Enable a Flagged Feature

Developers must take certain actions to ensure a newly developed feature works with a feature flag.

Before you can enable a flagged feature, you must set up feature flags on your site by adding the necessary code. See the Setting Up Feature Flags on a Commerce Site section of the Feature Flags overview for more information.

Coding the Feature Flag Trigger

The code for the new feature must include a means of being triggered by the feature flag created for it. For example, imagine that your development team has created a new custom checkout flow that you want to test on your site using a feature flag. The code for that new checkout flow would need to include something like the following to trigger the feature flag:

```javascript
// The function that is run when the checkout button is pressed
function onCheckoutButtonClick(...){
    // Check on the flag created in the UI
    if(featureFlags['New Checkout Flow Flag']['enabled']){
        triggerNewCheckoutFlow(...);
    } else{
        triggerOldCheckoutFlow(...);
    }
}
```

In this example, the new checkout flow is developed in the `triggerNewCheckoutFlow()` function, and the old checkout flow is contained in the `triggerOldCheckoutFlow()` function. `New Checkout Flow Flag` is the name of the feature flag you made in Monetate. (See Create a Feature Flag for more information.)

Turning on a Flagged Feature

To get the configuration for a feature flag, navigate to the `/features/1/flags` endpoint using your account tuple (for example, `https://sb.monetate.net/features/1/flags/a-123456/p/commerce.com/`).

The JSON entry returned shows the name and percentage chosen for available features.

```json
{"Test-red": {"percent":50}}
```

Monetate provides helper functions that allow you to effectively parse this data.
window.monetate.featureFlag

{hash: f, isFeatureEnabled: f, parseTogglesConfig: f}
  hash: f q(a,b)
  isFeatureEnabled: f r(a,b,c)
  parseTogglesConfig: f (a,b)

parseTogglesConfig(config, mid) adds an enabled field in the JSON entry with true or false based on the Monetate ID (MID) so that you can have the entire config ready for use.

var config = {'Foo': {'percent': 50}, 'Baz': {'percent': 100}};
var mid = "2.162744605.1589391933748";
console.log("Config as received: ", config);

Config as received: Foo: {...}, Baz: {...}
  Baz: {percent: 100}
  Foo: {percent: 50}
  _proto_: Object

monetate.featureFlag.parseTogglesConfig(config, mid);
console.log("Config as parsed: ", config);

Config as parsed: Foo: {...}, Baz: {...}
  Baz: {percent: 100, enabled: true}
  Foo: {percent: 50, enabled: false}
  _proto_: Object

isFeatureEnabled(featName, config, mid) is similar to parseTogglesConfig but returns true or false for an individual feature and leaves the config unchanged.

var config = {'Foo': {'percent': 50}, 'Baz': {'percent': 100}};
var mid = "2.162744605.1589391933748";
var fooEnabled = monetate.featureFlag.isFeatureEnabled('Foo', config, mid);
var bazEnabled = monetate.featureFlag.isFeatureEnabled('Baz', config, mid);
console.log("The feature 'Foo' is enabled? ", fooEnabled);
console.log("The feature 'Baz' is enabled? ", bazEnabled);

The feature 'Foo' is enabled? false
The feature 'Baz' is enabled? true

hash(mid, featName) ensures that every visitor gets a consistent percentile. By taking the MID and a name of a feature and turning it into a percentage, it ensures uniqueness and consistency amongst visitors, even distributions, and a standard for re-creation on other platforms. If used on another platform, the hash function can be recreated for a consistent user experience because the percentile is the same. The hash modulo 100 gives you the percentile and represents an enabled feature if less than the flag's percentage or disabled if equal to or greater than the flag's percentage.
var featName = "featureName";
var mid = "2.162744605.1589391933748";
var percentile = monetate.featureToggle.hash(mid, featName) % 100;
console.log("The user " + mid + " is in the " + percentile + " percentile for " + featName);

The user 2.162744605.1589391933748 is in the 67 percentile for featureName