

OBJECT IN JAVASCRIPT

```
748 $sort_order = array();
749
750 foreach ($quotes as $key => $value) {
751     $sort_order[$key] = $value['sort_order'];
752 }
753
754 array_multisort($sort_order, SORT_ASC, $quotes);
755
756 $this->session->data['lpa']['shipping_methods'] = $quotes;
757 $this->session->data['lpa']['address'] = $address;
758
759 if (empty($quotes)) {
760     $json['error'] = $this->language->get('
761         error_no_shipping_methods');
762 } else {
763     $json['quotes'] = $quotes;
764 }
765
766 if (isset($this->session->data['lpa']['shipping_method']) && !
767     empty($this->session->data['lpa']['shipping_method']) &&
768     isset($this->session->data['lpa']['shipping_method']['code']
769 )) {
770     $json['selected'] = $this->session->data['lpa']['
771         shipping_method']['code'];
772 } else {
773     $json['selected'] = '';
774 }
775
776 } else {
777     $json['error'] = $this->language->get('error_shipping_methods');
```

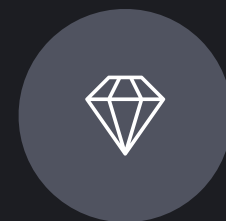
```
386     this.cycle(true)
387 }
388
389 this.interval = clearInterval(this.interval)
390
391 return this
392 }
393
394 Carousel.prototype.next = function () {
395     if (this.sliding) return
396     return this.slide('next')
397 }
398
399 Carousel.prototype.prev = function () {
400     if (this.sliding) return
401     return this.slide('prev')
402 }
403
404 Carousel.prototype.slide = function (type, next) {
405     var $active = this.$element.find('.item.active')
406     var $next = next || this.getActiveDirection(type, $active)
407     var direction = type == 'next' ? 'left' : 'right'
408     var fallback = type == 'next' ? 'first' : 'last'
409     var that = this
410
411     if (!$next.length) {
412         if (!this.options.wrap) return
413         $next = this.$element.find('.item')[fallback]()
414     }
415
416     if ($next.hasClass('active')) return (this.sliding = false)
417
418     var relatedTarget = $next[0]
419     var slideEvent = $.Event('slide.bs.carousel', {
420         relatedTarget: relatedTarget,
421         direction: direction
```


α_t

TYPES

α_t

TYPES



PRIMITIVE VALUE

Undefined, Null, Boolean,
Number, String and Symbol



REFERENCE VALUE

object, function, array

TYPES



PRIMITIVE VALUE

`const a = 2;`
`const b = a;` `=` `const a = 2;`
`const b = 2;`



REFERENCE VALUE

`const a = {};`
`const b = a;` `≠` `const a = {};`
`const b = {};`

α_t

this KEYWORD

Most of the time, **this** refers to the object that executes the code

this KEYWORD

■ Global

```
this;
```

The global object

■ Function call (ES5)

```
myFunc();
```

The global object or
undefined in strict mode

■ Method call

```
obj.myFunc();
```

The object that executes
the function

■ Constructor

```
new Obj();
```

The newly created object

■ Explicitly set (ES5)

```
myFunc.bind(obj);
```

explicitly set to an object,
Can not set for arrow function

■ Arrow function

```
() => {}
```

The object outside the
function

Note: should not use as
method

α_t

Prototype

Prototype is how inheritance
in javascript works

α_t

__proto__ property

α_t

__proto__ property

An object doesn't have prototype property but it has the __proto__ property. The inheritance object of an object is the __proto__ property.

α_t

prototype property

```
Box.prototype.show = function () {  
  // ...  
}
```

α_t

prototype property

```
Box.prototype.show = function () {  
    // ...  
}
```

The **prototype** property is created when function is created and only exists in functions.
It's only used for the **new** keyword

α_t

new keyword

```
const data = new Employee();
```


α_t

new keyword

```
const data = new Employee();
```

It does 4 things

α_t

new keyword

```
const data = new Employee();
```

- 1 Create a new plain object

It does 4 things

α_t

new keyword

```
const data = new Employee();
```

- 1 Create a new plain object
- 2 Assign that object's `__proto__` property to the `prototype` property of the function

It does 4 things

α_t

new keyword

```
const data = new Employee();
```

- 1 Create a new plain object
- 2 Assign that object's `__proto__` property to the `prototype` property of the function
- 3 Executes the function with `this` binds to the newly created object

It does 4 things

new keyword

```
const data = new Employee();
```

It does 4 things

- 1 Create a new plain object
- 2 Assign that object's `__proto__` property to the `prototype` property of the function
- 3 Executes the function with `this` binds to the newly created object
- 4 Return the object

α_t

Class

Class is just an implementation that
builds on top of the **prototype**

OVERVIEW

- Javascript is prototype-based
- Understand different of primitive value and reference value
- Understand 6 different scope of `this` keyword
- Understand `prototype`, `__proto__`
- Understand `class` keyword

α_t

THANK YOU
FOR LISTENING