

---

# Bubbles and trees with jQuery

Web 2.0 made easy

Bastian Feder <php@bastian-feder.de>

Jakob Westhoff <jakob@php.net>

WebtechCon

November 18, 2009

- Bastian Feder
  - Application developer
  - Using PHP since 2001
  - Using Javascript since 2002
  - Working on the papaya CMS since 01.2008

- Jakob Westhoff
  - Web-developer for more than 6 years
  - Author of Activebar2 (used by <http://ie6update.com>)
  - Computer science student at the TU Dortmund
  - Active in different Open Source projects

# Goals of this session

---

- You will learn about:

# Goals of this session

---

- You will learn about:
  - jQuery basics and need to know facts

# Goals of this session

---

- You will learn about:
  - jQuery basics and need to know facts
  - Effects and animation

# Goals of this session

---

- You will learn about:
  - jQuery basics and need to know facts
  - Effects and animation
  - Event-handling mechanisms

# Goals of this session

---

- You will learn about:
  - jQuery basics and need to know facts
  - Effects and animation
  - Event-handling mechanisms
  - Asynchronous request handling (AJAX)

# jQuery about itself

---

From the jQuery Website:

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

# jQuery about itself

---

From the jQuery Website:

jQuery is a fast and concise JavaScript **Library** that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

# jQuery about itself

---

From the jQuery Website:

jQuery is a **fast** and **concise** JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

# jQuery about itself

---

From the jQuery Website:

jQuery is a fast and concise JavaScript Library that simplifies HTML **document traversing**, **event handling**, **animating**, and **Ajax interactions** for rapid web development. jQuery is designed to change the way that you write JavaScript.

# jQuery about itself

---

From the jQuery Website:

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. **jQuery is designed to change the way that you write JavaScript.**

# Introduction to jQuery

---

- Compact
  - only 56kb minified
  - 19kb minified and gzipped

# Introduction to jQuery

---

- Compact
  - only 56kb minified
  - 19kb minified and gzipped
- Cross-browser compatible
  - Internet Explorer 6.0+
  - Firefox 2+
  - Safari 3.0+
  - Opera 9.0+
  - Chrome

# Introduction to jQuery

---

- Compact
  - only 56kb minified
  - 19kb minified and gzipped
- Cross-browser compatible
  - Internet Explorer 6.0+
  - Firefox 2+
  - Safari 3.0+
  - Opera 9.0+
  - Chrome
- Easily extendable

# jQuery Example

---

```
$("p#example").addClass("highlight").fadeIn("slow");
```

# jQuery Example

---

```
$("#p#example").addClass("highlight").fadeIn("slow");
```

- Find all paragraphs with the id example

# jQuery Example

---

```
$("p#example").addClass("highlight").fadeIn("slow");
```

- Find all paragraphs with the id `example`
- Add the css class `highlight` to them

# jQuery Example

---

```
$("p#example").addClass("highlight").fadeIn("slow");
```

- Find all paragraphs with the id `example`
- Add the css class `highlight` to them
- Fade in the paragraph slowly

# What comes next?

---

## Basics

# DOM Object proxy

---

- jQuery is accessed using \$ or jQuery

# DOM Object proxy

---

- jQuery is accessed using \$ or jQuery
- Document centric
  - \$(css selector).operation

# DOM Object proxy

---

- jQuery is accessed using \$ or jQuery
- Document centric
  - \$(css selector).operation

```
$(document).ready(  
  function() { ... }  
);
```

# DOM Object proxy

- jQuery is accessed using `$` or `jQuery`
- Document centric
  - `$(css selector).operation`

```
$(document).ready(  
  function() { ... }  
);
```

- Get the document object **extended** with jQuery functionality

# DOM Object proxy

- jQuery is accessed using `$` or `jQuery`
- Document centric
  - `$(css selector).operation`

```
$(document).ready(  
  function() { ... }  
);
```

- Get the document object **extended** with jQuery functionality
- Register an **event handler** executed once the document is **loaded**

# Interoperability

---

- Different Javascript libs use \$ as a shortcut

# Interoperability

---

- Different Javascript libs use \$ as a shortcut
- jQuery allows to rename its \$ shortcut

# Interoperability - Example

---

- Include jQuery **after** other libraries

```
1 <script src="prototype.js"></script>
```

```
2 <script src="jQuery.js"></script>
```

# Interoperability - Example

- Include jQuery **after** other libraries

```
1 <script src="prototype.js"></script>
2 <script src="jQuery.js"></script>
```

- Use its **noConflict** function

```
1 var $j = jQuery.noConflict();
2
3 // Use jQuery
4 $j(someElement).someFunction();
5
6 // Use prototype
7 $(someElement).someFunction();
```

# Fluent interface (Chaining)

---

- Every method returns the object currently working on

# Fluent interface (Chaining)

- Every method returns the object currently working on

```
1 $( 'span ' )  
2   .addClass( 'red ' )  
3   .css( 'text-weight ' , ' bold ' );
```

# Fluent interface (Chaining)

- Every method returns the object currently working on

```
1 $( 'span ' )  
2   .addClass( 'red ' )  
3   .css( 'text-weight ' , 'bold ' );
```

- Element selections are tracked and can be restored

# Fluent interface (Chaining)

- Every method returns the object currently working on

```
1 $( 'span' )
2   .addClass( 'red' )
3   .css( 'text-weight', 'bold' );
```

- Element selections are tracked and can be restored

```
1 $( 'span' )
2   .parent()
3     .addClass( 'neat' )
4     .parent()
5       .addClass( 'fancy' )
6       .end()
7     .end()
8   .addClass( 'red' )
```

# CSS 3 complaint selectors

---

- Basic

```
$('#id'), $('.class'), $('element')
```

# CSS 3 complaint selectors

- Basic

`$('#id')`, `$('.class')`, `$( 'element' )`

- Hierarchical

`$( 'ancestor descendant' )`, `$( 'parent > child' )`

# CSS 3 complaint selectors

- Basic

`$('#id')`, `$('.class')`, `$( 'element' )`

- Hierarchical

`$( 'ancestor descendant' )`, `$( 'parent > child' )`

- Filters

`$( 'p: first' )`, `$( 'div: visible' )`, `$( '*: header' )`

# CSS 3 complaint selectors

- Basic

`$('#id')`, `$('.class')`, `$( 'element' )`

- Hierarchical

`$( 'ancestor descendant' )`, `$( 'parent > child' )`

- Filters

`$( 'p: first' )`, `$( 'div: visible' )`, `$( '*: header' )`

- Many more

`$( 'div: has( img. thumbnail [ src$ = .png ] : not ( : hidden ) )' )`

# DOM Manipulation

---

- 1 Select the nodes to work with:

# DOM Manipulation

---

1 Select the nodes to work with:

```
$( 'p > div ' )
```

# DOM Manipulation

---

- 1 Select the nodes to work with:

```
$( 'p > div ' )
```

- 2 Use DOM manipulation functions on it

# DOM Manipulation

---

- 1 Select the nodes to work with:

```
$( 'p > div ' )
```

- 2 Use DOM manipulation functions on it
  - Insert nodes
  - Change content
  - Replace nodes
  - Remove nodes

# Insert nodes

---

- Inside

# Insert nodes

---

- Inside

- . `append ( content )`
  - . `prepend ( content )`

# Insert nodes

---

- Inside

- .append( content )
  - .prepend( content )

- Outside

# Insert nodes

---

## ■ Inside

- `.append( content )`
- `.prepend( content )`

## ■ Outside

- `.after( content )`
- `.before( content )`

# Insert nodes

---

- Inside

- .append( content )
  - .prepend( content )

- Outside

- .after( content )
  - .before( content )

- Around

# Insert nodes

---

## ■ Inside

- `.append( content )`
- `.prepend( content )`

## ■ Outside

- `.after( content )`
- `.before( content )`

## ■ Around

- `.wrap( content )`
- `.wrapAll( content )`

# Insert nodes

## ■ Inside

- `.append( content )`
- `.prepend( content )`

## ■ Outside

- `.after( content )`
- `.before( content )`

## ■ Around

- `.wrap( content )`
- `.wrapAll( content )`

`content` may be a **HTML snippet**, a **jQuery object** or a **DOM node**

# Changing contents

---

- Change **simple** text content

# Changing contents

---

- Change **simple** text content

```
$( 'p#id' ).text( 'foobar' );
```

# Changing contents

---

- Change **simple** text content

```
$( 'p#id ' ).text( 'foobar' );
```

- Change **HTML** content (innerHTML)

# Changing contents

---

- Change **simple** text content

```
$( 'p#id ' ). text( ' foobar ' );
```

- Change **HTML** content (innerHTML)

```
$( 'p#id ' ). html( '<b>foo </b>bar ' );
```

# Replace nodes

---

- Replace selected nodes with **new** nodes

# Replace nodes

---

- Replace selected nodes with **new** nodes

```
$( 'p#id ' ).replaceWith( '<div>foobar </div>' );
```

# Replace nodes

---

- Replace selected nodes with **new** nodes

```
$( 'p##id ' ).replaceWith( '<div>foobar </div>' );
```

- Replace selected nodes with **already existing** nodes

# Replace nodes

---

- Replace selected nodes with **new** nodes

```
$( 'p#id' ).replaceWith( '<div>foobar</div>' );
```

- Replace selected nodes with **already existing** nodes

```
$( 'p#id' ).replaceAll( 'div' );
```

# Replace nodes

- Replace selected nodes with **new** nodes

```
$( 'p#id' ).replaceWith( '<div>foobar</div>' );
```

- Replace selected nodes with **already existing** nodes

```
$( 'p#id' ).replaceAll( 'div' );
```

Replaces **all** elements which match the selector **div** with the contents of **p#id**

# Remove nodes

---

- Remove all child nodes

# Remove nodes

---

- Remove all child nodes

```
$( 'p#id ' ).empty();
```

# Remove nodes

---

- Remove all child nodes

```
$( 'p#id ' ). empty ( ) ;
```

- Remove the element itself

# Remove nodes

---

- Remove all child nodes

```
$( 'p#id' ).empty();
```

- Remove the element itself

```
$( 'p#id' ).remove();
```

- Iterate over an arbitrary set of elements:

- Iterate over an arbitrary set of elements:

```
1  $('p').each(  
2    function() {  
3      $(this).html('<b>Lorem ipsum... </b>');  
4    }  
5  );
```

- Iterate over an arbitrary set of elements:

```
1  $('p').each(  
2    function() {  
3      $(this).html('<b>Lorem ipsum... </b>');  
4    }  
5  );
```

- this is mapped to the **currently iterated** DOM element

# What comes next?

---

## Animation

# Predefined animations

---

jQuery comes with a number of **predefined** effects

# Predefined animations

---

jQuery comes with a number of **predefined** effects

- Fade in and out elements

```
$( 'p' ). fadeIn (1000);  
$( 'p' ). fadeOut (500);  
$( 'p' ). fadeTo (800, 0.5);
```

# Predefined animations

jQuery comes with a number of **predefined** effects

- Fade in and out elements

```
$( 'p' ). fadeIn ( 1000 );  
$( 'p' ). fadeOut ( 500 );  
$( 'p' ). fadeTo ( 800, 0.5 );
```

- Slide in and out

```
$( 'p' ). slideIn ( 1000 );  
$( 'p' ). slideOut ( 500 );  
$( 'p' ). slideToggle ( 800 );
```

# Predefined animations

---

- Show and hide elements by using a combination of fading and sliding

```
$( 'p' ).show(1000);  
$( 'p' ).hide(500);  
$( 'p' ).toggle(800);
```

# Predefined animations

---

- Show and hide elements by using a combination of fading and sliding

```
$( 'p' ). show ( 1000 );  
$( 'p' ). hide ( 500 );  
$( 'p' ). toggle ( 800 );
```

- Called **without** the duration time these three effects are **instantaneous**

# Predefined animations

- Show and hide elements by using a combination of fading and sliding

```
$( 'p' ).show(1000);  
$( 'p' ).hide(500);  
$( 'p' ).toggle(800);
```

- Called **without** the duration time these three effects are **instantaneous**
- All animation methods can be called with an **optional callback** executed after the animation **finished**.

# Custom animations

---

- Use the `animate` method for **custom** animations

# Custom animations

---

- Use the `animate` method for `custom` animations
- Every `numeric` property can be animated

# Custom animations

---

- Use the `animate` method for `custom` animations
- Every `numeric` property can be animated
  - relative values
  - absolute values

# Custom animations

- Use the `animate` method for **custom** animations
- Every **numeric** property can be animated
  - relative values
  - absolute values

```
1  $('p').animate(  
2    { top: '100px',  
3      height: '+=50px',  
4      opacity: '-=0.5'  
5    },  
6    1000  
7  );
```

# A small detour: Using the queue

---

- jQuery uses a **queue** to manage it's animations internally

# A small detour: Using the queue

---

- jQuery uses a **queue** to manage it's animations internally
- Can be used to **synchronize** any otherwise asynchronous operation

# A small detour: Using the queue

---

- jQuery uses a **queue** to manage it's animations internally
- Can be used to **synchronize** any otherwise asynchronous operation
- Queues are associated with DOM objects

# A small detour: Using the queue

- jQuery uses a **queue** to manage it's animations internally
- Can be used to **synchronize** any otherwise asynchronous operation
- Queues are associated with DOM objects

```
1  $('p').queue(function () {
2      someAsyncCallWithCallback(function () {
3          $(this).dequeue();
4      })
5  });
6
7  $('p').queue(function () {
8      /* ... */
9  });
```

# A small detour: Using the queue

- jQuery uses a **queue** to manage it's animations internally
- Can be used to **synchronize** any otherwise asynchronous operation
- Queues are associated with DOM objects

```
1 $( 'p' ).queue( function () {
2     someAsyncCallWithCallback( function () {
3         $( this ).dequeue();
4     })
5 });
6
7 $( 'p' ).queue( function () {
8     /* ... */
9 });
```

- Always used **\$(this).dequeue()** to keep the queue running

# What comes next?

---

## Events

# Event handling basics

---

- **Event abstraction** to not rely on browser specific behaviour

# Event handling basics

---

- **Event abstraction** to not rely on browser specific behaviour
- **Generic** event binding/unbinding methods

# Event handling basics

---

- **Event abstraction** to not rely on browser specific behaviour
- **Generic** event binding/unbinding methods
- **Shortcut helper** methods for different events

# Event handling basics

---

- **Event abstraction** to not rely on browser specific behaviour
- **Generic** event binding/unbinding methods
- **Shortcut helper** methods for different events
- **Live** event binding (event delegation)

# Event handling basics

---

- **Event abstraction** to not rely on browser specific behaviour
- **Generic** event binding/unbinding methods
- **Shortcut helper** methods for different events
- **Live** event binding (event delegation)
- **Event namespaces**

# Generic event handling

---

- Register an event handler for a specific element

```
$( 'p' ).bind( 'click ', function( e ) {} );
```

# Generic event handling

---

- Register an event handler for a specific element

```
$( 'p' ).bind( 'click', function( e ) {} );
```

- Remove a registered event handler

```
$( 'p' ).unbind( 'click', [ function ] );
```

# Generic event handling

- Register an event handler for a specific element

```
$( 'p' ).bind( 'click', function( e ) {} );
```

- Remove a registered event handler

```
$( 'p' ).unbind( 'click', [ function ] );
```

- Register event handler for one time execution

```
$( 'p' ).one( "mousemove", function( e ) {} );
```

# Generic event handling

- Register an event handler for a specific element

```
$( 'p' ).bind( 'click', function( e ) {} );
```

- Remove a registered event handler

```
$( 'p' ).unbind( 'click', [ function ] );
```

- Register event handler for one time execution

```
$( 'p' ).one( "mousemove", function( e ) {} );
```

- Trigger an event and all of its event handlers

```
$( 'p' ).trigger( "click" );
```

# Event object abstraction

---

- Event object **always send** to event handler
  - No need for `window.event`

# Event object abstraction

---

- Event object **always send** to event handler
  - No need for `window.event`
- Normalized according to W3C standard (`jQuery.Event`)

# Event object abstraction

- Event object **always send** to event handler
  - No need for `window.event`
- Normalized according to W3C standard (`jQuery.Event`)

```
function handler(event) {  
    event.type;  
    event.target;  
    event.pageX;  
    event.pageY;  
    /* ... */  
  
    event.preventDefault();  
    event.stopPropagation();  
    /* ... */  
}
```

# Event helper

---

Helper methods for common tasks

- Shortcut methods for all kinds of events

# Event helper

## Helper methods for common tasks

- Shortcut methods for all kinds of events

```
$( 'p' ).click( function () { /* ... */ } );
```

```
dblclick , keydown , keyup , mousedown ,  
mouseenter , mousemove , scroll , ...
```

## Helper methods for common tasks

- Shortcut methods for all kinds of events

```
$( 'p' ).click( function () { /* ... */ } );
```

```
dblclick , keydown , keyup , mousedown ,  
mouseenter , mousemove , scroll , ...
```

- Often needed functionality

## Helper methods for common tasks

- Shortcut methods for all kinds of events

```
$('#p').click(function() { /* ... */ });
```

```
dblclick, keydown, keyup, mousedown,  
mouseenter, mousemove, scroll, ...
```

- Often needed functionality

```
$('#p').hover(over, out);
```

```
$('#p').toggle(fn, fn2, fn3, fn4, ...);
```

- Bind event handlers to items which do not exist yet

- Bind event handlers to items which do not exist yet

```
$( 'p' ).live( "click", function( e ) {  
    $( this ).css( 'background-color', 'red' );  
});
```

# Live events

---

- Bind event handlers to items which do not exist yet

```
$( 'p' ).live( "click", function( e ) {  
    $( this ).css( 'background-color', 'red' );  
});
```

- Remove live event handlers again

# Live events

---

- Bind event handlers to items which do not exist yet

```
$( 'p' ).live( "click", function( e ) {  
    $( this ).css( 'background-color', 'red' );  
});
```

- Remove live event handlers again

```
$( 'p' ).die( "click" );
```

# Live events

- Bind event handlers to items which do not exist yet

```
$('#p').live("click", function(e) {  
    $(this).css('background-color', 'red');  
});
```

- Remove live event handlers again

```
$('#p').die("click");
```

live and die are using **event delegation**, which enhances performance with big element counts

# Event namespaces

---

- To safely **unbind** events you need their **callback function**

# Event namespaces

---

- To safely `unbind` events you need their `callback function`
- jQuery event `namespaces` solve this problem really elegant

# Event namespaces

- To safely **unbind** events you need their **callback function**
- jQuery event **namespaces** solve this problem really elegant

```
$( 'div' ).bind( 'click.mynamespace', function(e){  
    /* ... */  
});
```

# Event namespaces

- To safely `unbind` events you need their `callback function`
- jQuery event `namespaces` solve this problem really elegant

```
$( 'div' ).bind( 'click.mynamespace', function(e){  
    /* ... */  
});
```

- Unbinding the click event:

```
$( 'div' ).unbind( 'click.mynamespace' );
```

# Event namespaces

- To safely **unbind** events you need their **callback function**
- jQuery event **namespaces** solve this problem really elegant

```
$( 'div' ).bind( 'click.mynamespace', function(e){  
    /* ... */  
});
```

- Unbinding the click event:

```
$( 'div' ).unbind( 'click.mynamespace' );
```

- Unbinding **all events** in a namespace:

```
$( 'div' ).unbind( '.mynamespace' );
```

# What comes next?

---

Ajax

# Shortcuts for AJAX requests

---

- Load external HTML directly:

```
$( 'p#id' ).load( "somefile.html_#id_.class" )
```

# Shortcuts for AJAX requests

- Load external HTML directly:

```
$( 'p#id' ).load( "somefile.html_#id_.class" )
```

- Fire a HTTP **GET** request:

```
$.get( "http://..." , [data] , [callback] , [type] );
```

# Shortcuts for AJAX requests

- Load external HTML directly:

```
$( 'p#id' ).load( "somefile.html_#id_.class" )
```

- Fire a HTTP GET request:

```
$.get( "http://...", [data], [callback], [type] );
```

- Fire a HTTP POST request:

```
$.post( "http://...", [data], [callback], [type] );
```

# Shortcuts for AJAX requests

- Load external HTML directly:

```
$( 'p#id' ).load( "somefile.html_#id_.class" )
```

- Fire a HTTP GET request:

```
$.get( "http://...", [data], [callback], [type] );
```

- Fire a HTTP POST request:

```
$.post( "http://...", [data], [callback], [type] );
```

- Load and execute a remote `<script>`:

```
$.getScript( "somescript.js", [callback] );
```

# Shortcuts for AJAX requests

- Load external HTML directly:

```
$( 'p#id' ).load( "somefile.html_#id_.class" )
```

- Fire a HTTP GET request:

```
$.get( "http://...", [data], [callback], [type] );
```

- Fire a HTTP POST request:

```
$.post( "http://...", [data], [callback], [type] );
```

- Load and execute a remote `<script>`:

```
$.getScript( "somescript.js", [callback] );
```

Callback functions are **only** executed **on success**

# Advanced AJAX requests

---

- Manual complex AJAX requests are possible

# Advanced AJAX requests

- Manual complex AJAX requests are possible

```
1 $.ajax({
2     url: /* Request url */,
3     beforeSend: function(xhr) {},
4     complete: function(xhr) {},
5     success: function(code) {},
6     error: function(xhr, status, error) {},
7     ...
8 });
```

# Advanced AJAX requests

- Manual complex AJAX requests are possible

```
1 $.ajax({
2     url: /* Request url */,
3     beforeSend: function(xhr) {},
4     complete: function(xhr) {},
5     success: function(code) {},
6     error: function(xhr, status, error) {},
7     ...
8 });
```

Other possible options are: `async`, `cache`, `contentType`, `data`, `dataFilter`, `dataType`, `global`, `ifModified`, `jsonp`, `password`, `processData`, `scriptCharset`, `type`, `username`, `xhr`

## Questions, comments or annotations?

Slides: <http://slideshare.net>

### Contact:

- Jakob Westhoff <jakob@php.net>
  - <http://westhoffswelt.de>
  - @jakobwesthoff
- Bastian Feder <php@bastian-feder.de>
  - <http://bastian-feder.de>
  - @lapistano