الكولكشن

undefinedcount()2undefined

> The count method returns the total number of items in the collection:

> بترجع عدد المصفوفة

```
$collection = collect([1, 2, 3, 4]);

$collection->count();

// 4
```

undefinedcontains()3undefined

> The contains method determines whether the collection contains a given item:

> بتبحث عن كلمة داخل المصوفة وبترجع صح او خطأ علي حسب الكلمة

```
$collection = collect(['name' => 'Desk', 'price' => 100]);

$collection->contains('Desk');

// true

$collection->contains('New York');

// false
```

```php
$collection = collect([1, 2, 3, 4, 5]);

$collection->contains(function ($value, $key) {
    return $value > 5;
});

// false
```

undefineddiff()4undefined

> The diff method compares the collection against another collection or a plain PHP array based on its values. This method will return the values in the original collection that are not present in the given collection:

> دي بترجع الاختلاف مابين الدالتين

```php
$collection = collect([1, 2, 3, 4, 5]);

$diff = $collection->diff([2, 4, 6, 8]);

$diff->all();

// [1, 3, 5]
```

undefineddiffAssoc()5undefined

> The diffAssoc method compares the collection against another collection or a plain PHP array based on its keys and values. This method will return the key / value pairs in the original collection that are not present in the given collection:

> دي بترجع الاختلاف مابين الدوال مع المفاتيح الرئسية للدوال علي خلاف التانية بتراجع القيم بس
>

```php
$collection = collect([
    'color' => 'orange',
    'type' => 'fruit',
    'remain' => 6
]);

$diff = $collection->diffAssoc([
    'color' => 'yellow',
    'type' => 'fruit',
    'remain' => 3,
    'used' => 6
]);

$diff->all();

// ['color' => 'orange', 'remain' => 6]
```

undefinedduplicates()6undefined

> the duplicates method retrieves and returns duplicate values from the collection:

> دي بترجع التكرر الي في الدوال

```php
$collection = collect(['a', 'b', 'a', 'c', 'b']);

$collection->duplicates();

// [ 2 => 'a', 4 => 'b' ]
```

undefinedeach()7undefined

> The each method iterates over the items in the collection and passes each item to a callback:
> دي بتعمل دوران علي المصفوفة كلها مع تنفيذ الشرط بتاعها

```
$collection->each(function ($item, $key) {
    //
});
```

> If you would like to stop iterating through the items, you may return false from your callback:
> بتقدر تخرج من الدوران من خلال الشرط ده

```
$collection->each(function ($item, $key) {
    if (/* some condition */) {
        return false;
    }
});
```

every()

> The every method may be used to verify that all elements of a collection pass a given truth test:
> لو اي حالة حصل فيها تعارض حيرجع بخطا بخلاف الدالة التانية

```
collect([1, 2, 3, 4])->every(function ($value, $key) {
    return $value > 2;
});

// false
If the collection is empty, every will return true:

$collection = collect([]);

$collection->every(function($value, $key) {
```

```
    return $value > 2;
});

// true
```

filter()

> The filter method filters the collection using the given callback, keeping only those items that pass a given truth test:

> بيرجع بمصفوفة تحتوي علي الاشياء الصحيحة والمحققة للشرط

```
$collection = collect([1, 2, 3, 4]);

$filtered = $collection->filter(function ($value, $key) {
    return $value > 2;
});

$filtered->all();

// [3, 4]
```

> If no callback is supplied, all entries of the collection that are equivalent to false will be removed:

```
$collection = collect([1, 2, 3, null, false, '', 0, []]);

$collection->filter()->all();

// [1, 2, 3]
```

first()

> The first method returns the first element in the collection that passes a given truth test:

```
collect([1, 2, 3, 4])->first(function ($value, $key) {
    return $value > 2;
});

// 3
```

بترجع بأول عنصر محقق للشرط الي بيمرر من خلالها

firstWhere()

The firstWhere method returns the first element in the collection with the given key / value pair:

```
$collection = collect([
    ['name' => 'Regena', 'age' => null],
    ['name' => 'Linda', 'age' => 14],
    ['name' => 'Diego', 'age' => 23],
    ['name' => 'Linda', 'age' => 84],
]);

$collection->firstWhere('name', 'Linda');

// ['name' => 'Linda', 'age' => 14]
```

بترجع بأول مصفوفة محققة للشرط الخاص بيها

forPage()

The forPage method returns a new collection containing the items that would be present on a given page number. The method accepts the page number as its first argument and the number of items to show per page as its second argument:

```
$collection = collect([1, 2, 3, 4, 5, 6, 7, 8, 9]);

$chunk = $collection->forPage(2, 3);

$chunk->all();

// [4, 5, 6]
```

دي بترجع بعدد معين من المصفوفة
اول رقم رقم البداية الي بتبدي منو المصفوفة
ثاني رقم هو الرقم الي بترجعو من المصفوفة

get()

The get method returns the item at a given key. If the key does not exist, null is returned:

```
$collection = collect(['name' => 'taylor', 'framework' => 'laravel']);

$value = $collection->get('name');

// taylor
```

بترجع بقيمة الاسم الممر ليها من خلال المفتاح الرئيسي للدالة

implode()

The implode method joins the items in a collection. Its arguments depend on the type of items in the collection. If the collection contains arrays or objects, you should pass the key of the attributes you wish to join, and the "glue" string you wish to place between the values:

```
$collection = collect([
    ['account_id' => 1, 'product' => 'Desk'],
    ['account_id' => 2, 'product' => 'Chair'],
]);

$collection->implode('product', ', ');

// Desk, Chair
```

بتحول المصفوفة الي نص

isEmpty()

```
The isEmpty method returns true if the collection is empty; otherwise, false is
returned:

collect([])->isEmpty();

// true
```

الشرط محقق المصفوفة فارغة

isNotEmpty()

The isNotEmpty method returns true if the collection is not empty; otherwise, false is returned:

```
collect([])->isNotEmpty();

// false
```

الشرط غير محقق المصفوفة فارغة

except()

The except method returns all items in the collection except for those with the specified keys:

```
$collection = collect(['product_id' => 1, 'price' => 100, 'discount' => false]);

$filtered = $collection->except(['price', 'discount']);

$filtered->all();

// ['product_id' => 1]
```

معدا
دي معناها بترجع المصفوفة كاملة معدا المفاتيح الي بتتمرر ليها

only()

undefinedThe only method returns the items in the collection with the specified keys:undefined

```
$collection = collect(['product_id' => 1, 'name' => 'Desk', 'price' => 100,
'discount' => false]);

$filtered = $collection->only(['product_id', 'name']);
```

```
$filtered->all();

// ['product_id' => 1, 'name' => 'Desk']
```

فقط
بترجع كل القيم المررة ليها في الباروميتر

اكتر الدوال المستخدمة في المشاريع علي حسبundefined موقع خدمات الويبundefined

merge()

The merge method merges the given array or collection with the original collection. If a string key in the given items matches a string key in the original collection, the given items's value will overwrite the value in the original collection:

```
$collection = collect(['product_id' => 1, 'price' => 100]);

$merged = $collection->merge(['price' => 200, 'discount' => false]);

$merged->all();
// ['product_id' => 1, 'price' => 200, 'discount' => false]
```

If the given items's keys are numeric, the values will be appended to the end of the collection:

```
$collection = collect(['Desk', 'Chair']);

$merged = $collection->merge(['Bookcase', 'Door']);
```

```
$merged->all();

// ['Desk', 'Chair', 'Bookcase', 'Door']
```

دي بتدمج 2 كوليكشن مع بعض

بتدي الاولوية للمصفوفة الجديدة

union()

The union method adds the given array to the collection. If the given array contains keys that are already in the original collection, the original collection's values will be preferred:

```
$collection = collect([1 => ['a'], 2 => ['b']]);

$union = $collection->union([3 => ['c'], 1 => ['b']]);

$union->all();

// [1 => ['a'], 2 => ['b'], 3 => ['c']]
```

دي بتدمج 2 كوليكشن مع بعض

بتدي الاولوية للمصفوفة القديمة

unique()

The unique method returns all of the unique items in the collection. The returned collection keeps the original array keys, so in this example we'll use the values method to reset the keys to consecutively numbered indexes:

```
$collection = collect([1, 1, 2, 2, 3, 4, 2]);
```

```php
$unique = $collection->unique();

$unique->values()->all();

// [1, 2, 3, 4]
```

When dealing with nested arrays or objects, you may specify the key used to determine uniqueness:

```php
$collection = collect([
    ['name' => 'iPhone 6', 'brand' => 'Apple', 'type' => 'phone'],
    ['name' => 'iPhone 5', 'brand' => 'Apple', 'type' => 'phone'],
    ['name' => 'Apple Watch', 'brand' => 'Apple', 'type' => 'watch'],
    ['name' => 'Galaxy S6', 'brand' => 'Samsung', 'type' => 'phone'],
    ['name' => 'Galaxy Gear', 'brand' => 'Samsung', 'type' => 'watch'],
]);

$unique = $collection->unique('brand');

$unique->values()->all();

/*
    [
        ['name' => 'iPhone 6', 'brand' => 'Apple', 'type' => 'phone'],
        ['name' => 'Galaxy S6', 'brand' => 'Samsung', 'type' => 'phone'],
    ]
*/
```

بترجع قيم المصفوفة بدون تغير

toJson()

The toJson method converts the collection into a JSON serialized string:

```php
$collection = collect(['name' => 'Desk', 'price' => 200]);

$collection->toJson();

// '{"name":"Desk", "price":200}'
```

تحويل المصفوفة الي جيسون

toArray()

The toArray method converts the collection into a plain PHP array. If the collection's values are Eloquent models, the models will also be converted to arrays:

```php
$collection = collect(['name' => 'Desk', 'price' => 200]);

$collection->toArray();

/*
    [
        ['name' => 'Desk', 'price' => 200],
    ]
*/
```

تحويل الكوليكشن الي مصفوفة

---

when()

The when method will execute the given callback when the first argument given to the method evaluates to true:

```
$collection = collect([1, 2, 3]);

$collection->when(true, function ($collection) {
    return $collection->push(4);
});

$collection->when(false, function ($collection) {
    return $collection->push(5);
});

$collection->all();

// [1, 2, 3, 4]
```

اضافة او تنفيذ قيمة معينة عند تحقق الشرط

عكس الدالة اضافة او تنفيذ قيمة معينة عند حالة عدم تحقق الشرط

unless()

The unless method will execute the given callback unless the first argument given to the method evaluates to true:

```
$collection = collect([1, 2, 3]);

$collection->unless(true, function ($collection) {
    return $collection->push(4);
});

$collection->unless(false, function ($collection) {
    return $collection->push(5);
});

$collection->all();

// [1, 2, 3, 5]
```

---

take()

الاخذ

The take method returns a new collection with the specified number of items:

```
$collection = collect([0, 1, 2, 3, 4, 5]);

$chunk = $collection->take(3);

$chunk->all();
```

```
// [0, 1, 2]
```

```
$collection = collect([0, 1, 2, 3, 4, 5]);

$chunk = $collection->take(-2);

$chunk->all();

// [4, 5]
```

duplicates()

the duplicates method retrieves and returns duplicate values from the collection:

```
$collection = collect(['a', 'b', 'a', 'c', 'b']);

$collection->duplicates();

// [ 2 => 'a', 4 => 'b' ]
```

بترجع مصفوفة فيها القيم الي اتكررت

sum()

الجمع

The sum method returns the sum of all items in the collection:

```
collect([1, 2, 3, 4, 5])->sum();
```

```
// 15
```

If the collection contains nested arrays or objects, you should pass a key to use for determining which values to sum:

```
$collection = collect([
    ['name' => 'JavaScript: The Good Parts', 'pages' => 176],
    ['name' => 'JavaScript: The Definitive Guide', 'pages' => 1096],
]);

$collection->sum('pages');

// 1272
```

split()

The split method breaks a collection into the given number of groups:

```
$collection = collect([1, 2, 3, 4, 5]);

$groups = $collection->split(3);

$groups->toArray();

// [[1, 2], [3, 4], [5]]
```

pluck()

The pluck method retrieves all of the values for a given key:

```
$collection = collect([
    ['product_id' => 'prod-100', 'name' => 'Desk'],
    ['product_id' => 'prod-200', 'name' => 'Chair'],
]);

$plucked = $collection->pluck('name');

$plucked->all();

// ['Desk', 'Chair']
```

بتاخد الاندكس الخاص بالمصفوفة وبترجع القيم بتاعتو

pipe()

The pipe method passes the collection to the given callback and returns the result:

```
$collection = collect([1, 2, 3]);

$piped = $collection->pipe(function ($collection) {
    return $collection->sum();
});

// 6
```

undefinedبترجع قيمة المصفوفة كاملة في مع دمج النتيجةundefined

1. **undefinedwhere()undefined**
2. **undefinedwhereBetween()undefined**
3. **undefinedwhereIn()undefined**

4. **undefinedwhereNotIn()undefined**

where()

The where method filters the collection by a given key / value pair:

```
$collection = collect([
    ['product' => 'Desk', 'price' => 200],
    ['product' => 'Chair', 'price' => 100],
    ['product' => 'Bookcase', 'price' => 150],
    ['product' => 'Door', 'price' => 100],
]);

$filtered = $collection->where('price', 100);

$filtered->all();

/*
    [
        ['product' => 'Chair', 'price' => 100],
        ['product' => 'Door', 'price' => 100],
    ]
*/
```

whereBetween()

The whereBetween method filters the collection within a given range:

```
$collection = collect([
    ['product' => 'Desk', 'price' => 200],
    ['product' => 'Chair', 'price' => 80],
```

```
    ['product' => 'Bookcase', 'price' => 150],
    ['product' => 'Pencil', 'price' => 30],
    ['product' => 'Door', 'price' => 100],
]);

$filtered = $collection->whereBetween('price', [100, 200]);

$filtered->all();

/*
    [
        ['product' => 'Desk', 'price' => 200],
        ['product' => 'Bookcase', 'price' => 150],
        ['product' => 'Door', 'price' => 100],
    ]
*/
```

whereIn()

```
$collection = collect([
    ['product' => 'Desk', 'price' => 200],
    ['product' => 'Chair', 'price' => 100],
    ['product' => 'Bookcase', 'price' => 150],
    ['product' => 'Door', 'price' => 100],
]);

$filtered = $collection->whereIn('price', [150, 200]);

$filtered->all();

/*
```

```
    [
        ['product' => 'Bookcase', 'price' => 150],
        ['product' => 'Desk', 'price' => 200],
    ]
*/
```

whereBetween()

```
$collection = collect([
    ['product' => 'Desk', 'price' => 200],
    ['product' => 'Chair', 'price' => 80],
    ['product' => 'Bookcase', 'price' => 150],
    ['product' => 'Pencil', 'price' => 30],
    ['product' => 'Door', 'price' => 100],
]);

$filtered = $collection->whereBetween('price', [100, 200]);

$filtered->all();

/*
    [
        ['product' => 'Desk', 'price' => 200],
        ['product' => 'Bookcase', 'price' => 150],
        ['product' => 'Door', 'price' => 100],
    ]
*/
```

whereNotIn()

```php
$collection = collect([
    ['product' => 'Desk', 'price' => 200],
    ['product' => 'Chair', 'price' => 100],
    ['product' => 'Bookcase', 'price' => 150],
    ['product' => 'Door', 'price' => 100],
]);

$filtered = $collection->whereNotIn('price', [150, 200]);

$filtered->all();

/*
    [
        ['product' => 'Chair', 'price' => 100],
        ['product' => 'Door', 'price' => 100],
    ]
*/
```

> الفرق بين
> نفس الوظيفة مع اختلاف بيان المخرجات
> الاولي قيم
> الثانية دوال

get()

The get method returns the item at a given key. If the key does not exist, null is returned:

```php
$collection = collect(['name' => 'taylor', 'framework' => 'laravel']);

$value = $collection->get('name');

// taylor
```

only()

The only method returns the items in the collection with the specified keys:

```php
$collection = collect(['product_id' => 1, 'name' => 'Desk', 'price' => 100,
'discount' => false]);

$filtered = $collection->only(['product_id', 'name']);

$filtered->all();

// ['product_id' => 1, 'name' => 'Desk']
```

---

groupBy()

بتفصل جروبات علي حسب الاندكس الي بيدخل ليها

The groupBy method groups the collection's items by a given key:

```php
$collection = collect([
    ['account_id' => 'account-x10', 'product' => 'Chair'],
```

```
    ['account_id' => 'account-x10', 'product' => 'Bookcase'],
    ['account_id' => 'account-x11', 'product' => 'Desk'],
]);

$grouped = $collection->groupBy('account_id');

$grouped->toArray();

/*
    [
        'account-x10' => [
            ['account_id' => 'account-x10', 'product' => 'Chair'],
            ['account_id' => 'account-x10', 'product' => 'Bookcase'],
        ],
        'account-x11' => [
            ['account_id' => 'account-x11', 'product' => 'Desk'],
        ],
    ]
*/
```

has()

The has method determines if a given key exists in the collection:

```
$collection = collect(['account_id' => 1, 'product' => 'Desk', 'amount' => 5]);

$collection->has('product');

// true

$collection->has(['product', 'amount']);
```

```
// true

$collection->has(['amount', 'price']);

// false
```

بيتم تمرير المفتاح الرئيسي اذا كان موجود بيرجع صح او غلط اذا كان غير موجود

diffAssoc()

The diffAssoc method compares the collection against another collection or a plain PHP array based on its keys and values. This method will return the key / value pairs in the original collection that are not present in the given collection:

```
$collection = collect([
    'color' => 'orange',
    'type' => 'fruit',
    'remain' => 6
]);

$diff = $collection->diffAssoc([
    'color' => 'yellow',
    'type' => 'fruit',
    'remain' => 3,
    'used' => 6
]);
```

```
$diff->all();

// ['color' => 'orange', 'remain' => 6]
```

بتطلع الاختلاف مابين المصفوفتين

duplicates()

```
the duplicates method retrieves and returns duplicate values from the collection:

$collection = collect(['a', 'b', 'a', 'c', 'b']);

$collection->duplicates();

// [ 2 => 'a', 4 => 'b' ]
```

بتطلع القيم الي اتكررت وبتخليها في مصفوفة

merge()

```
The merge method merges the given array or collection with the original collection.
If a string key in the given items matches a string key in the original collection,
the given items's value will overwrite the value in the original collection:
```

```
$collection = collect(['product_id' => 1, 'price' => 100]);

$merged = $collection->merge(['price' => 200, 'discount' => false]);

$merged->all();

// ['product_id' => 1, 'price' => 200, 'discount' => false]
If the given items's keys are numeric, the values will be appended to the end of the collection:

$collection = collect(['Desk', 'Chair']);

$merged = $collection->merge(['Bookcase', 'Door']);

$merged->all();

// ['Desk', 'Chair', 'Bookcase', 'Door']
```

دمج 2 كولكشن مع بعض

pull()

```
The pull method removes and returns an item from the collection by its key:
```

```
$collection = collect(['product_id' => 'prod-100', 'name' => 'Desk']);

$collection->pull('name');
```

```
// 'Desk'

$collection->all();

// ['product_id' => 'prod-100']
```

تقوم بالحذف

push()

```
The push method appends an item to the end of the collection:
```

```
$collection = collect([1, 2, 3, 4]);

$collection->push(5);

$collection->all();

// [1, 2, 3, 4, 5]
```

تقوم باضافة عنصر جديد في اخر الكولكشن

put()

The put method sets the given key and value in the collection:

```
$collection = collect(['product_id' => 1, 'name' => 'Desk']);

$collection->put('price', 100);

$collection->all();

// ['product_id' => 1, 'name' => 'Desk', 'price' => 100]
```

تقوم باضافة عنصر في الكولكشن مع مفتاح وقمية

random()

The random method returns a random item from the collection:

```
$collection = collect([1, 2, 3, 4, 5]);

$collection->random();

// 4 - (retrieved randomly)
```

تقوم بارجاع قيم عشوائة

unless()

The unless method will execute the given callback unless the first argument given to the method evaluates to true:

```
$collection = collect([1, 2, 3]);

$collection->unless(true, function ($collection) {
    return $collection->push(4);
});

$collection->unless(false, function ($collection) {
    return $collection->push(5);
});

$collection->all();

// [1, 2, 3, 5]
```

تقوم بالدوران في حالة عدم تحقق الشرط تنفذ المطلوب