کامپایلر

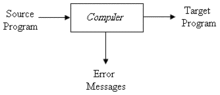
وظیفه این کامپایلر استتفاده از زبان javascript برای تبدیل یک زبان به زبان جاوااسکریپت است .

ویرایشگر آس مجموعه ای از کتابخانه ها است که میتوان با جای گذاری یک فانکشن مشخص در ابتدای پروژه از آن استفاده کرد و با توجه به نیاز از هر زبان برنامه نویسی که نیاز داریم میتوانیم کتابخانه مورد نیاز را مشخص کنیم تا ویرایشگر طبق آن عمل کند.

**کامپایلر** یا **فرودبر**  [برنامه](https://fa.wikipedia.org/wiki/%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87_(%D8%B1%D8%A7%DB%8C%D8%A7%D9%86%D9%87)) یا مجموعه‌ای از [برنامه‌های کامپیوتری](https://fa.wikipedia.org/wiki/%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87) است که متنی از زبان برنامه نویسی سطح بالا (زبان مبدأ) را به [زبانی سطح پایین](https://fa.wikipedia.org/wiki/%D8%B2%D8%A8%D8%A7%D9%86_%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C_%D8%B3%D8%B7%D8%AD_%D9%BE%D8%A7%DB%8C%DB%8C%D9%86) (زبان مقصد) .این برنامه ممکن است برای پردازش شدن توسط برنامه دیگری مثل [پیونددهنده](https://fa.wikipedia.org/wiki/%D9%BE%DB%8C%D9%88%D9%86%D8%AF%D8%AF%D9%87%D9%86%D8%AF%D9%87) مناسب باشد یا [فایل](https://fa.wikipedia.org/wiki/%D9%81%D8%A7%DB%8C%D9%84) متنی باشد که انسان نیز بتواند آنرا بخواند. به این علت فرودبر نامگذاری شده است که کد با زبان سطح بالا را به کد زبان سطح پایین تبدیل می کند بدون این که در رفتار کد تغییری حاصل شود. مهم‌ترین علت استفاده از ترجمه کد مبدأ، ایجاد برنامه اجرایی می‌باشد.

به بیان ساده، کامپایلر برنامه‌ای است که یک برنامه نوشته شده در یک زبان خاص [ساخت‌یافته](https://fa.wikipedia.org/w/index.php?title=%D8%B3%D8%A7%D8%AE%D8%AA%E2%80%8C%DB%8C%D8%A7%D9%81%D8%AA%D9%87&action=edit&redlink=1) را خوانده و آن را به یک برنامه مقصد (Target Language) تبدیل می‌نماید. در یکی از مهم‌ترین پروسه‌های این تبدیل، کامپایلر وجود [خطا](https://fa.wikipedia.org/wiki/%D8%AE%D8%B7%D8%A7) را در برنامه مبدأ اعلام می‌نماید.

شمایی از یک کامپایلر

[](https://fa.wikipedia.org/wiki/%D9%BE%D8%B1%D9%88%D9%86%D8%AF%D9%87:Compiler.GIF)

در اولین نگاه، تنوع کامپایلرها ممکن است به چشم نیاید. تعداد بسیار زیادی زبان‌های منبع وجود دارند که دامنه آنها از زبان‌های شناخته شده مانند [فرترن](https://fa.wikipedia.org/wiki/%D9%81%D8%B1%D8%AA%D8%B1%D9%86) و[پاسکال](https://fa.wikipedia.org/wiki/%D9%BE%D8%A7%D8%B3%DA%A9%D8%A7%D9%84) تا زبان‌های خاص منظوره گسترده است.

کامپایلرها به انواع [تک‌گذره](https://fa.wikipedia.org/w/index.php?title=%D8%AA%DA%A9%E2%80%8C%DA%AF%D8%B0%D8%B1%D9%87&action=edit&redlink=1)، [چند گذره](https://fa.wikipedia.org/w/index.php?title=%DA%86%D9%86%D8%AF_%DA%AF%D8%B0%D8%B1%D9%87&action=edit&redlink=1)، [باردهی و اجرا](https://fa.wikipedia.org/w/index.php?title=%D8%A8%D8%A7%D8%B1%D8%AF%D9%87%DB%8C_%D9%88_%D8%A7%D8%AC%D8%B1%D8%A7&action=edit&redlink=1)، [بهینه‌ساز](https://fa.wikipedia.org/w/index.php?title=%D8%A8%D9%87%DB%8C%D9%86%D9%87%E2%80%8C%D8%B3%D8%A7%D8%B2&action=edit&redlink=1)، [غلط یاب](https://fa.wikipedia.org/wiki/%D8%BA%D9%84%D8%B7_%DB%8C%D8%A7%D8%A8) و ... بسته با عمل انجام شده تقسیم می‌شوند. علی‌رغم این تنوع اعمال  اساسی که هر کامپایلر بایستی انجام دهد، مشابه هم می‌باشند.

نحوه کار :

ﻧﺤﻮﻩ ﯼ ﮐﺎﺭ ﺍﯾﻦ ﻣﺎﮊﻭﻝ ﻧﺴﺒﺘﺎ ﺳﺎﺩﻩ ﺍﺳﺖ ﻭ ﻧﯿﺎﺯ ﺑﻪ ﺗﻨﻈﯿﻤﺎﺕ ﺧﺎص ﺍﺯ ﺳﻤﺖ ﺷﻤﺎ ﻧﺪﺍﺭﺩ ﻭ ﺗﻨﻬﺎ ﮐﺎﺭﯼ ﮐﻪ ﺷﻤﺎ باید انجام دهید ﺗﻌﯿﯿﻦ برخی تنظیمان ساده است

مراحل کاری :

در این کامپایلر از ویرایشگر ace استفاده میشود و باید قبل از نوشتن هر کد این ویرایشگر را از سایت ویرایشگر به آدرس زیر دانلود و در پروژه قرار دهید .

[https://ace.c9.io/#nav=about](https://ace.c9.io/" \l "nav=about)

برای این که از قابلیت تم و رنگ دهی این ادیتور استفاده کنید باید ابتدا این فانکشن را در پروژه خود اجرا کنید :

editor.setTheme("ace/theme/twilight");

تنظیم زبان برنامه نویسی مورد نظر

به طور پیش فرض، ویرایشگر پشتیبانی از حالت متن ساده را دارد در حالی که همه ی زبان های موجود دیگر را در ماژول های مشخص و جدا از هم بنا بر نیاز برنامه نویس در بر دارد .

فایل ها :

Index.html

که شامل کد های html و همچنین اسکریپت دیگر فایل ها است

Test 1-5

که شامل کد هایی برای تست کامپایلر است می باشد.

detector.js

این فایل شامل کد های تابع دتکتور است که در واقع کاتالیزور کامپایلر است

در این ساختار کد نویسی سعی شده که تمامی کد ها به وسیله ی جاوا اسکریپت نوشته شود و از زبان های دیگر مثل رگولار و ... استفاده نشود .

function detector)) تابع دتکتور در واقع همین کار را انجام میدهد که با گرفتن مقادیر اولیه و انتهایی هر المان ورودی آن المان را برای ما return می کند.

* var searchNam
* var searchEnd
* var searchStart

که متغیر های ابتدایی و انتهایی و

script.js

var editor = ace.edit("editor")

در این فایل فانکشن هایی که در ویرایشگر آس نیاز داریم فراخوانده شده اند.

editor.setTheme("ace/theme/twilight");

تم هایی که با توجه به زبان میتوان از آس انتخاب کرد توسظ این خط کد مشخص میشود.

function setValueInEditor(\_txt)

function getEditorValue()

function setCompiledValue(\_txt)

این توابع در انتقال کد های ابتدایی به کامپایلر به ما کمک میکنند که میتوان کد های ابتدایی و مورد نظر را از ۳ روش به کامپایلر تزریق کرد

1. از خود مرورگر
2. آپلود
3. افزودن در فایل index\_code

که در صورت وارد نشدن هیچ کدی پیام خطای متناسب را نمایش می دهد.

و در خط آخر ؛ ( jQuery ) جی کوئری به فایل افزوده شده است.

فایل Module-function

که شامل فانکشن هایی است که در پروژه استفاده شده که یک به یک توضیح داده خواهد شد

**detectComments**

همان طور که گفته شد اصلی ترین تابع در این پروژه دتکتور است که میتواند جایگاه المان های مختلف را با استفاده از موقعیت ابتدایی و انتهایی آنها را مشخص کند .

**کامنت ها** به دو صورت تک خطی و چند خطی در کد نویسی میتوانند بیان شوند که در اینجا هر دو حالت در کد های ورودی به تابع دتکتور فرستاده میشوند و مقادیر آنها مشخص میشوند و با کامنت در استایل جاوا اسکریپت جایگزین میشوند .

هر کد ورودی دارای بخش ها و مقادیر مشخصی است که باید در جایگاه معین باشد برای مثال در زبان فاپلا باید در قالب ماژول باشند و یا در جاوا اسکریپت در فانکشن و این ماژول ها دارای نام معین هستند که این تابع اسم ماژول را گرفته و با استایل جاوا اسکریپت نمایش میدهد .

function detect\_module\_name

خب بعد از این که ماژول را گرفتیم اولین چیزی که مد نظر هر شخصی میآید نام آن ماژول است

پس بلا۰فاصله فانکشنی را مینویسیم که نام ماژال رو همانند ماژول از دتکتور دریافت کند .

function detect\_inputs

مثل هر ماژولی ماژل های این زبان هم دارای ورودی هایی است که ابتدا ارور را false در نظر میگیریم و آرایه result را خالی میسازیم .

به خاطر این که ممکن است دارای چندین و یا چند ورودی باش هر ماژول ما از while و حلقه استفاده میکنیم و به کمک دتکتور حالت هایی که ممکن است یک input وارد شوند و ورودی یک تابع قر ار گیرند را وارد میکنیم و درون این شرط نیاز است که برای جلو گیری از خطاها و باگ های احتمالی فواصل را بررسی کنیم که د رآرایه مات فاصله ای نباشد و به دلیل این که ممکن است در دیگر فانکشن ها هم این کار نیاز شود پس یک فانکشن را تعریف میکنیم به اسم seperatorNotExistکه در فایل main.js ساخته شده که وظیفه دارد هر چیزی که بهش پاس داده میشه رو از نظر انواع فاصله بررسی کند و به ازائ هر کدام از ورودی های این تابع حذف کند .پس تا اینجا ورودی های تابع را در شرط داریم مگر این که در جایی از شرط مشکلی وجود داشته باشد که که has error true میشود و #erorr نمایش داده میشود .

نکته : ممکن است در ماژول چند input وجود داشته باشد و کاربر همه را به صورا صحیح وارد کرده باشد که دتکتور آنها را پیدا میکند و مشکلی نیست اما مشکل در جایی است که یکی از این input ها با مشکل مواجه شده باشند یا سمیکالون نداشته باشندکه در این صورت کد ارور میدهد و ارور را نشان میدهد و کلاس ارور را اککتیو کرده و علامت قرمز ظاهر میشود .

function detect\_module\_output(\_text)

out put با استفاده از رگولار گرفته شده اما نیازی نبوده میتوان از دتکتور گرفت منتهی به خاطر عدم نیاز و عدم استفاده کاری بهش نداشتیم .

function detect\_content(\_text, \_string)

در این تابع ابتدا result به عنوان یک آراییه خالی تعریف میشود و بعد از آن با استسفاده از دتکتور کانتنت را پیدا کرده و به عنوان یک آرایه تحویل داده میشود وتوجه داشته باشید که در اکثر همه فانکشتن ها متن ورودی با استفاده از دتکتور گرفته میشود و به یعنوان یک آرایه ذخیره میشوند

function detect\_module\_content(\_text)

در این فانکشن با استفاده از دتکتور مقدار کانتنت و کد های اصلی ماژول را میگیریم و result را مساوی با آن قرار میدهیم ک

function detect\_while(\_text)

تابع شرط

بعد از این که ماژول و اینپوت ها مشخص شدند به سزاغ شروط میرویم .در شروع به خاطر داشتن دتکتور حالات مختلفی که یک ifمیتواند داشته باشد را بررسی میکنیم و در دتکتور بازخوانی میکنیم

به خاطر این که ممکن است در در هر تابع چندین شرط وجود داشته باشد لذا باید همه تعاریف و عملیات ها را در حلقه whileقرار دهیم .

بعد از این که تمام قسمت های if را در دتکتور گرفتیم نیاز به یک شرط داریم که در صورتی که همه ی قسمت های ifکامل بود و همگی صادق بودند آنگاه کد های ورودی را با کد های جاوا اسکریپت جایگزین کرده و برای جلو گیری از باگ های احتمالی

در آرایه ای که از قبل به عنوان result تعریف کرده ایم پوش میشوند و در غیر این صورت ارور را true بر میگرداند .

**نکته :**در این قسمت و همین طور بعد از while باید چک شود که وقتی if و whileگرفته میشوند در واقع یعنی ماژول مشخص شده و if و while از متن جدا شده اند ولی مشکلی که وجود دارد این است که همه متن را میگیرند و دیگر کدی را باقی نمیگذارند تا ما بقی فانکشن ها آن را بررسی کنند لذا نیاز به یک فانکشن کلی داریم که در result بتواند هر فانکشن هر چیزی را که نیاز دارد بگیرد و مابقه کد هایی که نیز ندارد را پس دهد تا فانکشن بعدی بتواند آن را دتکت کند .

**نکته۲:** در ابتدای فانکشن if ممکن است برخی از متغیر هایی که تعهریف شده اند گیج کننده باشند اما صرف جهت اطلاع این متغیر ها همان طور که در دتکتور گفته شد یک آرایه هسنتند که قسمت اول آنها key قسمت دوم : ابتدا و قسمت سومم انتها موردی اسست که دتکتور باید بگیرد .

**function detect\_while(\_text)**

فانکشن while تشابه خیلی زیادی به فانکشن if دارد چرا که کارایی متشابهی در جاوا اسکریپت دارند توضیحات مطرح شده در فانکشن ifدر اینجا هم صادق است پس از توضیح بیشتر صرف نظر میکنیم و به ادامه فانکشن ها میپردازیم

فایل check\_block همان طور که در تابع while و if گفته شد نیاز به سک فانکشن است تا باقیمانده کد ها را ادیت کند تا همه فانکشن ها بتوانند به خوبی وظیفه ی خود را انجام دهند و کد ورودی را هندل کنند که در این تابع باید همه توابع فایل های دیگر فراخوانده شوند و باقیمانده یکی ورودی تابع دیگر شود که در این صورت کد های باقیمانده تا آخرین فانکشن ادامه پیدا میکنند و همه ی فانکشن ها میتوانند کد را به خوبی هندل کنند و کد به درستی با کد های جاوا اسکریپت جایگزین شود که بتواند به درستی اجرا شود .

فایل دتکتور :

در این فایل هماطور که گفته شده به طور مختصر در فانکشن های گذشته به طور اختصار میتوان گفت که وظیفه ی اصلی این فانشن پیدا کردم پارت های مختلفی از کد است که ما به آنها نیاز داریم که با نقطه ابتدایی و انتهایی آن صورت میگیرد .

نکته : تصمیم هایی گرفته شد که به خاطر بعضی المان ها نظیر if و while که دارای موارد بیشتری از ابتدایی و انتهایی هستند لذا دتکتور را به بیش از دو نقطه گسترش بهدهیم به صورتی که بتواند نقاط ابتدایی و انتهایی و دو نقطه در وسط کد را بگیرد و و کد های پیدا شده رابه ما بر گرداند اما به خاطر هزینه بر بودن میبینیم که در تابع if و while در ابتدای فانکش ها چند بار موارد مختلفی را از دتکتور خواستیم که پیدا کند و این دلیل این همه تعدد است که در جای خودش توضیحات کافی داده شده و نیازی به ادامه نیست .

در این جا بیشتر به خود فانکشن دتکتور میپردازیم که این گونه عمل میکند :

نکته : برای این که بدونیم تعداد المان های درون یک آرایه چند تا است از arguments.length استفاده میشود .