

# Pixel documentation

## javascript load

Loads the boostable tracking software

browser requirements: \* ie8+ \* chrome 2.0+ \* safari 3.0+ \* firefox 4.0+ \* opera 6.0+

This snippet should be places in the header of any page wanting to be tracked (replace with id provided to you):

```
<!-- boostable initialize -->
<noscript></noscript>
<script>(function(b,s,t,p,x,l){b.bst||(b.bst=x=function(){x.event?x.event.apply(x,a
rguments):
x.q.push(arguments)},x.q=[],x.v="2.0",x.s=new
Date());l=s.createElement(t);l.src=p;l.async=!0;
l.defer=!0;s.getElementsByTagName(t)[0].parentNode.appendChild(l)})(window,document,
"script",
"//cdn.boostable.com/tr.js");</script>

<script>bst('init', '<BOOST_ID>');</script>
<!-- end boostable initialize -->
```

## objects

In order for the boostable tracking software to work as expected, calling the events with proper object parameters is a must. These objects are a guide as to what is expected in the events.

### EcommerceObject

boostable's representation of an ecommerce item. There is no constructor for this, but objects with these attributes should be used as parameters to the events.

attributes: \* **sku** {String} (*required*) unique product id \* **cost** {Number} (*required*) ISO-4217  
currency value \* **category** {String|Array[String]} (*optional*) the object category \* **quantity** {Number}  
(*optional*) default: 1 \* **currency** {String} (*optional*) ISO-4217 currency code default: 'USD' \* **name**  
{String} (*optional*) \* **description** {String} (*optional*)

usage:

```
var o = {
  sku: 'foo-123',
  name: 'foo bar',
  description: 'a little foo, a little bar',
  category: ['foo', 'bar'],
  cost: 1234,
```

```

    currency: 'USD',
    quantity: 2
  };

```

## EcommerceOrder

boostable's representation of an ecommerce order. There is no constructor for this, but objects with these attributes should be used.

attributes: \* **id** {String} (*required*) unique order id \* **total** {Number} (*required*) total revenue as [ISO-4217](#) currency value \* **cart** {EcommerceObject | Array[EcommerceObject]} (*required*) the items currently in the cart \* **currency** {Number} (*optional*) [ISO-4217](#) currency code default: 'USD' \* **subTotal** {Number} (*optional*) summation of costs prior to any shipping, fees, tax, etc. as [ISO-4217](#) currency value \* **shipping** {Number} (*optional*) any shipping cost applied as [ISO-4217](#) currency value \* **tax** {Number} (*optional*) any tax applied as [ISO-4217](#) currency value \* **discount** {Number} (*optional*) any discounts applied as [ISO-4217](#) currency value \* **fee** {Number} (*optional*) any additional fees applied as [ISO-4217](#) currency value

usage:

```

var o = {
  id: 'order-123',
  total: 4324,
  subTotal: 4000,
  tax: 324,
  shipping: 0,
  discount: 0,
  currency: 'USD',
  cart: { /*ecommerceObject*/ } // or [{ /*ecommerceObject 1*/ }, { /*ecommerceObject
2*/ } /*, ...*/ ]
};

```

## events

boostable events are triggered by calling the `bst` javascript function. These are the core of the boostable tracking software.

### init

initialize the boostable tracker. This should be placed in the header (and only once per page)

parameters: \* **id** {String} (*required*) the boostable id provided to you

usage:

```

bst('init', '<BOOST_ID>');

```

## cart

sends boostable the current cart status. This should be called after any cart update (add/remove).

parameters: \* cartItems {Array[EcommerceObject] | EcommerceObject} (*optional*) the current cart

usage:

```
// empty cart:
bst('cart');
// or
bst('cart', []);

// single item cart:
bst('cart', { /*ecommerceObject 1*/ });
// or
bst('cart', [{ /*ecommerceObject 1*/ }]);

// multi item cart:
bst('cart', [{ /*ecommerceObject 1*/ }, { /*ecommerceObject 2*/ } /*, ...*/ ]);
// or
bst('cart', { /*ecommerceObject 1*/ }, { /*ecommerceObject 2*/ } /*, ...*/ );
```

## purchase

track a purchase event. This should be called when an order is placed (and only after)

parameters: \* order {EcommerceOrder} (*required*) the order in question

usage:

```
bst('purchase', { /*ecommerceOrder*/ });
```

## view

trigger a page view. This should be called on any page, or any new modal (with any updated information).

parameters: \* item {EcommerceObject} (*optional*) the item or category being viewed

usage:

```
// simple page view
bst('view')

// ecommerce page view
bst('view', { /*ecommerceObject*/ })

// ecommerce category view
bst('view', { /*ecommerceObject with null sku and defined category*/ })
```

## examples

here are a view examples of calling the events.

ecommerce category view:

```
bst('view', {sku: null, category: ['foo', 'bar']});
```

single item cart:

```
bst('cart', {
  sku: 'foo-123',
  cost: 1234,
  name: 'foo bar'
});
```

multi item cart:

```
var c = [
  {
    sku: 'foo-123',
    cost: 123,
    name: 'foo 123'
  },
  {
    sku: 'foo-456',
    cost: 456,
    name: 'foo 456'
  }
];
bst('cart', c);
```

single item purchase:

```
var o = {
  id: 'order-123',
  total: 4324,
  subTotal: 4000,
  tax: 324,
  shipping: 0,
  discount: 0,
  currency: 'USD',
  cart: {
    sku: 'foo-4000',
    cost: 4000,
    name: 'foo 4000'
  }
};
```

```
bst('purchase', o);
```

## multi-item purchase

```
var c = [
  {
    sku: 'foo-123',
    cost: 123,
    name: 'foo 123'
  },
  {
    sku: 'foo-456',
    cost: 456,
    name: 'foo 456'
  }
];
var o = {
  id: 'order-703',
  total: 703
  subTotal: 579
  tax: 124
  fee: 0,
  currency: 'USD',
  cart: c
};
bst('purchase', o);
```

## see also:

this [fiddle](#) for a simple ecommerce simulation. Use as reference only.