



بدون پیش‌نیاز؛ فقط شروع کن

دوره‌ی مقدماتی · مخصوص تازه‌کارها

آموزش مقدماتی فرانت‌اند از اینترنت تا JavaScript

مسیر گام‌به‌گام یادگیری: پایه‌ی برنامه‌نویسی و اینترنت، CSS، HTML و مبانی JavaScript با مثال، فلوچارت و تمرین

مدرس دوره

مجتبی سجادفر

Mojtaba Sajadfar

۴ بخش · مثال و تمرین

JavaScript

CSS

HTML

درباره من



مجتبی سجادفر
Mojtaba Sajadfar

Angular

Ionic

NestJS

OpenLayers

TypeScript

immsr.ir



«من مجتبی سجادفر هستم؛ توسعه‌دهنده نرم‌افزار با علاقه به توسعه رابط کاربری، سامانه‌های مبتنی بر نقشه و ساخت اپلیکیشن‌های وب و موبایل. در این مسیر با فناوری‌هایی مانند Angular، Ionic، NestJS، OpenLayers و TypeScript کار می‌کنم و به یادگیری مستمر، حل مسئله و ساخت تجربه‌های دیجیتال کاربردی علاقه‌مندم.»

نقشه‌های تعاملی

WebGIS



توسعه رابط کاربری

Front-End Development



توسعه اپلیکیشن موبایل

Mobile Applications



توسعه وب‌اپلیکیشن

Web Applications



این دوره برای چه کسی است؟

روش یادگیری



هر مفهوم: توضیح ساده ← شکل ← مثال ← تمرین. کد را خودتان تایپ کنید.

پیش‌نیاز



فقط کار با کامپیوتر و اینترنت در حد کاربر و علاقه به یادگیری.

مخاطب



تازه‌کارهایی که می‌خواهند وارد برنامه‌نویسی و توسعه‌ی وب شوند.

در پایان دوره می‌توانید

۱ توضیح دهید وب چگونه کار می‌کند (Client، Server، HTTP، DNS)

۳ یک صفحه‌ی HTML معنایی و دسترس‌پذیر بنویسید

۵ با JavaScript پایه و DOM صفحه را تعاملی کنید

۲ مسئله را با الگوریتم و فلوچارت تحلیل کنید

۴ با CSS صفحه را استایل و ریسپانسیو کنید (Grid و Flexbox)

۶ داده را با Fetch API از یک API بگیرید و نمایش دهید

توسعه‌دهنده‌ی فرانت‌اند کیست؟

توسعه‌دهنده‌ی فرانت‌اند با HTML، CSS و JavaScript بخش‌های دیدنی و تعاملی وب‌سایت‌ها و اپلیکیشن‌ها را می‌سازد؛ هر دکمه، منو یا انیمیشنی که کاربر می‌بیند و با آن کار می‌کند.



JavaScript



رفتار و پویایی

صفحه را تعاملی می‌کند؛ زبان برنامه‌نویسی اصلی فرانت‌اند.

CSS



ظاهر و چیدمان

رنگ، فونت، فاصله و شکل نمایش عناصر را تغییر می‌دهد.

HTML



اسکلت و محتوای صفحه

هر چیزی که می‌خواهید نمایش دهید، با HTML چیده می‌شود.

کار روزمره: واکنش‌گرا بودن صفحه، دسترس‌پذیری، زیبایی و تجربه‌ی کاربری، و همکاری با طراح، بک‌اند و مدیر پروژه؛ فرانت‌اند کدنویسی واقعی است، نه فقط طراحی.



مهارت‌های فرانت‌اند و جای این دوره

۱ هسته: HTML، CSS و JavaScript

۴ مبانی SEO

۷ RESTful API (آشنایی و Fetch)

۱۰ فریم‌ورک‌ها (React، Angular، Vue)

۲ دسترس‌پذیری (Accessibility)

۵ ابزار توسعه مرورگر (DevTools)

۸ مبانی تجربه‌ی کاربری (UX)

۱۱ عملکرد وب (Web Performance)

۳ طراحی واکنش‌گرا (Responsive)

۶ تست و اشکال‌زدایی (مقدماتی)

۹ Git و کنترل نسخه

۱۲ TypeScript

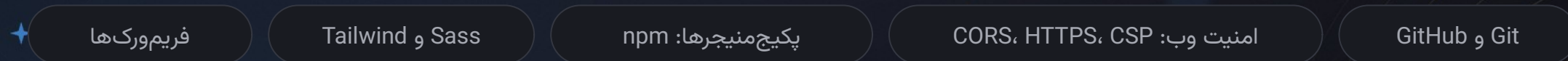
گام‌های بعدی

در این دوره

نقشه‌ی راه این دوره



ادامه‌ی نقشه‌ی راه پس از این دوره



پایه‌ی برنامه‌نویسی و اینترنت

کد چيست، الگوریتم و فلوچارت، و اینکه وب چگونه کار می‌کند

● مرورگر و DevTools

● URL و HTTP

● اینترنت و DNS

● برنامه‌نویسی و الگوریتم

برنامه نویسی چیست؟

برنامه نویسی یعنی نوشتن دستورهای دقیق و مرحله به مرحله برای کامپیوتر تا کاری مشخص را انجام دهد. کامپیوتر خلاقیت ندارد؛ همان چیزی را اجرا می کند که دقیقاً به او گفته ایم.

script.js

```
// یک دستور  
console.log("Hello, World!");
```



خروجی · Console

خروجی نمایشی

Hello, World!

سؤال کلاسی اگر ترتیب دستورها عوض شود، نتیجه ی برنامه چه تغییری می کند؟



دستور



یک فرمان مشخص؛ مثل «دو عدد را جمع کن»

برنامه



مجموعه ای از دستورها با ترتیب مشخص

نرم افزار



محصولی کامل برای کاربر؛ شامل برنامه ها، رابط کاربری و داده ها

ورودی، پردازش، خروجی



```
calculator.js

const a = 7;           // ورودی
const b = 5;           // پردازش
const sum = a + b;     // خروجی: 12
console.log(sum);
```

الگوریتم

ترتیبی مشخص از مراحل برای حل یک مسئله. **برنامه** همان الگوریتم است که با یک زبان برنامه‌نویسی نوشته می‌شود.

فعالیت کلاسی

ورودی، پردازش و خروجی را مشخص کنید.

الگوریتم و ساختارهای فکری آن

Repetition

تکرار



تا وقتی که شرطی برقرار است، مرحله‌ها را تکرار کن.

Selection

شرط



اگر ... آنگاه ... در غیر این صورت ...

Sequence

ترتیب



مراحل یکی پس از دیگری انجام می‌شوند.

شکستن مسئله



مسئله‌ی بزرگ را به مراحل کوچک‌تر تقسیم کنید؛ هر مرحله ورودی و خروجی روشن دارد.

نمونه: پیدا کردن عدد بزرگ‌تر از دو عدد

- ۱ ورودی: دو عدد a و b را بگیر
- ۲ اگر a از b بزرگ‌تر است، a را انتخاب کن
- ۳ در غیر این صورت b را انتخاب کن
- ۴ خروجی: عدد انتخاب‌شده را نمایش بده

شبه‌کد: فکر کردن پیش از کدنویسی

● ● ● pseudocode · معدل سه نمره

START

READ score1, score2, score3

average = (score1 + score2 + score3) / 3

IF average >= 10 THEN

PRINT "قبول"

ELSE

PRINT "مردود"

END IF

END

شبه‌کد چیست؟



توصیف مراحل حل مسئله با زبانی ساده، بدون وابستگی به نحو یک زبان خاص.

یک شبه‌کد خوب



هر مرحله در یک خط
ورودی و خروجی مشخص
شرط و تکرار با ساختار روشن

شبه‌کد زبان رسمی نیست؛ هدفش روشن کردن منطق پیش از نوشتن کد واقعی است.

فعالیت شبه‌کد «زوج یا فرد بودن یک عدد» را بنویسید.



فلوجارت: نمایش مسیر اجرا

نمادها

شروع / پایان

ورودی / خروجی

پردازش

تصمیم (شرط)



فلوچارت با حلقه: بزرگ‌ترین عدد لیست

ردیابی با لیست [۷، ۲، ۹، ۴]

i	لیست[i]	max
۲	۹	۹
۳	۲	۹
۴	۷	۹

شروع

max = اولین عدد لیست

i = ۲

i ≥ تعداد؟

خیر

پایان

نمایش max

بله

لیست[i] < max؟

بله

max = لیست[i]

خیر

i = i + 1

ایده‌ی کلیدی



هر عنصر را با بزرگ‌ترین مقدار دیده‌شده تا اینجا مقایسه می‌کنیم.

وقت تمرین

تمرین: الگوریتم و فلوچارت

۱

شبه‌کدی بنویسید که زوج یا فرد بودن یک عدد را مشخص کند. راهنما: باقی‌مانده‌ی تقسیم بر ۲ را بررسی کنید.

آسان

۲

فلوچارت قیمت نهایی را رسم کنید: اگر خرید بیش از ۱,۰۰۰,۰۰۰ تومان بود، ۱۰٪ تخفیف بدهید. راهنما: یک لوزی برای شرط لازم است.

آسان

۳

الگوریتمی برای جمع اعداد ۱ تا n بنویسید. راهنما: از حلقه و یک متغیر «جمع» استفاده کنید.

متوسط

۴

فلوچارت بررسی رمز عبور را رسم کنید: حداقل ۸ کاراکتر، وگرنه پیام خطا. راهنما: ورودی، شرط و دو خروجی دارید.

متوسط

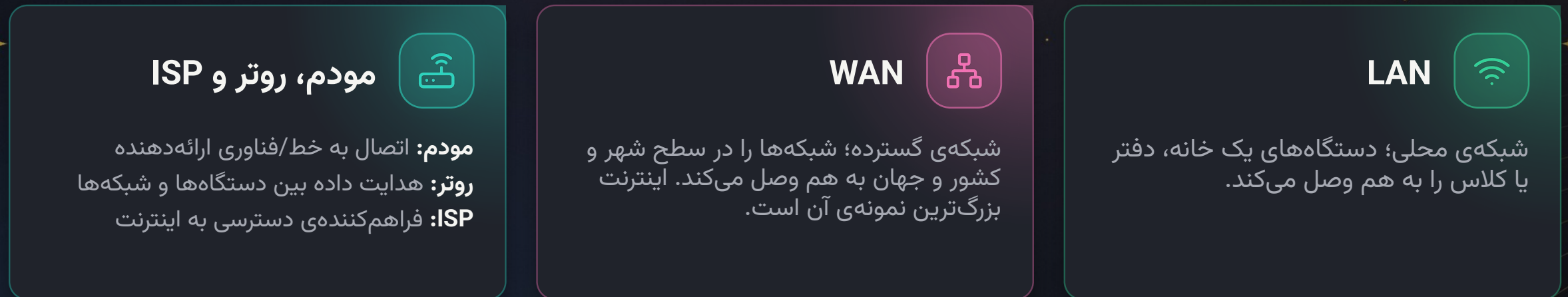
۵

الگوریتمی برای معکوس کردن یک متن بنویسید. راهنما: کاراکترها را از آخر به اول بخوانید.

چالشی

پاسخ‌ها را در بخش ۴ با JavaScript پیاده‌سازی می‌کنیم.

اینترنت: شبکه‌ای از شبکه‌ها



وب (WWW) و تفاوت آن با اینترنت

اینترنت - زیرساخت شبکه‌ی جهانی

وب (World Wide Web)

صفحه‌ها و منابع پیوسته به هم با URL، که با پروتکل HTTP/HTTPS و مرورگر در دسترس‌اند



بازی آنلاین

پروتکل‌های اختصاصی



انتقال فایل

FTP / SFTP



ایمیل

SMTP / IMAP

اینترنت



زیرساخت ارتباطی؛ مسیر و شبکه‌ای که داده از آن عبور می‌کند.

وب



یکی از سرویس‌های روی اینترنت برای دسترسی به صفحه‌ها و منابع.

پیوند میان آن‌ها

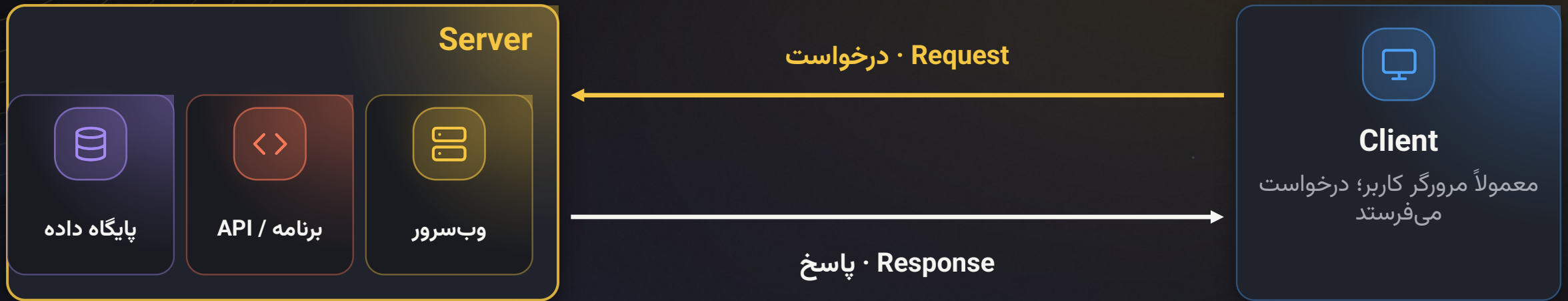


وب بدون اینترنت کار نمی‌کند؛ اما اینترنت فقط وب نیست.

سؤال کلاسی ایمیل و بازی آنلاین از اینترنت استفاده می‌کنند؛ آیا حتماً «وب» هستند؟



Client، Server و معماری Client-Server



معماری Client-Server

یک سرور به چندین کلاینت سرویس می‌دهد؛ کلاینت‌ها درخواست می‌دهند و سرور پاسخ می‌دهد.

وب‌سرور · نرم‌افزار

نرم‌افزاری مانند Nginx یا Apache که درخواست‌های HTTP را می‌گیرد؛ روی یک سرور چند نرم‌افزار اجرا می‌شود.

سرور · دستگاه

کامپیوتر یا ماشین مجازی که معمولاً همیشه روشن و به شبکه متصل است.

آدرس IP و Port

2001:db8::1**IPv6**

نوشتار شانزده‌شانزده‌گانه با «:»؛ ۱۲۸ بیتی و با فضای آدرس بسیار بزرگ‌تر.

203.0.113.10**IPv4**

چهار عدد ۰ تا ۲۵۵ که با نقطه جدا می‌شوند؛ حدود ۴/۳ میلیارد آدرس.

پورتهای رایج

HTTP

80

HTTPS

443

SSH

22

سرور توسعه‌ی محلی

3000

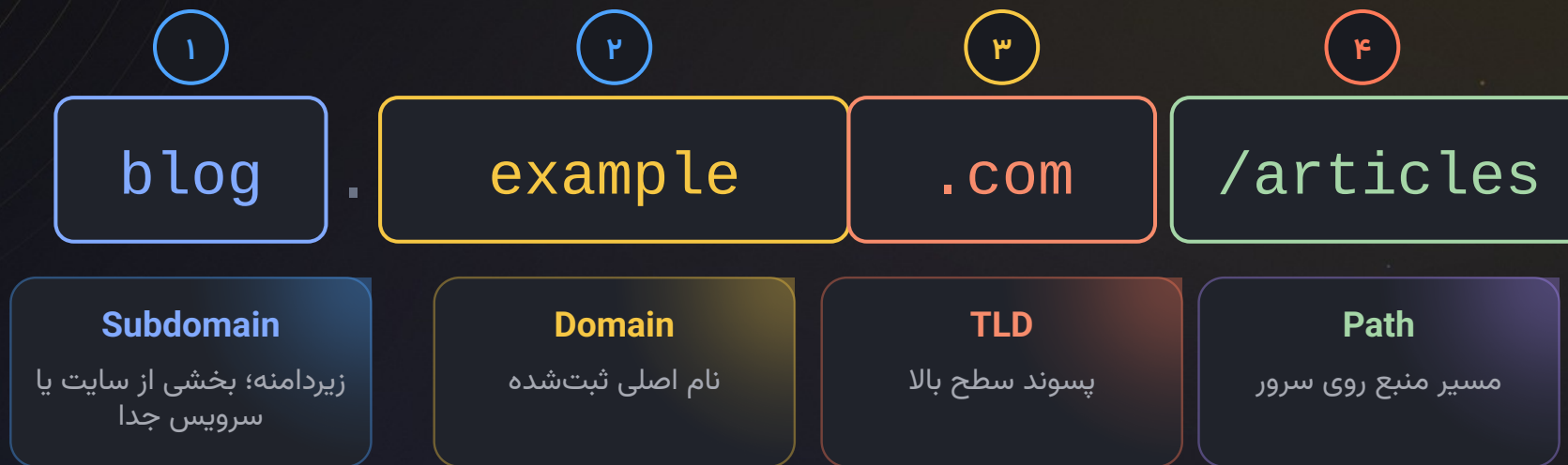
Port: کدام سرویس روی آن دستگاه؟**203.0.113.10**

IP: کدام دستگاه؟

:443

Port: کدام سرویس؟

Hosting و Domain، Subdomain، Path



هاست (Hosting)



فضا و سروری که فایل‌ها و برنامه‌ی سایت روی آن قرار می‌گیرند و به درخواست‌ها پاسخ می‌دهند.

اشاره می‌کند



دامنه (Domain)



نام و آدرسی که کاربر می‌بیند و به خاطر می‌سپارد؛ مثل `example.com`. دامنه با رکوردهای DNS به آدرس سرور اشاره می‌کند.

DNS: تبدیل نام دامنه به آدرس IP

```

bash — nslookup
$ nslookup example.com
Server: 192.168.1.1
Name: example.com
Address: 203.0.113.10
  
```

کش (Cache)

پاسخ‌های قبلی DNS موقتاً در مرورگر و سیستم‌عامل ذخیره می‌شوند تا جست‌وجوی بعدی سریع‌تر شود.

DNS Servers



DNS Resolver



Browser



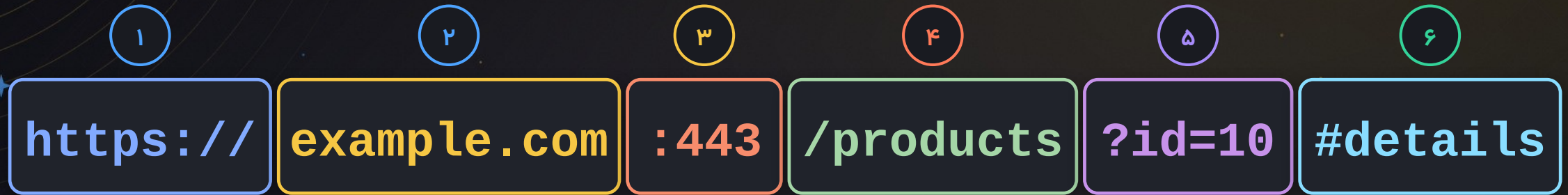
۱ example.com چه IP دارد؟

۲ جست‌وجو در سلسله‌مراتب DNS (اگر در کش نبود)

۳ IP: 203.0.113.10

۴ پاسخ: آدرس IP

اجزای یک URL



● Scheme

پروتکل ارتباط

مرورگر با چه قواعدی با سرور صحبت کند؛ مثل https

● Domain

نام سرور

آدرس خوانا که DNS آن را به IP تبدیل می‌کند

● Port

درگاه سرویس

شماره‌ی سرویس روی سرور؛ ۴۴۳ پیش‌فرض HTTPS است

● Path

مسیر منبع

کدام صفحه یا منبع روی سرور درخواست شده است

● Query String

پارامترها

اطلاعات اضافه به شکل کلید=مقدار، بعد از علامت ؟

● Fragment

بخش درون صفحه

بعد از #؛ فقط برای رفتن به بخشی از صفحه و بدون ارسال به سرور

وقت تمرین

تمرین: URL ها را تجزیه کنید

۱ کدام اجزا وجود دارند؟

url #1

`http://shop.example.org/cart`

۲ کدام اجزا وجود دارند؟

url #2

`https://www.example.com:8080/search?q=html&page=2`

۳ کدام اجزا وجود دارند؟

url #3

`https://docs.example.com/guide/intro#install`

۴ کدام اجزا وجود دارند؟

url #4

`https://example.com/users/42/profile`

نکته اگر Port نوشته نشود، مرورگر پورت پیش فرض پروتکل را به کار می برد: ۸۰ برای HTTP و ۴۴۳ برای HTTPS.



پروتکل، HTTP و HTTPS

پروتکل مجموعه‌ای از قواعد توافق‌شده است که مشخص می‌کند دو طرف ارتباط چگونه پیام‌ها را قالب‌بندی و مبادله کنند.



HTTPS = HTTP + TLS



HTTP



دیده‌شده در مسیر رمز شده

```
17 03 03 00 5a 8f e2 1c b4
0d 77 e9 c3 91 5e a6 3a 2c
f0 d8 41 9b 07 6e 22 d1 85
5c a9 e0 13 7f b8 46 0a d2
```

دیده‌شده در مسیر متن خوانا

```
POST /login HTTP/1.1
Host: example.com

username=ali&password=1234
```

TLS ارتباط را رمزنگاری و دست‌کاری را آشکار می‌کند
هویت سرور با گواهی بررسی می‌شود؛ این به معنی بی‌خطر بودن سایت نیست

داده‌ها بدون رمزنگاری جابه‌جا می‌شوند
در مسیر قابل شنود و دست‌کاری‌اند

ساختار Request و Response

سرور ← کلاینت · Response نمونه ساده شده

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 1256

<html> ... </html>
```

کلاینت ← سرور · Request نمونه ساده شده

```
GET /products?id=10 HTTP/1.1
Host: example.com
Accept: text/html
User-Agent: Mozilla/5.0
```

نسخه، کد وضعیت و عبارت آن

Status line

نوع و اندازه‌ی محتوا، کش و ...

Headers

خود محتوا: JSON، HTML، تصویر و ...

Body

متد (GET)، مسیر و نسخه‌ی پروتکل

خط اول

اطلاعاتی درباره‌ی درخواست و کلاینت

Headers

در GET معمولاً ندارد؛ در POST داده‌ی ارسالی

Body

متدهای HTTP

DELETE

حذف

حذف یک منبع

DELETE /students/7

Delete

PATCH

ویرایش جزئی

تغییر بخشی از یک منبع

PATCH /students/7

Update

PUT

جایگزینی

جایگزینی کامل یک منبع با داده‌ی جدید

PUT /students/7

Replace

POST

ایجاد

ارسال داده برای ایجاد یا پردازش؛ مثلاً ثبت یک مورد جدید

POST /students

Create

GET

دریافت

دریافت اطلاعات؛ نباید داده‌ای را تغییر دهد

GET /students

Read

این‌ها قراردادهای استاندارد HTTP هستند؛ اینکه هر مسیر دقیقاً چه کاری انجام دهد به طراحی API بستگی دارد.



کدهای وضعیت HTTP

400

Bad Request

درخواست نامعتبر یا ناقص است

۴XX · خطای درخواست

302

Found

تغییر مسیر موقت؛ آدرس اصلی همچنان معتبر است

۳XX · تغییر مسیر

301

Moved Permanently

آدرس برای همیشه تغییر کرده؛ به آدرس جدید بروید

۳XX · تغییر مسیر

200

OK

درخواست موفق بود و پاسخ ارسال شد

۲XX · موفق

500

Internal Server Error

خطای غیرمنتظره در سرور

۵XX · خطای سرور

404

Not Found

منبع درخواستی پیدا نشد

۴XX · خطای درخواست

403

Forbidden

هویت مشخص است، اما اجازهی دسترسی ندارید

۴XX · خطای درخواست

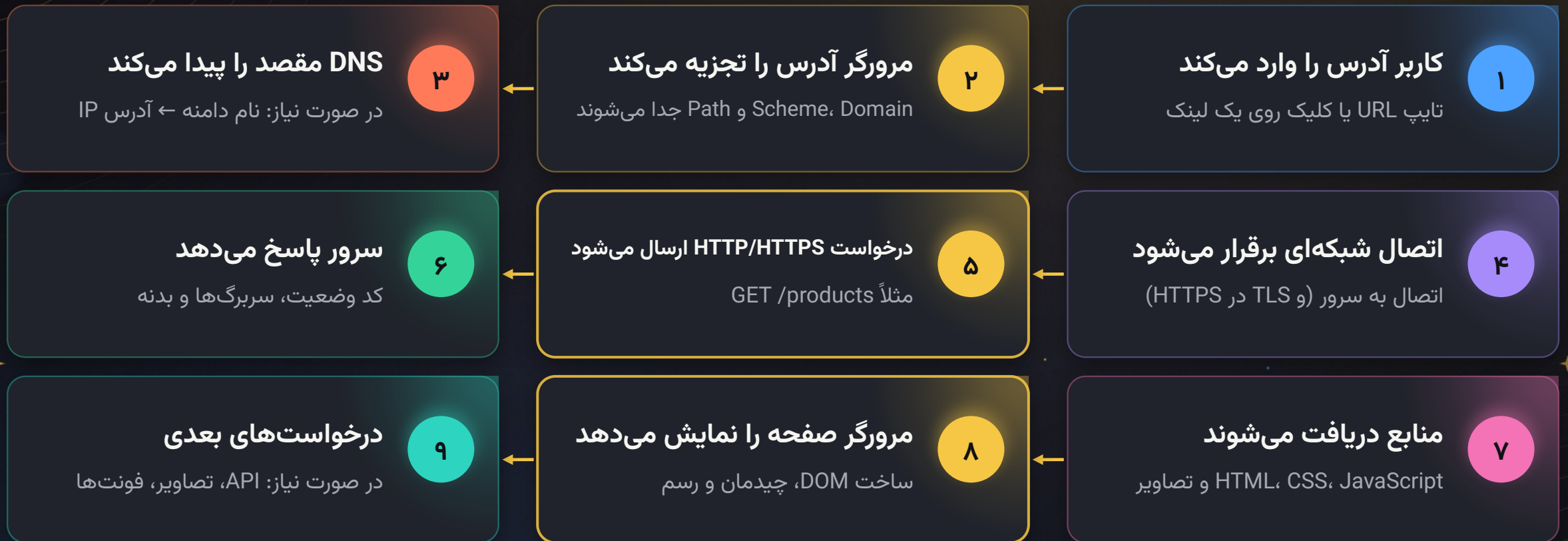
401

Unauthorized

هویت احراز نشده یا نامعتبر است؛ ورود لازم است

۴XX · خطای درخواست

وقتی یک وبسایت باز می‌شود



مدل ساده‌شده: Cache، Redirect، CDN، Service Worker و معماری سرور می‌توانند مسیر واقعی را تغییر دهند.

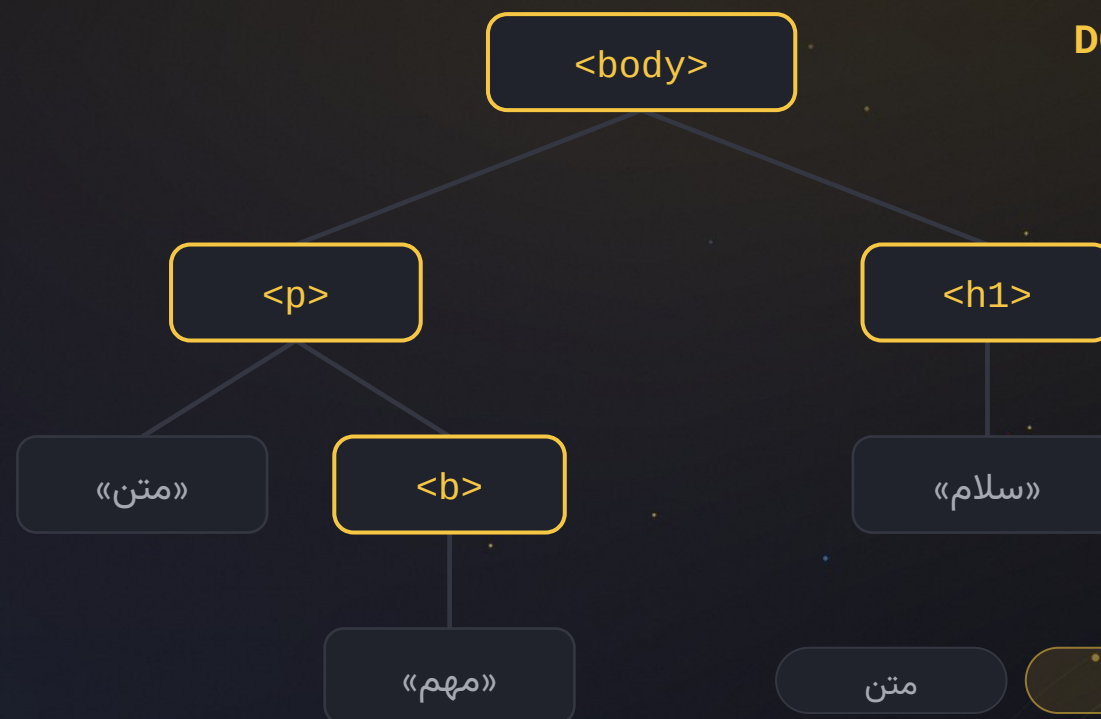
مرورگر چگونه صفحه را نمایش می‌دهد؟



مراحل همیشه کاملاً پشت‌سرهم نیستند: مرورگر هنگام دریافت HTML می‌تواند پردازش را شروع کند، بعضی کارها هم‌پوشانی دارند و تغییر DOM یا استایل می‌تواند Layout و Paint را دوباره اجرا کند.

از HTML تا DOM و Render Tree

درخت DOM



index.html

```

<body>
  <h1>سلام</h1>
  <p>متن <b>مهم</b></p>
</body>
  
```

style.css

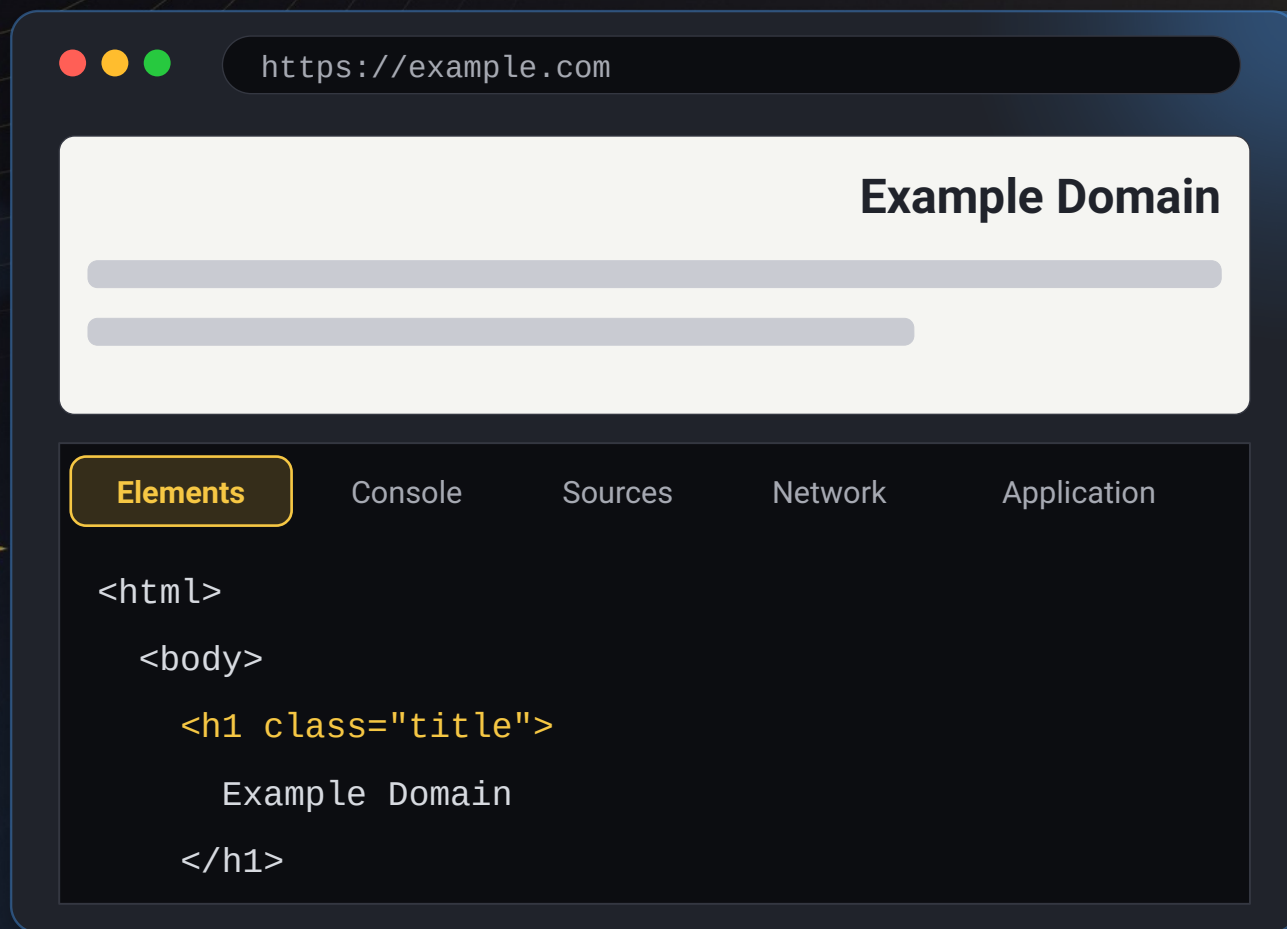
```

h1 { color: #F6C744; }
p { display: none; }
  
```

عنصر p در Render Tree نمی‌آید و رسم نمی‌شود.

DOM ساختار صفحه است؛ CSSOM قواعد استایل؛ و ترکیب دیدنی آن‌ها همان Render Tree.

Developer Tools: بررسی صفحه از داخل



F12

کلید میانبر در بیشتر مرورگرها

Ctrl

+

Shift

+

I

ویندوز و لینوکس

⌘

+

⌥

+

I

macOS

Inspect

راست کلیک روی عنصر و انتخاب

صفحه‌ی وب آنچه کاربر می‌بیند است؛ DevTools ابزار بررسی همان صفحه. تغییرات آن موقتی‌اند.

تب Network: درخواست‌ها و پاسخ‌ها

Name	Method	Status	Type	Time
index.html	GET	200	document	120 ms
style.css	GET	200	stylesheet	45 ms
script.js	GET	200	script	60 ms
hero.jpg	GET	200	jpeg	310 ms
logo.png	GET	404	png	80 ms

داده‌ی نمایشی

منبع بارگذاری نشده: ردیف قرمز، کد ۴۰۴ یا failed را در Status ببینید و Console را هم بررسی کنید.

Headers

سربرگ‌های درخواست و پاسخ

Payload

داده‌ی ارسالی با POST یا PUT

Response

پاسخ خام سرور

Timing

زمان هر مرحله‌ی دریافت

وقت تمرین

مرور و تمرین بخش اول

آسان

۱ تفاوت اینترنت و وب را در دو جمله توضیح دهید.

متوسط

۲ مراحل باز شدن `https://example.com/products?id=10` را به ترتیب بنویسید. راهنما: از URL تا نمایش صفحه.

متوسط

۳ اجزای این URL را مشخص کنید: `https://docs.example.com:8080/guide?lang=fa#intro`
راهنما: Scheme، Domain، Port، Path، Query، Fragment

متوسط

۴ تفاوت کد ۴۰۴ و ۵۰۰ و ۴۰۱ و ۴۰۳ را بگویید. راهنما: خطای کلاینت یا سرور؟

آسان

۵ در یک سایت با DevTools درخواست یک تصویر را پیدا کنید و Status و حجم آن را بخوانید. راهنما: تب Network را باز کنید.

بخش بعدی: HTML، اسکلت و ساختار هر صفحه‌ی وب.

HTML: اسکلت صفحه‌ی وب

از ساختار سند تا فرم‌ها، دسترس‌پذیری و SEO

● دسترس‌پذیری و SEO

● فرم‌ها

● HTML معنایی

● مبانی HTML

HTML چیست؟ تگ، ویژگی و عنصر

```
<p class="intro">سلام دنیا</p>
```

تگ آغازین

نام ویژگی

مقدار ویژگی

محتوا

تگ پایانی

تگ آغازین + ویژگی‌ها + محتوا + تگ پایانی = یک عنصر (Element)

فایل متنی ساده



فایل با پسوند .html در هر ویرایشگر متن قابل نوشتن و در مرورگر قابل اجراست.

تبدیل به DOM



مرورگر HTML را می‌خواند و آن را به یک درخت (DOM) تبدیل می‌کند.

زبان نشانه‌گذاری



HTML برنامه‌نویسی نیست؛ فقط ساختار و معنای محتوا را توصیف می‌کند.

ساختار یک سند HTML

index.html

```
<!DOCTYPE html>
<html lang="fa" dir="rtl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
    content="width=device-width, initial-scale=1">
  <title>صفحه‌ی اول من</title>
</head>
<body>
  <h1>سلام دنیا</h1>
  <p>. این اولین صفحه‌ی من است</p>
</body>
</html>
```

<DOCTYPE html!>

به مرورگر می‌گوید این سند HTML5 است.



dir و lang

زبان و جهت متن؛ برای فارسی fa و rtl.



<head>

اطلاعات صفحه (کدگذاری، عنوان، متا) که در صفحه دیده نمی‌شود.



<body>

محتوای دیدنی صفحه: متن، تصویر، لینک و فرم.



ساختار درختی و تو در تو بودن تگ‌ها

درخت عناصر



درست ✓

`<p>متن <b>مهم</b></p>`

نادرست ✗

`<p>متن <b>مهم</p></b>`

✓ هر تگ باز، بسته شود

تگ‌ها درست تو در تو شوند

T نام تگ‌ها با حروف کوچک

ویژگی‌ها (Attributes) و ویژگی‌های عمومی

index.html

```
<a href="https://example.com"
  title="سایت نمونه"
  target="_blank">
  مشاهده‌ی سایت
</a>
<p id="intro" class="lead big">
  متن مقدمه
</p>
```

index.html

مقدمه‌ی صفحه

[مشاهده‌ی سایت](#)

نتیجه در مرورگر

*-data و title

راهنمای کوتاه و داده‌ی
دلخواه برای JavaScript.



dir و lang

زبان و جهت یک بخش از
صفحه.



class

نام گروه؛ برای چند عنصر و
برای استایل‌دهی.



id

شناسه‌ی یکتا؛ در یک صفحه
فقط برای یک عنصر.



متن: عنوان‌ها، پاراگراف و تأکید

text.html

```
<h1>عنوان اصلی</h1>
<h2>زیرعنوان</h2>
<p>این متن <strong>مهم</strong>
    است <em>تأکیدی</em> و این
<hr>
<p>خط دوم<br>خط اول</p>
<!-- این یک کامنت است -->
```

index.html

عنوان اصلی

زیرعنوان

این متن مهم و این تأکیدی است

خط اول

خط دوم

نتیجه در مرورگر

p، br و hr

پاراگراف، شکستن خط و خط جداکننده.



strong و em

تأکید با معنا؛ برای فقط ظاهر از CSS استفاده کنید.



h1 تا h6

شش سطح عنوان؛ در هر صفحه یک h1 و ترتیب منطقی.



لینک‌ها (Anchor)

links.html

```
<a href="https://example.com">لینک خارجی</a>
<a href="about.html">صفحه‌ی درباره‌ی ما</a>
<a href="#contact">رفتن به بخش تماس</a>
<a href="mailto:me@example.com">ارسال ایمیل</a>
<a href="https://example.com"
  target="_blank" rel="noopener">
  باز شدن در تب جدید</a>
```

index.html

[لینک خارجی](#)[صفحه‌ی درباره‌ی ما](#)[رفتن به بخش تماس](#)[ارسال ایمیل](#)[باز شدن در تب جدید](#)

نتیجه در مرورگر

target و rel

باز شدن در تب جدید همراه
با `rel="noopener"`



لنگر (#)

رفتن به عنصری با id
مشخص در همان صفحه.



نسبی

نسبت به فایل فعلی؛ برای
صفحه‌های همین پروژه.



مطلق

آدرس کامل با `https://`؛
برای سایت‌های دیگر.



تصاویر و ویژگی alt

images.html

```

<figure>
  
  <figcaption>نمودار حضور هفتگی</figcaption>
</figure>
```

index.html



نمودار حضور هفتگی

نتیجه در مرورگر

فرمت‌ها

ابعد را مشخص کنید تا
چیدمان نیپرد.



PNG شفاف؛
JPEG عکس؛
SVG برداری.

height و width

توضیح متنی برای
صفحه‌خوان.



alt

مسیر فایل تصویر؛ نسبی یا
مطلق.



src

توضیح متنی برای
صفحه‌خوان.



لیست‌ها: نامرتب، مرتب و توضیحی

lists.html

```
<ul>
  <li>HTML</li>
  <li>CSS</li>
</ul>
<ol>
  <li>یادگیری HTML</li>
  <li>یادگیری CSS</li>
</ol>
```

index.html

HTML •

CSS •

۱. یادگیری HTML

۲. یادگیری CSS

HTML

نتیجه در مرورگر

زبان نشانه‌گذاری

dl و منوی سایت

dl برای اصطلاح و تعریف؛ منو معمولاً ul داخل nav است.



ol

فهرست مرتب با شماره؛ ترتیب مهم است.



ul و li

فهرست نامرتب؛ هر مورد داخل یک li.



جدول‌ها (Table)

table.html

```
<table>
  <thead>
    <tr><th>نام</th><th>نمره</th></tr>
  </thead>
  <tbody>
    <tr><td>مجتبی</td><td>۲۰</td></tr>
    <tr><td>علی</td><td>۱۸</td></tr>
  </tbody>
</table>
```

index.html

نام	نمره
مجتبی	۲۰
علی	۱۸

نتیجه در مرورگر

نه برای چیدمان

جدول فقط برای داده؛
چیدمان را با CSS بسازید.



rowspan و colspan

ادغام سلول‌ها در ستون یا
سطر.



thead / tbody

بخش‌بندی جدول؛ خوانایی
و دسترس‌پذیری بهتر.



th و td

th سلول عنوان و td سلول
داده است.



عناصر بلوکی و درون خطی

block-inline.html

```
<div>بلوک اول</div>
<div>بلوک دوم</div>
<span>درون خطی ۱</span>
<span>درون خطی ۲</span>
<span>درون خطی ۳</span>
```

بلوکی (Block)

div

div

درون خطی (Inline)

span

span

span

span و div



عنصرهای بدون معنا؛ فقط وقتی عنصر معنایی مناسب نیست از آن‌ها استفاده کنید.

درون خطی



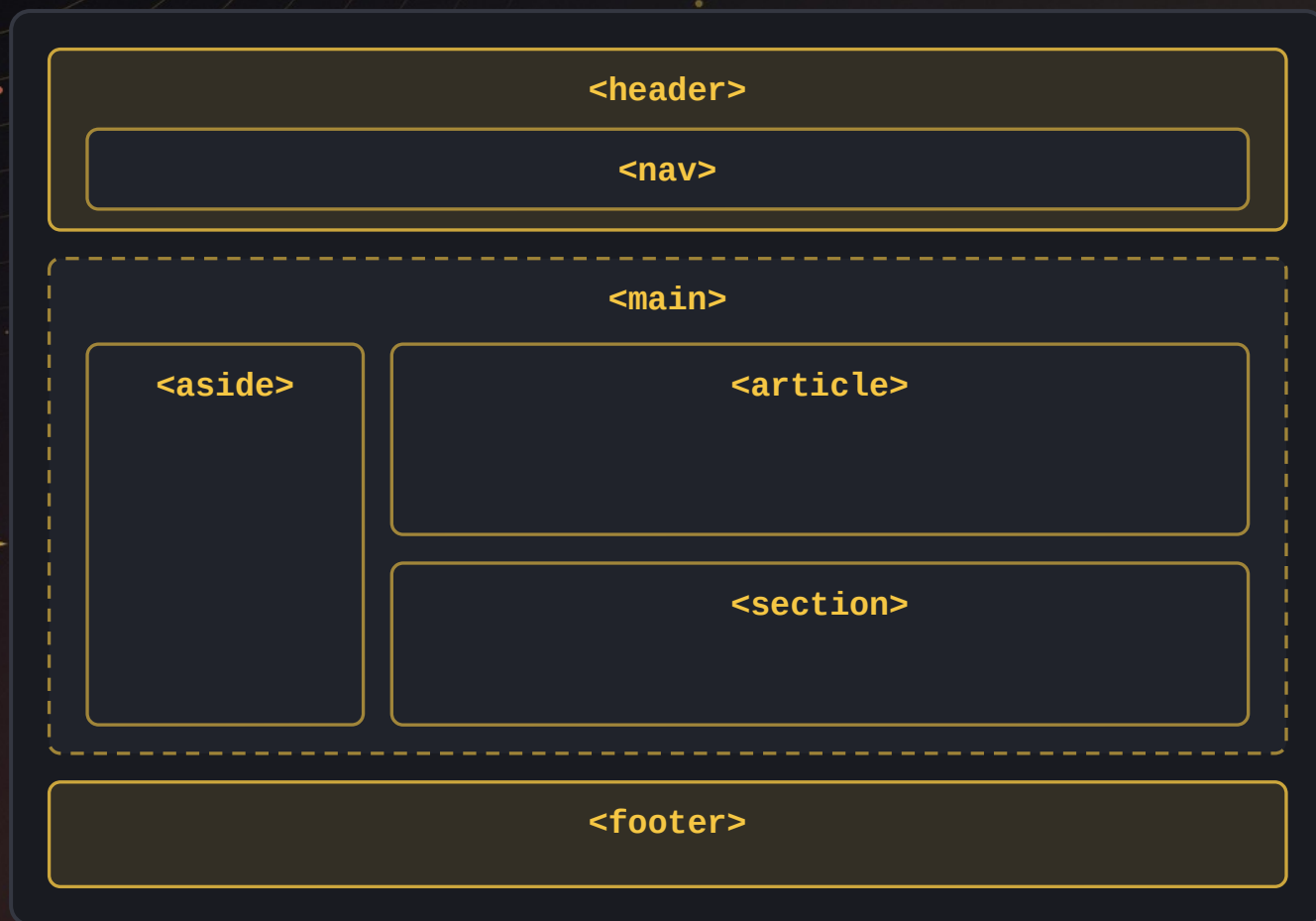
کنار هم قرار می‌گیرد و فقط به اندازه‌ی محتوا جا می‌گیرد: `span`، `a`، `strong`، `img`.

بلوکی



خط جدید می‌گیرد و تمام عرض را پر می‌کند: `div`، `p`، `h1`، `ul`، `section`.

عناصر معنایی: ساختار یک صفحه



سربرگ صفحه یا بخش؛ لوگو و عنوان

`<header>`

بخش لینک‌های اصلی ناوبری

`<nav>`

محتوای اصلی صفحه؛ یکی در هر صفحه

`<main>`

بخش موضوعی با عنوان خودش

`<section>`

محتوای مستقل؛ مثل مقاله یا خبر

`<article>`

محتوای فرعی؛ مثل سایدبار

`<aside>`

پاورقی؛ حقوق و لینک‌های تماس

`<footer>`

معنایی در برابر div های بی معنا

✓ معنایی

```
<header>
  <nav>...</nav>
</header>
<main>
  <article>...</article>
</main>
<footer>...</footer>
```

✗ بدون معنا

```
<div class="header">
  <div class="menu">...</div>
</div>
<div class="content">
  <div class="post">...</div>
</div>
<div class="footer">...</div>
```

خوانایی و نگهداری



کد خودش را توضیح می‌دهد و تغییرش ساده‌تر است.

SEO



موتورهای جست‌وجو ساختار و اهمیت محتوا را بهتر می‌فهمند.

دسترس‌پذیری



صفحه‌خوان‌ها از لندمارک‌ها برای پرش بین بخش‌ها استفاده می‌کنند.

فرم‌ها ۱: ساختار و label

form.html

```
<form action="/register" method="post">
  <label for="name">نام</label>
  <input type="text" id="name" name="name">
  <label for="email">ایمیل</label>
  <input type="email" id="email" name="email">
  <button type="submit">ثبت نام</button>
</form>
```

index.html

button

ارسال می‌کند.
فرم را `type="submit"`



name

فرستاده می‌شود.
کلید داده‌ای که به سرور



for و label

مرتبط با `id` دارد.
هر ورودی یک برچسب



method و action

مقصد و روش ارسال (GET یا POST).



فرم‌ها ۲: انواع کنترل‌ها

۱۴۰۴/۰۱/۱۵

type="date"

انتخاب تاریخ



۱۸

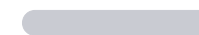
type="number"

عدد؛ min و max

.....

type="password"

رمز عبور؛ مخفی



type="text"

متن کوتاه



<textarea>

متن چندخطی



تهران

<select>

فهرست کشویی



type="radio"

یک انتخاب از چند گزینه



type="checkbox"

چند انتخاب مستقل

اعتبارسنجی فرم در HTML

validation.html

```
<input type="email" name="email" required>
<input type="password" name="pass"
  required minlength="8">
<input type="number" name="age"
  min="18" max="99">
<input type="text" name="code"
  pattern="[0-9]{5}" title="رقم ۵">
```

required پر کردن فیلد اجباری است.

minlength / min / max حداقل طول یا بازه‌ی عددی.

pattern الگوی عبارت باقاعده (Regex).

type=email قالب ایمیل را بررسی می‌کند.



اعتبارسنجی سرور

بررسی نهایی و امنیتی



ارسال به سرور

فقط اگر معتبر بود



اعتبارسنجی مرورگر

required، pattern و ...



کاربر فرم را ارسال می‌کند

کلیک روی دکمه

دسترس پذیری (Accessibility) در HTML

a11y.html

```

<!-- bad -->
<div onclick="send()">ارسال</div>
<!-- good -->
<button type="button">ارسال</button>

<label for="q">جست و جو</label>
<input id="q" type="search">
<button aria-label="بستن منو">x</button>

```

index.html

کنترل پذیری با صفحه کلید

فعال سازی

Enter

جابه جایی

Tab

ارسال

حالت فوکوس باید دیده شود

نتیجه در مرورگر

کنتراست رنگ

حداقل ۴٫۵ به ۱ برای متن
معمولی (WCAG).



صفحه کلید و فوکوس

همه چیز با Tab و Enter
قابل استفاده باشد.



alt و عنوان ها

alt معنادار و عنوان ها به
ترتیب (h1 تا h6).



عنصر درست

button برای دکمه، a برای
لینک، label برای فیلد.



مبانی SEO در HTML

head · seo

```
<head>
  <meta charset="UTF-8">
  <title>آموزش HTML برای مبتدی‌ها</title>
  <meta name="description"
        content="از صفر با مثال و تمرین یادگیری HTML">
  <link rel="canonical"
        href="https://example.com/html">
</head>
```

index.html

example.com > html

آموزش HTML برای مبتدی‌ها

یادگیری HTML از صفر با مثال و تمرین

نمونه‌ی نتیجه‌ی جست‌وجو

نتیجه در مرورگر

موبایل و سرعت

صفحه‌ی ریسپانسیو و سریع، تجربه و رتبه را بهتر می‌کند.



لینک و تصویر

متن لینک توصیفی و alt مناسب برای تصاویر.



عنوان‌ها و معنا

یک h1 و ساختار منطقی با عناصر معنایی.



description و title

عنوان و توضیح کوتاه دقیق برای هر صفحه.



دست به کد شو

پروژه‌ی کوچک ۱: صفحه‌ی رزومه

resume.html

```
<body>
  <header>
    <h1>مجتبی سجادفر</h1>
    <p>توسعه‌دهنده‌ی فرانت‌اند</p>
  </header>
  <main>
    <section>
      <h2>مهارت‌ها</h2>
      <ul><li>HTML</li><li>CSS</li></ul>
    </section>
  </main>
  <footer>sara@example.com</footer>
</body>
```

index.html

مجتبی سجادفر

توسعه‌دهنده‌ی فرانت‌اند

مهارت‌ها

- HTML •
- CSS •

sara@example.com

نتیجه در مرورگر

وقت تمرین

تمرین‌های HTML

آسان

۱ یک صفحه‌ی معرفی شخصی با عنوان، عکس، پاراگراف و لینک به یک سایت بسازید. راهنما: h1، p، a و alt

آسان

۲ برنامه‌ی هفتگی کلاس‌هایتان را در یک جدول با thead و tbody بنویسید. راهنما: th برای روزها و td برای درس‌ها

متوسط

۳ فرم ثبت‌نام با نام، ایمیل، رمز عبور، شهر (select) و تیک پذیرش قوانین بسازید. راهنما: label برای همه‌ی فیلدها و required

متوسط

۴ صفحه‌ای که با div ساخته شده را به عناصر معنایی (header، nav، main، footer) تبدیل کنید. راهنما: ساختار صفحه را اول روی کاغذ بکشید.

متوسط

۵ صفحه‌ی چندبخشی با منوی لینک‌های داخلی (#) بسازید که به بخش‌ها اسکرول کند. راهنما: id برای هر بخش و "href="#id"

چالشی

۶ صفحه‌ی منوی یک رستوران با لیست غذاها، تصویر، جدول قیمت و فرم سفارش بسازید. راهنما: همه‌ی مفاهیم این بخش را ترکیب کنید.

برای بررسی کار: صفحه را در مرورگر باز کنید و با DevTools ساختار DOM را ببینید.

حواست باشه

اشتباهات رایج در HTML

تو در تو کردن نادرست
`<b><i>..</b></i>` نادرست است.



فراموش کردن بستن تگ
 ساختار صفحه به هم می‌ریزد.



ساخت دکمه با `div`
 از `button` استفاده کنید.



نبودن `alt` برای تصویر
 دسترس‌پذیری و SEO ضعیف می‌شود.



نام فایل با فاصله یا حروف بزرگ
 از `my-page.html` استفاده کنید.



چند `h1` و پرش در سطح عنوان‌ها
 ترتیب `h1` تا `h6` را رعایت کنید.



فیلد بدون `label`
 هر ورودی یک `label` مرتبط می‌خواهد.



استفاده از جدول برای چیدمان
 جدول فقط برای داده است.



خلاصه‌ی سریع

خلاصه‌ی HTML (Cheat Sheet)

ریشه‌ی سند، زبان و جهت

<html lang dir>

نوع سند؛ اولین خط

<!DOCTYPE html>

عنوان‌ها

<h1>...<h6>

اطلاعات صفحه / محتوای دیدنی

<head> / <body>

لینک

<a href>

پاراگراف، شکستن خط، خط جداکننده

<p>
 <hr>

لیست‌ها

تصویر با متن جایگزین

عناصر معنایی

<header> <main> <footer>

جدول

<table> <tr> <td>

اعتبارسنجی فرم

required pattern

فرم و کنترل‌ها

<form> <input> <label>

پایه SEO

<title> <meta>

دسترس‌پذیری

alt label aria-*

CSS: ظاهر و چیدمان

از انتخابگرها و Box Model تا Flexbox، Grid و طراحی واکنش‌گرا

● واکنش‌گرا

● Grid و Flexbox

● Box Model

● مبانی CSS

CSS چیست و چگونه اضافه می‌شود؟

```
h1 { color: #F6C744; }
```

انتخابگر

ویژگی

مقدار

اعلان (Declaration)

قاعده (Rule) = انتخابگر + اعلان‌ها؛ هر اعلان یک جفت «ویژگی: مقدار» است

فایل خارجی (External)



```
<link rel="stylesheet" href="style.css">
```

بهترین روش؛ قابل استفاده در چند صفحه.

داخل صفحه (Internal)



```
<style> p { color: red; } </style>
```

برای صفحه‌ی تکی یا نمونه‌های کوتاه.

درون خطی (Inline)



```
<p style="color: red;">
```

فقط برای آزمایش؛ نگهداری سخت است.

انتخابگرها ۱: پایه

● ● ● style.css

```
p { color: #4A4E5A; }
.card { background: white; }
#title { font-size: 28px; }
h1, h2 { font-weight: 700; }
.card p { margin: 0; }
* { box-sizing: border-box; }
```

عنصر · p

همه‌ی عناصر آن تگ.



کلاس · .card

همه‌ی عناصری با "class="card"؛ پرکاربردترین.



شناسه · #title

فقط یک عنصر یکتا؛ در استایل‌دهی کم استفاده شود.



گروه · h1, h2

یک قاعده برای چند انتخابگر.



نوه‌ها · .card p

هر p داخل .card (در هر عمقی).



همه · *

همه‌ی عناصر صفحه.



انتخابگرها ۲: حالت‌ها و ترکیب‌ها

● ● ● style.css

```
a:hover { color: #F6C744; }
input:focus { outline: 2px solid #F6C744; }
li:first-child { font-weight: 700; }
li:nth-child(2n) { background: #eee; }
ul > li { list-style: none; }
h1 + p { margin-top: 0; }
input[type="email"] { direction: ltr; }
```

focus: و hover:

حالت ماوس روی عنصر و فوکوس صفحه‌کلید.



(nth-child: و first-child:

انتخاب بر اساس جایگاه در بین هم‌نوع‌ها.



فرزند مستقیم <

فقط فرزندان بلافاصله، نه همه‌ی نوه‌ها.



همسایه +

عنصری که بلافاصله بعد از عنصر دیگر است.



ویژگی [...=type]

انتخاب بر اساس مقدار یک attribute.



اولویت قاعده‌ها (Specificity) و وراثت

style.css

```
p { color: blue; }
.note { color: green; }
#msg { color: red; }
```

index.html

```
<p id="msg" class="note">سلام</p>
```

نتیجه: متن قرمز است



هر سه قاعده به همان `p` می‌رسند، اما انتخابگر شناسه (#msg) از کلاس و عنصر قوی‌تر است.

درون خطی

۱

روی خود عنصر "..." style="

شناسه

۲

#id

کلاس، شبه‌کلاس و ویژگی

۳

.class :hover [type]

عنصر

۴

p h1 div

وراثت: color و font-* به فرزندانش می‌رسند؛ margin نه.

رنگ‌ها و پس‌زمینه

style.css

```
.a { color: red; }
.b { color: #F6C744; }
.c { color: rgb(246 199 68); }
.d { color: hsl(45 92% 63%); }
.e { background: rgb(0 0 0 / 50%); }
.f { background: linear-gradient(
  to left, #F6C744, #111216); }
```

index.html



hsl()

rgb()

#F6C744

red



گرادیان خطی (نمایش ساده‌شده)

نتیجه در مرورگر

کنتراست

بین متن و پس‌زمینه، تفاوت کافی (۴/۵ به ۱).



hsl

فام، اشباع و روشنایی؛ ساخت سایه‌های هم‌خانواده.



نام، rgb، hex

red، #F6C744 و rgb(246 199 68).



تایپوگرافی: فونت و متن

style.css

```
body {
  font-family: "Vazirmatn", Tahoma, sans-serif;
  font-size: 16px;
  line-height: 1.8;
  color: #20232B;
}
h1 { font-size: 2rem; font-weight: 700; }
p { text-align: justify; }
```

index.html

عنوان اصلی صفحه

این یک پاراگراف نمونه است که با فاصله‌ی خط مناسب
نوشته شده تا خواندن آن آسان باشد؛ ارتفاع خط، خوانایی
متن فارسی را به‌طور محسوسی بهتر می‌کند.

نتیجه در مرورگر

فونت وب

با `font-face@` یا
سرویس فونت اضافه
می‌شود.



text-align

`right`، `center`،
`justify`؛ در RTL شروع
از راست.



font-size و line-height

برای فارسی حدود ۱/۷ تا ۲
مناسب است.



font-family

فهرست فونت‌ها با یک
فونت عمومی آخر.



واحدهای اندازه‌گیری در CSS

vw / vh

درصدی از عرض/ارتفاع
نمایشگر

```
height: 100vh;
```

rem

نسبت به فونت ریشه
(html)

```
font-size: 1.25rem;
```

em

نسبت به فونت همین
عنصر

```
padding: 1.5em;
```

%

نسبت به عنصر والد

```
width: 50%;
```

px

اندازه‌ی ثابت

```
border: 1px solid;
```

نکته

1rem معمولاً ۱۶ پیکسل است؛ **em** با تو در تو شدن می‌تواند انباشته شود.



پیشنهاد برای شروع

اندازه‌ی فونت و فاصله‌ها را با **rem** بنویسید؛ عرض‌ها را با **%** یا **max-width** و ارتفاع صفحه را با **vh**.



Box Model: هر عنصر یک جعبه است



style.css

```
.box {  
  width: 200px;  
  padding: 16px;  
  border: 2px solid #F6C744;  
  margin: 24px;  
  box-sizing: border-box;  
}
```

content-box



عرض کل = ۲۰۰ + ۳۲ + ۴ = ۲۳۶px

border-box



عرض کل همان ۲۰۰px می ماند؛
محاسبه ساده تر است.

margin، padding و border

style.css

```
.a { margin: 10px; }
/* top-bottom left-right */
.b { margin: 10px 20px; }
/* top right bottom left */
.c { padding: 10px 20px 30px 40px; }
.d { width: 300px; margin: 0 auto; }
.e { border: 1px solid #ccc;
    border-radius: 8px; }
```

۱ بالا

۴ چپ

عنصر

۲ راست

۳ پایین

ترتیب ساعت‌گرد

بالا ← راست ← پایین ← چپ؛ با دو مقدار: بالا/پایین و چپ/راست.



margin: 0 auto

عنصر بلوکی با عرض مشخص را افقی وسط می‌گذارد.



border-radius

گرد کردن گوشه‌ها؛ 50% برای دایره.



padding در برابر margin

margin بیرون از border و padding داخل آن است.



ویژگی display و رفتار عناصر

display: none;

دیده نمی‌شود

از چیدمان حذف می‌شود

display: inline-block;



کنار هم و با عرض/ارتفاع

display: inline;



کنار هم؛ عرض/ارتفاع نمی‌پذیرد

display: block;



تمام عرض و خط جدید

overflow

وقتی محتوا از جعبه بیرون می‌زند: **visible**، **hidden**، **scroll** یا **auto** (اسکرول در صورت نیاز).



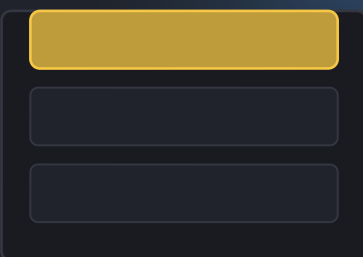
visibility: hidden در برابر **none**

display: none عنصر را از چیدمان حذف می‌کند؛ **visibility: hidden** نامرئی می‌کند ولی جایش می‌ماند.



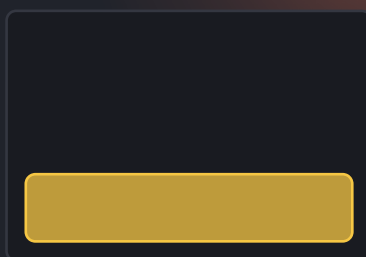
Position: جای‌گذاری دقیق

sticky



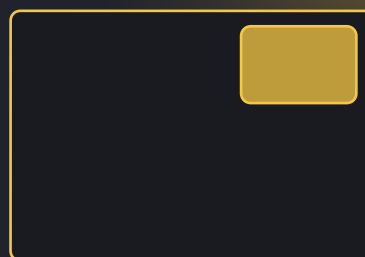
تا رسیدن به آستانه عادی، سپس ثابت.

fixed



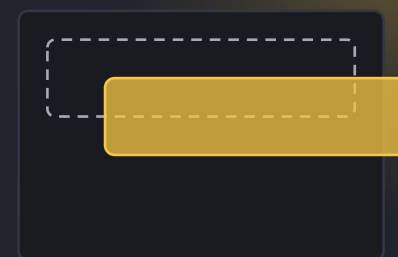
نسبت به پنجره؛ هنگام اسکرول ثابت می‌ماند.

absolute



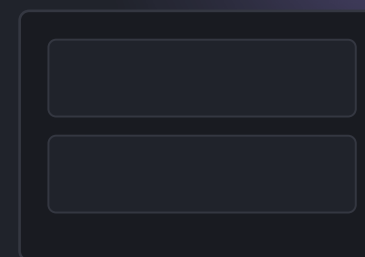
نسبت به نزدیک‌ترین والد دارای position.

relative



نسبت به جای عادی خودش جابه‌جا می‌شود.

static



جریان عادی صفحه؛ left و top اثر ندارند.

style.css

```
.parent { position: relative; }
.badge { position: absolute; top: 8px; left: 8px; }
```

z-index

ترتیب لایه‌ها؛ عدد بزرگ‌تر بالاتر است و فقط روی عناصر دارای position اثر دارد.



۱ Flexbox: مفهوم محور اصلی و متقاطع



style.css

```
.container {
  display: flex;
  flex-direction: row;
  gap: 16px;
}
```

display: flex

والد را کانتینر فلکس می‌کند.



flex-direction

جهت محور اصلی: row یا column.



gap

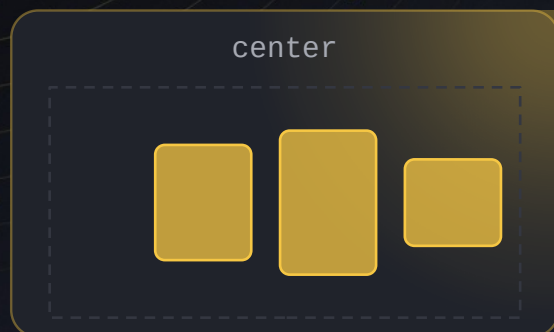
فاصله بین آیتم‌ها بدون margin.



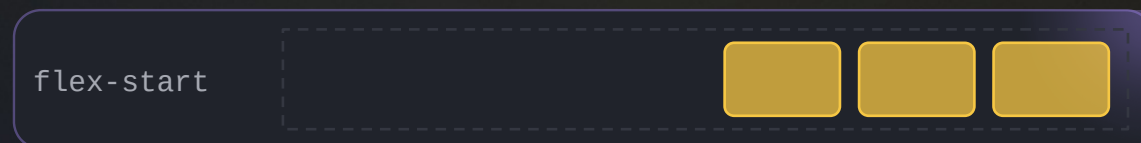
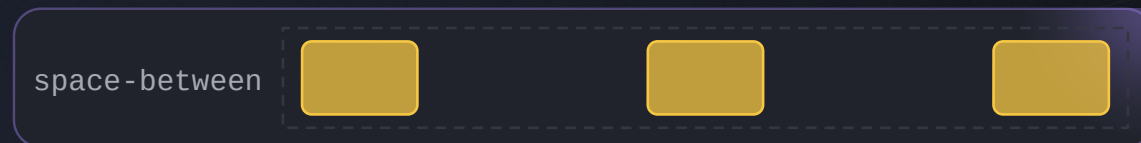
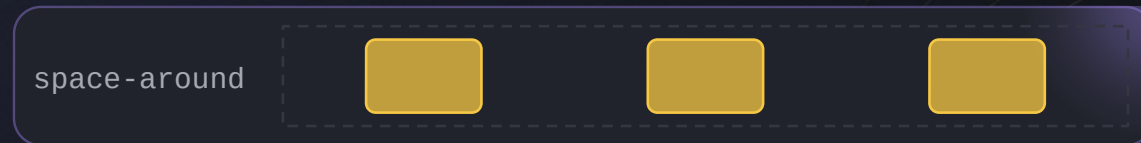
align-items و justify-content: flexbox ۲

align-items · محور متقاطع

justify-content · محور اصلی



در RTL شروع از راست است، پس flex-start یعنی سمت راست.



۳ Flexbox: مثال عملی، نوار ناوبری

style.css

```
nav {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  padding: 12px 24px;  
}  
.menu { display: flex; gap: 16px; }  
.search { flex: 1; }
```

index.html

لوگو خانه محصولات تماس ورود

نتیجه: لوگو در شروع، دکمه در انتها

نتیجه در مرورگر

flex-wrap: wrap

اگر جا کم بود آیتم‌ها به سطر بعد می‌روند.



flex: 1

آیتم تمام فضای باقی‌مانده را می‌گیرد.



center و space-between

توزیع روی محور اصلی و وسط‌چین عمودی.



۱ CSS Grid: سطر و ستون

style.css

```
.grid {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
  grid-template-rows: auto auto;
  gap: 16px;
}
```

index.html



۳ ستون، ۲ سطر

نتیجه در مرورگر

gap

فاصله‌ی بین سطرها و ستون‌ها.

**واحد fr**

سه‌می از فضای باقی‌مانده؛ 1fr 2fr 1fr.

**دوبعدی**

هم سطر و هم ستون را کنترل می‌کند.

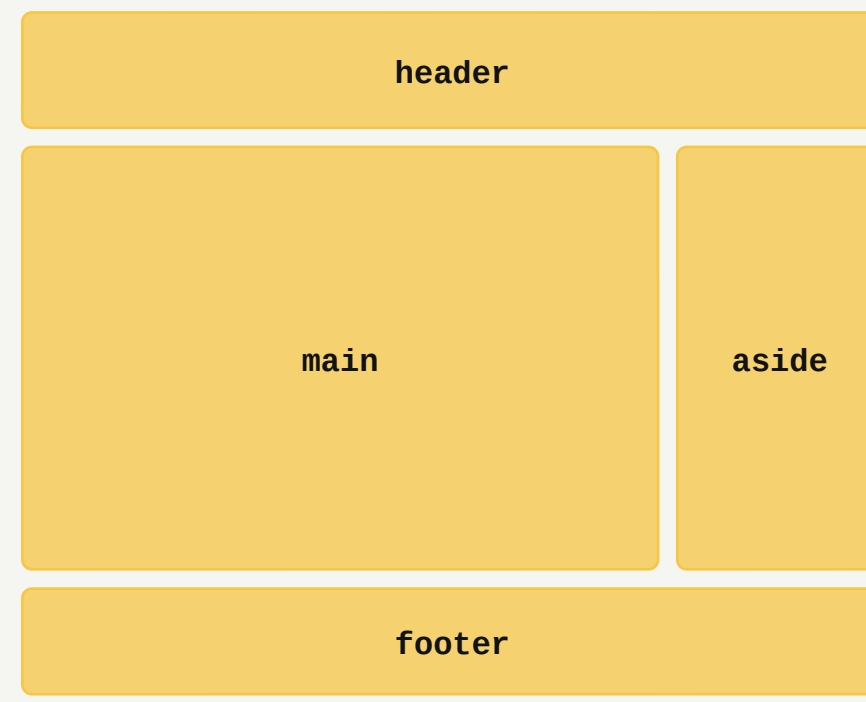


۲ CSS Grid: چیدمان صفحه با areas

style.css

```
.page {  
  display: grid;  
  grid-template-columns: 1fr 3fr;  
  grid-template-areas:  
    "header header"  
    "aside main"  
    "footer footer";  
}  
header { grid-area: header; }  
aside { grid-area: aside; }  
main { grid-area: main; }  
footer { grid-area: footer; }
```

index.html



نتیجه در مرورگر

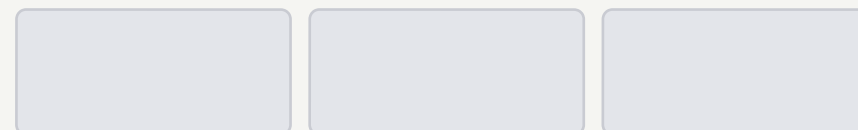
۳ CSS Grid: گالری واکنش‌گرا

style.css

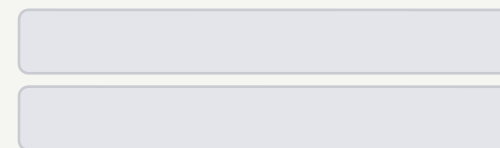
```
.cards {
  display: grid;
  grid-template-columns:
    repeat(auto-fit, minmax(220px, 1fr));
  gap: 16px;
}
```

index.html

صفحه‌ی عریض: ۳ ستون



صفحه‌ی باریک: ۱ ستون



نتیجه در مرورگر

بدون Media Query

چیدمان واکنش‌گرا فقط با یک خط.



minmax(220px, 1fr)

هر ستون حداقل ۲۲۰px و حداکثر سهم برابر.



auto-fit

تعداد ستون‌ها خودکار با فضای موجود.



Flexbox یا Grid؟



یک بعدی: سطر یا ستون

Flexbox



- نوار ناوبری و ابزارها
- وسط چین کردن یک آیتم
- توزیع فضا بین آیتمها

دو بعدی: سطر و ستون با هم

Grid



- چیدمان کلی صفحه
- گالری و کارت‌ها
- ناحیه بندی با areas

وسط چین کردن؛ مسئله‌ی کلاسیک

style.css

```
.a { text-align: center; }
.b { width: 300px; margin: 0 auto; }
.c { display: flex;
    justify-content: center;
    align-items: center; }
.d { display: grid; place-items: center; }
```

index.html

وسط

نتیجه در مرورگر

هر دو جهت · Grid

place-items:
center

هر دو جهت · Flex

justify-content
و align-items

افقی (بلوک)

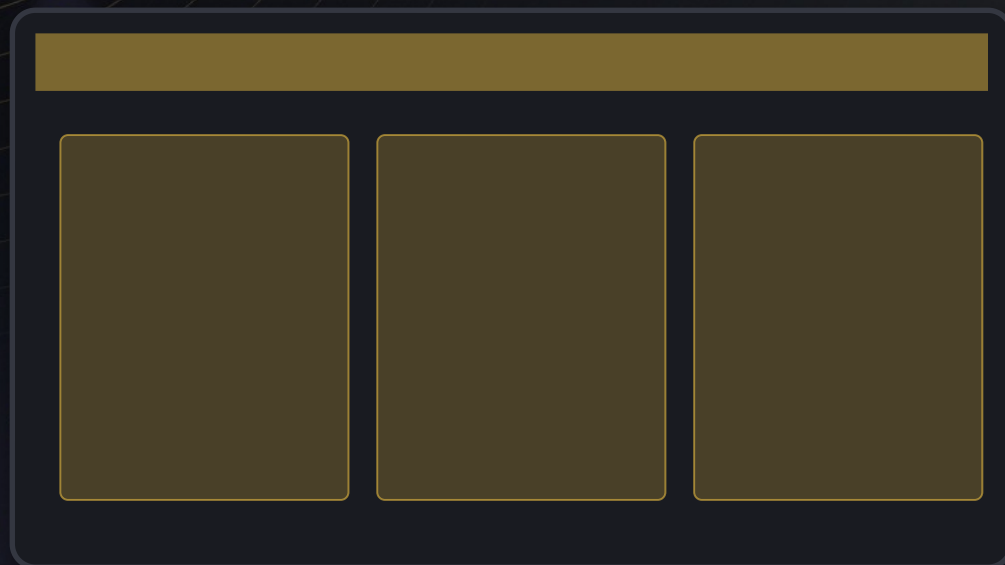
margin: 0 auto
با عرض

افقی (متن)

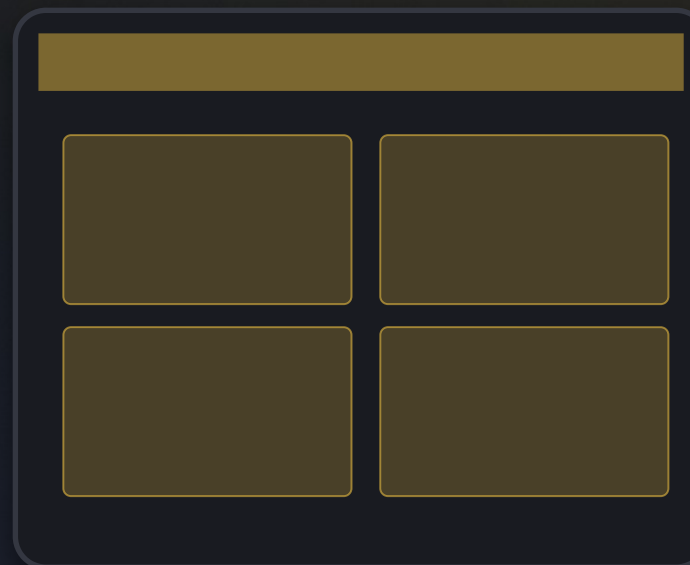
text-align: center



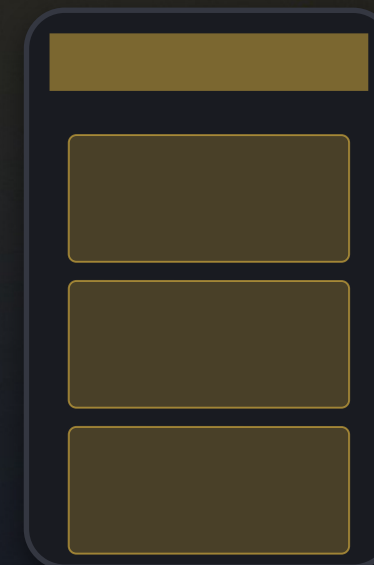
طراحی واکنش‌گرا (Responsive)



دسکتاپ سه ستون



تبلت دو ستون



موبایل یک ستون

آزمایش



DevTools در Device Toolbar

Media Query



قاعده‌های CSS مشروط به اندازه

Mobile-First



از کوچک‌ترین صفحه شروع کنید

Viewport



ناحیه‌ی نمایش صفحه روی دستگاه

Media Query و Viewport, Mobile-First

index.html · head

```
<meta name="viewport"
      content="width=device-width, initial-scale=1">
```

style.css · Mobile-First

```
.cards { display: grid; gap: 16px; }
@media (min-width: 768px) {
  .cards { grid-template-columns: 1fr 1fr; }
}
@media (min-width: 1024px) {
  .cards { grid-template-columns: repeat(3, 1fr); }
}
```

Viewport

عرض صفحه را برابر عرض دستگاه می‌گیرد.



Mobile-First

اول موبایل، سپس گسترش با `min-width`.



Media Query

قاعده‌های مشروط به اندازه‌ی صفحه.



آزمایش در اندازه‌های مختلف

Device Toolbar در DevTools



Media Query عملی: منو در موبایل

style.css

```
.menu {
  display: flex;
  gap: 16px;
}

@media (max-width: 600px) {
  .menu {
    flex-direction: column;
  }
}
```

index.html

عرض بیش از ۶۰۰px

خانه محصولات تماس

عرض $\geq 600px$

خانه

محصولات

تماس

نتیجه در مرورگر

تست



.DevTools در Device Toolbar

نقطه شکست‌های رایج



حدود ۶۰۰، ۷۶۸ و ۱۰۲۴ پیکسل.

min-width یا max-width

max-width کوچک‌تر؛ min-width.
Mobile-First

Animation و Transition، Transform

style.css

```
.btn {  
  transition: transform .2s, background .2s;  
}  
.btn:hover {  
  transform: translateY(-3px) scale(1.05);  
}  
@keyframes pulse {  
  50% { transform: scale(1.15); }  
}  
.dot { animation: pulse 1.5s infinite; }
```

index.html

← با transition نرم

hover

عادی

نتیجه در مرورگر

فریم‌های pulse: کوچک، بزرگ، کوچک

prefers-reduced-motion

برای کاربران حساس، حرکت را کم یا حذف کنید.

**keyframes@**

انیمیشن چندمرحله‌ای با animation.

**transform**

rotate و translate, scale



دست به کد شو

پروژه‌ی کوچک ۲: کارت پروفایل

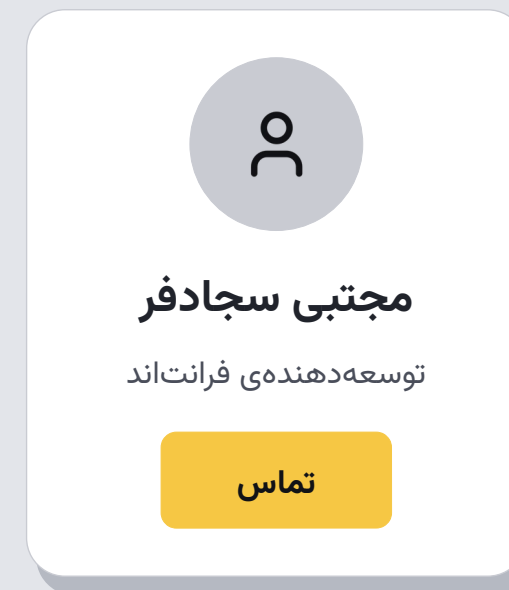
index.html

```
<article class="card">
  
  <h2>مجتبی سجادفر</h2>
  <p>توسعه‌دهنده‌ی فرانت‌اند</p>
  <a class="btn" href="#">تماس</a>
</article>
```

style.css

```
.card { max-width: 280px; padding: 24px;
  text-align: center; background: #fff;
  border-radius: 16px; box-shadow: 0 8px 24px #0003; }
.avatar { width: 96px; border-radius: 50%; }
.btn { padding: 8px 20px; background: #F6C744;
  border-radius: 8px; }
```

index.html



نتیجه در مرورگر

دست به کد شو

پروژه‌ی کوچک ۳: صفحه‌ی فرود ریسپانسیو

۱ ساختار HTML معنایی: header، main، section و footer را بنویسید.

۲ رنگ‌ها و فاصله‌ها را با متغیرهای CSS یا یک قاعده‌ی ساده تعریف کنید.

۳ ابتدا استایل موبایل (یک ستون) را کامل کنید.

۴ کارتها را با Grid بچینید: `auto-fit` و `minmax`.

۵ برای صفحه‌ی بزرگ‌تر با Media Query سه ستون کنید و در DevTools تست کنید.

header

hero

card

card

card

footer

موبایل

header + nav

hero

card

card

card

footer

دسکتاپ

نمونه‌ی ساختار؛ ظاهر دقیق را خودتان طراحی کنید.

چرا استایل من اعمال نمی‌شود؟

ابزارهای DevTools

پنل Styles، Computed و نمایش
Box Model در تب Elements.

نکته

تغییر موقت مقادارها در DevTools را
امتحان کنید و بعد در فایل اصلی اعمال
کنید.



وقت تمرین

تمرین‌های CSS

آسان

صفحه‌ی رزومه‌ی بخش HTML را با رنگ، فونت و فاصله‌ی مناسب استایل کنید. راهنما: color, font-family, line-height, margin

۱

آسان

کارت پروفایل با گوشه‌ی گرد، سایه و دکمه‌ی hover بسازید. راهنما: border-radius, box-shadow, transition

۲

متوسط

یک نوار ناوبری با لوگو، سه لینک و دکمه با Flexbox بسازید. راهنما: align-items و justify-content

۳

متوسط

گالری ۶ کارتی بسازید که با Grid در ۳ ستون و در موبایل در ۱ ستون چیده شود. راهنما: minmax و auto-fit یا Media Query

۴

متوسط

صفحه‌ی فرود پروژه‌ی ۳ را کامل و در سه اندازه‌ی صفحه تست کنید. راهنما: Mobile-First و Device Toolbar

۵

چالشی

یک تم تیره و روشن با متغیرهای CSS بسازید (**var(--bg)**). راهنما: `root` و متغیرها، `prefers-color-scheme`

۶

بعد از هر تمرین، ظاهر صفحه را با DevTools بررسی و ویرایش موقت کنید.

حواست باشه

اشتباهات رایج در CSS

استفاده‌ی زیاد از id و !important

اولویت‌ها را پیچیده می‌کند؛ از کلاس استفاده کنید.



فراموش کردن box-sizing

با border-box محاسبه‌ی عرض ساده‌تر است.



position: absolute بدون والد relative

مرجع جای‌گذاری را نامشخص می‌کند.



اندازه‌های ثابت px برای همه‌چیز

برای واکنش‌گرایی از rem، % و max-width استفاده کنید.



وسط‌چین کردن با روش نادرست

margin: auto برای بلوک؛ text-align برای متن؛ Flex/Grid برای دو جهت.



نبودن meta viewport

صفحه روی موبایل کوچک و ریز دیده می‌شود.



کنتراست کم

خوانایی و دسترس‌پذیری را از بین می‌برد.



غلط املایی در ویژگی یا مقدار

در Styles، خط‌خورده یا هشدار دیده می‌شود.



خلاصه‌ی سریع

خلاصه‌ی CSS (Cheat Sheet)

انتخابگرهای پایه

`.class #id element`

رنگ و پس‌زمینه

`color background`

واحدهای نسبی

`rem % vw vh`

محاسبه‌ی ساده‌ی عرض

`box-sizing: border-box`

توزیع و ترازبندی در Flex

`justify-content align-items`

جای‌گذاری دقیق

`position top left z-index`

حرکت و انیمیشن

`transition transform`

ساختار یک قاعده

`selector { p: v; }`

شبه‌کلاس‌ها

`:hover :focus`

تایپوگرافی

`font-family line-height`

Box Model

`padding border margin`

فعال کردن Flex یا Grid

`display: flex | grid`

ستون‌های Grid

`grid-template-columns`

Media Query

`@media (min-width: ...)`

JavaScript: مبانی برنامه‌نویسی در مرورگر

متغیر، شرط، حلقه، تابع، آرایه، شیء، DOM و Fetch API

Fetch API ●

DOM و رویدادها ●

تابع، آرایه و شیء ●

مبانی زبان ●

JavaScript چیست و کجا اجرا می‌شود؟

JavaScript زبان برنامه‌نویسی صفحه‌های وب است: به رویدادها پاسخ می‌دهد، محتوا را تغییر می‌دهد و با سرور حرف می‌زند. موتور آن داخل مرورگر است (و با Node.js روی سرور هم اجرا می‌شود).



فایل خارجی



```
<script src="app.js" defer></script>
```

بهترین روش؛ کد جدا از HTML.

تگ script در صفحه



```
<script> alert(1); </script>
```

برای نمونه‌های کوتاه.

Console مرورگر



```
console.log(2 + 3);
```

برای آزمایش سریع؛ Console ← F12.

اولین برنامه: خروجی، ورودی و کامنت

script.js

```
// a single-line comment
console.log("Hello, World!");
console.log("سلام دنیا");

/* a multi-line
   comment */
alert("پیام به کاربر");
const name = prompt("نام شما؟");
```

Console · خروجی

نمایشی

Hello, World!
سلام دنیا

console.log()

چاپ در Console؛ ابزار اصلی آزمایش و اشکال زدایی.

>_

alert() و prompt()

نمایش پیام و گرفتن متن از کاربر در یک پنجره.

💬

کامنت‌ها

// برای یک خط و /* */ برای چند خط؛ اجرا نمی‌شوند.

📄

دستورها و ;

هر دستور را معمولاً با سمی‌کولن تمام می‌کنیم.

✓

متغیرها: جعبه‌هایی برای نگه‌داشتن داده

script.js

```
let age = 20;
const name = "Mojtaba";

age = 21;           // allowed
// name = "Ali";    // TypeError: Assignment
//                  // to constant variable.

let userAge = 30;   // camelCase
```

قواعد نام‌گذاری

با حرف، _ یا \$ شروع شود (نه با عدد)
حساس به حروف بزرگ و کوچک است: **age** و **Age** متفاوت‌اند
نام معنادار و **camelCase: userName**
کلمه‌های رزرو مثل **class** یا **if** مجاز نیستند

حافظه: هر متغیر یک نام و یک مقدار دارد

ok

true

name

"Mojtaba"

age

21

نام ← مقدار



var

قدیمی و مستعد خطا؛
پرهیز.



const

مقدار ثابت؛
پیش‌فرض بهتر.



let

مقدار می‌تواند بعداً
تغییر کند.

انواع داده (Data Types)

script.js

```
typeof "salam"    // "string"
typeof 42         // "number"
typeof true       // "boolean"
typeof undefined  // "undefined"
typeof null       // "object" (old quirk)
typeof [1, 2]     // "object"
```

```
const name = "Mojtaba";
console.log(`Hi, ${name}`);
```

نکته‌ی جالب `typeof null` برابر `"object"` است؛ یک ویژگی قدیمی زبان که هنوز اصلاح نشده.

Number

عدد صحیح و اعشاری

42 3.14

String

متن؛ داخل گیومه یا بک‌تیک

"Mojtaba"

null / undefined

«خالی عمدی» و «مقداردهی نشده»

null

Boolean

درست یا نادرست؛ در شرط‌ها

true false

Array و Object {}

فهرست مرتب و مجموعه‌ی کلید-مقدار؛ داده‌های مرجع

[1, 2, 3]

{ age: 25 }

عملگرها: محاسبه و مقایسه

script.js

```
let a = 7, b = 2;
a + b          // 9
a % b          // 1   (remainder)
a ** b         // 49  (power)
a === 7        // true
"5" == 5       // true  (avoid)
"5" === 5      // false
a > 5 && b < 5  // true
!(a > 5)       // false
```

حسابی

++ -- / * % ** و ++ --



مقایسه‌ای

=== != > < >= <=; نتیجه‌ی boolean.



منطقی

&& (و)، || (یا) و ! (نقیض).



=== به جای ==

== نوع را تبدیل می‌کند؛ همیشه از === استفاده کنید.



شرطها: if، else if و else

script.js

```
const age = 20;

if (age >= 18) {
  console.log("adult");
} else if (age >= 13) {
  console.log("teen");
} else {
  console.log("child");
}
```

فقط اولین شاخه‌ی درست اجرا می‌شود؛ ترتیب شرطها مهم است.



switch، عملگر سه‌تایی و Truthy/Falsy

script.js

```
const color = "red";
switch (color) {
  case "red":
    console.log("stop");
    break;
  case "green":
    console.log("go");
    break;
  default:
    console.log("wait");
}
const msg = age >= 18 ? "adult" : "minor";
```

case و switch

مقایسه‌ی یک مقدار با چند حالت؛ **break** را فراموش نکنید.



عملگر سه‌تایی

شرط ؟ الف : ب؛ شکل کوتاه if/else برای مقداردهی.



False

.false، 0، ""، null، undefined، NaN



True

بقیه‌ی مقدارها؛ مثل "0"، [] و {}.



حلقه‌ها: for و while

```

script.js

for (let i = 1; i <= 3; i++) {
  console.log(i);
}

let n = 3;
while (n > 0) {
  console.log(n);
  n--;
}

```

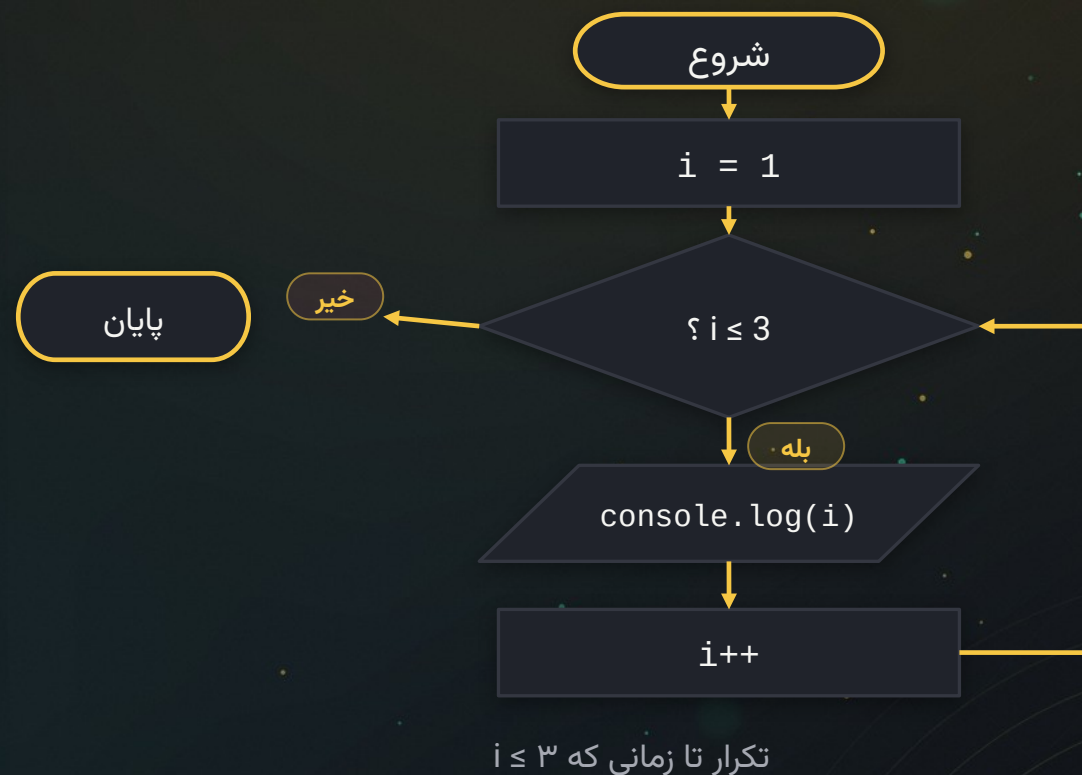
خروجی · Console

نمایشی

```

1 2 3 (for)
3 2 1 (while)

```



تابع‌ها: بلوک‌های قابل‌استفاده‌ی مجدد



script.js

```
function add(a, b) {
  return a + b;
}

const double = (n) => n * 2;

console.log(add(3, 4)); // 7
console.log(double(5)); // 10
```

پارامتر و آرگومان

پارامتر: نام در تعریف؛ آرگومان: مقدار هنگام فراخوانی.

return

مقدار را برمی‌گرداند و تابع را تمام می‌کند.

تابع arrow

شکل کوتاه تابع. $n * 2 \leq (n)$

دامنه‌ی متغیر (Scope)

script.js

```
const g = "global";

function f() {
  const x = "function";
  if (true) {
    let y = "block";
    console.log(g, x, y); // all visible
  }
}
```

قاعده‌ی دید



داخل می‌تواند بیرون را ببیند؛ بیرون داخل را نه.
let و **const** به بلوک `{ }` محدودند.
 متغیر سراسری زیاد، خطر تداخل نام‌ها دارد.

Global Scope

g

Function Scope

g · x

Block Scope

g · x · y

آرایه‌ها: فهرست مرتب از مقادارها

script.js

```
const fruits = ["apple", "banana", "cherry"];  
console.log(fruits[0]);           // "apple"  
console.log(fruits.length);      // 3  
fruits.push("mango");            // add to end  
fruits.pop();                    // remove from end  
fruits[1] = "kiwi";              // replace
```

اندیس از ۰

#

عنصر اول `fruits[0]` است.

length



تعداد عناصر آرایه.

push و pop

+

افزودن و حذف از انتها.

نمودار آرایه (LTR): اندیس از ۰ شروع می‌شود

fruits

0

"apple"

1

"banana"

2

"cherry"

3

"mango"

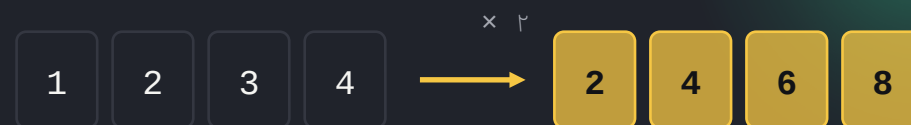
متدهای آرایه: forEach، map، filter، find

script.js

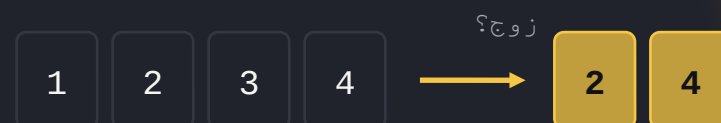
```
const nums = [1, 2, 3, 4];

nums.forEach((n) => console.log(n));
const doubled = nums.map((n) => n * 2);
// [2, 4, 6, 8]
const even = nums.filter((n) => n % 2 === 0);
// [2, 4]
const big = nums.find((n) => n > 2); // 3
nums.includes(3); // true
```

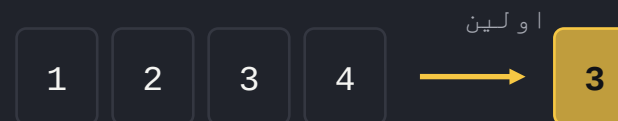
map(n => n * 2)



filter(n => n % 2 === 0)



find(n => n > 2)



map و filter آرایه‌ی جدید می‌سازند و آرایه‌ی اصلی را تغییر نمی‌دهند.

اشیا (Objects) و JSON

script.js

```
const user = {  
  name: "Sara",  
  age: 25,  
  skills: ["HTML", "CSS"],  
  address: { city: "Tabriz" }  
};  
console.log(user.name);           // "Sara"  
console.log(user["age"]);         // 25  
console.log(user.address.city);   // "Tabriz"  
user.age = 26;  
const text = JSON.stringify(user); // object -> JSON  
const back = JSON.parse(text);     // JSON -> object
```

کلید و مقدار

هر ویژگی یک نام (کلید) و یک مقدار دارد.



دسترسی با نقطه یا []

user.name یا user["name"]



JSON

قالب متنی تبادل داده؛ کلیدها با ""

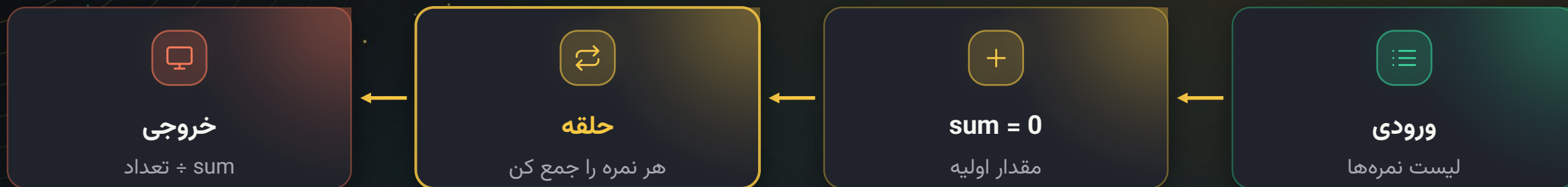


parse و stringify

تبدیل شیء به متن JSON و برعکس.



الگوریتم عملی ۱: میانگین نمرات



● ● ● script.js

```

const scores = [15, 18, 12, 20];
let sum = 0;
for (const s of scores) {
  sum += s;
}
const average = sum / scores.length;
console.log(average); // 16.25
  
```

جدول ردیابی (Trace)

مرحله	s	sum
شروع	—	0
دور ۱	15	15
دور ۲	18	33
دور ۳	12	45
دور ۴	20	65

$$۱۶٫۲۵ = ۴ \div ۶۵$$

پیاده‌سازی الگوریتم‌های بخش اول

● ● ● findMax.js · فلوارت بخش اول

```
function findMax(list) {
  let max = list[0];
  for (let i = 1; i < list.length; i++) {
    if (list[i] > max) {
      max = list[i];
    }
  }
  return max;
}
console.log(findMax([4, 9, 2, 7])); // 9
```

ربط با فلوارت

$\geq i$ همان `max = list[0]` همان «اولین عدد»؛ حلقه‌ی `for` همان لوزی `i` تعداد؛ `if` همان مقایسه.



● ● ● isEven.js

```
const isEven = (n) => n % 2 === 0;
console.log(isEven(8)); // true
```

● ● ● sumTo.js

```
function sumTo(n) {
  let total = 0;
  for (let i = 1; i <= n; i++) total += i;
  return total;
}
```

● ● ● reverse.js

```
const reverse = (s) => s.split('').reverse().join('');
```

وقت تمرین

تمرین‌های مبانی JavaScript

آسان

۱ تابعی بنویسید که دو عدد بگیرد و حاصل جمع، تفاضل، ضرب و تقسیم را در یک شیء برگرداند. راهنما: `return { sum: ..., diff: ... }`

آسان

۲ تابعی بنویسید که بگوید یک سن «کودک»، «نوجوان» یا «بزرگسال» است. راهنما: `if / else if / else`

متوسط

۳ از آرایه‌ی [۱ تا ۱۰] فقط اعداد زوج را جدا و مربع آن‌ها را چاپ کنید. راهنما: `filter` و `map` سپس

متوسط

۴ آرایه‌ای از شیء‌ها (نام و نمره‌ی چند دانشجو) بسازید و میانگین و بهترین نمره را پیدا کنید. راهنما: حلقه و `findMax`

متوسط

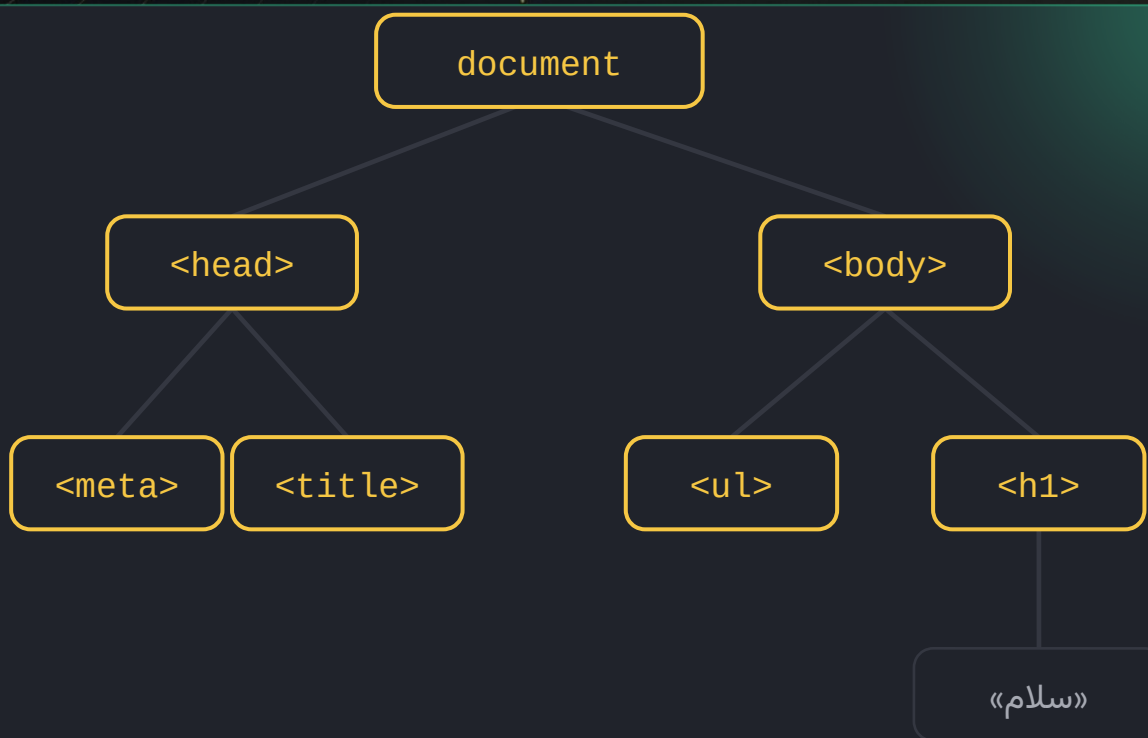
۵ تابعی بنویسید که تعداد حروف صدادار انگلیسی یک متن را بشمارد. راهنما: حلقه روی کاراکترها و `includes`

چالشی

۶ FizzBuzz: اعداد ۱ تا ۱۰۰؛ مضرب ۳ را Fizz، مضرب ۵ را Buzz و مضرب هر دو را FizzBuzz چاپ کنید. راهنما: اول شرط مضرب هر دو

برای هر تمرین ابتدا فلوچارت یا شبه‌کد بکشید و بعد کد بنویسید.

DOM: صفحه به عنوان یک درخت از اشیا



درخت DOM

Console

```

console.log(document.title);
console.log(document.body);
console.log(window.innerWidth);
  
```

document

ورودی دسترسی به صفحه.



خواندن و تغییر

عناصر را بخوانید، تغییر دهید، بسازید و حذف کنید.



window

شیء سراسری مرورگر؛ document زیرمجموعه‌ی آن است.



انتخاب عناصر در DOM

script.js

```
const title = document.getElementById("title");
const btn = document.querySelector(".btn");
const items = document.querySelectorAll("li");

console.log(items.length);
items.forEach((li) => console.log(li.textContent));
```

index.html

```
<h1 id="title">سلام</h1>
<button class="btn">کلیک</button>
<ul><li>الف</li><li>ب</li></ul>
```

getElementById

یک عنصر با شناسه‌ی مشخص.

#

querySelector

اولین عنصر مطابق با انتخابگر CSS؛ اگر نبود `null`.

Q

querySelectorAll

همه‌ی عناصر مطابق؛ فهرستی (NodeList) برای حلقه.

⋮

اجرا بعد از بارگذاری

اسکریپت را با `defer` یا انتهای `body` بیاورید.

⌚

تغییر محتوا، استایل و کلاس

script.js

```
title.textContent = "سلام، دنیا";  
title.style.color = "goldenrod";  
title.classList.add("big");  
title.classList.toggle("hidden");  
btn.disabled = true;  
img.src = "new.jpg";  
img.setAttribute("alt", "تصویر جدید");
```

index.html

قبل

سلام

بعد

سلام، دنیا

کلیک

نتیجه در مرورگر

classList

style را بهتر از **add**، **remove**، **toggle** مستقیم.



innerHTML

با ورودی کاربر خطر XSS دارد؛ پرهیز.



textContent

متن را امن جایگزین می‌کند.



ساخت و حذف عناصر

script.js

```
const list = document.querySelector("ul");

const li = document.createElement("li");
li.textContent = "مورد جدید";
list.append(li);

list.firstElementChild.remove();
```

index.html

فهرست پس از اجرا

مورد اول (حذف شد)

● مورد دوم

● مورد جدید

نتیجه در مرورگر

remove

عنصر را از DOM حذف می‌کند.



prepend و append

افزودن در انتها یا ابتدای والد.



createElement

عنصر جدید می‌سازد، اما هنوز در صفحه نیست.



رویدادها (Events)

script.js

```
const btn = document.querySelector("button");
const input = document.querySelector("input");

btn.addEventListener("click", (event) => {
  console.log("clicked", event.type);
});
input.addEventListener("input", () => {
  console.log(input.value);
});
```

addEventListener

یک تابع را به رویداد وصل می‌کند.



رویدادهای رایج

.click, input, change, submit, keydown



event شیء

اطلاعات رویداد: نوع، هدف و کلید فشرده‌شده.



preventDefault()

رفتار پیش‌فرض (مثل ارسال فرم) را متوقف می‌کند.



رویداد



هندلر



تغییر DOM



دست به کد شو

پروژه‌ی کوچک ۱: شمارنده

index.html

```

<h1 id="count">0</h1>
<button id="inc">+</button>
<button id="dec">-</button>
<script src="app.js" defer></script>

```

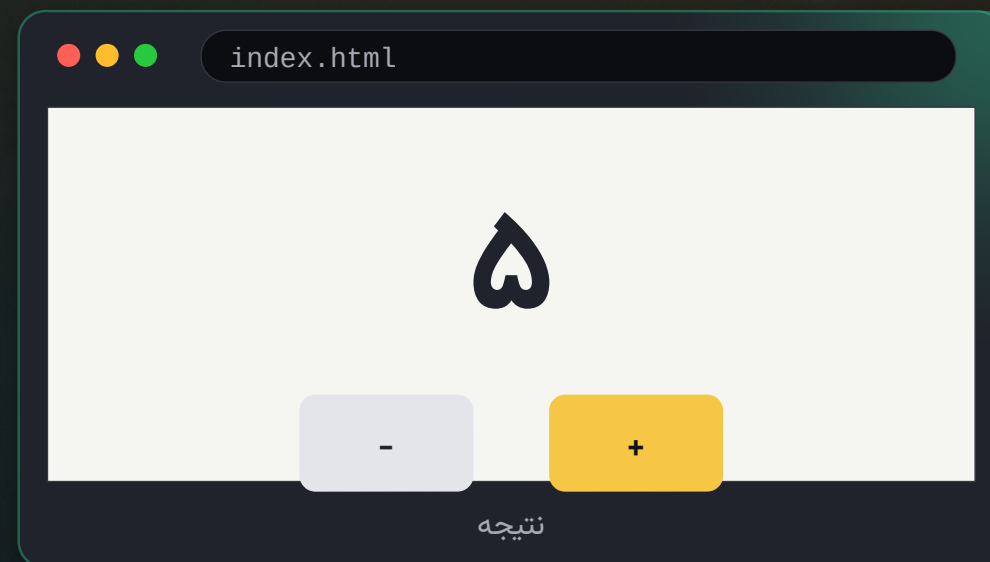
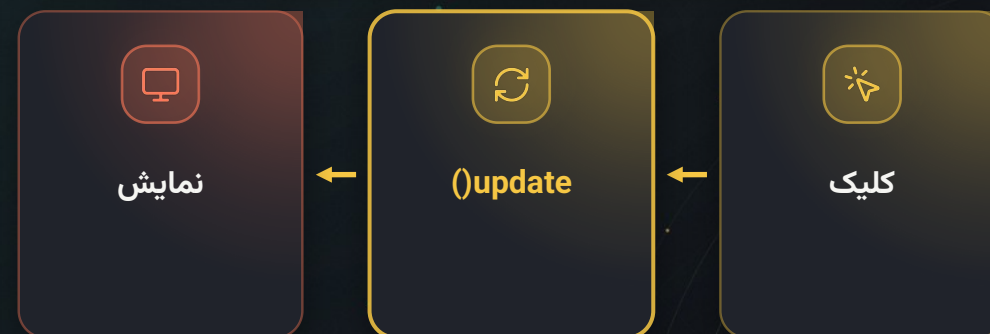
app.js

```

const out = document.querySelector("#count");
let count = 0;

function update(step) {
  count += step;
  out.textContent = count;
}
document.querySelector("#inc").onclick = () => update(1);
document.querySelector("#dec").onclick = () => update(-1);

```



پروژه‌ی کوچک ۲: لیست کارها، الگوریتم

دست به کد شو



index.html

```

<form id="todo-form">
  <input id="todo-input" placeholder="کار جدید">
  <button>افزودن</button>
</form>
<ul id="todo-list"></ul>

```

رویداد submit

به ارسال فرم گوش می‌دهیم.

()preventDefault

از بارگذاری دوباره‌ی صفحه جلوگیری می‌کند.

append و createElement

li را می‌سازیم و به لیست می‌افزاییم.

دست به کد شو

پروژه‌ی کوچک ۲: لیست کارها، کد

app.js

```
const form = document.querySelector("#todo-form");
const input = document.querySelector("#todo-input");
const list = document.querySelector("#todo-list");

form.addEventListener("submit", (e) => {
  e.preventDefault();
  const text = input.value.trim();
  if (text === "") return;

  const li = document.createElement("li");
  li.textContent = text;
  list.append(li);
  input.value = "";
});
```

index.html

افزودن

کار جدید

● خرید نان

● تمرین JavaScript

● مطالعه‌ی HTML

آخرین مورد تازه اضافه شده

نتیجه در مرورگر

اعتبارسنجی فرم با JavaScript

app.js

```
form.addEventListener("submit", (e) => {  
  const email = emailInput.value.trim();  
  const pass = passInput.value;  
  const errors = [];  
  
  if (!email.includes("@")) errors.push("ایمیل نامعتبر");  
  if (pass.length < 8) errors.push("رمز کوتاه است");  
  
  if (errors.length > 0) {  
    e.preventDefault();  
    msg.textContent = errors.join(", ");  
  }  
});
```

آرایه‌ی errors

همه‌ی خطاها را جمع می‌کنیم و یک‌جا نشان می‌دهیم.



()preventDefault

اگر خطا بود، ارسال فرم متوقف می‌شود.



سرور هم بررسی کند

بررسی سمت کلاینت قابل دور زدن است؛ فقط تجربه‌ی کاربری.



Fetch API: گرفتن داده از سرور



app.js

```

async function loadUsers() {
  const res = await fetch("https://jsonplaceholder.typicode.com/users");
  const users = await res.json();
  console.log(users[0].name);
}
loadUsers();
  
```

هر دو await داخل تابع async هستند؛ تا رسیدن پاسخ، صفحه متوقف نمی‌شود.

Promise

نتیجه‌ای که بعداً می‌رسد.



async / await

نوشتن کد ناهمزمان به صورت خوانا.



JSON

قالب رایج پاسخ API؛ با res.json().



نمایش داده‌ی API در صفحه

app.js

```
async function showUsers() {
  try {
    const res = await fetch("https://jsonplaceholder.typicode.com/users");
    if (!res.ok) throw new Error("HTTP " + res.status);
    const users = await res.json();
    list.innerHTML = "";
    users.slice(0, 5).forEach((u) => {
      const li = document.createElement("li");
      li.textContent = u.name;
      list.append(li);
    });
  } catch (err) {
    list.textContent = "خطا در دریافت داده";
  }
}
showUsers();
```

index.html

کاربران

- Leanne Graham ●
- Ervin Howell ●
- Clementine Bauch ●
- Patricia Lebsack ●
- Chelsey Dietrich ●

نمونه‌ی نتیجه

Promise و async/await



● ● ● async / await ✓

```
async function load() {
  try {
    const res = await fetch(url);
    const data = await res.json();
    console.log(data);
  } catch (err) {
    console.error(err);
  }
}
```

● ● ● then / catch

```
fetch(url)
  .then((res) => res.json())
  .then((data) => console.log(data))
  .catch((err) => console.error(err));
```

سه نوع خطا در برنامه نویسی

● ● ● Browser Console

```
console.log("Hello"  
SyntaxError: missing ) after argument list
```

Syntax Error · خطای نحوی

قواعد زبان رعایت نشده؛ اجرا شروع نمی‌شود.



● ● ● Browser Console

```
console.log(age);  
ReferenceError: age is not defined
```

Runtime Error · خطای زمان اجرا

برنامه شروع می‌شود، اما هنگام اجرا متوقف می‌شود.



● ● ● Browser Console

```
const avg = a + b + c / 3;  
// wrong result, no error message
```

Logical Error · خطای منطقی

برنامه اجرا می‌شود، اما نتیجه اشتباه است؛ پیام خطایی دیده نمی‌شود.



اشکال زدایی مرحله به مرحله



تقسیم مسئله به بخش‌های کوچک



هر بخش را جداگانه آزمایش کنید تا محل دقیق خطا مشخص شود.

● ● ● script.js · بررسی گام به گام

```
console.log("a =", a);  
console.log("b =", b);  
console.log("sum =", sum);
```

ویژگی‌های مدرن JavaScript (ES6+)

script.js

```
const name = "Sara";
console.log(`Hello, ${name}`);

const { age, city } = { age: 25, city: "Tabriz" };
const [first, ...rest] = [1, 2, 3];
const merged = [...rest, 4];

// math.js
export const PI = 3.14;
// app.js
import { PI } from "./math.js";
```

Template literal

متن با بک‌تیک و `${...}` برای درج مقدار.

Destructuring

برداشتن چند مقدار از آرایه یا شیء در یک خط.



rest و Spread

پخش کردن آرایه یا شیء و جمع کردن باقی‌مانده.



ماژول‌ها

`import` و `export` در HTML با `type="module"`.

وقت تمرین

تمرین‌ها: DOM، رویدادها و Fetch

آسان

شمارنده را طوری گسترش دهید که عدد از ۰ پایین‌تر نرود و دکمه‌ی «بازنشانی» داشته باشد. راهنما: `update()`

آسان

تغییر تم تیره/روشن با یک دکمه بسازید؛ با کلاس روی `body`. راهنما: `classList.toggle`

متوسط

لیست کارها را کامل کنید: با کلیک روی هر کار، آن را حذف کنید. راهنما: `li.addEventListener` و `remove()`

متوسط

فرم ثبت‌نام را با اعتبارسنجی JavaScript بسازید (ایمیل، رمز ≤ 8 و تکرار رمز). راهنما: آرایه‌ی `errors`

متوسط

ده کاربر را از `jsonplaceholder` بگیرید و در جدول HTML نمایش دهید (نام، ایمیل، شهر). راهنما: `fetch` و `createElement`

چالشی

جست‌وجوی زنده: با تایپ در `input`، لیست کاربران را فیلتر کنید. راهنما: رویداد `input` و `filter`در تمرین‌های Fetch حتماً خطای شبکه را با `try/catch` مدیریت کنید.

حواست باشه

اشتباهات رایج در JavaScript

استفاده از var

از let و const استفاده کنید.



استفاده از == به جای ===

مقایسه با تبدیل نوع نتایج عجیب می‌دهد.



اجرای اسکریپت پیش از بارگذاری HTML

defer بگذارید تا عناصر آماده باشند.



شروع اندیس از ۱

اندیس آرایه از ۰ شروع می‌شود.



فراموش کردن preventDefault در فرم

صفحه دوباره بارگذاری می‌شود.



پیدا نشدن عنصر (null)

انتخابگر را و زمان اجرا را بررسی کنید.



فراموش کردن await

به جای داده، یک Promise می‌گیرید.



innerHTML با ورودی کاربر

خطر XSS؛ textContent بهتر است.



خلاصه‌ی سریع

خلاصه‌ی JavaScript (Cheat Sheet)

انواع داده‌ی پایه

`string number boolean`

شرط

`if / else if / else`

تابع

`function () => {}`

شیء و JSON

`{ key: value }`

تغییر محتوا و ظاهر

`textContent classList`

رویداد

`addEventListener`

مدیریت خطا

`try / catch`

تعریف متغیر

`let const`

مقایسه و منطق

`=== && || !`

حلقه

`for while`

آرایه و متدها

`[ ] push map filter`

انتخاب عنصر

`querySelector`

ساخت و افزودن عنصر

`createElement append`

گرفتن داده‌ی API

`fetch await res.json()`

بعد از این دوره: ادامه‌ی نقشه‌ی راه

۰۳ پکیج‌منیجرها
نصب کتابخانه‌ها برای npm، yarn و pnpm

۰۲ امنیت وب
آشنایی با OWASP، CORS، HTTPS و

۰۵ فریم‌ورک‌ها
Vue یا React، Angular برای برنامه‌های بزرگ

۰۴ Tailwind و Sass
پیش‌پردازنده‌ها و معماری CSS

۰۷ تست و اشکال‌زدایی
تست خودکار و رفع خطای سیستماتیک

۰۶ TypeScript
نوع‌دهی به JavaScript؛ کد مطمئن‌تر

۰۸ عملکرد وب
سرعت بارگذاری و بهینه‌سازی تجربه یادگیری پایان ندارد

از اینجا شروع کن

Git و GitHub

ثبت تاریخچه‌ی تغییرات، کار تیمی و انتشار نمونه‌کار روی وب؛ مهم‌ترین مهارت پس از پایه‌ی HTML، CSS و JavaScript.

- تاریخچه‌ی تغییرات
- کار تیمی
- انتشار نمونه‌کار

برنامه‌ی پیشنهادی ده جلسه‌ای

۱ برنامه‌نویسی، الگوریتم و فلوچارت

تمرین: ۵ مسئله‌ی الگوریتمی

۲ اینترنت، HTTP، DNS و مرورگر

تمرین: تجزیه URL و بررسی Network

۳ HTML ۱: ساختار، متن، لینک، تصویر، لیست، جدول

تمرین: صفحه‌ی رزومه

۴ HTML ۲: معنایی، فرم، دسترس‌پذیری و SEO

تمرین: فرم ثبت‌نام

۵ CSS ۱: انتخابگرها، رنگ، تایپوگرافی و Box Model

تمرین: استایل رزومه

۶ Flexbox و CSS ۲: display، position

تمرین: نوار ناوبری

۷ CSS ۳: Grid، ریسپانسیو و انیمیشن

تمرین: صفحه‌ی فرود

۸ JS ۱: متغیر، شرط، حلقه و تابع

تمرین: الگوریتم‌های بخش اول

۹ JS ۲: آرایه، شیء، DOM و رویدادها

تمرین: شمارنده و لیست کارها

۱۰ JS ۳: Fetch API و پروژه‌ی نهایی

تمرین: پروژه‌ی پایانی

هر جلسه حدود ۲ ساعت؛ مجموعاً حدود ۲۰ ساعت. اگر زمان کم است: جلسه‌های ۱ و ۲ را ادغام و انیمیشن CSS و Fetch را معرفی‌وار مرور کنید.

دست به کد شو

پروژه پایانی: وبسایت شخصی تعاملی



CSS



- چیدمان با Grid و Flexbox
- ریسپانسیو برای موبایل و دسکتاپ
- حالت hover و تم رنگی

HTML



- ساختار معنایی: header, nav, main, footer
- بخش‌های درباره‌ی من، مهارت‌ها (لیست)، تماس (فرم)
- alt برای تصاویر و label برای فیلدها

کیفیت و تحویل



- بدون خطا در Console
- تست در سه اندازه‌ی صفحه
- پوشه‌ی منظم و نام‌گذاری درست

JavaScript



- دکمه‌ی تغییر تم تیره/روشن
- اعتبارسنجی فرم تماس
- نمایش داده‌ی یک API

جمع‌بندی مسیر فرانت‌اند



مجتبی سجادفر

Mojtaba Sajadfar

فرانت‌اند را با ساختن یاد می‌گیرید؛
هر روز کمی کد بزنید و پروژه بسازید.

JavaScript

CSS

HTML

پایه و اینترنت

نمونه‌کار

پروژه‌ی واقعی

فریم‌ورک

Git و GitHub

نمونه‌کارها و ادامه‌ی مسیر را در پرتفولیو ببینید

immsr.ir