

Simple Use Case: Making an Element Draggable

```
myDDObj = new YAHOO.util.DD("myDiv");
```

Makes the HTML element whose id attribute is "myDiv" draggable.

Constructor (YAHOO.util.DD, DDProxy, DDTarget)

```
YAHOO.util.DD(str | el ref target[, str group name, obj configuration]);
```

Arguments:

- (1) **Element:** ID or elem. ref. of the element to make draggable; deferral is supported if the element is not yet on the page.
- (2) **Group Name:** An optional string indicating the DD group; DD objects only "interact with" other objects that share a group.
- (3) **Configuration:** An object containing name-value pairs, used to set any of the DD object's properties.

Properties & Methods of YAHOO.util.DragDrop

<p>Properties:</p> <ul style="list-style-type: none"> available (b) dragOnly (b) useShim (b) groups (ar) id (s) invalidHandle Classes (s[]) invalidHandleIds (obj) isTarget (b) maintainOffset (b) padding (int[]) primaryButtonOnly (b) xTicks (int[]) yTicks (int[]) 	<p>Methods:</p> <ul style="list-style-type: none"> addInvalidHandle Class (s <i>cssClass</i>) addInvalidHandleId (s <i>id</i>) addInvalidHandle Type (s <i>tagName</i>) addToGroup (s <i>groupName</i>) clearTicks() clearConstraints() getDragEl() getEl() isLocked() lock() removeFromGroup(o <i>dd</i>, s <i>group</i>) removeInvalid HandleClass(s <i>cssClass</i>) removeInvalid HandleId(s <i>id</i>) removeInvalidHandle Type (s <i>tagName</i>) resetConstraints() setDragElId(s <i>id</i>) setHandleElId (s <i>id</i>) setOuterHandleElId (s <i>id</i>) setPadding(i <i>top</i>, i <i>right</i>, i <i>bottom</i>, i <i>left</i>) setXConstraint(i <i>left</i>, i <i>right</i>, i <i>tick size</i>) setYConstraint(i <i>up</i>, i <i>down</i>, i <i>tick size</i>) unlock() unreg()
---	--

Properties & Methods of YAHOO.util.DD & .DDProxy

Inherit from YAHOO.util.DragDrop and add the following:

YAHOO.util.DD Properties:	YAHOO.util.DDProxy Properties:
scroll (b)	centerFrame (b)
	resizeFrame (b)

Interesting Moments in Drag & Drop

Moment	Point Mode	Intersect Mode	Event (e)
onMouseDown	e	e	mousedown
startDrag	x, y	x, y	n/a
onDrag	e	e	mousemove
onDragEnter	e, id	e, DDArray	mousemove
onDragOver	e, id	e, DDArray	mousemove
onDragOut	e, id	e, DDArray	mousemove
onDragDrop	e, id	e, DDArray	mouseup
onInvalidDrop	e	e	mouseup
endDrag	e	e	mouseup
onMouseUp	e	e	mouseup

These "moments" are exposed as events on your DD instances; they are methods of YAHOO.util.DragDrop. The table above identifies the arguments passed to these methods in Point and Intersect modes.

Solutions

Add a drag handle to an existing DD object:

```
myDDObj.setHandleElId('myDragHandle');
```

Set the "padding" or "forgiveness zone" of a DD object:

```
myDDObj.setPadding(20, 30, 20, 30); //units are pixels, top/rt/bt/left
```

Get the 'best match' from an onDragDrop event in Intersect Mode where the dragged element is over more than one target:

```
myDDObj.onDragDrop = function(e, DDArray) {
    oDDBestMatch =
        YAHOO.util.DragDropMgr.getBestMatch(DDArray);
}
```

Override an interesting moment method for a DD object instance:

```
myDDObj = new YAHOO.util.DD("myDiv");
myDDObj.startDrag = function(x,y) {
    this.iStartX = x; this.iStartY = y;
}
```

Change the look and feel of the proxy element at the start of a drag event using YAHOO.util.DDProxy:

```
myDDObj.startDrag(x,y) {
    YAHOO.util.Dom.addClass(this.getDragEl(),
        "myCSSClass"); }
```

Lock Drag and Drop across the whole page:

```
YAHOO.util.DragDropMgr.lock();
```

Switch to Intersect Mode:

```
YAHOO.util.DragDropMgr.mode =
    YAHOO.util.DragDropMgr.INTERSECT;
```

Drag & Drop Manager: Properties

- clickPixelThresh (i)
- clickTimeThresh (i)
- useShim (b)
- mode either YAHOO.util.DragDropMgr.PC, NT or .INTERSECT
- preventDefault (b)
- stopPropagation (b)
- useCache (b)

Drag & Drop Manager: Methods

oDD=instance of DragDrop object

- getBestMatch(a [oDDs])
- getDDById(s id)
- getLocation(oDD)
- getRelated(oDD, b target only)
- isDragDrop(s id)
- isHandle(s DDId, s HandleId)
- isLegalTarget(oDD, oDD target)
- isLocked()
- lock()
- refreshCache()
- swapNode()
- unlock()

***Note:** YAHOO.util.DragDropMgr is a singleton; changes made to its properties (such as locking or unlocking) affect Drag and Drop globally throughout a page.

Dependencies

Drag & Drop requires the YAHOO object, DOM, and Event.