**برنامه نویسی مبتنی بر وب**

**استاد**: برنجی

علیرضا سلطانی نشان

ترم سوم

1399.07.15

فهرست مطالب

[**تفاوت بین Programming Language و Script Language** 3](#_Toc53504690)

[**دسته بندیها** 3](#_Toc53504691)

[**تفسیر** 3](#_Toc53504692)

[**توسعه** 4](#_Toc53504693)

[**سرعت** 4](#_Toc53504694)

[**منظور از شئگرايی چيست؟** 4](#_Toc53504695)

[**فريمورک** 6](#_Toc53504696)

[**برخی از مزایای استفاده از فریمورک ها:** 6](#_Toc53504697)

[**Editor و IDE، تفاوت در چیست؟** 6](#_Toc53504698)

[**انوع روش های برنامهنويسی** 7](#_Toc53504699)

[**برنامهنویسی ضروری** 7](#_Toc53504700)

[**برنامهنویسی ساختارمند** 8](#_Toc53504701)

[**برنامهنویسی تابعی (عملکردی)** 8](#_Toc53504702)

[**برنامهنویسی اظهاری** 10](#_Toc53504703)

[**تاريخچه PHP** 10](#_Toc53504704)

[**نوشتن کد در زبان PHP** 12](#_Toc53504705)

[**دستور echo** 12](#_Toc53504706)

[**دستور Print** 12](#_Toc53504707)

[**تفاوت echo با print** 13](#_Toc53504708)

[**متغیرها** 13](#_Toc53504709)

[**ثابت ها** 13](#_Toc53504710)

[**تفاوت Single quote و Double quote** 14](#_Toc53504711)

[**مراجع** 15](#_Toc53504712)

**تفاوت بین Programming Language و Script Language**

مهم ترین تفاوت بین زبان برنامه نویسی با زبان اسکریپت نویسی، به نوع اجرای[[1]](#footnote-1) آنها بر می­گردد. در **Programming** **Language** یک مترجم[[2]](#footnote-2) وجود دارد تا برنامه نوشته شده شما را از زبان سطح بالای[[3]](#footnote-3) برنامه نویسی به زبان سطح پایین ماشین[[4]](#footnote-4)، تبدیل کند. در زبان اسکریپت نویسی، دیگر خبری از یک مترجم نیست، بلکه یک مفسری[[5]](#footnote-5) در برنامه حاکم است تا کدهای شما را به صورت خط به خط اجرا کند و در این سیستم دیگر خبری از مترجمی نیست که کاملا یک برنامه را به زبان سطح پایین ترجمه کند و سپس خروجی را به توسعه دهنده نمایش دهد.

**تفاوت های اساسی PL با SL**

**دسته بندی­ها[[6]](#footnote-6)**

یک زبان برنامه نویسی دارای 5 بخش اصلی است، نسل اول، نسل دوم، نسل سوم، نسل چهارم، نسل پنجم.

اما در یک زبان اسکریپتی فقط دو بخش اصلی آشنا وجود دارد: بخش سمت سرویس دهنده [[7]](#footnote-7)و بخش سرویس گیرنده.[[8]](#footnote-8)

**تفسیر**

زبان های برنامه نویسی، با یک طراحی جمع و جور طراحی شده اند در حالی که نیازی به تفسیر کننده کد توسط برنامه یا زبان دیگری ندارند، در حالی که در زبان های اسکریپتی، با یک زبان نوشته شده اند و توسط دیگر برنامه ها تفسیر می­شوند. برای مثال اگر ما بخواهیم JavaScript را به صورت Native در کنار HTML بنویسیم، برنامه ای که این کد های جی اس را اجرا می­کند در واقع همان مرورگر است، یا مثلا در Node.js برای اجرای بلادرنگ [[9]](#footnote-9)برنامه نیاز به یک برنامه­ی دیگری بنام **nodemon** داریم که بتواند کدهای نوشته شده را تفسیر کند تا مرورگر در مرحله بعدی توسط لایه نمایش در معماری MVC، قسمت سمت کاربر را نمایش دهد.

**توسعه[[10]](#footnote-10)**

نوشتن کد با استفاده یک زبان برنامه نویسی مانند **سویفت** یا **سی**، نسبتا سخت می باشد چون که برای مثال برای نوشتن یک تابع، نیاز به کد خط­های زیادی است. در حالی که در یک زبان اسکریپتی مانند پایتون یا جی اس یا PHP، در زمان و تعداد کدهای کمتری خیلی راحت میتوان یک تابع یا کلاس را نوشت.

**سرعت**

سرعت برنامه های نوشته شده توسط زبان های برنامه نویسی نسبت به زبان های اسکریپتی، بیشتر است، زیرا زبان های برنامه نویسی یک بار کد را ترجمه می­کنند اگر مشکلی یا خطایی در فرآیند اجرای برنامه وجود داشته باشد، برنامه همان اولین زمان متوقف می­شوند، و خطا و هشدار را اعلام میکنند، در غیر این صورت اگر هیچ مشکلی نباشد کل برنامه را به صورت تمام و کمال اجرا خواهند کرد. اما در زبان های تفسیر کننده چون که برنامه به صورت خط به خط اجرا می­شوند، اگر در یکی از خط­ها مشکلی باشد کل برنامه متوقف می­شود و نتیجه ناقصی را خواهیم داشت.

**منظور از شئ­گرايی[[11]](#footnote-11) چيست؟**

برنامه نویسی شئ­گرا یکی از روش های برنامه نویسی است که بر پایه یکسری از مفاهیم کلاس ها و شئ ها است. شئ­گرایی به عنوان ساختار یک برنامه نرم­افزاری مورد استفاده قرار میگیرد، که میتوان از آنها به صورت مجدد بارها و بارها استفاده کرد، یعنی ما آنها را به صورت یک ساختار یا چهارچوب کلی ایجاد می­کنیم و می­توانیم نمونه های [[12]](#footnote-12)زیاد و مختلفی از آن ها داشته باشیم. به خاطر اینکه شئ­گرایی یک شیوه برنامه نویسی است ما در این زمینه زبان های زیادی مانند، سی ها، پایتون، جی اس و غیره را داریم.

یک برنامه نویس اطلاعات و رفتار هایی که می­توانند روی این اطلاعات تاثیر داشته باشند، در نرم­افزار خود از الگویی به نام **Class** استفاده می­کند.

در نوشتن این گزارش برای معرفی شئ ها از زبان JS استفاده کردم:

1. } const users = {
2. name: 'Alireza',
3. family: 'Soltani',
4. age:  getAge = (userBirthday) => {
5. const realAge = Date.now() - userBirthday
6. return realAge
7. },
8. field: "JS programmer"
9. }// This is a very simple object in JS

یک کلاس ساده برای دریافت نام و نام خانوادگی و سن او:

1. class users {
2. constructor(name, family, birthday) {
3. this.name = name
4. this.family = family
5. this.birthday = birthday
6. }
8. calculateUsers() {
9. const years = Date.now() - this.birthday
10. const name = this.name
11. const family = this.family
12. return {
13. name,
14. family,
15. years
16. }
17. }
18. }
20. const user1 = new users('Alireza', 'Soltani', 2000)
21. const alirezaSoltaniNeshan = user1.calculateUsers()
23. console.log(alirezaSoltaniNeshan)

**فريم­ورک**

فريم­ورک يک کتابخانه­ای برای استفاده در اهداف خاص و مشخص است که توسط يک زبان برنامه­نویسی معرفی شده. تا بتوانیم با استفاده از آن سرعت بیشتری برای تولید محصول و نتیجه خود داشته باشیم. برای مثال در زبان جی­اس فریم­ورک های زیادی در استفاده های گوناگونی وجود دارد. بر فرض مثال ما میخواهیم برنامه نویسی سمت کاربر را با یک فریم­ورک انجام بدیم که بسیار واکنش پذیر و پویا باشد پس از فریم­ورک وب **React** یا **Angular** استفاده می­کنیم. یا برای برنامه نویسی موبایل می­توانیم از زبان جی­اس با استفاده از فریم­ورک **React** **Native** و در زبان پایتون با استفاده از **Kivy** (**کیوی**) استفاده کنیم. یا اینکه اگر بخواهیم برنامه نویسی سمت سرور را داشته باشیم از پایتون فریم­ورک Django یا پلتفرم Node.js استفاده می­کنیم.

**برخی از مزایای استفاده از فریم­ورک ها:**

* کاهش زمان کد زدن
* افزایش بهره­وری و صرفه جویی در تعداد خط ها
* بسیار مناسب برای کارهای تیمی
* استفاده از مدل طراحی MVC

**Editor و IDE، تفاوت در چیست؟**

بطور کلی **در یک IDE** هنگام نصب و راه اندازی آن، یک نرم افزار به صورت یکپارچه با یکسری از تکنولوژی­ها و فریم­ورک ها نصب خواهند شد که ممکن است در ابتدای نصب نیازمند به اینترنت باشید که تمامی پکیج ها و برنامه های مورد نیاز دانلود شوند، و بعد از راه اندازی یک IDE، دیگر نیاز به هر بار نصب راه­اندازی یک پلتفرم یا اولیه سازی در هر بار نوشتن یک پروژه جدید نیست. برای مثال وقتی شما برنامه Android Studio را نصب میکنید، بعد از راه اندازی آن و آماده کردن Gradle دیگر نیاز به نصب دوباره پکیج ها ندارید. در حقیقت یک IDE محیطی یکپارچه را فراهم میکند که به توسعه پلتفرم خاص خود بپردازید.

**یک ویرایشگر یا (Editor)** حکم یک جعبه ابزار همه فن حریف را دارد که برای هر کاری مورد استفاده قرار گیرد،(کاملا یک ویرایش کنند حالا هر چیزی) برای مثال، شما می­توانید برنامه­ای با هر زبانی بنویسید، به شرط آنکه تمام ملزومات و کتابخانه های آن نصب باشد، به عنوان پیشفرض VS code، برای نوشتن یک برنامه فلاتر نیازمند نصب تمامی پکج ها در هر پروژه خود دارید، یا مثلا در هنگام برنامه نویسی سرور ساید در زبان جی­اس ابتدا نیاز به بسته های node\_moduals در هر پروژه دارید، که بایستی به وسیله پکیج منیجر npm کتابخانه ها و فریم­ورک های موردنظر را نصب کنید. این نصب و به اصطلاح init کردن در هنگام استفاده از ویرایش به صورت یکپارچه برای همیشگی نیست بلکه هربار پروژه زدن نیاز به دوباره کاری داریم، اما در IDE مانند Xcode هنگام پروژه زدن یا init کردن برنامه خود همه وابستگی ها، همان ابتدای نصب مشخص شده اند و نیازی به دوباره کاری نیست.

**انوع روش های برنامه­نويسی**

روش های برنامه نویسی، یعنی چه راه هایی برای نوشتن یک برنامه وجود دارد.

روش های برنامه نویسی هم به تعداد هستند: که در اینجا 4 نمونه از آشناترین آنها را برسی خواهم کرد.

**برنامه­نویسی ضروری[[13]](#footnote-13)**

کنترل جریان در برنامه­نویسی ضروری به صورت صریح است. یعنی دستورات نحوه محاسبه را قدم به قدم نشان می­دهند.

1. result = []
2. i = 0
3. start:
4. numPeople = length(people)
5. if i >= numPeople goto finished
6. p = people[i]
7. nameLength = length(p.name)
8. if nameLength <= 5 goto nextOne
9. upperName = toUpper(p.name)
10. addToList(result, upperName)
11. nextOne:
12. i = i + 1
13. goto start
14. finished:
15. return sort(result)

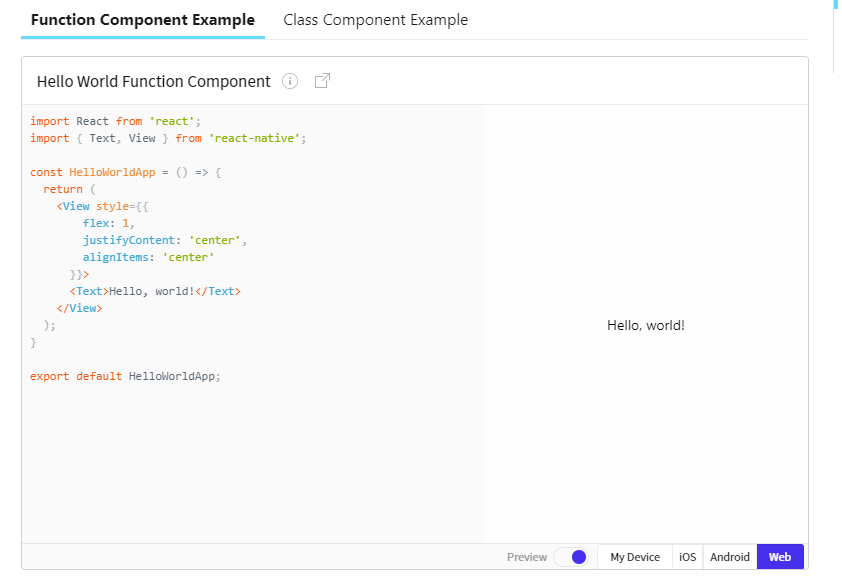
**برنامه­نویسی ساختارمند[[14]](#footnote-14)**

برنامه­نوسی ساختارمند هم از هم­نوع برنامه­نویسی ضروری میباشد با این تفاوت که کنترل جریان به صورت حلقه های تو در تو مشخص شده. مثال زیر یک نوع خیلی واضحی از برنامه ساختارمند است، در آن مشخص شده که از خانه اول آریه ای بنام people که در آن object هایی قرار دارد، افرادی که تعداد کلمه نام آنها بیشتر از 5 باشد را در لیست result پوش میکند.

1. result = [];
2. for i = 0; i < length(people); i++ {
3. p = people[i];
4. if length(p.name)) > 5 {
5. addToList(result, toUpper(p.name));
6. }
7. }
8. return sort(result);

**برنامه­نویسی تابعی (عملکردی)[[15]](#footnote-15)**

در برنامه­نویسی تابعی تمام علمیات درون یک تابع نوشته می­شود، در این یک تابع ممکن است تعداد کم یا زیادی از توابع برای انجام یکسری Taskها وجود داشته باشد. (یک مثال) در زمانی که شروع به یادگیری برنامه نویسی جی اس کردم و زمانی که خواستم بروی فریم­ورک ریکت نیتیو مشغول شوم، دو راه برای نوشتن برنامه های React Native با JS وجود داشت، یک برنامه نویسی extended Class (حالی نه مثل polymorphism) و دیگری برنامه­نویسی به صورت Functional. تا قبل از سال 2018 برنامه نویسی با فریم­ورک React Native به صورت تابعی، در حقیقت برای نمایش محتوای ثابت [[16]](#footnote-16)بود و برای پویایی و استفاده از کلاس ها و کانکشن ها دیگر، از extended class استفاده میشد، از سال 2018 به بعد دیگر فرقی نمیکند که شما به صورت تابعی برنامه نویسی میکنید یا کلاسی، در هر دو روش می­توانید به یک نتیجه برسید.



شکل 1اکسپو کامپایلر، روش تابعی یا کلاسی

**برنامه­نویسی اظهاری[[17]](#footnote-17)**

نیازی به توضیح اضافی نیست، در حقیقت این نوع از برنامه­نویسی را همه توسعه دهندگان با آن آشنا هستند، زبانی که به صورت واضح با استفاده از یکسری پرس و پاسخ هایی را انجام میدهیم. مانند زبان SQL.

1. select upper(name)  
   from people where length(name) > 5 order by name

**تاريخچه PHP**

**PHP** يک زبان اسکريپت­نویسی مخصوص توسعه وب می­باشد. در حقیقت توسط برنامه نویس دانمارک-کانادایی به نام Rasmus Lerdorf در سال 1994، ساخته شده است. **PHP** درواقع بر پایه Personal Home Page است، اما آن امروزه بر پایه **Hypertext** **Preprocessor** است. کدهای PHP معمولا توسط یک سرویس دهنده وب مانند (درحالت لوکال) Wamp، توسط مفسر PHP پردازش می­شوند و نتیجه توسط مرورگر نمایش داده میشود. در وب سرویس، نتیجه تفسیر و اجرای کد PHP که ممکن است هر نوع داده ای باشد، مانند داده هایی از قبیل HTML یا حتی Binary Image Data، که می­­خواهند توسط پروتکل HTTP به سمت کاربر پاسخ داده شوند. مانند Web Template های مختلف مانند Handlebar, EJS و غیره، PHP می­تواند لا به لای ساختار HTML نوشته شود. PHP میتواند در بسیاری از وظایف خارج از فضای وب هم مورد استفاده برنامه نویسان قرار گیرد، از قبیل برنامه های گرافیکی مستقل[[18]](#footnote-18)، یا کنترل ربات ها یا برنده های کنترلی. و همچنین مانند پلتفرم نود جی اس می­تواند به صورت خط فرمان [[19]](#footnote-19)هم مورد استفاده قرار گیرد.

استاندارد مفسر زبان PHP بر پایه و اساس Zend Engine، برنامه رایگان منتشر شده زیر نظر لایسنس PHP است. که درواقع Zend Engine یک برنامه متن باز به عنوان مفسر زبان اسکریپتی PHP است. PHP به طور گسترده و با محدوده بزرگی می­تواند در بسیاری از وب سرویس ها و تقریبا در هر سیستم عامل و پلتفرمی به طور آزاد مورد استفاده قرار گیرد.

**نسخه های PHP**

**نسخه 4.0**

الان PHP در نسخه 7 خود بسر میبرد. PHP از نسخه 4 شروع به توسعه شد، در 22 می سال 2000 تقریبا 18 ماه پس از اعلام رسمی در مورد ازسر گیری فعالیت­­های برنامه­نویسی برای نسخه جدید، PHP 4.0 به بیرون منتشر شد. بسیاری از مردم، نسخه 4 را جنجالی­ترین نسخه از این زبان می­دانستند و در بسیاری از فروم ها صحبت هایی در مورد آن میشد. بعد از چند ماه، سایت netcraf گزارشی بیرون داد که تخمین می­زد بیشتر از 3.6 میلیون دامنه، PHP را روی دامنه های خود نصب کرده اند.

**نسخه 5.0**

در نسخه پنجم PHP بسیاری از توابع اضافه شدند، توابعی مانند، destroy() و سازنده ها، تکثیر اشیاء، class abstraction، حوزه متغیرها، رابط­ها و ارتقای طریقه مدیرت کردن اشیا.

**نسخه 6.0**

در این نسخه از زبان PHP ، از کد های Unicode پشتیبانی شد، امنیت پیشرفت بسیار زیادی داشته،

**نسخه 7.0**

بعد از نشخه ششم اين زبان، ويژگی هایی زیادی معرفی شدند که در اینجا به چندتا از آنها اشاره می­کنم:

مدیریت بهتر خط ها، موتور Zend بهبود پیدا کرد، از نسخه 64 بیتی پشتیبانی نمود، انواع کلاس های بدون نام، کارایی و سرعت بیشتر نسبت به قبل، ایجاد عملگرهای جدید (مخالف <>)، مرتب سازی متغیر ها و تعریف کردن مقادیر برای هرکدام، قابلیت مدیریت عیب Single Thread بودن با استفاده از Promiseها و غیره.

**نوشتن کد در زبان PHP**

برای نوشتن برای های PHP می­توانیم در لابه­لای ساختار HTML مورد استفاده قرار گیرند، برای نوشتن یک برنامه PHP بایستی در وب سرویس شما این اتفاق بیوفتد چرا که این زبان یک زبان اسکریپتی سمت سرور می­باشد باید با فرمت PHP در وب سرویس شما اجرا شود، مهم نیست از چه وب سرویسی استفاده می­کنید، با کمی آگاهی فقط کافی است در آن فایلی با پسوند مربوط نوشته و در داخل آن به توسعه برنامه PHP خود بپردازید.

برای نوشتن هر کدام از کدهای PHP لازم است درون <?php?> نوشته شود تا کدهای HTML شما با PHP تمایز داشته باشند.

**دستور echo**

برای نمایش خروجی و یک پیام می­توان از دستور echo استفاده کرد.

1. <?php
2. echo “Hello, PHP”;
3. ?>

**دستور Print**

دستور Print هم مانند دستور echo می­باشد

1. <?php
2. print(“Hello, PHP”);
3. ?>

**تفاوت echo با print**

اما سرعت اجرا شدن در echo بیشتر از print است. در echo شما می­توانید چند تا پارامتر را به عنوان نمایش وارد کنید اما در Print همچین امکانی وجود ندارد و در هنگام استفاده از echo ما هیچ مقداری به عنوان مقدار برگشتی نخواهیم داشت، اما در Print مقدار عدد 1 برگشت داده می­شود.

**متغیرها**

برای معرفی متغیر به نحو زیر عمل می­کنیم:

1. <?php
2. $name = “Alireza”;
3. $age = 19;
4. echo $name, $age;
5. ?>

متغير ها قسمتی هایی از یک حافظه هستند که با توجه به نوع مشخصی یک قسمتی را اشغال می­کنند و به صورت موقتی تا زمان اجرای برنامه در دسترس می­باشند.

انوع مخلفی که حتما میدانیم وجود دارد:

رشته ها، اعداد (اعشاری و صحیح)، آرایه ها و آبجکت ها فانکشن ها و غیره.

**ثابت ها**

زمانی که از ثابت ها استفاده میکنیم دیگر نباید قصد تغییر مقدارشان را داشته باشیم:

1. <?php
2. define(“number”, 58 );
3. echo “The result number is”.number;
4. ?>

**تفاوت Single quote و Double quote**

در Single quote ما اگر متغیری را بنویسیم، فقط آنرا نمایش میدهند یعنی:

1. echo ‘$a’ //$a

اما وقتی از double quote استفاده میکنیم مانند آن است که داریم اسم متغیر را صدا میکنیم که برای فقط مقدارش را نمایش دهد.

1. echo “This is a simple number in double quote: $a”;//This is a simple number in double quote: 15

برای اینکه بخواهیم از نوع متغیری باخبر شویم از تابع gettype(your var) استفاده خوهیم کرد

**مراجع**

روش های برنامه­نویسی: https://cs.lmu.edu/~ray/notes/paradigms

1. Execute [↑](#footnote-ref-1)
2. Compiler [↑](#footnote-ref-2)
3. High Level [↑](#footnote-ref-3)
4. Low Level (Machine Level) [↑](#footnote-ref-4)
5. Interpreter [↑](#footnote-ref-5)
6. Categories [↑](#footnote-ref-6)
7. Server Side [↑](#footnote-ref-7)
8. Client Side [↑](#footnote-ref-8)
9. RealTime [↑](#footnote-ref-9)
10. Development [↑](#footnote-ref-10)
11. Object Oriented Programming [↑](#footnote-ref-11)
12. Instances [↑](#footnote-ref-12)
13. Imperative [↑](#footnote-ref-13)
14. Structed [↑](#footnote-ref-14)
15. Functional Programming [↑](#footnote-ref-15)
16. Static Contents [↑](#footnote-ref-16)
17. Declarative Programming [↑](#footnote-ref-17)
18. Standalone Graphical Applications [↑](#footnote-ref-18)
19. Command Line Interface [↑](#footnote-ref-19)