

Intention Economy

Taxonomy of Browser-Based Gestures

National Association of REALTORS®
May 2007

Executive Summary

Consumer gestures in the physical world are obvious and are important elements to customer service. Gestures signal consumer acceptance or rejection of value propositions. Gestures in Internet-based business are less obvious because the mechanisms for detecting them are different. In the physical world, the human senses are used whereas in the the Internet world, digital feedback must be used.

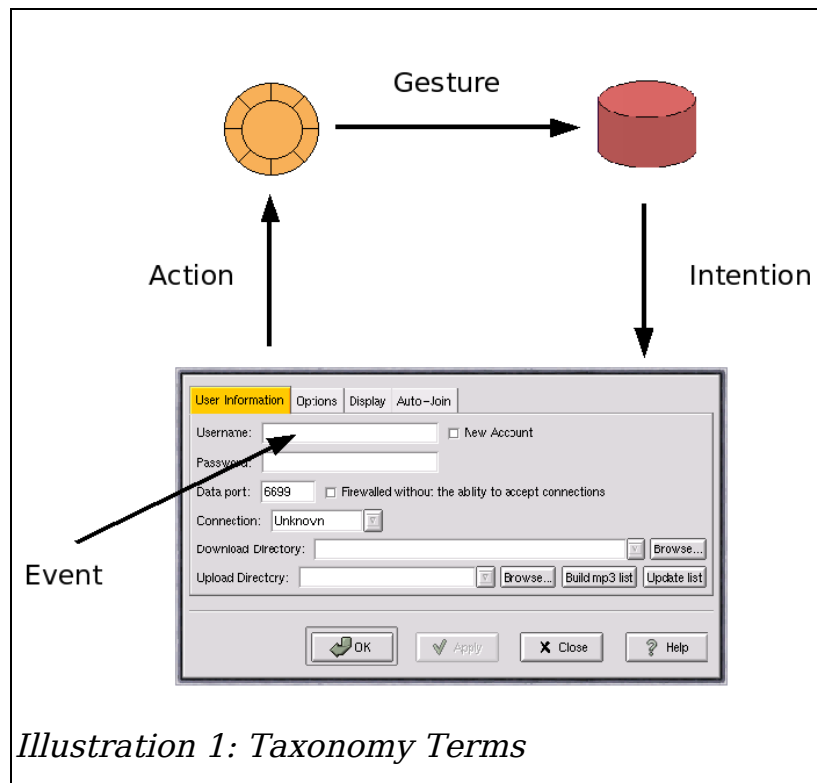


Illustration 1: Taxonomy Terms

Consumer gestures can be collected from browsers that render documents written in standard HTML. Gestures represent all mouse and keyboard usage including movement and clicking. Internet-based businesses have always relied on mouse clicks, but Intention can be determined by mouse movements as well.

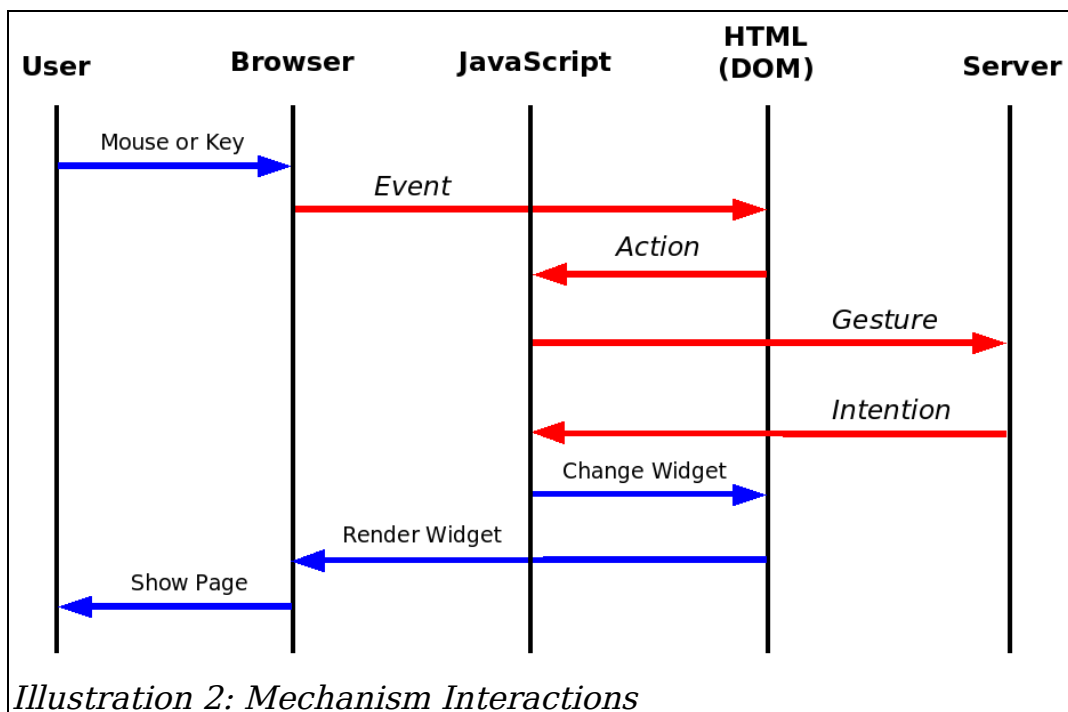
Once captured, gestures can be analyzed to determine the intention of the user bringing the marketing practices in the Internet one step closer to the physical world.

Definition of Terms

Terminology that will be used in this paper are presented as a stylized graphic in *Illustration 1: Taxonomy Terms*. Specifically these terms are:

- **Event** – A user initiated happening, such as moving a mouse, clicking on a button or changing the value of a field.
- **Action** – Another form of an event. Similar Events are grouped to provide better semantics; an abstraction of HTML events.
- **Gesture** – An action supplemented with unique user information. Gestures are sent from the consumer browser to the operator's server.
- **Intention** – Repeated gestures that signal the desire of a consumer. Intentions provide feedback to the consumer's browser.

The *Taxonomy Terms* illustration is a stylized, high level presentation of the concept that over simplifies the actual mechanism. A more accurate depiction is presented as an interaction diagram as presented in *Illustration 2: Mechanism Interactions*. This presentation shows the four terms that are being defined within the sequence of processing.



An *Event* is created when the user operates any input device, typically a keyboard or mouse. This *Event* is captured by the browser and uses the Domain Object Model (DOM), created when the HTML was loaded, to determine if the *Event* is significant.

The definition in the DOM is used to create a call to designer supplied JavaScript. The call abstracts the *Event* into an *Action*. Actions standardize the various Events for processing from the intention perspective. Actions are defined later in this document.

The next step in the sequence is to send the *Action* to the server in the form of a *Gesture*. The server analyzes Gestures and determines what the consumer is trying to do on the website. The determination is called an *Intention* and is returned from the server to the JavaScript running in the browser.

The form of an *Intention* is implementation dependent but represents feedback to the consumer. In an AJAX implementation, it can be as simple as HTML (within a CDATA structure of XML) targeted for an HTML <id> tag.

Events Generated from HTML

HTML tags can be used to capture user events within a browser session. These events are created by the browser as a result of HTML tags that call JavaScript functions when mouse or keyboard usage is detected.

- **<form>** - Generates the events *onSubmit* before sending the form to the server and *onReset* when resetting the fields to their original values.
- **<input>**- Represents a variety of GUI widgets. The use *type* attribute is used to specify the specific widget. What is common to all INPUT tags is the support for *onMouseOver* and *onMouseOut*. INPUT tags that have the TEXT, RADIO or CHECKBOX attributes also support the *onBlur*, *onChange* and *onSelect* events. Tags that have the BUTTON, SUBMIT or RESET attributes do not allow information entry. The *onBlur* and *onFocus* events require mouse clicks.
- **<select>** - Represents an entry area that allows selection from multiple options displayed in a drop-down box from within a <form> tag. This tag shares many of the characteristics of the <input> tag but has separate <option> tags for each alternative displayed to the user.

- **<textarea>** - Represents a multi-line text entry area within a <form> tag. This tag shares many of the characteristics of the <input> tag.
- **** - Displays an image on the page and supports the *onMouseOver*, *onMouseOut* and *onClick* events.
- **<media>** - Displays either audio or video on the page and supports the *onMouseOver*, *onMouseOut* and *onClick* events.
- **<body>** - Supports the *onLoad* event when the document is first loaded and the *onUnload* event when the browser leaves the page.
- **<a>** - Used to represent links to sections within the current document or other HTML documents. This tag has more robust capabilities than other HTML tags because events do not always require mouse clicks to be detected as found in *onMouseOver* and *onMouseOut*. The *onClick* event is called when the mouse is clicked.

Similarities in events can be found in the HTML tags even though the naming conventions are not always consistent. An example of this would be acquiring focus. Focus for a link is detected with *onMouseOver* whereas focus on a page is reported with the *onLoad* event.

In some cases, multiple events from the same HTML tag can be used to express the same Action. An example would be the *OnMouseOver* and *OnFocus* events of the <input> tag; both can be mapped to the Focus action. In these cases, only one of the events should be used to simplify processing. Because mouse movements happen before clicks, *OnMouseOver* is the preferred event.

The similarities, differences and ambiguities across events dictate the need to provide an abstraction layer before they can become useful.

Grouping Events into Actions

HTML events can be grouped into actions of a standard model that facilitates intention processing with a process called abstraction or normalization. HTML events are easier to process for intention purposes if they are abstracted first because the semantics of browsing and intention are different.

Intention actions differ from the model used for databases because they do not correspond to the Create, Read, Update and Delete (CRUD) model. In the Intention model, users do not create data elements, they act on them. The purpose of the mechanism is to analyze the “acts” of the user.

The usefulness of abstraction is evident when discussing the <input> tag. Some of the widgets are clearly intended to support user entry while others (like BUTTON, SUBMIT and RESET) are used to control the form.

Abstraction can also bring together similar content like and <media>. Pictures, audio and video are all display-only content types that the user can view and even click on, but will not change. In some ways, and <media> are similar to the <input> tag types BUTTON, SUBMIT and RESET because they are not used to collect user input.

Browsers have built-in event handlers that are designed to capture events performed on a HTML widgets. For example, moving the mouse over a text field would create an *OnMouseOver* event. The end result of the capture would be positioning the cursor within the text field. Not all HTML tags are rendered as widgets therefore common formatting elements such as <td> or <div> are not used in intention analysis.

Event abstraction yields the following action types:

- **FOCUS** – Moving user attention to a widget
- **UNFOCUS** – Moving user attention away from a widget
- **SUBMIT** – Sending information to a server
- **CLICK** – Clicking on a link with the mouse
- **CHANGE** – Changing widget information

Further abstraction can be performed to simplify a variety of HTML widget (or control) types. Widget abstraction yields the following action widgets:

- **PAGE** – A viewable page in the browser
- **LINK** – A link to another page
- **FORM** – A collection of data entry fields
- **INPUT** – Data entry fields including text, radio and check boxes
- **MEDIA** – Images or streams like audio and video.
- **BUTTON** – A visible button control

In practice, grouping events into actions can be achieved by assigning JavaScript events that are fired by the browser. Assignments can be made by overriding the built-in event handlers with custom handlers. Overriding can be achieved by either altering the HTML directly or with JavaScript that works directly with the Domain Object Model (DOM) model of the browser. Further discussion of how to implement actions are beyond the scope of this document.

A recommended mapping of HTML events to actions is presented in *Illustration 3: Recommended Action to Event Mapping*. This table uses a subset of all available events accounting for the redundancy and ambiguity of HTML.

Action Type	Action Widget	HTML Tag	HTML Type	HTML Event
Focus	PAGE	BODY	N/A	On Load
	LINK	A	N/A	On MouseOver
	BUTTON	INPUT	BUTTON	On MouseOver
	BUTTON	INPUT	RESET	On MouseOver
	BUTTON	INPUT	SUBMIT	On MouseOver
	INPUT	INPUT	CHECKBOX	On MouseOver
	INPUT	INPUT	RADIO	On MouseOver
	INPUT	INPUT	PASSWORD	On MouseOver
	INPUT	INPUT	TEXT	On MouseOver
	INPUT	SELECT	N/A	On MouseOver
	INPUT	INPUT	TEXTAREA	On MouseOver
	MEDIA	IMG	N/A	On MouseOver
MEDIA	MEDIA	N/A	On MouseOver	
UnFocus	LINK	A	N/A	On MouseOut
	BUTTON	INPUT	BUTTON	On MouseOut
	BUTTON	INPUT	RESET	On MouseOut
	BUTTON	INPUT	SUBMIT	On MouseOut
	INPUT	INPUT	CHECKBOX	On MouseOut
	INPUT	INPUT	RADIO	On MouseOut
	INPUT	INPUT	PASSWORD	On MouseOut
	INPUT	INPUT	TEXT	On MouseOut
	INPUT	SELECT	N/A	On MouseOut
	INPUT	INPUT	TEXTAREA	On MouseOut
	MEDIA	IMG	N/A	On MouseOut
	MEDIA	IMG	N/A	On MouseOut
PAGE	BODY	N/A	On Unload	
Submit	FORM	FORM	N/A	On Submit
Change	INPUT	INPUT	CHECKBOX	On Change
	INPUT	INPUT	RADIO	On Change
	INPUT	SELECT	N/A	On Change
	INPUT	INPUT	PASSWORD	On Change
	INPUT	INPUT	TEXT	On Change
	INPUT	INPUT	TEXTAREA	On Change
	FORM	FORM	N/A	On Change
	FORM	FORM	N/A	On Reset
Click	BUTTON	INPUT	BUTTON	On Click
	BUTTON	INPUT	RESET	On Click
	BUTTON	INPUT	SUBMIT	On Click
	MEDIA	IMG	N/A	On Click
	MEDIA	MEDIA	N/A	On Click
	LINK	A	N/A	On Click

Illustration 3: Recommended Action to Event Mapping

Actions Expressed as Gestures

A gesture is the server side representation of an action. Within the browser, HTML events are abstracted into actions before being sent to the server. The most common protocol to use for the server communication is HTTP. Additional information such as who, where and when are included as part of the HTTP message. A gesture is the combination of an action with some specific additional information.

Gestures are sent to a server by the browser for storage and analysis. One server can handle Intention analysis for multiple users therefore additional information needs to be added to the action. A gesture is composed of:

- Unique User
- Unique generation point
- Type of Action
- Name of the widget that generated the Action
- Address where the Action was detected
- Remote time the Action was detected (browser time)
- Local time the Action was processed (server time)

Gestures are stored on the server to facilitate analysis. Due to the real-time characteristics of intention processing, the entire transfer process can be handled with an Asynchronous Javascript and XML (AJAX) mechanism.

Generating Intention from Gestures

An intention is a computer derived, educated “guess” about what the user is interested in on a website. The “guess” is arrived at by either analyzing the last gesture captured or by analyzing the last gesture within the context of other gestures. There is an opportunity to perform this analysis every time an gesture is passed to the server.

An intention can take many forms and is passed back from the server to the browser for rendering. Examples of intention could be images, forms or changes to the menu selections. It can even be as simple as a piece of HTML that is intended to replace a HTML <div> tag on a page. If an AJAX process is used for communications, the resulting HTML can be sent back to the browser within a CDATA tag of XML.