1

Events

The keyword this

2

```
this.fieldName // access field
this.fieldName = value; // modify field
this.methodName(parameters); // call method
```
JS

- all JavaScript code actually runs inside of an object
- by default, code runs inside the global window object
  - all global variables and functions you declare become part of window
- the this keyword refers to the current object

Event handler binding

3

```
function pageLoad() {
	$("ok").onclick = okayClick; // bound to okButton
here
}
function okayClick() { // okayClick knows what DOM object
	this.innerHTML = "booyah"; // it was called on
}
window.onload = pageLoad;
```
JS

- event handlers attached unobtrusively are bound to the element
- inside the handler, that element becomes this (rather than the window)

Fixing redundant code with this

4

```
<fieldset>
	<label><input type="radio" name="ducks"
value="Huey" /> Huey</label>
	<label><input type="radio" name="ducks"
value="Dewey" /> Dewey</label>
	<label><input type="radio" name="ducks"
value="Louie" /> Louie</label>
</fieldset>
```
HTML

```
function processDucks() {
	if ($("huey").checked) {
		alert("Huey is checked!");
	} else if ($("dewey").checked) {
		alert("Dewey is checked!");
	} else {
		alert("Louie is checked!");
	}
	alert(this.value + " is checked!");
}
```
JS

More about events

5

abort	blur	change	click	dblclick	error	focus
keydown	keypress	keyup	load	mousedown	mousemove	mouseout
mouseover	mouseup	reset	resize	select	submit	unload

- the click event (onclick) is just one of many events that can be handled
- **problem:** events are tricky and have incompatibilities across browsers
  - reasons: fuzzy W3C event specs; IE disobeying web standards; etc.
- **solution:** Prototype includes many event-related features and fixes

Attaching event handlers the Prototype way

6

```
element.onevent = function;
element.observe("event", "function");
```
JS

```
// call the playNewGame function when the Play button is clicked
$("play").observe("click", playNewGame);
```
JS

- to use Prototype's event features, you must attach the handler using the DOM element
- object's observe method (added by Prototype)
  - pass the event of interest and the function to use as the handler
  - handlers must be attached this way for Prototype's event features to work

Attaching multiple event handlers with $$

7

```
// listen to clicks on all buttons with class "control" that
// are directly inside the section with ID "game"
window.onload = function() {
        var gameButtons = $$("#game > button.control");
        for (var i = 0; i < gameButtons.length; i++) {
                gameButtons[i].observe("click", gameButtonClick);
        }
};
function gameButtonClick() { ... }
```
JS

- you can use $$ and other DOM walking methods to unobtrusively attach event handlers to
- a group of related elements in your window.onload code

The Event object

8

```
function name(event) {
// an event handler function ...
}
```
JS

- Event handlers can accept an optional parameter to represent the event that is occurring. Event objects have the following properties / methods:

method / property name	description
type	what kind of event, such as "click" or "mousedown"
element() *	the element on which the event occurred
stop() **	cancels an event
stopObserving()	removes an event handler

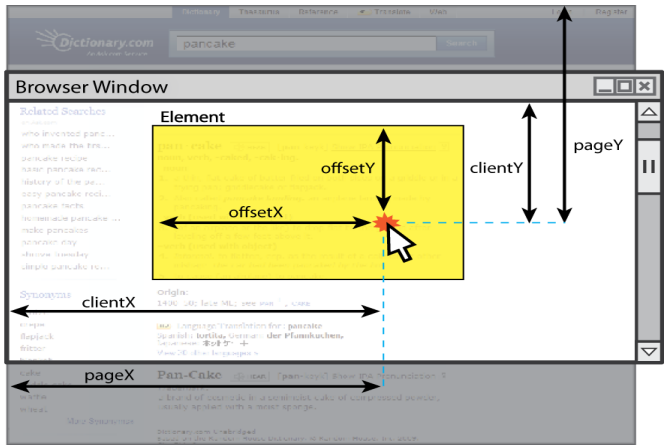
Mouse events

click	user presses/releases mouse button on this element
dblclick	user presses/releases mouse button twice on this element
mousedown	user presses down mouse button on this element
mouseup	user releases mouse button on this element

Mouse events

mouseover	mouse cursor enters this element's box
mouseout	mouse cursor exits this element's box
mousemove	mouse cursor moves around within this element's box

Mouse event objects



Mouse event objects

property/method	description
clientX, clientY	coordinates in browser window
screenX, screenY	coordinates in screen
offsetX, offsetY	coordinates in element
pointerX(), pointerY() *	coordinates in entire web page
isLeftClick() **	true if left button was pressed

* replaces non-standard properties pageX and pageY

** replaces non-standard properties button and which

The Event object

13

```
<pre id="target">Move the mouse over me!</pre>
```
HTML

```
window.onload = function() {
    $("target").observe("mousemove", showCoords);
};
function showCoords(event) {
       this.innerHTML =
       "pointer: (" + event.pointerX() + ", " +
event.pointerY() + ")\n"
       + "screen : (" + event.screenX + ", " +
       event.screenY + ")\n"
       + "client : (" + event.clientX + ", " +
event.clientY + ")";
```
JS

14

More Events and Validation

Page/window events

15

name	description
load	the browser loads the page
unload	the browser exits the page
resize	the browser window is resized
contextmenu	the user right-clicks to pop up a context menu
error	an error occurs when loading a document or an image

Page/window events

16

```
// best way to attach event handlers on page load
window.onload = function() { ... };
       document.observe("dom:loaded", function() {
       $("orderform").observe("submit", verify);
});
```
JS

Form events

event name	description
submit	form is being submitted
reset	form is being reset
change	the text or state of a form control has changed

```
window.observe("load", function() {
	$("orderform").observe("submit", verify);
});
function verify(event) {
	if ($("zipcode").value.length < 5) {
		event.stop(); // cancel form submission
unless
	} // zip code is 5 chars long
}
```
JS

Prototype and forms


```
$("id")["name"]
```
JS

- gets parameter with given name from form with given id

```
$F("id")
```
JS

- $F returns the value of a form control with the given id

```
var name = $F("username");
if (name.length < 4) {
	$("username").clear();
	$("login").disable();
}
```
JS

Prototype and forms

- other form control methods:

activate	clear	disable	enable
focus	getValue	present	select

Client-side validation code


```
<form id="exampleform" action="http://foo.com/foo.php">
```
HTML

```
window.onload = function() {
	$("exampleform").onsubmit = checkData;
};
function checkData(event) {
	if ($("city").value == "" ||
$("state").value.length != 2) {
		Event.stop(event);
		alert("Error, invalid city/state."); // show
error message
	}
}
```
JS

- forms expose onsubmit and onreset events
- to abort a form submission, call Prototype's Event.stop on the event

Regular expressions in JavaScript

- `string.match(regex)`
  - if string fits the pattern, returns the matching text; else returns null
  - can be used as a Boolean truthy/falsey test:

```
var name = $("name").value;
if (name.match(/[a-z]+/)) { ... }
```

- an i can be placed after the regex for a case-insensitive match
  - name.match(/Xenia/i) will match “xenia", “XeNiA", ...

Replacing text with regular expressions

- `string.replace(regex, "text")`
  - replaces the first occurrence of given pattern with the given text
  - `var str = “Xenia Mountrouidou";`
    `str.replace(/[a-z]/, "x") returns " Xxnia Mountrouidou"`
  - returns the modified string as its result; must be stored

```
str = str.replace(/[a-z]/, "x")
```

- a g can be placed after the regex for a global match (replace all occurrences)
  - `str.replace(/[a-z]/g, "x")` returns “Xxxxx Mxxxxxxxxxx"

Keyboard/text events

name	description
keydown	user presses a key while this element has keyboard focus
keyup	user releases a key while this element has keyboard focus
keypress	user presses and releases a key while this element has keyboard focus
focus	this element gains keyboard focus
blur	this element loses keyboard focus
select	this element's text is selected or deselected)

Key event objects

property name	description
keyCode rototype's key code constants	ASCII integer value of key that was pressed (convert to char with String.fromCharCode)
altKey, ctrlKey, shiftKey	true if Alt/Ctrl/Shift key is being held

Key event objects

Event.KEY_BACKSPACE	Event.KEY_DELETE	Event.KEY_DOWN	Event.KEY_END
Event.KEY_ESC	Event.KEY_HOME	Event.KEY_LEFT	Event.KEY_PAGEDOWN
Event.KEY_PAGEUP	Event.KEY_RETURN	Event.KEY_RIGHT	Event.KEY_TAB
Event.KEY_UP			

- issue: if the event you attach your listener to doesn't have the focus, you won't hear the event
  - possible solution: attach key listener to entire page body, outer element, etc.