraphQL . (یعنی فراخوانی API و کار با داده JSON).

6. مفاهیم پایه سئو (SEO): آگاهی از اصول اولیه سئو برای تولید کدهایی که توسط موتورهای جستجو مانند Google به راحتی crawl و index شوند.

# بررسی مفهوم Backend

Backend (بک‌اند) چیست؟

Backend یا سمت سرور (Server Side) به تمام بخش‌های پنهان یک وب‌سایت یا اپلیکیشن گفته می‌شود که کاربر به طور مستقیم آن را نمی‌بیند، اما برای عملکرد صحیح سیستم absolutely vital (کاملاً حیاتی) است . بک‌اند مسئول پردازش منطق کسب‌وکار، ذخیره‌سازی داده‌ها، مدیریت کاربران و در نهایت، پاسخگویی به درخواست‌هایی است که از سمت فرانت‌اند ارسال می‌شود.

به بیان ساده، اگر فرانت‌اند را مغز زیبا و کاربرپسند یک فروشگاه بدانیم، بک‌اند انبار، سیستم حسابداری، مدیریت موجودی و logistics (لجستیک) پشت صحنه آن است.

اجزای اصلی توسعه بک‌اند (Core Components)

یک سیستم بک‌اند معمولاً از چند بخش کلیدی تشکیل شده است:

|  |  |
| --- | --- |
| جزء | توضیح و مسئولیت |
| 1. سرور (Server) | یک کامپیوتر قدرتمند است که نرم‌افزار سرویس‌دهنده (مانند Nginx, Apache) روی آن نصب شده و مسئول دریافت و پاسخگویی به درخواست‌های کلاینت‌ها (مرورگرهای کاربران) است. |
| 2. برنامه (Application) | این همان کد اصلی ی است که توسعه‌دهنده بک‌اند می‌نویسد. این کد با استفاده از فریم‌ورک‌ها و زبان‌های برنامه‌نویسی (مانند Python/Django, PHP/Laravel, Node.js/Express) نوشته می‌شود و مسئول اجرای منطق کسب‌وکار (مثلاً محاسبه قیمت سبد خرید، بررسی اعتبار کاربر) است. |
| 3. پایگاه داده (Database) | | انبار داده برنامه است. تمام اطلاعات مهم مانند اطلاعات کاربران، محصولات، مقالات و تراکنش‌ها در اینجا ذخیره، مدیریت و بازیابی می‌شوند. مثال‌ها: MySQL, PostgreSQL, MongoDB. |
| Application Programming Interface API | پل ارتباطی بین فرانت‌اند و بک‌اند است. API一 قوانین و protocols (پروتکل‌هایی) است که مشخص می‌کند فرانت‌اند چگونه می‌تواند از بک‌اند درخواست داده کند یا داده ارسال کند (معمولاً از طریق REST یا GraphQL ). |

یک مقایسه ساده:

یک نمایش عروسکی را در نظر بگیرید:

فرانت‌اند: عروسک‌های زیبا و صحنه‌ای که تماشاگران می‌بینند.

بک‌اند: عروسک‌گردان‌ها، نویسنده داستان، نورپرداز و تمام عوامل پشت صحنه که نمایش را ممکن می‌کنند.

مهارت‌های مورد نیاز یک توسعه‌دهنده بک‌اند

یک توسعه‌دهنده بک‌اند موفق به مهارت‌های فنی و مفهومی خاصی نیاز دارد:

1. تسلط بر یک زبان برنامه‌نویسی سمت سرور: مانند Python (با فریم‌ورک Django یا Flask)، JavaScript (با فریم‌ورک Node.js/Express)، PHP (با فریم‌ورک Laravel یا Symfony)، Java (با Spring) یا C (با .NET).

2. دانش پایگاه‌های داده و SQL: توانایی Design (طراحی) ، ایجاد و نوشتن Query های بهینه برای پایگاه‌های داده رابطه‌ای (مانند MySQL) و/یا غیررابطه‌ای (NoSQL مانند MongoDB).

3. کار با APIها: توانایی Design و توسعه API های RESTful یا GraphQL که قابل اعتماد، امن و well documented (دارای مستندات خوب) باشند.

4. امنیت (Security): آگاهی از حفره‌های امنیتی رایج (مثل SQL Injection, XSS, CSRF) و روش‌های مقابله با آنها. مدیریت احراز هویت (Authentication) و مجوزدهی (Authorization) کاربران.

5. مدیریت سرور و Deployment (استقرار): آشنایی با مبانی DevOps مانند کار با سرورهای لینوکس، استفاده از Docker برای containerization، و platforms (پلتفرم‌هایی) مانند AWS , Azure یا Google Cloud برای میزبانی برنامه.

6. حل مسئله و تفکر الگوریتمی: از آنجایی که بک‌اند با منطق و داده‌های حجیم سروکار دارد، توانایی حل مسئله و نوشتن الگوریتم‌های بهینه بسیار crucial (حیاتی) است.

# معرفی HTML:

HTML چیست؟

HTML مخفف HyperText Markup Language (زبان نشانه‌گذاری ابرمتنی) است. HTML یک زبان برنامه‌نویسی نیست، بلکه یک زبان نشانه‌گذاری (Markup Language) است که برای ساختاردهی و سازماندهی محتوای صفحات وب استفاده می‌شود.

به بیان ساده، HTML اسکلت و ساختار اصلی یک صفحه وب را می‌سازد. تمام متون، تصاویر، لینک‌ها، جدول‌ها و عناصری که در مرورگر خود می‌بینید، ابتدا توسط HTML تعریف و ساخته می‌شوند.

HTML چگونه کار می‌کند؟

HTML از المان‌هایی به نام تگ (Tag) تشکیل شده است. هر تگ محتوای داخل خود را تعریف می‌کند. مرورگر این تگ‌ها را می‌خواند و سپس بر اساس آنها، محتوای صفحه را Render (نمایش) می‌دهد.

مثال یک تگ ساده:

|  |
| --- |
| <p>      این یک پاراگراف است.  </p> |

`<p>`: تگ باز کردن (شروع پاراگراف)

`این یک پاراگراف است.`: محتوای قابل مشاهده

`</p>`: تگ بستن (پایان پاراگراف)

ساختار اصلی یک سند HTML

هر صفحه HTML دارای یک ساختار پایه و استاندارد است:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>      <meta charset="UTF8">      <meta name="viewport" content="width=devicewidth, initialscale=1.0">      <title>عنوان صفحه من</title>  </head>  <body>      <! تمام محتوای قابل مشاهده صفحه اینجا قرار می‌گیرد >      <h1>این یک عنوان مهم است</h1>      <p>این یک پاراگراف نمونه است.</p>  </body>  </html> |

توضیح بخش‌های مختلف:

1. <!DOCTYPE html>`: به مرورگر می‌گوید که این سند از نوع HTML5 است.

2. <html>`: تگ ریشه (Root) صفحه است. ویژگی `lang="fa"` زبان صفحه را فارسی و `dir="rtl"` جهت نوشتار را راست‌چین (RighttoLeft) تعیین می‌کند.

3. <head>`: حاوی MetaInformation (اطلاعات فرامیانی) درباره صفحه است که به کاربر نمایش داده نمی‌شود (مانند عنوان صفحه در تب مرورگر، لینک به فایل‌های CSS و JavaScript، کلمات کلیدی برای موتورهای جستجو).

4. <body>`: حاوی تمام محتوای قابل مشاهده صفحه است. هر چیزی که کاربر در مرورگر می‌بیند، داخل این تگ قرار می‌گیرد.

5. < comment !>`: این یک توضیح (Comment) است. مرورگر آن را نادیده می‌گیرد و فقط برای توسعه‌دهندگان قابل مشاهده است.

معرفی مهم‌ترین تگ‌های HTML

تگ‌های HTML برای اهداف مختلفی استفاده می‌شوند. در جدول زیر برخی از پرکاربردترین آن‌ها را مشاهده می‌کنید:

|  |  |  |  |
| --- | --- | --- | --- |
| دسته‌بندی | تگ | توضیح | مثال |
| ساختار صفحه | >header>`, `<footer>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<div> | برای سازماندهی و تقسیم‌بندی بخش‌های مختلف صفحه به کار می‌روند | >header>  …  </header> |
| سرتیترها | (Headings) `<h1>`,  `<h2>`, `<h3>`,  `<h4>`, `<h5>`, `<h6> | برای ایجاد عنوان استفاده می‌شوند. `<h1>` مهم‌ترین و بزرگ‌ترین و `<h6>` کم‌اهمیت‌ترین و کوچک‌ترین است. | >h1> عنوان اصلی </h1> |
| متن‌ها | >p< | برای ایجاد پاراگراف | `>p< متن پاراگراف.  >p< |
| لینک‌ها | >a< | برای ایجاد هایپرلینک به صفحات دیگر. | >a href="<https://google.com><"  گوگل  />a< |
| تصاویر | >img< | برای قرار دادن تصویر. این تگ selfclosing است (تگ بسته ندارد). | >img src="image.jpg" alt<"توضیح تصویر"= |
| لیست‌ها | >ul>`, `<ol>`,  `>li< | برای ایجاد لیست‌های ترتیبی (Ordered) و غیرترتیبی | Unordered: ` <ul> <li<  آیتم  >/li< >li>  آیتم ۱  </li>  </ul< |
| جدول‌ها | >table>`, `<tr>`, `<td>`,  `>th< | برای ایجاد و نمایش داده‌ها در قالب جدول. | >table> <tr> <td> داده </td> </tr> </table> |
| فرم‌ها | >form>`, `<input>`, `<button>`, `<textarea>`, `<select> | برای دریافت اطلاعات از کاربر (مانند فرم تماس یا فرم لاگین). | >form> <input type="text"> <button>  ارسال </button> </form> |

ویژگی‌های تگ‌ها (Attributes)

هر تگ می‌تواند ویژگی‌هایی (Attributes) داشته باشد که اطلاعات بیشتری درباره آن تگ ارائه می‌دهند. ویژگی‌ها همیشه در تگ بازکننده و به صورت `name="value"` نوشته می‌شوند.

مثال‌های رایج:

href` در تگ `<a>`: آدرس مقصد لینک را مشخص می‌کند.

|  |
| --- |
| <a href="https://www.example.com">  این یک لینک است  </a> |

src` و alt` در تگ <img src=’’ alt=’’> مسیر تصویر و `alt` متن جایگزین برای تصویر (در صورت عدم نمایش) را مشخص می‌کند.

|  |
| --- |
| <img src="cat.jpg" alt="یک گربه ناز"> |

class` و id`: برای style دادن به عناصر با CSS یا دسترسی به آنها با JavaScript استفاده می‌شوند.

|  |
| --- |
| <p class="importanttext" id="mainparagraph">  این متن مهم است.  </p> |

# [معرفی Css](webdesign/css):

CSS چیست؟

CSS مخفف Cascading Style Sheets (شیوه‌نامه آبشاری) است. CSS یک زبان استایل‌دهی است که ظاهر، چیدمان و نمایش محتوای HTML را توصیف می‌کند.

به بیان ساده، اگر HTML را اسکلت و ساختار یک ساختمان در نظر بگیریم، CSS نما، رنگ دیوارها، دکوراسیون و چیدمان داخلی آن است. CSS است که یک صفحه متنی ساده را به یک وب‌سایت زیبا و جذاب تبدیل می‌کند.

CSS چگونه کار می‌کند؟

CSS با انتخاب المان‌های HTML و اعمال استایل‌های مختلف (مانند رنگ، فونت، اندازه، موقعیت) روی آنها کار می‌کند. این کار با استفاده از قوانینی به نام Rule Set انجام می‌شود.

ساختار یک قانون CSS:

|  |
| --- |
| selector {      property: value;      property: value;  } |

Selector (انتخاب‌گر): مشخص می‌کند که کدام المان(های) HTML باید استایل بگیرند.

Property (ویژگی): مشخص می‌کند کدام aspect (جنبه) از المان را می‌خواهید تغییر دهید (مثلاً `color` برای رنگ متن).

Value (مقدار): مشخص می‌کند چگونه می‌خواهید آن ویژگی را تغییر دهید (مثلاً `red` برای رنگ قرمز).

Declaration (اعلامیه): هر جفت `property: value` یک اعلامیه است.

مثال:

|  |
| --- |
| p {              color: blue;              font-size: 16px;          } |

[Declaration](webdesign/css/Declaration/Declaration.html)

این کد به مرورگر می‌گوید: "تمام تگ‌های `<p>` (پاراگراف) را با رنگ آبی (`blue`) و اندازه فونت 16 پیکسل (`16px) نمایش بده."

روش‌های اتصال CSS به HTML

سه روش اصلی برای اضافه کردن استایل‌های CSS به یک صفحه وب وجود دارد:

|  |  |  |
| --- | --- | --- |
| روش | توضیح | مثال |
| | 1. External CSS (خارجی) | بهترین و توصیه‌شده‌ترین روش. قوانین CSS در یک فایل جداگانه با پسوند `.css` ذخیره می‌شوند و سپس به HTML لینک می‌شوند. | در فایل  style.css:  p { color: red; }  درفایل  HTML:  <link rel="stylesheet" href="style.css"> |
| 2. Internal CSS (داخلی) | قوانین CSS داخل تگ `<>style>` در بخش `<>head>` سند HTML نوشته می‌شوند. | >head> <style>  p { color: red; } </style>  </head> |
| 3. Inline CSS (درون‌خطی) | استایل مستقیماً به یک المان HTML خاص با استفاده از attribute (ویژگی) `style` اضافه می‌شود. این روش کمتر توصیه می‌شود. | >p style="color: red">  این متن قرمز است.  </p> |

مهم‌ترین مفاهیم و ویژگی‌های CSS

1. Selectors (انتخاب‌گرها)

انتخاب‌گرها مشخص می‌کنند که استایل روی کدام المان‌ها اعمال شود.

|  |  |  |
| --- | --- | --- |
| نوع انتخاب‌گر | مثال | توضیح |
| Element Selector | P | تمام تگ‌های `<p>` را انتخاب می‌کند. |
| Class Selector | .myclass | تمام المان‌هایی که class="myclass"  دارند را انتخاب می‌کند (متدوال‌ترین روش). |
| ID Selector | #myid | المانی که  id="myid"  دارد را انتخاب می‌کند (منحصر به فرد) |
| Descendant Selector | div p | تمام تگ‌های  <p>  که درون یک  <div>    قرار دارند را انتخاب می‌کند. |

2. Box Model (مدل جعبه‌ای)

این یکی از مهم‌ترین مفاهیم CSS است. هر المان HTML در CSS به عنوان یک جعبه (Box) در نظر گرفته می‌شود که از چهار بخش تشکیل شده است:

Content (محتوای اصلی)

Padding (فاصله داخلی) بین محتوا و border

Border (حاشیه)

Margin (فاصله خارجی) بین border و المان‌های دیگر

|  |
| --- |
| div {      width: 300px; / عرض محتوا /      padding: 20px; / فاصله داخلی /      border: 5px solid black; / حاشیه /      margin: 10px; / فاصله خارجی /  } |

[Box Model](webdesign/css/BoxModel/index.html)

3. Layout (چیدمان)

کنترل موقعیت و چیدمان المان‌ها در صفحه.

Flexbox: یک مدل layout یک‌بعدی برای طراحی چیدمان‌های flexible (انعطاف‌پذیر) و responsive (واکنش‌گرا).

|  |
| --- |
| .container {  display: flex;  justifycontent: center;  alignitems: center;  } |

CSS Grid: یک مدل layout دو‌بعدی برای ایجاد چیدمان‌های شبکه‌ای پیچیده.

|  |
| --- |
| .container {  display: grid;  gridtemplatecolumns: 1fr 1fr 1fr;  gap: 10px;  } |

4. Responsive Design (طراحی واکنش‌گرا)

وب‌سایت در تمام دستگاه‌ها (موبایل، تبلت، دسکتاپ) به خوبی نمایش داده شود. این کار principalmente با Media Queries انجام می‌شود.

|  |
| --- |
| / استایل برای صفحه‌های با عرض کمتر از 600px (موبایل) /  @media (maxwidth: 600px) {  body {  backgroundcolor: lightblue;  }  .menu {  flexdirection: column;  }  } |

ویژگی‌های پرکاربرد CSS

|  |  |  |  |
| --- | --- | --- | --- |
| دسته‌بندی | ویژگی | توضیح | مثال |
| متن و فونت | color | رنگ متن | color: ff0000; |
|  | fontfamily | نوع فونت | fontfamily: Arial; |
|  | fontsize | اندازه فونت | fontsize: 18px; |
|  | textalign | تراز متن | textalign: center; |
| پس‌زمینه | backgroundcolor | رنگ پس‌زمینه | backgroundcolor: blue; |
|  | backgroundimage | تصویر پس‌زمینه | backgroundimage: url('image.jpg'); |
| ابعاد و فاصله | width / height | عرض / ارتفاع | width: 100px; |
|  | padding | فاصله داخلی | padding: 10px; |
|  | margin | فاصله خارجی | margin: 20px; |
| حاشیه | border | حاشیه | border: 1px solid black; |
| موقعیت‌یابی | display | نوع نمایش | display: none; display: block; |
|  | position | نوع موقعیت‌یابی | position: relative; |

# معرفی JavaScript

JavaScript چیست؟JavaScript یک زبان برنامه‌نویسی سطح بالا، پویا و مفسری است که عمدتاً برای توسعه وب استفاده می‌شود. این زبان به شما امکان می‌دهد صفحات وب تعاملی ایجاد کنید.

ویژگی‌های کلیدی JavaScript:

سطح بالا: نحو خوانا و نزدیک به زبان انسان

پویا: نوع‌دهی پویا (Dynamic Typing)

مفسری: بدون نیاز به کامپایل قبل از اجرا

# بررسی مفهوم SEO

بررسی کامل مفهوم SEO (بهینه‌سازی موتورهای جستجو)

تعریف SEO

SEO (Search Engine Optimization) به مجموعه‌ای از تکنیک‌ها و استراتژی‌ها گفته می‌شود که برای بهبود رتبه‌بندی یک وب‌سایت در نتایج جستجوی طبیعی (غیرپولی) موتورهای جستجو مانند گوگل استفاده می‌شود.

اهداف اصلی SEO

- افزایش visibility (قابلیت دیده شدن)

- جذب ترافیک ارگانیک (رایگان)

- افزایش نرخ تبدیل (Conversion Rate)

- بناء برند (Brand Building)

سه رکن اصلی SEO

1. سئوی فنی (Technical SEO)

• بهینه‌سازی سرعت سایت

• ساختار URL بهینه

• طراحی ریسپانسیو (موبایل فرندلی)

• استفاده از SSL/HTTPS

• نقشه سایت (Sitemap)

• معماری اطلاعات مناسب

2. سئوی محتوا (On-Page SEO)

• تحقیق کلمات کلیدی (Keyword Research)

• تولید محتوای با کیفیت و ارزشمند

• بهینه‌سازی تگ‌های عنوان (Title Tags)

• متا دیسکریپشن (Meta Descriptions)

• استفاده از هدینگ‌های مناسب (H1, H2, H3)

• بهینه‌سازی تصاویر (Alt Text)

3. سئوی خارجی (Off-Page SEO)

• لینک‌سازی خارجی (Backlinks)

• فعالیت در شبکه‌های اجتماعی

• برندسازی (Branding)

• نظرات کاربران (Reviews)

• مشارکت در فروم‌ها و انجمن‌ها

الگوریتم‌های مهم گوگل

|  |  |
| --- | --- |
| الگوریتم | هدف |
| Panda | مبارزه با محتوای کم‌کیفیت |
| Penguin | مبارزه با لینک‌های اسپم |
| Hummingbird | درک بهتر intent کاربر |
| BERT | درک زبان طبیعی |
| Core Updates | به روزرسانی‌های کلی |

**مراحل اجرای پروژه SEO**

مرحله ۱: تحلیل و بررسی

- آنالیز رقبا

- Audit فنی سایت

- تحلیل کلمات کلیدی

**مرحله ۲: بهینه‌سازی**

- رفع مشکلات فنی

- تولید و بهینه‌سازی محتوا

- لینک‌سازی داخلی

**مرحله ۳: لینک‌سازی**

- ساخت بک‌لینک‌های باکیفیت

- فعالیت در شبکه‌های اجتماعی

**مرحله ۴: مانیتورینگ**

- رصد رتبه‌ها

- تحلیل ترافیک

- گزارش‌گیری منظم

سئوی کلاه سیاه (منفی)

- کلیک‌فروشی (Click Fraud)

- محتوای کپی (Duplicate Content)

- لینک‌های اسپم (Spam Links)

- مخفی‌سازی محتوا (Cloaking)

روندهای جدید SEO در ۲۰۲۴

- تجربه کاربری (UX) اهمیت بیشتر

- جستجوی صوتی (Voice Search)

- AI و محتوای هوشمند

- E-A-T (تخصص، اعتماد، Authority)

- Core Web Vitals (معیارهای سرعت)

ابزارهای کاربردی SEO

- Google Search Console

- Google Analytics

- Ahrefs/SEMrush

- Screaming Frog

- GTmetrix

نکات کلیدی موفقیت در SEO

1. صبر و تحمل (فرآیند زمان‌بر است)

2. تمرکز بر کاربر (User-First)

3. به روز بودن (الگوریتم‌ها دائماً تغییر می‌کنند)

4. تحلیل داده‌محور

5. یکپارچگی با بازاریابی دیجیتال

SEO یک فرآیند مستمر است که نیاز به استراتژی بلندمدت، تخصص فنی و درک عمیق از رفتار کاربران دارد.

# بررسی مفهوم Progressive Enhancement بهبود تدریجی

بررسی مفهوم Progressive Enhancement بهبود تدریجی

تصور کنید می‌خواهید یک ساندویچ درست کنید:

Progressive Enhancement یعنی:

1. اول نان بگذارید (پایه اصلی)

2. بعد کره بمالید (اضافه کردن قابلیت اول)

3. سپس پنیر بگذارید (قابلیت دوم)

4. در آخر گوجه اضافه کنید (قابلیت پیشرفته)

نتیجه: حتی اگر گوجه نداشته باشید، باز هم یک ساندویچ دارید!

فلسفه اصلی به زبان ساده

"اول مطمئن شویم همه می‌توانند از پایه‌ای ترین حالت استفاده کنند، سپس برای کسانی که امکانات بهتر دارند، تجربه بهتری بسازیم."

مثال عملی: ساخت یک وبسایت خبری

مرحله ۱: پایه اصلی (همه می‌بینند)

کاربران با: موبایل قدیمی، اینترنت کم سرعت، مرورگر ساده

مرحله ۲: اضافه کردن استایل (زیبایی)

/ مثل رنگ کردن خانه /

کاربران با: مرورگرهای نسبتاً جدید

مرحله ۳: قابلیت‌های پیشرفته

// مثل نصب آسانسور در ساختمان

// نمایش نظرات بدون رفرش صفحه

// پیشنهاد اخبار مشابه

کاربران با: مرورگرهای جدید، اینترنت پرسرعت

مقایسه با دو روش دیگر

روش ۱: Progressive Enhancement (ما)

- اول متن ساده ⭠ سپس رنگ ⭠بعد انیمیشن

- مزیت: همه می‌توانند بخوانند

روش ۲: Graceful Degradation (برعکس ما)

- اول انیمیشن پیچیده اگر نشد،⭠ متن ساده نشان بده

- مشکل: ممکن است بعضی کاربران اصلاً محتوا نبینند

مثال‌های واقعی از زندگی

مثال ۱: ارسال پیام

- پایه: SMS ساده

- میانی: پیام با emoji

- پیشرفته: ویدیو، موقعیت مکانی، reactions

مثال ۲: تماشای فیلم

- پایه: دانلود با کیفیت پایین

- میانی: استریمینگ با کیفیت معمولی

- پیشرفته: 4K, HDR, واقعیت مجازی

نمودار درک مطلب

|  |
| --- |
| کاربران با امکانات محدود  ↓  محتوا و عملکرد اصلی ← همه می‌توانند استفاده کنند  ↓  کاربران با امکانات متوسط  ↓  ظاهر زیباتر + قابلیت‌های بیشتر  ↓  کاربران با امکانات پیشرفته  ↓  تجربه فوق‌العاده و تعاملی |

چرا این روش مهم است؟

۱. عدالت دیجیتال

- کاربران روستایی با اینترنت کند

- افراد با دستگاه‌های قدیمی

- افراد با معلولیت (نابینایان با screen reader)

۲. آمادگی برای شرایط مختلف

- وقتی اینترنت قطع می‌شود

- وقتی JavaScript load نمی‌شود

- وقتی مرورگر قدیمی است

۳. ساخت پایه محکم

- مثل ساختن ساختمان: اول اسکلت محکم، سپس نماکاری

تمرین فکری برای دانشجویان

سناریو: ساخت یک فرم ثبت‌نام

سوال: چگونه این فرم را با Progressive Enhancement بسازیم؟

پاسخ مرحله به مرحله:

1. پایه: فرم HTML ساده که بدون JavaScript کار می‌کند

2. مرحله ۲: اضافه کردن CSS برای زیبایی

3. مرحله ۳: JavaScript برای validation لحظه‌ای

4. مرحله ۴: autocomplete پیشرفته

**نکات کلیدی برای یادآوری**

1. اول مفید، سپس زیبا، بعد پیشرفته

2. همیشه به کاربران با امکانات محدود فکر کنید

3. پایه محکم = تجربه قابل اعتماد برای همه

4. این روش شبیه زندگی واقعی است

جمع‌بندی برای مبتدیان

Progressive Enhancement یعنی:

"ما یک محصول دیجیتال را طوری می‌سازیم که اول برای همه قابل استفاده باشد، سپس برای کسانی که امکانات بهتری دارند، تجربه بهتری ایجاد کنیم."

این روش نه تنها فنی، بلکه انسانی و منصفانه است!

# بررسی مفهوم Responsive Web Design

بررسی مفهوم Responsive Web Design برای مبتدیان مطلق

توضیح با یک قیاس ساده: مبلمان اتاق

تصور کنید می‌خواهید اتاقی را با مبلمان چیدمان کنید:

اتاق کوچک (موبایل):

- یک مبل کوچک

- میز عسلی کوچک

- چیدمان ساده و خطی

اتاق متوسط (تبلت):

- یک مبل 3 نفره

- دو مبل تک نفره

- میز بزرگتر

اتاق بزرگ (دسکتاپ):

- چندین مبل بزرگ

- میز ناهارخوری

- گلدان و دکوراسیون اضافی

وب‌سایت ریسپانسیو هم دقیقاً همین کار را می‌کند!

مفهوم اصلی به زبان کودکانه

"یک وب‌سایت ریسپانسیو مثل شنل جادویی است که وقتی روی صفحه‌های مختلف نمایش داده می‌شود، شکل و اندازه‌اش عوض می‌کند!"

مشکل قدیمی را ببینید

وب‌سایت‌های قدیمی (غیر-ریسپانسیو):

<!-- مثل یک میز ثابت که نمی‌شود کوچکش کرد -->

موبایل: [=== باید اسکرول کنی ===]

دسکتاپ: [محتوای کامل دیده می‌شود]

وب‌سایت‌های جدید (ریسپانسیو):

<!-- مثل آب که شکل ظرف را می‌گیرد -->

موبایل: [محتوا]

[به صورت]

[عمودی]

دسکتاپ: [محتوا به صورت افقی کنار هم]

سه ابرقدرت Responsive Design

1. قدرت انعطاف‌پذیری (Flexible Layout)

- مثل کش لاستیکی که می‌تواند کوچک و بزرگ شود

- عناصر صفحه می‌توانند نسبت به صفحه بزرگ و کوچک شوند

2. قدرت شرطی (Media Queries)

- مثل دماسنج هوشمند که:

- اگر هوا سرد شد، بخاری روشن کند

- اگر هوا گرم شد، کولر روشن کند

- اگر صفحه کوچک بود، چیدمان موبایل نشان دهد

- اگر صفحه بزرگ بود، چیدمان دسکتاپ نشان دهد

3. قدرت تصاویر جادویی (Flexible Images)

- تصاویر مثل آب در لیوان می‌شوند

- خودکار بزرگ و کوچک می‌شوند

یک داستان تصویری

داستان: سفر یک وب‌سایت به دستگاه‌های مختلف

صحنه ۱: ورود به موبایل کوچک

- منو تبدیل به دکمه ☰ می‌شود

- عکس‌ها کوچک و شفاف می‌شوند

- متن‌ها در ستون باریک نمایش داده می‌شوند

صحنه ۲: ورود به تبلت

- منو به صورت افقی نمایش داده می‌شود

- دو ستون اطلاعات کنار هم قرار می‌گیرند

- عکس‌ها متوسط می‌شوند

صحنه ۳: ورود به دسکتاپ

- منو کامل با زیرمنوها نمایش داده می‌شود

- چندین ستون کنار هم قرار می‌گیرند

- عکس‌ها بزرگ و با کیفیت نمایش داده می‌شوند

نمودار درک مطلب برای دانشجویان

|  |
| --- |
| صفحه نمایش کوچک (موبایل)  ↓  محتوای اصلی + چیدمان عمودی  ↓  صفحه نمایش متوسط (تبلت)  ↓  چیدمان ۲ ستونی + منوی ساده  ↓  صفحه نمایش بزرگ (دسکتاپ)  ↓  چیدمان چندستونی + منوی کامل |

تمرین فکری: طراحی یک روزنامه

سوال: چگونه یک روزنامه را ریسپانسیو کنیم؟

پاسخ مرحله به مرحله:

1. موبایل:

- تیترها بزرگ

- ستون‌ها عمودی پشت سر هم

- عکس‌ها کامل عرض

2. تبلت:

- دو ستون کنار هم

- عکس‌ها متوسط

- فونت‌ها متوسط

3. دسکتاپ:

- چندین ستون کنار هم

- عکس‌ها بزرگ

- منوهای کناری

مثال‌های واقعی از زندگی

مثال ۱: سینما vs تلویزیون

- سینما (دسکتاپ): صفحه عریض، کیفیت بالا

- تلویزیون (تبلت): صفحه متوسط، کیفیت خوب

- موبایل (موبایل): صفحه کوچک، کیفیت مناسب

مثال ۲: کتاب چاپی vs کتاب الکترونیکی

- کتاب چاپی: layout ثابت، اندازه ثابت

- کتاب الکترونیکی: متن خودکار با صفحه سازگار می‌شود

چرا Responsive Design مهم است؟

۱. کاربران مختلف، دستگاه‌های مختلف

- پدربزرگ با تبلت

- پدر با دسکتاپ

- فرزند با موبایل

۲. آماده برای آینده

- دستگاه‌های جدید با اندازه‌های مختلف

- تلویزیون‌های هوشمند

- ساعت‌های هوشمند

۳. تجربه کاربری بهتر

- نیازی به زوم کردن نیست

- خواندن راحت‌تر است

- navigation ساده‌تر است

جمع‌بندی نهایی

Responsive Design یعنی:

"یک وب‌سایت باهوش که می‌داند روی چه دستگاهی نمایش داده می‌شود و خودش را با آن دستگاه وفق می‌دهد!"

سه کلید طلایی برای یادآوری:

1. انعطاف‌پذیر باش (مثل آب شکل ظرف را بگیر)

2. هوشمند باش (شرایط مختلف را تشخیص بده)

3. ساده باش (همه بتوانند استفاده کنند)

این مفهوم یکی از مهم‌ترین مهارت‌ها در طراحی وب امروزی است!

# بررسی مفهوم Internet

بررسی مفهوم اینترنت (Internet) برای مبتدیان

تعریف ساده اینترنت

اینترنت مثل یک شبکه عظیم جهانی از جاده‌های اطلاعاتی است که کامپیوترهای سراسر جهان را به هم وصل می‌کند.

قیاس ساده: شبکه پست و نامه‌رسانی

تصور کنید اینترنت مثل سیستم پست بین‌المللی است:

- کامپیوترها = خانه‌های مردم

- داده‌ها = نامه‌ها و بسته‌ها

- روترها = اداره‌های پست

- کابل‌ها = جاده‌ها و مسیرهای حمل و نقل

اینترنت چگونه کار می‌کند؟ - یک داستان

داستان ارسال یک عکس از تهران به اصفهان:

1. فرستنده (تهران): عکس را به قطعات کوچک تقسیم می‌کند

2. بسته‌بندی: هر قطعه در یک پاکت با آدرس می‌رود

3. مسیریابی: هر پاکت از چند اداره پست (روتر) عبور می‌کند

4. دریافت (اصفهان): قطعات دوباره جمع می‌شوند و عکس ساخته می‌شود

اجزای اصلی اینترنت

1. سرورها (Servers) - کتابخانه‌های عمومی

- کامپیوترهای همیشه روشن

- اطلاعات را ذخیره و ارائه می‌دهند

- مثل: گوگل، فیسبوک، یوتیوب

2. کلاینت‌ها (Clients) - شما و من

- کامپیوترها، موبایل‌ها، تبلت‌ها

- اطلاعات را request می‌کنند

- مثل: لپ‌تاپ شما، موبایل دوستتان

3. روترها (Routers) - ایستگاه‌های راهنمایی

- داده‌ها را به مسیر درست هدایت می‌کنند

- از ترافیک جلوگیری می‌کنند

4. کابل‌ها (Cables) - جاده‌های اطلاعات

- کابل‌های زمینی، زیردریایی

- داده‌ها را حمل می‌کنند

انواع اتصال به اینترنت

1. ADSL - خط تلفن قدیمی

- مثل استفاده از همان لوله آب برای آشپزی و شستشو

- سرعت متوسط، در دسترس همه

2. فیبر نوری - لوله‌های جدید

- مثل اتوبان 8 بانده

- سرعت بسیار بالا

3. وای‌فای (WiFi) - بی‌سیم

- مثل ارتباط رادیویی

- در محدوده مشخص کار می‌کند

4. دیتا موبایل - اینترنت همراه

- مثل ارتباط تلفنی

- همه جا در دسترس

تاریخچه مختصر اینترنت

دهه 1960: تولد در جنگ سرد

- آمریکا می‌خواست شبکه‌ای بسازد که حتی با حمله اتمی هم کار کند

- نام اولیه: ARPANET

دهه 1980: شبکه دانشگاهی

- دانشگاه‌ها به هم وصل شدند

- ایمیل اختراع شد

دهه 1990: وب جهانی متولد شد

- تیم برنرزلی WWW را اختراع کرد

- مرورگرها ساخته شدند

دهه 2000: انفجار اینترنت

- گوگل، فیسبوک، یوتیوب

- اینترنت پرسرعت

امروز: اینترنت اشیا

- یخچال، ماشین، خانه هوشمند

- همه چیز به اینترنت وصل است

پروتکل‌ها - قوانین اینترنت

HTTP/HTTPS - زبان وب‌سایت‌ها

- مثل زبان رسمی برای صحبت با وب‌سایت‌ها

- HTTPS = HTTP + رمزنگاری (امن)

TCP/IP - زبان اصلی اینترنت

- مثل قوانین رانندگی بین‌المللی

- همه باید این زبان را بلد باشند

SMTP - زبان ایمیل

- مثل قوانین پست برای نامه‌ها

خدمات اصلی اینترنت

1. وب جهانی (WWW)

- وب‌سایت‌ها و صفحات

- مثل: گوگل، آمازون

2. پست الکترونیک (Email)

- ارسال نامه دیجیتال

- مثل: Gmail, Outlook

3. رسانه‌های اجتماعی

- ارتباط با دیگران

- مثل: اینستاگرام، تلگرام

4. ذخیره‌سازی ابری

- نگهداری فایل در اینترنت

- مثل: Google Drive, Dropbox

اینترنت چگونه زندگی ما را تغییر داده؟

ارتباطات:

- قبل: نامه 2 هفته‌ای

- بعد: پیام فوری 2 ثانیه‌ای

اطلاعات:

- قبل: رفتن به کتابخانه

- بعد: جستجوی 2 ثانیه‌ای در گوگل

کسب و کار:

- قبل: مغازه فیزیکی

- بعد: فروشگاه اینترنتی

آموزش:

- قبل: فقط کلاس حضوری

- بعد: دوره‌های آنلاین

امنیت در اینترنت - مثل خانه خود

قفل‌های دیجیتال:

- رمز عبور قوی = کلید محکم

- آنتی‌ویروس = سیستم alarm

- فایروال = دیوار محافظ

خطرات:

- هکرها = سارقان دیجیتال

- ویروس‌ها = بیماری‌های کامپیوتری

- فیشینگ = کلاهبرداری اینترنتی

آینده اینترنت

1. اینترنت 5G

- سرعت باورنکردنی

- تاخیر بسیار کم

2. اینترنت اشیا (IoT)

- همه دستگاه‌ها متصل

- خانه‌های هوشمند

3. واقعیت مجازی

- دنیاهای دیجیتال

- تجربیات غوطه‌وری

4. هوش مصنوعی

- اینترنت هوشمندتر

- پاسخ‌های شخصی‌سازی شده

نکات مهم برای مبتدیان

اینترنت چیست:

- شبکه جهانی کامپیوترها

- سیستم انتقال اطلاعات

- ابزار ارتباط و اطلاعات

اینترنت چیست نیست:

- یک کامپیوتر غول‌پیکر نیست

- یک مکان فیزیکی نیست

- فقط وب‌سایت نیست

جمع‌بندی نهایی

اینترنت مثل سیستم عصبی زمین است:

- مغز: سرورهای بزرگ

- اعصاب: کابل‌ها و امواج

- حواس: کاربران (ما)

سه نکته طلایی:

1. اینترنت شبکه‌ای از شبکه‌هاست

2. همه چیز با قوانین مشخص کار می‌کند

3. ابزاری قدرتمند برای ارتباط و اطلاعات است

اینترنت یکی از مهم‌ترین اختراعات بشر است که جهان را به دهکده‌ای جهانی تبدیل کرده است!

# بررسی مفهوم Protocol

تعریف ساده پروتکل

پروتکل یک سری قوانین و مقررات است که مشخص می‌کند چگونه دو یا چند سیستم با هم ارتباط برقرار کنند.

قیاس روزمره: قوانین رانندگی

تصور کنید پروتکل مثل قوانین راهنمایی و رانندگی است:

|  |  |
| --- | --- |
| قوانین رانندگی | معادل در پروتکل |
| چراغ قرمز = توقف | ERROR = Stop |
| خط کشی = مسیر حرکت | PATH = Route |
| بوق = سیگنال | SIGNAL = Alert |
| گواهینامه = مجوز | AUTH = Authentication |

پروتکل در ارتباطات - یک داستان

داستان دو نفر که می‌خواهند با زبان‌های مختلف صحبت کنند:

بدون پروتکل:

- 👨‍💼: "Hello!" (انگلیسی)

- 👩‍💼: "؟من متوجه نمی‌شوم" (فارسی)

- نتیجه: ارتباط ناموفق

با پروتکل:

- 👨‍💼: "Hello!" (انگلیسی)

- 👩‍💼: "Hello!" (انگلیسی - طبق پروتکل)

- نتیجه: ارتباط موفق

**چرا پروتکل لازم است؟**

### 1. زبان مشترک

- کامپیوترهای مختلف (ویندوز، مک، لینوکس)

- دستگاه‌های مختلف (موبایل، تبلت، کامپیوتر)

- همه باید یک زبان را بفهمند

2. قابل پیش‌بینی بودن

- مثل دستورالعمل آشپزی

- همه مراحل مشخص است

- نتیجه یکسان می‌شود

3. خطایابی آسان

- اگر مشکلی پیش بیاید، می‌دانیم کجای کار اشتباه شده

- مثل نقشه راه برای عیب‌یابی

انواع پروتکل‌های اینترنتی

1. TCP/IP - پادشاه پروتکل‌ها

مثل سیستم پست پیشرفته

- بسته‌ها را شماره‌گذاری می‌کند

- رسیدن را تأیید می‌کند

- اگر بسته گم شد، دوباره می‌فرستد

2. HTTP/HTTPS - پروتکل وب

مثل زبان وب‌سایت‌ها

HTTP: "لطفاً صفحه گوگل را بفرست"

HTTPS: "لطفاً صفحه گوگل را بفرست (رمزگذاری شده)"

3. FTP - پروتکل انتقال فایل

مثل پیک مخصوص فایل‌ها

"لطفاً این فایل 1GB را منتقل کن"

4. SMTP/POP3 - پروتکل ایمیل

مثل پستچی ایمیل

SMTP: "این ایمیل را بفرست"

POP3: "ایمیل‌های جدید را بگیر"

پروتکل در عمل: ارسال یک ایمیل

مرحله ۱: نوشتن ایمیل (SMTP)

فرستنده: ali@example.com

گیرنده: sara@test.com

موضوع: سلام

متن: چگونه هستی؟

مرحله ۲: بسته‌بندی (TCP)

بسته ۱: [هدر ایمیل]

بسته ۲: [موضوع]

بسته ۳: [متن ایمیل]

مرحله ۳: ارسال (IP)

از: IP آلی

به: IP سارا

مسیر: روتر ۱ → روتر ۲ → روتر ۳

مرحله ۴: دریافت (POP3)

سارا: "ایمیل جدید دارم؟"

سرور: "بله، یک ایمیل از علی"

ویژگی‌های یک پروتکل خوب

1. قابل اعتماد (Reliable)

- داده‌ها سالم می‌رسند

- اگر خطایی باشد، تشخیص داده می‌شود

2. کارا (Efficient)

- سریع کار می‌کند

- منابع کم مصرف می‌کند

3. امن (Secure)

- دیگران نمی‌توانند بخوانند

- هکرها نمی‌توانند تغییر دهند

4. قابل توسعه (Extensible)

- می‌توان آن را بهبود داد

- با تکنولوژی‌های جدید کار می‌کند

پروتکل‌های معروف در زندگی روزمره

1. بلوتوث - پروتکل بی‌سیم کوتاه‌برد

- هدفون بی‌سیم

- ماوس و کیبورد

- انتقال فایل بین موبایل‌ها

2. WiFi - پروتکل شبکه بی‌سیم

- اینترنت بدون سیم

- ارتباط در محدوده ساختمان

3. NFC - پروتکل ارتباط نزدیک

- پرداخت با موبایل

- کارت‌های هوشمند

مثال ساده: پروتکل دست دادن

پروتکل دست دادن بین دو نفر:

1. قدم ۱: تماس چشمی برقرار کن

2. قدم ۲: دست راست را جلو ببر

3. قدم ۳: محکم اما نه خیلی سفت بگیر

4. قدم ۴: 2-3 بار تکان بده

5. قدم ۵: رها کن

اگر کسی این پروتکل را رعایت نکند:

|  |
| --- |
| - دست دادن خیلی شل → بی‌احترامی  - دست دادن خیلی سفت → تهاجمی  - دست دادن طولانی → ناخوشایند |

پروتکل‌های امنیتی

1. SSL/TLS - پروتکل امن

- مثل حمل پول با خودروی زرهی

- اطلاعات را رمزگذاری می‌کند

- در آدرس وب می‌بینید: `https://`

2. VPN - پروتکل شبکه خصوصی

- مثل تونل امن زیرزمینی

- ارتباط را مخفی می‌کند

3. SSH - پروتکل امن برای سرور

- مثل کلید ویژه برای اتاق کنترل

- امنیت بالا برای مدیریت سرور

آینده پروتکل‌ها

1. پروتکل‌های کوانتومی

- امنیت بسیار بالا

- استفاده از فیزیک کوانتوم

2. پروتکل‌های هوش مصنوعی

- ارتباط هوشمند بین دستگاه‌ها

- پیش‌بینی نیازها

3. پروتکل‌های اینترنت اشیا

- میلیاردها دستگاه متصل

- ارتباط کارا و کم‌مصرف

نکات کلیدی برای مبتدیان

پروتکل چیست:

- مجموعه قوانین ارتباطی

- زبان مشترک بین سیستم‌ها

- تضمین کننده موفقیت ارتباط

پروتکل چیست نیست:

- یک برنامه یا نرم‌افزار نیست

- سخت‌افزار نیست

- اختیاری نیست (ضروری است)

جمع‌بندی نهایی

پروتکل مثل زبان مشترک بین ملت‌هاست:

- دستور زبان: قواعد پایه

- واژگان: پیام‌های قابل فهم

- لهجه: انواع مختلف پروتکل

سه نکته طلایی:

1. پروتکل‌ها زبان مشترک دستگاه‌ها هستند

2. بدون پروتکل‌ها، ارتباط غیرممکن است

3. هر نوع ارتباط، پروتکل مخصوص خود را دارد

پروتکل‌ها پایه و اساس تمام ارتباطات دیجیتال هستند!

# بررسی مفهوم WEB

تعریف ساده وب

وب مثل یک کتابخانه عظیم جهانی است که صفحات آن در سراسر جهان پخش شده و با هم مرتبط شده‌اند.

قیاس ساده: شهر جهانی اطلاعات

تصور کنید وب مثل یک شهر بسیار بزرگ است:

|  |  |
| --- | --- |
| بخش‌های شهر | معادل در وب |
| خانه‌ها | وب‌سایت‌ها |
| خیابان‌ها | لینک‌ها |
| تابلوهای راهنما | موتورهای جستجو |
| پست‌چی‌ها | مرورگرها |

تفاوت اینترنت و وب

- اینترنت: جاده‌ها و زیرساخت (سیم‌ها، روترها، سرورها)

- وب: ماشین‌هایی که روی جاده حرکت می‌کنند (وب‌سایت‌ها)

|  |
| --- |
| اینترنت (زیرساخت)  ↓  حمل و نقل داده‌ها  ↓  وب (محتوای قابل مشاهده) |

سه تکنولوژی اصلی وب

1. HTML - اسکلت ساختمان

<!-- مثل نقشه ساختمان -->

<title>فروشگاه من</title>

<header>سربرگ</header>

<nav>منوی</nav>

<main>محتوای اصلی</main>

<footer>پاورقی</footer>

2. CSS - دکوراسیون و زیبایی

/ مثل رنگ، مبلمان، دکور /

|  |
| --- |
| body {  background: lightblue;  font-family: Arial;  }  Header{  color: white;  background: darkblue;  } |

3. JavaScript - برق و سیستم‌های هوشمند

// مثل آسانسور، دزدگیر، سیستم روشنایی

// وقتی کاربر کلیک کند، اتفاقی بیفتد

|  |
| --- |
| button.addEventListener('click', function(){  alert('خوش آمدید!');  }); |

## تاریخچه وب به زبان ساده

1989: تولد وب

- تیم برنرزلی (Tim Berners-Lee) در CERN

- ایده: ارتباط اسناد با هایپرلینک

- اولین وب‌سایت: http://info.cern.ch

1990s: وب 1.0 - وب خواندنی

- صفحات استاتیک

- کاربر فقط می‌توانست بخواند

- مثل روزنامه آنلاین

2000s: وب 2.0 - وب خواندنی/نوشتنی

- صفحات داینامیک

- کاربران محتوا تولید می‌کنند

- مثل فیسبوک، ویکیپدیا

2010s: وب 3.0 - وب هوشمند

- هوش مصنوعی

- شخصی‌سازی محتوا

- اینترنت اشیا

انواع وب‌سایت‌ها

1. وب‌سایت استاتیک (ایستا)

- مثل بروشور کاغذی

- محتوا ثابت است

- تغییر سخت

- مثال: رزومه آنلاین

2. وب‌سایت داینامیک (پویا)

- مثل مجله زنده

- محتوا تغییر می‌کند

- کاربران تعامل می‌کنند

- مثال: شبکه‌های اجتماعی

3. وب اپلیکیشن‌ها

- مثل نرم‌افزار آنلاین

- عملکرد پیچیده

- مثال: Gmail, Google Docs

چگونه وب کار می‌کند؟ - یک داستان

داستان بازدید از یک وب‌سایت:

1. شما: آدرس را در مرورگر تایپ می‌کنید

https://www.example.com

2. مرورگر: با DNS صحبت می‌کند

"آدرس IP example.com چیست؟"

"IP آن 93.184.216.34 است"

3. مرورگر: با سرور ارتباط برقرار می‌کند

"لطفاً صفحه اصلی را بفرست"

4. سرور: صفحه را می‌فرستد

"این HTML، CSS و JavaScript شما است."

5. مرورگر: صفحه را نمایش می‌دهد

"صفحه آماده نمایش است!"

اجزای اصلی تجربه وب

1. مرورگر (Browser) - پنجره به دنیای وب

- گوگل کروم، فایرفاکس، سافاری

- مثل مترجم بین شما و وب‌سایت‌ها

2. موتور جستجو (Search Engine) - فهرست کتابخانه

- گوگل، بینگ، یاندکس

- کمک می‌کند محتوای مورد نظر را پیدا کنید

3. سرور (Server) - انبار اطلاعات

- کامپیوترهای همیشه روشن

- اطلاعات را ذخیره و ارائه می‌دهند

4. دامنه (Domain) - آدرس منحصر به فرد

- example.com

- مثل شماره پلاک خانه

تکنولوژی‌های سمت سرور vs سمت کلاینت

سمت کلاینت (Front-end) - جلوی صحنه:

- HTML: ساختار

- CSS: ظاهر

- JavaScript: رفتار

سمت سرور (Back-end) - پشت صحنه:

- پایگاه داده: ذخیره اطلاعات

- سرور: پردازش درخواست‌ها

- زبان‌های برنامه‌نویسی: PHP, Python, Java

طراحی وب مدرن - اصول مهم

1. Responsive Design - طراحی واکنش‌گرا

- صفحه در موبایل، تبلت، دسکتاپ خوب دیده می‌شود

- مثل آب که شکل ظرف را می‌گیرد

2. Accessibility - دسترسی‌پذیری

- افراد با معلولیت هم بتوانند استفاده کنند

- screen readerها بتوانند بخوانند

3. SEO - بهینه‌سازی برای موتورهای جستجو

- کمک به گوگل برای فهمیدن محتوا

- نمایش بهتر در نتایج جستجو

4. Performance - عملکرد

- سریع لود شود

- منابع کم مصرف کند

امنیت در وب

خطرات رایج:

- هک: دسترسی غیرمجاز

- فیشینگ: کلاهبرداری آنلاین

- بدافزار: نرم‌افزارهای مخرب

راه‌های محافظت:

- HTTPS: ارتباط امن

- پسورد قوی: مثل کلید محکم

- آپدیت: مثل تعمیر درب و پنجره

تاثیر وب بر زندگی ما

ارتباطات:

- قبل: نامه، تلفن

- بعد: ایمیل، شبکه‌های اجتماعی، ویدیوکال

اطلاعات:

- قبل: کتابخانه، دایرةالمعارف

- بعد: گوگل، ویکیپدیا

کسب و کار:

- قبل: مغازه فیزیکی

- بعد: فروشگاه اینترنتی

آینده وب

1. وب 3.0 - وب هوشمند

- هوش مصنوعی

- شخصی‌سازی پیشرفته

- واقعیت مجازی

2. Progressive Web Apps (PWA)

- ترکیب وب‌سایت و اپلیکیشن

- کار آفلاین

- نصب بدون استور

3. Voice Search - جستجوی صوتی

- صحبت با دستگاه‌ها

- دسترسی بدون صفحه‌کلید

4. WebAssembly

- اجرای برنامه‌های سنگین در مرورگر

- مثل نرم‌افزارهای دسکتاپ

مفاهیم کلیدی برای یادگیری

برای شروع کدنویسی:

1. HTML: ساختار صفحه

2. CSS: زیبایی و layout

3. JavaScript: تعامل و منطق

ابزارهای مفید:

- VS Code: ویرایشگر کد

- Git: کنترل نسخه

- Chrome DevTools: دیباگ کردن

جمع‌بندی نهایی

وب مثل یک جهان دیجیتال است:

- قاره‌ها: دسته‌های مختلف محتوا

- کشورها: وب‌سایت‌های مختلف

- جاده‌ها: لینک‌ها و ارتباطات

سه نکته طلایی:

1. وب روی اینترنت ساخته شده است

2. از سه تکنولوژی اصلی تشکیل شده

3. در حال تکامل و هوشمندتر شدن است

وب یکی از دموکراتیک‌ترین اختراعات بشر است که اطلاعات را در دسترس همه قرار داده است!

شروع یادگیری

اگر می‌خواهید وب را یاد بگیرید:

1. با HTML شروع کنید

2. سپس CSS را بیاموزید

3. در نهایت JavaScript را یاد بگیرید

یادگیری وب امروز = ساختن آینده فردا

# بررسی مفهوم HTTP

تعریف ساده HTTP

HTTP مثل زبان مشترک برای صحبت با وب‌سایت‌ها است. وقتی شما به یک وب‌سایت سر می‌زنید، مرورگر شما با سرور آن وب‌سایت به زبان HTTP صحبت می‌کند.

قیاس ساده: سفارش غذا در رستوران

تصور کنید HTTP مثل فرآیند سفارش غذا است:

|  |  |
| --- | --- |
| مرحله سفارش غذا | معادل در HTTP |
| مشتری منو می‌خواهد | GET /menu |
| مشتری غذا سفارش می‌دهد | POST /order |
| آشپز غذا را آماده می‌کند | Processing |
| گارسون غذا را می‌آورد | 200 OK |
| غذا تمام شده است | 404 Not Found |

HTTP چگونه کار می‌کند؟ - یک داستان

داستان بازدید از وب‌سایت:

کاربر (شما):

|  |
| --- |
| GET /index.html HTTP/1.1  Host: www.example.com |

سرور (وب‌سایت):

|  |
| --- |
| HTTP/1.1 200 OK  Content-Type: text/html  <html>  <head>  <title>  خوش آمدید  </title>  </head>  <body>  محتوا اینجاست  </body>  </html> |

ساختار پایه HTTP

1. درخواست (Request) - چیزی که می‌خواهیم

|  |
| --- |
| GET /about.html HTTP/1.1  Host: example.com  User-Agent: Chrome  Accept-Language: fa |

1. پاسخ (Response) - چیزی که می‌گیریم

|  |
| --- |
| HTTP/1.1 200 OK  Content-Type: text/html  Content-Length: 1256  <html>  ...محتوا...  </html> |

متدهای اصلی HTTP (Verbs)

1. GET - درخواست اطلاعات

- مثل پرسیدن سوال

- فقط داده می‌گیرد، چیزی تغییر نمی‌دهد

- مثال: مشاهده صفحه، دانلود فایل

2. POST - ارسال اطلاعات

- مثل پر کردن فرم

- داده جدید ایجاد می‌کند

- مثال: ثبت نام، ارسال نظر

3. PUT - آپدیت اطلاعات

- مثل تصحیح اشتباه

- داده موجود را تغییر می‌دهد

- مثال: ویرایش پروفایل

4. DELETE - حذف اطلاعات

- مثل پاره کردن کاغذ

- داده را حذف می‌کند

- مثال: حذف پست

کدهای وضعیت HTTP (Status Codes)

کدهای 1xx: اطلاعی

- `100 Continue`: ادامه بده

کدهای 2xx: موفقیت

- `200 OK`: همه چیز خوب است

- `201 Created`: چیز جدید ساخته شد

- `204 No Content`: موفق اما بدون محتوا

کدهای 3xx: تغییر مسیر

- `301 Moved Permanently`: آدرس تغییر کرده

- `304 Not Modified`: از کش استفاده کن

کدهای 4xx: خطای کلاینت

- `400 Bad Request`: درخواست بد

- `401 Unauthorized`: نیاز به لاگین

- `403 Forbidden`: دسترسی ممنوع

- `404 Not Found`: صفحه پیدا نشد

کدهای 5xx: خطای سرور

- `500 Internal Server Error`: مشکل در سرور

- `503 Service Unavailable`: سرور در دسترس نیست

تفاوت HTTP و HTTPS

HTTP - معمولی:

- مثل پستکار معمولی

- نامه‌ها قابل خواندن هستند

- مناسب برای اطلاعات عمومی

HTTPS - امن:

- مثل پستکار با خودروی زرهی

- نامه‌ها قفل شده‌اند

- مناسب برای اطلاعات محرمانه

HTTP: http://example.com

HTTPS: https://example.com

حالت‌های ارتباط HTTP

1. Stateless - بدون حافظه

- هر درخواست مستقل است

- سرور شما را به خاطر نمی‌سپارد

- مثل فروشنده غریبه

2. با کوکی‌ها - حافظه دار

- کوکی هویت شما را ذخیره می‌کند

- سرور شما را می‌شناسد

- مثل فروشنده آشنا

نسخه‌های HTTP

HTTP/1.1 (1999) - کلاسیک

- یک ارتباط برای هر درخواست

- کند اما سازگار با همه

HTTP/2 (2015) - مدرن

- چندین درخواست همزمان

- سریع‌تر و کارآمدتر

HTTP/3 (2022) - جدید

- از UDP вместо TCP

- بسیار سریع‌تر

HTTP در عمل - مثال‌های واقعی

مثال ۱: ورود به فیسبوک

1. `GET facebook.com` - صفحه لاگین

2. `POST /login` - ارسال نام کاربری و رمز

3. `200 OK` - ورود موفق

4. `GET /home` - صفحه اصلی

مثال ۲: جستجو در گوگل

1. `GET google.com` - صفحه اصلی

2. `GET /search?q=HTTP` - نتایج جستجو

3. `200 OK` - نمایش نتایج

مثال ۳: خرید آنلاین

1. `GET /products` - مشاهده محصولات

2. `POST /cart` - اضافه به سبد

3. `POST /checkout` - پرداخت

4. `201 Created` - سفارش ثبت شد

هدرهای مهم HTTP

هدرهای درخواست:

- `User-Agent`: مرورگر شما چیست؟

- `Accept`: چه نوع محتوایی می‌خواهی؟

- `Authorization`: هویت شما چیست؟

هدرهای پاسخ:

- `Content-Type`: نوع محتوا (HTML, JSON, etc.)

- `Set-Cookie`: ذخیره اطلاعات شما

- `Cache-Control`: چگونه کش شود؟

مشاهده HTTP در عمل

با DevTools مرورگر:

1. F12 را بزنید

2. به تب Network بروید

3. یک صفحه را باز کنید

4. درخواست‌های HTTP را ببینید

مثال واقعی:

|  |
| --- |
| Request URL: https://www.google.com/  Request Method: GET  Status Code: 200 OK  Content-Type: text/html |

ویژگی‌های پیشرفته HTTP

1. Caching - ذخیره موقت

- صفحاتی که تغییر نمی‌کنند را ذخیره می‌کند

- سرعت را افزایش می‌دهد

2. Compression - فشرده‌سازی

- داده‌ها را کوچک می‌کند

- مصرف اینترنت کاهش می‌یابد

3. Pipelining - ارسال چندتایی

- چند درخواست همزمان

- مانند چند مسافر در یک تاکسی

امنیت در HTTP

مشکلات امنیتی HTTP:

- اطلاعات قابل مشاهده توسط دیگران

- جعل هویت امکان‌پذیر است

- داده‌ها قابل تغییر هستند

راه حل: استفاده از HTTPS

- رمزگذاری end-to-end

- هویت سرور تأیید می‌شود

- داده‌ها محافظت می‌شوند

نکات کلیدی برای مبتدیان

### HTTP چیست:

- پروتکل ارتباط با وب‌سایت‌ها

- زبان مشترک مرورگر و سرور

- پایه و اساس وب

### HTTP چیست نیست:

- اینترنت نیست

- یک برنامه نیست

- فقط برای وب‌سایت نیست (APIها هم استفاده می‌کنند)

جمع‌بندی نهایی

HTTP مثل زبان دیپلماتیک بین مرورگرها و سرورهاست:

- دستور زبان: ساختار درخواست و پاسخ

- واژگان: متدها و کدهای وضعیت

- لهجه: هدرها و کوکی‌ها

سه نکته طلایی:

1. HTTP زبان ارتباطی وب است

2. هر عمل در وب یک درخواست HTTP است

3. HTTPS نسخه امن HTTP است

HTTP ستون فقرات ارتباطات وب است و بدون آن، وب امروزی وجود نداشت!

شروع یادگیری عملی

برای دیدن HTTP در عمل:

1. مرورگر خود را باز کنید

2. F12 بزنید → Network tab

3. به یک وب‌سایت سر بزنید

4. مکالمات HTTP را مشاهده کنید

# بررسی مفهوم Server

تعریف ساده سرور

سرور یک کامپیوتر همیشه روشن و متصل به اینترنت است که خدمات مختلفی به دیگر کامپیوترها (کلاینت‌ها) ارائه می‌دهد.

قیاس ساده: رستوران

تصور کنید سرور مثل یک رستوران است:

|  |  |
| --- | --- |
| بخش‌های رستوران | معادل در سرور |
| آشپزخانه | پردازش اطلاعات |
| منو | خدمات قابل ارائه |
| گارسون | نرم‌افزار سرور |
| مشتریان | کاربران (کلاینت‌ها) |

انواع سرورها

1. وب سرور (Web Server) - فروشگاه آنلاین

- وب‌سایت‌ها را میزبانی می‌کند

- مثال: Apache, Nginx

- مثل: یک مغازه که 24 ساعته باز است

2. فایل سرور (File Server) - کتابخانه

- فایل‌ها را ذخیره و به اشتراک می‌گذارد

- مثال: Google Drive, Dropbox

- مثل: کتابخانه عمومی

3. ایمیل سرور (Mail Server) - اداره پست

- ایمیل‌ها را مدیریت می‌کند

- مثال: Gmail, Outlook

- مثل: اداره پست الکترونیکی

4. گیم سرور (Game Server) - زمین بازی

- بازی‌های آنلاین را میزبانی می‌کند

- مثال: سرورهای PUBG, Minecraft

- مثل: زمین ورزشی مجازی

سرور چگونه کار می‌کند؟ - یک داستان

داستان درخواست یک صفحه وب:

کاربر (کلاینت):

"سلام سرور، لطفاً صفحه اصلی Google را بفرست"

سرور:

"بفرما! این صفحه HTML است...

همینطور فایل‌های CSS و JavaScript"

کاربر:

"ممنون! حالا یک عکس هم نیاز دارم"

سرور:

"عکس هم برات می‌فرستم"

اجزای اصلی یک سرور

1. سخت‌افزار (Hardware) - جسم سرور

- پردازنده (CPU): مغز سرور

- حافظه (RAM): حافظه موقت

- هارد دیسک (Storage): انبار اطلاعات

- شبکه (Network Card): ارتباط با اینترنت

2. نرم‌افزار (Software) - روح سرور

- سیستم عامل: Windows Server, Linux

- نرم‌افزار سرویس: Apache, MySQL

- امنیت: فایروال، آنتی‌ویروس

3. شبکه (Network) - راه‌های ارتباطی

- پهنای باند: عرض جاده اطلاعات

- آیپی ثابت: آدرس همیشگی

- Uptime: زمان فعالیت

سرور در مقابل کلاینت

سرور (خدمت‌دهنده):

- ✅ همیشه روشن

- ✅ قدرت پردازش بالا

- ✅ اتصال دائمی به اینترنت

- ✅ خدمات ارائه می‌دهد

کلاینت (خدمت‌گیرنده):

- ✅ گاهی خاموش/روشن

- ✅ قدرت پردازش معمولی

- ✅ اتصال موقت به اینترنت

- ✅ خدمات دریافت می‌کند

انواع میزبانی سرور

1. سرور اختصاصی (Dedicated Server) - ویلا

- فقط برای شما

- قدرت کامل

- هزینه بالا

- کنترل کامل

2. سرور مجازی (VPS) - آپارتمان

- بخشی از یک سرور بزرگ

- قدرت اشتراکی

- هزینه متوسط

- کنترل نسبی

3. میزبانی اشتراکی (Shared Hosting) - خوابگاه

- یک سرور با چندین کاربر

- قدرت بسیار محدود

- هزینه کم

- کنترل کم

4. ابری (Cloud) - هتل

- منابع قابل تغییر

- پرداخت به میزان استفاده

- انعطاف‌پذیر

- مقیاس‌پذیر

نمونه‌های واقعی سرور

سرورهای معروف:

- Google Server: جستجو، یوتیوب، جیمیل

- Facebook Server: فیسبوک، اینستاگرام، واتس‌اپ

- Amazon AWS: سرورهای ابری آمازون

- Microsoft Azure: سرورهای مایکروسافت

امنیت سرور

خطرات رایج:

- حمله DDoS: ترافیک مصنوعی زیاد

- هک: نفوذ به سیستم

- بدافزار: آلوده شدن سرور

راه‌های محافظت:

- فایروال: دیوار محافظ

- آنتی‌ویروس: سیستم ایمنی

- بک‌آپ: نسخه پشتیبان

- آپدیت: به‌روزرسانی منظم

ویژگی‌های یک سرور خوب

1. قابلیت اطمینان (Reliability)

- همیشه در دسترس باشد

- مثل: برق شهر

2. مقیاس‌پذیری (Scalability)

- بتواند بزرگ شود

- مثل: ساختمان قابل توسعه

3. امنیت (Security)

- در برابر threats مقاوم باشد

- مثل: بانک امن

4. کارایی (Performance)

- سریع پاسخ دهد

- مثل: آمبولانس سریع

سرورها در زندگی روزمره

وقتی که از سرور استفاده می‌کنید:

- وب‌گردی: سرورهای وب

- ارسال ایمیل: سرورهای ایمیل

- تماشای فیلم: سرورهای استریمینگ

- بازی آنلاین: سرورهای بازی

- ذخیره عکس: سرورهای ابری

چگونه یک سرور راه‌اندازی کنیم؟

مراحل ساده:

1. خرید سرور: سخت‌افزار یا سرویس ابری

2. نصب سیستم عامل: Windows Server یا Linux

3. نصب نرم‌افزار: وب سرور، پایگاه داده

4. پیکربندی شبکه: آیپی، دامنه

5. راه‌اندازی سرویس: وب‌سایت، اپلیکیشن

مفاهیم پیشرفته

1. Load Balancing - تقسیم بار

- چندین سرور با هم کار می‌کنند

- ترافیک بین آنها تقسیم می‌شود

- مثل: چندین گارسون در رستوران شلوغ

2. Clustering - خوشه‌سازی

- چند سرور مثل یک سرور عمل می‌کنند

- اگر یکی خراب شد، بقیه کار می‌کنند

- مثل: تیم پزشکی در اتاق عمل

3. Virtualization - مجازی‌سازی

- یک سرور فیزیکی به چند سرور مجازی تقسیم می‌شود

- صرفه‌جویی در منابع

- مثل: تقسیم یک خانه به چند آپارتمان

تفاوت سرور فیزیکی و مجازی

سرور فیزیکی:

کامپیوتر واقعی

- قدرت کامل

- هزینه بالا

- نگهداری سخت

سرور مجازی (VPS):

- ️ بخشی از یک سرور بزرگ

- ️ قدرت اشتراکی

- هزینه مناسب

- نگهداری آسان

هزینه‌های سرور

هزینه‌های معمول:

- خرید/اجاره سخت‌افزار

- برق و cooling

- پهنای باند اینترنت

- نگهداری و پشتیبانی

- امنیت و بک‌آپ

جمع‌بندی نهایی

سرور مثل یک خدمتکار وفادار است که:

- همیشه بیدار است

- قدرت زیادی دارد

- به همه خدمات می‌دهد

- در سایه کار می‌کند

سه نکته طلایی:

1. سرورها پایه و اساس اینترنت هستند

2. هر سرویس آنلاین روی سرورها اجرا می‌شود

3. سرورها همیشه در حال کار هستند

بدون سرورها، اینترنتی وجود نداشت و همه سرویس‌های آنلاین از کار می‌افتادند!

شروع یادگیری عملی

برای کار با سرورها می‌توانید:

1. یک VPS ارزان خریداری کنید

2. با Linux آشنا شوید

3. یک وب سرور نصب کنید

4. یک وب‌سایت ساده راه‌اندازی کنید

یادگیری سرورها = فهمیدن پایه‌های اینترنت

# بررسی مفهوم IP

IP چیست؟ (یک تشبیه ساده)

فرض کنید شما می‌خواهید یک نامه پستی برای دوست خود بفرستید. برای اینکه نامه به دست او برسد، باید آدرس دقیق خانه او را روی پاکت بنویسید. در دنیای شبکه‌های کامپیوتری و اینترنت، آدرس IP دقیقاً همان نقش "آدرس" را بازی می‌کند.

IP مخفف Internet Protocol (پروتکل اینترنت) است. هر دستگاه متصل به یک شبکه (مثل اینترنت یا حتی شبکه داخلی خانه شما) یک آدرس IP منحصربه‌فرد دریافت می‌کند تا بتواند با دیگر دستگاه‌ها ارتباط برقرار کند.

دستگاه شما (کامپیوتر، موبایل، تلویزیون هوشمند و...) = خانه شما

آدرس IP = شماره پلاک و نام خیابان خانه شما

آدرس IP چگونه است؟ (دو نسخه اصلی)

دو استاندارد اصلی برای آدرس‌های IP وجود دارد:

1. IPv4 (نسخه ۴)

این نسخه کلاسیک و رایج‌ترین فرمت است. آدرس‌های IPv4 از چهار عدد که با نقطه از هم جدا می‌شوند تشکیل شده‌اند. هر عدد می‌تواند بین ۰ تا ۲۵۵ باشد.

مثال: `192.168.1.1` یا `8.8.8.8` (آدرس سرورهای DNS گوگل)

با گسترش اینترنت، تعداد آدرس‌های منحصربه‌فرد IPv4 (حدود ۴.۳ میلیارد) در حال اتمام است.

2. IPv6 (نسخه ۶)

برای حل مشکل کمبود آدرس، نسخه جدیدی به نام IPv6 معرفی شد. این آدرس‌ها بسیار طولانی‌تر و از اعداد هگزادسیمال (۰-۹ و A-F) و دونقطه تشکیل شده‌اند.

\* مثال: `2001:0db8:85a3:0000:0000:8a2e:0370:7334`

این نسخه تعداد بسیار بیشتری آدرس (به میزان غیرقابل تصوری) در اختیار می‌گذارد.

انواع آدرس IP از نظر دسترسی

آدرس‌های IP به دو دسته اصلی تقسیم می‌شوند:

1. IP عمومی (Public IP)

این آدرس، شناسه اصلی شبکه شما در اینترنت است. وقتی به اینترنت متصل می‌شوید، شرکت ارائه‌دهنده اینترنت (ISP) مانند ایرانسل، شاتل یا پارس‌آنلاین یک IP عمومی به روتر (مودم) خانه شما اختصاص می‌دهد. تمام دستگاه‌های داخل خانه شما با یک IP عمومی به دنیای خارج دیده می‌شوند.

مثال: وقتی در گوگل "what is my ip" را جستجو می‌کنید، IP عمومی خود را می‌بینید.

2. IP خصوصی (Private IP)

در داخل شبکه محلی شما (مثلاً شبکه Wi-Fi خانه)، روتر به هر دستگاه یک IP خصوصی اختصاص می‌دهد. این آدرس‌ها فقط در داخل شبکه معتبر هستند و از بیرون قابل دسترسی نیستند. این کار شبیه دادن شماره اتاق‌های مختلف به ساکنان یک ساختمان بزرگ است.

مثال: آدرس‌هایی مانند `192.168.1.10` برای لپ‌تاپ شما و `192.168.1.11` برای موبایل شما.

نحوه ارتباط: وقتی لپ‌تاپ شما (`192.168.1.10`) یک درخواست به سایت گوگل می‌فرستد، روتر این درخواست را دریافت کرده، با استفاده از IP عمومی خود آن را به اینترنت ارسال می‌کند و وقتی پاسخ آمد، آن را به لپ‌تاپ شما برمی‌گرداند. این فرآیند ترجمه آدرس شبکه (NAT) نام دارد.

IP استاتیک (ثابت) در مقابل IP داینامیک (پویا)

IP داینامیک: بیشتر کاربران عادی اینترنت این نوع IP را دارند. IP عمومی شما پس از مدتی (مثلاً هر ۲۴ ساعت یا با هر بار restart کردن مودم) توسط ISP تغییر می‌کند. این روش برای ISPها مقرون‌به‌صرفه‌تر است.

IP استاتیک: این آدرس ثابت است و تغییر نمی‌کند. معمولاً سرورها، وب‌سایت‌ها یا دستگاه‌هایی که نیاز است همیشه از بیرون در دسترس باشند (مثل دوربین‌های مداربسته تحت شبکه) به IP استاتیک نیاز دارند و معمولاً هزینه اضافه دارند.

پروتکل اینترنت (Internet Protocol) چه می‌کند؟

همانطور که از نامش پیداست، IP یک پروتکل یا قوانین ارتباطی است. وظیفه اصلی آن مسیریابی (Routing) بسته‌های داده به مقصد صحیح است. IP مسئولیت تحویل بسته‌ها را بر عهده دارد، اما تضمینی برای رسیدن بدون خطای آن‌ها نمی‌دهد. تضمین تحویل مطمئن بر عهده پروتکل‌های دیگر مانند TCP است (که معمولاً together با آن به صورت TCP/IP شناخته می‌شوند).

چگونه IP خود را پیدا کنیم؟

IP عمومی: کافیست در مرورگر خود به سایتی مانند https://whatismyipaddress.com/ بروید.

IP خصوصی:

ویندوز: در Command Prompt دستور `ipconfig` را تایپ کنید.

مک: در Terminal دستور `ifconfig` را تایپ کنید.

اندروید و iOS: به تنظیمات WiFi رفته و روی شبکه متصل شده کلیک کنید تا جزئیات را ببینید.

جمع‌بندی نکات کلیدی

|  |  |
| --- | --- |
| ویژگی | توضیح |
| کاربرد اصلی | شناسایی و مسیریابی دستگاه‌ها در یک شبکه |
| فرمت‌های رایج | IPv4 (مثل ۱۹۲.۱۶۸.۱.۱) و IPv6 (مثل ۲۰۰۱:db8::۸a۲e) |
| انواع دسترسی | عمومی: برای اینترنت | خصوصی: برای شبکه داخلی |
| انواع تخصیص | داینامیک: تغییرکننده | استاتیک: ثابت |

# بررسی مفهوم Domain

دامنه چیست؟ (تشبیه ساده)

فرض کنید ما در دنیای اینترنت زندگی می‌کنیم. هر وب‌سایت یک خانه است که روی یک سرور (یک کامپیوتر قدرتمند همیشه روشن) قرار دارد. هر خانه یک آدرس IP منحصربه‌فرد دارد (مثل `۱۹۲.۰.۲.۴۴`).

مشکل اینجاست که به خاطر سپردن این اعداد برای انسان‌ها بسیار سخت است. آیا می‌توانید آدرس IP تمام وب‌سایت‌هایی که روزانه بازمی‌کنید را به خاطر بسپارید؟

اینجاست که نام دامنه وارد می‌شود. دامنه مانند یک پلاک خوانا و به یاد ماندنی برای آن آدرس IP است.

آدرس IP = `۱۹۲.۰.۲.۴۴` (مثل مختصات جغرافیایی یک خانه)

نام دامنه = `google.com` (مثل یک آدرس پستی ساده و قابل خواندن)

پس دامنه در واقع نامی است که شما در نوار آدرس مرورگر خود تایپ می‌کنید تا به یک وب‌سایت دسترسی پیدا کنید.

دامنه چگونه کار می‌کند؟ (سیستم DNS)

وقتی شما `www.google.com` را در مرورگر خود وارد می‌کنید، چه اتفاقی می‌افتد؟

1. مرورگر شما به سرورهای خاصی به نام DNS (سیستم نام دامنه) مراجعه می‌کند. DNS مانند دفترچه تلفن بسیار بزرگ اینترنت است.

2. مرورگر می‌پرسد: "آدرس IP مربوط به `google.com` چیست؟"

3. سرور DNS پاسخ می‌دهد: "آدرس IP آن `۱۷۲.۲۱۷.۱۶.۲۳۸` است."

4. حالا مرورگر با استفاده از آن آدرس IP، می‌تواند به سرور گوگل متصل شده و صفحه وب را برای شما بارگذاری کند.

این کل فرآیند در کسری از ثانیه اتفاق می‌افتد.

اجزای تشکیل‌دهنده یک نام دامنه

یک نام دامنه کامل از چند بخش تشکیل شده است. بیایید `https://www.example.ir` را بررسی کنیم:

1. زیردامنه (Subdomain):

بخش سمت چپ دامنه اصلی.

رایج‌ترین آن `www` است، اما می‌تواند چیزهای دیگری مثل `mail.google.com` یا `blog.example.com` باشد.

اختیاری است (امروزه بسیاری از سایت‌ها بدون `www` نیز کار می‌کنند).

2. نام دامنه (Second-Level Domain - SLD):

بخش اصلی و منحصربه‌فرد نامی که شما انتخاب می‌کنید.

در مثال ما `example` نام دامنه است.

این بخش هویت و برند وب‌سایت شما را می‌سازد.

3. پسوند دامنه (Top-Level Domain - TLD):

بخش سمت راست نقطه.

در مثال ما `.ir` پسوند است.

این پسوندها دسته‌بندی خاصی را نشان می‌دهند.

انواع پسوندهای دامنه (TLD)

پسوندها به دسته‌های مختلفی تقسیم می‌شوند:

دامنه‌های ملی (ccTLD - Country Code TLD):

مخصوص کشورها هستند.

مثال: `.ir` (ایران)، `.de` (آلمان)، `.us` (آمریکا)، `.tr` (ترکیه).

دامنه‌های عمومی (gTLD - Generic TLD):

معروف‌ترین و پرکاربردترین پسوندها.

مثال: `.com` (Commercial - تجاری)، `.org` (Organization - سازمان‌ها)، `.net` (Network - شبکه‌ها)، `.info` (Information - اطلاعات).

دامنه‌های جدید (New gTLD):

پسوندهای جدیدی که برای تنوع بیشتر ایجاد شده‌اند.

مثال: `.shop`، `.app`، `.blog`، `.tech`، `.bank`.

چرا به دامنه نیاز داریم؟ (اهداف و مزایا)

1. قابل حفظ کردن: به خاطر سپردن `google.com` بسیار ساده‌تر از `۱۷۲.۲۱۷.۱۶.۲۳۸` است.

2. حرفه‌ای بودن: داشتن یک دامنه اختصاصی (مثل `brandname.com`) بسیار معتبرتر از استفاده از یک آدرس رایگان (مثل `brandname.freehost.com`) است.

3. برندسازی: دامنه بخشی از هویت و برند کسب‌وکار شما در اینترنت می‌شود.

4. ثبات: حتی اگر سرور یا آدرس IP وب‌سایت شما تغییر کند، می‌توانید دامنه خود را به آدرس جدید اشاره کنید و کاربران همچنان با همان نام قدیمی به سایت شما دسترسی خواهند داشت.

چگونه یک دامنه ثبت کنیم؟

دامنه‌ها را نمی‌توان خرید، بلکه آن‌ها را برای یک دوره مشخص (معمولاً ۱ تا ۱۰ سال) ثبت (Register) می‌کنید. برای این کار باید به شرکت‌های ثبت‌کننده دامنه (Domain Registrar) مانند شرکت‌های داخلی (مثلاً `پارس‌پک`، `ایران‌سرور`) یا بین‌المللی (مثلاً `GoDaddy`، `Namecheap`) مراجعه کنید.

تفاوت دامنه و هاست (مهم!)

این دو مفهوم اغلب با هم اشتباه گرفته می‌شوند، اما کاملاً متفاوت هستند:

|  |  |  |
| --- | --- | --- |
| ویژگی | دامنه (Domain) | هاست (Hosting) |
| تعریف | آدرس وب‌سایت شما (مثل آدرس پستی) | فضای ذخیره‌سازی وب‌سایت شما (مثل زمین و ساختمان) |
| کاربرد | کاربران با تایپ آن به سایت شما می‌رسند | فایل‌ها، عکس‌ها و کدهای سایت شما روی آن قرار می‌گیرد |
| مثال | example.com | یک سرور فیزیکی در یک مرکز داده |

برای راه‌اندازی یک وب‌سایت، شما به هر دو نیاز دارید: هم یک دامنه و هم یک سرویس هاستینگ.

جمع‌بندی نکات کلیدی

|  |  |
| --- | --- |
| مفهوم | توضیح خلاصه |
| کاربرد اصلی | تبدیل آدرس‌های عددی سخت (IP) به نام‌های خوانا و به یاد ماندنی |
| اجزای اصلی | زیردامنه (مثل www) + نام (مثل example) + پسوند (مثل .com) |
| سیستم پشتیبان | DNS که مانند دفترچه تلفن، دامنه را به IP تبدیل می‌کند |
| ثبت دامنه | اجاره نام برای یک دوره مشخص (مثلاً ۱ سال) از طریق Registrarها |

# بررسی مفهوم DNS

DNS چیست؟ (تشبیه ساده)

فرض کنید شما می‌خواهید با یک دوست تماس بگیرید. شما نام او را در حافظه خود دارید (مثلاً "احمد")، اما تلفن برای برقراری ارتباط نیاز به شماره تلفن او دارد. در این حالت، شما از دفترچه تلفن استفاده می‌کنید تا نام "احمد" را به شماره تلفن او تبدیل کنید.

DNS (Domain Name System) دقیقاً همان نقش دفترچه تلفن جهانی اینترنت را بازی می‌کند.

نام دامنه (مثل `google.com`) = نام شخص (مثل "احمد")

آدرس IP (مثل `142.251.16.100`) = شماره تلفن

DNS = دفترچه تلفن

کار اصلی DNS ترجمه نام‌های قابل خواندن توسط انسان (دامنه) به آدرس‌های عددی ماشین‌ها (IP) است.

مشکل بزرگی که DNS حل می‌کند

بدون DNS، شما برای بازدید از هر وب‌سایتی مجبور بودید آدرس IP عددی و خشک آن را به خاطر بسپارید.

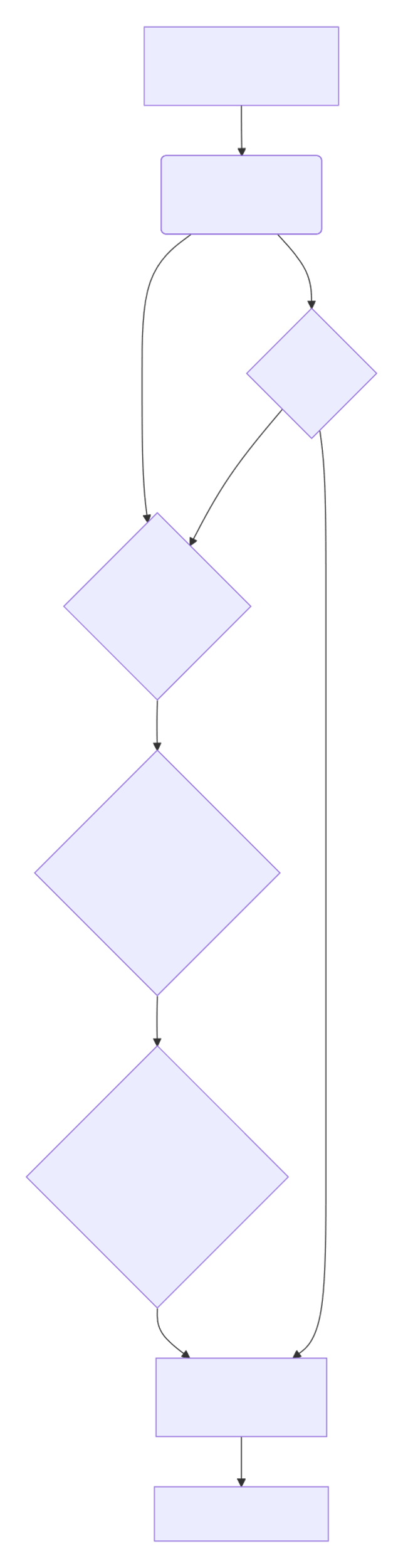
با DNS: شما `digikala.com` را تایپ می‌کنید.

بدون DNS: شما مجبور بودید چیزی شبیه `۵.۱۴۴.۱۳۱.۲۳` را به خاطر بسپارید و وارد کنید.

DNS زندگی را در اینترنت برای انسان‌ها ممکن ساخته است.

DNS چگونه کار می‌کند؟ (یک جستجوی معمولی)

وقتی شما `www.youtube.com` را در مرورگر خود وارد کرده و Enter می‌زنید، چه اتفاقی پشت صحنه می‌افتد؟ این فرآیند که جستجوی DNS (DNS Lookup) نام دارد، در چندین مرحله و در کسری از ثانیه انجام می‌شود. نمودار زیر این جریان را به وضوح نشان می‌دهد:



Website Loaded

IP Address : 142.251.16.100

Authoritative Server

IP for www.youtube.com?

yes

TLD Server

Where is youtube.com?

Root Server

Where is .com?

no

Cache Hit?

Cache Check

Resolver  
ISP/Recursive DNS

User enters  
www.youtube.com

در نمودار بالا دیدیم که چگونه چهار سرویس اصلی DNS برای تبدیل نام دامنه به آدرس IP با یکدیگر همکاری می‌کنند. حالا اجازه دهید نقش هرکدام را دقیق‌تر بررسی کنیم:

سرورهای درگیر در فرآیند DNS

1. DNS Recursor (Resolve کننده بازگشتی):

نقش: همان "تحقیق‌کننده" است. این سرور معمولاً توسط ISP (شرکت اینترنتی شما) ارائه می‌شود. درخواست شما را دریافت می‌کند و برای پیدا کردن پاسخ، به دیگر سرورها "سرک" می‌کشد.

مثال: سرور DNS شرکت‌هایی مانند ایرانسل، شاتل یا پارس‌آنلاین.

2. Root Name Server (سرور ریشه):

نقش: مانند فهرست بخش‌های مختلف دفترچه تلفن عمل می‌کند. این سرور آدرس سرورهای مربوط به پسوندها (TLD) مانند `.com`، `.org`، `.ir` را می‌داند.

کاربرد: به Resolver می‌گوید: "برای پیدا کردن `youtube.com`، باید به سرورهای مخصوص پسوند `.com` مراجعه کنی."

3. TLD Name Server (سرور پسوند دامنه):

نقش: اطلاعات مربوط به یک پسوند خاص را نگهداری می‌کند.

کاربرد: سرور مخصوص `.com` به Resolver می‌گوید: "برای اطلاعات دامنه `youtube.com`، باید به سرورهای معتبر (Authoritative) که مسئول این دامنه هستند، مراجعه کنی. آدرس آن سرورها این است..."

4. Authoritative Name Server (سرور معتبر):

نقش: این سرور منبع اصلی حقیقت برای یک دامنه خاص است. اطلاعات دقیق دامنه (مانند آدرس IP اصلی) در اینجا نگهداری می‌شود.

مالکیت: این سرورها توسط صاحب دامنه یا شرکت هاستینگ آن مدیریت می‌شوند.

نتیجه نهایی: این سرور به Resolver پاسخ نهایی را می‌دهد: "آدرس IP `www.youtube.com` برابر است با `۱۴۲.۲۵۱.۱۶.۱۰۰`."

کش DNS (DNS Cache): میان‌بری برای سرعت

اگر نمودار را دوباره نگاه کنید، یک مسیر جایگزین از `Cache Hit?` می‌بینید. اینجاست که کش وارد عمل می‌شود.

کش چیست؟ یک حافظه موقت سریع.

کاربرد: پس از اولین بار پیدا کردن آدرس IP یک دامنه، Resolver نتیجه را برای مدت مشخصی (TTL) در کش خود ذخیره می‌کند.

فایده: اگر شما یا کاربر دیگری در مدت زمان معینی دوباره به همان دامنه مراجعه کند، Resolver به جای طی کردن کل مسیر طولانی، بلافاصله پاسخ را از کش خود برمی‌گرداند. این کار سرعت را به شدت افزایش می‌دهد و از ترافیک شبکه می‌کاهد.

کش در سیستم عامل (ویندوز، مک) و حتی مرورگر شما نیز وجود دارد.

انواع رکوردهای DNS (Records)

سرورهای معتبر، اطلاعات یک دامنه را در قالب رکوردهای مختلفی ذخیره می‌کنند. هر رکورد یک وظیفه خاص دارد:

A Record: اصلی‌ترین رکورد که دامنه را به آدرس IPv4 تبدیل می‌کند.

AAAA Record: دامنه را به آدرس IPv6 تبدیل می‌کند.

CNAME Record: یک دامنه را به نام دامنه دیگری "هدایت می‌کند" (مثلاً `digikala.com` را به `www.digikala.com`).

MX Record: مشخص می‌کند ایمیل‌های مربوط به دامنه شما به کدام سرور ایمیل ارسال شوند.

TXT Record: برای ذخیره اطلاعات متنی مانند تأیید مالکیت دامنه یا تنظیمات امنیتی (SPF, DKIM) استفاده می‌شود.

اهمیت DNS: چرا DNS حیاتی است؟

1. کاربرپسند کردن اینترنت: پایه و اساس استفاده آسان از اینترنت است.

2. قابلیت اطمینان و افزونگی: سیستم DNS به گونه‌ای طراحی شده که بسیار مقاوم است و اگر یک سرور از کار بیفتد، سرورهای دیگر پاسخگو هستند.

3. انعطاف‌پذیری: به صاحبان وب‌سایت اجازه می‌دهد بدون تغییر دامنه، سرور یا هاستینگ خود را عوض کنند (فقط رکوردهای DNS را به روز می‌کنند).

4. توزیع بار (Load Balancing): DNS می‌تواند ترافیک را بین چندین سرور توزیع کند تا از overload شدن یک سرور جلوگیری شود.

جمع‌بندی نکات کلیدی

|  |  |
| --- | --- |
| مفهوم | توضیح خلاصه |
| کاربرد اصلی | تبدیل نام دامنه (مثل google.com) به آدرس IP (مثل ۱۴۲.۲۵۱.۱۶.۱۰۰) |
| تشبیه | دفترچه تلفن اینترنت |
| فرآیند اصلی | جستجوی DNS: درخواست کاربر -> Resolver -> Root -> TLD -> Authoritative -> IP |
| عنصر سرعت‌دهنده | کش (Cache): ذخیره موقت پاسخ‌ها برای دسترسی سریع‌تر در آینده |
| اجزای کلیدی | Recursive Resolver, Root Server, TLD Server, Authoritative Server |

# بررسی مفهوم Interanet

اینترانت چیست؟ (تشبیه ساده)

فرض کنید یک شرکت یا سازمان، یک شهر کوچک خصوصی است که دیوارهایی دور آن کشیده شده است. فقط ساکنان این شهر (کارکنان آن سازمان) اجازه ورود دارند و می‌توانند از امکانات داخل شهر مانند کتابخانه، تابلوهای اعلانات، ادارات و سیستم پستی داخلی استفاده کنند.

اینترانت دقیقاً همان شهر خصوصی است.

\* اینترانت = یک شبکه داخلی خصوصی که از فناوری‌های مشابه اینترنت (مانند وب‌سرورها و مرورگرها) استفاده می‌کند، اما فقط برای اعضای یک سازمان قابل دسترسی است.

تعریف فنی

اینترانت (Intranet) یک شبکه رایانه‌ای خصوصی است که از پروتکل‌های اینترنتی (مثل TCP/IP و HTTP) برای اشتراک‌گذاری اطلاعات، ابزارهای collaboration و سیستم‌های عملیاتی در داخل یک سازمان استفاده می‌کند. دسترسی به آن معمولاً به کارمندان سازمان محدود می‌شود و از طریق فایروال از دسترسی عمومی محافظت می‌گردد.

هدف اصلی اینترانت چیست؟

هدف اصلی، ایجاد یک پلتفرم مرکزی و امن برای موارد زیر است:

اشتراک‌گذاری اطلاعات: انتشار اخبار شرکت، دستورالعلها، راهنماهای کاری، فرم‌ها.

همکاری (Collaboration): فضای کاری مشترک، انجمن‌های گفتگو، ویکی‌های داخلی.

اتوماسیون فرآیندها: سیستم درخواست مرخصی، ثبت سفارش خرید، مدیریت پروژه.

دسترسی به برنامه‌های کاربردی: برنامه‌های نرم‌افزاری داخلی شرکت که فقط از طریق شبکه داخلی در دسترس هستند.

ویژگی‌های کلیدی اینترانت

1. دسترسی محدود و امن: مهمترین ویژگی. فقط افراد مجاز (کارکنان) می‌توانند به آن وارد شوند. این دسترسی اغلب با نام کاربری و رمز عبور کنترل می‌شود.

2. محرمانگی: اطلاعات حساس شرکت (مثل استراتژی‌ها، مالی، منابع انسانی) در محیطی امن قرار می‌گیرد.

3. متمرکز بودن: همه چیز در یک جای واحد قرار دارد و employees برای پیدا کردن اطلاعات لازم به چندین منبع مراجعه نمی‌کنند.

4. استفاده از مرورگر وب: کاربران با مرورگر آشنا (مثل Chrome, Firefox) به آن دسترسی دارند، بنابراین نیاز به آموزش کمی وجود دارد.

اینترانت چگونه کار می‌کند؟

اینترانت از همان زیرساخت یک شبکه محلی (LAN) استفاده می‌کند:

1. یک یا چند وب‌سرور در داخل شبکه سازمان نصب می‌شوند که صفحات اینترانت را میزبانی می‌کنند.

2. یک فایروال قوی بین شبکه داخلی و اینترنت عمومی قرار می‌گیرد تا از ورود غیرمجاز جلوگیری کند.

3. کارمندان از طریق کامپیوترهای خود در شبکه سازمان، با تایپ یک آدرس (مثل `http://intranet`) به آن متصل می‌شوند.

4. اگر کارمندی از راه دور بخواهد به اینترانت دسترسی داشته باشد، معمولاً باید از یک VPN (شبکه خصوصی مجازی) استفاده کند تا connection او امن شده و به عنوان بخشی از شبکه داخلی شناسایی شود.

مثال‌های عینی از اینترانت

سایت داخلی یک شرکت بزرگ: صفحه‌ای که با ورود به شرکت، همه کارمندان می‌بینند و شامل اخبار شرکت، لیست تلفن داخلی، لینک به نرم‌افزارهای داخلی و فرم درخواست مرخصی است.

سیستم مدیریت آموزش (LMS) داخلی: پورتالی که کارمندان برای گذراندن دوره‌های آموزشی اجباری از آن استفاده می‌کنند.

ویکی (Wiki) داخلی شرکت: دانشنامه‌ای که در آن روش‌های انجام کارها، troubleshooting و بهترین تجربیات به اشتراک گذاشته می‌شود.

مقایسه اینترانت با مفاهیم مشابه (مهم!)

برای درک بهتر، اینترانت را با اینترنت و اکسترانت مقایسه می‌کنیم:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | اینترنت (Internet) | اینترانت (Intranet) | اکسترانت (Extranet) |
| محدوده دسترسی | عمومی و جهانی - همه می‌توانند به بخش‌های عمومی دسترسی داشته باشند. | داخلی و خصوصی - فقط اعضای یک سازمان. | | نیمه خصوصی - اعضای سازمان + شرکای خارجی محدود (مانند تامین کنندگان، مشتریان خاص). |
| هدف | اشتراک‌گذاری اطلاعات در سطح جهانی. | اشتراک‌گذاری اطلاعات و collaboration در داخل سازمان. | اشتراک‌گذاری اطلاعات selected با شرکای خارج از سازمان. |
| میزان امنیت | امنیت نسبتاً پایین (فضای عمومی). | امنیت بسیار بالا (پشت فایروال). | امنیت بالا، با کنترل دسترسی دقیق برای کاربران خارجی. |
| مثال | `google.com`, `digikala.com` | آدرس داخلی شرکت برای اخبار و منابع انسانی | پورتالی که یک شرکت برای تامین کنندگان خود ایجاد کرده تا موجودی انبار را ببینند. |

مزایا و معایب اینترانت

مزایا:

افزایش بهره‌وری: دسترسی سریع و متمرکز به اطلاعات.

کاهش هزینه‌ها: کاهش مصرف کاغذ، بهبود ارتباطات.

بهبود فرهنگ سازمانی: ایجاد حس تعلق و یکپارچگی.

امنیت اطلاعات: حفاظت از داده‌های حساس شرکت.

معایب:

هزینه راه‌اندازی و نگهداری: نیاز به سرور، نرم‌افزار و پشتیبانی فنی دارد.

مقاومت در برابر تغییر: ممکن است کارمندان به استفاده از آن عادت نکنند.

نیاز به بروزرسانی: اطلاعات باید همیشه به روز باشند تا مفید واقع شوند.

جمع‌بندی نکات کلیدی

|  |  |
| --- | --- |
| مفهوم | توضیح خلاصه |
| تعریف | شبکه خصوصی یک سازمان که از تکنولوژی وب استفاده می‌کند |
| تشبیه | شهر خصوصی یا شبکه داخلی یک شرکت |
| کاربران | فقط کارکنان و اعضای سازمان |
| هدف اصلی | اشتراک‌گذاری اطلاعات و بهبود همکاری داخلی به صورت امن و متمرکز |
| تفاوت با اینترنت | اینترنت عمومی است، اینترانت خصوصی است. |

# بررسی مفهوم Extranet

فرض کنید یک شرکت، یک محله امن و خصوصی دارد (اینترانت آن). حالا این شرکت می‌خواهد برای شرکای تجاری خاصی (مثل تأمین‌کنندگان یا مشتریان ویژه) دسترسی کنترل‌شده‌ای به بخشی از این محله بدهد. اما نمی‌خواهد این افراد به تمام ساختمان‌ها دسترسی داشته باشند.

راه‌حل چیست؟ یک دریچه امن یا اتاق مهمان مخصوص در حصار این محله ایجاد می‌کند که دسترسی به چند بخش مشخص را فراهم می‌کند.

اکسترانت دقیقاً همان اتاق مهمان یا دریچه امن است.

اکسترانت = یک شبکه خصوصی گسترده که دسترسی محدود و امنی را به بخشی از اطلاعات یا سرویس‌های یک سازمان برای کاربران خارجی (مانند شرکا، تأمین‌کنندگان، مشتریان منتخب) فراهم می‌کند.

تعریف فنی

اکسترانت (Extranet) بخشی از شبکه خصوصی یک سازمان (اینترانت) است که به صورت کنترل‌شده و از طریق اینترنت، در اختیار کاربران مجاز خارج از سازمان قرار می‌گیرد. دسترسی معمولاً با نام کاربری، رمز عبور و سایر روش‌های احراز هویت شدیداً کنترل می‌شود.

هدف اصلی اکسترانت چیست؟

هدف اصلی، ایجاد همکاری امن و کارآمد با entities خارجی بدون به خطر انداختن امنیت شبکه داخلی است.

کاربردهای رایج اکسترانت

1. برای تأمین‌کنندگان (Suppliers):

مشاهده سطح موجودی انبار شرکت

دیدن برنامه‌ریزی تولید برای هماهنگی بهتر تحویل کالا

ثبت قیمت‌ها و شرکت در مناقصات داخلی

2. برای مشتریان بزرگ (Key Customers):

پیگیری سفارشات (مثلاً دیدن وضعیت تولید و حمل سفارش خود)

دسترسی به پایگاه دانش فنی و پشتیبانی ویژه

ثبت مستقیم سفارشات جدید

3. برای شعب یا نمایندگی‌های دیگر:

اشتراک‌گذاری اطلاعات فروش، بازاریابی و منابع آموزشی بین شعب مختلف یک شرکت بین‌المللی

4. برای پیمانکاران و شرکای پروژه:

دسترسی به اسناد، نقشه‌ها و زمان‌بندی‌های یک پروژه خاص برای هماهنگی بهتر

ویژگی‌های کلیدی اکسترانت

1. دسترسی بسیار کنترل‌شده: کاربران خارجی فقط به بخش‌های خاصی که برایشان تعریف شده دسترسی دارند ("نیاز به دانستن").

2. امنیت فوق‌العاده بالا: از فایروال‌های پیشرفته، رمزگذاری (Encryption) و روش‌های قوی احراز هویت (مانند کلیدهای امنیتی) استفاده می‌کند.

3. مبتنی بر اینترنت: دسترسی از طریق اینترنت عمومی انجام می‌شود، اما connection کاملاً امن است.

4. هدف تجاری و همکاری: ماهیت آن همکاری با entities خارجی برای بهبود زنجیره تأمین و روابط تجاری است.

اکسترانت چگونه کار می‌کند؟

1. یک سازمان بخشی از سرویس‌های اینترانت خود را (مثلاً یک برنامه یا پایگاه داده) برای دسترسی از بیرون آماده می‌کند.

2. یک درگاه امن (Secure Gateway) روی فایروال ایجاد می‌کند.

3. برای هر کاربر خارجی یک حساب کاربری (Account) با سطوح دسترسی مشخص ایجاد می‌شود.

4. کاربران خارجی از طریق اینترنت، به آدرس خاص اکسترانت متصل می‌شوند.

5. آن‌ها باید هویت خود را احراز کنند (با نام کاربری و رمز عبور قوی یا روش‌های دو مرحله‌ای).

6. پس از ورود، فقط به منابعی که مجاز هستند دسترسی پیدا می‌کنند.

یک مثال عینی

شرکت خودروسازی "الف" را در نظر بگیرید:

اینترانت شرکت الف: حاوی اطلاعات محرمانه مانند استراتژی‌های آینده، مالیات، حقوق کارکنان و طرح‌های جدید است. فقط کارکنان به آن دسترسی دارند.

اکسترانت شرکت الف: این شرکت یک اکسترانت برای تأمین‌کنندگان قطعات خود ایجاد کرده است.

تأمین‌کننده با ورود به اکسترانت می‌تواند:

موجودی انبار قطعات شرکت الف را ببیند تا بداند چه زمانی باید قطعات جدید ارسال کند.

برنامه تولید هفته آینده شرکت الف را مشاهده کند.

صورتحساب‌های خود را بارگذاری و وضعیت پرداخت آن‌ها را پیگیری کند.

اما تأمین‌کننده هرگز نمی‌تواند به اطلاعات حقوق کارکنان یا طرح‌های محرمانه جدید دسترسی پیدا کند.

مزایا و معایب اکسترانت

مزایا:

بهبود کارایی زنجیره تأمین: هماهنگی سریع‌تر و دقیق‌تر با تأمین‌کنندگان و مشتریان.

کاهش هزینه‌ها: کاهش خطاهای ارتباطی، کاغذبازی و زمان پردازش سفارشات.

تقویت روابط با شرکا: شفافیت و اعتماد را افزایش می‌دهد.

امنیت بالا: اطلاعات حساس شرکت در عین به اشتراک‌گذاری محدود، محافظت می‌شوند.

معایب:

هزینه پیاده‌سازی و نگهداری بالا: نیاز به زیرساخت امنیتی قوی دارد.

پیچیدگی مدیریت: مدیریت کاربران خارجی و سطوح دسترسی آن‌ها چالش‌برانگیز است.

ریسک امنیتی بالقوه: با وجود امنیت بالا، باز کردن یک دریچه به بیرون همیشه یک ریسک محسوب می‌شود و نیاز به نظارت دائمی دارد.

# بررسی موتورهای رندرگیری مرورگرها

موتور رندرگیری مرورگر، هسته اصلی هر مرورگر وب است که مسئول تفسیر کدهای HTML و CSS و نمایش بصری صفحات وب برای شماست. در جدول زیر میتوانید مهمترین موتورهای رندر و مرورگرهای مرتبط با آنها را ببینید.

|  |  |  |
| --- | --- | --- |
| موتور رندر | مرورگرهای شاخص | توضیحات |
| Blink | گوگل کروم، مایکروسافت اج (نسخه جدید)، اپرا، ادج | موتور اصلی و پیشرو در دنیای مرورگرها که توسط گوگل توسعه یافته است. |
| WebKit | سافاری (اپل) | موتور اصلی مورد استفاده در محصولات اپل. همه مرورگرهای iOS موظفند از این موتور استفاده کنند. |
| Gecko | موزیلا فایرفاکس | موتور متن باز توسعه یافته توسط موزیلا که به عنوان جایگزینی مستقل برای Blink شناخته می‌شود. |
| Trident | اینترنت اکسپلورر (IE) | موتور قدیمی مایکروسافت که امروزه منسوخ شده است. |
| EdgeHTML | مایکروسافت اج (نسخه قدیمی) | جانشین Trident که بعداً جای خود را به Blink داد. |

چرا شناخت موتور رندر مهم است؟

تفاوت در موتورهای رندر می‌تواند دلیل اصلی تفاوت در نمایش یک صفحه وب در مرورگرهای مختلف باشد. این موضوع به ویژه برای طراحان وب بسیار حیاتی است، زیرا آن‌ها باید اطمینان حاصل کنند که وب‌سایت طراحی شده، در تمامی موتورهای رندر اصلی به درستی و یکسان نمایش داده می‌شود.

# بررسی URL

URL چیست؟ (تشبیه ساده)

فرض کنید شما در یک شهر بزرگ زندگی می‌کنید. هر خانه در این شهر یک آدرس پستی منحصربه‌فرد دارد که شامل نام خیابان، شماره پلاک، نام شهر و کدپستی است. با این آدرس، پستچی می‌داند دقیقاً باید نامه را به کجا برساند.

URL دقیقاً همان "آدرس پستی" برای اطلاعات در دنیای وب است.

\* URL = نشانی یکتای منابع در وب که به مرورگر می‌گوید دقیقاً چه چیزی را و از کجا باید بگیرد.

URL مخفف Uniform Resource Locator به معنای "نشانی‌یاب یکنواخت منبع" است.

URL چگونه است؟ (اجزای تشکیل‌دهنده)

یک URL کامل از چند بخش تشکیل شده است. بیایید آدرس زیر را بررسی کنیم:

`https://www.example.com:8080/path/to/myfile.html?key1=value1&key2=value2section1`

1. پروتکل (Protocol):

مثال: `https://`

نقش: قوانین ارتباطی بین مرورگر و سرور را مشخص می‌کند. مشخص می‌کند که داده‌ها چگونه منتقل شوند.

رایج‌ترین پروتکل: `HTTP` (ناآمنه) و `HTTPS` (امن و رمزگذاری شده). پروتکل‌های دیگر عبارتند از `ftp://` (برای انتقال فایل) و `mailto://` (برای ایمیل).

2. نام میزبان (Hostname):

مثال: `www.example.com`

نقش: آدرس سروری که منبع (مثل وب‌سایت) روی آن قرار دارد را مشخص می‌کند. این بخش معمولاً یک نام دامنه است.

3. پورت (Port):

مثال: `:8080`

نقش: یک "درگاه" خاص روی سرور را برای ارتباط مشخص می‌کند. مانند شماره داخلی یک ساختمان بزرگ.

پورت‌های پیش‌فرض: اگر پورت مشخص نشود، مرورگر به طور خودکار از پورت استاندارد استفاده می‌کند (پورت ۸۰ برای HTTP و پورت ۴۴۳ برای HTTPS). بنابراین معمولاً در URLها دیده نمی‌شوند.

4. مسیر (Path):

مثال: `/path/to/myfile.html`

نقش: محل دقیق منبع درخواستی (مثل یک صفحه HTML، یک عکس یا یک فایل) را در ساختار پوشه‌های سرور مشخص می‌کند. مانند مسیر یک فایل در کامپیوتر شما.

5. پارامترهای Query (کوئری):

مثال: `?key1=value1&key2=value2`

نقش: اطلاعات اضافی را به سرور ارسال می‌کند. این بخش معمولاً پس از علامت سؤال (`?`) می‌آید و شامل جفت‌های `key=value` است که با `&` از هم جدا می‌شوند.

کاربرد: برای جستجو در سایت، فیلتر کردن نتایج یا ارسال داده‌های فرم استفاده می‌شود. مثلاً وقتی در یوتیوب جستجو می‌کنید، URL به شکل `youtube.com/results?search\_query=query` درمی‌آید.

6. لنگر (Fragment یا Anchor):

مثال: `section1`

نقش: به مرورگر دستور می‌دهد که پس از بارگذاری صفحه، به یک قسمت خاص از همان صفحه پرش کند (مثل یک تیتر یا بخش مشخص).

نکته: این بخش به سرور ارسال نمی‌شود و فقط توسط مرورگر پردازش می‌گردد.

یک مثال واقعی از URL

آدرس `https://digikala.com/search/?q=mobile` را در نظر بگیرید:

پروتکل: `https://` (ارتباط امن)

نام میزبان: `digikala.com`

مسیر: `/search/` (به صفحه جستجو برو)

پارامترهای Query: `?q=mobile` (عبارت جستجو شده "mobile" است)

تفاوت URL و Domain (مهم!)

این دو مفهوم مرتبط اما متفاوت هستند:

دامنه (Domain): فقط بخش نام و آدرس اصلی وب‌سایت است (مثلاً `google.com`).

URL (نشانی وب): یک آدرس کامل و دقیق است که شامل پروتکل، دامنه، مسیر و سایر پارامترها می‌شود تا یک منبع خاص را پیدا کند (مثلاً `https://google.com/maps`).

به زبان ساده: دامنه مانند نام شهر است، اما URL مانند آدرس کامل یک ساختمان خاص در آن شهر است.

انواع URL

1. URL مطلق (Absolute): شامل تمام بخش‌های لازم (پروتکل، میزبان و مسیر) برای پیدا کردن منبع است.

مثال: `https://example.com/images/photo.jpg`

2. URL نسبی (Relative): فقط مسیر را نسبت به آدرس فعلی مشخص می‌کند. این نوع URL در طراحی صفحات وب بسیار کاربرد دارد.

اگر در صفحه `https://example.com/about/` باشیم، یک URL نسبی مانند `../images/photo.jpg` به آدرس `https://example.com/images/photo.jpg` اشاره می‌کند.

چرا URL مهم است؟

1. مسیریابی: بدون URL، مرورگر نمی‌داند چگونه اطلاعات را از سرور درخواست کند.

2. ساختاردهی به وب: URLها به سازماندهی منطقی اطلاعات در وب کمک می‌کنند.

3. سئو (SEO): URLهای خوانا و توصیفی (مثل `example.com/blog/seo-tips`) برای موتورهای جستجو بهتر هستند و رتبه بهتری دارند.

4. قابلیت اشتراک‌گذاری: کاربران می‌توانند آدرس دقیق یک صفحه را با دیگران به اشتراک بگذارند.

جمع‌بندی نکات کلیدی

|  |  |
| --- | --- |
| مفهوم | توضیح خلاصه |
| تعریف | آدرس کامل و دقیق یک منبع (صفحه، عکس، فایل) در وب |
| تشبیه | آدرس پستی کامل یک ساختمان |
| اجزای اصلی | پروتکل (قوانین) + میزبان (آدرس سرور) + مسیر (محل فایل) + پارامترها (داده اضافی) |
| تفاوت با دامنه | دامنه بخشی از URL است. URL یک آدرس کامل‌تر است |
| مثال | <https://www.google.com/maps> |

# بررسی مفهوم HTTP:

HTTP چیست؟

فرض کنید شما در یک رستوران هستید. شما (مرورگر) یک درخواست (Request) به پیشخدمت (سرور) می‌دهید: "یک ساندویچ مرغ می‌خواهم."

پیشخدمت درخواست شما را می‌شنود، به آشپزخانه می‌رود و سپس یک پاسخ (Response) برای شما می‌آورد: "بسیار خوب، این هم ساندویچ مرغ شما."

HTTP دقیقاً نقش قوانین و زبان مشترک برای این تبادل درخواست و پاسخ بین مرورگر و سرور را بازی می‌کند.

مرورگر (Client) = مشتری رستوران

سرور (Server) = پیشخدمت رستوران

HTTP = پروتکل و قوانین ارتباطی بین مشتری و پیشخدمت

HTTP مخفف Hypertext Transfer Protocol به معنای "پروتکل انتقال ابرمتن" است.

تعریف فنی

HTTP یک پروتکل لایه کاربرد (Application Layer) است که برای انتقال ابرمتن‌ها (مانند صفحات وب) و دیگر محتوای چندرسانه‌ای در شبکه جهانی وب استفاده می‌شود. HTTP پایه و اساس هرگونه تبادل داده در وب است.

HTTP چگونه کار می‌کند؟ (درخواست-پاسخ)

کاربر در مرورگر خود آدرس سایتی مانند `http://example.com` را وارد می‌کند.

1. درخواست (Request): مرورگر یک درخواست HTTP به سرور می‌فرستد. این درخواست شامل اطلاعاتی مانند:

روش (Method): نوع عمل درخواستی (مثلاً GET برای دریافت داده).

مسیر (URL): آدرس منبع درخواستی.

هدرها (Headers): اطلاعات اضافی درباره درخواست.

2. پاسخ (Response): سرور درخواست را پردازش کرده و یک پاسخ HTTP برمی‌گرداند. این پاسخ شامل:

کد وضعیت (Status Code): عددی که نشان می‌دهد درخواست موفق بود یا خطایی رخ داده.

هدرهای پاسخ (Response Headers): اطلاعاتی درباره پاسخ.

بدنه (Body): محتوای اصلی درخواست شده (مثلاً کد HTML صفحه).

ویژگی‌های کلیدی HTTP

1. بدون حالت (Stateless):

مفهوم: سرور، وضعیت قبلی مرورگر (مثلاً اینکه درخواست قبلی چه بوده) را به خاطر نمی‌سپارد. هر درخواست مستقل از درخواست‌های قبل است.

مشکل: چگونه وب‌سایت‌ها شما را به خاطر می‌سپارند (مثلاً لاگین می‌مانید)؟

راه‌حل: از کوکی‌ها (Cookies) استفاده می‌شود. کوکی‌ها اطلاعات را در سمت مرورگر ذخیره کرده و با هر درخواست به سرور ارسال می‌کنند تا سرور "شما" را بشناسد.

2. مبتنی بر متن (Text-based):

دستورات و هدرهای HTTP به صورت متن ساده و قابل خواندن برای انسان هستند.

روش‌های رایج HTTP (HTTP Methods)

این روش‌ها عمل مورد نظر مرورگر را مشخص می‌کنند (مثل فعل در جمله):

|  |  |  |
| --- | --- | --- |
| روش | کاربرد | مثال |
| GET | دریافت داده از سرور | بارگذاری یک صفحه وب |
| POST | ارسال داده به سرور | ارسال اطلاعات فرم لاگین |
| PUT | به‌روزرسانی داده موجود در سرور | ویرایش پروفایل کاربر |
| DELETE | حذف داده از سرور | پاک کردن یک پست |

کدهای وضعیت HTTP (HTTP Status Codes)

این کدهای سه‌رقمی نتیجه درخواست را به صورت عددی نشان می‌دهند (مثلاً آیا موفق بود یا خطایی رخ داد؟):

|  |  |  |
| --- | --- | --- |
| دسته | معنی | مثال‌های معروف |
| 1xx (اطلاعاتی) | درخواست دریافت شده و پردازش ادامه دارد | - |
| 2xx (موفقیت) | درخواست با موفقیت دریافت و پردازش شد | 200 OK (موفقیت‌آمیز) |
| 3xx (تغییر مسیر) | برای تکمیل درخواست، عمل اضافی لازم است | 301 Moved Permanently (صفحه به آدرس جدید منتقل شده) |
| 4xx (خطای سمت کلاینت) | درخواست حاوی خطا یا غیرمعتبر است | 404 Not Found (صفحه پیدا نشد) |
| 5xx (خطای سمت سرور) | سرور در پردازش درخواست موفق نبود | 500 Internal Server Error (خطای داخلی سرور) |

معایب HTTP (چرا HTTPS ایجاد شد؟)

اطلاعات در HTTP به صورت متن ساده (Plain Text) منتقل می‌شوند. این یعنی:

اگر کسی ارتباط شما را استراق سمع کند، می‌تواند تمام اطلاعاتی که رد و بدل می‌شود (مانند رمز عبور، اطلاعات کارت بانکی) را به راحتی بخواند.

این یک مشکل بزرگ امنیتی است.

تفاوت HTTP با HTTPS

HTTPS (HTTP Secure) نسخه امن HTTP است.

مکانیزم: HTTPS از پروتکل SSL/TLS برای رمزگذاری (Encrypt) کردن تمام ارتباط بین مرورگر و سرور استفاده می‌کند.

نتیجه: حتی اگر داده‌ها intercepted شوند، برای فرد مهاجم قابل خواندن نیستند.

شناخت: وجود یک قفل در نوار آدرس مرورگر، نشان‌دهنده استفاده از HTTPS است.

امروزه استفاده از HTTPS برای تمام وب‌سایت‌ها به یک استاندارد ضروری تبدیل شده است.

# بررسی مفهوم HTTPS

حتماً! در ادامه به بررسی مفهوم HTTPS (پروتکل امن انتقال ابرمتن) به زبان ساده و جامع می‌پردازیم.

HTTPS چیست؟ (تشبیه ساده)

فرض کنید می‌خواهید یک محموله بسیار باارزش را برای دوست خود بفرستید. اگر آن را در یک جعبه معمولی بگذارید (مثل HTTP)، هر کسی در مسیر می‌تواند جعبه را باز کند، محتویات را ببیند یا حتی تغییر دهد.

اما اگر محموله را داخل یک جعبه امنیتی ضدگلوله قرار دهید که فقط شما و دوستتان کلید باز کردن آن را دارید (مثل HTTPS)، حتی اگر دزدان راهی هم جعبه را بدزدند، نمی‌توانند آن را باز کنند یا محتویات را بفهمند.

HTTPS دقیقاً همان جعبه امنیتی برای ارتباطات اینترنتی است.

HTTPS = نسخه امن شده HTTP که ارتباط بین مرورگر و سرور را رمزگذاری (Encrypt) می‌کند.

HTTPS مخفف Hypertext Transfer Protocol Secure به معنای "پروتکل امن انتقال ابرمتن" است.

تعریف فنی

HTTPS یک پروتکل ارتباطی امن برای انتقال داده‌ها در شبکه اینترنت است که با استفاده از پروتکل SSL/TLS، یک لایه امنیتی به پروتکل استاندارد HTTP اضافه می‌کند. این پروتکل از سه جنبه مهم محافظت می‌کند:

1. رمزگذاری (Encryption)

2. اعتبارسنجی (Authentication)

3. یکپارچگی داده (Data Integrity)

HTTPS چگونه کار می‌کند؟ (SSL/TLS Handshake)

وقتی شما به یک سایت HTTPS متصل می‌شوید (مثلاً `https://bank.com`)، قبل از شروع تبادل داده‌های اصلی، یک فرآیند امنیتی به نام SSL/TLS Handshake بین مرورگر و سرور انجام می‌شود.

مزایای کلیدی HTTPS

1. رمزگذاری داده (Encryption):

مهمترین مزیت. تمام داده‌هایی که بین شما و سایت رد و بدل می‌شود (رمز عبور، اطلاعات کارت بانکی، پیام‌های خصوصی) به صورت رمزگذاری شده انتقال می‌یابند.

حتی اگر کسی ارتباط را استراق سمع کند، نمی‌تواند محتوای آن را بخواند.

2. اعتبارسنجی (Authentication):

HTTPS با استفاده از گواهی SSL تأیید می‌کند که شما در حال ارتباط با وب‌سایت واقعی و قانونی هستید، نه یک سایت جعلی که خود را به جای آن جا زده است.

این گواهی را یک مرجع معتبر (Certificate Authority) صادر می‌کند.

3. حفظ یکپارچگی داده (Data Integrity):

HTTPS تضمین می‌کند که داده‌ها در مسیر انتقال دستکاری یا تغییر نکرده‌اند.

چگونه بفهمیم سایتی از HTTPS استفاده می‌کند؟

در نوار آدرس مرورگر خود به دنبال این نشانه‌ها باشید:

1. قفل (🔒) بسته: اصلی‌ترین نشانه. معمولاً کنار آدرس سایت نمایش داده می‌شود.

2. شروع آدرس با `https://`: به جای `http://`.

3. علامت "امن" (Secure): در برخی مرورگرها کلمه "امن" یا "Secure" نوشته می‌شود.

تفاوت HTTP با HTTPS (مقایسه کلی)

|  |  |  |
| --- | --- | --- |
| ویژگی | HTTPS | HTTP |
| امنیت | ندارد - داده‌ها به صورت متن ساده منتقل می‌شوند | دارد - داده‌ها کاملاً رمزگذاری می‌شوند |
| پورت پیش‌فرض | پورت ۸۰ | پورت ۴۴۳ |
| الگوریتم رمزنگاری | استفاده نمی‌کند | از SSL/TLS استفاده می‌کند |
| سرعت | کمی سریع‌تر (به دلیل عدم overhead رمزگذاری) | کمی کندتر (به دلیل پردازش‌های رمزگذاری) |
| امنیت برای | مناسب اطلاعات عمومی و غیرحساس | ضروری برای اطلاعات حساس (لاگین، بانک، پرداخت) |
| نماد در مرورگر | ⓘ یا "ناامن" | 🔒 یا "امن" |

چرا HTTPS امروزه ضروری است؟

1. حفاظت از اطلاعات کاربران: اولین و مهمترین دلیل.

2. اعتماد و اعتبار (Credibility): کاربران به سایت‌هایی که قفل سبز رنگ دارند بیشتر اعتماد می‌کنند.

3. سئو (SEO): گوگل به سایت‌های دارای HTTPS رتبه بهتری می‌دهد.

4. امکان استفاده از قابلیت‌های جدید وب: بسیاری از APIهای مدرن مرورگر (مانند دسترسی به مکان) فقط برای سایت‌های HTTPS فعال هستند.

5. افزایش امنیت کلی اینترنت.

گواهی SSL (SSL Certificate) چیست؟

برای راه‌اندازی HTTPS، صاحب وب‌سایت باید یک گواهی SSL از یک مرجع معتبر خریداری یا دریافت کند. این گواهی مانند یک کارت شناسایی دیجیتال برای وب‌سایت است که:

هویت صاحب سایت را تأیید می‌کند.

حاوی یک کلید عمومی برای برقراری ارتباط امن است.

# بررسی سایت‌های Static و Dynamic

سایت‌های استاتیک و داینامیک دو نوع اصلی از وبسایت‌ها هستند که از نظر فنی، کاربرد و مدیریت با یکدیگر تفاوت‌های بنیادینی دارند. انتخاب بین این دو به طور کامل به اهداف و نیازهای کسب‌وکار شما بستگی دارد.

در جدول زیر می‌توانید خلاصه‌ای از مهم‌ترین تفاوت‌های این دو نوع سایت را مشاهده کنید:

|  |  |  |
| --- | --- | --- |
| ویژگی | سایت استاتیک (ایستا) | سایت داینامیک (پویا) |
| محتوا و تعامل | محتوای ثابت و از پیش نوشته شده. فاقد تعامل کاربر (مانند ثبت‌نام یا نظر دادن) . | محتوای پویا و تغییرپذیر. تعامل کامل کاربر (ثبت‌نام، خرید، سیستم کاربری و غیره) . |
| فناوری‌های مورد استفاده | عمدتاً HTML و CSS، گاهی جاوااسکریپت . | زبان‌های سمت سرور (مانند PHP، ASP.NET) و یک پایگاه داده (Database) |
| مدیریت و به‌روزرسانی | برای هر تغییر کوچک نیاز به دانش فنی و تغییر کدها توسط برنامه‌نویس است . | از طریق پنل مدیریت (CMS)؛ به راحتی و بدون نیاز به کدنویسی قابل مدیریت است . |
| هزینه | هزینه طراحی اولیه کمتر، اما هزینه به‌روزرسانی‌های مکرر می‌تواند بالا باشد . | هزینه طراحی اولیه بالاتر، اما به‌صرفه‌تر در بلندمدت به دلیل مدیریت آسان . |
| سرعت بارگذاری | سرعت بسیار بالا به دلیل سادگی ساختار و نبود درخواست به پایگاه داده . | سرعت نسبتاً کمتر (به دلیل پردازش‌های سمت سرور)، اما قابل بهینه‌سازی |
| امنیت | امنیت بالاتر به دلیل سادگی ساختار و نداشتن پایگاه داده و نقطه ورود پیچیده . | نیاز به تمهیدات امنیتی بیشتر به دلیل پیچیدگی و وجود پایگاه داده . |
| مثال‌های کاربردی | سایت‌های معرفی (پورتفولیو)، رزومه آنلاین، صفحات لندینگ، بروشورهای الکترونیک | فروشگاه‌های اینترنتی، سایت‌های خبری، شبکه‌های اجتماعی، وبلاگ‌ها، پرتال‌های سازمانی |

چگونه بین سایت استاتیک و داینامیک انتخاب کنیم؟

انتخاب صحیح به طور مستقیم به پاسخ این سوال بستگی دارد: هدف اصلی شما از راه‌اندازی وبسایت چیست؟

- یک سایت استاتیک مناسب شماست اگر:

- نیاز به یک بروشور آنلاین ساده و ثابت دارید که اطلاعات پایه شرکت یا خدمات شما را نمایش دهد.

- محتوای سایت شما به ندرت نیاز به تغییر دارد (مثلاً یک یا دو بار در سال).

- بودجه اولیه کمی دارید و اولویت شما سرعت بسیار بالا و امنیت است.

- نیازی به ویژگی‌های پیچیده‌ای مانند عضویت کاربران، سبد خرید یا درگاه پرداخت ندارید .

- یک سایت داینامیک انتخاب بهتری است اگر:

- قصد دارید به طور منظم محتوا منتشر کنید (مثلاً مقاله، خبر یا محصول جدید).

- نیازمند تعامل با کاربران هستید (مانند نظردهی، ثبت‌نام، سیستم سفارش‌دهی).

- می‌خواهید خودتان به راحتی و بدون وابستگی به برنامه‌نویس، سایت را مدیریت کنید.

- کسب‌وکار شما در حال رشد است و به قابلیت‌های گسترش‌پذیر نیاز دارید .

یک نکته مهم: راه حل‌های مدرن (Static Site Generators)

امروزه با فناوری‌هایی مانند Static Site Generators (مثل Gatsby, Next.js) می‌توان تا حدی مرز بین این دو نوع سایت را کمرنگ کرد. این فناوری‌ها امکان ایجاد سایت‌هایی با سرعت و امنیت سایت‌های استاتیک را فراهم می‌کنند، در حالی که محتوای آن‌ها می‌تواند از یک سیستم مدیریت محتوا (CMS) به صورت پویا استخراج شود. این گزینه می‌تواند برای پروژه‌های خاصی بسیار ایده‌آل باشد.

# بررسی request و Response های یک وب سایت

Request و Response چیست؟

فرض کنید شما به یک رستوران رفتید:

1. شما (مرورگر) به پیشخدمت (سرور) می‌گویید: "یک پیتزا پپرونی سفارش می‌دهم."

→ این یک Request (درخواست) است.

2. پیشخدمت (سرور) به آشپزخانه می‌رود، پیتزا را آماده می‌کند و برای شما (مرورگر) می‌آورد: "بفرمایید پیتزای شما."

→ این یک Response (پاسخ) است.

این مکالمه پایه و اساس تمام ارتباطات در دنیای وب است.

تعریف فنی

Request (درخواست): پیامی است که مرورگر (Client) به سرور می‌فرستد تا اطلاعات یا سرویس خاصی را درخواست کند.

Response (پاسخ): پیامی است که سرور پس از پردازش Request، برای مرورگر می‌فرستد و حاوی نتیجه درخواست (داده‌های مورد نظر یا خطا) است.

Request و Response چگونه کار می‌کنند؟

وقتی شما آدرس یک سایت را در مرورگر وارد می‌کنید یا روی لینکی کلیک می‌کنید، این فرآیند به طور مداوم تکرار می‌شود.

اجزای تشکیل‌دهنده یک Request (درخواست)

هر Request معمولاً شامل سه بخش اصلی است:

۱. خط شروع (Request Line)

- متد (Method): نوع عمل درخواستی را مشخص می‌کند.

- آدرس (URL): منبع درخواستی را مشخص می‌کند.

- ورژن پروتکل: مانند `HTTP/1.1`

مثال: `GET /products/index.html HTTP/1.1`

۲. هدرها (Headers)

اطلاعات اضافی درباره درخواست را شامل می‌شوند:

- `Host: example.com` (دامنه سایت مقصد)

- `User-Agent: Mozilla/5.0...` (اطلاعات مرورگر و سیستم عامل)

- `Cookie: session\_id=abc123` (اطلاعات ذخیره شده در مرورگر)

- `Accept: text/html,application/xhtml+xml` (نوع داده‌هایی که مرورگر می‌تواند دریافت کند)

۳. بدنه (Body) - اختیاری

حاوی داده‌ای است که به سرور ارسال می‌شود (مثلاً اطلاعات فرم ثبت نام).

انواع متدهای رایج Request (HTTP Methods)

|  |  |  |
| --- | --- | --- |
| متد | کاربرد | مثال |
| GET | دریافت اطلاعات از سرور | بارگذاری یک صفحه وب |
| POST | ارسال اطلاعات به سرور | ثبت نام کاربر، ارسال نظر |
| PUT | به‌روزرسانی اطلاعات موجود | ویرایش پروفایل کاربر |
| DELETE | حذف اطلاعات | پاک کردن یک پست |

اجزای تشکیل‌دهنده یک Response (پاسخ)

هر Response نیز شامل سه بخش اصلی است:

۱. خط وضعیت (Status Line)

- ورژن پروتکل: مانند `HTTP/1.1`

- کد وضعیت (Status Code): عدد سه رقمی که نتیجه درخواست را نشان می‌دهد.

- توضیح وضعیت: شرح متنی کد وضعیت.

مثال: `HTTP/1.1 200 OK`

۲. هدرهای پاسخ (Response Headers)

اطلاعات اضافی درباره پاسخ:

- `Content-Type: text/html` (نوع محتوای بازگشتی)

- `Content-Length: 1024` (حجم داده بازگشتی)

- `Set-Cookie: session\_id=xyz456` (ذخیره اطلاعات در مرورگر)

۳. بدنه (Body)

محتوای اصلی پاسخ (مثلاً کد HTML صفحه، داده JSON، یا یک فایل).

یک مثال واقعی از Request و Response

موقعیت: کاربر روی لینک "تماس با ما" کلیک می‌کند.

Request ارسالی توسط مرورگر:

```

GET /contact-us.html HTTP/1.1

Host: example.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36

Accept: text/html,application/xhtml+xml

Cookie: user\_lang=fa

```

Response دریافتی از سرور:

```

HTTP/1.1 200 OK

Content-Type: text/html; charset=utf-8

Content-Length: 1542

Date: Sat, 01 Jan 2024 12:00:00 GMT

HTML Response come here:

```

ابزارهای مشاهده Request و Response (برای توسعه‌دهندگان)

- Developer Tools مرورگر: در Chrome/Firefox با F12 باز می‌شود. تب Network تمام Request/Responseها را نشان می‌دهد.

- Postman: ابزاری برای تست و بررسی APIها با ساخت Requestهای دستی.

- cURL: یک ابزار خط فرمان برای ارسال Request.

# بررسی نحوه رندرگیری کدها توسط مرورگر

مرورگرها برای تبدیل کدهای HTML، CSS و JavaScript به صفحات وب قابل مشاهده، مراحل پیچیده و به هم پیوستهای را طی می کنند. درک این فرآیند به شما کمک میکند تا وبسایتهای سریعتر و کاراتری توسعه دهید.

در جدول زیر مراحل اصلی این فرآیند بهطور خلاصه آمده است:

|  |  |  |
| --- | --- | --- |
| مرحله | توضیح | خروجی |
| ۱. پارس کردن HTML | مرورگر کد HTML را خط به خط میخواند و به یک ساختار درختی از اشیاء تبدیل میکند. | DOM (Document Object Model) |
| ۲. پارس کردن CSS | استایلهای CSS نیز به یک ساختار درختی قابل فهم برای مرورگر تبدیل میشوند | CSSOM (CSS Object Model |
| ۳. ساخت درخت رندر | DOM و CSSOM با هم ترکیب میشوند تا درختی ساخته شود که فقط شامل عناصر قابل مشاهده صفحه است (عناصری مانند `display: none` در این درخت قرار نمیگیرند). | Render Tree (درخت رندر) |
| ۴. لایه بندی (Layout) | مرورگر موقعیت و اندازه دقیق هر عنصر در صفحه را محاسبه میکند. این مرحله گاهی Reflow نیز نامیده میشود. | موقعیت و ابعاد هندسی |
| ۵. رنگآمیزی (Painting) | مرورگر پیکسلهای هر عنصر (مانند رنگها، تصاویر و حاشیهها) را رسم میکند. این مرحله Repaint نیز نام دارد. | لایه های پیکسلی |
| ۶. ترکیب (Compositing) | در نهایت، لایههای مختلفی که رنگآمیزی شدهاند، به ترتیب صحیح روی هم قرار میگیرند تا صفحه نهایی تشکیل شود. | صفحه وب نهایی |

نقش JavaScript و بهینه سازی عملکرد

JavaScript به طور خاص بر روند رندر تأثیر میگذارد، زیرا معمولاً تجزیه و اجرای آن باعث توقف ساخت DOM میشود. برای کاهش این تأثیر منفی:

- جاوااسکریپت را در انتهای صفحه (قبل از تگ بسته `body`) قرار دهید تا ابتدا محتوای HTML بارگذاری و پارس شود.

- از ویژگیهای async` یا defer` برای اسکریپت های خارجی استفاده کنید تا بارگیری آنها به صورت غیرمسدودکننده انجام شود.

- مسیر رندر بحرانی را بهینه کنید: تعداد منابع (فایلهای CSS و JS) را کاهش دهید، حجم آنها را با فشردهسازی (Minify) کم کنید و با تکنیکهایی مانند Lazy Loading بارگذاری منابع غیرضروری برای نمایش اولیه را به تأخیر بیندازید.

تفاوت Repaint و Reflow

دو مفهوم مهم در بهروزرسانی صفحات هستند:

- Repaint (رنگآمیزی مجدد): هنگامی رخ میدهد که ظاهر یک عنصر تغییر کند، اما بر Layout صفحه تأثیر نگذارد (مانند تغییر رنگ پسزمینه). این عمل نسبتاً سبک است.

- Reflow (بازچینی): هنگامی رخ میدهد که تغییری در صفحه ایجاد شود که بر Layout و موقعیت عناصر تأثیر بگذارد (مانند تغییر اندازه پنجره مرورگر یا اضافه کردن یک عنصر DOM). این عمل پرهزینه است و ممکن است باعث شود مرورگر بخشی یا تمام صفحه را مجدداً محاسبه کند.

نتیجه گیری

درک دقیق فرآیند رندر مرورگر یک مزیت بزرگ برای توسعهدهندگان وب است. با بهینهسازی هر یک از این مراحل، به ویژه مسیر رندر بحرانی، میتوانید تجربه کاربری بسیار بهتری از نظر سرعت بارگذاری و تعامل روان ایجاد کنید.

اگر درباره تأثیر یک المان یا تکنیک خاص (مثلاً استفاده از فونتهای وب یا تصاویر بزرگ) بر رندر صفحه سؤال خاصی دارید، خوشحال میشوم در مورد آن نیز صحبت کنیم.

# آموزش HTML

## [کامنت ها در HTML](webdesign/html/comments)

کامنت‌ها در HTML ابزار بسیار مهمی برای توسعه‌دهندگان هستند که توسط مرورگر نمایش داده نمی‌شوند اما برای اهداف مختلفی استفاده می‌شوند.

نحوه نوشتن کامنت در HTML

|  |
| --- |
| <!-- این یک کامنت تک خطی است -->  <!--  این یک کامنت  چند خطی است  -->  <div class="container">  <!-- کامنت داخل عناصر -->  <p>محتوا</p>  </div> |

[کامنت ها مثال اول](webdesign/html/comments/first.html)

کاربردهای اصلی کامنت‌ها

1. توضیح کد (Documentation)

|  |
| --- |
| <!-- نوار ناوبری اصلی سایت -->  <nav>  <ul>  <li><a href="/">خانه</a></li>  <li><a href="/about">درباره ما</a></li>  </ul>  </nav>  <!-- بخش مقالات -->  <section id="articles">  <!-- هر مقاله در یک کارت نمایش داده می‌شود -->  <article class="card">  <h3>عنوان مقاله</h3>  <p>متن مقاله...</p>  </article>  </section> |

[Documentation](webdesign/html/comments/Documentation.html)

1. غیرفعال کردن موقت کد (Debugging)

|  |
| --- |
| <!-- این بخش در حال توسعه است -->  <!--  <div class="feature">  <p>این ویژگی به زودی اضافه می‌شود</p>  </div>  -->  <!-- کد قدیمی که ممکن است بعداً نیاز شود -->  <!--  <button class="old-button">دکمه قدیمی</button>  --> |

[Debugging](webdesign/html/comments/Debugging.html)

1. علامت‌گذاری و سازماندهی

|  |
| --- |
| <!-- ===== HEADER START ===== -->  <header>  <!-- لوگو -->  <div class="logo">سایت من</div>    <!-- منوی اصلی -->  <nav>...</nav>  </header>  <!-- ===== HEADER END ===== -->  <!-- ===== MAIN CONTENT START ===== -->  <main>...</main>  <!-- ===== MAIN CONTENT END ===== --> |

[categorized](webdesign/html/comments/cat.html)

نکات مهم درباره کامنت‌ها

نحوه پردازش توسط مرورگر:

- کامنت‌ها جزئی از DOM نمی‌شوند

- در Render Tree لحاظ نمی‌شوند

- بر عملکرد مرورگر تأثیر منفی نمی‌گذارند

- در سورس کد صفحه قابل مشاهده هستند

بهترین تمرین :

|  |
| --- |
| <!-- خوب: کامنت واضح و مختصر -->  <!-- منوی کاربر -->  <div class="user-menu">...</div>  <!-- بد: کامنت بی‌معنی -->  <!-- div -->  <div>...</div>  <!-- خوب: کامنت برای کد پیچیده -->  <!-- محاسبه قیمت با درنظرگیری تخفیف و مالیات -->  <div class="price-calculator">  <script>  // کدهای پیچیده JavaScript  </script>  </div> |

[practices](webdesign/html/comments/practices.html)

کامنت‌های شرطی برای مرورگرهای قدیمی

|  |
| --- |
| <!-- نمایش پیام برای کاربران IE قدیمی -->  <!--[if lt IE 9]>  <div class="browser-warning">  لطفاً مرورگر خود را آپدیت کنید  </div>  <![endif]--> |

[if](webdesign/html/comments/if.html)

نکات پیشرفته

کامنت‌ها در ابزارهای توسعه:

کامنت‌ها در DevTools مرورگر قابل مشاهده هستند و به دیباگ کمک می‌کنند.

تأثیر بر سئو:

کامنت‌ها معمولاً بر سئو تأثیر مستقیم ندارند، اما کامنت‌های زیاد ممکن است نسبت کد به محتوا را کاهش دهند.

امنیت:

هرگز اطلاعات حساس در کامنت نگذارید:

|  |
| --- |
| <!-- بد: اطلاعات حساس -->  <!-- ادمین: admin, رمز: 123456 -->  <!-- بد: کد مخفی -->  <!-- API Key: ak\_123456789 --> |

[sensetive](webdesign/html/comments/sensetive.html)

جمع‌بندی

کامنت‌ها ابزار قدرتمندی برای:

- بهبود خوانایی کد

- دیباگ راحت‌تر

- مستندسازی پروژه

- همکاری تیمی بهتر

اما به یاد داشته باشید:

- زیاده‌روی نکنید

- اطلاعات حساس نگذارید

- کدهای قدیمی را حذف کنید

## [معرفی tag ها در Html و انواع آن (singel & pairs)](webdesign/html/singel-pairs)

HTML از تگ‌ها (Tags) برای ساختاردهی و نمایش محتوای وب استفاده می‌کند. تگ‌ها به دو دسته اصلی تقسیم می‌شوند:

انواع تگ‌ها از نظر ساختاری

۱. تگ‌های جفتی (Paired Tags)

این تگ‌ها دارای تگ آغازین و پایانی هستند.

|  |
| --- |
| <tag>محتوای داخل تگ</tag> |

[PairedTags](webdesign/html/singel-pairs/PairedTags.html)

مثال‌ها:

|  |
| --- |
| <p>این یک پاراگراف است</p>  <div>این یک بخش است</div>  <h1>این یک عنوان است</h1>  <a href="https://example.com">این یک لینک است</a> |

[Example](webdesign/html/singel-pairs/sample.html)

۲. تگ‌های تکی (Single Tags)

این تگ‌ها فقط تگ آغازین دارند و محتوایی بین آنها قرار نمی‌گیرد.

|  |
| --- |
| <tag> |

[Singletag](webdesign/html/singel-pairs/Singletag.html)

مثال‌ها:

|  |
| --- |
| <br> <!-- خط شکست -->  <hr> <!-- خط افقی -->  <img> <!-- تصویر -->  <input> <!-- فیلد ورودی -->  <meta> <!-- متادیتا -->  <link> <!-- لینک به فایل خارجی --> |

[Samples2](webdesign/html/singel-pairs/samples2.html)

جدول مقایسه تگ‌های جفتی و تکی

|  |  |  |
| --- | --- | --- |
| ویژگی | تگ‌های جفتی | تگ‌های تکی |
| ساختار | `<tag>  محتوا  </tag>` | <tag /> |
| محتوای داخلی | دارد | ندارد |
| تگ پایانی | الزامی | ندارد |
| نحوه بستن | `</tag>` | <tag /> اختیاری |
| مثال | <p>  متن پاراگراف  </p> | <br> |

دسته‌بندی تگ‌ها بر اساس کاربرد

## [ساختار تگ ها در HTML](webdesign/html/tagsStructure)

درک پایه: تگ + صفت = عنصر کامل

فرض کنید هر تگ HTML مانند یک آدمک خام است و صفت‌ها مانند لباس‌ها و ابزار هستند که به آن ویژگی می‌دهند!

ساختار کلی:

|  |
| --- |
| <!-- attr => صفت است -->  <!-- value => مقدار صفت است که درون کوتیشن اجباری درج میشود-->  <tag attr="value">      محتوای تگ  </tag> |

[base](webdesign/html/tagsStructure/base.html)

انواع ساختار با صفت‌ها

مثال‌های ملموس:

|  |
| --- |
| <a href="https://google.com">برو به گوگل</a> |

[Sample](webdesign/html/tagsStructure/sample.html)

مثل نوشتن آدرس روی یک بلیط

|  |
| --- |
| <p dir="rtl">      این متن راست به چپ است  </p>  <p dir="ltr">      This content is left to right.  </p> |

[dir](webdesign/html/tagsStructure/dir.html)

مثل دادن یک اسم به جعبه

|  |
| --- |
| <div class="container">      محتوای داخل کانتینر  </div> |

[container](webdesign/html/tagsStructure/container.html)

۲. تگ‌های تکی با صفت

|  |
| --- |
| <tag attr="value"> |

## معرفی VS Code و پلاگین های آن

(material icons, material themes , open in browser , prettier ,rainbow brackets , path intellisense)

ویژوال استودیو کد (VS Code) یک ویرایشگر کد سبک‌وزن، قدرتمند و رایگان از مایکروسافت است که محبوبیت بسیار زیادی در بین توسعه‌دهندگان دارد. این ویرایشگر به لطف معماری مبتنی بر افزونه، امکان شخصی‌سازی و گسترش امکانات را تا حد زیادی فراهم می‌کند.

در ادامه، افزونه‌هایی که شما نام بردید را به طور کامل معرفی می‌کنم.

پلاگین‌های ظاهری و بصری

این دسته از افزونه‌ها به زیباتر شدن و کاربردی‌تر شدن محیط کدنویسی کمک می‌کنند.

|  |  |
| --- | --- |
| افزونه | توضیح |
| Material Icon Theme | آیکن‌های زیبا و مدرن بر اساس اصول Material Design برای فایل‌ها و پوشه‌های شما درون VS Code. |
| Material Theme | یک پوسته رنگی کامل که تمام محیط VS Code را بر اساس طراحی متریال زیبا می‌کند. |

پلاگین‌های افزایش بهره‌وری و عملکرد

این افزونه‌ها مستقیماً به شما در کدنویسی سریع‌تر، تمیزتر و با خطای کمتر کمک می‌کنند.

|  |  |
| --- | --- |
| افزونه | توضیح |
| Prettier | یک قالب‌بند (Formatter) کد که به صورت خودکار کدهای شما را مرتب و زیبا می‌کند. |
| Path Intellisense | هنگام نوشتن آدرس فایل‌ها (مثلاً در تگ `img` یا دستور `import`)، به صورت خودکار نام فایل‌ها و پوشه‌ها را پیشنهاد می‌دهد. |
| Rainbow Brackets | پرانتزها، کروشه‌ها و آکولادها را به صورت رنگین‌کمانی رنگ می‌کند |
| Open in Browser | این افزونه امکان می‌دهد فایل HTMLای که در حال ویرایش آن هستید را مستقیماً و با یک کلیک در مرورگر پیش‌فرض سیستم یا مرورگر دلخواهتان باز کنید. |

💡 نکات نصب و استفاده

- برای نصب، به تب Extensions در VS Code رفته (آیکون چهارگوش در نوار کناری) و نام افزونه را جستجو کنید.

- پس از نصب Prettier، بهتر است آن را به عنوان فرمت‌کننده پیش‌فرض برای زبان‌های مورد نظرتان در تنظیمات VS Code تنظیم کنید تا به طور خودکار upon save کدهایتان را مرتب کند.

- برای Open in Browser معمولاً از کلیدهای `Alt + B` برای بازکردن در مرورگر پیش‌فرض و `Shift + Alt + B` برای انتخاب از بین مرورگرها استفاده می‌شود.

## [بررسی ساختار یک سند Html](webdesign/html/HtmlStructure)

ساختار پایه HTML5

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>عنوان صفحه</title>  </head>  <body>  <!-- محتوای اصلی صفحه -->  </body>  </html> |

تگ `<!DOCTYPE html>`

نقش و اهمیت:

- اولین خط در هر سند HTML5

- تعیین نوع سند برای مرورگر

- اطمینان از رندر صحیح صفحه

سیر تکامل:

|  |
| --- |
| <!-- HTML 4.01 Strict -->  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  "http://www.w3.org/TR/html4/strict.dtd">  <!-- XHTML 1.0 -->  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <!-- HTML5 (ساده شده) -->  <!DOCTYPE html> |

تگ `<html>`

ویژگی‌های اصلی:

|  |
| --- |
| <html lang="fa" dir="rtl"> |

|  |  |  |
| --- | --- | --- |
| ویژگی | مقدار | توضیح |
| lang | fa`, `en`, `ar | زبان اصلی سند |
| dir | rtl`, `ltr | جهت نوشتار |

اهمیت `lang`:

- سئو (بهینه‌سازی برای موتورهای جستجو)

- دسترسی‌پذیری (برای نرم‌افزارهای صفحه‌خوان)

- ترجمه خودکار مرورگر

تگ `<head>` - مغز صفحه

محتوای تگ head:

|  |
| --- |
| <head>  <!-- اطلاعات متا -->  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">    <!-- عنوان و لینک‌ها -->  <title>فروشگاه اینترنتی</title>  <link rel="stylesheet" href="styles.css">    <!-- اسکریپت‌ها -->  <script src="script.js"></script>  </head> |

متا تگ‌های ضروری

1. تعیین encoding:

|  |
| --- |
| <meta charset="UTF-8"> |

- پشتیبانی از زبان فارسی

- نمایش صحیح کاراکترها

۲. تنظیم viewport:

|  |
| --- |
| <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

- ریسپانسیو کردن سایت

- تنظیم اندازه برای موبایل

۳. توضیحات صفحه:

|  |
| --- |
| <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

- تاثیر مستقیم بر سئو

- نمایش در نتایج جستجو

۴. کلمات کلیدی:

|  |
| --- |
| <meta name="viewport" content="width=device-width, initial-scale=1.0"> |

1. نویسنده صفحه:

|  |
| --- |
| <meta name="author" content="نام شما"> |

تگ `<title>`

اهمیت:

- نمایش در تب مرورگر

- عنوان در نتایج جستجو

- ذخیره در بوکمارک‌ها

مثال‌های خوب و بد:

|  |
| --- |
| <!-- بد: بدون عنوان -->  <title></title>  <!-- بد: عنوان عمومی -->  <title>صفحه جدید</title>  <!-- خوب: توصیفی و خاص -->  <title>فروشگاه اینترنتی دیجی‌کالا - خرید آنلاین</title> |

تگ `<body>` - بدنه اصلی

ساختار معمول:

|  |
| --- |
| <body>  <header>  <!-- هدر صفحه -->  </header>    <nav>  <!-- منوی -->  </nav>    <main>  <!-- محتوای اصلی -->  <article>  <!-- مقاله یا پست -->  </article>    <aside>  <!-- محتوای جانبی -->  </aside>  </main>    <footer>  <!-- پاورقی -->  </footer>  </body> |

تگ‌های معنایی جدید HTML5 در body:

|  |  |
| --- | --- |
| تگ | کاربرد |
| `<header>` | سربرگ صفحه یا بخش |
| `<nav>` | منوی ناوبری |
| `<main>` | محتوای اصلی (یکبار استفاده) |
| `<article>` | محتوای مستقل (پست، مقاله) |
| `<section>` | بخش بندی محتوا |
| `<aside>` | محتوای جانبی |
| `<footer>` | پاورقی |

مثال کامل یک سند HTML5

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta name="description" content="وبلاگ شخصی علی محمدی - مقالات برنامه‌نویسی">  <meta name="keywords" content="برنامه‌نویسی, وب, آموزش, HTML5">  <meta name="author" content="علی محمدی">  <title>وبلاگ شخصی علی محمدی - برنامه‌نویسی وب</title>  <link rel="stylesheet" href="css/styles.css">  </head>  <body>  <header>  <h1>وبلاگ شخصی علی محمدی</h1>  <p>مقالات آموزشی برنامه‌نویسی وب</p>  </header>    <nav>  <ul>  <li><a href="home">خانه</a></li>  <li><a href="articles">مقالات</a></li>  <li><a href="about">درباره من</a></li>  </ul>  </nav>    <main>  <article>  <h2>آموزش HTML5 برای مبتدیان</h2>  <p>محتوای مقاله...</p>  </article>  </main>    <footer>  <p>© 2024 کلیه حقوق محفوظ است.</p>  </footer>  </body>  </html> |

[full](webdesign/html/HtmlStructure/full.html)

نکات کلیدی

بهترین روش‌ها:

1. همیشه با `<!DOCTYPE html>` شروع کنید

2. زبان و جهت صفحه را مشخص کنید

3. viewport را برای موبایل تنظیم کنید

4. از تگ‌های معنایی HTML5 استفاده کنید

5. عنوان توصیفی و منحصر به فرد انتخاب کنید

خطاهای رایج:

|  |
| --- |
| <!-- ❌ فراموش کردن DOCTYPE -->  <html>  <head>...</head>  <!-- ❌ قرار دادن محتوای قابل مشاهده در head -->  <head>  <h1>این درست نیست!</h1>  </head>  <!-- ❌ چندین تگ main -->  <main>...</main>  <main>...</main> |

## معرفی BOM و DOM

BOM - Browser Object Model (مدل شیءگرای مرورگر)

مفهوم کلی:

BOM مربوط به مرورگر است و به شما امکان کنترل پنجره مرورگر، تب‌ها، تاریخچه و... را می‌دهد.

ساختار سلسله مراتبی BOM:

|  |
| --- |
| window (پنجره مرورگر)  ├── navigator (اطلاعات مرورگر)  ├── location (آدرس صفحه)  ├── history (تاریخچه)  ├── screen (اطلاعات صفحه نمایش)  └── document (سند HTML - که خود DOM است) |

اجزای اصلی BOM:

1. window - شیء اصلی

|  |
| --- |
| // باز کردن پنجره جدید  window.open('https://example.com', '\_blank');  // بستن پنجره جاری  window.close();  // اندازه پنجره  console.log(window.innerWidth); // عرض داخلی  console.log(window.innerHeight); // ارتفاع داخلی |

1. navigator - اطلاعات مرورگر

|  |
| --- |
| console.log(navigator.userAgent); // شناسه مرورگر  console.log(navigator.language); // زبان مرورگر  console.log(navigator.onLine); // وضعیت اتصال اینترنت |

1. location - مدیریت آدرس

|  |
| --- |
| // تغییر مسیر  location.href = 'https://google.com';  // رفرش صفحه  location.reload();  // اطلاعات آدرس  console.log(location.protocol); // https:  console.log(location.hostname); // example.com  console.log(location.pathname); // /page.html |

1. history - مدیریت تاریخچه

|  |
| --- |
| // بازگشت به صفحه قبل  history.back();  // رفتن به صفحه بعد  history.forward();  // رفتن به چند صفحه قبل/بعد  history.go(-2); // دو صفحه قبل |

1. screen - اطلاعات نمایشگر

|  |
| --- |
| console.log(screen.width); // عرض کل نمایشگر  console.log(screen.height); // ارتفاع کل نمایشگر  console.log(screen.colorDepth); // عمق رنگ |

### DOM - Document Object Model (مدل شیءگرای سند)

مفهوم کلی:

DOM مربوط به سند HTML است و به شما امکان دسترسی و تغییر محتوای صفحه را می‌دهد.

ساختار درختی DOM:

|  |
| --- |
| document (ریشه)  ├── html  ├── head  │ ├── title  │ └── meta  └── body  ├── h1  ├── p  └── div |

انواع اصلی نودها در DOM:

|  |  |  |
| --- | --- | --- |
| نوع نود | مثال | توضیح |
| Element | P , div | تگ‌های HTML |
| Attribute | class="red" | ویژگی‌های تگ‌ها |
| Text | سلام دنیا | متن داخل تگ‌ها |
| Comment | `<!-- کامنت -->` | کامنت‌ها |

## نمایش درخت Html

درخت DOM چیست؟

DOM Tree یک نمایش سلسله مراتبی و درختی از ساختار HTML صفحه است که مرورگر آن را می‌سازد.

مثال ساده:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>صفحه من</title>  </head>  <body>  <h1>سلام دنیا</h1>  <p>این یک پاراگراف است</p>  </body>  </html> |

درخت DOM مربوطه:

|  |
| --- |
| document (ریشه)  │  ├── <!DOCTYPE html>  │  └── <html>  ├── <head>  │ └── <title>  │ └── "صفحه من" (متن)  │  └── <body>  ├── <h1>  │ └── "سلام دنیا" (متن)  │  └── <p>  └── "این یک پاراگراف است" (متن) |

ساختار کامل درخت DOM

مثال پیچیده‌تر:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <title>فروشگاه اینترنتی</title>  </head>  <body>  <header class="main-header">  <h1 id="site-title">فروشگاه من</h1>  <nav>  <ul>  <li><a href="home">خانه</a></li>  <li><a href="products">محصولات</a></li>  </ul>  </nav>  </header>    <main>  <article>  <h2>محصولات ویژه</h2>  <div class="product">  <img src="product1.jpg" alt="محصول ۱">  <p>توضیحات محصول</p>  </div>  </article>  </main>  </body>  </html> |

درخت DOM کامل:

|  |
| --- |
| document  │  ├── <!DOCTYPE html>  │  └── <html lang="fa">  ├── <head>  │ ├── <meta charset="UTF-8">  │ └── <title>  │ └── "فروشگاه اینترنتی"  │  └── <body>  ├── <header class="main-header">  │ ├── <h1 id="site-title">  │ │ └── "فروشگاه من"  │ │  │ └── <nav>  │ └── <ul>  │ ├── <li>  │ │ └── <a href="home">  │ │ └── "خانه"  │ │  │ └── <li>  │ └── <a href="products">  │ └── "محصولات"  │  └── <main>  └── <article>  ├── <h2>  │ └── "محصولات ویژه"  │  └── <div class="product">  ├── <img src="product1.jpg" alt="محصول ۱">  └── <p>  └── "توضیحات محصول" |

انواع Node در درخت DOM

1. Element Node (گره عنصر)

|  |
| --- |
| <div>, <p>, <span>, <img> |

1. Text Node (گره متن)

|  |
| --- |
| "این یک متن است" |

1. Attribute Node (گره ویژگی)

|  |
| --- |
| class="header", id="title", src="image.jpg" |

1. Comment Node (گره توضیح)

|  |
| --- |
| <!-- این یک توضیح است --> |

نمایش گرافیکی درخت DOM

|  |
| --- |
| document  │  ┌──────┴──────┐  <!DOCTYPE> <html>  │  ┌──────┴──────┐  <head> <body>  │ │  ┌──────┴──────┐ ┌───┴───┐  <meta charset> <title> <header> <main>  │ │  ┌───┴───┐ │  <h1> <nav> <article>  │ │ │  "فروشگاه من" <ul> ┌─┴─┐  │ <h2> <div>  ┌─┴─┐ │ │  <li><li> │ ┌─┴─┐  │ │ │ │ │  <a> <a> "محصولات" <img> <p>  │ │ │ │  "خانه" "محصولات" │ "توضیحات"  │  "محصول ۱" (alt) |

## بررسی مهفوم Semantic

فرض کنید می‌خواهید یک کتاب بنویسید. به جای اینکه همه چیز را فقط «فصل» بنامید، از بخش‌های معناداری مانند «پیشگفتار»، «فهرست»، «فصل‌های اصلی»، «نتیجه‌گیری» و «نمایه» استفاده می‌کنید. تگ‌های معنایی در HTML دقیقاً همین کار را برای صفحات وب انجام می‌دهند. آن‌ها به مرورگرها، موتورهای جستجو مانند گوگل و توسعه‌دهندگان کمک می‌کنند تا معنای هر بخش از صفحه را به سرعت درک کنند، نه فقط اینکه آن بخش چگونه به نظر می‌رسد.

مزایای اصلی استفاده از این تگ‌ها عبارتند از:

سئوی بهتر: موتورهای جستجو محتوای صفحه شما را بهتر درک می‌کنند.

دسترسی‌پذیری: برای کاربرانی که از نرم‌افزارهای صفحه‌خوان استفاده می‌کنند، درک ساختار صفحه آسان‌تر می‌شود.

کد خوانا و قابل نگهداری: سازماندهی و فهم کد برای توسعه‌دهندگان، به ویژه در پروژه‌های بزرگ، ساده‌تر می‌شود.

جدول کامل تگ‌های سمانتیک:

|  |  |
| --- | --- |
| نام تگ | توضیح کاربرد |
| article | محتوای مستقل و خودکفا را تعریف می‌کند، مانند یک پست وبلاگ، مقاله خبری یا نظر کاربر. |
| aside | محتوایی را تعریف می‌کند که جدا از محتوای اصلی صفحه است، مانند یک نوار کناری (سایدبار). |
| details | جزئیات اضافی را مشخص می‌کند که کاربر می‌تواند به صورت اختیاری آن را باز یا بسته کند (یک ویجت تعاملی). |
| figcaption | یک عنوان برای عنصر `<figure>` تعریف می‌کند (توضیح برای عکس یا نمودار). |
| figure | محتوای خود-contained مانند عکس، نمودار یا قطعه کد را مشخص می‌کند. |
| footer | یک پاورقی برای سند یا یک بخش خاص تعریف می‌کند. معمولاً شامل اطلاعات تماس، حق تکثیر و لینک‌های مرتبط است. |
| header | یک سربرگ برای سند یا یک بخش خاص تعریف می‌کند. اغلب شامل لوگو، عنوان و منوی می‌شود. |
| main | محتوای اصلی و منحصربه‌فرد سند را مشخص می‌کند. این تگ باید در هر صفحه فقط یک بار استفاده شود |
| Mark | متنی را که باید هایلایت یا علامت‌گذاری شود، تعریف می‌کند |
| nav | بخشی از صفحه که حاوی لینک‌های ناوبری اصلی است را تعریف می‌کند، مانند منوی سایت. |
| section | یک بخش عمومی در یک سند را تعریف می‌کند. معمولاً برای گروه‌بندی محتوای مرتبط حول یک موضوع واحد استفاده می‌شود. |
| summary | یک عنوان قابل مشاهده برای عنصر `<details>` تعریف می‌کند. این اولین چیزی است که کاربر می‌بیند و با کلیک روی آن، جزئیات باز یا بسته می‌شود. |
| time | یک زمان (یا تاریخ) را به صورت قابل خواندن توسط ماشین تعریف می‌کند. |

چگونه از تگ‌های معنایی استفاده کنیم؟

برای شروع، سعی کنید در پروژه بعدی خود به جای تکیه بر تگ‌های عمومی مانند `<div>` برای همه چیز، از تگ‌های معنایی مناسب استفاده کنید. برای مثال، ساختار اصلی صفحه خود را با `header، main` و `<footer>` بچینید و منوها را داخل `<nav>` قرار دهید.

این کار در ابتدا ممکن است کمی تمرین نیاز داشته باشد، اما به مرور کدهای شما حرفه‌ای‌تر و سازمان‌یافته‌تر خواهند شد.

## [تگهای HTML](webdesign/html/tags)

## [معرفی تگ p و display آن](webdesign/html/tags/p)

تگ `<p>` (پاراگراف)

تعریف ساده:

تگ `<p>` برای ایجاد پاراگراف‌های متنی در صفحات وب استفاده می‌شود.

ساختار پایه:

|  |
| --- |
| <p>این یک پاراگراف نمونه است.</p>  <p>این پاراگراف دوم است.</p> |

[p](webdesign/html/tags/p/p.html)

## [معرفی تگ های Heading](webdesign/html/tags/heading)

معرفی تگ‌های Heading (عنوان‌بندی)

تگ‌های Heading چیست؟

تگ‌های Heading برای عنوان‌بندی و ساختاردهی محتوای صفحه استفاده می‌شوند. این تگ‌ها از `h1` تا `h6` وجود دارند.

سطوح مختلف Heading:

|  |
| --- |
| <h1>مهمترین عنوان (سطح 1)</h1>      <h2>عنوان سطح 2</h2>      <h3>عنوان سطح 3</h3>      <h4>عنوان سطح 4</h4>      <h5>عنوان سطح 5</h5>      <h6>کمترین اهمیت (سطح 6)</h6> |

[Levels](webdesign/html/tags/heading/index.html)

جدول مشخصات تگ‌های Heading

|  |  |  |  |
| --- | --- | --- | --- |
| تگ | اندازه پیش‌فرض | کاربرد | اهمیت سئو |
| h1 | 2em | عنوان اصلی صفحه | ⭐⭐⭐⭐⭐ |
| h2 | 1.5em | بخش‌های اصلی | ⭐⭐⭐⭐ |
| h3 | 1.17em | زیربخش‌ها | ⭐⭐⭐ |
| h4 | 1em | عناوین فرعی | ⭐⭐ |
| h5 | 0.83em | عناوین کوچک | ⭐ |
| h6 | 0.67em | کمترین اهمیت | ☆ |

ویژگی‌های اصلی Headingها

1. سلسله مراتب اهمیت

|  |
| --- |
| <h1>فروشگاه اینترنتی دیجی‌کالا</h1>      <h2>دسته‌بندی محصولات</h2>      <h3>لوازم الکترونیکی</h3>      <h4>موبایل و تبلت</h4>      <h5>گوشی‌های سامسونگ</h5>      <h6>مدل گلکسی S24</h6> |

[hierarchy](webdesign/html/tags/heading/hierarchy%20.html)

۲. display پیش‌فرض: block

- هر Heading در خط جدید شروع می‌شود

- فاصله بالا و پایین خودکار دارد

- عرض کامل والد را اشغال می‌کند

۳. font-weight پیش‌فرض: bold

بهترین روش‌های استفاده

ساختار درست:

|  |
| --- |
| <h1>عنوان اصلی صفحه</h1>      <h2>مقدمه</h2>      <p>متن مقدمه...</p>      <h2>محصولات</h2>      <h3>محصولات الکترونیکی</h3>      <h4>لپ‌تاپ</h4>      <p>متن درباره لپ‌تاپ...</p>      <h2>نتیجه‌گیری</h2>      <p>متن نتیجه‌گیری...</p> |

❌ روش‌های نادرست:

|  |
| --- |
| <h1>عنوان اصلی صفحه</h1>  <!-- اشتباه: پرش سطح -->   <h3>محصولات الکترونیکی</h3>     <h1 style=”font-size:4em”>عنوان اصلی صفحه</h1> |

نباید هدینگ H2 جا می‌افتاد در بخش بعد نباید صرفاً برای بزرگ کردن متن از هدینگ استفاده کرد.

اهمیت در سئو (SEO)

بهترین روش‌ها برای سئو:

|  |
| --- |
| <!-- ✅ درست: ساختار منطقی -->  <h1>آموزش برنامه‌نویسی وب</h1>  <h2>آموزش HTML</h2>  <h3>تگ‌های Heading</h3>  <h2>آموزش CSS</h2>  <h3>استایل‌دهی متن</h3>  <!-- ✅ استفاده از کلمات کلیدی -->  <h1>خرید لباس ورزشی ارزان - فروشگاه آنلاین</h1>  <h2>لیست قیمت لباس‌های ورزشی</h2> |

[SEO](webdesign/html/tags/heading/seo.html)

## [معرفی تگ hr و نحوه Display آن](webdesign/html/tags/hr)

تگ `<hr>` (مخفف Horizontal Rule) یک تگ ساده اما کاربردی در HTML است که برای ایجاد یک خط افقی در صفحه وب استفاده می‌شود.

ویژگی‌های اصلی:

- یک تگ self-closing است (نیازی به تگ پایانی ندارد)

- به صورت پیش‌فرض یک خط افقی ایجاد می‌کند

- برای جداسازی محتوا و ایجاد وقفه بصری استفاده می‌شود

مثال ساده:

|  |
| --- |
| <p>این محتوای بالای خط است.</p>  <hr>  <p>این محتوای پایین خط است.</p> |

[semple](webdesign/html/tags/hr/smple.html)

استایل‌دهی با CSS:

می‌توانید ظاهر تگ `<hr>` را با CSS سفارشی کنید:

|  |
| --- |
| hr {  border: none;  height: 2px;  background-color: #333;  margin: 20px 0;  } |

[Style](webdesign/html/tags/hr/style.html)

کاربردهای رایج:

- جداسازی بخش‌های مختلف محتوا

- ایجاد وقفه بین مطالب

- تفکیک بصری بخش‌های مقاله یا پست

این تگ در HTML5 معنای معنایی بیشتری پیدا کرده و نشان‌دهنده "تغییر موضوع" در محتوا است.

## [معرفی لیست‌های ترتیبی و غیر ترتیبی و display آنها](webdesign/html/tags/lists)

لیست‌های غیرترتیبی (Unordered Lists) - ul

لیست‌هایی که ترتیب آیتم‌های آن مهم نیست و معمولاً با علامت‌هایی مانند نقطه، دایره یا مربع نمایش داده می‌شوند.

ساختار پایه:

|  |
| --- |
| <ul>  <li>آیتم اول</li>  <li>آیتم دوم</li>  <li>آیتم سوم</li>  </ul> |

[ul](webdesign/html/tags/lists/ul.html)

انواع استایل‌های نشانگر:

|  |
| --- |
| <ul style="list-style-type: disc;"> <!-- پیش‌فرض - نقطه سیاه -->  <li>آیتم با نقطه</li>  </ul>  <ul style="list-style-type: circle;"> <!-- دایره توخالی -->  <li>آیتم با دایره</li>  </ul>  <ul style="list-style-type: square;"> <!-- مربع -->  <li>آیتم با مربع</li>  </ul>  <ul style="list-style-type: none;"> <!-- بدون نشانگر -->  <li>آیتم بدون نشانگر</li>  </ul> |

[ul-style](webdesign/html/tags/lists/ul-style.html)

لیست‌های ترتیبی (Ordered Lists) - ol

لیست‌هایی که ترتیب آیتم‌ها مهم است و با اعداد، حروف یا اعداد رومی نمایش داده می‌شوند.

ساختار پایه:

|  |
| --- |
| <ol>  <li>آیتم اول</li>  <li>آیتم دوم</li>  <li>آیتم سوم</li>  </ol> |

[ol](webdesign/html/tags/lists/ol.html)

انواع استایل‌های شماره‌گذاری:

|  |
| --- |
| <ol type="1"> <!-- پیش‌فرض - اعداد (1, 2, 3) -->  <li>آیتم عددی</li>  </ol>  <ol type="A"> <!-- حروف بزرگ انگلیسی (A, B, C) -->  <li>آیتم حرف بزرگ</li>  </ol>  <ol type="a"> <!-- حروف کوچک انگلیسی (a, b, c) -->  <li>آیتم حرف کوچک</li>  </ol>  <ol type="I"> <!-- اعداد رومی بزرگ (I, II, III) -->  <li>آیتم رومی بزرگ</li>  </ol>  <ol type="i"> <!-- اعداد رومی کوچک (i, ii, iii) -->  <li>آیتم رومی کوچک</li>  </ol> |

[ol-style](webdesign/html/tags/lists/ol-style.html)

شروع از شماره خاص:

|  |
| --- |
| <ol start="5">  <li>این آیتم پنجم خواهد بود</li>  <li>ششم</li>  <li>هفتم</li>  </ol> |

[Ol Count](webdesign/html/tags/lists/ol-count.html)

لیست‌های تو در تو (Nested Lists):

می‌توان لیست‌ها را درون یکدیگر قرار داد:

|  |
| --- |
| <ul>  <li>میوه‌ها  <ul>  <li>سیب</li>  <li>پرتقال</li>  </ul>  </li>  <li>سبزی‌ها  <ol>  <li>هویج</li>  <li>کلم</li>  </ol>  </li>  </ul> |

[NestedLists](webdesign/html/tags/lists/NestedLists.html)

## آموزش EMMET

امت (Emmet) یک ابزار بسیار قدرتمند برای توسعه‌دهندگان وب است که با استفاده از اختصارات ساده (شبیه به CSS) به شما امکان می‌دهد کدهای HTML و CSS را با سرعت بسیار بالایی تولید کنید. این ابزار به صورت پیش‌فرض در ویرایشگرهای محبوبی مانند VS Code وجود دارد و کار کدنویسی را لذت‌بخش‌تر می‌کند.

برای شروع کار با امت، ابتدا باید ساختار اصلی یک سند HTML را سریعاً بسازید. سپس می‌توانید با یادگیری عملگرهای مختلف، المان‌های پیچیده‌تری ایجاد کنید.

شروع فوری با امت

|  |  |  |
| --- | --- | --- |
| کاربرد | عبارت Emmet | خروجی (پس از فشردن Tab) |
| ساختار پایه HTML | ! | یک سند HTML5 کامل با تگ‌های اصلی |
| تکرار المان | ul>li\*5 | یک لیست با 5 آیتم |
| افزایش عددی (ایندکس) | ul>li.item$\*3 | آیتم لیست با کلاس‌های `item1`, `item2`, `item3` |
| متن لورم ایپسوم | p>lorem20 | یک پاراگراف حاوی 20 کلمه متن تصادفی |

عملگرهای اصلی برای ساختاردهی

درک این عملگرها پایه یادگیری امت است:

> (فرزند مستقیم): المان سمت راست را داخل المان سمت چپ قرار می‌دهد. مثال: `nav>ul>li`

+ (خواهر یا هم‌سطح): المان‌ها را در یک سطح کنار هم قرار می‌دهد. مثال: `div+p+span`

^ (بالا رفتن از سطح): یک سطح در ساختار کد به بالا برمی‌گردد. مثال: `div>p^span` (اسپن دیگر فرزند دیو نیست).

() (گروه‌بندی): بخشی از عبارت را گروه‌بندی می‌کند تا اولویت‌ها را تغییر دهد. مثال: `(div>h2)+p`

افزودن کلاس، آی‌دی و ویژگی‌ها

امت یک syntax شبیه به CSS برای افزودن attributes دارد:

کلاس (.): برای افزودن کلاس از نقطه استفاده کنید. مثال: `div.container` یا `p.text.large` برای چندین کلاس.

آی‌دی (#): برای افزودن آی‌دی از نماد `#` استفاده کنید. مثال: `div#header`

ویژگی‌های سفارشی ([ ]): سایر ویژگی‌ها را داخل کروشه بنویسید. مثال: `img[src="image.jpg" alt="description"]`

متن ({ }): محتوای متنی را داخل آکولاد قرار دهید. مثال: `a{کلیک کنید}`

میانبرهای مفید CSS

امت فقط برای HTML نیست و برای CSS نیز میانبرهای بسیار سریعی دارد. معمولاً حروف اول propertyها به همراه مقدارشان است:

|  |
| --- |
| m10 → margin: 10px;  p20-30 → padding: 20px 30px;  w50p → width: 50%;  bgc#f00 → background-color: #ff0000;  df → display: flex;  tac → text-align: center; |

تمرین و تسلط

بهترین راه برای یادگیری امت، تمرین مستمر است. شروع کنید به تایپ کردن این اختصارات در ویرایشگر خود (مانند VS Code) و دیدن نتیجه جادویی آن. به مرور، سرعت شما در کدنویسی به طور چشمگیری افزایش خواهد یافت.

برگهٔ راهنمای Emmet به شما کمک میکند تا با استفاده از خلاصهنویسی (Abbreviations)، کدهای HTML و CSS را بسیار سریعتر بنویسید. در ادامه، دستورات اصلی بههمراه مثالهای کاربردی را برای شما آوردهام.

دستورات پایه Emmet

جدول زیر اصلیترین سینتکس Emmet را نشان میدهد که برای شروع کدنویسی سریع به آنها نیاز دارید:

|  |  |  |
| --- | --- | --- |
| دستور و نماد | کاربرد | مثال و خروجی |
| > | ایجاد عنصر فرزند | nav>ul>li → <nav><ul><li></li></ul></nav> |
| + | ایجاد عنصر همسطح | div+p+bq → <div></div><p></p><blockquote></blockquote> |
| \* | تکرارعنصر | <ul>li\*3 → <ul><li></li><li></li><li></li></ul> |
| ^ | بالا رفتن یک سطح | div>p>span+em^bq → <div><p><span></span><em></em></p><blockquote></blockquote></div> |
| () | گروه‌بندی دستورات | div>(header>ul>li\*2>a)+footer>p -> <div><header><ul><li><a href=""></a></li><li><a href="#"></a></li></ul></header><footer><p></p></footer></div> |
| . و # | افزودن کلاس و آی دی | form#search.wide → <form id="search" class="wide"></form> |
| [] | افزودن ویژگی (Attribute) | p[title="Hello"]→ <p title="Hello"></p> |
| {} | افزودن متن | a{Click me} → <a href="">Click me</a> |
| $ | شماره‌گذاری خودکار | ul>li.item$\*3 → <li class="item1"></li><li class="item2"></li><li class="item3"></li> |

ترفندها و مثالهای کاربردی

برای کارایی بیشتر، این ترفندها را در نظر بگیرید:

- تولید ساختار اولیه HTML: با تایپ ! و زدن کلید Tab، یک ساختار اولیه کامل HTML برای شما ساخته میشود.

- تولید متن بی‌معنای لورم ایپسوم: از دستور lorem استفاده کنید. برای کنترل تعداد کلمات، عدد مورد نظر را کنار آن بنویسید (مثلاً `lorem10`).

- شمارهگذاری پیشرفته: برای شمارهگذاری معکوس از `$@-` و برای شروع از عدد خاصی از `$@3` استفاده کنید (مثلاً `ul>li.item$@3\*5`).

- نامگذاری خودکار تگها: Emmet بهطور هوشمند تگ مناسب را انتخاب میکند. مثلاً نوشتن `.container` بهطور پیشفرش یک `<div>` میسازد، اما اگر داخل `<ul>` باشد، یک `<li>` تولید میکند.

راهاندازی و استفاده

- فعالسازی: Emmet به‌طور پیشفرض روی بسیاری از ویرایشگرهای مدرن مانند VS Code و Sublime Text نصب و فعال است. برای اطمینان، میتوانید افزونه مربوطه را از مارکتپلیس ویرایشگر خود نصب کنید.

- طریقه استفاده: پس از تایپ خلاصه دستور (مثلاً ul>li\*5)، کلید Tab را بفشارید تا کد کامل به‌طور خودکار تولید شود.

## [معرفی attribute های تگ های HTML](webdesign/html/Attributes)

فهرست مطالب

[مفهوم-attribute-ها](#_مفهوم_Attribute_ها)

[انواع-attribute-ها](#_انواع_Attribute_ها)

[attribute-های-جهانی](#_1._Attribute_های)

[attribute-های-خاص-تگ‌ها](#_2._Attribute_های)

نکات-مهم

### مفهوم Attribute ها

Attribute ها ویژگی‌های اضافی هستند که به تگ‌های HTML اضافه می‌شوند تا رفتار یا ظاهر آنها را تغییر دهند.

ساختار کلی:

|  |
| --- |
| <tag attribute="value">محتوا</tag> |

[base](webdesign/html/Attributes/base.html)

### انواع Attribute ها

### 1. Attribute های جهانی (Global Attributes)

این Attribute ها برای تمام تگ‌های HTML قابل استفاده هستند:

|  |
| --- |
| <!-- class - برای استایل‌دهی -->  <div class="container main-class">محتوا</div>  <!-- id - شناسه یکتا -->  <section id="main-section">بخش اصلی</section>  <!-- style - استایل inline -->  <p style="color: blue; font-size: 16px;">متن رنگی</p>  <!-- title - راهنمای ابزاری -->  <a href="#" title="این یک لینک است">لینک</a>  <!-- data-\* - ذخیره داده‌های سفارشی -->  <div data-user-id="123" data-role="admin">کاربر</div>  <!-- aria-\* - دسترسی‌پذیری -->  <button aria-label="بستن پنجره">×</button> |

[Global](webdesign/html/Attributes/global.html)

### 2. Attribute های خاص تگ‌ها

تگ‌های فرم:

|  |
| --- |
| <!-- input -->  <input type="text" name="username" placeholder="نام کاربری" required>  <!-- select -->  <select multiple size="3">  <option value="1">گزینه ۱</option>  <option value="2" selected>گزینه ۲</option>  </select>  <!-- textarea -->  <textarea rows="4" cols="50" maxlength="200"></textarea> |

[form](webdesign/html/Attributes/form.html)

تگ‌های رسانه:

|  |
| --- |
| <!-- img -->  <img src="image.jpg" alt="توضیح تصویر" width="300" height="200" loading="lazy">  <!-- video -->  <video controls autoplay muted loop>  <source src="video.mp4" type="video/mp4">  </video>  <!-- audio -->  <audio controls preload="metadata">  <source src="audio.mp3" type="audio/mpeg">  </audio> |

[Media](webdesign/html/Attributes/media.html)

تگ‌های لینک و ناوبری:

|  |
| --- |
| <!-- a -->      <a href="page.html" target="\_blank" rel="noopener noreferrer">لینک خارجی</a>      <!-- link -->      <link rel="stylesheet" href="style.css" media="screen and (min-width: 768px)"> |

[a](webdesign/html/Attributes/a.html)

Attribute های جهانی مهم

1. کلاس و آی‌دی:

|  |
| --- |
| <div class="box warning active" id="main-box">محتوا</div> |

1. داده‌های سفارشی (data-\*):

|  |
| --- |
| <div data-product-id="456" data-price="29.99" data-category="electronics">          محصول      </div>      <!-- دسترسی در JavaScript -->      <script>          const element = document.querySelector('div');          console.log(element.dataset.productId); // "456"      </script> |

1. ویژگی‌های دسترسی‌پذیری:

|  |
| --- |
| <button aria-expanded="false" aria-controls="menu">منو</button>      <div id="menu" aria-hidden="true">محتوای منو</div> |

[Global2](webdesign/html/Attributes/global2.html)

نکات مهم

1. نوشتن صحیح مقدار Attribute ها:

|  |
| --- |
| <!-- صحیح -->  <input type="text" value="مقدار پیش‌فرض">  <!-- اشتباه -->  <input type=text value=مقدار پیش‌فرض> |

1. Attribute های بولین:

|  |
| --- |
| <!-- این‌ها معادل هستند -->  <input disabled>  <input disabled="disabled">  <input disabled>  <!-- Attribute های بولین رایج -->  <video controls autoplay muted loop></video>  <select multiple></select>  <form novalidate></form> |

1. ترتیب Attribute ها:

|  |
| --- |
| <!-- توصیه شده: ترتیب منطقی -->  <img src="image.jpg" alt="توضیح" width="300" height="200" class="img-responsive">  <!-- بهتر است class و id در ابتدا یا انتها باشند -->  <div id="container" class="main-wrapper" data-role="page"> |

1. امنیت و عملکرد:

|  |
| --- |
| <!-- امن: جلوگیری از حملات tabnabbing -->  <a href="https://example.com" target="\_blank" rel="noopener noreferrer">  لینک امن  </a>  <! بهینه‌سازی بارگذاری تصاویر ---->  <img src="image.jpg" alt="تصویر" loading="lazy" decoding="async"> |

1. ویژگی‌های جدید HTML5:

|  |
| --- |
| <! ذخیره‌سازی محلی ---->  <input type="text" list="suggestions">  <datalist id="suggestions">  <option value="تهران">  <option value="مشهد">  </datalist>  <!-- اعتبارسنجی -->  <input type="email" required pattern="[a-z0-9.\_%+-]+@[a-z0-9.-]+\.[a-z]{2,}$"> |

بهترین روش‌ها

1. استفاده معنادار از id: هر id باید در صفحه یکتا باشد

2. نام‌گذاری مناسب class: استفاده از naming convention مانند BEM

3. اولویت استایل‌دهی با class: به جای style inline از class استفاده کنید

4. رعایت دسترسی‌پذیری: استفاده از aria-\* برای کاربران معلول

5. بهینه‌سازی سئو: استفاده صحیح از alt برای تصاویر

## [آموزش لیست‌های Nested](#_معرفی_لیست‌های_ترتیبی)

## معرفی پلاگین Live server در Vs code

پلاگین Live Server یک افزونه محبوب در VS Code است که با راه‌اندازی یک سرور توسعه محلی، قابلیت «بارگذاری مجدد زنده» (Live Reload) را برای شما فراهم می‌کند. این به آن معناست که با هر بار ذخیره‌سازی کدهای HTML، CSS یا JavaScript، صفحه مرورگر شما به طور خودکار به‌روزرسانی می‌شود و شما می‌توانید تغییرات را بلافاصله مشاهده کنید .

نحوه نصب و راه‌اندازی

برای نصب و شروع استفاده از Live Server این مراحل را دنبال کنید:

1. باز کردن پنل Extensions: در VS Code، روی آیکون Extensions در نوار کناری کلیک کنید یا از کلیدهای ترکیبی `Ctrl+Shift+X` (در ویندوز/لینوکس) یا `Cmd+Shift+X` (در مک) استفاده کنید .

2. جستجو و نصب: در نوار جستجو، عبارت "Live Server" را تایپ کنید. نتیجه مربوط بهRitwick Dey را پیدا کرده و روی دکمه Install کلیک کنید .

3. راه‌اندازی سرور: پس از نصب، برای شروع به کار یکی از روش‌های زیر را انتخاب کنید:

روی دکمه "Go Live" که در پایین سمت راست پنجره VS Code (نوار وضعیت) ظاهر می‌شود، کلیک کنید .

روی فایل HTML مورد نظر خود کلیک راست کرده و گزینه "Open with Live Server" را انتخاب کنید .

از کلیدهای ترکیبی `Alt+L, Alt+O` (در ویندوز) یا `Cmd+L, Cmd+O` (در مک) استفاده کنید .

پس از انجام این مراحل، صفحه پروژه شما در مرورگر پیش‌فرضتان (معمولاً روی آدرس `http://127.0.0.1:5500`) باز می‌شود .

ویژگی‌ها و قابلیت‌های کلیدی

این افزونه امکانات مفید دیگری نیز در اختیار شما قرار می‌دهد:

پشتیبانی از workspaceهای چندگانه: اگر روی چندین پروژه به طور همزمان کار می‌کنید، Live Server به خوبی از این حالت پشتیبانی می‌کند .

قابلیت سفارشی‌سازی: می‌توانید شماره پورت، مرورگر پیش‌فرض و ریشه سرور را از طریق تنظیمات VS Code (`File > Preferences > Settings`) تغییر دهید .

اتصال از طریق شبکه محلی (WLAN): با پیکربندی مناسب، می‌توانید پروژه خود را از دستگاه‌های دیگر (مانند تلفن همراه) در همان شبکه Wi-Fi مشاهده و تست کنید .

پشتیبانی از دیباگ کردن مرورگر کروم: این قابلیت امکان عیب‌یابی دقیق‌تر کدهایتان را فراهم می‌سازد .

راهنمای عیب‌یابی

اگر با مشکل مواجه شدید، این راهکارها می‌توانند کمک‌کننده باشند:

دکمه "Go Live" نمایش داده نمی‌شود: مطمئن شوید که یک فایل HTML در یک پوشه پروژه (Workspace) باز کرده‌اید و نه فقط یک فایل مجزا. در صورت نیاز، از منوی `File` گزینه `Open Folder` را انتخاب کنید .

سرور شروع به کار نمی‌کند: VS Code را یک بار restart کنید. همچنین بررسی کنید که افزونه‌های مشابه دیگری (مانند "Live Preview") که ممکن است تداخل ایجاد کنند، غیرفعال باشند .

مرورگر به‌طور خودکار به‌روزرسانی نمی‌شود: مطمئن شوید که پس از ایجاد تغییرات، فایل خود را ذخیره (`Ctrl+S`) کرده‌اید .

جمع‌بندی

پلاگین Live Server با حذف نیاز به بارگذاری مجدد دستی مرورگر، یکی از ضروری‌ترین ابزارها برای توسعه‌دهندگان front-end به حساب می‌آید. این افزونه روند توسعه را بسیار سریع‌تر و روان‌تر می‌کند .

## [معرفی تگ blockqoute و dislayآن](webdesign/html/tags/blockqoute)

تگ `<blockquote>` چیست؟

تگ `<blockquote>` یک عنصر سطح بلاک در HTML است که برای نقل قول کردن بخش‌های بزرگی از متن از یک منبع دیگر استفاده می‌شود. این تگ به مرورگرها، موتورهای جستجو و خوانندگان صفحه می‌فهماند که محتوای داخل آن یک نقل قول است.

ویژگی‌های اصلی و ظاهری

به طور پیش‌فرض، مرورگرها معمولاً محتوای داخل `<blockquote>` را به صورت زیر نمایش می‌دهند:

تورفتگی (Indentation) از هر دو سمت راست و چپ.

گاهی اوقات یک حاشیه (Margin) در بالا و پایین به آن اضافه می‌کنند.

ممکن است از ایتالیک (Italic) نیز استفاده کنند.

البته شما می‌توانید با CSS این ظاهر پیش‌فرض را کاملاً تغییر دهید.

نحوه استفاده (Syntax)

|  |
| --- |
| <blockquote>  اینجا متن نقل قول شده قرار می گیرد. این می‌تواند یک پاراگراف، چندین پاراگراف یا هر محتوای دیگری باشد.  </blockquote> |

[Syntax](webdesign/html/tags/blockqoute/Syntax.html)

ویژگی مهم: `cite`

تگ `<blockquote>` یک ویژگی اختیاری به نام `cite` دارد. از این ویژگی برای اشاره به آدرس منبع (URL) نقل قول استفاده می‌کنید. این آدرس برای موتورهای جستجو و ابزارهای خاص بسیار مفید است، اما در ظاهر صفحه به کاربر نمایش داده نمی‌شود.

|  |
| --- |
| <blockquote cite="https://example.com/source-page">  این یک نقل قول بسیار معروف از یک وب‌سایت است.  </blockquote> |

[cite](webdesign/html/tags/blockqoute/cite.html)

1. نقل قول بلند: از `<blockquote>` برای نقل قول‌های چندخطی و نسبتاً بلند استفاده کنید. برای نقل قول‌های کوتاه و درون خطی، از تگ `<q>` استفاده می‌شود.

2. منبع‌گذاری: همیشه سعی کنید منبع نقل قول را مشخص کنید. بهترین روش این است که از ویژگی `cite` برای آدرس منبع و از تگ `<cite>` برای ذکر نام منبع (مثلاً نام کتاب یا نویسنده) استفاده کنید. معمولاً نام منبع را بعد از بسته شدن تگ `<blockquote>` می‌آورند.

مثال کامل و استاندارد

|  |
| --- |
| <blockquote cite="https://fa.wikipedia.org/wiki/%D8%B3%D8%B9%D8%AF%DB%8C">          <p>بنی آدم اعضای یک پیکرند که در آفرینش ز یک گوهرند<br>              چو عضوی به درد آورَد روزگار، دگر عضوها را نمانَد قرار</p>      </blockquote>      <p>— <cite>سعدی شیرازی</cite></p> |

[full](webdesign/html/tags/blockqoute/full.html)

## تگ Pre و Display آن

تگ `<pre>` چیست؟

تگ `<pre>` مخفف Preformatted Text به معنای "متن از پیش قالب‌بندی شده" است. این تگ برای نمایش متنی استفاده می‌شود که فضاها، تو رفتگی‌ها و شکستن خطوط در آن باید دقیقاً همانطور که در کد HTML نوشته شده است، حفظ شود.

مشکل اصلی که `<pre>` حل می‌کند

به طور معمول، HTML فضاهای خالی متوالی (Spaceهای پشت سر هم) و Enterها را نادیده می‌گیرد. برای مثال:

|  |
| --- |
| <p>این یک متن  معمولی است.  فضاها و خطوط جدید نادیده گرفته می‌شوند.  </p> |

[اما با استفاده از `<pre>`:](webdesign/html/tags/pre)

|  |
| --- |
| <pre>این    یک    متن              از پیش قالب‌بندی شده است.              فضاها و خطوط جدید حفظ می‌شوند.      </pre> |

[index](webdesign/html/tags/pre/index.html)

ویژگی‌های ظاهری پیش‌فرض

مرورگرها معمولاً محتوای داخل `<pre>` را با ویژگی‌های زیر نمایش می‌دهند:

فونت مونواسپیس (Monospace): مانند `Courier New` - که در آن همه کاراکترها عرض یکسانی دارند.

حفظ فضای سفید: تمام Spaceها، Tabها و خطوط جدید دقیقاً نمایش داده می‌شوند.

نمایش به صورت بلوک: یک عنصر سطح بلاک است.

کاربردهای اصلی تگ `<pre>`

1. نمایش کدهای کامپیوتری (رایج‌ترین کاربرد)

2. نمایش شعر و متون ادبی که فرمت خاصی دارند

3. نمایش خروجی برنامه‌ها (Command Line Output)

4. ایجاد ASCII Art (نقاشی با کاراکترها)

مثال‌های کاربردی

مثال ۱: نمایش کد برنامه‌نویسی

|  |
| --- |
| <pre>  function helloWorld() {  console.log("Hello World!");  for (let i = 0; i < 5; i++) {  console.log("Number: " + i);  }  }  </pre> |

[programing](webdesign/html/tags/pre/programing.html)

مثال ۲: نمایش شعر با فرمت اصلی

|  |
| --- |
| <pre>  غزل شماره ۱ سعدی  بنى آدم اعضای یک پیکرند  که در آفرینش ز یک گوهرند  چو عضوى به درد آورد روزگار  دگر عضوها را نماند قرار  </pre> |

[poem](webdesign/html/tags/pre/poem.html)

ترکیب با تگ `<code>`

برای نمایش کد، بهترین روش این است که تگ `<pre>` را با تگ `<code>` ترکیب کنید. تگ `<code>` معنای معنایی (Semantic) دارد و به مرورگر و موتورهای جستجو می‌فهماند که این محتوا "کد کامپیوتری" است.

|  |
| --- |
| <pre><code>  function calculate(a, b) {  return a + b;  }  let result = calculate(5, 3);  console.log(result); // Output: 8  </code></pre> |

[code](webdesign/html/tags/pre/code.html)

## معرفی Webstorm و نحوه نصب و کرک آن

معرفی پلاگین های Webstorm و استفاده از

آن ها (atom material icons, material theme ui lite, rainbow brackets )

WebStorm یک محیط توسعه یکپارچه (IDE) تخصصی و قدرتمند برای برنامهنویسی وب است که توسط شرکت JetBrains توسعه داده شده است. این نرمافزار از زبانها و فناوریهای مدرن وب مانند JavaScript، TypeScript، HTML، CSS و همچنین فریمورکهای محبوبی مانند React، Angular، Vue.js و Node.js پشتیبانی کامل میکند.

در ادامه به معرفی امکانات، روش نصب و پلاگینهای محبوب آن میپردازم.

امکانات و ویژگی های کلیدی WebStorm

WebStorm با بهرهگیری از هوش مصنوعی، ابزارهای پیشرفتهای را در اختیار توسعهدهندگان قرار میدهد که روند کدنویسی را کارآمدتر میکند.

ویرایشگر هوشمند: این ویرایشگر با تجزیه و تحلیل پروژه، بهترین پیشنهادات را برای تکمیل خودکار کد (autocomplete) ارائه میدهد و خطاها را بهصورت زنده و در حین تایپ کردن تشخیص میدهد.

- اشکالزدایی و تست واحد: WebStorm دارای دیباگر داخلی برای کدهای سمت کلاینت و برنامههای Node.js است. همچنین با ابزارهای تستی مانند Karma و Mocha ادغام میشود.

- یکپارچگی با ابزارهای توسعه: این نرمافزار از سیستمهای کنترل نسخه مانند Git پشتیبانی میکند و امکان اجرای وظایف Grunt، Gulp یا npm را مستقیماً از داخل محیط IDE فراهم میسازد.

- پشتیبانی از فریمورکهای مدرن: این IDE کمکهای ویژهای برای توسعه با فریمورکهایی مانند Angular، React، و Meteor ارائه میدهد.

روش نصب WebStorm

برای نصب WebStorm، بهتر است آخرین نسخه رسمی آن را مستقیماً از وبسایت JetBrains دانلود کنید. توجه داشته باشید که استفاده از نسخه‌های کرک شده که در برخی سایتهای غیررسمی ارائه می‌شوند، میتواند همراه با خطرات امنیتی مانند نصب بدافزار یا عدم پشتیبانی باشد.

شرکت JetBrains برای دانشجویان و توسعهدهندگان فعال، لایسنس رایگان یا دورههای trial در نظر گرفته است که استفاده از این روشها به مراتب ایمنتر و مطمئنتر است.

معرفی پلاگین های محبوب

نصب پلاگین در WebStorm بسیار ساده است. از منوی File > Settings (در ویندوز/لینوکس) یا WebStorm > Preferences (در مک) به بخش Plugins بروید. سپس از طریق زبانه Marketplace میتوانید پلاگینهای زیر را جستجو و نصب کنید.

در جدول below سه پلاگین محبوب درخواستی شما معرفی شده است:

|  |  |  |
| --- | --- | --- |
| نام پلاگین | کاربرد اصلی | ویژگی های کلیدی |
| Material Theme UI | تغییر ظاهر کلی IDE به سبک Material Design | ارائه چندین تم مختلف (دارک و روشن)، تغییر شکل دکمهها و تبها، قابل تنظیم بودن عمق رنگها و المانها. |
| Atom Material Icons | اضافه کردن آیکونهای زیبا و معنادار به کنار نام فایلها و پوشهها در ساختار پروژه | کمک به شناسایی سریعتر نوع فایل (مثلاً فایلهای کامپوننت، استایل، یا تصویر)، کاهش خستگی چشم. |
| Rainbow Brackets | رنگآمیزی پرانتزها، کروشهها و آکولادها با رنگهای مختلف بهصورت جفت | دنبال کردن پرانتزهای تودرتو در کد بسیار آسانتر میشود، قابلیت شخصیسازی رنگها و غیرفعال کردن آن برای زبانهای خاص |

جمع بندی

WebStorm با ویژگیهای قدرتمند خود میتواند تجربه توسعه وب را بسیار لذتبخشتر و حرفهایتر کند. استفاده از پلاگینهای معرفیشده، بهویژه Material Theme UI و Rainbow Brackets، میتواند این تجربه را از نظر بصری و کارایی further further ارتقا دهد.

## بررسی تفاوت‌های HTML4 و HTML5

تفاوت‌های بین HTML4 و HTML5 بسیار گسترده و اساسی است. HTML5 یک انقلاب در توسعه وب به حساب می‌آید که نه تنها تگ‌های جدیدی اضافه کرد، بلکه رویکرد توسعه وب را به کلی تغییر داد.

تفاوت‌های ساختاری و فلسفی

HTML4:

- بیشتر بر روی نمایش محتوا متمرکز بود

- نیاز به پلاگین‌های خارجی (مثل Flash) برای مالتی‌مدیا

- ساختار سند کمتر معنایی (semantic)

HTML5:

- روی معنای محتوا تأکید دارد

- قابلیت‌های مالتی‌مدیا داخلی

- طراحی شده برای برنامه‌های تحت وب پیچیده

جدول مقایسه‌ای تفاوت‌های کلیدی

|  |  |  |
| --- | --- | --- |
| ویژگی | HTML4 | HTML5 |
| تگ‌های معنایی | محدود (div, span) | گسترده (header, footer, section, article) |
| پشتیبانی مالتی‌مدیا | نیاز به Flash/Silverlight | audio, video تگ‌های داخلی |
| ذخیره‌سازی داده | Cookie فقط | LocalStorage, SessionStorage |
| فرم‌ها | input typeهای محدود | email, url, date, range, color و... |
| گرافیک | نیاز به پلاگین | Canvas, SVG داخلی |
| APIها | بسیار محدود | Geolocation, Drag&Drop, Web Workers و... |

مهم‌ترین ویژگی‌های جدید HTML5

1. تگ‌های معنایی (Semantic Tags)

|  |
| --- |
| <!-- HTML4 -->  <div id="header">  <div id="nav">  <div class="article">  <!-- HTML5 -->  <header>  <nav>  <article>  <section>  <aside>  <footer> |

1. تگ‌های مالتی‌مدیا

|  |
| --- |
| <video controls width="250">  <source src="video.mp4" type="video/mp4">  </video>  <audio controls>  <source src="audio.mp3" type="audio/mpeg">  </audio> |

1. کنترل‌های پیشرفته فرم

|  |
| --- |
| <input type="email" required>  <input type="url">  <input type="date">  <input type="color">  <input type="range" min="0" max="100"> |

1. Canvas برای گرافیک

|  |
| --- |
| <canvas id="myCanvas" width="200" height="100"></canvas> |

تفاوت‌های فنی مهم

Doctype ساده‌تر

|  |
| --- |
| <!-- HTML4 -->  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  "http://www.w3.org/TR/html4/loose.dtd">  <!-- HTML5 -->  <!DOCTYPE html> |

کدگذاری کاراکتر ساده‌تر

|  |
| --- |
| <!-- HTML4 -->  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <!-- HTML5 -->  <meta charset="UTF-8"> |

مزایای اصلی HTML5

1. سازگاری بهتر با موبایل

- طراحی ریسپانسیو آسان‌تر

- عملکرد بهتر در دستگاه‌های لمسی

2. عملکرد بالاتر

- نیاز کمتر به پلاگین‌های خارجی

- بارگذاری سریع‌تر صفحات

3. تجربه کاربری بهتر

- برنامه‌های تحت وب غنی‌تر (Rich Web Applications)

- قابلیت کار آفلاین

4. استانداردهای باز

وابستگی کمتر به شرکت‌های خاص

جامعه توسعه‌دهنده بزرگ

چالش‌های مهاجرت از HTML4 به HTML5

سازگاری با مرورگرهای قدیمی

- نیاز به polyfill برای برخی ویژگی‌ها

- تست گسترده در مرورگرهای مختلف

یادگیری مفاهیم جدید

- APIهای جدید JavaScript

- مفاهیم برنامه‌نویسی غیرهمزمان (Async)

🚀 نتیجه‌گیری

HTML5 نه تنها یک بروزرسانی ساده، بلکه یک تحول اساسی در توسعه وب بود. امروزه HTML5 استاندارد اصلی توسعه وب است و تمام پروژه‌های جدید باید با این استاندارد توسعه داده شوند.

مهم‌ترین توصیه: برای پروژه‌های جدید حتماً از HTML5 استفاده کنید و اگر پروژه‌های قدیمی HTML4 دارید، به تدریج آن‌ها را به HTML5 مهاجرت دهید.

## معرفی Doctype

حتماً! در ادامه یک معرفی کامل و ساختاریافته از `Doctype` در HTML ارائه میدهم.

معرفی Doctype (نوع سند)

Doctype یا Document Type Declaration یک دستورالعمل ضروری در ابتدای سندهای HTML و XHTML است. وظیفه اصلی آن اعلام نوع سند و نسخه زبان نشانه‌گذاری (مانند HTML5) به مرورگر وب است.

هدف اصلی Doctype: فعال کردن حالت استاندارد (Standards Mode)

مهمترین وظیفه `Doctype` این است که به مرورگر بگوید سند شما باید بر اساس کدام مجموعه از قوانین (استانداردها) نمایش داده شود.

بدون Doctype: مرورگر وارد حالت شگفت‌انگیز (Quirks Mode) می‌شود. در این حالت، مرورگر برای سازگاری با صفحات وب بسیار قدیمی (مربوط به دهه 90 و اوایل 2000) رفتار می‌کند. این باعث می‌شود استایل‌ها و لایه‌بندی (Layout) صفحه شما به طور نادرست و غیرقابل پیش‌بینی نمایش داده شود.

با Doctype صحیح: مرورگر وارد حالت استاندارد (Standards Mode) می‌شود. در این حالت، مرورگر سند HTML و CSS شما را مطابق با استانداردهای مدرن و مشخص‌شده پردازش و رندر می‌کند. این امر تضمین می‌کند که صفحه شما در تمام مرورگرهای مدرن به یک شکل و به درستی نمایش داده می‌شود.

نحوه نوشتن Doctype

نحوه نوشتن `Doctype` در نسخه‌های مختلف HTML متفاوت بود و بسیار طولانی و پیچیده بود، اما در HTML5 این کار به شدت ساده شده است.

Doctype در HTML5

در HTML5، تنها کاری که باید بکنید این است که در اولین خط سند HTML خود بنویسید:

|  |
| --- |
| <!DOCTYPE html> |

مثال یک سند کامل HTML5:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>عنوان صفحه</title>  </head>  <body>  <h1>سلام دنیا!</h1>  <p>این یک پاراگراف است.</p>  </body>  </html> |

ویژگی‌های Doctype در HTML5:

بسیار کوتاه و ساده است.

به حروف بزرگ و کوچک حساس نیست (Case-insensitive)، اما معمولاً با حروف بزرگ نوشته می‌شود (`<!DOCTYPE html>`).

تنها کاری که می‌کند فعال کردن حالت استاندارد در مرورگر است.

نمونه‌هایی از Doctype در نسخه‌های قدیمی HTML و XHTML

برای درک بهتر سادگی HTML5، در زیر `Doctype`های پیچیده نسخه‌های قدیمی را می‌بینید:

HTML 4.01 Strict

|  |
| --- |
| <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"> |

XHTML 1.0 Transitional

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> |

همانطور که می‌بینید، این تعاریف شامل ارجاع به یک فایل DTD (تعریف نوع سند) خارجی بودند که مشخصات دقیق تگ‌های مجاز را تعریف می‌کرد. خوشبختانه در HTML5 دیگر نیازی به این کار نیست.

نکات کلیدی و خلاصه

1. اجباری است: همیشه `<!DOCTYPE html>` را در خط اول سند HTML خود قرار دهید.

2. فقط یک بار: فقط یک `Doctype` برای هر صفحه مجاز است.

3. هدف اصلی: فعال کردن حالت استاندارد (Standards Mode) در مرورگر برای رندر صحیح و یکسان صفحه.

4. ساده‌سازی در HTML5: نوشتن آن در HTML5 بسیار آسان شده و نیازی به یادآوری آدرس‌های طولانی نیست.

5. این یک تگ HTML نیست: `Doctype` یک تگ معمولی HTML مانند `<html>` یا `<body>` نیست؛ بلکه یک اعلامیه (Declaration) برای مرورگر است.

با رعایت این نکته ساده اما حیاتی، می‌توانید از نمایش صحیح و یکسان وب‌سایت خود در مرورگرهای مختلف اطمینان حاصل کنید.

## [Semantic](#_بررسی_مهفوم_Semantic)

## معرفی تگ main و Display آن

حتماً! در ادامه یک معرفی کامل و ساختاریافته از تگ `<main>` در HTML ارائه می‌دهم.

معرفی تگ `<main>`

تگ `<main>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای تعیین محتوای اصلی و محوری یک سند HTML استفاده می‌شود.

هدف اصلی تگ `<main>`

محتوایی که درون این تگ قرار می‌گیرد، باید مستقیماً به محتوای مرکزی صفحه مرتبط باشد و محتوای تکراری (مانند نوار کناری، هدر، فوتر، منو) نباید در آن قرار بگیرد.

ویژگی‌های کلیدی تگ `<main>`

1. عنصر معنایی (Semantic):

- به موتورهای جستجو و مرورگرها کمک می‌کند ساختار صفحه را بهتر درک کنند.

- برای دسترسی‌پذیری (Accessibility) بسیار مهم است و به صفحه‌خوان‌ها (Screen Readers) کمک می‌کند تا محتوای اصلی را سریع‌تر پیدا کنند.

2. یگانه در صفحه (Unique):

- تگ `<main>` باید در هر صفحه فقط یک بار استفاده شود.

- برخلاف تگ‌هایی مانند `<header>` یا `<footer>` که می‌توانند چندین بار در یک صفحه ظاهر شوند.

3. والد مستقیم `<body>`:

- تگ `<main>` باید مستقیماً درون تگ `<body>` قرار گیرد.

- نباید درون تگ‌های دیگر مانند `<article>`, `<aside>`, `<div>`, `<footer>`, `<header>`, یا `<nav>` قرار بگیرد.

نحوه استفاده از تگ `<main>`

مثال صحیح:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>وبلاگ من</title>  </head>  <body>  <header>  <h1>سایت آموزشی</h1>  <nav></nav>  </header>  <main>  <h1>عنوان مقاله اصلی</h1>  <p>این پاراگراف محتوای اصلی صفحه است.</p>  <article>  <h2>مقالات مرتبط</h2>  <p>محتوای مقاله...</p>  </article>  </main>  <aside>  <h2>نوار کناری</h2>  <p>تبلیغات یا لینک‌های مرتبط</p>  </aside>  <footer>  <p>کپی‌رایت © 2023</p>  </footer>  </body>  </html> |

مثال نادرست (استفاده دو بار از `<main>`):

|  |
| --- |
| <body>  <main>  <h1>محتوای اصلی</h1>  </main>  <main> <!-- ❌ نادرست! -->  <h1>محتوای اصلی دیگر</h1>  </main>  </body> |

تفاوت `<main>` با تگ‌های مشابه

مزایای استفاده از تگ `<main>`

1. سئو (SEO):

- به موتورهای جستجو کمک می‌کند محتوای اصلی صفحه را بهتر تشخیص دهند

2. دسترسی‌پذیری (Accessibility):

- صفحه‌خوان‌ها می‌توانند کاربر را مستقیماً به محتوای اصلی هدایت کنند

- تجربه کاربری بهتری برای افراد دارای معلولیت فراهم می‌کند

3. خوانایی کد:

- کد HTML تمیزتر و قابل درک‌تری ایجاد می‌کند

- نگهداری و توسعه صفحه را آسان‌تر می‌کند

پشتیبانی مرورگرها

تگ `<main>` در تمام مرورگرهای مدرن به طور کامل پشتیبانی می‌شود:

## تگ Header و display آن

معرفی تگ `<header>`

تگ `<header>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای تعیین سرصفحه یا بخش introductory یک صفحه یا بخش استفاده می‌شود.

هدف اصلی تگ `<header>`

این تگ معمولاً شامل محتوای مقدماتی، عناوین، لوگو، ناوبری (Navigation) و دیگر اطلاعات مربوط به معرفی آن بخش یا صفحه است.

ویژگی‌های کلیدی تگ `<header>`

1. عنصر معنایی (Semantic):

- ساختار صفحه را برای موتورهای جستجو و مرورگرها شفاف می‌کند

- به صفحه‌خوان‌ها (Screen Readers) در درک بهتر ساختار صفحه کمک می‌کند

2. قابل استفاده چندباره:

- برخلاف تگ `<main>`، می‌توانید چندین تگ `<header>` در یک صفحه داشته باشید

- هر تگ `<header>` مخصوص بخش parent خود است

3. والدهای مجاز:

- می‌تواند در تگ‌های مختلفی قرار گیرد:

- مستقیماً در `<body>` (برای هدر اصلی صفحه)

- درون `<article>`

- درون `<section>`

- درون دیگر تگ‌های block-level

نحوه استفاده از تگ `<header>`

مثال ۱: هدر اصلی صفحه

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>فروشگاه اینترنتی</title>  </head>  <body>  <header>  <div class="logo">  <img src="logo.png" alt="لوگو فروشگاه">  <h1>فروشگاه اینترنتی ما</h1>  </div>  <nav>  <ul>  <li><a href="/">خانه</a></li>  <li><a href="/products">محصولات</a></li>  <li><a href="/about">درباره ما</a></li>  <li><a href="/contact">تماس با ما</a></li>  </ul>  </nav>  </header>  <main>  <h2>محصولات ویژه</h2>  <!-- محتوای اصلی -->  </main>  <footer>  <p>تمامی حقوق محفوظ است</p>  </footer>  </body>  </html> |

مثال ۲: هدر برای مقاله

|  |
| --- |
| <article>  <header>  <h2>عنوان مقاله</h2>  <p>نویسنده: <span>محمد رضایی</span></p>  <p>تاریخ انتشار: <time datetime="2023-10-01">مهر ۱۴۰۲</time></p>  </header>    <div class="content">  <p>این محتوای اصلی مقاله است...</p>  </div>  </article> |

مثال ۳: هدر برای بخش‌های مختلف

|  |
| --- |
| <section>  <header>  <h3>بخش نظرات</h3>  <p>نظرات کاربران درباره این محصول</p>  </header>    <div class="comments">  <!-- لیست نظرات -->  </div>  </section> |

محتوای مجاز در `<header>`

می‌توانید این عناصر را در تگ `<header>` قرار دهید:

- عنوان‌ها (`<h1>` تا `<h6>`)

- لوگو یا تصویر

- ناوبری (`<nav>`)

- فرم جستجو

- اطلاعات نویسنده یا تاریخ

- جدول محتوا (برای مقالات طولانی)

نکات مهم در استفاده از `<header>`

1. اجباری نیست: استفاده از `<header>` اجباری نیست، اما توصیه می‌شود

2. جایگزین `<div>`: به جای `<div class="header">` از `<header>` استفاده کنید

3. موقعیت‌یابی: معمولاً در بالای صفحه یا بخش قرار می‌گیرد، اما از نظر HTML اجباری نیست

4. تودرتو نباشد: تگ‌های `<header>` نباید درون یکدیگر قرار گیرند

5. پشت سر هم نباشد: تگ‌های `<header>` نباید مستقیماً پس از یکدیگر بیایند

پشتیبانی مرورگرها

تگ `<header>` در تمام مرورگرهای مدرن به طور کامل پشتیبانی می‌شود.

مزایای استفاده از تگ `<header>`

1. سئو (SEO):

- به موتورهای جستجو در درک ساختار صفحه کمک می‌کند

- محتوای مهم را برجسته می‌کند

2. دسترسی‌پذیری (Accessibility):

- صفحه‌خوان‌ها می‌توانند کاربر را به بخش‌های مختلف هدایت کنند

- navigation بین بخش‌های صفحه آسان‌تر می‌شود

3. خوانایی کد:

- کد HTML تمیزتر و معنایی‌تر می‌شود

- نگهداری و توسعه پروژه آسان‌تر می‌شود

## تگ Footer و Display آن

معرفی تگ `<footer>`

تگ `<footer>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای تعیین پاصفحه یا بخش پایانی یک صفحه یا بخش استفاده می‌شود.

هدف اصلی تگ `<footer>`

این تگ معمولاً شامل اطلاعات تکمیلی، حقوق قانونی، لینک‌های مرتبط، اطلاعات تماس و دیگر محتوای مربوط به پایان بخش یا صفحه است.

ویژگی‌های کلیدی تگ `<footer>`

1. عنصر معنایی (Semantic):

- ساختار صفحه را برای موتورهای جستجو و مرورگرها شفاف می‌کند

- به صفحه‌خوان‌ها (Screen Readers) در درک بهتر ساختار صفحه کمک می‌کند

2. قابل استفاده چندباره:

- می‌توانید چندین تگ `<footer>` در یک صفحه داشته باشید

- هر تگ `<footer>` مخصوص بخش parent خود است

3. والدهای مجاز:

- می‌تواند در تگ‌های مختلفی قرار گیرد:

- مستقیماً در `<body>` (برای فوتر اصلی صفحه)

- درون `<article>`

- درون `<section>`

- درون `<nav>`

- درون دیگر تگ‌های block-level

نحوه استفاده از تگ `<footer>`

مثال ۱: فوتر اصلی صفحه

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>فروشگاه اینترنتی</title>  </head>  <body>  <header>  <!-- هدر صفحه -->  </header>  <main>  <!-- محتوای اصلی -->  </main>  <footer>  <div class="footer-content">  <div class="footer-section">  <h3>درباره ما</h3>  <p>ما یک فروشگاه اینترنتی معتبر با سال‌ها تجربه هستیم.</p>  </div>    <div class="footer-section">  <h3>لینک‌های سریع</h3>  <ul>  <li><a href="/about">درباره ما</a></li>  <li><a href="/contact">تماس با ما</a></li>  <li><a href="/privacy">حریم خصوصی</a></li>  </ul>  </div>    <div class="footer-section">  <h3>تماس با ما</h3>  <p>ایمیل: info@example.com</p>  <p>تلفن: ۰۲۱-۱۲۳۴۵۶۷۸</p>  </div>  </div>    <div class="footer-bottom">  <p>&copy; ۱۴۰۲ - تمامی حقوق برای فروشگاه اینترنتی ما محفوظ است.</p>  </div>  </footer>  </body>  </html> |

مثال ۲: فوتر برای مقاله

|  |
| --- |
| <article>  <header>  <h2>عنوان مقاله</h2>  <p>نویسنده: محمد رضایی</p>  </header>    <div class="content">  <p>این محتوای اصلی مقاله است...</p>  </div>    <footer>  <p>تاریخ انتشار: <time datetime="2023-10-01">مهر ۱۴۰۲</time></p>  <div class="article-tags">  <span>برچسب‌ها:</span>  <a href="/tag/html">HTML</a>  <a href="/tag/css">CSS</a>  <a href="/tag/web">وب</a>  </div>  </footer>  </article> |

مثال ۳: فوتر برای بخش‌های مختلف

|  |
| --- |
| <section>  <h3>نظرات کاربران</h3>    <div class="comment">  <p>این یک نظر نمونه است.</p>  <footer>  <span>نویسنده: علی محمدی</span>  <span>تاریخ: ۱۴۰۲/۰۷/۱۵</span>  </footer>  </div>    <div class="comment">  <p>این نظر دیگری است.</p>  <footer>  <span>نویسنده: زهرا کریمی</span>  <span>تاریخ: ۱۴۰۲/۰۷/۱۶</span>  </footer>  </div>  </section> |

محتوای مجاز در `<footer>`

می‌توانید این عناصر را در تگ `<footer>` قرار دهید:

- اطلاعات تماس

- لینک‌های مرتبط

- اطلاعات حقوق قانونی (کپی‌رایت)

- اطلاعات نویسنده (در مقالات)

- نقشه سایت

- دکمه‌های اشتراک‌گذاری در شبکه‌های اجتماعی

- فرم ثبت‌نامه الکترونیکی

- اطلاعات تاریخ و زمان

نکات مهم در استفاده از `<footer>`

1. اجباری نیست: استفاده از `<footer>` اجباری نیست، اما توصیه می‌شود

2. جایگزین `<div>`: به جای `<div class="footer">` از `<footer>` استفاده کنید

3. موقعیت‌یابی: معمولاً در پایین صفحه یا بخش قرار می‌گیرد، اما از نظر HTML اجباری نیست

4. اطلاعات تماس: برای اطلاعات تماس از تگ `<address>` درون `<footer>` استفاده کنید

5. تودرتو نباشد: تگ‌های `<footer>` نباید درون یکدیگر قرار گیرند

مثال با تگ `<address>`:

|  |
| --- |
| <footer>  <address>  <p>آدرس: تهران، خیابان ولیعصر، پلاک ۱۲۳</p>  <p>تلفن: <a href="tel:+982112345678">۰۲۱-۱۲۳۴۵۶۷۸</a></p>  <p>ایمیل: <a href="mailto:info@example.com">info@example.com</a></p>  </address>  </footer> |

تگ `<footer>` در تمام مرورگرهای مدرن به طور کامل پشتیبانی می‌شود.

## تگ Section و Display آن

معرفی تگ `<section>`

تگ `<section>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای گروه‌بندی محتوای مرتبط به هم استفاده می‌شود. هر section باید یک موضوع مستقل و مشخص داشته باشد.

هدف اصلی تگ `<section>`

این تگ برای تقسیم‌بندی منطقی محتوای صفحه به بخش‌های مجزا که هر کدام مفهوم مستقل دارند استفاده می‌شود.

ویژگی‌های کلیدی تگ `<section>`

1. عنصر معنایی (Semantic):

- ساختار صفحه را برای موتورهای جستجو و مرورگرها شفاف می‌کند

- به سازماندهی محتوای مرتبط کمک می‌کند

2. قابل استفاده چندباره:

- می‌توانید چندین تگ `<section>` در یک صفحه داشته باشید

- هر section باید یک عنوان (`<h1>`-`<h6>`) داشته باشد

3. برای محتوای مستقل:

- هر section باید محتوای مستقل و قابل تفکیک از سایر sections داشته باشد

نحوه استفاده از تگ `<section>`

مثال ۱: صفحه اصلی با sections مختلف

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>شرکت طراحی وب</title>  </head>  <body>  <header>  <h1>شرکت طراحی وب ما</h1>  <nav><!-- منوی ناوبری --></nav>  </header>  <main>  <section id="intro">  <h2>درباره ما</h2>  <p>ما یک شرکت طراحی وب با ۱۰ سال تجربه هستیم...</p>  </section>  <section id="services">  <h2>خدمات ما</h2>  <div class="service">  <h3>طراحی وبسایت</h3>  <p>طراحی وبسایت‌های مدرن و ریسپانسیو</p>  </div>  <div class="service">  <h3>سئو و بهینه‌سازی</h3>  <p>بهبود رتبه در موتورهای جستجو</p>  </div>  </section>  <section id="portfolio">  <h2>نمونه کارها</h2>  <article>  <h3>پروژه ۱</h3>  <p>توضیحات پروژه اول...</p>  </article>  <article>  <h3>پروژه ۲</h3>  <p>توضیحات پروژه دوم...</p>  </article>  </section>  </main>  <footer>  <!-- فوتر صفحه -->  </footer>  </body>  </html> |

مثال ۲: بخش‌های یک مقاله طولانی

|  |
| --- |
| <article>  <header>  <h1>آموزش کامل HTML5</h1>  <p>نویسنده: محمد رضایی</p>  </header>  <section id="introduction">  <h2>مقدمه</h2>  <p>HTML5 آخرین نسخه از زبان نشانه‌گذاری ابرمتن است...</p>  </section>  <section id="semantic-elements">  <h2>تگ‌های معنایی</h2>  <p>تگ‌های معنایی به درک بهتر ساختار صفحه کمک می‌کنند...</p>  <ul>  <li>تگ section</li>  <li>تگ article</li>  <li>تگ header</li>  </ul>  </section>  <section id="multimedia">  <h2>المان‌های چندرسانه‌ای</h2>  <p>HTML5 قابلیت‌های چندرسانه‌ای قدرتمندی ارائه می‌دهد...</p>  </section>  </article> |

مثال ۳: بخش‌های مختلف یک صفحه محصول

|  |
| --- |
| <main>  <section class="product-gallery">  <h2 class="visually-hidden">گالری محصول</h2>  <!-- تصاویر محصول -->  </section>  <section class="product-info">  <h1>لپ‌تپ ایسوس ROG</h1>  <p>لپ‌تپ گیمینگ با پردازنده Core i7</p>  <div class="price">۱۵,۰۰۰,۰۰۰ تومان</div>  </section>  <section class="product-specs">  <h2>مشخصات فنی</h2>  <table>  <tr>  <th>پردازنده</th>  <td>Intel Core i7</td>  </tr>  <tr>  <th>رم</th>  <td>16GB</td>  </tr>  </table>  </section>  <section class="customer-reviews">  <h2>نظرات مشتریان</h2>  <article class="review">  <h3>عالی و قدرتمند</h3>  <p>این لپ‌تپ واقعاً برای گیمینگ عالیه...</p>  </article>  </section>  </main> |

تفاوت `<section>` با تگ‌های مشابه

<section>` vs `<article>`

- <article>`: برای محتوای مستقل و قابل انتشار مجدا (مانند پست وبلاگ، خبر)

- <section>`: برای گروه‌بندی محتوای مرتبط (نیاز به context صفحه دارد)

موارد استفاده صحیح از `<section>`

1. فصل‌های یک مقاله

2. بخش‌های مختلف صفحه اصلی

3. تب‌های محتوا

4. بخش‌های مختلف یک برنامه تحت وب

نکات مهم در استفاده از `<section>`

1. همیشه عنوان داشته باشد:

|  |
| --- |
| <!-- صحیح -->  <section>  <h2>عنوان بخش</h2>  <p>محتوای بخش...</p>  </section>  <!-- نادرست -->  <section>  <p>محتوای بدون عنوان...</p>  </section> |

2. برای استایل‌دهی استفاده نشود:

- از `<div>` برای استایل‌دهی استفاده کنید

- از `<section>` فقط برای معناسازی محتوا استفاده کنید

3. تودرتو بودن مجاز:

|  |
| --- |
| <section>  <h2>بخش اصلی</h2>  <section>  <h3>زیربخش اول</h3>  <p>محتوای زیربخش...</p>  </section>  </section> |

تگ `<section>` در تمام مرورگرهای مدرن به طور کامل پشتیبانی می‌شود.

## تگ article و Display آن

معرفی تگ `<article>`

تگ `<article>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای محتوای مستقل و خودکفا استفاده می‌شود. محتوای داخل `<article>` باید به تنهایی معنای کامل داشته باشد و قابل انتشار مجدد باشد.

هدف اصلی تگ `<article>`

این تگ برای محتوایی استفاده می‌شود که می‌تواند به تنهایی و خارج از context صفحه فعلی معنا داشته باشد و قابل انتشار در سایت‌های دیگر باشد.

ویژگی‌های کلیدی تگ `<article>`

1. محتوای مستقل (Self-contained):

- محتوا باید به تنهایی معنای کامل داشته باشد

- قابل درک باشد حتی اگر از بقیه صفحه جدا شود

2. قابل توزیع مجدد:

- می‌تواند در RSS feeds، سایت‌های دیگر یا اپلیکیشن‌ها استفاده شود

- برای سندیکاسیون (Syndication) مناسب است

3. قابل استفاده چندباره:

- می‌توانید چندین تگ `<article>` در یک صفحه داشته باشید

- هر article محتوای مستقل و مجزایی دارد

نحوه استفاده از تگ `<article>`

مثال ۱: پست‌های وبلاگ

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>وبلاگ آموزشی</title>  </head>  <body>  <header>  <h1>وبلاگ آموزش برنامه‌نویسی</h1>  </header>  <main>  <article class="blog-post">  <header>  <h2>آموزش کامل تگ‌های HTML5</h2>  <p>  <time datetime="2023-10-01">۱ مهر ۱۴۰۲</time> |  نویسنده: <span>محمد رضایی</span>  </p>  </header>    <section>  <h3>مقدمه</h3>  <p>HTML5 آخرین نسخه از زبان نشانه‌گذاری ابرمتن است که...</p>  </section>    <section>  <h3>تگ‌های معنایی</h3>  <p>تگ‌های معنایی به درک بهتر ساختار صفحه کمک می‌کنند...</p>  </section>    <footer>  <div class="tags">  <span>برچسب‌ها: </span>  <a href="/tag/html">HTML</a>  <a href="/tag/html5">HTML5</a>  <a href="/tag/web">وب</a>  </div>  <div class="comments">  <h4>نظرات (۵)</h4>  <!-- لیست نظرات -->  </div>  </footer>  </article>  <article class="blog-post">  <header>  <h2>آموزش Flexbox در CSS</h2>  <p>  <time datetime="2023-09-25">۲۵ شهریور ۱۴۰۲</time> |  نویسنده: <span>زهرا کریمی</span>  </p>  </header>  <p>Flexbox یک روش مدرن برای چیدمان در CSS است...</p>  <!-- محتوای کامل مقاله -->  </article>  </main>  <footer>  <!-- فوتر سایت -->  </footer>  </body>  </html> |

مثال ۲: اخبار و مطالب خبری

|  |
| --- |
| <section class="news-feed">  <h2>آخرین اخبار</h2>    <article class="news-item">  <h3>انتشار نسخه جدید مرورگر فایرفاکس</h3>  <p>شرکت موزیلا امروز نسخه ۱۲۰ مرورگر فایرفاکس را منتشر کرد...</p>  <footer>  <span>منبع: خبرگزاری فناوری</span> |  <time datetime="2023-10-15">۱۵ مهر ۱۴۰۲</time>  </footer>  </article>    <article class="news-item">  <h3>کنفرانس وب ایران ۱۴۰۲</h3>  <p>کنفرانس سالانه توسعه دهندگان وب در تاریخ ۲۰ آبان برگزار می‌شود...</p>  <footer>  <span>منبع: انجمن برنامه‌نویسان ایران</span> |  <time datetime="2023-10-10">۱۰ مهر ۱۴۰۲</time>  </footer>  </article>  </section> |

مثال ۳: نظرات کاربران

|  |
| --- |
| <section class="user-comments">  <h2>نظرات کاربران</h2>    <article class="comment">  <header>  <h3>تجربه استفاده از محصول</h3>  <div class="comment-meta">  <span>نویسنده: علی محمدی</span>  <time datetime="2023-10-05">۵ مهر ۱۴۰۲</time>  <div class="rating">★★★★☆</div>  </div>  </header>  <p>این محصول واقعاً عالی است. کیفیت ساخت فوق‌العاده‌ای دارد...</p>  <footer>  <button>پاسخ</button>  <button>پسندیدم</button>  </footer>  </article>    <article class="comment">  <header>  <h3>پیشنهاد بهبود</h3>  <div class="comment-meta">  <span>نویسنده: سارا احمدی</span>  <time datetime="2023-10-04">۴ مهر ۱۴۰۲</time>  </div>  </header>  <p>محصول خوبی است اما می‌تواند در بخش رابط کاربری بهبود یابد...</p>  </article>  </section> |

مثال ۴: محصولات در فروشگاه

|  |
| --- |
| <section class="products">  <h2>محصولات پرطرفدار</h2>    <article class="product">  <img src="laptop.jpg" alt="لپ‌تپ ایسوس">  <h3>لپ‌تپ ایسوس ROG</h3>  <p>لپ‌تپ گیمینگ با پردازنده Core i7 و کارت گرافیک RTX 4060</p>  <div class="price">۲۵,۰۰۰,۰۰۰ تومان</div>  <button>افزودن به سبد خرید</button>  </article>    <article class="product">  <img src="phone.jpg" alt="گوشی سامسونگ">  <h3>گوشی سامسونگ Galaxy S23</h3>  <p>گوشی هوشمند با دوربین ۲۰۰ مگاپیکسل و پردازنده اسنپدراگون</p>  <div class="price">۱۵,۰۰۰,۰۰۰ تومان</div>  <button>افزودن به سبد خرید</button>  </article>  </section> |

تفاوت `<article>` با تگ‌های مشابه

<article>` vs `<section>`

- <article>`: محتوای مستقل و کامل (مانند پست وبلاگ، خبر)

- <section>`: گروه‌بندی محتوای مرتبط (نیاز به context صفحه دارد)

<article>` vs `<div>`

- <div>`: تگ غیرمعنایی برای گروه‌بندی عمومی

- <article>`: تگ معنایی برای محتوای مستقل و کامل

ساختار معمول یک `<article>`

|  |
| --- |
| <article>  <header>  <h2>عنوان مقاله</h2>  <p>اطلاعات متا (نویسنده، تاریخ، دسته‌بندی)</p>  </header>    <section>  <h3>بخش اول</h3>  <p>محتوای بخش اول...</p>  </section>    <section>  <h3>بخش دوم</h3>  <p>محتوای بخش دوم...</p>  </section>    <footer>  <p>اطلاعات تکمیلی (برچسب‌ها، آمار، لینک‌های مرتبط)</p>  </footer>  </article> |

موارد استفاده صحیح از `<article>`

1. پست‌های وبلاگ

2. اخبار و مقالات خبری

3. نظرات کاربران

4. پست‌های فروم

5. ویجت‌های مستقل

6. کارت‌های محصول

نکات مهم در استفاده از `<article>`

1. عنوان داشته باشد:

|  |
| --- |
| <!-- صحیح -->  <article>  <h2>عنوان مقاله</h2>  <p>محتوای مقاله...</p>  </article> |

1. تودرتو بودن مجاز:

|  |
| --- |
| <article>  <h2>مقاله اصلی</h2>  <p>مقدمه...</p>    <article>  <h3>نظر خواننده</h3>  <p>متن نظر...</p>  </article>  </article> |

3. قابل شناسایی:

- بهتر است از id یا class برای شناسایی استفاده شود

- برای سندیکاسیون در RSS مفید است

پشتیبانی مرورگرها

مزایای استفاده از تگ `<article>`

1. سئو (SEO):

- موتورهای جستجو محتوای مستقل را بهتر درک می‌کنند

- برای featured snippets و rich results مناسب است

2. دسترسی‌پذیری (Accessibility):

- صفحه‌خوان‌ها می‌توانند بین article‌ها پیمایش کنند

- ساختار برای کاربران نابینا شفاف می‌شود

3. انتشار مجدد:

- قابل استفاده در RSS feeds

- قابل اشتراک‌گذاری در پلتفرم‌های مختلف

4. خوانایی کد:

- کد HTML سازمان‌یافته‌تر و معنایی‌تر می‌شود

چه زمانی از `<article>` استفاده کنیم؟

- وقتی محتوا مستقل است

- وقتی محتوا قابل انتشار مجدد است

- وقتی محتوا به تنهایی معنا دارد

چه زمانی از `<article>` استفاده نکنیم؟

- برای محتوای وابسته به context صفحه (از `<section>` استفاده کنید)

- برای گروه‌بندی عمومی (از `<div>` استفاده کنید)

خلاصه نکات مهم

- تگ `<article>` برای محتوای مستقل و خودکفا استفاده می‌شود

- باید به تنهایی معنای کامل داشته باشد

- می‌تواند چندین بار در یک صفحه استفاده شود

- برای محتوای قابل انتشار مجدد مناسب است

- یک تگ معنایی است که برای سئو و دسترسی‌پذیری مفید است

- معمولاً شامل عنوان، محتوا و اطلاعات متا است

با استفاده صحیح از تگ `<article>`، محتوای شما ساختار معنایی بهتری پیدا کرده و برای موتورهای جستجو و کاربران قابل درک‌تر خواهد بود.

## معرفی تگ aside و Display آن

مثال

نمایش مقداری محتوا جدا از محتوایی که در آن قرار گرفته است:

|  |
| --- |
| <p>My family and I visited The Epcot center this summer. The weather was nice, and Epcot was amazing! I had a great summer together with my family!</p>  <aside>  <h4>Epcot Center</h4>  <p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>  </aside> |

نمونه‌های بیشتر «خودتان امتحان کنید» در زیر آمده است.

تعریف و کاربرد

این <aside>تگ، جدا از محتوایی که در آن قرار می‌گیرد، محتوای دیگری را نیز تعریف می‌کند.

محتوای کناری باید به طور غیرمستقیم با محتوای اطراف مرتبط باشد.

نکته: محتوا <aside>اغلب به عنوان یک نوار کناری در یک سند قرار می‌گیرد.

نکته: این <aside>عنصر در مرورگر به صورت خاصی نمایش داده نمی‌شود. با این حال، می‌توانید از CSS برای استایل‌دهی به <aside> عنصر استفاده کنید (به مثال زیر مراجعه کنید).

ویژگی‌های سراسری

این <aside>تگ همچنین از ویژگی‌های سراسری (Global Attributes) در HTML پشتیبانی می‌کند .

ویژگی‌های رویداد

این <aside>تگ همچنین از ویژگی‌های رویداد در HTML پشتیبانی می‌کند .

تبلیغات

حذف تبلیغات

## معرفی تگ nav و Display آن

معرفی تگ `<nav>`

تگ `<nav>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای تعیین بخش ناوبری (Navigation) یا منوی راهبری یک صفحه وب استفاده می‌شود.

هدف اصلی تگ `<nav>`

این تگ برای گروه‌بندی لینک‌های مهم ناوبری که کاربران برای حرکت در سایت از آنها استفاده می‌کنند، طراحی شده است.

ویژگی‌های کلیدی تگ `<nav>`

1. عنصر معنایی (Semantic):

- ساختار صفحه را برای موتورهای جستجو و مرورگرها شفاف می‌کند

- به صفحه‌خوان‌ها (Screen Readers) در شناسایی بخش ناوبری اصلی کمک می‌کند

2. برای لینک‌های مهم:

- فقط برای لینک‌های ناوبری اصلی استفاده می‌شود

- نه برای همه لینک‌های موجود در صفحه

3. قابل استفاده چندباره:

- می‌توانید چندین تگ `<nav>` در یک صفحه داشته باشید

- هر nav برای گروه متفاوتی از لینک‌های ناوبری

نحوه استفاده از تگ `<nav>`

مثال ۱: منوی ناوبری اصلی سایت

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>فروشگاه اینترنتی</title>  </head>  <body>  <header>  <div class="logo">  <h1>فروشگاه ما</h1>  </div>    <nav aria-label="منوی اصلی">  <ul>  <li><a href="/">خانه</a></li>  <li><a href="/products">محصولات</a></li>  <li><a href="/about">درباره ما</a></li>  <li><a href="/contact">تماس با ما</a></li>  <li><a href="/blog">بلاگ</a></li>  </ul>  </nav>  </header>  <main>  <h2>محصولات ویژه</h2>  <!-- محتوای اصلی -->  </main>  <footer>  <nav aria-label="لینک‌های فوتر">  <ul>  <li><a href="/privacy">حریم خصوصی</a></li>  <li><a href="/terms">قوانین</a></li>  <li><a href="/sitemap">نقشه سایت</a></li>  </ul>  </nav>  </footer>  </body>  </html> |

مثال ۲: منوی ناوبری با dropdown

|  |
| --- |
| <nav class="main-navigation">  <ul>  <li><a href="/">خانه</a></li>  <li class="dropdown">  <a href="/products">محصولات ▼</a>  <ul class="dropdown-menu">  <li><a href="/products/laptops">لپ‌تپ</a></li>  <li><a href="/products/phones">موبایل</a></li>  <li><a href="/products/tablets">تبلت</a></li>  </ul>  </li>  <li><a href="/services">خدمات</a></li>  <li><a href="/support">پشتیبانی</a></li>  </ul>  </nav> |

مثال ۳: ناوبری breadcrumb

|  |
| --- |
| <nav aria-label="پagiناسیون مقالات">  <ul class="pagination">  <li><a href="/blog/page1">۱</a></li>  <li><a href="/blog/page2">۲</a></li>  <li><a href="/blog/page3" aria-current="page">۳</a></li>  <li><a href="/blog/page4">۴</a></li>  <li><a href="/blog/page5">۵</a></li>  </ul>  </nav> |

مثال ۵: ناوبری در sidebar

|  |
| --- |
| <aside>  <nav aria-label="دسته‌بندی مقالات">  <h3>دسته‌بندی‌ها</h3>  <ul>  <li><a href="/category/html">HTML</a></li>  <li><a href="/category/css">CSS</a></li>  <li><a href="/category/javascript">JavaScript</a></li>  <li><a href="/category/react">React</a></li>  </ul>  </nav>    <nav aria-label="مقالات محبوب">  <h3>مقالات پرطرفدار</h3>  <ul>  <li><a href="/article/1">آموزش Flexbox</a></li>  <li><a href="/article/2">آموزش Grid</a></li>  <li><a href="/article/3">بهینه‌سازی سئو</a></li>  </ul>  </nav>  </aside> |

چه لینک‌هایی باید در `<nav>` قرار گیرند؟

- منوی اصلی سایت

- منوی فوتر

- منوی دسته‌بندی‌ها

- Breadcrumb

- منوی سریع (Quick Links)

نامناسب برای nav :

- لینک‌های موجود در متن مقاله

- لینک‌های تبلیغاتی

- دکمه‌های اشتراک‌گذاری در شبکه‌های اجتماعی

- لینک‌های فرعی و کم اهمیت

نکات مهم در استفاده از `<nav>`

1. استفاده از لیست‌ها:

|  |
| --- |
| <!-- صحیح -->  <nav>  <ul>  <li><a href="/">خانه</a></li>  <li><a href="/about">درباره</a></li>  </ul>  </nav>  <!-- کمتر ایده‌آل -->  <nav>  <a href="/">خانه</a>  <a href="/about">درباره</a>  </nav> |

1. دسترسی‌پذیری (Accessibility):

|  |
| --- |
| <!-- استفاده از aria-label -->  <nav aria-label="منوی اصلی">  <!-- لینک‌ها -->  </nav>  <!-- یا استفاده از aria-labelledby -->  <h2 id="main-nav">ناوبری اصلی</h2>  <nav aria-labelledby="main-nav">  <!-- لینک‌ها -->  </nav> |

3. موقعیت‌یابی:

- معمولاً در `<header>` قرار می‌گیرد

- می‌تواند در footer>`، `<aside یا مستقیماً در `<body>` باشد

مزایای استفاده از تگ `<nav>`

1. سئو (SEO):

- موتورهای جستجو ساختار ناوبری سایت را بهتر درک می‌کنند

- لینک‌های مهم برای خزنده‌ها شناسایی می‌شوند

2. دسترسی‌پذیری (Accessibility):

- صفحه‌خوان‌ها می‌توانند کاربر را مستقیم به منوی ناوبری هدایت کنند

- کاربران نابینا به راحتی می‌توانند در سایت پیمایش کنند

3. خوانایی کد:

- کد HTML تمیزتر و معنایی‌تر می‌شود

- توسعه و نگهداری پروژه آسان‌تر می‌شود

4. تجربه کاربری:

- مرورگرها و دستگاه‌ها می‌توانند رفتارهای خاصی برای ناوبری پیاده‌سازی کنند

نمونه کامل با ساختار استاندارد

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>سایت آموزشی</title>  </head>  <body>  <a href="#main-content" class="skip-link">پرش به محتوای اصلی</a>    <header>  <nav aria-label="منوی اصلی">  <ul>  <li><a href="/" aria-current="page">خانه</a></li>  <li><a href="/courses">دوره‌ها</a></li>  <li><a href="/blog">بلاگ</a></li>  <li><a href="/about">درباره ما</a></li>  <li><a href="/contact">تماس</a></li>  </ul>  </nav>  </header>  <main id="main-content">  <!-- محتوای اصلی -->  </main>  <footer>  <nav aria-label="لینک‌های فوتر">  <ul>  <li><a href="/privacy">حریم خصوصی</a></li>  <li><a href="/terms">قوانین</a></li>  <li><a href="/help">راهنما</a></li>  </ul>  </nav>    <nav aria-label="شبکه‌های اجتماعی">  <ul>  <li><a href="#">توییتر</a></li>  <li><a href="#">اینستاگرام</a></li>  <li><a href="#">لینکدین</a></li>  </ul>  </nav>  </footer>  </body>  </html> |

خلاصه نکات مهم

- تگ `<nav>` برای بخش ناوبری اصلی سایت استفاده می‌شود

- می‌تواند چندین بار در یک صفحه استفاده شود

- فقط برای لینک‌های مهم ناوبری استفاده شود

- حتماً از لیست‌ها (`<ul>`, `<ol>`) برای ساختار داخلی استفاده کنید

- برای دسترسی‌پذیری از `aria-label` یا `aria-labelledby` استفاده کنید

- یک تگ معنایی است که برای سئو و دسترسی‌پذیری بسیار مفید است

با استفاده صحیح از تگ `<nav>`، ساختار ناوبری سایت شما برای موتورهای جستجو، مرورگرها و کاربران به خصوص افراد دارای معلولیت بهینه می‌شود.

## [معرفی تگ address و display آن](webdesign/html/tags/address)

معرفی تگ `<address>`

تگ `<address>` یکی از تگ‌های معنایی (Semantic) در HTML5 است که برای ارائه اطلاعات تماس مربوط به نویسنده، صاحب مقاله، یا صاحب سایت استفاده می‌شود.

هدف اصلی تگ `<address>`

این تگ برای مشخص کردن اطلاعات تماس مربوط به بخش parent خود (مانند `<article>` یا `<body>`) به کار می‌رود.

ویژگی‌های کلیدی تگ `<address>`

1. عنصر معنایی (Semantic):

- به موتورهای جستجو و مرورگرها می‌فهماند که این بخش حاوی اطلاعات تماس است

- برای دسترسی‌پذیری (Accessibility) بسیار مفید است

2. محدوده کاربرد:

- اطلاعات تماس نویسنده یک مقاله

- اطلاعات تماس صاحب سایت یا صفحه

- اطلاعات تماس سازمان یا شرکت

3. نحوه نمایش:

- به طور پیش‌فرض به صورت ایتالیک نمایش داده می‌شود

- می‌توان با CSS استایل آن را تغییر داد

نحوه استفاده از تگ `<address>`

مثال ۱: اطلاعات تماس در فوتر سایت

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>شرکت فناوری اطلاعات</title>  </head>  <body>  <header>  <h1>شرکت فناوری اطلاعات نوآوران</h1>  </header>  <main>  <article>  <h2>مقاله درباره HTML5</h2>  <p>محتوای مقاله...</p>    <address>  نویسنده: <a href="mailto:author@example.com">محمد رضایی</a><br>  تلفن: <a href="tel:+982112345678">۰۲۱-۱۲۳۴۵۶۷۸</a>  </address>  </article>  </main>  <footer>  <address>  <strong>شرکت نوآوران</strong><br>  آدرس: تهران، خیابان ولیعصر، پلاک ۱۲۳<br>  تلفن: <a href="tel:+982112345678">۰۲۱-۱۲۳۴۵۶۷۸</a><br>  ایمیل: <a href="mailto:info@example.com">info@example.com</a><br>  وبسایت: <a href="https://example.com">example.com</a>  </address>    <p>&copy; ۱۴۰۲ - تمامی حقوق محفوظ است.</p>  </footer>  </body>  </html> |

[footer](webdesign/html/tags/address/footer.html)

مثال ۲: اطلاعات نویسنده در انتهای مقاله

|  |
| --- |
| <article>  <header>  <h1>آموزش کامل HTML5</h1>  <p>تاریخ انتشار: <time datetime="2023-10-01">۱ مهر ۱۴۰۲</time></p>  </header>    <section>  <h2>مقدمه</h2>  <p>محتوای مقاله...</p>  </section>    <footer>  <address>  <h3>اطلاعات نویسنده</h3>  <p>نام: محمد رضایی</p>  <p>تخصص: توسعه front-end</p>  <p>ایمیل: <a href="mailto:m.rezaei@example.com">m.rezaei@example.com</a></p>  <p>وبسایت: <a href="https://mrezaei.blog.ir">mrezaei.blog.ir</a></p>  </address>  </footer>  </article> |

[writer](webdesign/html/tags/address/writer.html)

مثال ۳: اطلاعات شرکت در صفحه درباره ما

|  |
| --- |
| <main>  <h1>درباره شرکت ما</h1>  <p>شرکت ما با ۱۰ سال سابقه در زمینه توسعه وب فعالیت می‌کند...</p>    <address>  <h2>راه‌های ارتباطی</h2>  <div class="contact-info">  <div class="contact-item">  <strong>آدرس دفتر مرکزی:</strong>  <p>تهران، میدان ونک، خیابان ملاصدرا، پلاک ۴۵</p>  </div>  <div class="contact-item">  <strong>تلفن:</strong>  <p><a href="tel:+982188776655">۰۲۱-۸۸۷۷۶۶۵۵</a></p>  </div>  <div class="contact-item">  <strong>ایمیل:</strong>  <p><a href="mailto:contact@company.com">contact@company.com</a></p>  </div>  <div class="contact-item">  <strong>ساعات کاری:</strong>  <p>شنبه تا چهارشنبه: ۸:۰۰ تا ۱۷:۰۰</p>  </div>  </div>  </address>  </main> |

[about](webdesign/html/tags/address/about.html)

مثال ۴: استفاده در کنار تگ‌های دیگر

|  |
| --- |
| <section class="contact-section">  <h2>با ما در ارتباط باشید</h2>    <address>  <div class="contact-method">  <h3>پشتیبانی فنی</h3>  <p>ایمیل: <a href="mailto:support@example.com">support@example.com</a></p>  <p>تلفن: <a href="tel:+982155667788">۰۲۱-۵۵۶۶۷۷۸۸</a></p>  </div>    <div class="contact-method">  <h3>فروش</h3>  <p>ایمیل: <a href="mailto:sales@example.com">sales@example.com</a></p>  <p>تلفن: <a href="tel:+982199887766">۰۲۱-۹۹۸۸۷۷۶۶</a></p>  </div>    <div class="contact-method">  <h3>دفتر مرکزی</h3>  <p>آدرس: کرج، بلوار موذن، پلاک ۱۲۳</p>  <p>کد پستی: ۳۱۴۵۶۷۸۹۱۰</p>  </div>  </address>  </section> |

[side](webdesign/html/tags/address/side.html)

تفاوت `<address>` با تگ‌های مشابه

`<address>` vs `<p>`

- <p>`: برای پاراگراف‌های عمومی متن

- <address>`: مخصوص اطلاعات تماس با معنای خاص

<address>` vs `<div>`

- <div>`: تگ غیرمعنایی برای گروه‌بندی عمومی

- <address>`: تگ معنایی مخصوص اطلاعات تماس

چه محتوایی در `<address>` قرار گیرد؟

مناسب برای address:

- آدرس فیزیکی

- آدرس ایمیل

- شماره تلفن

- لینک به صفحه تماس

- اطلاعات نویسنده

- اطلاعات شرکت

- موقعیت جغرافیایی

نامناسب برای address:

- تاریخ انتشار

- برچسب‌های مقاله

- محتوای اصلی

- نظرات کاربران

- اطلاعات غیرمرتبط با تماس

نکات مهم در استفاده از `<address>`

1. محدوده معنی‌دهی:

|  |
| --- |
| <!-- صحیح - اطلاعات تماس مربوط به مقاله -->  <article>  <h1>مقاله نمونه</h1>  <p>محتوای مقاله...</p>  <address>  نویسنده: محمد رضایی<br>  ایمیل: author@example.com  </address>  </article>  <!-- صحیح - اطلاعات تماس مربوط به کل صفحه -->  <body>  <main>...</main>  <footer>  <address>  اطلاعات شرکت...  </address>  </footer>  </body> |

[correct](webdesign/html/tags/address/correct.html)

1. استفاده از لینک‌های قابل کلیک:

|  |
| --- |
| <address>  <!-- استفاده از mailto و tel -->  ایمیل: <a href="mailto:info@example.com">info@example.com</a><br>  تلفن: <a href="tel:+982112345678">۰۲۱-۱۲۳۴۵۶۷۸</a><br>  وبسایت: <a href="https://example.com">example.com</a>  </address> |

[link](webdesign/html/tags/address/link.html)

1. قالب‌بندی با CSS:

|  |
| --- |
| <style>  address {  font-style: normal; /\* حذف ایتالیک پیش‌فرض \*/  line-height: 1.6;  background: #f9f9f9;  padding: 15px;  border-right: 3px solid #007bff;  }    address a {  color: #007bff;  text-decoration: none;  }    address a:hover {  text-decoration: underline;  }  </style> |

[css](webdesign/html/tags/address/css.html)

مزایای استفاده از تگ `<address>`

1. سئو (SEO):

- موتورهای جستجو اطلاعات تماس را به راحتی شناسایی می‌کنند

- برای کسب‌وکارهای محلی بسیار مفید است

2. دسترسی‌پذیری (Accessibility):

- صفحه‌خوان‌ها بخش اطلاعات تماس را به درستی شناسایی می‌کنند

- کاربران نابینا به راحتی می‌توانند اطلاعات تماس را پیدا کنند

3. معناسازی (Semantics):

- کد HTML معنایی‌تر و قابل درک‌تر می‌شود

- توسعه‌دهندگان به راحتی می‌توانند بخش اطلاعات تماس را پیدا کنند

4. تجربه کاربری:

- کاربران به راحتی می‌توانند با کسب‌وکار تماس بگیرند

- لینک‌های mailto و tel تجربه تماس را آسان می‌کنند

نمونه کامل با ساختار استاندارد

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>شرکت طراحی وب</title>      <style>          .contact-card {              background: #f8f9fa;              border-radius: 8px;              padding: 20px;              margin: 20px 0;          }          .contact-card h3 {              color: #333;              margin-bottom: 15px;          }      </style>  </head>  <body>      <header>          <h1>شرکت طراحی وب پیشرو</h1>      </header>      <main>          <article>              <h2>آخرین پروژه ما</h2>              <p>توضیحات پروژه...</p>              <div class="contact-card">                  <address>                      <h3>اطلاعات مدیر پروژه</h3>                      <p><strong>نام:</strong> زهرا محمدی</p>                      <p><strong>سمت:</strong> مدیر فنی</p>                      <p><strong>ایمیل:</strong>                          <a href="mailto:z.mohammadi@example.com">z.mohammadi@example.com</a>                      </p>                      <p><strong>تلفن:</strong>                          <a href="tel:+989121234567">۰۹۱۲-۱۲۳-۴۵۶۷</a>                      </p>                  </address>              </div>          </article>      </main>      <footer>          <div class="footer-content">              <address>                  <h3>دفتر مرکزی</h3>                  <p>آدرس: اصفهان، خیابان چهارباغ، مجتمع تجاری قیصر</p>                  <p>تلفن: <a href="tel:+983112345678">۰۳۱-۱۲۳۴۵۶۷۸</a></p>                  <p>ایمیل: <a href="mailto:info@peshro-web.com">info@peshro-web.com</a></p>                  <p>ساعات کاری: ۸:۰۰ تا ۱۶:۰۰</p>              </address>          </div>      </footer>  </body>  </html> |

[stan](webdesign/html/tags/address/stan.html)

## [معرفی تگ em و display آن](webdesign/html/tags/em)

تگ `<em>` چیست؟

تگ `<em>` یک تگ معنایی (Semantic) در HTML است که برای تأکید روی یک بخش از متن استفاده می‌شود. کلمه `em` مخفف emphasis به معنی "تأکید" است.

کاربرد اصلی

هدف اصلی از استفاده از `<em>` این است که به مرورگر، موتورهای جستجو و خوانندگان صفحه (Screen Readers) بفهماند که این قسمت از متن دارای اهمیت خاصی است و باید با حالت تأکیدی خوانده یا تفسیر شود.

نحوه استفاده

|  |
| --- |
| <p>این یک متن معمولی است، اما <em>این قسمت</em> بسیار مهم است.</p> |

[use](webdesign/html/tags/em/use.html)

نتیجه در مرورگر:

این یک متن معمولی است، اما *این قسمت* بسیار مهم است.

ویژگی‌های ظاهری (پیش‌فرض مرورگر)

به طور پیش‌فرض، مرورگرها متن داخل `<em>` را به صورت مورب (کج) نمایش می‌دهند. اما این فقط یک نمایش بصری است و ماهیت اصلی تگ، معنایی است.

تفاوت `<em>` با `<i>`

این یک تفاوت مهم است:

|  |  |
| --- | --- |
| تگ <em> تأکید معنایی | تگ  <i> سبک ظاهری |
| معنای خاصی به متن اضافه می‌کند (تأکید). | فقط حالت ظاهری متن را تغییر می‌دهد (مورب). |
| برای خوانندگان صفحه و سئو مهم است. | فاقد بار معنایی است. |
| مثال: "من **حتما** آنجا خواهم بود." | مثال: اصطلاحات فنی، کلمات خارجی مانند "<i>et cetera</i>". |

تفاوت `<em>` با `<strong>`

هر دو برای تأکید هستند اما سطح تأکید آن‌ها متفاوت است:

`<em>`: برای تأکید معمولی به کار می‌رود. مانند تغییر لحن در یک جمله.

`<strong>`: برای تأکید قوی‌تر یا نشان دادن اهمیت و فوریت استفاده می‌شود. مانند یک هشدار.

|  |
| --- |
| <p>لطفا <em>قبل از</em> اقدام، <strong>حتما</strong> با مدیر خود مشورت کنید.</p> |

[dif](webdesign/html/tags/em/dif.html)

نتیجه در مرورگر:

لطفا *قبل از* اقدام، **حتما** با مدیر خود مشورت کنید.

نکات مهم

1. تغییر استایل:شما می‌توانید با CSS استایل پیش‌فرض تگ `<em>` (مورب) را تغییر دهید. برای مثال می‌توانید آن را به رنگ قرمز، زیرخط‌دار یا با پس‌زمینه خاص نمایش دهید.

|  |
| --- |
| em {  font-style: normal; /\* حذف حالت مورب \*/  color: red;  background-color: yellow;  } |

[style](webdesign/html/tags/em/style.html)

1. سطح تأکید: می‌توانید تگ‌های `<em>` را درون هم قرار دهید که سطح تأکید را بیشتر می‌کند.

|  |
| --- |
| <p>این <em>بخش خیلی <em>خیلی</em> مهم</em> است.</p> |

[Emphasis](webdesign/html/tags/em/emphasis.html)

3. دسترسی (Accessibility): خوانندگان صفحه (Screen Readers) هنگام رسیدن به متن داخل `<em>`، لحن صدای خود را تغییر می‌دهند و روی آن تأکید می‌کنند. این دلیل اصلی و مهم استفاده از این تگ است.

جمع‌بندی

از تگ `<em>` هرگاه می‌خواهید روی یک کلمه یا عبارت در جمله تأکید معنایی بگذارید استفاده کنید. این کار نه تنها ظاهر متن را زیباتر می‌کند، بلکه معنای آن را نیز غنی‌تر کرده و برای موتورهای جستجو و افراد نابینا قابل درک‌تر می‌سازد.

## [بررسی تفاوت‌های display های Inline و block](webdesign/html/inline-block)

حتما! در ادامه یک بررسی کامل و ساختاریافته از تفاوت‌های اصلی بین المان‌های Inline و Block در CSS ارائه می‌شود.

خلاصه سریع و کلیدی

نمایش بلوکی (`display: block`): مانند یک آجر است. کل عرض موجود را می‌گیرد و در یک خط جدید شروع می‌شود. می‌توانید به آن ارتفاع و عرض بدهید و مانند یک جعبه بزرگ آن را استایل دهید.

نمایش درون‌خطی (`display: inline`): مانند یک کلمه در یک پاراگراف است. فقط به اندازه محتوایش فضا می‌گیرد و در کنار عناصر دیگر در یک خط قرار می‌گیرد. نمی‌توانید ارتفاع و عرض مستقیم به آن بدهید.

مقایسه جزئی تفاوت‌ها

|  |  |  |
| --- | --- | --- |
| ویژگی | `display: block` (بلوکی) | `display: inline` (درون‌خطی) |
| شروع و پایان | همیشه در یک خط جدید شروع می‌شود. | در جریان عادی متن و در کنار عناصر دیگر قرار می‌گیرد. |
| عرض (Width) | به طور پیش‌فرض کل عرض والد را اشغال می‌کند. | فقط به عرض محتوای خودش است. تنظیم `width` بی‌اثر است. |
| ارتفاع (Height) | می‌توان `height` را تنظیم کرد. | تنظیم `height` بی‌اثر است. ارتفاع از خط متن (line-height) تأثیر می‌گیرد. |
| حاشیه (Margin) | چهار طرف (بالا، راست، پایین، چپ) قابل تنظیم است. | فقط حاشیه چپ و راست (افقی) قابل تنظیم است. حاشیه بالا و پایین بی‌اثر است و روی عناصر دیگر تأثیر نمی‌گذارد. |
| پدینگ (Padding) | چهار طرف قابل تنظیم است. | چهار طرف قابل تنظیم است، اما پدینگ بالا و پایین ممکن است روی عناصر مجاور همپوشانی (Overlap) داشته باشد. |
| قابلیت قرارگیری درون | می‌تواند سایر عناصر Block و Inline را در خود جای دهد. | فقط می‌تواند داده‌های متنی و سایر عناصر Inline را در خود جای دهد. نباید یک عنصر Block درون یک عنصر Inline قرار بگیرد. |

مثال عملی برای درک بهتر

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .block-example {  display: block;  background-color: lightblue;  margin: 10px 0; /\* حاشیه بالا و پایین دارد \*/  padding: 20px;  width: 80%; /\* عرض قابل تنظیم است \*/  height: 50px; /\* ارتفاع قابل تنظیم است \*/  }  .inline-example {  display: inline;  background-color: lightcoral;  margin: 0 20px; /\* فقط حاشیه چپ و راست دارد \*/  padding: 10px 20px; /\* پدینگ بالا و پایین دارد اما ممکن است همپوشانی کند \*/  /\* width و height در اینجا کار نمی‌کنند \*/  }  </style>  </head>  <body>  <div class="block-example">این یک دیو بلوکی است.</div>  <p class="block-example">این یک پاراگراف بلوکی دیگر است.</p>  <p>این یک پاراگراف است و <span class="inline-example">این یک اسپن درون‌خطی</span> است که در کنار متن قرار می‌گیرد. <strong class="inline-example">این یک المان قوی درون‌خطی دیگر است.</strong></p>  </body>  </html> |

[simple](webdesign/html/inline-block/simple.html)

نتیجه بصری کد بالا:

1. دو المان `div` و `p` که `block` هستند، هر کدام در یک خط جدید و در کل عرض موجود (یا ۸۰٪ی تعریف شده) نمایش داده می‌شوند و حاشیه و پدینگ آن‌ها به وضوح فاصله ایجاد می‌کند.

2. دو المان `span` و `strong` که `inline` هستند، در وسط خط متن قرار گرفته‌اند و فقط حاشیه‌های چپ و راست آن‌ها فاصله ایجاد می‌کند. پدینگ بالا و پایین آن‌ها ممکن است با خطوط متن بالا و پایین همپوشانی داشته باشد.

حالت سوم: `display: inline-block` (ترکیب هر دو)

گاهی به ویژگی‌های هر دو نیاز داریم. اینجاست که `inline-block` به کمک می‌آید.

\* مانند Inline است: در یک خط و در کنار عناصر دیگر قرار می‌گیرد.

\* مانند Block است: می‌توان به آن `width`، `height`، `margin` (از همه جهات) و `padding` (از همه جهات) داد بدون اینکه باعث همپوشانی شود.

مثال کاربردی: ساخت منوی ناوبری که آیتم‌های آن در یک خط هستند اما می‌توان به آن‌ها اندازه و حاشیه دقیق داد.

|  |
| --- |
| <style>  .nav-item {  display: inline-block; /\* کلید جادویی \*/  width: 100px;  height: 30px;  margin: 10px; /\* حاشیه از همه جهات رعایت می‌شود \*/  padding: 5px;  background-color: #eee;  text-align: center;  }  </style>  <div class="nav-item">خانه</div>  <div class="nav-item">محصولات</div>  <div class="nav-item">تماس با ما</div> |

جمع‌بندی نهایی

|  |  |
| --- | --- |
| وقتی که می‌خواهید... | از این نمایش استفاده کنید |
| یک المان به عنوان یک "جعبه" یا "بخش" مستقل عمل کند (مانند `div`, `section`, `p`, `h1`). | `block` |
| یک المان در جریان متن و بدون شکستن خط باشد (مانند `span`, `strong`, `a`, `img`). | `inline` |
| یک المان در کنار عناصر دیگر قرار گیرد اما بتوان به آن ارتفاع، عرض و حاشیه دقیق داد (مانند آیتم‌های منو، ستون‌های کوچک). | `inline-block` |

با درک این تفاوت‌های پایه‌ای، کنترل بسیار بهتری روی چیدمان و استایل‌دهی صفحات وب خود خواهید داشت.

## معرفی تگ strong و display آن

حتما! در ادامه یک معرفی کامل و کاربردی از تگ <strong>` ارائه می‌شود.

خلاصه سریع و کلیدی

تگ `<strong>` یک تگ معنایی (Semantic) در HTML است که برای تأکید قوی روی یک بخش از متن استفاده می‌شود. این تگ به مرورگرها، موتورهای جستجو و خوانندگان صفحه می‌گوید که محتوای درون آن از اهمیت بالایی برخوردار است.

ویژگی‌های اصلی

1. معنایی (Semantic): هدف اصلی آن بیان معنای "اهمیت قوی" است، نه فقط تغییر ظاهری متن.

2. ظاهر پیش‌فرض:در بیشتر مرورگرها، متن درون این تگ به صورت پررنگ (Bold) نمایش داده می‌شود.

3. تودرتویی (Nesting):هرچه تعداد این تگ‌ها درون هم بیشتر باشد، میزان "اهمیت" آن بخش از متن بیشتر در نظر گرفته می‌شود (اگرچه از نظر بصری تغییری نمی‌کند).

کاربرد و مثال‌ها

مثال ساده:

|  |
| --- |
| <p>  این یک متن معمولی است و <strong>این بخش بسیار مهم است</strong>.  </p> |

نتیجه در مرورگر:

این یک متن معمولی است و این بخش بسیار مهم است

مثال‌های کاربردی در موقعیت‌های واقعی:

هشدارهای مهم:

|  |
| --- |
| <p>  <strong>هشدار:</strong> قبل از نصب، از بکاپ گیری اطلاعات خود مطمئن شوید.  </p> |

کلیدواژه‌های اصلی در یک پاراگراف:

|  |
| --- |
| <p>  برای موفقیت در سئو، تولید <strong>محتوای با کیفیت</strong> و <strong>لینک‌سازی طبیعی</strong> ضروری است.  </p> |

[key](webdesign/html/tags/strong/key.html)

در فرم‌ها برای فیلدهای اجباری:

|  |
| --- |
| <label for="name">نام <strong>\*</strong></label>  <input type="text" id="name" required> |

تفاوت `<strong>` با `<b>` (بسیار مهم)

این یک تفاوت کلیدی و مفهومی است که بسیاری از توسعه‌دهندگان تازه‌کار در آن اشتباه می‌کنند.

|  |  |  |
| --- | --- | --- |
| ویژگی | تگ `<strong>` | تگ `<b>` |
| نوع تگ | معنایی (Semantic) | ظاهری (Presentational) |
| هدف | نشان دادن اهمیت قوی و ضرورت محتوا. | فقط پررنگ کردن متن بدون انتقال مفهوم خاصی. |
| تأثیر بر سئو | دارد. به موتورهای جستجو می‌فهماند که این بخش مهم است. | ندارد. موتورهای جستجو به آن وزن معنایی خاصی نمی‌دهند. |
| تأثیر بر خوانندگان صفحه | دارد. نرم‌افزارهای خواننده صفحه، هنگام برخورد با این تگ، تغییر لحن داده و کلمه را با تأکید بیشتری می‌خوانند. | ندارد. نرم‌افزارهای خواننده صفحه، آن را مانند متن عادی می‌خوانند. |
| ظاهر پیش‌فرض | پررنگ (Bold) | پررنگ (Bold) |

مثال برای درک تفاوت:

|  |
| --- |
| <p>من <b>گربه</b> را دیدم. آن یک <strong>گربهٔ وحشی</strong> بود!</p> |

[b](webdesign/html/tags/strong/b.html)

در این مثال، `<b>` فقط کلمه "گربه" را پررنگ کرده چون ممکن است طراح بخواهد از نظر بصری به آن توجه شود، اما معنای خاصی ندارد.

اما `<strong>` روی "گربهٔ وحشی" تأکید دارد تا به کاربر هشدار دهد که این یک موضوع مهم و بالقوه خطرناک است.

استایل‌دهی با CSS

اگرچه ظاهر پیش‌فرض `<strong>` پررنگ است، اما شما می‌توانید با CSS این ظاهر را به کلی تغییر دهید تا هم معنای مهم بودن را برساند و هم از نظر بصری به شکل دیگری دیده شود.

|  |
| --- |
| <style>  strong {  font-weight: normal; /\* پررنگی پیش‌فرض را حذف می‌کند \*/  color: red;  background-color: yellow;  padding: 2px 5px;  border-left: 3px solid red;  }  </style>  <p>لطفا <strong>قوانین و مقررات</strong> را به دقت مطالعه کنید.</p> |

[style](webdesign/html/tags/strong/style.html)

در این مثال، متن مهم دیگر پررنگ نیست، اما با رنگ و پس‌زمینه متمایز شده است، اما همچنان معنای "اهمیت" خود را حفظ کرده است.

نکات مهم و بهترین روش‌ها

\* استفاده به جا: فقط از `<strong>` برای بخش‌هایی که واقعاً اهمیت بالایی دارند استفاده کنید. استفاده بی‌رویه از آن، تأثیرگذاری آن را از بین می‌برد.

\* جایگزین نیست: این تگ جایگزین تگ‌های هدینگ (مانند `<h1>`, `<h2>`) نیست. برای ساختار اصلی صفحه حتماً از تگ‌های هدینگ استفاده کنید.

\* دسترسی‌پذیری (Accessibility): استفاده صحیح از `<strong>` به کاربرانی که از خوانندگان صفحه استفاده می‌کنند کمک می‌کند تا بر روی بخش‌های مهم متن تمرکز کنند.

جمع‌بندی نهایی

تگ <strong>` یک ابزار قدرتمند در HTML است که با استفاده صحیح از آن، می‌توانید:

1. تجربه کاربری بهتری ایجاد کنید.

2. سئوی صفحه خود را بهبود ببخشید.

3. دسترسی‌پذیری وبسایت را افزایش دهید.

همیشه به خاطر داشته باشید که HTML برای ساختار و معناست و CSS برای ظاهر. بنابراین، برای پررنگ کردن متن‌های "مهم" از `<strong>` و برای پررنگ کردن متن‌های "غیرمهم" (فقط برای زیبایی) از CSS (مثلاً `font-weight: bold;`) یا در موارد نهایی از `<b>` استفاده کنید.

## [معرفی تگ b و display آن](webdesign/html/tags/b)

خلاصه سریع و کلیدی

تگ `<b>` یک تگ ظاهری (Presentational) در HTML است که متن درون آن را فقط به صورت پررنگ (Bold) نمایش می‌دهد، بدون اینکه هیچ معنای خاص یا اهمیتی به آن متن بدهد.

ویژگی‌های اصلی

1. غیرمعنایی (Non-semantic): این تگ هیچ معنای اضافی مانند "اهمیت" یا "تأکید" را منتقل نمی‌کند.

2. ظاهر پیش‌فرض: متن درون این تگ در مرورگر به صورت پررنگ نمایش داده می‌شود.

3. هدف: صرفاً تغییر ظاهر متن برای اهداف زیبایی‌شناختی یا توجه بصری.

کاربرد و مثال‌ها

مثال ساده:

|  |
| --- |
| <p>این یک متن معمولی است و <b>این بخش پررنگ شده</b>.</p> |

[Simple](webdesign/html/tags/b/simple.html)

نتیجه در مرورگر:

این یک متن معمولی است و این بخش پررنگ شده.

مثال‌های کاربردی در موقعیت‌های واقعی:

هایلایت کردن عناوین محصول در یک لیست:

|  |
| --- |
| <ul>  <li><b>نام محصول:</b> لپ‌تاپ ایسوس</li>  <li><b>قیمت:</b> ۱۵ میلیون تومان</li>  </ul> |

[list](webdesign/html/tags/b/list.html)

جلب توجه بصری به یک کلمه کلیدی (بدون اهمیت معنایی):

|  |
| --- |
| <p>کلمه <b>HTML</b> مخفف HyperText Markup Language است.</p> |

[attention](webdesign/html/tags/b/attention.html)

در طراحی‌های خاص که نیاز به متن پررنگ است:

|  |
| --- |
| <p>ما به شما <b>تضمین</b> بهترین کیفیت را می‌دهیم.</p> |

[special](webdesign/html/tags/b/special.html)

تفاوت `<b>` با `<strong>` (مقایسه دقیق)

این مقایسه برای درک عمیق‌تر تفاوت این دو تگ حیاتی است.

|  |  |  |
| --- | --- | --- |
| ویژگی | تگ `<b>` | تگ `<strong>` |
| نوع تگ | ظاهری (Presentational) | معنایی (Semantic) |
| هدف اصلی | فقط پررنگ کردن ظاهری متن | نشان دادن اهمیت قوی و ضرورت محتوا |
|  |  |  |
| معنا و مفهوم | هیچ معنای خاصی ندارد | به موتورهای جستجو و خوانندگان صفحه می‌گوید این بخش مهم است |
| تأثیر بر سئو | ندارد | دارد (موتورهای جستجو به آن وزن می‌دهند) |
| تأثیر بر خوانندگان صفحه | ندارد (متن مانند متن عادی خوانده می‌شود) | دارد (با تأکید و لحن متفاوت خوانده می‌شود) |
| کاربرد مناسب | وقتی فقط می‌خواهید متن از نظر بصری پررنگ باشد | وقتی می‌خواهید روی اهمیت محتوا تأکید کنید |

مثال برای درک بهتر تفاوت:

|  |
| --- |
| <p>در این <b>مقاله</b> می‌خوانیم که <strong>رعایت نکات ایمنی</strong> بسیار حیاتی است.</p> |

[strong](webdesign/html/tags/b/strong.html)

تجزیه و تحلیل:

- `مقاله` با `<b>` پررنگ شده چون فقط می‌خواهیم توجه بصری به آن جلب شود.

- `رعایت نکات ایمنی` با `<strong>` پررنگ شده چون واقعاً مهم و حیاتی است.

استایل‌دهی با CSS

از آنجا که `<b>` یک تگ ظاهری است، می‌توانید ظاهر آن را به راحتی با CSS تغییر دهید:

|  |
| --- |
| <style>  b {  font-weight: normal; /\* پررنگی پیش‌فرض را حذف می‌کنیم \*/  color: blue;  text-decoration: underline;  background-color: #f0f0f0;  padding: 2px 4px;  }  </style>  <p>این یک <b>متن نمونه</b> برای نمایش استایل‌دهی است.</p> |

[style](webdesign/html/tags/b/style.html)

نکات مهم و بهترین روش‌ها

1. استفاده محدود: در HTML مدرن، استفاده از `<b>` باید محدود باشد. در بسیاری از موارد بهتر است از CSS استفاده کنید.

2. جایگزین CSS: برای پررنگ کردن معمولی، این روش بهتر است:

|  |
| --- |
| <p>این یک <span style="font-weight: bold;">متن پررنگ</span> است.</p>  <!-- یا با کلاس CSS -->  <style>.bold-text { font-weight: bold; }</style>  <p>این یک <span class="bold-text">متن پررنگ</span> است.</p> |

[span](webdesign/html/tags/b/span.html)

3. موارد استفاده مجاز `<b>`:

- هایلایت کردن کلمات کلیدی

- عناوین در لیست‌ها

- جلب توجه بصری بدون معنای خاص

4. مواردی که نباید از `<b>` استفاده کنید:

- برای متن‌های مهم و حیاتی

- برای تأکید روی محتوای کلیدی

- برای عناوین اصلی (به جای `<h1>`, `<h2>` و...)

جمع‌بندی نهایی

تگ <b>`:

- فقط برای تغییر ظاهر متن به صورت پررنگ است

- هیچ معنای خاصی ندارد

- برای سئو مفید نیست

- در HTML مدرن استفاده از آن محدود شده

توصیه مهم:

- برای متن‌های مهم از <strong>` استفاده کنید

- برای پررنگ کردن ظاهری معمولاً از CSS استفاده کنید

- از <b>` فقط در موارد خاص و محدود استفاده کنید

این رویکرد به شما کمک می‌کند کدهای تمیزتر، معنایی‌تر و بهینه‌تری داشته باشید.

## [تگ i و Display آن](webdesign/html/tags/i)

خلاصه سریع و کلیدی

تگ `<i>` یک تگ در HTML است که برای نمایش متن به صورت کج (ایتالیک) استفاده می‌شود. برخلاف گذشته که صرفاً یک تگ ظاهری بود، در HTML5 معنای جدیدی به آن داده شده است.

ویژگی‌های اصلی

1. معنایی (Semantic) در HTML5: برای نشان دادن متونی با صدای یا حالت متمایز از متن معمولی استفاده می‌شود.

2. ظاهر پیش‌فرض: متن درون این تگ در مرورگر به صورت کج (ایتالیک) نمایش داده می‌شود.

3. هدف: نشان دادن متونی که به نوعی از جریان عادی متن متمایز هستند.

کاربردهای صحیح `<i>` در HTML5

برخلاف `<b>` که هنوز ظاهری است، `<i>` در HTML5 معنای语义 مشخصی پیدا کرده است:

1. اصطلاحات فنی یا تخصصی

|  |
| --- |
| <p>مفهوم <i>relative positioning</i> در CSS بسیار مهم است.</p> |

[special](webdesign/html/tags/i/special.html)

1. کلمات خارجی

|  |
| --- |
| <p>کلمه <i>bonjour</i> در زبان فرانسوی به معنای سلام است.</p> |

[foreign](webdesign/html/tags/i/foreign.html)

1. اندیشه یا رویاپردازی

|  |
| --- |
| <p>او در فکر بود: <i>آیا این تصمیم درست است؟</i></p> |

[imagenation](webdesign/html/tags/i/imagenation.html)

1. نام کشتی‌ها و هواپیماها

|  |
| --- |
| <p>کشتی <i>Titanic</i> در سال ۱۹۱۲ غرق شد.</p> |

[names](webdesign/html/tags/i/names.html)

1. واژه‌های به عنوان نمونه

|  |
| --- |
| <p>مثلاً گیاهان <i>گلدار</i> و <i>بی‌گل</i> دو دسته اصلی هستند.</p> |

[sampels](webdesign/html/tags/i/sampeles.html)

تفاوت `<i>` با `<em>` (مهم!)

این تفاوت دقیقاً مشابه تفاوت `<b>` با `<strong>` است.

|  |  |  |
| --- | --- | --- |
| ویژگی | تگ `<i>` | تگ `<em>` |
| نوع تگ | معنایی (با مفهوم متمایز بودن) | معنایی (با مفهوم تأکید) |
| هدف اصلی | نشان دادن متن با حالت یا صدای متمایز | نشان دادن تأکید معنایی روی محتوا |
| تأثیر بر خوانندگان صفحه | ممکن است با لحن متفاوتی خوانده شود | با تأکید و استرس بیشتر خوانده می‌شود |
| ظاهر پیش‌فرض | ایتالیک (کج) | ایتالیک (کج) |
| مثال | کلمات خارجی، اصطلاحات فنی | تأکید روی اهمیت یک مفهوم |

مثال برای درک تفاوت:

|  |
| --- |
| <p>کلمه <i>pizza</i> ایتالیایی است، اما من <em>واقعاً</em> پیتزا را دوست دارم!</p> |

[em](webdesign/html/tags/i/em.html)

استایل‌دهی با CSS

می‌توانید ظاهر پیش‌فرض `<i>` را با CSS تغییر دهید:

|  |
| --- |
| <style>  i {  font-style: normal; /\* حالت ایتالیک پیش‌فرض را حذف می‌کند \*/  color: purple;  background-color: #f8f8f8;  padding: 2px 4px;  border-radius: 3px;  }  </style>  <p>این یک <i>متن نمونه</i> برای نمایش استایل‌دهی است.</p> |

استفاده به عنوان آیکون (روش رایج)

یکی از کاربردهای بسیار رایج `<i>` در کنار فونت‌آیکون‌ها است:

|  |
| --- |
| <!-- استفاده از Font Awesome -->  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">  <i class="fas fa-user"></i> پروفایل کاربر  <i class="fas fa-search"></i> جستجو  <i class="fas fa-home"></i> خانه |

[icons](webdesign/html/tags/i/icon.html)

در این حالت، `<i>` صرفاً یک عنصر نگهدارنده برای آیکون است و معنای خاصی ندارد.

نکات مهم و بهترین روش‌ها

1. استفاده معنایی: برای متن‌هایی که واقعاً "حالت متمایز" دارند از `<i>` استفاده کنید.

2. برای تأکید از `<em>` استفاده کنید: اگر هدف شما تأکید روی اهمیت متن است، از `<em>` استفاده نمایید.

3. جایگزین CSS: اگر فقط می‌خواهید متن را کج کنید (بدون معنای خاص):

|  |
| --- |
| <span style="font-style: italic;">متن کج</span> |

[span](webdesign/html/tags/i/span.html)

4. خوانایی: استفاده بیش از حد از متن ایتالیک می‌تواند خوانایی را کاهش دهد.

جمع‌بندی نهایی

تگ <i>`:

- در HTML5 معنادار شده است

- برای متن‌های با حالت متمایز استفاده می‌شود

- ظاهر پیش‌فرض: ایتالیک

- برای آیکون‌ها نیز کاربرد گسترده‌ای دارد

توصیه نهایی:

- برای متن‌های متمایز از <i>`

- برای تأکید از <em>`

- برای ایتالیک ظاهری از CSS

این رویکرد به شما کمک می‌کند کدهای معنایی‌تر و استانداردتری بنویسید.

## [معرفی تگ S و dispaly آن](webdesign/html/tags/s)

خلاصه سریع و کلیدی

تگ `<s>` یک تگ معنایی در HTML است که برای نمایش متن‌هایی استفاده می‌شود که دیگر دقیق، مرتبط یا معتبر نیستند. این تگ به صورت پیش‌فرض با یک خط روی متن (Strikethrough) نمایش داده می‌شود.

ویژگی‌های اصلی

1. معنایی (Semantic): هدف آن نشان دادن "عدم صحت یا مرتبط بودن" متن است، نه فقط خط کشیدن روی آن.

2. ظاهر پیش‌فرض: در مرورگرها، متن درون این تگ با یک خط افقی روی آن نمایش داده می‌شود.

3. کاربرد: برای نشان دادن تغییرات، به‌روزرسانی‌ها یا موارد باطل شده.

کاربرد و مثال‌ها

مثال ساده:

|  |
| --- |
| <p>قیمت اصلی: <s>۱۵۰,۰۰۰ تومان</s></p>  <p>قیمت با تخفیف: ۱۲۰,۰۰۰ تومان</p> |

نتیجه در مرورگر:

قیمت اصلی: ۱۵۰,۰۰۰ تومان

قیمت با تخفیف: ۱۲۰,۰۰۰ تومان

مثال‌های کاربردی:

\* لیست کارها:

|  |
| --- |
| <ul>  <li><s>خرید لباسشویی</s></li>  <li>تماس با پشتیبانی</li>  <li>ارسال ایمیل به مدیر</li>  </ul> |

[لیست](webdesign/html/tags/s/list.html)

محصولات تمام شده:

|  |
| --- |
| <div class="product">  <h3>لپ‌تاپ ایسوس</h3>  <p><s>موجود در انبار</s></p>  <p style="color: red;">اتمام موجودی</p>  </div> |

[product](webdesign/html/tags/s/product.html)

اطلاعات قدیمی:

|  |
| --- |
| <p>  تیم ما در  <s>  خیابان آزادی واقع بود  </s>  دفتر جدید واقع در خیابان ولیعصر مستقر است.  </p> |

تفاوت `<s>` با `<del>` (مهم!)

این تفاوت کلیدی است که بسیاری از توسعه‌دهندگان در آن اشتباه می‌کنند.

|  |  |  |
| --- | --- | --- |
| ویژگی | تگ `<s>` | تگ `<del>` |
| معنا و مفهوم | نشان دادن متن غیرمرتبط یا غیردقیق | نشان دادن متن حذف شده از سند |
| تأثیر بر سئو | ندارد | دارد (موتورهای جستجو می‌فهمند متن حذف شده) |
| جفت تگ مکمل | ندارد | با تگ `<ins>` (متن درج شده) استفاده می‌شود |
| کاربرد معمول | قیمت قدیمی، اطلاعات منسوخ | ویرایش‌های مستندات، تغییرات نسخه |

مثال برای درک تفاوت:

|  |
| --- |
| <p>قیمت: <s>۲۰۰,۰۰۰ تومان</s> ۱۵۰,۰۰۰ تومان</p>  <p>مستندات پروژه: <del>نسخه ۱.۲</del> <ins>نسخه ۲.۰</ins></p> |

[del](webdesign/html/tags/s/del.html)

تجزیه و تحلیل:

- در مثال اول، `<s>` فقط نشان می‌دهد قیمت ۲۰۰,۰۰۰ تومان دیگر معتبر نیست.

- در مثال دوم، `<del>` و `<ins>` نشان می‌دهند که نسخه ۱.۲ حذف و نسخه ۲.۰ جایگزین شده است.

استایل‌دهی با CSS

اگرچه ظاهر پیش‌فرض `<s>` خط خورده است، اما می‌توانید استایل آن را تغییر دهید:

|  |
| --- |
| <style>  s {  text-decoration: none; /\* خط پیش‌فرض را حذف می‌کند \*/  color: #888;  background-color: #ffe6e6;  padding: 2px 5px;  border-radius: 3px;  font-style: italic;  }  </style>  <p>این محصول <s>گارانتی مادام العمر</s> گارانتی ۲ ساله دارد.</p> |

[style](webdesign/html/tags/s/style.html)

نکات مهم و بهترین روش‌ها

1. استفاده معنایی: فقط برای متن‌هایی که واقعاً غیردقیق یا غیرمرتبط هستند از `<s>` استفاده کنید.

2. برای حذف واقعی از `<del>` استفاده کنید: اگر متن از یک سند حذف شده، از `<del>` استفاده نمایید.

3. جایگزین CSS: اگر فقط می‌خواهید خط روی متن بکشید (بدون معنای خاص):

|  |
| --- |
| <span style="text-decoration: line-through;">متن خط خورده</span> |

[span](webdesign/html/tags/s/span.html)

4. دسترسی‌پذیری: خوانندگان صفحه (Screen Readers) ممکن است متن درون `<s>` را با لحن متفاوتی اعلام کنند.

جمع‌بندی نهایی

تگ <s>`:

- برای نشان دادن متن‌های غیردقیق یا منسوخ استفاده می‌شود

- ظاهر پیش‌فرض: خط خورده

- معنایی است و فقط برای اهداف ظاهری نباید استفاده شود

توصیه نهایی:

- برای متن‌های غیرمرتبط از <s>`

- برای متن‌های حذف شده از <del>`

- برای خط خوردگی ظاهری از CSS

این رویکرد به شما کمک می‌کند کدهای استانداردتر و معنایی‌تری داشته باشید.

## [معرفی تگ u و display آن](webdesign/html/tags/u)

تگ‌های متنی و معنایی

۱. `<u>` (Underline)

- کاربرد: نمایش متن با خط زیردار

- معنا: در HTML5 برای نشان دادن متن‌هایی با تفاوت سبکی استفاده می‌شود

- مثال:

|  |
| --- |
| <p>این یک <u>متن خط دار</u> است.</p> |

## [معرفی تگ small و display آن](webdesign/html/tags/small)

کاربرد: نمایش متن با سایز کوچک‌تر

- معنا: برای نمایش اطلاعات فرعی مانند حق چاپ، توضیحات کوچک

- مثال:

|  |
| --- |
| <p>قیمت: ۱۵۰,۰۰۰ تومان <small>شامل مالیات</small></p> |

[index](webdesign/html/tags/small/index.html)

## معرفی تگ q و display آن

- کاربرد: نقل قول کوتاه و درون خطی

- معنا: نقل قول‌های کوتاه که نیاز به پاراگراف جداگانه ندارند

- ویژگی: مرورگر به طور خودکار کوتیشن اضافه می‌کند

- مثال:

|  |
| --- |
| <p>همانطور که <q>دانش قدرت است</q>، پس بیاموزیم.</p> |

[index](webdesign/html/tags/q/index.html)

## [معرفی تگ abbr و display آن](webdesign/html/tags/abbr)

- کاربرد: نشان دادن مخفف‌ها و اختصارات

- ویژگی: با attribute `title` توضیح کامل نشان داده می‌شود

- مثال:

|  |
| --- |
| <p><abbr title="HyperText Markup Language">HTML</abbr> زبان استاندارد وب است.</p> |

[index](webdesign/html/tags/abbr/index.html)

## [بررسی حالات nesting در element ها](webdesign/html/Nesting)

قوانین کلی:

- تگ‌های درون‌خطی می‌توانند در تگ‌های بلوکی قرار گیرند

- تگ‌های درون‌خطی می‌توانند در یکدیگر قرار گیرند

- تگ‌های بلوکی نباید در تگ‌های درون‌خطی قرار گیرند

مثال Nesting صحیح:

|  |
| --- |
| <p>  این یک پاراگراف است با  <strong>متن مهم و <mark>هایلایت شده</mark></strong>  و یک <abbr title="Example">EX</abbr>.  </p> |

[index](webdesign/html/Nesting/index.html)

مثال Nesting نادرست:

|  |
| --- |
| <!-- نادرست -->  <span><div>این اشتباه است</div></span> |

## [معرفی تگ sub و display آن](webdesign/html/tags/sub/index.html)

`<sub>` (Subscript)

- کاربرد: نمایش متن به صورت زیرنویس

- کاربردها: فرمول‌های شیمیایی، ریاضی

- مثال:

|  |
| --- |
| <p>فرمول آب: H<sub>2</sub>O</p> |

[index](webdesign/html/tags/sub/index.html)

## [معرفی تگ sup و display آن](webdesign/html/tags/sup)

- کاربرد: نمایش متن به صورت بالانویس

- کاربردها: توان در ریاضی، شماره‌گذاری پاورقی

- مثال:

|  |
| --- |
| <p>مساحت دایره: πr<sup>2</sup></p> |

[index](webdesign/html/tags/sup/index.html)

## [معرفی تگ mark و display آن](webdesign/html/tags/mark)

- کاربرد: هایلایت کردن متن

- معنا: نشان دادن متن برای مرجع یا اهمیت

- مثال:

|  |
| --- |
| <p>لطفا <mark>قسمت مهم</mark> را بخوانید.</p> |

[index](webdesign/html/tags/mark/index.html)

## معرفی تگ ins و display آن

- کاربرد: نشان دادن متن اضافه شده

- معنا: محتوای جدیدی که به سند اضافه شده

- ظاهر پیش‌فرض: خط زیردار

- مثال:

|  |
| --- |
| <p>برنامه ما: <del>پنجشنبه</del> <ins>جمعه</ins> برگزار می‌شود.</p> |

## معرفی تگ del و display آن

- کاربرد: نشان دادن متن حذف شده

- معنا: محتوایی که از سند حذف شده

- ظاهر پیش‌فرض: خط خورده

- مثال:

|  |
| --- |
| <p>قیمت: <del>۲۰۰,۰۰۰</del> ۱۵۰,۰۰۰ تومان</p> |

## [معرفی تگ br و display آن](webdesign/html/tags/br)

- کاربرد: ایجاد خط جدید بدون ایجاد پاراگراف جدید

- نوع: تگ تکی (Self-closing)

- مثال:

|  |
| --- |
| <p>نام: علی<br>سن: ۲۵<br>شهر: تهران</p> |

## [معرفی تگ div و display آن](webdesign/html/tags/div)

- کاربرد: نگهدارنده عمومی برای گروه‌بندی عناصر

- نوع: بلوکی (Block)

- کاربردها: ایجاد بخش‌های مختلف صفحه، گروه‌بندی برای استایل‌دهی

- مثال:

|  |
| --- |
| <div class="header">          <h1>عنوان سایت</h1>          <nav>منوی</nav>      </div> |

[index](webdesign/html/tags/div/index.html)

## [معرفی تگ span و display آن](webdesign/html/tags/span)

- کاربرد: نگهدارنده عمومی برای بخش‌های درون خطی

- نوع: درون‌خطی (Inline)

- کاربردها: استایل‌دهی به بخش‌های کوچک متن

- مثال:

|  |
| --- |
| <p>این یک <span class="highlight">متن ویژه</span> است.</p> |

[span](webdesign/html/tags/span/index.html)

## [معرفی id و class و بررسی تفاوت آن ها](webdesign/html/Attributes/idclass)

1. معرفی ID و Class در HTML

ID (شناسه)

- مشخصه منحصربفرد برای یک المان HTML

- فقط یک بار در هر صفحه قابل استفاده است

- با `#` در CSS و JavaScript شناسایی می‌شود

|  |
| --- |
| <div id="header">سرصفحه سایت</div>  <div id="main-content">محتوای اصلی</div>  #header {  background-color: blue;  color: white;  } |

[index](webdesign/html/Attributes/idclass/index.html)

Class (کلاس)

- مشخصه قابل استفاده مجدد برای چندین المان

- می‌تواند چندین بار در یک صفحه استفاده شود

- با `.` در CSS شناسایی می‌شود

|  |
| --- |
| <div class="button">دکمه 1</div>  <div class="button primary">دکمه 2</div>  <span class="button">دکمه 3</span>  .button {  padding: 10px;  border: 1px solid #ccc;  }  .primary {  background-color: blue;  color: white;  } |

تفاوت‌های اصلی ID و Class

|  |  |  |
| --- | --- | --- |
| ویژگی | ID | Class |
| تعداد استفاده | یک بار در صفحه | چندین بار |
| انتخاب‌گر CSS | `#` | `.` |
| اولویت | بالاتر | پایین‌تر |
| JavaScript | دسترسی مستقیم | نیاز به حلقه |
| هدف | المان منحصربفرد | گروه المان‌ها |

## [معرفی global attributes](#_1._Attribute_های)

ویژگی‌هایی که روی تمام المان‌های HTML قابل استفاده هستند:

|  |
| --- |
| <!-- class - برای استایل‌دهی -->  <div class="container main"></div>  <!-- id - برای شناسایی -->  <p id="description">توضیحات</p>  <!-- style - استایل inline -->  <span style="color: red; font-size: 16px;"></span>  <!-- title - راهنمای ابزار -->  <button title="برای ذخیره کلیک کنید">ذخیره</button>  <!-- data-\* - ذخیره داده‌های سفارشی -->  <div data-user-id="123" data-role="admin"></div>  <!-- lang - زبان محتوا -->  <p lang="fa">متن فارسی</p>  <!-- dir - جهت متن -->  <p dir="rtl">متن راست به چپ</p>  <!-- hidden - مخفی کردن -->  <div hidden>این مخفی است</div>  <!-- tabindex - ترتیب فوکوس -->  <input tabindex="1">  <button tabindex="2">کلیک</button>  <!-- accesskey - دسترسی سریع -->  <button accesskey="s">ذخیره (Alt+S)</button> |

[Global3](webdesign/html/Attributes/global3.html)

## [بررسی character escapes](webdesign/html/Entities)

برای نمایش کاراکترهای خاص در HTML:

کاراکترهای رایج:

|  |
| --- |
| &amp; <!-- & -->  &lt; <!-- < -->  &gt; <!-- > -->  &quot; <!-- " -->  &apos; <!-- ' -->  &nbsp; <!-- فاصله غیرقابل شکست -->  &copy; <!-- © -->  &reg; <!-- ® --> |

مثال‌های کاربردی:

|  |
| --- |
| <p>برای نمایش تگ از &lt;div&gt; استفاده کنید</p>  <p>شرکت ما&nbsp;تحت&nbsp;قانون&nbsp;کپی‌رایت&nbsp;است</p>  <p>قیمت: 100&euro;</p>  <p>&quot;متن درون نقل قول&quot;</p> |

```html

<p>برای نمایش تگ از &lt;div&gt; استفاده کنید</p>

<p>شرکت ما&nbsp;تحت&nbsp;قانون&nbsp;کپی‌رایت&nbsp;است</p>

<p>قیمت: 100&euro;</p>

<p>&quot;متن درون نقل قول&quot;</p>

## آموزش دستورات پایه CMD

دستورات اصلی:

|  |
| --- |
| :: پوشه جاری  cd  chdir  :: تغییر پوشه  cd C:\Users  cd ..  :: نمایش محتوای پوشه  dir  :: ساخت پوشه جدید  mkdir نام\_پوشه  md نام\_پوشه  :: ساخت فایل جدید  echo > filename.txt  type nul > filename.txt  :: حذف فایل  del filename.txt  :: حذف پوشه  rmdir نام\_پوشه  rd نام\_پوشه  :: کپی فایل  copy source.txt destination.txt  :: انتقال فایل  move source.txt destination.txt  :: تغییر نام  ren oldname.txt newname.txt  :: نمایش محتوای فایل  type filename.txt  :: پاک کردن صفحه  cls  :: اطلاعات سیستم  systeminfo  :: آدرس IP  ipconfig  :: ارتباط شبکه  ping google.com  :: مسیر فایل  path  :: تاریخ و زمان  date  time  :: تاریخچه دستورات  doskey /history |

مثال‌های کاربردی CMD:

|  |
| --- |
| :: رفتن به پوشه Documents  cd C:\Users\Username\Documents  :: ساخت پوشه پروژه جدید  mkdir MyProject  cd MyProject  :: ساخت فایل‌های پروژه  echo > index.html  echo > style.css  :: مشاهده لیست فایل‌ها  dir  :: بازگشت به پوشه قبل  cd ..  :: کپی یک پوشه  xcopy MyProject MyProjectBackup /E  :: مشاهده آدرس IP  ipconfig | findstr "IPv4"  :: تست اتصال به اینترنت  ping 8.8.8.8 |

نکات مهم CMD:

- برای اجرای دستورات مدیریتی، CMD را Run as Administrator باز کنید

- از کلید Tab برای تکمیل خودکار نام فایل/پوشه استفاده کنید

- با کلیدهای ↑ و ↓ در تاریخچه دستورات حرکت کنید

- برای متوقف کردن دستور در حال اجرا از Ctrl+C استفاده کنید

## معرفی node js

Node.js یک محیط اجرای جاوااسکریپت سمت سرور است که بر پایه موتور V8 کروم ساخته شده است.

ویژگی‌های اصلی:

- غیرمسدودکننده (Non-blocking)

- رویدادمحور (Event-driven)

- مناسب برای برنامه‌های Real-time

# آموزش گام به گام نصب و استفاده از Node.js

مرحله 1: دانلود Node.js

روش اول: از سایت رسمی

1. به آدرس (https://nodejs.org) بروید

2. نسخه LTS (پیشنهادی) را دانلود کنید

3. فایل نصب را اجرا کنید

روش دوم: با Package Manager (لینوکس/مک)

|  |
| --- |
| # Ubuntu/Debian  curl -fsSL https://deb.nodesource.com/setup\_lts.x | sudo -E bash -  sudo apt-get install -y nodejs  # macOS با Homebrew  brew install node  # Windows با Chocolatey  choco install nodejs |

مرحله 2: بررسی نصب

بررسی نسخه‌ها در ترمینال/CMD:

|  |
| --- |
| # بررسی نسخه Node.js  node --version  # یا  node -v  # بررسی نسخه NPM  npm --version  # یا  npm -v  # بررسی مسیر نصب  which node # لینوکس/مک  where node # ویندوز |

خروجی مورد انتظار:

|  |
| --- |
| node -v → v18.17.0 (یا نسخه بالاتر)  npm -v → 9.6.7 (یا نسخه بالاتر) |

مرحله 3: ایجاد اولین پروژه

قدم به قدم:

1. ایجاد پوشه پروژه:

|  |
| --- |
| # ایجاد پوشه جدید  mkdir my-first-node-project  # ورود به پوشه  cd my-first-node-project |

1. مقداردهی اولیه پروژه:

|  |
| --- |
| # ایجاد فایل package.json  npm init -y |

1. مشاهده ساختار پروژه:

|  |
| --- |
| my-first-node-project/  ├── package.json  └── (فایل‌های دیگر) |

مرحله 4: ایجاد اولین فایل Node.js

فایل `app.js` ایجاد کنید:

|  |
| --- |
| // app.js  console.log("سلام Node.js!");  // محاسبه ساده  const a = 5;  const b = 3;  console.log(`جمع ${a} و ${b} برابر است با: ${a + b}`);  // ماژول ساده  function greet(name) {  return `سلام ${name}! خوش آمدید!`;  }  console.log(greet("کاربر")); |

اجرای فایل:

|  |
| --- |
| node app.js |

خروجی:

|  |
| --- |
| سلام Node.js!  جمع 5 و 3 برابر است با: 8  سلام کاربر! خوش آمدید! |

مرحله 5: ایجاد سرور وب ساده

فایل `server.js`:

|  |
| --- |
| // server.js  const http = require('http');  // ایجاد سرور  const server = http.createServer((req, res) => {  // تنظیم هدر  res.writeHead(200, {  'Content-Type': 'text/html; charset=utf-8'  });    // مسیرهای مختلف  if (req.url === '/') {  res.end(`  <!DOCTYPE html>  <html dir="rtl">  <head>  <meta charset="UTF-8">  <title>سرور Node.js</title>  </head>  <body>  <h1>سلام! به سرور Node.js خوش آمدید</h1>  <p>این یک صفحه ساده است</p>  <a href="/about">درباره ما</a>  </body>  </html>  `);  } else if (req.url === '/about') {  res.end(`  <!DOCTYPE html>  <html dir="rtl">  <head>  <meta charset="UTF-8">  <title>درباره ما</title>  </head>  <body>  <h1>درباره ما</h1>  <p>این صفحه درباره ما است</p>  <a href="/">بازگشت به خانه</a>  </body>  </html>  `);  } else {  res.end(`  <!DOCTYPE html>  <html dir="rtl">  <body>  <h1>404 - صفحه پیدا نشد</h1>  <a href="/">بازگشت به خانه</a>  </body>  </html>  `);  }  });  // راه‌اندازی سرور  const PORT = 3000;  server.listen(PORT, () => {  console.log(`سرور در حال اجرا در: http://localhost:${PORT}`);  console.log('برای توقف سرور: Ctrl+C');  }); |

اجرای سرور:

|  |
| --- |
| node server.js |

تست سرور:

1. مرورگر را باز کنید

2. به آدرس `http://localhost:3000` بروید

3. صفحات مختلف را تست کنید

مرحله 6: کار با NPM و نصب پکیج

نصب پکیج‌های مفید:

|  |
| --- |
| # نصب Express (فریمورک وب)  npm install express  # نصب Nodemon (برای توسعه - ریستارت خودکار)  npm install --save-dev nodemon  # نصب چند پکیج با هم  npm install axios cors dotenv |

فایل `package.json` پس از نصب:

|  |
| --- |
| {  "name": "my-first-node-project",  "version": "1.0.0",  "description": "",  "main": "app.js",  "scripts": {  "start": "node server.js",  "dev": "nodemon server.js"  },  "dependencies": {  "express": "^4.18.0",  "axios": "^1.0.0",  "cors": "^2.8.5",  "dotenv": "^16.0.0"  },  "devDependencies": {  "nodemon": "^2.0.0"  }  } |

مرحله 7: ایجاد سرور با Express

نصب Express:

|  |
| --- |
| npm install express |

فایل `express-server.js`:

|  |
| --- |
| // express-server.js  const express = require('express');  const app = express();  const PORT = 3000;  // میدلور برای پارس کردن JSON  app.use(express.json());  // میدلور برای فایل‌های استاتیک  app.use(express.static('public'));  // روت اصلی  app.get('/', (req, res) => {  res.send(`  <!DOCTYPE html>  <html dir="rtl">  <head>  <meta charset="UTF-8">  <title>Express App</title>  <style>  body { font-family: Arial, sans-serif; margin: 40px; }  h1 { color: #333; }  </style>  </head>  <body>  <h1>سلام! این سرور Express است</h1>  <p>کاربران:</p>  <ul id="users"></ul>  <script>  // دریافت کاربران از API  fetch('/api/users')  .then(response => response.json())  .then(users => {  const list = document.getElementById('users');  users.forEach(user => {  const li = document.createElement('li');  li.textContent = user.name;  list.appendChild(li);  });  });  </script>  </body>  </html>  `);  });  // API ساده  app.get('/api/users', (req, res) => {  const users = [  { id: 1, name: 'علی' },  { id: 2, name: 'رضا' },  { id: 3, name: 'سارا' }  ];  res.json(users);  });  // روت POST  app.post('/api/users', (req, res) => {  const newUser = req.body;  console.log('کاربر جدید:', newUser);  res.json({ message: 'کاربر با موفقیت ایجاد شد', user: newUser });  });  // راه‌اندازی سرور  app.listen(PORT, () => {  console.log(`سرور Express در حال اجرا در: http://localhost:${PORT}`);  }); |

اجرای سرور Express:

|  |
| --- |
| # روش عادی  node express-server.js  # یا با Nodemon (برای توسعه)  npx nodemon express-server.js |

مرحله 8: تنظیم Scripts در package.json

ویرایش فایل `package.json`:

|  |
| --- |
| {  "name": "my-first-node-project",  "version": "1.0.0",  "main": "app.js",  "scripts": {  "start": "node server.js",  "dev": "nodemon server.js",  "express": "node express-server.js",  "express-dev": "nodemon express-server.js",  "test": "echo \"Error: no test specified\" && exit 1"  },  "dependencies": {  "express": "^4.18.0"  },  "devDependencies": {  "nodemon": "^2.0.0"  }  } |

استفاده از Scripts:

|  |
| --- |
| # اجرای سرور اصلی  npm start  # اجرای سرور با Nodemon (توسعه)  npm run dev  # اجرای سرور Express  npm run express  # اجرای سرور Express با Nodemon  npm run express-dev |

مرحله 9: ساختار پروژه حرفه‌ای

ایجاد ساختار پوشه‌ها:

|  |
| --- |
| # ایجاد پوشه‌های مختلف  mkdir routes models controllers public views utils  # ایجاد فایل‌های اصلی  touch app.js server.js .env .gitignore  # ایجاد فایل‌های درون پوشه‌ها  touch routes/users.js models/user.js controllers/userController.js  touch public/style.css public/script.js |

ساختار نهایی:

|  |
| --- |
| my-node-project/  ├── app.js # فایل اصلی  ├── server.js # راه‌انداز سرور  ├── package.json  ├── .env # متغیرهای محیطی  ├── .gitignore  ├── routes/ # مسیرها  │ └── users.js  ├── models/ # مدل‌های داده  │ └── user.js  ├── controllers/ # کنترلرها  │ └── userController.js  ├── public/ # فایل‌های استاتیک  │ ├── style.css  │ └── script.js  ├── views/ # template ها  └── utils/ # ابزارهای کمکی |

مرحله 10: عیب‌یابی و دیباگ

روش‌های دیباگ:

|  |
| --- |
| // 1. استفاده از console.log  console.log('مقدار متغیر:', variable);  // 2. استفاده از console.table برای آرایه‌ها و آبجکت‌ها  console.table(users);  // 3. استفاده از debugger  function calculate(a, b) {  debugger; // توقف در این خط  return a + b;  }  // 4. استفاده از ماژول util برای آبجکت‌های پیچیده  const util = require('util');  console.log(util.inspect(myObject, { depth: null })); |

اجرای دیباگ:

|  |
| --- |
| # اجرا با inspect  node inspect app.js  # یا با فلگ --inspect  node --inspect app.js |

مرحله 11: پروژه عملی - مدیریت وظایف

فایل `todo-app.js`:

|  |
| --- |
| // todo-app.js  const express = require('express');  const app = express();  const PORT = 3000;  app.use(express.json());  app.use(express.static('public'));  let todos = [  { id: 1, task: 'یادگیری Node.js', completed: false },  { id: 2, task: 'ایجاد پروژه عملی', completed: false }  ];  // صفحه اصلی  app.get('/', (req, res) => {  res.sendFile(\_\_dirname + '/public/todo.html');  });  // API دریافت همه وظایف  app.get('/api/todos', (req, res) => {  res.json(todos);  });  // API افزودن وظیفه جدید  app.post('/api/todos', (req, res) => {  const newTodo = {  id: todos.length + 1,  task: req.body.task,  completed: false  };  todos.push(newTodo);  res.json(newTodo);  });  // API به‌روزرسانی وظیفه  app.put('/api/todos/:id', (req, res) => {  const id = parseInt(req.params.id);  const todo = todos.find(t => t.id === id);  if (todo) {  todo.completed = req.body.completed;  res.json(todo);  } else {  res.status(404).json({ error: 'وظیفه پیدا نشد' });  }  });  app.listen(PORT, () => {  console.log(`📝 برنامه مدیریت وظایف در حال اجرا در: http://localhost:${PORT}`);  }); |

فایل `public/todo.html`:

|  |
| --- |
| <!DOCTYPE html>  <html dir="rtl">  <head>  <meta charset="UTF-8">  <title>مدیریت وظایف</title>  <style>  body { font-family: Arial, sans-serif; margin: 40px; }  .completed { text-decoration: line-through; color: #888; }  li { margin: 10px 0; cursor: pointer; }  </style>  </head>  <body>  <h1>✅ مدیریت وظایف</h1>    <div>  <input type="text" id="newTask" placeholder="وظیفه جدید...">  <button onclick="addTask()">افزودن</button>  </div>    <ul id="todoList"></ul>  <script>  // دریافت وظایف  async function loadTasks() {  const response = await fetch('/api/todos');  const todos = await response.json();  renderTasks(todos);  }  // نمایش وظایف  function renderTasks(todos) {  const list = document.getElementById('todoList');  list.innerHTML = '';    todos.forEach(todo => {  const li = document.createElement('li');  li.textContent = todo.task;  li.className = todo.completed ? 'completed' : '';  li.onclick = () => toggleTask(todo.id);  list.appendChild(li);  });  }  // افزودن وظیفه جدید  async function addTask() {  const input = document.getElementById('newTask');  const task = input.value.trim();    if (task) {  await fetch('/api/todos', {  method: 'POST',  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify({ task })  });    input.value = '';  loadTasks();  }  }  // تغییر وضعیت وظیفه  async function toggleTask(id) {  const todo = (await fetch('/api/todos')).json()  .then(todos => todos.find(t => t.id === id));    await fetch(`/api/todos/${id}`, {  method: 'PUT',  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify({ completed: !todo.completed })  });    loadTasks();  }  // بارگذاری اولیه  loadTasks();  </script>  </body>  </html> |

خلاصه مراحل نصب و استفاده:

1. دانلود و نصب Node.js از سایت رسمی

2. بررسی نصب با دستورات `node -v` و `npm -v`

3. ایجاد پروژه با `npm init -y`

4. نوشتن کد در فایل `.js`

5. اجرای کد با `node filename.js`

6. نصب پکیج با `npm install package-name`

7. ایجاد سرور با Express یا http ماژول

8. استفاده از Scripts در package.json

9. ساختاردهی پروژه به صورت حرفه‌ای

10. عیب‌یابی و توسعه برنامه

## معرفی npm

NPM مدیریت بسته‌های جاوااسکریپت و بزرگترین ریپازیتوری کتابخانه‌های جهان است.

دستورات مهم NPM:

|  |
| --- |
| # بررسی نسخه‌ها  node --version  npm --version  # ایجاد پروژه جدید  npm init  npm init -y  # نصب پکیج  npm install package-name  npm install package-name --save-dev # برای توسعه  npm install package-name -g # نصب  # حذف پکیج  npm uninstall package-name  # اجرای اسکریپت‌ها  npm run script-name  # آپدیت NPM  npm install -g npm@latest |

فایل package.json:

|  |
| --- |
| {  "name": "my-project",  "version": "1.0.0",  "scripts": {  "start": "node app.js",  "dev": "nodemon app.js"  },  "dependencies": {  "express": "^4.18.0"  },  "devDependencies": {  "nodemon": "^2.0.0"  }  } |

## [معرفی live-server و دستورات آن](#_معرفی_پلاگین_Live)

یک سرور توسعه سبک برای تست صفحات وب

نصب و استفاده:

|  |
| --- |
| نصب  npm install -g live-server  اجرا در پوشه جاری  live-server  اجرا در پورت خاص  live-server --port=8080  عدم باز شدن خودکار مرورگر  live-server --no-browser  مشخص کردن پوشه  live-server ./public |

دستورات live-server:

|  |
| --- |
| live-server --port=3000 # پورت مشخص  live-server --host=localhost # هاست مشخص  live-server --open=about.html # باز کردن فایل خاص  live-server --wait=1000 # تاخیر قبل از ریلود  live-server --cors # فعال کردن CORS |

## [معرفی تگ a و display آن](webdesign/html/tags/a)

تگ `<a>` (لینک):

|  |
| --- |
| <a href="https://example.com">متن لینک</a> |

[index](webdesign/html/tags/a/index.html)

Property Display تگ `<a>`:

- به صورت inline نمایش داده می‌شود

- می‌تواند با CSS به block یا inline-block تغییر کند

|  |
| --- |
| /\* حالت پیش‌فرض - inline \*/  a {  display: inline;  }  /\* تغییر به block \*/  a.block-link {  display: block;  padding: 10px;  background: blue;  color: white;  }  /\* تغییر به inline-block \*/  a.button {  display: inline-block;  padding: 10px 20px;  background: green;  color: white;  text-decoration: none;  } |

[style](webdesign/html/tags/a/style.html)

معرفی Attribute href

انواع مقادیر href:

|  |
| --- |
| <!-- لینک به سایت خارجی -->  <a href="https://google.com">گوگل</a>  <!-- لینک به صفحه داخلی -->  <a href="about.html">درباره ما</a>  <!-- لینک به پوشه -->  <a href="/products/">محصولات</a>  <!-- لینک به فایل -->  <a href="files/document.pdf">دانلود PDF</a>  <!-- لینک به ایمیل -->  <a href="mailto:example@email.com">ارسال ایمیل</a>  <!-- لینک به تلفن -->  <a href="tel:+98123456789">تماس با ما</a>  <!-- لینک به بخشی از صفحه -->  <a href="#section1">برو به بخش 1</a>  <!-- لینک جاوااسکریپت -->  <a href="javascript:void(0)">لینک JS</a> |

[Href](webdesign/html/tags/a/href.html)

معرفی URL های Relative و Absolute

Absolute URL (مطلق):

|  |
| --- |
| <!-- کامل و مستقل -->  <a href="https://www.example.com/products/index.html">  محصولات  </a>  <a href="http://localhost:3000/about">  درباره ما  </a> |

[url](webdesign/html/tags/a/url.html)

Relative URL (نسبی):

|  |
| --- |
| <!-- نسبت به پوشه جاری -->  <a href="contact.html">تماس با ما</a>  <!-- پوشه والد -->  <a href="../admin/dashboard.html">پنل مدیریت</a>  <!-- پوشه فرزند -->  <a href="images/photo.jpg">عکس</a>  <!-- از root سایت -->  <a href="/css/style.css">استایل</a> |

لینک کردن تگ a به بخشی از سایت و استفاده

**ایجاد لینک‌های داخلی:**

|  |
| --- |
| <!-- منوی ناوبری -->  <nav>  <a href="#home">خانه</a>  <a href="#about">درباره</a>  <a href="#services">خدمات</a>  <a href="#contact">تماس</a>  </nav>  <!-- بخش‌های صفحه -->  <section id="home">  <h2>خانه</h2>  <p>محتوای بخش خانه...</p>  </section>  <section id="about">  <h2>درباره ما</h2>  <p>محتوای بخش درباره ما...</p>  </section>  <section id="services">  <h2>خدمات</h2>  <p>محتوای بخش خدمات...</p>  </section> |

[inner](webdesign/html/tags/a/inner.html)

از پراپرتی scroll-behavior

**CSS برای اسکرول نرم:**

|  |
| --- |
| html {  scroll-behavior: smooth;  }  /\* یا برای مرورگرهای خاص \*/  section {  scroll-margin-top: 50px; /\* برای fixed header \*/  } |

[scroll](webdesign/html/tags/a/scroll.html)

**جاوااسکریپت برای اسکرول سفارشی:**

|  |
| --- |
| document.querySelectorAll('a[href^="#"]').forEach(anchor => {  anchor.addEventListener('click', function (e) {  e.preventDefault();  const target = document.querySelector(this.getAttribute('href'));  target.scrollIntoView({  behavior: 'smooth',  block: 'start'  });  });  }); |

[script](webdesign/html/tags/a/script.html)

بررسی Attribute Target

**مقادیر مختلف Target:**

|  |
| --- |
| <!-- باز شدن در تب جدید -->  <a href="https://example.com" target="\_blank">  باز کردن در تب جدید  </a>  <!-- باز شدن در فریم والد -->  <a href="page.html" target="\_parent">  باز کردن در والد  </a>  <!-- باز شدن در همان فریم -->  <a href="page.html" target="\_self">  باز کردن در همین فریم (پیش‌فرض)  </a>  <!-- باز شدن در بالاترین فریم -->  <a href="page.html" target="\_top">  باز کردن در بالاترین فریم  </a>  <!-- باز شدن در فریم خاص -->  <a href="page.html" target="frame-name">  باز کردن در فریم مشخص  </a> |

[Target](webdesign/html/tags/a/target.html)

**امنیت با rel="noopener noreferrer":**

|  |
| --- |
| <a href="https://external-site.com"  target="\_blank"  rel="noopener noreferrer">  لینک امن به سایت خارجی  </a> |

[**noopener**](webdesign/html/tags/a/noopener.html)

|  |
| --- |
| rel="noopener"  مسدود کردن دسترسی صفحه جدید به window.opener  جلوگیری از حملات phishing و tabnabbing  حفظ حریم خصوصی صفحه اصلی  rel="noreferrer"  عدم ارسال اطلاعات منبع (Referrer Header)  مخفی کردن منبع ترافیک  حفظ اطلاعات Analytics داخلی |

این ویژگی‌ها به‌طور عمده برای افزایش امنیت و حفظ حریم خصوصی به کار می‌روند و موتورهای جستجو مانند گوگل، آن‌ها را در الگوریتم‌های رتبه‌بندی خود لحاظ نمی‌کنند.

noopener چگونه امنیت را حفظ می‌کند - مثال عملی

|  |
| --- |
| // در صفحه malicious-site.com  if (window.opener) {  // تغییر آدرس صفحه اصلی به یک سایت فیشینگ  window.opener.location = 'https://fake-bank.com/login';    // یا دسترسی به اطلاعات حساس  const sensitiveData = window.opener.document.cookie;    // یا اجرای کدهای مخرب دیگر  window.opener.alert('شما هک شده‌اید!');  } |

|  |
| --- |
| window.opener.document.body.innerHTML = fakeLoginPage; |

## بررسی mail links

**لینک‌های ایمیل:**

|  |
| --- |
| <!-- ایمیل ساده -->  <a href="mailto:info@company.com">  ارسال ایمیل  </a>  <!-- ایمیل با موضوع -->  <a href="mailto:info@company.com?subject=سوال">  ایمیل با موضوع  </a>  <!-- ایمیل با متن -->  <a href="mailto:info@company.com?body=سلام، من علاقه‌مند هستم">  ایمیل با متن  </a>  <!-- ایمیل با موضوع و متن -->  <a href="mailto:info@company.com?subject=سوال&body=لطفا با من تماس بگیرید">  ایمیل کامل  </a>  <!-- ایمیل به چند نفر -->  <a href="mailto:person1@email.com,person2@email.com">  ایمیل به چند نفر  </a>  <!-- ایمیل با CC و BCC -->  <a href="mailto:main@email.com?cc=cc@email.com&bcc=bcc@email.com">  ایمیل با CC و BCC  </a> |

[mail](webdesign/html/tags/a/email.html)

## بررسی tel links

**لینک‌های تلفن:**

|  |
| --- |
| <!-- شماره ساده -->  <a href="tel:+982112345678">  021-12345678  </a>  <!-- با کد کشور -->  <a href="tel:+989121234567">  0912-123-4567  </a>  <!-- برای دستگاه‌های مختلف -->  <a href="tel:+982112345678"  onclick="ga('send', 'event', 'Contact', 'Call', 'Phone');">  تماس با پشتیبانی  </a>  <!-- فرمت‌های مختلف -->  <a href="tel:02112345678">02112345678</a>  <a href="tel:+98-21-1234-5678">+98-21-1234-5678</a>  <a href="tel:00982112345678">00982112345678</a> |

[tell](webdesign/html/tags/a/Tel.html)

**استایل‌دهی لینک‌های تلفن:**

|  |
| --- |
| /\* فقط نمایش روی دستگاه‌های تلفن \*/  @media (max-width: 768px) {  .phone-link {  display: block;  background: green;  color: white;  padding: 15px;  text-align: center;  text-decoration: none;  border-radius: 5px;  }  }  /\* آیکون تلفن \*/  .phone-link::before {  content: "📞 ";  margin-left: 5px;  } |

[phone-style](webdesign/html/tags/a/phone-style.html)

**مثال کاربردی کامل:**

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>لینک‌های ارتباطی</title>  <style>  html {  scroll-behavior: smooth;  }    body {  font-family: Arial, sans-serif;  margin: 0;  padding: 0;  }    nav {  position: fixed;  top: 0;  width: 100%;  background: #333;  padding: 15px;  text-align: center;  }    nav a {  color: white;  text-decoration: none;  margin: 0 15px;  padding: 10px 20px;  border-radius: 5px;  transition: background 0.3s;  }    nav a:hover {  background: #555;  }    section {  padding: 100px 50px;  min-height: 500px;  }    #home { background: #f0f0f0; }  #contact { background: #e0e0e0; }    .contact-links {  display: flex;  gap: 20px;  flex-wrap: wrap;  }    .contact-links a {  display: inline-block;  padding: 15px 25px;  background: #007bff;  color: white;  text-decoration: none;  border-radius: 5px;  transition: background 0.3s;  }    .contact-links a:hover {  background: #0056b3;  }  </style>  </head>  <body>  <nav>  <a href="#home">خانه</a>  <a href="#contact">تماس با ما</a>  </nav>    <section id="home">  <h1>خانه</h1>  <p>به سایت ما خوش آمدید...</p>  </section>    <section id="contact">  <h2>تماس با ما</h2>  <div class="contact-links">  <a href="mailto:info@company.com?subject=سوال&body=سلام">ارسال ایمیل</a>  <a href="tel:+982112345678">تماس تلفنی</a>  <a href="https://wa.me/989121234567" target="\_blank">واتساپ</a>  </div>  </section>  </body>  </html> |

[**real**](webdesign/html/tags/a/real.html)

## [معرفی تگ img و display آن](webdesign/html/tags/img)

تگ `<img>` یکی از اساسی‌ترین تگ‌ها در HTML برای نمایش تصاویر در صفحات وب است. این تگ یک تگ تکی است و تگ بسته ندارد.

|  |  |  |
| --- | --- | --- |
| ویژگی (Attribute) | توضیح | اهمیت |
| src` | مسیر فایل تصویر (URL) را مشخص می‌کند | اجباری |
| alt` | متن جایگزین برای توصیف تصویر | ضروری برای دسترسی پذیری و سئو |
| width`/`height` | تعیین عرض و ارتفاع تصویر (به پیکسل) | اختیاری، برای کنترل layout |
| `title` | ارائه اطلاعات اضافی (نمایش در tooltip) | اختیاری |

مثال:

|  |
| --- |
| <img src="path/to/image.jpg" alt="توضیح تصویر"> |

[sample](webdesign/html/tags/img/images/image.jpg)

ویژگی‌های (Attributes) مهم تگ <img>

ویژگی src (اجباری)

مشخص‌کننده مسیر تصویر

|  |
| --- |
| <img src="images/photo.jpg">  <img src="https://example.com/image.png"> |

ویژگی alt (بسیار مهم)

متن جایگزین برای دسترسی‌پذیری و سئو

|  |
| --- |
| <img src="logo.png" alt="لوگو شرکت ما">  <img src="cat.jpg" alt="یک گربه نارنجی در حال بازی"> |

[alt](webdesign/html/tags/img/alt.html)

معرفی تصاویر (vector & raster) و بررسی انواع آن ها

تصاویر دیجیتال عموماً در دو دسته اصلی رستر و وکتور قرار می‌گیرند که هر کدام ویژگی‌ها و کاربردهای متفاوتی دارند.

|  |  |  |
| --- | --- | --- |
| ویژگی | تصاویر رستر (Raster) | تصاویر وکتور (Vector) |
| ساختار | شبکه‌ای از پیکسل‌ها (نقاط رنگی) | مبتنی بر معادلات و نقاط ریاضی |
| مقیاس‌پذیری | با بزرگ‌نمایی، کیفیت از دست می‌رود و پیکسل‌ها نمایان می‌شوند | بی‌نهایت - بدون افت کیفیت در هر اندازه‌ای |
| کاربردهای متداول | عکاسی، طراحی وب، گرافیک‌های پیچیده با رنگ‌های متنوع | طراحی لوگو، آیکون، تصویرسازی، فونت‌ها |
| فرمت‌های رایج | `JPEG`، `PNG`، `GIF`، `WebP` | `SVG`، `AI` (ادوبی ایلاستریتور)، `EPS` |
| حجم فایل | معمولاً سنگین‌تر (به ویژه در رزولوشن‌های بالا) | معمولاً سبک‌تر |

بررسی فرمت های عکس ها

فرمت‌های مختلفی برای ذخیره‌سازی تصاویر وجود دارد که هر کدام نقاط قوت و ضعف خاص خود را دارند.

- JPEG (یا JPG): این فرمت برای عکس‌هایی با رنگ‌های طبیعی و گرادیان‌های پیچیده (مانند عکس‌های واقعی) بسیار مناسب است. JPEG از فشرده‌سازی "Lossy" استفاده می‌کند؛ یعنی برای کاهش حجم فایل، بخشی از اطلاعات تصویر را حذف می‌کند. این فرمت برای تصاویری که متن بخش مهمی از آن‌هاست یا نیاز به پس‌زمینه شفاف دارند، مناسب نیست.

- PNG: این فرمت از فشرده‌سازی "بی‌اتلاف" (Lossless) پشتیبانی می‌کند، بنابراین کیفیت تصویر حفظ می‌شود. PNG از پس‌زمینه شفاف (Transparent) پشتیبانی کامل می‌کند و برای نمایش متن و گرافیک‌هایی با لبه‌های واضح (مانند لوگوها و آیکون‌ها) ایده‌آل است. حجم فایل‌های PNG معمولاً از JPEG بیشتر است.

- GIF: این فرمت تنها از ۲۵۶ رنگ پشتیبانی می‌کند، بنابراین برای عکس‌های واقعی مناسب نیست. با این حال، از آن برای ایجاد تصاویر متحرک ساده و با حجم کم استفاده می‌شود. GIF نیز مانند PNG از شفافیت پشتیبانی می‌کند.

- SVG: یک فرمت وکتوری است که برای لوگوها، آیکون‌ها و نمودارها ایده‌آل است. تصاویر SVG در هر اندازه‌ای بدون کاهش کیفیت نمایش داده می‌شوند و معمولاً حجم فایل بسیار کمی دارند. این فرمت برای صفحه‌نمایش‌های با وضوح بالا (Retina) گزینه‌ای عالی است.

- WebP: یک فرمت مدرن است که توسط گوگل توسعه یافته و هدف آن جایگزینی برای JPEG و PNG با حفظ کیفیت و حجم کمتر است. اگرچه پشتیبانی مرورگرها از آن روز به روز در حال بهبود است، اما هنوز به طور جهانی در همه جا پشتیبانی نمی‌شود.

بررسی انواع color mode

حالت‌های رنگ، سیستم‌هایی برای سازماندهی و نمایش رنگ‌ها هستند که انتخاب آن‌ها به رسانه خروجی (مانند نمایشگر یا چاپ) بستگی دارد.

- RGB (قرمز، سبز، آبی): یک حالت رنگ افزودنی است که در آن رنگ‌ها از طریق ترکیب نورهای قرمز، سبز و آبی ساخته می‌شوند. این حالت به طور انحصاری برای خروجی‌های دیجیتال مانند صفحات نمایش، طراحی وب و اپلیکیشن‌ها استفاده می‌شود.

- CMYK (فیروزه‌ای، ارغوانی، زرد، مشکی): یک حالت رنگ تفریقی است که در آن رنگ‌ها با ترکیب رنگ‌های فیروزه‌ای، ارغوانی، زرد و مشکی ایجاد می‌شوند. از این حالت برای چاپ (مانند پوستر، بروشور و بسته‌بندی) استفاده می‌شود. هنگام آماده‌سازی طرح برای چاپ، باید فایل را به حالت CMYK تبدیل کنید.

- سایر حالت‌ها: حالت‌های دیگری مانند Grayscale (سایه‌های خاکستری)، Indexed Color (محدود به ۲۵۶ رنگ برای کاهش حجم فایل) و Bitmap (فقط سیاه و سفید خالص) برای کاربردهای خاص تری مورد استفاده قرار می‌گیرند.

معرفی mimetype

MIME Type (که قبلاً با نام Multipurpose Internet Mail Extensions شناخته می‌شد) یک استاندارد برای شناسایی نوع و قالب محتوای یک فایل در اینترنت است. هنگامی که یک مرورگر وب از سرور درخواست فایلی می‌کند، سرور نوع فایل (مثلاً تصویر، استایل‌شیت یا جاوااسکریپت) را در هدر `Content-Type` پاسخ مشخص می‌کند. این به مرورگر می‌گوید که چگونه داده‌های دریافتی را تفسیر و پردازش کند.

ساختار MIME Type به شکل `type/subtype` است. برای مثال:

- تصاویر: `image/jpeg`، `image/png`، `image/svg+xml`

- متن: `text/html` (برای صفحات HTML)، `text/css` (برای فایل‌های CSS)، `text/javascript` (برای فایل‌های جاوااسکریپت)

- کاربردهای دیگر: `application/pdf` (برای فایل‌های PDF)، `application/json` (برای داده‌های JSON)

## [بررسی تفاوت display های block و inline و inline-block](webdesign/html/inline-block)

1. Display: Block (بلوکی)

- کل عرض موجود را اشغال می‌کند

- شروع خط جدید - المان بعدی در خط جدید قرار می‌گیرد

- قابل تنظیم width و height

- padding و margin از چهار طرف اعمال می‌شود

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>Display Block</title>  <style>  .block-example {  display: block;  width: 80%;  height: 100px;  background: #3498db;  color: white;  padding: 20px;  margin: 10px 0;  border: 2px solid #2980b9;  }  </style>  </head>  <body>  <div class="block-example">بلاک ۱ - کل عرض را می‌گیرد</div>  <div class="block-example">بلاک ۲ - در خط جدید قرار می‌گیرد</div>  <span style="background: yellow;">این اسپان بعد از بلاک ها می‌آید</span>  </body>  </html> |

[block](webdesign/html/inline-block/block.html)

المان‌های block به صورت پیش‌فرض:

|  |
| --- |
| <div>, <p>, <h1>-<h6>, <section>, <article>, <header>, <footer>, <ul>, <ol>, <li> |

Display: Inline (درون خطی)

- فقط به اندازه محتوا فضا اشغال می‌کند

- شروع خط جدید نمی‌کند

width و height قابل تنظیم نیستند

- padding و margin فقط از چپ و راست اعمال می‌شود

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>Display Inline</title>  <style>  .inline-example {  display: inline;  background: #e74c3c;  color: white;  padding: 10px; /\* فقط چپ و راست \*/  margin: 0 15px; /\* فقط چپ و راست \*/  /\* width و height بی اثر هستند \*/  }  </style>  </head>  <body>  <p>  متن معمولی  <span class="inline-example">این یک المان inline است</span>  <span class="inline-example">این یکی دیگر است</span>  و ادامه متن در همان خط  </p>  </body>  </html> |

[inline](webdesign/html/inline-block/inline.html)

المان‌های inline به صورت پیش‌فرض:

|  |
| --- |
| <span>, <a>, <strong>, <em>, <img>, <button>, <input> |

Display: Inline-Block (درون خطی-بلوکی)

- ترکیب ویژگی‌های inline و block

- شروع خط جدید نمی‌کند (مانند inline)

- قابل تنظیم width و height (مانند block)

- padding و margin از چهار طرف اعمال می‌شود

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>Display Inline-Block</title>  <style>  .inline-block-example {  display: inline-block;  width: 120px;  height: 60px;  background: #27ae60;  color: white;  padding: 10px;  margin: 5px;  text-align: center;  line-height: 60px;  border: 2px solid #219652;  }  </style>  </head>  <body>  <div class="inline-block-example">آیتم ۱</div>  <div class="inline-block-example">آیتم ۲</div>  <div class="inline-block-example">آیتم ۳</div>  <div class="inline-block-example">آیتم ۴</div>  </body>  </html> |

[Inline-block](webdesign/html/inline-block/inline-block.html)

مقایسه کامل Display ها

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | Block | Inline | Inline-Block |
| شروع خط جدید | ✅ بله | ❌ خیر | ❌ خیر |
| عرض و ارتفاع | ✅ قابل تنظیم | ❌ غیرقابل تنظیم | ✅ قابل تنظیم |
| Padding | ✅ چهار طرف | ✅ فقط چپ/راست | ✅ چهار طرف |
| Margin | ✅ چهار طرف | ✅ فقط چپ/راست | ✅ چهار طرف |
| همراه با متن | ❌ خیر | ✅ بله | ✅ بله |

## [معرفی تگ table و display آن](webdesign/html/tags/table)

ساختار پایه جدول:

|  |
| --- |
| <table>  <!-- محتوای جدول -->  </table> |

معرفی تگ های tr و th و td

`<tr>` (Table Row)

- تعریف یک سطر در جدول

|  |
| --- |
| <tr>  <!-- سلول های سطر -->  </tr> |

`<td>` (Table Data)

- تعریف یک سلول معمولی (داده)

|  |
| --- |
| <td>محتویات سلول</td> |

`<th>` (Table Header)

- تعریف یک سلول هدر (تیتر)

به صورت Bold نمایش داده می‌شود.

|  |
| --- |
| <th>عنوان ستون</th> |

مثال کامل:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>جدول ساده</title>  <style>  table {  width: 100%;  border-collapse: collapse;  margin: 20px 0;  font-family: Arial, sans-serif;  }    th, td {  border: 1px solid #ddd;  padding: 12px;  text-align: right;  }    th {  background-color: #3498db;  color: white;  font-weight: bold;  }    tr:nth-child(even) {  background-color: #f2f2f2;  }    tr:hover {  background-color: #e3f2fd;  }  </style>  </head>  <body>  <table>  <tr>  <th>ردیف</th>  <th>نام محصول</th>  <th>قیمت</th>  <th>تعداد</th>  </tr>  <tr>  <td>۱</td>  <td>لپ تاپ</td>  <td>۱۵,۰۰۰,۰۰۰ تومان</td>  <td>۱۰</td>  </tr>  <tr>  <td>۲</td>  <td>موبایل</td>  <td>۸,۵۰۰,۰۰۰ تومان</td>  <td>۲۵</td>  </tr>  <tr>  <td>۳</td>  <td>تبلت</td>  <td>۶,۲۰۰,۰۰۰ تومان</td>  <td>۱۵</td>  </tr>  </table>  </body>  </html> |

[complate](webdesign/html/tags/table/complete.html)

معرفی Attribute colspan & rowspan

colspan (ادغام افقی)

- ادغام چند سلول در یک سطر

|  |
| --- |
| <table border="1" style="width: 100%;">  <tr>  <th colspan="2">اطلاعات شخصی</th>  <th colspan="2">اطلاعات تماس</th>  </tr>  <tr>  <td>نام</td>  <td>فامیلی</td>  <td>ایمیل</td>  <td>تلفن</td>  </tr>  <tr>  <td>علی</td>  <td>رضایی</td>  <td>ali@example.com</td>  <td>۰۹۱۲۱۲۳۴۵۶۷</td>  </tr>  </table> |

[colspan](webdesign/html/tags/table/colspan.html)

rowspan (ادغام عمودی)

- ادغام چند سلول در یک ستون

|  |
| --- |
| <table border="1" style="width: 100%;">  <tr>  <th rowspan="2">ردیف</th>  <th colspan="2">نام</th>  <th rowspan="2">سن</th>  </tr>  <tr>  <th>نام کوچک</th>  <th>نام خانوادگی</th>  </tr>  <tr>  <td>۱</td>  <td>محمد</td>  <td>کریمی</td>  <td>۲۵</td>  </tr>  <tr>  <td>۲</td>  <td>فاطمه</td>  <td>احمدی</td>  <td>۳۰</td>  </tr>  </table> |

[rowspan](webdesign/html/tags/table/rowspan.html)

مثال ترکیبی:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>ادغام سلول ها</title>  <style>  .complex-table {  width: 100%;  border-collapse: collapse;  }  .complex-table th, .complex-table td {  border: 1px solid #333;  padding: 10px;  text-align: center;  }  .complex-table th {  background: #2c3e50;  color: white;  }  .merged-cell {  background: #3498db;  color: white;  font-weight: bold;  }  </style>  </head>  <body>  <table class="complex-table">  <tr>  <th rowspan="2">شهر</th>  <th colspan="2">جمعیت</th>  <th rowspan="2">مساحت (km²)</th>  </tr>  <tr>  <th>مرد</th>  <th>زن</th>  </tr>  <tr>  <td>تهران</td>  <td>۴,۵۰۰,۰۰۰</td>  <td>۴,۳۰۰,۰۰۰</td>  <td>۷۳۰</td>  </tr>  <tr>  <td>مشهد</td>  <td>۱,۸۰۰,۰۰۰</td>  <td>۱,۷۵۰,۰۰۰</td>  <td>۳۵۰</td>  </tr>  <tr>  <td colspan="3" class="merged-cell">جمع کل</td>  <td>۱۲,۳۵۰,۰۰۰</td>  </tr>  </table>  </body>  </html> |

[both](webdesign/html/tags/table/both.html)

معرفی تگ caption در table

- اضافه کردن عنوان به جدول

- باید بلافاصله بعد از تگ `<table>` قرار گیرد

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>جدول با عنوان</title>  <style>  .captioned-table {  width: 80%;  margin: 20px auto;  border-collapse: collapse;  }  .captioned-table caption {  font-size: 1.5em;  font-weight: bold;  color: #2c3e50;  margin-bottom: 10px;  caption-side: top;  }  .captioned-table th, .captioned-table td {  border: 1px solid #bdc3c7;  padding: 8px;  text-align: center;  }  .captioned-table th {  background: #34495e;  color: white;  }  </style>  </head>  <body>  <table class="captioned-table">  <caption>لیست کارمندان شرکت - سال ۱۴۰۳</caption>  <tr>  <th>کد پرسنلی</th>  <th>نام و نام خانوادگی</th>  <th>سمت</th>  <th>حقوق (تومان)</th>  </tr>  <tr>  <td>۱۰۱</td>  <td>علی محمدی</td>  <td>مدیر فنی</td>  <td>۲۵,۰۰۰,۰۰۰</td>  </tr>  <tr>  <td>۱۰۲</td>  <td>فاطمه کریمی</td>  <td>برنامه نویس</td>  <td>۱۸,۰۰۰,۰۰۰</td>  </tr>  </table>  </body>  </html> |

[caption](webdesign/html/tags/table/caption.html)

معرفی تگ های thead و tbody و tfoot

ساختار حرفه‌ای جدول:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>جدول حرفه‌ای</title>  <style>  .professional-table {  width: 90%;  margin: 20px auto;  border-collapse: collapse;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  font-family: Arial, sans-serif;  }    /\* استایل سر جدول \*/  .professional-table thead {  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  color: white;  }    .professional-table thead th {  padding: 15px;  font-size: 1.1em;  text-align: center;  }    /\* استایل بدنه جدول \*/  .professional-table tbody tr {  transition: background-color 0.3s ease;  }    .professional-table tbody tr:nth-child(even) {  background-color: #f8f9fa;  }    .professional-table tbody tr:nth-child(odd) {  background-color: #ffffff;  }    .professional-table tbody tr:hover {  background-color: #e3f2fd;  cursor: pointer;  }    .professional-table tbody td {  padding: 12px;  border-bottom: 1px solid #e0e0e0;  text-align: center;  }    /\* استایل پاورقی جدول \*/  .professional-table tfoot {  background-color: #2c3e50;  color: white;  font-weight: bold;  }    .professional-table tfoot td {  padding: 15px;  text-align: center;  }    .highlight {  background-color: #fff3cd !important;  color: #856404;  }  </style>  </head>  <body>  <table class="professional-table">  <caption>گزارش مالی سه ماهه اول ۱۴۰۳</caption>    <!-- سر جدول -->  <thead>  <tr>  <th>ماه</th>  <th>فروش (میلیون تومان)</th>  <th>هزینه (میلیون تومان)</th>  <th>سود خالص (میلیون تومان)</th>  <th>رشد (%)</th>  </tr>  </thead>    <!-- پاورقی جدول -->  <tfoot>  <tr>  <td>جمع کل</td>  <td>۲,۸۵۰</td>  <td>۱,۹۰۰</td>  <td>۹۵۰</td>  <td>+۱۸٪</td>  </tr>  </tfoot>    <!-- بدنه جدول -->  <tbody>  <tr>  <td>فروردین</td>  <td>۸۰۰</td>  <td>۶۰۰</td>  <td>۲۰۰</td>  <td>+۵٪</td>  </tr>  <tr class="highlight">  <td>اردیبهشت</td>  <td>۱,۱۰۰</td>  <td>۷۰۰</td>  <td>۴۰۰</td>  <td>+۲۵٪</td>  </tr>  <tr>  <td>خرداد</td>  <td>۹۵۰</td>  <td>۶۰۰</td>  <td>۳۵۰</td>  <td>+۱۵٪</td>  </tr>  </tbody>  </table>  </body>  </html> |

[thead-tfoot-tbody](webdesign/html/tags/table/thead-tfoot-tbody.html)

مثال نهایی: جدول پیشرفته

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>جدول پیشرفته</title>  <style>  .advanced-table {  width: 95%;  margin: 30px auto;  border-collapse: collapse;  box-shadow: 0 4px 15px rgba(0,0,0,0.1);  border-radius: 10px;  overflow: hidden;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }    .advanced-table caption {  font-size: 1.8em;  font-weight: bold;  color: #2c3e50;  margin: 20px 0;  padding: 15px;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  color: white;  border-radius: 8px 8px 0 0;  }    .advanced-table thead {  background: #34495e;  }    .advanced-table thead th {  padding: 18px;  font-size: 1.1em;  text-align: center;  color: white;  border-bottom: 3px solid #2980b9;  }    .advanced-table tbody tr {  transition: all 0.3s ease;  }    .advanced-table tbody tr:nth-child(even) {  background: #f8f9fa;  }    .advanced-table tbody tr:nth-child(odd) {  background: #ffffff;  }    .advanced-table tbody tr:hover {  background: #e3f2fd;  transform: scale(1.01);  box-shadow: 0 2px 8px rgba(0,0,0,0.1);  }    .advanced-table tbody td {  padding: 15px;  text-align: center;  border-bottom: 1px solid #ecf0f1;  transition: all 0.3s ease;  }    .advanced-table tbody tr:hover td {  border-bottom-color: #3498db;  }    .advanced-table tfoot {  background: #2c3e50;  color: white;  font-weight: bold;  }    .advanced-table tfoot td {  padding: 18px;  text-align: center;  font-size: 1.1em;  }    .positive {  color: #27ae60;  font-weight: bold;  }    .negative {  color: #e74c3c;  font-weight: bold;  }    .highlight {  background: #fff3cd !important;  border-left: 4px solid #ffc107;  }    /\* ریسپانسیو برای موبایل \*/  @media (max-width: 768px) {  .advanced-table {  font-size: 0.9em;  }    .advanced-table thead {  display: none;  }    .advanced-table tbody td {  display: block;  text-align: left;  }    .advanced-table tbody td:before {  content: attr(data-label);  font-weight: bold;  display: inline-block;  width: 120px;  color: #2c3e50;  }  }  </style>  </head>  <body>  <table class="advanced-table">  <caption>کارنامه دانشجویان ترم بهار ۱۴۰۳</caption>    <thead>  <tr>  <th>شماره دانشجویی</th>  <th>نام و نام خانوادگی</th>  <th>رشته تحصیلی</th>  <th>معدل</th>  <th>وضعیت</th>  <th>عملیات</th>  </tr>  </thead>    <tbody>  <tr>  <td data-label="شماره دانشجویی">۹۸۱۲۳۴۵</td>  <td data-label="نام">علی رضایی</td>  <td data-label="رشته">مهندسی کامپیوتر</td>  <td data-label="معدل" class="positive">۱۸.۷۵</td>  <td data-label="وضعیت" class="positive">✅ قبول</td>  <td data-label="عملیات">  <button style="padding: 5px 10px; background: #3498db; color: white; border: none; border-radius: 3px; cursor: pointer;">  مشاهده جزئیات  </button>  </td>  </tr>    <tr class="highlight">  <td data-label="شماره دانشجویی">۹۸۱۲۳۴۶</td>  <td data-label="نام">فاطمه محمدی</td>  <td data-label="رشته">مهندسی برق</td>  <td data-label="معدل" class="positive">۱۹.۲۵</td>  <td data-label="وضعیت" class="positive">✅ قبول</td>  <td data-label="عملیات">  <button style="padding: 5px 10px; background: #3498db; color: white; border: none; border-radius: 3px; cursor: pointer;">  مشاهده جزئیات  </button>  </td>  </tr>    <tr>  <td data-label="شماره دانشجویی">۹۸۱۲۳۴۷</td>  <td data-label="نام">محمد کریمی</td>  <td data-label="رشته">مهندسی عمران</td>  <td data-label="معدل" class="negative">۱۱.۸۰</td>  <td data-label="وضعیت" class="negative">❌ مشروط</td>  <td data-label="عملیات">  <button style="padding: 5px 10px; background: #e74c3c; color: white; border: none; border-radius: 3px; cursor: pointer;">  اخطار تحصیلی  </button>  </td>  </tr>  </tbody>    <tfoot>  <tr>  <td colspan="6">تعداد کل دانشجویان: ۳ نفر | میانگین معدل: ۱۶.۶۰</td>  </tr>  </tfoot>  </table>  </body>  </html> |

[final](webdesign/html/tags/table/final.html)

## [بررسی چگونگی عمکرد form ها](webdesign/html/tags/form)

فرم‌ها در HTML ابزارهایی برای جمع‌آوری اطلاعات از کاربران هستند و داده‌ها را برای پردازش به سرور ارسال می‌کنند. درک نحوه عملکرد و اجزای تشکیل‌دهنده آنها برای ایجاد صفحات وب تعاملی ضروری است.

تگ `<form>` و ویژگی‌های اصلی

تگ `<form>` به عنوان کانتینر اصلی، تمامی عناصر فرم را در بر می‌گیرد و با ویژگی‌های خود، نحوه ارسال داده‌ها را کنترل می‌کند.

|  |
| --- |
| <form action="/submit.php" method="post" enctype="multipart/form-data">  <!-- عناصر فرم در اینجا قرار می‌گیرند -->  </form> |

- `action`: آدرس URLای را مشخص می‌کند که داده‌های فرم پس از ارسال، به آنجا فرستاده می‌شوند. در صورتی که این ویژگی مشخص نشود، داده‌ها به همان صفحه فعلی ارسال می‌گردند.

- `method`: روش HTTP مورد استفاده برای ارسال داده‌ها را تعیین می‌کند که عمدتاً `GET` یا `POST` است.

- `enctype`: نحوه کدبندی داده‌ها را هنگام ارسال مشخص می‌کند. این ویژگی به ویژه زمانی که فرم شامل آپلود فایل است (`type="file"`) و باید روی `multipart/form-data` تنظیم شود، اهمیت دارد.

معرفی تگ form و attribute های action و method و enctype

بررسی method های get و post

|  |  |  |
| --- | --- | --- |
| ویژگی | روش GET | روش POST |
| کاربرد | دریافت داده (عمدتاً جستجو) | ارسال داده (ثبت نام، ورود) |
| نمایش داده | در نوار آدرس مرورگر قابل مشاهده است | در نوار آدرس نمایش داده نمی‌شود |
| امنیت | برای داده‌های حساس مناسب نیست | از امنیت بیشتری برخوردار است |
| محدودیت حجم | دارد (حدود ۲۰۴۸ کاراکتر) | ندارد |
| قابلیت بوکمارک | دارد | ندارد |

معرفی تگ input و انواع type های آن

Text ,password, search, email, url, tel, submit, reset, button,radio,checkbox, file,hidden,date, time, number, range, color,

تگ `<input>` پرکاربردترین عنصر برای دریافت ورودی از کاربر است و رفتار آن به شدت تحت تأثیر ویژگی `type` آن قرار می‌گیرد.

|  |
| --- |
| <input type="text" name="username" placeholder="نام کاربری خود را وارد کنید">  <input type="password" name="password" placeholder="رمز عبور">  <input type="email" name="email" required>  <input type="submit" value="ارسال اطلاعات"> |

انواع رایج `type`ها:

- `text`: برای دریافت متن تک خطی

- `password`: برای دریافت رمز عبور (کاراکترها پنهان می‌شوند)

- `email`, `url`, `tel`: برای دریافت ایمیل، آدرس وب و شماره تلفن (برخی مرورگرها اعتبارسنجی اولیه انجام می‌دهند)

- `submit`, `reset`, `button`: برای ایجاد دکمه‌های ارسال، بازنشانی و یک دکمه معمولی

- `radio`: برای انتخاب یک گزینه از چند گزینه (همه دکمه‌های رادیویی یک نام مشترک دارند)

- `checkbox`: برای انتخاب یک یا چند گزینه از چند گزینه

- `file`: برای انتخاب و آپلود فایل

- `hidden`: برای ارسال داده‌های پنهان (مخفی از کاربر)

- `date`, `time`, `number`, `range`, `color`: برای دریافت انواع داده‌های خاص و نمایش ویجت‌های مناسب

معرفی attribute های

Value, name, placeholder, min-length, max-length, required, diabled, readonly,

در ادامه، به معرفی و بررسی کامل تگ‌ها و ویژگی‌های (Attributes) مورد نظر شما می‌پردازیم. این اطلاعات به شما کمک می‌کند فرم‌های حرفه‌ای، دسترسی‌پذیر و خوش‌ساخت در HTML ایجاد کنید.

معرفی ویژگی‌های (Attributes) پرکاربرد

ویژگی‌های زیر رفتار و کارکرد عناصر فرم، به ویژه تگ `<input>` را کنترل می‌کنند:

|  |  |
| --- | --- |
| ویژگی | توضیح و کاربرد |
| `value` | مقدار اولیه و پیش‌فرض فیلد را مشخص می‌کند. این مقدار قابل تغییر توسط کاربر است و هنگام ارسال فرم به سرور فرستاده می‌شود. |
| name` | نام فیلد را مشخص می‌کند. این نام، هنگام ارسال فرم، به عنوان کلید برای داده‌ها استفاده می‌شود و اهمیت بسیار زیادی دارد. اگر فیلدی `name` نداشته باشد، مقدار آن ارسال نخواهد شد. |
| placeholder` | یک متن راهنما به صورت کمرنگ در فیلد نمایش می‌دهد تا کاربر بداند چه اطلاعاتی وارد کند. این متن با شروع تایپ کاربر ناپدید می‌شود و هرگز جایگزین برچسب (`<label>`) نمی‌شود. |
| required` | فیلد را به یک فیلد اجباری تبدیل می‌کند. کاربر باید پیش از ارسال فرم، این فیلد را پر کند. |
| disabled` | فیلد را غیرفعال می‌کند. فیلد غیرفعال، قابل کلیک یا فوکوس نیست و مقدار آن نیز ارسال نمی‌شود. |
| readonly` | فیلد را فقط-خواندنی می‌کند. کاربر نمی‌تواند مقدار آن را تغییر دهد، اما فیلد قابل فوکوس است و مقدار آن ارسال می‌شود. |
| maxlength` | حداکثر تعداد کاراکتر مجاز برای ورود در فیلدهای متنی را مشخص می‌کند. |

نکته مهم: تفاوت اصلی `disabled` و `readonly` در این است که مقدار فیلدهای `readonly` هنگام ارسال فرم به سرور فرستاده می‌شود، اما مقدار فیلدهای `disabled` ارسال نمی‌شود.

معرفی تگ textarea و display آن

از این تگ برای ایجاد یک کادر متنی چندخطی استفاده می‌شود که برای دریافت نظرات، آدرس یا هر متن طولانی از کاربر مناسب است.

- نمایش (Display): این تگ به صورت inline-block` نمایش داده می‌شود.

- ویژگی‌های کلیدی:

- rows`: تعداد خطوط قابل مشاهده را مشخص می‌کند.

- cols`: عرض کادر متن را بر اساس تعداد کاراکترها مشخص می‌کند.

- resize`: به صورت پیش‌فرض، کاربر می‌تواند اندازه `textarea` را با ماوس تغییر دهد. با استفاده از CSS می‌توان این قابلیت را غیرفعال کرد: `resize: none;`.

|  |
| --- |
| <label for="message">پیام شما:</label>  <textarea id="message" name="user\_message" rows="4" cols="50" placeholder="متن پیام خود را اینجا بنویسید..."></textarea> |

معرفی تگ Select و option و optgroup

از این تگ‌ها برای ایجاد فهرست‌های کشویی استفاده می‌شود.

- `<select>`: کانتینر اصلی برای ایجاد فهرست کشویی است. ویژگی `name` آن برای ارسال داده ضروری است.

- <option>`: برای تعریف هر یک از گزینه‌ها درون تگ `<select>` استفاده می‌شود. ویژگی `value` مشخص می‌کند که در صورت انتخاب آن گزینه، چه مقداری به سرور ارسال شود.

- <optgroup>`: برای دسته‌بندی کردن گزینه‌های مرتبط درون فهرست استفاده می‌شود. ویژگی `label` آن، عنوان دسته را مشخص می‌کند.

|  |
| --- |
| <label for="car">انتخاب خودرو:</label>  <select id="car" name="car">  <optgroup label="خودروهای ایرانی">  <option value="dena">دنا</option>  <option value="peugeot">پژو</option>  </optgroup>  <optgroup label="خودروهای خارجی">  <option value="benz">مرسدس بنز</option>  <option value="bmw">بی ام و</option>  </optgroup>  </select> |

معرفی تگ label و display آن

تگ `<label>` یک برچسب متنی برای عناصر فرم ایجاد می‌کند. استفاده از آن دو مزیت بزرگ دارد:

1. افزایش دسترسی‌پذیری (Accessibility): نرم‌افزارهای صفحه‌خوان (Screen Readers) متن برچسب را برای کاربران نابینا می‌خوانند.

2. تجربه کاربری بهتر: با کلیک روی متن برچسب، خود فیلد مرتبط، فوکوس یا فعال می‌شود که به ویژه برای المان‌های کوچک مانند چک‌باکس‌ها مفید است.

- نمایش (Display): این تگ به صورت inline` نمایش داده می‌شود.

اتصال `<label>` به `<input>` با دو روش انجام می‌شود:

اتصال صریح (روش ترجیحی): با استفاده از ویژگی `for` در `<label>` و `id` در `<input>`.

|  |
| --- |
| <label for="username">نام کاربری:</label>  <input type="text" id="username" name="username">  <!-- با کلیک روی متن "نام کاربری"، کرسر در فیلد ورودی قرار می‌گیرد --> |

. اتصال ضمنی: با قرار دادن تگ `<input>` درون تگ `<label>`.

|  |
| --- |
| <label>  <input type="checkbox" name="agree"> با شرایط موافقم  </label>  <!-- با کلیک روی متن "با شرایط موافقم"، چک‌باکس فعال یا غیرفعال می‌شود --> |

معرفی تگ fieldset و legend

از این تگ‌ها برای گروه‌بندی بصری و معنایی چندین عنصر مرتبط در یک فرم استفاده می‌شود. این کار درک و پیمایش فرم را برای همه کاربران، به ویژه کاربران صفحه‌خوان، بسیار آسان‌تر می‌کند.

- `<fieldset>`: به عنوان کانتینر اصلی، دور عناصر مرتبط یک کادر می‌کشد.

- <legend>`: که باید درون `<fieldset>` قرار گیرد، عنوانی برای آن گروه مشخص می‌کند.

|  |
| --- |
| <form>  <fieldset>  <legend>اطلاعات تماس</legend>  <label for="phone">تلفن:</label>  <input type="tel" id="phone" name="phone"><br>  <label for="email">ایمیل:</label>  <input type="email" id="email" name="email">  </fieldset>  </form> |

یک فرم کامل

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>فرم نمونه جامع</title>  <style>  /\* استایل‌های عمومی و ریست \*/  \* {  box-sizing: border-box;  margin: 0;  padding: 0;  }    body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  line-height: 1.6;  color: #333;  background-color: #f4f4f4;  padding: 20px;  }    .form-container {  max-width: 800px;  margin: 0 auto;  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 0 20px rgba(0,0,0,0.1);  }    h1 {  text-align: center;  color: #2c3e50;  margin-bottom: 30px;  font-size: 2.2em;  }    h2 {  color: #3498db;  margin: 25px 0 15px 0;  padding-bottom: 10px;  border-bottom: 2px solid #ecf0f1;  }    /\* استایل‌های عمومی فرم \*/  .form-group {  margin-bottom: 20px;  }    label {  display: block;  margin-bottom: 8px;  font-weight: bold;  color: #2c3e50;  }    .required::after {  content: " \*";  color: #e74c3c;  }    /\* استایل‌های فیلدهای ورودی \*/  input[type="text"],  input[type="password"],  input[type="email"],  input[type="url"],  input[type="tel"],  input[type="number"],  input[type="date"],  input[type="time"],  input[type="search"],  select,  textarea {  width: 100%;  padding: 12px;  border: 2px solid #bdc3c7;  border-radius: 6px;  font-size: 16px;  transition: all 0.3s ease;  background-color: #fff;  }    input:focus,  select:focus,  textarea:focus {  outline: none;  border-color: #3498db;  box-shadow: 0 0 8px rgba(52, 152, 219, 0.3);  }    /\* استایل برای گروه‌بندی فیلدها \*/  .fieldset-group {  border: 2px solid #ecf0f1;  border-radius: 8px;  padding: 20px;  margin-bottom: 25px;  background-color: #f8f9fa;  }    legend {  font-weight: bold;  color: #2c3e50;  padding: 0 10px;  font-size: 1.1em;  }    /\* استایل‌های خاص برای انواع مختلف ورودی \*/  .input-row {  display: flex;  gap: 15px;  margin-bottom: 15px;  }    .input-group {  flex: 1;  }    /\* استایل رادیو و چک‌باکس \*/  .radio-group,  .checkbox-group {  margin: 10px 0;  }    input[type="radio"],  input[type="checkbox"] {  margin-left: 10px;  }    /\* استایل رنج \*/  input[type="range"] {  width: 100%;  height: 8px;  margin: 10px 0;  }    .range-value {  text-align: center;  font-weight: bold;  color: #3498db;  }    /\* استایل فایل \*/  input[type="file"] {  width: 100%;  padding: 10px;  border: 2px dashed #bdc3c7;  border-radius: 6px;  background-color: #f8f9fa;  }    /\* استایل دکمه‌ها \*/  .button-group {  display: flex;  gap: 10px;  justify-content: flex-end;  margin-top: 30px;  }    button,  input[type="submit"],  input[type="reset"],  input[type="button"] {  padding: 12px 25px;  border: none;  border-radius: 6px;  font-size: 16px;  cursor: pointer;  transition: all 0.3s ease;  }    input[type="submit"],  button[type="submit"] {  background-color: #27ae60;  color: white;  }    input[type="submit"]:hover,  button[type="submit"]:hover {  background-color: #219652;  }    input[type="reset"] {  background-color: #e74c3c;  color: white;  }    input[type="reset"]:hover {  background-color: #c0392b;  }    input[type="button"] {  background-color: #3498db;  color: white;  }    input[type="button"]:hover {  background-color: #2980b9;  }    /\* استایل برای موبایل \*/  @media (max-width: 768px) {  .input-row {  flex-direction: column;  gap: 0;  }    .button-group {  flex-direction: column;  }    button,  input[type="submit"],  input[type="reset"],  input[type="button"] {  width: 100%;  margin-bottom: 10px;  }  }  </style>  </head>  <body>  <div class="form-container">  <h1>فرم ثبت نام جامع</h1>    <form action="/submit" method="post" enctype="multipart/form-data">    <!-- بخش اطلاعات شخصی -->  <div class="fieldset-group">  <legend>اطلاعات شخصی</legend>    <div class="input-row">  <div class="input-group">  <label for="fname" class="required">نام</label>  <input type="text" id="fname" name="first\_name" required  placeholder="نام خود را وارد کنید">  </div>  <div class="input-group">  <label for="lname" class="required">نام خانوادگی</label>  <input type="text" id="lname" name="last\_name" required  placeholder="نام خانوادگی خود را وارد کنید">  </div>  </div>    <div class="form-group">  <label for="email" class="required">ایمیل</label>  <input type="email" id="email" name="email" required  placeholder="example@domain.com">  </div>    <div class="form-group">  <label for="password" class="required">رمز عبور</label>  <input type="password" id="password" name="password" required  minlength="8" placeholder="حداقل ۸ کاراکتر">  </div>  </div>    <!-- بخش اطلاعات تماس -->  <div class="fieldset-group">  <legend>اطلاعات تماس</legend>    <div class="input-row">  <div class="input-group">  <label for="phone">تلفن همراه</label>  <input type="tel" id="phone" name="phone"  placeholder="09xxxxxxxxx">  </div>  <div class="input-group">  <label for="website">وبسایت</label>  <input type="url" id="website" name="website"  placeholder="https://example.com">  </div>  </div>    <div class="form-group">  <label for="search">جستجو</label>  <input type="search" id="search" name="search"  placeholder="متن مورد نظر برای جستجو...">  </div>  </div>    <!-- بخش اطلاعات تکمیلی -->  <div class="fieldset-group">  <legend>اطلاعات تکمیلی</legend>    <div class="input-row">  <div class="input-group">  <label for="birthdate">تاریخ تولد</label>  <input type="date" id="birthdate" name="birthdate">  </div>  <div class="input-group">  <label for="age">سن</label>  <input type="number" id="age" name="age" min="18" max="100"  placeholder="بین ۱۸ تا ۱۰۰">  </div>  </div>    <div class="form-group">  <label for="favcolor">رنگ مورد علاقه</label>  <input type="color" id="favcolor" name="favcolor" value="#3498db">  </div>    <div class="form-group">  <label for="salary">محدوده حقوق درخواستی:  <span id="salaryValue" class="range-value">50%</span>  </label>  <input type="range" id="salary" name="salary" min="0" max="100"  value="50" oninput="document.getElementById('salaryValue').textContent = this.value + '%'">  </div>  </div>    <!-- بخش انتخاب‌ها -->  <div class="fieldset-group">  <legend>انتخاب‌ها</legend>    <div class="form-group">  <label>جنسیت</label>  <div class="radio-group">  <input type="radio" id="male" name="gender" value="male">  <label for="male">مرد</label>  </div>  <div class="radio-group">  <input type="radio" id="female" name="gender" value="female">  <label for="female">زن</label>  </div>  </div>    <div class="form-group">  <label>مهارت‌ها</label>  <div class="checkbox-group">  <input type="checkbox" id="html" name="skills" value="html">  <label for="html">HTML</label>  </div>  <div class="checkbox-group">  <input type="checkbox" id="css" name="skills" value="css">  <label for="css">CSS</label>  </div>  <div class="checkbox-group">  <input type="checkbox" id="js" name="skills" value="javascript">  <label for="js">JavaScript</label>  </div>  </div>    <div class="form-group">  <label for="country">کشور</label>  <select id="country" name="country">  <option value="">لطفا انتخاب کنید</option>  <option value="ir">ایران</option>  <option value="us">آمریکا</option>  <option value="ca">کانادا</option>  <option value="uk">انگلستان</option>  </select>  </div>    <div class="form-group">  <label for="bio">بیوگرافی</label>  <textarea id="bio" name="bio" rows="4"  placeholder="درباره خودتان بنویسید..."></textarea>  </div>    <div class="form-group">  <label for="resume">ارسال رزومه</label>  <input type="file" id="resume" name="resume"  accept=".pdf,.doc,.docx">  </div>    <input type="hidden" id="user\_id" name="user\_id" value="12345">  </div>    <!-- دکمه‌های فرم -->  <div class="button-group">  <input type="reset" value="پاک کردن فرم">  <input type="button" value="ذخیره موقت"  onclick="alert('ذخیره شد!')">  <input type="submit" value="ثبت نهایی">  </div>  </form>  </div>  </body>  </html> |

[index](webdesign/html/tags/form/index.html)

## [معرفی تگ iframe](webdesign/html/tags/iframe)

تگ `<iframe>` برای نمایش یک صفحه وب درون صفحه وب دیگر استفاده می‌شود.

ویژگی‌های اصلی:

|  |
| --- |
| <iframe src="https://example.com"  width="800"  height="600"  title="نمونه iframe"  frameborder="0"  allowfullscreen>  </iframe> |

[index](webdesign/html/tags/iframe/index.html)

ویژگی‌های مهم:

- src`: آدرس صفحه‌ای که می‌خواهید نمایش دهید

- width` و `height`: اندازه قاب

- title`: برای دسترسی‌پذیری (accessibility)

- frameborder`: نمایش یا عدم نمایش حاشیه

- allowfullscreen`: اجازه نمایش تمام صفحه

استفاده از `loading="lazy"` برای iframeها باعث عملکرد بهینه میشود.

مثال‌های کاربردی:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>نمونه iframe</title>      <style>          .iframe-container {              margin: 20px 0;              border: 2px solid #3498db;              border-radius: 10px;              overflow: hidden;          }          .responsive-iframe {              width: 100%;              height: 400px;              border: none;          }          .h\_iframe-aparat\_embed\_frame {              position: relative;          }          .h\_iframe-aparat\_embed\_frame .ratio {              display: block;              width: 100%;              height: auto;          }          .h\_iframe-aparat\_embed\_frame iframe {              position: absolute;              top: 0;              left: 0;              width: 100%;              height: 100%;          }      </style>  </head>  <body>      <h1>نمونه‌های iframe</h1>      <!-- نمایش نقشه گوگل -->      <div class="iframe-container">          <iframe              src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d700.5838411288692!2d-122.4364637635371!3d45.50367163544069!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x5495988bed8ee02d%3A0x9790f57856d15574!2s970%20NW%20Eastman%20Pkwy%2C%20Gresham%2C%20OR%2097030%2C%20USA!5e0!3m2!1sen!2s!4v1759649336276!5m2!1sen!2s"              width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy"              referrerpolicy="no-referrer-when-downgrade"></iframe>          </iframe>      </div>      <!-- نمایش ویدیوی یوتیوب -->      <div class="iframe-container">          <iframe width="100%" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0"              allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"              allowfullscreen>          </iframe>      </div>      <!-- نمایش ویدیوی یوتیوب -->      <div class="h\_iframe-aparat\_embed\_frame">          <span style="display: block;padding-top: 57%"></span>          <iframe src="https://www.aparat.com/video/video/embed/videohash/hvfb02d/vt/frame" allowFullScreen="true"              webkitallowfullscreen="true" mozallowfullscreen="true">          </iframe>      </div>      <!-- iframe ریسپانسیو -->      <div style="position: relative; width: 100%; height: 0; padding-bottom: 56.25%;">          <iframe src="https://example.com" style="position: absolute; top: 0; left: 0; width: 100%; height: 100%;"              frameborder="0">          </iframe>      </div>  </body>  </html> |

[working](webdesign/html/tags/iframe/working.html)

امنیت iframe:

|  |
| --- |
| <!-- استفاده از sandbox برای افزایش امنیت -->  <iframe src="https://example.com"  sandbox="allow-scripts allow-same-origin"  allow="camera; microphone">  </iframe> |

## [معرفی تگ video و display آن](webdesign/html/tags/video)

تگ `<video>` برای نمایش ویدیو در صفحات وب استفاده می‌شود.

### Display پیش‌فرض: `inline`

ساختار پایه:

|  |
| --- |
| <video controls width="640">  <source src="movie.mp4" type="video/mp4">  <source src="movie.webm" type="video/webm">  مرورگر شما از تگ ویدیو پشتیبانی نمی‌کند.  </video> |

[base](webdesign/html/tags/video/base.html)

ویژگی‌های مهم:

- controls`: نمایش کنترل‌های پخش

- autoplay`: پخش خودکار ویدیو

- loop`: تکرار ویدیو

- muted`: بی‌صدا کردن ویدیو

- poster`: تصویر پیش‌نمایش

- width` و `height`: اندازه ویدیو

فرمت‌های پشتیبانی شده:

ویدیو: MP4, WebM, OGV

- استفاده از `preload="none"` برای صرفه‌جویی در پهنای باند

مثال کامل ویدیو:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>پخش کننده ویدیو</title>  <style>  .video-container {  max-width: 800px;  margin: 20px auto;  background: #000;  border-radius: 10px;  overflow: hidden;  box-shadow: 0 4px 15px rgba(0,0,0,0.3);  }    .custom-video {  width: 100%;  height: auto;  display: block;  }    .video-controls {  background: #333;  padding: 10px;  display: flex;  justify-content: center;  gap: 10px;  }    .control-btn {  padding: 8px 15px;  background: #3498db;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  }  </style>  </head>  <body>  <h1>پخش کننده ویدیو</h1>    <!-- ویدیو ساده با کنترل‌ها -->  <div class="video-container">  <video controls class="custom-video" poster="thumbnail.jpg">  <source src="video/sample.mp4" type="video/mp4">  <source src="video/sample.webm" type="video/webm">  مرورگر شما از تگ ویدیو پشتیبانی نمی‌کند.  </video>  </div>    <!-- ویدیو با کنترل‌های سفارشی -->  <div class="video-container">  <video id="myVideo" class="custom-video" poster="thumbnail.jpg">  <source src="video/sample.mp4" type="video/mp4">  مرورگر شما از تگ ویدیو پشتیبانی نمی‌کند.  </video>  <div class="video-controls">  <button class="control-btn" onclick="playVideo()">▶ پخش</button>  <button class="control-btn" onclick="pauseVideo()">⏸ توقف</button>  <button class="control-btn" onclick="muteVideo()">🔇 بی‌صدا</button>  <input type="range" id="volume" min="0" max="1" step="0.1" value="1" onchange="setVolume(this.value)">  </div>  </div>    <!-- ویدیو با قابلیت autoplay و loop -->  <div class="video-container">  <video autoplay muted loop class="custom-video">  <source src="video/background.mp4" type="video/mp4">  ویدیوی پس‌زمینه  </video>  </div>  <script>  const video = document.getElementById('myVideo');    function playVideo() {  video.play();  }    function pauseVideo() {  video.pause();  }    function muteVideo() {  video.muted = !video.muted;  }    function setVolume(volume) {  video.volume = volume;  }  </script>  </body>  </html> |

[Complete](webdesign/html/tags/video/complete.html)

## [معرفی تگ audio و display آن](webdesign/html/tags/audio)

تگ `<audio>` برای پخش فایل‌های صوتی استفاده می‌شود.

Display پیش‌فرض: `inline`

ساختار پایه:

|  |
| --- |
| <audio controls>  <source src="audio/sample.mp3" type="audio/mpeg">  <source src="audio/sample.ogg" type="audio/ogg">  مرورگر شما از تگ audio پشتیبانی نمی‌کند.  </audio> |

[base](webdesign/html/tags/audio/base.html)

ویژگی‌های مهم:

- `controls`: نمایش کنترل‌های پخش

- autoplay`: پخش خودکار صوت

- loop`: تکرار صوت

- muted`: بی‌صدا کردن

- preload`: پیش‌بارگذاری

فرمت‌های پشتیبانی شده:

صوت: MP3, WAV, OGG

- استفاده از `preload="none"` برای صرفه‌جویی در پهنای باند

مثال کامل audio:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>پخش کننده صوت</title>  <style>  .audio-player {  max-width: 600px;  margin: 20px auto;  padding: 20px;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  border-radius: 15px;  box-shadow: 0 4px 15px rgba(0,0,0,0.2);  color: white;  }    .track-info {  text-align: center;  margin-bottom: 15px;  }    .track-title {  font-size: 1.2em;  font-weight: bold;  margin-bottom: 5px;  }    .track-artist {  font-size: 0.9em;  opacity: 0.8;  }    .custom-audio {  width: 100%;  height: 40px;  border-radius: 20px;  }    .playlist {  margin-top: 30px;  }    .playlist-item {  padding: 10px;  margin: 5px 0;  background: rgba(255,255,255,0.1);  border-radius: 5px;  cursor: pointer;  transition: background 0.3s;  }    .playlist-item:hover {  background: rgba(255,255,255,0.2);  }    .audio-container {  display: inline-block;  margin: 10px;  vertical-align: top;  }  </style>  </head>  <body>  <h1>پخش کننده صوت</h1>    <!-- پخش کننده ساده -->  <div class="audio-player">  <div class="track-info">  <div class="track-title">آهنگ نمونه</div>  <div class="track-artist">هنرمند ناشناس</div>  </div>  <audio controls class="custom-audio">  <source src="audio/sample.mp3" type="audio/mpeg">  <source src="audio/sample.ogg" type="audio/ogg">  مرورگر شما از تگ audio پشتیبانی نمی‌کند.  </audio>  </div>    <!-- چند پخش کننده در کنار هم -->  <div style="text-align: center;">  <div class="audio-container">  <h3>آهنگ ۱</h3>  <audio controls>  <source src="audio/track1.mp3" type="audio/mpeg">  </audio>  </div>    <div class="audio-container">  <h3>آهنگ ۲</h3>  <audio controls>  <source src="audio/track2.mp3" type="audio/mpeg">  </audio>  </div>    <div class="audio-container">  <h3>آهنگ ۳</h3>  <audio controls>  <source src="audio/track3.mp3" type="audio/mpeg">  </audio>  </div>  </div>    <!-- پخش کننده پیشرفته با پلی‌لیست -->  <div class="audio-player">  <div class="track-info">  <div class="track-title" id="currentTrack">لطفا آهنگی انتخاب کنید</div>  <div class="track-artist" id="currentArtist">-</div>  </div>  <audio id="mainAudio" controls class="custom-audio">  مرورگر شما از تگ audio پشتیبانی نمی‌کند.  </audio>    <div class="playlist">  <h3>پلی‌لیست</h3>  <div class="playlist-item" onclick="playTrack('audio/track1.mp3', 'آهنگ زیبا', 'خواننده اول')">  🎵 آهنگ زیبا - خواننده اول  </div>  <div class="playlist-item" onclick="playTrack('audio/track2.mp3', 'ملودی شب', 'خواننده دوم')">  🎵 ملودی شب - خواننده دوم  </div>  <div class="playlist-item" onclick="playTrack('audio/track3.mp3', 'ریتم تابستان', 'خواننده سوم')">  🎵 ریتم تابستان - خواننده سوم  </div>  </div>  </div>  <script>  function playTrack(src, title, artist) {  const audio = document.getElementById('mainAudio');  const currentTrack = document.getElementById('currentTrack');  const currentArtist = document.getElementById('currentArtist');    audio.src = src;  currentTrack.textContent = title;  currentArtist.textContent = artist;    audio.play().catch(e => {  console.log('پخش خودکار ممکن نیست:', e);  });  }  </script>  </body>  </html> |

[complete](webdesign/html/tags/audio/complete.html)

مثال ترکیبی

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <title>مدیا پلیر ترکیبی</title>  <style>  .media-grid {  display: grid;  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  gap: 20px;  padding: 20px;  }    .media-card {  background: white;  border-radius: 10px;  padding: 15px;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  }    .media-title {  font-weight: bold;  margin-bottom: 10px;  color: #2c3e50;  }  </style>  </head>  <body>  <h1>مدیا پلیر ترکیبی</h1>    <div class="media-grid">  <!-- ویدیو -->  <div class="media-card">  <div class="media-title">ویدیوی آموزشی</div>  <video controls width="100%">  <source src="video/tutorial.mp4" type="video/mp4">  </video>  </div>    <!-- صوت -->  <div class="media-card">  <div class="media-title">پادکست هفتگی</div>  <audio controls style="width: 100%">  <source src="audio/podcast.mp3" type="audio/mpeg">  </audio>  </div>    <!-- iframe -->  <div class="media-card">  <div class="media-title">نقشه تعاملی</div>  <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3239.8..."  width="100%"  height="200"  frameborder="0"  style="border:0;"  allowfullscreen>  </iframe>  </div>  </div>  </body>  </html> |

نکات مهم:

## معرفی تگ picture

# [آموزش CSS](webdesign/css)

## معرفی declaration

در CSS، Declaration (اعلان) به هر دستور تنظیمی گفته می‌شود که یک ویژگی (property) و مقدار (value) را مشخص می‌کند.

ساختار کلی:

|  |
| --- |
| property: value; |

مثال:

|  |
| --- |
| color: red; |

در اینجا:

\* `color` → ویژگی (property)

\* `red` → مقدار (value)

هر declaration باید با نقطه‌ویرگول (;) تمام شود.

## [معرفی selector و property و value](webdesign/css/selector)

Selector برای انتخاب المان (عنصر)‌های خاص در صفحه‌ی HTML استفاده می‌شود تا بتوانیم به آن‌ها استایل بدهیم.

ساختار کلی:

|  |
| --- |
| selector {  property: value;  } |

مثال:

|  |
| --- |
| p {  color: blue;  } |

[index](webdesign/css/selector/index.html)

در این مثال:

\* `p` → selector است (المان‌های `<p>` را انتخاب می‌کند)

\* `{ color: blue; }` → declaration block است

Property (ویژگی) و Value (مقدار):

Property → ویژگی‌ای است که می‌خواهید تنظیم کنید (مثل رنگ، اندازه، پس‌زمینه و غیره).

Value → مقدار آن ویژگی است.

مثال:

|  |
| --- |
| font-size: 20px;  background-color: yellow; |

[Property](webdesign/css/selector/Property.html)

## معرفی element selector

Element Selector ساده‌ترین نوع selector است.

با استفاده از نام تگ HTML، تمام عناصر از همان نوع انتخاب می‌شوند.

مثال:

|  |
| --- |
| h1 {  color: green;  }  p {  font-size: 16px;  } |

[element](webdesign/css/selector/element.html)

در این مثال:

تمام `<h1>`ها سبز می‌شوند.

تمام `<p>`ها اندازه‌ی فونت ۱۶ پیکسل خواهند داشت.

## معرفی id selector(local & global)

ID Selector (انتخاب‌گر شناسه) : ID selector برای انتخاب یک المان خاص در صفحه استفاده می‌شود.

هر المان می‌تواند فقط یک id داشته باشد و هر id باید منحصربه‌فرد باشد.

نحوه نوشتن:

در CSS، برای اشاره به id از علامت # استفاده می‌کنیم.(علامت # در برنامه نویسی شارپ خوانده می‌شود، شارپ اولین بار توسط زبان C استفاده شد و در آن هنگام این اسم از دنیای موسیقی وام گرفته شده بود. در موسیقی نت شارپ نیم پرده نت کروماتیک را بالاتر میآورد.)

|  |
| --- |
| #myBox {  background-color: lightblue;  color: white;  } |

[idSelector](webdesign/css/selector/idSelector.html)

تفاوت Local و Global در id:

• Local ID → اگر در یک component یا محدوده خاص استفاده شود (مثلاً در فریم‌ورک‌هایی مثل React یا Vue).

• Global ID → در کل سند HTML منحصربه‌فرد است و در CSS جهانی تعریف می‌شود.

معمولاً در HTML سنتی، همه idها به‌صورت global رفتار می‌کنند.

## معرفی class selector(local & global)

Class selector برای انتخاب یک یا چند المان با ویژگی مشترک استفاده می‌شود.

برخلاف id، می‌توان از یک class در چندین المان استفاده کرد.

نحوه نوشتن:

در CSS، از علامت . همان Dot برای اشاره به کلاس استفاده می‌کنیم.

|  |
| --- |
| .text-red {  color: red;  } |

[class](webdesign/css/selector/class.html)

## بررسی تفاوت Style های inline , internal ,external

|  |  |  |  |
| --- | --- | --- | --- |
| **نوع استایل** | **محل تعریف** | **مثال** | **کاربرد** |
| **Inline** | درون خود تگ HTML با ویژگی style | <p style="color: blue;"> | برای تغییر سریع یا خاص یک المان |
| **Internal** | درون تگ <style> در <head> | <style> p { color: blue; } </style> | برای صفحات کوچک یا تست |
| **External** | در فایل جداگانه CSS | link rel="stylesheet" href="style.css" | برای پروژه‌های بزرگ و ساختارمند |

## بررسی اولویت های style دهی

وقتی چند قانون CSS روی یک المان اعمال شوند، مرورگر باید تصمیم بگیرد کدام قانون برنده است.

ترتیب اولویت از زیاد به کم:

|  |  |  |  |
| --- | --- | --- | --- |
| **سطح اولویت** | **نوع انتخاب‌گر** | **مثال** | **توضیح** |
| **Inline style** | داخل تگ HTML | <p style="color:red;"> | بالاترین اولویت |
| **ID selector** | #box {} |  | هر id خاص‌تر از class و element است |
| **Class, Attribute, Pseudo-class** | .btn, [type="text"], :hover |  | سطح متوسط |
| **Element selector** | p, div, h1 |  | پایین‌تر از بقیه |
| **Universal selector** | \* {} |  | کمترین اولویت |

نکته:

اگر دو قانون دقیقاً با یک سطح اولویت تعریف شوند،

آخرین قانونی که در CSS آمده باشد اعمال می‌شود.

مثال کامل:

|  |
| --- |
| <p id="txt" class="red" style="color: green;">Hello CSS</p>  p { color: blue; }  .red { color: red; }  #txt { color: orange; } |

## [Complete](webdesign/css/selector/complete.html)

🎨 نتیجه نهایی → رنگ متن سبز است

(چون inline style بالاترین اولویت را دارد)

## [کامنت ها در CSS](webdesign/css/comments)

کامنت‌ها برای نوشتن توضیح در کد استفاده می‌شن و مرورگر اون‌ها رو اجرا نمی‌کنه.

یعنی فقط برای خوانایی و مستندسازی به کار می‌رن.

نحوه نوشتن کامنت در CSS:

|  |
| --- |
| /\* این یک کامنت است \*/  p {  color: blue; /\* این هم یک کامنت در کنار کد \*/  } |

[base](webdesign/css/comments/base.html)

ویژگی‌ها:

\* با `/\*` شروع و با `\*/` تمام می‌شود.

\* می‌توانند در هر جای فایل CSS نوشته شوند.

\* چند خطی هم می‌تونن باشن:

|  |
| --- |
| /\*  این یک کامنت چندخطی است  برای توضیح کدهای زیر  \*/ |

[Multi line](webdesign/css/comments/multyline.html)

## [معرفی !important و اولویت آن](webdesign/css/important)

`!important` برای افزایش اولویت اجرای یک قانون CSS استفاده می‌شود.

اگر چند قانون متناقض وجود داشته باشد، قانون دارای `!important` همیشه برنده است (حتی اگر اولویت انتخاب‌گر پایین‌تر باشد).

مثال:

|  |
| --- |
| p {  color: blue !important;  }  #text {  color: red;  } |

[example](webdesign/css/important/example.html)

نتیجه:

حتی اگر `#text` از نظر خاصیت (specificity) قوی‌تر است،

رنگ آبی اعمال می‌شود، چون `!important` دارد.

نکات مهم درباره `!important`:

باید فقط در شرایط خاص و ضروری استفاده شود (چون بعداً تغییر دادن استایل سخت می‌شود).

اگر چند قانون مختلف هر دو `!important` داشته باشند، آنگاه دوباره specificity و ترتیب قرارگیری تعیین‌کننده خواهد بود.

مثال:

|  |
| --- |
| p { color: blue !important; }  p.special { color: red !important; } |

[multy](webdesign/css/important/multy.html)

نتیجه → متن قرمز می‌شود، چون `.special` خاص‌تر است.

## معرفی group selector

وقتی چند انتخاب‌گر قرار است استایل یکسانی بگیرند،

می‌توان آن‌ها را در یک قانون با استفاده از ویرگول `,` با هم ترکیب کرد.

مثال:

|  |
| --- |
| h1, h2, h3 {  color: darkblue;  text-align: center;  } |

[group](webdesign/css/selector/group.html)

نتیجه:

تمام عناصر `h1`، `h2 و `h3` یک استایل مشترک می‌گیرند.

مزیت:

کد کوتاه‌تر و خواناتر می‌شود.

مدیریت استایل راحت‌تر است.

جمع‌بندی سریع:

|  |  |  |
| --- | --- | --- |
| مفهوم | توضیح کوتاه | مثال |
| Comment | توضیح در کد (اجرا نمی‌شود) | `/\* متن توضیح \*/` |
| !important | افزایش اولویت استایل | color: red !important; |
| Group Selector | اعمال استایل مشترک به چند المان | h1, h2, p { ... } |

## [معرفی CSS Units](webdesign/css/Units)

در CSS برای تعیین اندازه‌ها (مثل طول، عرض، فاصله، اندازه فونت و غیره) از واحدها استفاده می‌کنیم.

واحدها به دو دسته اصلی تقسیم می‌شوند:

الف) واحدهای مطلق (Absolute Units)

این واحدها مستقل از صفحه یا دستگاه هستند؛ یعنی همیشه اندازه ثابتی دارند.

|  |  |  |
| --- | --- | --- |
| واحد | توضیح | معادل تقریبی |
| `px` | پیکسل – متداول‌ترین واحد در طراحی وب | ۱ پیکسل در صفحه |
| `pt` | پوینت – معمولاً در چاپ استفاده می‌شود | ۱pt = 1/72 اینچ |
| `cm` | سانتی‌متر |  |
| `mm` | میلی‌متر |  |
| `in` | اینچ | 1in = 96px |
| `pc` | پیکا | 1pc = 12pt |

مثال:

|  |
| --- |
| div {  width: 200px;  font-size: 12pt;  } |

[example](webdesign/css/Units/example.html)

ب) واحدهای نسبی (Relative Units)

این واحدها نسبت به یک چیز دیگر (مثل اندازه صفحه، فونت والد یا اندازه Viewport) محاسبه می‌شوند.

|  |  |  |
| --- | --- | --- |
| واحد | نسبت به | توضیح |
| `%` | والد | درصدی از اندازه والد |
| `em` | اندازه فونت والد | 1em = اندازه فونت والد |
| `rem` | اندازه فونت ریشه (html) | پایدارتر از em |
| `vw` | عرض صفحه (Viewport Width) | 1vw = 1٪ از عرض پنجره مرورگر |
| `vh` | ارتفاع صفحه (Viewport Height) | 1vh = 1٪ از ارتفاع پنجره مرورگر |
| `vmin` | کوچکتر بین vw و vh | برای واکنش‌گرایی (responsive) مفید است |
| `vmax` | بزرگتر بین vw و vh | مشابه مورد بالا |
| `ch` | عرض عدد صفر در فونت فعلی | برای طراحی دقیق متن‌ها |
| `ex` | ارتفاع x کوچک در فونت فعلی | کمتر استفاده می‌شود |

مثال:

|  |
| --- |
| body {  font-size: 16px;  }  p {  font-size: 1.5em; /\* 1.5 × 16px = 24px \*/  }  h1 {  font-size: 2rem; /\* 2 × اندازه فونت root \*/  }  div {  width: 50vw; /\* نصف عرض صفحه مرورگر \*/  } |

[Example2](webdesign/css/Units/example2.html)

نکته‌ی کلیدی:

برای طراحی واکنش‌گرا (Responsive) بهتر است از واحدهای نسبی مثل `em`, `rem`, `vw`, `%` استفاده شود.

واحدهای مطلق مثل `px` معمولاً برای طراحی دقیق و ثابت به کار می‌روند.

## معرفی Font-family

ویژگی `font-family` نوع فونت نوشته‌ها را مشخص می‌کند.

می‌توان چند فونت را پشت سر هم نوشت تا اگر یکی در سیستم کاربر موجود نبود، دیگری استفاده شود.

ساختار کلی:

|  |
| --- |
| font-family: "Font1", "Font2", generic-family; |

مثال:

|  |
| --- |
| p {  font-family: "Tahoma", "Arial", sans-serif;  } |

[example](webdesign/css/font/example.html)

در این مثال:

مرورگر ابتدا سعی می‌کند فونت Tahoma را نمایش دهد.

اگر Tahoma در سیستم کاربر موجود نبود، از Arial استفاده می‌کند.

اگر هیچ‌کدام نبودند، از sans-serif به‌عنوان فونت عمومی استفاده می‌کند.

خانواده‌های عمومی (Generic Font Families)

|  |  |
| --- | --- |
| نام خانواده | ویژگی ظاهری |
| `serif` | حروف دارای دُم (مثل Times New Roman) |
| `sans-serif` | حروف ساده بدون دُم (مثل Arial, Helvetica) |
| `monospace` | عرض ثابت برای هر کاراکتر (مثل Courier New) |
| `cursive` | فونت دست‌نویس (مثل Comic Sans MS) |
| `fantasy` | فونت تزئینی یا فانتزی |

مثال:

|  |
| --- |
| h1 {  font-family: "Times New Roman", serif;  } |

[Generic](webdesign/css/font/Generic.html)

نکته‌ها:

اگر نام فونت شامل فاصله باشد، باید داخل گیومه (" ") نوشته شود.

بهتر است همیشه در انتهای لیست فونت‌ها یک generic family قرار دهید تا در صورت نبود فونت‌های سفارشی، ظاهر متن خراب نشود.

جمع‌بندی سریع:

|  |  |  |
| --- | --- | --- |
| مفهوم | توضیح کوتاه | مثال |
| Absolute Units | اندازه‌های ثابت (px, pt, cm) | width: 200px; |
| Relative Units | وابسته به محیط (%, em, rem, vw) | font-size: 2rem; |
| font-family | تعیین نوع فونت متن | font-family: "Arial", sans-serif; |

استفاده از سایت transfonter

استفاده از سایت https://transfonter.org

Transfonter سایتی است برای تبدیل فونت‌های دسکتاپ (مثل TTF یا OTF) به فرمت‌های مناسب وب (مثل WOFF و WOFF2).

مراحل استفاده:

1. وارد سایت شوید:

https://transfonter.org

2. روی دکمه “Add fonts” کلیک کن و فونت مورد نظر خودت (مثل `IranSans.ttf`) رو انتخاب کن.

3. در قسمت تنظیمات:

\* تیک گزینه‌ی “Convert to WOFF/WOFF2” رو بزن.

\* می‌تونی گزینه‌ی “Include @font-face CSS” رو هم فعال کنی تا فایل CSS خودش بسازه.

4. روی دکمه‌ی “Convert” کلیک کن.

5. فایل ZIP دانلود می‌شه که شامل:

فونت‌های تبدیل‌شده (`.woff`, `.woff2`)

و یک فایل CSS آماده است.

نحوه استفاده در پروژه:

بعد از استخراج فایل ZIP، معمولاً باید فایل CSS فونت رو به پروژه لینک کنی:

|  |
| --- |
| <link rel="stylesheet" href="fonts/font.css"> |

استفاده از font-face

دستور `@font-face` بهت اجازه می‌ده فونت دلخواهت رو مستقیماً در سایت خودت تعریف و استفاده کنی — حتی اگه اون فونت در سیستم کاربر نصب نباشه.

ساختار کلی:

|  |
| --- |
| @font-face {  font-family: 'MyFont';  src: url('fonts/MyFont.woff2') format('woff2'),  url('fonts/MyFont.woff') format('woff');  font-weight: normal;  font-style: normal;  } |

[fontface](webdesign/css/font/fontface.html)

سپس از اون فونت در هر جای سایت می‌تونی استفاده کنی:

|  |
| --- |
| body {  font-family: 'MyFont', sans-serif;  } |

توضیح قسمت‌ها:

|  |  |
| --- | --- |
| ویژگی | توضیح |
| `font-family` | نامی دلخواه که برای فونت انتخاب می‌کنی (در CSS از همین اسم استفاده می‌کنی). |
| `src` | مسیر فایل‌های فونت (می‌تونه چند نوع فرمت داشته باشه). |
| format() | نوع فرمت فایل (مثلاً woff, woff2, ttf). |
| font-weight / font-style | برای مشخص کردن حالت نرمال، بولد یا ایتالیک فونت. |

نکات:

بهتره از فرمت‌های WOFF و WOFF2 استفاده کنی چون فشرده‌تر و مخصوص وب هستن.

مسیر (`url`) باید نسبت به فایل CSS تنظیم بشه.

برای فونت‌های فارسی مثل IRANSans، Vazir، یا Yekan Bakh هم می‌تونی از همین روش استفاده کنی.

بررسی مفهوم font stack

Font Stack یعنی نوشتن چند فونت پشت سر هم در خاصیت `font-family` تا اگر یکی در مرورگر یا سیستم کاربر موجود نبود، دیگری جایگزین شود.

مثال:

|  |
| --- |
| body {  font-family: "Vazir", "Tahoma", "Arial", sans-serif;  } |

[font-stack](webdesign/css/font/font-stack.html)

ترتیب بررسی مرورگر:

1. ابتدا سعی می‌کند فونت Vazir را نمایش دهد.

2. اگر وجود نداشت، سراغ Tahoma می‌رود.

3. بعد Arial را امتحان می‌کند.

4. اگر هیچ‌کدام نبود، از خانواده‌ی عمومی sans-serif استفاده می‌کند.

مزیت Font Stack:

اطمینان از اینکه متن همیشه خوانا و زیبا باقی می‌ماند.

انعطاف‌پذیری بین دستگاه‌ها و سیستم‌عامل‌های مختلف.

مثال ترکیبی (همه با هم):

فرض کن فونت سفارشی "IRANSans" داری و با Transfonter تبدیلش کردی:

|  |
| --- |
| @font-face {  font-family: 'IRANSans';  src: url('fonts/IRANSans.woff2') format('woff2'),  url('fonts/IRANSans.woff') format('woff');  font-weight: normal;  font-style: normal;  }  body {  font-family: 'IRANSans', 'Tahoma', 'Arial', sans-serif;  } |

[combine](webdesign/css/font/combine.html)

نتیجه:

اگر IRANSans در دسترس باشد → از آن استفاده می‌شود.

اگر نباشد → مرورگر به ترتیب به فونت‌های بعدی می‌رود.

جمع‌بندی سریع:

|  |  |  |
| --- | --- | --- |
| مفهوم | توضیح | مثال |
| Transfonter | تبدیل فونت‌های TTF/OTF به WOFF/WOFF2 برای وب | سایت transfonter.org |
| @font-face | معرفی فونت سفارشی در CSS | `@font-face { font-family: 'MyFont'; src: url(...); }` |
| Font Stack | لیست چند فونت جایگزین برای نمایش مطمئن‌تر | `font-family: "Vazir", "Tahoma", sans-serif;` |

Font-size

ویژگی `font-size` برای تعیین اندازه‌ی متن استفاده می‌شود.

ساختار کلی:

|  |
| --- |
| selector {  font-size: value;  } |

مقادیر قابل استفاده:

|  |  |  |
| --- | --- | --- |
| نوع مقدار | مثال | توضیح |
| px (پیکسل) | `font-size: 16px; | اندازه ثابت، متداول‌ترین واحد |
| em | `font-size: 1.2em;` | نسبت به اندازه‌ی فونت والد |
| rem | `font-size: 1.5rem;` | نسبت به اندازه‌ی فونت root (معمولاً html) |
| % | `font-size: 120%;` | درصدی از اندازه‌ی فونت والد |
| کلمات کلیدی | `font-size: small;`, `large;`, `medium;` | اندازه‌های از پیش تعریف‌شده توسط مرورگر |

مثال:

|  |
| --- |
| p {  font-size: 18px;  }  h1 {  font-size: 2rem;  }  span {  font-size: 120%;  } |

[Font-size](webdesign/css/font/font-size.html)

نتیجه:

متن `<p>` با اندازه‌ی ثابت ۱۸px نمایش داده می‌شود.

`<h1>` نسبت به root بزرگ‌تر است.

`<span>` کمی بزرگ‌تر از والد خود است.

Font-weight

ویژگی `font-weight` (ضخامت فونت)

ویژگی `font-weight` میزان ضخامت یا نازکی حروف را مشخص می‌کند.

ساختار کلی:

|  |
| --- |
| selector {  font-weight: value;  } |

مقادیر قابل استفاده:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `normal` | ضخامت معمولی (معادل 400) |
| `bold` | ضخیم (معادل 700) |
| `bolder` | ضخیم‌تر از والد |
| `lighter` | نازک‌تر از والد |
| `100` تا `900` | ضخامت عددی (100 = نازک‌ترین، 900 = ضخیم‌ترین) |

مثال:

|  |
| --- |
| h1 {  font-weight: bold;  }  p {  font-weight: 300;  } |

[font-weight](webdesign/css/font/font-weight.html)

نکته: همه‌ی فونت‌ها از همه‌ی وزن‌ها پشتیبانی نمی‌کنند (مثلاً برخی فقط normal و bold دارند).

Font-style

ویژگی `font-style` (سبک فونت)

ویژگی `font-style` برای تعیین سبک نوشتاری فونت استفاده می‌شود.

ساختار کلی:

|  |
| --- |
| selector {  font-style: value;  } |

مقادیر قابل استفاده:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `normal` | فونت عادی |
| `italic` | فونت ایتالیک (کج‌شده واقعی) |
| `oblique` | حالت مایل مصنوعی (کمی متفاوت از italic) |

مثال:

|  |
| --- |
| p {  font-style: italic;  }  em {  font-style: oblique;  } |

[font-style](webdesign/css/font/font-style.html)

تفاوت:

`italic` معمولاً نسخه‌ی مخصوص از فونت را استفاده می‌کند (طراحی‌شده برای ایتالیک).

`oblique` فقط حروف را به‌صورت مصنوعی کمی کج می‌کند.

مثال ترکیبی:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <style>  body {  font-family: "Vazir", sans-serif;  }  h1 {  font-size: 2.5rem;  font-weight: 700;  font-style: normal;  }  p {  font-size: 18px;  font-weight: 300;  font-style: italic;  }  </style>  </head>  <body>  <h1>عنوان نمونه</h1>  <p>این یک متن نمونه با فونت ایتالیک و وزن نازک است.</p>  </body>  </html> |

[combine2](webdesign/css/font/combine2.html)

جمع‌بندی سریع:

|  |  |  |
| --- | --- | --- |
| ویژگی | توضیح | مثال |
| font-size | اندازه‌ی فونت | `font-size: 16px;` |
| font-weight | ضخامت فونت | `font-weight: bold;` |
| font-style | سبک نوشتاری (کج، معمولی) | `font-style: italic;` |

Font-variant

این ویژگی برای تغییر حالت خاصی از نمایش فونت استفاده می‌شود، مثل تبدیل حروف کوچک به حالت small-caps (حروف بزرگ کوچک‌نمایی‌شده).

نحوه‌ی استفاده:

|  |
| --- |
| p {  font-variant: small-caps;  } |

مقادیر ممکن:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `normal` | حالت عادی فونت (پیش‌فرض) |
| `small-caps` | حروف کوچک را به حالت حروف بزرگ کوچک‌شده تبدیل می‌کند (مثلاً `Hello` → `HELLO` با حروف کوچک‌تر) |
| `all-small-caps` | همه‌ی حروف را به small-caps تبدیل می‌کند (در فونت‌های خاص CSS Fonts Level 4) |
| `petite-caps`, `unicase`, `titling-caps` | حالت‌های خاص دیگر در برخی فونت‌ها (بیشتر برای طراحی حرفه‌ای تایپوگرافی) |

مثال:

|  |
| --- |
| h1 {  font-variant: small-caps;  } |

[Font-variant](webdesign/css/font/Font-variant.html)

نتیجه: حروف کوچک به صورت حروف بزرگ ولی با اندازه‌ی کوچک‌تر نمایش داده می‌شوند.

Font

ویژگی font یک شورت‌هند (Shorthand) است، یعنی ترکیبی از چند ویژگی دیگر مرتبط با فونت، مثل:

• font-style

• font-variant

• font-weight

• font-size

• line-height

• font-family

ساختار کلی:

|  |
| --- |
| font: [style] [variant] [weight] size[/line-height] family; |

مثال ساده:

|  |
| --- |
| p {  font: italic small-caps bold 16px/1.5 "Times New Roman", serif;  } |

[font](webdesign/css/font/font.html)

این مثال معادل است با:

|  |
| --- |
| p {  font-style: italic;  font-variant: small-caps;  font-weight: bold;  font-size: 16px;  line-height: 1.5;  font-family: "Times New Roman", serif;  } |

نکته مهم:

وقتی از font استفاده می‌کنی، باید حداقل font-size و font-family را مشخص کنی.

اگر این دو مورد را ننویسی، مرورگر آن را نادیده می‌گیرد.

مثال نهایی:

|  |
| --- |
| body {  font: normal small-caps 600 18px/1.4 "Open Sans", sans-serif;  } |

[final](webdesign/css/font/font.html)

نتیجه:

فونت با ضخامت 600، اندازه px18، فاصله خط 1.4، با small-caps و فونت Open Sans نمایش داده می‌شود.

## [Color](webdesign/css/color)

ویژگی `color` در CSS برای تعیین رنگ متن (text) در عناصر HTML استفاده می‌شود.

این یکی از پرکاربردترین ویژگی‌ها در طراحی وب است.

کاربرد اصلی

|  |
| --- |
| p {  color: red;  } |

[base](webdesign/css/color/base.html)

این کد رنگ متن داخل تگ `<p>` را قرمز می‌کند.

انواع روش‌های تعریف رنگ در CSS

با نام رنگ (Color Names)

CSS بیش از 140 نام رنگ استاندارد دارد.

|  |
| --- |
| h1 {  color: blue;  }  p {  color: darkgreen;  } |

[name](webdesign/css/color/name.html)

نمونه‌هایی از نام رنگ‌ها:

|  |
| --- |
| `red`, `blue`, `green`, `black`, `white`, `gray`, `purple`, `orange`, `gold`, `crimson`, ... |

با مقدار HEX (هگزادسیمال)

فرمت: `#RRGGBB`

هر جفت عدد (دو رقم) نشان‌دهنده میزان قرمز (R)، سبز (G) و آبی (B) است.

|  |
| --- |
| p {  color: #ff0000; /\* قرمز \*/  }  h2 {  color: #00ff00; /\* سبز \*/  } |

[hex](webdesign/css/color/hex.html)

نسخه کوتاه‌تر: `#RGB`

|  |
| --- |
| p {  color: #f00; /\* همان قرمز \*/  } |

[shortHex](webdesign/css/color/shortHex.html)

با تابع rgb()

فرمت: `rgb(red, green, blue)`

مقادیر بین 0 تا 255 هستند.

|  |
| --- |
| p {  color: rgb(255, 0, 0); /\* قرمز \*/  } |

[rgb](webdesign/css/color/rgb.html)

با تابع `rgba()` (دارای شفافیت)

فرمت: `rgba(red, green, blue, alpha)`

مقدار alpha بین 0 (کاملاً شفاف) تا 1 (کامل پررنگ) است.

|  |
| --- |
| p {  color: rgba(255, 0, 0, 0.6); /\* قرمز نیمه‌شفاف \*/  } |

[rgba](webdesign/css/color/rgba.html)

با تابع `hsl()` و `hsla()`

فرمت: `hsl(hue, saturation, lightness)`

hue: زاویه رنگ (0 تا 360)

saturation: شدت رنگ (٪)

lightness: روشنایی رنگ (٪)

|  |
| --- |
| p {  color: hsl(120, 100%, 25%); /\* سبز تیره \*/  } |

[hsl](webdesign/css/color/hsl.html)

با شفافیت hsla():

|  |
| --- |
| p {  color: hsla(200, 80%, 50%, 0.5);  } |

[hsla](webdesign/css/color/hsla.html)

نکات مهم

ویژگی `color` فقط رنگ متن را تغییر می‌دهد، نه پس‌زمینه. برای پس‌زمینه از `background-color` استفاده می‌شود.

اگر می‌خواهی رنگ متن لینک‌ها را تغییر دهی، باید از انتخاب‌گرهای خاص استفاده کنی:

|  |
| --- |
| a:link { color: blue; }  a:visited { color: purple; }  a:hover { color: red; }  a:active { color: orange; } |

[a](webdesign/css/color/a.html)

## [Selectors](#_معرفی_selector_و)

معرفی descendant selectors

Descendant Selector (انتخاب‌گر نسل‌ها / نوادگان)

این انتخاب‌گر همه‌ی عناصر درون (در هر سطحی) را هدف می‌گیرد.

در واقع اگر یک عنصر درون عنصر دیگر باشد (حتی چند سطح پایین‌تر)، انتخاب می‌شود.

نحوه‌ی نوشتن:

|  |
| --- |
| ancestor descendant { ... } |

یعنی: هر عنصری با نام descendant که درون ancestor قرار دارد.

مثال:

|  |
| --- |
| <div>  <p>این پاراگراف درون div است.</p>  <section>  <p>این هم داخل section که درون div است.</p>  </section>  </div> |

|  |
| --- |
| div p {  color: blue;  } |

[descendant](webdesign/css/selector/descendant.html)

نتیجه:

هر <p> که در هر سطحی داخل <div> باشد، آبی می‌شود (چه مستقیم، چه غیرمستقیم).

معرفی child selectors

Child Selector (انتخاب‌گر فرزند مستقیم)

این انتخاب‌گر فقط عناصر فرزند مستقیم یک عنصر را انتخاب می‌کند (نه نوه‌ها و پایین‌تر).

نحوه‌ی نوشتن:

|  |
| --- |
| parent > child { ... } |

مثال:

|  |
| --- |
| <div>  <p>فرزند مستقیم div</p>  <section>  <p>نوه div</p>  </section>  </div> |

|  |
| --- |
| div > p {  color: red;  } |

[child](webdesign/css/selector/child.html)

نتیجه:

<p> اول (فرزند مستقیم) قرمز می‌شود.

<p> دوم (داخل <section>) تغییر نمی‌کند.

معرفی next-sibling selector

Next-Sibling Selector (انتخاب‌گر خواهر/برادر بعدی)

این انتخاب‌گر عنصر هم‌سطح بعدی (بلافاصله بعد از عنصر مشخص‌شده) را انتخاب می‌کند.

نحوه‌ی نوشتن:

|  |
| --- |
| element1 + element2 { ... } |

یعنی: element2 که بلافاصله بعد از element1 آمده.

مثال:

|  |
| --- |
| <h2>عنوان</h2>  <p>پاراگراف بلافاصله بعد از h2</p>  <p>پاراگراف دوم</p> |

|  |
| --- |
| h2 + p {  color: green;  } |

[Next-Sibling](webdesign/css/selector/Next-Sibling.html)

نتیجه:

فقط اولین <p> که بلافاصله بعد از <h2> آمده سبز می‌شود.

پاراگراف دوم تغییری نمی‌کند.

جمع‌بندی سریع:

|  |  |  |  |
| --- | --- | --- | --- |
| انتخاب‌گر | نماد | توضیح | مثال |
| Descendant | (فاصله) | تمام نوادگان در هر سطح | `div p` |
| Child | `>` | فقط فرزندان مستقیم | `div > p` |
| Next-sibling | `+` | فقط عنصر هم‌سطح بعدی | `h2 + p` |

معرفی Subsequent-sibling selectors

(به فارسی: انتخاب‌گر خواهر/برادرهای بعدی یا عمومی‌تر از next-sibling)

نماد: ~

این انتخاب‌گر، تمام عناصر هم‌سطح بعدی (نه فقط اولین) را که بعد از یک عنصر خاص قرار گرفته‌اند، انتخاب می‌کند.

نحوه‌ی نوشتن:

|  |
| --- |
| element1 ~ element2 { ... } |

یعنی هر element2 که بعد از element1 در همان سطح (هم‌والد) آمده است.

مثال:

|  |
| --- |
| <h2>عنوان</h2>  <p>پاراگراف اول بعد از h2</p>  <p>پاراگراف دوم بعد از h2</p>  <div>div بعد از h2</div> |

[siblingselectors](webdesign/css/selector/siblingselectors.html)

نتیجه:

هر دو <p> که بعد از <h2> آمده‌اند (در همان سطح) آبی می‌شوند.

اما <div> تغییری نمی‌کند.

تفاوت با + (Next-Sibling):

|  |  |  |
| --- | --- | --- |
| انتخاب‌گر | نماد | انتخاب می‌کند |
| `+` | Next-sibling | فقط اولین عنصر هم‌سطح بعدی |
| `~` | Subsequent-sibling | همه‌ی عناصر هم‌سطح بعدی از همان نوع |

مثال مقایسه‌ای:

|  |
| --- |
| <h2>عنوان</h2>  <p>پاراگراف ۱</p>  <p>پاراگراف ۲</p> |

|  |
| --- |
| h2 + p { color: red; } /\* فقط پاراگراف ۱ \*/  h2 ~ p { color: blue; } /\* هر دو پاراگراف \*/ |

[compare](webdesign/css/selector/cpmpare.html)

معرفی universal selector

به فارسی: انتخاب‌گر عمومی یا سراسری)

نماد: \*

این انتخاب‌گر تمام عناصر موجود در صفحه را انتخاب می‌کند.

یعنی بدون استثنا روی همه‌ی تگ‌های HTML اعمال می‌شود.

نحوه‌ی نوشتن:

|  |
| --- |
| \* {  margin: 0;  padding: 0;  } |

[universal](webdesign/css/selector/universal.html)

این کد همه‌ی تگ‌ها را بدون فاصله‌ی پیش‌فرض (reset) می‌کند.

اغلب در ابتدای فایل CSS استفاده می‌شود تا ظاهر مرورگرهای مختلف یکسان شود.

همچنین می‌توان از \* در ترکیب با انتخاب‌گرهای دیگر استفاده کرد:

|  |
| --- |
| div \* {  color: green;  } |

[universal2](webdesign/css/selector/universal2.html)

یعنی تمام عناصری که داخل هر <div> قرار دارند (در هر سطحی)، رنگشان سبز شود.

جمع‌بندی سریع

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| انتخاب‌گر | نماد | کاربرد | مثال | توضیح |
| Next-sibling | `+` | فقط اولین عنصر هم‌سطح بعدی | `h2 + p` | اولین `<p>` بعد از `<h2>` |
| Subsequent-sibling | `~` | همه‌ی عناصر هم‌سطح بعدی | `h2 ~ p` | تمام `<p>`های بعد از `<h2>` |
| Universal | `\*` | همه‌ی عناصر صفحه | `\* { margin: 0; }` | برای تنظیم کلی یا انتخاب سراسری |

## [استایل متن](webdesign/css/text)

Line-height

(ارتفاع خط / فاصله بین خطوط متن)

این ویژگی فاصله‌ی عمودی بین خطوط متن را تعیین می‌کند.

در واقع مشخص می‌کند هر خط چه‌قدر از خط بعدی فاصله داشته باشد.

نحوه‌ی نوشتن:

|  |
| --- |
| p {  line-height: 1.6;  } |

[line-height](webdesign/css/text/line-height.html)

مقدارها:

|  |  |  |
| --- | --- | --- |
| نوع مقدار | مثال | توضیح |
| عدد (بدون واحد) | `1.5` | فاصله ۱.۵ برابر اندازه فونت |
| با واحد پیکسلی یا درصدی | `24px`, `150%` | فاصله مشخص بر حسب واحد |
| نرمال (پیش‌فرض) | `normal` | فاصله عادی مرورگر |

مثال:

|  |
| --- |
| p {  font-size: 16px;  line-height: 1.8;  } |

[line-height](webdesign/css/text/line-height2.html)

متن خواناتر می‌شود چون خطوط کمی از هم فاصله دارند.

Text-indent

(تورفتگی ابتدای پاراگراف)

این ویژگی میزان تورفتگی (فاصله از سمت چپ یا راست، بسته به زبان متن) در ابتدای خط اول یک پاراگراف را تعیین می‌کند.

نحوه‌ی نوشتن:

|  |
| --- |
| p {  text-indent: 40px;  } |

[indent](webdesign/css/text/indent.html)

خط اول پاراگراف ۴۰ پیکسل از سمت چپ (در متن انگلیسی) یا از سمت راست (در متن فارسی) فاصله می‌گیرد.

مقدارها:

|  |  |  |
| --- | --- | --- |
| نوع مقدار | مثال | توضیح |
| طول (پیکسل، em، درصد) | `50px`, `2em`, `10%` | مقدار فاصله از ابتدای خط |
| منفی | `-20px` | باعث می‌شود خط اول به بیرون از محدوده بلوک برود (کمتر استفاده می‌شود) |

مثال فارسی:

|  |
| --- |
| p {  direction: rtl;  text-indent: 2em;  } |

[text-indent2](webdesign/css/text/text-indent2.html)

پاراگراف فارسی با فاصله‌ی طبیعی از سمت راست شروع می‌شود.

Text-align

(تراز متن)

این ویژگی جهت ترازبندی متن درون یک عنصر را مشخص می‌کند.

نحوه‌ی نوشتن:

|  |
| --- |
| p {  text-align: justify;  } |

[text-align](webdesign/css/text/text-align.html)

مقدارها:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `left` | تراز از چپ |
| `right` | تراز از راست |
| `center` | تراز در مرکز |
| `justify` | تراز دوطرفه (مثل روزنامه‌ها، پر کردن تمام عرض خط) |
| `start` / `end` | بسته به جهت زبان (در CSS جدیدتر) |

مثال:

|  |
| --- |
| p {  text-align: justify;  }  h1 {  text-align: center;  } |

[text-align2](webdesign/css/text/text-align2.html)

جمع‌بندی سریع:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | توضیح | مثال | نتیجه |
| `line-height` | فاصله بین خطوط متن | `line-height: 1.8;` | خوانایی بیشتر |
| `text-indent` | تورفتگی ابتدای پاراگراف | `text-indent: 30px;` | شروع منظم پاراگراف |
| `text-align` | ترازبندی متن | `text-align: center;` | متن وسط‌چین یا راست‌چین و ... |

Text-decoration

(تزیین یا خط‌های مربوط به متن)

این ویژگی برای اضافه‌کردن یا حذف خط‌هایی مثل زیرخط، خط وسط، یا بالاخط به متن استفاده می‌شود.

نحوه نوشتن:

|  |
| --- |
| a {  text-decoration: none;  } |

[text-decoration](webdesign/css/text/text-decoration.html)

مقادیر رایج:

|  |  |  |
| --- | --- | --- |
| مقدار | توضیح | مثال |
| `none` | بدون هیچ خطی | حذف زیرخط لینک‌ها |
| `underline` | خط زیر متن | زیرخط برای تأکید یا لینک |
| `overline` | خط بالای متن |  |
| `line-through` | خط وسط متن | برای نشان‌دادن حذف یا تخفیف |
| `underline overline` | چند خط هم‌زمان |  |

مثال:

|  |
| --- |
| h1 {  text-decoration: underline overline;  }  del {  text-decoration: line-through;  } |

[text-decoration2](webdesign/css/text/text-decoration2.html)

نتیجه:

<h1> هم بالا و هم پایین خط دارد

<del> (حذف شده) خط وسط دارد

CSS3 جزئی‌تر:

می‌توان جزئیات را دقیق‌تر تنظیم کرد:

|  |
| --- |
| text-decoration-line: underline;  text-decoration-color: red;  text-decoration-style: wavy; |

نتیجه: زیرخط قرمز موج‌دار (برای طراحی زیبا)

Text-transform

(تبدیل حروف متن)

برای تغییر حالت حروف (کوچک/بزرگ) در متن استفاده می‌شود.

نحوه نوشتن:

|  |
| --- |
| p {  text-transform: uppercase;  } |

[transform](webdesign/css/text/transform.html)

مقادیر:

|  |  |  |
| --- | --- | --- |
| مقدار | توضیح | مثال |
| `none` | بدون تغییر | متن عادی |
| `uppercase` | تمام حروف بزرگ | `hello` → `HELLO` |
| `lowercase` | تمام حروف کوچک | `HELLO` → `hello` |
| `capitalize` | حرف اول هر کلمه بزرگ | `hello world` → `Hello World` |

مثال:

|  |
| --- |
| h1 {  text-transform: capitalize;  }  p {  text-transform: lowercase;  } |

[transform2](webdesign/css/text/transform2.html)

برای عنوان‌ها (capitalize) عالیه

و برای کنترل استاندارد متن (lowercase) هم کاربردیه.

Letter-spacing

)فاصله بین حروف)

این ویژگی فاصله‌ی افقی بین حروف یک متن را مشخص می‌کند.

برای زیبایی تایپوگرافی و خوانایی استفاده می‌شود.

نحوه نوشتن:

|  |
| --- |
| h1 {  letter-spacing: 2px;  } |

[spacing](webdesign/css/text/spacing.html)

مقدارها:

|  |  |
| --- | --- |
| مقدار | توضیح |
| عدد با واحد (px, em, ...) | فاصله دلخواه بین حروف |
| `normal` | مقدار پیش‌فرض مرورگر |

مثال:

|  |
| --- |
| h1 {  letter-spacing: 5px;  }  p {  letter-spacing: -1px;  } |

[spacing3](webdesign/css/text/spacing3.html)

در <h1> حروف از هم فاصله می‌گیرند،

اما در <p> فاصله کمتر از حالت عادی می‌شود.

جمع‌بندی سریع:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | توضیح | نمونه مقدار | کاربرد |
| `text-decoration` | خط‌دار کردن یا حذف خط متن | `underline`, `line-through` | لینک‌ها و متون حذف‌شده |
| `text-transform` | تغییر حالت حروف | `uppercase`, `capitalize` | ظاهر تیتر یا نام‌ها |
| `letter-spacing` | تنظیم فاصله بین حروف | `2px`, `normal` | زیباسازی تایپوگرافی |

Word-spacing

(فاصله بین کلمات)

این ویژگی برای تنظیم فاصله‌ی **بین کلمات (نه حروف)** استفاده می‌شود.

نحوه‌ی نوشتن:

|  |
| --- |
| p {  word-spacing: 10px;  } |

[Word-spacing](webdesign/css/text/Word-spacing.html)

در این حالت بین هر کلمه در پاراگراف ۱۰ پیکسل فاصله اضافه می‌شود.

مقدارها:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `normal` | فاصله‌ی پیش‌فرض مرورگر |
| `طول (px, em, %)` | فاصله دلخواه بین کلمات |
| `منفی` | کاهش فاصله بین کلمات (نزدیک‌تر شدن) |

مثال:

|  |
| --- |
| p {  word-spacing: 5px;  }  h1 {  word-spacing: -3px;  } |

[Word-spacing2](webdesign/css/text/Word-spacing2.html)

در پاراگراف، کلمات کمی از هم فاصله می‌گیرن،

و در عنوان حروف به هم نزدیک‌تر می‌شن.

Text-shadow

(سایه‌ی متن)

این ویژگی برای افزودن سایه‌ی تزئینی به متن استفاده می‌شود.

می‌تواند به نوشته عمق یا جلوه‌ی زیبایی بدهد.

نحوه‌ی نوشتن:

|  |
| --- |
| text-shadow: h-shadow v-shadow blur color; |

h-shadow: فاصله سایه در محور افقی

v-shadow: فاصله سایه در محور عمودی

blur: میزان تاری سایه (اختیاری)

color: رنگ سایه

مثال ساده:

|  |
| --- |
| h1 {  text-shadow: 2px 2px 4px gray;  } |

سایه خاکستری با فاصله ۲px در دو جهت و تاری ۴px ایجاد می‌شود.

چند سایه با هم:

|  |
| --- |
| h2 {  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;  } |

[text-shadow2](webdesign/css/text/text-shadow2.html)

سایه‌های متعدد ترکیب می‌شن و جلوه درخشان یا سه‌بعدی ایجاد می‌کنن

Whitespace

(کنترل نحوه‌ی رفتار فاصله‌ها و شکست خط‌ها)

این ویژگی مشخص می‌کند مرورگر با فاصله‌ها (space، تب، خط جدید) چطور رفتار کند.

به‌صورت پیش‌فرض مرورگر فاصله‌های اضافی را نادیده می‌گیرد و خطوط را می‌شکند.

مقادیر مهم:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `normal` | حالت پیش‌فرض؛ فاصله‌های اضافی حذف و متن در عرض عنصر می‌شکند |
| `nowrap` | شکستن خطوط ممنوع؛ همه در یک خط می‌ماند |
| `pre` | فاصله‌ها و خطوط دقیقاً مثل HTML (مثل تگ `<pre>`) حفظ می‌شوند |
| `pre-line` | فاصله‌ها نادیده گرفته می‌شوند ولی خط‌های جدید حفظ می‌شوند |
| `pre-wrap` | هم فاصله‌ها و هم خط‌های جدید حفظ می‌شوند و متن نیز در صورت نیاز می‌شکند |

مثال‌ها:

|  |
| --- |
| p.normal {  white-space: normal;  }  p.nowrap {  white-space: nowrap;  }  p.pre {  white-space: pre;  } |

|  |
| --- |
| <p class="normal">متن با فاصله‌های زیاد و چند خط.</p>  <p class="nowrap">متن طولانی بدون شکستن خط.</p>  <p class="pre">  متن با  فاصله‌ها و خطوط  دقیقاً مثل اینجا.  </p> |

[Whitespace](webdesign/css/text/Whitespace.html)

بسته به مقدار white-space، مرورگر یا فاصله‌ها رو نادیده می‌گیره یا دقیق نمایش می‌ده.

جمع‌بندی سریع:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | توضیح | نمونه مقدار | کاربرد |
| `word-spacing` | فاصله بین کلمات | `10px`, normal | تنظیم فضا بین کلمات |
| `text-shadow` | افزودن سایه به متن | 2px 2px 4px gray` | جلوه‌ی سه‌بعدی یا نئون |
| `white-space` | کنترل فاصله و شکست خط | `normal`, `pre`, `nowrap` | حفظ یا حذف فاصله‌ها و خطوط |

[Vertical-align](webdesign/css/vertical/base.html)

ویژگی vertical-align مشخص می‌کند که **یک عنصر inline یا inline-block** (مثل img, span, یا متن) نسبت به **خط پایه (baseline)** یا وسط خط یا بالای خط چطور تراز شود.

**نحوه نوشتن:**

|  |
| --- |
| <p>این یک تصویر <img src="icon.png" alt="آیکن"> در خط است.</p> |

|  |
| --- |
| img {  vertical-align: middle;  } |

[base](webdesign/css/vertical/base.html)

کاربردهای اصلی

الف) در عناصر inline یا inline-block

در این حالت، vertical-align تعیین می‌کند که عنصر نسبت به خط متن (baseline) کجا قرار بگیرد.

مثال:

|  |
| --- |
| img {  vertical-align: middle;  } |

حالا تصویر دقیقاً در وسط ارتفاع متن کنار خودش تراز می‌شود.

مقادیر متداول:

|  |  |  |
| --- | --- | --- |
| مقدار | توضیح | کاربرد |
| `baseline` | تراز با خط پایه‌ی متن (پیش‌فرض) | مناسب متن و آیکن |
| `top` | تراز در بالای بلندترین عنصر خط |  |
| `middle` | تراز در وسط خط | معمولاً برای آیکن در کنار متن |
| `bottom` | تراز در پایین خط |  |
| `text-top` | تراز با بالای متن اطراف |  |
| `text-bottom` | تراز با پایین متن اطراف |  |
| عدد یا درصد | مقدار دلخواه نسبت به baseline | مثلاً `vertical-align: 10px;` یا `-20%` |

مثال تصویری (متن + آیکن):

|  |
| --- |
| <p>  <img src="icon.png" style="vertical-align: middle;"> متن وسط‌چین  </p> |

[example](webdesign/css/vertical/example.html)

آیکن دقیقاً در وسط ارتفاع متن کنار آن قرار می‌گیرد (نه بالاتر، نه پایین‌تر).

**ب) در سلول‌های جدول (table-cell)**

در این حالت، vertical-align کنترل می‌کند محتوا در سلول جدول بالا، وسط یا پایین قرار بگیرد.

**مثال:**

|  |
| --- |
| <table border="1">  <tr>  <td style="height:100px; vertical-align: top;">بالا</td>  <td style="height:100px; vertical-align: middle;">وسط</td>  <td style="height:100px; vertical-align: bottom;">پایین</td>  </tr>  </table> |

[table](webdesign/css/vertical/table.html)

نتیجه:  
متن هر سلول در موقعیت مشخص‌شده (بالا، وسط، پایین) قرار می‌گیرد.

**نکته مهم**

vertical-align روی **عناصر block-level مثل div** اثری ندارد.  
اگر می‌خواهی محتوای داخل div را عموداً وسط قرار دهی، باید از روش‌های دیگر مثل **Flexbox** یا **Grid** استفاده کنی:

|  |
| --- |
| div {  display: flex;  align-items: center; /\* معادل vertical-align برای بلاک‌ها \*/  justify-content: center; /\* وسط افقی \*/  } |

[معرفی رنگ ها](#_Color)

[معرفی HEX color Mode](#_با_مقدار_HEX)

[معرفی RGB color mode](#_با_تابع_`rgb()`)

[معرفی RGBa color mode](#_با_تابع_`rgba()`)

[Color:](#_Color)

[Direction](webdesign/css/Direction)

(جهت نوشتار و چیدمان متن)

این ویژگی جهت قرارگیری متن و ترتیب نمایش عناصر درون یک بلوک را تعیین می‌کند.

معمولاً برای زبان‌هایی مثل فارسی و عربی (`rtl`) یا انگلیسی (`ltr`) استفاده می‌شود.

نحوه نوشتن:

|  |
| --- |
| p {  direction: rtl;  } |

[base](webdesign/css/Direction/base.html)

مقادیر:

|  |  |
| --- | --- |
| مقدار | توضیح |
| `ltr` | چپ به راست \*(Left To Right)\* — پیش‌فرض برای انگلیسی |
| `rtl` | راست به چپ \*(Right To Left)\* — مناسب برای فارسی و عربی |
| `inherit` | جهت را از عنصر والد به ارث می‌برد |

مثال:

|  |
| --- |
| <p style="direction: rtl;">این متن راست‌چین است.</p>  <p style="direction: ltr;">This text is left to right.</p> |

[sample](webdesign/css/Direction/sample.html)

در `rtl`، متن از سمت راست شروع می‌شود و نقطه‌گذاری‌ها هم بر اساس زبان تنظیم می‌گردد.

نکته:

برای نمایش درست در زبان‌های راست‌چین، معمولاً `direction` را همراه با `text-align` استفاده می‌کنیم:

|  |
| --- |
| body {  direction: rtl;  text-align: right;  } |

[farsi](webdesign/css/Direction/farsi.html)

## [Width](webdesign/css/Width)

(عرض عنصر)

این ویژگی میزان عرض (پهنای) یک عنصر را تعیین می‌کند.

نحوه نوشتن:

|  |
| --- |
| div {  width: 200px;  } |

[base](webdesign/css/Width/base.html)

مقدارها:

|  |  |  |
| --- | --- | --- |
| نوع مقدار | مثال | توضیح |
| پیکسل (px) | width: 300px; | عرض ثابت |
| درصد (%) | width: 50%; | عرض نسبت به عنصر والد |
| auto | مقدار پیش‌فرض — مرورگر به‌صورت خودکار تعیین می‌کند |  |
| max-content /min-content | بر اساس طول محتوای داخل تنظیم می‌شود (در CSS3) |  |

مثال:

|  |
| --- |
| <div style="width: 50%; background: lightblue;">نصف عرض صفحه</div> |

[sample](webdesign/css/Width/sample.html)

عرض این <div> نصف عرض والدش (مثلاً صفحه یا کانتینر) خواهد بود.

عرض این `<div>` نصف عرض والدش (مثلاً صفحه یا کانتینر) خواهد بود.

## [Height](webdesign/css/height)

(ارتفاع عنصر)

این ویژگی میزان ارتفاع (بلندی) یک عنصر را تعیین می‌کند.

نحوه نوشتن:

|  |
| --- |
| Div{  height: 150px;  } |

[base](webdesign/css/height/base.html)

مقدارها:

|  |  |  |
| --- | --- | --- |
| نوع مقدار | مثال | توضیح |
| پیکسل (px) | `height: 200px;` | ارتفاع ثابت |
| درصد (%) | `height: 50%;` | ارتفاع نسبت به والد (اگر والد ارتفاع مشخصی داشته باشد) |
| `auto` | ارتفاع متناسب با محتوای داخل (پیش‌فرض) |  |
| `max-content` / `min-content` | ارتفاع بر اساس محتوای واقعی متن |  |

مثال:

|  |
| --- |
| <div style="height: 100px; background: lightcoral;">  این باکس ارتفاع ثابت دارد.  </div> |

[sample](webdesign/css/height/sample.html)

باکس همیشه ۱۰۰ پیکسل ارتفاع دارد، حتی اگر متنش بیشتر شود (ممکن است متن بیرون بزند مگر overflow را کنترل کنید).

نکته مهم:

اگر از `height: 100%; استفاده می‌کنی، باید عنصر والد نیز ارتفاع مشخصی داشته باشد تا درصد معنی پیدا کند.

برای محتوای زیاد معمولاً بهتر است از `min-height` استفاده شود تا در صورت نیاز ارتفاع بیشتر شود:

|  |
| --- |
| div {  min-height: 200px;  } |

[min](webdesign/css/height/min.html)

## [Opacity](webdesign/css/opacity)

Opacity در CSS

خاصیت `opacity` میزان شفافیت یک المنت را مشخص می‌کند.

مقدار آن بین ۰ (کاملاً شفاف) تا ۱ (کاملاً مات) است.

نمونه کد:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .box {  width: 200px;  height: 200px;  background-color: blue;  margin: 20px;  }  .half-transparent {  opacity: 0.5; /\* نیمه شفاف \*/  }  .fully-transparent {  opacity: 0; /\* کاملاً شفاف (دیده نمی‌شود) \*/  }  </style>  </head>  <body>  <div class="box">مات (opacity:1)</div>  <div class="box half-transparent">نیمه شفاف (opacity:0.5)</div>  <div class="box fully-transparent">نامرئی (opacity:0)</div>  </body>  </html> |

[base](webdesign/css/opacity/base.html)

نکته مهم:

- وقتی `opacity` را روی یک المنت اعمال می‌کنید، تمام محتوای داخل آن (متن، عکس، فرزندان) هم شفاف می‌شوند.

- برای پس‌زمینه فقط از `background: rgba(255,0,0,0.5)` استفاده کنید اگر می‌خواهید فقط پس‌زمینه شفاف شود و محتوا مات بماند.

## معرفی attribute selector ها و regex

|  |  |  |
| --- | --- | --- |
| انتخابگر | مثال | توضیح کاربرد |
| `[attr]` | `[title]` | انتخاب عناصری که ویژگی `title` را دارند (مقدار مهم نیست). |
|  |  |  |
| `[attr=value]` | `[target="\_blank"]` | انتخاب عناصری که ویژگی آن‌ها دقیقاً برابر با مقدار مشخص شده است. |
| `[attr~=value]` | `[title~="flower"]` | انتخاب عناصری که در مقدار ویژگی آن‌ها، یک کلمهٔ کامل به نام `flower` وجود داشته باشد. |
| [attr|=value] | [class|="top"] | انتخاب عناصری که مقدار ویژگی آن‌ها دقیقاً با `top` شروع شود یا به صورت `top-` باشد (مثل `top-header`). |
| `[attr^=value]` | `[class^="top"]` | انتخاب عناصری که مقدار ویژگی آن‌ها با `top` شروع شود (مهم نیست کلمه کامل باشد). |
| `[attr$=value]` | `[class$="test"]` | انتخاب عناصری که مقدار ویژگی آن‌ها با `test` پایان یابد. |
| `[attr\*=value]` | `[class\*="te"]` | انتخاب عناصری که در مقدار ویژگی آن‌ها حاوی زیررشته `te` باشند. |

مثال:

|  |
| --- |
| a[target="\_blank"] {  background-color: yellow;  padding: 5px;  border: 1px solid black;  } |

## Background

Background-color

`background-color` - رنگ پس‌زمینه

تعریف:

تعیین رنگ زمینه برای یک المنت

مقادیر ممکن:

|  |
| --- |
| div {  background-color: red; /\* نام رنگ \*/  background-color: #ff0000; /\* کد هگز \*/  background-color: rgb(255,0,0); /\* RGB \*/  background-color: rgba(255,0,0,0.5); /\* RGB با شفافیت \*/  background-color: transparent; /\* شفاف \*/  } |

[Background-color](webdesign/css/Background/Background-color.html)

مثال عملی:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .box {  width: 200px;  height: 100px;  margin: 10px;  padding: 20px;  border: 2px solid black;  }    .solid-color {  background-color: #3498db; /\* آبی \*/  }    .transparent {  background-color: rgba(52, 152, 219, 0.3); /\* آبی نیمه شفاف \*/  }    .gradient {  background: linear-gradient(to right, #3498db, #2ecc71); /\* گرادیان \*/  }  </style>  </head>  <body>  <div class="box solid-color">پس‌زمینه آبی</div>  <div class="box transparent">پس‌زمینه نیمه شفاف</div>  <div class="box gradient">پس‌زمینه گرادیان</div>  </body>  </html> |

[Background-color2](webdesign/css/Background/background-color2.html)

Background-image

تعریف:

تعیین عکس به عنوان پس‌زمینه

کد:

|  |
| --- |
| selector {  background-image: url('آدرس-عکس');  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  width: 400px;  height: 300px;  border: 2px solid #333;  margin: 20px;  }    .nature-bg {  background-image: url('https://example.com/nature.jpg');  }    .pattern-bg {  background-image: url('https://example.com/pattern.png');  }    /\* چندین عکس پس‌زمینه \*/  .multiple-bg {  background-image:  url('https://example.com/texture.jpg'),  linear-gradient(45deg, #3498db, #2ecc71);  }  </style>  <div class="container nature-bg">پس‌زمینه عکس طبیعت</div>  <div class="container pattern-bg">پس‌زمینه پترن</div>  <div class="container multiple-bg">چندین پس‌زمینه</div> |

[Background-image](webdesign/css/Background/Background-image.html)

Background-repeat

تعریف:

تعیین نحوه تکرار عکس پس‌زمینه

مقادیر ممکن:

|  |
| --- |
| div {  background-repeat: repeat; /\* تکرار در دو جهت (پیش‌فرض) \*/  background-repeat: repeat-x; /\* تکرار فقط در جهت افقی \*/  background-repeat: repeat-y; /\* تکرار فقط در جهت عمودی \*/  background-repeat: no-repeat; /\* عدم تکرار \*/  background-repeat: space; /\* تکرار با فاصله \*/  background-repeat: round; /\* تکرار با کشیده شدن \*/  } |

مثال جامع و کاربردی

کد کامل HTML/CSS:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <style>  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  }    body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  padding: 20px;  background-color: #f5f5f5;  }    .header {  background-color: #2c3e50;  color: white;  padding: 20px;  text-align: center;  margin-bottom: 30px;  }    .demo-container {  display: flex;  flex-wrap: wrap;  gap: 20px;  justify-content: center;  }    .demo-box {  width: 300px;  height: 200px;  border: 3px solid #34495e;  border-radius: 10px;  padding: 15px;  display: flex;  align-items: center;  justify-content: center;  text-align: center;  font-weight: bold;  color: #2c3e50;  }    /\* مثال‌های background-color \*/  .color-solid {  background-color: #e74c3c;  }    .color-transparent {  background-color: rgba(231, 76, 60, 0.5);  }    /\* مثال‌های background-image \*/  .image-normal {  background-image: url('https://picsum.photos/300/200');  background-size: cover;  }    .image-pattern {  background-image: url('https://www.transparenttextures.com/patterns/45-degree-fabric-light.png');  }    /\* مثال‌های background-repeat \*/  .repeat-normal {  background-image: url('https://www.transparenttextures.com/patterns/checker-pattern.png');  background-repeat: repeat;  }    .repeat-x {  background-image: url('https://www.transparenttextures.com/patterns/striped-pattern.png');  background-repeat: repeat-x;  background-color: #3498db;  }    .repeat-y {  background-image: url('https://www.transparenttextures.com/patterns/vertical-stripes.png');  background-repeat: repeat-y;  background-color: #2ecc71;  }    .no-repeat {  background-image: url('https://picsum.photos/100/100');  background-repeat: no-repeat;  background-position: center;  background-color: #f1c40f;  }    .info-panel {  background-color: white;  padding: 20px;  margin: 20px auto;  max-width: 800px;  border-radius: 10px;  box-shadow: 0 4px 6px rgba(0,0,0,0.1);  }    .code-block {  background-color: #2c3e50;  color: #ecf0f1;  padding: 15px;  border-radius: 5px;  margin: 10px 0;  font-family: monospace;  direction: ltr;  text-align: left;  }  </style>  </head>  <body>  <div class="header">  <h1>🎨 آموزش خصوصیات Background در CSS</h1>  <p>Background-color | Background-image | Background-repeat</p>  </div>    <div class="info-panel">  <h2>🔍 نمونه‌های عملی</h2>  <div class="demo-container">  <div class="demo-box color-solid">background-color: #e74c3c</div>  <div class="demo-box color-transparent">background-color: rgba(231, 76, 60, 0.5)</div>  <div class="demo-box image-normal">background-image: url('image.jpg')</div>  <div class="demo-box image-pattern">background-image: pattern</div>  <div class="demo-box repeat-normal">background-repeat: repeat</div>  <div class="demo-box repeat-x">background-repeat: repeat-x</div>  <div class="demo-box repeat-y">background-repeat: repeat-y</div>  <div class="demo-box no-repeat">background-repeat: no-repeat</div>  </div>  </div>    <div class="info-panel">  <h2>📝 خلاصه و نکات کلیدی</h2>    <h3>🎯 Background-color:</h3>  <div class="code-block">  .element {<br>  &nbsp;&nbsp;background-color: #3498db;<br>  &nbsp;&nbsp;background-color: rgba(52, 152, 219, 0.5);<br>  &nbsp;&nbsp;background-color: transparent;<br>  }  </div>    <h3>🖼️ Background-image:</h3>  <div class="code-block">  .element {<br>  &nbsp;&nbsp;background-image: url('path/to/image.jpg');<br>  &nbsp;&nbsp;background-size: cover;<br>  &nbsp;&nbsp;background-position: center;<br>  }  </div>    <h3>🔁 Background-repeat:</h3>  <div class="code-block">  .element {<br>  &nbsp;&nbsp;background-repeat: repeat; /\* تکرار کامل \*/<br>  &nbsp;&nbsp;background-repeat: repeat-x; /\* تکرار افقی \*/<br>  &nbsp;&nbsp;background-repeat: repeat-y; /\* تکرار عمودی \*/<br>  &nbsp;&nbsp;background-repeat: no-repeat; /\* بدون تکرار \*/<br>  }  </div>  </div>    <div class="info-panel">  <h2>💡 نکات مهم برای دانشجویان</h2>  <ul style="line-height: 2; padding-right: 20px;">  <li>✅ <strong>Background-color</strong> همیشه در زیر Background-image نمایش داده می‌شود</li>  <li>✅ می‌توانید از چندین عکس به عنوان پس‌زمینه استفاده کنید</li>  <li>✅ مقدار پیش‌فرض <strong>background-repeat</strong> برابر با <strong>repeat</strong> است</li>  <li>✅ برای عکس‌های بزرگ از <strong>no-repeat</strong> استفاده کنید</li>  <li>✅ برای پترن‌های کوچک از <strong>repeat</strong> استفاده کنید</li>  <li>✅ از <strong>background-size: cover</strong> برای پوشش کامل زمینه استفاده کنید</li>  </ul>  </div>  </body>  </html> |

[complete](webdesign/css/Background/complete.html)

Background-position

۱. ویژگی `background-position`

این ویژگی موقعیت تصویر زمینه را مشخص می‌کند.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش background-position</title>  <style>  .box {  width: 300px;  height: 200px;  border: 2px solid #333;  background-image: url('image.jpg');  background-repeat: no-repeat;  margin: 20px;  }    .center {  background-position: center;  }    .top-right {  background-position: top right;  }    .custom {  background-position: 70% 25%;  }  </style>  </head>  <body>  <div class="box center">موقعیت: مرکز</div>  <div class="box top-right">موقعیت: بالا راست</div>  <div class="box custom">موقعیت: 70% 25%</div>  </body>  </html> |

[Background-position](webdesign/css/Background/Background-position.html)

Background-origin

این ویژگی مشخص می‌کند که تصویر زمینه نسبت به کدام ناحیه (حاشیه، padding یا محتوا) قرار بگیرد.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش background-origin</title>  <style>  .box {  width: 300px;  height: 150px;  border: 15px dashed #666;  padding: 30px;  background-image: url('pattern.png');  background-repeat: no-repeat;  margin: 20px;  }    .border-box {  background-origin: border-box;  }    .padding-box {  background-origin: padding-box;  }    .content-box {  background-origin: content-box;  }  </style>  </head>  <body>  <div class="box border-box">background-origin: border-box</div>  <div class="box padding-box">background-origin: padding-box</div>  <div class="box content-box">background-origin: content-box</div>  </body>  </html> |

[Background-origin](webdesign/css/Background/Background-origin.html)

Background-attachment

این ویژگی مشخص می‌کند که تصویر زمینه هنگام اسکرول ثابت بماند یا همراه با محتوا حرکت کند.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش background-attachment</title>  <style>  body {  height: 200vh;  }    .section {  height: 100vh;  background-image: url('background.jpg');  background-repeat: no-repeat;  background-size: cover;  display: flex;  justify-content: center;  align-items: center;  color: white;  font-size: 2em;  }    .scroll {  background-attachment: scroll;  }    .fixed {  background-attachment: fixed;  }    .local {  background-attachment: local;  height: 150vh;  overflow: auto;  }  </style>  </head>  <body>  <div class="section scroll">background-attachment: scroll</div>  <div class="section fixed">background-attachment: fixed</div>  <div class="section local">  <div style="height: 200vh">background-attachment: local</div>  </div>  </body>  </html> |

[Background-attachment](webdesign/css/Background/Background-attachment.html)

مثال ترکیبی

در این مثال، هر سه ویژگی را با هم ترکیب کرده‌ایم:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>مثال ترکیبی</title>  <style>  .hero {  height: 100vh;  background-image: url('hero-image.jpg');  background-position: center center;  background-origin: border-box;  background-attachment: fixed;  background-repeat: no-repeat;  background-size: cover;  border: 10px solid #333;  padding: 20px;  color: white;  display: flex;  justify-content: center;  align-items: center;  font-size: 2em;  }  </style>  </head>  <body>  <div class="hero">  <h1>بخش هیرو با تصویر زمینه ثابت</h1>  </div>  </body>  </html> |

[combine](webdesign/css/Background/combine.html)

Background-size

Background-size

خاصیت `background-size` برای کنترل اندازه تصویر پس‌زمینه استفاده می‌شود. این خاصیت تعیین می‌کند که تصویر پس‌زمینه چه اندازه‌ای داشته باشد و چگونه در عنصر نمایش داده شود.

مقادیر اصلی background-size:

- `auto`: اندازه اصلی تصویر حفظ می‌شود

- `cover`: تصویر کل عنصر را می‌پوشاند (ممکن است بخشی از تصویر بریده شود)

- `contain`: کل تصویر در عنصر نمایش داده می‌شود (ممکن است فضای خالی ایجاد شود)

- `طول و عرض`: مقادیر پیکسلی یا درصدی برای عرض و ارتفاع

- `initial`: مقدار پیش‌فرض

نکته: برای `background-size: cover` تصویر کوچک نمی‌شود و همیشه کل عنصر را می‌پوشاند، اما ممکن است لبه‌های تصویر بریده شوند.

|  |
| --- |
| <style>  .box {  width: 300px;  height: 200px;  margin: 20px;  border: 2px solid #333;  padding: 20px;  color: black;  font-weight: bold;  }  /\* پس‌زمینه کامل با تمام ویژگی‌ها \*/  .background-full {  background:  url('https://via.placeholder.com/100x100/ff6b6b/ffffff?text=BG1')  no-repeat  center  fixed  #f8f9fa;  }  /\* چندین لایه پس‌زمینه \*/  .background-multiple {  background:  url('https://via.placeholder.com/100x100/4ecdc4/ffffff?text=IMG1') top left no-repeat,  url('https://via.placeholder.com/100x100/45b7d1/ffffff?text=IMG2') bottom right no-repeat,  linear-gradient(45deg, #ff9a9e, #fad0c4);  }  /\* فقط رنگ پس‌زمینه \*/  .background-color-only {  background: #e9ecef;  }  /\* تصویر با موقعیت و رنگ مشخص \*/  .background-image-position {  background:  url('https://via.placeholder.com/150x100/ff6b6b/ffffff')  right bottom  no-repeat  #dee2e6;  }  </style>  <div class="box background-full">پس‌زمینه کامل: تصویر + موقعیت + تکرار + ثابت + رنگ</div>  <div class="box background-multiple">چندین لایه پس‌زمینه (تصاویر و گرادیان)</div>  <div class="box background-color-only">فقط رنگ پس‌زمینه</div>  <div class="box background-image-position">تصویر در گوشه پایین راست با رنگ زمینه</div> |

[Background-size](webdesign/css/Background/Background-size.html)

Background

خاصیت `background` در CSS یک خاصیت جامع و مختصر است که امکان تعیین چندین ویژگی پس‌زمینه را به صورت همزمان فراهم می‌کند. این خاصیت می‌تواند شامل رنگ پس‌زمینه، تصویر، موقعیت، تکرار، اندازه و سایر ویژگی‌ها باشد.

مهمترین ویژگی‌هایی که می‌توان با background کنترل کرد:

- `background-color`: رنگ پس‌زمینه

- `background-image`: تصویر پس‌زمینه

- `background-position`: موقعیت تصویر

- `background-size`: اندازه تصویر

- `background-repeat`: نحوه تکرار تصویر

- `background-attachment`: ثابت یا متحرک بودن تصویر

نکته مهم: ترتیب نوشتن مقادیر در background مهم نیست، اما توصیه می‌شود برای خوانایی بهتر از ترتیب استاندارد استفاده شود.

|  |
| --- |
| <style>  .box {  width: 300px;  height: 200px;  margin: 20px;  border: 2px solid #333;  padding: 20px;  color: black;  font-weight: bold;  }  /\* پس‌زمینه کامل با تمام ویژگی‌ها \*/  .background-full {  background:  url('https://via.placeholder.com/100x100/ff6b6b/ffffff?text=BG1')  no-repeat  center  fixed  #f8f9fa;  }  /\* چندین لایه پس‌زمینه \*/  .background-multiple {  background:  url('https://via.placeholder.com/100x100/4ecdc4/ffffff?text=IMG1') top left no-repeat,  url('https://via.placeholder.com/100x100/45b7d1/ffffff?text=IMG2') bottom right no-repeat,  linear-gradient(45deg, #ff9a9e, #fad0c4);  }  /\* فقط رنگ پس‌زمینه \*/  .background-color-only {  background: #e9ecef;  }  /\* تصویر با موقعیت و رنگ مشخص \*/  .background-image-position {  background:  url('https://via.placeholder.com/150x100/ff6b6b/ffffff')  right bottom  no-repeat  #dee2e6;  }  </style>  <div class="box background-full">پس‌زمینه کامل: تصویر + موقعیت + تکرار + ثابت + رنگ</div>  <div class="box background-multiple">چندین لایه پس‌زمینه (تصاویر و گرادیان)</div>  <div class="box background-color-only">فقط رنگ پس‌زمینه</div>  <div class="box background-image-position">تصویر در گوشه پایین راست با رنگ زمینه</div> |

[background](webdesign/css/Background/background.html)

Linear-gradient

تابع `linear-gradient` برای ایجاد gradient (تدریج رنگ) خطی استفاده می‌شود. این تابع یک تصویر gradient ایجاد می‌کند که شامل انتقال تدریجی بین دو یا چند رنگ است.

ساختار اصلی:

|  |
| --- |
| linear-gradient(جهت, رنگ-اول, رنگ-دوم, ...) |

جهت‌های مختلف:

- `to top`: از پایین به بالا

- `to bottom`: از بالا به پایین (پیش‌فرض)

- `to left`: از راست به چپ

- `to right`: از چپ به راست

- `to top left`: به گوشه بالا چپ

- `45deg`: زاویه 45 درجه

انواع gradient:

- گرادیان خطی (Linear Gradient): انتقال رنگ در یک خط مستقیم

- گرادیان دایره‌ای (Radial Gradient): انتقال رنگ از مرکز به بیرون

|  |
| --- |
| <style>  .gradient-box {  width: 300px;  height: 200px;  margin: 20px;  border: 2px solid #333;  display: inline-block;  color: black;  font-weight: bold;  text-align: center;  padding-top: 80px;  }  /\* گرادیان‌های خطی پایه \*/  .linear-simple {  background: linear-gradient(#ff6b6b, #4ecdc4);  }  .linear-direction-right {  background: linear-gradient(to right, #ff6b6b, #4ecdc4);  }  .linear-diagonal {  background: linear-gradient(45deg, #ff6b6b, #4ecdc4);  }  .linear-angle {  background: linear-gradient(120deg, #ff6b6b, #4ecdc4);  }  /\* گرادیان با چند رنگ \*/  .linear-multi {  background: linear-gradient(to right, #ff6b6b, #ffd166, #06d6a0, #118ab2);  }  /\* گرادیان با موقعیت مشخص رنگ‌ها \*/  .linear-position {  background: linear-gradient(to right, #ff6b6b 0%, #4ecdc4 50%, #45b7d1 100%);  }  .linear-sharp {  background: linear-gradient(to right, #ff6b6b 0%, #ff6b6b 50%, #4ecdc4 50%, #4ecdc4 100%);  }  /\* گرادیان با شفافیت \*/  .linear-transparent {  background: linear-gradient(to bottom, rgba(255,107,107,0.8), rgba(78,205,196,0.3));  }  /\* گرادیان‌های دایره‌ای \*/  .radial-simple {  background: radial-gradient(circle, #ff6b6b, #4ecdc4);  }  .radial-ellipse {  background: radial-gradient(ellipse, #ff6b6b, #4ecdc4, #45b7d1);  }  .radial-position {  background: radial-gradient(circle at top left, #ff6b6b, #4ecdc4);  }  .radial-size {  background: radial-gradient(50px 50px at center, #ff6b6b, #4ecdc4);  }  </style>  <div class="gradient-box linear-simple">گرادیان عمودی ساده</div>  <div class="gradient-box linear-direction-right">گرادیان افقی (راست)</div>  <div class="gradient-box linear-diagonal">گرادیان 45 درجه</div>  <div class="gradient-box linear-angle">گرادیان 120 درجه</div>  <div class="gradient-box linear-multi">گرادیان چندرنگی</div>  <div class="gradient-box linear-position">گرادیان با موقعیت رنگ</div>  <div class="gradient-box linear-sharp">گرادیان با مرز مشخص</div>  <div class="gradient-box linear-transparent">گرادیان شفاف</div>  <div class="gradient-box radial-simple">گرادیان دایره‌ای ساده</div>  <div class="gradient-box radial-ellipse">گرادیان بیضی شکل</div>  <div class="gradient-box radial-position">گرادیان دایره‌ای در گوشه</div>  <div class="gradient-box radial-size">گرادیان دایره‌ای با سایز مشخص</div> |

[Linear-gradient](webdesign/css/Background/Linear-gradient.html)

Radial-gradient

گرادیان دایره‌ای (Radial-gradient) یک گرادیان است که از مرکز به سمت بیرون پخش می‌شود. برخلاف linear-gradient که در یک خط مستقیم حرکت می‌کند، radial-gradient به صورت دایره‌ای یا بیضی شکل گسترش می‌یابد.

ساختار اصلی:

|  |
| --- |
| background: radial-gradient(شکل at موقعیت, رنگ-شروع, رنگ-پایان); |

پارامترهای مهم:

- شکل: `circle` (دایره) یا `ellipse` (بیضی)

- موقعیت: `center`، `top`، `left`، `bottom`، `right` یا مختصات دقیق

- اندازه: `closest-side`، `farthest-side`، `closest-corner`، `farthest-corner`

|  |
| --- |
| <style>  .gradient-box {  width: 250px;  height: 200px;  margin: 15px;  border: 2px solid #333;  display: inline-block;  color: white;  font-weight: bold;  text-align: center;  padding-top: 80px;  }  /\* گرادیان دایره‌ای ساده \*/  .radial-simple {  background: radial-gradient(circle, #ff6b6b, #4ecdc4);  }  /\* گرادیان بیضی شکل \*/  .radial-ellipse {  background: radial-gradient(ellipse, #ff6b6b, #4ecdc4, #45b7d1);  }  /\* گرادیان با موقعیت مشخص \*/  .radial-position {  background: radial-gradient(circle at top left, #ff6b6b, #4ecdc4);  }  .radial-position-pixel {  background: radial-gradient(circle at 70% 30%, #ff6b6b, #4ecdc4);  }  /\* گرادیان با اندازه مشخص \*/  .radial-size {  background: radial-gradient(50px 50px at center, #ff6b6b, #4ecdc4);  }  /\* گرادیان با چند رنگ \*/  .radial-multi {  background: radial-gradient(circle, #ff6b6b, #ffd166, #06d6a0, #118ab2);  }  /\* گرادیان با موقعیت رنگ‌ها \*/  .radial-color-stops {  background: radial-gradient(circle, #ff6b6b 0%, #ffd166 30%, #06d6a0 70%, #118ab2 100%);  }  /\* گرادیان با اندازه‌های مختلف \*/  .radial-closest-side {  background: radial-gradient(closest-side circle at 30% 30%, #ff6b6b, #4ecdc4);  }  .radial-farthest-corner {  background: radial-gradient(farthest-corner circle at 30% 30%, #ff6b6b, #4ecdc4);  }  </style>  <div class="gradient-box radial-simple">دایره‌ای ساده</div>  <div class="gradient-box radial-ellipse">بیضی شکل</div>  <div class="gradient-box radial-position">موقعیت: بالا چپ</div>  <div class="gradient-box radial-position-pixel">موقعیت: 70% 30%</div>  <div class="gradient-box radial-size">سایز: 50px 50px</div>  <div class="gradient-box radial-multi">چند رنگ</div>  <div class="gradient-box radial-color-stops">موقعیت رنگ‌ها</div>  <div class="gradient-box radial-closest-side">closest-side</div>  <div class="gradient-box radial-farthest-corner">farthest-corner</div> |

[Radial-gradient](webdesign/css/Background/Radial-gradient.html)

Repeating-linear-gradient

گرادیان خطی تکراری مشابه linear-gradient است، اما الگو به صورت متناوب تکرار می‌شود. این ویژگی برای ایجاد الگوهای خطی منظم بسیار کاربردی است.

ساختار اصلی:

|  |
| --- |
| background: repeating-linear-gradient(جهت, رنگ-موقعیت, رنگ-موقعیت, ...); |

مثال:

|  |
| --- |
| <style>  .gradient-box {  width: 250px;  height: 200px;  margin: 15px;  border: 2px solid #333;  display: inline-block;  color: black;  font-weight: bold;  text-align: center;  padding-top: 80px;  }  /\* گرادیان خطی تکراری ساده \*/  .repeating-linear-simple {  background: repeating-linear-gradient(45deg, #ff6b6b, #ff6b6b 10px, #4ecdc4 10px, #4ecdc4 20px);  }  /\* گرادیان خطی تکراری افقی \*/  .repeating-linear-horizontal {  background: repeating-linear-gradient(to right, #ff6b6b, #ff6b6b 15px, #4ecdc4 15px, #4ecdc4 30px);  }  /\* گرادیان خطی تکراری عمودی \*/  .repeating-linear-vertical {  background: repeating-linear-gradient(to bottom, #ff6b6b, #ff6b6b 10px, #4ecdc4 10px, #4ecdc4 20px);  }  /\* گرادیان خطی تکراری با چند رنگ \*/  .repeating-linear-multi {  background: repeating-linear-gradient(  to right,  #ff6b6b,  #ff6b6b 10px,  #ffd166 10px,  #ffd166 20px,  #06d6a0 20px,  #06d6a0 30px,  #118ab2 30px,  #118ab2 40px  );  }  /\* گرادیان خطی تکراری نرم \*/  .repeating-linear-soft {  background: repeating-linear-gradient(  -45deg,  rgba(255, 107, 107, 0.1),  rgba(255, 107, 107, 0.1) 10px,  rgba(78, 205, 196, 0.1) 10px,  rgba(78, 205, 196, 0.1) 20px  );  }  /\* الگوی راه‌راه \*/  .repeating-linear-stripes {  background: repeating-linear-gradient(  0deg,  transparent,  transparent 5px,  #f8f9fa 5px,  #f8f9fa 10px  );  }  </style>  <div class="gradient-box repeating-linear-simple">مورب تکراری</div>  <div class="gradient-box repeating-linear-horizontal">افقی تکراری</div>  <div class="gradient-box repeating-linear-vertical">عمودی تکراری</div>  <div class="gradient-box repeating-linear-multi">چند رنگ تکراری</div>  <div class="gradient-box repeating-linear-soft">تکراری نرم</div>  <div class="gradient-box repeating-linear-stripes">الگوی راه‌راه</div> |

[Repeating-linear-gradient](webdesign/css/Background/Repeating-linear-gradient.html)

Repeating-radial-gradient

گرادیان دایره‌ای تکراری مشابه radial-gradient است، اما الگو به صورت متناوب از مرکز به بیرون تکرار می‌شود.

ساختار اصلی:

|  |
| --- |
| background: repeating-radial-gradient(شکل at موقعیت, رنگ-موقعیت, رنگ-موقعیت, ...); |

مثال:

|  |
| --- |
| <style>  .gradient-box {  width: 250px;  height: 200px;  margin: 15px;  border: 2px solid #333;  display: inline-block;  color: white;  font-weight: bold;  text-align: center;  padding-top: 80px;  }  /\* گرادیان دایره‌ای تکراری ساده \*/  .repeating-radial-simple {  background: repeating-radial-gradient(  circle,  #ff6b6b,  #ff6b6b 10px,  #4ecdc4 10px,  #4ecdc4 20px  );  }  /\* گرادیان دایره‌ای تکراری با فاصله بیشتر \*/  .repeating-radial-spaced {  background: repeating-radial-gradient(  circle,  #ff6b6b,  #ff6b6b 20px,  #4ecdc4 20px,  #4ecdc4 40px  );  }  /\* گرادیان دایره‌ای تکراری با چند رنگ \*/  .repeating-radial-multi {  background: repeating-radial-gradient(  circle,  #ff6b6b,  #ff6b6b 10px,  #ffd166 10px,  #ffd166 20px,  #06d6a0 20px,  #06d6a0 30px,  #118ab2 30px,  #118ab2 40px  );  }  /\* گرادیان دایره‌ای تکراری با موقعیت \*/  .repeating-radial-position {  background: repeating-radial-gradient(  circle at 30% 20%,  #ff6b6b,  #ff6b6b 10px,  #4ecdc4 10px,  #4ecdc4 20px  );  }  /\* گرادیان بیضی تکراری \*/  .repeating-radial-ellipse {  background: repeating-radial-gradient(  ellipse,  #ff6b6b,  #ff6b6b 10px,  #4ecdc4 10px,  #4ecdc4 20px,  #45b7d1 20px,  #45b7d1 30px  );  }  /\* الگوی دایره‌ای ظریف \*/  .repeating-radial-subtle {  background: repeating-radial-gradient(  circle at center,  rgba(255, 107, 107, 0.3),  rgba(255, 107, 107, 0.3) 8px,  rgba(78, 205, 196, 0.3) 8px,  rgba(78, 205, 196, 0.3) 16px  );  }  </style>  <div class="gradient-box repeating-radial-simple">دایره‌ای تکراری ساده</div>  <div class="gradient-box repeating-radial-spaced">دایره‌ای با فاصله</div>  <div class="gradient-box repeating-radial-multi">دایره‌ای چند رنگ</div>  <div class="gradient-box repeating-radial-position">دایره‌ای با موقعیت</div>  <div class="gradient-box repeating-radial-ellipse">بیضی تکراری</div>  <div class="gradient-box repeating-radial-subtle">دایره‌ای ظریف</div> |

[Repeating-radial-gradient](webdesign/css/Background/Repeating-radial-gradient.html)

نکات مهم :

1. تفاوت اصلی:

- `radial-gradient`: یک بار اجرا می‌شود

- `repeating-\*`: الگو به صورت نامحدود تکرار می‌شود

2. کاربردها:

- ایجاد الگوهای پس‌زمینه

- طراحی‌های خلاقانه

- افکت‌های بصری جذاب

3. پشتیبانی مرورگرها: تمام مرورگرهای مدرن از این ویژگی‌ها پشتیبانی می‌کنند.

## بررسی سایت css gradient generator

ابزارهای تولید گرادیان CSS

ابزارهای آنلاین مختلفی برای ساخت گرادیان وجود دارند که هر کدام ویژگی‌های منحصربه‌فردی ارائه می‌دهند:

|  |  |  |
| --- | --- | --- |
| ابزار | ویژگی‌های کلیدی | مناسب برای |
| [Ultimate CSS Gradient Generator](https://www.colorzilla.com/gradient-editor/) | رابط فتوشاپی، پشتیبانی از چند ایست رنگ، کنترل opacity، سازگاری کراس-براوزر | کاربران حرفه‌ای که نیاز به دقت بالا دارند |
| [CSS Gradient](https://cssgradient.io/) | رابط ساده و بصری، گالری از گرادیان‌های آماده، کد خروجی تمیز | شروع سریع و الهام گرفتن از نمونه‌های آماده |
| [Gradient Generator (Josh Comeau)](https://www.joshwcomeau.com/gradient-generator/) | الگوریتم‌های پیشرفته برای ایجاد گرادیان‌های بصری‌تر و طبیعی‌تر | طراحی‌های با کیفیت بالا و حرفه‌ای |

## [ساخت کامپوننت parallax](webdesign/css/parallax)

کد نمونه: پارالاکس ساده با `background-attachment`

|  |
| --- |
| <style>  .parallax-section {  height: 100vh; /\* بلوک تمام ارتفاع صفحه \*/  background-image: url('path-to-your-image.jpg');  background-attachment: fixed; /\* ایجاد افکت پارالاکس \*/  background-position: center;  background-size: cover;  }  </style>  <div class="parallax-section"></div> |

[base](webdesign/css/parallax/base.html)

## @import

[تقسیم بندی ماژولار css با استفاده از @import](webdesign/css/import)

مفهوم ماژولار سازی در CSS

ماژولار سازی به معنای تقسیم کدهای CSS به فایل‌های کوچک‌تر و سازمان‌یافته بر اساس functionality است. این کار مزایای زیر را دارد:

- قابلیت نگهداری بهتر

- قابلیت استفاده مجدد

- توسعه تیمی آسان‌تر

- خطایابی سریع‌تر

ساختار پیشنهادی برای پروژه:

|  |
| --- |
| css/  ├── base/  │ ├── reset.css  │ ├── typography.css  │ └── variables.css  ├── components/  │ ├── buttons.css  │ ├── forms.css  │ └── cards.css  ├── layout/  │ ├── header.css  │ ├── footer.css  │ └── grid.css  └── main.css |

نحوه استفاده از import@

|  |
| --- |
| /\* فایل main.css \*/  @import url('./base/reset.css');  @import url('./base/variables.css');  @import url('./base/typography.css');  @import url('./layout/grid.css');  @import url('./layout/header.css');  @import url('./layout/footer.css');  @import url('./components/buttons.css');  @import url('./components/forms.css');  /\* استایل‌های اصلی پروژه \*/  .container {  max-width: 1200px;  margin: 0 auto;  } |

[base](webdesign/css/import/base.html)

## [Padding](webdesign/css/Padding)

تعریف Padding

Paddingفضای بین محتوای یک element و border آن است. Padding باعث می‌شود عناصر فضای تنفسی داشته باشند و خوانایی آن‌ها افزایش یابد.

روش‌های تعریف Padding

1. روش تک مقداری (یکسان برای همه جهات)

|  |
| --- |
| .element {  padding: 20px;  }  /\*  بالا: 20px  راست: 20px  پایین: 20px  چپ: 20px  \*/ |

[All](webdesign/css/Padding/all.html)

1. روش دو مقداری (عمودی - افقی)

|  |
| --- |
| .element {  padding: 15px 25px;  }  /\*  بالا: 15px  راست: 25px  پایین: 15px  چپ: 25px  \*/ |

1. روش سه مقداری

|  |
| --- |
| .element {  padding: 10px 20px 15px;  }  /\*  بالا: 10px  راست: 20px  پایین: 15px  چپ: 20px  \*/ |

1. روش چهار مقداری (همه جهات جداگانه)

|  |
| --- |
| .element {  padding: 10px 15px 20px 25px;  }  /\*  بالا: 10px  راست: 15px  پایین: 20px  چپ: 25px  \*/ |

Properties جداگانه برای هر جهت

|  |
| --- |
| .element {  padding-top: 10px;  padding-right: 15px;  padding-bottom: 20px;  padding-left: 25px;  } |

مثال‌های عملی

|  |
| --- |
| <style>  .box1 {  background-color: lightblue;  padding: 20px;  }  .box2 {  background-color: lightgreen;  padding: 30px 15px;  }  .box3 {  background-color: lightyellow;  padding: 10px 25px 40px;  }  .box4 {  background-color: lightpink;  padding: 5px 10px 15px 20px;  }  </style>  <div class="box1">پدینگ یکسان از همه طرف</div>  <div class="box2">پدینگ عمودی و افقی</div>  <div class="box3">پدینگ بالا، چپ-راست، پایین</div>  <div class="box4">پدینگ همه جهات جداگانه</div> |

[inAction](webdesign/css/Padding/inaction.html)

## [border](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border)

Border-style

Border-Style مشخص کننده‌ی سبک و طرح حاشیه یک المان است. این ویژگی تعیین می‌کند که خطوط حاشیه به چه شکلی نمایش داده شوند.

- solid: خط ساده و پیوسته

- dashed: خط چین (خط‌های بلند)

- dotted: نقطه چین (نقاط کوچک)

- double: خط دوتایی

- groove: حاشیه فرورفته سه بعدی

- ridge: حاشیه برجسته سه بعدی

- inset: المان فرورفته

- outset: المان برجسته

- none: بدون حاشیه

- hidden: حاشیه مخفی (مشابه none)

|  |
| --- |
| <style>  .border-example {  background-color: #f0f0f0;  margin: 5px 0;  padding: 10px;  }  .solid { border-style: solid; }  .dashed { border-style: dashed; }  .dotted { border-style: dotted; }  .double { border-style: double; }  .groove { border-style: groove; }  .ridge { border-style: ridge; }  .inset { border-style: inset; }  .outset { border-style: outset; }  .none { border-style: none; }  .hidden { border-style: hidden; }  </style>  <div class="border-example solid">border-style: solid</div>  <div class="border-example dashed">border-style: dashed</div>  <div class="border-example dotted">border-style: dotted</div>  <div class="border-example double">border-style: double</div>  <div class="border-example groove">border-style: groove</div>  <div class="border-example ridge">border-style: ridge</div>  <div class="border-example inset">border-style: inset</div>  <div class="border-example outset">border-style: outset</div>  <div class="border-example none">border-style: none</div>  <div class="border-example hidden">border-style: hidden</div> |

[Border-style](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\Border-style.html)

در این تمرین: تمامی استایل‌های مختلف border-style را مشاهده کردید. هر کدام از این استایل‌ها طرح متفاوتی برای حاشیه ایجاد می‌کنند.

Border-width

Border-Width ضخامت حاشیه را مشخص می‌کند. این ویژگی می‌تواند با واحدهای مختلف یا کلمات کلیدی تعریف شود.

- کلمات کلیدی: thin (نازک), medium (متوسط), thick (ضخیم)

- مقادیر عددی: با واحدهایی مانند px, em, rem و...

- مقادیر ترکیبی: می‌توان برای هر طرف المان ضخامت متفاوت تعریف کرد

|  |
| --- |
| <style>  .width-example {  background-color: #f0f0f0;  border-style: solid;  margin: 5px 0;  padding: 10px;  }  .thin { border-width: thin; }  .medium { border-width: medium; }  .thick { border-width: thick; }  .pixel1 { border-width: 1px; }  .pixel3 { border-width: 3px; }  .pixel5 { border-width: 5px; }  .pixel10 { border-width: 10px; }  .mixed { border-width: 2px 5px 10px 15px; }  </style>  <div class="width-example thin">border-width: thin</div>  <div class="width-example medium">border-width: medium</div>  <div class="width-example thick">border-width: thick</div>  <div class="width-example pixel1">border-width: 1px</div>  <div class="width-example pixel3">border-width: 3px</div>  <div class="width-example pixel5">border-width: 5px</div>  <div class="width-example pixel10">border-width: 10px</div>  <div class="width-example mixed">border-width: 2px 5px 10px 15px</div> |

[Border-width](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\Border-width.html)

در این تمرین: مشاهده کردید که چگونه می‌توان ضخامت حاشیه را با کلمات کلیدی و مقادیر عددی تنظیم کرد. همچنین یاد گرفتید که چگونه برای هر چهار طرف المان ضخامت متفاوت تعریف کنید.

Border-color

Border-Color رنگ حاشیه را مشخص می‌کند. این ویژگی می‌تواند با روش‌های مختلف تعریف رنگ استفاده شود.

- نام رنگ: red, blue, green و...

- کد HEX: #ff6b6b, #ffffff و...

- RGB: rgb(255, 0, 0), rgb(106, 90, 205) و...

- مقادیر ترکیبی: رنگ‌های مختلف برای هر طرف

- transparent: حاشیه شفاف

|  |
| --- |
| <style>  .color-example {  background-color: #f0f0f0;  border-style: solid;  border-width: 3px;  margin: 5px 0;  padding: 10px;  }  .red { border-color: red; }  .blue { border-color: blue; }  .green { border-color: green; }  .hex { border-color: #ff6b6b; }  .rgb { border-color: rgb(106, 90, 205); }  .mixed { border-color: red blue green purple; }  .transparent { border-color: transparent; }  </style>  <div class="color-example red">border-color: red</div>  <div class="color-example blue">border-color: blue</div>  <div class="color-example green">border-color: green</div>  <div class="color-example hex">border-color: #ff6b6b</div>  <div class="color-example rgb">border-color: rgb(106, 90, 205)</div>  <div class="color-example mixed">border-color: red blue green purple</div>  <div class="color-example transparent">border-color: transparent</div> |

[Border-color](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\Border-color.html)

در این تمرین: با روش‌های مختلف تعریف رنگ برای حاشیه آشنا شدید. یاد گرفتید که چگونه از نام رنگ، کد HEX و مقادیر RGB استفاده کنید و همچنین چگونه برای هر طرف المان رنگ متفاوت تعریف نمایید.

Border

Border Shorthand در CSS

هدف آموزش

در این جلسه می‌خواهیم با روش مختصرنویسی (Shorthand) برای border آشنا شویم. تاکنون با ویژگی‌های جداگانه border آشنا شدید، اکنون یاد می‌گیرید چگونه می‌توان همه این ویژگی‌ها را در یک خط کد نوشت.

مرور مفاهیم قبلی

تا اینجا با این ویژگی‌ها آشنا شدید:

- border-style: تعیین سبک حاشیه (solid, dashed, dotted, ...)

- border-width: تعیین ضخامت حاشیه (thin, medium, thick, px)

- border-color: تعیین رنگ حاشیه (red, #ff0000, rgb(255,0,0))

معرفی Border Shorthand

Border Shorthand یک روش کوتاه‌تر برای تعریف تمام ویژگی‌های border در یک خط است.

ساختار کلی:

|  |
| --- |
| border: width style color; |

روش‌های استفاده از Border Shorthand

روش ۱: تعریف کامل border

|  |
| --- |
| <style>  .box1 { border: 2px solid red; }  .box2 { border: 5px dashed blue; }  .box3 { border: 3px dotted green; }  </style>  <div class="box1">border: 2px solid red</div>  <div class="box2">border: 5px dashed blue</div>  <div class="box3">border: 3px dotted green</div> |

[Border Shorthand](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\Border%20Shorthand.html)

روش ۲: تعریف border برای هر طرف بصورت جداگانه

|  |
| --- |
| <style>  .box4 {  border-top: 4px solid red;  border-right: 2px dashed blue;  border-bottom: 3px dotted green;  border-left: 5px double purple;  }  </style>  <div class="box4">Borderهای مختلف برای هر طرف</div> |

[separately](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\separately.html)

تمرین عملی

|  |
| --- |
| <style>  .example {  background-color: #f8f9fa;  margin: 10px 0;  padding: 15px;  }  /\* تعریف border کامل \*/  .full-border {  border: 3px solid #007bff;  }  /\* تعریف border با رنگ متفاوت \*/  .colored-border {  border: 4px dashed #28a745;  }  /\* تعریف border نازک \*/  .thin-border {  border: 1px solid #dc3545;  }  /\* تعریف border برای یک طرف خاص \*/  .single-side {  border-bottom: 5px solid #ffc107;  }  /\* تعریف borderهای مختلف برای جهات مختلف \*/  .multiple-sides {  border-top: 2px dotted #6f42c1;  border-right: 4px dashed #fd7e14;  border-bottom: 3px double #20c997;  border-left: 1px solid #e83e8c;  }  </style>  <div class="example full-border">border: 3px solid #007bff</div>  <div class="example colored-border">border: 4px dashed #28a745</div>  <div class="example thin-border">border: 1px solid #dc3545</div>  <div class="example single-side">border-bottom: 5px solid #ffc107</div>  <div class="example multiple-sides">Borderهای مختلف برای هر جهت</div> |

[inAction](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\inAction.html)

نکات مهم برای دانشجویان

ترتیب مقادیر در Border Shorthand:

1. Width (ضخامت)

2. Style (سبک)

3. Color (رنگ)

مزایای استفاده از Border Shorthand:

- کد کمتر و تمیزتر

- خوانایی بهتر

- صرفه‌جویی در زمان

- کاهش احتمال خطا

Border-radius

تعریف:

Border-radius برای گرد کردن گوشه‌های یک المان استفاده می‌شود.

ساختار:

|  |
| --- |
| border-radius: مقدار; |

روش‌های مختلف استفاده:

|  |
| --- |
| <style>  .box {  background-color: #f8f9fa;  border: 2px solid #007bff;  padding: 20px;  margin: 10px;  width: 200px;  }  .radius-small { border-radius: 10px; }  .radius-medium { border-radius: 20px; }  .radius-large { border-radius: 30px; }  .radius-circle { border-radius: 50%; }  .radius-elliptical { border-radius: 20px 40px; }  .radius-different { border-radius: 10px 20px 30px 40px; }  </style>  <div class="box radius-small">border-radius: 10px</div>  <div class="box radius-medium">border-radius: 20px</div>  <div class="box radius-large">border-radius: 30px</div>  <div class="box radius-circle">border-radius: 50% (دایره)</div>  <div class="box radius-elliptical">border-radius: 20px 40px (بیضی)</div>  <div class="box radius-different">border-radius: 10px 20px 30px 40px</div> |

[Border-radius](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\Border-radius.html)

border-collapse - border-spacing

معرفی Border-collapse و Border-spacing

تعریف:

این دو ویژگی مخصوص جدول‌ها (table) هستند:

- border-collapse: تعیین می‌کند حاشیه‌های سلول‌های جدول از هم جدا باشند یا به هم بچسبند

- border-spacing: فاصله بین سلول‌های جدول را مشخص می‌کند

مقادیر border-collapse:

- `separate` (پیش‌فرض): حاشیه‌ها جدا از هم

- `collapse`: حاشیه‌ها به هم می‌چسبند

تمرین عملی جدول:

|  |
| --- |
| <style>  .table-container {  margin: 20px 0;  }  table {  width: 100%;  border: 2px solid #333;  }  th, td {  border: 1px solid #666;  padding: 10px;  text-align: center;  }  /\* جدا بودن حاشیه‌ها با فاصله \*/  .separate-table {  border-collapse: separate;  border-spacing: 10px;  }  /\* چسبیدن حاشیه‌ها \*/  .collapse-table {  border-collapse: collapse;  }  /\* فاصله‌های مختلف \*/  .spacing-table {  border-collapse: separate;  border-spacing: 5px 15px;  }  </style>  <div class="table-container">  <h3>border-collapse: separate - border-spacing: 10px</h3>  <table class="separate-table">  <tr>  <th>ستون ۱</th>  <th>ستون ۲</th>  <th>ستون ۳</th>  </tr>  <tr>  <td>داده ۱</td>  <td>داده ۲</td>  <td>داده ۳</td>  </tr>  </table>  </div>  <div class="table-container">  <h3>border-collapse: collapse</h3>  <table class="collapse-table">  <tr>  <th>ستون ۱</th>  <th>ستون ۲</th>  <th>ستون ۳</th>  </tr>  <tr>  <td>داده ۱</td>  <td>داده ۲</td>  <td>داده ۳</td>  </tr>  </table>  </div>  <div class="table-container">  <h3>border-spacing: 5px 15px (افقی عمودی)</h3>  <table class="spacing-table">  <tr>  <th>ستون ۱</th>  <th>ستون ۲</th>  <th>ستون ۳</th>  </tr>  <tr>  <td>داده ۱</td>  <td>داده ۲</td>  <td>داده ۳</td>  </tr>  </table>  </div> |

[table](file:///E:\personal%20job\sam\projects\classes\webdesign\css\border\table.html)

بررسی سایت border radius generator

بررسی چند ابزار آنلاین محبوب و مفید برای تولید کد `border-radius` را در ادامه می‌بینید. این ابزارها به شما کمک میکنند تا بدون نیاز به نوشتن و تست دستی کد، مقادیر مختلف را بهصورت بصری تنظیم و نتیجه را فورا مشاهده کنید.

در جدول زیر برخی از تولیدکنندههای `border-radius` محبوب را مشاهده میکنید:

|  |  |  |
| --- | --- | --- |
| ابزار | ویژگی‌های کلیدی | بهترین برای |
| <https://border-radius.com/> | رابط کاربری بسیار ساده و مینیمال | شروع سریع و تنظیمات ساده |
| [CSS Generator از MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_backgrounds_and_borders/Border-radius_generator) | ابزار رسمی جامعه MDN، پشتیبانی از شعاع بیضوی | توسعه‌دهندگانی که به منابع معتبر اعتماد دارند |
| [CSS Generator از 10015.io](https://10015.io/tools/css-border-radius-generator) | کنترل پیشرفته (8 نقطه)، پشتیبانی از تصویر و گرادیان | طراحی‌های پیچیده و هنری |
| [Elementor Border Radius Generator](https://elementor.com/tools/border-radius-generator/) | پیش‌نمایش زنده، توضیحات آموزشی کامل | کاربران المنتور و تازه‌کارها |
| [Corner Inverter](https://corner-inverter.douiri.org/) | ایجاد گوشه‌های معکوس (Inverted Corners) | طرح‌های مدرن و انحصاری |

راهنمای استفاده از ابزارها

اکثر این تولیدکننده‌ها به یک شکل کار میکنند:

1. مقادیر را تنظیم کنید: معمولاً با حرکت دادن نوارهای لغزنده (Sliders) یا نقطهها روی یک مستطیل پیش‌نمایش.

2. کد را کپی کنید: کد CSS تولید شده در جعبه‌ای نمایش داده می‌شود که می‌توانید آن را مستقیماً در استایل‌شیت خود کپی کنید.

3. پیش‌نمایش ببینید: تغییرات شما بلافاصله روی یک شکل نمونه اعمال می‌شود.

ابزار پیشرفته10015.io حتی به شما امکان می‌دهد تا شعاع هر محور از هر گوشه را به طور جداگانه کنترل کنید تا اشکال ارگانیک و پیچیده‌تری ایجاد کنید. از سوی دیگر، ابزار Corner Inverter برای ایجاد "شعاع border معکوس" طراحی شده است که یک اثر طراحی بسیار مدرن است و پیاده‌سازی دستی آن در CSS بسیار پیچیده است.

انتخاب ابزار مناسب

اگر به یک ابزار سریع و ساده برای گرد کردن گوشه‌های دکمه‌ها یا تصاویر نیاز دارید، Border-Radius.com یا ابزار MDN انتخاب‌های بسیار خوبی هستند.

اگر هدف شما خلق طرح‌های پیچیده و غیراستاندارد است، 10015.io یا Corner Inverter قدرت و انعطاف بیشتری در اختیار شما می‌گذارند.

## outline

Outline-style

معرفی Outline-style

تعریف:

Outline-style برای تعیین سبک outline استفاده می‌شود.

مقادیر ممکن:

|  |
| --- |
| outline-style: solid; /\* خط پیوسته \*/  outline-style: dashed; /\* خط چین \*/  outline-style: dotted; /\* نقطه چین \*/  outline-style: double; /\* خط دوتایی \*/  outline-style: groove; /\* فرورفته سه بعدی \*/  outline-style: ridge; /\* برجسته سه بعدی \*/  outline-style: inset; /\* المان فرورفته \*/  outline-style: outset; /\* المان برجسته \*/  outline-style: none; /\* بدون outline \*/  outline-style: hidden; /\* outline مخفی \*/ |

تمرین عملی Outline-style:

|  |
| --- |
| <style>  .container {  display: flex;  flex-direction: column;  gap: 15px;  padding: 20px;  }  .example {  background-color: #f8f9fa;  padding: 20px;  border: 2px solid #dee2e6;  }  .solid { outline-style: solid; }  .dashed { outline-style: dashed; }  .dotted { outline-style: dotted; }  .double { outline-style: double; }  .groove { outline-style: groove; }  .ridge { outline-style: ridge; }  .inset { outline-style: inset; }  .outset { outline-style: outset; }  .none { outline-style: none; }  </style>  <div class="container">  <div class="example solid">outline-style: solid</div>  <div class="example dashed">outline-style: dashed</div>  <div class="example dotted">outline-style: dotted</div>  <div class="example double">outline-style: double</div>  <div class="example groove">outline-style: groove</div>  <div class="example ridge">outline-style: ridge</div>  <div class="example inset">outline-style: inset</div>  <div class="example outset">outline-style: outset</div>  <div class="example none">outline-style: none</div>  </div> |

[Outline-style](file:///E:\personal%20job\sam\projects\classes\webdesign\css\outline\Outline-style.html)

Outline-width

تعریف:

Outline-width برای تعیین ضخامت outline استفاده می‌شود.

مقادیر ممکن:

|  |
| --- |
| outline-width: thin; /\* نازک \*/  outline-width: medium; /\* متوسط \*/  outline-width: thick; /\* ضخیم \*/  outline-width: 2px; /\* مقدار عددی \*/  outline-width: 0.5em; /\* واحدهای نسبی \*/ |

تمرین عملی Outline-width:

|  |
| --- |
| <style>  .width-example {  background-color: #e9ecef;  padding: 15px;  margin-bottom: 10px;  outline-style: solid;  outline-color: #007bff;  }  .thin { outline-width: thin; }  .medium { outline-width: medium; }  .thick { outline-width: thick; }  .pixel2 { outline-width: 2px; }  .pixel5 { outline-width: 5px; }  .pixel10 { outline-width: 10px; }  </style>  <div class="width-example thin">outline-width: thin</div>  <div class="width-example medium">outline-width: medium</div>  <div class="width-example thick">outline-width: thick</div>  <div class="width-example pixel2">outline-width: 2px</div>  <div class="width-example pixel5">outline-width: 5px</div>  <div class="width-example pixel10">outline-width: 10px</div> |

Outline-color

تعریف:

Outline-color برای تعیین رنگ outline استفاده می‌شود.

روش‌های تعریف رنگ:

|  |
| --- |
| outline-color: red; /\* نام رنگ \*/  outline-color: #ff0000; /\* کد HEX \*/  outline-color: rgb(255, 0, 0); /\* RGB \*/  outline-color: hsl(0, 100%, 50%); /\* HSL \*/  outline-color: transparent; /\* شفاف \*/ |

تمرین عملی Outline-color:

|  |
| --- |
| <style>  .color-example {  background-color: #f8f9fa;  padding: 15px;  margin-bottom: 10px;  outline-style: solid;  outline-width: 3px;  }  .red { outline-color: red; }  .blue { outline-color: blue; }  .green { outline-color: green; }  .hex { outline-color: #ff6b6b; }  .rgb { outline-color: rgb(106, 90, 205); }  .transparent { outline-color: transparent; }  </style>  <div class="color-example red">outline-color: red</div>  <div class="color-example blue">outline-color: blue</div>  <div class="color-example green">outline-color: green</div>  <div class="color-example hex">outline-color: #ff6b6b</div>  <div class="color-example rgb">outline-color: rgb(106, 90, 205)</div>  <div class="color-example transparent">outline-color: transparent</div> |

[Outline-color](file:///E:\personal%20job\sam\projects\classes\webdesign\css\outline\Outline-color.html)

تمرین ترکیبی:

|  |
| --- |
| <style>  .combined-example {  background-color: #ffffff;  padding: 20px;  margin-bottom: 15px;  border: 1px solid #dee2e6;  }  .focus-example {  background-color: #f8f9fa;  padding: 15px;  margin-bottom: 10px;  }  .focus-example:focus {  outline: 3px solid #28a745;  }  .button-example {  padding: 10px 20px;  background-color: #007bff;  color: white;  border: none;  cursor: pointer;  display: block;  margin-bottom: 10px;  }  .button-example:focus {  outline: 2px dashed #ffc107;  }  .custom-outline {  outline: 4px dotted #6f42c1;  background-color: #f8f9fa;  padding: 15px;  margin-bottom: 10px;  }  </style>  <div class="focus-example" tabindex="0">  روی این المان کلیک کنید یا با دکمه Tab آن را انتخاب کنید  </div>  <button class="button-example" tabindex="0">  دکمه با outline سفارشی  </button>  <div class="custom-outline">  outline سفارشی: 4px dotted #6f42c1  </div> |

[combine](file:///E:\personal%20job\sam\projects\classes\webdesign\css\outline\combine.html)

Outline-offset

معرفی Outline-offset

تعریف:

Outline-offset فاصله بین outline و border المان را مشخص می‌کند.

ویژگی‌ها:

- می‌تواند مقادیر مثبت و منفی بگیرد

- فقط روی outline اثر می‌گذارد

- واحدهای مختلفی می‌پذیرد (px, em, rem, ...)

مقادیر ممکن:

|  |
| --- |
| outline-offset: 5px; /\* فاصله مثبت \*/  outline-offset: -3px; /\* فاصله منفی \*/  outline-offset: 1em; /\* واحد نسبی \*/  outline-offset: 0; /\* بدون فاصله \*/ |

تمرین عملی Outline-offset:

|  |
| --- |
| <style>  .offset-example {  background-color: #f8f9fa;  padding: 20px;  margin-bottom: 20px;  border: 2px solid #dee2e6;  outline: 3px solid #007bff;  }  .positive { outline-offset: 10px; }  .negative { outline-offset: -5px; }  .mixed { outline-offset: 15px; }  .zero { outline-offset: 0; }  .em-unit { outline-offset: 1em; }  </style>  <div class="offset-example positive">outline-offset: 10px (مثبت)</div>  <div class="offset-example negative">outline-offset: -5px (منفی)</div>  <div class="offset-example mixed">outline-offset: 15px (فاصله زیاد)</div>  <div class="offset-example zero">outline-offset: 0 (بدون فاصله)</div>  <div class="offset-example em-unit">outline-offset: 1em (واحد نسبی)</div> |

[Outline-offset](file:///E:\personal%20job\sam\projects\classes\webdesign\css\outline\Outline-offset.html)

Outline

معرفی Outline Shorthand:

Outline Shorthand یک روش کوتاه برای تعریف outline در یک خط است.

ساختار:

|  |
| --- |
| outline: width style color; |

ترتیب مقادیر:

1. outline-width (ضخامت)

2. outline-style (سبک)

3. outline-color (رنگ)

تمرین عملی Outline Shorthand:

|  |
| --- |
| <style>  .shorthand-example {  background-color: #e9ecef;  padding: 20px;  margin-bottom: 15px;  border: 1px solid #ced4da;  }  .basic { outline: 2px solid red; }  .dashed { outline: 4px dashed blue; }  .dotted { outline: 3px dotted green; }  .thick { outline: thick double purple; }  .thin { outline: thin groove orange; }  .combined {  outline: 5px ridge #ff6b6b;  outline-offset: 8px;  }  </style>  <div class="shorthand-example basic">outline: 2px solid red</div>  <div class="shorthand-example dashed">outline: 4px dashed blue</div>  <div class="shorthand-example dotted">outline: 3px dotted green</div>  <div class="shorthand-example thick">outline: thick double purple</div>  <div class="shorthand-example thin">outline: thin groove orange</div>  <div class="shorthand-example combined">outline: 5px ridge #ff6b6b + offset</div> |

[outline](file:///E:\personal%20job\sam\projects\classes\webdesign\css\outline\Outline.html)

## [Margin](file:///E:\personal%20job\sam\projects\classes\webdesign\css\Margin)

تعریف Margin

Margin فضای خارج از border یک المان است. این ویژگی فاصله بین المان‌ها را کنترل می‌کند.

روش‌های تعریف Margin

1. روش تک مقداری

|  |
| --- |
| .element {  margin: 20px;  }  /\* همه جهات: 20px \*/ |

1. روش دو مقداری

|  |
| --- |
| .element {  margin: 15px 25px;  }  /\* عمودی: 15px, افقی: 25px \*/ |

1. روش سه مقداری

|  |
| --- |
| .element {  margin: 10px 20px 15px;  }  /\* بالا: 10 px, چپ/راست: 20px, پایین: 15px \*/ |

1. روش چهار مقداری

|  |
| --- |
| .element {  margin: 10px 15px 20px 25px;  }  /\* بالا: 10px, راست: 15px, پایین: 20px, چپ: 25px \*/ |

مثال‌های عملی Margin

|  |
| --- |
| <style>  .box {  background-color: #f0f0f0;  border: 2px solid #333;  padding: 20px;  }  .margin-single { margin: 30px; }  .margin-double { margin: 20px 40px; }  .margin-triple { margin: 10px 30px 50px; }  .margin-quad { margin: 15px 25px 35px 45px; }  .margin-top { margin-top: 40px; }  .margin-right { margin-right: 30px; }  .margin-bottom { margin-bottom: 20px; }  .margin-left { margin-left: 50px; }  </style>  <div class="box margin-single">margin: 30px (همه جهات)</div>  <div class="box margin-double">margin: 20px 40px (عمودی/افقی)</div>  <div class="box margin-triple">margin: 10px 30px 50px (بالا/چپ-راست/پایین)</div>  <div class="box margin-quad">margin: 15px 25px 35px 45px (همه جهات جداگانه)</div>  <div class="box margin-top">margin-top: 40px</div>  <div class="box margin-right">margin-right: 30px</div>  <div class="box margin-bottom">margin-bottom: 20px</div>  <div class="box margin-left">margin-left: 50px</div> |

[sample](file:///E:\personal%20job\sam\projects\classes\webdesign\css\Margin\sample.html)

بررسی مفهوم collapsing margin

Collapsing Margin زمانی اتفاق می‌افتد که marginهای عمودی دو المان مجاور با هم ادغام می‌شوند و فقط بزرگترین margin نمایش داده می‌شود.

شرایط ایجاد Collapsing Margin:

- بین المان‌های block-level مجاور

- بین parent و first/last child

- در المان‌های خالی

مثال Collapsing Margin:

|  |
| --- |
| <style>  .parent {  background-color: #e0e0e0;  padding: 10px;  }  .child1 {  background-color: #ffcccc;  margin-bottom: 40px;  padding: 15px;  }  .child2 {  background-color: #ccffcc;  margin-top: 30px;  padding: 15px;  }  .no-collapse {  background-color: #ccccff;  margin-top: 20px;  margin-bottom: 20px;  padding: 1px; /\* جلوگیری از collapsing \*/  }  </style>  <div class="parent">  <div class="child1">Child 1 - margin-bottom: 40px</div>  <div class="child2">Child 2 - margin-top: 30px</div>  <p>فاصله عمودی بین دو المان: 40px (نه 70px)</p>  </div>  <div class="no-collapse">  این المان با padding از collapsing جلوگیری می‌کند  </div> |

[Collapsing](file:///E:\personal%20job\sam\projects\classes\webdesign\css\Margin\Collapsing.html)

بررسی margin در تگ های inline

ویژگی‌های Margin در المان‌های Inline:

- فقط marginهای چپ و راست (افقی) کار می‌کنند

- marginهای بالا و پایین (عمودی) نادیده گرفته می‌شوند

- روی المان‌های inline-block همه marginها کار می‌کنند

مثال Margin در المان‌های Inline:

|  |
| --- |
| <style>  .inline-example {  background-color: #f8f9fa;  padding: 20px;  }  .span-margin {  background-color: #ffeb3b;  margin: 20px; /\* فقط margin افقی اعمال می‌شود \*/  }  .span-horizontal {  background-color: #e91e63;  margin-left: 30px;  margin-right: 30px;  color: white;  }  .span-vertical {  background-color: #4caf50;  margin-top: 50px; /\* اعمال نمی‌شود \*/  margin-bottom: 50px; /\* اعمال نمی‌شود \*/  color: white;  }  .inline-block {  display: inline-block;  background-color: #2196f3;  margin: 20px; /\* همه marginها کار می‌کنند \*/  color: white;  padding: 10px;  }  </style>  <div class="inline-example">  متن معمولی  <span class="span-margin">این span دارای margin: 20px است (فقط افقی)</span>  متن ادامه دارد  </div>  <div class="inline-example">  متن معمولی  <span class="span-horizontal">margin افقی: 30px</span>  متن ادامه دارد  </div>  <div class="inline-example">  متن معمولی  <span class="span-vertical">margin عمودی اعمال نمی‌شود</span>  متن ادامه دارد  </div>  <div class="inline-example">  متن معمولی  <span class="inline-block">inline-block - همه marginها کار می‌کنند</span>  متن ادامه دارد  </div> |

[Inline](file:///E:\personal%20job\sam\projects\classes\webdesign\css\Margin\Inline.html)

بررسی negative margins

Negative Margins امکان استفاده از مقادیر منفی برای margin را فراهم می‌کند. این ویژگی برای ایجاد افکت‌های خاص و کنترل layout استفاده می‌شود.

کاربردهای Negative Margins:

- همپوشانی المان‌ها

- تنظیم موقعیت المان‌ها

- ایجاد افکت‌های بصری

مثال Negative Margins:

|  |
| --- |
| <style>  .container {  background-color: #f0f0f0;  padding: 20px;  }  .normal-box {  background-color: #ffcccc;  padding: 20px;  margin-bottom: 10px;  }  .negative-top {  background-color: #ccffcc;  padding: 20px;  margin-top: -30px;  }  .negative-left {  background-color: #ccccff;  padding: 20px;  margin-left: -50px;  }  .overlap-container {  position: relative;  height: 120px;  background-color: #e0e0e0;  padding: 10px;  }  .box1 {  background-color: rgba(255, 0, 0, 0.7);  padding: 20px;  position: absolute;  top: 20px;  left: 20px;  z-index: 1;  }  .box2 {  background-color: rgba(0, 255, 0, 0.7);  padding: 20px;  position: absolute;  top: 40px;  left: 40px;  margin-left: -30px;  z-index: 2;  }  </style>  <div class="container">  <div class="normal-box">متن معمولی با margin مثبت</div>  <div class="negative-top">این المان با margin-top منفی به بالا کشیده شده است</div>  </div>  <div class="container">  <div class="negative-left">این المان با margin-left منفی به چپ کشیده شده است</div>  </div>  <div class="overlap-container">  <div class="box1">المان اول</div>  <div class="box2">المان دوم با margin منفی برای همپوشانی</div>  </div> |

[Negative](E:\\personal job\\sam\\projects\\classes\\webdesign\\css\\Margin\\Negative.html)

## [بررسی مفهوم box model](webdesign/css/BoxModel)

تعریف Box Model

هر element در HTML یک جعبه (box) است که از چهار بخش اصلی تشکیل شده است:

اجزای Box Model:

1. Content: محتوای اصلی المنت

2. Padding: فضای بین محتوا و border

3. Border: حاشیه دور padding

4. Margin: فضای خارج از border

نمایش گرافیکی:

هر element در HTML یک جعبه (box) است که از چهار بخش اصلی تشکیل شده است:

اجزای Box Model:

1. Content: محتوای اصلی المنت

2. Padding فضای بین محتوا و border

3. Border: حاشیه دور padding

4. Margin: فضای خارج از border

نمایش گرافیکی:

|  |
| --- |
| ┌─────────────────────────┐ ↑  │ Margin │ |  │ ┌───────────────────┐ │ |  │ │ Border │ │ |  │ │ ┌─────────────┐ │ │ |  │ │ │ Padding │ │ │ |  │ │ │ ┌───────┐ │ │ | ارتفاع کل  │ │ │ │Content│ │ │ │ |  │ │ │ └───────┘ │ │ │ |  │ │ └─────────────┘ │ │ |  │ └───────────────────┘ │ |  └─────────────────────────┘ ↓  ← عرض کل → |

محاسبه اندازه کل element:

عرض کل = width + padding-left + padding-right + border-left + border-right + margin-left + margin-right

ارتفاع کل = height + padding-top + padding-bottom + border-top + border-bottom + margin-top + margin-bottom

مثال‌های عملی:

|  |
| --- |
| <style>  .box-model-demo {  width: 200px;  height: 100px;  padding: 20px;  border: 5px solid #333;  margin: 10px;  background-color: #f0f0f0;  }  /\* محاسبه اندازه کل:  عرض: 200 + 20+20 + 5+5 + 10+10 = 270px  ارتفاع: 100 + 20+20 + 5+5 + 10+10 = 170px  \*/  </style>  <div class="box-model-demo">محتوای داخل جعبه</div> |

ویژگیهای box model

|  |
| --- |
| /\* حالت پیش فرض \*/  .default-box {  box-sizing: content-box; /\* فقط content در width/height محاسبه می‌شود \*/  width: 200px;  padding: 20px; /\* عرض کل: 200 + 40 = 240px \*/  }  .border-box {  box-sizing: border-box; /\* padding و border در width/height محاسبه می‌شوند \*/  width: 200px;  padding: 20px; /\* عرض کل: 200px \*/  } |

[box-sizing property](webdesign/css/BoxModel/box-sizing%20property.html)

## Overflow

تعریف Overflow

وقتی محتوای یک element از اندازه تعریف شده آن بزرگ‌تر باشد، overflow رخ می‌دهد.

مقادیر مختلف overflow:

1. overflow: visible (پیش‌فرض)

|  |
| --- |
| .visible-overflow {  width: 200px;  height: 100px;  overflow: visible;  border: 2px solid red;  } |

[visible](webdesign/css/Overflow/visible.html)

محتوای اضافی نمایش داده می‌شود و از element خارج می‌شود.

1. overflow: hidden

|  |
| --- |
| .hidden-overflow {  width: 200px;  height: 100px;  overflow: hidden;  border: 2px solid blue;  } |

[hidden](webdesign/css/Overflow/hidden.html)

محتوای اضافی مخفی می‌شود.

1. overflow: scroll

|  |
| --- |
| .scroll-overflow {  width: 200px;  height: 100px;  overflow: scroll;  border: 2px solid green;  } |

[scroll](webdesign/css/Overflow/scroll.html)

همیشه scrollbar نمایش داده می‌شود.

۴. overflow: auto

|  |
| --- |
| .auto-overflow {  width: 200px;  height: 100px;  overflow: auto;  border: 2px solid orange;  } |

[auto](webdesign/css/Overflow/auto.html)

فقط در صورت نیاز scrollbar نمایش داده می‌شود.

overflow برای جهت‌های خاص:

|  |
| --- |
| .directional-overflow {  width: 200px;  height: 100px;  overflow-x: hidden; /\* مخفی کردن افقی \*/  overflow-y: auto; /\* اسکرول عمودی در صورت نیاز \*/  border: 2px solid purple;  } |

[x-y](webdesign/css/Overflow/x-y.html)

مثال کاربردی:

|  |
| --- |
| <style>  .text-container {  width: 300px;  height: 150px;  border: 2px solid #333;  padding: 10px;  margin: 10px;  }  .scrollable {  overflow: auto;  }  .hidden-content {  overflow: hidden;  }  .long-text {  width: 400px; /\* عمدا بزرگتر از container \*/  }  </style>  <div class="text-container scrollable">  <div class="long-text">  این یک متن بسیار طولانی است که از container خارج می‌شود...  </div>  </div>  <div class="text-container hidden-content">  <div class="long-text">  این متن به دلیل hidden بودن، قسمت اضافی آن دیده نمی‌شود.  </div>  </div> |

[inaction](webdesign/css/Overflow/inaction.html)

## [Box-sizing](#_بررسی_مفهوم_box)

## Root

معرفی css variables با استفاده از :root و var()

`:root` یک pseudo-class است که به المان ریشه سند (معمولاً `<html>`) اشاره می‌کند. از این ویژگی برای تعریف متغیرهای CSS استفاده می‌شود.

کاربردهای :root:

- تعریف متغیرهای CSS

- ذخیره رنگ‌ها، فونت‌ها، سایزها و سایر مقادیر قابل استفاده مجدد

مثال عملی :root

|  |
| --- |
| <style>  :root {  --primary-color: #007bff;  --secondary-color: #6c757d;  --font-size-large: 24px;  --spacing-unit: 16px;  --border-radius: 8px;  }  .box1 {  background-color: var(--primary-color);  color: white;  padding: var(--spacing-unit);  border-radius: var(--border-radius);  }  .box2 {  background-color: var(--secondary-color);  color: white;  padding: calc(var(--spacing-unit) \* 2);  font-size: var(--font-size-large);  }  .box3 {  border: 2px solid var(--primary-color);  padding: var(--spacing-unit);  border-radius: var(--border-radius);  }  </style>  <div class="box1">این با متغیرهای :root استایل شده</div>  <div class="box2">این هم با متغیرهای :root استایل شده</div>  <div class="box3">این با border و متغیرهای :root استایل شده</div> |

[root](webdesign/css/Root/root.html)

## [calc()](webdesign/css/calc)

معرفی تابع ریاضی calc()

Calc() یک تابع CSS است که امکان انجام محاسبات ریاضی را در مقادیر CSS فراهم می‌کند.

ویژگی‌های calc():

- پشتیبانی از چهار عمل اصلی (+, -, \*, /)

- امکان ترکیب واحدهای مختلف

- استفاده در هر propertyای که مقدار عددی می‌پذیرد

مثال‌های عملی calc():

|  |
| --- |
| <style>  .calc-example {  background-color: #f0f0f0;  margin-bottom: 10px;  padding: 15px;  border: 1px solid #ccc;  }  .width-calc {  width: calc(100% - 100px);  background-color: #ffcccc;  }  .height-calc {  height: calc(50vh - 50px);  background-color: #ccffcc;  }  .font-calc {  font-size: calc(16px + 0.5vw);  background-color: #ccccff;  }  .margin-calc {  margin: calc(10px + 2%);  background-color: #ffffcc;  }  .padding-calc {  padding: calc(20px - 1em) calc(15px + 2%);  background-color: #ffccff;  }  </style>  <div class="calc-example width-calc">width: calc(100% - 100px)</div>  <div class="calc-example height-calc">height: calc(50vh - 50px)</div>  <div class="calc-example font-calc">font-size: calc(16px + 0.5vw)</div>  <div class="calc-example margin-calc">margin: calc(10px + 2%)</div>  <div class="calc-example padding-calc">padding: calc(20px - 1em) calc(15px + 2%)</div> |

[calc](webdesign/css/calc/calc.html)

## Display

Display یکی از مهمترین ویژگی‌های CSS است که نحوه نمایش و layout یک المان را تعیین می‌کند.

مقادیر اصلی display:

۱. display: block

- المان کل عرض parent را می‌گیرد

- در خط جدید شروع می‌شود

- می‌توان width و height تنظیم کرد

۲. display: inline

- فقط به اندازه محتوا فضا می‌گیرد

- در همان خط نمایش داده می‌شود

- نمی‌توان width و height تنظیم کرد

۳. display: inline-block

- مانند inline در خط نمایش داده می‌شود

- مانند block می‌توان width و height تنظیم کرد

۴. display: none

- المان کاملاً مخفی می‌شود

- فضایی اشغال نمی‌کند

مثال‌های عملی display

|  |
| --- |
| <style>  .container {  background-color: #f8f9fa;  padding: 15px;  margin-bottom: 20px;  border: 1px solid #dee2e6;  }  .block-example {  display: block;  background-color: #ff6b6b;  color: white;  padding: 10px;  margin: 5px 0;  width: 200px;  height: 50px;  }  .inline-example {  display: inline;  background-color: #4ecdc4;  color: white;  padding: 10px;  margin: 0 5px;  /\* width و height کار نمی‌کنند \*/  }  .inline-block-example {  display: inline-block;  background-color: #45b7d1;  color: white;  padding: 10px;  margin: 0 5px;  width: 120px;  height: 40px;  }  .hidden-example {  display: none;  background-color: #96ceb4;  padding: 10px;  }  .visible-after-hidden {  background-color: #feca57;  padding: 10px;  }  </style>  <div class="container">  <div class="block-example">display: block - المان اول</div>  <div class="block-example">display: block - المان دوم</div>  </div>  <div class="container">  <span class="inline-example">display: inline - اول</span>  <span class="inline-example">display: inline - دوم</span>  <span class="inline-example">display: inline - سوم</span>  </div>  <div class="container">  <div class="inline-block-example">inline-block اول</div>  <div class="inline-block-example">inline-block دوم</div>  <div class="inline-block-example">inline-block سوم</div>  </div>  <div class="container">  <div class="hidden-example">این المان display: none دارد</div>  <div class="visible-after-hidden">این المان بعد از المان مخفی است</div>  </div> |

[display](webdesign/css/Display/Display.html)

## [Box-shadow](webdesign/css/box-shadow)

Box-shadow برای ایجاد سایه دور المان‌ها استفاده می‌شود. این ویژگی می‌تواند افکت‌های عمق و بعد به طراحی اضافه کند.

ساختار box-shadow:

|  |
| --- |
| box-shadow: offset-x offset-y blur-radius spread-radius color inset; |

پارامترها:

- offset-x: فاصله افقی سایه (مثبت: راست، منفی: چپ)

- offset-y: فاصله عمودی سایه (مثبت: پایین، منفی: بالا)

- blur-radius: میزان محو شدن سایه

- spread-radius: اندازه سایه

- color: رنگ سایه

- inset: سایه داخلی

مثال‌های عملی box-shadow:

|  |
| --- |
| <style>  .shadow-example {  background-color: white;  padding: 30px;  margin: 20px;  border: 1px solid #ddd;  }  .shadow-simple {  box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.3);  }  .shadow-blur {  box-shadow: 0 0 20px rgba(0, 0, 0, 0.2);  }  .shadow-spread {  box-shadow: 0 0 0 10px rgba(255, 0, 0, 0.3);  }  .shadow-inset {  box-shadow: inset 0 0 10px rgba(0, 0, 0, 0.5);  }  .shadow-multiple {  box-shadow:  5px 5px 10px rgba(0, 0, 0, 0.3),  -5px -5px 10px rgba(255, 0, 0, 0.3);  }  .shadow-colored {  box-shadow: 8px 8px 15px rgba(106, 17, 203, 0.4);  }  </style>  <div class="shadow-example shadow-simple">سایه ساده</div>  <div class="shadow-example shadow-blur">سایه با blur زیاد</div>  <div class="shadow-example shadow-spread">سایه با spread</div>  <div class="shadow-example shadow-inset">سایه داخلی (inset)</div>  <div class="shadow-example shadow-multiple">چند سایه</div>  <div class="shadow-example shadow-colored">سایه رنگی</div> |

[box-shadow](webdesign/css/box-shadow/box-shadow.html)

## [لیست ها](webdesign/css/list)

List-style-type

لیست‌ها در CSS

معرفی لیست‌ها در CSS

در CSS می‌توانیم ظاهر و رفتار لیست‌های HTML را کنترل کنیم. لیست‌ها شامل دو نوع اصلی هستند:

ویژگی‌های اصلی استایل‌دهی به لیست‌ها

۱. `list-style-type` - تعیین نوع مارکر

برای لیست‌های غیرترتیبی (`<ul>`):

|  |
| --- |
| ul.disc { list-style-type: disc; } /\* پیش‌فرض - دایره پر \*/  ul.circle { list-style-type: circle; } /\* دایره توخالی \*/  ul.square { list-style-type: square; } /\* مربع \*/  ul.none { list-style-type: none; } /\* بدون مارکر \*/ |

[ul](webdesign/css/list/ul.html)

برای لیست‌های ترتیبی (`<ol>`):

|  |
| --- |
| ol.decimal { list-style-type: decimal; } /\* اعداد لاتین \*/  ol.arabic { list-style-type: arabic-indic; } /\* اعداد عربی \*/  ol.persian { list-style-type: persian; } /\* اعداد فارسی \*/  ol.roman { list-style-type: upper-roman; } /\* اعداد رومی بزرگ \*/  ol.alpha { list-style-type: lower-alpha; } /\* حروف کوچک \*/ |

[ol](webdesign/css/list/ol.html)

۲. `list-style-position` - موقعیت مارکر

|  |
| --- |
| ul.inside { list-style-position: inside; } /\* مارکر داخل جعبه \*/  ul.outside { list-style-position: outside; } /\* مارکر خارج جعبه (پیش‌فرض) \*/ |

[position](webdesign/css/list/position.html)

3. list-style-image - تصویر به جای مارکر

|  |
| --- |
| ul.custom { list-style-image: url('bullet.png'); } |

[image](webdesign/css/list/image.html)

مقادیر نمادین:

|  |
| --- |
| .symbolic { list-style-type: "★ "; } /\* استفاده از کاراکتر خاص \*/ |

نکات فنی مهم

- مقدار پیش‌فرض برای ul: disc

- مقدار پیش‌فرض برای ol: decimal

- با noneمی‌توان مارکرها را حذف کرد

- می‌توان از::marker برای استایل مستقیم مارکرها استفاده کرد

- از direction: rtl برای لیست‌های فارسی استفاده کنید

این ویژگی‌ها به شما کنترل کامل روی ظاهر و چیدمان لیست‌ها در صفحات وب می‌دهد.

## [فونت آیکون ها:](webdesign/css/webIcons)

فونت‌های آیکون مجموعهای از نمادها و شکلها هستند که به‌ جای حروف در یک فونت قرار گرفته‌اند و میتوانید مانند متن معمولی، اندازه و رنگ آنها را با CSS تغییر دهید و چون به صورت برداری هستند، بدون افت کیفیت بزرگنمایی کنید

|  |  |  |
| --- | --- | --- |
| نام کتابخانه | توضیحات | تعداد آیکون (تقریبی) |
| [Font Awesome](https://fontawesome.com/download) | یکی از محبوب‌ترین و کامل‌ترین کتابخانه‌ها | +۱,۶۰۰ (رایگان) |
| [Google Icons (Material Design)](https://fonts.google.com/icons) | مجموعه رسمی و توصیه‌شده توسط گوگل، با طراحی مدرن و یکدست | +۹۰۰ |
| [Bootstrap Icons](https://icons.getbootstrap.com/) | آیکون‌های SVG ساخته شده برای فریمورک بوت‌استرپ، اما قابل استفاده در هر پروژه‌ای |  |
| [Typicons](https://www.s-ings.com/typicons/) | مجموعه‌ای کاملاً رایگان و میزبانی شده در گیت‌هاب | ۳۳۰ |
| [phosphoricons](https://phosphoricons.com/) | طراحی نیمه‌شکسته و زیبا، مجموعه عظیم، مناسب برای وب و موبایل | ۹,۰۰۰+ |
| [Devicons](https://devicon.dev/) | مناسب برای نمایش لوگوها و آرم‌های فناوری‌ها و برندهای مختلف | ۱۲۳ |
| [Tabler Icons](https://tabler.io/icons) | آیکون‌های "pixel-perfect"، کاملاً رایگان و متن‌باز. | ۵,۷۰۰+ |
| [Feather Icons](https://feathericons.dev/) | طراحی ساده، کاربرپسند و بسیار سبک، ایده‌آل برای پروژه‌های مینیمال. | ۲۸۷+ |
| [Heroicons](https://heroicons.com/) | طراحی دست‌ساز و باکیفیت توسط سازندگان Tailwind CSS، در دو استایل Outline و Solid | ۳۱۶+ |

مزایای کلی استفاده از فونت آیکون

فونت‌های آیکون به دلایل زیر به یک انتخاب محبوب در طراحی وب تبدیل شده‌اند:

قابلیت مقیاس‌پذیری: چون بردار هستند، بدون از دست دادن کیفیت، بزرگ یا کوچک می‌شوند.

استایل‌دهی آسان با CSS: به راحتی می‌توان رنگ، سایه، اندازه و سایز را تغییر داد.

سرعت بارگیری بالا: حجم بسیار کمی دارند و عملکرد سایت را بهبود می‌بخشند.

پشتیبانی گسترده از مرورگرها : در تقریباً همه مرورگرها به خوبی نمایش داده می‌شوند.

شفافیت : به دلیل داشتن پس‌زمینه شفاف، روی هر رنگی قابل قرارگیری هستند.

روش استفاده از فونت آیکون ها

به طور کلی دو راه متداول برای استفاده از این کتابخانه‌ها وجود دارد:

1. دانلود و میزبانی داخلی: دانلود فایل‌های فونت و CSS و اضافه کردن آن‌ها به پروژه خود.

2. استفاده از CDN: اتصال مستقیم به فایل‌های کتابخانه از طریق یک لینک (تگ لینک) در بخش `head` سایت که ساده‌تر است.

حتماً! در ادامه نحوه استفاده از هر کدام از کتابخانه‌های معروف فونت آیکون با CDN را با مثال عملی مشاهده می‌کنید:

### Font Awesome

اتصال CDN در `<head>`:

|  |
| --- |
| <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">  <i class="fas fa-home"></i> <!-- آیکون خانه -->  <i class="fab fa-twitter"></i> <!-- آیکون توییتر -->  <i class="fas fa-search"></i> <!-- آیکون جستجو --> |

[Font Awesome](webdesign/css/webIcons/fontawsome.html)

### Google Material Icons

اتصال CDN در `<head>`:

|  |
| --- |
| <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">  <span class="material-icons">home</span> *<!-- آیکون خانه -->*  <span class="material-icons">favorite</span> *<!-- آیکون قلب -->*  <span class="material-icons">search</span> *<!-- آیکون جستجو -->* |

[Google](webdesign/css/webIcons/Google.html)

### Bootstrap Icons

اتصال CDN در `<head>`:

|  |
| --- |
| <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.0/font/bootstrap-icons.css">  <i class="bi bi-house"></i> <!-- آیکون خانه -->  <i class="bi bi-heart"></i> <!-- آیکون قلب -->  <i class="bi bi-search"></i> <!-- آیکون جستجو --> |

[Bootstrap](webdesign/css/webIcons/Bootstrap.html)

### Phosphor Icons

### Tabler Icons

|  |
| --- |
| <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@tabler/icons-webfont@latest/dist/tabler-icons.min.css" />  <i class="ti ti-heart"></i>  <i class="ti ti-star"></i>  <i class="ti ti-home"></i> |

[Tabler](webdesign/css/webIcons/Tabler.html)

### Feather Icons

|  |
| --- |
| <!-- https://feathericons.dev/?search=heart&iconset=feather -->  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" class="main-grid-item-icon" fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2">  <path d="M20.84 4.61a5.5 5.5 0 0 0-7.78 0L12 5.67l-1.06-1.06a5.5 5.5 0 0 0-7.78 7.78l1.06 1.06L12 21.23l7.78-7.78 1.06-1.06a5.5 5.5 0 0 0 0-7.78z" />  </svg>  <!-- https://feathericons.dev/?search=star&iconset=feather -->  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" class="main-grid-item-icon" fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2">  <polygon points="12 2 15.09 8.26 22 9.27 17 14.14 18.18 21.02 12 17.77 5.82 21.02 7 14.14 2 9.27 8.91 8.26 12 2" />  </svg>  <!-- https://feathericons.dev/?search=home&iconset=feather -->  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" width="24" height="24" class="main-grid-item-icon" fill="none" stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2">  <path d="M3 9l9-7 9 7v11a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2z" />  <polyline points="9 22 9 12 15 12 15 22" />  </svg> |

[Feather](webdesign/css/webIcons/Feather.html)

### Heroicons

|  |
| --- |
| <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="red" width="30" height="30">  <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4.318 6.318a4.5 4.5 0 000 6.364L12 20.364l7.682-7.682a4.5 4.5 0 00-6.364-6.364L12 7.636l-1.318-1.318a4.5 4.5 0 00-6.364 0z"/>  </svg>  *<!-- Star Icon -->*  <svg xmlns="http://www.w3.org/2000/svg" fill="gold" viewBox="0 0 24 24" stroke="gold" width="30" height="30">  <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M11.049 2.927c.3-.921 1.603-.921 1.902 0l1.519 4.674a1 1 0 00.95.69h4.915c.969 0 1.371 1.24.588 1.81l-3.976 2.888a1 1 0 00-.363 1.118l1.518 4.674c.3.922-.755 1.688-1.538 1.118l-3.976-2.888a1 1 0 00-1.176 0l-3.976 2.888c-.783.57-1.838-.197-1.538-1.118l1.518-4.674a1 1 0 00-.363-1.118l-3.976-2.888c-.784-.57-.38-1.81.588-1.81h4.914a1 1 0 00.951-.69l1.519-4.674z"/>  </svg>  *<!-- Home Icon -->*  <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke="blue" width="30" height="30">  <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1 1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6"/>  </svg> |

[Heroicons](webdesign/css/webIcons/Heroicons.html)

### Remix Icon

|  |
| --- |
| <link  href="https://cdn.jsdelivr.net/npm/remixicon@4.7.0/fonts/remixicon.css"  rel="stylesheet"  />  <i class="ri-heart-line"></i>  <i class="ri-heart-2-line"></i>  <i class="ri-star-line"></i>  <i class="ri-home-5-line"></i> |

Remix

## [pseudo](webdesign/css/pseudo)

معرفی pseudo class selector های زیر:

Link

`:Link` - لینک مشاهده نشده

توضیح: برای لینک‌هایی که هنوز کلیک نشده‌اند

|  |
| --- |
| <style>  a:link {  color: blue;  text-decoration: none;  }  </style>  <a href="#page1">لینک مشاهده نشده</a> |

[link](webdesign/css/pseudo/Link.html)

Visited

`:Visited` - لینک بازدید شده

توضیح: برای لینک‌هایی که قبلاً بازدید شده‌اند

|  |
| --- |
| <style>  a:visited {  color: purple;  opacity: 0.7;  }  </style>  <a href="#page2">لینک بازدید شده</a> |

Hover

`:Hover` - هاور کردن موس

توضیح: وقتی موس روی عنصر قرار می‌گیرد

|  |
| --- |
| <style>  a:hover {  color: red;  background-color: yellow;  padding: 2px 5px;  }  </style>  <a href="#page3">هاور کنید روی من</a> |

[Hover](webdesign/css/pseudo/Hover.html)

Active

`:Active` - هنگام کلیک

توضیح: در لحظه‌ای که روی لینک کلیک می‌شود

|  |
| --- |
| <style>  a:active {  color: white;  background-color: black;  transform: scale(0.95);  }  </style>  <a href="#page4">کلیک کنید و نگه دارید</a> |

[Active](webdesign/css/pseudo/Active.html)

ترکیب این موارد

|  |
| --- |
| <style>  a {  color: blue;  text-decoration: none;  font-size: 18px;  transition: all 0.3s;  padding: 5px 10px;  border-radius: 4px;  }    a:link {  color: blue;  background-color: #f0f8ff;  }    a:visited {  color: purple;  background-color: #f8f0ff;  }    a:hover {  color: white;  background-color: #0066cc;  transform: translateY(-2px);  }    a:active {  color: red;  background-color: #ffcccc;  transform: translateY(1px);  }  </style>  <a href="#test">این یک لینک نمونه است</a> |

[combine](webdesign/css/pseudo/combine.html)

Focus

`:Focus` - حالت فوکوس

توضیح: وقتی روی یک فیلد input کلیک می‌کنید یا با کلید Tab آن را انتخاب می‌کنید، این حالت فعال می‌شود

|  |
| --- |
| <style>  input:focus {  background-color: #fff8dc;  border: 2px solid #4CAF50;  outline: none;  padding: 8px;  box-shadow: 0 0 5px rgba(76, 175, 80, 0.5);  }  </style>  <input type="text" placeholder="اینجا کلیک کنید یا با Tab انتخابش کنید"> |

[Focus](webdesign/css/pseudo/Focus.html)

## [transition](webdesign/css/transition)

معرفی مفهوم transition

معرفی Transition

Transition به شما امکان می‌دهد تغییرات CSS به صورت نرم و در مدت زمان مشخصی انجام شوند.

Transition-property

`transition-property` - مشخص کردن ویژگی‌های متغیر

توضیح: تعیین می‌کند کدام ویژگی‌های CSS باید transition داشته باشند

|  |
| --- |
| <style>  div {  width: 100px;  height: 100px;  background: blue;  transition-property: width, background;  }  div:hover {  width: 200px;  background: red;  }  </style>  <div>هاور کنید</div> |

[property](webdesign/css/transition/property.html)

Transition-duration

transition-duration- مدت زمان transition

توضیح: تعیین می‌کند transition چقدر طول بکشد (ثانیه یا میلی‌ثانیه)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: green;  transition-property: all;  transition-duration: 2s;  }  .box:hover {  width: 300px;  background: orange;  }  </style>  <div class="box">هاور کنید (۲ ثانیه)</div> |

[duration](webdesign/css/transition/duration.html)

Transition-timing-functioning

transition-timing-function - تابع زمان‌بندی

توضیح: تعیین می‌کند transition با چه سرعتی در طول زمان پیش برود

|  |
| --- |
| <style>  .box {  width: 100px;  height: 50px;  background: blue;  margin: 10px;  transition: all 2s;  }  .ease { transition-timing-function: ease; }  .linear { transition-timing-function: linear; }  .ease-in { transition-timing-function: ease-in; }  .ease-out { transition-timing-function: ease-out; }  .box:hover {  width: 300px;  }  </style>  <div class="box ease">ease</div>  <div class="box linear">linear</div>  <div class="box ease-in">ease-in</div>  <div class="box ease-out">ease-out</div> |

[timing](webdesign/css/transition/timing.html)

Transition-delay

transition-delay - تأخیر در شروع

توضیح: تعیین می‌کند transition چه تأخیری قبل از شروع داشته باشد

|  |
| --- |
| <style>  .delayed {  width: 100px;  height: 50px;  background: green;  transition: all 1s;  transition-delay: 0.5s;  }  .delayed:hover {  width: 250px;  background: red;  }  </style>  <div class="delayed">هاور کنید (۰.۵ ثانیه تأخیر)</div> |

[delay](webdesign/css/transition/delay.html)

Transition

transition - shorthand

توضیح: ترکیب تمام ویژگی‌ها در یک خط

|  |
| --- |
| <style>  button {  padding: 10px 20px;  background: #4CAF50;  color: white;  border: none;  /\* property | duration | timing-function | delay \*/  transition: all 0.5s ease-in-out 0.1s;  }  button:hover {  background: #2196F3;  transform: translateY(-5px);  }  </style>  <button>هاور کنید</button> |

[Shorthand](webdesign/css/transition/shorthand.html)

مثال ترکیبی کامل

|  |
| --- |
| <style>  .card {  width: 200px;  height: 100px;  background: #f0f0f0;  padding: 20px;  /\* property: all, duration: 0.8s, timing: ease, delay: 0s \*/  transition: all 0.8s ease 0s;  }  .card:hover {  background: #e3f2fd;  transform: scale(1.05);  box-shadow: 0 5px 15px rgba(0,0,0,0.2);  }  </style>  <div class="card">کارت نمونه - هاور کنید</div> |

[combine](webdesign/css/transition/combine.html)

بررسی وب سایت cubic-bezier

|  |  |
| --- | --- |
| ویژگی | توضیح |
| آدرس وب‌سایت | [https://cubic-bezier.com](%20https:/cubic-bezier.com%20) |
| کاربرد اصلی | ایجاد و ویرایش دیداری منحنی‌های `cubic-bezier` برای انیمیشن‌های CSS |
| توضیح ساده | این ابزار به شما کمک می‌کند بدون نیاز به حدس زدن اعداد، با حرکت دادن نقاط کنترل، منحنی سرعت انیمیشن خود را به راحتی طراحی و نتیجه آن را بلافاصله مشاهده کنید. |

این تابع یک Timing Function است که نحوه تغییر سرعت یک انیمیشن در طول زمان را کنترل می‌کند و به جای حرکت یکنواخت، به انیمیشن شما حس طبیعی‌تر یا دراماتیک‌تری می‌دهد. ابزار [Cubic-Bezier.com](https://cubic-bezier.com/) یک رابط گرافیکی برای کار با این تابع ارائه می‌دهد.

مثال ساده:

کد زیر یک انیمیشن برای یک جعبه ایجاد می‌کند که در هنگام هاور، با یک منحنی شتاب‌دار (`ease-in`) به سمت راست حرکت می‌کند.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <style>  .box {  width: 100px;  height: 100px;  background-color: #4a69bd;  transition: transform 0.5s cubic-bezier(0.42, 0, 1, 1); /\* ease-in \*/  }  .box:hover {  transform: translateX(100px);  }  </style>  </head>  <body>  <div class="box"></div>  </body>  </html> |

[cubic-bezier](webdesign/css/transition/cubic-bezier.html)

بررسی سایت hover.css

|  |  |
| --- | --- |
| ویژگی | توضیح |
| آدرس پروژه | https://ianlunn.github.io/Hover/ |
| کاربرد اصلی | یک کتابخانه آماده از افکت‌های هاور برای لینک‌ها، دکمه‌ها، آیکون‌ها و ... |
| توضیح ساده | این مجموعه شامل ده‌ها افکت از پیش طراحی شده (مانند جلوه‌های سایه، پس‌زمینه، و گفتگو) است که می‌توانید به سادگی با اضافه کردن یک کلاس به المان خود از آن‌ها استفاده کنید. |

<https://ianlunn.github.io/Hover/>

این کتابخانه حجم وسیعی از افکت‌های هاور دو بعدی، سه بعدی، تغییر پس‌زمینه و حاشیه را در خود جای داده است. استفاده از آن بسیار ساده است و نیاز به نوشتن انیمیشن از صفر را تا حد زیادی کاهش می‌دهد.

مثال ساده:

برای استفاده، ابتدا فایل CSS کتابخانه را در صفحه خود قرار دهید، سپس کلاس‌های مورد نظر را به المان اضافه کنید. مثال زیر افکت "جلوهدادن" (`float`) را روی یک دکمه پیاده می‌کند.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <!-- لینک به فایل Hover.css (از CDN) -->  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/hover.css/2.3.1/css/hover-min.css">  <style>  /\* استایل پایه دکمه \*/  .btn {  padding: 15px 30px;  background-color: #2c3e50;  color: white;  text-decoration: none;  border-radius: 5px;  display: inline-block;  }  </style>  </head>  <body>  <!-- اعمال افکت هاور از کتابخانه -->  <a href="#" class="btn hvr-float">دکمه من</a>  </body>  </html> |

[Hover](webdesign/css/transition/Hover.html)

[معرفی سایر pseudo class selector ها](webdesign/css/pseudo)

[Root](#_Root)

Empty

توضیح:

این سلکتور برای انتخاب المان‌هایی استفاده می‌شود که هیچ محتوایی ندارند (حتی فاصله یا متن خالی)

|  |
| --- |
| <div class="box"></div>  <div class="box">محتوا دارم</div>  <div class="box"> </div>  <div class="box"><!-- کامنت --></div>  <style>  .box {  width: 100px;  height: 50px;  margin: 10px;  border: 1px solid black;  }  .box:empty {  background-color: red;  }  </style> |

[Empty](webdesign/css/pseudo/Empty.html)

نتیجه واقعی:

✅ اولین div - قرمز می‌شود (کاملاً خالی)

❌ دومی - سفید می‌ماند (محتوا دارد)

❌ سومی - سفید می‌ماند (فاصله دارد)

✅ چهارمی - قرمز می‌شود (کاملاً خالی - کامنت تاثیری ندارد)

First-child

توضیح: این سلکتور اولین المان فرزند را از بین همه فرزندان والدش انتخاب می‌کند.

|  |
| --- |
| <ul>  <li>آیتم اول</li>  <li>آیتم دوم</li>  <li>آیتم سوم</li>  </ul>  <div>  <p>پاراگراف اول</p>  <p>پاراگراف دوم</p>  <span>این اسپن است</span>  </div>  <style>  li:first-child {  color: blue;  font-weight: bold;  }  p:first-child {  background-color: yellow;  }  </style> |

[First-child](webdesign/css/pseudo/First-child.html)

نتیجه:

- اولین `li` آبی و پررنگ می‌شود

- اولین `p` زرد می‌شود

Last-child

توضیح:

این سلکتور آخرین المان فرزند را از بین همه فرزندان والدش انتخاب می‌کند.

|  |
| --- |
| <ul>  <li>آیتم اول</li>  <li>آیتم دوم</li>  <li>آیتم آخر</li>  </ul>  <div class="container">  <button>دکمه 1</button>  <button>دکمه 2</button>  <button>دکمه آخر</button>  </div>  <style>  li:last-child {  color: green;  font-style: italic;  }  button:last-child {  background-color: purple;  color: white;  }  </style> |

[Last-child](webdesign/css/pseudo/Last-child.html)

نتیجه:

- آخرین `li` سبز و ایتالیک می‌شود

- آخرین `button` بنفش با متن سفید می‌شود

Only-child

توضیح:

این سلکتور المان‌هایی را انتخاب می‌کند که تنها فرزند والد خود باشند.

|  |
| --- |
| <div class="parent">  <p>من تنها فرزند هستم</p>  </div>  <div class="parent">  <p>فرزند اول</p>  <p>فرزند دوم</p>  </div>  <div class="parent">  <span>من اسپن تنها هستم</span>  </div>  <style>  .parent {  border: 2px solid #ccc;  margin: 10px;  padding: 10px;  }  p:only-child {  background-color: gold;  padding: 10px;  }  span:only-child {  background-color: lightblue;  padding: 10px;  }  </style> |

[Only-child](webdesign/css/pseudo/Only-child.html)

First-of-type

توضیح:

این سلکتور اولین المان از نوع خود را در بین فرزندان والدش انتخاب می‌کند.

|  |
| --- |
| <div class="container">  <span>اسپن اول</span>  <p>پاراگراف اول</p>  <span>اسپن دوم</span>  <p>پاراگراف دوم</p>  <div>دیو اول</div>  </div>  <style>  .container {  border: 2px solid #333;  padding: 15px;  }  p:first-of-type {  background-color: lightgreen;  padding: 8px;  }  span:first-of-type {  background-color: lightcoral;  padding: 8px;  }  div:first-of-type {  background-color: lightyellow;  padding: 8px;  }  </style> |

[First-of-type](webdesign/css/pseudo/First-of-type.html)

نتیجه:

- اولین `p` سبز روشن می‌شود

- اولین `span` قرمز روشن می‌شود

- `div` زرد روشن می‌شود

Last-of-type

توضیح:

این سلکتور آخرین المان از نوع خود را در بین فرزندان والدش انتخاب می‌کند.

|  |
| --- |
| <div class="content">  <p>پاراگراف اول</p>  <span>اسپن اول</span>  <p>پاراگراف آخر</p>  <span>اسپن آخر</span>  <div>دیو آخر</div>  </div>  <style>  .content {  border: 2px solid #666;  padding: 15px;  margin-top: 20px;  }  p:last-of-type {  background-color: lightseagreen;  padding: 8px;  }  span:last-of-type {  background-color: plum;  padding: 8px;  }  div:last-of-type {  background-color: orange;  padding: 8px;  }  </style> |

[Last-of-type](webdesign/css/pseudo/Last-of-type.html)

نتیجه:

- آخرین `p` سبز آبی می‌شود

- آخرین `span` بنفش کمرنگ می‌شود

- `div` نارنجی می‌شود

Only-of-type

توضیح:

این سلکتور المان‌هایی را انتخاب می‌کند که تنها المان از نوع خود در بین فرزندان والدشان باشند

|  |
| --- |
| <div class="container">  <p>من تنها پاراگراف هستم</p>  <span>اسپن اول</span>  <span>اسپن دوم</span>  <div>من تنها دیو از این نوع هستم</div>  </div>  <style>  .container {  border: 2px solid #333;  padding: 15px;  margin: 10px;  }  p:only-of-type {  background-color: lightgreen;  padding: 10px;  }  div:only-of-type {  background-color: lightblue;  padding: 10px;  }  span:only-of-type {  background-color: gold;  padding: 10px;  }  </style> |

[Only-of-type](webdesign/css/pseudo/Only-of-type.html)

نتیجه:

- `p` سبز می‌شود (تنها پاراگراف است)

- `div` آبی می‌شود (تنها دیو از این نوع است)

- `span`ها تغییری نمی‌کنند (چون دو اسپن وجود دارد)

Nth-child()

توضیح:

این سلکتور المان‌ها را بر اساس موقعیت شمارشی بین تمام فرزندان والد انتخاب می‌کند.

|  |
| --- |
| <ul>  <li>آیتم 1</li>  <li>آیتم 2</li>  <li>آیتم 3</li>  <li>آیتم 4</li>  <li>آیتم 5</li>  <li>آیتم 6</li>  </ul>  <style>  li:nth-child(3) {  background-color: red;  color: white;  }  li:nth-child(2n) {  background-color: #f0f0f0;  }  li:nth-child(2n+1) {  background-color: #e0e0e0;  }  li:nth-child(even) {  border-right: 3px solid blue;  }  li:nth-child(odd) {  border-left: 3px solid green;  }  </style> |

[Nth-child](webdesign/css/pseudo/Nth-child.html)

نتیجه:\*

- آیتم 3: قرمز با متن سفید - که با پس زمینه خاکستری در 2n+1 جابجا میشود.

- آیتم‌های زوج (2,4,6): پس‌زمینه خاکستری روشن + حاشیه آبی

- آیتم‌های فرد (1,3,5): پس‌زمینه خاکستری + حاشیه سبز

Nth-last-child()

توضیح:

این سلکتور مانند `nth-child` است اما شمارش از انتها به ابتدا انجام می‌شود.

|  |
| --- |
| <div class="items">  <div>آیتم 1 (آخر از آخر)</div>  <div>آیتم 2</div>  <div>آیتم 3</div>  <div>آیتم 4</div>  <div>آیتم 5 (اول از آخر)</div>  </div>  <style>  .items {  border: 2px solid #666;  padding: 10px;  margin: 10px;  }  div:nth-last-child(1) {  background-color: gold;  font-weight: bold;  }  div:nth-last-child(2) {  background-color: silver;  }  div:nth-last-child(3) {  background-color: #cd7f32; /\* برنز \*/  }  div:nth-last-child(2n) {  border-bottom: 2px dashed red;  }  div:nth-last-child(2n+1) {  border-top: 2px dotted blue;  }  </style> |

[Nth-last-child](webdesign/css/pseudo/Nth-last-child.html)

نتیجه:

- آیتم 5: طلایی و پررنگ (اول از آخر)

- آیتم 4: نقره‌ای (دوم از آخر)

- آیتم 3: برنزی (سوم از آخر)

- آیتم‌های زوج از آخر (2,4): خط چین قرمز در پایین

- آیتم‌های فرد از آخر (1,3,5): نقطه چین آبی در بالا

Nth-of-type()

توضیح:

این سلکتور المان‌ها را بر اساس موقعیت شمارشی بین المان‌های هم نوع خود انتخاب می‌کند.

|  |
| --- |
| <div class="container">  <p>پاراگراف 1</p>  <span>اسپن 1</span>  <p>پاراگراف 2</p>  <div>دیو 1</div>  <p>پاراگراف 3</p>  <span>اسپن 2</span>  <p>پاراگراف 4</p>  </div>  <style>  .container {  border: 2px solid #333;  padding: 15px;  margin: 10px;  }  /\* هر پاراگراف دوم از نوع خود \*/  p:nth-of-type(2) {  background-color: gold;  padding: 8px;  }  /\* پاراگراف‌های فرد از نوع خود \*/  p:nth-of-type(odd) {  border-right: 3px solid blue;  }  /\* پاراگراف‌های زوج از نوع خود \*/  p:nth-of-type(even) {  border-left: 3px solid green;  }  /\* اسپن دوم از نوع خود \*/  span:nth-of-type(2) {  background-color: lightcoral;  padding: 8px;  }  </style> |

[Nth-of-type](webdesign/css/pseudo/Nth-of-type.html)

نتیجه:

- پاراگراف 2: طلایی (دومین پاراگراف)

- پاراگراف 1 و 3: حاشیه آبی (پاراگراف‌های فرد)

- پاراگراف 2 و 4: حاشیه سبز (پاراگراف‌های زوج)

- اسپن 2: قرمز روشن (دومین اسپن)

Nth-last-of-type()

توضیح:

این سلکتور مانند nth-of-type است اما شمارش از انتها بین المان‌های هم نوع انجام می‌شود.

|  |
| --- |
| <div class="content">  <p>پاراگراف اول (چهارم از آخر)</p>  <span>اسپن اول (دوم از آخر)</span>  <div>دیو اول</div>  <p>پاراگراف دوم (سوم از آخر)</p>  <span>اسپن دوم (اول از آخر)</span>  <p>پاراگراف سوم (دوم از آخر)</p>  <p>پاراگراف چهارم (اول از آخر)</p>  </div>  <style>  .content {  border: 2px solid #666;  padding: 15px;  margin: 10px;  }  /\* پاراگراف اول از آخر \*/  p:nth-last-of-type(1) {  background-color: lightgreen;  padding: 8px;  }  /\* پاراگراف دوم از آخر \*/  p:nth-last-of-type(2) {  background-color: lightblue;  padding: 8px;  }  /\* پاراگراف سوم از آخر \*/  p:nth-last-of-type(3) {  background-color: lightyellow;  padding: 8px;  }  /\* اسپن اول از آخر \*/  span:nth-last-of-type(1) {  border: 2px solid red;  padding: 5px;  }  </style> |

[Nth-last-of-type](webdesign/css/pseudo/Nth-last-of-type.html)

\*نتیجه:

- پاراگراف 4: سبز روشن (اول از آخر)

- پاراگراف 3: آبی روشن (دوم از آخر)

- پاراگراف 2: زرد روشن (سوم از آخر)

- اسپن 2: حاشیه قرمز (اول از آخر)

Enabled

توضیح:

این سلکتور المان‌های فرم را که فعال و قابل استفاده هستند انتخاب می‌کند

|  |
| --- |
| <input type="text" placeholder="این اینپوت فعال است">  <input type="text" disabled placeholder="این غیرفعال است">  <input type="email" placeholder="ایمیل فعال">  <button>دکمه فعال</button>  <button disabled>دکمه غیرفعال</button>  <style>  input:enabled {  background-color: #e8f5e8;  border: 2px solid green;  }  button:enabled {  background-color: #4CAF50;  color: white;  cursor: pointer;  }  </style> |

[Enabled](webdesign/css/pseudo/Enabled.html)

نتیجه:

- اینپوت‌های فعال: پس‌زمینه سبز روشن + حاشیه سبز

- دکمه فعال: سبز + نشانگر دست

- المان‌های غیرفعال تغییری نمی‌کنند

Disabled

توضیح: این سلکتور المان‌های فرم را که غیرفعال شده‌اند انتخاب می‌کند.

|  |
| --- |
| <input type="text" placeholder="فعال">  <input type="text" disabled placeholder="غیرفعال">  <input type="password" disabled placeholder="پسورد غیرفعال">  <select disabled>  <option>انتخاب غیرفعال</option>  </select>  <textarea disabled>این متن غیرفعال است</textarea>  <style>  input:disabled {  background-color: #f5f5f5;  border: 1px solid #ccc;  color: #999;  cursor: not-allowed;  }  select:disabled, textarea:disabled {  background-color: #f0f0f0;  color: #666;  cursor: not-allowed;  }  </style> |

[Disabled](webdesign/css/pseudo/Disabled.html)

نتیجه:

- تمام المان‌های غیرفعال: پس‌زمینه خاکستری + متن کمرنگ + نشانگر ممنوع

Checked

توضیح: این سلکتور المان‌های رادیو و چک‌باکس که انتخاب شده‌اند را انتخاب می‌کند.

|  |
| --- |
| <label>  <input type="checkbox" name="option1"> گزینه اول  </label>  <label>  <input type="checkbox" name="option2" checked> گزینه دوم (پیش‌انتخاب)  </label>  <label>  <input type="radio" name="gender" value="male"> مرد  </label>  <label>  <input type="radio" name="gender" value="female" checked> زن (پیش‌انتخاب)  </label>  <style>  input[type="checkbox"]:checked,  input[type="radio"]:checked {  transform: scale(1.2);  }  input[type="checkbox"]:checked {  accent-color: green;  }  input[type="radio"]:checked {  accent-color: blue;  }  label {  display: block;  margin: 10px 0;  cursor: pointer;  }  </style> |

[Checked](webdesign/css/pseudo/Checked.html)

نتیجه:

- چک‌باکس‌های انتخاب شده: بزرگ‌تر + رنگ سبز

- رادیوهای انتخاب شده: بزرگ‌تر + رنگ آبی

Not()

توضیح: این سلکتور تمام المان‌ها به جز موارد مشخص شده را انتخاب می‌کند.

|  |
| --- |
| <div class="container">  <p class="special">پاراگراف ویژه</p>  <p>پاراگراف معمولی</p>  <p class="special">پاراگراف ویژه دیگر</p>  <p>پاراگراف دیگر</p>  <div>این دیو است</div>  </div>  <input type="text" placeholder="متن معمولی">  <input type="text" disabled placeholder="متن غیرفعال">  <style>  /\* تمام پاراگراف‌ها به جز آنهایی که کلاس special دارند \*/  p:not(.special) {  background-color: #ffe6e6;  padding: 8px;  }  /\* تمام اینپوت‌ها به جز نوع ایمیل \*/  input:not([type="email"]) {  border: 2px solid #333;  margin: 5px;  }  /\* تمام المان‌ها به جز آنهایی که غیرفعال هستند \*/  input:not(:disabled) {  background-color: #f9f9f9;  }  /\* ترکیب چند شرط \*/  p:not(.special):not(:first-child) {  border-right: 3px solid orange;  }  </style> |

[Not](webdesign/css/pseudo/Not.html)

نتیجه:

- پاراگراف‌های بدون کلاس special: پس‌زمینه قرمز روشن

- اینپوت‌های غیرایمیل: حاشیه مشکی

- اینپوت‌های فعال: پس‌زمینه سفید

- پاراگراف‌های معمولی (به جز اولی): حاشیه نارنجی

معرفی pseudo element selector های زیر

First-line

توضیح:

این pseudo-element اولین خط یک المان متنی را انتخاب می‌کند.

|  |
| --- |
| <p class="text">این یک پاراگراف نمونه است که چندین خط متن دارد. این متن برای نمایش دادن نحوه کار first-line نوشته شده است. می‌بینید که چگونه اولین خط متفاوت از بقیه خطوط نمایش داده می‌شود.</p>  <div class="content">این یک دیو با محتوای متنی است. اولین خط این المان نیز استایل متفاوتی خواهد داشت. بقیه متن به صورت عادی نمایش داده می‌شود.</div>  <style>  .text::first-line {  color: red;  font-weight: bold;  font-size: 1.2em;  text-transform: uppercase;  }  .content::first-line {  background-color: yellow;  color: blue;  font-style: italic;  }  </style> |

[First-line](webdesign/css/pseudo/First-line.html)

نتیجه:

- اولین خط پاراگراف: قرمز، پررنگ، بزرگ‌تر و با حروف بزرگ

- اولین خط دیو: زرد، آبی و ایتالیک

First-letter

**توضیح: این pseudo-element اولین حرف یک المان متنی را انتخاب می‌کند.**

|  |
| --- |
| <p class="story">روزی روزگاری در سرزمینی دور، پادشاهی زندگی می‌کرد که...</p>  <div class="poem">به نام خداوند جان و خرد<br>کزین برتر اندیشه برنگذرد</div>  <style>  .story::first-letter {  font-size: 3em;  color: #8B4513;  float: right;  margin-left: 5px;  font-weight: bold;  line-height: 1;  }  .poem::first-letter {  font-size: 2.5em;  color: #2E8B57;  background-color: #F0FFF0;  padding: 5px 10px;  border-radius: 5px;  margin-left: 10px;  }  </style> |

[First-letter](webdesign/css/pseudo/First-letter.html)

نتیجه:

- حرف اول داستان: قهوه‌ای، بزرگ و شناور

- حرف اول شعر: سبز، با پس‌زمینه سبز بسیار روشن

Before

توضیح: این pseudo-element محتوایی را قبل از محتوای اصلی المان اضافه می‌کند.

|  |
| --- |
| <p class="note">این یک یادداشت مهم است</p>  <ul>  <li class="item">آیتم اول</li>  <li class="item">آیتم دوم</li>  <li class="item">آیتم سوم</li>  </ul>  <div class="price">۵۰,۰۰۰ تومان</div>  <style>  .note::before {  content: "📌 ";  color: red;  font-weight: bold;  }  .item::before {  content: "✔ ";  color: green;  margin-left: 10px;  }  .price::before {  content: "قیمت: ";  color: #666;  font-size: 0.9em;  }  </style> |

[Before](webdesign/css/pseudo/Before.html)

نتیجه:

- قبل از یادداشت: آیکون پین قرمز

- قبل از هر آیتم: تیک سبز

- قبل از قیمت: متن "قیمت:"

After

**توضیح: این pseudo-element محتوایی را بعد از محتوای اصلی المان اضافه می‌کند.**

|  |
| --- |
| <a href="#" class="link">مشاهده بیشتر</a>  <p class="required">نام کاربری</p>  <div class="tooltip">این متن راهنما است  <span class="info">i</span>  </div>  <style>  .link::after {  content: " →";  color: #007bff;  font-weight: bold;  }  .required::after {  content: " \*";  color: red;  font-weight: bold;  }  .info::after {  content: " این یک راهنمای متنی است";  background-color: #333;  color: white;  padding: 5px 10px;  border-radius: 4px;  font-size: 0.8em;  position: absolute;  white-space: nowrap;  opacity: 0;  transition: opacity 0.3s;  }  .info:hover::after {  opacity: 1;  }  </style> |

[**after**](webdesign/css/pseudo/After.html)

نتیجه:

- بعد از لینک: فلش آبی

- بعد از فیلد اجباری: ستاره قرمز

- هنگام هاور روی آیکون: نمایش متن راهنما

مثال ترکیبی پیشرفته:

|  |
| --- |
| <blockquote class="fancy-quote">  بهترین راه برای پیش‌بینی آینده، ساختن آن است.  </blockquote>  <div class="feature-list">  <div class="feature">امنیت بالا</div>  <div class="feature">سرعت فوق‌العاده</div>  <div class="feature">پشتیبانی ۲۴ ساعته</div>  </div>  <style>  .fancy-quote::before {  content: "“";  font-size: 4em;  color: #ddd;  float: right;  line-height: 0.8;  margin-left: 10px;  }  .fancy-quote::after {  content: "”";  font-size: 4em;  color: #ddd;  float: left;  line-height: 0.8;  margin-right: 10px;  }  .fancy-quote::first-line {  font-weight: bold;  color: #2c3e50;  }  .feature::before {  content: "✓ ";  color: #27ae60;  font-weight: bold;  margin-left: 10px;  }  .feature::after {  content: " ★";  color: #f39c12;  font-size: 0.8em;  }  .feature::first-letter {  font-size: 1.2em;  color: #e74c3c;  font-weight: bold;  }  </style> |

[complex](webdesign/css/pseudo/complex.html)

نتیجه:

- نقل قول: با گیومه بزرگ، خط اول پررنگ

- ویژگی‌ها: قبل با تیک سبز، بعد با ستاره طلایی، حرف اول بزرگ و قرمز

نکات مهم:

- `::before` و `::after` حتماً نیاز به `content` دارند (حتی اگر خالی باشد: `content: ""`)

- `::first-line` و `::first-letter` فقط روی المان‌های block-level کار می‌کنند

- این pseudo-elementها با `:` دو نقطه نوشته می‌شوند (در CSS3)

## [ساخت یک پروژه ساده](webdesign/css/project-1)

ساختار پیشرفته پروژه با استفاده از @import

|  |
| --- |
| project/  │  ├── index.html  │  ├── assets/  │ ├── css/  │ │ ├── main.css  │ │ ├── header.css  │ │ ├── footer.css  │ │ ├── hero.css  │ │ ├── about.css  │ │ ├── skills.css  │ │ ├── projects.css  │ │ ├── contact.css  │ │ ├── responsive.css  │ │ ├── variables.css  │ │ └── fonts/  │ │ ├── iransans/  │ │ │ ├── iransans.woff  │ │ │ └── iransans.woff2  │ │ └── vazir/  │ │ ├── vazir.woff  │ │ └── vazir.woff2  │ │  │ └── images/  │ ├── profile.jpg  │ ├── project1.jpg  │ ├── project2.jpg  │ └── project3.jpg |

فایل index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>نمونه کار من</title>      <link rel="stylesheet" href="assets/css/main.css">  </head>  <body>      <!-- هدر -->      <header class="header">          <div class="container">              <div class="logo">                  <h1>نمونه کار من</h1>              </div>              <nav class="navigation">                  <ul class="nav-list">                      <li class="nav-item"><a href="#home" class="nav-link">خانه</a></li>                      <li class="nav-item"><a href="#about" class="nav-link">درباره من</a></li>                      <li class="nav-item"><a href="#skills" class="nav-link">مهارت‌ها</a></li>                      <li class="nav-item"><a href="#projects" class="nav-link">پروژه‌ها</a></li>                      <li class="nav-item"><a href="#contact" class="nav-link">تماس</a></li>                  </ul>              </nav>          </div>      </header>      <!-- بخش اصلی -->      <main>          <!-- بخش هیرو -->          <section id="home" class="hero">              <div class="container">                  <div class="hero-content">                      <h2 class="hero-title">سلام، من یک توسعه‌دهنده وب هستم</h2>                      <p class="hero-description">به نمونه کار من خوش آمدید. من علاقه‌مند به ایجاد وبسایت‌های زیبا و کاربردی هستم.</p>                      <a href="#projects" class="hero-button">مشاهده پروژه‌ها</a>                  </div>                  <div class="hero-image">                      <img src="assets/images/profile.png" alt="تصویر پروفایل" class="profile-img">                  </div>              </div>          </section>          <!-- بخش درباره من -->          <section id="about" class="about">              <div class="container">                  <h2 class="section-title">درباره من</h2>                  <div class="about-content">                      <p class="about-text">من یک توسعه‌دهنده وب با تجربه در HTML و CSS هستم. به طراحی وبسایت‌های واکنش‌گرا و کاربرپسند علاقه دارم و همیشه در حال یادگیری تکنولوژی‌های جدید هستم.</p>                      <p class="about-text">هدف من ایجاد تجربه‌های کاربری عالی و رابط‌های کاربری جذاب است که نیازهای کاربران را به خوبی برآورده کند.</p>                  </div>              </div>          </section>          <!-- بخش مهارت‌ها -->          <section id="skills" class="skills">              <div class="container">                  <h2 class="section-title">مهارت‌های من</h2>                  <div class="skills-list">                      <div class="skill-item">                          <h3 class="skill-title">HTML</h3>                          <div class="skill-bar">                              <div class="skill-progress" data-skill="90%"></div>                          </div>                          <span class="skill-percent">90%</span>                      </div>                      <div class="skill-item">                          <h3 class="skill-title">CSS</h3>                          <div class="skill-bar">                              <div class="skill-progress" data-skill="85%"></div>                          </div>                          <span class="skill-percent">85%</span>                      </div>                      <div class="skill-item">                          <h3 class="skill-title">JavaScript</h3>                          <div class="skill-bar">                              <div class="skill-progress" data-skill="70%"></div>                          </div>                          <span class="skill-percent">70%</span>                      </div>                  </div>              </div>          </section>          <!-- بخش پروژه‌ها -->          <section id="projects" class="projects">              <div class="container">                  <h2 class="section-title">پروژه‌های من</h2>                  <div class="projects-grid">                      <div class="project-card">                          <img src="assets/images/project1.jpg" alt="پروژه ۱" class="project-image">                          <div class="project-info">                              <h3 class="project-title">پروژه اول</h3>                              <p class="project-description">توضیحات کوتاه درباره پروژه اول و تکنولوژی‌های استفاده شده در آن.</p>                              <a href="#" class="project-link">مشاهده پروژه</a>                          </div>                      </div>                      <div class="project-card">                          <img src="assets/images/project2.jpg" alt="پروژه ۲" class="project-image">                          <div class="project-info">                              <h3 class="project-title">پروژه دوم</h3>                              <p class="project-description">توضیحات کوتاه درباره پروژه دوم و تکنولوژی‌های استفاده شده در آن.</p>                              <a href="#" class="project-link">مشاهده پروژه</a>                          </div>                      </div>                      <div class="project-card">                          <img src="assets/images/project3.jpg" alt="پروژه ۳" class="project-image">                          <div class="project-info">                              <h3 class="project-title">پروژه سوم</h3>                              <p class="project-description">توضیحات کوتاه درباره پروژه سوم و تکنولوژی‌های استفاده شده در آن.</p>                              <a href="#" class="project-link">مشاهده پروژه</a>                          </div>                      </div>                  </div>              </div>          </section>          <!-- بخش تماس -->          <section id="contact" class="contact">              <div class="container">                  <h2 class="section-title">تماس با من</h2>                  <div class="contact-content">                      <div class="contact-info">                          <h3 class="contact-subtitle">اطلاعات تماس</h3>                          <p class="contact-text">ایمیل: example@example.com</p>                          <p class="contact-text">تلفن: ۰۹۱۲۳۴۵۶۷۸۹</p>                      </div>                      <form class="contact-form">                          <div class="form-group">                              <label for="name" class="form-label">نام</label>                              <input type="text" id="name" class="form-input" required>                          </div>                          <div class="form-group">                              <label for="email" class="form-label">ایمیل</label>                              <input type="email" id="email" class="form-input" required>                          </div>                          <div class="form-group">                              <label for="message" class="form-label">پیام</label>                              <textarea id="message" class="form-textarea" rows="5" required></textarea>                          </div>                          <button type="submit" class="form-button">ارسال پیام</button>                      </form>                  </div>              </div>          </section>      </main>      <!-- فوتر -->      <footer class="footer">          <div class="container">              <p class="footer-text">© 2023 تمامی حقوق محفوظ است.</p>          </div>      </footer>  </body>  </html> |

[index](webdesign/css/project-1/index.html)

فایل main.css (فایل اصلی با @import)

|  |
| --- |
| /\* ایمپورت تمام فایل‌های CSS \*/  @import url("face.css");  @import url('variables.css');  @import url('header.css');  @import url('hero.css');  @import url('about.css');  @import url('skills.css');  @import url('projects.css');  @import url('contact.css');  @import url('footer.css');  @import url('responsive.css');  /\* استایل‌های عمومی \*/  \* {      box-sizing: border-box;      margin: 0;      padding: 0;  }  html {      scroll-behavior: smooth;  }  body {      font-family: var(--font-primary);      font-size: 16px;      line-height: 1.6;      color: var(--text-color);      background-color: var(--white);  }  .container {      max-width: var(--container-width);      margin: 0 auto;      padding: 0 20px;  }  .section-title {      font-size: 2rem;      text-align: center;      margin-bottom: 40px;      color: var(--secondary-color);      position: relative;  }  .section-title::after {      content: '';      display: block;      width: 50px;      height: 3px;      background-color: var(--primary-color);      margin: 10px auto 0;  } |

[Main.css](webdesign/css/project-1/assets/css/main.css)

فایل variables.css

|  |
| --- |
| :root {  /\* رنگ‌ها \*/  --primary-color: #3498db;  --secondary-color: #2c3e50;  --accent-color: #e74c3c;  --light-color: #ecf0f1;  --dark-color: #2c3e50;  --text-color: #333;  --text-light: #777;  --white: #fff;  --black: #000;    /\* فونت‌ها \*/  --font-primary: 'Vazir', 'IRANSans', Tahoma, sans-serif;    /\* اندازه‌ها \*/  --container-width: 1200px;  --header-height: 70px;  --section-padding: 80px 0;  --border-radius: 5px;    /\* سایه‌ها \*/  --shadow-light: 0 2px 5px rgba(0, 0, 0, 0.1);  --shadow-medium: 0 5px 15px rgba(0, 0, 0, 0.1);  --shadow-heavy: 0 10px 30px rgba(0, 0, 0, 0.15);    /\* انتقال‌ها \*/  --transition-fast: 0.3s ease;  --transition-medium: 0.5s ease;  } |

فایل header.css

|  |
| --- |
| .header {      background-color: var(--white);      box-shadow: var(--shadow-light);      position: sticky;      top: 0;      z-index: 10;  }  .header .container {      display: table;      width: 100%;      height: 100%;  }  .logo {      display: table-cell;      vertical-align: middle;      width: 30%;  }  .logo h1 {      color: var(--primary-color);      font-size: 1.5rem;  }  .navigation {      text-align: right;      background-color: white;      font-size: 0.6rem;  }  .nav-list {      list-style-type: none;  }  .nav-item {      display: inline;      margin-left: 20px;  }  .nav-link {      text-decoration: none;      color: var(--dark-color);      font-weight: 500;      padding: 5px 10px;      border-radius: var(--border-radius);      transition: color var(--transition-fast);  }  .nav-link:hover {      color: var(--primary-color);  } |

## Float

ویژگی float تعیین می‌کند که یک عنصر باید در سمت چپ یا راست محتوای خود شناور شود و عناصر دیگر در اطراف آن جریان یابند.

مقادیر ممکن:

- left: عنصر در سمت چپ شناور می‌شود

- right: عنصر در سمت راست شناور می‌شود

- none (پیش‌فرض): عنصر شناور نمی‌شود

- inherit: مقدار از والد به ارث می‌رسد

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Float</title>  <style>  .container {  width: 80%;  margin: 0 auto;  border: 2px solid #333;  padding: 20px;  background-color: #f9f9f9;  }    .float-left {  float: left;  width: 150px;  height: 150px;  background-color: #4CAF50;  margin: 0 15px 15px 0;  padding: 10px;  color: white;  }    .float-right {  float: right;  width: 150px;  height: 150px;  background-color: #2196F3;  margin: 0 0 15px 15px;  padding: 10px;  color: white;  }    .content {  text-align: justify;  line-height: 1.6;  }  </style>  </head>  <body dir="rtl">  <div class="container">  <div class="float-left">عنصر شناور چپ</div>  <div class="float-right">عنصر شناور راست</div>  <div class="content">  <h2>مثال ویژگی float</h2>  <p>این یک متن نمونه است که در اطراف عناصر شناور جریان می‌یابد. عنصر سبز رنگ در سمت چپ و عنصر آبی رنگ در سمت راست شناور شده‌اند. متن به طور خودکار در اطراف این عناصر جریان پیدا می‌کند و فضای خالی را پر می‌کند.</p>  <p>ویژگی float برای ایجاد طرح‌بندی‌های ساده و قرار دادن عناصر در کنار یکدیگر بسیار مفید است. با این حال، برای طرح‌بندی‌های پیچیده‌تر، معمولاً از Flexbox یا Grid استفاده می‌شود.</p>  </div>  </div>  </body>  </html> |

[float](webdesign/css/Float/index.html)

Clear

ویژگی `clear` کنترل می‌کند که یک عنصر در کنار عناصر شناور قرار گیرد یا در زیر آنها.

مقادیر ممکن:

- `none` (پیش‌فرض): عنصر می‌تواند در کنار عناصر شناور قرار گیرد

- `left`: عنصر در زیر عناصر شناور چپ قرار می‌گیرد

- `right`: عنصر در زیر عناصر شناور راست قرار می‌گیرد

- `both`: عنصر در زیر تمام عناصر شناور قرار می‌گیرد

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Clear</title>  <style>  .container {  width: 80%;  margin: 0 auto;  border: 2px solid #333;  padding: 20px;  background-color: #f9f9f9;  }    .float-box {  float: left;  width: 100px;  height: 100px;  background-color: #FF9800;  margin: 0 15px 15px 0;  padding: 10px;  color: white;  }    .normal-paragraph {  background-color: #E0E0E0;  padding: 10px;  margin-bottom: 10px;  }    .cleared-paragraph {  background-color: #E0E0E0;  padding: 10px;  clear: both; /\* این پاراگراف در زیر عنصر شناور قرار می‌گیرد \*/  }  </style>  </head>  <body dir="rtl">  <div class="container">  <div class="float-box">عنصر شناور</div>  <p class="normal-paragraph">این پاراگراف در کنار عنصر شناور قرار می‌گیرد و متن در اطراف آن جریان می‌یابد.</p>  <p class="cleared-paragraph">این پاراگراف با ویژگی clear:both در زیر عنصر شناور قرار می‌گیرد و از ادامه جریان متن جلوگیری می‌کند.</p>  </div>  </body>  </html> |

[clear](webdesign/css/Float/clear.html)

Clearfix

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <style>  .parent {  border: 2px solid red;  background: #f0f0f0;  margin: 10px;  padding: 10px;  }    .child {  float: right;  width: 100px;  height: 100px;  background: blue;  margin: 10px;  }    /\* Clearfix ساده \*/  .clearfix::after {  content: "";  display: table;  clear: both;  }  </style>  </head>  <body>  <h3>بدون Clearfix (مشکل)</h3>  <div class="parent">  <div class="child"></div>  <div class="child"></div>  </div>  <h3>با Clearfix (حل شده)</h3>  <div class="parent clearfix">  <div class="child"></div>  <div class="child"></div>  </div>  </body>  </html> |

[clearfix](webdesign/css/Float/Clearfix.html)

## Shape-outside

ویژگی shape-outside شکل هندسی را تعریف می‌کند که متن در اطراف آن جریان می‌یابد. این ویژگی فقط روی عناصر شناور (float) کار می‌کند.

مقادیر ممکن:

- `none` (پیش‌فرض): شکل مستطیلی معمولی

- circle(): شکل دایره‌ای

- ellipse(): شکل بیضی

- polygon(): شکل چندضلعی

- url(): شکل بر اساس تصویر با آلفا کانال

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Shape-outside</title>  <style>  .container {  width: 80%;  margin: 0 auto;  border: 2px solid #333;  padding: 20px;  background-color: #f9f9f9;  }    .circle {  width: 200px;  height: 200px;  float: left;  shape-outside: circle(50%);  clip-path: circle(50%);  background-color: #9C27B0;  margin: 0 20px 20px 0;  }    .triangle {  width: 200px;  height: 200px;  float: right;  shape-outside: polygon(0 0, 100% 100%, 0 100%);  clip-path: polygon(0 0, 100% 100%, 0 100%);  background-color: #F44336;  margin: 0 0 20px 20px;  }    .content {  text-align: justify;  line-height: 1.6;  }  </style>  </head>  <body dir="rtl">  <div class="container">  <div class="circle"></div>  <div class="triangle"></div>  <div class="content">  <h2>مثال ویژگی shape-outside</h2>  <p>در این مثال، از ویژگی shape-outside برای ایجاد شکل‌های غیرمستطیلی استفاده شده است. عنصر بنفش یک دایره است و عنصر قرمز یک مثلث. متن به طور خودکار در اطراف این شکل‌ها جریان می‌یابد و طرح‌بندی جذابی ایجاد می‌کند.</p>  <p>ویژگی shape-outside به طراحان این امکان را می‌دهد که طرح‌بندی‌های خلاقانه‌تری ایجاد کنند و متن را به شکل‌های غیرمعمول در اطراف عناصر جریان دهند.</p>  <p>برای نمایش بصری شکل‌ها، از ویژگی clip-path نیز استفاده شده است که شکل عنصر را به شکل تعریف شده در shape-outside محدود می‌کند.</p>  </div>  </div>  </body>  </html> |

[Shape-outside](webdesign/css/Float/Shape-outside.html)

ابزارهای آنلاین برای ساخت

|  |  |
| --- | --- |
| نام ابزار | ویژگی‌های کلیدی |
| [Clippy](https://bennettfeely.com/clippy/) | محبوبترین ابزار؛ پیش‌طرح‌های آماده (مثل مثلث، ستاره)؛ پشتیبانی از `inset()` با قابلیت گرد کردن گوشه |
| [CSS Portal Generator](https://www.cssportal.com/css-clip-path-generator/) | رابط کاربری بصری؛ آپلود تصویر شخصی برای تست شکل؛ تولید خودکار کد |
| [UnusedCSS Generator](https://unused-css.com/tools/clip-path-generator) | سیستم Grid پیشرفته برای تراز حرفه‌ای نقاط؛ قابلیت ایجاد حفره در شکل |
| [SEOMI Tool](https://seomi.ir/%D8%A7%D8%A8%D8%B2%D8%A7%D8%B1-%D8%B3%D8%A7%D8%AE%D8%AA-%D8%B4%DA%A9%D9%84-%D8%A8%D8%A7-css-clip-path/) | کنترل جزئیات (تعداد نقاط، گرادیان، انیمیشن)؛ رابط کاربری فارسی |

## Shape-margin

ویژگی shape-margin فاصله بین شکل تعریف شده توسط shape-outside و محتوای اطراف آن را تنظیم می‌کند.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Shape-margin</title>  <style>  .container {  width: 80%;  margin: 0 auto;  border: 2px solid #333;  padding: 20px;  background-color: #f9f9f9;  }    .shape {  width: 200px;  height: 200px;  float: left;  shape-outside: circle(50%);  clip-path: circle(50%);  background-color: #009688;  }    .no-margin {  shape-margin: 0;  margin: 0 20px 20px 0;  }    .with-margin {  shape-margin: 30px;  margin: 0 20px 20px 0;  }    .content {  text-align: justify;  line-height: 1.6;  }    .comparison {  display: flex;  justify-content: space-between;  margin-bottom: 30px;  }    .example {  width: 48%;  border: 1px solid #ccc;  padding: 15px;  background-color: white;  }  </style>  </head>  <body dir="rtl">  <div class="container">  <h2>مقایسه shape-margin</h2>    <div class="comparison">  <div class="example">  <h3>بدون shape-margin</h3>  <div class="shape no-margin"></div>  <div class="content">  <p>در این مثال، از shape-margin استفاده نشده است. متن مستقیماً در کنار شکل دایره‌ای قرار می‌گیرد و فاصله‌ای بین متن و شکل وجود ندارد.</p>  </div>  </div>    <div class="example">  <h3>با shape-margin: 30px</h3>  <div class="shape with-margin"></div>  <div class="content">  <p>در این مثال، از shape-margin: 30px استفاده شده است. همانطور که مشاهده می‌کنید، فاصله 30 پیکسلی بین شکل دایره‌ای و متن ایجاد شده است که خوانایی را بهبود می‌بخشد.</p>  </div>  </div>  </div>    <div class="content">  <h3>نتیجه‌گیری</h3>  <p>ویژگی shape-margin برای ایجاد فاصله بین شکل تعریف شده و محتوای اطراف آن استفاده می‌شود. این ویژگی خوانایی را بهبود می‌بخشد و ظاهر زیباتری به طرح‌بندی می‌دهد.</p>  </div>  </div>  </body>  </html> |

[Shape-margin](webdesign/css/Float/Shape-margin.html)

مثال ترکیبی از تمام ویژگی‌ها

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>مثال ترکیبی Float, Clear, Shape-outside, Shape-margin</title>  <style>  .container {  width: 90%;  margin: 0 auto;  border: 2px solid #333;  padding: 20px;  background-color: #f9f9f9;  }    .custom-shape {  width: 250px;  height: 250px;  float: left;  shape-outside: polygon(0 0, 100% 0, 80% 100%, 0% 100%);  clip-path: polygon(0 0, 100% 0, 80% 100%, 0% 100%);  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  shape-margin: 20px;  margin: 0 25px 25px 0;  }    .floating-box {  float: right;  width: 150px;  height: 150px;  background-color: #FF5722;  margin: 0 0 15px 25px;  padding: 10px;  color: white;  }    .cleared-section {  clear: both;  background-color: #E1F5FE;  padding: 20px;  margin-top: 20px;  border-radius: 5px;  }    .content {  text-align: justify;  line-height: 1.7;  }    h2 {  color: #333;  border-bottom: 2px solid #667eea;  padding-bottom: 10px;  }  </style>  </head>  <body dir="rtl">  <div class="container">  <div class="custom-shape"></div>  <div class="floating-box">عنصر شناور ساده</div>    <div class="content">  <h2>مثال ترکیبی از ویژگی‌های CSS</h2>  <p>در این مثال، از چندین ویژگی CSS به صورت ترکیبی استفاده شده است. عنصر آبی-بنفش با ویژگی‌های float، shape-outside و shape-margin یک شکل چندضلعی ایجاد کرده و متن در اطراف آن جریان یافته است.</p>  <p>عنصر نارنجی نیز یک عنصر شناور ساده است که در سمت راست قرار گرفته است. ترکیب این عناصر با یکدیگر یک طرح‌بندی پویا و جذاب ایجاد کرده است.</p>  <p>ویژگی‌های shape-outside و shape-margin به طراحان وب این امکان را می‌دهند که طرح‌بندی‌های خلاقانه‌تری ایجاد کنند و از طرح‌بندی‌های مستطیلی سنتی فراتر روند.</p>  </div>    <div class="cleared-section">  <h3>بخش cleared</h3>  <p>این بخش با ویژگی clear:both در زیر تمام عناصر شناور قرار گرفته است. از این ویژگی برای اطمینان از اینکه یک عنصر در زیر عناصر شناور قرار می‌گیرد استفاده می‌شود.</p>  </div>  </div>  </body>  </html> |

[all](webdesign/css/Float/all.html)

نکات مهم:

1. Float برای طرح‌بندی‌های ساده مفید است اما برای طرح‌بندی‌های پیچیده‌تر، Flexbox و Grid گزینه‌های بهتری هستند.

2. Clear برای حل مشکلات مربوط به float استفاده می‌شود و اطمینان می‌دهد که عناصر به درستی در زیر عناصر شناور قرار می‌گیرند.

3. Shape-outside و Shape-margin ویژگی‌های نسبتاً جدیدی هستند و پشتیبانی مرورگرها را بررسی کنید.

4. برای عناصر شناور، همیشه عرض (width) مشخص کنید تا نتایج قابل پیش‌بینی‌تری داشته باشید.

این ویژگی‌ها در کنار هم می‌توانند برای ایجاد طرح‌بندی‌های خلاقانه و جذاب در طراحی وب مورد استفاده قرار گیرند.

## [Position(static,relative,absolute,fixed,sticky)](webdesign/css/Position)

ویژگی `position`

این ویژگی نحوه قرارگیری یک عنصر در صفحه را مشخص می‌کند.

۱. static (پیش‌فرض)

|  |
| --- |
| <div class="box static">Static</div>  .static {  position: static; /\* عنصر در جریان عادی صفحه قرار می‌گیرد \*/  top: 20px; /\* ❌ تاثیری ندارد \*/  left: 20px; /\* ❌ تاثیری ندارد \*/  } |

[static](webdesign/css/Position/static.html)

relative

relative: نسبت به موقعیت اصلی خودش جابه‌جا می‌شود

|  |
| --- |
| <div class="container">  <div class="box relative">Relative</div>  </div>  .relative {  position: relative;  top: 20px; /\* 20px به سمت پایین حرکت می‌کند \*/  left: 30px; /\* 30px به سمت راست حرکت می‌کند \*/  background-color: lightblue;  } |

[relative](webdesign/css/Position/relative.html)

absolute

absolute: نسبت به نزدیکترین والد خود از جنس relative قرار می‌گیرد

|  |
| --- |
| <div class="container" style="position: relative;">  <div class="box absolute">Absolute</div>  </div>  .absolute {  position: absolute;  top: 10px; /\* 10px از بالای والد نسبی \*/  right: 15px; /\* 15px از راست والد نسبی \*/  background-color: lightcoral;  } |

[absolute](webdesign/css/Position/Absolute.html)

fixed

fixed: نسبت به صفحه نمایش ثابت می‌ماند

|  |
| --- |
| <div class="box fixed">Fixed - همیشه ثابت</div>  .fixed {  position: fixed;  bottom: 20px; /\* 20px از پایین صفحه \*/  right: 20px; /\* 20px از راست صفحه \*/  background-color: lightgreen;  z-index: 1000;  } |

[fixed](webdesign/css/Position/fixed.html)

sticky

sticky: بین relative و fixed رفتار می‌کند یعنی یک موقعیت در صفحه دارد مثلا top:0 اما تا زمانیکه اسکرول مرورگر به آن نقطه نرسد کاری انجام نمیدهد.

|  |
| --- |
| <div class="sticky-header">Sticky Header</div>  <div style="height: 2000px;">محتوای بلند...</div>  .sticky-header {  position: sticky;  top: 0; /\* وقتی به بالای صفحه می‌رسد، می‌چسبد \*/  background-color: #333;  color: white;  padding: 15px;  z-index: 100;  } |

[sticky](webdesign/css/Position/sticky.html)

## Top , left , right , bottom

این ویژگی‌ها موقعیت عنصر را نسبت به لبه‌ها مشخص می‌کنند.

|  |
| --- |
| .element {  position: absolute;  top: 50px; /\* 50px از بالا \*/  right: 100px; /\* 100px از راست \*/  bottom: 20px; /\* 20px از پایین \*/  left: 30px; /\* 30px از چپ \*/  } |

[Top-left-right-bottom](webdesign/css/Position/top-r-l-b.html)

## Z index

ترتیب نمایش عناصر روی هم را کنترل می‌کند.

|  |
| --- |
| <div class="box box1">زمینه 1 (z-index: 1)</div>  <div class="box box2">زمینه 2 (z-index: 2)</div>  <div class="box box3">زمینه 3 (z-index: 3)</div>  .box {  position: absolute;  width: 200px;  height: 200px;  }  .box1 {  background-color: red;  z-index: 1;  top: 50px;  left: 50px;  }  .box2 {  background-color: blue;  z-index: 2; /\* روی box1 قرار می‌گیرد \*/  top: 100px;  left: 100px;  }  .box3 {  background-color: green;  z-index: 3; /\* روی همه قرار می‌گیرد \*/  top: 150px;  left: 150px;  } |

[z-index](webdesign/css/Position/Z-index.html)

مثال کاربردی ترکیبی

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <style>  \* {  box-sizing: border-box;  }    body {  margin: 0;  height: 2000px;  font-family: Arial, sans-serif;  }    .header {  position: sticky;  top: 0;  background: #2c3e50;  color: white;  padding: 15px;  z-index: 1000;  text-align: center;  }    .container {  position: relative;  width: 80%;  height: 400px;  margin: 50px auto;  border: 2px solid #34495e;  background: #ecf0f1;  }    .relative-box {  position: relative;  top: 20px;  left: -30px;  width: 200px;  height: 100px;  background: #3498db;  color: white;  padding: 10px;  }    .absolute-box {  position: absolute;  top: 50px;  right: 50px;  width: 150px;  height: 150px;  background: #e74c3c;  color: white;  padding: 10px;  z-index: 10;  }    .fixed-button {  position: fixed;  bottom: 30px;  left: 30px;  background: #27ae60;  color: white;  padding: 15px 25px;  border: none;  border-radius: 5px;  cursor: pointer;  z-index: 900;  }    .tooltip {  position: absolute;  top: -40px;  left: 50%;  transform: translateX(-50%);  background: #333;  color: white;  padding: 5px 10px;  border-radius: 3px;  z-index: 20;  }  </style>  </head>  <body>  <div class="header">هدر Sticky - هنگام اسکرول می‌چسبد</div>    <div class="container">  <div class="relative-box">  جعبه Relative  <div class="tooltip">Tooltip با Absolute</div>  </div>    <div class="absolute-box">جعبه Absolute</div>  </div>    <button class="fixed-button">دکمه Fixed</button>  </body>  </html> |

[combine](webdesign/css/Position/combine.html)

|  |  |  |
| --- | --- | --- |
| Position Type | نسبت به چه چیزی موقعیت‌دهی می‌شود | در جریان عادی صفحه |
| `static` | - | ✓ بله |
| `relative` | موقعیت اصلی خودش | ✓ بله |
| `absolute` | نزدیکترین والد positioned | ✌ خیر |
| `fixed` | viewport (صفحه نمایش) | ✌ خیر |
| `sticky` | scroll container | ✓ بله |

نکات کلیدی:

- `z-index` فقط روی عناصر positioned کار می‌کند

- برای `absolute`، والد باید `position: relative` داشته باشد

- `sticky` بین `relative` و `fixed` رفتار می‌کند

- مقادیر منفی در `z-index` مجاز هستند

## ساخت منو nested

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>سایت آموزشی - منوی بازشونده</title>  <style>  /\* استایل‌های پایه \*/  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }  body {  background-color: #f5f5f5;  line-height: 1.6;  }  /\* هدر \*/  header {  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  color: white;  padding: 1rem 2rem;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  }  .header-content {  display: flex;  justify-content: space-between;  align-items: center;  max-width: 1200px;  margin: 0 auto;  }  .logo {  font-size: 1.8rem;  font-weight: bold;  }  /\* ناوبری اصلی \*/  nav {  background-color: white;  padding: 0 2rem;  box-shadow: 0 2px 5px rgba(0,0,0,0.1);  position: sticky;  top: 0;  z-index: 100;  }  .nav-menu {  list-style: none;  display: flex;  max-width: 1200px;  margin: 0 auto;  }  .nav-item {  position: relative;  }  .nav-link {  display: block;  padding: 1rem 1.5rem;  color: #333;  text-decoration: none;  transition: all 0.3s ease;  border-bottom: 3px solid transparent;  }  .nav-link:hover {  background-color: #f8f9fa;  border-bottom: 3px solid #667eea;  color: #667eea;  }  /\* منوی بازشونده \*/  .dropdown-menu {  position: absolute;  top: 100%;  right: 0;  background-color: white;  min-width: 200px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  border-radius: 5px;  opacity: 0;  visibility: hidden;  transform: translateY(-10px);  transition: all 0.3s ease;  list-style: none;  }  .dropdown-item {  border-bottom: 1px solid #f0f0f0;  }  .dropdown-item:last-child {  border-bottom: none;  }  .dropdown-link {  display: block;  padding: 0.8rem 1rem;  color: #333;  text-decoration: none;  transition: all 0.3s ease;  }  .dropdown-link:hover {  background-color: #667eea;  color: white;  padding-right: 1.5rem;  }  /\* نمایش منوی بازشونده هنگام هاور \*/  .nav-item:hover .dropdown-menu {  opacity: 1;  visibility: visible;  transform: translateY(0);  }  /\* محتوای اصلی \*/  .main-content {  max-width: 1200px;  margin: 2rem auto;  padding: 0 2rem;  }  .content-section {  background: white;  padding: 2rem;  border-radius: 10px;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  margin-bottom: 2rem;  }  h1 {  color: #333;  margin-bottom: 1rem;  }  p {  color: #666;  margin-bottom: 1rem;  }  /\* فوتر \*/  footer {  background-color: #333;  color: white;  text-align: center;  padding: 2rem;  margin-top: 3rem;  }  </style>  </head>  <body>  <!-- هدر سایت -->  <header>  <div class="header-content">  <div class="logo">آکادمی برنامه‌نویسی</div>  <div>آموزش HTML, CSS, JavaScript</div>  </div>  </header>  <!-- ناوبری اصلی -->  <nav>  <ul class="nav-menu">  <li class="nav-item">  <a href="#" class="nav-link">خانه</a>  </li>  <li class="nav-item">  <a href="#" class="nav-link">دوره‌ها</a>  <!-- منوی بازشونده -->  <ul class="dropdown-menu">  <li class="dropdown-item">  <a href="#" class="dropdown-link">HTML & CSS</a>  </li>  <li class="dropdown-item">  <a href="#" class="dropdown-link">JavaScript</a>  </li>  <li class="dropdown-item">  <a href="#" class="dropdown-link">React.js</a>  </li>  <li class="dropdown-item">  <a href="#" class="dropdown-link">Node.js</a>  </li>  </ul>  </li>  <li class="nav-item">  <a href="#" class="nav-link">مقالات</a>  </li>  <li class="nav-item">  <a href="#" class="nav-link">درباره ما</a>  </li>  <li class="nav-item">  <a href="#" class="nav-link">تماس با ما</a>  </li>  </ul>  </nav>  <!-- محتوای اصلی -->  <main class="main-content">  <section class="content-section">  <h1>به سایت آموزشی ما خوش آمدید</h1>  <p>این یک سایت نمونه برای آموزش ایجاد منوی بازشونده با HTML و CSS خالص است.</p>  <p>منوی بازشونده در بخش "دوره‌ها" قابل مشاهده است. روی آن هاور کنید تا زیرمنو نمایش داده شود.</p>  </section>  </main>  <!-- فوتر -->  <footer>  <p>© 2024 آکادمی برنامه‌نویسی - تمام حقوق محفوظ است</p>  </footer>  </body>  </html> |

[nested](webdesign/css/menu/nested.html)

۱. ساختار HTML:

<header>: بخش بالایی سایت (هدر)

<nav>: بخش ناوبری اصلی

<ul class="nav-menu">: لیست منوی اصلی

<ul class="dropdown-menu">: لیست منوی بازشونده

<main>: محتوای اصلی سایت

۲. مفاهیم CSS مهم:

Position برای منوی بازشونده:

|  |
| --- |
| .nav-item {  position: relative; /\* مبدا برای منوی بازشونده \*/  }  .dropdown-menu {  position: absolute; /\* نسبت به والد relative قرار می‌گیرد \*/  top: 100%; /\* دقیقاً زیر والد \*/  right: 0;  } |

نمایش و مخفی کردن منو:

|  |
| --- |
| .dropdown-menu {  opacity: 0;  visibility: hidden;  transform: translateY(-10px);  }  .nav-item:hover .dropdown-menu {  opacity: 1;  visibility: visible;  transform: translateY(0);  } |

Transition برای انیمیشن:

|  |
| --- |
| .dropdown-menu {  transition: all 0.3s ease; /\* انیمیشن نرم \*/  } |

position: relative در .nav-item مبدا را برای منوی بازشونده مشخص می‌کند

position: absolute در .dropdown-menu منو را از جریان عادی خارج می‌کند

hover با CSS منو را نمایش/مخفی می‌کند

transition برای ایجاد انیمیشن نرم استفاده شده

z-index اطمینان می‌دهد منو روی سایر عناصر نمایش داده شود

## ساخت mega menu

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>فروشگاه اینترنتی - مگامنو</title>  <style>  /\* استایل‌های پایه \*/  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }  body {  background-color: #f8f9fa;  }  /\* هدر \*/  header {  background: linear-gradient(135deg, #2c3e50, #3498db);  color: white;  padding: 1rem 0;  box-shadow: 0 2px 15px rgba(0,0,0,0.1);  }  .header-top {  padding: 0 2rem;  max-width: 1400px;  margin: 0 auto;  }  .logo {  font-size: 2rem;  font-weight: bold;  color: white;  }  .search-box input {  padding: 0.8rem;  border: none;  border-radius: 5px 0 0 5px;  font-size: 1rem;  }  .search-box button {  padding: 0.8rem 1.5rem;  background: #e74c3c;  color: white;  border: none;  border-radius: 0 5px 5px 0;  cursor: pointer;  }  .user-actions a {  color: white;  text-decoration: none;  margin-right: 1rem;  }  /\* مگامنو \*/  .mega-menu {  background: white;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  position: relative;  z-index: 1000;  }  .mega-menu-container {  max-width: 1400px;  margin: 0 auto;  padding: 0 2rem;  position: relative;  }  .main-nav {  list-style: none;  }  .nav-link {  display: block;  padding: 1.2rem 1.5rem;  color: #2c3e50;  text-decoration: none;  font-weight: 600;  transition: all 0.3s ease;  border-bottom: 3px solid transparent;  }  .nav-link:hover {  color: #3498db;  border-bottom: 3px solid #3498db;  background: #f8f9fa;  }  /\* محتوای مگامنو \*/  .mega-content {  position: absolute;  top: 100%;  left: 0;  right: 0;  background: white;  box-shadow: 0 10px 30px rgba(0,0,0,0.15);  border-top: 3px solid #3498db;  opacity: 0;  visibility: hidden;  transform: translateY(-10px);  transition: all 0.3s ease;  padding: 2rem;  }  .mega-content-grid {  max-width: 1400px;  margin: 0 auto;  }  .mega-column{  float: left;  width: 25%;  }  .mega-content-grid::after {  content: "";  display: table;  clear: both;  }  .mega-column h3 {  color: #2c3e50;  margin-bottom: 1rem;  padding-bottom: 0.5rem;  border-bottom: 2px solid #3498db;  }  .mega-column ul {  list-style: none;  }  .mega-column li {  margin-bottom: 0.5rem;  }  .mega-column a {  color: #555;  text-decoration: none;  transition: all 0.3s ease;  display: block;  padding: 0.3rem 0;  }  .mega-column a:hover {  color: #3498db;  padding-right: 0.5rem;  }  .nav-item{  display: inline-block;  width: fit-content;  }  /\* نمایش مگامنو \*/  .nav-item:hover .mega-content {  opacity: 1;  visibility: visible;  transform: translateY(0);  }  /\* محتوای اصلی \*/  .main-content {  max-width: 1200px;  margin: 3rem auto;  padding: 0 2rem;  text-align: center;  }  .main-content h1 {  color: #2c3e50;  margin-bottom: 1rem;  }  .main-content p {  color: #666;  line-height: 1.8;  font-size: 1.1rem;  }  </style>  </head>  <body>  <!-- هدر سایت -->  <header>  <div class="header-top">  <div class="logo">فروشگاه اینترنتی</div>    <div class="search-box">  <input type="text" placeholder="جستجوی محصولات...">  <button>جستجو</button>  </div>    <div class="user-actions">  <a href="#">ورود</a>  <a href="#">ثبت نام</a>  <a href="#">سبد خرید</a>  </div>  </div>  </header>  <!-- مگامنو -->  <nav class="mega-menu">  <div class="mega-menu-container">  <ul class="main-nav">  <li class="nav-item">  <a href="#" class="nav-link">کالای دیجیتال</a>  <div class="mega-content">  <div class="mega-content-grid">  <div class="mega-column">  <h3>موبایل و تبلت</h3>  <ul>  <li><a href="#">گوشی موبایل</a></li>  <li><a href="#">تبلت</a></li>  <li><a href="#">لوازم جانبی موبایل</a></li>  <li><a href="#">سیم کارت</a></li>  <li><a href="#">قاب و کاور</a></li>  </ul>  </div>  <div class="mega-column">  <h3>لپ تاپ و کامپیوتر</h3>  <ul>  <li><a href="#">لپ تاپ</a></li>  <li><a href="#">کامپیوتر رومیزی</a></li>  <li><a href="#">مانیتور</a></li>  <li><a href="#">قطعات کامپیوتر</a></li>  <li><a href="#">پرینتر و اسکنر</a></li>  </ul>  </div>  <div class="mega-column">  <h3>صوتی و تصویری</h3>  <ul>  <li><a href="#">تلویزیون</a></li>  <li><a href="#">هدفون و هدست</a></li>  <li><a href="#">اسپیکر</a></li>  <li><a href="#">سیستم صوتی</a></li>  <li><a href="#">دوربین</a></li>  </ul>  </div>  <div class="mega-column">  <h3>گیمینگ</h3>  <ul>  <li><a href="#">کنسول بازی</a></li>  <li><a href="#">دسته بازی</a></li>  <li><a href="#">گیم پی سی</a></li>  <li><a href="#">نرم افزار بازی</a></li>  <li><a href="#">تجهیزات گیمینگ</a></li>  </ul>  </div>  </div>  </div>  </li>  <li class="nav-item">  <a href="#" class="nav-link">پوشاک</a>  <div class="mega-content">  <div class="mega-content-grid">  <div class="mega-column">  <h3>مردانه</h3>  <ul>  <li><a href="#">پیراهن</a></li>  <li><a href="#">شلوار</a></li>  <li><a href="#">تیشرت</a></li>  <li><a href="#">کت و شلوار</a></li>  <li><a href="#">کفش</a></li>  </ul>  </div>  <div class="mega-column">  <h3>زنانه</h3>  <ul>  <li><a href="#">لباس مجلسی</a></li>  <li><a href="#">بلوز و شلوار</a></li>  <li><a href="#">مانتو</a></li>  <li><a href="#">شال و روسری</a></li>  <li><a href="#">کفش</a></li>  </ul>  </div>  <div class="mega-column">  <h3>بچه گانه</h3>  <ul>  <li><a href="#">پسرانه</a></li>  <li><a href="#">دخترانه</a></li>  <li><a href="#">نوزاد</a></li>  <li><a href="#">اسباب بازی</a></li>  <li><a href="#">کفش کودک</a></li>  </ul>  </div>  <div class="mega-column">  <h3>ورزشی</h3>  <ul>  <li><a href="#">لباس ورزشی</a></li>  <li><a href="#">کفش ورزشی</a></li>  <li><a href="#">تجهیزات ورزشی</a></li>  <li><a href="#">کوله پشتی</a></li>  <li><a href="#">کلاه و عینک</a></li>  </ul>  </div>  </div>  </div>  </li>  <li class="nav-item">  <a href="#" class="nav-link">خانه و آشپزخانه</a>  <div class="mega-content">  <div class="mega-content-grid">  <div class="mega-column">  <h3>لوازم خانگی</h3>  <ul>  <li><a href="#">یخچال و فریزر</a></li>  <li><a href="#">ماشین لباسشویی</a></li>  <li><a href="#">اجاق گاز</a></li>  <li><a href="#">مایکروویو</a></li>  <li><a href="#">هود</a></li>  </ul>  </div>  <div class="mega-column">  <h3>دکوراسیون</h3>  <ul>  <li><a href="#">مبلمان</a></li>  <li><a href="#">پرده</a></li>  <li><a href="#">فرش و گلیم</a></li>  <li><a href="#">آینه و تابلو</a></li>  <li><a href="#">ساعت دیواری</a></li>  </ul>  </div>  <div class="mega-column">  <h3>آشپزخانه</h3>  <ul>  <li><a href="#">ظروف آشپزخانه</a></li>  <li><a href="#">قابلمه و تابه</a></li>  <li><a href="#">چاقو</a></li>  <li><a href="#">لوازم برقی</a></li>  <li><a href="#">سرویس ظروف</a></li>  </ul>  </div>  <div class="mega-column">  <h3>خواب و حمام</h3>  <ul>  <li><a href="#">رختخواب</a></li>  <li><a href="#">ملحفه</a></li>  <li><a href="#">پتو</a></li>  <li><a href="#">لوازم حمام</a></li>  <li><a href="#">حوله</a></li>  </ul>  </div>  </div>  </div>  </li>  <li class="nav-item">  <a href="#" class="nav-link">سوپرمارکت</a>  </li>  <li class="nav-item">  <a href="#" class="nav-link">فروش ویژه</a>  </li>  </ul>  </div>  </nav>  <!-- محتوای اصلی -->  <main class="main-content">  <h1>به فروشگاه اینترنتی ما خوش آمدید</h1>  <p>این یک سایت نمونه با مگامنو است. روی آیتم‌های منو هاور کنید تا مگامنوهای مختلف را مشاهده کنید.</p>  <p>مگامنوها محتوای گسترده‌ای از دسته‌بندی‌های مختلف را نمایش می‌دهند.</p>  </main>  </body>  </html> |

[megamenu](webdesign/css/menu/megamenu.html)

## ساخت منو همبرگری

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>منوهای متحرک</title>  <style>  body {  background-color: #efefef;  font-family: 'Roboto', sans-serif;  color: #333;  margin: 0;  padding: 20px;  }  .container {  max-width: 800px;  margin: 0 auto;  }  h1 {  text-align: center;  margin-bottom: 30px;  }  .menu-section {  background-color: white;  border-radius: 8px;  padding: 20px;  margin-bottom: 20px;  box-shadow: 0 2px 5px rgba(0,0,0,0.1);  }  .menu-section h2 {  margin-top: 0;  margin-bottom: 15px;  font-size: 1.2rem;  color: #555;  }  .menu-container {  display: flex;  flex-wrap: wrap;  gap: 20px;  justify-content: center;  }  .menu {  position: relative;  width: 60px;  height: 60px;  cursor: pointer;  transition-duration: 0.5s;  }  /\* استایل‌های مشترک برای آیکون‌ها \*/  .menu .icon,  .menu .icon-left,  .menu .icon-right,  .menu .icon-E,  .menu .icon-X {  transition-duration: 0.5s;  position: absolute;  height: 8px;  background-color: #212121;  }  .menu .icon::before,  .menu .icon::after,  .menu .icon-left::before,  .menu .icon-left::after,  .menu .icon-right::before,  .menu .icon-right::after,  .menu .icon-E::before,  .menu .icon-E::after,  .menu .icon-X::before,  .menu .icon-X::after {  transition-duration: 0.5s;  position: absolute;  width: 100%;  height: 8px;  background-color: #212121;  content: "";  }  /\* استایل‌های خاص برای هر منو \*/  /\* منوی 1 \*/  .btn1 .icon-left {  width: 30px;  top: 30px;  left: 0px;  }  .btn1 .icon-left::before {  top: -20px;  }  .btn1 .icon-left::after {  top: 20px;  }  .btn1 .icon-right {  width: 30px;  top: 30px;  left: 30px;  }  .btn1 .icon-right::before {  top: -20px;  }  .btn1 .icon-right::after {  top: 20px;  }  .btn1.open .icon-left {  background: transparent;  }  .btn1.open .icon-left::before {  transform: rotateZ(45deg) scaleX(1.4) translate(4px, 4px);  }  .btn1.open .icon-left::after {  transform: rotateZ(-45deg) scaleX(1.4) translate(4px, -4px);  }  .btn1.open .icon-right {  background: transparent;  }  .btn1.open .icon-right::before {  transform: rotateZ(-45deg) scaleX(1.4) translate(-4px, 4px);  }  .btn1.open .icon-right::after {  transform: rotateZ(45deg) scaleX(1.4) translate(-4px, -4px);  }  /\* منوی 2 \*/  .btn2 .icon {  width: 60px;  top: 30px;  }  .btn2 .icon::before {  top: -20px;  }  .btn2 .icon::after {  top: 20px;  }  .btn2.open .icon {  background: transparent;  }  .btn2.open .icon::before {  transform: rotateZ(45deg) scaleX(1.25) translate(13px, 13px);  }  .btn2.open .icon::after {  transform: rotateZ(-45deg) scaleX(1.25) translate(12px, -12px);  }  /\* منوی 3 \*/  .btn3 .icon {  width: 60px;  top: 30px;  }  .btn3 .icon::before {  top: -20px;  }  .btn3 .icon::after {  top: 20px;  }  .btn3.open .icon::before {  transform: rotateZ(135deg) scaleX(1.25) translate(13px, -13px);  }  .btn3.open .icon::after {  transform: rotateZ(-135deg) scaleX(1.25) translate(12px, 12px);  }  /\* منوی 4 \*/  .btn4 .icon {  width: 60px;  top: 30px;  height: 0px;  }  .btn4 .icon::before {  top: -20px;  }  .btn4 .icon::after {  top: 20px;  }  .btn4.open .icon::before {  transform: rotateZ(45deg) scaleX(1.25) translate(12px, 12px);  }  .btn4.open .icon::after {  transform: rotateZ(-45deg) scaleX(1.25) translate(12px, -12px);  }  /\* منوی 5 \*/  .btn5 .icon {  width: 60px;  top: 30px;  }  .btn5 .icon::before {  top: -21px;  }  .btn5 .icon::after {  top: 20px;  }  .btn5.open .icon::before {  transform: rotateZ(-45deg) scaleX(0.75) translate(-20px, -6px);  }  .btn5.open .icon::after {  transform: rotateZ(45deg) scaleX(0.75) translate(-20px, 6px);  }  /\* منوی 6 \*/  .btn6 .icon {  width: 60px;  top: 30px;  }  .btn6 .icon::before {  top: -20px;  }  .btn6 .icon::after {  top: 20px;  }  .btn6.open .icon {  transform: rotateZ(180deg);  }  .btn6.open .icon::before {  transform: rotateZ(45deg) scaleX(0.75) translate(20px, -6px);  }  .btn6.open .icon::after {  transform: rotateZ(-45deg) scaleX(0.75) translate(20px, 6px);  }  /\* منوی 7 \*/  .btn7 .icon-E {  width: 60px;  top: 30px;  transition-delay: 0.5s;  }  .btn7 .icon-E::before {  top: -20px;  }  .btn7 .icon-E::after {  top: 20px;  }  .btn7 .icon-X {  width: 200px;  top: 30px;  height: 0px;  transform: scaleX(0);  transform-origin: 30px 0px;  }  .btn7 .icon-X::before {  top: -20px;  transform: rotateZ(45deg) scaleX(1.25) translate(12px, 12px);  }  .btn7 .icon-X::after {  top: 20px;  transform: rotateZ(-45deg) scaleX(1.25) translate(12px, -12px);  }  .btn7.open .icon-E {  transform: scaleX(0);  transition-delay: 0s;  }  .btn7.open .icon-X {  transform: scaleX(1);  transition-delay: 0.5s;  }  /\* منوی 8 \*/  .btn8 .icon-E {  width: 60px;  top: 30px;  transform-origin: 0px 0px;  transition-delay: 0.5s;  }  .btn8 .icon-E::before {  top: -20px;  }  .btn8 .icon-E::after {  top: 20px;  }  .btn8 .icon-X {  width: 60px;  top: 30px;  height: 0px;  transform-origin: 0px 0px;  transform: scaleY(0);  }  .btn8 .icon-X::before {  top: -20px;  transform: rotateZ(45deg) scaleX(1.25) translate(12px, 12px);  }  .btn8 .icon-X::after {  top: 20px;  transform: rotateZ(-45deg) scaleX(1.25) translate(12px, -12px);  }  .btn8.open .icon-E {  transform: scaleY(0);  transition-delay: 0s;  }  .btn8.open .icon-X {  transform: scaleY(1);  transition-delay: 0.5s;  }  /\* منوی 9 \*/  .btn9 .icon {  width: 60px;  top: 0;  height: 0px;  transform: rotateZ(90deg);  transform-origin: 30px 30px;  left: -15px;  }  .btn9 .icon::before {  width: 30px;  height: 60px;  top: 0;  }  .btn9 .icon::after {  width: 30px;  height: 60px;  top: 0;  left: 30px;  }  .btn9.open .icon {  transform: rotateZ(0deg);  }  .btn9.open .icon::before {  width: 22.5px;  }  .btn9.open .icon::after {  transform-origin: 30px 60px;  transform: scaleX(0.75);  }  .btn9.play .icon {  transform: rotateZ(0deg);  }  .btn9.play .icon::before {  width: 500px;  }  .btn9.play .icon::after {  transform-origin: 30px 60px;  transform: scaleX(2);  }  /\* منوی 10 \*/  .btn10 .icon {  width: 60px;  top: 30px;  }  .btn10 .icon::before {  top: -20px;  }  .btn10 .icon::after {  top: 20px;  }  .btn10.open .icon {  transform: rotateZ(-180deg);  }  .btn10.open .icon::before {  transform: rotateZ(-45deg) scaleX(0.75) translate(-20px, -6px);  }  .btn10.open .icon::after {  transform: rotateZ(45deg) scaleX(0.75) translate(-20px, 6px);  }  /\* منوی 11 \*/  .btn11 .icon-left {  width: 15px;  height: 4px;  top: 30px;  }  .btn11 .icon-left::before {  width: 15px;  height: 4px;  top: -10px;  }  .btn11 .icon-left::after {  width: 15px;  height: 4px;  top: 10px;  }  .btn11 .icon-right {  width: 15px;  height: 4px;  top: 30px;  left: 15px;  }  .btn11 .icon-right::before {  width: 15px;  height: 4px;  top: -10px;  }  .btn11 .icon-right::after {  width: 15px;  height: 4px;  top: 10px;  }  .btn11.open .icon-left {  background: transparent;  }  .btn11.open .icon-left::before {  transform: rotateZ(45deg) scaleX(1.4) translate(2px, 2px);  }  .btn11.open .icon-left::after {  transform: rotateZ(-45deg) scaleX(1.4) translate(2px, -2px);  }  .btn11.open .icon-right {  background: transparent;  }  .btn11.open .icon-right::before {  transform: rotateZ(-45deg) scaleX(1.4) translate(-2px, 2px);  }  .btn11.open .icon-right::after {  transform: rotateZ(45deg) scaleX(1.4) translate(-2px, -2px);  }  /\* منوی 12 \*/  .btn12 .icon {  width: 30px;  height: 4px;  top: 30px;  }  .btn12 .icon::before {  width: 30px;  height: 4px;  top: -10px;  }  .btn12 .icon::after {  width: 30px;  height: 4px;  top: 10px;  }  .btn12.open .icon {  background: transparent;  }  .btn12.open .icon::before {  transform: rotateZ(45deg) scaleX(1.25) translate(6.5px, 6.5px);  }  .btn12.open .icon::after {  transform: rotateZ(-45deg) scaleX(1.25) translate(6px, -6px);  }  /\* منوی 13 \*/  .btn13 .icon {  width: 30px;  height: 4px;  top: 30px;  }  .btn13 .icon::before {  width: 30px;  height: 4px;  top: -10px;  }  .btn13 .icon::after {  width: 30px;  height: 4px;  top: 10px;  }  .btn13.open .icon::before {  transform: rotateZ(-45deg) scaleX(0.75) translate(-10px, -3px);  }  .btn13.open .icon::after {  transform: rotateZ(45deg) scaleX(0.75) translate(-10px, 3px);  }  /\* منوی 14 \*/  .btn14 .icon-circle {  position: absolute;  height: 60px;  width: 60px;  top: 2px;  border-radius: 100%;  background: transparent;  border: 4px solid black;  left: -15px;  transition-duration: 0.5s;  }  .btn14 .icon-circle:hover {  background: #fff;  }  .btn14 .icon {  width: 30px;  height: 4px;  top: 30px;  border-radius: 4px;  }  .btn14 .icon::before {  width: 30px;  height: 4px;  top: -10px;  border-radius: 4px;  }  .btn14 .icon::after {  width: 30px;  height: 4px;  top: 10px;  border-radius: 4px;  }  .btn14.open .icon-circle {  background: #fff;  }  .btn14.open .icon {  background: transparent;  }  .btn14.open .icon::before {  transform: rotateZ(45deg) scaleX(1.25) translate(6.5px, 6.5px);  }  .btn14.open .icon::after {  transform: rotateZ(-45deg) scaleX(1.25) translate(6px, -6px);  }  /\* منوی 15 \*/  .btn15 .icon {  width: 30px;  height: 4px;  top: 30px;  left: 15px;  border-radius: 4px;  }  .btn15 .icon::before {  width: 30px;  height: 4px;  top: -10px;  border-radius: 4px;  }  .btn15 .icon::after {  width: 30px;  height: 4px;  top: 10px;  border-radius: 4px;  }  .btn15:hover {  background: #fff;  }  .btn15.open .icon {  background: transparent;  }  .btn15.open .icon::before {  transform: rotateZ(45deg) scaleX(1.25) translate(6.5px, 6.5px);  }  .btn15.open .icon::after {  transform: rotateZ(-45deg) scaleX(1.25) translate(6px, -6px);  }  /\* منوهای جدید \*/  .all-btn {  display: flex;  flex-wrap: wrap;  gap: 20px;  justify-content: center;  margin-top: 30px;  }  .menu-btn-1, .menu-btn-2, .menu-btn-3, .menu-btn-4,  .menu-btn-5, .menu-btn-6, .menu-btn-7, .menu-btn-8 {  height: 32px;  width: 40px;  cursor: pointer;  }  .menu-btn-1 span, .menu-btn-1 span::before, .menu-btn-1 span::after,  .menu-btn-2 span, .menu-btn-2 span::before, .menu-btn-2 span::after,  .menu-btn-3 span, .menu-btn-3 span::before, .menu-btn-3 span::after,  .menu-btn-4 span, .menu-btn-4 span::before, .menu-btn-4 span::after,  .menu-btn-6 span, .menu-btn-6 span::before, .menu-btn-6 span::after {  background: #212121;  border-radius: 3px;  content: '';  position: absolute;  width: 40px;  height: 6px;  margin-top: 13px;  transition: .3s ease-in-out;  }  .menu-btn-1 span::before, .menu-btn-2 span::before, .menu-btn-3 span::before,  .menu-btn-4 span::before, .menu-btn-6 span::before {  margin-top: -12px;  }  .menu-btn-1 span::after, .menu-btn-2 span::after, .menu-btn-3 span::after,  .menu-btn-4 span::after, .menu-btn-6 span::after {  margin-top: 12px;  }  .menu-btn-1.active span, .menu-btn-2.active span, .menu-btn-3.active span {  background: transparent;  }  .menu-btn-1.active span::before, .menu-btn-2.active span::before,  .menu-btn-3.active span::before, .menu-btn-6.active span::before {  margin-top: 0;  transform: rotate(45deg);  }  .menu-btn-1.active span::after, .menu-btn-2.active span::after,  .menu-btn-3.active span::after, .menu-btn-6.active span::after {  margin-top: 0;  transform: rotate(-45deg);  }  .menu-btn-2 span, .menu-btn-2 span::before, .menu-btn-2 span::after,  .menu-btn-3 span, .menu-btn-3 span::before, .menu-btn-3 span::after {  transform: rotateY(180deg);  transition: .5s ease-in-out;  }  .menu-btn-2.active {  transform: rotateY(180deg);  transition: .5s ease-in-out;  }  .menu-btn-4 span, .menu-btn-4 span::before, .menu-btn-4 span::after {  height: 3px;  transition: .5s ease-in-out;  }  .menu-btn-4.active span {  border-radius: 1px 0px 0px 1px;  }  .menu-btn-4.active span::before {  right: 24px;  width: 17px;  top: 7.8px;  border-radius: 1px 0px 0px 1px;  transform: rotate(-34deg);  }  .menu-btn-4.active span::after {  right: 24px;  width: 17px;  top: -7.8px;  border-radius: 1px 0px 0px 1px;  transform: rotate(34deg);  }  .menu-btn-5 span:nth-child(1), .menu-btn-5 span:nth-child(1)::before,  .menu-btn-5 span:nth-child(1)::after {  background: #E91E63;  content: '';  position: absolute;  width: 40px;  height: 3px;  margin-top: 13px;  transition: .3s ease-in-out;  }  .menu-btn-5 span:nth-child(1)::before {  margin-top: -12px;  transition: .3s ease-in-out .3s;  }  .menu-btn-5 span:nth-child(1)::after {  margin-top: 12px;  transition: .3s ease-in-out .3s;  }  .menu-btn-5 span:nth-child(2) {  background: #E91E63;  content: '';  position: absolute;  width: 0px;  height: 3px;  margin-top: 13px;  transform: rotate(-45deg);  transition: .3s ease-in-out;  }  .menu-btn-5.active span:nth-child(1)::before,  .menu-btn-5.active span:nth-child(1)::after {  margin-top: 0;  }  .menu-btn-5.active span:nth-child(1) {  transform: rotate(45deg);  transition: .5s ease-in-out .5s;  }  .menu-btn-5.active span:nth-child(2) {  width: 40px;  transition: .3s ease-in-out .8s;  }  .menu-btn-6 span, .menu-btn-6 span::before, .menu-btn-6 span::after {  background: #795548;  height: 3px;  transition: .3s ease-in-out .3s;  }  .menu-btn-6 span::before, .menu-btn-6 span::after {  transition: .3s ease-in-out;  }  .menu-btn-6.active span {  background: transparent;  transform: rotate(-90deg);  transition: .5s ease-in-out;  }  .menu-btn-6.active span::before {  transform: rotate(-45deg);  transition-delay: .5s;  }  .menu-btn-6.active span::after {  transform: rotate(45deg);  transition-delay: .5s;  }  .menu-btn-7, .menu-btn-8 {  height: 40px;  width: 40px;  cursor: pointer;  position: relative;  border-radius: 100%;  transition: .5s ease-in-out;  }  .menu-btn-7 {  border: 4px solid #00bcd4;  }  .menu-btn-8 {  border: 4px solid #9C27B0;  }  .menu-btn-7 span:nth-child(1), .menu-btn-7 span:nth-child(2),  .menu-btn-8 span:nth-child(1), .menu-btn-8 span:nth-child(2) {  content: '';  position: absolute;  width: 20px;  height: 5px;  margin-top: 18px;  }  .menu-btn-7 span:nth-child(1), .menu-btn-7 span:nth-child(2) {  background: #00BCD4;  }  .menu-btn-8 span:nth-child(1), .menu-btn-8 span:nth-child(2) {  background: #9C27B0;  transition: .5s ease-in-out;  }  .menu-btn-7 span:nth-child(1), .menu-btn-8 span:nth-child(1) {  right: 15.1px;  transform: rotate(45deg);  }  .menu-btn-7 span:nth-child(2), .menu-btn-8 span:nth-child(2) {  right: 4.5px;  transform: rotate(-45deg);  }  .menu-btn-7.active {  transform: rotateX(-180deg);  }  .menu-btn-8.active span:nth-child(1), .menu-btn-8.active span:nth-child(2) {  margin-top: 16px;  }  .menu-btn-8.active span:nth-child(1) {  transform: rotate(-45deg);  }  .menu-btn-8.active span:nth-child(2) {  transform: rotate(45deg);  }  </style>  </head>  <body>  <div class="container">  <h1>انواع منوهای متحرک</h1>    <!-- بخش اول: منوهای 1 تا 8 -->  <div class="menu-section">  <h2>منوهای 1 تا 8</h2>  <div class="menu-container">  <div class="menu btn1" data-menu="1">  <div class="icon-left"></div>  <div class="icon-right"></div>  </div>  <div class="menu btn2" data-menu="2">  <div class="icon"></div>  </div>  <div class="menu btn3" data-menu="3">  <div class="icon"></div>  </div>  <div class="menu btn4" data-menu="4">  <div class="icon"></div>  </div>  <div class="menu btn5" data-menu="5">  <div class="icon"></div>  </div>  <div class="menu btn6" data-menu="6">  <div class="icon"></div>  </div>  <div class="menu btn7" data-menu="7">  <div class="icon-E"></div>  <div class="icon-X"></div>  </div>  <div class="menu btn8" data-menu="8">  <div class="icon-E"></div>  <div class="icon-X"></div>  </div>  </div>  </div>    <!-- بخش دوم: منوهای 9 تا 15 -->  <div class="menu-section">  <h2>منوهای 9 تا 15</h2>  <div class="menu-container">  <div class="menu btn9" data-menu="9">  <div class="icon"></div>  </div>  <div class="menu btn10" data-menu="10">  <div class="icon"></div>  </div>  <div class="menu btn11" data-menu="11">  <div class="icon-left"></div>  <div class="icon-right"></div>  </div>  <div class="menu btn12" data-menu="12">  <div class="icon"></div>  </div>  <div class="menu btn13" data-menu="13">  <div class="icon"></div>  </div>  <div class="menu btn14" data-menu="14">  <div class="icon-circle"></div>  <div class="icon"></div>  </div>  <div class="menu btn15" data-menu="15">  <div class="icon"></div>  </div>  </div>  </div>    <!-- بخش سوم: منوهای جدید -->  <div class="menu-section">  <h2>منوهای جدید</h2>  <div class="all-btn">  <div class="menu-btn-1" onclick="menuBtnFunction(this)">  <span></span>  </div>  <div class="menu-btn-2" onclick="menuBtnFunction(this)">  <span></span>  </div>  <div class="menu-btn-3" onclick="menuBtnFunction(this)">  <span></span>  </div>  <div class="menu-btn-4" onclick="menuBtnFunction(this)">  <span></span>  </div>  <div class="menu-btn-5" onclick="menuBtnFunction(this)">  <span></span>  <span></span>  </div>  <div class="menu-btn-6" onclick="menuBtnFunction(this)">  <span></span>  </div>  <div class="menu-btn-7" onclick="menuBtnFunction(this)">  <span></span>  <span></span>  </div>  <div class="menu-btn-8" onclick="menuBtnFunction(this)">  <span></span>  <span></span>  </div>  </div>  </div>  </div>  <script>  // تابع برای فعال‌سازی منوها  function menuBtnFunction(element) {  element.classList.toggle("active");  }  // اضافه کردن event listener برای منوهای اصلی  document.querySelectorAll('.menu').forEach(menu => {  menu.addEventListener('click', function() {  this.classList.toggle('open');  });  });  </script>  </body>  </html> |

[hamburger](webdesign/css/menu/hamburger.html)

## ساخت منو دیجی کالا

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>digimenu - منوی دسته‌بندی</title>  <style>  \* {  box-sizing: border-box;  margin: 0;  padding: 0;  }  body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  line-height: 1.6;  background-color: #f5f5f5;  }  header {  width: 100%;  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  background-color: #fff;  position: relative;  z-index: 1000;  }  nav ul {  list-style: none;  align-items: center;  }  .main-list {  float: right;  position: relative;  }  .list-link {  display: block;  padding: 1rem 1.5rem;  text-decoration: none;  color: #333;  font-weight: 500;  transition: all 0.3s ease;  border-bottom: 3px solid transparent;  }  .list-link:hover {  background-color: #f8f9fa;  border-bottom: 3px solid #007bff;  }  .submenu {  position: absolute;  top: 100%;  right: 0;  width: 900px;  background-color: #fff;  border: 1px solid #e0e0e0;  border-radius: 4px;  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);  visibility: hidden;  opacity: 0;  z-index: 1001;  padding: 1.5rem;  max-height: 70vh;  overflow: hidden;  transition: all 0.3s ease;  display: flex;  }  .base-submenu {  overflow-y: auto;  width: 30%;  border-left: 1px solid #e0e0e0;  padding-right: 1rem;  height: 100%;  max-height: calc(70vh - 3rem);  }  .base-menu-list {  margin-bottom: 0.5rem;  }  .base-menu-link {  display: block;  padding: 0.75rem 1rem;  text-decoration: none;  color: #555;  border-radius: 4px;  transition: all 0.2s ease;  }  .base-menu-link:hover, .base-menu-list.active .base-menu-link {  background-color: #f0f8ff;  color: #007bff;  }  .second-submenu {  width: 70%;  padding-right: 1.5rem;  height: 100%;  max-height: calc(70vh - 3rem);  overflow-y: auto;  }  .category-content {  display: none;  }  .category-content.active {  display: block;  }  .second-submenu h1 {  font-size: 1.2rem;  margin-bottom: 1rem;  padding-bottom: 0.5rem;  border-bottom: 1px solid #e0e0e0;  }  .second-submenu h1 a {  text-decoration: none;  color: #333;  }  .row::after {  display: table;  content: "";  clear: both;  float: none;  }  .w-1\/4 {  width: 25%;  padding: 0 0.75rem;  float: right;  }  .second-submenu li {  margin-bottom: 0.75rem;  }  .second-submenu li a {  text-decoration: none;  color: #333;  font-weight: 500;  font-size: 0.8rem;  }  .gray li a {  color: #666;  font-weight: normal;  font-size: 0.9rem;  }  .gray li {  margin-bottom: 0.4rem;  }  .main-list:hover .submenu {  visibility: visible;  opacity: 1;  }  .float-right {  float: right;  }  .w-full {  width: 100%;  }  /\* استایل برای موبایل \*/  @media (max-width: 768px) {  .submenu {  position: static;  width: 100%;  display: none;  flex-direction: column;  }  .main-list:hover .submenu {  display: flex;  }  .base-submenu,  .second-submenu {  width: 100%;  border: none;  padding: 0;  }  .row {  flex-direction: column;  }  .w-1\/4 {  width: 100%;  margin-bottom: 1.5rem;  }  }  </style>  </head>  <body>  <header>  <nav>  <ul>  <li class="main-list">  <a href="#" class="list-link">  دسته بندی  </a>  <div class="submenu">  <div class="base-submenu">  <ul>  <li class="base-menu-list active" data-category="mobile">  <a href="#" class="base-menu-link">  موبایل  </a>  </li>  <li class="base-menu-list" data-category="stationery">  <a href="#" class="base-menu-link">  لوازم تحریر  </a>  </li>  <li class="base-menu-list" data-category="laptop">  <a href="#" class="base-menu-link">  لپ تاپ  </a>  </li>  <li class="base-menu-list" data-category="home-appliances">  <a href="#" class="base-menu-link">  کالای خانه و لوازم خانگی  </a>  </li>  <li class="base-menu-list" data-category="cosmetics">  <a href="#" class="base-menu-link">  آرایشی  </a>  </li>  <li class="base-menu-list" data-category="fashion">  <a href="#" class="base-menu-link">  مد و پوشاک  </a>  </li>  <li class="base-menu-list" data-category="automotive">  <a href="#" class="base-menu-link">  خودرو  </a>  </li>  <li class="base-menu-list" data-category="tools">  <a href="#" class="base-menu-link">  ابزارآلات  </a>  </li>  <li class="base-menu-list" data-category="gold">  <a href="#" class="base-menu-link">  طلا و نقره  </a>  </li>  <li class="base-menu-list" data-category="equipment">  <a href="#" class="base-menu-link">  تجهیزات  </a>  </li>  <li class="base-menu-list" data-category="books">  <a href="#" class="base-menu-link">  کتاب و هنر  </a>  </li>  <li class="base-menu-list" data-category="sports">  <a href="#" class="base-menu-link">  ورزش و سفر  </a>  </li>  <li class="base-menu-list" data-category="gift-cards">  <a href="#" class="base-menu-link">  کارت هدیه  </a>  </li>  <li class="base-menu-list" data-category="supermarket">  <a href="#" class="base-menu-link">  سوپر مارکت  </a>  </li>  <li class="base-menu-list" data-category="toys">  <a href="#" class="base-menu-link">  اسباب بازی  </a>  </li>  <li class="base-menu-list" data-category="local-products">  <a href="#" class="base-menu-link">  محصولات بومی  </a>  </li>  <li class="base-menu-list" data-category="pet-shop">  <a href="#" class="base-menu-link">  پت شاپ  </a>  </li>  </ul>  </div>  <div class="second-submenu">  <!-- محتوای موبایل -->  <div id="mobile" class="category-content active">  <h1>  <a href="#">  همه محصولات موبایل  </a>  </h1>  <div class="row">  <ul class="float-right w-1/4">  <li>  <a href="#">گوشی‌های هوشمند</a>  <ul class="gray">  <li><a href="#">سامسونگ</a></li>  <li><a href="#">اپل</a></li>  <li><a href="#">شیائومی</a></li>  <li><a href="#">هوآوی</a></li>  <li><a href="#">نوکیا</a></li>  </ul>  </li>  <li>  <a href="#">اپل (آیفون)</a>  <ul class="gray">  <li><a href="#">آیفون 15 سری</a></li>  <li><a href="#">آیفون 14 سری</a></li>  <li><a href="#">آیفون 13 سری</a></li>  <li><a href="#">آیفون 12 سری</a></li>  <li><a href="#">آیفون 11 سری</a></li>  <li><a href="#">آیفون SE</a></li>  <li><a href="#">آیفون X سری</a></li>  <li><a href="#">آیفون 8 سری</a></li>  <li><a href="#">آیفون 7 سری</a></li>  </ul>  </li>  <li>  <a href="#">گوشی‌های معمولی</a>  <ul class="gray">  <li><a href="#">گوشی ساده</a></li>  <li><a href="#">گوشی کلیددار</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">لوازم جانبی موبایل</a>  <ul class="gray">  <li><a href="#">کیف و کاور</a></li>  <li><a href="#">پاوربانک</a></li>  <li><a href="#">هدفون و هندزفری</a></li>  <li><a href="#">پایه نگهدارنده</a></li>  </ul>  </li>  <li>  <a href="#">شارژر و کابل</a>  <ul class="gray">  <li><a href="#">شارژر دیواری</a></li>  <li><a href="#">شارژر خودرو</a></li>  <li><a href="#">کابل USB</a></li>  </ul>  </li>    <li>  <a href="#">گوشی براساس قیمت</a>  <ul class="gray">  <li><a href="#">اقتصادی (زیر ۵ میلیون تومان)</a></li>  <li><a href="#">میان‌رده (۵ تا ۱۰ میلیون تومان)</a></li>  <li><a href="#">بالارده (۱۰ تا ۲۰ میلیون تومان)</a></li>  <li><a href="#">فلاگشیپ (۲۰ تا ۳۰ میلیون تومان)</a></li>  <li><a href="#">لاکچری (بالای ۳۰ میلیون تومان)</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">محافظ صفحه</a>  <ul class="gray">  <li><a href="#">محافظ شیشه‌ای</a></li>  <li><a href="#">محافظ نانو</a></li>  <li><a href="#">محافظ مات</a></li>  </ul>  </li>  <li>  <a href="#">هندزفری</a>  <ul class="gray">  <li><a href="#">هندزفری سیمی</a></li>  <li><a href="#">هندزفری بلوتوث</a></li>  <li><a href="#">هندزفری ورزشی</a></li>  </ul>  </li>    <li>  <a href="#">گوشی براساس رزولوشن تصویر</a>  <ul class="gray">  <li><a href="#">HD (720p)</a></li>  <li><a href="#">Full HD (1080p)</a></li>  <li><a href="#">Quad HD (1440p)</a></li>  <li><a href="#">4K Ultra HD (2160p)</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">سیم کارت</a>  <ul class="gray">  <li><a href="#">سیم کارت دائمی</a></li>  <li><a href="#">سیم کارت اعتباری</a></li>  <li><a href="#">سیم کارت اینترنتی</a></li>  </ul>  </li>  <li>  <a href="#">حافظه جانبی</a>  <ul class="gray">  <li><a href="#">کارت حافظه</a></li>  <li><a href="#">فلش USB</a></li>  <li><a href="#">هارد اکسترنال</a></li>  </ul>  </li>    <li>  <a href="#">گوشی براساس رزولوشن تصویر</a>  <ul class="gray">  <li><a href="#">HD (720p)</a></li>  <li><a href="#">Full HD (1080p)</a></li>  <li><a href="#">Quad HD (1440p)</a></li>  <li><a href="#">4K Ultra HD (2160p)</a></li>  </ul>  </li>  </ul>  </div>  </div>    <!-- محتوای لوازم تحریر -->  <div id="stationery" class="category-content">  <h1>  <a href="#">  همه محصولات لوازم تحریر  </a>  </h1>  <div class="row">  <ul class="float-right w-1/4">  <li>  <a href="#">مداد و خودکار</a>  <ul class="gray">  <li><a href="#">مداد سیاه</a></li>  <li><a href="#">مداد رنگی</a></li>  <li><a href="#">خودکار</a></li>  <li><a href="#">خودکار رنگی</a></li>  </ul>  </li>  <li>  <a href="#">دفتر و کاغذ</a>  <ul class="gray">  <li><a href="#">دفتر مشق</a></li>  <li><a href="#">دفتر یادداشت</a></li>  <li><a href="#">کاغذ A4</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">تراش و پاک‌کن</a>  <ul class="gray">  <li><a href="#">تراش</a></li>  <li><a href="#">پاک‌کن</a></li>  <li><a href="#">مدادتراش رومیزی</a></li>  </ul>  </li>  <li>  <a href="#">کیف و جامدادی</a>  <ul class="gray">  <li><a href="#">جامدادی</a></li>  <li><a href="#">کیف مدرسه</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">رنگ و نقاشی</a>  <ul class="gray">  <li><a href="#">مداد شمعی</a></li>  <li><a href="#">گواش</a></li>  <li><a href="#">آبرنگ</a></li>  </ul>  </li>  <li>  <a href="#">خط کش و گونیا</a>  <ul class="gray">  <li><a href="#">خط کش</a></li>  <li><a href="#">گونیا</a></li>  <li><a href="#">نقاله</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">منگنه و سوزن</a>  <ul class="gray">  <li><a href="#">منگنه</a></li>  <li><a href="#">سوزن منگنه</a></li>  <li><a href="#">پونز</a></li>  </ul>  </li>  <li>  <a href="#">چسب و قیچی</a>  <ul class="gray">  <li><a href="#">چسب ماتیکی</a></li>  <li><a href="#">چسب مایع</a></li>  <li><a href="#">قیچی</a></li>  </ul>  </li>  </ul>  </div>  </div>    <!-- محتوای لپ تاپ -->  <div id="laptop" class="category-content">  <h1>  <a href="#">  همه محصولات لپ تاپ  </a>  </h1>  <div class="row">  <ul class="float-right w-1/4">  <li>  <a href="#">لپ تاپ‌های تجاری</a>  <ul class="gray">  <li><a href="#">لنوو</a></li>  <li><a href="#">دل</a></li>  <li><a href="#">اچ پی</a></li>  </ul>  </li>  <li>  <a href="#">لپ تاپ‌های گیمینگ</a>  <ul class="gray">  <li><a href="#">ایسوس</a></li>  <li><a href="#">ام اس آی</a></li>  <li><a href="#">ایسر</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">لپ تاپ‌های سبک</a>  <ul class="gray">  <li><a href="#">مک‌بوک</a></li>  <li><a href="#">مایکروسافت</a></li>  <li><a href="#">سرفیس</a></li>  </ul>  </li>  <li>  <a href="#">لپ تاپ‌های ارزان</a>  <ul class="gray">  <li><a href="#">لپ تاپ دانشجویی</a></li>  <li><a href="#">لپ تاپ کارکرده</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">کیف لپ تاپ</a>  <ul class="gray">  <li><a href="#">کیف حمل</a></li>  <li><a href="#">کوله پشتی</a></li>  </ul>  </li>  <li>  <a href="#">کول پد</a>  <ul class="gray">  <li><a href="#">کول پد فعال</a></li>  <li><a href="#">کول پد غیرفعال</a></li>  </ul>  </li>  </ul>  <ul class="float-right w-1/4">  <li>  <a href="#">ماوس و کیبورد</a>  <ul class="gray">  <li><a href="#">ماوس بی‌سیم</a></li>  <li><a href="#">ماوس گیمینگ</a></li>  <li><a href="#">کیبورد مکانیکی</a></li>  </ul>  </li>  <li>  <a href="#">داک استیشن</a>  <ul class="gray">  <li><a href="#">داک USB-C</a></li>  <li><a href="#">داک تاندربولت</a></li>  </ul>  </li>  </ul>  </div>  </div>    <!-- محتوای سایر دسته‌بندی‌ها -->  <div id="home-appliances" class="category-content">  <h1>  <a href="#">  همه محصولات لوازم خانگی  </a>  </h1>  <div class="row">  <!-- محتوای لوازم خانگی -->  </div>  </div>    <!-- محتوای سایر دسته‌بندی‌ها به همین ترتیب -->    </div>  </div>  </li>  <li class="main-list">  <a href="#" class="list-link">  شگفت انگیز ها  </a>  </li>  <li class="main-list">  <a href="#" class="list-link">  سوپر مارکت  </a>  </li>  <li class="main-list">  <a href="#" class="list-link">  طلای دیجیتال  </a>  </li>  <li class="main-list">  <a href="#" class="list-link">  پرفروشترین ها  </a>  </li>  <li class="main-list">  <a href="#" class="list-link">  سوالی دارید  </a>  </li>  <li class="main-list">  <a href="#" class="list-link">  فروشنده شوید  </a>  </li>  </ul>  </nav>  </header>  <script>  document.addEventListener('DOMContentLoaded', function() {  const categoryItems = document.querySelectorAll('.base-menu-list');  categoryItems.forEach(item => {  item.addEventListener('mouseenter', function() {  // حذف کلاس active از همه آیتم‌ها  categoryItems.forEach(i => i.classList.remove('active'));    // اضافه کردن کلاس active به آیتم فعلی  this.classList.add('active');    // دریافت شناسه دسته‌بندی  const categoryId = this.getAttribute('data-category');    // مخفی کردن همه محتواها  const allContents = document.querySelectorAll('.category-content');  allContents.forEach(content => content.classList.remove('active'));    // نمایش محتوای مربوطه  const targetContent = document.getElementById(categoryId);  if (targetContent) {  targetContent.classList.add('active');  }  });  });  });  </script>  </body>  </html> |

## Transform

Translate()

این تابع، عنصر را در صفحه جابجا می‌کند. مانند این است که شما یک قطعه کاغذ را روی میز بلند کنید و در جای دیگری قرار دهید.

- کاربرد: حرکت دادن عنصر در جهت‌های مختلف.

- پارامترها:

- `translateX(move)`: عنصر را به اندازه `move` پیکسل (یا هر واحد دیگر) به چپ یا راست حرکت می‌دهد. مقدار مثبت به راست، منفی به چپ.

- `translateY(move)`: عنصر را به اندازه `move` پیکسل به بالا یا پایین حرکت می‌دهد. مقدار مثبت به پایین، منفی به بالا.

- `translate(x, y)`: هر دو حرکت را همزمان انجام می‌دهد.

مثال عملی:

یک دکمه را در نظر بگیرید. با ماوس‌هاور (hover) آن را کمی به پایین و راست حرکت می‌دهیم.

|  |
| --- |
| <style>  .button {  display: block;  width: 250px;  background-color: #034e03;  text-align: center;  color: white;  padding: 1rem;  border-radius: 20px;  text-decoration: none;  transition: transform 0.3s;  transition: all 0.2s ease;  }  .button:hover {  transform: translate(10px, 20px);  /\* یا به صورت جداگانه:  transform: translateX(10px) translateY(20px);  \*/  }  </style>  <a href="#" class="button">  دکمه  </a> |

[translate](webdesign/css/Transform/Translate.html)

نتیجه: وقتی نشانگر ماوس را روی دکمه ببرید، 10 پیکسل به راست و 20 پیکسل به پایین می‌رود.

Rotate()

این تابع، عنصر را به دور یک نقطه مرکزی (به طور پیش‌فرض مرکز خودش) می‌چرخاند.

- کاربرد: ایجاد افکت‌های چرخشی، مانند آیکون‌های در حال چرخش یا کارت‌های معکوس.

- پارامتر: یک مقدار زاویه می‌گیرد. واحدهای رایج:

- deg (درجه): مثلاً `45deg`

- turn (دور): مثلاً `0.25turn` (معادل 90 درجه)

- rad (رادیان)

مثال عملی:

چرخش 45 درجه‌ای یک عکس هنگام هاور.

|  |
| --- |
| .photo {  transition: transform 0.5s;  }  .photo:hover {  transform: rotate(45deg);  } |

[Rotate](webdesign/css/Transform/Rotate.html)

نتیجه: عکس به اندازه 45 درجه در جهت عقربه‌های ساعت می‌چرخد. برای چرخش برعکس از مقادیر منفی استفاده کنید (مثلاً `-45deg`).

Skew()

این تابع، عنصر را به صورت مورب و کج تغییر شکل می‌دهد. مانند این است که به گوشه‌های یک عکس کشش وارد کنید.

- کاربرد: ایجاد جلوه‌های پرسپکتیو ساده، افکت‌های tilt (کج شدن) و طراحی‌های خلاقانه.

- پارامترها:

- `skewX(angle)`: عنصر را در راستای محور X کج می‌کند. (گوشه‌های بالا و پایین ثابت می‌مانند و کناره‌ها حرکت می‌کنند).

- `skewY(angle)`: عنصر را در راستای محور Y کج می‌کند. (گوشه‌های چپ و راست ثابت می‌مانند و بالا و پایین حرکت می‌کنند).

- `skew(x-angle, y-angle)`: هر دو را همزمان اعمال می‌کند.

مثال عملی:

کج کردن یک کادر متن هنگام هاور.

|  |
| --- |
| .text-box {  transition: transform 0.3s;  }  .text-box:hover {  transform: skewX(15deg);  /\* یا:  transform: skew(15deg, 5deg);  \*/  } |

[skew](webdesign/css/Transform/skew.html)

نتیجه: کادر متن به اندازه 15 درجه در جهت محور X کشیده و کج می‌شود.

Scale()

توضیح:

تابع scale() برای تغییر اندازه عناصر استفاده میشه. مثل این میمونه که یه عکس رو زوم می‌کنی یا ازش دور می‌شی.

- scale(2) = عنصر 2 برابر بزرگ میشه

- scale(0.5) = عنصر نصف اندازه اصلی میشه

- scale(1) = اندازه اصلی (بدون تغییر)

کد تمرین:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .box {  width: 100px;  height: 100px;  background: #e74c3c;  margin: 30px;  display: inline-block;  transition: transform 0.3s;  }  .big:hover {  transform: scale(2);  }  .small:hover {  transform: scale(0.5);  }  </style>  </head>  <body>  <div class="box big">هاور کن - بزرگ میشه</div>  <div class="box small">هاور کن - کوچک میشه</div>  </body>  </html> |

[scale](webdesign/css/Transform/scale.html)

Transform-origin

توضیح:

transform-origin نقطه شروع تغییرات رو مشخص می‌کنه. یعنی عنصر از کدوم نقطه شروع به بزرگ شدن یا چرخش کنه.

مقادیر مهم:

- center = از مرکز

- top left = از گوشه بالا چپ

- bottom right = از گوشه پایین راست

کد تمرین:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .box {  width: 100px;  height: 100px;  background: #2ecc71;  margin: 30px;  display: inline-block;  transition: transform 0.3s;  }  .from-center:hover {  transform: scale(2);  transform-origin: center;  }  .from-corner:hover {  transform: scale(2);  transform-origin: top left;  }  </style>  </head>  <body>  <div class="box from-center">هاور کن - از مرکز بزرگ میشه</div>  <div class="box from-corner">هاور کن - از گوشه بزرگ میشه</div>  </body>  </html> |

[origin](webdesign/css/Transform/origin.html)

### Perspective

ویژگی perspective برای ایجاد عمق و جلوه سه‌بعدی در عناصر استفاده می‌شود. این ویژگی مشخص می‌کند که بیننده چقدر از عنصر فاصله دارد و عمق صحنه را کنترل می‌کند.

- مقادیر:

- `perspective(500px)` = فاصله دید ۵۰۰ پیکسلی

- `perspective(1000px)` = فاصله دید ۱۰۰۰ پیکسلی

- هرچه مقدار کمتر باشد: اثر سه‌بعدی قوی‌تر است

- هرچه مقدار بیشتر باشد: اثر سه‌بعدی ملایم‌تر است

|  |
| --- |
| <style>  .container {  perspective: 800px;  margin: 50px;  }    .box {  width: 100px;  height: 100px;  background: #9b59b6;  margin: 30px;  display: inline-block;  transition: transform 0.5s;  }  .rotate-3d:hover {  transform: rotateY(45deg);  }    .deep-perspective:hover {  transform: rotateX(60deg);  }  </style>  <div class="container">  <div class="box rotate-3d">هاور کن - چرخش سه‌بعدی</div>  <div class="box deep-perspective">هاور کن - عمق بیشتر</div>  </div> |

[perspective](webdesign/css/Transform/Perspective.html)

در این مثال:

- مربع اول: با چرخش حول محور Y اثر سه‌بعدی ایجاد می‌کند

- مربع دوم: با چرخش حول محور X عمق بیشتری نشان می‌دهد

## Perspective-origin

ویژگی `perspective-origin` نقطه نگاه بیننده را مشخص می‌کند. یعنی بیننده از کدام زاویه به صحنه سه‌بعدی نگاه می‌کند.

- مقادیر پیش‌فرض: `50% 50%` (مرکز)

- مقادیر ممکن:

- left top = نگاه از بالا-چپ

- right bottom = نگاه از پایین-راست

- 20% 80% = نگاه از ۲۰% چپ و ۸۰% بالا

|  |
| --- |
| <style>  .scene {  perspective: 1000px;  perspective-origin: center center;  margin: 50px;  }    .box {  width: 100px;  height: 100px;  background: #e67e22;  margin: 30px;  display: inline-block;  transition: transform 0.5s;  }  .view-top:hover {  transform: rotateX(45deg);  }    .view-left:hover {  transform: rotateY(45deg);  }    /\* تغییر نقطه نگاه \*/  .scene-different {  perspective: 1000px;  perspective-origin: left top;  margin: 50px;  }  </style>  <div class="scene">  <div class="box view-top">هاور کن - نگاه از مرکز</div>  <div class="box view-left">هاور کن - چرخش از مرکز</div>  </div>  <div class="scene-different">  <div class="box view-top">هاور کن - نگاه از بالا-چپ</div>  <div class="box view-left">هاور کن - چرخش از بالا-چپ</div>  </div> |

[Origin2](webdesign/css/Transform/orgin2.html)

در این مثال:

- دو مربع اول: از نقطه دید مرکزی مشاهده می‌شوند

- دو مربع دوم: از نقطه دید بالا-چپ مشاهده می‌شوند

## [Filter](webdesign/css/Filter)

Blur()

تابع blur()برای محو کردن یا ایجاد افکت تاری در عناصر استفاده می‌شود. این تابع بخشی از فیلترهای CSS است.

مقادیر:

- blur(0px) = بدون تاری

- blur(2px) = تاری کم

- blur(5px) = تاری متوسط

- blur(10px) = تاری شدید

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e74c3c;  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .blur-light:hover {  filter: blur(2px);  }  .blur-medium:hover {  filter: blur(5px);  }  .blur-strong:hover {  filter: blur(10px);  }  </style>  <div class="box blur-light">هاور کن - تاری کم</div>  <div class="box blur-medium">هاور کن - تاری متوسط</div>  <div class="box blur-strong">هاور کن - تاری شدید</div> |

[blur](webdesign/css/Filter/blur.html)

Brightness()

تابع `brightness()` برای تنظیم روشنایی عناصر استفاده می‌شود. این تابع می‌تواند عناصر را تاریک‌تر یا روشن‌تر کند.

- مقادیر:

- `brightness(0.5)` = ۵۰% روشنایی (تاریک)

- `brightness(1)` = ۱۰۰% روشنایی (نرمال)

- `brightness(1.5)` = ۱۵۰% روشنایی (روشن)

- `brightness(2)` = ۲۰۰% روشنایی (بسیار روشن)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e74c3c;  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .dark:hover {  filter: brightness(0.5);  }  .normal:hover {  filter: brightness(1);  }  .bright:hover {  filter: brightness(1.5);  }  .very-bright:hover {  filter: brightness(2);  }  </style>  <div class="box dark">هاور کن - تاریک</div>  <div class="box normal">هاور کن - نرمال</div>  <div class="box bright">هاور کن - روشن</div>  <div class="box very-bright">هاور کن - بسیار روشن</div> |

[brightness](webdesign/css/Filter/brightness.html)

Contrast()

تابع `contrast()` برای تنظیم کنتراست (تضاد) عناصر استفاده می‌شود. کنتراست تفاوت بین تاریک‌ترین و روشن‌ترین نقاط یک تصویر را کنترل می‌کند.

- مقادیر:

- `contrast(0.5)` = ۵۰% کنتراست (کم)

- `contrast(1)` = ۱۰۰% کنتراست (نرمال)

- `contrast(1.5)` = ۱۵۰% کنتراست (زیاد)

- `contrast(2)` = ۲۰۰% کنتراست (بسیار زیاد)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #000, #fff);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .low-contrast:hover {  filter: contrast(0.5);  }  .normal-contrast:hover {  filter: contrast(1);  }  .high-contrast:hover {  filter: contrast(1.5);  }  .very-high-contrast:hover {  filter: contrast(2);  }  </style>  <div class="box low-contrast">هاور کن - کنتراست کم</div>  <div class="box normal-contrast">هاور کن - کنتراست نرمال</div>  <div class="box high-contrast">هاور کن - کنتراست زیاد</div>  <div class="box very-high-contrast">هاور کن - کنتراست بسیار زیاد</div> |

[contrast](webdesign/css/Filter/contrast.html)

Grayscale()

تابع grayscale برای تبدیل عناصر به مقیاس خاکستری استفاده می‌شود. این تابع تصاویر رنگی را به سیاه و سفید تبدیل می‌کند.

- مقادیر:

- grayscale(0) = بدون تغییر (رنگی)

<=grayscale(0.5) ۵۰% خاکستری

<=grayscale(1) ۱۰۰% خاکستری (کاملاً سیاه و سفید)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #e74c3c, #3498db, #f1c40f);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .gray-50:hover {  filter: grayscale(0.5);  }  .gray-100:hover {  filter: grayscale(1);  }  </style>  <div class="box gray-50">هاور کن - ۵۰% خاکستری</div>  <div class="box gray-100">هاور کن - ۱۰۰% خاکستری</div> |

[grayscale](webdesign/css/Filter/grayscale.html)

hue-rotate()

تابع hue-rotate برای چرخش رنگ‌های عنصر استفاده می‌شود. این تابع تمام رنگ‌ها را در چرخه رنگ به اندازه مشخصی می‌چرخاند.

- مقادیر:

- hue-rotate(0deg) = بدون تغییر

- hue-rotate(90deg) = چرخش ۹۰ درجه‌ای

- hue-rotate(180deg) = چرخش ۱۸۰ درجه‌ای (مکمل)

- hue-rotate(360deg) = چرخش کامل

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #e74c3c, #3498db, #f1c40f);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .rotate-90:hover {  filter: hue-rotate(90deg);  }  .rotate-180:hover {  filter: hue-rotate(180deg);  }  .rotate-360:hover {  filter: hue-rotate(360deg);  }  </style>  <div class="box rotate-90">هاور کن - چرخش ۹۰ درجه</div>  <div class="box rotate-180">هاور کن - چرخش ۱۸۰ درجه</div>  <div class="box rotate-360">هاور کن - چرخش کامل</div> |

[hue-rotate](webdesign/css/Filter/hue.html)

invert()

تابع invert برای معکوس کردن رنگ‌های عنصر استفاده می‌شود. این تابع رنگ‌ها را به رنگ مکمل خود تبدیل می‌کند.

- مقادیر:

- invert(0)= بدون تغییر

<= invert(0.5) ۵۰% معکوس

<= invert(1) ۱۰۰% معکوس (نگاتیو کامل)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #e74c3c, #3498db);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .invert-50:hover {  filter: invert(0.5);  }  .invert-100:hover {  filter: invert(1);  }  </style>  <div class="box invert-50">هاور کن - ۵۰% معکوس</div>  <div class="box invert-100">هاور کن - ۱۰۰% معکوس</div> |

[invert](webdesign/css/Filter/invert.html)

saturate()

تابع saturate برای تنظیم شدت رنگ (اشباع رنگ) عناصر استفاده می‌شود.

- مقادیر:

- `saturate(0)` = بدون رنگ (سیاه و سفید)

= saturate(0.5)` ۵۰% اشباع رنگ

= saturate(1)` ۱۰۰% اشباع رنگ (نرمال)

= saturate(2)` ۲۰۰% اشباع رنگ (بسیار پررنگ)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #e74c3c, #3498db);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .desaturate:hover {  filter: saturate(0.5);  }  .normal-saturate:hover {  filter: saturate(1);  }  .super-saturate:hover {  filter: saturate(2);  }  </style>  <div class="box desaturate">هاور کن - کم رنگی</div>  <div class="box normal-saturate">هاور کن - نرمال</div>  <div class="box super-saturate">هاور کن - پررنگ</div> |

[saturate](webdesign/css/Filter/saturate.html)

sepia()

تابع sepia برای ایجاد افکت سِپیا (عکس قدیمی قهوه‌ای رنگ) استفاده می‌شود.

- مقادیر:

- `sepia(0)` = بدون تغییر

= sepia(0.5)` ۵۰% افکت سپیا

= sepia(1)` ۱۰۰% افکت سپیا (کاملاً قدیمی)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: linear-gradient(45deg, #e74c3c, #3498db, #f1c40f);  margin: 30px;  display: inline-block;  transition: filter 0.3s;  }  .sepia-50:hover {  filter: sepia(0.5);  }  .sepia-100:hover {  filter: sepia(1);  }  </style>  <div class="box sepia-50">هاور کن - ۵۰% سپیا</div>  <div class="box sepia-100">هاور کن - ۱۰۰% سپیا</div> |

[sepia](webdesign/css/Filter/sepia.html)

## [animation](webdesign/css/animation)

معرفی animation ها و نحوه استفاده

ویژگی `animation` در CSS به شما امکان می‌دهد انیمیشن‌های پیچیده و کنترل شده ایجاد کنید. این ویژگی ترکیبی از چند زیرویژگی است.

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e74c3c;  margin: 30px;  display: inline-block;  }  .basic-animation {  animation: move 2s infinite;  }  </style>  <div class="box basic-animation">انیمیشن پایه</div> |

[animation](webdesign/css/animation/index.html)

@keyframes

قاعده `@keyframes` برای تعریف مراحل و سبک‌های یک انیمیشن استفاده می‌شود. این قاعده مشخص می‌کند که انیمیشن در هر مرحله چه ویژگی‌هایی باید داشته باشد.

- ساختار: `from` / `to` یا درصدهای مختلف (`0%`, `50%`, `100%`)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #3498db;  margin: 30px;  display: inline-block;  }  @keyframes move {  from { transform: translateX(0); }  to { transform: translateX(200px); }  }  @keyframes color-change {  0% { background: #3498db; }  50% { background: #e74c3c; }  100% { background: #2ecc71; }  }  .move-animation {  animation: move 3s infinite;  }  .color-animation {  animation: color-change 4s infinite;  }  </style>  <div class="box move-animation">حرکت افقی</div>  <div class="box color-animation">تغییر رنگ</div> |

[keyframes](webdesign/css/animation/keyframes.html)

animation-name

این ویژگی نام انیمیشنی که با `keyframes` تعریف شده را مشخص می‌کند.

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #9b59b6;  margin: 30px;  display: inline-block;  }  @keyframes bounce {  0%, 100% { transform: translateY(0); }  50% { transform: translateY(-50px); }  }  @keyframes rotate {  from { transform: rotate(0deg); }  to { transform: rotate(360deg); }  }  .bounce-box {  animation-name: bounce;  animation-duration: 1s;  animation-iteration-count: infinite;  }  .rotate-box {  animation-name: rotate;  animation-duration: 2s;  animation-iteration-count: infinite;  }  </style>  <div class="box bounce-box">پرش</div>  <div class="box rotate-box">چرخش</div> |

[animation-name](webdesign/css/animation/animation-name.html)

animation-duration

این ویژگی مدت زمان اجرای یک سیکل انیمیشن را مشخص می‌کند.

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e67e22;  margin: 30px;  display: inline-block;  }  @keyframes slide {  from { transform: translateX(0); }  to { transform: translateX(100px); }  }  .fast {  animation-name: slide;  animation-duration: 0.5s;  animation-iteration-count: infinite;  }  .slow {  animation-name: slide;  animation-duration: 3s;  animation-iteration-count: infinite;  }  </style>  <div class="box fast">سریع (0.5s)</div>  <div class="box slow">آهسته (3s)</div> |

[animation-duration](webdesign/css/animation/animation-duration.html)

animation-timing-functioning

این ویژگی منحنی سرعت انیمیشن را کنترل می‌کند.

- مقادیر رایج:

- `ease` = شروع و پایان آهسته (پیش‌فرض)

- `linear` = سرعت ثابت

- `ease-in` = شروع آهسته

- `ease-out` = پایان آهسته

- `ease-in-out` = شروع و پایان آهسته

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #2ecc71;  margin: 30px;  display: inline-block;  }  @keyframes move-right {  from { transform: translateX(0); }  to { transform: translateX(200px); }  }  .ease { animation: move-right 3s ease infinite; }  .linear { animation: move-right 3s linear infinite; }  .ease-in { animation: move-right 3s ease-in infinite; }  .ease-out { animation: move-right 3s ease-out infinite; }  </style>  <div class="box ease">ease</div>  <div class="box linear">linear</div>  <div class="box ease-in">ease-in</div>  <div class="box ease-out">ease-out</div> |

[timing](webdesign/css/animation/timing.html)

animation-delay

این ویژگی تأخیر قبل از شروع انیمیشن را مشخص می‌کند.

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e74c3c;  margin: 30px;  display: inline-block;  }  @keyframes fade-in {  from { opacity: 0; }  to { opacity: 1; }  }  .no-delay {  animation: fade-in 2s infinite;  }  .delay-1s {  animation: fade-in 2s 1s infinite;  }  .delay-2s {  animation: fade-in 2s 2s infinite;  }  </style>  <div class="box no-delay">بدون تأخیر</div>  <div class="box delay-1s">تأخیر 1s</div>  <div class="box delay-2s">تأخیر 2s</div> |

[animation-delay](webdesign/css/animation/delay.html)

animation-iteration-count

این ویژگی تعداد دفعات تکرار انیمیشن را مشخص می‌کند.

- مقادیر:

- `1` = یک بار اجرا شود

- `3` = سه بار اجرا شود

- `infinite` = بی‌نهایت بار تکرار شود

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #3498db;  margin: 30px;  display: inline-block;  }  @keyframes pulse {  0% { transform: scale(1); }  50% { transform: scale(1.2); }  100% { transform: scale(1); }  }  .once {  animation: pulse 1s 1;  }  .three-times {  animation: pulse 1s 3;  }  .infinite {  animation: pulse 1s infinite;  }  </style>  <div class="box once">یک بار</div>  <div class="box three-times">سه بار</div>  <div class="box infinite">بی‌نهایت</div> |

[iteration](webdesign/css/animation/iteration.html)

animation-direction

این ویژگی جهت اجرای انیمیشن را کنترل می‌کند.

- مقادیر:

- `normal` = از اول به آخر (پیش‌فرض)

- `reverse` = از آخر به اول

- `alternate` = یکی در میان normal و reverse

- `alternate-reverse` = یکی در میان reverse و normal

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #9b59b6;  margin: 30px;  display: inline-block;  }  @keyframes slide {  from { transform: translateX(0); }  to { transform: translateX(100px); }  }  .normal { animation: slide 2s infinite normal; }  .reverse { animation: slide 2s infinite reverse; }  .alternate { animation: slide 2s infinite alternate; }  </style>  <div class="box normal">normal</div>  <div class="box reverse">reverse</div>  <div class="box alternate">alternate</div> |

[animation-direction](webdesign/css/animation/animation-direction.html)

animation-fill-mode

این ویژگی مشخص می‌کند که عنصر قبل و بعد از انیمیشن چه سبکی باید داشته باشد.

- مقادیر:

- `none` = بدون تغییر (پیش‌فرض)

- `forwards` = سبک پایان انیمیشن حفظ شود

- `backwards` = سبک شروع انیمیشن از ابتدا اعمال شود

- `both` = ترکیب forwards و backwards

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #e67e22;  margin: 30px;  display: inline-block;  opacity: 0.3;  }  @keyframes fade-in {  from { opacity: 0; transform: scale(0.5); }  to { opacity: 1; transform: scale(1); }  }  .none { animation: fade-in 2s 1 none; }  .forwards { animation: fade-in 2s 1 forwards; }  .backwards { animation: fade-in 2s 1 backwards; }  .both { animation: fade-in 2s 1 both; }  </style>  <div class="box none">none</div>  <div class="box forwards">forwards</div>  <div class="box backwards">backwards</div>  <div class="box both">both</div> |

[fill](webdesign/css/animation/fill.html)

Animation shorthand

ویژگی `animation` به صورت خلاصه می‌تواند تمام زیرویژگی‌ها را در یک خط دریافت کند.

- ترتیب: `name duration timing-function delay iteration-count direction fill-mode`

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #2ecc71;  margin: 30px;  display: inline-block;  }  @keyframes complex-animation {  0% { transform: translateX(0) rotate(0deg); background: #2ecc71; }  50% { transform: translateX(100px) rotate(180deg); background: #e74c3c; }  100% { transform: translateX(0) rotate(360deg); background: #3498db; }  }  .shorthand {  /\* name, duration, timing-function, delay, iteration-count, direction \*/  animation: complex-animation 3s ease-in-out 1s infinite alternate;  }  .detailed {  animation-name: complex-animation;  animation-duration: 3s;  animation-timing-function: ease-in-out;  animation-delay: 1s;  animation-iteration-count: infinite;  animation-direction: alternate;  }  </style>  <div class="box shorthand">Shorthand</div>  <div class="box detailed">Detailed</div> |

[shorthand](webdesign/css/animation/shorthand.html)

معرفی کتاب خانه animate.css

کتابخانه Animate.css یک کتابخانه آماده برای اضافه کردن انیمیشن‌های زیبا به عناصر وب‌سایت است. این کتابخانه شامل بیش از 100 انیمیشن مختلف می‌باشد.

نصب و راه‌اندازی

روش ۱: استفاده از CDN

|  |
| --- |
| <head>  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css">  </head> |

روش ۲: دانلود فایل CSS

روش‌های دیگر نصب شامل استفاده از npm (npm install animate.css --save) یا  
 Yarn (yarn add animate.css) است

ساختار پایه استفاده

برای استفاده از انیمیشن‌ها، دو کلاس اصلی نیاز است:

- `animate\_\_animated` (کلاس پایه)

- `animate\_\_[نام-انیمیشن]` (نام انیمیشن خاص)

|  |
| --- |
| <style>  .box {  width: 100px;  height: 100px;  background: #3498db;  margin: 30px;  display: inline-block;  }  </style>  <div class="box animate\_\_animated animate\_\_bounce">انیمیشن Bounce</div>  <div class="box animate\_\_animated animate\_\_fadeIn">انیمیشن FadeIn</div>  <div class="box animate\_\_animated animate\_\_slideInLeft">انیمیشن SlideIn</div> |

[animate.css](webdesign/css/animation/animate.css.html)

دسته‌بندی انیمیشن‌ها

انیمیشن‌های ورود (Entrance):

|  |
| --- |
| <div class="animate\_\_animated animate\_\_fadeInUp">Fade In Up</div>  <div class="animate\_\_animated animate\_\_zoomIn">Zoom In</div>  <div class="animate\_\_animated animate\_\_slideInDown">Slide In Down</div> |

انیمیشن‌های خروج (Exit):

|  |
| --- |
| <div class="animate\_\_animated animate\_\_fadeOut">Fade Out</div>  <div class="animate\_\_animated animate\_\_zoomOut">Zoom Out</div>  <div class="animate\_\_animated animate\_\_slideOutUp">Slide Out Up</div> |

انیمیشن‌های توجه‌دهنده (Attention Seekers):

|  |
| --- |
| <div class="animate\_\_animated animate\_\_pulse">Pulse</div>  <div class="animate\_\_animated animate\_\_tada">Tada</div>  <div class="animate\_\_animated animate\_\_wobble">Wobble</div>  <div class="animate\_\_animated animate\_\_heartBeat">Heart Beat</div> |

کلاس‌های کمکی

کنترل سرعت انیمیشن:

|  |
| --- |
| <div class="animate\_\_animated animate\_\_fadeIn animate\_\_slow">آهسته (2s)</div>  <div class="animate\_\_animated animate\_\_fadeIn animate\_\_slower">بسیار آهسته (3s)</div>  <div class="animate\_\_animated animate\_\_fadeIn animate\_\_fast">سریع (800ms)</div>  <div class="animate\_\_animated animate\_\_fadeIn animate\_\_faster">بسیار سریع (500ms)</div> |

کنترل تأخیر انیمیشن:

|  |
| --- |
| <div class="animate\_\_animated animate\_\_fadeIn animate\_\_delay-1s">تأخیر 1 ثانیه</div>  <div class="animate\_\_animated animate\_\_fadeIn animate\_\_delay-2s">تأخیر 2 ثانیه</div>  <div class="animate\_\_animated animate\_\_fadeIn animate\_\_delay-3s">تأخیر 3 ثانیه</div> |

کنترل تکرار انیمیشن:

|  |
| --- |
| <div class="animate\_\_animated animate\_\_bounce animate\_\_repeat-1">تکرار 1 بار</div>  <div class="animate\_\_animated animate\_\_bounce animate\_\_repeat-2">تکرار 2 بار</div>  <div class="animate\_\_animated animate\_\_bounce animate\_\_repeat-3">تکرار 3 بار</div>  <div class="animate\_\_animated animate\_\_bounce animate\_\_infinite">تکرار بی‌نهایت</div> |

مثال‌های کاربردی

کارت با انیمیشن ورود:

|  |
| --- |
| <style>  .card {  width: 200px;  padding: 20px;  background: #f8f9fa;  border-radius: 10px;  box-shadow: 0 2px 10px rgba(0,0,0,0.1);  margin: 20px;  }  </style>  <div class="card animate\_\_animated animate\_\_fadeInUp animate\_\_delay-1s">  <h3>کارت نمونه</h3>  <p>این کارت با انیمیشن fadeInUp نمایش داده می‌شود</p>  </div> |

دکمه با افکت پالس:

|  |
| --- |
| <style>  .btn {  padding: 10px 20px;  background: #e74c3c;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  margin: 10px;  }  </style>  <button class="btn animate\_\_animated animate\_\_pulse animate\_\_infinite animate\_\_slower">  دکمه با پالس مداوم  </button> |

لیست با انیمیشن مرحله‌ای:

|  |
| --- |
| <style>  .list-item {  padding: 10px;  background: #ecf0f1;  margin: 5px;  border-radius: 5px;  }  </style>  <div class="list-item animate\_\_animated animate\_\_fadeInLeft">آیتم ۱</div>  <div class="list-item animate\_\_animated animate\_\_fadeInLeft animate\_\_delay-1s">آیتم ۲</div>  <div class="list-item animate\_\_animated animate\_\_fadeInLeft animate\_\_delay-2s">آیتم ۳</div> |

نکات مهم

1. ترتیب کلاس‌ها: همیشه کلاس `animate\_\_animated` را اول قرار دهید

2. پیشوند: تمام کلاس‌ها با پیشوند `animate\_\_` شروع می‌شوند

3. کارایی: از انیمیشن‌های سنگین روی عناصر زیاد خودداری کنید

4. تجربه کاربر: انیمیشن‌ها را به گونه‌ای استفاده کنید که باعث سردرگمی کاربر نشوند

## Resize

Resize در CSS:

ویژگی `resize` در CSS به کاربران امکان تغییر اندازه عناصر را می‌دهد. این ویژگی معمولاً روی عناصر `<textarea>` و `<div>` با `overflow` مشخص استفاده می‌شود.

- مقادیر:

- `none` = غیرفعال کردن تغییر اندازه

- `both` = تغییر اندازه در هر دو جهت (عرض و ارتفاع)

- `horizontal` = تغییر اندازه فقط در جهت افقی

- `vertical` = تغییر اندازه فقط در جهت عمودی

|  |
| --- |
| <style>  .container {  width: 400px;  padding: 20px;  background: #f8f9fa;  border-radius: 10px;  margin: 30px auto;  font-family: Arial, sans-serif;  }  .resizable-box {  width: 300px;  height: 120px;  padding: 15px;  background: #3498db;  color: white;  border: 2px solid #2980b9;  margin: 15px 0;  overflow: auto;  border-radius: 5px;  }  .resize-none {  resize: none;  background: #e74c3c;  border-color: #c0392b;  }  .resize-both {  resize: both;  background: #2ecc71;  border-color: #27ae60;  }  .resize-horizontal {  resize: horizontal;  background: #f39c12;  border-color: #d35400;  }  .resize-vertical {  resize: vertical;  background: #9b59b6;  border-color: #8e44ad;  }  .textarea-base {  width: 100%;  height: 80px;  padding: 12px;  border: 2px solid #bdc3c7;  border-radius: 5px;  font-family: Arial, sans-serif;  font-size: 14px;  margin: 10px 0;  box-sizing: border-box;  }  .title {  color: #2c3e50;  margin-bottom: 20px;  text-align: center;  }  .section-title {  color: #34495e;  margin: 25px 0 15px 0;  border-bottom: 2px solid #ecf0f1;  padding-bottom: 8px;  }  </style>  <div class="container">  <h2 class="title">آموزش ویژگی Resize در CSS</h2>    <h3 class="section-title">انواع resize روی divها:</h3>    <div class="resizable-box resize-none">  <strong>resize: none</strong><br>  تغییر اندازه این باکس غیرفعال است  </div>    <div class="resizable-box resize-both">  <strong>resize: both</strong><br>  می‌توانید این باکس را در هر دو جهت (عرض و ارتفاع) تغییر اندازه دهید  </div>    <div class="resizable-box resize-horizontal">  <strong>resize: horizontal</strong><br>  فقط می‌توانید عرض این باکس را تغییر دهید  </div>    <div class="resizable-box resize-vertical">  <strong>resize: vertical</strong><br>  فقط می‌توانید ارتفاع این باکس را تغییر دهید  </div>  <h3 class="section-title">انواع resize روی textarea:</h3>    <textarea class="textarea-base resize-none" placeholder="این textarea قابل تغییر اندازه نیست"></textarea>    <textarea class="textarea-base resize-both" placeholder="این textarea در هر دو جهت قابل تغییر است"></textarea>    <textarea class="textarea-base resize-horizontal" placeholder="فقط عرض این textarea قابل تغییر است"></textarea>    <textarea class="textarea-base resize-vertical" placeholder="فقط ارتفاع این textarea قابل تغییر است"></textarea>  </div> |

[Resize](webdesign/css/Resize/index.html)

نکات مهم:

1. الزامی بودن overflow: برای کارکرد `resize` روی عناصر غیر از `<textarea>`، باید ویژگی `overflow` روی `auto`, `scroll` یا `hidden` تنظیم شود.

2. مقادیر پیش‌فرض:

- `<textarea>`: `resize: both` (پیش‌فرض مرورگر)

- سایر عناصر: `resize: none`

3. کاربردهای عملی:

- ایجاد ویرایشگر متن قابل تنظیم

- پنل‌های قابل تغییر اندازه

- ویجت‌های قابل شخصی‌سازی

## [Table](webdesign/css/table)

قاب یا border در جدول

برای ایجاد جلوه جدول، border باید به خود جدول و سلول‌ها (th, td) اعمال شود.

ویژگی border-collapse:

- collapse: حاشیه‌ها در هم ادغام می‌شوند

- separate: حاشیه‌ها جداگانه نمایش داده می‌شوند

|  |
| --- |
| <style>  table {  width: 100%;  border-collapse: collapse;  }  table, th, td {  border: 1px solid black;  }  </style>  <table>  <tr>  <td>سلول اول</td>  <td>سلول دوم</td>  </tr>  <tr>  <td>سلول سوم</td>  <td>سلول چهارم</td>  </tr>  </table> |

[table](webdesign/css/table/index.html)

empty-cells

کنترل نمایش سلول‌های خالی

مقادیر:

- show: سلول‌های خالی نمایش داده می‌شوند (پیش‌فرض)

- hide: سلول‌های خالی مخفی می‌شوند

شرط کارکرد:

فقط با `border-collapse: separate;` کار می‌کند

مثال:

|  |
| --- |
| <style>  table {  border-collapse: separate;  empty-cells: hide;  }  table, th, td {  border: 1px solid black;  }  </style>  <table>  <tr>  <td>محتوا دار</td>  <td></td> <!-- خالی - مخفی می‌شود -->  </tr>  </table> |

[empty-cells](webdesign/css/table/empty-cells.html)

table-layout

الگوریتم طرح‌بندی جدول

مقادیر:

- auto: اندازه سلول‌ها بر اساس محتوا تنظیم می‌شود (پیش‌فرض)

- fixed: اندازه سلول‌ها ثابت و بر اساس عرض ستون‌ها تنظیم می‌شود

مثال:

|  |
| --- |
| <style>  table {  width: 100%;  table-layout: fixed;  }  td {  border: 1px solid black;  width: 33%; /\* هر ستون 33% عرض را می‌گیرد \*/  }  </style>  <table>  <tr>  <td>ستون اول</td>  <td>ستون دوم با محتوای طولانی‌تر</td>  <td>ستون سوم</td>  </tr>  </table> |

[table-layout](webdesign/css/table/table-layout.html)

caption-side  
موقعیت عنوان جدول (caption)

مقادیر:

- top: عنوان در بالای جدول (پیش‌فرض)

- bottom: عنوان در زیر جدول

مثال:

|  |
| --- |
| <style>  table {  border-collapse: collapse;  width: 100%;  }  table, th, td {  border: 1px solid black;  }  caption {  caption-side: bottom;  font-weight: bold;  padding: 10px;  }  </style>  <table>  <caption>این عنوان در زیر جدول نمایش داده می‌شود</caption>  <tr>  <td>ردیف اول</td>  <td>محتوا</td>  </tr>  <tr>  <td>ردیف دوم</td>  <td>محتوا</td>  </tr>  </table> |

[caption-side](webdesign/css/table/caption-side.html)

## [معرفی swiper js](webdesign/js/swiper)

Swiper.js یک کتابخانه جاوااسکریپت برای ساخت اسلایدرهای مدرن و واکنش‌گرا است.

نصب و راه‌اندازی

روش ۱: استفاده از CDN (ساده‌ترین روش)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <!-- اضافه کردن CSS Swiper -->  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css" />  <style>  /\* استایل پایه برای اسلایدر \*/  .swiper {  width: 100%;  height: 400px;  }    .swiper-slide {  text-align: center;  font-size: 18px;  background: #fff;  display: flex;  justify-content: center;  align-items: center;  }    .swiper-slide img {  display: block;  width: 100%;  height: 100%;  object-fit: cover;  }  </style>  </head>  <body>  <!-- ساختار HTML اسلایدر -->  <div class="swiper mySwiper">  <div class="swiper-wrapper">  <div class="swiper-slide">اسلاید 1</div>  <div class="swiper-slide">اسلاید 2</div>  <div class="swiper-slide">اسلاید 3</div>  <div class="swiper-slide">اسلاید 4</div>  <div class="swiper-slide">اسلاید 5</div>  </div>    <!-- دکمه‌های قبلی و بعدی -->  <div class="swiper-button-next"></div>  <div class="swiper-button-prev"></div>    <!-- نقاط pagination -->  <div class="swiper-pagination"></div>  </div>  <!-- اضافه کردن JS Swiper -->  <script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>  <script>  // راه‌اندازی ساده Swiper  var swiper = new Swiper(".mySwiper", {  spaceBetween: 30,  centeredSlides: true,  autoplay: {  delay: 2500,  disableOnInteraction: false,  },  pagination: {  el: ".swiper-pagination",  clickable: true,  },  navigation: {  nextEl: ".swiper-button-next",  prevEl: ".swiper-button-prev",  },  });  </script>  </body>  </html> |

[swiper](webdesign/js/swiper/index.html)

انواع پیکربندی‌های ساده

۱. اسلایدر ساده با navigation

|  |
| --- |
| var swiper = new Swiper('.mySwiper', {  direction: 'horizontal',  loop: true,    navigation: {  nextEl: '.swiper-button-next',  prevEl: '.swiper-button-prev',  },  }); |

۲. اسلایدر با pagination (نقاط)

|  |
| --- |
| var swiper = new Swiper('.mySwiper', {  pagination: {  el: '.swiper-pagination',  type: 'bullets',  clickable: true  },  }); |

۳. اسلایدر اتوماتیک

|  |
| --- |
| var swiper = new Swiper('.mySwiper', {  autoplay: {  delay: 3000,  },  loop: true  }); |

۴. اسلایدر با چند اسلاید در یک view

|  |
| --- |
| var swiper = new Swiper('.mySwiper', {  slidesPerView: 3,  spaceBetween: 30,  pagination: {  el: ".swiper-pagination",  clickable: true,  },  }); |

مثال کامل و عملی

اسلایدر تصاویر

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>اسلایدر تصاویر - Swiper.js</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.css" />  <style>  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }    body {  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  min-height: 100vh;  display: flex;  justify-content: center;  align-items: center;  padding: 20px;  }    .slider-container {  width: 100%;  max-width: 1200px;  background: white;  border-radius: 20px;  box-shadow: 0 20px 40px rgba(0,0,0,0.1);  padding: 30px;  margin: 50px auto;  }    .slider-title {  text-align: center;  color: #333;  font-size: 2.5em;  margin-bottom: 30px;  font-weight: bold;  background: linear-gradient(45deg, #667eea, #764ba2);  -webkit-background-clip: text;  -webkit-text-fill-color: transparent;  }    .swiper {  width: 100%;  height: 500px;  border-radius: 15px;  overflow: hidden;  }    .swiper-slide {  position: relative;  background: #f8f9fa;  }    .slide-image {  width: 100%;  height: 100%;  object-fit: cover;  }    .slide-content {  position: absolute;  bottom: 0;  right: 0;  left: 0;  background: linear-gradient(transparent, rgba(0,0,0,0.8));  color: white;  padding: 40px 30px 30px;  transform: translateY(0);  transition: transform 0.3s ease;  }    .swiper-slide-active .slide-content {  transform: translateY(0);  }    .slide-title {  font-size: 1.8em;  margin-bottom: 10px;  font-weight: bold;  }    .slide-description {  font-size: 1.1em;  opacity: 0.9;  line-height: 1.6;  }    .swiper-pagination-bullet {  width: 12px;  height: 12px;  background: white;  opacity: 0.5;  transition: all 0.3s ease;  }    .swiper-pagination-bullet-active {  background: #667eea;  opacity: 1;  transform: scale(1.2);  }    .swiper-button-next, .swiper-button-prev {  color: white;  background: rgba(102, 126, 234, 0.8);  width: 50px;  height: 50px;  border-radius: 50%;  transition: all 0.3s ease;  }    .swiper-button-next:after, .swiper-button-prev:after {  font-size: 20px;  font-weight: bold;  }    .swiper-button-next:hover, .swiper-button-prev:hover {  background: #667eea;  transform: scale(1.1);  }    /\* کلاس‌های مخصوص تصاویر \*/  .nature-image {  background: linear-gradient(45deg, #4facfe, #00f2fe);  }    .mountain-image {  background: linear-gradient(45deg, #43e97b, #38f9d7);  }    .city-image {  background: linear-gradient(45deg, #fa709a, #fee140);  }    .beach-image {  background: linear-gradient(45deg, #667eea, #764ba2);  }    .forest-image {  background: linear-gradient(45deg, #96e6a1, #d4fc79);  }    /\* استایل برای نمایش روی موبایل \*/  @media (max-width: 768px) {  .swiper {  height: 400px;  }    .slider-title {  font-size: 2em;  }    .slide-title {  font-size: 1.5em;  }    .slide-description {  font-size: 1em;  }    .swiper-button-next, .swiper-button-prev {  width: 40px;  height: 40px;  }  }    @media (max-width: 480px) {  .swiper {  height: 300px;  }    .slider-title {  font-size: 1.8em;  }    .slide-content {  padding: 20px 15px 15px;  }    .slide-title {  font-size: 1.3em;  }    .slide-description {  font-size: 0.9em;  }  }  </style>  </head>  <body>  <div class="slider-container">  <h1 class="slider-title">گالری تصاویر زیبا</h1>    <div class="swiper mySwiper">  <div class="swiper-wrapper">  <!-- اسلاید ۱ -->  <div class="swiper-slide">  <div class="slide-image nature-image"></div>  <div class="slide-content">  <h2 class="slide-title">طبیعت بکر</h2>  <p class="slide-description">منظره‌ای زیبا از طبیعت دست‌نخورده و رودخانه‌های خروشان</p>  </div>  </div>    <!-- اسلاید ۲ -->  <div class="swiper-slide">  <div class="slide-image mountain-image"></div>  <div class="slide-content">  <h2 class="slide-title">کوه‌های سر به فلک کشیده</h2>  <p class="slide-description">نمایی باشکوه از قله‌های برفی و ابرهای سفید</p>  </div>  </div>    <!-- اسلاید ۳ -->  <div class="swiper-slide">  <div class="slide-image city-image"></div>  <div class="slide-content">  <h2 class="slide-title">شهر مدرن</h2>  <p class="slide-description">نمای شبانه شهر با آسمان‌خراش‌های درخشان</p>  </div>  </div>    <!-- اسلاید ۴ -->  <div class="swiper-slide">  <div class="slide-image beach-image"></div>  <div class="slide-content">  <h2 class="slide-title">ساحل طلایی</h2>  <p class="slide-description">دریای آبی و شن‌های طلایی در غروب آفتاب</p>  </div>  </div>    <!-- اسلاید ۵ -->  <div class="swiper-slide">  <div class="slide-image forest-image"></div>  <div class="slide-content">  <h2 class="slide-title">جنگل اسرارآمیز</h2>  <p class="slide-description">عمق جنگل با درختان سرسبز و نورهای طلایی</p>  </div>  </div>  </div>    <!-- نقاط pagination -->  <div class="swiper-pagination"></div>    <!-- دکمه‌های navigation -->  <div class="swiper-button-next"></div>  <div class="swiper-button-prev"></div>  </div>  </div>  <script src="https://cdn.jsdelivr.net/npm/swiper@11/swiper-bundle.min.js"></script>  <script>  // راه‌اندازی Swiper  var swiper = new Swiper(".mySwiper", {  // تنظیمات پایه  direction: 'horizontal',  loop: true,  speed: 800,  grabCursor: true,    // اتوپلی  autoplay: {  delay: 5000,  disableOnInteraction: false,  pauseOnMouseEnter: true  },    // pagination (نقاط)  pagination: {  el: ".swiper-pagination",  clickable: true,  dynamicBullets: true,  renderBullet: function (index, className) {  return '<span class="' + className + '">' + (index + 1) + '</span>';  }  },    // navigation (دکمه‌ها)  navigation: {  nextEl: ".swiper-button-next",  prevEl: ".swiper-button-prev",  },    // افکت‌ها  effect: 'slide',  fadeEffect: {  crossFade: true  },    // breakpoints برای ریسپانسیو  breakpoints: {  320: {  slidesPerView: 1,  spaceBetween: 10  },  768: {  slidesPerView: 1,  spaceBetween: 20  },  1024: {  slidesPerView: 1,  spaceBetween: 30  }  },    // رویدادها  on: {  init: function () {  console.log('Swiper initialized successfully!');  },  slideChange: function () {  console.log('Slide changed to: ', this.activeIndex);  }  }  });    // متوقف کردن اتوپلی وقتی ماوس روی اسلایدر است  const swiperContainer = document.querySelector('.swiper');  swiperContainer.addEventListener('mouseenter', function() {  swiper.autoplay.stop();  });    swiperContainer.addEventListener('mouseleave', function() {  swiper.autoplay.start();  });  </script>  </body>  </html> |

[image](webdesign/js/swiper/image.html)

تنظیمات مهم و پرکاربرد

|  |  |  |
| --- | --- | --- |
| تنظیم | توضیح | مقدار پیش‌فرض |
| `direction` | جهت اسلایدر | 'horizontal' |
| `loop` | حلقه‌ای شدن | false |
| `speed` | سرعت انتقال (میلی‌ثانیه) | 300 |
| `autoplay` | پخش خودکار | false |
| `pagination` | نقاط پایین اسلایدر | false |
| `navigation` | دکمه‌های قبلی/بعدی | false |
| `slidesPerView` | تعداد اسلایدهای visible | 1 |
| `spaceBetween` | فاصله بین اسلایدها (px) | 0 |

نکات مهم برای شروع:

1. حتما ساختار HTML را رعایت کنید

2. ابعاد اسلایدر را در CSS مشخص کنید

3. فایل‌های CSS و JS را فراموش نکنید

4. برای ریسپانسیو از breakpoints استفاده کنید

## [Flex](webdesign/css/Flex)

Flexbox یک مدل layout در CSS است که طراحی ریسپانسیو و چیدمان عناصر را آسان می‌کند.

flex-direction

تعریف:

تعیین جهت اصلی چیدمان عناصر در container

مقادیر:

|  |
| --- |
| .container {  flex-direction: row; /\* پیش‌فرض - (در LTR) \*/  flex-direction: row-reverse; /\* از راست به چپ (در LTR) \*/  flex-direction: column; /\* از بالا به پایین \*/  flex-direction: column-reverse; /\* از پایین به بالا \*/  } |

مثال عملی:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <style>  .container {  display: flex;  width: 400px;  height: 200px;  border: 2px solid #333;  margin: 20px 0;  }    .item {  padding: 20px;  margin: 5px;  background: #4CAF50;  color: white;  text-align: center;  }    .row { flex-direction: row; }  .row-reverse { flex-direction: row-reverse; }  .column { flex-direction: column; }  .column-reverse { flex-direction: column-reverse; }  </style>  </head>  <body>  <h3>flex-direction: row</h3>  <div class="container row">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>    <h3>flex-direction: row-reverse</h3>  <div class="container row-reverse">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>    <h3>flex-direction: column</h3>  <div class="container column">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>    <h3>flex-direction: column-reverse</h3>  <div class="container column-reverse">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>  </body>  </html> |

[flex-direction](webdesign/css/Flex/flex-direction.html)

flex-wrap

تعریف:

تعیین اینکه آیا عناصر در یک خط بمانند یا در چند خط شکسته شوند

مقادیر:

|  |
| --- |
| .container {  flex-wrap: nowrap; /\* پیش‌فرض - همه در یک خط \*/  flex-wrap: wrap; /\* شکسته شدن به خطوط جدید \*/  flex-wrap: wrap-reverse; /\* شکسته شدن به خطوط جدید از پایین \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  width: 300px;  border: 2px solid #333;  margin: 20px 0;  }    .item {  width: 100px;  padding: 20px;  margin: 5px;  background: #2196F3;  color: white;  text-align: center;  }    .nowrap { flex-wrap: nowrap; }  .wrap { flex-wrap: wrap; }  .wrap-reverse { flex-wrap: wrap-reverse; }  </style>  <div class="container nowrap">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  <div class="item">۴</div>  </div>  <div class="container wrap">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  <div class="item">۴</div>  </div>  <div class="container wrap-reverse">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  <div class="item">۴</div>  </div> |

[flex-wrap](webdesign/css/Flex/flex-wrap.html)

flex-flow

تعریف:

shortcut برای `flex-direction` و `flex-wrap`

ساختار:

|  |
| --- |
| .container {  flex-flow: <flex-direction> <flex-wrap>;  } |

مثال:

|  |
| --- |
| .container {  flex-flow: row wrap; /\* جهت ردیفی + شکسته شدن \*/  flex-flow: column nowrap; /\* جهت ستونی + بدون شکسته شدن \*/  flex-flow: row-reverse wrap; /\* جهت معکوس + شکسته شدن \*/  } |

justify-content

تعریف:

تراز کردن عناصر در جهت اصلی (محور x در row)

مقادیر:

|  |
| --- |
| .container {  justify-content: flex-start; /\* ابتدا (راست در RTL) \*/  justify-content: flex-end; /\* انتها (چپ در RTL) \*/  justify-content: center; /\* وسط \*/  justify-content: space-between; /\* فاصله بین عناصر \*/  justify-content: space-around; /\* فاصله اطراف عناصر \*/  justify-content: space-evenly; /\* فاصله یکسان \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  width: 500px;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 20px;  background: #FF9800;  color: white;  text-align: center;  }    .start { justify-content: flex-start; }  .end { justify-content: flex-end; }  .center { justify-content: center; }  .between { justify-content: space-between; }  .around { justify-content: space-around; }  .evenly { justify-content: space-evenly; }  </style>  <div class="container start">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>  <div class="container end">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div>  <div class="container between">  <div class="item">۱</div>  <div class="item">۲</div>  <div class="item">۳</div>  </div> |

[justify-content](webdesign/css/Flex/justify-content.html)

align-items

تعریف:

تراز کردن عناصر در جهت متقاطع (محور y در row)

مقادیر:

|  |
| --- |
| .container {  align-items: stretch; /\* پیش‌فرض - کشیده شدن \*/  align-items: flex-start; /\* ابتدا \*/  align-items: flex-end; /\* انتها \*/  align-items: center; /\* وسط \*/  align-items: baseline; /\* تراز بر اساس خط پایه متن \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  height: 150px;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 10px 20px;  background: #9C27B0;  color: white;  text-align: center;  }    .item:nth-child(2) { padding: 30px 20px; }  .item:nth-child(3) { padding: 15px 20px; }    .stretch { align-items: stretch; }  .start { align-items: flex-start; }  .end { align-items: flex-end; }  .center { align-items: center; }  .baseline { align-items: baseline; }  </style>  <div class="container stretch">  <div class="item">آیتم ۱</div>  <div class="item">آیتم ۲</div>  <div class="item">آیتم ۳</div>  </div>  <div class="container center">  <div class="item">آیتم ۱</div>  <div class="item">آیتم ۲</div>  <div class="item">آیتم ۳</div>  </div>  <div class="container baseline">  <div class="item">آیتم ۱</div>  <div class="item">آیتم ۲</div>  <div class="item">آیتم ۳</div>  </div> |

[align-items](webdesign/css/Flex/align-items.html)

align-content

تعریف:

تراز کردن خطوط در جهت متقاطع وقتی چندین خط وجود دارد

مقادیر:

|  |
| --- |
| .container {  align-content: stretch; /\* پیش‌فرض \*/  align-content: flex-start; /\* ابتدا \*/  align-content: flex-end; /\* انتها \*/  align-content: center; /\* وسط \*/  align-content: space-between; /\* فاصله بین خطوط \*/  align-content: space-around; /\* فاصله اطراف خطوط \*/  } |

مثال:

|  |
| --- |
| <style>  .container {  display: flex;  flex-wrap: wrap;  height: 300px;  width: 400px;  border: 2px solid #333;  margin: 20px 0;  }    .item {  width: 180px;  padding: 20px;  margin: 5px;  background: #795548;  color: white;  text-align: center;  }    .content-start { align-content: flex-start; }  .content-center { align-content: center; }  .content-between { align-content: space-between; }  </style>  <div class="container content-start">  <div class="item">خط ۱</div>  <div class="item">خط ۲</div>  <div class="item">خط ۳</div>  </div>  <div class="container content-center">  <div class="item">خط ۱</div>  <div class="item">خط ۲</div>  <div class="item">خط ۳</div>  </div> |

[align-content](webdesign/css/Flex/align-content.html)

align-self

تعریف:

تراز کردن یک آیتم خاص در جهت متقاطع (override برای align-items)

مقادیر:

|  |
| --- |
| .item {  align-self: auto; /\* پیش‌فرض - ارث بری از container \*/  align-self: flex-start; /\* ابتدا \*/  align-self: flex-end; /\* انتها \*/  align-self: center; /\* وسط \*/  align-self: stretch; /\* کشیده شدن \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  height: 200px;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 20px;  margin: 5px;  background: #607D8B;  color: white;  text-align: center;  }    .self-start { align-self: flex-start; }  .self-end { align-self: flex-end; }  .self-center { align-self: center; }  .self-stretch { align-self: stretch; }  </style>  <div class="container">  <div class="item">عادی</div>  <div class="item self-start">شروع</div>  <div class="item">عادی</div>  <div class="item self-center">وسط</div>  <div class="item self-end">پایان</div>  </div> |

[align-self](webdesign/css/Flex/align-self.html)

Order

تعریف:

تعیین ترتیب نمایش آیتم‌ها

مقدار:

|  |
| --- |
| .item {  order: <number>; /\* عدد صحیح - پیش‌فرض: 0 \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 20px;  margin: 5px;  background: #E91E63;  color: white;  text-align: center;  }    .first { order: -1; }  .last { order: 1; }  .middle { order: 0; }  </style>  <div class="container">  <div class="item last">۳ (order: 1)</div>  <div class="item first">۱ (order: -1)</div>  <div class="item middle">۲ (order: 0)</div>  </div> |

[Order](webdesign/css/Flex/Order.html)

flex-grow

تعریف:

تعیین میزان رشد آیتم نسبت به دیگر آیتم‌ها

مقدار:

|  |
| --- |
| .item {  flex-grow: <number>; /\* پیش‌فرض: 0 \*/  } |

مثال عملی:

|  |
| --- |
| <style>  .container {  display: flex;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 20px;  margin: 5px;  background: #00BCD4;  color: white;  text-align: center;  }    .grow-1 { flex-grow: 1; }  .grow-2 { flex-grow: 2; }  .grow-3 { flex-grow: 3; }  </style>  <div class="container">  <div class="item grow-1">grow: 1</div>  <div class="item grow-2">grow: 2</div>  <div class="item grow-3">grow: 3</div>  </div> |

[flex-grow](webdesign/css/Flex/flex-grow.html)

flex-shrink

تعریف:

تعیین میزان جمع شدن آیتم نسبت به دیگر آیتم‌ها

مقدار:

|  |
| --- |
| .item {  flex-shrink: <number>; /\* پیش‌فرض: 1 \*/  } |

مثال:

|  |
| --- |
| <style>  .container {  display: flex;  width: 400px;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  width: 200px;  padding: 20px;  margin: 5px;  background: #8BC34A;  color: white;  text-align: center;  }    .shrink-0 { flex-shrink: 0; } /\* جمع نمی‌شود \*/  .shrink-1 { flex-shrink: 1; } /\* جمع شدن نرمال \*/  .shrink-2 { flex-shrink: 2; } /\* دو برابر جمع می‌شود \*/  </style>  <div class="container">  <div class="item shrink-0">عدم جمع شدن</div>  <div class="item shrink-1">جمع شدن نرمال</div>  <div class="item shrink-2">جمع شدن دوبرابر</div>  </div> |

[flex-shrink](webdesign/css/Flex/flex-shrink.html)

flex-basis

تعریف:

تعیین اندازه اولیه آیتم قبل از اعمال فضای اضافی

مقدار:

|  |
| --- |
| .item {  flex-basis: auto | <width>; /\* پیش‌فرض: auto \*/  } |

flex-shorthand

تعریف:

shortcut برای `flex-grow`, `flex-shrink`, `flex-basis`

ساختار:

|  |
| --- |
| .item {  flex: <flex-grow> <flex-shrink> <flex-basis>;  } |

مثال‌های پرکاربرد:

|  |
| --- |
| .item {  flex: 1; /\* flex: 1 1 0 \*/  flex: 0 0 auto; /\* اندازه ثابت \*/  flex: 2 1 200px; /\* رشد:2, جمع شدن:1, پایه:200px \*/  flex: auto; /\* flex: 1 1 auto \*/  flex: none; /\* flex: 0 0 auto \*/  } |

مثال عملی کامل:

|  |
| --- |
| <style>  .container {  display: flex;  border: 2px solid #333;  margin: 20px 0;  padding: 10px;  }    .item {  padding: 20px;  margin: 5px;  background: #FF5722;  color: white;  text-align: center;  }    .flex-1 { flex: 1; }  .flex-2 { flex: 2; }  .flex-auto { flex: auto; }  .flex-none { flex: none; width: 100px; }  </style>  <div class="container">  <div class="item flex-1">flex: 1</div>  <div class="item flex-2">flex: 2</div>  <div class="item flex-auto">flex: auto</div>  <div class="item flex-none">flex: none</div>  </div> |

[flex](webdesign/css/Flex/index.html)

خلاصه نکات مهم:

|  |  |  |
| --- | --- | --- |
| ویژگی | کاربرد | مقدار پیش‌فرض |
| `flex-direction` | جهت چیدمان | `row` |
| `flex-wrap` | شکسته شدن خطوط | `nowrap` |
| `justify-content` | تراز محور اصلی | `flex-start` |
| `align-items` | تراز محور متقاطع | `stretch` |
| `flex-grow` | میزان رشد | `0` |
| `flex-shrink` | میزان جمع شدن | `1` |
| `order` | ترتیب نمایش | `0` |

## [آموزش وسط چین کردن تگ ها](webdesign/css/center)

1. وسط‌چین متن با `text-align`

2. وسط‌چین عناصر اینلاین

3. وسط‌چین با Flexbox

4. وسط‌چین با Margin Auto

5. وسط‌چین عمودی و افقی

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش وسط چین کردن</title>  <style>  /\* استایل پایه \*/  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  }  body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  line-height: 1.6;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  min-height: 100vh;  padding: 20px;  }  .container {  max-width: 1200px;  margin: 0 auto;  }  .section-title {  color: #333;  margin-bottom: 30px;  text-align: center;  font-size: 2em;  }  /\* وسط چین متن \*/  .text-center-example {  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  margin-bottom: 30px;  }  .centered-text {  text-align: center;  font-size: 1.2em;  color: #667eea;  margin-bottom: 15px;  padding: 10px;  background: #f8f9fa;  border-radius: 5px;  }  .normal-text {  text-align: right;  color: #666;  padding: 10px;  background: #f8f9fa;  border-radius: 5px;  }  /\* وسط چین عناصر اینلاین \*/  .inline-center-example {  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  margin-bottom: 30px;  }  .inline-container {  text-align: center;  padding: 20px;  background: #f8f9fa;  border-radius: 8px;  border: 2px dashed #ddd;  }  .inline-item {  display: inline-block;  padding: 10px 20px;  margin: 5px;  background: #667eea;  color: white;  border-radius: 5px;  font-weight: bold;  }  /\* وسط چین با Flexbox \*/  .flex-center-example {  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  margin-bottom: 30px;  }  /\* وسط چین افقی \*/  .flex-horizontal {  display: flex;  justify-content: center;  gap: 15px;  padding: 20px;  background: #e3f2fd;  border-radius: 8px;  margin-bottom: 20px;  border: 2px dashed #90caf9;  }  /\* وسط چین عمودی \*/  .flex-vertical {  display: flex;  align-items: center;  height: 150px;  gap: 15px;  padding: 20px;  background: #f3e5f5;  border-radius: 8px;  margin-bottom: 20px;  border: 2px dashed #ce93d8;  }  /\* وسط چین کامل (افقی و عمودی) \*/  .flex-both {  display: flex;  justify-content: center;  align-items: center;  height: 200px;  padding: 20px;  background: #e8f5e8;  border-radius: 8px;  border: 2px dashed #a5d6a7;  }  .flex-item {  padding: 15px 25px;  background: #333;  color: white;  border-radius: 5px;  font-weight: bold;  text-align: center;  min-width: 100px;  }  /\* وسط چین با Margin Auto \*/  .margin-center-example {  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  margin-bottom: 30px;  }  /\* وسط چین افقی با margin auto \*/  .block-center-horizontal {  width: 70%;  margin: 20px auto;  padding: 30px;  background: #fff3e0;  border: 2px dashed #ffb74d;  border-radius: 8px;  text-align: center;  font-weight: bold;  color: #e65100;  }  /\* وسط چین عمودی و افقی با ترکیب روش‌ها \*/  .block-center-vertical {  height: 300px;  position: relative;  background: #fce4ec;  border: 2px dashed #f48fb1;  border-radius: 8px;  margin: 20px 0;  }  .centered-content {  position: absolute;  top: 50%;  left: 50%;  transform: translate(-50%, -50%);  padding: 30px;  background: #ad1457;  color: white;  border-radius: 8px;  text-align: center;  font-weight: bold;  width: 80%;  max-width: 400px;  }  /\* مثال ترکیبی - کارت پروفایل \*/  .combined-example {  background: white;  padding: 30px;  border-radius: 10px;  box-shadow: 0 5px 15px rgba(0,0,0,0.1);  }  .profile-card {  max-width: 400px;  margin: 0 auto;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  border-radius: 15px;  overflow: hidden;  box-shadow: 0 10px 30px rgba(0,0,0,0.2);  }  .profile-header {  padding: 40px 30px 20px;  text-align: center;  background: rgba(255,255,255,0.1);  }  .profile-image {  width: 120px;  height: 120px;  border-radius: 50%;  border: 4px solid white;  margin: 0 auto 20px;  display: block;  background: #ddd;  }  .profile-name {  color: white;  font-size: 1.5em;  margin-bottom: 5px;  }  .profile-title {  color: rgba(255,255,255,0.8);  font-size: 1em;  }  .profile-stats {  display: flex;  justify-content: space-around;  padding: 30px 20px;  background: rgba(255,255,255,0.1);  }  .stat-item {  text-align: center;  color: white;  }  .stat-number {  display: block;  font-size: 1.8em;  font-weight: bold;  margin-bottom: 5px;  }  .stat-label {  font-size: 0.9em;  opacity: 0.8;  }  .profile-actions {  display: flex;  justify-content: center;  gap: 15px;  padding: 30px;  background: rgba(255,255,255,0.05);  }  .btn {  padding: 12px 25px;  border: none;  border-radius: 25px;  font-weight: bold;  cursor: pointer;  transition: all 0.3s ease;  }  .btn-primary {  background: #4CAF50;  color: white;  }  .btn-secondary {  background: transparent;  color: white;  border: 2px solid white;  }  .btn:hover {  transform: translateY(-2px);  box-shadow: 0 5px 15px rgba(0,0,0,0.2);  }  /\* رسپانسیو \*/  @media (max-width: 768px) {  .container {  padding: 10px;  }    .section-title {  font-size: 1.5em;  }    .profile-stats {  flex-direction: column;  gap: 20px;  }    .profile-actions {  flex-direction: column;  align-items: center;  }    .btn {  width: 200px;  }  }  </style>  </head>  <body>  <!-- 1. وسط‌چین متن با text-align -->  <div class="container">  <div class="text-center-example">  <h2 class="section-title">وسط چین متن با text-align</h2>  <p class="centered-text">این متن در وسط قرار گرفته است</p>  <p class="normal-text">این متن به صورت عادی راست چین است</p>  </div>  </div>  <!-- 2. وسط‌چین عناصر اینلاین در یک تگ بلاک -->  <div class="container">  <div class="inline-center-example">  <h2 class="section-title">وسط چین عناصر اینلاین</h2>  <div class="inline-container">  <span class="inline-item">عنصر اینلاین ۱</span>  <span class="inline-item">عنصر اینلاین ۲</span>  <span class="inline-item">عنصر اینلاین ۳</span>  </div>  </div>  </div>  <!-- 3. وسط‌چین با Flexbox -->  <div class="container">  <div class="flex-center-example">  <h2 class="section-title">وسط چین با Flexbox</h2>    <div class="flex-horizontal">  <div class="flex-item">آیتم ۱</div>  <div class="flex-item">آیتم ۲</div>  <div class="flex-item">آیتم ۳</div>  </div>    <div class="flex-vertical">  <div class="flex-item">آیتم عمودی ۱</div>  <div class="flex-item">آیتم عمودی ۲</div>  </div>    <div class="flex-both">  <div class="flex-item">وسط چین کامل</div>  </div>  </div>  </div>  <!-- 4. وسط‌چین با Margin Auto -->  <div class="container">  <div class="margin-center-example">  <h2 class="section-title">وسط چین با Margin Auto</h2>    <div class="block-center-horizontal">  این div با margin auto وسط چین شده است  </div>    <div class="block-center-vertical">  <div class="centered-content">  این محتوا هم افقی و هم عمودی وسط چین شده  </div>  </div>  </div>  </div>  <!-- 5. مثال نهایی ترکیبی -->  <div class="container">  <div class="combined-example">  <h2 class="section-title">مثال ترکیبی - کارت پروفایل</h2>    <div class="profile-card">  <div class="profile-header">  <div class="profile-image"></div>  <h3 class="profile-name">محمد رضایی</h3>  <p class="profile-title">توسعه‌دهنده فرانت‌اند</p>  </div>    <div class="profile-stats">  <div class="stat-item">  <span class="stat-number">۱۲۴</span>  <span class="stat-label">پروژه</span>  </div>  <div class="stat-item">  <span class="stat-number">۸۹</span>  <span class="stat-label">مشتری</span>  </div>  <div class="stat-item">  <span class="stat-number">۵</span>  <span class="stat-label">سال تجربه</span>  </div>  </div>    <div class="profile-actions">  <button class="btn btn-primary">ارتباط با من</button>  <button class="btn btn-secondary">مشاهده رزومه</button>  </div>  </div>  </div>  </div>  </body>  </html> |

[آموزش وسط چین کردن تگ ها](webdesign/css/center/index.html)

خلاصه روش‌های وسط‌چین:

|  |  |  |
| --- | --- | --- |
| روش | کاربرد | کد مثال |
| `text-align: center` | متن و عناصر اینلاین | `.text { text-align: center; }` |
| `margin: 0 auto` | عناصر بلاک | `.block { width: 50%; margin: 0 auto; }` |
| `display: flex` + `justify-content: center` | وسط چین افقی | `.container { display: flex; justify-content: center; }` |
| `display: flex` + `align-items: center` | وسط چین عمودی | `.container { display: flex; align-items: center; }` |
| `position: absolute` + `transform` | وسط چین کامل | `.element { position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); }` |

## [media query](webdesign/css/media%20query)

معرفی Media Query

Media Query یک تکنیک در CSS است که به ما امکان می‌دهد استایل‌های مختلفی بر اساس ویژگی‌های دستگاه (مانند عرض صفحه، ارتفاع، جهت و وضوح) اعمال کنیم.

ساختار پایه Media Query

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  /\* استایل پایه برای همه دستگاه‌ها \*/  .container {  width: 100%;  padding: 20px;  background-color: #f0f0f0;  }  /\* برای دستگاه‌های با عرض کمتر از 768px \*/  @media (max-width: 768px) {  .container {  background-color: #e0e0e0;  padding: 10px;  }  }  /\* برای دستگاه‌های با عرض بیشتر از 1200px \*/  @media (min-width: 1200px) {  .container {  width: 80%;  margin: 0 auto;  background-color: #d0d0d0;  }  }  </style>  </head>  <div class="container">  <p>این یک محتوای نمونه است که بر اساس اندازه صفحه تغییر می‌کند.</p>  </div> |

[index](webdesign/css/media%20query/index.html)

انواع Media Query

|  |
| --- |
| <style>  /\* بر اساس عرض صفحه \*/  @media (min-width: 768px) and (max-width: 1024px) {  .box {  width: 50%;  float: left;  }  }  /\* بر اساس جهت دستگاه \*/  @media (orientation: landscape) {  .header {  height: 100px;  }  }  @media (orientation: portrait) {  .header {  height: 150px;  }  }  /\* بر اساس وضوح صفحه \*/  @media (min-resolution: 2dppx) {  .image {  background-image: url('image@2x.png');  }  }  </style>  </head> |

[media-query](webdesign/css/media%20query/media-query.html)

نکات مهم Media Query:

- از `min-width` برای طراحی Mobile-First استفاده کنید

- از `max-width` برای طراحی Desktop-First استفاده کنید

- Breakpointهای رایج: 768px (تبلت)، 1024px (دسکتاپ کوچک)، 1200px (دسکتاپ بزرگ)

- می‌توانید چندین شرط را با `and` ترکیب کنید

معرفی media query و ساخت صفحات responsive

مثال کامل یک Layout واکنش‌گرا

|  |
| --- |
| <style>  \* {  box-sizing: border-box;  margin: 0;  padding: 0;  }  .container {  width: 100%;  margin: 0 auto;  padding: 15px;  }  .header {  background-color: #333;  color: white;  padding: 20px;  text-align: center;  }  .nav {  background-color: #444;  padding: 10px;  }  .nav ul {  list-style-type: none;  display: flex;  justify-content: space-around;  }  .nav a {  color: white;  text-decoration: none;  }  .main-content {  display: flex;  flex-wrap: wrap;  gap: 20px;  margin: 20px 0;  }  .sidebar {  flex: 1;  min-width: 250px;  background-color: #f4f4f4;  padding: 20px;  }  .content {  flex: 3;  min-width: 300px;  background-color: white;  padding: 20px;  border: 1px solid #ddd;  }  .footer {  background-color: #333;  color: white;  text-align: center;  padding: 20px;  margin-top: 20px;  }  /\* موبایل - عرض کمتر از 768px \*/  @media (max-width: 767px) {  .nav ul {  flex-direction: column;  align-items: center;  }    .nav li {  margin: 5px 0;  }    .main-content {  flex-direction: column;  }    .sidebar, .content {  min-width: 100%;  }  }  /\* تبلت - بین 768px و 1024px \*/  @media (min-width: 768px) and (max-width: 1024px) {  .container {  max-width: 90%;  }    .sidebar {  flex: 1;  }    .content {  flex: 2;  }  }  /\* دسکتاپ - بیشتر از 1024px \*/  @media (min-width: 1025px) {  .container {  max-width: 1200px;  }  }  </style>  <div class="container">  <header class="header">  <h1>سایت واکنش‌گرا</h1>  </header>    <nav class="nav">  <ul>  <li><a href="#">صفحه اصلی</a></li>  <li><a href="#">درباره ما</a></li>  <li><a href="#">خدمات</a></li>  <li><a href="#">تماس با ما</a></li>  </ul>  </nav>    <div class="main-content">  <aside class="sidebar">  <h3>منوی کناری</h3>  <p>محتوای منوی کناری</p>  </aside>    <main class="content">  <h2>محتوای اصلی</h2>  <p>این محتوا بر اساس اندازه صفحه تغییر می‌کند.</p>  </main>  </div>    <footer class="footer">  <p>کپی رایت ۲۰۲۴</p>  </footer>  </div> |

[responsive](webdesign/css/media%20query/responsive.html)

نکات مهم طراحی Responsive:

- همیشه از `viewport` meta tag استفاده کنید

- از واحدهای نسبی (%, vw, vh) به جای واحدهای مطلق استفاده کنید

- از Flexbox یا Grid برای Layout استفاده کنید

- تصاویر را با `max-width: 100%` واکنش‌گرا کنید

## معرفی srcset و x-descriptor و w-descriptor

x-descriptor (تراکم پیکسلی)

|  |
| --- |
| <style>  .responsive-img {  max-width: 100%;  height: auto;  }  </style>  <img  src="image-1x.jpg"  srcset="image-1x.jpg 1x,  image-2x.jpg 2x,  image-3x.jpg 3x"  alt="تصویر نمونه"  class="responsive-img"> |

[descriptor](webdesign/css/descriptor/index.html)

w-descriptor

|  |
| --- |
| <style>  .responsive-img {  width: 100%;  height: auto;  }  </style>  <img  src="image-400.jpg"  srcset="image-400.jpg 400w,  image-800.jpg 800w,  image-1200.jpg 1200w,  image-1600.jpg 1600w"  sizes="(max-width: 600px) 100vw,  (max-width: 1200px) 50vw,  400px"  alt="تصویر نمونه"  class="responsive-img"> |

[w-descriptor](webdesign/css/descriptor/w-descriptor.html)

## معرفی owl carousel

Owl Carousel یک کتابخانه جاوااسکریپت (بر پایه jQuery) برای ساخت کاروسل‌های (اسلایدرهای) واکنش‌گرا است. این کتابخانه با وجود قدیمی شدن، هنوز هم به دلیل سادگی در استفاده و انعطاف‌پذیری بالا محبوب است.

ویژگی‌های اصلی:

|  |  |
| --- | --- |
| ویژگی | توضیح |
| واکنش‌گرا | به طور خودکار با اندازه صفحه سازگار می‌شود |
| پشتیبانی از تاچ | قابلیت کشیدن با انگشت در موبایل |
| تغییر سرعت اتوماتیک | قابلیت حرکت خودکار اسلایدها |
| حلقه‌ای | امکان حرکت بی‌نهایت اسلایدها |
| تنظیمات گسترده | بیش از ۸۰ گزینه برای شخصی‌سازی |

مثال عملی برای دانشجویان

مرحله ۱: ساختار HTML

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Owl Carousel</title>    <!-- لینک فایل CSS Owl Carousel -->  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css">  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.theme.default.min.css">    <style>  .item {  padding: 50px;  text-align: center;  background: #4DC7A0;  margin: 10px;  color: white;  font-size: 20px;  border-radius: 10px;  }    /\* استایل برای نمایش بهتر در موبایل \*/  .owl-carousel .owl-stage {  padding: 10px 0;  }  </style>  </head>  <body>  <!-- کانتینر کاروسل -->  <div class="owl-carousel owl-theme">  <div class="item">  <h3>اسلاید ۱</h3>  <p>محتوای اولین اسلاید</p>  </div>  <div class="item">  <h3>اسلاید ۲</h3>  <p>محتوای دومین اسلاید</p>  </div>  <div class="item">  <h3>اسلاید ۳</h3>  <p>محتوای سومین اسلاید</p>  </div>  <div class="item">  <h3>اسلاید ۴</h3>  <p>محتوای چهارمین اسلاید</p>  </div>  <div class="item">  <h3>اسلاید ۵</h3>  <p>محتوای پنجمین اسلاید</p>  </div>  </div>  <!-- اضافه کردن jQuery و اسکریپت‌های لازم -->  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <script src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/owl.carousel.min.js"></script>  <script>  // راه‌اندازی کاروسل  $(document).ready(function(){  $(".owl-carousel").owlCarousel({  loop: true, // حرکت حلقه‌ای  margin: 10, // فاصله بین آیتم‌ها  nav: true, // نمایش دکمه‌های قبلی/بعدی  responsive: { // تنظیمات واکنش‌گرا  0: {  items: 1 // در موبایل: ۱ اسلاید  },  600: {  items: 2 // در تبلت: ۲ اسلاید  },  1000: {  items: 3 // در دسکتاپ: ۳ اسلاید  }  }  });  });  </script>  </body>  </html> |

### مرحله ۲: توضیح تنظیمات مهم

|  |
| --- |
| $(".owl-carousel").owlCarousel({  // تنظیمات پایه  items: 3, // تعداد اسلایدهای visible  loop: true, // حرکت بی‌نهایت  margin: 20, // فاصله بین آیتم‌ها    // ناوبری  nav: true, // نمایش دکمه‌های navigation  navText: ["قبلی", "بعدی"], // متن دکمه‌ها    // اتوپلی  autoplay: true, // حرکت خودکار  autoplayTimeout: 3000, // مدت زمان نمایش هر اسلاید    // نمایش نقاط (dots)  dots: true, // نمایش نقاط پایین کاروسل    // واکنش‌گرایی  responsive: {  0: { items: 1 }, // از 0px تا 599px: 1 اسلاید  600: { items: 2 }, // از 600px تا 999px: 2 اسلاید  1000: { items: 3 } // از 1000px به بالا: 3 اسلاید  }  }); |

نکات مهم برای دانشجویان

1. ترتیب بارگذاری فایل‌ها مهم است:

- اول jQuery

- سپس Owl Carousel JS

- در آخر اسکریپت شخصی شما

2. کلاس‌های ضروری:

- `owl-carousel` : کلاس اصلی

- `owl-theme` : برای استایل پیش‌فرض

3. مشکل رایج: اگر کاروسل کار نکرد، کنسول مرورگر را بررسی کنید.

## 🔄 جایگزین‌های مدرن

اگر پروژه جدیدی شروع می‌کنید، این گزینه‌ها را نیز در نظر بگیرید:

- Swiper.js - مدرن‌تر و قدرتمندتر

- Glide.js - سبک‌وزن و ساده

- Slick Carousel - شبیه Owl اما به‌روزتر

tiny-slider – پیشنهاد خود owl برای جایگذینی

## پروژه تبدیل UI به صفحات وب

AI هایی مانند  
 https://dev.animaapp.com/  
<https://www.builder.io/>

<https://teleporthq.io/>.

<https://uizard.io/>

<https://10web.io/>

<https://www.wix.com/>

<https://durable.co/>

[آموزش ساختن یک فروشگاه کامل از روی عکس](https://www.figma.com/design/V7CPtySoLXCFHlXtsd40Fw/eCommerce-Website-%7C-Web-Page-Design-%7C-UI-KIT-%7C-Interior-Landing-Page--Community-?node-id=0-1&p=f&t=qJRTtGtlL6CAoORI-0)

## نحوه خرید دامنه و هاست

خرید هاست از یکی از هاست های هاست ایران، لیمو، پارسپک، نت افراز. نحوه کار با Cpanel نحوه استفاده از Filezilla

## معرفی grid css

CSS Grid یک سیستم لایه‌بندی دو بعدی برای چیدمان عناصر در صفحه است.

|  |
| --- |
| <style>  .container {  display: grid;  grid-template-columns: 100px 100px 100px;  grid-template-rows: 100px 100px;  gap: 10px;  background-color: #f0f0f0;  padding: 10px;  margin-bottom: 30px;  }  .item {  background-color: #4CAF50;  color: white;  padding: 20px;  text-align: center;  border-radius: 5px;  }  </style>  <div class="container">  <div class="item">1</div>  <div class="item">2</div>  <div class="item">3</div>  <div class="item">4</div>  <div class="item">5</div>  <div class="item">6</div>  </div> |

نکات مهم:

- `display: grid` المان را به یک grid container تبدیل می‌کند

- `grid-template-columns` عرض ستون‌ها را تعریف می‌کند

- `grid-template-rows` ارتفاع سطرها را تعریف می‌کند

- `gap` فاصله بین سلول‌های گرید را مشخص می‌کند

grid column

ستون‌های عمودی در سیستم گرید که با `grid-template-columns` تعریف می‌شوند.

|  |
| --- |
| <style>  .columns-container {  display: grid;  grid-template-columns: 1fr 2fr 1fr;  gap: 15px;  background-color: #f8f9fa;  padding: 20px;  margin-bottom: 30px;  }  .column-item {  background-color: #673AB7;  color: white;  padding: 20px;  text-align: center;  border-radius: 8px;  }  .span-2-columns {  grid-column: span 2;  background-color: #E91E63;  }  .start-from-2 {  grid-column: 2 / 4;  background-color: #009688;  }  .full-columns {  grid-column: 1 / -1;  background-color: #FF9800;  }  </style>  <div class="columns-container">  <div class="column-item">ستون 1</div>  <div class="column-item">ستون 2</div>  <div class="column-item">ستون 3</div>    <div class="column-item span-2-columns">2 ستون را گرفته</div>  <div class="column-item">ستون 3</div>    <div class="column-item">ستون 1</div>  <div class="column-item start-from-2">از ستون 2 شروع شده</div>    <div class="column-item full-columns">تمام ستون‌ها</div>  </div> |

نکات مهم Grid Columns:

- `fr` واحد نسبی برای تقسیم فضای باقیمانده

- `span` تعداد ستون‌هایی که المان باید بگیرد

- `grid-column: start / end` موقعیت شروع و پایان

- `1 / -1` از اولین تا آخرین ستون

grid row

سطرهای افقی در سیستم گرید که با `grid-template-rows` تعریف می‌شوند.

|  |
| --- |
| <style>  .rows-container {  display: grid;  grid-template-columns: repeat(3, 1fr);  grid-template-rows: 80px 120px 100px;  gap: 10px;  background-color: #e1f5fe;  padding: 15px;  margin-bottom: 30px;  }  .row-item {  background-color: #3F51B5;  color: white;  padding: 15px;  text-align: center;  border-radius: 6px;  }  .tall-row {  grid-row: span 2;  background-color: #F44336;  }  .custom-row {  grid-row: 2 / 4;  background-color: #4CAF50;  }  .full-rows {  grid-row: 1 / -1;  background-color: #FFC107;  color: black;  }  </style>  <div class="rows-container">  <div class="row-item">سطر 1 - ستون 1</div>  <div class="row-item tall-row">2 سطر گرفته</div>  <div class="row-item">سطر 1 - ستون 3</div>    <div class="row-item">سطر 2 - ستون 1</div>  <div class="row-item">سطر 2 - ستون 2</div>  <div class="row-item custom-row">سطر 2 تا 4 - ستون 3</div>    <div class="row-item full-rows">تمام سطرها - ستون 1</div>  <div class="row-item">سطر 3 - ستون 2</div>  <div class="row-item">سطر 4 - ستون 3</div>  </div> |

نکات مهم Grid Rows:

- `grid-row: span 2` المان دو سطر را می‌گیرد

- `grid-row: 2 / 4` از سطر 2 تا سطر 4

- `1 / -1` از اولین تا آخرین سطر

grid cell

کوچکترین واحد در گرید - تقاطع یک سطر و یک ستون.

|  |
| --- |
| <style>  .cells-container {  display: grid;  grid-template-columns: repeat(4, 1fr);  grid-template-rows: repeat(3, 80px);  gap: 5px;  background-color: #fff3e0;  padding: 10px;  margin-bottom: 30px;  }  .cell {  background-color: #FF9800;  border: 2px solid #F57C00;  padding: 10px;  text-align: center;  display: flex;  align-items: center;  justify-content: center;  font-weight: bold;  }  .specific-cell {  background-color: #8BC34A;  border-color: #689F38;  grid-column: 3;  grid-row: 2;  }  .another-cell {  background-color: #03A9F4;  border-color: #0288D1;  grid-column: 2;  grid-row: 3;  }  </style>  <div class="cells-container">  <div class="cell">(1,1)</div>  <div class="cell">(1,2)</div>  <div class="cell">(1,3)</div>  <div class="cell">(1,4)</div>    <div class="cell">(2,1)</div>  <div class="cell">(2,2)</div>  <div class="cell specific-cell">(2,3) مشخص</div>  <div class="cell">(2,4)</div>    <div class="cell">(3,1)</div>  <div class="cell another-cell">(3,2) مشخص</div>  <div class="cell">(3,3)</div>  <div class="cell">(3,4)</div>  </div> |

نکات مهم Grid Cell:

- هر سلول یک واحد مستقل است

- فقط یک المان می‌تواند در هر سلول قرار گیرد

- آدرس‌دهی با `(row, column)`

grid area

مجموعه‌ای از سلول‌ها که یک ناحیه مستطیلی شکل تشکیل می‌دهند.

|  |
| --- |
| <style>  .areas-container {  display: grid;  grid-template-columns: 1fr 1fr 1fr 1fr;  grid-template-rows: 100px 200px 100px;  grid-template-areas:  "header header header header"  "sidebar content content content"  "footer footer footer footer";  gap: 10px;  background-color: #f5f5f5;  padding: 15px;  margin-bottom: 30px;  }  .area-item {  background-color: #9C27B0;  color: white;  padding: 15px;  text-align: center;  border-radius: 8px;  display: flex;  align-items: center;  justify-content: center;  }  .header-area {  grid-area: header;  background-color: #2196F3;  }  .sidebar-area {  grid-area: sidebar;  background-color: #4CAF50;  }  .content-area {  grid-area: content;  background-color: #FF9800;  }  .footer-area {  grid-area: footer;  background-color: #F44336;  }  .custom-area-container {  display: grid;  grid-template-columns: repeat(4, 1fr);  grid-template-rows: repeat(4, 80px);  gap: 10px;  background-color: #e8f5e8;  padding: 15px;  }  .custom-area {  grid-area: 2 / 2 / 4 / 4;  background-color: #607D8B;  color: white;  display: flex;  align-items: center;  justify-content: center;  border-radius: 8px;  }  </style>  <div class="areas-container">  <div class="area-item header-area">ناحیه سربرگ</div>  <div class="area-item sidebar-area">ناحیه کناری</div>  <div class="area-item content-area">ناحیه محتوا</div>  <div class="area-item footer-area">ناحیه پاورقی</div>  </div>  <div class="custom-area-container">  <div class="custom-area">ناحیه سفارشی (2,2 تا 4,4)</div>  </div> |

نکات مهم Grid Area:

- `grid-template-areas` برای تعریف layout با نام

- `grid-area: name` برای اختصاص المان به ناحیه

- `grid-area: row-start / col-start / row-end / col-end` برای تعریف با خطوط

- نقطه (`.`) برای سلول‌های خالی در `grid-template-areas`

grid line

خطوط جداکننده بین ستون‌ها و سطرها هستند. این خطوط از عدد 1 شماره‌گذاری می‌شوند.

|  |
| --- |
| <style>  .lines-container {  display: grid;  grid-template-columns: [col1-start] 100px [col2-start] 100px [col3-start] 100px [col3-end];  grid-template-rows: [row1-start] 100px [row2-start] 100px [row2-end];  gap: 10px;  background-color: #e3f2fd;  padding: 10px;  margin-bottom: 30px;  }  .line-item {  background-color: #2196F3;  color: white;  padding: 20px;  text-align: center;  border-radius: 5px;  }  .line-item-1 {  grid-column-start: col1-start;  grid-column-end: col3-start;  background-color: #FF5722;  }  .line-item-4 {  grid-row-start: row2-start;  grid-row-end: row2-end;  background-color: #4CAF50;  }  </style>  <div class="lines-container">  <div class="line-item line-item-1">خط ستون 1 تا 3</div>  <div class="line-item">2</div>  <div class="line-item">3</div>  <div class="line-item line-item-4">خط سطر 2 تا 3</div>  <div class="line-item">5</div>  <div class="line-item">6</div>  </div> |

نکات مهم Grid Lines:

- خطوط از 1 شروع به شمارش می‌کنند

- می‌توان به خطوط نام داد: `[line-name]`

- `grid-column-start` و `grid-column-end` موقعیت شروع و پایان در ستون‌ها

- `grid-row-start` و `grid-row-end` موقعیت شروع و پایان در سطرها

معرفی property های grid

grid-template-columns

تعریف ستون‌های گرید و عرض آنها

|  |
| --- |
| <style>  .container-1 {  display: grid;  grid-template-columns: 100px 200px 150px;  gap: 10px;  background-color: #f0f0f0;  padding: 10px;  margin-bottom: 20px;  }  .container-2 {  display: grid;  grid-template-columns: 1fr 2fr 1fr;  gap: 10px;  background-color: #e8f4f8;  padding: 10px;  margin-bottom: 20px;  }  .item {  background-color: #4CAF50;  color: white;  padding: 20px;  text-align: center;  border-radius: 5px;  }  </style>  <div class="container-1">  <div class="item">100px</div>  <div class="item">200px</div>  <div class="item">150px</div>  </div>  <div class="container-2">  <div class="item">1fr</div>  <div class="item">2fr</div>  <div class="item">1fr</div>  </div> |

مقادیر:

px, %, em - واحدهای ثابت

fr - واحد نسبی (fraction)

auto - اندازه خودکار

repeat(3, 1fr) - تکرار الگو

minmax(100px, 1fr) - محدوده اندازه

grid-gap- - grid-column-gap- grid-row-gap

تعریف فاصله بین سلول‌های گرید

|  |
| --- |
| <style>  .container-4 {  display: grid;  grid-template-columns: 1fr 1fr 1fr;  grid-template-rows: 100px 100px;  gap: 20px 10px; /\* row-gap column-gap \*/  background-color: #f8f9fa;  padding: 15px;  margin-bottom: 20px;  }  .container-5 {  display: grid;  grid-template-columns: 1fr 1fr 1fr;  grid-template-rows: 100px 100px;  grid-row-gap: 30px;  grid-column-gap: 15px;  background-color: #e8f5e8;  padding: 15px;  margin-bottom: 20px;  }  .item {  background-color: #2196F3;  color: white;  padding: 20px;  text-align: center;  border-radius: 5px;  }  </style>  <div class="container-4">  <div class="item">gap: 20px 10px</div>  <div class="item">ستون 2</div>  <div class="item">ستون 3</div>  <div class="item">سطر 2</div>  <div class="item">سطر 2</div>  <div class="item">سطر 2</div>  </div>  <div class="container-5">  <div class="item">row-gap: 30px</div>  <div class="item">ستون 2</div>  <div class="item">ستون 3</div>  <div class="item">column-gap: 15px</div>  <div class="item">ستون 2</div>  <div class="item">ستون 3</div>  </div> |

نکته: grid-gap نام قدیمی است، از gap استفاده کنید

grid-template-rows

تعریف سطرهای گرید و ارتفاع آنها

|  |
| --- |
| <style>  .container-3 {  display: grid;  grid-template-columns: 1fr 1fr 1fr;  grid-template-rows: 80px 120px 100px;  gap: 10px;  background-color: #fff3e0;  padding: 10px;  margin-bottom: 20px;  }  .item {  background-color: #FF9800;  color: white;  padding: 15px;  text-align: center;  border-radius: 5px;  }  </style>  <div class="container-3">  <div class="item">سطر 1: 80px</div>  <div class="item">سطر 1: 80px</div>  <div class="item">سطر 1: 80px</div>  <div class="item">سطر 2: 120px</div>  <div class="item">سطر 2: 120px</div>  <div class="item">سطر 2: 120px</div>  <div class="item">سطر 3: 100px</div>  <div class="item">سطر 3: 100px</div>  <div class="item">سطر 3: 100px</div>  </div> |

grid-column - grid-row

تعریف موقعیت المان در ستون‌ها و سطرها

|  |
| --- |
| <style>  .container-6 {  display: grid;  grid-template-columns: 1fr 1fr 1fr 1fr;  grid-template-rows: 100px 100px 100px;  gap: 10px;  background-color: #f0f0f0;  padding: 15px;  margin-bottom: 20px;  }  .item {  background-color: #9C27B0;  color: white;  padding: 15px;  text-align: center;  border-radius: 5px;  }  .item-1 {  grid-column: 1 / 3; /\* از ستون 1 تا 3 \*/  background-color: #FF5722;  }  .item-2 {  grid-row: 1 / 3; /\* از سطر 1 تا 3 \*/  background-color: #4CAF50;  }  .item-3 {  grid-column: span 2; /\* 2 ستون بگیر \*/  grid-row: span 2; /\* 2 سطر بگیر \*/  background-color: #2196F3;  }  </style>  <div class="container-6">  <div class="item item-1">grid-column: 1 / 3</div>  <div class="item item-2">grid-row: 1 / 3</div>  <div class="item item-3">span 2 ستون و 2 سطر</div>  <div class="item">4</div>  <div class="item">5</div>  <div class="item">6</div>  <div class="item">7</div>  <div class="item">8</div>  </div> |

grid-column-start-grid-column-end-grid-row-start-grid-row-end

تعریف دقیق شروع و پایان المان

|  |
| --- |
| <style>  .container-7 {  display: grid;  grid-template-columns: [col1] 1fr [col2] 1fr [col3] 1fr [col4];  grid-template-rows: [row1] 100px [row2] 100px [row3] 100px [row4];  gap: 10px;  background-color: #e3f2fd;  padding: 15px;  margin-bottom: 20px;  }  .item {  background-color: #607D8B;  color: white;  padding: 15px;  text-align: center;  border-radius: 5px;  }  .positioned-item {  grid-column-start: col1;  grid-column-end: col3;  grid-row-start: row2;  grid-row-end: row4;  background-color: #E91E63;  }  </style>  <div class="container-7">  <div class="item">1</div>  <div class="item">2</div>  <div class="item">3</div>  <div class="item">4</div>  <div class="item positioned-item">موقعیت دقیق</div>  <div class="item">6</div>  <div class="item">7</div>  <div class="item">8</div>  <div class="item">9</div>  </div> |

grid-template-areas- grid-area

تعریف layout با نام‌های معنادار

|  |
| --- |
| <style>  .container-8 {  display: grid;  grid-template-columns: 1fr 2fr 1fr;  grid-template-rows: 80px 1fr 80px;  grid-template-areas:  "header header header"  "sidebar main aside"  "footer footer footer";  gap: 15px;  background-color: #f5f5f5;  padding: 20px;  margin-bottom: 20px;  height: 400px;  }  .area-item {  padding: 20px;  text-align: center;  border-radius: 8px;  color: white;  }  .header {  grid-area: header;  background-color: #2196F3;  }  .sidebar {  grid-area: sidebar;  background-color: #4CAF50;  }  .main {  grid-area: main;  background-color: #FF9800;  }  .aside {  grid-area: aside;  background-color: #9C27B0;  }  .footer {  grid-area: footer;  background-color: #F44336;  }  </style>  <div class="container-8">  <div class="area-item header">Header</div>  <div class="area-item sidebar">Sidebar</div>  <div class="area-item main">Main Content</div>  <div class="area-item aside">Aside</div>  <div class="area-item footer">Footer</div>  </div> |

justify-content- align-content

ترازبندی کل گرید در container

|  |
| --- |
| <style>  .container-9 {  display: grid;  grid-template-columns: 100px 100px;  grid-template-rows: 100px 100px;  gap: 10px;  justify-content: center;  align-content: center;  background-color: #fff8e1;  padding: 20px;  margin-bottom: 20px;  height: 400px;  border: 2px dashed #FFC107;  }  .justify-item {  background-color: #FF9800;  color: white;  padding: 20px;  text-align: center;  border-radius: 5px;  }  </style>  <div class="container-9">  <div class="justify-item">1</div>  <div class="justify-item">2</div>  <div class="justify-item">3</div>  <div class="justify-item">4</div>  </div> |

مقادیر justify-content و align-content:

start - ابتدا

end - انتها

center - وسط

stretch - کشیده (پیش‌فرض)

space-between - فاصله مساوی بین

space-around - فاصله مساوی دور

space-evenly - فاصله کاملاً مساوی

justify-items- align-items

ترازبندی محتوای داخل سلول‌ها

|  |
| --- |
| <style>  .container-10 {  display: grid;  grid-template-columns: 1fr 1fr 1fr;  grid-template-rows: 150px;  gap: 10px;  justify-items: center;  align-items: end;  background-color: #e8f5e8;  padding: 20px;  margin-bottom: 20px;  height: 200px;  }  .items-item {  background-color: #4CAF50;  color: white;  padding: 10px;  border-radius: 5px;  width: 80px;  }  </style>  <div class="container-10">  <div class="items-item">محتوا 1</div>  <div class="items-item">محتوا 2</div>  <div class="items-item">محتوا 3</div>  </div> |

مقادیر justify-items و align-items:

start - ابتدا

end - انتها

center - وسط

stretch - کشیده (پیش‌فرض)

justify-self - align-self

ترازبندی فردی یک المان خاص

|  |
| --- |
| <style>  .container-11 {  display: grid;  grid-template-columns: 1fr 1fr 1fr;  grid-template-rows: 150px;  gap: 10px;  background-color: #fce4ec;  padding: 20px;  margin-bottom: 20px;  height: 200px;  }  .self-item {  background-color: #E91E63;  color: white;  padding: 10px;  border-radius: 5px;  width: 80px;  }  .self-1 {  justify-self: start;  align-self: start;  }  .self-2 {  justify-self: center;  align-self: center;  }  .self-3 {  justify-self: end;  align-self: end;  }  </style>  <div class="container-11">  <div class="self-item self-1">start</div>  <div class="self-item self-2">center</div>  <div class="self-item self-3">end</div>  </div> |

مقادیر justify-self و align-self:

start - ابتدا

end - انتها

center - وسط

stretch - کشیده (پیش‌فرض)

auto - از مقدار container پیروی کند

# آموزش جاوا اسکریپت Java script

## معرفی زبان JavaScript

یک زبان برنامه‌نویسی سطح بالا، پویا و مفسری است که عمدتاً برای توسعه وب استفاده می‌شود.

ویژگی‌های کلیدی:

- مفسری (Interpreted): نیازی به کامپایل ندارد

- پویا (Dynamic): تایپ داده‌ای پویا

- شیءگرا (Object-Oriented): مبتنی بر پروتوتایپ

- چندپارادایمی: پشتیبانی از برنامه‌نویسی تابعی، رویدادمحور و غیره

- کراس-پلتفرم: اجرا در تمام مرورگرها و پلتفرم‌ها

نمونه کد ساده:

|  |
| --- |
| // Hello World در JavaScript  console.log("Hello, World!");  // تعریف متغیر  let name = "John";  const age = 25;  // تابع  function greet(person) {  return `Hello, ${person}!`;  } |

## معرفی کاربرد های زبان JavaScript

۱. توسعه فرانت‌اند (Frontend)

- تعامل با صفحات وب

- اعتبارسنجی فرم‌ها

- انیمیشن و افکت‌های بصری

- برنامه‌های تک صفحه‌ای (SPA)

۲. توسعه بک‌اند (Backend)

- سرور با Node.js

- API Development

- Real-time Applications

۳. توسعه اپلیکیشن موبایل

- React Native

- Ionic

-NativeScript

۴. توسعه دسکتاپ

- Electron (VS Code, Slack, Discord)

- NW.js

۵. سایر کاربردها

- بازی‌سازی (با Three.js, Phaser)

- اینترنت اشیاء (IoT)

- هوش مصنوعی و یادگیری ماشین

## معرفی فریمورک ها و کتابخانه های این زبان

۱. React (کتابخانه)

|  |
| --- |
| // نمونه کامپوننت React  import React, { useState } from 'react';  function Counter() {  const [count, setCount] = useState(0);    return (  <div>  <p>Count: {count}</p>  <button onClick={() => setCount(count + 1)}>  Increment  </button>  </div>  );  } |

۲. Vue.js

|  |
| --- |
| // نمونه کامپوننت Vue  const app = Vue.createApp({  data() {  return {  message: 'Hello Vue!'  }  }  }); |

3. Angular

|  |
| --- |
| // نمونه کامپوننت Angular  import { Component } from '@angular/core';  @Component({  selector: 'app-root',  template: '<h1>Hello {{name}}!</h1>'  })  export class AppComponent {  name = 'Angular';  } |

فریمورک‌های بک‌اند:

۱. Node.js + Express

|  |
| --- |
| // نمونه سرور Express  const express = require('express');  const app = express();  app.get('/', (req, res) => {  res.send('Hello World!');  });  app.listen(3000, () => {  console.log('Server running on port 3000');  }); |

#### ۲. Next.js (React Framework)

- رندرینگ سمت سرور

- تولید سایت استاتیک

- مسیریابی خودکار

۳. Nuxt.js (Vue Framework)

- فریمورک همه‌کاره برای Vue

کتابخانه‌های کاربردی:

۱. State Management

- Redux (برای React)

- Vuex/Pinia (برای Vue)

- NgRx (برای Angular)

۲. UI Libraries

- Material-UI (React)

- Vuetify (Vue)

- Ant Design

۳. Testing

- Jest

- Mocha

- Cypress

۴. Build Tools

- Webpack

- Vite

- Parcel

## بررسی تفاوت های زبان های مفسری و کامپایلری

زبان‌های مفسری (Interpreted Languages)

ویژگی‌ها:

- \*\*اجرای خط به خط\*\* کد منبع

- \*\*نیاز به مفسر\*\* در زمان اجرا

- \*\*عدم تولید فایل اجرایی\*\* مستقل

مثال‌ها:

- \*\*JavaScript\*\* - Python - PHP - Ruby

مزایا:

- \*\*توسعه سریعتر\*\* - تست فوری

- \*\*قابلیت اجرای کراس-پلتفرم\*\*

- \*\*عیب‌یابی آسان‌تر\*\*

معایب:

- \*\*سرعت اجرای کمتر\*\*

- \*\*وابستگی به مفسر\*\*

زبان‌های کامپایلری (Compiled Languages)

ویژگی‌ها:

- \*\*تبدیل کامل\*\* به کد ماشین قبل از اجرا

- \*\*تولید فایل اجرایی\*\* مستقل

- \*\*بهینه‌سازی در زمان کامپایل\*\*

مثال‌ها:

- \*\*C/C++\*\* - Java - Go - Rust

مزایا:

- \*\*سرعت اجرای بسیار بالا\*\*

- \*\*امنیت بیشتر\*\*

- \*\*اجرای مستقل\*\*

معایب:

- \*\*زمان توسعه طولانی‌تر\*\*

- \*\*وابستگی به پلتفرم\*\*

## معرفی تگ script

ساختار پایه:

|  |
| --- |
| <script>  // کد JavaScript در اینجا قرار می‌گیرد  console.log("Hello from script tag!");  </script> |

ویژگی‌های مهم:

- \*\*`src`\*\* : بارگیری فایل خارجی

- \*\*`async`\*\* : بارگیری و اجرای ناهمگام

- \*\*`defer`\*\* : بارگیری ناهمگام، اجرا پس از پارس HTML

- \*\*`type`\*\* : نوع اسکریپت (مثل `module`)

## بررسی قرار دادن تگ script در جای های مختلف

روش‌های مختلف قرار دادن تگ `<script>`

۱. داخل `<head>` - روش‌های مدیریت لودینگ

روش مشکل‌ساز (بدون مدیریت لودینگ):

|  |
| --- |
| <head>  <script>  // ❌ این کار درست نیست - DOM هنوز لود نشده  const element = document.getElementById("myElement");  console.log(element); // null  </script>  </head>  <body>  <div id="myElement">محتوای من</div>  </body> |

روش‌های صحیح برای اسکریپت در `<head>`:

روش ۱: استفاده از `DOMContentLoaded`

|  |
| --- |
| <head>  <script>  // ✅ منتظر می‌مانیم تا DOM کاملاً لود شود  document.addEventListener('DOMContentLoaded', function() {  console.log('DOM کاملاً لود شد!');    // حالا می‌توانیم به عناصر دسترسی داشته باشیم  const element = document.getElementById("myElement");  console.log(element.textContent); // "محتوای من"    // اجرای توابع اصلی  initializeApp();  });    function initializeApp() {  console.log('برنامه راه‌اندازی شد');  }  </script>  </head>  <body>  <div id="myElement">محتوای من</div>  </body> |

\*\*روش ۲: استفاده از `window.onload`\*\*

|  |
| --- |
| <head>  <script>  // ✅ منتظر می‌مانیم تا تمام منابع (تصاویر، استایل‌ها) لود شوند  window.onload = function() {  console.log('تمامی صفحه و منابع لود شدند');    const element = document.getElementById("myElement");  const images = document.querySelectorAll('img');  console.log(`تعداد تصاویر لود شده: ${images.length}`);  };  </script>  </head> |

\*\*روش ۳: استفاده از `defer` (توصیه شده)\*\*

|  |
| --- |
| <head>  <!-- ✅ اسکریپت بلافاصله بارگیری می‌شود اما پس از لود DOM اجرا می‌شود -->  <script defer src="app.js"></script>  <script defer src="utils.js"></script>  </head>  <body>  <div id="myElement">محتوای من</div>  </body> |

\*\*محتوای فایل `app.js` با `defer`:\*\*

|  |
| --- |
| // با defer، این کد پس از لود کامل DOM اجرا می‌شود  console.log('این اسکریپت پس از لود DOM اجرا می‌شود');  const element = document.getElementById("myElement");  if (element) {  element.style.color = 'blue';  console.log('عنصر با موفقیت پیدا و تغییر کرد');  } |

۲. انتهای `<body>` (ساده‌ترین روش)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>صفحه نمونه</title>  </head>  <body>  <div id="myElement">محتوای من</div>  <button id="myButton">کلیک کن</button>    <script>  // ✅ DOM قطعاً لود شده است  const element = document.getElementById("myElement");  const button = document.getElementById("myButton");    element.style.backgroundColor = 'yellow';  button.addEventListener('click', function() {  alert('دکمه کلیک شد!');  });    console.log('همه عناصر در دسترس هستند');  </script>  </body>  </html> |

۳. استفاده از `async` برای اسکریپت‌های مستقل

|  |
| --- |
| <head>  <!-- ✅ برای اسکریپت‌های مستقل مانند آنالیتیکس -->  <script async src="analytics.js"></script>  </head> |

## معرفی اتریبیوت های defer و async و تفاوت آن ها

\*\*defer\*\* یک ویژگی در تگ script است که باعث می‌شود:

- اسکریپت در پس‌زمینه بارگیری شود

- اجرای اسکریپت بعد از لود کامل صفحه انجام شود

- ترتیب اجرای اسکریپت‌ها حفظ می‌شود

مثال:

|  |
| --- |
| <head>  <script defer src="script1.js"></script>  <script defer src="script2.js"></script>  </head> |

نتیجه:

- هر دو اسکریپت موازی بارگیری می‌شوند

- script1 حتماً قبل از script2 اجرا می‌شود

- بعد از کامل شدن HTML اجرا می‌شوند

async چیست؟

async\*\* یک ویژگی در تگ script است که:

- اسکریپت در پس‌زمینه بارگیری شود

- به محض آماده شدن اجرا می‌شود

- ترتیب اجرا مهم نیست

مثال:

|  |
| --- |
| <head>  <script async src="analytics.js"></script>  <script async src="ads.js"></script>  </head> |

\*\*نتیجه:\*\*

- هر کدام زودتر بارگیری شود، زودتر اجرا می‌شود

- ممکن است ads.js قبل از analytics.js اجرا شود

مقایسه defer و async

|  |  |  |
| --- | --- | --- |
| ویژگی | defer | async |
| ترتیب اجرا | ✅ حفظ می‌شود | ❌ حفظ نمی‌شود |
| زمان اجرا | بعد از لود HTML | به محض بارگیری |
| مناسب برای | اسکریپت‌های اصلی | اسکریپت‌های مستقل |

## نحوه نوشتن JavaScript به صورت external

مرحله ۱: ایجاد فایل JavaScript

یک فایل جدید با پسوند `.js` ایجاد کنید، مثلاً

|  |
| --- |
| // این یک فایل JavaScript خارجی است  function showMessage() {  alert("سلام! این از فایل خارجی است");  }  function changeBackground() {  document.body.style.backgroundColor = "lightblue";  }  console.log("فایل اسکریپت بارگیری شد"); |

### مرحله ۲: اتصال به HTML

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>اتصال فایل خارجی</title>  </head>  <body>  <h1>صفحه نمونه</h1>  <button onclick="showMessage()">کلیک کن</button>  <button onclick="changeBackground()">تغییر رنگ</button>  <!-- اتصال فایل JavaScript -->  <script src="script.js"></script>  </body>  </html> |

## چاپ کردن دیتا با java script

Document.write

توضیح:

- مستقیم محتوا را در صفحه HTML می‌نویسد

- فقط برای تست و یادگیری مناسب است

مثال‌ها:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <script>  // نوشتن متن ساده  document.write("سلام دنیا!");    // نوشتن HTML  document.write("<h1>عنوان بزرگ</h1>");  document.write("<p style='color: red;'>این یک پاراگراف قرمز است</p>");    // ترکیب با متغیرها  let name = "آرش";  let age = 25;  document.write("<p>نام: " + name + "</p>");  document.write("<p>سن: " + age + " سال</p>");  </script>  </body>  </html> |

نکات مهم:

- ❌ \*\*استفاده نکنید\*\* در پروژه‌های واقعی

- ❌ محتوای قبلی صفحه را پاک می‌کند اگر بعد از لود صفحه استفاده شود

Document.writeln

توضیح:

- دقیقاً مثل `document.write` کار می‌کند

- فقط یک \*\*خط جدید\*\* (`\n`) در انتها اضافه می‌کند

مثال:

|  |
| --- |
| <script>  // تفاوت بین write و writeln  document.write("خط اول");  document.write("خط دوم");  // نتیجه: خط اولخط دوم    document.writeln("خط اول");  document.writeln("خط دوم");  // نتیجه: خط اول  // خط دوم  </script> |

نکته:

- در HTML، خط جدید معمولاً نمایش داده نمی‌شود

- برای دیدن اثر آن باید از تگ `<pre>` استفاده کنید

Alert

توضیح:

- نمایش پیام در یک پنجره pop-up

- کاربر باید OK کند تا ادامه دهد

مثال‌ها:

|  |
| --- |
| // پیام ساده  alert("سلام! خوش آمدید");  // پیام با متغیر  let userName = "محمد";  alert("سلام " + userName + "! عزیز");  // هشدار خطا  alert("⚠️ خطا: فایل پیدا نشد");  // ترکیب با محاسبات  let result = 10 + 5;  alert("نتیجه محاسبه: " + result); |

ویژگی‌ها:

- ❌ \*\*مسدود کننده\*\* - کد بعدی اجرا نمی‌شود تا کاربر OK کند

- ✅ ساده برای دیباگ و هشدارهای فوری

Console.log

### توضیح:

- نمایش اطلاعات در کنسول مرورگر

- \*\*بهترین روش\*\* برای دیباگ و تست

### روش باز کردن کنسول:

- \*\*Windows/Linux\*\*: `F12` یا `Ctrl+Shift+I`

- \*\*Mac\*\*: `Cmd+Option+I`

مثال‌های کامل:

|  |
| --- |
| // چاپ متن ساده  console.log("این یک پیام ساده است");  // چاپ متغیرها  let name = "سارا";  let age = 30;  console.log("نام:", name);  console.log("سن:", age);  // چاپ چند متغیر با هم  console.log("اطلاعات کاربر:", name, age);  // چاپ اشیاء  let user = {  firstName: "علی",  lastName: "محمدی",  city: "تهران"  };  console.log("شیء کاربر:", user);  // چاپ آرایه  let colors = ["قرمز", "سبز", "آبی"];  console.log("آرایه رنگ‌ها:", colors); |

انواع مختلف console:

|  |
| --- |
| // انواع مختلف متدهای کنسول  console.error("این یک خطاست!"); // ❌ قرمز  console.warn("این یک هشدار است!"); // ⚠️ زرد  console.info("این یک اطلاعات است!"); // ℹ️ آبی  // جدول برای نمایش داده‌های ساختاریافته  let students = [  { name: "رضا", grade: 18 },  { name: "فاطمه", grade: 20 },  { name: "امیر", grade: 16 }  ];  console.table(students); |

مثال کاربردی کامل

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>نمایش داده در JavaScript</title>  </head>  <body>  <h1>روش‌های مختلف نمایش داده</h1>    <button onclick="showAlert()">نمایش Alert</button>  <button onclick="showInConsole()">نمایش در Console</button>    <script>  function showAlert() {  let userName = "نازنین";  let score = 95;  alert("کاربر " + userName + " امتیاز " + score + " گرفت!");  }    function showInConsole() {  // اطلاعات مختلف در کنسول  console.log("=== اطلاعات سیستم ===");    let user = {  name: "کامیاب",  age: 28,  email: "kamiab@example.com"  };    let products = ["لپ‌تاپ", "موبایل", "تبلت"];    console.log("کاربر:", user);  console.log("محصولات:", products);  console.warn("توجه: این فقط برای تست است");  }    // نمایش در کنسول هنگام لود صفحه  console.log("صفحه با موفقیت لود شد!");  </script>  </body>  </html> |

مقایسه روش‌ها

|  |  |  |  |
| --- | --- | --- | --- |
| روش | کاربرد | مخاطب | مناسب برای |
| \*\*`document.write`\*\* | نمایش در صفحه | همه کاربران | ❌ فقط آموزش |
| \*\*`alert`\*\* | پیام فوری | کاربر نهایی | هشدارهای مهم |
| \*\*`console.log`\*\* | دیباگ و لاگ | توسعه‌دهنده | ✅ پروژه‌های واقعی |

توصیه‌های مهم

1. \*\*همیشه از `console.log` برای دیباگ استفاده کنید\*\*

2. \*\*از `alert` فقط برای هشدارهای بسیار مهم استفاده کنید\*\*

3. \*\*هرگز از `document.write` در پروژه‌های واقعی استفاده نکنید\*\*

4. \*\*کنسول مرورگر بهترین دوست یک توسعه‌دهنده است!\*\*

|  |
| --- |
| // مثال نهایی از استفاده درست  let temperature = 25;  let city = "مشهد";  console.log(`دما در ${city}: ${temperature} درجه`);  // ✅ بهترین روش برای توسعه‌دهندگان  // فقط در صورت نیاز واقعی:  if (temperature > 40) {  alert(`⚠️ هشدار: دمای ${city} بسیار بالا است!`);  } |

## نحوه کامنت گذاری

کامنت‌ها برای توضیح کد و غیرفعال کردن موقت بخش‌هایی از کد استفاده می‌شوند:

|  |
| --- |
| // این یک کامنت تک خطی است  /\*  این یک کامنت چند خطی است  که می‌تواند چندین خط را پوشش دهد  \*/  /\*\*  \* این کامنت برای مستندسازی توابع استفاده می‌شود  \* @param {string} name - نام کاربر  \* @returns {string} پیام خوشامدگویی  \*/  function greet(name) {  return "Hello, " + name;  } |

## معرفی متغیر ها

معرفی متغیرها

متغیرها برای ذخیره و مدیریت داده‌ها در برنامه استفاده می‌شوند.

روش‌های تعریف متغیر:

|  |
| --- |
| // var (قدیمی - توصیه نمی‌شود)  var oldVariable = "مقدار قدیمی";  // let (متغیر قابل تغییر)  let changeableVariable = "می‌توان تغییر داد";  // const (ثابت - غیرقابل تغییر)  const constantVariable = "ثابت و غیرقابل تغییر"; |

بررسی نحوه نام گذاری متغیر ها

قوانین نام‌گذاری:

باید با حرف، $ یا \_ شروع شود

نمی‌تواند با عدد شروع شود

حساس به حروف بزرگ و کوچک (case-sensitive)

نمی‌تواند از کلمات رزرو شده استفاده کند

سبک‌های نام‌گذاری:

|  |
| --- |
| // camelCase (توصیه شده در JavaScript)  let firstName = "John";  let userAge = 25;  let isActive = true;  // snake\_case  let last\_name = "Doe";  let user\_age = 25;  // PascalCase (برای کلاس‌ها)  class UserProfile {  constructor(name) {  this.name = name;  }  }  // UPPER\_CASE (برای ثابت‌ها)  const MAX\_USERS = 100;  const API\_KEY = "abc123"; |

نحوه کار با متغیر ها

مثال ساده و کامل:

|  |
| --- |
| // تعریف متغیرها  let userName = "علی";  const userBirthYear = 1995;  let userScore = 0;  // کامنت توضیحی  // محاسبه سن کاربر بر اساس سال تولد  const currentYear = 2024;  let userAge = currentYear - userBirthYear;  // نمایش اطلاعات  console.log("نام کاربر: " + userName);  console.log("سن کاربر: " + userAge + " سال");  // تغییر مقدار متغیر  userScore = 85;  console.log("امتیاز کاربر: " + userScore);  // افزایش امتیاز  userScore += 15;  console.log("امتیاز جدید: " + userScore);  // کار با متغیرهای boolean  let isPremiumUser = true;  console.log("کارگر ویژه: " + isPremiumUser);  // کار با آرایه  let userHobbies = ["خواندن", "نوشتن", "ورزش"];  console.log("علاقه‌مندی‌ها: " + userHobbies);  // کار با object  let userProfile = {  name: userName,  age: userAge,  isActive: true,  hobbies: userHobbies  };  console.log("پروفایل کاربر:");  console.log(userProfile); |

خروجی کد بالا:

|  |
| --- |
| نام کاربر: علی  سن کاربر: 29 سال  امتیاز کاربر: 85  امتیاز جدید: 100  کارگر ویژه: true  علاقه‌مندی‌ها: خواندن,نوشتن,ورزش  پروفایل کاربر:  {  name: 'علی',  age: 29,  isActive: true,  hobbies: ['خواندن', 'نوشتن', 'ورزش']  } |

نکات مهم:

همیشه از const استفاده کنید مگر اینکه نیاز به تغییر مقدار داشته باشید

از نام‌های توصیفی برای متغیرها استفاده کنید

از camelCase برای نام‌گذاری استفاده کنید

متغیرها را در ابتدای scope تعریف کنید

## معرفی var و let و const

var (قدیمی - توصیه نمی‌شود)

|  |
| --- |
| // تعریف با var  var name = "علی";  var age = 25;  // مشکلات var  console.log(x); // undefined (Hoisting)  var x = 10;  var score = 100;  var score = 200; // قابل تعریف مجدد  console.log(score); // 200  // مشکل scope  if (true) {  var y = 20;  }  console.log(y); // 20 (دسترسی خارج از بلوک) |

let (متغیر قابل تغییر)

|  |
| --- |
| // تعریف با let  let userName = "محمد";  let userAge = 30;  // قابل تغییر اما قابل تعریف مجدد نیست  userName = "رضا"; // ✅  // let userName = "احمد"; // ❌ Error  // scope بلوکی  if (true) {  let z = 30;  console.log(z); // 30  }  // console.log(z); // ❌ Error  // Hoisting متفاوت  // console.log(temp); // ❌ Error  let temp = "موقت"; |

const (ثابت - غیرقابل تغییر)

|  |
| --- |
| // تعریف با const  const PI = 3.14159;  const APP\_NAME = "MyApp";  // غیرقابل تغییر  // PI = 3.14; // ❌ Error  // const APP\_NAME = "NewApp"; // ❌ Error  // نکته: برای objectها و آرایه‌ها، reference ثابت می‌ماند  const person = {  name: "علی",  age: 25  };  person.age = 26; // ✅ (تغییر property مجاز)  // person = {}; // ❌ (تغییر reference ممنوع)  const numbers = [1, 2, 3];  numbers.push(4); // ✅  // numbers = [5, 6, 7]; // ❌ |

## معرفی انواع داده

Primitive Types (انواع اولیه)

|  |
| --- |
| // String  let name = "علی رضایی";  let message = 'سلام جهان';  let template = `Hello ${name}`;  // Number  let age = 25;  let price = 99.99;  let temperature = -10;  // Boolean  let isActive = true;  let isCompleted = false;  // Undefined  let undefinedVar;  console.log(undefinedVar); // undefined  // Null  let emptyValue = null;  // Symbol (ایجاد مقدار یکتا)  let id = Symbol("id");  let id2 = Symbol("id");  console.log(id === id2); // false  // BigInt (اعداد بزرگ)  let bigNumber = 1234567890123456789012345678901234567890n; |

Reference Types (انواع مرجع)

|  |
| --- |
| // Object  let person = {  firstName: "علی",  lastName: "محمدی",  age: 30,  isStudent: false  };  // Array  let colors = ["قرمز", "سبز", "آبی"];  let numbers = [1, 2, 3, 4, 5];  // Function  function greet(name) {  return `سلام ${name}`;  }  // Date  let now = new Date(); |

## معرفی typeof

نحوه استفاده از typeof

|  |
| --- |
| // بررسی انواع داده‌های اولیه  console.log(typeof "Hello"); // "string"  console.log(typeof 42); // "number"  console.log(typeof 3.14); // "number"  console.log(typeof true); // "boolean"  console.log(typeof undefined); // "undefined"  console.log(typeof null); // "object" (⚠️ این یک باگ تاریخی است!)  console.log(typeof Symbol("id")); // "symbol"  console.log(typeof 123n); // "bigint"  // بررسی انواع مرجع  console.log(typeof {name: "Ali"}); // "object"  console.log(typeof [1, 2, 3]); // "object" (⚠️ آرایه هم object نشان داده می‌شود)  console.log(typeof function() {}); // "function"  console.log(typeof new Date()); // "object" |

مثال کاربردی از typeof

|  |
| --- |
| // تابع برای بررسی نوع داده  function checkType(value) {  const type = typeof value;  console.log(`مقدار: ${value}, نوع: ${type}`);  return type;  }  // تست با انواع مختلف  checkType("سلام"); // string  checkType(42); // number  checkType(true); // boolean  checkType(undefined); // undefined  checkType(null); // object  checkType([1, 2, 3]); // object  checkType({name: "Ali"}); // object  checkType(function() {}); // function  // تشخیص آرایه  function isArray(value) {  return Array.isArray(value);  }  console.log(isArray([1, 2, 3])); // true  console.log(isArray({})); // false |

مثال جامع و کاربردی

|  |
| --- |
| // برنامه مدیریت کاربران  const users = [];  function addUser(name, age, isActive) {  // بررسی نوع داده‌های ورودی  if (typeof name !== "string") {  console.error("نام باید رشته باشد");  return;  }    if (typeof age !== "number" || age < 0) {  console.error("سن باید عدد مثبت باشد");  return;  }    if (typeof isActive !== "boolean") {  console.error("وضعیت فعال باید boolean باشد");  return;  }    const newUser = {  id: Symbol("user"),  name: name,  age: age,  isActive: isActive,  createdAt: new Date()  };    users.push(newUser);  console.log(`کاربر ${name} اضافه شد`);  }  // استفاده از تابع  addUser("علی", 25, true); // ✅  addUser("رضا", "30", false); // ❌ خطا  addUser("محمد", -5, true); // ❌ خطا  // نمایش اطلاعات  console.log("لیست کاربران:", users);  console.log("تعداد کاربران:", users.length);  console.log("نوع متغیر users:", typeof users); // object  console.log("آیا users آرایه است؟", Array.isArray(users)); // true |

خلاصه تفاوت‌های var, let, const:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | var | let | const |
| قابل تعریف مجدد | ✅ | ❌ | ❌ |
| قابل تغییر | ✅ | ✅ | ❌ |
| Scope | تابع | بلوک | بلوک |
| Hoisting | ✅ | ❌ | ❌ |

نکات مهم:

همیشه از const استفاده کنید مگر نیاز به تغییر داشته باشید

از let برای متغیرهای قابل تغییر استفاده کنید

از var استفاده نکنید

typeof null برابر "object" است (باگ تاریخی)

برای تشخیص آرایه از Array.isArray() استفاده کنید

## بررسی concat در js

ادغام متن با concat

برای متن‌ها:

|  |
| --- |
| let name = "علی";  let family = "محمدی";  // روش ساده با +  let fullName = name + " " + family;  console.log(fullName); // "علی محمدی"  // روش با concat  let fullName2 = name.concat(" ", family);  console.log(fullName2); // "علی محمدی" |

برای لیست‌ها (آرایه):

|  |
| --- |
| let list1 = ["سیب", "موز"];  let list2 = ["پرتقال", "انگور"];  let allFruits = list1.concat(list2);  console.log(allFruits); // ["سیب", "موز", "پرتقال", "انگور"] |

## بررسی بک تیک و template string

نوشتن متن با ` (بک‌تیک)

|  |
| --- |
| let firstName = "رضا";  let age = 20;  // روش قدیمی  let text1 = "نام: " + firstName + " - سن: " + age;  // روش جدید با `  let text2 = `نام: ${firstName} - سن: ${age}`;  console.log(text2); // "نام: رضا - سن: 20" |

مثال‌های بیشتر:

|  |
| --- |
| let product = "کتاب";  let price = 50000;  let message = `قیمت ${product} برابر ${price} تومان است`;  console.log(message); // "قیمت کتاب برابر 50000 تومان است"  // محاسبه در متن  let a = 10;  let b = 5;  console.log(`حاصل جمع: ${a + b}`); // "حاصل جمع: 15" |

## نحوه ورودی گرفتن از کاربر

دریافت نام:

|  |
| --- |
| let userName = prompt("لطفا نام خود را وارد کنید:");  console.log("سلام " + userName); |

دریافت عدد:

|  |
| --- |
| let userAge = prompt("لطفا سن خود را وارد کنید:");  console.log(`سن شما ${userAge} سال است`); |

**مثال کاربردی ساده:**

|  |
| --- |
| // برنامه خوشامدگویی  let name = prompt("نام شما چیست؟");  let age = prompt("چند سال دارید؟");  let welcomeMessage = `سلام ${name}!  سن شما ${age} سال است.  خوش آمدید!`;  console.log(welcomeMessage);  alert(welcomeMessage); |

مثال کلی و کاربردی

|  |
| --- |
| // برنامه معرفی شخصی  let firstName = prompt("نام شما چیست؟");  let lastName = prompt("نام خانوادگی شما چیست؟");  let birthYear = prompt("سال تولد شما؟");  // محاسبه سن  let currentYear = 2024;  let age = currentYear - birthYear;  // ایجاد متن معرفی  let introduction = `  === معرفی ===  نام: ${firstName}  نام خانوادگی: ${lastName}  سال تولد: ${birthYear}  سن: ${age} سال  ==============  `;  console.log(introduction);  alert(introduction); |

## آموزش casting در js

تبدیل نوع داده (Casting)

تبدیل به رشته (String):

|  |
| --- |
| let number = 123;  let text = String(number);  console.log(text); // "123"  console.log(typeof text); // "string"  // روش ساده‌تر:  let age = 25;  let ageText = age + "";  console.log(ageText); // "25" |

**تبدیل به عدد (Number):**

|  |
| --- |
| let text = "456";  let number = Number(text);  console.log(number); // 456  console.log(typeof number); // "number"  // روش ساده‌تر:  let price = "15000";  let realPrice = +price;  console.log(realPrice); // 15000 |

**تبدیل به بولین (Boolean):**

|  |
| --- |
| let text = "سلام";  let boolValue = Boolean(text);  console.log(boolValue); // true  let emptyText = "";  let boolValue2 = Boolean(emptyText);  console.log(boolValue2); // false |

## معرفی block scope ها و متغیر های globalو local

**متغیر سراسری (Global):**

|  |
| --- |
| let globalVar = "من سراسری هستم";  function showMessage() {  console.log(globalVar); // قابل دسترسی  }  showMessage();  console.log(globalVar); // قابل دسترسی |

**متغیر محلی (Local):**

|  |
| --- |
| function calculate() {  let localVar = "من محلی هستم"; // فقط داخل این تابع قابل دسترسی است  console.log(localVar); // کار می‌کند  }  calculate();  // console.log(localVar); // خطا می‌دهد |

محدوده بلوکی (Block Scope):

|  |
| --- |
| if (true) {  let blockVar = "من فقط در این بلوک هستم";  console.log(blockVar); // کار می‌کند  }  // console.log(blockVar); // خطا می‌دهد |

## معرفی arithmetic operator ها

|  |
| --- |
| let a = 10;  let b = 3;  console.log(a + b); // 13 (جمع)  console.log(a - b); // 7 (تفریق)  console.log(a \* b); // 30 (ضرب)  console.log(a / b); // 3.333 (تقسیم)  console.log(a % b); // 1 (باقی‌مانده)  // افزایش و کاهش  let count = 5;  count++; // count = count + 1  console.log(count); // 6  count--; // count = count - 1  console.log(count); // 5 |

## معرفی assignment operator ها

|  |
| --- |
| let x = 10;  x += 5; // x = x + 5  console.log(x); // 15  x -= 3; // x = x - 3  console.log(x); // 12  x \*= 2; // x = x \* 2  console.log(x); // 24  x /= 4; // x = x / 4  console.log(x); // 6 |

## معرفی comparison operator ها

|  |
| --- |
| let a = 10;  let b = 5;  console.log(a > b); // true (بزرگتر)  console.log(a < b); // false (کوچکتر)  console.log(a >= b); // true (بزرگتر یا مساوی)  console.log(a <= b); // false (کوچکتر یا مساوی)  console.log(a == b); // false (برابری مقدار)  console.log(a != b); // true (عدم برابری مقدار)  // تفاوت == و ===  console.log(5 == "5"); // true (مقایسه مقدار)  console.log(5 === "5"); // false (مقایسه مقدار و نوع) |

## معرفی ternary operator

|  |
| --- |
| let age = 20;  // روش معمولی:  let message;  if (age >= 18) {  message = "بزرگسال";  } else {  message = "کودک";  }  // روش کوتاه با ternary:  let message2 = age >= 18 ? "بزرگسال" : "کودک";  console.log(message2); // "بزرگسال"  // مثال دیگر:  let score = 85;  let result = score >= 50 ? "قبول" : "مردود";  console.log(result); // "قبول" |

## معرفی logical operator ها

|  |
| --- |
| let hasTicket = true;  let hasMoney = false;  // AND (و) - هر دو شرط باید true باشد  console.log(hasTicket && hasMoney); // false  // OR (یا) - یکی از شرط‌ها true باشد  console.log(hasTicket || hasMoney); // true  // NOT (نفی) - معکوس کردن مقدار  console.log(!hasTicket); // false  console.log(!hasMoney); // true  // مثال کاربردی:  let isSunny = true;  let isWeekend = false;  let shouldGoOut = isSunny && isWeekend;  console.log(shouldGoOut); // false |

## معرفی nan و تابع isNan

|  |
| --- |
| let result = "سلام" / 5;  console.log(result); // NaN (Not a Number)  let invalidNumber = Number("متن");  console.log(invalidNumber); // NaN |

**تشخیص NaN با isNaN:**

|  |
| --- |
| console.log(isNaN("123")); // false (چون "123" به عدد تبدیل می‌شود)  console.log(isNaN("علی")); // true (چون به عدد تبدیل نمی‌شود)  console.log(isNaN(123)); // false  // مثال کاربردی:  let userInput = "15000";  if (isNaN(userInput)) {  console.log("لطفاً یک عدد وارد کنید!");  } else {  console.log("عدد معتبر است: " + userInput);  } |

معرفی شرط ها

شرط‌ها به کامپیوتر کمک می‌کنند تا بر اساس شرایط مختلف، تصمیم‌های متفاوت بگیرد. مثل وقتی که ما می‌گوییم: "اگر باران ببارد، چتر برمی‌دارم"

If

1. دستور if (اگر)

مفهوم:

"اگر این شرط درست بود، این کار را انجام بده"

ساختار:

|  |
| --- |
| if (شرط) {  // کدهایی که اگر شرط درست باشد اجرا می‌شوند  } |

مثال‌های کاربردی:

|  |
| --- |
| // مثال 1: بررسی سن برای رای دادن  let age = 20;  if (age >= 18) {  console.log("می‌توانی رای بدهی!");  }  // مثال 2: بررسی موجودی کافی  let money = 50000;  let productPrice = 30000;  if (money >= productPrice) {  console.log("می‌تونی این محصول رو بخری");  } |

Else

2. دستور if...else (اگر...در غیر این صورت)

مفهوم:

"اگر این شرط درست بود، این کار را انجام بده، در غیر این صورت آن کار را انجام بده"

ساختار:

|  |
| --- |
| if (شرط) {  // اگر شرط درست باشد  } else {  // اگر شرط نادرست باشد  } |

مثال‌های کاربردی:

|  |
| --- |
| // مثال 1: قبولی یا مردودی  let score = 75;  if (score >= 50) {  console.log("آفرین! قبول شدی 🎉");  } else {  console.log("متاسفانه مردود شدی 😢");  }  // مثال 2: ورود به سیستم  let password = "123456";  let correctPassword = "123456";  if (password === correctPassword) {  console.log("خوش آمدی! وارد شدی ✅");  } else {  console.log("رمز اشتباه است ❌");  } |

Else if

3. دستور else if (در غیر این صورت اگر)

مفهوم:

"اگر شرط اول درست بود این کار را انجام بده، در غیر این صورت اگر شرط دوم درست بود آن کار را انجام بده، در غیر این صورت..."

ساختار:

|  |
| --- |
| if (شرط۱) {  // اگر شرط۱ درست باشد  } else if (شرط۲) {  // اگر شرط۲ درست باشد  } else if (شرط۳) {  // اگر شرط۳ درست باشد  } else {  // اگر هیچکدام درست نباشند  } |

**مثال‌های کاربردی:**

|  |
| --- |
| // مثال 1: سیستم نمره‌دهی  let score = 85;  if (score >= 90) {  console.log("نمره: A - عالی!");  } else if (score >= 80) {  console.log("نمره: B - خوب");  } else if (score >= 70) {  console.log("نمره: C - قابل قبول");  } else {  console.log("نمره: D - نیاز به تلاش بیشتر");  }  // مثال 2: تشخیص زمان روز  let hour = 14; // ساعت 2 بعدازظهر  if (hour < 12) {  console.log("صبح بخیر!");  } else if (hour < 18) {  console.log("عصر بخیر!");  } else {  console.log("شب بخیر!");  } |

Switch

دستور switch

مفهوم:

"بر اساس مقدار این متغیر، کارهای مختلفی انجام بده"

ساختار:

|  |
| --- |
| switch (متغیر) {  case مقدار۱:  // کدهای مربوط به مقدار۱  break;  case مقدار۲:  // کدهای مربوط به مقدار۲  break;  default:  // کدهایی که اگر هیچکدام مطابقت نکردند  } |

**مثال‌های کاربردی:**

|  |
| --- |
| // مثال 1: انتخاب روز هفته  let dayNumber = 3;  let dayName;  switch (dayNumber) {  case 1:  dayName = "شنبه";  break;  case 2:  dayName = "یکشنبه";  break;  case 3:  dayName = "دوشنبه";  break;  case 4:  dayName = "سه شنبه";  break;  default:  dayName = "روز نامعتبر";  }  console.log("امروز " + dayName + " است");  // مثال 2: ماشین حساب ساده  let operator = "+";  let number1 = 10;  let number2 = 5;  let result;  switch (operator) {  case "+":  result = number1 + number2;  console.log("حاصل جمع: " + result);  break;  case "-":  result = number1 - number2;  console.log("حاصل تفریق: " + result);  break;  case "\*":  result = number1 \* number2;  console.log("حاصل ضرب: " + result);  break;  case "/":  result = number1 / number2;  console.log("حاصل تقسیم: " + result);  break;  default:  console.log("عملگر نامعتبر است!");  } |

پروژه عملی: سیستم تشخیص وضعیت آب و هوا

|  |
| --- |
| // برنامه تشخیص وضعیت آب و هوا  let temperature = 25; // دمای هوا  console.log("دمای هوا: " + temperature + " درجه");  if (temperature > 35) {  console.log("❌ هوا بسیار گرم است - مراقب باشید!");  } else if (temperature > 25) {  console.log("☀️ هوا گرم است - می‌توانید بروید بیرون");  } else if (temperature > 15) {  console.log("🌤️ هوا مطبوع است - روز خوبی برای گردش");  } else if (temperature > 5) {  console.log("⛅ هوا خنک است - ژاکت بپوشید");  } else {  console.log("❄️ هوا بسیار سرد است - در خانه بمانید");  } |

نکات مهم برای یادگیری

تفاوت‌های اصلی:

if: برای یک شرط ساده

if...else: برای دو حالت ممکن (درست/نادرست)

else if: برای چندین شرط مختلف

switch: برای مقایسه یک مقدار با گزینه‌های مشخص

تمرین‌های کوچک:

تمرین ۱: برنامه‌ای بنویسید که بفهمد یک عدد مثبت است یا منفی

|  |
| --- |
| let number = -5;  if (number > 0) {  console.log("عدد مثبت است");  } else if (number < 0) {  console.log("عدد منفی است");  } else {  console.log("عدد صفر است");  } |

تمرین ۲: برنامه‌ای بنویسید که تشخیص دهد امروز آخر هفته است یا نه

|  |
| --- |
| let day = "جمعه";  if (day === "جمعه" || day === "پنجشنبه") {  console.log("🎉 امروز آخر هفته است - استراحت کن!");  } else {  console.log("📚 امروز روز کاری است - درس بخوان!");  } |

نتیجه‌گیری

شرط‌ها مانند مغز برنامه‌نویسی هستند! آن‌ها به برنامه‌های ما هوش می‌دهند و کمک می‌کنند در شرایط مختلف، رفتارهای متفاوتی داشته باشند.

## معرفی توابع در js

معرفی توابع (Functions)

توابع چیستند؟

توابع بلوک‌های کدی هستند که:

یک کار خاص انجام می‌دهند

می‌توانند پارامتر بگیرند (ورودی)

می‌توانند مقدار برگردانند (خروجی)

قابل استفاده مجدد هستند

1. روش‌های ساختن تابع

روش اول: Function Declaration

|  |
| --- |
| // تعریف تابع  function sayHello() {  console.log("Hello everyone!");  }  // فراخوانی تابع  sayHello(); // "Hello everyone!" |

روش دوم: Function Expression

|  |
| --- |
| // تعریف تابع در یک متغیر  const welcome = function() {  console.log("Welcome!");  };  // فراخوانی تابع  welcome(); // "Welcome!" |

روش سوم: Arrow Function

|  |
| --- |
| // روش مدرن و خلاصه  const goodbye = () => {  console.log("Goodbye!");  };  goodbye(); // "Goodbye!" |

## نحوه ورودی گرفتن توابع

توابع با پارامتر ساده:

|  |
| --- |
| // تابع با یک پارامتر  function greet(name) {  console.log("Hello " + name + "!");  }  // استفاده از تابع  greet("Ali"); // "Hello Ali!"  greet("Reza"); // "Hello Reza!" |

توابع با چندین پارامتر:

|  |
| --- |
| // تابع با دو پارامتر  function addNumbers(num1, num2) {  let result = num1 + num2;  console.log("Sum: " + result);  }  // استفاده از تابع  addNumbers(5, 3); // "Sum: 8"  addNumbers(10, 7); // "Sum: 17" |

توابع با مقدار پیش‌فرض:

|  |
| --- |
| // پارامتر با مقدار پیش‌فرض  function multiply(a, b = 1) {  return a \* b;  }  console.log(multiply(5)); // 5 (b=1)  console.log(multiply(5, 3)); // 15 |

## بررسی تفاوت توابع void و غیره

تفاوت توابع با return و بدون return

🔄 تابع بدون return (void):

|  |
| --- |
| // این تابع فقط کار انجام می‌دهد و چیزی برنمی‌گرداند  function showMessage(message) {  console.log("Message: " + message);  }  let result = showMessage("Hello");  console.log(result); // undefined |

تابع با return:

|  |
| --- |
| // این تابع یک مقدار برمی‌گرداند  function calculateSum(a, b) {  return a + b;  }  let sum = calculateSum(10, 5);  console.log(sum); // 15 |

مقایسه عملی:

|  |
| --- |
| // تابع void - فقط چاپ می‌کند  function printAge(age) {  console.log("Age: " + age);  }  // تابع با return - مقدار برمی‌گرداند  function getNextYearAge(age) {  return age + 1;  }  // استفاده  printAge(25); // "Age: 25" - فقط چاپ می‌کند  let nextAge = getNextYearAge(25); // مقدار را برمی‌گرداند  console.log("Next year: " + nextAge); // "Next year: 26" |

## بررسی توابع بازگشتی

مفهوم بازگشت:

تابعی که خودش را صدا می‌زند

مثال ساده: محاسبه فاکتوریل

|  |
| --- |
| // محاسبه فاکتوریل با بازگشت  function factorial(n) {  // شرط توقف  if (n === 0 || n === 1) {  return 1;  }  // فراخوانی بازگشتی  return n \* factorial(n - 1);  }  console.log(factorial(5)); // 120 (5×4×3×2×1) |

مثال: جمع اعداد از 1 تا n

|  |
| --- |
| // جمع اعداد از 1 تا n  function sumToN(n) {  if (n === 1) {  return 1;  }  return n + sumToN(n - 1);  }  console.log(sumToN(5)); // 15 (5+4+3+2+1) |

مثال: نمایش شمارش معکوس

|  |
| --- |
| // شمارش معکوس  function countdown(number) {  if (number < 0) {  return; // توقف  }  console.log(number);  countdown(number - 1); // فراخوانی بازگشتی  }  countdown(5);  // خروجی:  // 5  // 4  // 3  // 2  // 1  // 0 |

مثال‌های کاربردی ترکیبی

ماشین حساب ساده:

|  |
| --- |
| function calculator(operation, a, b) {  switch(operation) {  case 'add':  return a + b;  case 'subtract':  return a - b;  case 'multiply':  return a \* b;  case 'divide':  return a / b;  default:  return "Invalid operation";  }  }  console.log(calculator('add', 10, 5)); // 15  console.log(calculator('multiply', 3, 4)); // 12 |

**سیستم تخفیف:**

|  |
| --- |
| function calculateDiscount(price, age) {  let discount = 0;    if (age < 18) {  discount = 0.2; // 20% تخفیف برای کودکان  } else if (age >= 60) {  discount = 0.3; // 30% تخفیف برای سالمندان  }    let finalPrice = price \* (1 - discount);  return finalPrice;  }  console.log(calculateDiscount(10000, 16)); // 8000  console.log(calculateDiscount(10000, 65)); // 7000 |

نکات مهم

✅ نکات کلیدی:

تابع بدون return مانند void عمل می‌کند

تابع با return می‌تواند در محاسبات استفاده شود

توابع بازگشتی باید شرط توقف داشته باشند

نام توابع باید واضح و توصیفی باشد

❌ مواردی که باید اجتناب کنید:

توابع بازگشتی بی‌نهایت (فراموش کردن شرط توقف)

نام‌های نامشخص برای توابع

توابع خیلی طولانی و پیچیده

تمرین‌های پیشنهادی:

تمرین ۱: تابعی بنویسید که یک عدد بگیرد و زوج یا فرد بودن آن را برگرداند

تمرین ۲: تابع بازگشتی بنویسید که توان یک عدد را محاسبه کند

|  |
| --- |
| // پاسخ تمرین ۱  function isEven(number) {  return number % 2 === 0;  }  // پاسخ تمرین ۲  function power(base, exponent) {  if (exponent === 0) {  return 1;  }  return base \* power(base, exponent - 1);  } |

## معرفی حلقه ها

For

حلقه for معرفی: حلقه for برای تکرار یک کد به تعداد مشخص استفاده می‌شود.

ساختار:

|  |
| --- |
| for (start;condition;step) {  // کدی که تکرار می‌شود  } |

مثال ساده:

|  |
| --- |
| // چاپ اعداد 1 تا 5  for (let i = 1; i <= 5; i++) {  console.log(i);  } |

مثال کاربردی:

|  |
| --- |
| // محاسبه جمع اعداد 1 تا 10  let sum = 0;  for (let i = 1; i <= 10; i++) {  sum = sum + i;  }  console.log("Sum: " + sum); // Sum: 55 |

While

معرفی:

حلقه while تا زمانی که شرط true باشد، کد را اجرا می‌کند.

ساختار:

|  |
| --- |
| while (شرط) {  // کدی که تکرار می‌شود  } |

مثال ساده:

|  |
| --- |
| // چاپ اعداد 1 تا 5  let count = 1;  while (count <= 5) {  console.log(count);  count++;  } |

مثال کاربردی:

|  |
| --- |
| // کاهش عدد تا رسیدن به صفر  let number = 5;  while (number > 0) {  console.log("Countdown: " + number);  number--;  } |

Do while

معرفی:

حلقه do while حداقل یک بار اجرا می‌شود، حتی اگر شرط false باشد.

ساختار:

|  |
| --- |
| do {  // کدی که تکرار می‌شود  } while (شرط); |

مثال ساده:

|  |
| --- |
| // چاپ اعداد 1 تا 5  let num = 1;  do {  console.log(num);  num++;  } while (num <= 5); |

مثال کاربردی:

|  |
| --- |
| // دریافت ورودی از کاربر تا زمانی که عدد مثبت باشد  let userInput;  do {  userInput = prompt("Enter a positive number:");  } while (userInput <= 0);  console.log("Thank you! You entered: " + userInput); |

معرفی حلقه های تو در تو

معرفی:

حلقه‌های تو در تو وقتی استفاده می‌شوند که نیاز به کار با ساختارهای دو بعدی داریم.

مثال ساده:

|  |
| --- |
| // ایجاد الگوی مربعی  for (let row = 1; row <= 3; row++) {  let line = "";  for (let col = 1; col <= 3; col++) {  line += "\* ";  }  console.log(line);  } |

**مثال کاربردی:**

|  |
| --- |
| // ایجاد جدول ضرب  for (let i = 1; i <= 5; i++) {  let row = "";  for (let j = 1; j <= 5; j++) {  row += (i \* j) + "\t";  }  console.log(row);  } |

مثال‌های کاربردی بیشتر

حلقه for با آرایه:

|  |
| --- |
| let fruits = ["apple", "banana", "orange", "grape"];  for (let i = 0; i < fruits.length; i++) {  console.log("Fruit: " + fruits[i]);  } |

حلقه while برای بازی:

|  |
| --- |
| let attempts = 3;  let correctPassword = "1234";  while (attempts > 0) {  let guess = prompt("Enter password (" + attempts + " attempts left):");    if (guess === correctPassword) {  console.log("Access granted!");  break;  } else {  attempts--;  console.log("Wrong password!");  }  }  if (attempts === 0) {  console.log("Access denied!");  } |

حلقه do while برای منو:

|  |
| --- |
| let choice;  do {  console.log("1. Play game");  console.log("2. View scores");  console.log("3. Exit");  choice = prompt("Enter your choice (1-3):");    switch(choice) {  case "1":  console.log("Starting game...");  break;  case "2":  console.log("Showing scores...");  break;  case "3":  console.log("Goodbye!");  break;  default:  console.log("Invalid choice!");  }  } while (choice !== "3"); |

خلاصه تفاوت‌ها:

|  |  |  |
| --- | --- | --- |
| حلقه | کاربرد | تعداد اجرا |
| for | وقتی تعداد تکرار مشخص است | مشخص |
| while | وقتی شرط نامشخص است | نامشخص |
| do while | وقتی حداقل یک بار اجرا نیاز است | حداقل یک بار |

## معرفی آرایه ها

معرفی آرایه ها (Arrays)

آرایه ها ساختارهای داده ای هستند که امکان ذخیره چندین مقدار را در یک متغیر فراهم می کنند.

|  |
| --- |
| // روش های مختلف ایجاد آرایه  let students = ["علی", "رضا", "مریم"];  let numbers = new Array(1, 2, 3, 4, 5);  let mixed = ["متن", 42, true, null]; |

نحوه دسترسی به عناصر آرایه و معرفی index

هر عنصر در آرایه یک شماره index دارد که از 0 شروع می شود.

|  |
| --- |
| let colors = ["قرمز", "آبی", "سبز", "زرد"];  console.log(colors[0]); // "قرمز"  console.log(colors[2]); // "سبز"  console.log(colors[colors.length - 1]); // آخرین عنصر: "زرد" |

آپدیت ارایه ها

می توانید مقادیر آرایه را تغییر دهید:

|  |
| --- |
| let fruits = ["سیب", "موز", "پرتقال"];  fruits[1] = "انگور"; // آپدیت عنصر دوم  console.log(fruits); // ["سیب", "انگور", "پرتقال"] |

نحوه استفاده از length در ارایه ها

خصوصیت length تعداد عناصر آرایه را برمی گرداند:

|  |
| --- |
| let courses = ["HTML", "CSS", "JavaScript", "React"];  console.log(courses.length); // 4  // استفاده در حلقه  for (let i = 0; i < courses.length; i++) {  console.log(courses[i]);  } |

معرفی توابع push و pop و shift و unshift

تابع push - اضافه کردن به انتها

|  |
| --- |
| let stack = [1, 2, 3];  stack.push(4);  console.log(stack); // [1, 2, 3, 4] |

تابع pop- حذف از انتها

|  |
| --- |
| let stack = [1, 2, 3];  let lastItem = stack.pop();  console.log(stack); // [1, 2]  console.log(lastItem); // 3 |

تابع shift- حذف از ابتدا

|  |
| --- |
| let queue = [1, 2, 3];  let firstItem = queue.shift();  console.log(queue); // [2, 3]  console.log(firstItem); // 1 |

تابع unshift - اضافه کردن به ابتدا

|  |
| --- |
| let queue = [2, 3];  queue.unshift(1);  console.log(queue); // [1, 2, 3] |

## معرفی تابع isArray

برای بررسی اینکه یک متغیر آرایه است یا خیر:

|  |
| --- |
| let arr = [1, 2, 3];  let str = "Hello";  console.log(Array.isArray(arr)); // true  console.log(Array.isArray(str)); // false  console.log(Array.isArray([1, 2, 3])); // true |

مثال کاربردی در طراحی وب

|  |
| --- |
| // مدیریت منوی navigation  let menuItems = ["Home", "About", "Services", "Contact"];  // اضافه کردن آیتم جدید  menuItems.push("Blog");  // حذف اولین آیتم  menuItems.shift();  // نمایش منو  menuItems.forEach((item, index) => {  console.log(`${index + 1}. ${item}`);  }); |

## معرفی associative array ها

در JavaScript، آرایه های انجمنی (Associative Arrays) به صورت رسمی وجود ندارند، اما می توان از Object برای این منظور استفاده کرد:

|  |
| --- |
| // استفاده از Object به عنوان آرایه انجمنی  let student = {  name: "علی رضایی",  age: 20,  grade: "A",  courses: ["طراحی وب", "پایگاه داده"]  };  // دسترسی به مقادیر  console.log(student["name"]); // "علی رضایی"  console.log(student.age); // 20  // اضافه کردن property جدید  student["email"] = "ali@university.com";  // حلقه زدن through properties  for (let key in student) {  console.log(`${key}: ${student[key]}`);  } |

## معرفی تابع toString

تبدیل آرایه به رشته متنی:

|  |
| --- |
| let fruits = ["سیب", "موز", "پرتقال"];  let result = fruits.toString();  console.log(result); // "سیب,موز,پرتقال"  console.log(typeof result); // "string"  // استفاده عملی در نمایش داده ها  let numbers = [1, 2, 3, 4, 5];  console.log("اعداد: " + numbers.toString()); // "اعداد: 1,2,3,4,5" |

## معرفی تابع join

تبدیل آرایه به رشته با جداکننده دلخواه:

|  |
| --- |
| let colors = ["قرمز", "سبز", "آبی"];  console.log(colors.join()); // "قرمز,سبز,آبی" (پیشفرض کاما)  console.log(colors.join(" - ")); // "قرمز - سبز - آبی"  console.log(colors.join(" | ")); // "قرمز | سبز | آبی"  console.log(colors.join("")); // "قرمزسبزآبی"  // مثال کاربردی در تولید HTML  let menuItems = ["Home", "About", "Contact"];  let menuHTML = "<ul><li>" + menuItems.join("</li><li>") + "</li></ul>";  console.log(menuHTML);  // <ul><li>Home</li><li>About</li><li>Contact</li></ul> |

## معرفی تابع delete

حذف عنصر از آرایه (جا خالی می گذارد):

|  |
| --- |
| let numbers = [10, 20, 30, 40, 50];  delete numbers[2]; // حذف عنصر با index 2  console.log(numbers); // [10, 20, empty, 40, 50]  console.log(numbers[2]); // undefined  console.log(numbers.length); // 5 (طول تغییر نکرده!) |

## معرفی تابع Splice

برای اضافه کردن، حذف یا جایگزینی عناصر:

|  |
| --- |
| let courses = ["HTML", "CSS", "JavaScript", "React", "Node.js"];  // حذف عناصر  let removed = courses.splice(2, 2); // از index 2، دو عنصر حذف کن  console.log(courses); // ["HTML", "CSS", "Node.js"]  console.log(removed); // ["JavaScript", "React"]  // اضافه کردن عناصر  courses.splice(1, 0, "Bootstrap", "jQuery");  console.log(courses); // ["HTML", "Bootstrap", "jQuery", "CSS", "Node.js"]  // جایگزینی عناصر  courses.splice(2, 1, "Vue.js");  console.log(courses); // ["HTML", "Bootstrap", "Vue.js", "CSS", "Node.js"] |

## معرفی تابع concat

ادغام دو یا چند آرایه:

|  |
| --- |
| let frontend = ["HTML", "CSS", "JavaScript"];  let backend = ["Node.js", "Python", "PHP"];  let databases = ["MySQL", "MongoDB"];  // ادغام دو آرایه  let fullstack = frontend.concat(backend);  console.log(fullstack);  // ["HTML", "CSS", "JavaScript", "Node.js", "Python", "PHP"]  // ادغام چندین آرایه  let allTechnologies = frontend.concat(backend, databases);  console.log(allTechnologies);  // ["HTML", "CSS", "JavaScript", "Node.js", "Python", "PHP", "MySQL", "MongoDB"]  // ادغام با مقادیر جداگانه  let withTools = frontend.concat("Git", "Webpack");  console.log(withTools);  // ["HTML", "CSS", "JavaScript", "Git", "Webpack"] |

مثال کاربردی در طراحی وب

|  |
| --- |
| // مدیریت تگ های یک مقاله وبلاگ  let articleTags = ["javascript", "web", "programming"];  // اضافه کردن تگ های جدید  let newTags = ["frontend", "es6"];  let allTags = articleTags.concat(newTags);  // نمایش تگ ها به صورت رشته با هشتگ  let tagsString = allTags.join(" #");  console.log("#" + tagsString); // "#javascript #web #programming #frontend #es6"  // مدیریت نظرات کاربران  let comments = [  {user: "user1", text: "مقاله عالی!"},  {user: "user2", text: "ممنون از آموزش"}  ];  // حذف یک نظر  comments.splice(0, 1); // حذف اولین نظر |

## معرفی تابع Slice

ایجاد کپی از بخشی از آرایه بدون تغییر آرایه اصلی:

|  |
| --- |
| let fruits = ["سیب", "موز", "پرتقال", "انگور", "هلو"];  // کپی از index 1 تا 3 (عدم شامل شدن index 3)  let citrus = fruits.slice(1, 3);  console.log(citrus); // ["موز", "پرتقال"]  console.log(fruits); // آرایه اصلی تغییر نکرده  // کپی از index 2 تا انتها  let fromIndex2 = fruits.slice(2);  console.log(fromIndex2); // ["پرتقال", "انگور", "هلو"]  // کپی از انتها (اعداد منفی)  let lastTwo = fruits.slice(-2);  console.log(lastTwo); // ["انگور", "هلو"]  // کپی کل آرایه  let fruitsCopy = fruits.slice();  console.log(fruitsCopy); // کپی کامل آرایه |

## معرفی تابع sortNumeric

|  |
| --- |
| let numbers = [40, 100, 1, 5, 25, 10];  // sort پیشفرض (بر اساس رشته)  let wrongSort = numbers.sort();  console.log(wrongSort); // [1, 10, 100, 25, 40, 5] - نادرست!  // sort عددی صحیح  let correctSort = numbers.sort(function(a, b) {  return a - b;  });  console.log(correctSort); // [1, 5, 10, 25, 40, 100] - صحیح  // با arrow function  let ascending = numbers.sort((a, b) => a - b);  let descending = numbers.sort((a, b) => b - a);  console.log(ascending); // صعودی: [1, 5, 10, 25, 40, 100]  console.log(descending); // نزولی: [100, 40, 25, 10, 5, 1] |

## معرفی متد foreach

اجرای یک تابع برای هر عنصر آرایه:

|  |
| --- |
| let students = ["علی", "مریم", "رضا", "فاطمه"];  // روش ساده  students.forEach(function(student) {  console.log("دانشجو: " + student);  });  // با arrow function و index  students.forEach((student, index) => {  console.log(`دانشجو ${index + 1}: ${student}`);  });  // مثال کاربردی در طراحی وب  let products = [  { name: "لپ تاپ", price: 1500 },  { name: "موبایل", price: 800 },  { name: "تبلت", price: 500 }  ];  products.forEach(product => {  console.log(`${product.name}: $${product.price}`);  }); |

## معرفی متد map

ایجاد آرایه جدید با تبدیل هر عنصر:

|  |
| --- |
| let numbers = [1, 2, 3, 4, 5];  // تبدیل هر عدد به مربع آن  let squares = numbers.map(function(num) {  return num \* num;  });  console.log(squares); // [1, 4, 9, 16, 25]  // با arrow function  let doubled = numbers.map(num => num \* 2);  console.log(doubled); // [2, 4, 6, 8, 10]  // مثال کاربردی در تولید HTML  let menuItems = ["Home", "About", "Contact"];  let menuLinks = menuItems.map(item => {  return `<a href="${item.toLowerCase()}.html">${item}</a>`;  });  console.log(menuLinks);  // ['<a href="home.html">Home</a>', ...] |

## بررسی تافوت foreach و map

تفاوت های کلیدی:

|  |  |  |
| --- | --- | --- |
| ویژگی | `forEach` | `map` |
| \*\*مقدار بازگشتی\*\* | `undefined` | آرایه جدید |
| \*\*تغییر آرایه اصلی\*\* | امکان دارد | تغییر نمی دهد |
| \*\*زنجیره ای کردن\*\* | نمی شود | می شود |
| \*\*کاربرد\*\* | اجرای عملیات | تبدیل داده ها |

## معرفی متد filter

ایجاد آرایه جدید با عناصری که شرط خاصی را دارا هستند:

|  |
| --- |
| let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  // فیلتر اعداد زوج  let evenNumbers = numbers.filter(function(number) {  return number % 2 === 0;  });  console.log(evenNumbers); // [2, 4, 6, 8, 10]  // با arrow function  let oddNumbers = numbers.filter(num => num % 2 !== 0);  console.log(oddNumbers); // [1, 3, 5, 7, 9]  // مثال کاربردی در مدیریت محصولات  let products = [  { name: "لپ‌تاپ", price: 1500, category: "electronics" },  { name: "کتاب", price: 20, category: "education" },  { name: "هدفون", price: 100, category: "electronics" },  { name: "ماوس", price: 50, category: "electronics" }  ];  // فیلتر محصولات الکترونیکی  let electronics = products.filter(product =>  product.category === "electronics"  );  console.log(electronics);  // فیلتر محصولات با قیمت بالای 100  let expensiveProducts = products.filter(product =>  product.price > 100  );  console.log(expensiveProducts); |

## معرفی متد indexOf و lastIndexOf

indexOf - پیدا کردن اولین index

|  |
| --- |
| let fruits = ["سیب", "موز", "پرتقال", "موز", "انگور"];  let firstBanana = fruits.indexOf("موز");  console.log(firstBanana); // 1  let firstOrange = fruits.indexOf("پرتقال");  console.log(firstOrange); // 2  // وقتی عنصر پیدا نشود  let notFound = fruits.indexOf("هلو");  console.log(notFound); // -1  // جستجو از index خاص  let secondBanana = fruits.indexOf("موز", 2);  console.log(secondBanana); // 3 |

پیدا کردن آخرین index

|  |
| --- |
| let fruits = ["سیب", "موز", "پرتقال", "موز", "انگور"];  let lastBanana = fruits.lastIndexOf("موز");  console.log(lastBanana); // 3  let lastOrange = fruits.lastIndexOf("پرتقال");  console.log(lastOrange); // 2  // جستجو به صورت معکوس  let searchFrom = fruits.lastIndexOf("موز", 2);  console.log(searchFrom); // 1 |

## معرفی object

اشیاء (Objects) برای ذخیره داده‌های ساختاریافته استفاده می‌شوند:

|  |
| --- |
| // ایجاد object با literal syntax  let student = {  firstName: "علی",  lastName: "رضایی",  age: 20,  grade: "A",  courses: ["طراحی وب", "پایگاه داده"],  isActive: true,    // متد (تابع داخل object)  getFullName: function() {  return this.firstName + " " + this.lastName;  },    // روش جدیدتر تعریف متد  getInfo() {  return `${this.getFullName()} - سن: ${this.age}`;  }  }; |

## نحوه دسترسی به عناصر object

روش‌های مختلف دسترسی:

|  |
| --- |
| let car = {  brand: "تویوتا",  model: "کمری",  year: 2023,  "fuel-type": "بنزین" // property با نام غیر استاندارد  };  // 1. Dot Notation (نقطه)  console.log(car.brand); // "تویوتا"  console.log(car.model); // "کمری"  // 2. Bracket Notation (کروشه)  console.log(car["brand"]); // "تویوتا"  console.log(car["model"]); // "کمری"  // موارد ضروری استفاده از bracket notation  console.log(car["fuel-type"]); // "بنزین" -因为有横线  console.log(car["year"]); // 2023  // با استفاده از متغیر  let propertyName = "brand";  console.log(car[propertyName]); // "تویوتا" |

**کاربردهای عملی در طراحی وب:**

|  |
| --- |
| // مدیریت پروفایل کاربر  let userProfile = {  id: 1,  username: "web\_designer",  email: "designer@example.com",  profile: {  firstName: "مریم",  lastName: "محمدی",  avatar: "/images/avatar.jpg",  bio: "طراح وب و توسعه‌دهنده فرانت‌اند"  },  preferences: {  theme: "dark",  language: "fa",  notifications: true  },    // متدها  getDisplayName() {  return `${this.profile.firstName} ${this.profile.lastName}`;  },    updateEmail(newEmail) {  this.email = newEmail;  return "ایمیل با موفقیت به‌روزرسانی شد";  }  };  // دسترسی به properties  console.log(userProfile.username); // "web\_designer"  console.log(userProfile.profile.firstName); // "مریم"  console.log(userProfile["preferences"]["theme"]); // "dark"  // فراخوانی متدها  console.log(userProfile.getDisplayName()); // "مریم محمدی"  console.log(userProfile.updateEmail("new@example.com"));  // اضافه کردن property جدید  userProfile.registrationDate = "2024-01-15";  userProfile["socialLinks"] = {  twitter: "https://twitter.com/user",  github: "https://github.com/user"  };  // حذف property  delete userProfile.preferences.notifications; |

مثال ترکیبی با آرایه‌ها:

|  |
| --- |
| // سیستم مدیریت مقالات وبلاگ  let blog = {  name: "طراحی وب مدرن",  author: "علی احمدی",  articles: [  {  id: 1,  title: "یادگیری JavaScript",  tags: ["javascript", "programming", "web"],  published: true,  views: 1500  },  {  id: 2,  title: "CSS Grid布局 کامل",  tags: ["css", "layout", "design"],  published: true,  views: 800  },  {  id: 3,  title: "مقاله در حال نوشتن",  tags: ["draft"],  published: false,  views: 0  }  ],    // متد برای پیدا کردن مقالات منتشر شده  getPublishedArticles() {  return this.articles.filter(article => article.published);  },    // متد برای پیدا کردن مقالات پرطرفدار  getPopularArticles(minViews = 1000) {  return this.articles.filter(article =>  article.published && article.views >= minViews  );  },    // متد برای جستجو در مقالات  searchArticles(keyword) {  return this.articles.filter(article =>  article.title.toLowerCase().includes(keyword.toLowerCase()) &&  article.published  );  }  };  // استفاده از متدها  console.log("مقالات منتشر شده:", blog.getPublishedArticles());  console.log("مقالات پرطرفدار:", blog.getPopularArticles());  console.log("مقالات مربوط به JavaScript:", blog.searchArticles("javascript")); |

## آموزش event ها

Event ها اتفاقاتی هستند که در مرورگر رخ می‌دهند و ما می‌توانیم با استفاده از JavaScript به آنها پاسخ دهیم.

|  |
| --- |
| // ساختار پایه Event Listener  element.addEventListener('event-type', function() {  // کدی که هنگام رخ دادن event اجرا می‌شود  }); |

Click

Event click زمانی رخ می‌دهد که کاربر روی یک element کلیک می‌کند.

پیاده سازی پایه:

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش Event کلیک</title>  <style>  .button {  padding: 15px 30px;  background-color: #007bff;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  margin: 10px;  transition: background-color 0.3s;  }  .button:hover {  background-color: #0056b3;  }  .counter {  font-size: 24px;  margin: 20px;  padding: 20px;  border: 2px solid #333;  display: inline-block;  }  </style>  </head>  <body>  <button id="clickBtn" class="button">روی من کلیک کن!</button>  <div id="clickCounter" class="counter">تعداد کلیک‌ها: 0</div>  <script>  let count = 0;  const button = document.getElementById('clickBtn');  const counter = document.getElementById('clickCounter');  button.addEventListener('click', function() {  count++;  counter.textContent = `تعداد کلیک‌ها: ${count}`;  this.style.backgroundColor = '#28a745';    // بازگشت به رنگ اصلی بعد از 200 میلی‌ثانیه  setTimeout(() => {  this.style.backgroundColor = '#007bff';  }, 200);  });  </script>  </body>  </html> |

کاربردهای عملی:

|  |
| --- |
| // مدیریت منوی navigation  const menuButton = document.getElementById('menuBtn');  const navigation = document.getElementById('navMenu');  menuButton.addEventListener('click', function() {  navigation.classList.toggle('active');  });  // Modal window  const openModalBtn = document.getElementById('openModal');  const closeModalBtn = document.getElementById('closeModal');  const modal = document.getElementById('myModal');  openModalBtn.addEventListener('click', function() {  modal.style.display = 'block';  });  closeModalBtn.addEventListener('click', function() {  modal.style.display = 'none';  }); |

Mouse enter

Event mouseenter زمانی رخ می‌دهد که ماوس وارد محدوده یک element شود.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <title>Event ورود ماوس</title>  <style>  .card {  width: 200px;  height: 150px;  background-color: #f0f0f0;  border: 2px solid #ddd;  border-radius: 10px;  padding: 20px;  margin: 20px;  transition: all 0.3s ease;  }  .hover-effect {  transform: scale(1.05);  box-shadow: 0 5px 15px rgba(0,0,0,0.2);  background-color: #e3f2fd;  }  .tooltip {  display: none;  position: absolute;  background: #333;  color: white;  padding: 5px 10px;  border-radius: 4px;  }  </style>  </head>  <body>  <div id="hoverCard" class="card">  ماوس را روی من بیاورید  </div>  <button id="tooltipBtn" class="button">دکمه با راهنما</button>  <div id="tooltip" class="tooltip">این یک راهنمای کوچک است!</div>  <script>  const card = document.getElementById('hoverCard');  const tooltipBtn = document.getElementById('tooltipBtn');  const tooltip = document.getElementById('tooltip');  // اثر hover روی کارت  card.addEventListener('mouseenter', function() {  this.classList.add('hover-effect');  this.textContent = 'ماوس وارد شد!';  });  // نمایش tooltip  tooltipBtn.addEventListener('mouseenter', function(e) {  tooltip.style.display = 'block';  tooltip.style.left = (e.pageX + 10) + 'px';  tooltip.style.top = (e.pageY + 10) + 'px';  });  </script>  </body>  </html> |

Mouseleave

Event mouseleave زمانی رخ می‌دهد که ماوس از محدوده یک element خارج شود.

|  |
| --- |
| // ادامه کدهای قبلی  card.addEventListener('mouseleave', function() {  this.classList.remove('hover-effect');  this.textContent = 'ماوس را روی من بیاورید';  });  tooltipBtn.addEventListener('mouseleave', function() {  tooltip.style.display = 'none';  });  // مثال کاربردی: گالری تصاویر  const galleryItems = document.querySelectorAll('.gallery-item');  galleryItems.forEach(item => {  item.addEventListener('mouseenter', function() {  const overlay = this.querySelector('.image-overlay');  overlay.style.opacity = '1';  });    item.addEventListener('mouseleave', function() {  const overlay = this.querySelector('.image-overlay');  overlay.style.opacity = '0';  });  }); |

Keyup

Event keyup زمانی رخ می‌دهد که یک کلید روی صفحه کلید رها شود.

|  |
| --- |
| // اعتبارسنجی real-time  const emailInput = document.getElementById('emailInput');  const passwordInput = document.getElementById('passwordInput');  emailInput.addEventListener('keyup', function() {  const email = this.value;  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;    if (emailRegex.test(email)) {  this.style.borderColor = '#28a745';  } else {  this.style.borderColor = '#dc3545';  }  });  passwordInput.addEventListener('keyup', function() {  const password = this.value;  const strength = checkPasswordStrength(password);    updatePasswordStrengthIndicator(strength);  });  function checkPasswordStrength(password) {  let strength = 0;    if (password.length >= 8) strength++;  if (/[A-Z]/.test(password)) strength++;  if (/[0-9]/.test(password)) strength++;  if (/[^A-Za-z0-9]/.test(password)) strength++;    return strength;  }  function updatePasswordStrengthIndicator(strength) {  const indicator = document.getElementById('passwordStrength');  const colors = ['#dc3545', '#ffc107', '#17a2b8', '#28a745'];  const texts = ['ضعیف', 'متوسط', 'قوی', 'خیلی قوی'];    indicator.style.width = `${(strength / 4) \* 100}%`;  indicator.style.backgroundColor = colors[strength] || '#dc3545';  indicator.textContent = texts[strength] || 'ضعیف';  } |

Keydown

Event keydown زمانی رخ می‌دهد که یک کلید روی صفحه کلید فشرده شود.

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <title>Event های صفحه کلید</title>  <style>  .input-field {  padding: 10px;  margin: 10px;  width: 300px;  border: 2px solid #ddd;  border-radius: 5px;  }  .character-counter {  margin: 10px;  color: #666;  }  .message {  padding: 10px;  margin: 10px;  border-radius: 5px;  display: none;  }  .success {  background-color: #d4edda;  color: #155724;  border: 1px solid #c3e6cb;  }  </style>  </head>  <body>  <input type="text" id="textInput" class="input-field" placeholder="متن خود را تایپ کنید...">  <div id="charCount" class="character-counter">تعداد کاراکترها: 0</div>  <div id="message" class="message"></div>  <script>  const textInput = document.getElementById('textInput');  const charCount = document.getElementById('charCount');  const message = document.getElementById('message');  // شمارش کاراکترها  textInput.addEventListener('keydown', function(e) {  // به روزرسانی شمارنده با کمی تاخیر  setTimeout(() => {  charCount.textContent = `تعداد کاراکترها: ${this.value.length}`;  }, 0);  });  // تشخیص کلیدهای خاص  textInput.addEventListener('keydown', function(e) {  // کلید Enter  if (e.key === 'Enter') {  showMessage('کلید Enter فشرده شد!', 'success');  }    // کلید Escape  if (e.key === 'Escape') {  this.value = '';  showMessage('متن پاک شد!', 'success');  }    // ترکیب کلیدها (Ctrl + A)  if (e.ctrlKey && e.key === 'a') {  e.preventDefault(); // جلوگیری از رفتار پیشفرض  this.select();  showMessage('تمام متن انتخاب شد!', 'success');  }  });  function showMessage(text, type) {  message.textContent = text;  message.className = `message ${type}`;  message.style.display = 'block';    setTimeout(() => {  message.style.display = 'none';  }, 2000);  }  </script>  </body>  </html> |

مثال جامع: سیستم جستجوی Real-time

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa">  <head>  <meta charset="UTF-8">  <title>سیستم جستجوی Real-time</title>  <style>  .search-container {  margin: 50px auto;  width: 80%;  max-width: 600px;  }  .search-box {  width: 100%;  padding: 15px;  font-size: 16px;  border: 2px solid #ddd;  border-radius: 25px;  outline: none;  transition: border-color 0.3s;  }  .search-box:focus {  border-color: #007bff;  }  .results-container {  margin-top: 20px;  border: 1px solid #ddd;  border-radius: 10px;  max-height: 300px;  overflow-y: auto;  display: none;  }  .result-item {  padding: 15px;  border-bottom: 1px solid #eee;  cursor: pointer;  transition: background-color 0.2s;  }  .result-item:hover {  background-color: #f8f9fa;  }  .result-item:last-child {  border-bottom: none;  }  </style>  </head>  <body>  <div class="search-container">  <input type="text" id="searchInput" class="search-box" placeholder="جستجو کنید...">  <div id="searchResults" class="results-container"></div>  </div>  <script>  const searchInput = document.getElementById('searchInput');  const searchResults = document.getElementById('searchResults');    // داده‌های نمونه  const products = [  'لپ‌تاپ ایسوس',  'موبایل سامسونگ',  'تبلت اپل',  'هدفون سونی',  'ماوس لاجیتک',  'کیبورد مکانیکی',  'مانیتور الجی',  'دوربین کانن'  ];  let searchTimeout;  searchInput.addEventListener('keyup', function(e) {  // پاک کردن timeout قبلی  clearTimeout(searchTimeout);    // ایجاد timeout جدید برای جلوگیری از جستجوی بیش از حد  searchTimeout = setTimeout(() => {  const searchTerm = this.value.trim().toLowerCase();    if (searchTerm.length === 0) {  searchResults.style.display = 'none';  return;  }    // فیلتر کردن نتایج  const filteredResults = products.filter(product =>  product.toLowerCase().includes(searchTerm)  );    // نمایش نتایج  displayResults(filteredResults);  }, 300); // تاخیر 300 میلی‌ثانیه  });  // Event برای وقتی کاربر جستجو را رها می‌کند  searchInput.addEventListener('blur', function() {  setTimeout(() => {  searchResults.style.display = 'none';  }, 200);  });  // Event برای وقتی کاربر دوباره روی جستجو کلیک می‌کند  searchInput.addEventListener('focus', function() {  if (this.value.trim().length > 0) {  searchResults.style.display = 'block';  }  });  function displayResults(results) {  searchResults.innerHTML = '';    if (results.length === 0) {  searchResults.innerHTML = '<div class="result-item">نتیجه‌ای یافت نشد</div>';  } else {  results.forEach(result => {  const resultItem = document.createElement('div');  resultItem.className = 'result-item';  resultItem.textContent = result;    resultItem.addEventListener('click', function() {  searchInput.value = result;  searchResults.style.display = 'none';  });    searchResults.appendChild(resultItem);  });  }    searchResults.style.display = 'block';  }  </script>  </body>  </html> |

جمع‌بندی

Click: برای تعامل مستقیم کاربر

Mouseenter/Mouseleave: برای effects و hover states

Keydown: برای تشخیص فشردن کلیدها

Keyup: برای اعتبارسنجی real-time و جستجو

## نحوه css دادن با js

تغییر استایل مستقیم (Inline Styles)

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>آموزش CSS با JS</title>  <style>  .box {  width: 200px;  height: 200px;  margin: 20px;  padding: 20px;  border: 2px solid #333;  text-align: center;  line-height: 200px;  transition: all 0.3s ease;  }  button {  padding: 10px 20px;  margin: 5px;  cursor: pointer;  border: none;  border-radius: 5px;  background-color: #007bff;  color: white;  }  </style>  </head>  <body>  <div id="myBox" class="box">این یک div است</div>    <button onclick="changeColor()">تغییر رنگ</button>  <button onclick="changeSize()">تغییر سایز</button>  <button onclick="resetStyles()">بازنشانی</button>  <script>  const box = document.getElementById('myBox');  function changeColor() {  // تغییر استایل مستقیم  box.style.backgroundColor = '#ff6b6b';  box.style.color = 'white';  box.style.borderColor = '#ff4757';  }  function changeSize() {  box.style.width = '300px';  box.style.height = '150px';  box.style.lineHeight = '150px';  box.style.fontSize = '20px';  }  function resetStyles() {  // حذف استایل‌های inline  box.style.backgroundColor = '';  box.style.color = '';  box.style.borderColor = '';  box.style.width = '';  box.style.height = '';  box.style.lineHeight = '';  box.style.fontSize = '';  }  </script>  </body>  </html> |

اضافه کردن و حذف کلاس‌های CSS

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>مدیریت کلاس با JS</title>  <style>  .card {  width: 300px;  padding: 20px;  margin: 20px;  border: 2px solid #ddd;  border-radius: 10px;  transition: all 0.3s ease;  }    .active {  background-color: #e3f2fd;  border-color: #2196f3;  box-shadow: 0 4px 8px rgba(0,0,0,0.1);  }    .highlight {  background-color: #fff3cd;  border-color: #ffc107;  }    .large-text {  font-size: 24px;  font-weight: bold;  }    .hidden {  display: none;  }    .rotate {  transform: rotate(5deg);  }  </style>  </head>  <body>  <div id="myCard" class="card">  <h3>کارت نمونه</h3>  <p>این یک کارت نمونه است که می‌توانیم استایل آن را تغییر دهیم.</p>  </div>  <button onclick="toggleActive()">فعال/غیرفعال</button>  <button onclick="addHighlight()">هایلایت</button>  <button onclick="removeHighlight()">حذف هایلایت</button>  <button onclick="toggleVisibility()">نمایش/مخفی</button>  <button onclick="addMultipleClasses()">چند کلاس</button>  <script>  const card = document.getElementById('myCard');  function toggleActive() {  card.classList.toggle('active');  }  function addHighlight() {  card.classList.add('highlight');  }  function removeHighlight() {  card.classList.remove('highlight');  }  function toggleVisibility() {  card.classList.toggle('hidden');  }  function addMultipleClasses() {  card.classList.add('large-text', 'rotate');  }  // بررسی وجود کلاس  console.log('آیا کلاس active وجود دارد؟', card.classList.contains('active'));    // جایگزینی کلاس  function replaceClass() {  if (card.classList.contains('active')) {  card.classList.replace('active', 'highlight');  }  }  </script>  </body>  </html> |

ایجاد استایل داینامیک با JavaScript

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>استایل داینامیک</title>  <style>  .container {  padding: 20px;  margin: 20px;  border: 1px solid #ccc;  }  .color-preview {  width: 100px;  height: 100px;  margin: 10px;  border: 1px solid #333;  }  </style>  </head>  <body>  <div class="container">  <h3>تنظیم کننده استایل داینامیک</h3>    <label>عرض (px): </label>  <input type="range" id="widthRange" min="100" max="500" value="200">  <span id="widthValue">200px</span>    <br><br>    <label>ارتفاع (px): </label>  <input type="range" id="heightRange" min="50" max="300" value="100">  <span id="heightValue">100px</span>    <br><br>    <label>رنگ背景: </label>  <input type="color" id="colorPicker" value="#ff6b6b">    <br><br>    <label>شعاع حاشیه (px): </label>  <input type="range" id="borderRadiusRange" min="0" max="50" value="0">  <span id="borderRadiusValue">0px</span>    <br><br>    <div id="previewBox" class="color-preview"></div>  </div>  <script>  const previewBox = document.getElementById('previewBox');  const widthRange = document.getElementById('widthRange');  const heightRange = document.getElementById('heightRange');  const colorPicker = document.getElementById('colorPicker');  const borderRadiusRange = document.getElementById('borderRadiusRange');  const widthValue = document.getElementById('widthValue');  const heightValue = document.getElementById('heightValue');  const borderRadiusValue = document.getElementById('borderRadiusValue');  // تغییر عرض  widthRange.addEventListener('input', function() {  previewBox.style.width = this.value + 'px';  widthValue.textContent = this.value + 'px';  });  // تغییر ارتفاع  heightRange.addEventListener('input', function() {  previewBox.style.height = this.value + 'px';  heightValue.textContent = this.value + 'px';  });  // تغییر رنگ  colorPicker.addEventListener('input', function() {  previewBox.style.backgroundColor = this.value;  });  // تغییر شعاع حاشیه  borderRadiusRange.addEventListener('input', function() {  previewBox.style.borderRadius = this.value + 'px';  borderRadiusValue.textContent = this.value + 'px';  });  </script>  </body>  </html> |

ایجاد و مدیریت استایل‌شیت داینامیک

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>استایل‌شیت داینامیک</title>  <style>  .theme-box {  width: 200px;  height: 100px;  margin: 10px;  padding: 20px;  border: 2px solid;  display: inline-block;  text-align: center;  cursor: pointer;  }  </style>  </head>  <body>  <h3>انتخاب تم</h3>    <div class="theme-box" onclick="applyTheme('light')">تم روشن</div>  <div class="theme-box" onclick="applyTheme('dark')">تم تیره</div>  <div class="theme-box" onclick="applyTheme('blue')">تم آبی</div>  <div class="theme-box" onclick="applyTheme('green')">تم سبز</div>    <div id="content">  <h2>محتوای نمونه</h2>  <p>این یک محتوای نمونه است که استایل آن بر اساس تم انتخاب شده تغییر می‌کند.</p>  <button>دکمه نمونه</button>  </div>  <script>  function applyTheme(themeName) {  // حذف استایل‌شیت قبلی اگر وجود دارد  const oldStyle = document.getElementById('dynamic-style');  if (oldStyle) {  oldStyle.remove();  }  // ایجاد استایل‌شیت جدید  const style = document.createElement('style');  style.id = 'dynamic-style';    let cssRules = '';    switch(themeName) {  case 'light':  cssRules = `  body { background-color: white; color: black; }  #content { background-color: #f8f9fa; padding: 20px; }  button { background-color: #007bff; color: white; }  `;  break;    case 'dark':  cssRules = `  body { background-color: #343a40; color: white; }  #content { background-color: #495057; padding: 20px; }  button { background-color: #6c757d; color: white; }  `;  break;    case 'blue':  cssRules = `  body { background-color: #e3f2fd; color: #1565c0; }  #content { background-color: #bbdefb; padding: 20px; }  button { background-color: #1976d2; color: white; }  `;  break;    case 'green':  cssRules = `  body { background-color: #e8f5e8; color: #2e7d32; }  #content { background-color: #c8e6c9; padding: 20px; }  button { background-color: #388e3c; color: white; }  `;  break;  }    style.textContent = cssRules;  document.head.appendChild(style);  }  // اعتماد تم پیشفرض  applyTheme('light');  </script>  </body>  </html> |

مثال کاربردی: دکمه تغییر حالت روز/شب

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>دکمه تغییر حالت</title>  <style>  body {  margin: 0;  padding: 20px;  font-family: Arial, sans-serif;  transition: all 0.3s ease;  }    .container {  max-width: 800px;  margin: 0 auto;  padding: 20px;  }    .theme-toggle {  position: fixed;  top: 20px;  left: 20px;  padding: 10px 15px;  background-color: #007bff;  color: white;  border: none;  border-radius: 20px;  cursor: pointer;  z-index: 1000;  }    /\* استایل‌های پیشفرض (حالت روز) \*/  body.light-mode {  background-color: white;  color: black;  }    body.dark-mode {  background-color: #1a1a1a;  color: white;  }    .card {  background-color: #f8f9fa;  padding: 20px;  margin: 10px 0;  border-radius: 10px;  transition: all 0.3s ease;  }    .dark-mode .card {  background-color: #2d2d2d;  }  </style>  </head>  <body class="light-mode">  <button class="theme-toggle" onclick="toggleTheme()">حالت شب 🌙</button>    <div class="container">  <h1>سایت نمونه</h1>    <div class="card">  <h2>کارت اول</h2>  <p>این یک کارت نمونه است. محتوای کارت در اینجا نمایش داده می‌شود.</p>  </div>    <div class="card">  <h2>کارت دوم</h2>  <p>کارت دیگری برای نمایش قابلیت تغییر تم.</p>  </div>  </div>  <script>  function toggleTheme() {  const body = document.body;  const themeButton = document.querySelector('.theme-toggle');    if (body.classList.contains('light-mode')) {  // تغییر به حالت شب  body.classList.remove('light-mode');  body.classList.add('dark-mode');  themeButton.textContent = 'حالت روز ☀️';    // تغییر استایل‌های اضافی با JS  themeButton.style.backgroundColor = '#ffc107';  themeButton.style.color = 'black';  } else {  // تغییر به حالت روز  body.classList.remove('dark-mode');  body.classList.add('light-mode');  themeButton.textContent = 'حالت شب 🌙';    // بازگشت به استایل اولیه  themeButton.style.backgroundColor = '#007bff';  themeButton.style.color = 'white';  }  }  </script>  </body>  </html> |

جمع‌بندی روش‌ها

element.style.property: تغییر مستقیم استایل‌های inline

classList: مدیریت کلاس‌های CSS (add, remove, toggle, contains)

ایجاد استایل‌شیت داینامیک: برای تغییرات گسترده

تغییر استایل بر اساس رویدادها: تعامل با کاربر

## ساخت ماشین حساب با js

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>ماشین حساب JavaScript</title>  <style>  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }  body {  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  display: flex;  justify-content: center;  align-items: center;  min-height: 100vh;  padding: 20px;  }  .calculator {  background: #2c3e50;  border-radius: 20px;  padding: 25px;  box-shadow: 0 20px 40px rgba(0, 0, 0, 0.3);  width: 100%;  max-width: 400px;  }  .display {  background: #34495e;  border-radius: 15px;  padding: 20px;  margin-bottom: 20px;  text-align: left;  direction: ltr;  }  #result {  width: 100%;  background: none;  border: none;  color: white;  font-size: 2.5em;  text-align: right;  outline: none;  direction: ltr;  }  #operation {  color: #bdc3c7;  font-size: 1.2em;  text-align: right;  margin-bottom: 10px;  min-height: 25px;  direction: ltr;  }  .buttons {  display: grid;  grid-template-columns: repeat(4, 1fr);  gap: 15px;  }  button {  border: none;  border-radius: 15px;  padding: 20px;  font-size: 1.4em;  font-weight: bold;  cursor: pointer;  transition: all 0.2s ease;  color: white;  }  button:active {  transform: scale(0.95);  }  .number {  background: #34495e;  }  .number:hover {  background: #4a6572;  }  .operator {  background: #e67e22;  }  .operator:hover {  background: #f39c12;  }  .equals {  background: #27ae60;  }  .equals:hover {  background: #2ecc71;  }  .clear {  background: #e74c3c;  }  .clear:hover {  background: #c0392b;  }  .zero {  grid-column: span 2;  }  .memory-btn {  background: #8e44ad;  font-size: 1em;  padding: 15px;  }  .memory-btn:hover {  background: #9b59b6;  }  .memory-display {  color: #bdc3c7;  font-size: 0.9em;  text-align: center;  margin-top: 10px;  min-height: 20px;  }  </style>  </head>  <body>  <div class="calculator">  <div class="display">  <div id="operation"></div>  <input type="text" id="result" value="0" readonly>  </div>    <div class="buttons">  <!-- ردیف اول - دکمه های حافظه -->  <button class="memory-btn" onclick="memoryClear()">MC</button>  <button class="memory-btn" onclick="memoryRecall()">MR</button>  <button class="memory-btn" onclick="memoryAdd()">M+</button>  <button class="memory-btn" onclick="memorySubtract()">M-</button>    <!-- ردیف دوم -->  <button class="clear" onclick="clearAll()">C</button>  <button class="clear" onclick="clearEntry()">CE</button>  <button class="operator" onclick="deleteLast()">⌫</button>  <button class="operator" onclick="appendOperator('/')">÷</button>    <!-- ردیف سوم -->  <button class="number" onclick="appendNumber('7')">7</button>  <button class="number" onclick="appendNumber('8')">8</button>  <button class="number" onclick="appendNumber('9')">9</button>  <button class="operator" onclick="appendOperator('\*')">×</button>    <!-- ردیف چهارم -->  <button class="number" onclick="appendNumber('4')">4</button>  <button class="number" onclick="appendNumber('5')">5</button>  <button class="number" onclick="appendNumber('6')">6</button>  <button class="operator" onclick="appendOperator('-')">−</button>    <!-- ردیف پنجم -->  <button class="number" onclick="appendNumber('1')">1</button>  <button class="number" onclick="appendNumber('2')">2</button>  <button class="number" onclick="appendNumber('3')">3</button>  <button class="operator" onclick="appendOperator('+')">+</button>    <!-- ردیف ششم -->  <button class="number zero" onclick="appendNumber('0')">0</button>  <button class="number" onclick="appendDecimal()">.</button>  <button class="equals" onclick="calculate()">=</button>  </div>    <div class="memory-display" id="memoryDisplay"></div>  </div>  <script>  // متغیرهای  let currentInput = '0';  let previousInput = '';  let operation = null;  let shouldResetScreen = false;  let memory = 0;  // عناصر DOM  const resultDisplay = document.getElementById('result');  const operationDisplay = document.getElementById('operation');  const memoryDisplay = document.getElementById('memoryDisplay');  // به روزرسانی نمایشگر  function updateDisplay() {  resultDisplay.value = currentInput;  operationDisplay.textContent = previousInput + (operation || '');  updateMemoryDisplay();  }  // به روزرسانی نمایشگر حافظه  function updateMemoryDisplay() {  memoryDisplay.textContent = memory !== 0 ? `M: ${memory}` : '';  }  // اضافه کردن عدد  function appendNumber(number) {  if (currentInput === '0' || shouldResetScreen) {  currentInput = number;  shouldResetScreen = false;  } else {  currentInput += number;  }  updateDisplay();  }  // اضافه کردن اعشار  function appendDecimal() {  if (shouldResetScreen) {  currentInput = '0.';  shouldResetScreen = false;  } else if (!currentInput.includes('.')) {  currentInput += '.';  }  updateDisplay();  }  // انتخاب عملگر  function appendOperator(nextOperation) {  if (operation !== null && !shouldResetScreen) {  calculate();  }    previousInput = currentInput;  operation = nextOperation;  shouldResetScreen = true;  updateDisplay();  }  // محاسبه  function calculate() {  if (operation === null || shouldResetScreen) return;    const prev = parseFloat(previousInput);  const current = parseFloat(currentInput);    if (isNaN(prev) || isNaN(current)) return;    let computation;  switch (operation) {  case '+':  computation = prev + current;  break;  case '-':  computation = prev - current;  break;  case '\*':  computation = prev \* current;  break;  case '/':  if (current === 0) {  alert('خطا: تقسیم بر صفر مجاز نیست!');  clearAll();  return;  }  computation = prev / current;  break;  default:  return;  }    currentInput = roundResult(computation).toString();  operation = null;  previousInput = '';  shouldResetScreen = true;  updateDisplay();  }  // گرد کردن نتیجه  function roundResult(num) {  return Math.round(num \* 100000000) / 100000000;  }  // پاک کردن همه  function clearAll() {  currentInput = '0';  previousInput = '';  operation = null;  shouldResetScreen = false;  updateDisplay();  }  // پاک کردن ورودی جاری  function clearEntry() {  currentInput = '0';  updateDisplay();  }  // حذف آخرین رقم  function deleteLast() {  if (currentInput.length === 1 || (currentInput.length === 2 && currentInput.startsWith('-'))) {  currentInput = '0';  } else {  currentInput = currentInput.slice(0, -1);  }  updateDisplay();  }  // توابع حافظه  function memoryClear() {  memory = 0;  updateMemoryDisplay();  }  function memoryRecall() {  currentInput = memory.toString();  shouldResetScreen = true;  updateDisplay();  }  function memoryAdd() {  const current = parseFloat(currentInput);  if (!isNaN(current)) {  memory += current;  updateMemoryDisplay();  }  }  function memorySubtract() {  const current = parseFloat(currentInput);  if (!isNaN(current)) {  memory -= current;  updateMemoryDisplay();  }  }  // مدیریت صفحه کلید  document.addEventListener('keydown', function(event) {  if (event.key >= '0' && event.key <= '9') {  appendNumber(event.key);  } else if (event.key === '.') {  appendDecimal();  } else if (event.key === '+' || event.key === '-' || event.key === '\*' || event.key === '/') {  appendOperator(event.key);  } else if (event.key === 'Enter' || event.key === '=') {  event.preventDefault();  calculate();  } else if (event.key === 'Escape') {  clearAll();  } else if (event.key === 'Backspace') {  deleteLast();  }  });  // مقداردهی اولیه  updateDisplay();  </script>  </body>  </html> |

1. عملیات اصلی

جمع، تفریق، ضرب و تقسیم

پشتیبانی از اعداد اعشاری

مدیریت خطا (تقسیم بر صفر)

2. قابلیت‌های پیشرفته

حافظه (MC, MR, M+, M-)

پاک کردن انتخابی (C, CE)

حذف آخرین رقم (⌫)

3. طراحی واکنش‌گرا

طراحی مدرن و زیبا

سازگار با موبایل و تبلت

انیمیشن‌های روان

4. پشتیبانی از صفحه کلید

استفاده از اعداد صفحه کلید

Enter برای مساوی

Escape برای پاک کردن

Backspace برای حذف

## معرفی متد های string

Substr

substr- دریافت بخشی از رشته

|  |
| --- |
| let text = "Hello World!";  console.log(text.substr(0, 5)); // "Hello"  console.log(text.substr(6)); // "World!"  console.log(text.substr(6, 5)); // "World"  console.log(text.substr(-6)); // "World!" (از انتها شمرده می‌شود)  // مثال کاربردی  let email = "user@example.com";  let username = email.substr(0, email.indexOf('@'));  console.log(username); // "user" |

Substring

دریافت زیررشته

|  |
| --- |
| let text = "JavaScript Programming";  console.log(text.substring(0, 10)); // "JavaScript"  console.log(text.substring(11)); // "Programming"  console.log(text.substring(4, 10)); // "Script"  // تفاوت با substr - پارامتر دوم طول نیست، index پایان است  let str = "HelloWorld";  console.log(str.substring(3, 7)); // "loWo" (از index 3 تا 7)  console.log(str.substr(3, 7)); // "loWorld" (از index 3 به طول 7)  // اگر start > end باشد، آنها را جابجا می‌کند  console.log(text.substring(10, 0)); // "JavaScript" |

Slice

برش دادن رشته

|  |
| --- |
| let text = "Hello Beautiful World";  console.log(text.slice(6, 15)); // "Beautiful"  console.log(text.slice(6)); // "Beautiful World"  console.log(text.slice(-5)); // "World" (از انتها)  console.log(text.slice(-10, -6)); // "iful"  // مثال عملی - استخراج دامنه از ایمیل  let email = "test@gmail.com";  let domain = email.slice(email.indexOf('@') + 1);  console.log(domain); // "gmail.com"  // تفاوت slice با substring در پذیرش اعداد منفی  let str = "abcdef";  console.log(str.slice(2, -1)); // "cde"  console.log(str.substring(2, -1)); // "ab" (منفی را به 0 تبدیل می‌کند) |

Tolowercase

تبدیل به حروف کوچک

|  |
| --- |
| let text = "Hello WORLD!";  console.log(text.toLowerCase()); // "hello world!"  // مثال کاربردی - مقایسه بدون در نظر گرفتن بزرگی و کوچکی حروف  let userInput = "Admin";  let storedValue = "admin";  if (userInput.toLowerCase() === storedValue.toLowerCase()) {  console.log("ورودی صحیح است");  }  // نرمال سازی داده‌های کاربر  let userName = "JoHn DoE";  let normalizedName = userName.toLowerCase();  console.log(normalizedName); // "john doe" |

toUppercase

تبدیل به حروف بزرگ

|  |
| --- |
| let text = "hello world!";  console.log(text.toUpperCase()); // "HELLO WORLD!"  // مثال کاربردی - ایجاد کدهای شناسایی  let productName = "laptop";  let productCode = productName.toUpperCase() + "\_001";  console.log(productCode); // "LAPTOP\_001"  // هشدارها و پیام‌های مهم  let warning = "attention: system update required";  let importantWarning = warning.toUpperCase();  console.log(importantWarning); // "ATTENTION: SYSTEM UPDATE REQUIRED" |

Replace

- جایگزینی بخشی از رشته

|  |
| --- |
| let text = "I love apples. Apples are delicious.";  // جایگزینی اولین مورد  console.log(text.replace("apples", "oranges")); // "I love oranges. Apples are delicious."  // جایگزینی همه موارد با regex  console.log(text.replace(/apples/gi, "oranges")); // "I love oranges. oranges are delicious."  // مثال کاربردی - فرمت‌دهی متن  let phone = "123-456-7890";  let formattedPhone = phone.replace(/-/g, " ");  console.log(formattedPhone); // "123 456 7890"  // حذف فاصله‌های اضافی  let messyText = "Hello World !";  let cleanText = messyText.replace(/\s+/g, " ");  console.log(cleanText); // "Hello World !" |

Trim

حذف فاصله از ابتدا و انتها

|  |
| --- |
| let text = " Hello World! ";  console.log(text.trim()); // "Hello World!"  // انواع trim  let spacedText = " test ";  console.log(spacedText.trim()); // "test"  console.log(spacedText.trimStart()); // "test "  console.log(spacedText.trimEnd()); // " test"  // مثال کاربردی - اعتبارسنجی فرم  let userInput = " admin ";  if (userInput.trim() === "admin") {  console.log("ورودی معتبر است");  }  // تمیز کردن داده‌های ورودی کاربر  function cleanInput(input) {  return input.trim().toLowerCase();  }  let username = cleanInput(" ADMIN ");  console.log(username); // "admin" |

charAt

دریافت کاراکتر در موقعیت مشخص

|  |
| --- |
| let text = "JavaScript";  console.log(text.charAt(0)); // "J"  console.log(text.charAt(4)); // "S"  console.log(text.charAt(10)); // "" (رشته خالی)  // مقایسه با براکت نویسی  console.log(text[0]); // "J"  console.log(text[20]); // undefined  // مثال کاربردی - بررسی اولین کاراکتر  let filename = "document.pdf";  if (filename.charAt(0) === '.') {  console.log("این یک فایل مخفی است");  }  // ایجاد اختصار نام  let fullName = "John Doe";  let initials = fullName.charAt(0) + fullName.charAt(5);  console.log(initials); // "JD" |

Indexof

پیدا کردن index اولین occurrence

|  |
| --- |
| let text = "Hello world, welcome to the world.";  console.log(text.indexOf("world")); // 6  console.log(text.indexOf("World")); // -1 (پیدا نشد)  console.log(text.indexOf("o")); // 4  console.log(text.indexOf("o", 5)); // 7 (جستجو از index 5)  // مثال کاربردی - بررسی وجود کلمه  let sentence = "The quick brown fox jumps over the lazy dog";  let searchWord = "fox";  if (sentence.indexOf(searchWord) !== -1) {  console.log("کلمه پیدا شد");  }  // استخراج بخشی از URL  let url = "https://example.com/products?id=123";  let idIndex = url.indexOf("id=");  if (idIndex !== -1) {  let productId = url.substring(idIndex + 3);  console.log(productId); // "123"  } |

Lastindexof

پیدا کردن index آخرین occurrence

|  |
| --- |
| let text = "Hello world, welcome to the world.";  console.log(text.lastIndexOf("world")); // 24  console.log(text.lastIndexOf("o")); // 25  console.log(text.lastIndexOf("o", 20)); // 17  // مثال کاربردی - استخراج پسوند فایل  let filename = "document.backup.pdf";  let lastDotIndex = filename.lastIndexOf(".");  let extension = filename.substring(lastDotIndex + 1);  console.log(extension); // "pdf"  // پیدا کردن آخرین occurrence از یک کاراکتر  let path = "/home/user/documents/file.txt";  let lastSlashIndex = path.lastIndexOf("/");  let fileName = path.substring(lastSlashIndex + 1);  console.log(fileName); // "file.txt" |

Include

بررسی وجود زیررشته

|  |
| --- |
| let text = "Hello world, welcome to JavaScript!";  console.log(text.includes("world")); // true  console.log(text.includes("World")); // false  console.log(text.includes("world", 10)); // false (جستجو از index 10)  // مثال کاربردی - فیلتر کردن محتوا  let articles = [  "Introduction to JavaScript",  "Python Programming Guide",  "JavaScript Advanced Topics",  "Java Basics Tutorial"  ];  let javascriptArticles = articles.filter(article =>  article.includes("JavaScript")  );  console.log(javascriptArticles);  // بررسی ایمیل  let email = "user@company.com";  if (email.includes("@") && email.includes(".")) {  console.log("قالب ایمیل صحیح است");  } |

Startwith

بررسی شروع رشته

|  |
| --- |
| let text = "Hello world!";  console.log(text.startsWith("Hello")); // true  console.log(text.startsWith("world")); // false  console.log(text.startsWith("world", 6)); // true  // مثال کاربردی - بررسی پروتکل URL  let url = "https://example.com";  if (url.startsWith("https://")) {  console.log("اتصال امن است");  }  // دسته‌بندی فایل‌ها  let files = ["document.pdf", "image.jpg", "script.js", "style.css"];  let jsFiles = files.filter(file => file.startsWith("script"));  let cssFiles = files.filter(file => file.endsWith(".css"));  console.log(jsFiles); // ["script.js"]  console.log(cssFiles); // ["style.css"] |

Endwith

بررسی پایان رشته

|  |
| --- |
| let text = "Hello world!";  console.log(text.endsWith("world!")); // true  console.log(text.endsWith("Hello")); // false  console.log(text.endsWith("world", 11)); // true (در نظر گرفتن 11 کاراکتر اول)  // مثال کاربردی - بررسی نوع فایل  let filename = "document.pdf";  if (filename.endsWith(".pdf")) {  console.log("این یک فایل PDF است");  } else if (filename.endsWith(".jpg") || filename.endsWith(".png")) {  console.log("این یک فایل تصویری است");  }  // اعتبارسنجی شماره تلفن  let phoneNumber = "+981234567890";  if (phoneNumber.endsWith("0")) {  console.log("شماره تلفن معتبر است");  } |

مثال جامع و کاربردی

|  |
| --- |
| // سیستم پردازش و اعتبارسنجی متن  function textProcessor(inputText) {  // تمیز کردن متن  let cleanedText = inputText.trim();    // بررسی شرایط مختلف  let stats = {  length: cleanedText.length,  startsWithHello: cleanedText.toLowerCase().startsWith('hello'),  endsWithDot: cleanedText.endsWith('.'),  containsJavaScript: cleanedText.toLowerCase().includes('javascript'),  firstWord: cleanedText.substring(0, cleanedText.indexOf(' ') || cleanedText.length),  lastWord: cleanedText.substring(cleanedText.lastIndexOf(' ') + 1)  };    return stats;  }  // تست تابع  let sampleText = " Hello World! Welcome to JavaScript programming. ";  let result = textProcessor(sampleText);  console.log("آمار متن:");  console.log(`طول متن: ${result.length}`);  console.log(`شروع با Hello: ${result.startsWithHello}`);  console.log(`پایان با نقطه: ${result.endsWithDot}`);  console.log(`شامل JavaScript: ${result.containsJavaScript}`);  console.log(`کلمه اول: ${result.firstWord}`);  console.log(`کلمه آخر: ${result.lastWord}`);  // مثال دیگر: ایجاد slug از عنوان  function createSlug(title) {  return title  .toLowerCase()  .trim()  .replace(/\s+/g, '-')  .replace(/[^\w\-]+/g, '')  .replace(/\-\-+/g, '-')  .replace(/^-+/, '')  .replace(/-+$/, '');  }  let articleTitle = " Introduction to JavaScript Programming! ";  let slug = createSlug(articleTitle);  console.log(slug); // "introduction-to-javascript-programming" |

جمع‌بندی متدها

|  |  |  |
| --- | --- | --- |
| متد | کاربرد | مقدار بازگشتی |
| `substr()` | دریافت بخشی از رشته با شروع و طول | زیررشته |
| `substring()` | دریافت زیررشته با شروع و پایان | زیررشته |
| `slice()` | برش رشته با پشتیبانی از اعداد منفی | زیررشته |
| `toLowerCase()` | تبدیل به حروف کوچک | رشته |
| `toUpperCase()` | تبدیل به حروف بزرگ | رشته |
| `replace()` | جایگزینی بخشی از رشته | رشته جدید |
| `trim()` | حذف فاصله از دو طرف | رشته تمیز شده |
| `charAt()` | دریافت کاراکتر در موقعیت | کاراکتر |
| `indexOf()` | پیدا کردن اولین موقعیت | index یا -1 |
| `lastIndexOf()` | پیدا کردن آخرین موقعیت | index یا -1 |
| `includes()` | بررسی وجود زیررشته | true/false |
| `startsWith()` | بررسی شروع رشته | true/false |
| `endsWith()` | بررسی پایان رشته | true/false |

## نحوه دسترسی به کارکتر های string

استفاده از Brackets Notation []

|  |
| --- |
| let text = "JavaScript";  // دسترسی به کاراکترهای مختلف  console.log(text[0]); // "J"  console.log(text[4]); // "S"  console.log(text[9]); // "t"  // دسترسی به کاراکترهای خارج از محدوده  console.log(text[10]); // undefined  console.log(text[-1]); // undefined  // حلقه زدن through کاراکترها  for (let i = 0; i < text.length; i++) {  console.log(`کاراکتر ${i}: ${text[i]}`);  }  // مثال کاربردی - شمارش نوع کاراکترها  function countCharacters(str) {  let letters = 0;  let digits = 0;  let spaces = 0;    for (let char of str) {  if (char >= 'a' && char <= 'z' || char >= 'A' && char <= 'Z') {  letters++;  } else if (char >= '0' && char <= '9') {  digits++;  } else if (char === ' ') {  spaces++;  }  }    return { letters, digits, spaces };  }  let result = countCharacters("Hello World 123!");  console.log(result); // { letters: 10, digits: 3, spaces: 2 } |

استفاده از charAt

|  |
| --- |
| let text = "Programming";  console.log(text.charAt(0)); // "P"  console.log(text.charAt(3)); // "g"  console.log(text.charAt(20)); // "" (رشته خالی)  // مقایسه با bracket notation  console.log(text[20]); // undefined  // مثال کاربردی - ایجاد اختصار  function getInitials(fullName) {  let names = fullName.split(' ');  let initials = '';    for (let name of names) {  if (name.length > 0) {  initials += name.charAt(0).toUpperCase();  }  }    return initials;  }  console.log(getInitials("john doe")); // "JD"  console.log(getInitials("mary ann smith")); // "MAS" |

استفاده از charCodeAt- دریافت کد ASCII

|  |
| --- |
| let text = "ABCabc";  console.log(text.charCodeAt(0)); // 65 (A)  console.log(text.charCodeAt(1)); // 66 (B)  console.log(text.charCodeAt(3)); // 97 (a)  // مثال کاربردی - بررسی نوع کاراکتر  function analyzeString(str) {  for (let i = 0; i < str.length; i++) {  let char = str[i];  let code = str.charCodeAt(i);    console.log(`کاراکتر: ${char}, کد: ${code}, نوع: ${  code >= 48 && code <= 57 ? 'عدد' :  code >= 65 && code <= 90 ? 'حرف بزرگ' :  code >= 97 && code <= 122 ? 'حرف کوچک' : 'سایر'  }`);  }  }  analyzeString("Hello123!"); |

استفاده از String.fromCharCode() - ایجاد رشته از کدها

|  |
| --- |
| // ایجاد رشته از کدهای ASCII  console.log(String.fromCharCode(65, 66, 67)); // "ABC"  console.log(String.fromCharCode(97, 98, 99)); // "abc"  // مثال کاربردی - تولید رشته تصادفی  function generateRandomString(length) {  let result = '';  for (let i = 0; i < length; i++) {  // اعداد 65-90 (حروف بزرگ) و 97-122 (حروف کوچک)  let randomCode = Math.random() < 0.5 ?  Math.floor(Math.random() \* 26) + 65 :  Math.floor(Math.random() \* 26) + 97;  result += String.fromCharCode(randomCode);  }  return result;  }  console.log(generateRandomString(8)); // مثلاً: "XyZabCdE" |

## معرفی متد split

اساس کار متد split

|  |
| --- |
| let text = "apple,banana,orange,grape";  // تقسیم با جداکننده کاما  let fruits = text.split(",");  console.log(fruits); // ["apple", "banana", "orange", "grape"]  // تقسیم با جداکننده فاصله  let sentence = "Hello world from JavaScript";  let words = sentence.split(" ");  console.log(words); // ["Hello", "world", "from", "JavaScript"]  // تقسیم هر کاراکتر  let chars = "Hello".split("");  console.log(chars); // ["H", "e", "l", "l", "o"] |

محدود کردن تعداد بخش‌ها

|  |
| --- |
| let text = "one,two,three,four,five";  // محدود کردن به 3 بخش  let limited = text.split(",", 3);  console.log(limited); // ["one", "two", "three"]  // مثال کاربردی - استخراج نام دامنه  let email = "username@domain.com";  let parts = email.split("@", 2);  console.log(parts); // ["username", "domain.com"]  let domainParts = parts[1].split(".", 2);  console.log(domainParts); // ["domain", "com"] |

استفاده از Regex در split

|  |
| --- |
| let text = "apple, banana; orange. grape";  // تقسیم با چند جداکننده مختلف  let fruits = text.split(/[,;.]\s\*/);  console.log(fruits); // ["apple", "banana", "orange", "grape"]  // تقسیم بر اساس اعداد  let data = "item123price456quantity789";  let items = data.split(/\d+/);  console.log(items); // ["item", "price", "quantity", ""]  // تقسیم بر اساس فاصله‌های متعدد  let messyText = "Hello World !";  let cleanWords = messyText.split(/\s+/);  console.log(cleanWords); // ["Hello", "World", "!"] |

مثال‌های کاربردی و واقعی

مثال 1: پردازش CSV داده

|  |
| --- |
| // شبیه‌سازی داده CSV  let csvData = "name,age,city,email\nJohn,30,New York,john@email.com\nJane,25,London,jane@email.com";  function parseCSV(csv) {  let lines = csv.split("\n");  let result = [];    // عنوان‌ها (ستون‌ها)  let headers = lines[0].split(",");    // داده‌ها  for (let i = 1; i < lines.length; i++) {  if (lines[i].trim() === "") continue;    let obj = {};  let currentLine = lines[i].split(",");    for (let j = 0; j < headers.length; j++) {  obj[headers[j]] = currentLine[j];  }    result.push(obj);  }    return result;  }  let parsedData = parseCSV(csvData);  console.log(parsedData);  // [  // { name: "John", age: "30", city: "New York", email: "john@email.com" },  // { name: "Jane", age: "25", city: "London", email: "jane@email.com" }  // ] |

مثال 2: تجزیه و تحلیل URL

|  |
| --- |
| function parseURL(url) {  // جدا کردن پروتکل  let protocol = url.split('://')[0];    // جدا کردن domain و path  let withoutProtocol = url.split('://')[1];  let domain = withoutProtocol.split('/')[0];  let path = withoutProtocol.split('/').slice(1).join('/');    // جدا کردن query parameters  let pathParts = path.split('?');  let actualPath = pathParts[0];  let queryString = pathParts[1] || '';    // تجزیه query parameters  let queryParams = {};  if (queryString) {  queryString.split('&').forEach(pair => {  let [key, value] = pair.split('=');  queryParams[key] = value;  });  }    return {  protocol,  domain,  path: actualPath,  queryParams  };  }  let url = "https://example.com/products/page?category=electronics&sort=price";  let parsed = parseURL(url);  console.log(parsed); |

مثال 3: پردازش متن و آمار

|  |
| --- |
| function textStatistics(text) {  // تقسیم به کلمات  let words = text.split(/\s+/).filter(word => word.length > 0);    // تقسیم به جملات  let sentences = text.split(/[.!?]+/).filter(sentence => sentence.trim().length > 0);    // تقسیم به پاراگراف‌ها  let paragraphs = text.split(/\n+/).filter(para => para.trim().length > 0);    // محاسبه میانگین طول کلمات  let totalWordLength = words.reduce((sum, word) => sum + word.length, 0);  let averageWordLength = totalWordLength / words.length;    return {  totalCharacters: text.length,  totalWords: words.length,  totalSentences: sentences.length,  totalParagraphs: paragraphs.length,  averageWordLength: averageWordLength.toFixed(2),  longestWord: words.reduce((longest, current) =>  current.length > longest.length ? current : longest, ""  )  };  }  let sampleText = `این یک متن نمونه است.  این متن شامل چندین جمله می‌باشد!  آیا این کار می‌کند؟ بله، به خوبی کار می‌کند.`;  let stats = textStatistics(sampleText);  console.log(stats); |

مثال 4: اعتبارسنجی و فرمت‌دهی

|  |
| --- |
| // اعتبارسنجی شماره تلفن  function validatePhoneNumber(phone) {  let cleaned = phone.split(' ').join('').split('-').join('');    if (cleaned.length !== 11) {  return "شماره تلفن باید 11 رقمی باشد";  }    if (!cleaned.startsWith('09')) {  return "شماره تلفن باید با 09 شروع شود";  }    return "شماره تلفن معتبر است";  }  console.log(validatePhoneNumber("0912-345-6789")); // معتبر  console.log(validatePhoneNumber("9123456789")); // نامعتبر  // فرمت‌دهی شماره تلفن  function formatPhoneNumber(phone) {  let cleaned = phone.split(' ').join('').split('-').join('');    if (cleaned.length === 11) {  return cleaned.split('').map((char, index) => {  if (index === 4 || index === 7) return '-' + char;  return char;  }).join('');  }    return phone;  }  console.log(formatPhoneNumber("09123456789")); // "0912-345-6789" |

مثال 5: سیستم تگ‌گذاری

|  |
| --- |
| function processTags(input) {  // تقسیم تگ‌ها با جداکننده‌های مختلف  let tags = input.split(/[,\s]+/).filter(tag => tag.length > 0);    // حذف تگ‌های تکراری  let uniqueTags = [...new Set(tags)];    // نرمال سازی (حروف کوچک و حذف فاصله)  let normalizedTags = uniqueTags.map(tag =>  tag.trim().toLowerCase().replace(/\s+/g, '-')  );    return normalizedTags;  }  let tagInput = "JavaScript, Web Development, javascript, web development, CSS";  let processedTags = processTags(tagInput);  console.log(processedTags); // ["javascript", "web-development", "css"]  // ایجاد HTML برای تگ‌ها  function createTagHTML(tags) {  return tags.map(tag =>  `<span class="tag">${tag}</span>`  ).join(' ');  }  let tagHTML = createTagHTML(processedTags);  console.log(tagHTML); |

## آموزش ساخت modal

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>مودال ساده</title>  <style>  body {  font-family: Arial, sans-serif;  padding: 50px;  text-align: center;  background-color: #f5f5f5;  }  .open-btn {  padding: 15px 30px;  background: #007bff;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  font-size: 18px;  }  .modal {  display: none;  position: fixed;  top: 0;  left: 0;  width: 100%;  height: 100%;  background: rgba(0, 0, 0, 0.5);  }  .modal-content {  position: absolute;  top: 50%;  left: 50%;  transform: translate(-50%, -50%);  background: white;  padding: 30px;  border-radius: 10px;  width: 300px;  text-align: center;  }  .close-btn {  padding: 10px 20px;  background: #dc3545;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  margin-top: 15px;  }  </style>  </head>  <body>  <button class="open-btn" data-modal-open="myModal">  باز کردن مودال  </button>  <div id="myModal" class="modal" data-modal>  <div class="modal-content">  <h2>مودال ساده</h2>  <p>این یک مودال بسیار ساده است</p>  <button class="close-btn" data-modal-close>  بستن  </button>  </div>  </div>  <script>  // انتخاب تمام دکمه‌های باز کردن مودال  const openButtons = document.querySelectorAll('[data-modal-open]');    // انتخاب تمام دکمه‌های بستن مودال  const closeButtons = document.querySelectorAll('[data-modal-close]');  // اضافه کردن event listener برای دکمه‌های باز کردن  openButtons.forEach(button => {  button.addEventListener('click', function() {  const modalId = this.getAttribute('data-modal-open');  const modal = document.getElementById(modalId);  modal.style.display = 'block';  });  });  // اضافه کردن event listener برای دکمه‌های بستن  closeButtons.forEach(button => {  button.addEventListener('click', function() {  const modal = this.closest('[data-modal]');  modal.style.display = 'none';  });  });  // بستن مودال با کلیک خارج از آن  document.addEventListener('click', function(event) {  if (event.target.hasAttribute('data-modal')) {  event.target.style.display = 'none';  }  });  </script>  </body>  </html> |

توضیح کد:

data attributes استفاده شده:

data-modal-open="myModal" - برای دکمه باز کردن مودال

data-modal - برای خود مودال

data-modal-close - برای دکمه بستن مودال

ID استفاده شده:

id="myModal" - برای شناسایی مودال

کارکرد:

دکمه با data-modal-open هنگام کلیک، مودال مربوطه را باز می‌کند

دکمه با data-modal-close مودال والد خود را می‌بندد

کلیک روی پس‌زمینه (مودال) آن را می‌بندد

## معرفی Math در js

PI

|  |
| --- |
| console.log(Math.PI); // 3.141592653589793  // مثال: محاسبه محیط دایره  let radius = 5;  let circumference = 2 \* Math.PI \* radius;  console.log("محیط دایره:", circumference); // 31.41592653589793  // مثال: محاسبه مساحت دایره  let area = Math.PI \* Math.pow(radius, 2);  console.log("مساحت دایره:", area); // 78.53981633974483 |

E

عدد اویلر (e) یک عدد ثابت ریاضی است که تقریباً برابر با 2.71828 می‌باشد. این عدد پایه و اساس لگاریتم طبیعی است و در بسیاری از پدیده‌های طبیعی و ریاضی ظاهر می‌شود.

|  |
| --- |
| console.log(Math.E); // 2.718281828459045  // مثال: محاسبه رشد نمایی  let time = 2;  let growthRate = 0.1;  let finalValue = Math.pow(Math.E, growthRate \* time);  console.log("مقدار نهایی:", finalValue); // 1.2214027581601699 |

این یعنی اگر چیزی با نرخ رشد 10% به صورت پیوسته رشد کند، بعد از 2 واحد زمانی به حدود 1.2214 برابر مقدار اولیه خود می‌رسد.

Sqrt

|  |
| --- |
| console.log(Math.sqrt(16)); // 4  console.log(Math.sqrt(2)); // 1.4142135623730951  console.log(Math.sqrt(9)); // 3  // مثال: محاسبه وتر مثلث قائم الزاویه  let a = 3;  let b = 4;  let c = Math.sqrt(a \* a + b \* b);  console.log("طول وتر:", c); // 5  // مثال: حل معادله درجه دوم  function solveQuadratic(a, b, c) {  let discriminant = Math.sqrt(b \* b - 4 \* a \* c);  let x1 = (-b + discriminant) / (2 \* a);  let x2 = (-b - discriminant) / (2 \* a);  return [x1, x2];  }  console.log(solveQuadratic(1, -3, 2)); // [2, 1] |

Round

|  |
| --- |
| console.log(Math.round(4.7)); // 5  console.log(Math.round(4.4)); // 4  console.log(Math.round(4.5)); // 5  console.log(Math.round(-4.5)); // -4  // مثال: محاسبه میانگین با گرد کردن  let scores = [85.6, 92.3, 78.9, 88.1];  let average = scores.reduce((sum, score) => sum + score, 0) / scores.length;  let roundedAverage = Math.round(average);  console.log("میانگین گرد شده:", roundedAverage);  // مثال: تبدیل دمای اعشاری به صحیح  let temperature = 23.7;  console.log("دمای گرد شده:", Math.round(temperature)); // 24 |

Floor گرد کردن به پایین

|  |
| --- |
| console.log(Math.floor(4.7)); // 4  console.log(Math.floor(4.4)); // 4  console.log(Math.floor(4.9)); // 4  console.log(Math.floor(-4.1)); // -5  // مثال: محاسبه سن  let birthYear = 1990;  let currentYear = 2024;  let age = Math.floor((new Date() - new Date(birthYear, 0, 1)) / (365.25 \* 24 \* 60 \* 60 \* 1000));  console.log("سن:", age);  // مثال: تقسیم صفحات در صفحه‌بندی  let totalItems = 47;  let itemsPerPage = 10;  let totalPages = Math.floor(totalItems / itemsPerPage);  console.log("تعداد صفحات:", totalPages); // 4 |

Ceil گرد کردن به بالا

|  |
| --- |
| console.log(Math.ceil(4.1)); // 5  console.log(Math.ceil(4.7)); // 5  console.log(Math.ceil(4.0)); // 4  console.log(Math.ceil(-4.1)); // -4  // مثال: محاسبه تعداد خودروهای مورد نیاز  let passengers = 23;  let carCapacity = 5;  let carsNeeded = Math.ceil(passengers / carCapacity);  console.log("تعداد خودروهای مورد نیاز:", carsNeeded); // 5  // مثال: محاسبه هزینه با احتساب مالیات  let price = 123.45;  let taxRate = 0.09;  let total = Math.ceil(price \* (1 + taxRate));  console.log("هزینه نهایی:", total); // 135 |

Trunc حذف بخش اعشاری

|  |
| --- |
| console.log(Math.trunc(4.7)); // 4  console.log(Math.trunc(4.1)); // 4  console.log(Math.trunc(-4.1)); // -4  console.log(Math.trunc(4.9)); // 4  // مثال: استخراج قسمت صحیح عدد  let decimalNumber = 15.789;  let integerPart = Math.trunc(decimalNumber);  console.log("قسمت صحیح:", integerPart); // 15  // مثال: تبدیل ساعت اعشاری به ساعت و دقیقه  let decimalHours = 8.75;  let hours = Math.trunc(decimalHours);  let minutes = Math.round((decimalHours - hours) \* 60);  console.log(`${hours} ساعت و ${minutes} دقیقه`); // 8 ساعت و 45 دقیقه |

Random تولید عدد تصادفی

|  |
| --- |
| // تولید عدد تصادفی بین 0 و 1  console.log(Math.random()); // مثلاً: 0.548213456789  // تولید عدد تصادفی بین 0 و 10  let random1 = Math.random() \* 10;  console.log(random1); // مثلاً: 7.892345  // تولید عدد صحیح تصادفی بین 1 و 100  let randomInt = Math.floor(Math.random() \* 100) + 1;  console.log(randomInt); // مثلاً: 42  // مثال: تولید رنگ HEX تصادفی  function getRandomColor() {  let color = '#' + Math.floor(Math.random() \* 16777215).toString(16);  return color;  }  console.log("رنگ تصادفی:", getRandomColor());  // مثال: انتخاب تصادفی از آرایه  let fruits = ["سیب", "موز", "پرتقال", "انگور"];  let randomFruit = fruits[Math.floor(Math.random() \* fruits.length)];  console.log("میوه تصادفی:", randomFruit);  // مثال: تولید رمز عبور تصادفی  function generatePassword(length) {  let chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  let password = "";  for (let i = 0; i < length; i++) {  password += chars[Math.floor(Math.random() \* chars.length)];  }  return password;  }  console.log("رمز عبور:", generatePassword(8)); // مثلاً: "aB3x9pL2" |

مثال جامع: بازی حدس عدد

|  |
| --- |
| // بازی حدس عدد تصادفی  function numberGuessingGame() {  const min = 1;  const max = 100;  const secretNumber = Math.floor(Math.random() \* (max - min + 1)) + min;  let attempts = 0;    console.log(`من عددی بین ${min} و ${max} در نظر گرفته‌ام. حدس بزن!`);    // این قسمت در محیط واقعی با input کاربر کار می‌کند  // برای مثال چند حدس تصادفی می‌زنیم  let guesses = [50, 25, 75, 37, 42];    guesses.forEach(guess => {  attempts++;  if (guess === secretNumber) {  console.log(`🎉 آفرین! در ${attempts} مرحله حدس زدی.`);  } else if (guess < secretNumber) {  console.log(`📈 عدد بزرگ‌تر از ${guess} است`);  } else {  console.log(`📉 عدد کوچک‌تر از ${guess} است`);  }  });    console.log(`عدد مورد نظر ${secretNumber} بود`);  }  numberGuessingGame(); |

جدول مقایسه متدهای گرد کردن

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| عدد | `round()` | `floor()` | `ceil()` | `trunc()` |
| 4.3 | 4 | 4 | 5 | 4 |
| 4.7 | 5 | 4 | 5 | 4 |
| 4.5 | 5 | 4 | 5 | 4 |
| -4.3 | -4 | -5 | -4 | -4 |
| -4.7 | -5 | -5 | -4 | -4 |

## **ساخت captcha**

توضیح:

CAPTCHA یک تست برای تشخیص انسان از ربات است. در این مثال، یک CAPTCHA ساده تولید می‌کنیم که کاربر باید عدد نمایش داده شده را وارد کند.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>CAPTCHA ساده</title>  </head>  <body>  <div id="captcha"></div>  <input type="text" id="userInput">  <button onclick="checkCaptcha()">بررسی</button>  <script>  // تولید عدد تصادفی بین ۱۰۰۰ تا ۹۹۹۹  let captchaValue = Math.floor(1000 + Math.random() \* 9000);  document.getElementById("captcha").innerText = captchaValue;  function checkCaptcha() {  let userInput = document.getElementById("userInput").value;  if (userInput == captchaValue) {  alert("کپچا صحیح است!");  } else {  alert("کپچا نادرست است!");  }  }  </script>  </body>  </html> |

## **معرفی arrow function ها**

توضیح:

Arrow Function ها روشی ساده تر برای نوشتن توابع در جاوااسکریپت هستند. این توابع syntax کوتاه تری دارند و this در آن ها به صورت lexical binding کار می کند.

مثال:

|  |
| --- |
| // تابع معمولی  function add(a, b) {  return a + b;  }  // Arrow Function  const addArrow = (a, b) => a + b;  // استفاده از هر دو  console.log(add(2, 3)); // خروجی: ۵  console.log(addArrow(2, 3)); // خروجی: ۵  // مثال دیگر با یک پارامتر  const square = x => x \* x;  console.log(square(5)); // خروجی: ۲۵ |

## **معرفی dom و bom**

DOM (Document Object Model)

توضیح کامل:

DOM یک رابط برنامه نویسی برای اسناد HTML و XML است. وقتی مرورگر یک صفحه HTML را بارگذاری می کند، یک مدل شیء گرایی از صفحه می سازد که به آن DOM می گویند. DOM صفحه را به صورت یک ساختار درختی از اشیا نمایش می دهد که می توان با جاوااسکریپت آن را تغییر داد.

اجزای اصلی DOM:

Document: نماینده کل سند HTML

Element: نماینده تگ های HTML

Attribute: نماینده ویژگی های تگ ها

Text: نماینده محتوای متنی

مثال DOM:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h1 id="title">سلام دنیا</h1>  <p id="demo">این یک متن است.</p>  <button onclick="changeContent()">تغییر محتوا</button>  <script>  function changeContent() {  // دسترسی به DOM و تغییر متن  document.getElementById("title").innerText = "عنوان تغییر کرد!";  document.getElementById("demo").innerHTML = "<strong>متن جدید با استایل</strong>";    // ایجاد عنصر جدید  let newElement = document.createElement("p");  newElement.textContent = "این یک پاراگراف جدید است";  document.body.appendChild(newElement);  }  </script>  </body>  </html> |

BOM (Browser Object Model)

توضیح کامل:

BOM به توسعه دهندگان اجازه می دهد با مرورگر تعامل داشته باشند. بر خلاف DOM که استاندارد است، BOM استاندارد رسمی ندارد اما همه مرورگرها از آن پشتیبانی می کنند.

اجزای اصلی BOM:

window: بالاترین شیء در BOM

navigator: اطلاعات درباره مرورگر

screen: اطلاعات درباره صفحه نمایش

history: تاریخچه مرورگر

location: اطلاعات درباره URL فعلی

مثال BOM:

|  |
| --- |
| // کار با window  window.alert("این یک پیام است");  // باز کردن پنجره جدید  function openNewWindow() {  window.open("https://example.com", "\_blank", "width=600,height=400");  }  // اطلاعات مرورگر  console.log("نام مرورگر: " + navigator.userAgent);  console.log("زبان مرورگر: " + navigator.language);  // اطلاعات صفحه نمایش  console.log("عرض صفحه: " + screen.width);  console.log("ارتفاع صفحه: " + screen.height);  // کار با location  console.log("آدرس فعلی: " + location.href);  // location.reload(); // بارگذاری مجدد صفحه  // location.href = "https://google.com"; // تغییر آدرس  // کار با history  // history.back(); // بازگشت به صفحه قبل  // history.forward(); // رفتن به صفحه بعد |

تفاوت DOM و BOM:

DOM مربوط به محتوای صفحه است

BOM مربوط به مرورگر و محیط اجرا است

DOM یک استاندارد رسمی دارد

BOM استاندارد رسمی ندارد اما همه مرورگرها از آن پشتیبانی می کنند

## **دسترسی به dom با استفاده از id و classname و tagname**

دسترسی با استفاده از ID

توضیح:

برای دسترسی به عناصری که دارای id هستند از getElementById استفاده می‌کنیم.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="header">عنوان سایت</div>  <p id="description">این یک پاراگراف است</p>  <script>  // دسترسی با ID  let header = document.getElementById("header");  let description = document.getElementById("description");    // تغییر محتوا  header.innerText = "عنوان تغییر کرد";  description.style.color = "red";  </script>  </body>  </html> |

دسترسی با استفاده از ClassName

توضیح:

برای دسترسی به عناصری که دارای class هستند از getElementsByClassName استفاده می‌کنیم.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div class="box">جعبه ۱</div>  <div class="box">جعبه ۲</div>  <div class="box">جعبه ۳</div>  <script>  // دسترسی با ClassName  let boxes = document.getElementsByClassName("box");    // تغییر همه عناصر با کلاس box  for (let i = 0; i < boxes.length; i++) {  boxes[i].style.backgroundColor = "lightblue";  boxes[i].style.padding = "10px";  boxes[i].style.margin = "5px";  }  </script>  </body>  </html> |

دسترسی با استفاده از TagName

توضیح:

برای دسترسی به عناصر بر اساس نام تگ از getElementsByTagName استفاده می‌کنیم.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>پاراگراف اول</p>  <p>پاراگراف دوم</p>  <div>این یک div است</div>  <p>پاراگراف سوم</p>  <script>  // دسترسی با TagName  let paragraphs = document.getElementsByTagName("p");    // تغییر همه پاراگراف‌ها  for (let i = 0; i < paragraphs.length; i++) {  paragraphs[i].style.fontSize = "18px";  paragraphs[i].style.border = "1px solid black";  }    // تغییر div  let divs = document.getElementsByTagName("div");  divs[0].style.backgroundColor = "yellow";  </script>  </body>  </html> |

## **استفاده از queryselector**

دسترسی با استفاده از QuerySelector

توضیح:

querySelector و querySelectorAll روش‌های مدرن و قدرتمند برای انتخاب عناصر هستند که از syntax شبیه CSS استفاده می‌کنند.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="container">  <p class="text">متن اول</p>  <p class="text special">متن ویژه</p>  <div class="item">آیتم ۱</div>  <div class="item">آیتم ۲</div>  </div>  <script>  // انتخاب اولین عنصر با کلاس text  let firstText = document.querySelector(".text");  firstText.style.color = "blue";    // انتخاب عنصر با کلاس special  let specialText = document.querySelector(".special");  specialText.style.fontWeight = "bold";    // انتخاب تمام عناصر با کلاس item  let allItems = document.querySelectorAll(".item");  allItems.forEach(item => {  item.style.backgroundColor = "lightgreen";  });    // انتخاب تمام پاراگراف‌های داخل container  let containerParagraphs = document.querySelectorAll("#container p");  containerParagraphs.forEach(p => {  p.style.padding = "5px";  });    // انتخاب با ترکیب مختلف  let firstItem = document.querySelector("div.item");  firstItem.innerHTML = "اولین آیتم تغییر کرد";  </script>  </body>  </html> |

مثال جامع

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>مثال جامع DOM</title>  <style>  .active { background-color: yellow; }  .hidden { display: none; }  </style>  </head>  <body>  <header id="main-header">  <h1>وبسایت من</h1>  </header>    <nav class="navigation">  <ul>  <li class="nav-item">خانه</li>  <li class="nav-item active">درباره ما</li>  <li class="nav-item">تماس با ما</li>  </ul>  </nav>    <main id="content">  <article class="post">  <h2>مقاله اول</h2>  <p>این محتوای مقاله اول است.</p>  </article>  <article class="post">  <h2>مقاله دوم</h2>  <p>این محتوای مقاله دوم است.</p>  </article>  </main>    <footer id="main-footer">  <p>تمامی حقوق محفوظ است</p>  </footer>  <script>  // دسترسی با ID  let header = document.getElementById("main-header");  header.style.backgroundColor = "#f0f0f0";    // دسترسی با ClassName  let navItems = document.getElementsByClassName("nav-item");  for (let item of navItems) {  item.style.cursor = "pointer";  item.style.padding = "10px";  }    // دسترسی با TagName  let articles = document.getElementsByTagName("article");  for (let article of articles) {  article.style.border = "1px solid #ccc";  article.style.margin = "10px";  article.style.padding = "15px";  }    // دسترسی با QuerySelector  let activeItem = document.querySelector(".nav-item.active");  activeItem.style.color = "red";    let allPosts = document.querySelectorAll(".post");  allPosts.forEach((post, index) => {  post.querySelector("h2").innerText = `مقاله ${index + 1} - ویرایش شده`;  });    // انتخاب footer با روش‌های مختلف  let footer1 = document.getElementById("main-footer");  let footer2 = document.querySelector("#main-footer");  let footer3 = document.querySelector("footer");    console.log("همه روش‌ها یک عنصر را انتخاب کردند:", footer1 === footer2 && footer2 === footer3);  </script>  </body>  </html> |

نکات مهم

getElementById فقط یک عنصر برمی‌گرداند

getElementsByClassName و getElementsByTagName یک collection برمی‌گردانند

querySelector اولین عنصر منطبق را برمی‌گرداند

querySelectorAll یک NodeList از تمام عناصر منطبق برمی‌گرداند

برای کار با collectionها باید از حلقه استفاده کنید

querySelector و querySelectorAll از تمام selectorهای CSS پشتیبانی می‌کنند

## **معرفی innerhtml و innertext**

innerHTML

توضیح:

innerHTML محتوای HTML داخل یک عنصر را برمی‌گرداند یا تنظیم می‌کند. این ویژگی امکان اضافه کردن تگ‌های HTML جدید را فراهم می‌کند.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="container">  <p>این یک <strong>متن</strong> تست است</p>  </div>  <script>  let container = document.getElementById("container");    // خواندن محتوای HTML  console.log(container.innerHTML);  // خروجی: <p>این یک <strong>متن</strong> تست است</p>    // تغییر محتوای HTML  container.innerHTML = `  <h2>عنوان جدید</h2>  <p>این یک پاراگراف <span style="color: red;">جدید</span> است</p>  <button>کلیک کنید</button>  `;  </script>  </body>  </html> |

innerText

توضیح:

innerText فقط متن قابل مشاهده یک عنصر را برمی‌گرداند یا تنظیم می‌کند. تگ‌های HTML در نظر گرفته نمی‌شوند.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="content">  <p>این یک <strong>متن</strong> تست است</p>  <div style="display: none;">این متن مخفی است</div>  </div>  <script>  let content = document.getElementById("content");    // خواندن متن قابل مشاهده  console.log(content.innerText);  // خروجی: این یک متن تست است    // تغییر متن  content.innerText = "این یک متن جدید است <strong>تگ HTML کار نمی‌کند</strong>";  </script>  </body>  </html> |

مقایسه innerHTML و innerText\*\*

| ویژگی | HTML را تفسیر می‌کند | متن مخفی را نشان می‌دهد | عملکرد |
| --- | --- | --- | --- |
| innerHTML | بله | خیر | سریع‌تر |
| innerText | خیر | بله | کندتر |

## **معرفی .attribute**

توضیح:

می‌توانیم مستقیماً به ویژگی‌های عناصر دسترسی داشته باشیم و آن‌ها را تغییر دهیم.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <img id="myImage" src="default.jpg" alt="تصویر پیش‌فرض" width="200">  <a id="myLink" href="https://google.com">برو به گوگل</a>  <input id="myInput" type="text" value="مقدار اولیه">  <script>  let image = document.getElementById("myImage");  let link = document.getElementById("myLink");  let input = document.getElementById("myInput");    // خواندن ویژگی‌ها  console.log(image.src);  console.log(link.href);  console.log(input.value);    // تغییر ویژگی‌ها  image.src = "new-image.jpg";  image.alt = "تصویر جدید";  image.width = 300;    link.href = "https://github.com";  link.innerText = "برو به گیت‌هاب";    input.value = "مقدار جدید";  input.type = "password";  </script>  </body>  </html> |

## **معرفی setattribute**

توضیح:

setAttribute برای تنظیم یا تغییر ویژگی‌های عناصر استفاده می‌شود. این روش زمانی مفید است که بخواهیم ویژگی‌های سفارشی یا ویژگی‌هایی که مستقیماً در object وجود ندارند را تغییر دهیم.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="myDiv" data-info="اطلاعات اولیه">یک div معمولی</div>  <button id="myButton">یک دکمه</button>  <input id="myInput">  <script>  let div = document.getElementById("myDiv");  let button = document.getElementById("myButton");  let input = document.getElementById("myInput");    // استفاده از setAttribute  div.setAttribute("data-info", "اطلاعات جدید");  div.setAttribute("class", "container active");  div.setAttribute("title", "این یک tooltip است");    button.setAttribute("disabled", "true");  button.setAttribute("data-action", "submit");    input.setAttribute("placeholder", "نام خود را وارد کنید");  input.setAttribute("required", "true");  input.setAttribute("data-validation", "email");    // خواندن ویژگی با getAttribute  console.log(div.getAttribute("data-info")); // اطلاعات جدید  console.log(button.getAttribute("disabled")); // true  console.log(input.getAttribute("placeholder")); // نام خود را وارد کنید  </script>  </body>  </html> |

مثال جامع

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>مثال جامع</title>  <style>  .card {  border: 1px solid #ccc;  padding: 20px;  margin: 10px;  }  .active {  background-color: yellow;  }  .hidden {  display: none;  }  </style>  </head>  <body>  <div id="app">  <div id="user-card" class="card" data-user-id="123">  <h2 id="user-name">نام کاربر</h2>  <p id="user-email">email@example.com</p>  <button id="edit-btn">ویرایش</button>  </div>    <div id="product-list">  <div class="product" data-price="100000">محصول ۱</div>  <div class="product" data-price="200000">محصول ۲</div>  </div>  </div>  <script>  // کار با innerHTML و innerText  let userName = document.getElementById("user-name");  console.log("innerText:", userName.innerText);  console.log("innerHTML:", userName.innerHTML);    // تغییر با innerHTML (ایجاد تگ جدید)  userName.innerHTML = "علیرضا <span style='color: green;'>(آنلاین)</span>";    // کار با attribute مستقیم  let userCard = document.getElementById("user-card");  console.log("کلاس فعلی:", userCard.className);  userCard.className = "card active";    // کار با setAttribute  userCard.setAttribute("data-user-role", "admin");  userCard.setAttribute("title", "کاربر مدیر سیستم");    // تغییر ویژگی‌های دکمه  let editBtn = document.getElementById("edit-btn");  editBtn.setAttribute("onclick", "editUser()");  editBtn.setAttribute("data-action", "edit");    // کار با محصولات  let products = document.querySelectorAll(".product");  products.forEach((product, index) => {  // خواندن data attribute  let price = product.getAttribute("data-price");  console.log(`قیمت محصول ${index + 1}: ${price}`);    // تغییر با innerHTML  product.innerHTML = `  محصول ${index + 1}  <br>  <small style="color: green;">قیمت: ${price} تومان</small>  `;    // اضافه کردن ویژگی جدید  product.setAttribute("data-index", index);  });    // تابع ویرایش  function editUser() {  let userCard = document.getElementById("user-card");  let currentRole = userCard.getAttribute("data-user-role");    if (currentRole === "admin") {  userCard.setAttribute("data-user-role", "user");  userCard.style.border = "1px solid blue";  } else {  userCard.setAttribute("data-user-role", "admin");  userCard.style.border = "1px solid red";  }    alert("نقش کاربر تغییر کرد");  }  </script>  </body>  </html> |

خلاصه و نکات مهم

تفاوت‌های کلیدی:

innerHTML vs innerText:

innerHTML تگ‌های HTML را تفسیر می‌کند

innerText فقط متن خالص را نمایش می‌دهد

attribute مستقیم vs setAttribute:

attribute مستقیم: element.id = "newId"

setAttribute: element.setAttribute("id", "newId")

موارد استفاده:

برای تغییر محتوای HTML از innerHTML

برای تغییر متن ساده از innerText

برای ویژگی‌های استاندارد از attribute مستقیم

برای ویژگی‌های سفارشی از setAttribute

امنیت:

هنگام استفاده از innerHTML مراقب حملات XSS باشید

برای داده‌های کاربر از textContent یا innerText استفاده کنید

## **معرفی .clasename**

توضیح:

ویژگی className برای خواندن یا تغییر کلاس‌های CSS یک عنصر استفاده می‌شود. این ویژگی یک رشته را برمی‌گرداند که شامل تمام کلاس‌های عنصر است.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .red { color: red; }  .bold { font-weight: bold; }  .large { font-size: 24px; }  .border { border: 2px solid black; }  </style>  </head>  <body>  <div id="myDiv" class="red bold">این یک div است</div>  <button onclick="changeClass()">تغییر کلاس</button>  <script>  let myDiv = document.getElementById("myDiv");    // خواندن کلاس‌ها  console.log("کلاس‌های فعلی:", myDiv.className); // خروجی: "red bold"    function changeClass() {  // تغییر کلاس‌ها  myDiv.className = "large border";    // اضافه کردن کلاس جدید  // myDiv.className += " blue"; // روش ساده اما مشکل‌دار    // روش بهتر برای اضافه کردن کلاس  myDiv.className = myDiv.className + " blue";    console.log("کلاس‌های جدید:", myDiv.className);  }  </script>  </body>  </html> |

مثال کاربردی:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .active { background-color: yellow; }  .hidden { display: none; }  .error { color: red; border: 1px solid red; }  .success { color: green; border: 1px solid green; }  </style>  </head>  <body>  <div id="message" class="success">عملیات با موفقیت انجام شد</div>  <button id="toggleBtn">تغییر وضعیت</button>  <script>  let message = document.getElementById("message");  let toggleBtn = document.getElementById("toggleBtn");    toggleBtn.addEventListener("click", function() {  if (message.className === "success") {  message.className = "error";  message.innerText = "خطا در انجام عملیات!";  } else {  message.className = "success";  message.innerText = "عملیات با موفقیت انجام شد";  }  });  </script>  </body>  </html> |

## **معرفی .href**

توضیح:

ویژگی href برای خواندن یا تغییر آدرس URL لینک‌ها استفاده می‌شود. این ویژگی می‌تواند آدرس کامل (absolute URL) یا آدرس نسبی (relative URL) باشد.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <a id="myLink" href="https://google.com">برو به گوگل</a>  <br>  <button onclick="changeLink()">تغییر لینک</button>  <button onclick="showLink()">نمایش لینک</button>  <script>  let myLink = document.getElementById("myLink");    function showLink() {  // خواندن آدرس لینک  alert("آدرس لینک: " + myLink.href);  }    function changeLink() {  // تغییر آدرس لینک  myLink.href = "https://github.com";  myLink.innerText = "برو به گیت‌هاب";  myLink.target = "\_blank"; // باز کردن در تب جدید  }  </script>  </body>  </html> |

مثال کاربردی:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h3>منوی ناوبری</h3>  <nav>  <a id="homeLink" href="#home">خانه</a> |  <a id="aboutLink" href="#about">درباره ما</a> |  <a id="contactLink" href="#contact">تماس با ما</a>  </nav>    <div style="margin-top: 20px;">  <button onclick="makeExternal()">تبدیل به لینک خارجی</button>  <button onclick="addTracking()">اضافه کردن tracking</button>  </div>  <script>  function makeExternal() {  let homeLink = document.getElementById("homeLink");  let aboutLink = document.getElementById("aboutLink");  let contactLink = document.getElementById("contactLink");    homeLink.href = "https://example.com/home";  aboutLink.href = "https://example.com/about";  contactLink.href = "https://example.com/contact";    // تنظیم برای باز شدن در تب جدید  homeLink.target = "\_blank";  aboutLink.target = "\_blank";  contactLink.target = "\_blank";  }    function addTracking() {  let links = document.querySelectorAll("a");  links.forEach(link => {  // اضافه کردن پارامتر tracking به URL  if (link.href.includes("example.com")) {  link.href += "?source=website&medium=button";  }  });  }  </script>  </body>  </html> |

## **معرفی .src**

توضیح:

ویژگی src برای خواندن یا تغییر منبع (source) عناصری مانند تصاویر، iframeها، اسکریپت‌ها و ویدیوها استفاده می‌شود.

مثال:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <img id="myImage" src="default.jpg" alt="تصویر پیش‌فرض" width="300">  <br>  <button onclick="changeImage()">تغییر تصویر</button>  <button onclick="showSource()">نمایش منبع</button>  <script>  let myImage = document.getElementById("myImage");    function showSource() {  // خواندن آدرس تصویر  alert("آدرس تصویر: " + myImage.src);  }    function changeImage() {  // تغییر آدرس تصویر  myImage.src = "new-image.jpg";  myImage.alt = "تصویر جدید";  myImage.title = "این یک تصویر جدید است";  }  </script>  </body>  </html> |

**مثال کاربردی:**

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <h3>گالری تصاویر</h3>  <img id="galleryImage" src="image1.jpg" width="400" alt="تصویر گالری">  <br>  <div>  <button onclick="prevImage()">تصویر قبلی</button>  <button onclick="nextImage()">تصویر بعدی</button>  <button onclick="toggleVideo()">تبدیل به ویدیو</button>  </div>  <script>  let images = [  "image1.jpg",  "image2.jpg",  "image3.jpg",  "image4.jpg"  ];    let currentIndex = 0;  let galleryImage = document.getElementById("galleryImage");  let isVideo = false;    function showCurrentImage() {  galleryImage.src = images[currentIndex];  galleryImage.alt = `تصویر ${currentIndex + 1}`;  }    function nextImage() {  if (isVideo) return;    currentIndex = (currentIndex + 1) % images.length;  showCurrentImage();  }    function prevImage() {  if (isVideo) return;    currentIndex = (currentIndex - 1 + images.length) % images.length;  showCurrentImage();  }    function toggleVideo() {  if (!isVideo) {  // تغییر به ویدیو  galleryImage.src = "video-poster.jpg";  galleryImage.alt = "پوستر ویدیو";  isVideo = true;  } else {  // بازگشت به تصویر  showCurrentImage();  isVideo = false;  }  }  </script>  </body>  </html> |

مثال جامع

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .profile-card {  border: 2px solid #333;  padding: 20px;  margin: 10px;  text-align: center;  border-radius: 10px;  }  .active {  background-color: #e3f2fd;  border-color: #2196f3;  }  .inactive {  background-color: #f5f5f5;  border-color: #9e9e9e;  }  </style>  </head>  <body>  <div id="profile" class="profile-card active">  <img id="avatar" src="default-avatar.jpg" width="100" alt="آواتار">  <h2 id="username">کاربر مهمان</h2>  <a id="profileLink" href="/profile">مشاهده پروفایل</a>  <br><br>  <button onclick="updateProfile()">بروزرسانی پروفایل</button>  </div>  <script>  function updateProfile() {  // دسترسی به عناصر  let profile = document.getElementById("profile");  let avatar = document.getElementById("avatar");  let username = document.getElementById("username");  let profileLink = document.getElementById("profileLink");    // تغییر className  if (profile.className.includes("active")) {  profile.className = "profile-card inactive";  } else {  profile.className = "profile-card active";  }    // تغییر src تصویر  let newAvatar = prompt("آدرس آواتار جدید را وارد کنید:", "new-avatar.jpg");  if (newAvatar) {  avatar.src = newAvatar;  avatar.alt = "آواتار کاربر";  }    // تغییر متن و href لینک  let newUsername = prompt("نام کاربری جدید را وارد کنید:", "علیرضا");  if (newUsername) {  username.innerText = newUsername;  profileLink.href = `/user/${newUsername}`;  profileLink.innerText = `پروفایل ${newUsername}`;  profileLink.target = "\_blank";  }    // نمایش اطلاعات  console.log("کلاس‌های پروفایل:", profile.className);  console.log("آدرس تصویر:", avatar.src);  console.log("آدرس لینک:", profileLink.href);  }  </script>  </body>  </html> |

نکات مهم

className:

برای اضافه کردن کلاس جدید از classList.add() استفاده کنید

className تمام کلاس‌های قبلی را پاک می‌کند

برای نگهداری کلاس‌های قبلی: element.className += " new-class"

href:

href همیشه آدرس کامل را برمی‌گرداند

برای آدرس نسبی از getAttribute("href") استفاده کنید

src:

src همیشه آدرس کامل را برمی‌گرداند

تغییر src بلافاصله منبع جدید را بارگذاری می‌کند

## **معرفی regular expression**

توضیح:

Regular Expression (Regex) الگوهایی برای جستجو و مدیریت متن هستند. از Regex برای اعتبارسنجی، جستجو، جایگزینی و استخراج متن استفاده می‌شود.

|  |
| --- |
| // روش ۱: استفاده از literal (توصیه شده)  let regex1 = /pattern/;  // روش ۲: استفاده از constructor  let regex2 = new RegExp("pattern"); |

فلگ‌های مهم

|  |
| --- |
| // g - جستجوی سراسری  let text = "hello hello hello";  console.log(text.match(/hello/g)); // ["hello", "hello", "hello"]  // i - حساس نبودن به بزرگی و کوچکی  console.log("Hello".match(/hello/i)); // ["Hello"]  // m - جستجوی چند خطی  let multiLineText = "first line\nsecond line\nthird line";  console.log(multiLineText.match(/^second/m)); // ["second"] |

مثال‌های ساده

۱. جستجوی ساده

|  |
| --- |
| let text = "Hello World! Hello JavaScript!";  let regex = /Hello/;  // تست وجود الگو  console.log(regex.test(text)); // true  // پیدا کردن اولین مورد  console.log(text.match(regex)); // ["Hello"]  // پیدا کردن همه موارد  console.log(text.match(/Hello/g)); // ["Hello", "Hello"] |

اعتبارسنجی ایمیل

|  |
| --- |
| <input type="text" id="email" placeholder="ایمیل خود را وارد کنید">  <button onclick="validateEmail()">اعتبارسنجی</button>  <p id="result"></p>  <script>  function validateEmail() {  let email = document.getElementById("email").value;  let regex = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;    if (regex.test(email)) {  document.getElementById("result").innerText = "ایمیل معتبر است";  document.getElementById("result").style.color = "green";  } else {  document.getElementById("result").innerText = "ایمیل نامعتبر است";  document.getElementById("result").style.color = "red";  }  }  </script> |

اعتبارسنجی شماره موبایل ایرانی

|  |
| --- |
| <input type="text" id="mobile" placeholder="شماره موبایل را وارد کنید">  <button onclick="validateMobile()">اعتبارسنجی</button>  <p id="mobileResult"></p>  <script>  function validateMobile() {  let mobile = document.getElementById("mobile").value;  let regex = /^09[0-9]{9}$/;    if (regex.test(mobile)) {  document.getElementById("mobileResult").innerText = "شماره موبایل معتبر است";  document.getElementById("mobileResult").style.color = "green";  } else {  document.getElementById("mobileResult").innerText = "شماره موبایل نامعتبر است";  document.getElementById("mobileResult").style.color = "red";  }  }  </script> |

کاراکترهای خاص در Regex

کلاس‌های کاراکتر

|  |
| --- |
| let text = "abc123 ABC!@#";  // اعداد  console.log(text.match(/[0-9]/g)); // ["1", "2", "3"]  // حروف کوچک  console.log(text.match(/[a-z]/g)); // ["a", "b", "c"]  // حروف بزرگ  console.log(text.match(/[A-Z]/g)); // ["A", "B", "C"]  // کاراکترهای خاص  console.log(text.match(/[!@#]/g)); // ["!", "@", "#"] |

متا کاراکترها

|  |
| --- |
| let text = "cat bat rat mat";  // شروع با c  console.log(text.match(/^c/g)); // ["c"]  // پایان با t  console.log(text.match(/t$/g)); // null (چون متن با t پایان نمی‌یابد)  // هر کاراکتری  console.log(text.match(/.at/g)); // ["cat", "bat", "rat", "mat"] |

متدهای مهم Regex

۱متد test()

|  |
| --- |
| let regex = /hello/i; // i برای حساس نبودن به بزرگی و کوچکی  console.log(regex.test("Hello World")); // true  console.log(regex.test("Hi World")); // false |

متد exec()

|  |
| --- |
| let regex = /hello/g;  let text = "hello world hello javascript";  let result;  while ((result = regex.exec(text)) !== null) {  console.log(`Found: ${result[0]} at position: ${result.index}`);  } |

متد replace()

|  |
| --- |
| let text = "Hello World! Hello JavaScript!";  // جایگزینی ساده  console.log(text.replace(/Hello/, "Hi")); // "Hi World! Hello JavaScript!"  // جایگزینی همه موارد  console.log(text.replace(/Hello/g, "Hi")); // "Hi World! Hi JavaScript!"  // جایگزینی با تابع  console.log(text.replace(/Hello/g, (match) => match.toUpperCase())); // "HELLO World! HELLO JavaScript!" |

مثال‌های کاربردی

۱. استخراج اعداد از متن

|  |
| --- |
| let text = "من 25 سال دارم و 70 کیلوگرم وزن دارم";  let numbers = text.match(/\d+/g);  console.log(numbers); // ["25", "70"] |

حذف فاصله‌های اضافی

|  |
| --- |
| let text = "این یک متن با فاصله‌های اضافی است";  let cleanText = text.replace(/\s+/g, " ");  console.log(cleanText); // "این یک متن با فاصله‌های اضافی است" |

اعتبارسنجی رمز عبور

|  |
| --- |
| function validatePassword(password) {  // حداقل ۸ کاراکتر، شامل حروف بزرگ، کوچک، عدد و کاراکتر خاص  let regex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/;  return regex.test(password);  }  console.log(validatePassword("StrongPass1!")); // true  console.log(validatePassword("weak")); // false |

نکات مهم

کاراکترهای فرار: برای جستجوی کاراکترهای خاص از \ استفاده کنید

|  |
| --- |
| console.log("1+1=2".match(/1\+1=2/)); // ["1+1=2"] |

کوانتیفایرها:

|  |
| --- |
| // \* - صفر یا بیشتر  // + - یک یا بیشتر  // ? - صفر یا یک  // {n} - دقیقا n بار  // {n,} - حداقل n بار  // {n,m} - بین n تا m بار |

|  |
| --- |
| // \* - صفر یا بیشتر  console.log("aabc".match(/a\*b/)); // ["aab"]  // + - یک یا بیشتر  console.log("aabc".match(/a+b/)); // ["aab"]  // ? - صفر یا یک  console.log("color".match(/colou?r/)); // ["color"]  // {n} - دقیقا n بار  console.log("aaa".match(/a{3}/)); // ["aaa"]  // {n,} - حداقل n بار  console.log("aaa".match(/a{2,}/)); // ["aaa"]  // {n,m} - بین n تا m بار  console.log("aaa".match(/a{2,3}/)); // ["aaa"] |

ستاره \* - صفر یا بیشتر

|  |
| --- |
| // الگو: a\* یعنی صفر یا چند حرف a  console.log("b".match(/a\*b/)); // ["b"] - صفر a و سپس b  console.log("ab".match(/a\*b/)); // ["ab"] - یک a و سپس b  console.log("aab".match(/a\*b/)); // ["aab"] - دو a و سپس b  console.log("aaab".match(/a\*b/)); // ["aaab"] - سه a و سپس b |

مثبت + - یک یا بیشتر

|  |
| --- |
| // الگو: a+ یعنی حداقل یک حرف a  console.log("b".match(/a+b/)); // null - هیچ a وجود ندارد  console.log("ab".match(/a+b/)); // ["ab"] - یک a و سپس b  console.log("aab".match(/a+b/)); // ["aab"] - دو a و سپس b  console.log("aaab".match(/a+b/)); // ["aaab"] - سه a و سپس b |

علامت سوال ? - صفر یا یک

|  |
| --- |
| // الگو: s? یعنی s می‌تواند باشد یا نباشد  console.log("cat".match(/cats?/)); // ["cat"] - بدون s  console.log("cats".match(/cats?/)); // ["cats"] - با s |

آکولاد {n} - دقیقاً n بار

|  |
| --- |
| // الگو: a{3} یعنی دقیقاً سه a  console.log("aa".match(/a{3}/)); // null - فقط دو a  console.log("aaa".match(/a{3}/)); // ["aaa"] - دقیقاً سه a  console.log("aaaa".match(/a{3}/)); // ["aaa"] - فقط سه a اول |

آکولاد {n,} - حداقل n بار

|  |
| --- |
| // الگو: a{2,} یعنی حداقل دو a  console.log("a".match(/a{2,}/)); // null - فقط یک a  console.log("aa".match(/a{2,}/)); // ["aa"] - دو a  console.log("aaa".match(/a{2,}/)); // ["aaa"] - سه a |

آکولاد {n,m} - بین n تا m بار

|  |
| --- |
| // الگو: a{2,4} یعنی بین ۲ تا ۴ a  console.log("a".match(/a{2,4}/)); // null - فقط یک a  console.log("aa".match(/a{2,4}/)); // ["aa"] - دو a  console.log("aaa".match(/a{2,4}/)); // ["aaa"] - سه a  console.log("aaaa".match(/a{2,4}/)); // ["aaaa"] - چهار a  console.log("aaaaa".match(/a{2,4}/)); // ["aaaa"] - فقط چهار a اول |

مثال‌های کاربردی

|  |
| --- |
| // کد ملی (دقیقاً ۱۰ رقم)  console.log("1234567890".match(/^\d{10}$/)); // ["1234567890"]  console.log("12345".match(/^\d{10}$/)); // null  // نام کاربری (۳ تا ۲۰ کاراکتر)  console.log("ali".match(/^[a-z]{3,20}$/)); // ["ali"]  console.log("al".match(/^[a-z]{3,20}$/)); // null  // رمز عبور (حداقل ۸ کاراکتر)  console.log("password".match(/^.{8,}$/)); // ["password"]  console.log("pass".match(/^.{8,}$/)); // null  // شماره تلفن (۱۰ یا ۱۱ رقم)  console.log("0912345678".match(/^\d{10,11}$/)); // ["0912345678"]  console.log("09123".match(/^\d{10,11}$/)); // null |

گروه‌بندی:

|  |
| --- |
| let text = "name: John, age: 30";  let match = text.match(/name: (\w+), age: (\d+)/);  console.log(match[1]); // "John"  console.log(match[2]); // "30" |

پرانتز معمولی () - گروه‌بندی

|  |
| --- |
| // مثال ۱: گروه‌بندی برای تکرار  const text1 = "abcabcabc";  console.log(text1.match(/(abc)+/)); // ["abcabcabc", "abc"]  // مثال ۲: گروه‌بندی برای انتخاب  const text2 = "cat and dog";  console.log(text2.match(/(cat|dog)/)); // ["cat", "cat"]  console.log(text2.match(/(cat|dog)/g)); // ["cat", "dog"] |

پرانتز با ?: - گروه بدون ذخیره‌سازی

|  |
| --- |
| // تفاوت بین () و (?:)  const text = "hello world";  // با () - گروه ذخیره می‌شود  const withGroup = /(hello) (world)/;  const match1 = text.match(withGroup);  console.log(match1); // ["hello world", "hello", "world"]  // با (?:) - گروه ذخیره نمی‌شود  const withoutGroup = /(?:hello) (world)/;  const match2 = text.match(withoutGroup);  console.log(match2); // ["hello world", "world"] |

Lookahead (?=) - شرط بعدی

|  |
| --- |
| // پیدا کردن کلماتی که بعدشان "world" آمده  const text3 = "hello world, test world, example";  console.log(text3.match(/\w+(?= world)/g)); // ["hello", "test"]  // بررسی پسورد که عدد دارد (بدون مصرف کردن کاراکترها)  const password = "Password123";  console.log(/(?=.\*\d)/.test(password)); // true - شامل عدد است |

Negative Lookahead (?!) - شرط منفی بعدی

|  |
| --- |
| // پیدا کردن کلماتی که بعدشان "world" نیامده  const text4 = "hello test, hello world, hello example";  console.log(text4.match(/hello(?! world)/g)); // ["hello", "hello"]  // بررسی که ایمیل با @example.com تمام نشود  const emails = ["user@gmail.com", "admin@example.com", "test@yahoo.com"];  emails.forEach(email => {  if (/^(?!.\*@example\.com$).\*$/.test(email)) {  console.log(email); // user@gmail.com, test@yahoo.com  }  }); |

Lookbehind (?<=) - شرط قبلی

|  |
| --- |
| // پیدا کردن اعدادی که بعد از "price:" آمده‌اند  const text5 = "price:100, cost:200, value:300";  console.log(text5.match(/(?<=price:)\d+/g)); // ["100"]  // پیدا کردن اعدادی که بعد از "$" آمده‌اند  const prices = "The price is $50 and $100";  console.log(prices.match(/(?<=\$)\d+/g)); // ["50", "100"] |

Negative Lookbehind (?<!) - شرط منفی قبلی

|  |
| --- |
| // پیدا کردن اعدادی که قبلشان "$" نیامده  const text6 = "price 50, $100, value 200";  console.log(text6.match(/(?<!\$)\b\d+\b/g)); // ["50", "200"]  // پیدا کردن کلمات که قبلشان "not" نیامده  const sentence = "I am happy, I am not sad, I am excited";  console.log(sentence.match(/(?<!\bnot\s)\bam\b/g)); // ["am", "am"] |

مثال‌های کاربردی

|  |
| --- |
| // مثال ۱: استخراج نام و نام خانوادگی  const fullName = "علی محمدی";  const nameRegex = /(\S+)\s+(\S+)/;  const nameMatch = fullName.match(nameRegex);  console.log(nameMatch[1]); // "علی"  console.log(nameMatch[2]); // "محمدی"  // مثال ۲: استخراج تاریخ  const date = "تاریخ: 1402/10/15";  const dateRegex = /(\d{4})\/(\d{2})\/(\d{2})/;  const dateMatch = date.match(dateRegex);  console.log(dateMatch[1]); // "1402" - سال  console.log(dateMatch[2]); // "10" - ماه  console.log(dateMatch[3]); // "15" - روز  // مثال ۳: اعتبارسنجی ایمیل با گروه‌بندی  function parseEmail(email) {  const emailRegex = /(\w+)@(\w+)\.(\w+)/;  const match = email.match(emailRegex);  if (match) {  return {  username: match[1],  domain: match[2],  tld: match[3]  };  }  return null;  }  console.log(parseEmail("test@example.com"));  // { username: "test", domain: "example", tld: "com" } |

فرم ورود با اعتبارسنجی Real-time

در این مثال یک فرم ورود ایجاد می‌کنیم که در حین تایپ کاربر و هنگام خروج از هر فیلد، اعتبارسنجی انجام می‌دهد.

|  |
| --- |
| <style>  \* {  box-sizing: border-box;  margin: 0;  padding: 0;  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  }  body {  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  min-height: 100vh;  display: flex;  justify-content: center;  align-items: center;  padding: 20px;  }  .login-container {  background: white;  padding: 40px;  border-radius: 15px;  box-shadow: 0 15px 35px rgba(0, 0, 0, 0.1);  width: 100%;  max-width: 450px;  }  h2 {  text-align: center;  margin-bottom: 30px;  color: #333;  font-size: 28px;  }  .form-group {  margin-bottom: 25px;  position: relative;  }  label {  display: block;  margin-bottom: 8px;  color: #555;  font-weight: 600;  }  input {  width: 100%;  padding: 12px 15px;  border: 2px solid #ddd;  border-radius: 8px;  font-size: 16px;  transition: all 0.3s ease;  outline: none;  }  input:focus {  border-color: #667eea;  box-shadow: 0 0 0 3px rgba(102, 126, 234, 0.1);  }  input.valid {  border-color: #28a745;  }  input.invalid {  border-color: #dc3545;  }  .error-message {  color: #dc3545;  font-size: 14px;  margin-top: 5px;  height: 20px;  display: block;  }  .submit-btn {  width: 100%;  padding: 12px;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  color: white;  border: none;  border-radius: 8px;  font-size: 18px;  cursor: pointer;  transition: all 0.3s ease;  }  .submit-btn:hover {  transform: translateY(-2px);  box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);  }  .submit-btn:disabled {  background: #ccc;  cursor: not-allowed;  transform: none;  box-shadow: none;  }  .requirements {  font-size: 12px;  color: #666;  margin-top: 5px;  }  </style>  <div class="login-container">  <h2>فرم ثبت نام</h2>  <form id="loginForm">  <div class="form-group">  <label for="username">نام کاربری (فقط انگلیسی)</label>  <input type="text" id="username" name="username"  placeholder="نام کاربری خود را وارد کنید"  oninput="validateUsername()"  onblur="validateUsername()">  <span class="error-message" id="usernameError"></span>  <div class="requirements">نام کاربری باید فقط شامل حروف و اعداد انگلیسی باشد</div>  </div>  <div class="form-group">  <label for="firstName">نام (فقط فارسی)</label>  <input type="text" id="firstName" name="firstName"  placeholder="نام خود را وارد کنید"  oninput="validateFirstName()"  onblur="validateFirstName()">  <span class="error-message" id="firstNameError"></span>  <div class="requirements">نام باید فقط شامل حروف فارسی باشد</div>  </div>  <div class="form-group">  <label for="lastName">نام خانوادگی (فقط فارسی)</label>  <input type="text" id="lastName" name="lastName"  placeholder="نام خانوادگی خود را وارد کنید"  oninput="validateLastName()"  onblur="validateLastName()">  <span class="error-message" id="lastNameError"></span>  <div class="requirements">نام خانوادگی باید فقط شامل حروف فارسی باشد</div>  </div>  <div class="form-group">  <label for="password">رمز عبور</label>  <input type="password" id="password" name="password"  placeholder="رمز عبور خود را وارد کنید"  oninput="validatePassword()"  onblur="validatePassword()">  <span class="error-message" id="passwordError"></span>  <div class="requirements">رمز عبور باید حداقل ۸ کاراکتر و شامل حروف کوچک، بزرگ و کاراکترهای خاص باشد</div>  </div>  <button type="submit" class="submit-btn" id="submitBtn" disabled>ثبت نام</button>  </form>  </div>  <script>  // اعتبارسنجی نام کاربری (فقط انگلیسی)  function validateUsername() {  const username = document.getElementById('username');  const error = document.getElementById('usernameError');  const regex = /^[a-zA-Z0-9\_]+$/;    if (!username.value.trim()) {  showError(username, error, 'نام کاربری الزامی است');  return false;  }    if (!regex.test(username.value)) {  showError(username, error, 'نام کاربری باید فقط شامل حروف و اعداد انگلیسی باشد');  return false;  }    showSuccess(username, error);  return true;  }  // اعتبارسنجی نام (فقط فارسی)  function validateFirstName() {  const firstName = document.getElementById('firstName');  const error = document.getElementById('firstNameError');  const regex = /^[\u0600-\u06FF\s]+$/;    if (!firstName.value.trim()) {  showError(firstName, error, 'نام الزامی است');  return false;  }    if (!regex.test(firstName.value)) {  showError(firstName, error, 'نام باید فقط شامل حروف فارسی باشد');  return false;  }    showSuccess(firstName, error);  return true;  }  // اعتبارسنجی نام خانوادگی (فقط فارسی)  function validateLastName() {  const lastName = document.getElementById('lastName');  const error = document.getElementById('lastNameError');  const regex = /^[\u0600-\u06FF\s]+$/;    if (!lastName.value.trim()) {  showError(lastName, error, 'نام خانوادگی الزامی است');  return false;  }    if (!regex.test(lastName.value)) {  showError(lastName, error, 'نام خانوادگی باید فقط شامل حروف فارسی باشد');  return false;  }    showSuccess(lastName, error);  return true;  }  // اعتبارسنجی رمز عبور  function validatePassword() {  const password = document.getElementById('password');  const error = document.getElementById('passwordError');    if (!password.value.trim()) {  showError(password, error, 'رمز عبور الزامی است');  return false;  }    // حداقل ۸ کاراکتر، شامل حروف کوچک، بزرگ و کاراکترهای خاص  const regex = /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[!@#$%^&\*()\_+\-=\[\]{};':"\\|,.<>\/?]).{8,}$/;    if (!regex.test(password.value)) {  showError(password, error, 'رمز عبور باید حداقل ۸ کاراکتر و شامل حروف کوچک، بزرگ و کاراکترهای خاص باشد');  return false;  }    showSuccess(password, error);  return true;  }  // نمایش خطا  function showError(input, errorElement, message) {  input.classList.remove('valid');  input.classList.add('invalid');  errorElement.textContent = message;  updateSubmitButton();  }  // نمایش موفقیت  function showSuccess(input, errorElement) {  input.classList.remove('invalid');  input.classList.add('valid');  errorElement.textContent = '';  updateSubmitButton();  }  // بروزرسانی وضعیت دکمه ثبت  function updateSubmitButton() {  const submitBtn = document.getElementById('submitBtn');  const isFormValid = validateUsername() &&  validateFirstName() &&  validateLastName() &&  validatePassword();    submitBtn.disabled = !isFormValid;  }  // مدیریت ارسال فرم  document.getElementById('loginForm').addEventListener('submit', function(e) {  e.preventDefault();    if (validateUsername() && validateFirstName() && validateLastName() && validatePassword()) {  alert('فرم با موفقیت ثبت شد!');  // اینجا می‌توانید اطلاعات را به سرور ارسال کنید  }  });  // اعتبارسنجی اولیه هنگام لود صفحه  document.addEventListener('DOMContentLoaded', function() {  updateSubmitButton();  });  </script> |

این کد یک فرم ثبت نام با اعتبارسنجی Real-time ایجاد می‌کند که:

نام کاربری: فقط حروف و اعداد انگلیسی می‌پذیرد

نام و نام خانوادگی: فقط حروف فارسی می‌پذیرد

رمز عبور: حداقل ۸ کاراکتر و شامل حروف کوچک، بزرگ و کاراکترهای خاص

ویژگی‌ها:

اعتبارسنجی در حین تایپ (oninput)

اعتبارسنجی هنگام خروج از فیلد (onblur)

نمایش پیام خطای واضح

تغییر رنگ border فیلدها بر اساس وضعیت اعتبارسنجی

غیرفعال بودن دکمه ثبت تا زمانی که همه فیلدها معتبر نباشند

طراحی ریسپانسیو و زیبا

توضیح گام به گام یک رجکس:

|  |
| --- |
| /^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[!@#$%^&\*]).{8,}$/; |

گام ۱: شروع و پایان رشته

|  |
| --- |
| ^ ... $ |

^ → شروع رشته

$ → پایان رشته

نتیجه: کل رشته باید با الگو مطابقت داشته باشد

Lookahead اول - حروف کوچک

|  |
| --- |
| (?=.\*[a-z]) |

(?= ) → lookahead مثبت (شرط را چک می‌کند اما مصرف نمی‌کند)

.\* → هر تعداد کاراکتر (صفر یا بیشتر)

[a-z] → حداقل یک حرف کوچک انگلیسی

معنی: رشته باید حداقل یک حرف کوچک داشته باشد

|  |
| --- |
| console.log(/(?=.\*[a-z])/.test("A1!")); // false - حرف کوچک ندارد  console.log(/(?=.\*[a-z])/.test("aA1!")); // true - حرف کوچک دارد |

Lookahead دوم - حروف بزرگ

|  |
| --- |
| (?=.\*[A-Z]) |

[A-Z] → حداقل یک حرف بزرگ انگلیسی

معنی: رشته باید حداقل یک حرف بزرگ داشته باشد

|  |
| --- |
| console.log(/(?=.\*[A-Z])/.test("a1!")); // false - حرف بزرگ ندارد  console.log(/(?=.\*[A-Z])/.test("aA1!")); // true - حرف بزرگ دارد |

Lookahead سوم – اعداد

|  |
| --- |
| (?=.\*\d) |

\d → حداقل یک عدد (معادل [0-9])

معنی: رشته باید حداقل یک عدد داشته باشد

مثال:

|  |
| --- |
| console.log(/(?=.\*\d)/.test("aA!")); // false - عدد ندارد  console.log(/(?=.\*\d)/.test("aA1!")); // true - عدد دارد |

Lookahead چهارم - کاراکترهای خاص

|  |
| --- |
| (?=.\*[!@#$%^&\*]) |

[!@#$%^&\*] → حداقل یکی از این کاراکترهای خاص

معنی: رشته باید حداقل یک کاراکتر خاص داشته باشد

مثال:

|  |
| --- |
| console.log(/(?=.\*[!@#$%^&\*])/.test("aA1")); // false - کاراکتر خاص ندارد  console.log(/(?=.\*[!@#$%^&\*])/.test("aA1!")); // true - کاراکتر خاص دارد |

گام ۶: شرط طول

|  |
| --- |
| .{8,} |

. → هر کاراکتر (به جز خط جدید)

{8,} → حداقل ۸ بار

معنی: رشته باید حداقل ۸ کاراکتر داشته باشد

مثال:

|  |
| --- |
| console.log(/.{8,}/.test("1234567")); // false - ۷ کاراکتر  console.log(/.{8,}/.test("12345678")); // true - ۸ کاراکتر |

شرح رجکس:

|  |
| --- |
| **۱. الگوی پایه برای متن فارسی**  /^[\u0600-\u06FF\s]+$/  ۲. الگوی کامل‌تر با علائم نگارشی فارسی /^[\u0600-\u06FF\s\u200C\u200D،؛؟!.]+$/  ۳. الگوی پیشرفته با اعداد فارسی /^[\u0600-\u06FF\s\u200C\u200D،؛؟!.\u06F0-\u06F9]+$/ |

^ → شروع رشته

[\u0600-\u06FF] → محدوده یونیکد حروف فارسی

\s → فاصله و whitespace

+ → یک یا بیشتر

$ → پایان رشته

|  |
| --- |
| // الگوی ساده فارسی  const persianRegex = /^[\u0600-\u06FF\s]+$/;  console.log(persianRegex.test("سلام دنیا")); // true  console.log(persianRegex.test("Hello")); // false  console.log(persianRegex.test("123")); // false  console.log(persianRegex.test("سلام123")); // false |

الگوهای مختلف بر اساس نیاز:

۱. فقط حروف فارسی (بدون فاصله):

|  |
| --- |
| /^[\u0600-\u06FF]+$/ |

فارسی با فاصله:

|  |
| --- |
| /^[\u0600-\u06FF\s]+$/ |

فارسی با علائم نگارشی:

|  |
| --- |
| /^[\u0600-\u06FF\s،؛؟!.]+$/ |

فارسی با اعداد:

|  |
| --- |
| /^[\u0600-\u06FF\s\u06F0-\u06F9]+$/ |

فارسی کامل (همه موارد):

|  |
| --- |
| /^[\u0600-\u06FF\s\u200C\u200D،؛؟!.\u06F0-\u06F9]+$/ |

## **استفاده از متد های search و replace**

متد search، معرفی:

متد search اولین ایندکس تطابق regex را در رشته پیدا می‌کند.

ساختار:

|  |
| --- |
| string.search(regex) |

مثال‌ها:

|  |
| --- |
| const text = "Hello World! Hello JavaScript!";  // جستجوی ساده  console.log(text.search(/Hello/)); // 0  // جستجوی با حروف بزرگ و کوچک  console.log(text.search(/hello/i)); // 0  // جستجوی کلمه‌ای که وجود ندارد  console.log(text.search(/Python/)); // -1  // جستجوی عدد در رشته  console.log(text.search(/\d/)); // -1  // جستجوی کاراکتر خاص  console.log(text.search(/!/)); // 11 |

مثال‌های کاربردی:

|  |
| --- |
| // بررسی وجود کلمه در متن  const article = "این یک مقاله درباره جاوااسکریپت است";  if (article.search(/جاوااسکریپت/) !== -1) {  console.log("مقاله درباره جاوااسکریپت است");  }  // پیدا کردن موقعیت اولین فاصله  const name = "علی رضا محمدی";  console.log(name.search(/\s/)); // 2 (موقعیت اولین فاصله) |

متد replace()

معرفی: متد replace() مقادیر تطابق‌یافته را با مقدار جدید جایگزین می‌کند.

ساختار:

|  |
| --- |
| string.replace(regex, replacement) |

مثال‌ها:

|  |
| --- |
| const text = "Hello World! Hello JavaScript!";  // جایگزینی ساده  console.log(text.replace(/Hello/, "Hi"));  // "Hi World! Hello JavaScript!"  // جایگزینی (همه موارد)  console.log(text.replace(/Hello/g, "Hi"));  // "Hi World! Hi JavaScript!"  // جایگزینی با حروف بزرگ و کوچک  console.log(text.replace(/hello/gi, "Hi"));  // "Hi World! Hi JavaScript!" |

مثال‌های پیشرفته:

|  |
| --- |
| // فرمت کردن شماره تلفن  const phone = "09123456789";  const formatted = phone.replace(/^(\d{4})(\d{3})(\d{4})$/, "$1-$2-$3");  console.log(formatted); // "0912-345-6789"  // حذف اعداد از متن  const mixedText = "من 25 سال دارم و 70 کیلوگرم وزن دارم";  const noNumbers = mixedText.replace(/\d+/g, "");  console.log(noNumbers); // "من سال دارم و کیلوگرم وزن دارم"  // جایگزینی با تابع  const text = "price: 100, price: 200, price: 300";  const increased = text.replace(/\d+/g, (match) => parseInt(match) + 50);  console.log(increased); // "price: 150, price: 250, price: 350" |

## **معرفی modifier ها در search**

معرفی Modifierها:

Modifierها تنظیمات global regex هستند که رفتار جستجو را تغییر می‌دهند.

انواع Modifierها:

g – Global

|  |
| --- |
| const text = "hello world hello";  console.log(text.match(/hello/)); // ["hello"] (فقط اولین)  console.log(text.match(/hello/g)); // ["hello", "hello"] (همه موارد) |

i - Case Insensitive

|  |
| --- |
| const text = "Hello WORLD";  console.log(text.match(/hello/)); // null  console.log(text.match(/hello/i)); // ["Hello"]  console.log(text.match(/world/i)); // ["WORLD"] |

m – Multiline

|  |
| --- |
| const text = "first line\nsecond line\nthird line";  // بدون m - فقط ابتدای کل متن  console.log(text.match(/^second/)); // null  // با m - ابتدای هر خط  console.log(text.match(/^second/m)); // ["second"] |

s - Dot All

|  |
| --- |
| const text = "hello\nworld";  // بدون s - نقطه شامل خط جدید نمی‌شود  console.log(text.match(/hello.world/)); // null  // با s - نقطه شامل خط جدید می‌شود  console.log(text.match(/hello.world/s)); // ["hello\nworld"] |

u – Unicode

|  |
| --- |
| const text = "😀";  // بدون u - مشکل با emoji  console.log(text.match(/./)); // ["�"]  // با u - پشتیبانی از unicode  console.log(text.match(/./u)); // ["😀"] |

مثال ترکیبی با Modifierها:

|  |
| --- |
| const text = "Hello World!\nhello JavaScript!\nHELLO Node.js!";  // جستجوی global + case insensitive  const matches = text.match(/hello/gi);  console.log(matches); // ["Hello", "hello", "HELLO"]  // جایگزینی global + case insensitive  const replaced = text.replace(/hello/gi, "Hi");  console.log(replaced);  // "Hi World!\nHi JavaScript!\nHi Node.js!"  // جستجوی چند خطی  const lineStarts = text.match(/^hello/gim);  console.log(lineStarts); // ["Hello", "hello", "HELLO"] |

## استفاده از parentnode و childrennode و firstchild و lastchild و nextsibiling و previoussibiling

۱. parentNode - عنصر والد

|  |
| --- |
| <div id="parent">  <p id="child">این یک پاراگراف است</p>  </div>  <script>  const child = document.getElementById("child");  console.log(child.parentNode.id); // "parent"  </script> |

۲. children - فرزندان مستقیم

|  |
| --- |
| <ul id="list">  <li>آیتم ۱</li>  <li>آیتم ۲</li>  <li>آیتم ۳</li>  </ul>  <script>  const list = document.getElementById("list");  console.log(list.children.length); // 3  console.log(list.children[0].textContent); // "آیتم ۱"  </script> |

۳. firstChild / firstElementChild

|  |
| --- |
| <div id="container">  <span>اولین فرزند</span>  <p>دومین فرزند</p>  </div>  <script>  const container = document.getElementById("container");    // firstChild - شامل text nodes هم می‌شود  console.log(container.firstChild); // #text (فاصله)    // firstElementChild - فقط element nodes  console.log(container.firstElementChild.textContent); // "اولین فرزند"  </script> |

۴. lastChild / lastElementChild

|  |
| --- |
| <div id="container">  <p>اولین فرزند</p>  <span>آخرین فرزند</span>  </div>  <script>  const container = document.getElementById("container");  console.log(container.lastElementChild.textContent); // "آخرین فرزند"  </script> |

۵. nextSibling / nextElementSibling

|  |
| --- |
| <ul>  <li id="item1">آیتم ۱</li>  <li id="item2">آیتم ۲</li>  <li id="item3">آیتم ۳</li>  </ul>  <script>  const item1 = document.getElementById("item1");  console.log(item1.nextElementSibling.textContent); // "آیتم ۲"    const item2 = document.getElementById("item2");  console.log(item2.nextElementSibling.textContent); // "آیتم ۳"  </script> |

۶. previousSibling / previousElementSibling

|  |
| --- |
| <ul>  <li id="item1">آیتم ۱</li>  <li id="item2">آیتم ۲</li>  <li id="item3">آیتم ۳</li>  </ul>  <script>  const item3 = document.getElementById("item3");  console.log(item3.previousElementSibling.textContent); // "آیتم ۲"    const item2 = document.getElementById("item2");  console.log(item2.previousElementSibling.textContent); // "آیتم ۱"  </script> |

## معرفی createElement و append و insertBefore و remove

۱. createElement - ایجاد عنصر جدید

|  |
| --- |
| // ایجاد یک div جدید  const newDiv = document.createElement("div");  newDiv.textContent = "این یک div جدید است";  newDiv.className = "my-class";  // ایجاد یک دکمه جدید  const newButton = document.createElement("button");  newButton.textContent = "کلیک کنید";  newButton.id = "myButton"; |

۲. append - اضافه کردن به انتها

|  |
| --- |
| <div id="container"></div>  <script>  const container = document.getElementById("container");    // ایجاد عناصر جدید  const title = document.createElement("h2");  title.textContent = "عنوان جدید";    const paragraph = document.createElement("p");  paragraph.textContent = "این یک پاراگراف جدید است";    // اضافه کردن به انتهای container  container.append(title, paragraph);  </script> |

۳. insertBefore - اضافه کردن قبل از عنصر مشخص

|  |
| --- |
| <ul id="myList">  <li id="item2">آیتم ۲</li>  <li id="item3">آیتم ۳</li>  </ul>  <script>  const list = document.getElementById("myList");  const item2 = document.getElementById("item2");    // ایجاد آیتم جدید  const newItem = document.createElement("li");  newItem.textContent = "آیتم ۱";  newItem.id = "item1";    // اضافه کردن قبل از item2  list.insertBefore(newItem, item2);  </script> |

۴. remove - حذف عنصر

|  |
| --- |
| <div id="container">  <p id="paragraph1">پاراگراف ۱</p>  <p id="paragraph2">پاراگراف ۲</p>  <p id="paragraph3">پاراگراف ۳</p>  </div>  <button onclick="removeSecond()">حذف پاراگراف دوم</button>  <script>  function removeSecond() {  const paragraph2 = document.getElementById("paragraph2");  paragraph2.remove();  }  </script> |

مثال کاربردی کامل

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .container { border: 2px solid #333; padding: 20px; margin: 10px; }  .item { background: #f0f0f0; margin: 5px; padding: 10px; border: 1px solid #ccc; }  button { margin: 5px; padding: 8px 15px; }  </style>  </head>  <body>  <div id="app">  <div class="container" id="listContainer">  <div class="item" id="item1">آیتم ۱</div>  <div class="item" id="item2">آیتم ۲</div>  <div class="item" id="item3">آیتم ۳</div>  </div>    <button onclick="addItem()">اضافه کردن آیتم</button>  <button onclick="removeLast()">حذف آخرین آیتم</button>  <button onclick="showInfo()">نمایش اطلاعات</button>  </div>  <script>  function addItem() {  const container = document.getElementById("listContainer");    // ایجاد آیتم جدید  const newItem = document.createElement("div");  newItem.className = "item";  newItem.textContent = `آیتم ${container.children.length + 1}`;    // اضافه کردن به انتها  container.append(newItem);  }    function removeLast() {  const container = document.getElementById("listContainer");    if (container.lastElementChild) {  container.lastElementChild.remove();  }  }    function showInfo() {  const container = document.getElementById("listContainer");  const firstItem = container.firstElementChild;  const lastItem = container.lastElementChild;    let info = `تعداد فرزندان: ${container.children.length}\n`;  info += `اولین آیتم: ${firstItem ? firstItem.textContent : 'ندارد'}\n`;  info += `آخرین آیتم: ${lastItem ? lastItem.textContent : 'ندارد'}`;    alert(info);  }  </script>  </body>  </html> |

مثال پیشرفته - مدیریت لیست پویا

|  |
| --- |
| <div id="taskManager">  <h3>مدیریت وظایف</h3>  <input type="text" id="taskInput" placeholder="وظیفه جدید...">  <button onclick="addTask()">اضافه کردن</button>  <ul id="taskList"></ul>  </div>  <script>  function addTask() {  const taskInput = document.getElementById("taskInput");  const taskList = document.getElementById("taskList");  const taskText = taskInput.value.trim();    if (taskText === "") return;    // ایجاد آیتم جدید  const newTask = document.createElement("li");  newTask.innerHTML = `  ${taskText}  <button onclick="completeTask(this)">انجام شد</button>  <button onclick="removeTask(this)">حذف</button>  `;    // اضافه کردن به ابتدای لیست  taskList.insertBefore(newTask, taskList.firstChild);    // خالی کردن input  taskInput.value = "";  }    function completeTask(button) {  const taskItem = button.parentNode;  taskItem.style.textDecoration = "line-through";  taskItem.style.color = "green";  }    function removeTask(button) {  const taskItem = button.parentNode;  taskItem.remove();  }  </script> |

💡 نکات مهم:

Element نسخه‌ها بهتر از Node نسخه‌ها هستند (مثلاً firstElementChild بهتر از firstChild)

append() می‌تواند چندین عنصر را بپذیرد

insertBefore() نیاز به reference element دارد

remove() ساده‌تر از removeChild() است

## معرفی bom

BOM یا Browser Object Model به توسعه‌دهندگان اجازه می‌دهد با مرورگر تعامل داشته باشند. برخلاف DOM که استاندارد است، BOM استاندارد رسمی ندارد اما همه مرورگرها از آن پشتیبانی می‌کنند.

Screen.width

|  |
| --- |
| console.log(`عرض کل صفحه: ${screen.width}px`);  // مثال: 1920px |

Screen.height

|  |
| --- |
| console.log(`ارتفاع کل صفحه: ${screen.height}px`);  // مثال: 1080px |

Screen.availwidth - عرض قابل استفاده

|  |
| --- |
| console.log(`عرض قابل استفاده: ${screen.availWidth}px`);  // مثال: 1920px (منهای تسکبار) |

Screen.availheight - **ارتفاع قابل استفاده**

|  |
| --- |
| console.log(`ارتفاع قابل استفاده: ${screen.availHeight}px`);  // مثال: 1040px (منهای تسکبار) |

مثال کاربردی:

|  |
| --- |
| <button onclick="showScreenInfo()">نمایش اطلاعات صفحه</button>  <div id="screenInfo"></div>  <script>  function showScreenInfo() {  const info = `  📺 اطلاعات صفحه نمایش:  • عرض کل: ${screen.width}px  • ارتفاع کل: ${screen.height}px  • عرض قابل استفاده: ${screen.availWidth}px  • ارتفاع قابل استفاده: ${screen.availHeight}px  `;  document.getElementById('screenInfo').innerHTML = info;  }  </script> |

Innerwidth- window.innerWidth - عرض داخلی پنجره

|  |
| --- |
| console.log(`عرض داخلی پنجره: ${window.innerWidth}px`);  // شامل اسکرولبار می‌شود |

Innerheight - window.innerHeight - ارتفاع داخلی پنجره

|  |
| --- |
| console.log(`ارتفاع داخلی پنجره: ${window.innerHeight}px`);  // شامل اسکرولبار می‌شود |

مثال کاربردی:

|  |
| --- |
| <button onclick="showWindowInfo()">نمایش اندازه پنجره</button>  <div id="windowInfo"></div>  <script>  function showWindowInfo() {  const info = `  🪟 اندازه پنجره مرورگر:  • عرض داخلی: ${window.innerWidth}px  • ارتفاع داخلی: ${window.innerHeight}px  `;  document.getElementById('windowInfo').innerHTML = info;  }  // ردیابی تغییر اندازه پنجره  window.addEventListener('resize', function() {  console.log(`پنجره تغییر کرد: ${window.innerWidth}x${window.innerHeight}`);  });  </script> |

Location Object - اطلاعات آدرس

Location.href - آدرس کامل

|  |
| --- |
| console.log(`آدرس کامل: ${location.href}`);  // مثال: https://example.com/products/page.html |

Location.hostname - نام هاست

|  |
| --- |
| console.log(`نام هاست: ${location.hostname}`);  // مثال: example.com |

Location.pathname – مسیر

|  |
| --- |
| console.log(`مسیر: ${location.pathname}`);  // مثال: /products/page.html |

Location.protocol - **پروتکل**

|  |
| --- |
| console.log(`پروتکل: ${location.protocol}`);  // مثال: https: |

مثال کاربردی:

|  |
| --- |
| <button onclick="showLocationInfo()">نمایش اطلاعات آدرس</button>  <button onclick="redirectToGoogle()">برو به گوگل</button>  <div id="locationInfo"></div>  <script>  function showLocationInfo() {  const info = `  🌐 اطلاعات آدرس فعلی:  • آدرس کامل: ${location.href}  • نام هاست: ${location.hostname}  • مسیر: ${location.pathname}  • پروتکل: ${location.protocol}  `;  document.getElementById('locationInfo').innerHTML = info;  }  function redirectToGoogle() {  location.href = 'https://google.com';  }  // تغییر مسیر پس از 5 ثانیه  // setTimeout(() => {  // location.href = 'https://example.com';  // }, 5000);  </script> |

مثال جامع BOM

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <style>  .info-box {  border: 2px solid #333;  padding: 20px;  margin: 10px;  border-radius: 10px;  background: #f9f9f9;  }  .screen { border-color: #ff6b6b; }  .window { border-color: #4ecdc4; }  .location { border-color: #45b7d1; }  button {  padding: 10px 20px;  margin: 5px;  background: #007bff;  color: white;  border: none;  border-radius: 5px;  cursor: pointer;  }  </style>  </head>  <body>  <h2>نمایش اطلاعات BOM</h2>    <div class="info-box screen">  <h3>📺 اطلاعات صفحه نمایش</h3>  <button onclick="updateScreenInfo()">بروزرسانی</button>  <div id="screenInfo"></div>  </div>    <div class="info-box window">  <h3>🪟 اطلاعات پنجره</h3>  <button onclick="updateWindowInfo()">بروزرسانی</button>  <div id="windowInfo"></div>  </div>    <div class="info-box location">  <h3>🌐 اطلاعات آدرس</h3>  <button onclick="updateLocationInfo()">بروزرسانی</button>  <div id="locationInfo"></div>  </div>  <script>  // اطلاعات صفحه نمایش  function updateScreenInfo() {  document.getElementById('screenInfo').innerHTML = `  <p>عرض کل: <strong>${screen.width}px</strong></p>  <p>ارتفاع کل: <strong>${screen.height}px</strong></p>  <p>عرض قابل استفاده: <strong>${screen.availWidth}px</strong></p>  <p>ارتفاع قابل استفاده: <strong>${screen.availHeight}px</strong></p>  `;  }    // اطلاعات پنجره  function updateWindowInfo() {  document.getElementById('windowInfo').innerHTML = `  <p>عرض داخلی: <strong>${window.innerWidth}px</strong></p>  <p>ارتفاع داخلی: <strong>${window.innerHeight}px</strong></p>  `;  }    // اطلاعات آدرس  function updateLocationInfo() {  document.getElementById('locationInfo').innerHTML = `  <p>آدرس کامل: <strong>${location.href}</strong></p>  <p>نام هاست: <strong>${location.hostname}</strong></p>  <p>مسیر: <strong>${location.pathname}</strong></p>  <p>پروتکل: <strong>${location.protocol}</strong></p>  `;  }    // بروزرسانی خودکار هنگام تغییر اندازه پنجره  window.addEventListener('resize', updateWindowInfo);    // بارگذاری اولیه  updateScreenInfo();  updateWindowInfo();  updateLocationInfo();  </script>  </body>  </html> |

مثال کاربردی - تشخیص دستگاه

|  |
| --- |
| function detectDevice() {  const isMobile = window.innerWidth <= 768;  const isTablet = window.innerWidth > 768 && window.innerWidth <= 1024;  const isDesktop = window.innerWidth > 1024;    console.log(`عرض صفحه: ${window.innerWidth}px`);    if (isMobile) {  console.log('📱 دستگاه موبایل');  } else if (isTablet) {  console.log('📟 دستگاه تبلت');  } else {  console.log('💻 دستگاه دسکتاپ');  }  }  // اجرا هنگام بارگذاری و تغییر اندازه  detectDevice();  window.addEventListener('resize', detectDevice); |

## معرفی setTimeOut و setInterval

setTimeout - اجرای تابع پس از تاخیر

|  |
| --- |
| setTimeout(تابع, زمان\_تاخیر, آرگومان1, آرگومان2, ...) |

مثال‌های ساده:

|  |
| --- |
| // مثال ۱: اجرای ساده پس از ۲ ثانیه  setTimeout(function() {  console.log("این پیام پس از ۲ ثانیه نمایش داده می‌شود");  }, 2000);  // مثال ۲: استفاده از arrow function  setTimeout(() => {  console.log("سلام بعد از ۳ ثانیه!");  }, 3000);  // مثال ۳: ارسال آرگومان به تابع  setTimeout((name, age) => {  console.log(`سلام ${name}, سن شما ${age} سال است`);  }, 1000, "علی", 25); |

مثال کاربردی:

|  |
| --- |
| <button onclick="showMessage()">کلیک کن</button>  <div id="message"></div>  <script>  function showMessage() {  const messageDiv = document.getElementById("message");  messageDiv.innerHTML = "در حال بارگذاری...";    // پنهان کردن پیام پس از ۳ ثانیه  setTimeout(() => {  messageDiv.innerHTML = "عملیات کامل شد!";  }, 3000);  }  </script> |

setInterval - اجرای تابع در فواصل زمانی

ساختار:

|  |
| --- |
| setInterval(تابع, فاصله\_زمانی, آرگومان1, آرگومان2, ...) |

مثال‌های ساده:

|  |
| --- |
| // مثال ۱: نمایش زمان هر ثانیه  setInterval(() => {  console.log(`زمان: ${new Date().toLocaleTimeString()}`);  }, 1000);  // مثال ۲: شمارش معکوس  let count = 10;  const countdown = setInterval(() => {  console.log(count);  count--;    if (count < 0) {  clearInterval(countdown);  console.log("شمارش تمام شد!");  }  }, 1000); |

مثال کاربردی:

|  |
| --- |
| <div id="timer">10</div>  <button onclick="startTimer()">شروع تایمر</button>  <button onclick="stopTimer()">توقف تایمر</button>  <script>  let timerInterval;  let timeLeft = 10;  function startTimer() {  const timerDiv = document.getElementById("timer");    timerInterval = setInterval(() => {  timeLeft--;  timerDiv.textContent = timeLeft;    if (timeLeft <= 0) {  clearInterval(timerInterval);  timerDiv.textContent = "زمان تمام شد!";  }  }, 1000);  }  function stopTimer() {  clearInterval(timerInterval);  }  </script> |

متوقف کردن setTimeout و setInterval

clearTimeout - متوقف کردن setTimeout

|  |
| --- |
| // ذخیره timeout در یک متغیر  const timeoutId = setTimeout(() => {  console.log("این اجرا نمی‌شود");  }, 5000);  // متوقف کردن قبل از اجرا  clearTimeout(timeoutId); |

clearInterval - متوقف کردن setInterval

|  |
| --- |
| let counter = 0;  const intervalId = setInterval(() => {  counter++;  console.log(`شماره: ${counter}`);    if (counter >= 5) {  clearInterval(intervalId);  console.log("متوقف شد!");  }  }, 1000); |

مثال‌های پیشرفته

مثال ۱: اسلایدشو تصاویر

|  |
| --- |
| <div id="slideshow">  <img id="slide" src="image1.jpg" width="400">  </div>  <button onclick="startSlideshow()">شروع اسلایدشو</button>  <button onclick="stopSlideshow()">توقف اسلایدشو</button>  <script>  let slideshowInterval;  let currentSlide = 1;  const totalSlides = 5;  function startSlideshow() {  slideshowInterval = setInterval(() => {  currentSlide = (currentSlide % totalSlides) + 1;  document.getElementById("slide").src = `image${currentSlide}.jpg`;  }, 3000);  }  function stopSlideshow() {  clearInterval(slideshowInterval);  }  </script> |

مثال ۲: تایپ کردن متنی

|  |
| --- |
| <div id="typingText"></div>  <script>  function typeWriter(text, elementId, speed = 100) {  const element = document.getElementById(elementId);  let i = 0;    element.innerHTML = "";    const typingInterval = setInterval(() => {  if (i < text.length) {  element.innerHTML += text.charAt(i);  i++;  } else {  clearInterval(typingInterval);  }  }, speed);  }  // استفاده  typeWriter("این یک متن نمونه است که تایپ می‌شود...", "typingText", 50);  </script> |

## **ساخت ساعت دیجیتال**

|  |
| --- |
| <!DOCTYPE html>  <html lang="fa" dir="rtl">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>ساعت دیجیتال زیبا</title>  <style>  \* {  margin: 0;  padding: 0;  box-sizing: border-box;  }  body {  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);  min-height: 100vh;  display: flex;  justify-content: center;  align-items: center;  padding: 20px;  }  .clock-container {  background: rgba(255, 255, 255, 0.1);  backdrop-filter: blur(10px);  border-radius: 20px;  padding: 40px;  box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);  border: 1px solid rgba(255, 255, 255, 0.2);  text-align: center;  min-width: 400px;  }  .clock-title {  color: white;  font-size: 24px;  margin-bottom: 30px;  text-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);  }  .digital-clock {  background: rgba(0, 0, 0, 0.7);  border-radius: 15px;  padding: 30px 20px;  margin: 20px 0;  border: 2px solid rgba(255, 255, 255, 0.3);  }  .time {  font-size: 4rem;  font-weight: bold;  color: #00ff88;  text-shadow: 0 0 20px #00ff88;  font-family: 'Courier New', monospace;  letter-spacing: 5px;  }  .date {  font-size: 1.5rem;  color: #ffffff;  margin-top: 15px;  text-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);  }  .day {  font-size: 1.2rem;  color: #ffd700;  margin-top: 10px;  text-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);  }  .controls {  margin-top: 30px;  display: flex;  gap: 15px;  justify-content: center;  }  .control-btn {  background: rgba(255, 255, 255, 0.2);  border: none;  color: white;  padding: 12px 25px;  border-radius: 25px;  cursor: pointer;  font-size: 14px;  transition: all 0.3s ease;  backdrop-filter: blur(10px);  }  .control-btn:hover {  background: rgba(255, 255, 255, 0.3);  transform: translateY(-2px);  }  .theme-toggle {  position: absolute;  top: 20px;  left: 20px;  background: rgba(255, 255, 255, 0.2);  border: none;  color: white;  padding: 10px 15px;  border-radius: 20px;  cursor: pointer;  backdrop-filter: blur(10px);  }  /\* تم تاریک \*/  body.dark-theme {  background: linear-gradient(135deg, #2c3e50 0%, #34495e 100%);  }  body.dark-theme .digital-clock {  background: rgba(255, 255, 255, 0.1);  border: 2px solid rgba(255, 255, 255, 0.1);  }  body.dark-theme .time {  color: #ff6b6b;  text-shadow: 0 0 20px #ff6b6b;  }  /\* انیمیشن \*/  @keyframes glow {  0%, 100% { opacity: 1; }  50% { opacity: 0.8; }  }  .time {  animation: glow 2s ease-in-out infinite;  }  /\* ریسپانسیو \*/  @media (max-width: 480px) {  .clock-container {  min-width: 90%;  padding: 30px 20px;  }    .time {  font-size: 3rem;  letter-spacing: 3px;  }    .date {  font-size: 1.2rem;  }  }  </style>  </head>  <body>  <button class="theme-toggle" onclick="toggleTheme()">تغییر تم</button>    <div class="clock-container">  <h1 class="clock-title">🕒 ساعت دیجیتال</h1>    <div class="digital-clock">  <div class="time" id="time">00:00:00</div>  <div class="date" id="date">--- -- ----</div>  <div class="day" id="day">------</div>  </div>    <div class="controls">  <button class="control-btn" onclick="toggleFormat()">تغییر فرمت</button>  <button class="control-btn" onclick="toggleSeconds()">نمایش/مخفی ثانیه</button>  <button class="control-btn" onclick="playSound()">پخش صدا</button>  </div>  </div>  <script>  // تنظیمات اولیه  let is24Hour = true;  let showSeconds = true;  let currentTheme = 'light';  // نام روزهای هفته به فارسی  const days = ['یکشنبه', 'دوشنبه', 'سه شنبه', 'چهارشنبه', 'پنجشنبه', 'جمعه', 'شنبه'];    // نام ماه‌ها به فارسی  const months = [  'فروردین', 'اردیبهشت', 'خرداد', 'تیر', 'مرداد', 'شهریور',  'مهر', 'آبان', 'آذر', 'دی', 'بهمن', 'اسفند'  ];  function updateClock() {  const now = new Date();    // زمان  let hours = now.getHours();  let minutes = now.getMinutes();  let seconds = now.getSeconds();    // فرمت 12 ساعته  if (!is24Hour) {  const ampm = hours >= 12 ? 'PM' : 'AM';  hours = hours % 12 || 12;  }    // اضافه کردن صفر به اعداد یک رقمی  hours = hours.toString().padStart(2, '0');  minutes = minutes.toString().padStart(2, '0');  seconds = seconds.toString().padStart(2, '0');    // نمایش زمان  const timeString = showSeconds ?  `${hours}:${minutes}:${seconds}` :  `${hours}:${minutes}`;    document.getElementById('time').textContent = timeString;    // تاریخ  const day = days[now.getDay()];  const date = now.getDate();  const month = months[now.getMonth()];  const year = now.getFullYear();    document.getElementById('date').textContent = `${date} ${month} ${year}`;  document.getElementById('day').textContent = day;  }  // تغییر فرمت 12/24 ساعته  function toggleFormat() {  is24Hour = !is24Hour;  updateClock();  }  // نمایش/مخفی کردن ثانیه  function toggleSeconds() {  showSeconds = !showSeconds;  updateClock();  }  // تغییر تم  function toggleTheme() {  currentTheme = currentTheme === 'light' ? 'dark' : 'light';  document.body.classList.toggle('dark-theme');  }  // پخش صدای تیک تاک  function playSound() {  const audio = new Audio('data:audio/wav;base64,UklGRigAAABXQVZFZm10IBAAAAABAAEARKwAAIhYAQACABAAZGF0YQQAAAAAAA==');  audio.play().catch(e => console.log('پخش صدا امکان پذیر نیست'));  }  // به روز رسانی هر ثانیه  setInterval(updateClock, 1000);    // اجرای اولیه  updateClock();  // انیمیشن برای تغییرات  function animateTimeChange() {  const timeElement = document.getElementById('time');  timeElement.style.transform = 'scale(1.1)';  setTimeout(() => {  timeElement.style.transform = 'scale(1)';  }, 300);  }  // اضافه کردن انیمیشن به تغییرات ثانیه  const originalUpdateClock = updateClock;  updateClock = function() {  originalUpdateClock();  if (showSeconds) {  animateTimeChange();  }  };  </script>  </body>  </html> |

## **معرفی json**

JSON (JavaScript Object Notation) یک فرمت سبک برای تبادل داده است که هم برای انسان قابل خواندن و هم برای ماشین قابل پردازش است.

ساختار پایه:

|  |
| --- |
| {  "name": "علی",  "age": 25,  "isStudent": true,  "courses": ["ریاضی", "فیزیک", "شیمی"],  "address": {  "city": "تهران",  "street": "ولیعصر"  }  } |

معرفی توابع .parse و . stringfy

۱. JSON.parse - تبدیل JSON به Object

|  |
| --- |
| // JSON string  const jsonString = '{"name": "علی", "age": 25, "city": "تهران"}';  // تبدیل به object  const obj = JSON.parse(jsonString);  console.log(obj.name); // "علی"  console.log(obj.age); // 25 |

۲. JSON.stringify() - تبدیل Object به JSON

|  |
| --- |
| // JavaScript object  const person = {  name: "علی",  age: 25,  isStudent: true,  courses: ["ریاضی", "فیزیک"]  };  // تبدیل به JSON string  const jsonString = JSON.stringify(person);  console.log(jsonString);  // '{"name":"علی","age":25,"isStudent":true,"courses":["ریاضی","فیزیک"]}' |

بررسی تفاوت json و object

JSON:

|  |
| --- |
| {  "name": "علی",  "age": 25,  "isStudent": true,  "courses": ["ریاضی", "فیزیک"]  } |

JavaScript Object:

|  |
| --- |
| const person = {  name: "علی",  age: 25,  isStudent: true,  courses: ["ریاضی", "فیزیک"],  // می‌تواند شامل توابع باشد  greet: function() {  return `سلام، من ${this.name} هستم`;  },  // می‌تواند شامل undefined باشد  nickname: undefined  }; |

مقایسه:

|  |  |  |
| --- | --- | --- |
| ویژگی | JSON | JavaScript Object |
| توابع | پشتیبانی نمی‌شود | پشتیبانی می‌شود |
| undefined | پشتیبانی نمی‌شود | پشتیبانی می‌شود |
| تاریخ | پشتیبانی نمی‌شود | پشتیبانی می‌شود |
| کلید بدون دابل کوتیشن | پشتیبانی نمی‌شود | پشتیبانی می‌شود |
| توضیحات | پشتیبانی نمی‌شود | پشتیبانی می‌شود |

بررسی تفاوت json و xml

JSON:

|  |
| --- |
| {  "students": [  {  "id": 1,  "name": "علی",  "age": 20,  "courses": ["ریاضی", "فیزیک"]  },  {  "id": 2,  "name": "رضا",  "age": 22,  "courses": ["شیمی", "زیست"]  }  ]  } |

XML:

|  |
| --- |
| <students>  <student>  <id>1</id>  <name>علی</name>  <age>20</age>  <courses>  <course>ریاضی</course>  <course>فیزیک</course>  </courses>  </student>  <student>  <id>2</id>  <name>رضا</name>  <age>22</age>  <courses>  <course>شیمی</course>  <course>زیست</course>  </courses>  </student>  </students> |

مقایسه JSON و XML:

|  |  |  |
| --- | --- | --- |
| معیار | JSON | XML |
| حجم | کوچک‌تر | بزرگ‌تر |
| خوانایی | بهتر | پیچیده‌تر |
| پردازش | سریع‌تر | کندتر |
| ساختار | ساده‌تر | پیچیده‌تر |
| پشتیبانی زبان‌ها | عالی | خوب |

بررسی انواع داده ها در json

String (رشته)

|  |
| --- |
| {  "name": "علی",  "description": "این یک متن نمونه است"  } |

Number (عدد)

|  |
| --- |
| {  "age": 25,  "price": 29.99,  "temperature": -5  } |

Boolean (صحیح/غلط)

|  |
| --- |
| {  "isActive": true,  "isStudent": false  } |

Array (آرایه)

|  |
| --- |
| {  "colors": ["قرمز", "سبز", "آبی"],  "numbers": [1, 2, 3, 4, 5]  } |

Object (شیء)

|  |
| --- |
| {  "person": {  "firstName": "علی",  "lastName": "محمدی"  }  } |

null (تهی)

|  |
| --- |
| {  "middleName": null  } |

داده‌های غیرمجاز در JSON:

|  |
| --- |
| // این موارد در JSON مجاز نیستند:  const invalidJSON = {  date: new Date(), // تاریخ  func: function() {}, // تابع  undef: undefined, // undefined  nan: NaN, // NaN  infinity: Infinity // بی‌نهایت  }; |

ساخت فایل json

روش ۱: ایجاد فایل دستی

یک فایل جدید با پسوند .json ایجاد کنید

محتوای JSON را در آن بنویسید

ذخیره با encoding UTF-8

مثال: students.json

|  |
| --- |
| {  "students": [  {  "id": 1,  "name": "علی رضایی",  "age": 20,  "major": "کامپیوتر",  "grades": {  "math": 18.5,  "physics": 17,  "programming": 19.25  },  "isActive": true  },  {  "id": 2,  "name": "فاطمه محمدی",  "age": 21,  "major": "برق",  "grades": {  "math": 19,  "physics": 18.75,  "programming": 16.5  },  "isActive": true  }  ],  "totalStudents": 2,  "lastUpdate": "2024-01-15"  } |

روش ۲: ایجاد فایل با JavaScript

|  |
| --- |
| // ایجاد داده  const data = {  company: "شرکت نمونه",  employees: [  { id: 1, name: "علی", position: "برنامه‌نویس" },  { id: 2, name: "مریم", position: "طراح" }  ],  established: 2020  };  // تبدیل به JSON  const jsonData = JSON.stringify(data, null, 2);  // ایجاد فایل و دانلود  function downloadJSON() {  const blob = new Blob([jsonData], { type: 'application/json' });  const url = URL.createObjectURL(blob);  const a = document.createElement('a');  a.href = url;  a.download = 'company-data.json';  a.click();  URL.revokeObjectURL(url);  } |

مثال‌های کاربردی

مثال ۱: ذخیره و بازیابی از LocalStorage

|  |
| --- |
| // ذخیره در localStorage  const userSettings = {  theme: "dark",  language: "fa",  notifications: true,  fontSize: 14  };  localStorage.setItem('userSettings', JSON.stringify(userSettings));  // بازیابی از localStorage  const savedSettings = JSON.parse(localStorage.getItem('userSettings'));  console.log(savedSettings.theme); // "dark" |

مثال ۲: پردازش داده‌های پیچیده

|  |
| --- |
| const universityData = {  "university": "دانشگاه تهران",  "departments": [  {  "name": "کامپیوتر",  "head": "دکتر احمدی",  "students": 300,  "courses": ["پایگاه داده", "هوش مصنوعی", "شبکه"]  },  {  "name": "برق",  "head": "دکتر رضایی",  "students": 250,  "courses": ["مدارهای الکتریکی", "ماشین‌های الکتریکی"]  }  ]  };  // استفاده از داده‌ها  console.log(JSON.stringify(universityData, null, 2)); |

مثال ۳: اعتبارسنجی JSON

|  |
| --- |
| function isValidJSON(str) {  try {  JSON.parse(str);  return true;  } catch (e) {  return false;  }  }  // تست  console.log(isValidJSON('{"name": "علی"}')); // true  console.log(isValidJSON('{name: "علی"}')); // false |

💡 نکات مهم:

همیشه از دابل کوتیشن استفاده کنید

کلیدها باید string باشند

مقادیر می‌توانند: string, number, boolean, array, object, null باشند

از JSON.stringify برای ذخیره‌سازی و ارسال داده استفاده کنید

از JSON.parse برای خواندن داده استفاده کنید

## **بررسی تفاوت sync و async**

برنامه‌نویسی Synchronous (همگام)

مفهوم:

در برنامه‌نویسی همگام، دستورات به ترتیب و یکی پس از دیگری اجرا می‌شوند. هر دستور باید کامل شود تا دستور بعدی اجرا شود.

مثال:

|  |
| --- |
| console.log("شروع برنامه");  function task1() {  console.log("وظیفه ۱ انجام شد");  }  function task2() {  console.log("وظیفه ۲ انجام شد");  }  task1();  task2();  console.log("پایان برنامه");  // خروجی:  // شروع برنامه  // وظیفه ۱ انجام شد  // وظیفه ۲ انجام شد  // پایان برنامه |

ویژگی‌ها:

اجرای خطی و ترتیبی

مسدود کننده (Blocking)

ساده برای درک

کند برای عملیات زمان‌بر

برنامه‌نویسی Asynchronous (ناهمگام)

مفهوم:

در برنامه‌نویسی ناهمگام، دستورات می‌توانند بدون منتظر ماندن برای تکمیل دستورات قبلی اجرا شوند.

مثال:

|  |
| --- |
| console.log("شروع برنامه");  setTimeout(() => {  console.log("وظیفه ۱ انجام شد (بعد از ۲ ثانیه)");  }, 2000);  console.log("پایان برنامه");  // خروجی:  // شروع برنامه  // پایان برنامه  // وظیفه ۱ انجام شد (بعد از ۲ ثانیه) |

ویژگی‌ها:

غیر مسدود کننده (Non-blocking)

کارآمد برای عملیات I/O

پیچیده‌تر برای مدیریت

مناسب برای شبکه و فایل

## **استفاده از callback در async**

مفهوم Callback:

یک تابع که به عنوان آرگومان به تابع دیگر پاس داده می‌شود و بعد از تکمیل یک عملیات ناهمگام فراخوانی می‌شود.

مثال‌های Callback:

مثال ۱: Callback ساده

|  |
| --- |
| function fetchData(callback) {  console.log("در حال دریافت داده...");  setTimeout(() => {  const data = { name: "علی", age: 25 };  callback(data);  }, 2000);  }  // استفاده از callback  fetchData(function(result) {  console.log("داده دریافت شد:", result);  });  console.log("این خط بلافاصله اجرا می‌شود"); |

مثال ۲: Callback با خطا

|  |
| --- |
| function readFile(callback) {  const success = Math.random() > 0.5;    setTimeout(() => {  if (success) {  callback(null, "محتوای فایل");  } else {  callback("خطا در خواندن فایل", null);  }  }, 1000);  }  // استفاده  readFile(function(error, data) {  if (error) {  console.error("خطا:", error);  } else {  console.log("داده:", data);  }  }); |

مثال ۳: Callback Hell (جهنم callback).

|  |
| --- |
| // مشکل: callbackهای تودرتو  function getUserData(userId, callback) {  setTimeout(() => {  console.log("دریافت اطلاعات کاربر");  callback({ id: userId, name: "علی" });  }, 1000);  }  function getUserPosts(userId, callback) {  setTimeout(() => {  console.log("دریافت پست‌های کاربر");  callback(["پست ۱", "پست ۲"]);  }, 1000);  }  function getPostComments(postId, callback) {  setTimeout(() => {  console.log("دریافت نظرات پست");  callback(["نظر ۱", "نظر ۲"]);  }, 1000);  }  // استفاده (Callback Hell)  getUserData(1, function(user) {  getUserPosts(user.id, function(posts) {  getPostComments(posts[0], function(comments) {  console.log("کاربر:", user);  console.log("پست‌ها:", posts);  console.log("نظرات:", comments);  });  });  }); |

## **استفاده از promise در async**

مفهوم Promise:

یک شیء که نشان دهنده تکمیل یا شکست یک عملیات ناهمگام است.

ساختار Promise:

|  |
| --- |
| const myPromise = new Promise((resolve, reject) => {  // عملیات ناهمگام  const success = true;    if (success) {  resolve("عملیات موفق");  } else {  reject("عملیات ناموفق");  }  }); |

مثال‌های Promise:

مثال ۱: Promise ساده

|  |
| --- |
| function fetchUserData() {  return new Promise((resolve, reject) => {  setTimeout(() => {  const success = Math.random() > 0.3;    if (success) {  resolve({ id: 1, name: "علی", email: "ali@example.com" });  } else {  reject("خطا در دریافت اطلاعات کاربر");  }  }, 2000);  });  }  // استفاده از Promise  fetchUserData()  .then(user => {  console.log("اطلاعات کاربر:", user);  })  .catch(error => {  console.error("خطا:", error);  })  .finally(() => {  console.log("عملیات تکمیل شد");  }); |

مثال ۲: زنجیره‌ای کردن Promises

|  |
| --- |
| function getUser(userId) {  return new Promise((resolve) => {  setTimeout(() => {  resolve({ id: userId, name: "علی" });  }, 1000);  });  }  function getUserPosts(userId) {  return new Promise((resolve) => {  setTimeout(() => {  resolve(["پست ۱", "پست ۲", "پست ۳"]);  }, 1000);  });  }  function getPostComments(postId) {  return new Promise((resolve) => {  setTimeout(() => {  resolve(["نظر ۱", "نظر ۲"]);  }, 1000);  });  }  // زنجیره‌ای کردن  getUser(1)  .then(user => {  console.log("کاربر:", user);  return getUserPosts(user.id);  })  .then(posts => {  console.log("پست‌ها:", posts);  return getPostComments(posts[0]);  })  .then(comments => {  console.log("نظرات:", comments);  })  .catch(error => {  console.error("خطا:", error);  }); |

مثال ۳: Promise.all - اجرای همزمان

|  |
| --- |
| function getUsers() {  return new Promise(resolve => {  setTimeout(() => resolve(["علی", "رضا", "مریم"]), 1000);  });  }  function getProducts() {  return new Promise(resolve => {  setTimeout(() => resolve(["لپ تاپ", "موبایل", "تبلت"]), 1500);  });  }  function getOrders() {  return new Promise(resolve => {  setTimeout(() => resolve(["سفارش ۱", "سفارش ۲"]), 2000);  });  }  // اجرای همزمان همه Promiseها  Promise.all([getUsers(), getProducts(), getOrders()])  .then(results => {  const [users, products, orders] = results;  console.log("کاربران:", users);  console.log("محصولات:", products);  console.log("سفارش ها:", orders);  })  .catch(error => {  console.error("خطا در یکی از عملیات:", error);  }); |

مثال ۴: Promise.race - اولین تکمیل شده

|  |
| --- |
| function fastOperation() {  return new Promise(resolve => {  setTimeout(() => resolve("عملیات سریع"), 500);  });  }  function slowOperation() {  return new Promise(resolve => {  setTimeout(() => resolve("عملیات کند"), 2000);  });  }  // اولین عملیات که تکمیل شود  Promise.race([fastOperation(), slowOperation()])  .then(result => {  console.log("نتیجه:", result); // "عملیات سریع"  }); |

مقایسه Callback و Promise

Callback:

|  |
| --- |
| // مشکلات Callback  getData(function(data1) {  processData(data1, function(data2) {  saveData(data2, function(data3) {  sendData(data3, function(result) {  console.log("نتیجه:", result);  });  });  });  }); |

Promise:

|  |
| --- |
| // راه حل با Promise  getData()  .then(processData)  .then(saveData)  .then(sendData)  .then(result => {  console.log("نتیجه:", result);  })  .catch(error => {  console.error("خطا:", error);  }); |

مزایای Promise نسبت به Callback:

* قابل خواندن تر - کد تمیز تر و خطی تر
* مدیریت خطا بهتر - استفاده از catch برای همه خطاها
* ترکیب پذیری - امکان استفاده از Promise.all و Promise.race
* اجتناب از Callback Hell - ساختار مسطح تر

مثال کاربردی: سیستم لاگین

|  |
| --- |
| // شبیه سازی عملیات ناهمگام  function checkUserExists(username) {  return new Promise((resolve, reject) => {  setTimeout(() => {  const users = ["ali", "reza", "maryam"];  if (users.includes(username)) {  resolve(true);  } else {  reject("کاربر یافت نشد");  }  }, 1000);  });  }  function verifyPassword(username, password) {  return new Promise((resolve, reject) => {  setTimeout(() => {  if (password === "123456") {  resolve({ username, token: "abc123" });  } else {  reject("رمز عبور نادرست");  }  }, 1000);  });  }  function getUserProfile(token) {  return new Promise((resolve) => {  setTimeout(() => {  resolve({  name: "علی محمدی",  email: "ali@example.com",  role: "user"  });  }, 1000);  });  }  // فرآیند لاگین  function login(username, password) {  console.log("شروع فرآیند لاگین...");    checkUserExists(username)  .then(() => verifyPassword(username, password))  .then(userData => getUserProfile(userData.token))  .then(profile => {  console.log("لاگین موفق");  console.log("پروفایل کاربر:", profile);  })  .catch(error => {  console.error("خطا در لاگین:", error);  })  .finally(() => {  console.log("فرآیند لاگین تکمیل شد");  });  }  // تست  login("ali", "123456");  // login("unknown", "123456"); // تست خطا |

نکات مهم:

* از Sync برای عملیات سریع استفاده کنید
* از Async برای عملیات I/O و شبکه استفاده کنید
* Promiseها جایگزین بهتری برای Callback هستند
* همیشه خطاها را با catch مدیریت کنید
* برای کدهای مدرن از async/await استفاده کنید

## **استفاده از await در async**

مفهوم async/await:

یک روش مدرن و خوانا برای کار با کدهای ناهمگام که شبیه کدهای همگام به نظر می‌رسد.

ساختار پایه:

|  |
| --- |
| async function myFunction() {  const result = await someAsyncOperation();  return result;  } |

مثال‌های async/await:

مثال ۱: تبدیل Promise به async/await

|  |
| --- |
| // قبل (با Promise)  function fetchData() {  return fetch('https://api.example.com/data')  .then(response => response.json())  .then(data => console.log(data))  .catch(error => console.error(error));  }  // بعد (با async/await)  async function fetchData() {  try {  const response = await fetch('https://api.example.com/data');  const data = await response.json();  console.log(data);  } catch (error) {  console.error(error);  }  } |

مثال ۲: چندین عملیات ناهمگام

|  |
| --- |
| async function getUserData(userId) {  try {  console.log("دریافت اطلاعات کاربر...");  const user = await fetchUser(userId);    console.log("دریافت پست‌های کاربر...");  const posts = await fetchUserPosts(userId);    console.log("دریافت نظرات...");  const comments = await fetchComments(posts[0].id);    return {  user,  posts,  comments  };  } catch (error) {  console.error("خطا در دریافت داده:", error);  }  }  // توابع شبیه‌سازی شده  function fetchUser(id) {  return new Promise(resolve => {  setTimeout(() => resolve({ id, name: "علی", email: "ali@example.com" }), 1000);  });  }  function fetchUserPosts(userId) {  return new Promise(resolve => {  setTimeout(() => resolve([{ id: 1, title: "پست اول" }, { id: 2, title: "پست دوم" }]), 1000);  });  }  function fetchComments(postId) {  return new Promise(resolve => {  setTimeout(() => resolve(["نظر عالی", "ممنون از پست"]), 1000);  });  }  // استفاده  getUserData(1).then(result => console.log(result)); |

مثال ۳: اجرای همزمان با Promise.all

|  |
| --- |
| async function fetchAllData() {  try {  const [users, products, orders] = await Promise.all([  fetch('/api/users'),  fetch('/api/products'),  fetch('/api/orders')  ]);    const usersData = await users.json();  const productsData = await products.json();  const ordersData = await orders.json();    return {  users: usersData,  products: productsData,  orders: ordersData  };  } catch (error) {  console.error("خطا در دریافت داده‌ها:", error);  }  } |

مثال ۴: حلقه با async/await

|  |
| --- |
| async function processUsers(userIds) {  const results = [];    for (const userId of userIds) {  try {  const user = await fetchUser(userId);  results.push(user);  console.log(`کاربر ${userId} پردازش شد`);  } catch (error) {  console.error(`خطا در پردازش کاربر ${userId}:`, error);  }  }    return results;  }  // استفاده  const userIds = [1, 2, 3, 4, 5];  processUsers(userIds).then(users => console.log(users)); |

## **معرفی ajax و api**

AJAX (Asynchronous JavaScript And XML):

تکنیکی برای ارسال و دریافت داده از سرور بدون نیاز به رفرش صفحه.

مثال AJAX با Fetch API:

|  |
| --- |
| // ارسال درخواست GET  async function fetchUsers() {  try {  const response = await fetch('https://jsonplaceholder.typicode.com/users');    if (!response.ok) {  throw new Error('خطا در دریافت داده');  }    const users = await response.json();  displayUsers(users);  } catch (error) {  console.error('خطا:', error);  }  }  function displayUsers(users) {  const container = document.getElementById('users-container');  container.innerHTML = users.map(user => `  <div class="user-card">  <h3>${user.name}</h3>  <p>ایمیل: ${user.email}</p>  <p>شهر: ${user.address.city}</p>  </div>  `).join('');  } |

API (Application Programming Interface):

رابطی که امکان ارتباط بین نرم‌افزارهای مختلف را فراهم می‌کند.

انواع API:

|  |
| --- |
| // مثال‌هایی از API‌های مختلف  const apis = {  // API عمومی  weather: 'https://api.openweathermap.org/data/2.5/weather',    // API رایگان برای تست  jsonPlaceholder: 'https://jsonplaceholder.typicode.com',    // API داخلی  localApi: '/api/users'  }; |

## **بررسی تفاوت soap و Rest api**

بررسی تفاوت SOAP و REST API

REST API:

|  |
| --- |
| // مثال درخواست REST  async function restExample() {  // GET - دریافت داده  const response = await fetch('https://api.example.com/users/1');  const user = await response.json();    // POST - ایجاد داده جدید  await fetch('https://api.example.com/users', {  method: 'POST',  headers: {  'Content-Type': 'application/json',  },  body: JSON.stringify({  name: 'علی',  email: 'ali@example.com'  })  });    // PUT - آپدیت کامل  await fetch('https://api.example.com/users/1', {  method: 'PUT',  headers: {  'Content-Type': 'application/json',  },  body: JSON.stringify({  name: 'علی محمدی',  email: 'ali@example.com'  })  });    // DELETE - حذف داده  await fetch('https://api.example.com/users/1', {  method: 'DELETE'  });  } |

SOAP API:

|  |
| --- |
| // مثال درخواست SOAP (پیچیده‌تر)  const soapRequest = `  <?xml version="1.0" encoding="UTF-8"?>  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Body>  <GetUser xmlns="http://example.com/webservice">  <UserID>1</UserID>  </GetUser>  </soap:Body>  </soap:Envelope>  `;  async function soapExample() {  const response = await fetch('https://api.example.com/soap', {  method: 'POST',  headers: {  'Content-Type': 'text/xml',  'SOAPAction': 'GetUser'  },  body: soapRequest  });    const xmlData = await response.text();  // پردازش XML پاسخ...  } |

مقایسه SOAP و REST:

|  |  |  |
| --- | --- | --- |
| ویژگی | REST API | SOAP API |
| قالب داده | JSON, XML, HTML | فقط XML |
| ساختار | ساده، مبتنی بر HTTP | پیچیده، مبتنی بر XML |
| عملکرد | سریع‌تر | کندتر |
| امنیت | SSL/TLS | WS-Security |
| یادگیری | آسان | پیچیده |
| انعطاف‌پذیری | بالا | پایین |

## **معرفی get در api**

درخواست GET ساده:

|  |
| --- |
| // دریافت لیست کاربران  async function getUsers() {  try {  const response = await fetch('https://jsonplaceholder.typicode.com/users');    if (!response.ok) {  throw new Error(`خطا: ${response.status}`);  }    const users = await response.json();  return users;  } catch (error) {  console.error('خطا در دریافت کاربران:', error);  return [];  }  } |

GET با پارامترهای Query:

|  |
| --- |
| // جستجوی کاربران  async function searchUsers(query, limit = 10) {  try {  // ساخت URL با پارامترهای query  const url = new URL('https://jsonplaceholder.typicode.com/users');  url.searchParams.append('q', query);  url.searchParams.append('\_limit', limit);    const response = await fetch(url);  const users = await response.json();  return users;  } catch (error) {  console.error('خطا در جستجو:', error);  return [];  }  }  // استفاده  searchUsers('ali', 5).then(users => console.log(users)); |

GET با هدرهای سفارشی:

|  |
| --- |
| // درخواست با احراز هویت  async function getProtectedData() {  try {  const token = 'your-auth-token-here';    const response = await fetch('https://api.example.com/protected-data', {  method: 'GET',  headers: {  'Authorization': `Bearer ${token}`,  'Content-Type': 'application/json',  'X-API-Key': 'your-api-key'  }  });    if (response.status === 401) {  throw new Error('احراز هویت ناموفق');  }    const data = await response.json();  return data;  } catch (error) {  console.error('خطا:', error);  return null;  }  } |

مثال کاربردی کامل:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>مدیریت کاربران</title>  <style>  .user-card {  border: 1px solid #ddd;  padding: 15px;  margin: 10px;  border-radius: 8px;  }  .loading {  color: blue;  }  .error {  color: red;  }  </style>  </head>  <body>  <h1>سیستم مدیریت کاربران</h1>    <button onclick="loadUsers()">بارگذاری کاربران</button>  <button onclick="loadUser(1)">دریافت کاربر ۱</button>    <div id="loading" class="loading" style="display: none;">  در حال بارگذاری...  </div>    <div id="error" class="error" style="display: none;"></div>    <div id="users-container"></div>  <script>  // نمایش loading  function showLoading() {  document.getElementById('loading').style.display = 'block';  document.getElementById('error').style.display = 'none';  }    // مخفی کردن loading  function hideLoading() {  document.getElementById('loading').style.display = 'none';  }    // نمایش خطا  function showError(message) {  document.getElementById('error').style.display = 'block';  document.getElementById('error').textContent = message;  }    // دریافت همه کاربران  async function loadUsers() {  showLoading();    try {  const response = await fetch('https://jsonplaceholder.typicode.com/users');    if (!response.ok) {  throw new Error(`خطای سرور: ${response.status}`);  }    const users = await response.json();  displayUsers(users);    } catch (error) {  showError(`خطا در دریافت کاربران: ${error.message}`);  } finally {  hideLoading();  }  }    // دریافت کاربر خاص  async function loadUser(userId) {  showLoading();    try {  const response = await fetch(`https://jsonplaceholder.typicode.com/users/${userId}`);    if (!response.ok) {  throw new Error(`خطای سرور: ${response.status}`);  }    const user = await response.json();  displayUser(user);    } catch (error) {  showError(`خطا در دریافت کاربر: ${error.message}`);  } finally {  hideLoading();  }  }    // نمایش لیست کاربران  function displayUsers(users) {  const container = document.getElementById('users-container');  container.innerHTML = users.map(user => `  <div class="user-card">  <h3>${user.name}</h3>  <p><strong>ایمیل:</strong> ${user.email}</p>  <p><strong>تلفن:</strong> ${user.phone}</p>  <p><strong>شرکت:</strong> ${user.company.name}</p>  </div>  `).join('');  }    // نمایش کاربر تکی  function displayUser(user) {  const container = document.getElementById('users-container');  container.innerHTML = `  <div class="user-card">  <h3>${user.name}</h3>  <p><strong>نام کاربری:</strong> ${user.username}</p>  <p><strong>ایمیل:</strong> ${user.email}</p>  <p><strong>آدرس:</strong> ${user.address.street}, ${user.address.city}</p>  <p><strong>تلفن:</strong> ${user.phone}</p>  <p><strong>وبسایت:</strong> ${user.website}</p>  <p><strong>شرکت:</strong> ${user.company.name}</p>  </div>  `;  }  </script>  </body>  </html> |

نکات مهم GET API:

* ایمن - داده‌ها را تغییر نمی‌دهد
* قابل کش - پاسخ‌ها قابل ذخیره هستند
* قابل بوکمارک - URLها قابل ذخیره‌سازی هستند
* محدودیت طول - پارامترها در URL محدودیت طول دارند
* استفاده برای - دریافت داده، جستجو، فیلتر کردن

## **XmlHttpRequest**

معرفی:

قدیمی‌ترین روش برای ارسال درخواست‌های HTTP در JavaScript

مثال XHR:

|  |
| --- |
| // ایجاد درخواست GET با XHR  function getUsersXHR() {  return new Promise((resolve, reject) => {  const xhr = new XMLHttpRequest();    xhr.open('GET', 'https://jsonplaceholder.typicode.com/users', true);    xhr.onload = function() {  if (xhr.status >= 200 && xhr.status < 300) {  const users = JSON.parse(xhr.responseText);  resolve(users);  } else {  reject(new Error(`خطا: ${xhr.status}`));  }  };    xhr.onerror = function() {  reject(new Error('خطای شبکه'));  };    xhr.send();  });  }  // استفاده  getUsersXHR()  .then(users => console.log('کاربران:', users))  .catch(error => console.error('خطا:', error)); |

درخواست POST با XHR:

|  |
| --- |
| function createUserXHR(userData) {  return new Promise((resolve, reject) => {  const xhr = new XMLHttpRequest();    xhr.open('POST', 'https://jsonplaceholder.typicode.com/users', true);  xhr.setRequestHeader('Content-Type', 'application/json');    xhr.onload = function() {  if (xhr.status >= 200 && xhr.status < 300) {  const response = JSON.parse(xhr.responseText);  resolve(response);  } else {  reject(new Error(`خطا: ${xhr.status}`));  }  };    xhr.onerror = function() {  reject(new Error('خطای شبکه'));  };    xhr.send(JSON.stringify(userData));  });  }  // استفاده  createUserXHR({  name: 'علی',  email: 'ali@example.com'  }).then(user => console.log('کاربر ایجاد شد:', user)); |

## **Promise**

معرفی:

یک شیء که نشان‌دهنده تکمیل یا شکست یک عملیات ناهمگام است

ایجاد Promise:

|  |
| --- |
| // ایجاد Promise سفارشی  function delay(ms) {  return new Promise((resolve, reject) => {  setTimeout(() => {  if (ms > 0) {  resolve(`عملیات بعد از ${ms} میلی‌ثانیه تکمیل شد`);  } else {  reject('زمان نمی‌تواند منفی باشد');  }  }, ms);  });  }  // استفاده  delay(2000)  .then(message => console.log(message))  .catch(error => console.error(error)); |

ترکیب Promise‌ها:

|  |
| --- |
| // Promise.all - اجرای همزمان  function fetchMultipleData() {  const promise1 = fetch('https://jsonplaceholder.typicode.com/users/1');  const promise2 = fetch('https://jsonplaceholder.typicode.com/posts/1');  const promise3 = fetch('https://jsonplaceholder.typicode.com/comments/1');    return Promise.all([promise1, promise2, promise3])  .then(responses => {  return Promise.all(responses.map(response => response.json()));  })  .then(([user, post, comment]) => {  return { user, post, comment };  });  }  // Promise.race - اولین پاسخ  function fetchWithTimeout(url, timeout = 5000) {  const fetchPromise = fetch(url);  const timeoutPromise = new Promise((\_, reject) => {  setTimeout(() => reject(new Error('زمان درخواست به پایان رسید')), timeout);  });    return Promise.race([fetchPromise, timeoutPromise]);  } |

## **Fetch**

معرفی:

مدرن‌ترین روش built-in برای کار با درخواست‌های HTTP

درخواست GET با Fetch:

|  |
| --- |
| // دریافت داده  async function getUsersFetch() {  try {  const response = await fetch('https://jsonplaceholder.typicode.com/users');    if (!response.ok) {  throw new Error(`خطای HTTP: ${response.status}`);  }    const users = await response.json();  return users;  } catch (error) {  console.error('خطا در دریافت داده:', error);  throw error;  }  } |

درخواست POST با Fetch:

|  |
| --- |
| // ارسال داده  async function createUserFetch(userData) {  try {  const response = await fetch('https://jsonplaceholder.typicode.com/users', {  method: 'POST',  headers: {  'Content-Type': 'application/json',  'Authorization': 'Bearer your-token-here'  },  body: JSON.stringify(userData)  });    if (!response.ok) {  throw new Error(`خطای HTTP: ${response.status}`);  }    const newUser = await response.json();  return newUser;  } catch (error) {  console.error('خطا در ایجاد کاربر:', error);  throw error;  }  } |

مدیریت Upload/Download:

|  |
| --- |
| // آپلود فایل  async function uploadFile(file) {  const formData = new FormData();  formData.append('file', file);  formData.append('userId', '123');    try {  const response = await fetch('/api/upload', {  method: 'POST',  body: formData  // headers نباید تنظیم شود - مرورگر خودش تنظیم می‌کند  });    const result = await response.json();  return result;  } catch (error) {  console.error('خطا در آپلود:', error);  throw error;  }  }  // دانلود فایل  async function downloadFile(url, filename) {  try {  const response = await fetch(url);  const blob = await response.blob();    // ایجاد لینک دانلود  const downloadUrl = window.URL.createObjectURL(blob);  const link = document.createElement('a');  link.href = downloadUrl;  link.download = filename;  link.click();    // آزاد کردن memory  window.URL.revokeObjectURL(downloadUrl);  } catch (error) {  console.error('خطا در دانلود:', error);  throw error;  }  } |

## **Axios**

معرفی:

یک کتابخانه محبوب برای ارسال درخواست‌های HTTP

نصب و راه‌اندازی:

|  |
| --- |
| <!-- استفاده از CDN -->  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script> |

|  |
| --- |
| // یا با npm  // npm install axios |

درخواست GET با Axios:

|  |
| --- |
| // دریافت داده  async function getUsersAxios() {  try {  const response = await axios.get('https://jsonplaceholder.typicode.com/users');    // داده به صورت خودکار parse می‌شود  return response.data;  } catch (error) {  console.error('خطا در دریافت داده:', error);  throw error;  }  } |

درخواست POST با Axios:

|  |
| --- |
| // ارسال داده  async function createUserAxios(userData) {  try {  const response = await axios.post(  'https://jsonplaceholder.typicode.com/users',  userData,  {  headers: {  'Content-Type': 'application/json',  'Authorization': 'Bearer your-token-here'  },  timeout: 10000 // 10 ثانیه  }  );    return response.data;  } catch (error) {  if (error.response) {  // سرور پاسخ با status code خارج از 2xx داده  console.error('خطای سرور:', error.response.status);  } else if (error.request) {  // درخواست ارسال شده اما پاسخی دریافت نشده  console.error('خطای شبکه:', error.request);  } else {  // خطای دیگر  console.error('خطا:', error.message);  }  throw error;  }  } |

ویژگی‌های پیشرفته Axios:

|  |
| --- |
| // ایجاد instance با تنظیمات پیشفرض  const apiClient = axios.create({  baseURL: 'https://api.example.com',  timeout: 10000,  headers: {  'Content-Type': 'application/json'  }  });  // اضافه کردن interceptor برای درخواست‌ها  apiClient.interceptors.request.use(  config => {  // اضافه کردن token به همه درخواست‌ها  const token = localStorage.getItem('authToken');  if (token) {  config.headers.Authorization = `Bearer ${token}`;  }  return config;  },  error => {  return Promise.reject(error);  }  );  // اضافه کردن interceptor برای پاسخ‌ها  apiClient.interceptors.response.use(  response => {  return response;  },  error => {  if (error.response?.status === 401) {  // redirect به صفحه لاگین  window.location.href = '/login';  }  return Promise.reject(error);  }  );  // استفاده از instance  async function getProtectedData() {  try {  const response = await apiClient.get('/protected-data');  return response.data;  } catch (error) {  console.error('خطا:', error);  throw error;  }  } |

📊 مقایسه کامل

کد مقایسه‌ای برای درخواست GET:

XmlHttpRequest:

|  |
| --- |
| function getWithXHR(url) {  return new Promise((resolve, reject) => {  const xhr = new XMLHttpRequest();  xhr.open('GET', url);  xhr.onload = () => resolve(JSON.parse(xhr.responseText));  xhr.onerror = () => reject(new Error('خطای شبکه'));  xhr.send();  });  } |

Fetch:

|  |
| --- |
| async function getWithFetch(url) {  const response = await fetch(url);  if (!response.ok) throw new Error('خطای HTTP');  return response.json();  } |

Axios

|  |
| --- |
| async function getWithAxios(url) {  const response = await axios.get(url);  return response.data;  } |

جدول مقایسه:

|  |  |  |  |
| --- | --- | --- | --- |
| ویژگی | XmlHttpRequest | Fetch | Axios |
| ساختار | قدیمی، مبتنی onload | مدرن، مبتنی بر Promise | کتابخانه، مبتنی بر Promise |
| پشتیبانی مرورگر | همه مرورگرها | مرورگرهای مدرن | همه مرورگرها (با polyfill) |
| JSON تبدیل خودکار | ❌ | ❌ | ✅ |
| Cancel درخواست | ✅ | ✅ (با AbortController) | ✅ |
| Upload progress | ✅ | ❌ | ✅ |
| Timeout خودکار | ❌ | ❌ | ✅ |
| Interceptors | ❌ | ❌ | ✅ |
| سادگی استفاده | ❌ | ✅ | ✅✅ |

مثال کاربردی کامل

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>مقایسه روش‌های HTTP</title>  <style>  .container { margin: 20px; padding: 20px; border: 1px solid #ddd; }  .method { margin: 10px 0; padding: 10px; background: #f5f5f5; }  button { margin: 5px; padding: 8px 15px; cursor: pointer; }  .result { margin: 10px 0; padding: 10px; background: white; }  </style>  </head>  <body>  <div class="container">  <h1>مقایسه روش‌های ارسال درخواست HTTP</h1>    <div class="method">  <h3>XmlHttpRequest</h3>  <button onclick="testXHR()">تست XHR</button>  <div id="xhr-result" class="result"></div>  </div>    <div class="method">  <h3>Fetch API</h3>  <button onclick="testFetch()">تست Fetch</button>  <div id="fetch-result" class="result"></div>  </div>    <div class="method">  <h3>Axios</h3>  <button onclick="testAxios()">تست Axios</button>  <div id="axios-result" class="result"></div>  </div>  </div>  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>  <script>  const API\_URL = 'https://jsonplaceholder.typicode.com/users/1';    // XmlHttpRequest  async function testXHR() {  const resultDiv = document.getElementById('xhr-result');  resultDiv.innerHTML = 'در حال بارگذاری...';    try {  const user = await getWithXHR(API\_URL);  resultDiv.innerHTML = `<pre>${JSON.stringify(user, null, 2)}</pre>`;  } catch (error) {  resultDiv.innerHTML = `خطا: ${error.message}`;  }  }    function getWithXHR(url) {  return new Promise((resolve, reject) => {  const xhr = new XMLHttpRequest();  xhr.open('GET', url);  xhr.onload = function() {  if (xhr.status === 200) {  resolve(JSON.parse(xhr.responseText));  } else {  reject(new Error(`خطا: ${xhr.status}`));  }  };  xhr.onerror = function() {  reject(new Error('خطای شبکه'));  };  xhr.send();  });  }    // Fetch API  async function testFetch() {  const resultDiv = document.getElementById('fetch-result');  resultDiv.innerHTML = 'در حال بارگذاری...';    try {  const user = await getWithFetch(API\_URL);  resultDiv.innerHTML = `<pre>${JSON.stringify(user, null, 2)}</pre>`;  } catch (error) {  resultDiv.innerHTML = `خطا: ${error.message}`;  }  }    async function getWithFetch(url) {  const response = await fetch(url);  if (!response.ok) {  throw new Error(`خطای HTTP: ${response.status}`);  }  return response.json();  }    // Axios  async function testAxios() {  const resultDiv = document.getElementById('axios-result');  resultDiv.innerHTML = 'در حال بارگذاری...';    try {  const user = await getWithAxios(API\_URL);  resultDiv.innerHTML = `<pre>${JSON.stringify(user, null, 2)}</pre>`;  } catch (error) {  resultDiv.innerHTML = `خطا: ${error.message}`;  }  }    async function getWithAxios(url) {  const response = await axios.get(url);  return response.data;  }  </script>  </body>  </html> |

نکات انتخاب روش مناسب:

* برای پروژه‌های ساده: از Fetch استفاده کنید
* برای پروژه‌های پیچیده: از Axios استفاده کنید
* برای پشتیبانی از مرورگرهای قدیمی: از XHR استفاده کنید
* برای کنترل پیشرفته: از XHR یا Axios استفاده کنید

## **معرفی ecmascript**

چیست ECMAScript؟

ECMAScript استاندارد رسمی زبان JavaScript است. هر سال نسخه‌های جدیدی با ویژگی‌های تازه منتشر می‌شود.

نسخه‌های مهم:

ES5 (2009): ویژگی‌های پایه

ES6/ES2015: کلاس‌ها، ماژول‌ها، arrow functions

ES2016-ES2023: ویژگی‌های مدرن

بر اساس نتایج جستجو، ECMAScript 2025 (ES2025) در ژوئن 2025 به عنوان یک استاندارد رسمی منتشر شد و ویژگی‌های جدید و کاربردی را به جاوااسکریپت اضافه کرده است . در ادامه، این ویژگی‌ها را به همراه مثال‌های ساده بررسی می‌کنیم.

جدول زیر خلاصه‌ای از مهم‌ترین ویژگی‌های جدید در ES2025 را نشان می‌دهد:

|  |  |  |
| --- | --- | --- |
| ویژگی | توضیح | مثال ساده |
| متدهای جدید Iterator | متدهایی برای پردازش آرایه‌ها و داده‌ها بدون ایجاد آرایه‌های میانی | Iterator.from([1,2,3]).filter(x => x>1).toArray() |
| متدهای جدید Set | متدهایی برای انجام عملیات ریاضی روی مجموعه‌ها | setA.union(setB) |
| ایمپورت ماژول‌های JSON | امکان ایمپورت مستقیم فایل‌های JSON | import data from './data.json' with { type: 'json' }; |
| RegExp.escape() | متدی برای فرار کردن کاراکترهای خاص در عبارات منظم | RegExp.escape("1+1") |
| Promise.try() | متدی برای یکسان‌سازی مدیریت خطا در توابع همگام و ناهمگام | Promise.try(() => func()).catch(...) |

Web API چیست؟

یک Web API یا رابط برنامه‌نویسی کاربردی تحت وب، مجموعه‌ای از ابزارها و پروتکل‌هاست که به برنامه‌های مختلف اجازه می‌دهد از طریق پروتکل استاندارد HTTP با یکدیگر ارتباط برقرار کنند. این مفهوم یک فناوری خاص نیست و می‌توان آن را با تکنولوژی‌های مختلفی مانند جاوا یا دات‌نت پیاده‌سازی کرد.

معرفی insertAdjacentHTML

متد insertAdjacentHTML یکی از APIهای مفید مرورگر برای درج کارآمد محتوای HTML در صفحه وب است.

کاربرد و عملکرد: این متد یک رشته حاوی کد HTML را تجزیه کرده و گره‌های حاصل را در موقعیت مشخصی نسبت به یک عنصر در درخت DOM درج می‌کند. برخلاف innerHTML که کل محتوای داخلی یک عنصر را جایگزین می‌کند، insertAdjacentHTML محتوای موجود را خراب نمی‌کند و سریع‌تر است.

نحوه استفاده: این متد دو پارامتر می‌گیرد: موقعیت قرارگیری و رشته HTML.

موقعیت‌های قابل استفاده:

* beforebegin: قبل از خود عنصر
* afterbegin: درون عنصر، قبل از اولین فرزند
* beforeend: درون عنصر، بعد از آخرین فرزند
* afterend: بعد از خود عنصر

|  |
| --- |
| // مثال: درج یک پاراگراف جدید بعد از یک عنصر هدر  const headerElement = document.getElementById("myHeader");  const newHTML = '<p class="info">این یک پاراگراف جدید است.</p>';  headerElement.insertAdjacentHTML('afterend', newHTML); |

دیگر Web APIهای مهم و کاربردی

جدول زیر برخی دیگر از APIهای مرورگر که در توسعه فرانت‌اند بسیار پرکاربرد هستند را نشان می‌دهد:

|  |  |
| --- | --- |
| API | کاربرد اصلی |
| Fetch API | ارسال درخواست‌های شبکه (HTTP) و دریافت داده از سرور |
| Canvas API | ترسیم گرافیک و اشکال پویا، دستکاری و ویرایش تصاویر |
| Geolocation API | دریافت موقعیت جغرافیایی کاربر |
| Web Storage API | ذخیره‌سازی داده در سمت کلاینت (localStorage, sessionStorage) |
| Intersection Observer API | تشخیص زمانی که یک عنصر در viewport کاربر ظاهر می‌شود |
| History API | دستکاری تاریخچه مرورگر و ساخت اپلیکیشن‌های تک صفحه‌ای (SPA) |
| Drag and Drop API | پیاده‌سازی قابلیت کشیدن و رها کردن (Drag & Drop) عناصر |

روندهای آینده Web APIها

در سال 2025، توسعه Web APIها به سمت معماری‌های سرورلس (Serverless) و مدیریت هوشمند پیش می‌رود. همچنین امنیت با پروتکل‌های احراز هویت پیشرفته‌تری مانند "Zero-trust" بیش از گذشته اهمیت یافته است.

## **نحوه array destructing**

مفهوم:

استخراج مقادیر از آرایه و قرار دادن در متغیرهای جداگانه

مثال‌های پایه:

|  |
| --- |
| // آرایه ساده  const numbers = [1, 2, 3, 4, 5];  // Destructuring  const [first, second, third] = numbers;  console.log(first); // 1  console.log(second); // 2  console.log(third); // 3 |

موارد استفاده پیشرفته:

۱. رد کردن عناصر:

|  |
| --- |
| const colors = ['red', 'green', 'blue', 'yellow'];  // رد کردن عناصر با کاما  const [first, , third] = colors;  console.log(first); // 'red'  console.log(third); // 'blue' |

مقدار پیشفرض:

|  |
| --- |
| const fruits = ['apple', 'banana'];  // مقدار پیشفرض برای موقعیت‌های خالی  const [fruit1, fruit2, fruit3 = 'orange'] = fruits;  console.log(fruit1); // 'apple'  console.log(fruit2); // 'banana'  console.log(fruit3); // 'orange' (پیشفرض) |

باقی‌مانده آرایه:

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5];  // گرفتن باقی‌مانده آرایه  const [first, second, ...rest] = numbers;  console.log(first); // 1  console.log(second); // 2  console.log(rest); // [3, 4, 5] |

جابجایی مقادیر:

|  |
| --- |
| let a = 10;  let b = 20;  // جابجایی بدون متغیر موقت  [a, b] = [b, a];  console.log(a); // 20  console.log(b); // 10 |

با توابع:

|  |
| --- |
| // تابعی که آرایه برمی‌گرداند  function getCoordinates() {  return [40.7128, -74.0060];  }  // Destructuring نتیجه تابع  const [latitude, longitude] = getCoordinates();  console.log(latitude); // 40.7128  console.log(longitude); // -74.0060 |

## **نحوه object destructing**

مفهوم:

استخراج ویژگی‌ها از آبجکت و قرار دادن در متغیرهای جداگانه

مثال‌های پایه:

|  |
| --- |
| // آبجکت کاربر  const user = {  name: 'علی',  age: 25,  email: 'ali@example.com',  address: {  city: 'تهران',  street: 'ولیعصر'  }  };  // Destructuring  const { name, age, email } = user;  console.log(name); // 'علی'  console.log(age); // 25  console.log(email); // 'ali@example.com' |

موارد استفاده پیشرفته:

۱. نام‌گذاری مجدد:

|  |
| --- |
| const person = {  firstName: 'علی',  lastName: 'محمدی'  };  // نام‌گذاری مجدد متغیرها  const { firstName: fname, lastName: lname } = person;  console.log(fname); // 'علی'  console.log(lname); // 'محمدی' |

مقدار پیشفرض:

|  |
| --- |
| const product = {  name: 'لپ‌تاپ',  price: 1000  };  // مقدار پیشفرض برای ویژگی‌های موجود نبوده  const { name, price, category = 'الکترونیک' } = product;  console.log(name); // 'لپ‌تاپ'  console.log(price); // 1000  console.log(category); // 'الکترونیک' (پیشفرض) |

باقی‌مانده ویژگی‌ها:

|  |
| --- |
| const student = {  name: 'رضا',  age: 20,  major: 'کامپیوتر',  grade: 18.5  };  // گرفتن باقی‌مانده ویژگی‌ها  const { name, ...otherInfo } = student;  console.log(name); // 'رضا'  console.log(otherInfo); // { age: 20, major: 'کامپیوتر', grade: 18.5 } |

Destructuring تودرتو:

|  |
| --- |
| const company = {  name: 'شرکت نمونه',  CEO: {  name: 'محمد',  age: 45  },  employees: ['علی', 'رضا', 'مریم']  };  // Destructuring تودرتو  const {  name,  CEO: { name: ceoName, age: ceoAge },  employees: [firstEmployee]  } = company;  console.log(name); // 'شرکت نمونه'  console.log(ceoName); // 'محمد'  console.log(ceoAge); // 45  console.log(firstEmployee); // 'علی' |

در پارامترهای تابع:

|  |
| --- |
| // بدون destructuring  function displayUser(user) {  console.log(`نام: ${user.name}, سن: ${user.age}`);  }  // با destructuring  function displayUser({ name, age, email = 'ندارد' }) {  console.log(`نام: ${name}, سن: ${age}, ایمیل: ${email}`);  }  // استفاده  const user = { name: 'علی', age: 25 };  displayUser(user); // نام: علی, سن: 25, ایمیل: ندارد |

انواع Export:

۱. Named Export (صادر کردن نام‌دار):

|  |
| --- |
| // math.js  export const PI = 3.14159;  export function add(a, b) {  return a + b;  }  export function multiply(a, b) {  return a \* b;  }  // یا در انتهای فایل  const PI = 3.14159;  function add(a, b) { return a + b; }  function multiply(a, b) { return a \* b; }  export { PI, add, multiply }; |

Default Export (صادر کردن پیشفرض):

|  |
| --- |
| // userService.js  const API\_URL = 'https://api.example.com';  export default class UserService {  static getUser(id) {  return fetch(`${API\_URL}/users/${id}`)  .then(response => response.json());  }    static createUser(userData) {  return fetch(`${API\_URL}/users`, {  method: 'POST',  body: JSON.stringify(userData)  }).then(response => response.json());  }  }  // یا برای یک تابع ساده  export default function logger(message) {  console.log(`[LOG]: ${message}`);  } |

Mixed Export (ترکیبی):

|  |
| --- |
| // utils.js  export const VERSION = '1.0.0';  export function formatDate(date) {  return date.toLocaleDateString('fa-IR');  }  export default function helper() {  console.log('Helper function');  } |

## **نحوه import و export**

ماژول چیست؟

ماژول‌ها فایل‌های جداگانه JavaScript هستند که می‌توانند کدهای مربوطه را در خود نگهداری و export کنند.

مزایای استفاده از ماژول‌ها:

* سازماندهی بهتر کد
* قابلیت استفاده مجدد
* اجتناب از تداخل نام‌ها
* بارگذاری بهینه

انواع Export

1. Named Export (صادر کردن نام‌دار)

صادر کردن چندین مقدار از یک ماژول با نام‌های مشخص

روش اول: صادر کردن مستقیم

|  |
| --- |
| // math.js  export const PI = 3.14159;  export function add(a, b) {  return a + b;  }  export function multiply(a, b) {  return a \* b;  }  export class Calculator {  constructor() {  this.result = 0;  }    add(x) {  this.result += x;  return this;  }  } |

روش دوم: صادر کردن در انتهای فایل

|  |
| --- |
| // utils.js  const VERSION = '1.0.0';  function formatDate(date) {  return date.toLocaleDateString('fa-IR');  }  function logger(message) {  console.log(`[LOG]: ${message}`);  }  // صادر کردن همه در یکجا  export { VERSION, formatDate, logger }; |

روش سوم: صادر کردن با نام‌گذاری مجدد

|  |
| --- |
| // api.js  function fetchUsers() {  return fetch('/api/users');  }  function createUser(userData) {  return fetch('/api/users', {  method: 'POST',  body: JSON.stringify(userData)  });  }  // صادر کردن با نام‌های متفاوت  export {  fetchUsers as getUsers,  createUser as postUser  }; |

Default Export (صادر کردن پیشفرض)

صادر کردن یک مقدار اصلی از ماژول

صادر کردن تابع به عنوان پیشفرض

|  |
| --- |
| // logger.js  export default function(message) {  console.log(`[APP]: ${message}`);  }  // یا به این صورت:  // function log(message) {  // console.log(`[APP]: ${message}`);  // }  // export default log; |

صادر کردن کلاس به عنوان پیشفرض

|  |
| --- |
| // User.js  export default class User {  constructor(name, email) {  this.name = name;  this.email = email;  }    getInfo() {  return `${this.name} (${this.email})`;  }  } |

صادر کردن آبجکت به عنوان پیشفرض

|  |
| --- |
| // config.js  const appConfig = {  apiUrl: 'https://api.example.com',  version: '1.0.0',  timeout: 5000  };  export default appConfig; |

Mixed Export (صادر کردن ترکیبی)

ترکیب صادرات پیشفرض و نام‌دار در یک ماژول

|  |
| --- |
| // auth.js  // صادرات نام‌دار  export const API\_KEY = 'abc123';  export const TOKEN\_NAME = 'auth\_token';  // صادرات پیشفرض  export default function authenticate(user, pass) {  return user === 'admin' && pass === '1234';  }  // صادرات نام‌دار اضافی  export function logout() {  localStorage.removeItem(TOKEN\_NAME);  } |

انواع Import

1. Import Named (وارد کردن نام‌دار)

وارد کردن مقادیر خاص

|  |
| --- |
| // وارد کردن مقادیر مشخص  import { PI, add } from './math.js';  console.log(PI); // 3.14159  console.log(add(2, 3)); // 5 |

وارد کردن با نام‌گذاری مجدد

|  |
| --- |
| // وارد کردن با نام جدید  import { multiply as mul, Calculator as Calc } from './math.js';  console.log(mul(2, 3)); // 6  const calc = new Calc(); |

وارد کردن همه صادرات نام‌دار

|  |
| --- |
| // وارد کردن همه صادرات نام‌دار به عنوان یک آبجکت  import \* as MathUtils from './math.js';  console.log(MathUtils.PI); // 3.14159  console.log(MathUtils.add(5, 3)); // 8 |

Import Default (وارد کردن پیشفرض)

|  |
| --- |
| // وارد کردن صادرات پیشفرض  import User from './User.js';  import logger from './logger.js';  import appConfig from './config.js';  const user = new User('علی', 'ali@example.com');  logger('برنامه شروع شد');  console.log(appConfig.apiUrl); |

Import ترکیبی

ترکیب صادرات پیشفرض و نام‌دار

|  |
| --- |
| // وارد کردن همزمان پیشفرض و نام‌دار  import authenticate, { API\_KEY, logout } from './auth.js';  const isAuthenticated = authenticate('admin', '1234');  console.log(API\_KEY);  logout(); |

وارد کردن همه با هم

|  |
| --- |
| // وارد کردن پیشفرض و همه صادرات نام‌دار  import authFunction, \* as AuthUtils from './auth.js';  authFunction('user', 'pass');  console.log(AuthUtils.API\_KEY);  AuthUtils.logout(); |

مثال‌های کاربردی کامل

ساختار پروژه ماژولار:

|  |
| --- |
| my-app/  ├── utils/  │ ├── math.js  │ ├── date.js  │ └── index.js  ├── services/  │ └── api.js  ├── models/  │ └── User.js  └── main.js |

فایل‌های ماژول:

utils/math.js

|  |
| --- |
| // صادرات نام‌دار  export const PI = 3.14159;  export function add(...numbers) {  return numbers.reduce((sum, num) => sum + num, 0);  }  export function multiply(a, b) {  return a \* b;  }  export function power(base, exponent) {  return Math.pow(base, exponent);  } |

utils/date.js

|  |
| --- |
| // صادرات پیشفرض  export default function formatDate(date, locale = 'fa-IR') {  return new Intl.DateTimeFormat(locale, {  year: 'numeric',  month: 'long',  day: 'numeric'  }).format(date);  }  // صادرات نام‌دار  export function getCurrentYear() {  return new Date().getFullYear();  }  export function getPersianMonth(monthIndex) {  const months = [  'فروردین', 'اردیبهشت', 'خرداد', 'تیر', 'مرداد', 'شهریور',  'مهر', 'آبان', 'آذر', 'دی', 'بهمن', 'اسفند'  ];  return months[monthIndex];  } |

utils/index.js (Barrel File)

|  |
| --- |
| // Barrel File - صادر کردن مجدد از ماژول‌های مختلف  export { PI, add, multiply, power } from './math.js';  export { default as formatDate, getCurrentYear, getPersianMonth } from './date.js'; |

services/api.js

|  |
| --- |
| const BASE\_URL = 'https://jsonplaceholder.typicode.com';  // صادرات نام‌دار برای توابع API  export async function fetchUsers() {  const response = await fetch(`${BASE\_URL}/users`);  if (!response.ok) throw new Error('خطا در دریافت کاربران');  return response.json();  }  export async function fetchUserPosts(userId) {  const response = await fetch(`${BASE\_URL}/users/${userId}/posts`);  if (!response.ok) throw new Error('خطا در دریافت پست‌ها');  return response.json();  }  export async function createPost(postData) {  const response = await fetch(`${BASE\_URL}/posts`, {  method: 'POST',  headers: {  'Content-Type': 'application/json',  },  body: JSON.stringify(postData)  });  if (!response.ok) throw new Error('خطا در ایجاد پست');  return response.json();  }  // صادرات پیشفرض برای کلاس API  export default class ApiService {  static async getComments(postId) {  const response = await fetch(`${BASE\_URL}/posts/${postId}/comments`);  return response.json();  }  } |

models/User.js

|  |
| --- |
| // صادرات پیشفرض  export default class User {  constructor(id, name, email) {  this.id = id;  this.name = name;  this.email = email;  }    displayInfo() {  return `${this.name} - ${this.email}`;  }    static createFromApi(data) {  return new User(data.id, data.name, data.email);  }  } |

فایل اصلی main.js:

|  |
| --- |
| // وارد کردن از Barrel File  import { add, multiply, formatDate, getCurrentYear } from './utils/index.js';  // وارد کردن از سرویس‌ها  import ApiService, { fetchUsers, fetchUserPosts } from './services/api.js';  // وارد کردن مدل‌ها  import User from './models/User.js';  // استفاده از ماژول‌ها  console.log('جمع اعداد:', add(1, 2, 3, 4, 5));  console.log('ضرب:', multiply(4, 5));  console.log('تاریخ امروز:', formatDate(new Date()));  console.log('سال جاری:', getCurrentYear());  // کار با API  async function displayUserData() {  try {  const users = await fetchUsers();  const firstUser = users[0];    // ایجاد نمونه User از داده‌های API  const user = User.createFromApi(firstUser);  console.log('کاربر:', user.displayInfo());    // دریافت پست‌های کاربر  const posts = await fetchUserPosts(user.id);  console.log('تعداد پست‌ها:', posts.length);    // دریافت نظرات با کلاس API  const comments = await ApiService.getComments(posts[0].id);  console.log('تعداد نظرات:', comments.length);    } catch (error) {  console.error('خطا:', error.message);  }  }  displayUserData(); |

نکات مهم و بهترین روش‌ها

1. ساختار ماژولار مناسب

|  |
| --- |
| // ❌ بد  import { func1, func2, const1, const2, class1 } from './bad-structure.js';  // ✅ خوب  import { helpers } from './utils/helpers.js';  import { constants } from './utils/constants.js';  import { ApiService } from './services/api.js'; |

استفاده از Barrel Files

|  |
| --- |
| // utils/index.js  export \* from './math.js';  export \* from './date.js';  export \* from './string-utils.js';  // سپس در main.js  import { add, formatDate, capitalize } from './utils/index.js'; |

نام‌گذاری مناسب

|  |
| --- |
| // برای صادرات پیشفرض از نام‌های معنادار استفاده کنید  import UserModel from './models/User.js'; // خوب  import User from './models/User.js'; // خوب  import U from './models/User.js'; // بد |

مدیریت خطا در Import

|  |
| --- |
| // بارگذاری پویا با مدیریت خطا  try {  const module = await import('./dynamic-module.js');  module.doSomething();  } catch (error) {  console.error('ماژول بارگذاری نشد:', error);  } |

جمع‌بندی

قوانین اصلی:

* از Named Export برای صادر کردن چندین مقدار استفاده کنید
* از Default Export برای صادر کردن مقدار اصلی ماژول استفاده کنید
* می‌توانید از هر دو روش در یک ماژول استفاده کنید
* از Barrel Files برای سازماندهی بهتر استفاده کنید

سینتکس‌های مهم:

|  |
| --- |
| // Export  export const name = 'value';  export function func() {}  export class ClassName {}  export { name1, name2 };  export default value;  // Import  import { name } from './module.js';  import defaultExport from './module.js';  import \* as alias from './module.js';  import { name as newName } from './module.js'; |

## **استفاده از allias و \* در import و export**

استفاده از Alias (نام مستعار) در Import

|  |
| --- |
| // math.js  export function add(a, b) {  return a + b;  }  export function multiply(a, b) {  return a \* b;  }  // main.js  import { add as جمع, multiply as ضرب } from './math.js';  console.log(جمع(2, 3)); // 5  console.log(ضرب(4, 5)); // 20 |

استفاده از \* برای Import همه چیز

|  |
| --- |
| // utils.js  export const PI = 3.14159;  export const VERSION = "1.0.0";  export function log(message) {  console.log(message);  }  // main.js  import \* as Utilities from './utils.js';  console.log(Utilities.PI); // 3.14159  console.log(Utilities.VERSION); // "1.0.0"  Utilities.log("سلام دنیا!"); // "سلام دنیا!" |

استفاده از Alias در Export

|  |
| --- |
| // api.js  function fetchUsers() {  return fetch('/api/users');  }  function createUser(data) {  return fetch('/api/users', {  method: 'POST',  body: JSON.stringify(data)  });  }  // Export با نام مستعار  export {  fetchUsers as getUsers,  createUser as postUser  }; |

## **معرفی مفاهیم شی گرایی و کلاس ها**

چهار اصل اصلی شی‌گرایی:

* کپسوله‌سازی (Encapsulation)
* وراثت (Inheritance)
* چندریختی (Polymorphism)
* انتزاع (Abstraction)

## **ساخت کلاس در js**

**ساخت کلاس پایه**

|  |
| --- |
| class Person {  constructor(name, age) {  this.name = name;  this.age = age;  }    introduce() {  return `سلام، من ${this.name} هستم و ${this.age} سال دارم.`;  }  }  // استفاده از کلاس  const person1 = new Person("علی", 25);  console.log(person1.introduce()); // "سلام، من علی هستم و 25 سال دارم." |

## **معرفی متغیر ها و توابع public و private**

معرفی متغیرها و توابع Public و Private

متغیرها و توابع Public (عمومی)

|  |
| --- |
| class BankAccount {  constructor(owner, balance) {  // متغیرهای public  this.owner = owner;  this.balance = balance;  }    // تابع public  getBalance() {  return this.balance;  }  }  const account = new BankAccount("علی", 1000);  console.log(account.owner); // "علی" - قابل دسترسی از خارج  console.log(account.getBalance()); // 1000 - قابل دسترسی از خارج |

**متغیرها و توابع Private خصوصی**

|  |
| --- |
| class BankAccount {  #accountNumber; // متغیر private با #  #pin;    constructor(owner, balance, accountNumber, pin) {  this.owner = owner; // public  this.balance = balance; // public  this.#accountNumber = accountNumber; // private  this.#pin = pin; // private  }    // تابع public  getAccountInfo() {  return `مالک: ${this.owner} - موجودی: ${this.balance}`;  }    // تابع private  #validatePin(inputPin) {  return this.#pin === inputPin;  }    // تابع public که از تابع private استفاده می‌کند  withdraw(amount, inputPin) {  if (this.#validatePin(inputPin)) {  if (amount <= this.balance) {  this.balance -= amount;  return `برداشت موفق. موجودی جدید: ${this.balance}`;  }  return "موجودی کافی نیست";  }  return "PIN نادرست";  }  }  const account = new BankAccount("رضا", 5000, "123456", "1234");  console.log(account.owner); // "رضا" - قابل دسترسی  // console.log(account.#accountNumber); // خطا - قابل دسترسی نیست  console.log(account.withdraw(1000, "1234")); // "برداشت موفق. موجودی جدید: 4000" |

## **معرفی setter و getter**

استفاده از Setter و Getter

|  |
| --- |
| class Student {  #name;  #grade;    constructor(name, grade) {  this.#name = name;  this.grade = grade; // از setter استفاده می‌کند  }    // Getter برای name  get name() {  return this.#name;  }    // Setter برای name  set name(newName) {  if (newName.length > 0) {  this.#name = newName;  } else {  console.log("نام نمی‌تواند خالی باشد");  }  }    // Getter برای grade  get grade() {  return this.#grade;  }    // Setter برای grade با اعتبارسنجی  set grade(newGrade) {  if (newGrade >= 0 && newGrade <= 20) {  this.#grade = newGrade;  } else {  console.log("نمره باید بین 0 و 20 باشد");  }  }    // Getter محاسبه‌شده  get status() {  return this.#grade >= 10 ? "قبول" : "مردود";  }  }  // استفاده  const student = new Student("مریم", 16);  console.log(student.name); // "مریم" - استفاده از getter  console.log(student.grade); // 16 - استفاده از getter  console.log(student.status); // "قبول" - استفاده از getter محاسبه‌شده  student.name = "زهرا"; // استفاده از setter  student.grade = 18; // استفاده از setter  // student.grade = 25; // "نمره باید بین 0 و 20 باشد" |

## **معرفی constructor**

Constructor پایه

|  |
| --- |
| class Car {  constructor(brand, model, year) {  this.brand = brand;  this.model = model;  this.year = year;  this.speed = 0;  this.isRunning = false;  }    start() {  this.isRunning = true;  return `${this.brand} ${this.model} روشن شد`;  }    accelerate(speed) {  if (this.isRunning) {  this.speed += speed;  return `سرعت فعلی: ${this.speed} km/h`;  }  return "لطفا اول ماشین را روشن کنید";  }  }  const myCar = new Car("پژو", "206", 2020);  console.log(myCar.start()); // "پژو 206 روشن شد"  console.log(myCar.accelerate(50)); // "سرعت فعلی: 50 km/h" |

Constructor پیشرفته با مقادیر پیشفرض

|  |
| --- |
| class Product {  constructor(name, price, category = "عمومی", discount = 0) {  this.name = name;  this.price = price;  this.category = category;  this.discount = discount;  this.createdAt = new Date();  }    get finalPrice() {  return this.price - (this.price \* this.discount / 100);  }    get productInfo() {  return `  نام محصول: ${this.name}  قیمت: ${this.price} تومان  دسته‌بندی: ${this.category}  تخفیف: ${this.discount}%  قیمت نهایی: ${this.finalPrice} تومان  `;  }  }  // استفاده  const product1 = new Product("لپ‌تاپ", 20000000);  const product2 = new Product("ماوس", 500000, "لوازم جانبی", 10);  console.log(product1.productInfo);  console.log(product2.productInfo); |

## **معرفی وراثت در شی گرایی**

وراثت ساده

|  |
| --- |
| // کلاس والد  class Animal {  constructor(name, sound) {  this.name = name;  this.sound = sound;  }    makeSound() {  return `${this.name} می‌گوید: ${this.sound}`;  }    eat() {  return `${this.name} در حال غذا خوردن است`;  }  }  // کلاس فرزند  class Dog extends Animal {  constructor(name, breed) {  super(name, "واق واق!"); // فراخوانی constructor والد  this.breed = breed;  }    // Override کردن متد والد  makeSound() {  return `${this.name} (نژاد: ${this.breed}) پارس می‌کند: ${this.sound}`;  }    // متد جدید مختص کلاس فرزند  fetch() {  return `${this.name} در حال آوردن توپ است`;  }  }  // کلاس فرزند دیگر  class Cat extends Animal {  constructor(name, color) {  super(name, "میو میو!");  this.color = color;  }    makeSound() {  return `${this.name} (رنگ: ${this.color}) میو میو می‌کند: ${this.sound}`;  }    climb() {  return `${this.name} در حال بالا رفتن از درخت است`;  }  }  // استفاده  const dog = new Dog("رکس", "ژرمن شپرد");  const cat = new Cat("پشمی", "نارنجی");  console.log(dog.makeSound()); // "رکس (نژاد: ژرمن شپرد) پارس می‌کند: واق واق!"  console.log(cat.makeSound()); // "پشمی (رنگ: نارنجی) میو میو می‌کند: میو میو!"  console.log(dog.fetch()); // "رکس در حال آوردن توپ است"  console.log(cat.eat()); // "پشمی در حال غذا خوردن است" (متد ارث‌بری شده) |

وراثت پیشرفته

|  |
| --- |
| // کلاس پایه  class Employee {  #id;    constructor(name, position, salary) {  this.#id = this.#generateId();  this.name = name;  this.position = position;  this.salary = salary;  }    #generateId() {  return Math.random().toString(36).substr(2, 9);  }    get id() {  return this.#id;  }    getInfo() {  return `${this.name} - ${this.position} - حقوق: ${this.salary}`;  }    calculateAnnualSalary() {  return this.salary \* 12;  }  }  // کلاس مدیر  class Manager extends Employee {  constructor(name, salary, department) {  super(name, "مدیر", salary);  this.department = department;  this.team = [];  }    addTeamMember(employee) {  this.team.push(employee);  }    // Override کردن متد والد  getInfo() {  return `${super.getInfo()} - بخش: ${this.department}`;  }    // متد جدید  getTeamSize() {  return this.team.length;  }  }  // کلاس توسعه‌دهنده  class Developer extends Employee {  constructor(name, salary, programmingLanguage) {  super(name, "توسعه‌دهنده", salary);  this.programmingLanguage = programmingLanguage;  this.projects = [];  }    addProject(project) {  this.projects.push(project);  }    getInfo() {  return `${super.getInfo()} - زبان برنامه‌نویسی: ${this.programmingLanguage}`;  }  }  // استفاده  const manager = new Manager("احمد", 10000000, "فناوری اطلاعات");  const dev1 = new Developer("رضا", 7000000, "JavaScript");  const dev2 = new Developer("مریم", 7500000, "Python");  manager.addTeamMember(dev1);  manager.addTeamMember(dev2);  console.log(manager.getInfo());  console.log(dev1.getInfo());  console.log(`تعداد اعضای تیم: ${manager.getTeamSize()}`); |

## **معرفی کلاس های static**

متدها و ویژگی‌های Static

|  |
| --- |
| class MathUtils {  // ویژگی static  static PI = 3.14159;  static VERSION = "1.0.0";    // متد static  static add(a, b) {  return a + b;  }    static multiply(a, b) {  return a \* b;  }    static circleArea(radius) {  return this.PI \* radius \* radius;  }    // متد معمولی (نیاز به نمونه دارد)  instanceMethod() {  return "این یک متد معمولی است";  }  }  // استفاده از static بدون ایجاد نمونه  console.log(MathUtils.PI); // 3.14159  console.log(MathUtils.add(5, 3)); // 8  console.log(MathUtils.circleArea(5)); // 78.53975  // ❌ این خط کار نمی‌کند:  // const util = new MathUtils();  // util.add(2, 3); // خطا |

کلاس کامل با Static

|  |
| --- |
| class User {  static #usersCount = 0;  static #onlineUsers = 0;    constructor(username, email) {  this.username = username;  this.email = email;  this.isOnline = false;    User.#usersCount++;  }    // متدهای معمولی  login() {  if (!this.isOnline) {  this.isOnline = true;  User.#onlineUsers++;  return `${this.username} وارد سیستم شد`;  }  return "کاربر از قبل آنلاین است";  }    logout() {  if (this.isOnline) {  this.isOnline = false;  User.#onlineUsers--;  return `${this.username} از سیستم خارج شد`;  }  return "کاربر از قبل آفلاین است";  }    // متدهای static  static getTotalUsers() {  return this.#usersCount;  }    static getOnlineUsers() {  return this.#onlineUsers;  }    static getUserStats() {  return `  آمار کاربران:  - کل کاربران: ${this.#usersCount}  - کاربران آنلاین: ${this.#onlineUsers}  - کاربران آفلاین: ${this.#usersCount - this.#onlineUsers}  `;  }  }  // استفاده  const user1 = new User("ali", "ali@example.com");  const user2 = new User("maryam", "maryam@example.com");  console.log(user1.login()); // "ali وارد سیستم شد"  console.log(user2.login()); // "maryam وارد سیستم شد"  console.log(User.getTotalUsers()); // 2  console.log(User.getOnlineUsers()); // 2  console.log(User.getUserStats());  console.log(user1.logout()); // "ali از سیستم خارج شد"  console.log(User.getOnlineUsers()); // 1 |

مثال کاربردی کامل

|  |
| --- |
| // کلاس پایه  class BankAccount {  static #totalAccounts = 0;  #accountNumber;  #balance;    constructor(owner, initialBalance = 0) {  this.owner = owner;  this.#balance = initialBalance;  this.#accountNumber = this.#generateAccountNumber();  BankAccount.#totalAccounts++;  }    #generateAccountNumber() {  return Math.random().toString(36).substr(2, 10).toUpperCase();  }    // Getterها  get accountNumber() {  return this.#accountNumber;  }    get balance() {  return this.#balance;  }    // متدها  deposit(amount) {  if (amount > 0) {  this.#balance += amount;  return `مبلغ ${amount} تومان واریز شد. موجودی جدید: ${this.#balance}`;  }  return "مبلغ واریزی باید مثبت باشد";  }    withdraw(amount) {  if (amount > 0 && amount <= this.#balance) {  this.#balance -= amount;  return `مبلغ ${amount} تومان برداشت شد. موجودی جدید: ${this.#balance}`;  }  return "مبلغ برداشت نامعتبر است";  }    // متد static  static getTotalAccounts() {  return this.#totalAccounts;  }  }  // کلاس فرزند  class SavingsAccount extends BankAccount {  #interestRate;    constructor(owner, initialBalance, interestRate = 5) {  super(owner, initialBalance);  this.#interestRate = interestRate;  }    addInterest() {  const interest = this.balance \* (this.#interestRate / 100);  this.deposit(interest);  return `سود ${interest} تومان اضافه شد`;  }    getInfo() {  return `  حساب پس‌انداز  مالک: ${this.owner}  شماره حساب: ${this.accountNumber}  موجودی: ${this.balance} تومان  نرخ سود: ${this.#interestRate}%  `;  }  }  // استفاده  const account1 = new BankAccount("علی رضایی", 1000);  const savings1 = new SavingsAccount("مریم محمدی", 5000, 7);  console.log(account1.deposit(2000)); // "مبلغ 2000 تومان واریز شد. موجودی جدید: 3000"  console.log(savings1.getInfo());  console.log(savings1.addInterest()); // "سود 350 تومان اضافه شد"  console.log(`تعداد کل حساب‌ها: ${BankAccount.getTotalAccounts()}`); // 2 |

## **معرفی es-toolkit.dev**

es-toolkit یک کتابخانه کاربردی مدرن برای جاوااسکریپت است که مجموعه‌ای از توابع قدرتمند و بهینه را برای کارهای روزمره برنامه‌نویسی ارائه می‌دهد. این کتابخانه بر روی اندازه bundle کوچک و عملکرد بسیار بالا در محیط‌های مدرن جاوااسکریپت تمرکز دارد.

چرا از es-toolkit استفاده کنیم؟

مهم‌ترین دلایل برای انتخاب es-toolkit نسبت به کتابخانه‌های مشابه مانند Lodash عبارتند از:

عملکرد برتر: این کتابخانه به طور متوسط ۲ تا ۳ برابر سریع‌تر از کتابخانه‌های جایگزین عمل می‌کند و در برخی توابع مانند omit تا ۱۱ برابر بهبود عملکرد دارد.

حجم بسیار کمتر: تا ۹۷٪ کد جاوااسکریپت کمتری نسبت به کتابخانه‌های جایگزین ارسال می‌کند که منجر به بارگذاری سریع‌تر صفحات وب می‌شود.

پشتیبانی کامل از TypeScript: دارای تایپ‌های از پیش تعریف شده و ساده اما قدرتمند برای تمامی توابع است.

پوشش تست کامل: با ۱۰۰٪ پوشش تست، از قابلیت اطمینان و استحکام بالایی برخوردار است.

سازگاری کامل با Lodash: یک لایه سازگاری کامل برای جایگزینی بی‌دردسر Lodash ارائه می‌دهد.

پشتیبانی از تمام محیط‌ها: از جمله Node.js، Deno، Bun و مرورگرها.

نصب و راه‌اندازی

es-toolkit از طریق پکیج منیجرهای مختلف در دسترس است:

|  |  |
| --- | --- |
| محیط | دستور نصب |
| Node.js | npm install es-toolkit |
| Deno | deno add @es-toolkit/es-toolkit |
| Bun | bun add es-toolkit |
| مرورگر (CDN) | <script src="https://cdn.jsdelivr.net/npm/es-toolkit@^1"></script> |

نمونه‌هایی از کاربرد

در ادامه نمونه‌هایی از پرکاربردترین توابع این کتابخانه را مشاهده می‌کنید:

کار با آرایه‌ها: توابعی برای مدیریت و پردازش آسان آرایه‌ها

|  |
| --- |
| import { chunk, uniq, filter } from 'es-toolkit';  const arr = [1, 2, 2, 3, 4, 4, 5];  const uniqueArr = uniq(arr); // نتیجه: [1, 2, 3, 4, 5]  const array = [1, 2, 3, 4, 5, 6];  const chunkedArray = chunk(array, 2); // نتیجه: [[1, 2], [3, 4], [5, 6]]  const users = [{ id: 1, active: true }, { id: 2, active: false }];  const activeUsers = filter(users, user => user.active); // نتیجه: [{ id: 1, active: true }] |

کنترل اجرای توابع: توابع کمکی برای مدیریت بهتر فراخوانی توابع

|  |
| --- |
| import { debounce, throttle } from 'es-toolkit';  const searchInputHandler = (query) => console.log('Searching for:', query);  const debouncedSearch = debounce(searchInputHandler, 300); // 300 میلی‌ثانیه تاخیر  const handleScrollEvent = () => console.log('Scrolling...');  const throttledScroll = throttle(handleScrollEvent, 100); // حداکثر یک فراخوانی در 100 میلی‌ثانیه |

کار با آبجکت‌ها: توابعی برای مدیریت و دستکاری آبجکت‌ها

|  |
| --- |
| import { get, set, pick } from 'es-toolkit';  const data = { user: { profile: { name: 'Reza' } } };  const userName = get(data, 'user.profile.name'); // نتیجه: 'Reza'  const config = {};  set(config, 'app.settings.theme', 'dark'); // نتیجه: { app: { settings: { theme: 'dark' } } }  const fullUser = { id: 'abc', name: 'Reza', email: 'Reza@example.com' };  const publicUser = pick(fullUser, ['id', 'name']); // نتیجه: { id: 'abc', name: 'Reza' } |

دسته‌بندی توابع

es-toolkit توابع خود را در دسته‌بندی‌های مختلفی ارائه می‌دهد که مهم‌ترین آن‌ها عبارتند از:

* Array: توابعی مانند uniq و difference برای دستکاری آرایه
* Function: ابزارهایی برای کنترل اجرای تابع از جمله debounce و throttle
* Math: ابزارهای عددی مانند sum و round
* Object: توابعی برای کار با آبجکت‌های جاوااسکریپت مانند pick و omit
* Promise: ابزارهای ناهمزمان مانند delay
* String: توابعی برای دستکاری رشته

## چندپارادایمی:

پشتیبانی از برنامه‌نویسی شیءگرا، تابعی و رویه‌ای: توضیح چندپارادایمی بودن جاوااسکریپت

جاوااسکریپت یک زبان برنامه‌نویسی چندپارادایمی است که از سبک‌های مختلف برنامه‌نویسی پشتیبانی می‌کند. در این مثال، هر سه پارادایم اصلی را نمایش می‌دهم.

### برنامه نویسی تابعی یا رویه‌ایی:

|  |
| --- |
| // برنامه‌نویسی رویه‌ای تمرکز بر توابع و رویه‌ها  // محاسبه مساحت دایره  function calculateCircleArea(radius) {  return Math.PI radius radius;  }  // محاسبه محیط دایره  function calculateCircleCircumference(radius) {  return 2 Math.PI radius;  }  // استفاده از توابع  const radius = 5;  const area = calculateCircleArea(radius);  const circumference = calculateCircleCircumference(radius);  console.log(`مساحت دایره: ${area}`);  console.log(`محیط دایره: ${circumference}`); |

### برنامه نویسی شیئ گرا (ObjectOriented Programming)

|  |
| --- |
| // برنامه‌نویسی شیءگرا استفاده از کلاس‌ها و اشیا  // تعریف کلاس  class Person {      constructor(name, age) {          this.name = name;          this.age = age;      }        // متد      greet() {          return `سلام، من ${this.name} هستم و ${this.age} سال دارم.`;      }        // متد دیگر      haveBirthday() {          this.age++;          return `تولدت مبارک! حالا ${this.age} ساله شدی.`;      }  }  // ارث‌بری  class Student extends Person {      constructor(name, age, studentId) {          super(name, age);          this.studentId = studentId;      }        study() {          return `${this.name} در حال مطالعه است.`;      }  }  // ایجاد نمونه‌ها  const person1 = new Person("امین", 30);  const student1 = new Student("سارا", 22, "12345");  console.log(person1.greet());  console.log(student1.greet());  console.log(student1.study());  console.log(person1.haveBirthday()); |

[مثال بالا در دنیای وب](webdesign/js/Object/js/ObjectOrientedProgramming.js)

### برنامه‌نویسی تابعی (Functional Programming)

|  |
| --- |
| // برنامه‌نویسی تابعی استفاده از توابع خالص و توابع مرتبه بالا  // توابع خالص (همان ورودی → همیشه همان خروجی)  const add = (a, b) => a + b;  const multiply = (a, b) => a b;  // تابعی که تابع برمی‌گرداند (تابع ساز)  const createMultiplier = (factor) => (number) => number factor;  // توابع مرتبه بالا (HigherOrder Functions)  const numbers = [1, 2, 3, 4, 5];  // map تبدیل آرایه  const doubled = numbers.map(n => n 2);  // filter فیلتر کردن آرایه  const evens = numbers.filter(n => n % 2 === 0);  // reduce کاهش آرایه به یک مقدار  const sum = numbers.reduce((acc, n) => acc + n, 0);  // ترکیب توابع  const pipe = (...fns) => (x) => fns.reduce((v, f) => f(v), x);  const double = x => x 2;  const increment = x => x + 1;  const square = x => x x;  const transform = pipe(double, increment, square);  const result = transform(5); // (52 + 1)^2 = 121  console.log("اعداد دوبرابر:", doubled);  console.log("اعداد زوج:", evens);  console.log("مجموع اعداد:", sum);  console.log("نتیجه ترکیب توابع:", result); |

[مثال بالا در دنیای وب](webdesign/js/Object/js/FunctionalProgramming.js)

ترکیب پارادایم‌ها

|  |
| --- |
| // ترکیب پارادایم‌های مختلف  // رویه‌ای + تابعی  function processUserData(users, filterFunc, mapFunc) {      return users          .filter(filterFunc)          .map(mapFunc);  }  // شیءگرا + تابعی  class Calculator {      constructor() {          this.operations = [];      }        // ذخیره عملیات به صورت تابع      addOperation(name, operationFunc) {          this.operations.push({ name, func: operationFunc });      }        executeOperation(name, ...args) {          const operation = this.operations.find(op => op.name === name);          if (operation) {              return operation.func(...args);          }          throw new Error(`Operation ${name} not found`);      }  }  // استفاده از ترکیب پارادایم‌ها  const users = [      { name: "Ali", age: 25, active: true },      { name: "Sara", age: 30, active: false },      { name: "Reza", age: 22, active: true }  ];  const activeUsers = processUserData(      users,      user => user.active,  // تابع فیلتر (تابعی)  (user) => {return user.active;}      user => user.name      // تابع تبدیل (تابعی) (user) => {return user.name;}  ); // call processUserData() function with specefic arugomans  console.log("کاربران فعال:", activeUsers);  // ایجاد ماشین حساب و اضافه کردن عملیات  const calc = new Calculator();  // شیءگرا  calc.addOperation("add", (a, b) => a + b);  // تابعی  calc.addOperation("multiply", (a, b) => a b);  console.log("جمع:", calc.executeOperation("add", 5, 3));  console.log("ضرب:", calc.executeOperation("multiply", 5, 3)); |

[ترکیب پارادایم‌ها](webdesign/js/Object/js/paradim.js)

این مثال‌ها نشان می‌دهند که چگونه جاوااسکریپت از چندین پارادایم برنامه‌نویسی پشتیبانی می‌کند و چگونه می‌توان این پارادایم‌ها را با هم ترکیب کرد تا کدهای انعطاف‌پذیر و قدرتمندی ایجاد شود.

[آموزش استفاده از map](#_Map_در_جاوا)   
[آموزش استفاده از filter](#_آموزش_استفاده_از)

[آموزش استفاده از reduce](#_آموزش_استفاده_از_1)

[آموزش استفاده از Rest parameters](#_آموزش_استفاده_از_2) و [آموزش استفاده از Spread Operator](#_آموزش_استفاده_از_2)

جاوااسکریپت به شما امکان می‌دهد از هر یک از این پارادایم‌ها یا ترکیبی از آن‌ها استفاده کنید که این انعطاف‌پذیری باعث می‌شود بتوانید بهترین روش را برای حل هر مسئله انتخاب نمایید.

کاربردهای اصلی:

افزودن تعامل به صفحات وب

ایجاد برنامه‌های تحت وب (Web Applications)

توسعه سمت سرور با Node.js

ساخت اپلیکیشن‌های موبایل

توسعه بازی‌های تحت وب

|  |
| --- |
| // نمایش پیام در کنسول  console.log("Hello, World!");  // تعریف متغیر  let name = "John";  const age = 25;  // تابع ساده  function greet(person) {      return "Hello, " + person + "!";  }  // استفاده از تابع  console.log(greet(name)); |

[مثال ساده](webdesign/js/Object/js/simple.js)

محیط‌های اجرا:

مرورگرهای وب (Chrome, Firefox, Safari)

Node.js برای اجرای سمت سرور

JavaScript امروزه به یکی از محبوب‌ترین زبان‌های برنامه‌نویسی جهان تبدیل شده و تقریباً در تمام وبسایت‌های مدرن استفاده می‌شود.

## آموزش استفاده از Map در جاوا اسکریپت

### مقدمه‌ای بر Map

`Map` یک ساختار داده‌ای در جاوااسکریپت است که به شما امکان ذخیره‌سازی جفت‌های کلید-مقدار (key-value) را می‌دهد. برخلاف Object، کلیدهای Map می‌توانند از هر نوع داده‌ای باشند.

|  |
| --- |
| // ایجاد یک Map جدید  const myMap = new Map();  // یا مقداردهی اولیه با آرایه‌ای از جفت‌های [key, value]  const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  console.log(initializedMap.get("name")); |

[Map()](webdesign/js/Object/js/map.js)

#### نکته برای دسترسی به عناصر Map از متد `get()` استفاده می‌شود. در مورد مثال شما:

##### روش‌های دسترسی به مقادیر Map

1. استفاده از متد `get()`

|  |
| --- |
| const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  // دسترسی به مقدار 'name'  const nameValue = initializedMap.get('name');  console.log(nameValue); // خروجی: 'Alice'  // دسترسی به سایر مقادیر  const numberValue = initializedMap.get(1);  console.log(numberValue); // خروجی: 'number one'  const booleanValue = initializedMap.get(true);  console.log(booleanValue); // خروجی: 'boolean true' |

[get()](webdesign/js/Object/js/get.js)

* 1. بررسی وجود کلید قبل از دسترسی

|  |
| --- |
| // بررسی وجود کلید قبل از دسترسی  const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  if (initializedMap.has('name')) {  const name = initializedMap.get('name');  console.log('Name found:', name); // خروجی: 'Name found: Alice'  } else {  console.log('Key not found');  } |

[get()  بررسی وجود کلید قبل از دسترسی](webdesign/js/Object/js/getExist.js)

1. دسترسی به همه مقادیر با تکرار (Iteration)

|  |
| --- |
| // getIteration.js  const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  // روش ۱: استفاده از for...of  for (const [key, value] of initializedMap) {  console.log(key, ':', value);  }  // خروجی:  // name : Alice  // 1 : number one  // true : boolean true  // روش ۲: استفاده از forEach  initializedMap.forEach((value, key) => {  console.log(`${key} → ${value}`);  }); |

[getIteration](webdesign/js/Object/js/getIteration.js)

1. دریافت همه کلیدها و مقادیر

|  |
| --- |
| // getAll.js  const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  // دریافت همه کلیدها  const keys = Array.from(initializedMap.keys());  console.log(keys); // خروجی: ['name', 1, true]  // دریافت همه مقادیر  const values = Array.from(initializedMap.values());  console.log(values); // خروجی: ['Alice', 'number one', 'boolean true']  // دریافت همه entries به صورت آرایه  const entries = Array.from(initializedMap.entries());  console.log(entries);  // خروجی: [['name', 'Alice'], [1, 'number one'], [true, 'boolean true']] |

[getAll.js](webdesign/js/Object/js/getAll.js)

##### مثال‌های کاربردی بیشتر

مثال ۱: مدیریت خطا برای کلیدهای ناموجود

|  |
| --- |
| const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  function safeGet(map, key, defaultValue = null) {  return map.has(key) ? map.get(key) : defaultValue;  }  const name = safeGet(initializedMap, 'name', 'Unknown');  const age = safeGet(initializedMap, 'age', 25); // کلید وجود ندارد  console.log(name); // 'Alice'  console.log(age); // 25 (مقدار پیش‌فرض) |

[getDefault](webdesign/js/Object/js/getDefault.js)

مثال ۲: استفاده در تابع

|  |
| --- |
| const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  function getUserInfo(map) {  return {  name: map.get('name') || 'N/A',  id: map.get('id') || 0,  isActive: map.get('isActive') || false  };  }  const userInfo = getUserInfo(initializedMap);  console.log(userInfo);  // خروجی: { name: 'Alice', id: 0, isActive: false } |

[get() استفاده در تابع](webdesign/js/Object/js/getFunction.js)

مثال ۳: تبدیل Map به Object

|  |
| --- |
| function mapToObject(map) {  const obj = {};  for (const [key, value] of map) {  obj[key] = value;  }  return obj;  }  const mapObject = mapToObject(initializedMap);  console.log(mapObject.name); // 'Alice'  console.log(mapObject[1]); // 'number one'  console.log(mapObject[true]); // 'boolean true' |

[getToObject](webdesign/js/Object/js/getToObject.js)

##### نکات مهم

1. کلیدها case-sensitive هستند:

|  |
| --- |
| const myMap = new Map([['NAME', 'Bob']]);  console.log(myMap.get('name')); // undefined  console.log(myMap.get('NAME')); // 'Bob' |

1. اشیا به عنوان کلید:

|  |
| --- |
| const objKey = { id: 1 };  const mapWithObject = new Map([[objKey, 'object value']]);  console.log(mapWithObject.get(objKey)); // 'object value'  console.log(mapWithObject.get({ id: 1 })); // undefined (اشیا مختلف) |

1. مقدار پیش‌فرض برای کلیدهای ناموجود:

|  |
| --- |
| // استفاده از || برای مقدار پیش‌فرض  const value = initializedMap.get('nonexistent') || 'Default Value';  console.log(value); // 'Default Value'  // استفاده از Nullish Coalescing (??)  const value2 = initializedMap.get('nonexistent') ?? 'Default';  console.log(value2); // 'Default' |

جمع‌بندی

برای دسترسی به مقادیر Map از `map.get(key)` استفاده کنید. همیشه بهتر است ابتدا با `map.has(key)` وجود کلید را بررسی کنید تا از خطاهای `undefined` جلوگیری شود.

|  |
| --- |
| const initializedMap = new Map([  ['name', 'Alice'],  [1, 'number one'],  [true, 'boolean true']  ]);  // بهترین روش دسترسی  if (initializedMap.has('name')) {  const name = initializedMap.get('name');  console.log('Name:', name); // خروجی: 'Name: Alice'  }  // یا با مقدار پیش‌فرض  const name = initializedMap.get('name') || 'Unknown';  console.log(name); // 'Alice' |

تفاوت Map با Object

#### مقایسه Map و Object

|  |  |  |
| --- | --- | --- |
| ویژگی | Map | Map |
| نوع کلیدها | هر نوع داده‌ای | فقط String یا Symbol |
| ترتیب عناصر | حفظ ترتیب درج | ترتیب تضمین شده نیست |
| سایز | size` property | باید manually محاسبه شود |
| پرفورمنس | بهتر برای افزودن/حذف مکرر | بهتر برای موارد ساده |
| تکرار | built-in | نیاز به Object.keys() دارد |

مثال مقایسه‌ای:

|  |
| --- |
| // Object  const obj = {};  obj[1] = 'number'; // کلید به string تبدیل می‌شود: '1'  obj['1'] = 'string'; // مقدار قبلی overwrite می‌شود  // Map  const map = new Map();  map.set(1, 'number'); // کلید number می‌ماند  map.set('1', 'string'); // کلید جداگانه برای string  console.log(obj); // { '1': 'string' }  console.log(map); // Map(2) { 1 → 'number', '1' → 'string' } |

#### متدهای اصلی Map

|  |
| --- |
| const map = new Map();  // 1. set() -  map.set('name', 'Alice');  map.set('age', 30);  map.set('name', 'Bob'); // Update مقدار  // 2. get() - دریافت مقدار  console.log(map.get('name')); // 'Bob'  console.log(map.get('age')); // 30  // 3. has() - بررسی وجود کلید  console.log(map.has('name')); // true  console.log(map.has('city')); // false  // 4. delete() - حذف یک کلید  map.delete('age');  console.log(map.has('age')); // false  // 5. clear() - حذف همه عناصر  map.clear();  console.log(map.size); // 0  // 6. size - تعداد عناصر  console.log(map.size); // 0 |

[getMethod](webdesign/js/Object/js/getMethod.js)

متدهای تکرار (Iteration):

|  |
| --- |
| const userMap = new Map([  ['name', 'Alice'],  ['age', 30],  ['city', 'New York']  ]);  // 1. keys() - کلیدها  for (const key of userMap.keys()) {  console.log(key); // 'name', 'age', 'city'  }  // 2. values() - مقادیر  for (const value of userMap.values()) {  console.log(value); // 'Alice', 30, 'New York'  }  // 3. entries() - جفت‌های [key, value] (پیش‌فرض)  for (const [key, value] of userMap.entries()) {  console.log(key, value);  }  // 4. forEach() - مانند آرایه  userMap.forEach((value, key) => {  console.log(key, value);  }); |

[mapIteration2](webdesign/js/Object/js/mapIteration2.js)

#### مثال‌های کاربردی

مثال 1: مدیریت کاربران

|  |
| --- |
| class UserManager {  constructor() {  this.users = new Map();  }  addUser(id, userData) {  this.users.set(id, { ...userData, createdAt: new Date() });  }  getUser(id) {  return this.users.get(id);  }  updateUser(id, updates) {  if (this.users.has(id)) {  const user = this.users.get(id);  this.users.set(id, { ...user, ...updates });  }  }  deleteUser(id) {  this.users.delete(id);  }  getAllUsers() {  return Array.from(this.users.entries());  }  }  // استفاده  const manager = new UserManager();  manager.addUser(1, { name: 'Alice', email: 'alice@example.com' });  manager.addUser(2, { name: 'Bob', email: 'bob@example.com' });  console.log(manager.getUser(1)); |

[mapUserManagement](webdesign/js/Object/js/mapUserManagement.js)

مثال 2: کش داده‌ها (Caching)

|  |
| --- |
| class DataCache {  constructor(maxSize = 100) {  this.cache = new Map();  this.maxSize = maxSize;  }  set(key, value) {  if (this.cache.size >= this.maxSize) {  // حذف قدیمی‌ترین آیتم (اولین آیتم)  const firstKey = this.cache.keys().next().value;  this.cache.delete(firstKey);  }  this.cache.set(key, value);  }  get(key) {  if (this.cache.has(key)) {  const value = this.cache.get(key);  // بروزرسانی به عنوان جدیدترین آیتم  this.cache.delete(key);  this.cache.set(key, value);  return value;  }  return null;  }  clear() {  this.cache.clear();  }  } |

[mapCaching](webdesign/js/Object/js/mapCaching.js)

مثال 3: شمارش تکرار عناصر

|  |
| --- |
| // mapCounting.js  function countOccurrences(array) {  const countMap = new Map();    for (const item of array) {  console.log("item",item);  console.log("countMap.get(item):",countMap.get(item));  const currentCount = countMap.get(item) || 0;  console.log("currentCount:",currentCount);  countMap.set(item, currentCount + 1);  }    return countMap;  }  const fruits = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple'];  const fruitCounts = countOccurrences(fruits);  console.log(fruitCounts);  // Map(3) { 'apple' → 3, 'banana' → 2, 'orange' → 1 } |

[mapCounting](webdesign/js/Object/js/mapCounting.js)

مثال 4: تبدیل بین ساختارها

|  |  |
| --- | --- |
| // تبدیل Object به Map  // const obj = { a: 1, b: 2, c: 3 };  // const mapFromObj = new Map(Object.entries(obj));  // // تبدیل Map به Object  // const objFromMap = Object.fromEntries(mapFromObj);  // console.log(objFromMap);  // // تبدیل Map به Array  // const arrayFromMap = Array.from(mapFromObj);  // console.log(arrayFromMap);  // // یا  // const arrayFromMap2 = [...mapFromObj];  // console.log(arrayFromMap2);  // // تبدیل Array به Map  // const array = [['a', 1], ['b', 2], ['c', 3]];  // const mapFromArray = new Map(array);  // console.log(mapFromArray); |  |

[mapToObject2](webdesign/js/Object/js/mapToObject2.js)

### نکات پیشرفته

استفاده از کلیدهای Object

|  |
| --- |
| const user1 = { id: 1, name: 'Alice' };  const user2 = { id: 2, name: 'Bob' };  const userSettings = new Map();  userSettings.set(user1, { theme: 'dark', language: 'en' });  userSettings.set(user2, { theme: 'light', language: 'fa' });  console.log(userSettings.get(user1)); // { theme: 'dark', language: 'en' } |

[mapObjectKeys](webdesign/js/Object/js/mapObjectKeys.js)

#### کلیدهای NaN

|  |
| --- |
| const map = new Map();  map.set(NaN, 'This is NaN');  console.log(map.get(NaN)); // 'This is NaN'  console.log(map.get(Number('abc'))); // 'This is NaN' - چون NaN === NaN |

[mapNaN](webdesign/js/Object/js/mapNaN.js)

#### زنجیره‌ای کردن متد set

|  |
| --- |
| const map = new Map();  // زنجیره‌ای کردن  map.set('a', 1)  .set('b', 2)  .set('c', 3);  console.log(map); // Map(3) { 'a' → 1, 'b' → 2, 'c' → 3 } |

[mapChainSets](webdesign/js/Object/js/mapChainSets.js)

استفاده با توابع

|  |
| --- |
| function createCounter() {  const counts = new Map();    return {  increment(key) {  counts.set(key, (counts.get(key) || 0) + 1);  },  getCount(key) {  return counts.get(key) || 0;  },  getAllCounts() {  return Array.from(counts.entries());  }  };  }  const counter = createCounter();  counter.increment('apple');  counter.increment('banana');  counter.increment('apple');  console.log(counter.getCount('apple')); // 2  console.log(counter.getAllCounts()); // [['apple', 2], ['banana', 1]] |

[mapFunctions2](webdesign/js/Object/js/mapFunctions2.js)

##### جمع‌بندی

چه زمانی از Map استفاده کنیم:

1. کلیدهای غیر-string نیاز دارید

2. ترتیب عناصر مهم است

3. تعداد عناصر را زیاد نیاز دارید

4. عملکرد افزودن/حذف مکرر مهم است

5. نیاز به تکرار ساده دارید

##### مزایای Map:

- کلیدهای با انواع مختلف

- حفظ ترتیب درج

- سایز built-in

- تکرار آسان

- پرفورمنس بهتر برای عملیات مکرر

محدودیت‌ها:

- تبدیل به JSON نیاز به کار اضافه دارد

- سینتکس ساده‌تر Object برای موارد ساده

|  |
| --- |
| // تبدیل Map به JSON و برعکس  const map = new Map([['name', 'Alice'], ['age', 30]]);  // به JSON  const json = JSON.stringify(Array.from(map.entries()));  // از JSON  const reconstructedMap = new Map(JSON.parse(json)); |

[map()   JSON](webdesign/js/Object/js/mapJSON.js)

Map یک ابزار قدرتمند در جاوااسکریپت است که در بسیاری از سناریوها می‌تواند جایگزین بهتری برای Object باشد، به خصوص وقتی که با داده‌های پویا و پیچیده کار می‌کنید.

### تمرینات اضافه

1. مپ کردن آرایه:

|  |
| --- |
| const array = [1, 4, 9, 16];  // Pass a function to map  const mapped = array.map((x) => x 2);  console.log(mapped);  // Expected output: Array [2, 8, 18, 32] |

[mapArray](webdesign/js/Object/js/mapArray.js)

1. مپ کردن آرایه با مربع عناصر:

|  |
| --- |
| //js/mapSqrt.js  const numbers = [1, 4, 9];  const roots = numbers.map((num) => Math.sqrt(num));  // roots is now [1, 2, 3]  // numbers is still [1, 4, 9]  console.log(numbers);  console.log(roots); |

[mapSqrt](webdesign/js/Object/js/mapSqrt.js)

1. تبدیل آرایه رشته‌ها به عناصر li با استفاده از Map در جاوااسکریپت

روش‌های مختلف برای تبدیل آرایه به عناصر li

استفاده از `map()` و `join()`

|  |
| --- |
| const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  // روش 1: استفاده از map و join  const htmlList = fruits.map(fruit => `<li>${fruit}</li>`).join('\n');  console.log(htmlList);  // خروجی:  // <li>سیب</li>  // <li>موز</li>  // <li>پرتقال</li>  // <li>انگور</li>  // قرار دادن در ul  const fullHtml = `<ul>\n${htmlList}\n</ul>`;  console.log(fullHtml); |

[Map join()](webdesign/js/Object/js/mapJoin.js)

2. ایجاد عناصر DOM واقعی

|  |
| --- |
| const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  // روش 2: ایجاد عناصر DOM واقعی  const listItems = fruits.map(fruit => {  const li = document.createElement('li');  li.textContent = fruit;  li.className = 'fruit-item';  return li;  });  // اضافه کردن به DOM  const ulElement = document.createElement('ul');  ulElement.append(...listItems);  document.body.appendChild(ulElement); |

[mapDOM](webdesign/js/Object/js/mapDOM.js)

1. استفاده با template literals

|  |
| --- |
| const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  // روش 3: با قالب‌بزی پیشرفته  const htmlList = fruits.map(fruit => `  <li class="fruit-item" data-fruit="${fruit}">  <span>${fruit.toUpperCase()}</span>  </li>  `).join('');  const fullHtml = `  <ul class="fruits-list">  ${htmlList}  </ul>  `;  console.log(fullHtml); |

[mapLiterals](webdesign/js/Object/js/mapLiterals.js)

مثال‌های کاربردی

مثال 1: لیست کاربران

|  |
| --- |
| const users = ['علی محمدی', 'فاطمه احمدی', 'محمد رضایی', 'زهرا حسینی'];  const userList = users.map(user => `  <li class="user-item">  <i class="icon-user"></i>  <span>${user}</span>  </li>  `).join('');  const usersHtml = `  <div class="users-container">  <h3>لیست کاربران</h3>  <ul class="users-list">  ${userList}  </ul>  </div>  `;  // قرار دادن در DOM  document.getElementById('app').innerHTML = usersHtml; |

[mapUserList](webdesign/js/Object/js/mapUserList.js)

مثال 2: لیست با شماره

|  |
| --- |
| const tasks = ['تمرین جاوااسکریپت', 'پروژه وب', 'مطالعه ریاضی', 'ورزش'];  const numberedList = tasks.map((task, index) => `  <li class="task-item">  <span class="task-number">${index + 1}.</span>  <span class="task-text">${task}</span>  <button onclick="completeTask(${index})">انجام شد</button>  </li>  `).join('');  const tasksHtml = `  <div class="tasks-wrapper">  <h2>لیست کارهای روزانه</h2>  <ol class="tasks-list">  ${numberedList}  </ol>  </div>  `; |

[mapUserOlList](webdesign/js/Object/js/mapUserOlList.js)

مثال 3: لیست با آیکون

|  |
| --- |
| const features = [  'پشتیبانی 24/7',  'امنیت بالا',  'رابط کاربری آسان',  'پشتیبانی از تمام دستگاه‌ها'  ];  const icons = ['🕒', '🔒', '💻', '📱'];  const featureList = features.map((feature, index) => `  <li class="feature-item">  <span class="feature-icon">${icons[index]}</span>  <span class="feature-text">${feature}</span>  </li>  `).join('');  const featuresHtml = `  <section class="features">  <h2>ویژگی‌های محصول</h2>  <ul class="features-list">  ${featureList}  </ul>  </section>  `; |

[mapListIcon](webdesign/js/Object/js/mapListIcon.js)

بهینه‌سازی عملکرد

استفاده از [DocumentFragment](#_آموزش_کار_با) برای عملکرد بهتر

|  |
| --- |
| const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  function createListItems(items) {  const fragment = document.createDocumentFragment();    items.map(item => {  const li = document.createElement('li');  li.textContent = item;  li.className = 'fruit-item';  fragment.appendChild(li);  return li;  });    return fragment;  }  const ulElement = document.createElement('ul');  ulElement.appendChild(createListItems(fruits));  // فقط یک بار DOM را آپدیت می‌کنیم  document.getElementById('app').appendChild(ulElement); |

[mapDocumentFragment](webdesign/js/Object/js/mapDocumentFragment.js)

استفاده از innerHTML برای کارایی بالا

|  |
| --- |
| const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  function renderList(items) {  const html = items.map(item => `<li>${item}</li>`).join('');  return `<ul>${html}</ul>`;  }  document.getElementById('app').innerHTML = renderList(fruits); |

[mapinnerHTML](webdesign/js/Object/js/mapinnerHTML.js)

تابع قابل استفاده مجدد

|  |
| --- |
| function arrayToHtmlList(items, options = {}) {  const {  listType = 'ul',  itemClass = '',  listClass = '',  transform = item => item  } = options;    const listItems = items.map(item => `  <li class="${itemClass}">${transform(item)}</li>  `).join('');    return `  <${listType} class="${listClass}">  ${listItems}  </${listType}>  `;  }  // استفاده  const fruits = ['سیب', 'موز', 'پرتقال', 'انگور'];  const html = arrayToHtmlList(fruits, {  listType: 'ul',  listClass: 'fruits-list',  itemClass: 'fruit-item',  transform: fruit => `<strong>${fruit}</strong>`  });  let app = document.getElementById("app");  app.innerHTML = html; |

[mapFunction](webdesign/js/Object/js/mapFunction.js)

جمع‌بندی

استفاده از `map()` برای تبدیل آرایه رشته‌ها به عناصر li بهترین روش است زیرا:

1. کد تمیز و خوانا

2. قابل استفاده مجدد

3. عملکرد عالی ⚡

4. قابل ترکیب با سایر متدها 🧩

|  |
| --- |
| // mapSimple2.js  // ساده‌ترین روش  const items = ['آیتم ۱', 'آیتم ۲', 'آیتم ۳'];  const html = items.map(item => `<li>${item}</li>`).join('');  // یا برای عناصر واقعی  const listElements = items.map(item => {      const li = document.createElement('li');      li.textContent = item;      return li;  });  let app = document.getElementById("app");  app.innerHTML = html;  listElements.forEach(li => app.appendChild(li)); |

[mapSimple2](webdesign/js/Object/js/mapSimple2.js)

این روش برای ایجاد لیست‌های پویا، منوها، نتایج جستجو و هر نوع لیستی در وب اپلیکیشن‌ها عالی کار می‌کند.

## آموزش کار با createDocumentFragment()

یک نود خالی برای درج عناصر Html درون آن. DocumentFragment جزئی از درخت نود صفحه ما نیست. این عنصر بعداً در Html درج خواهد شد و هنگام درج تنها فرزندانش را در نود صفحه مشاهده خواهید کرد. استفاده از DocumentFragment باعث بهبود عملکرد در برخی مرورگرهای قدیمی‌تر می‌شود.

|  |
| --- |
| const fragment = new DocumentFragment(); |

|  |
| --- |
| const element = document.getElementById("app"); // assuming ul exists  const fragment = document.createDocumentFragment();  const browsers = ["Firefox", "Chrome", "Opera", "Safari"];  browsers.forEach((browser) => { const li = document.createElement("li"); li.textContent = browser; fragment.appendChild(li); });  element.appendChild(fragment); |

[createDocumentFragment](webdesign/js/Object/js/createDocumentFragment.js)

## آموزش استفاده از filter

جاوااسکریپت یکی از قدرتمندترین زبان‌های برنامه‌نویسی برای توسعه وب است و متد `filter()` یکی از ابزارهای کاربردی برای کار با آرایه‌ها محسوب می‌شود. در ادامه به طور کامل با نحوه استفاده از این متد آشنا می‌شویم.

### آموزش جامع استفاده از متد filter() در جاوااسکریپت

### مقدمه‌ای بر متد filter()

متد `filter()` یک متد داخلی برای آرایه‌ها در جاوااسکریپت است که برای فیلتر کردن عناصر آرایه بر اساس یک شرط خاص به کار می‌رود. این متد یک آرایه جدید شامل عناصری که شرط مشخص شده را دارند برمی‌گرداند و آرایه اصلی را تغییر نمی‌دهد.

#### ویژگی‌های کلیدی:

تغییر ندادن آرایه اصلی (immutable)

بازگرداندن آرایه جدید با عناصر فیلتر شده

پشتیبانی از تمام مرورگرهای مدرن

این متد بخشی از ویژگی‌های استاندارد ECMAScript 5 (ES5) است و در همه محیط‌های اجرایی جاوااسکریپت از جمله مرورگرها و Node.js در دسترس می‌باشد .

### سینتکس و پارامترها

سینتکس اصلی متد filter() به صورت زیر است:

|  |
| --- |
| let newArray = array.filter(function(element, index, array) {  // شرط انتخاب عنصر  }, thisArg); |

#### پارامترها:

|  |  |  |
| --- | --- | --- |
| پارامتر | توضیح | اختیاری |
| element | عنصر فعلی که پردازش می‌شود | اجباری |
| index | ایندکس عنصر فعلی در آرایه |  |
| array | آرایه اصلی که filter روی آن فراخوانی شده |  |
| thisArg | مقداری که به عنوان this برای تابع استفاده می‌شود |  |

#### مقدار بازگشتی:

یک آرایه جدید شامل تمام عناصری که تابع callback شرط true را برگرداند

اگر هیچ عنصری شرط را برآورده نکند، یک آرایه خالی بازگردانده می‌شود

### موارد استفاده و مثال‌های کاربردی

مثال 1: فیلتر کردن اعداد زوج

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];  const evenNumbers = numbers.filter(number => number % 2 === 0);  console.log(evenNumbers); // خروجی: [2, 4, 6, 8, 10] |

[filterEven](webdesign/js/Object/js/filterEven.js)

مثال 2: فیلتر کردن بر اساس طول رشته

|  |
| --- |
| const words = ['hello', 'world', 'javascript', 'filter', 'code', 'web'];  const longWords = words.filter(word => word.length > 5);  console.log(longWords); // خروجی: ['javascript', 'filter'] |

[filterLength](webdesign/js/Object/js/filterLength.js)

مثال 3: فیلتر کردن اشیاء در آرایه

|  |
| --- |
| const products = [    { name: 'Laptop', price: 1000, category: 'electronics' },    { name: 'Book', price: 15, category: 'education' },    { name: 'Phone', price: 500, category: 'electronics' },    { name: 'Pen', price: 2, category: 'office' },    { name: 'eboard', price: 400, category: 'education' }  ];  const expensiveProducts = products.filter(product => product.price > 100);  console.log(expensiveProducts);  // خروجی: [{ name: 'Laptop', price: 1000, category: 'electronics' },  //         { name: 'Phone', price: 500, category: 'electronics' },  //         {name: 'eboard', price: 400, category: 'education'}]  const electronicProducts = products.filter(product => product.category === 'education');  console.log(electronicProducts);  // خروجی: [{name: 'Book', price: 15, category: 'education'},  //         {name: 'eboard', price: 400, category: 'education'}] |

[filterObject](webdesign/js/Object/js/filterObject.js)

مثال 4: استفاده از index در فیلتر کردن

|  |
| --- |
| const numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];  // انتخاب عناصر با ایندکس زوج  const evenIndexNumbers = numbers.filter((number, index) => index % 2 === 0);  console.log(evenIndexNumbers); // خروجی: [0, 2, 4, 6, 8] |

[filterIndex](webdesign/js/Object/js/filterIndex.js)

مثال 5: حذف مقادیر falsy از آرایه

|  |
| --- |
| const mixedValues = [0, 1, false, 2, '', 3, null, undefined, 4, NaN];  const truthyValues = mixedValues.filter(value => value);  // یا به صورت explicit: mixedValues.filter(value => Boolean(value))  console.log(truthyValues); // خروجی: [1, 2, 3, 4] |

[filterfalsy](webdesign/js/Object/js/filterfalsy.js)

### تفاوت filter با سایر متدهای آرایه

مقایسه متدهای رایج آرایه در جاوااسکریپت

|  |  |  |  |
| --- | --- | --- | --- |
| متد | هدف اصلی | مقدار بازگشتی | تغییر آرایه اصلی |
| Filter() | فیلتر کردن عناصر بر اساس شرط | آرایه جدید با عناصر فیلتر شده | خیر |
| Map() | تبدیل هر عنصر به یک عنصر جدید | آرایه جدید با نتایج تابع callback | خیر |
| forEach() | اجرای تابع برای هر عنصر | توسط کاربر تعیین میشود | خیر |
| Find() | یافتن اولین عنصر که شرط را دارد | عنصر یافت شده یا undefined | خیر |
| [Reduce()](#_آموزش_استفاده_از_1) | کاهش آرایه به یک مقدار | مقدار نهایی حاصل از کاهش | خیر |

مثال مقایسه‌ای:

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5];  // filter: فقط اعداد زوج  const filtered = numbers.filter(n => n % 2 === 0); // [2, 4]  // map: تبدیل هر عدد به مربع آن  const mapped = numbers.map(n => n n); // [1, 4, 9, 16, 25]  // find: یافتن اولین عدد بزرگتر از 3  const found = numbers.find(n => n > 3); // 4  // reduce: جمع همه اعداد  const reduced = numbers.reduce((sum, n) => sum + n, 0); // 15 |

[filterVs](webdesign/js/Object/js/filterVs.js)

### بهترین practices و نکات مهم

نکات کلیدی برای استفاده بهینه از filter():

1. عدم تغییر آرایه اصلی: همیشه به یاد داشته باشید که filter() یک آرایه جدید برمی‌گرداند و آرایه اصلی را تغییر نمی‌دهد.

2. استفاده از توابع arrow: برای کدهای مختصر و خوانا، از توابع arrow استفاده کنید.

3. بهینه‌سازی عملکرد: برای آرایه‌های بسیار بزرگ، از پیچیدگی محاسباتی تابع callback آگاه باشید.

4. قابلیت ترکیب: filter() را می‌توان با سایر متدهای آرایه مانند map()و reduce() ترکیب کرد.

5. خوانایی کد: برای شرایط پیچیده، از نام‌های توصیفی برای متغیرها استفاده کنید.

مواردی که باید اجتناب کنید:

1. استفاده از filter() برای side effects: از filter() فقط برای فیلتر کردن استفاده کنید، نه برای اثرات جانبی.

عدم استفاده از filter() برای Side Effects با مثال

مشکل استفاده نادرست از filter() برای Effects جانبی

متد `filter()` در جاوااسکریپت تنها برای فیلتر کردن عناصر آرایه بر اساس یک شرط طراحی شده است. استفاده از آن برای انجام عملیات جانبی (side effects) یک الگوی ضد طراحی (antipattern) محسوب می‌شود.

#### مقایسه استفاده صحیح و نادرست

مثال نادرست استفاده از filter() برای side effects:

|  |
| --- |
| const users = [  { id: 1, name: 'Alice', active: true },  { id: 2, name: 'Bob', active: false },  { id: 3, name: 'Charlie', active: true }  ];  // استفاده نادرست: انجام عملیات جانبی در filter  const activeUsers = users.filter(user => {  console.log(`Checking user: ${user.name}`); // side effect  sendLogToServer(user.id); // side effect خطرناک  return user.active;  });  function sendLogToServer(userId) {  // شبیه‌سازی ارسال به سرور  console.log(`Logging user ${userId} to server...`);  }  // خروجی:  // Checking user: Alice  // Logging user 1 to server...  // Checking user: Bob  // Logging user 2 to server...  // Checking user: Charlie  // Logging user 3 to server...  console.log(activeUsers);  // [{ id: 1, name: 'Alice', active: true }, { id: 3, name: 'Charlie', active: true }] |

[filterSideEffects](webdesign/js/Object/js/filterSideEffects.js)

مثال صحیح جداسازی concerns

|  |
| --- |
| const users = [  { id: 1, name: 'Alice', active: true },  { id: 2, name: 'Bob', active: false },  { id: 3, name: 'Charlie', active: true }  ];  const sendLogToServer = (userId)=>{  console.log(`this user id is ${userId}`);  }  // ابتدا عملیات جانبی را انجام دهید (اگر لازم است)  users.forEach(user => {  console.log(`Checking user: ${user.name}`); // ✅ مناسب برای side effects  sendLogToServer(user.id); // ✅ در جای مناسب  });  // سپس فیلتر کردن را انجام دهید  const activeUsers = users.filter(user => user.active); // ✅ فقط فیلتر کردن  console.log(activeUsers);  // [{ id: 1, name: 'Alice', active: true }, { id: 3, name: 'Charlie', active: true }] |

[filterConcerns](webdesign/js/Object/js/filterConcerns.js)

خطرات استفاده نادرست از filter() برای side effects

1. اجرای مضاعف عملیات

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5];  let count = 0;  // خطرناک: ممکن است چندین بار اجرا شود  const evenNumbers = numbers.filter(num => {  count++; // side effect  console.log(`Processing number: ${num}`);  return num % 2 === 0;  });  console.log(`Total operations: ${count}`); // 5 - اما ممکن است غیرمنتظره باشد |

[filterOver](webdesign/js/Object/js/filterOver.js)

1. مشکلات عملکردی در زنجیره متدها

|  |
| --- |
| // تشخیص منبع side effect سخت می‌شود  const processedData = sourceData.filter(item => {  updateCache(item); // side effect پنهان  validateItem(item); // side effect دیگر  return item.isValid;  }); |

[filterChain](webdesign/js/Object/js/filterChain.js)

1. مشکلات در دیباگ کردن

|  |
| --- |
| const users = [  { id: 1, name: 'Alice', premium: true },  { id: 2, name: 'Bob', premium: false },  { id: 3, name: 'Charlie', premium: true }  ];  const premiumUsers = [];  const logs = [];  // انجام همزمان فیلتر و logging با کنترل کامل  for (const user of users) {  console.log(`Processing user: ${user.name}`);  logs.push(`Processed user ${user.id}`);  if (user.premium) {  premiumUsers.push(user);  }  }  console.log(premiumUsers);  console.log(logs); |

[filterDebug](webdesign/js/Object/js/filterDebug.js)

2. بازگرداندن مقادیر غیرboolean: تابع callback باید مقدار boolean بازگرداند.

3. نادیده گرفتن مقدار بازگشتی: اگر به آرایه جدید نیاز ندارید، شاید forEach() یا for...of مناسب‌تر باشد.

### خلاصه مزایای استفاده از filter():

خوانایی کد: کد شما را تمیزتر و خواناتر می‌کند

عدم تغییرپذیری: آرایه اصلی را تغییر نمی‌دهد

قابلیت ترکیب: با سایر متدهای آرایه ترکیب می‌شود

پشتیبانی گسترده: در تمام محیط‌های اجرایی جاوااسکریپت پشتیبانی می‌شود

یادگیری و تسلط بر متد `filter()` و سایر متدهای آرایه مانند `map()` و `reduce` برای تبدیل شدن به یک توسعه‌دهنده ماهر جاوااسکریپت ضروری است . با تمرین و به کارگیری این مفاهیم در پروژه‌های واقعی، می‌توانید کدهای کارآمدتر و تمیزتری بنویسید.

## آموزش استفاده از reduce

آموزش کامل متد `reduce` در جاوااسکریپت

متد `reduce` یکی از قدرتمندترین متدهای آرایه در جاوااسکریپت است که برای محاسبه یک مقدار بر اساس تمام عناصر آرایه استفاده می‌شود.

ساختار پایه

|  |
| --- |
| array.reduce(callback(accumulator, currentValue, index, array), initialValue) |

accumulator: مقدار انباشته شده از فراخوانی‌های قبلی

currentValue: عنصر فعلی آرایه که در حال پردازش است

index: اندیس عنصر فعلی (اختیاری)

array: آرایه اصلی (اختیاری)

initialValue: مقدار اولیه برای accumulator (اختیاری)

### مثال‌های کاربردی

1. جمع کردن اعداد آرایه

|  |
| --- |
| const numbers = [1, 2, 3, 4, 5];  // بدون مقدار اولیه  const sum = numbers.reduce((acc, curr) => acc + curr);  console.log(sum); // 15  // با مقدار اولیه  const sumWithInitial = numbers.reduce((acc, curr) => acc + curr, 10);  console.log(sumWithInitial); // 25 |

[reduceSum](webdesign/js/Object/js/reduceSum.js)

1. محاسبه حاصل ضرب

|  |
| --- |
| const numbers = [2, 3, 4];  const product = numbers.reduce((acc, curr) => acc curr, 1);  console.log(product); // 24 |

[reduceMultiply](webdesign/js/Object/js/reduceMultiply.js)

1. پیدا کردن بیشترین مقدار

|  |
| --- |
| const numbers = [12, 45, 7, 32, 89, 3];  const max = numbers.reduce((acc, curr) => Math.max(acc, curr));  console.log(max); // 89 |

[reduceMax](webdesign/js/Object/js/reduceMax.js)

1. تبدیل آرایه به شیء

|  |
| --- |
| const fruits = ['apple', 'banana', 'orange'];  const fruitObject = fruits.reduce((acc, curr, index) => {  acc[index] = curr;  return acc;  }, {});  console.log(fruitObject);  // {0: "apple", 1: "banana", 2: "orange"} |

[reduceArrayToObject](webdesign/js/Object/js/reduceArrayToObject.js)

1. شمارش تکرار عناصر

|  |
| --- |
| const words = ['apple', 'banana', 'apple', 'orange', 'banana', 'apple'];  const wordCount = words.reduce((acc, curr) => {  acc[curr] = (acc[curr] || 0) + 1;  return acc;  }, {});  console.log(wordCount);  // {apple: 3, banana: 2, orange: 1} |

[reduceCounter](webdesign/js/Object/js/reduceCounter.js)

1. مسطح کردن آرایه‌های تو در تو

|  |
| --- |
| const nestedArray = [[1, 2], [3, 4], [5, 6]];  const flatArray = nestedArray.reduce((acc, curr) => acc.concat(curr), []);  console.log(flatArray); // [1, 2, 3, 4, 5, 6] |

[reduceFlat](webdesign/js/Object/js/reduceFlat.js)

1. گروه‌بندی اشیا بر اساس ویژگی

|  |
| --- |
| const people = [  { name: 'Alice', age: 25 },  { name: 'Bob', age: 30 },  { name: 'Charlie', age: 25 },  { name: 'David', age: 30 }  ];  const groupedByAge = people.reduce((acc, curr) => {  const age = curr.age;  if (!acc[age]) {  acc[age] = [];  }  acc[age].push(curr);  return acc;  }, {});  console.log(groupedByAge);  /  {  25: [{ name: 'Alice', age: 25 }, { name: 'Charlie', age: 25 }],  30: [{ name: 'Bob', age: 30 }, { name: 'David', age: 30 }]  }  / |

[reduceCat](webdesign/js/Object/js/reduceCat.js)

### تمرین‌ :

#### تمرین 1: محاسبه میانگین

|  |
| --- |
| const scores = [85, 90, 78, 92, 88];  const average = scores.reduce((acc, curr, index, array) => {  acc += curr;  if (index === array.length-1) {  return acc / array.length;  }  return acc;  }, 0);  console.log(average); // 86.6 |

[reduceAverage](webdesign/js/Object/js/reduceAverage.js)

تمرین 2: حذف مقادیر تکراری

|  |
| --- |
| const numbers = [1, 2, 2, 3, 4, 4, 5, 5, 5];  const uniqueNumbers = numbers.reduce((acc, curr) => {  if (!acc.includes(curr)) {  acc.push(curr);  }  return acc;  }, []);  console.log(uniqueNumbers); // [1, 2, 3, 4, 5] |

[reduceRepeat](webdesign/js/Object/js/reduceRepeat.js)

تمرین 3: ترکیب چندین آرایه

|  |
| --- |
| const arrays = [[1, 2], [3, 4], [5, 6]];  const combined = arrays.reduce((acc, curr) => [...acc, ...curr], []);  console.log(combined); // [1, 2, 3, 4, 5, 6] |

[reduceArrayConcat](webdesign/js/Object/js/reduceArrayConcat.js)

تمرین 4: تبدیل آرایه به رشته با فرمت خاص

|  |
| --- |
| // reduceArrayToString  const items = ['apple', 'banana', 'orange'];  const sentence = items.reduce((acc, curr, index) => {  if (index === 0) {  return curr;  } else if (index === items.length-1) {  return `${acc} and ${curr}`;  } else {  return `${acc}, ${curr}`;  }  }, '');  console.log(sentence); // "apple, banana and orange" |

[reduceArrayToString](webdesign/js/Object/js/reduceArrayToString.js)

نکات مهم

1. همیشه مقدار اولیه (`initialValue`) ارائه دهید تا از خطاها جلوگیری شود.

2. بدون مقدار اولیه، reduce از اولین عنصر آرایه به عنوان accumulator اولیه استفاده می‌کند.

3. reduce می‌تواند برای پیاده سازی بسیاری از متدهای دیگر مانند map، filter و find استفاده شود.

جمع‌بندی :

متد `reduce` یک ابزار بسیار قدرتمند و انعطاف‌پذیر برای پردازش آرایه‌ها در جاوااسکریپت است. با تمرین و استفاده از مثال‌های فوق، می‌توانید به خوبی با این متد آشنا شده و از آن در پروژه‌های خود استفاده کنید.

## آموزش استفاده از Rest parameters آموزش استفاده از Spread Operator:

عملگر Spread و Rest در جاوااسکریپت

عملگر `...` که در پارامترهای تابع به صورت `(…fns)` استفاده می‌شود، در جاوااسکریپت به دو صورت مختلف استفاده می‌شود:

### 1. Rest Parameters (پارامترهای rest)

وقتی `...` در تعریف پارامترهای تابع استفاده می‌شود، به آن Rest Parameters می‌گویند. این عملگر تمام آرگومان‌های باقی‌مانده را در یک آرایه جمع می‌کند.

مثال‌هایی از Rest Parameters:

|  |
| --- |
| // جمع کردن تمام آرگومان‌ها  function sum(...numbers) {  return numbers.reduce((total, num) => total + num, 0);  }  console.log(sum(1, 2, 3, 4, 5)); // 15  // ترکیب با پارامترهای عادی  function greet(greeting, ...names) {  return names.map(name => `${greeting}, ${name}!`);  }  console.log(greet('Hello', 'Alice', 'Bob', 'Charlie'));  // ['Hello, Alice!', 'Hello, Bob!', 'Hello, Charlie!']  // در arrow functions  const multiply = (multiplier, ...numbers) => {  return numbers.map(n => n multiplier);  };  console.log(multiply(2, 1, 2, 3)); // [2, 4, 6] |

[RestParameters](webdesign/js/Object/js/RestParameters.js)

### 2. Spread Operator (عملگر spread)

وقتی `...` در فراخوانی تابع یا با آرایه‌ها و آبجکت‌ها استفاده می‌شود، به آن Spread Operator می‌گویند. این عملگر عناصر یک آرایه یا ویژگی‌های یک آبجکت را گسترش می‌دهد.

مثال‌هایی از Spread Operator:

با آرایه‌ها:

|  |
| --- |
| // ترکیب آرایه‌ها  const arr1 = [1, 2, 3];  const arr2 = [4, 5, 6];  const combined = [...arr1, ...arr2];  console.log(combined); // [1, 2, 3, 4, 5, 6]  // کپی آرایه  const original = [1, 2, 3];  const copy = [...original];  console.log(copy); // [1, 2, 3]  // استفاده در فراخوانی تابع  const numbers = [1, 2, 3, 4, 5];  console.log(Math.max(...numbers)); // 5 |

[SpreadOperator](webdesign/js/Object/js/SpreadOperator.js)

با آبجکت‌ها:

|  |
| --- |
| // ترکیب آبجکت‌ها  const obj1 = { a: 1, b: 2 };  const obj2 = { c: 3, d: 4 };  const merged = { ...obj1, ...obj2 };  console.log(merged); // { a: 1, b: 2, c: 3, d: 4 }  // کپی آبجکت  const originalObj = { x: 10, y: 20 };  const copyObj = { ...originalObj };  console.log(copyObj); // { x: 10, y: 20 }  // overwrite properties  const base = { a: 1, b: 2, c: 3 };  const updated = { ...base, b: 20, d: 4 };  console.log(updated); // { a: 1, b: 20, c: 3, d: 4 } |

[SpreadOperatorObject](webdesign/js/Object/js/SpreadOperatorObject.js)

تفاوت‌های کلیدی:

|  |  |  |
| --- | --- | --- |
| ویژگی | Spread Operator | Rest Parameters |
| موقعیت استفاده | در تعریف پارامترهای تابع | در فراخوانی تابع، آرایه‌ها و آبجکت‌ها |
| هدف | جمع‌آوری چندین مقدار در یک آرایه | | گسترش یک آرایه/آبجکت به عناصر/ویژگی‌های جداگانه |
| نوع داده | همیشه یک آرایه ایجاد می‌کند | روی آرایه‌ها، آبجکت‌ها و iterableها کار می‌کند |

مثال‌های ترکیبی:

|  |
| --- |
| // ترکیب Rest و Spread  function processData(first, second, ...rest) {  console.log(`First: ${first}`);  console.log(`Second: ${second}`);  console.log(`Rest: ${rest.join(', ')}`);    // استفاده از Spread برای پردازش بیشتر  const allData = [first, second, ...rest];  return allData.map(item => item 2);  }  const result = processData(1, 2, 3, 4, 5);  console.log(result); // [2, 4, 6, 8, 10]  // در Destructuring  const [first, ...others] = [1, 2, 3, 4, 5];  console.log(first); // 1  console.log(others); // [2, 3, 4, 5]  const { a, ...rest } = { a: 1, b: 2, c: 3, d: 4 };  console.log(a); // 1  console.log(rest); // { b: 2, c: 3, d: 4 } |

[SpreadOperatorRestParameters](webdesign/js/Object/js/SpreadOperatorRestParameters.js)

کاربردهای پیشرفته:

|  |
| --- |
| // کلون عمیق ساده (فقط برای آبجکت‌های ساده)  const original = { a: 1, b: { c: 2 } };  const clone = { ...original, b: { ...original.b } };  // ادغام آبجکت‌ها با ویژگی‌های تکراری  const defaults = { theme: 'light', fontSize: 16 };  const userPreferences = { theme: 'dark' };  const finalConfig = { ...defaults, ...userPreferences };  console.log(finalConfig); // { theme: 'dark', fontSize: 16 }  // ایجاد آرایه بدون مقادیر تکراری  const withDuplicates = [1, 2, 2, 3, 4, 4, 5];  const unique = [...new Set(withDuplicates)];  console.log(unique); // [1, 2, 3, 4, 5] |

[SpreadOperatorPros](webdesign/js/Object/js/SpreadOperatorPros.js)

این عملگر یکی از قدرتمندترین ویژگی‌های ES6+ است که کد را خوانا‌تر و مختصرتر می‌کند.

### آموزش کار با تابع join()

تابع join() از توابع کار با آرایه‌هاست. این متد آرایه را به یک رشته متنی تبدیل میکند و برای اتصال عناصر از کاراکتر خاصی میتواند استفاده کند.

مثال:

|  |
| --- |
| const elements = ["Fire", "Air", "Water"];  console.log(elements.join());  // Expected output: "Fire,Air,Water"  console.log(elements.join(""));  // Expected output: "FireAirWater"  console.log(elements.join("-"));  // Expected output: "Fire-Air-Water" |

[join](webdesign/js/Object/js/join.js)

## آموزش Destructuring Assignment

معرفی Destructuring

Destructuring Assignment یک قابلیت قدرتمند در ES6 است که به شما امکان می‌دهد مقادیر را از آرایه‌ها یا ویژگی‌ها را از آبجکت‌ها استخراج کرده و در متغیرهای جداگانه قرار دهید.

Destructuring آرایه‌ها (Array Destructuring):

مبانی پایه:

|  |
| --- |
| // آرایه سنتی  const numbers = [1, 2, 3];  const a = numbers[0];  const b = numbers[1];  const c = numbers[2];  // با Destructuring  const [x, y, z] = [1, 2, 3];  console.log(x); // 1  console.log(y); // 2  console.log(z); // 3 |

[Destructuring](webdesign/js/Object/js/Destructuring.js)

مقداردهی پیشفرض

|  |
| --- |
| const [a = 10, b = 20, c = 30] = [1, 2];  console.log(a); // 1  console.log(b); // 2  console.log(c); // 30 (پیش‌فرض) |

[DestructuringPreValue](webdesign/js/Object/js/DestructuringPreValue.js)

رد کردن عناصر

|  |
| --- |
| const [first, , third] = [1, 2, 3, 4];  console.log(first); // 1  console.log(third); // 3 |

[DestructuringOmitted](webdesign/js/Object/js/DestructuringOmitted.js)

جمع‌آوری باقی‌مانده‌ها

|  |
| --- |
| const [first, second, ...rest] = [1, 2, 3, 4, 5];  console.log(first); // 1  console.log(second); // 2  console.log(rest); // [3, 4, 5] |

[DestructuringSum](webdesign/js/Object/js/DestructuringSum.js)

تعویض مقادیر (Swap)

|  |
| --- |
| let a = 1;  let b = 2;  // روش سنتی  let temp = a;  a = b;  b = temp;  // با Destructuring  [a, b] = [b, a];  console.log(a); // 2  console.log(b); // 1 |

[DestructuringSwap](webdesign/js/Object/js/DestructuringSwap.js)

### Destructuring آبجکت‌ها (Object Destructuring)

|  |
| --- |
| const person = {  name: 'علی',  age: 30,  city: 'تهران'  };  // روش سنتی  const name = person.name;  const age = person.age;  // با Destructuring  const { name, age, city } = person;  console.log(name); // 'علی'  console.log(age); // 30  console.log(city); // 'تهران' |

[DestructuringObject](webdesign/js/Object/js/DestructuringObject.js)

نام‌گذاری مجدد متغیرها

|  |
| --- |
| const person = {  firstName: 'فاطمه',  lastName: 'احمدی'  };  const { firstName: fname, lastName: lname } = person;  console.log(fname); // 'فاطمه'  console.log(lname); // 'احمدی' |

[DestructuringRename](webdesign/js/Object/js/DestructuringRename.js)

مقادیر پیش‌فرض

|  |
| --- |
| const { name = 'ناشناس', age = 0, city = 'نامشخص' } = { name: 'محمد' };  console.log(name); // 'محمد'  console.log(age); // 0 (پیش‌فرض)  console.log(city); // 'نامشخص' (پیش‌فرض) |

[DestructuringPreValue2](webdesign/js/Object/js/DestructuringPreValue2.js)

ترکیب نام‌گذاری مجدد و پیش‌فرض

|  |
| --- |
| const user = {  username: 'user123'  };  const {  username: uname = 'مهمان',  role: userRole = 'user'  } = user;  console.log(uname); // 'user123'  console.log(userRole); // 'user' (پیش‌فرض) |

[DestructuringPreValue3](webdesign/js/Object/js/DestructuringPreValue3.js)

Destructuring تو در تو

|  |
| --- |
| const company = {  name: 'فناوری اطلاعات',  address: {  city: 'مشهد',  street: 'آزادی',  zip: '12345'  }  };  const {  name,  address: {  city,  street: mainStreet,  zip: postalCode = '00000'  }  } = company;  console.log(name); // 'فناوری اطلاعات'  console.log(city); // 'مشهد'  console.log(mainStreet); // 'آزادی'  console.log(postalCode); // '12345' |

[DestructuringComplex](webdesign/js/Object/js/DestructuringComplex.js)

### کاربردهای عملی Destructuring

#### پارامترهای تابع

|  |
| --- |
| // روش سنتی  function printPerson(person) {  console.log(person.name, person.age);  }  // با Destructuring  function printPerson({ name, age, city = 'نامشخص' }) {  console.log(`${name} - ${age} سال - ${city}`);  }  printPerson({ name: 'علی', age: 30 }); // علی - 30 سال - نامشخص |

[DestructuringParamiter](webdesign/js/Object/js/DestructuringParamiter.js)

1. پاسخ‌های API

|  |
| --- |
| // شبیه‌سازی پاسخ API  const apiResponse = {  status: 'success',  data: {  users: [  { id: 1, name: 'User1' },  { id: 2, name: 'User2' }  ],  total: 2  }  };  // Destructuring برای دسترسی به داده‌ها  const {  status,  data: {  users,  total  }  } = apiResponse;  console.log(status); // 'success'  console.log(users); // [{...}, {...}]  console.log(total); // 2 |

[DestructuringAPi](webdesign/js/Object/js/DestructuringAPi.js)

1. پیکربندی و تنظیمات

|  |
| --- |
| const defaultConfig = {  theme: 'light',  language: 'fa',  notifications: true  };  function initApp(userConfig = {}) {  const {  theme = 'light',  language = 'fa',  notifications = true,  fontSize = 16  } = userConfig;  console.log(`پیکربندی: ${theme}, ${language}, ${fontSize}`);  }  initApp({ theme: 'dark', fontSize: 18 }); |

[DestructuringSetting](webdesign/js/Object/js/DestructuringSetting.js)

Destructuring با آرایه و آبجکت ترکیبی

|  |
| --- |
| const users = [  { name: 'علی', age: 30 },  { name: 'فاطمه', age: 25 },  { name: 'محمد', age: 35 }  ];  // Destructuring در map  const names = users.map(({ name }) => name);  console.log(names); // ['علی', 'فاطمه', 'محمد']  // Destructuring در for-of  for (const { name, age } of users) {  console.log(`${name}: ${age} سال`);  } |

[DestructuringArrayObj](webdesign/js/Object/js/DestructuringArrayObj.js)

آبجکت‌های پیچیده

|  |
| --- |
| const project = {  title: 'وبسایت شرکت',  team: [  { name: 'علی', role: 'developer' },  { name: 'فاطمه', role: 'designer' }  ],  deadline: '2024-12-31'  };  const {  title,  team: [firstMember, secondMember],  deadline: dueDate  } = project;  console.log(title); // 'وبسایت شرکت'  console.log(firstMember); // { name: 'علی', role: 'developer' }  console.log(dueDate); // '2024-12-31' |

[DestructuringComplex2](webdesign/js/Object/js/DestructuringComplex2.js)

نکات مهم و خطاهای رایج

Destructuring آرایه‌های کوتاه:

|  |
| --- |
| const [a, b, c] = [1, 2];  console.log(a); // 1  console.log(b); // 2  console.log(c); // undefined  // با پیش‌فرض  const [x, y, z = 0] = [1, 2];  console.log(z); // 0 |

[DestructuringArray3](webdesign/js/Object/js/DestructuringArray3.js)

نام ویژگی‌های آبجکت

|  |
| --- |
| const obj = { 'first-name': 'علی', 'last-name': 'محمدی' };  // باید از نام واقعی ویژگی استفاده کرد  const { 'first-name': firstName, 'last-name': lastName } = obj;  console.log(firstName); // 'علی' |

[DestructuringObject3](webdesign/js/Object/js/DestructuringObject3.js)

تمرین‌های عملی

تمرین 1: تبدیل مختصات

|  |
| --- |
| const point = [10, 20, 30];  // تبدیل به آبجکت با Destructuring  const [x, y, z] = point;  const pointObj = { x, y, z };  console.log(pointObj); // { x: 10, y: 20, z: 30 } |

[DestructuringRemap](webdesign/js/Object/js/DestructuringRemap.js)

تمرین 2: پردازش کاربر

|  |
| --- |
| const userData = {      personalInfo: {          name: 'زهرا',          family: 'رضایی',          birthDate: '1995-05-15'      },      contact: {          email: 'zahra@example.com',          phone: '09123456789'      }  };  // Destructuring برای استخراج اطلاعات  const {      personalInfo: { name, family },      contact: { email, phone }  } = userData;  console.log(`نام: ${name} ${family}`);  console.log(`تماس: ${email} - ${phone}`); |

[DestructuringUsers](webdesign/js/Object/js/DestructuringUsers.js)

تمرین 3: مدیریت خطا

|  |
| --- |
| function safeDestructuring(obj, path, defaultValue) {      const keys = path.split('.');      let current = obj;        for (const key of keys) {          if (current === null || current === undefined) {              return defaultValue;          }          current = current[key];      }        return current !== undefined ? current : defaultValue;  }  // استفاده  const data = { user: { profile: { name: 'علی' } } };  const userName = safeDestructuring(data, 'user.profile.name', 'ناشناس');  console.log(userName); // 'علی' |

[DestructuringErrorHandling](webdesign/js/Object/js/DestructuringErrorHandling.js)

جمع‌بندی

✅ مزایای Destructuring:

- کد تمیزتر و خواناتر

- کاهش خطوط کد تکراری

- مدیریت آسان پارامترهای اختیاری

- استخراج سریع داده‌های پیچیده

موارد استفاده:

- پارامترهای تابع

- پاسخ‌های API

- پیکربندی‌ها

- کار با آرایه‌ها و آبجکت‌ها

- ایونت هندلینگ

نکات امنیتی:

- همیشه از پیش‌فرض استفاده کنید

- در برابر null/undefined محافظت کنید

- برای ویژگی‌های تودرتو احتیاط کنید

Destructuring یکی از مفیدترین ویژگی‌های مدرن جاوااسکریپت است که به شما کمک می‌کد کدهای تمیزتر و قابل نگهداری‌تری بنویسید!

# Async در جاوا اسکریپت

# Defer در جاوا اسکریپت