**Getting Started with Underscore.js**

[Underscore.js](http://underscorejs.org/) is a JavaScript library, written by [Jeremy Ashkenas](https://en.wikipedia.org/wiki/Jeremy_Ashkenas), that provides functional utilities for a variety of use cases that we, as developers, may come across when facing a web project.

It makes for code which is easier to read:

\_.isEmpty({});

// true

You’ll need to grab a copy of the library, for example from your favourite CDN

<script src= "https://cdnjs.cloudflare.com/ajax/libs/underscore.js/1.8.3/underscore-min.js"></script>

**\_.each: Write Readable Loops**

There is not a single project that does not have something similar to this snippet at some point in the code:

var artists = ['Pharrel Williams', 'Led Zeppelin', 'Rolling Stones'];

for(var i = 0; i < artists.length; i++) {

console.log('artist: ' + artists[i]);

}

Underscore enables you to write equivalent code, using a syntax that is more readable:

var artists = ['Pharrel Williams', 'Led Zeppelin', 'Rolling Stones'];

\_.each(artists, function(artist, index, artists) {

console.log('artist: ' + artist);

});

**\_.each() takes two parameters:**

* The array (or object) to iterate over.
* A callback function.

For each element in our array \_.each() will invoke the callback function (referred to in the documentation as *iteratee*). Inside the callback we get access to a further three parameters:

* The value of the array for the current iteration index (artist). For example, for the snippet above we’d get “Pharrel Williams” for the first iteration.
* The number of the current iteration (index), which in our case will vary from 0 to 2.
* The array that we are iterating through (artists).

As you can see the code is more readable and we can access the individual elements in the array without the need for artists[i], as we saw in the example that used a for loop.

Result:

Pharrel Williams

Led Zeppelin

Rolling Stones

## **Collection Functions (Arrays or Objects)**

**each**\_.each(list, iteratee, [context]) Alias: **forEach**

It iterates over a **list** of elements, yielding each in turn to an **iteratee** function. The **iteratee** is bound to the **context** object, if one is passed. Each invocation of **iteratee** is called with three arguments: (element, index, list). If **list** is a JavaScript object, **iteratee**'s arguments will be (value, key, list). Returns the **list** for chaining.

\_.each([1, 2, 3], alert);

=> alerts each number in turn...

\_.each({one: 1, two: 2, three: 3}, alert);

=> alerts each number value in turn...

*Note: Collection functions work on arrays, objects, and array-like objects such as* arguments, NodeList *and similar.*

*But it works by duck-typing, so avoid passing objects with a numeric* length *property. It's also good to note that an* each *loop cannot be broken out of — to break, use* ***\_.find*** *instead.*