An Example of JavaScript Closure Using The Date Object (http://vegibit.com/an-example-of-javascript-closure-using-the-date-object/)

(http://vegibit.com/an-example-of-javascript-closure-using-the-date-object/)

You can never have enough examples of closure in JavaScript. In this quick tutorial, we'll take a look at a few examples of accessing variables both *with* and *without closure* to show how scoping changes in this type of code. The purpose of this exercise is to have a look at the difference between standard JavaScript functions (http://vegibit.com/javascript-string-functions/) vs nested type functions that create closure and allow variables to persist in memory longer than they would if not in a closure. We'll do this using the Date object in JavaScript.

Running the date example without closure

This first example is simply to demonstrate calling a function two times, with a difference in time to show that on each function call, a different value is accessed in the `date` variable. Let's walk through the code. We

first define a function named `withoutClosure()`. This function assigns a new date object to the `date` variable. We then simply return the current milliseconds using `date.getMilliseconds()`. The next two lines fetch the div we are going to populate with this value, and we call the function to actually put some milliseconds inside the div. After this, we use a `setTimeout()` function so that we can call this function again, but 500 milliseconds later. By doing so, we demonstrate that the milliseconds have changed – hence we are not accessing the same variable. A new date object gets created on each call of the `withoutClosure()` function. Go ahead and click the button as many times as you like to see if you can get the milliseconds to be equal. Try as you might, you will never get the same value twice!

```
function withoutClosure() {

    var date = new Date();

    return date.getMilliseconds();
}

var withoutclosurehtml = document.getElementById('withoutclosure');
withoutclosurehtml.innerHTML = withoutClosure();

window.setTimeout(function () {
    withoutclosurehtml.innerHTML += '<br>' + withoutClosure();
}, 500);
```

Click Clear

Running the date example with closure

Now we will tackle the same problem, but while using a *nested function* which will produce **closure** around the `date` object. This is a great example of how a variable will “hang around in memory” for as long as is

required by the inner function which is producing the closure. In this example, we can see that there are two times which the `withClosure()` function gets called. The second time the `withClosure()` function gets called, it is within a `setTimeout()` function call. This is simply to add a delay to the second call of the `withClosure()` function. In reality, we are actually calling the anonymous function within the `withClosure()` function. This is because `withClosure()` itself is actually *returning a function*, not a computed value. Remember, in JavaScript, **functions are first class citizens**, which means we can pass them around just like any other value or variable. So to be fair, it is actually the method `date.getMilliseconds()` that gets called twice in the program. The first time around when it gets called, it fetches the milliseconds value of the date object. 500 milliseconds later, it gets called again. In this example, no matter what you do – you will always get the same exact value for milliseconds for each of the two output values. Try to run the program as many times as you like to get a different value, they will always be the same! This is because of **closure**, and the fact that on each call, you are accessing the same date object. This differs from the first example where on each call, a new date object is being accessed, not the same one. Check out the Doug Crockford article (http://vegibit.com/douglas-crockford-the-good-parts-examples/#closure) to find more fun examples of closure.

```
function withClosure() {

    var date = new Date();

    return function () {
        return date.getMilliseconds();
    }
}

var withclosurehtml = document.getElementById('withclosure');
var closure = withClosure();
withclosurehtml.innerHTML = closure();

window.setTimeout(function () {
    withclosurehtml.innerHTML += '<br>' + closure();
}, 500);
```

Click Clear

Running the example with a module like closure pattern

The final example of closure with the date object is just a slight modification of the second example, but in a more module like pattern. In this example, instead of returning a function out of withClousre2(), we actually return an object where the key holds a reference to the nested function within `withClosure2()`. This example works mostly the same as example two above, so we will not step through the code line by line. Again, you can run the program as many times as you like, and the value produced by the call to `date.getMilliseconds()` will always be the same. Again this is because via closure, the same date object is being access on each function call.

```
var withClosure2 = function () {

    var date = new Date();

    var nestedfunction = function () {
        return date.getMilliseconds();
    }

    return {
        nestedfunction: nestedfunction
    }

};

var withclosurehtml2 = document.getElementById('withclosure2');
var closure = new withClosure2();
withclosurehtml2.innerHTML = closure.nestedfunction();

window.setTimeout(function () {
    withclosurehtml2.innerHTML += '<br>' + closure.nestedfunction();
}, 500);
```

Click Clear

Final Example

As one final example, we'll simply add a few more calls to the `date.getMilliseconds()` method and demonstrate that it is accessing the same date object each time. This is a good example of how closure will persist those variables you need for as long as is required. We could add a hundred calls to the `date.getMilliseconds()` method and the same result would be returned every time.

```
var withClosure3 = function () {
    var date = new Date();
    var nestedfunction = function () {
        return date.getMilliseconds();
    };

    return {
        nestedfunction: nestedfunction
    }

};

var withclosurehtml3 = document.getElementById('withclosure3');
var closure = new withClosure3();
withclosurehtml3.innerHTML = closure.nestedfunction();

window.setTimeout(function () {
    withclosurehtml3.innerHTML += '<br>' + closure.nestedfunction();
}, 500);

window.setTimeout(function () {
    withclosurehtml3.innerHTML += '<br>' + closure.nestedfunction();
}, 1000);

window.setTimeout(function () {
    withclosurehtml3.innerHTML += '<br>' + closure.nestedfunction();
}, 1500);

window.setTimeout(function () {
    withclosurehtml3.innerHTML += '<br>' + closure.nestedfunction();
}, 2500);
```

Click Clear

An Example of JavaScript Closure Using The Date Object Summary

This was a fun little example of exploring **JavaScript Closure** with a few different examples using the built in Date object in JavaScript. Have Fun!

Related

(http://vegibit.com/javascript-prototype-pattern/)

JavaScript Prototype Pattern (http://vegibit.com/javascript-prototype-pattern/)
In "web development"

(http://vegibit.com/mithril-javascript-tutorial/)

Mithril JavaScript Tutorial (http://vegibit.com/mithril-javascript-tutorial/)
In "web development"

(http://vegibit.com/using-variables-and-conditionals-in-javascript/)

Using Variables and Conditionals in JavaScript (http://vegibit.com/using-variables-and-conditionals-in-javascript/)
In "web development"

javascript (http://vegibit.com/tag/javascript/)

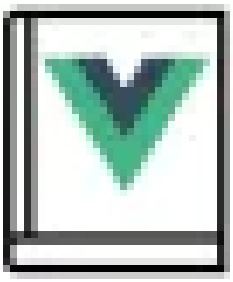
← Previous (http://vegibit.com/24-popular-python-repositories/)

Next → (http://vegibit.com/javascript-prototype-pattern/)

Search

Top Posts & Pages

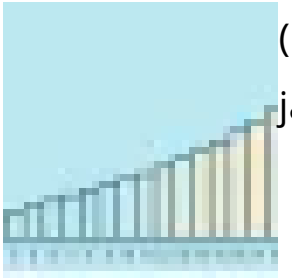
(http://vegibit.com/vue-js-tutorial/)



Vue.js Tutorial (http://vegibit.com/vue-js-tutorial/)

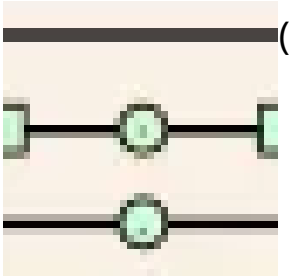
(http://vegibit.com/laravel-repository-pattern/)

Laravel Repository Pattern (http://vegibit.com/laravel-repository-pattern/)



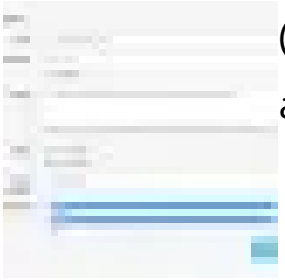
(http://vegibit.com/create-a-bar-chart-with-d3-javascript/)

Create A Bar Chart With D3 JavaScript (http://vegibit.com/create-a-bar-chart-with-d3-javascript/)



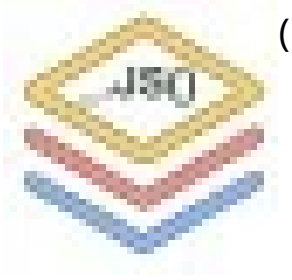
(http://vegibit.com/json-in-laravel/)

JSON in Laravel (http://vegibit.com/json-in-laravel/)



(http://vegibit.com/create-form-elements-using-laravel-and-bootstrap/)

Create Form Elements Using Laravel and Bootstrap (http://vegibit.com/create-form-elements-using-laravel-and-bootstrap/)



(http://vegibit.com/underscore-js-map-function/)

Underscore JS Map Function (http://vegibit.com/underscore-js-map-function/)

(http://vegibit.com/laravel-eloquent-orm-tutorial/)

Laravel Eloquent ORM Tutorial (http://vegibit.com/laravel-eloquent-orm-tutorial/)

(http://vegibit.com/php-string-functions/)

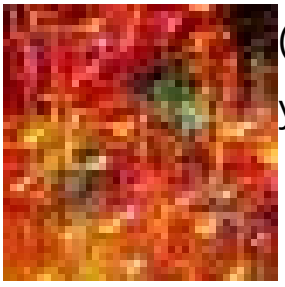
The Ultimate PHP String Functions List (http://vegibit.com/php-string-functions/)

(http://vegibit.com/getting-started-with-testing-in-laravel/)

Getting Started With Testing In Laravel (http://vegibit.com/getting-started-with-testing-in-laravel/)

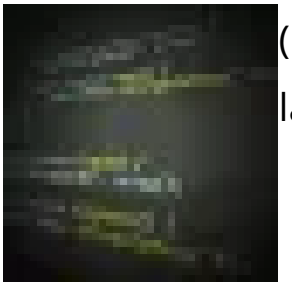
(http://vegibit.com/laravel-cache-tutorial/)

Laravel Cache Tutorial (http://vegibit.com/laravel-cache-tutorial/)



(http://vegibit.com/super-easy-crud-with-gii-and-the-yii2-framework/)

Super Easy CRUD With Gii And The Yii2 Framework (http://vegibit.com/super-easy-crud-with-gii-and-the-yii2-framework/)

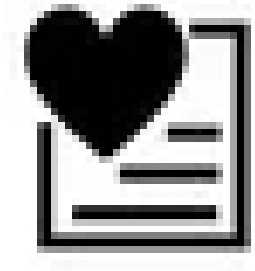


(http://vegibit.com/many-to-many-relationships-in-laravel/)

Many to Many Relationships in Laravel (http://vegibit.com/many-to-many-relationships-in-laravel/)

(http://vegibit.com/laravel-file-structure/)

Laravel File Structure (http://vegibit.com/laravel-file-structure/)



(http://vegibit.com/how-to-write-a-listicle/)

14 Steps To Write A Great Listicle (http://vegibit.com/how-to-write-a-listicle/)

(http://vegibit.com/how-to-install-the-advanced-template-in-yii2/)

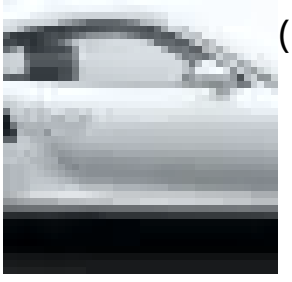
How To Install The Advanced Template In Yii2 (http://vegibit.com/how-to-install-the-advanced-template-in-yii2/)

(http://vegibit.com/most-useful-php-array-functions/)

The 27 Most Useful PHP Array Functions You Need To Know! (http://vegibit.com/most-useful-php-array-functions/)

(http://vegibit.com/upgrading-vuejs/)

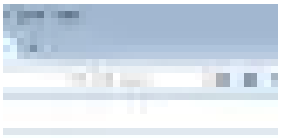
Upgrading VueJS (http://vegibit.com/upgrading-vuejs/)



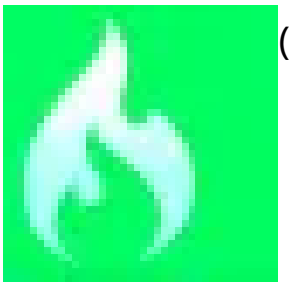
(http://vegibit.com/what-is-a-laravel-interface/)

What is a Laravel Interface? (http://vegibit.com/what-is-a-laravel-interface/)

(http://vegibit.com/vue-js-for-interactive-web-interfaces/)

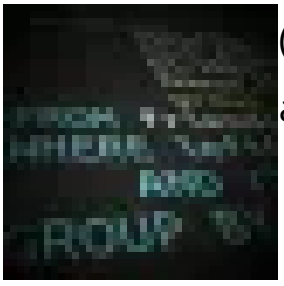


Vue.js for Interactive Web Interfaces (http://vegibit.com/vue-js-for-interactive-web-interfaces/)



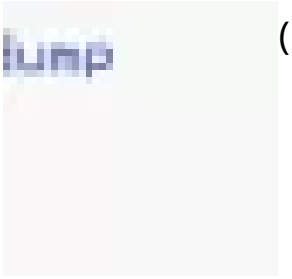
(http://vegibit.com/codeigniter-model-tutorial/)

Codeigniter Model Tutorial (http://vegibit.com/codeigniter-model-tutorial/)



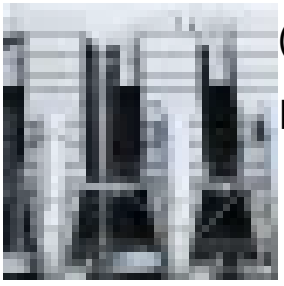
(http://vegibit.com/mysql-group-by-having-limit-offset-and-more/)

MySQL Group By Having Limit Offset and More! (http://vegibit.com/mysql-group-by-having-limit-offset-and-more/)



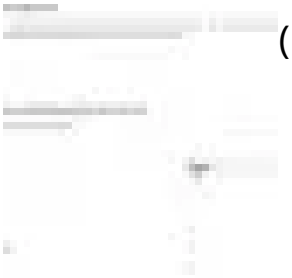
(http://vegibit.com/custom-helper-functions-in-laravel/)

Custom Helper Functions in Laravel (http://vegibit.com/custom-helper-functions-in-laravel/)



(http://vegibit.com/creating-static-and-dynamic-web-pages-in-laravel/)

Creating Static And Dynamic Web Pages In Laravel (http://vegibit.com/creating-static-and-dynamic-web-pages-in-laravel/)



(http://vegibit.com/developing-with-vuejs-and-php/)

Developing With VueJS and PHP (http://vegibit.com/developing-with-vuejs-and-php/)

(http://vegibit.com/mithril-javascript-tutorial/)

Mithril JavaScript Tutorial (http://vegibit.com/mithril-javascript-tutorial/)

(http://vegibit.com/easy-php-dates-and-times-with-carbon/)

Easy PHP Dates and Times With Carbon (http://vegibit.com/easy-php-dates-and-times-with-carbon/)

(http://vegibit.com/laravel-collections-tutorial/)

Laravel Collections Tutorial (http://vegibit.com/laravel-collections-tutorial/)

 © 2015 Vegibit.com