

# Y! YUI Library: Event Utility & Custom Event

v0.10

## Simple Use Case: Adding Event Listeners

```
YAHOO.util.Event.addListener("myDiv", "click",  
    fnCallback);
```

Adds the function `fnCallback` as a listener for the click event on an HTML element whose id attribute is "myDiv".

## Invocation (addListener)

```
YAHOO.util.Event.addListener(str | el ref | arr  
    target[s], str event, fn callback[, obj  
    associated object, b scope]);
```

Arguments:

- (1) **Element or elements:** You may pass a single element or group of elements in an array; references may be id strings or direct element references.
- (2) **Event:** A string indicating the event ('click', 'keypress', etc.).
- (3) **Callback:** The function to be called when the event fires.
- (4) **Associated object:** Object to which your callback will have access; often the callback's parent object.
- (5) **Scope:** Boolean — if true, the callback runs in the scope of the associated object.

## Event Utility Solutions

Using **onAvailable**:

```
fnCallback = function() { //will fire when element  
    becomes available}  
YAHOO.util.Event.onAvailable('myDiv', fnCallback);
```

Using Event's **convenience methods**:

```
fnCallback = function(e, obj) {  
    myTarget = YAHOO.util.Event.getTarget(e, 1);  
    //2nd argument tells Event to resolve text nodes  
}  
YAHOO.util.Event.addListener('myDiv', 'mouseover',  
    fnCallback, obj);
```

**Prevent the event's default behavior** from proceeding:

```
YAHOO.util.Event.preventDefault(e);
```

**Remove listener:**

```
YAHOO.util.Event.removeListener('myDiv',  
    'mouseover', fnCallback);
```

## Dependencies

Event requires the YAHOO object.

## Simple Use Case: Custom Event

```
myEvt = new YAHOO.util.CustomEvent("my event");  
mySubscriber = function(type, args) {  
    alert(args[0]); //alerts the first argument}  
myEvt.subscribe(mySubscriber);  
myEvt.fire("hello world");
```

Creates a new Custom Event instance and a subscriber function. The subscriber is set to listen to the Custom Event and alert the first argument, "hello world", when the event is fired.

## Constructor (Custom Event)

```
YAHOO.util.CustomEvent(str event name[, obj scope  
    object]);
```

Arguments:

- (1) **Event name:** A string identifying the event.
- (2) **Scope object:** The default scope in which subscribers will run; can be overridden in subscribe method.

## Subscribing to a Custom Event

```
myEvt.subscribe(fn callback[, obj associated object,  
    b scope]);
```

Arguments:

- (1) **Callback:** The function to be called when the event fires.
- (2) **Associated object:** Object to which your callback will have access; often the callback's parent object.
- (3) **Scope:** Boolean — if true, the callback runs in the scope of the associated object.

Your callback gets three arguments:

- (1) **Type:** The type of Custom Event, a string.
- (2) **Arguments:** All arguments passed in during `fire`, as an array.
- (3) **Associated object:** The associated object passed in during `subscribe`, if present.

## DOM Events

### Mouse Events:

click  
dblclick  
mousedown  
mouseout  
mouseover  
mouseup  
mousemove

### Keyboard Events:

keydown  
keypress  
keyup  
  
HTML Events:  
load  
unload  
abort

error  
select  
change  
submit  
reset  
resize  
scroll  
blur

## Event Utility Methods:

**addListener(...)**  
**getCharCode(e)**  
**getPageX(e)**  
**getPageY(e)**  
**getRelatedTarget(e)**  
**getTarget(e)**  
**getTime(e)**  
**getXY(e):** returns array  
    [pageX, pageY]  
**onAvailable(...)**  
**preventDefault(e)**  
**removeListener(...)**  
**stopEvent(e):** same as  
    preventDefault plus  
    stopPropagation  
**stopPropagation(e)**

## DOM Event Object Properties & Methods:

**altKey** (b)  
**bubbles** (b)  
**button** (i)  
**cancelable** (b)  
**cancelBubble** (b)  
**\*charcode** (i)  
**clientX** (i)  
**clientY** (i)  
**ctrlKey** (b)  
**currentTarget** (el)  
**detail** (i)  
**eventPhase** (i)  
**isChar** (b)  
**keyCode** (i)  
**metaKey** (i)  
**\*pageX** (i)  
**\*pageY** (i)  
**\*preventDefault()**  
**\*relatedTarget** (el)  
**screenX** (i)  
**screenY** (i)  
**shiftKey** (b)  
**\*stopPropagation()**  
**\*target** (el)  
**\*timestamp** (long)  
**type** (s)

[ \*use Event Utility method ]