هوک وردپرس و استفاده از اکشن و فیلتر در توسعه پذیری وردپرس

WordPress Hooks API

نویسنده: مهرشاد درزی

کارشناس ، مشاور و توسعه دهنده وردپرس

وب سایت: RealWp.net

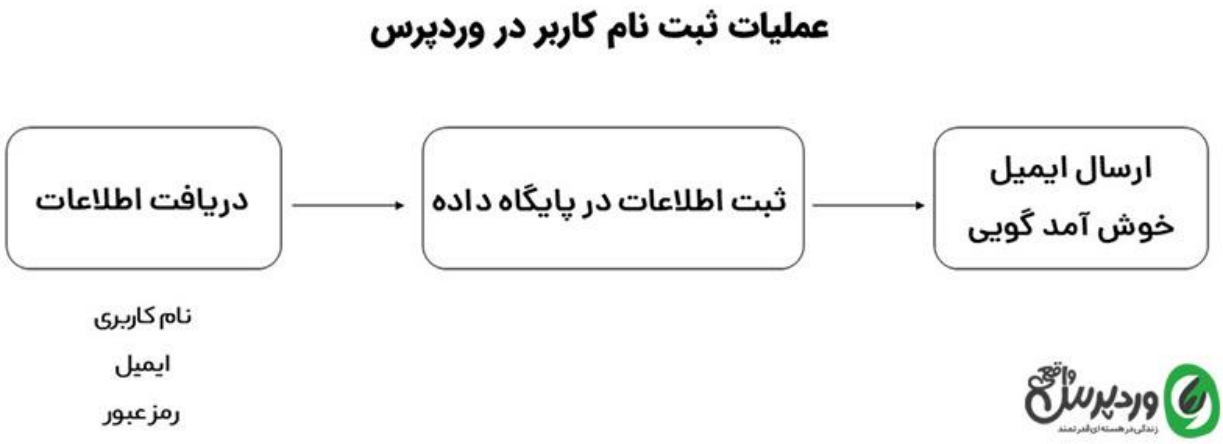
هوک وردپرس (WordPress Hooks) که به آن به فارسی گاها قلاب نیز گفته می شود ، در واقع میتوان مهم ترین عامل در قابلیت توسعه پذیری سیستم **وردپرس** در ساخت انواع پروژه ها در فضای وب دانست.اگر بخواهیم در یک کلمه هوک وردپرس را تعریف کنیم میتوان گفت" **هوک یعنی تاثیر گزاری**."

تاثیر گزاری بر تمام وقایع و قابلیت های وردپرس که هر لحظه در درخواست های مختلف انجام می شود. این تاثیر گزاری را می توان افزودن ، حذف کردن یا تغییر در تمامی پارامتر هایی که موجودیت ابتدایی دستورات در وردپرس هست دانست.

هوک وردپرس چرا بوجود آمد ؟

فرض کنید شما سازنده وردپرس بودید و می خواستید به نحوی این سیستم را پیاده سازی کنید که کاربران یا یک **توسعه دهنده وردپرس** بتواند براحتی قابلیت هایی به آن اضافه یا کم کند به شکلی که کد های اصلی سورس وردپرس به هیچ عنوان دستکاری نشود.

شما در بخشی از این سیستم خود یک عملیات خواهید داشت که وظیفه ی ثبت نام کاربر در سیستم وردپرس را بر عهده دارد. به عنوان مثال تابعی وجود دارد که نام کاربری (user_login) ، ایمیل (user_email)و رمز عبور (Password) را دریافت می کند. پس از تجزیه و تحلیل اطلاعات ، کاربر را در بانک اطلاعاتی وردپرس شما ثبت نام کرده و در نهایت یک ایمیل خوش آمد گوشی برای شخص ارسال می کند .



اگر سیستم بالا کاملا ثابت باشد دیگر نمی توان از آن برای توسعه پروژه های مختلف استفاده کرد ؟ چطور ؟ کاملا واضح هست مثلا ما می خواهیم بعد از ثبت نام یک کاربر در وب سایت فروشگاهی مان یک پیامک برای او برود ، یا اصلا یک کد تخفیف برای او ایجاد شود. در صورتی که در حالت عادی سیستم بالا تنها یک ایمیل برای شخص ارسال می کند.

اینجاست که واژه **اکشن وردپرس (**WordPress Action**)** متولد شد. یعنی اکشن به ما قابلیتی می دهد که **پس از انجام یک عملیات ما بتوانیم عملیات دل خواه دیگر را اجرا کنیم** یا به اصطلاح قلاب آن کنیم.بوسیله اکشن در وردپرس شما این قابلیت رو دارا خواهید بود پس از انجام یک عملیات مثل ثبت نام کاربر ، با توجه به نیاز های خود کارهایی را انجام دهید مثل ایجاد کد تخفیف برای کاربر یا ارسال پیامک و...

اگر کمی فکر کنید هنوز هم سیستم بالا در جایی می لنگد ، فرض کنید کاربری اصلا دوست دارد در وب سایت وردپرسی خود کاربران با شماره موبایل ثبت نام کنند و ایمیل نداشته باشند. در صورتی که ورودی سیستم ثبت نام کاربر در بالا بر اساس سه پارامتر نام کاربری ، ایمیل و رمز عبور هست. این مسئله به چه صورت حل شده ؟

با **فیلتر وردپرس (WordPress Filter)** شما این قابلیت را دارید که **متغیر ها و پارامتر هایی که در جریان عملیات وردپرس وجود دارند را دستکاری کنید** ، این دستکاری می تواند شامل اضافه کردن ، کم کردن یا تغییر در مقدار هر یک از مقادیر باشد. مثلا اگر می خواهیم وب سایت ما تنها با شماره موبایل کاربران ثبت نام انجام دهد با قابلیت فیلتر وردپرس الزامی بودن email را کاملا از بین می بریم و فرض رو بر آن می گزاریم که نام کاربری همان شماره همراه باشد.

کدهای هسته وردپرس که در پوشه های wp-admin و wp-includes و فایل های PHP که در پوشه اصلی وردپرس هستند ، تحت هیچ شرایطی نبایستی توسط شما یا توسعه دهنده وردپرس دست کاری شود. زیرا به محض آپدیت و بروز رسانی وردپرس با ورژن جدید تمامی دستکاری های شما از بین خواهد رفت.اصلا برای همین هست که هوک های وردپرس خلق شدند تا امنیت وردپرس بالا برود.

هوک وردپرس به چند دسته تقسیم می شود ؟

حال که با مفهوم هوک های وردپرس که شامل اکشن و فیلتر هست آشنا شدیم و تاثیر معجزه آمیز آن در توسعه وردپرس کاملا آشکار شده ، بیایید کمی بر تعریف رسمی مستندات وردپرس تاکید کنیم تا بتوانیم برنامه نویسی هوک وردپرس را به شکل حرفه ای شروع کنیم.

اکشن وردپرس (Actions)

اکشن ها در وردپرس مسیری را برای ما باز می کنند که بتوانیم یک یا چند عملیات دل خواه را در زمان انجام شدن عملیات های وردپرس ، بعد از آن اجرا کنیم. این عملیات ها به ترتیب و بر اساس **priority** یا حق تقدم هایی که ما به هسته ی وردپرس اعلام می کنیم اجرا خواهند شد.

فیلتر وردپرس (Filters)

فیلتر وردپرس راهی هست که بوسیله آن می توان تغییر در داده های عملکردی عملیات وردپرس انجام داد. دقت کنید که فیلتر تنها وظیفه ی تغییر در عملکرد همان متغیری دارد که به عنوان ورودی به آن داده شده است و ما نباید دستورات غیر از این را در فیلتر های وردپرس پردازش کنیم. اگر قصد دارید عملیاتی مجزا در وردپرس انجام دهید جای آن اکشن وردپرس اس ن فیلتر وردپرس.

تفاوت اصلی اکشن و فیلتر در وردپرس

اگر بخواهیم در یک جمله تفاوت اکشن و فیلتر در وردپرس را بیان کنیم این است که در اکشن ما یک عملیات را انجام می دهیم و هیچ مقداری بازگردانده نمی شود به سیستم ، در صورتی که در فیلتر ما یک مقدار را دریافت می کنیم سپس بر روی آن مقدار تغییراتی انجام می دهیم و در نهایت آن را به سیستم باز میگردانیم تا ادامه پردازش انجام شود.

پس اگر در مراحل پردازش یک اکشن دل خواه در وردپرس ، سیستم دچار مشکلی شود همانند حالت طبیعی PHP کاربر با خطا در صفحه مواجه خواهد شد و اگر در فیلتر های وردپرس ما متغیر را پاس ندهیم به ادامه سیستم یا به نوعی مقدار دهی کنیم که سیستم از ادامه کار باز بماند مثلا یک آرایه دریافت کنیم و آن را تبدیل به عدد کنیم و بازگردانیم در صورتی که سیستم نیاز به آرایه داشت. در این حالت هم سرویس به خطا برخورد خواهد کرد.

هوک وردپرس در کجای هسته ی وردپرس قرار دارد؟

کلاس WP_HOOK که در زیر شاخه ی wp-includes/class-wp-hook.php قرار دارد، پایه و اساس سیستم هوک وردپرس می باشد. این کلاس به صورت کاملا پایه نوشته شده است به شکلی که شما می توانید از این فایل در پروژه های غیر وردپرسی نیز استفاده کنید. به عنوان مثال پکیج های زیادی

در فریم ورک لاراول مثل tormjens/eventy بر اساس همین فایل طراحی شده که امکان استفاده از سیستم هوک وردپرس را در لاراول مهیا می سازد.

همواره یکی از سیاست های وردپرس بر این بوده که برای راحتی در توسعه و برنامه نویسی وردپرس کاربر را با Template Function ها آشنا کند ، این بحث در بخش هوک های وردپرس هم انجام شده به همین دلیل توابعی که ما میخواهیم از دل کلاس WP_HOOK بیرون بکشیم بدون هیچ سختی در فایل wp-includes/plugin.php قرار داده شده.

اگر این فایل را مشاهده کنید ، ابتدا فایل کلاس اصلی را بارگزاری نموده سپس یکسری تابع برای ما مهیا ساخته است.

```
23
24  // Initialize the filter globals.
25  require __DIR__ . '/class-wp-hook.php';
26
27  /** @var WP_Hook[] $wp_filter */
28  global $wp_filter, $wp_actions, $wp_current_filter;
29
30  if ( $wp_filter ) {
31          $wp_filter = WP_Hook::build_preinitialized_hooks( $wp_filter );
32  } else {
33          $wp_filter = array();
34  }
35
```

فرآیند و ساز و کار هوک وردپرس

سه ویژگی اصلی برای اکشن ها و فیلتر های وردپرس وجود دارد که ساز و کار آن را شامل می شود:

مرحله جذب هوک وردپرس

هوک های وردپرس ابتدا در نقطه ای از کدنویسی وردپرس مشخص می شوند ، این نقطه جذب یک شناسه دارد تا بوسیله آن ما بتوانیم عملیات هایی را به آن قلاب کنیم .عموما این شناسه با یک نام انگلیسی مشخص می شود. در اکشن ها برای ایجاد نقطه ی جذب از **do_action** استفاده می کنیم و در فیلتر ها برای ایجاد نقطه ی جذب از **apply_filters** استفاده می کنیم.

مرحله اتصال عملیات ها

بعد از این که نقطه ی جذب مشخص شد ، حال ما میتوانیم بی نهایت تابع یا عملیات را به آن نقطه ی جذب اتصال دهیم. برای اتصال به نقطه ی جذب در اکشن ها از **add_action** و در فیلتر ها از **add_filter** بهره میگیریم.

صف بندی و لیست عملیات ها

اگر تعریف مرحله ی قبل را به دقت خوانده باشید متوجه خواهید شد که ما محدودیتی در اتصال عملیات به یک نقطه ی جذب نداریم. پس وردپرس از کجا می فهمد که کدام عملیات را زود تر و کدام را دیرتر انجام دهد ؟

این ویژگی که همان **priority** یا حق تقدم هست در بخشی از تعریف نقاط اتصال در add_action یا add_filter قرار دارد که به ما این امکان را می دهد که بتوانیم صف بندی را کاملا اصولی و به ترتیب انجام دهیم.

وردپس از کجا لیست تمامی نقاط اتصال را نگه میدارد تا در آن لحظه اجرا کنید ؟

در تصویر بالا اگر به خط دوم کدنویسی دقت کرده باشید ، وردپرس ابتدا دو متغیر به صورت Global در هسته ی خود ایجاد می کند به نام **$wp_filter** و **$wp_actions.**
در هر مرحله ای که ما یک اتصال جدید به نقاط جذب اضافه می کنیم یک آرایه به عضو این متغیر ها اضافه می شود. به همین شکل هست که وردپرس لیست تمامی اتصال ها به نقاط جذب را در هر مرحله دارد و در زمان به خصوص می تواند آن را صدا بزند.

بیایید با هم کار برنامه نویسی هوک های وردپرس را آغاز کنیم.

ابتدا یک افزونه وردپرس با **WP-CLI** ایجاد می کنیم به نام wp-hook تا بتوانیم کدهای خود را در آن تست کنیم.

```
$ wp scaffold plugin wp-hook
```

اکشن در وردپرس (WordPress Action)

ایجاد نقطه ی جذب در اکشن وردپرس با do_action

برای ایجاد نقطه ی جذب در اکشن وردپرس از تابع do_action استفاده می کنیم. این تابع دو مقدار را میتواند در خود جای دهد اولی نام نقطه ی جذب ، و دومی پارامتر هایی که میخواهید به عملیات ها پاس داده شود.

اگر فایل **wp-settings.php** را در پوشه اصلی وردپرس باز کنید ، در خطوط پایانی آن کد زیر را مشاهده می کنید:

```
/**
 * This hook is fired once WP, all plugins, and the theme are fully loaded and instantiated.
 *
 * Ajax requests should use wp-admin/admin-ajax.php. admin-ajax.php can handle requests for
 * users not logged in.
 *
 * @link https://codex.wordpress.org/AJAX_in_Plugins
 *
 * @since 3.0.0
 */
do_action( 'wp_loaded' );
```

همان طور که می بینید برنامه نویسان وردپرس در این قسمت از فایل یک نقطه ی جذب ایجاد کردند که آن را wp_loaded نام گزاری نمودند. در کامنت این تابع نوشته شده است که دلیل ایجاد این نقطه ی جذب ابن بوده که اگر کاربر بخواهد در زمانی که تمامی افزونه ها و کدهای وردپرس لود شدند و قبل از نمایش در مرورگر کاربر ، عملیاتی انجام شود این هوک بتواند وظیفه این کار را به دوش بکشد.

اتصال به نقطه ی اکشن با add_action

حال ما می خواهیم یک اکشن در این لحظه ایجاد کنیم کافیست از تابع add_action بهره بگیریم.

```
add_action('wp_loaded', 'show_my_site_name');
function show_my_site_name(){
        echo 'This is My WordPress WebSite';
        exit;
}
```

اگر وب سایت وردپرس خود را در مرورگر بارگزاری کنید خواهید دید که در تمامی صفحات تنها با پیغام This is My WordPress WebSite مواجه می شوید و دیگر چیز دیگری لود نخواهد شد زیرا ما در خط پایان از exit برای بسته شدن عملیات استفاده کردیم.

در تابع add_action ابتدا گفتیم که میخواهیم به کدام نقطه ی جذب متصل شویم ، سپس نام تابعی که میخواهد متصل شود را بیان کردیم که همان show_my_site_name بوده است.به همین سادگی وردپرس به محضی که به نقطه ی جذب wp_loaded برخورد کند این تابع را نیز فراخوانی می کند بعد از آن.

ترتیب اجرا شدن اکشن ها در وردپرس

حال فرض کنید ما چند عملیات برای اکشن wp_loaded میخواهیم قرار دهیم و آن ها را در زمان اجرا اولویت بندی کنیم. تابع add_action پارامتر سومی دارد که این قابلیت را به ما می دهد که حق تقدم را بر اساس عدد بیان کنیم.

```
add_action('wp_loaded', 'wporg_callback_run_me_late', 11);
add_action('wp_loaded', 'wporg_callback_run_me_normal');
add_action('wp_loaded', 'wporg_callback_run_me_early', 9);
add_action('wp_loaded', 'wporg_callback_run_me_later', 11);
```

بر اساس مستندات وردپرس عدد حق تقدم یا **priority** شامل قوانین زیر هست:

۱. این عدد در حالت پیش فرض **10** گرفته می شود. پس اگر چیزی برای پارامتر سوم وارد نکنیم عدد ۱۰ محسوب می شود.

۲. پیشنهاد همیشه بر این است که این عدد کم تر از ۱۰۰ باشد. اما به شخصه خیلی جاها دیدم که اعدادی مثل ۹۹۹ و چیزهای دیگر را هم استفاده می کنند تا مطمئن شوند کد آنها دقیقا آخر از همه ی عملیات ها اجرا شود.

الان در مثال بالا که ما چهار تا تابع را قلاب کردیم ترتیب اجرا به چه صورت هست ؟

۱. ابتدا تابع خط سوم اجرا می شود زیرا حق تقدم آن از همه پایین تر هست. خط دوم زمانی که عددی برای حق تقدم ندارد به معنای ۱۰ می باشد پس چون خط سوم حق تقدم آن ۹ هست زود تر از همه می باشد.

۲. در مرحله دوم تابع خط شماره ی دو اجرا می شود چون حق تقدم آن برابر با ۱۰ هست و کم تر از ۱۱ می باشد.

۳. در مرحله س سوم ما دو تابع داریم که هر دو با حق تقدم ۱۱ بیان شده است در اینجا چون خط شماره یک زودتر به سیستم معرفی شده است ابتدا خط شماره یک اجرا می شود و سپس خط شماره آخر.

مفهوم حق تقدم در اجرای هوک های وردپرس برای اکشن و فیلتر ها کاملا یکی می باشد.

استفاده از هوک های وردپرس در برنامه نویسی شیءگرایی و کلاس

فک کنم شما هم مثل من به حرفه ای تر شدن فکر می کنید. پس باید **برنامه نویسی OOP** در وردپرس را یاد بگیریم. اگر می خواهید در کلاس های PHP از هوک های وردپرس استفاده کنیم می بایست یک پارامتر به بخش دوم add_action اضافه کنیم تا آن را متوجه سازیم که تابع در کدام کلاس قرار دارد. مثال زیر را ببینید:

```php
/**
 * Class WP_HOOK_TEST.
 */
class WP_HOOK_TEST {

    /**
     * Constructor
     */
    public function __construct() {
        add_action( 'wp_loaded', array( $this, 'show_my_website_name' ), 11 );
    }

    /**
     * Handle wp_loaded action.
     */
    public function show_my_website_name() {
        // do stuff here...
    }
}

$wphooktest = new WP_HOOK_TEST();
```

در مثال بالا ما افزونه ی وردپرس خود را به شکل حرفه ای در قالب یک کلاس ایجاد کردیم. در متد فراخوانی آن زمانی که خواستیم اکشن وردپرس را معرفی کنیم این بار پارامتر دوم دیگر نام تابع نیست بله شامل یک **آرایه** می باشد که پارامتر اول بیان گر این است که این تابع در کدام کلاس قرار دارد که ما در اینجا عبارت **$this** قرار دادیم یعنی همین کلاس و در پارامتر دوم نام تابع در این کلاس بیان شده است.

پس شما به راحتی می توانید نام هر کلاس را در پارامتر ابتدایی قرار دهید و آن را قلاب کنید. اگر تابع شما به شکل استاتیک قرار داده است طبق اصول برنامه نویسی PHP دیگر استفاده از this کاملا

اشتباه می باشد. درآن زمان می توانید از عبارت **__CLASS__** یا تابع **get_called_class** در PHP استفاده کنید. که البته به ندرت این حالت اتفاق میفتد اما دانستن آن خالی از لطف نیست.

```
class WP_HOOK_TEST {
    public static function init() {
       add_action( 'save_post', array( __CLASS__, 'run_after_save_post_first' ), 11 );
       add_action( 'save_post', array( get_called_class(), 'run_after_save_post' ), 12 );
    }
    public static function run_after_save_post_first(){
    }
    public static function run_after_save_post(){
    }
}
WP_HOOK_TEST::init();
```

پاس دادن داده ها بین اکشن های وردپرس

بیایید به مثال ابتدایی این مقاله برگردیم. زمانی که یک کاربر در وردپرس ثبت نام می کند ، یک هوک اجرا می شود. این هوک در مستندات وردپرس به نام user_register نامگذاری شده است. سوال اینجاست اگر ما بخواهیم مقداری را برای یک کاربر زمانی که ثبت نام کرد به عنوان کد تخفیف ایجاد کنیم اصلا از کجا بفهمیم چه کاربری ثبت نام کرد در آن لحظه ؟

اینجاست که نقطه ی جذب اکشن در وردپرس امکان ایجاد متغیر و پاس دادن آن به عملیات را برای ما فراهم می کند. اگر تابع wp_insert_user در وردپرس را مشاهده کنید. در آن نقطه ی جذب اکشن بدین شکل بیان شده است:

```
1900        /**
1901         * Fires immediately after a new user is registered.
1902         *
1903         * @since 1.5.0
1904         *
1905         * @param int $user_id User ID.
1906         */
1907        do_action( 'user_register', $user_id );
1908    }
```

تابع do_action این قابلیت را دارد که بی نهایت پارامتر به آن متصل کنیم که از این متغیر ها در زمان قلاب کردن استفاده کنیم. در مثال بالا زمانی که کاربر در تابع wp_insert_user ثبت نام شد و در پایگاه داده اطلاعات آن قرار گرفت ، شناسه کاربر به ما پاس داده می شود تا بتوانیم تشخیص دهیم کدام کاربر هم اکنون قرار هست هوک ها روی آن اجرا شود.

مثلا اگر بخواییم یک کد تخفیف برای او ایجاد کنیم در قالب یک یوزر متا (user meta) خواهیم داشت:

```
add_action('user_register', 'add_new_coupon_after_register');
function add_new_coupon_after_register($user_id) {
        update_user_meta($user_id, 'coupon_code', 'realwp20');
}
```

متغیر user_id که در زمان ایجاد نقطه ی جذب ارائه گشت ، در زمان ایجاد اکشن های متفاوت به عنوان **ورودی تابع** ما میتوان استفاده شود.ما در این مثال user_id را دریافت کردیم و برای آن در وردپرس یک یوزر متا ایجاد کردیم به نام coupon_code و مقدار آن را realwp20 قرار دادیم.

تعداد ورودی پارامتر ها اکشن وردپرس

تعداد پارامتر هایی که ما می توانیم در زمان نقطه ی جذب ایجاد کنیم کاملا نامحدود هست. مثلا یک اکشن در وردپرس وجود دارد به نام save_post که در زمان بروز رسانی یک نوشته ، عملیاتی را میتواند قلاب کند. این اکشن دارای سه پارامتر هست:

```
do_action( 'save_post', $post_ID, $post, $update );
```

در این اکشن که در وردپرس ایجاد شده سه پارامتر باز میگردد. اولی شناسه پست ، دومی یک آرایه از داده های پست مثل تاریخ انتشار و در پارامتر سوم بیان می کند که آیا پست ایجاد شده است یا آپدیت (بروزرسانی).

فرض کنید ما یک وب سایت داریم که دارای چندین نویسنده هست و ما به عنوان مدیر قصد داریم هر زمانی که مطلبی ویرایش شد از حالت انتشار در وب سایت خارج شود و به حالت پیش نویس برود تا ما دوباره آن را بازبینی کنیم و انتشار دهیم ، این کار را می توانیم با کد زیر انجام دهیم:

```
add_action('save_post', 'set_post_status_to_draft', 10, 3);
function set_post_status_to_draft($post_ID, $post, $update) {
	if($update ===true) {
	  $my_post = array(
		'ID'           => $post_ID,
		'post_status'   => 'draft',
	  );
	  wp_update_post( $my_post );
	}
}
```

اگر تعداد ورودی اکشن ما بیش تر از یک پارامتر باشد. حتما می بایست تعداد پارامتر های ورودی در زمان add_action بیان شود. پس اگر بخواهیم آناتومی کامل تابع add_action را بیان کنیم خواهیم داشت:

```
add_action( string $tag, callable $function_to_add, int $priority = 10, int $accepted_args = 1 )
```

لیست پارامتر ها:

۱. پارامتر tag همان نام هوکی می باشد که قرار است به آن قلاب شویم.

۲. پارامتر function_to_add همان نام تابعی هست که قرار است اجرا شود و داده ها به آن پاس داده شود. که می تواند هم در کلاس باشد یا یک تابع ساده.

۳. پارامتر priority همان حق تقدم اجرای تابع هست که به صورت پیش فرض ۱۰ می باشد.

٤. مقدار accepted_args تعداد پارامتر هایی است که از هوک اصلی برای تابع ما به عنوان ورودی برمیگردد اگر تنها یک متغیر باشد یا اصلا متغیری نباشد مثل هوک wp_loaded نیازی به نوشتن نیست ولی اگر بیش از یک پارامتر باشد مثل اکشن save_post می بایست عدد وارد شود.

در زمان مواجه با پارامتر های زیاد در اکشن وردپرس راه کار چیست ؟

اگر تعداد پارامتر های ورودی برای یک اکشن زیاد باشد و ما تنها به یک یا چند پارامتر نیاز داشته باشیم. کد نویسی آن زیاد جالب به نظر نمی رسد. زیرا مجبوریم تمامی متغیر های ابتدایی را به عنوان ورودی تعریف کنیم. برای حل این مشکل وردپرس به جای do_action پیشنهاد می کند از تابع **do_action_ref_array** استفاده کنیم.

پس تفاوت do_action و do_action_ref_array در آن هست که در do_action تک تک پارامتر ها معرفی می شوند ، اما در تابع do_action_ref_array تمامی پارامتر ها در قالب یک آرایه با اندیس می تواند پاس داده شود.

```
// Define Hook
do_action('my_action', $arg1, $arg2, $arg3, $arg4 );

// Use Hook
add_action('my_action', 'my_callback', 10, 4 );
function my_callback( $arg1, $arg2, $arg3, $arg4 ) {
    //access values with $args1, $args2 etc.
}

// Define Hook with Array
do_action_ref_array('my_action', array($arg1, $arg2, $arg3, $arg4) );

// Use Hook Array
add_action('my_action', 'my_callback' );
function my_callback( $args ) {
    //access values with $args[0], $args[1] etc.
}
```

در مثال بالا کاملا تفاوت بین do_action و do_action_ref_array مشخص هست و می توانید نحوه ی استفاده از هر کدام را تشخیص دهید.

هوک های وردپرس مشخص کننده شروط اجرا کننده نیستن

یکی از نکاتی که باید دقت کنید آن است که کدهای هوک وردپرس هر زمان که توسط نقطه جذب صدا زده شوند اجرا خواهند شد و اصلا به موقعیت ها و درخواست ها توجهی ندارند. و این شمایید که باید این شروط را در آن لحاظ کنید. مثلا برای هوک user_register که برای ثبت نام کاربر هست. دیگر تفاوتی ندارد که این کاربر در وب سایت خودش فرم ثبت نام را پر کرده و یا اینکه مدیریت وردپرس آن را دستی در قسمت افزودن کاربر مدیریت وردپرس ایجاد نموده است.

```
add_action('user_register', 'add_new_coupon_after_register');
function add_new_coupon_after_register($user_id) {
	if(!is_admin()) {
		update_user_meta($user_id, 'coupon_code', 'realwp20');
	}
}
```

در مثال بالا ما شرط گزاشتیم که تنها در صورتی که کاربر در وب سایت ثبت نام کرده ن در بخش مدیریت ، کد تخفیف ایجاد شود.

فیلتر در وردپرس (WordPress Filter)

اگر مثال های اکشن را درک کرده باشید ، فیلتر ها براحتی برایتان قابل هضم هست. فقط با یک تفاوت اینکه فیلتر نیاز به بازگرداندن یک متغیر دارد و تنها اجرا کننده نیست.

ایجاد نقطه ی جذب در فیلتر وردپرس با apply_filters

بیایید مثالی از افزونه ووکامرس بزنیم برای فیلترها. در ووکامرس بخشی وجود دارد که شخص می تواند درصد گرفتن مالیات بر ارزش افزوده فاکتور ها را مشخص کند. برنامه نویس ووکامرس می بایست قابلیت توسعه پذیری این مقدار را فعال کند تا هر شخص بتواند هر مقدار عددی برای ارزش افزوده هست را وارد نماید.

تابعی داریم برای دریافت عدد مالیات مثل زیر:

```
function get_tax_number() {
	$tax_number = apply_filters( 'tax_number_in_factor', 9 );
	return $tax_number;
}
```

در این تابع ، متغیر tax_number فیلتر گزاری شده است که دیگر توسعه دهندگان بتوانند مقدار آن را تغییر دهند. تابع apply_filters کاملا مشابه do_action هست و دارای دو پارامتر هست. پارامتر ابتدایی نام هوک و پارامتر های بعدی متغیر هایی است که پاس داده می شوند.

در این مثال تفاوت بین do_action و apply_filters کاملا محسوس است . زیرا در زمان do_action ما این تابع را برابر با هیچ متغیر قرار نمیدادیم زیرا تنها یک رویداد بود که قرار است اجرا شود.اما در apply_filters ما مقدار نهایی را برابر با یک متغیر قرار میدهیم که در ادامه عملکرد تابع از آن استفاده کنیم.

در مواردی که تعداد پارامتر های ورودی برای نقطه ی جذب در فیلتر وردپرس زیاد باشد می توانید از تابع apply_filters_ref_array استفاده کنیـــد. که کامـلا ساز و کار آن مشـابه تابع همـــتای آن

do_action_ref_array می باشد و برای قرار دادن متغیر های آرایه ای در هوک های وردپرس است.

ایجاد فیلتر در وردپرس با add_filter

تابع add_filter نیز کاملا مشابه add_action می باشد. و همانند آن ، دارای چهار پارامتر نام تگ ، نام تابعی که قرار است صدا زده شود ، حق تقدم و تعداد پارامتر های ورودی آن فیلتر می باشد.

```
add_filter('tax_number_in_order', 'change_tax_number');
function change_tax_number($tax_number) {
        // Set New Value
        $tax_number = 20;

        // Return Data
        return $tax_number;
}
```

در فیلتر بالا ما متغیر tax_number را برابر با ۲۰ مقدار دهی کردیم و در آخر return کردیم که باز گردد به متغیر اصلی.

البته می توانستید تنها با نوشتن return 20 هم این کار را انجام دهید. تا خلاصه تر شود.

به عنوان مثال در هسته ی وردپرس یک فیلتر تعریف شده برای تغییر عنوان صفحات مدیریت وردپرس به نام admin_title که دارای دو پارامتر ورودی برای ما می باشد.

```
add_filter('admin_title', 'my_admin_title', 10, 2);
function my_admin_title($admin_title, $title)
{
    return 'RealWP'. ' | '. $title;
}
```

اگر کدهای بالا را ذخیره کنید می بینید که به ابتدای عنوان تمامی صفحات مدیریت وردپرس کلمه RealWP اضافه گردید.

از بحث فیلتر وردپرس برای مقادیر آری یا خیر بسیار استفاده می شود. به عنوان مثال در هسته ی وردپرس فیلتری وجود دارد که مشخص می کند ادمین بار یا نوار مدیریت وردپرس در وب سایت نمایش داده شود یا خیر. این فیلتر به نام show_admin_bar نامگذاری شده است که مقدار آن می تواند true یا false باشد.

برای اینکه در یک خط بتوان این مقدار را مشخص کرد میتوان از عبارت کوتاه استفاده کرد:

```
add_filter( 'show_admin_bar', '__return_false' );
```

نکات پیشرفته هوک های وردپرس

حذف یک هوک در زمان مشخص

سیستم هوک به شما این امکان را می دهد که در درخواست های خاص بتوانید برخی از اکشن ها و فیلتر ها را اجرا نکنید و به اصطلاح آنرا از لیست حذف کنید. برا اکشن ها میتوان از تابع remove_action و برای فیلتر ها میتوان از تابع remove_filter استفاده کرد.

هر دوی این تابع ها دو ورودی به خود میگیرند پارامتر اول نام هوک یا نقطه ی جذب می باشد و پارامتر دوم نام تابعی هست که نباید اجرا شود.

```php
add_action('admin_init', 'remove_title_hook');
function remove_title_hook(){
	global $pagenow;
	if($pagenow =="index.php") {
		remove_filter('admin_title', 'my_admin_title');
	}
}
```

در مثال بالا ابتدا یک اکشن در زمان لود شدن صفحات مدیریت قلاب کردیم که نام آن در هسته ی وردپرس admin_init هست. سپس در داخل آن شرطی ایجاد نمودیم که اگر کاربر در صفحه index.phpیا همان پیشخوان مدیریت وردپرس قرار داشت ،تابع my_admin_title اجرا نشود و از لیست هوک حذف گردد.

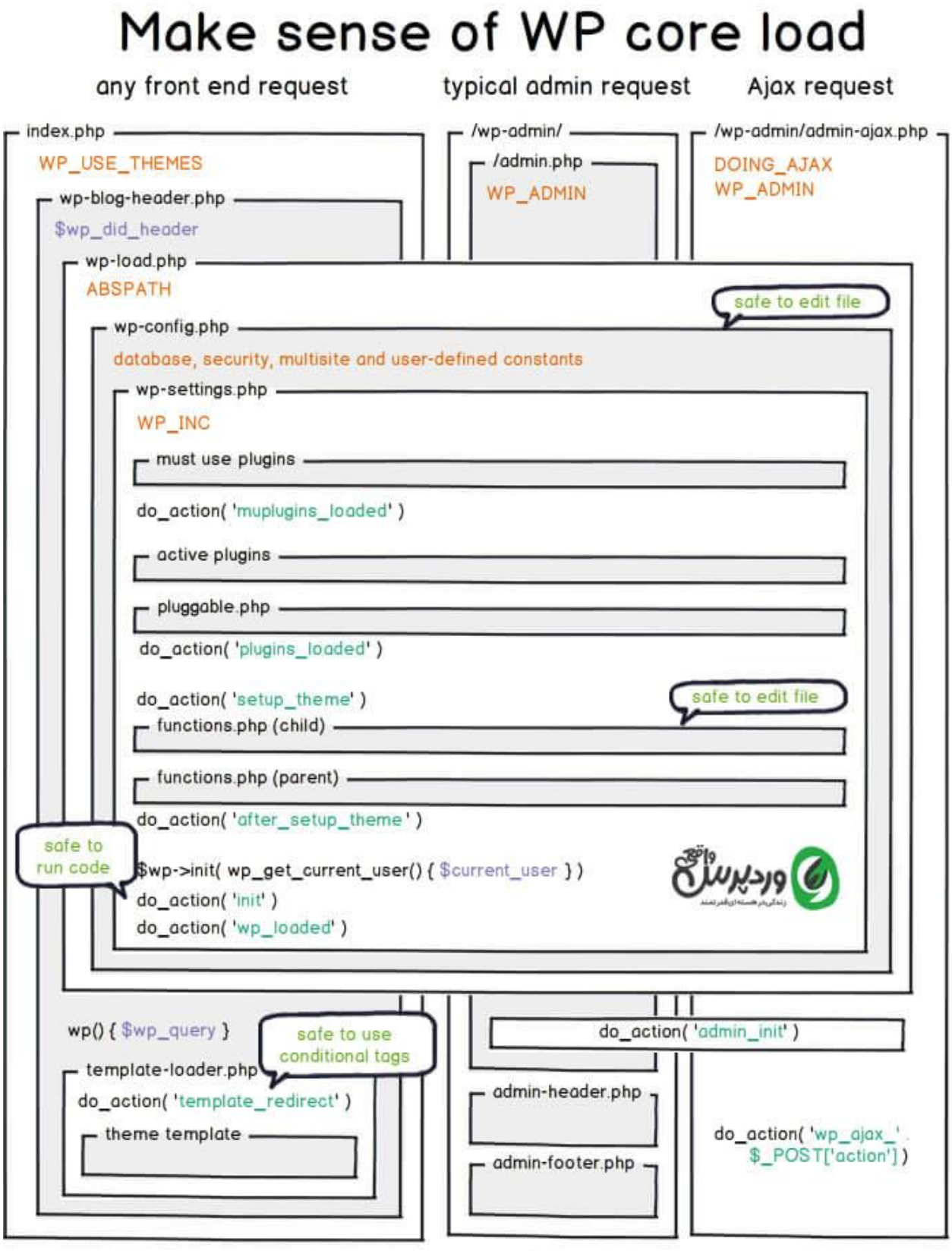
ترتیب اجرای هوک ها در هسته ی وردپرس به چه صورت هست ؟

یک سوالی که در کد بالا مورد بحث هست این می باشد که ما توانستیم در داخل اکشن admin_init برای هوک admin_title یک فیلتر قرار دهیم.آیا بر عکس این قضیه هم اتفاق می تواند بیفتد ؟

قطعا پاسخ این سوال خیر می باشد. زیرا اگر **ترتیب اجرای هوک ها در وردپرس** را بشناسیم. میدانستیم که ابتدا هسته ی هوک در بخش مدیریت یعنی admin_init لود می شود و بعد admin_title که در صفحه نمایش داده می شود.

برای اینکه بتوانیم کاملا مسیر لود شدن صفحات در وردپرس را بشناسیم کاملا بر میگردد به میزان تسلط شما بر هسته ی وردپرس و تجربه کاری شما.در این لینک لیستی از ترتیب اجرای اکشن ها زمانی که یک صفحه از صفحات وب سایت وردپرس نمایش داده می شود وجود دارد.

همچنین در زیر به صورت اینفوگرافیک مراحل لود شدن یک صفحه و اکشن هایی که اتفاق میفتد، نمایش داده شده است.مثلا هوک setup_theme همیشه بعد از هوک plugins_loaded اجرا می شود. پس این بدین معنی است که ما می توانیم در هوک plugins_loaded بر روی setup_theme تاثیر بزاریم و هوکی را از آن حذف یا اضافه کنیم.



لیست هوک های وردپرس را از کجا بیابیم ؟

لیست کامل هوک های وردپرس در مستندات وردپرس یا همان WordPress Code Reference وجود دارد و در هر بار آپدیت وردپرس این لیست نیز کاملا بروز رسانی می شود.تماما قابلیت جستجو و نمایش کد ها و توضیحات را دارد.

من به شخصه برای اینکه متوجه بشوم برای فلان بخش از وردپرس ، آیا هوکی وجود دارد یا خیر ؟ اولین کار در گوگل جستجو میکنم و اگر نشد هسته ی وردپرس را در یک واریشگر مثل PHPStorm یا visual Studio Code باز میکنم و سپس بین فایل های هسته ی وردپرس جستجو میکنم. این روش را حتما انجام دهید. زیرا گاها پیش آمده که متغیر هایی در وردپرس عوض شده ولی هنوز در پاسخ هایی که در StackOverflow یا وب سایت های مشابه آن داده می شود کاملا قدیمی هست.

راه حل سوم نیز استفاده از **WP-CLI** و پکیجی که بنده برای آن نوشتم می باشد.توسط پکـــیج WP-CLI Reference Command می توانید در همان خط فرمان سیستم به کل مستندات وردپرس دسترسی پیدا کنید.

استفاده از تابع های نامشخص یا Anonymous Function را به حداقل برسانید.

گاها در برخی از شرکت ها بین توسعه دهندگان وردپرس میبینم که هوک های وردپرس را بجای می نویسند. به عنوان مثال Anonymous اینکه به یک تابع قلاب کنند همان جا به صورت:

```
add_action('user_register', function($user_id) {
        update_user_meta($user_id, 'coupon_code', 'realwp20');
});
```

این کد اگر چه خلاصه تر هست و کار هم می کند. اما به هیچ عنوان توسط وردپرس و سازندگان پیشنهاد نمی شود.

دلیل این امر آن است که شما دیگر قادر نخواهید بود که عملیات استاندارد هوک ها رو روی این مورد کدنویسی ها اجرا کنید. مثلا اگر به همین صورت بنویسید دیگر قابلیت remove_action را نمیتوانیم برا آن استفاده کنیم.همچنین در توابعی که در ادامه اشاره میکنیم به عنوان تست هوک ها در وردپرس ، این مدل توابع اصلا در لیست استاندارد وردپرس قرار نخواهند گرفت.

نام هوک ها هم میتواند داینامیک باشد

ما گفتیم برای ایجاد یک نقطه ی جذب نیاز به یک شناسه داریم تا دیگر عملیات روی آن قلاب شود. اما می شود همان شناسه قلاب را هم داینامیک کرد خیلی ساده و کاربردی.به مثال زیر دقت کنید: در هسته ی وردپرس یک اکشن وجود دارد به نام save_post که این اکشن زمانی که یک محتوا بروز رسانی می شود عملیات هایش انجام می شود.اما همان طور که می دانید در وردپرس ما پست تایپ (Post Type) های مختلفی داریم مثل نوشته ها (Post) ، برگه ها (Page) ، رسانه ها (Attachment)، منوها (NavMenu) و یا اگر ووکامرس نصب کنید برای محصولات (Product) .

وردپرس برای این که کاربر راحت تر باشد و بخواهد فقط یک اکشن را در زمان بروز رسانی یک برگه استفاده کند آمده و شناسه را متغیر کرده.

```
/**
 * Fires once a post has been saved.
 *
 * The dynamic portion of the hook name, `$post->post_type`, refers to
 * the post type slug.
 *
 * @since 3.7.0
 *
 * @param int     $post_ID Post ID.
 * @param WP_Post $post    Post object.
 * @param bool    $update  Whether this is an existing post being updated or not.
 */
do_action( "save_post_{$post->post_type}", $post_ID, $post, $update );
```

اگر کد بالا را مشاهده کنید خواهید دید که نام هوک در بخش آخر شامل یک متغیر هست که بر اساس پست تایپ تغییر میکند.یعنی اگر ما بخواهیم در زمان مثلا بروز رسانی یک برگه یک اکشن ایجاد کنیم خواهیم داشت:

```
add_action('save_post_page', 'run_when_update_page', 10, 3);
function run_when_update_page($post_ID, $post, $update) {
// Code
}
```

یا مثلا در زمان آپدیت شدن یک محصول در ووکامرس اکشن ما اجرا شود:

```
add_action('save_post_product', 'run_when_update_product', 10, 3);
function run_when_update_product($post_ID, $post, $update) {
// Code
}
```

حذف تمامی هوک های قلاب شده به یک شناسه

مثال مبلغ مالیات که در بالا بیان شد را به یاد بیاورید . درآن ما یک فیلتر ایجاد کردیم که عدد را از ۹ به ۲۰ تغییر میداد. همیشه در بحث توسعه وردپرس به همین سادگی ها نیست. فرض کنید شما چند افزونه مختلف در زمینه ووکامرس نصب کردید و هر کدام طبق حق تقدم تاثیری در این عدد مالیات می گزارند و در نهایت عدد نهایی خروجی داده می شود.

اگر شما بخواهید تمامی هوک های که به این قضیه قلاب شده اند را حذف کنید مسلما دیگر تابع remove_filterیا remove_action به کار شما نمی یاد. زیرا این توابع فقط میتوانند یک تابع را از لیست خارج کنند.

برای این کار وردپرس دو تابع با نام های remove_all_filters و remove_all_actionsرا معرفی کرده است.توسط این دو تابع شما می توانید تمامی هوک های که قبل تر قلاب شده اند را خنثی کنید مثلا خواهیم داشت:

```
remove_all_filters('the_content');
```

در کد بالا ما تمامی فیلترهایی که بر روی the_content قلاب شده است را حذف کردیم و بعد از آن با خیالت راحت میتوانیم یک add_filter اضافه کنیم.

این دو تابع یک قابلیت جالب دیگر هم دارند ، مثلا شما میتوانید تمامی هوک هایی که با حق تقدم ۲۰ در وردپرس وجود دارد را حذف کنید. من همیشه برای برنامه زیری هایی که قبل از شروع کد نویسی روی بخش های مختلف انجام میدهم از این شیوه بسیار استفاده میکنم. و برای هر عدد یک شناسه به خصوص در ذهن خودم در نظر میگیرم.

```
remove_all_filters('admin_title', 15);
```

در مثال بالا تمامی هوک هایی که با حق تقدم ۱۵ به admin_title قلاب شده بودند از لیست خارج می شوند.

تشخیص قلاب شدن یک تابع بر هوک

این مورد هم بسیار برای من پیش آمده است ، زمانی هست که در پروژه های بزرگی که با وردپرس میزدم و هوک های بسیار در آن استفاده می شد. گاها نیاز به آن داشتم که در بخشی از کدها ابتدا چک کنم آیا قبلا این تابع یکبار قلاب شده است یا خیر.

برای این کار می توانید از تابع has_filter یا has_action در وردپرس استفاده کنید.

```
if(!has_filter('admin_title', 'my_admin_title')) {
        add_filter('admin_title', 'function_name');
}
```

در مثال بالا ابتدا چک کرده ایم که آیا قبلا تابع my_admin_title بر روی admin_title قلاب شده است یا خیر . اگر پاسخ منفی بود تابع function_name را بر روی admin_title قلاب می کند.

دریافت نام فیلتر یا اکشنی که در آن قرار داریم

گاهی پیش میاید که شما میخواهید یک **عملیات در چند هوک قلاب** شود. مثلا میخواهید کاربر زمانی که ثبت کرد و زمانی که یک فرمی را پر کرد و یا زمانی که وارد صفحه ای شد به آن کد تخفیف بدهید. یعنی سه قلاب بازگشت کند به یک تابع کد تخفیف دادن به کاربر .در این مواقع شما میتوانید با تابع current_filter یا current_action متوجه شوید که الان توسط کدام قلاب تابع در حال اجرا شدن می باشد.

```
function content_my_filter() {
    echo current_filter(); // 'the_content'
}
add_filter( 'the_content', 'content_my_filter' );
```

در تابع بالا مقدار تابع current_filter برابر با the_content هست زیاد تابع ما توسط فیلتر the_content در حال اجرا شدن است. اگر جای the_content هر هوک دیگری بود آن نمایش داده می شد.

قابلیت تکرار شوندگی در فیلتر و اکشن وردپرس

اگر دقت کرده باشید فیلتر ها در وردپرس روی یک متغیر تاثیر می گزارند. در مثالی که در مورد مقدار مالیات زدیم توسط فیلتر change_tax_number آن را به مقدار ۲۰ تغییر دادیم. حال سوال اینجاست اگر چندین بار همین فیلتر در صفحه ما اجرا شود چه اتفاقی می افتد ؟

برای فیلتر ها اتفاق خاصی رخ نمی دهد چون فیلتر تنها وظیفه تغییر در مقدار داده ها دارد. اما در اکشن وضعیت فرق میکند زیرا ما همیشه در اکشن عملیاتی را انجام می دهیم که تکرار آن عوامل به هیچ عنوان جایز نیست. مثلا فرض کنید عملیات ثبت پرداخت کاربر دو بار انجام شود و طبیعی است این قضیه در سیستم وردپرس کاملا اشتباه هست.

برای جلوگیری از این قضیه تنها در هوک های مدل اکشن تابعی ایجاد شده است به نام did_action که به ما اعلام میکند در حال حاضر چندمین بار هست که اکشن در حال اجرا شدن است.

این ویژگی مخصوصا برای اکشن های **wp_head**و **wp_footer** در قالب های وردپرس، که ما قصد داریم کد های Html به صفحه اضافه کنیم بسیار بدردمان میخورد.

```
add_action('wp_head', 'add_no_index_page');
function add_no_index_page(){
	if(is_search() and did_action('wp_head') == 1) {
		echo '<meta name="robots" content="noindex,nofollow">';
	}
}
```

در مثال بالا ما قصد داشتیم که کد متا تگ Noindex را در صفحه جستجوی وب سایت وردپرس قرار

دهیم تا ربات های گوگل آن را ایندکس نکنند. در زمان گزاشتن شرط بخشی را اضافه کردیم که تنها زمانی که برای بار اول wp_head فراخوانی می شود آن را قرار دهد. و اگر گاها باز هم در صفحه هوک wp_headدر جای دیگر فراخوانی شد ، دوباره این کد تولید نشود.

گرفتن گزارش و لاگ برنامه نویسی از هوک های وردپرس

اگر می خواهید از لیست تمامی توابعی که قلاب شده اند در وردپرس با خبر شوید ، تنها کافی است در پایان صفحه متغیر های wp_filter و wp_actions را خروجی بگیرید تا لیست کامل برایتان نمایش داده شود.

اگر در وب سایت وردپرس هستید می توانید آن را به هوک wp_footer قلاب کنید و اگر در محیط ادمین وردپرس هستید می توانید از هوک admin_footer استفاده کنید.

```
add_action('wp_footer', 'show_all_hooks');
function show_all_hooks(){
	global $wp_filter;

	echo '<pre>';
	print_r($wp_filter);
	echo '</pre>';
}
```

متغیر wp_filter یک آرایه هست که هر هوک در آن به صورت کلید و لیست عملیات قلاب شده عضوی از آن آرایه می باشند.

همچنین اگر میخواهید در زمان اجرا شدن هوک ها آن را هدف گزاری کنید و تست های سرعت بگیرید بهترین کار استفاده از تابع doing_actionو doing_filter می باشد. توسط این تابع شما قادرید متوجه شوید الان چه هوکی شروع به انجام عملیات خود در سیستم وردپرس کرده است.

مثلا برای این که تست بگیرید در حال حاضر هوک های مربوط با save_post آغاز به انجام عملیات کردن خواهیم داشت:

```
if ( doing_action( 'save_post' ) ) {
    // Do awesome here.
}
```

برنامه نویسی استاندارد وردپرس مبتنی بر هوک چیست ؟

بعد از تمامی مطالبی که در بالا از هوک های وردپرس آموختیم.حال وقت آن رسیده است که با یک مثال کاملا متوجه شویم که چگونه حتما باید برنامه نویسی و توسعه در وردپرس بر مبنای هوک های وردپرس باشد تا کاملا قابلیت پیشرفت و دستکاری برای کاربران برای آن در نظر گرفته شود.

میخواهیم چه کار کنیم ؟ قصد داریم یک فرم در صفحه ایجاد کنیم که کاربر بعد از تکمیل آن فرم ، در وب سایت وردپرس ما ثبت نام کند و اطلاعات وارد پایگاه داده می شود.
این فرم باید کاملا قابلیت افزودن یا کاستن فیلد ها را داشته باشد و به صورت پیش فرض ما سه فیلد ایمیل ، شماره موبایل و رمز عبور را درخواست می کنیم.

ابتدا یک تابع می نویسیم که بتواند برای ما فرم ایجاد کند ، اما فیلد های آن فرم را در قالب یک نقطه ی جذب فیلتر می بریم:

```php
add_shortcode( 'register-form', 'create_register_form' );
function create_register_form(){

	// Set Default Input
	$default_input = array(
		'user_login' => array('type' => 'text', 'label' => 'UserName'),
		'user_mobile' => array('type' => 'tel', 'label' => 'Mobile'),
		'user_pass' => array('type' => 'password', 'label' => 'Password')
	);

	// Apply filter List Of input
	$inputs = apply_filters('register_forms_inputs', $default_input);

	$r = '<form action="" method="post">';
	$r .= '<table>';

	foreach($inputs as $input_name => $input) {
		$r .= '<tr>';
		$r .= '<td>';
		$r .= $input['label'];
		$r .= '</td>';
		$r .= '<td>';
		$r .= '<input type="'.$input['type'].'" name="'.$input_name.'">';
		$r .= '</td>';
		$r .= '</tr>';
	}

	$r .= '<tr><td colspan="2"><iput type="submit" value="Register User"
name="register_wordpress_user"></td></tr>';
	$r .= '</form>';

	return $r;
}
```

کافی است شورت کد [register-form] که در کد بالا ایجاد کرده ایم را در ویرایشگر یک برگه وردپرس ذخیره کنید و نتیجه را ببینید.

مهم ترین بخش این کد ورودی های فرم هستند که در قالب یک آرایه ایجاد شده و به توسعه دهندگان اجازه آن را می دهد که توسط فیلتر register_forms_inputs ، فیلد هایی را به آن اضافه یا کم کنند.

مثلا فرض کنید من میخوام فیلد موبایل را از این فرم ثبت نام حذف کنم و فیلد جدید برای دریافت نام و نام خانوادگی ایجاد کنم پس در افزونه ی خودم می نویسم:

```php
add_filter('register_forms_inputs', 'change_register_form_input');
function change_register_form_input($default_input) {

        // Remove mobile input
        unset($default_input['user_mobile']);

        // Add Full Name Field
        $default_input['user_full_name'] = array('type' => 'text', 'label' => 'FullName');

        // Return Data
        return $default_input;
}
```

حال نتیجه را مشاهده کنید و ببینید که چقدر زیبا میتوان هسته ی وردپرس را توسعه داد و از آن لذت برد.