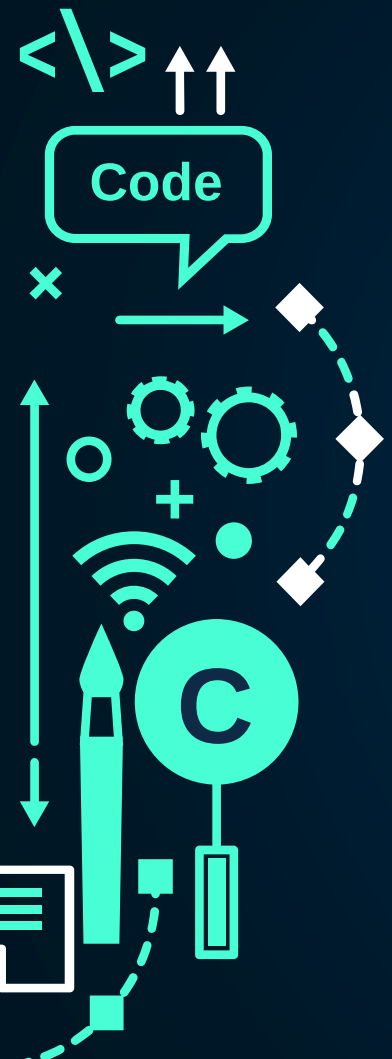


بسم الله الرحمن الرحیم

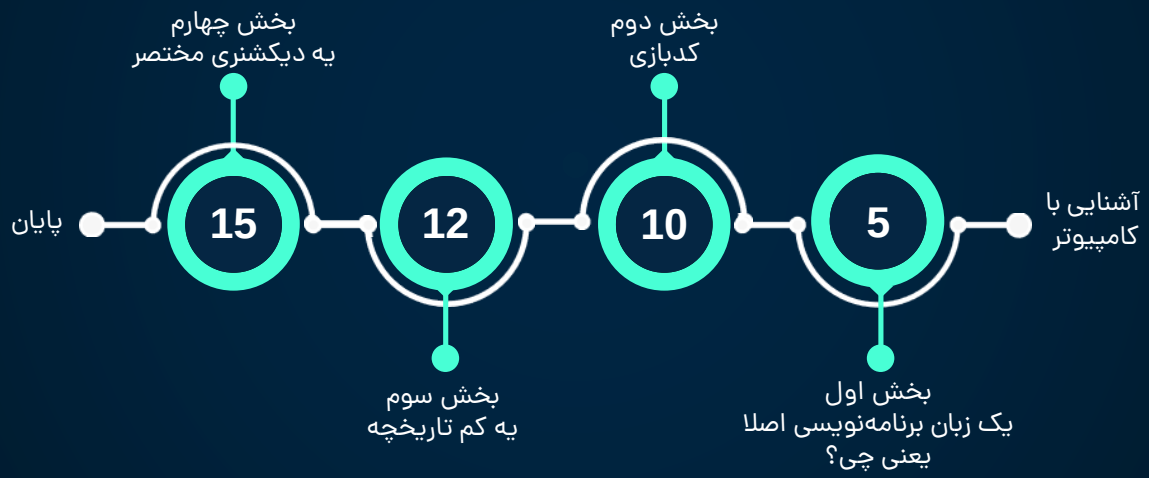
آشنایی با کامپیوتر

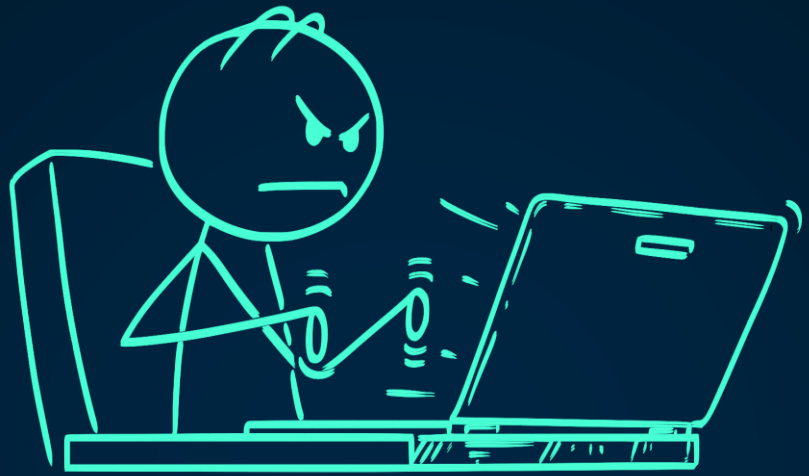
جلسه دوم

اولین چیزهایی که نوزادان یاد می‌گیرند، این است که چطور باید پدر و مادر خود را صدا بزنند و چطور می‌توانند با کلمات، با دنیای اطراف خود صحبت کنند. در دنیای علم هم چنین قانونی حکم می‌کند و هر کسی که می‌خواهد دانشی را کسب کند، باید در ابتدا به کلمات دنیای آن تا حدی به تسلط رسیده باشد تا بتواند با دیگر افراد آن دنیا ارتباط برقرار کند.

دانش کامپیوتر هم مثل بسیاری از علوم دیگر است که برای قدم گذاشتن در آن، نیاز داریم تا با اصطلاحات اولیه‌ی آن آشنا شویم. در این دستورکار هم می‌خواهیم علاوه بر آشنا شدن با تعدادی از اصطلاحات، کمی هم درمورد برنامه‌نویسی و تاریخچه‌ی زبان C بخوانیم.

فهرست


آشنایی با کامپیوتر
5
بخش اول
یک زبان برنامه‌نویسی اصلا یعنی چی؟
10
بخش دوم
کدبازی
12
بخش سوم
یه کم تاریخچه
15
بخش چهارم
یه دیکشنری مختصر
پایان



در دستورکار قبل کمی نسبت به طرز کار سخت افزار کامپیوتر دید پیدا کردید. در آینده در درس "معماری کامپیوتر" به طور کامل‌تری با این بخش‌ها آشنا خواهید شد.

در مرحله‌ی بعد، می‌خواهیم پا به دنیای برنامه‌نویسی گذاشته و درک کنیم برای تبدیل شدن به یک برنامه‌نویس[1]، چه حداقل‌هایی را باید رعایت کنیم و از چه امکاناتی می‌توانیم استفاده کنیم.

برای شروع باید یک زبان برنامه‌نویسی را بلد باشیم...

1- developer or programmer

بخش اول: یک زبان برنامه‌نویسی اصلا یعنی چی؟

وقتی به جهان نگاه می‌کنید، تمام چیزهایی که به هر نحوی با یکدیگر در ارتباط باشند، طبق یک سری قانون و قاعده (پروتکل[1]) این ارتباط را شکل می‌دهند. در ریز بینانه‌ترین حالت، این پروتکل‌ها تحت قوانین فیزیک تعریف می‌شوند و در دید بزرگ‌تر، با چیزهایی مثل زبان‌های طبیعی (فارسی، انگلیسی، عربی، الخ) سروکار دارند.

طبق همین روند، برای برقراری ارتباط با کامپیوترها و ارسال دستورالعمل به آن‌ها، نیاز به یک سری پروتکل و زبان خاص داریم.

در پایین‌ترین سطح، کامپیوترها فقط توانایی ذخیره‌ی ۰ ها و ۱ ها را دارند. می‌توانید این‌طور تصور کنید که کامپیوتر پر است از یک سری سوییچ (کلید روشن/خاموش) خیلی ریز که حالت‌های خاموش یا روشن دارند و از این‌ها به عنوان حافظه‌ی کامپیوتر استفاده می‌شود.

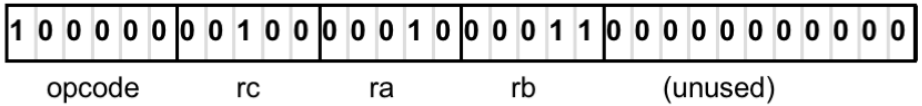
1- Protocol

- پس دستوراتی که به یک کامپیوتر می‌دهیم (مثلا ۳ را با ۵ جمع کن) و داده‌هایی که می‌خواهیم برایمان ذخیره کند (مثلا قسمت موردعلاقه‌تان از سریال برکینگ بد :دی)، همگی به همین شکل ذخیره می‌شوند، که کامپیوتر به وسیله‌ی یک سری بخش‌ها یا سطوح نرم‌افزاری و سخت‌افزاری، این‌ها را جوری تعبیر می‌کند که بتواند فرکانس‌های مختلف صدا را از اسپیکر پخش کند یا پیکسل‌های صفحه‌ی مانیتور را بر حسب نیاز تغییر دهد و ....

- در حال حاضر لازم نیست درگیر شوید که کامپیوتر چطور این کار را انجام می‌دهد. شما قرار است در طول دوران تحصیل خود مرحله به مرحله با این موضوعات بیشتر آشنا شوید که به این مراحل در علم کامپیوتر، سطوح abstraction گفته می‌شود. صرفا در حد یک درک اولیه، عکس صفحه‌ی بعد را ببینید...

Programming Languages

32-bit (4-byte) ADD instruction:



Means, to the BETA, `Reg[4] ← Reg[2] + Reg[3]`

We'd rather write in *assembly language*:

```
ADD(R2, R3, R4)
```

Today

or better yet a *high-level language*:

```
a = b + c;
```

Coming up

نترسید! این یه جمع خیلی ساده‌ست که توی سه سطح تجرید[1] توضیح داده شده.

- هرچقدر که یک زبان برنامه‌نویسی به زبان خود ما نزدیک‌تر باشد، آن زبان، سطح بالاتری دارد. ما در درس مبانی برنامه‌نویسی با بالاترین سطح زبان (یعنی چیزی شبیه به بخش نارنجی رنگ داخل شکل) کار داریم.

- در بخش آبی رنگ نمونه‌ای از زبان اسمبلی را می‌بینید که یک نوع زبان برنامه‌نویسی سطح پایین است. زبان‌های سطح پایین عمدتا به زبان ۰ و ۱ نزدیک‌ترند. رابط بین زبان‌های برنامه‌نویسی سطح بالا و سطح پایین، چیزی به اسم کامپایلر است که یک برنامه به زبان سطح بالا را به یک برنامه به زبان سطح پایین ترجمه می‌کند.
توضیحات بیش‌تر درباره‌ی زبان اسمبلی و جزئیاتش را در درس "ریزپردازنده و زبان اسمبلی" یاد خواهید گرفت.

- تا اینجا با وجود نزدیک‌تر شدن به دنیای ۰ و ۱ها، در این سطح هنوز کاملا نمی‌توانیم چنین دستوری را به کامپیوتر بفهمانیم. پس باید یک سطح پایین‌تر رفته و این دستورات را در قالب ۰ و ۱ به کمک سخت‌افزار کامپیوتر اجرا کنیم.

- این کار توسط همان ثبات‌هایی که اول دستورکار توضیح داده شد، انجام می‌شود...
همانطور که در شکل هم دیدید، برای اجرا کردن عملیات جمع، یک دستور شامل تعداد زیادی ۰ و ۱، اما کاملا با معنا توسط CPU اجرا خواهد شد. در این دستور گفته شده که محتوای ثبات‌های ra و rb که به ترتیب ثبات‌های شماره‌ی ۲ و ۳ هستند به ALU داده شود و حاصل جمع آن‌ها که خروجی ALUست در ثبات rc ذخیره شود.

- این‌که دقیقا چطور این کارها انجام می‌شود و CPU چطور چنین برداشتی از روی چند تا ۰ و ۱ دارد را به طور کامل در درس "معماری کامپیوتر" فرا خواهید گرفت.

بخش دوم: کدبازی

- حال که کمی با زبان برنامه نویسی و ماهیت‌ش آشنا شدیم، می‌خواهیم با یک بازی هیجان انگیز آشنا شویم که ما را با مدل دستورات زبان‌های برنامه‌نویسی تا حدی آشنا می‌کند. می‌دانیم که در این دنیا همه‌ی دستورات باید خیلی ساده‌تر و جزئی‌تر از دنیای ما انسان‌ها تعریف شوند تا کامپیوتر ما آن‌ها را متوجه شود و قدم به قدم اجرا کند.
- سایت زیر یکی از سایت‌هایی است که به کمک بازی، ما را با طرز فکری که برای برنامه‌نویسی لازم است آشنا می‌کند:

Studio code https://studio.code.org

- در ادامه لینک‌هایی از بازی‌های این سایت آورده شده:

Classic Maze https://studio.code.org/hoc/2

Ice Age Play Lab https://studio.code.org/s/iceage/stage/1/puzzle/3

Flappy Code https://studio.code.org/flappy/1

نکته‌ی جالب و با نمک(!) این بازی‌ها این است که بعد از تمام شدن مراحل بازی، گواهی حضور شما در این تورنمنت درون کلاسی! هم به شما داده می‌شود.

حال کاری که باید انجام دهید این است که هر کدام از بازی‌های سایت که دوست دارید را انتخاب کنید و انجام بدید و در نهایت گواهی حضورتان را بفرستید ؛)

توجه داشته باشید که بازی‌ای که انتخاب می‌کنید می‌تواند جزو موارد معرفی شده باشد و یا به انتخاب خودتان یکی از بازی‌های دیگر سایت را انجام دهید.

بخش سوم: یه کم تاریخچه

- اولین زبان برنامه‌نویسی سطح بالا که طراحی شد، زبانی به اسم Plankalkül (از آلمانی: plan=برنامه‌ریختن + kalkul=محاسبه‌کردن) بود که در سال ۱۹۴۸ معرفی شد و تقریبا هیچ‌کس درباره‌ش چیزی نشنیده! این زبان صرفا یک طراحی انتزاعی بود که تا ۳۰ سال بعد از معرفی شدنش، هیچ کامپایلری برای آن معرفی نشد. اما نکته‌ی مهم این است که طراحی این زبان منجر به طراحی زبان‌های خانواده‌ی ALGOL شد.
- زبان ALGOL [1] و نسخه‌های مختلفش توسط ایده‌ها و طراحی مستقیم افراد بزرگی در حیطه‌ی علوم کامپیوتر، از جمله John Backus، John McCarthy، Peter Naur و Donald Knuth ساخته شدن و اولین نسخه‌ی این خانواده، یعنی ALGOL 58 در سال ۱۹۵۸ تولید شد که خیلی زبان کاربردی‌ای نبود. بعدا از روی ALGOL 58 زبان بسیار کاربردی (در زمان خودش البته!) ALGOL 60 را ساختند. بعد از کلی طراحی‌های جدید که با آزمون و خطای بسیار پیاده‌سازی شدند، نهایتا زبان برنامه‌نویسی B توسط Ken Thompson و Dennis Ritchie ساخته شد.

1- Algorithmic Language

```
void continue_last_game(FILE * fp_user, char name_game[]); void decide_for_saving(FILE *
fp_user, char name_game[]); struct node * create_node(FILE * FP, char C[], int i);
struct node * create_list(); struct node * random(struct node * list); struct node *
check_list(struct node * list);
int main(int argc, char const *argv[]){ printf("Please enter your name: "); struct game
user_name; scanf("%s", user_name.name); int save_res = save_name(user_name.name);
```

بعد از گذشت مدتی، خود طراحان زبان B به نتیجه رسیدند که به زبان کامل‌تری نیاز دارند و زبان C را خلق کردند. البته از آن‌جا که هنگام طراحی این زبان می‌خواستند هم‌چنان با زبان B تطابق داشته باشد، در نتیجه این زبان هم ایراداتی دارد که در آینده خودتان متوجه آن‌ها خواهید شد.

شاید برایتان جالب باشد که زبان B و C داریم، ولی زبان A نداریم. (البته اگر بخواهید می‌توانید ALGOL را به عنوان زبان A در نظر بگیرید).

```
int i, a; if(save_res == 0){ printf("Hello %s :)\n", user_name.name); printf("Welcome to
your first game...\n"); printf("You can quit the game by pressing 0\n");
make_new_game(&user_name); }
else if(save_res == 1){ printf("Hello %s :)\n", user_name.name); printf("Which one do
you like?\n"); printf("1) Play a new game\n2) Continue your last game\n");
scanf("%d", &a);
if(a == 1){ printf("Welcome to your new game...\n"); printf("You can quit the game by
pressing 0\n"); make_new_game(&user_name); }
```

- بعد از پیدایش زبان C و محبوبیت بی‌اندازه‌ش به دلیل استفاده شدن در هسته‌ی سیستم‌عامل Unix (که بعدا شد Linux)، این زبان به استاندارد زبان‌های imperative [1] تبدیل شد و سینتکسش [2] (Syntax) تبدیل به استانداردی شد که اکثر زبان‌های برنامه‌نویسی‌ای که امروزه از آن‌ها استفاده می‌شود، از این زبان الگو گرفته‌اند. (جز زبان‌هایی که بر اساس Lisp نوشته شدن که جلوتر یه کم درباره‌شون حرف می‌زنیم.)

- این زبان استانداردهای مختلفی دارد که هر کدام نام مختص به خود را دارند. معروف‌ترین آن‌ها C99 (سال ۱۹۹۹ استاندارد شد) و جدیدترین‌شان C18 (سال ۲۰۱۸ استاندارد شد) هستند. شما با توجه به محیطی که برای برنامه‌نویسی خود انتخاب می‌کنید، احتمالا با یکی از این دو ورژن سر و کار خواهید داشت (اما نگران نباشید! تفاوت بین آن‌ها خیلی زیاد نیست؛ طوری که برای برنامه‌نویس‌های تازه‌کار، تقریبا اصلا به چشم نمی‌آید).

۱- یعنی زبان‌هایی که می‌توانیم در آن‌ها مقدار یک متغیر را تغییر بدیم.
۲- یعنی قوانین مربوط به چگونگی نوشتن دستورهای یک زبان

بخش چهارم: یه دیکشنری مختصر

توی رشته‌ی ما، یه سری اصطلاح رو خیلی زیاد باهاشون مواجه می‌شید. برای این‌که خیلی گیج نشید و یه تصور اولیه ازشون داشته‌باشید، اونا رو این زیر براتون توضیح می‌دیم:

سیستم عامل

ترجمه‌ی عبارت Operating System (یا به اختصار، OS) هست. سیستم‌عامل برنامه‌ایه که دستورات Load شدنش (که بهش می‌گن Firmware)، توی ROM کامپیوتر ذخیره می‌شه که باعث می‌شه موقع روشن شدن کامپیوتر، اولین برنامه‌ای باشه که اجرا می‌شه. وظیفه‌ی اصلی سیستم‌عامل، مدیریت کردن تعداد زیادی برنامه به صورت همروند و ایجاد هماهنگی بین اون برنامه‌هاست. سیستم‌عامل‌ها انواع و ورژن‌های مختلفی دارن. یکی از این انواع کم‌تر شناخته‌شده بین تازه‌کارها، سیستم عامل Linux هست که توزیع‌های بسیار زیادی داره، از جمله Ubuntu, Fedora, CentOS, ArchLinux و... که ویژگی مشترک بین این‌ها، هسته یا کرنل لینوکسه. کرنل یک سیستم‌عامل، کدیه که قسمت‌های اصلی سیستم‌عامل رو توی خودش داره و در مورد Linux، به زبان C نوشته شده. (در مورد این بحث بیش‌تر در درس "سیستم‌عامل" می‌خونید)

زبان‌های برنامه‌نویسی

در مورد خودشون توضیح دادیم، اما خوبه که بدونید چندین پارادایم (می‌شه گفت طرز تفکر) مختلف در زبان‌های برنامه‌نویسی وجود دارن که یه توضیح مختصر و مثال از بعضی‌هاشون میاریم. تفاوت بین این طرز تفکر ها، نحوه‌ای هست که مسائل برنامه‌نویسی رو به قسمت‌های کوچیک‌تر می‌شکنن.

برنامه‌نویسی procedural

این تفکر یه مسئله رو به متغیرها، ساختمان‌های داده (توی درس ساختمان داده و الگوریتم، به اختصار DS می‌خونید) و توابع می‌شکنه و برنامه‌هاش صرفا از پشت هم اجرا شدن یک سری تابع (function, method, procedure) تشکیل شدن. این توابع صرفا یک‌سری دستور هستن که یک‌بار نوشته می‌شن و چندین‌بار و با پارامترهای مختلف اجرا می‌شن. (جلوتر توی همین درس می‌خونید درباره‌شون) که این زبان‌ها همگی imperative هستن. زبان‌های C، Fortran و Pascal جزو این دسته هستند.

برنامه‌نویسی Object-Oriented یا OOP

این تفکر مسئله رو به قسمت‌هایی به اسم Object می‌شکنه که این Objectها مجموعه‌ای از توابع و متغیرها در یک بسته هستند و برنامه‌های این زبان از ارتباط این بسته‌های مختلف با هم‌دیگه تشکیل می‌شن. (توی درس برنامه‌نویسی پیش‌رفته با این بحث بیش‌تر آشنا می‌شید). زبان‌هایی مثل C ++ و Java و C # جزو این دسته هستند.

برنامه‌نویسی Functional

تفکری هست که تنها استفاده از مقدارهای ثابت (constant) رو مجاز می‌دونه، هیچ مقداری قابل تغییر نیست و ایجاد مقدارهای جدید، تنها با ایجاد یک constant جدید ممکن هست. دلیلش هم این هست که با این تفکر می‌شه بهتر درستی عملکرد یک برنامه رو اثبات کرد. (درباره‌ی این پارادایم بیش‌تر در درس «زبان‌های برنامه‌نویسی» می‌خونید) زبان‌هایی مثل Lisp و Haskell و ML جزو این دسته هستند.

و بسیار پارادیم‌های دیگه هم وجود دارن که در این‌جا فرصت نمی‌کنیم بیاریم‌شون ولی پیشنهاد می‌کنیم حتما از این صفحه‌ی ویکی‌پدیا یه مطالعه‌ای روشون داشته‌باشید:

Programming paradigm https://b2n.ir/769612

پایگاه داده یا Database (به اختصار: DB)

مجموعه‌ای از داده‌ها هستند که با نظم خاص و با استفاده از تکنیک‌های مختلفی کنار هم چیده می‌شن. استفاده از Databaseها به این دلیل هست که دسترسی‌مون به داده‌هایی که برای برنامه‌هامون نیاز داریم سریع‌تر بشه و یک سطح Abstraction رو وارد کارمون با دیتا بکنیم. (مثلا به جای این‌که بگیم برو فایل فلان رو بخون و خطی که کاراکتر اولش عدد ۳ هست رو بردار، به دیتابیس می‌گیم داده‌ای که idش شماره ۳ هست رو برای من بفرست.)

محیط توسعه یا Integrated Development Environment یا IDE

برنامه‌هایی هستند که با کنار هم آوردن چندین و چند ابزار کنار هم، روند توسعه‌ی برنامه رو برای برنامه‌نویس آسون می‌کنن. این‌ها با یک ویرایش‌گر متن ساده کار خودشون رو شروع می‌کنن و ابزارهایی مثل Linter (برای مرتب کردن کد)، Compiler، ابزارهای کنترل ورژن (مثل Git که جلوتر می‌بینیدش)، Debugger (برای رفع ایراد کد)، Syntax Highlighter (برای عوض کردن رنگ قسمت‌های مختلف کد) و خیلی آپشن‌های دیگه رو به اون ویرایش‌گر اضافه می‌کنن. محیط‌های توسعه‌ی زیادی داریم که اون‌ها رو این زیر لیست کردیم:

محصولات شرکت JetBrains مثل CLion برای زبان C و IntelliJ برای زبان Java و PyCharm برای زبان Python.

ویژوال استودیو (Visual Studio)

برای توسعه‌ی زبان‌های بسیار زیادی ازش می‌تونه استفاده بشه؛ ولی استفاده‌ی اصلیش برای زبان‌های توسعه‌یافته‌شده توسط مایکروسافت مثل C# و Microsoft C++ هست.
ویژوال استودیو کد (VS Code)، در ابتدا یک ویرایش‌گر متن ساده هست؛ ولی به سرعت با استفاده از ابزارهاش که به صورت Open-Source منتشر شدن، تبدیل به یک IDE می‌شه و در سال‌های اخیر هم به شدت مورد مقبولیت قرار گرفته.
ساب‌لایم (Sublime Text) و Notepad++ که باز ویرایش‌گرهای متن هستن ولی می‌تونن قابلیت‌های دیگه‌ای هم به صورت محدود خودشون اضافه کنن.

سرور - کلاینت

سرور به طور خیلی خلاصه، کد یا کامپیوتری هست که با گرفتن درخواست‌های کلاینت‌ها (کاربران)، پردازش‌هایی روی درخواست‌هاشون انجام می‌ده و جواب رو بهشون برمی‌گردونه. مثال خیلی ساده‌ی سرور می‌شه سایت گوگل که درخواست سرچ شما (کلاینت) رو می‌گیره و یه پاسخی بهتون برمی‌گردونه.

بک‌اند - فرانت‌اند

خیلی از برنامه‌ها از دو قسمت تشکیل شدن. یک قسمت که کاربر باهاش ارتباط برقرار می‌کنه و بهش می‌گن Front-End یا User Interface = UI و یک قسمت که پردازش رو روی داده‌های کاربر انجام می‌ده و بهش می‌گن Back-End که این قسمت از دید کاربر محفوظ هست. (این اصطلاح‌ها رو اکثرا در حوزه‌ی برنامه‌نویسی موبایل یا برنامه‌نویسی وب خواهید دید.)

;