

---

# Piece by piece

An introduction to jQuery-UI widget development

Jakob Westhoff <jakob@php.net>

itemis Votragsreihe Dortmund

6. June, 2011

Jakob Westhoff

@jakobwesthoff

<http://westhoffswelt.de>

## About You?

Questions?

Questions!

# Questions answered today

---

- 1 What is jQuery?
- 2 What is jQuery UI?
- 3 What features and widgets does jQuery UI provide?
- 4 How does the jQuery UI Theme generation work?
- 5 In which way can jQuery UI be used to write own widgets?

# What comes next?

---

jQuery

# 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.**

# Working with jQuery

---

```
$(".tooltip").addClass("highlight").fadeIn("slow");
```

# Working with jQuery

```
$(".tooltip").addClass("highlight").fadeIn("slow");
```

- Document centric
- Operates on **sets** accessed using \$ or jQuery
  - `$(css selector).operation`
  - `jQuery(css selector).operation`
- Fluent interface paradigm
  - `operation().operation().operation()`

# What comes next?

---

## jQuery-UI

# jQuery-UI about itself

---

From the jQuery-UI Website:

jQuery UI provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.

# jQuery-UI about itself

---

From the jQuery-UI Website:

jQuery UI provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, **built on top of the jQuery JavaScript Library**, that you can use to build highly interactive web applications.

# jQuery-UI about itself

---

From the jQuery-UI Website:

jQuery UI provides abstractions for low-level interaction and animation, advanced effects and **high-level, themeable widgets**, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.

# jQuery-UI about itself

---

From the jQuery-UI Website:

jQuery UI provides abstractions for **low-level interaction and animation**, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.

# jQuery-UI about itself

---

From the jQuery-UI Website:

jQuery UI provides abstractions for low-level interaction and animation, **advanced effects** and high-level, themeable widgets, built on top of the jQuery JavaScript Library, that you can use to build highly interactive web applications.

# jQuery-UI - A short overview

- A set of high level widgets
- Low level framework to create own Widgets, Behaviors and Effects
- Compatible with all major Browsers (thanks to jQuery)
- Modular code base between 5-220kb
- Fully themeable using a graphical tool: ThemeRoller
- Highly extensible
- Download: <http://jqueryui.com>

# What comes next?

---

## Widgets

# Widgets included in jQuery UI

---

- Accordion
- Autocomplete
- Button
- Dialog
- Slider
- Tabs
- Datepicker
- Progressbar

Live Demo - jQuery UI Widgets

## Live Demo

▶ Next section

## Section 1

Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam. Integer ut neque. Vivamus nisi metus, molestie vel, gravida in, condimentum sit amet, nunc. Nam a nibh. Donec suscipit eros. Nam mi. Proin viverra leo ut odio. Curabitur malesuada. Vestibulum a velit eu ante scelerisque vulputate.

## Section 2

## Section 3

## Section 4

## Section 1

## Section 2

Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum sit amet purus. Vivamus hendrerit, dolor at aliquet laoreet, mauris turpis porttitor velit, faucibus interdum tellus libero ac justo. Vivamus non quam. In suscipit faucibus urna.

## Section 3

## Section 4

## Section 1

## Section 2

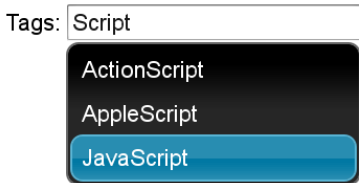
## Section 3

Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisque lobortis. Phasellus pellentesque purus in massa. Aenean in pede. Phasellus ac libero ac tellus pellentesque semper. Sed ac felis. Sed commodo, magna quis lacinia ornare, quam ante aliquam nisi, eu iaculis leo purus venenatis dui.

- List item one
- List item two
- List item three

## Section 4

# Autocomplete



# Buttons

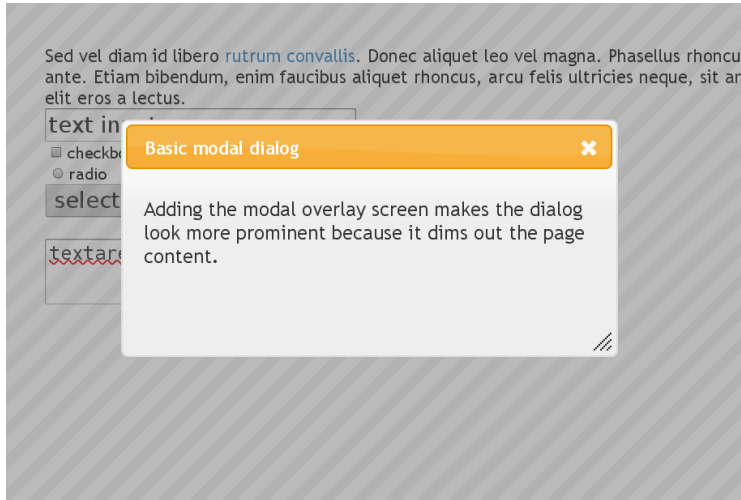
---

A button element

A submit button

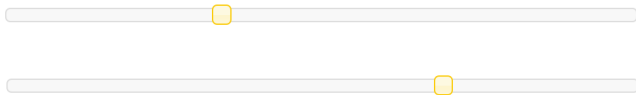
An anchor

# Dialog



# Slider

---



# Tabs

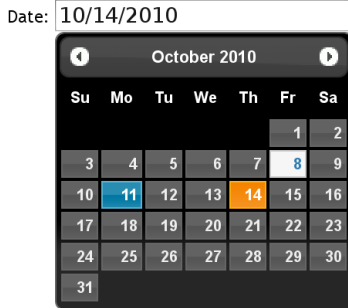
Nunc tincidunt

Proin dolor

Aenean lacinia

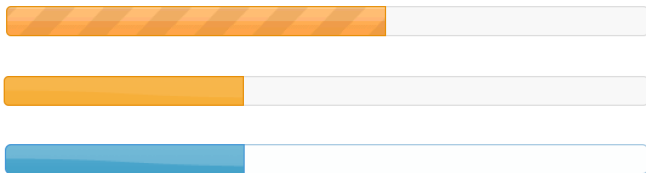
Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.

# Datepicker



# Progressbar

---



# What comes next?

---

## Behaviors

# Behaviors included in jQuery UI

---

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

# What comes next?

---

## Effects

# Effects included in jQuery UI

---

- Blind
- Bounce
- Clip
- Drop
- Explode
- Fold
- Highlight
- Pulsate
- ...

# What comes next?

---

## Animation Enhancements

# Animation Enhancements over default jQuery

---

- Color animations
- New easing functions
  - easeInBounce
  - easeInQuad
  - ...
- CSS class based animation

Live Demo - Animation Enhancements

Live Demo

# What comes next?

---

## Calling conventions

# Calling conventions

---

- For each widget (behavior) **one** function is registered
- The function can be called on **any** created jQuery **set** (`jQuery.fn`)
- The **widgets name** is used as function name
- **All** interaction with the Widget is accomplished through this function

# Calling conventions

---

- Widget creation

```
$("#id").autocomplete();  
$("#id").autocomplete({...});
```

# Calling conventions

---

- Widget creation

```
$("#id").autocomplete();  
$("#id").autocomplete({...});
```

- Invoking a method without arguments

```
$("#id").autocomplete("method");
```

# Calling conventions

- Widget creation

```
$("#id").autocomplete();  
$("#id").autocomplete({...});
```

- Invoking a method without arguments

```
$("#id").autocomplete("method");
```

- Invoking a method with arguments

```
$("#id").autocomplete("method", arg1, arg2, ...);
```

Live Demo - Calling jQuery UI

Live Demo

# What comes next?

---

## Creating your own Widget

# Creating a Widget without jQuery-UI

- Widgets usually have a state
  - Excessive use of `.data` method
  - Nested closures and custom objects
- Different initialization phases
- Handling of default and user options
- Multiple public methods
- Destruct and remove Widget on request

# What comes next?

---

## Widget Factory

# The Widget factory of jQuery-UI

---

- Instance based persistent states
- "Magic" methods for different initialization phases
- Automatic merging of default and user supplied options
- Custom methods without namespace pollution
- Distinguish between public and private methods

# What comes next?

---

## Example Widget

# The SparkleBar Widget

- A custom made progressbar
- Less flexible due to heavy usage of images
- More sophisticated graphical design possible



Live Demo - SparkleBar

Live Demo

# Filename conventions

---

- Name of this widget: `sparklebar`
- Namespace of this widget: `ui`
- All widgets should follow a certain filename convention:
  - `jquery.ui.sparklebar.js`
  - `jquery.ui.sparklebar.css`
  - `jquery.ui.sparklebar/image.png`
  - ...

# The widget factory in action

---

- Call `$.widget` to create a new Widget
- First argument: namespace and identifier
- Namespaces **do not** protect against naming conflicts
- All default widgets use the `ui` namespace
- Using the `ui` namespace for your own widgets is just fine

# The widget factory in action

---

```
1 $.widget(  
2   'ui.sparklebar',  
3   {  
4     /* Widget implementation */  
5   }  
6 );
```

# What comes next?

---

## Initialization

# "Magic" `_create` method

---

- `_create` function is automatically invoked on widget creation
- Only called once for each widget

# Access to the targeted element

---

- Targeted element stored in `element` property
- Saved as jQuery set
- Other magic properties exist: `options`, `widgetName`, `widgetEventPrefix`

# "Magic" \_create method

```
1 $.widget( "ui.sparklebar", {
2   _create: function() {
3
4     this.element.empty();
5
6     $( '<img_>' ).appendTo(
7       this.element
8     );
9
10    // ... Set needed properties of img ...
11
12  }
13 });
```

# "Magic" \_create method

```
1 $.widget( "ui.sparklebar", {  
2   _create: function() {  
3  
4     this.element.empty();  
5  
6     $( '<img_>' ).appendTo(  
7       this.element  
8     );  
9  
10    // ... Set needed properties of img ...  
11  
12  }  
13 });
```

# "Magic" \_create method

```
1 $.widget( "ui.sparklebar", {
2   _create: function() {
3
4     this.element.empty();
5
6     $( '<img_>' ).appendTo(
7       this.element
8     );
9
10    // ... Set needed properties of img ...
11
12  }
13 });
```

# "Magic" \_create method

```
1 $.widget( "ui.sparklebar", {
2   _create: function() {
3
4     this.element.empty();
5
6     $( '<img_>' ).appendTo(
7       this.element
8     );
9
10    // ... Set needed properties of img ...
11
12  }
13 });
```

# "Magic" `_init` method

- `_init` is invoked every time the widget function is called without a method name
  - `$("#id").sparklebar()`
  - `$("#id").sparklebar({...})`
- Little brother of the `_create` function
- Called **after** `_create` has been called

# "Magic" \_init method

```
1 $.widget( "ui.sparklebar", {
2   _init: function() {
3
4     // Set DOM properties based on provided options
5     // Initialize state of the widget
6
7   }
8 });
```

# "Magic" \_init method

```
1 $.widget( "ui.sparklebar", {  
2   _init: function() {  
3  
4     // Set DOM properties based on provided options  
5     // Initialize state of the widget  
6  
7   }  
8 });
```

# "Magic" \_init method

```
1 $.widget( "ui.sparklebar", {
2   _init: function() {
3
4     // Set DOM properties based on provided options
5     // Initialize state of the widget
6
7   }
8 });
```

# `_init` and `_create`

---

- Use `_create` to **prepare the DOM** for the widget
- Use `_init` to **initialize widget** based on options

# What comes next?

---

## Methods / Properties

# Widget methods and properties

---

- Use custom properties and methods to structure your code
- Both are defined in the Widget object

# Widget methods and properties

- Use custom properties and methods to structure your code
- Both are defined in the Widget object

```
1 $.widget(  
2   'ui.sparklebar',  
3   {  
4     'currentValue': 42  
5     'value': function() {...}  
6   }  
7 );
```

# One instance per Widget

---

- A new instance is created for each Widget invocation
- Properties are persistent

# One instance per Widget

- A new instance is created for each Widget invocation
- Properties are persistent

```
1 $.widget( "ui.sparklebar", {
2
3     "value": function(val) {
4         this.currentValue = val;
5
6         // Display progressbar change
7     }
8
9 });
```

# Fluent interface and returned values

---

- Widget factory takes care of fluent interface handling
- Returning a value from a method circumvents this behavior

# Fluent interface and returned values

- Widget factory takes care of fluent interface handling
- Returning a value from a method circumvents this behavior

```
1 $.widget( "ui.sparklebar", {  
2     getValue: function(val) {  
3         return this.currentValue;  
4     }  
5 });
```

# Private methods

---

- Methods prefixed with underscore (`_`) are private
- Properties are always private

# Private methods

- Methods prefixed with underscore (`_`) are private
- Properties are always private

```
1 $.widget( "ui.sparklebar", {  
2     _updateVisualRepresentation: function() {  
3         ...  
4     }  
5 });
```

# What comes next?

---

## Options

# Configuration options for your widget

---

- Most widgets need configuration in order to be reusable
- Named options can be provided during widget creation
- Remember: `$("#id").sparklebar({...})`

# Handling of Options

---

- Options are stored in the *magic* options property
- Options are always **optional**
- Default values need to be provided for each option

# Handling of Options - options property

```
1 $.widget( "ui.sparklebar", {  
2   options: {  
3     color: 'orange',  
4     initialValue: 0,  
5     animate: true  
6   },  
7 });
```

# Handling of Options - Access merged options

---

- Default options are automatically merged with user options
- Result is written back to the `options` property

# Handling of Options - Access merged options

```
1 $("#id").sparklebar({initialValue: 40});
2
3 $.widget( 'ui.sparklebar', {
4     _init: function() {
5         this.currentValue = this.options.initialValue;
6     },
7 });
```

# What comes next?

---

## Destruction

# Destruction of Widgets

---

- Every widget inherits certain methods
- Their default behavior can be extended if necessary
- `destroy()` is one of them
- Others are: `option`, `widget`, `_setOption`, `enable` and `disable`

# Default behavior of the `destroy` method

---

- Remove all events namespaced with the widget identifier
- Remove the DOM node stored in the `element` property
- Remove the DOM node returned by the `widget` method

# Extending methods - The destroy method

```
1 $.widget( "ui.sparklebar", {
2   destroy: function() {
3     // Call the base implementation of destroy
4     $.Widget.prototype.destroy.apply( this );
5
6     // Do more cleanup here
7   },
8 });
```

# What comes next?

---

## Event namespaces

# Event namespaces - a mostly unknown feature

- Group registered events by a certain identifier
- As usual use `bind` in order to listen for events
- Namespaces and event types separated using a **dot** (`.`)
  - eg. `click.namespace`
- Namespaces allow easy de-registration/management
  - `.unbind("click.namespace")`
  - `.unbind(".namespace")`

# Event namespaces in widget development

---

- **Always** use event namespaces inside your widgets
- Use the widget identifier as namespace
- No conflict between your event handlers and others
- Easy de-registration possible without remembering the callback

# What comes next?

---

## Visual Appearance

# CSS formatting in widgets

---

- Use CSS rules for visual formatting if possible
- Decoupling of functionality and representation
- Store CSS in appropriate position
  - `jquery.ui.sparklebar.css`
- Always use classes not ids (multiple invocation)
- Use widget name and namespace as prefix
  - `ui-sparklebar-container`
- Follow the hierarchy of your elements
  - `ui-sparklebar-container-bar`

# ThemeRoller - An online theme creator

---

- ThemeRoller is a WYSIWYG application to design themes
- Specifically written for jQuery UI
- Realized as a web application
- Creates needed CSS themes & images on the fly

Live Demo - ThemeRoller

Live Demo

# ThemeRoller

---

- Custom Widgets can profit from ThemeRoller as well
- Use jQuery-UI CSS framework for your elements
- Just a set of predefined CSS classes

- Structural helper
  - `ui-helper-hidden`, `ui-helper-clearfix`, ...
- Widget look and feel
  - `ui-widget-header`, `ui-widget-content`, ...
- Button and input element marker
  - `ui-priority-primary`, `ui-state-default`, ...
- Visual states
  - `ui-state-highlight`, `ui-state-error`, ...
- Icons
  - `ui-icon`, `ui-icon-folder-collapsed`, ...

# What comes next?

---

## Conclusion

# What you have learned today - In general

---

- 1 Obey the filename rules: `jquery.namespace.widget.js`

# What you have learned today - In general

---

- 1 Obey the filename rules: `jquery.namespace.widget.js`
- 2 Use the Widget factory (`$.widget`)

# What you have learned today - In general

---

- 1 Obey the filename rules: `jquery.namespace.widget.js`
- 2 Use the Widget factory (`$.widget`)
- 3 Don't be afraid of using many options

# What you have learned today - In general

---

- 1 Obey the filename rules: `jquery.namespace.widget.js`
- 2 Use the Widget factory (`$.widget`)
- 3 Don't be afraid of using many options
- 4 Always use Event namespaces to register events

# What you have learned today - In general

- 1 Obey the filename rules: `jquery.namespace.widget.js`
- 2 Use the Widget factory (`$.widget`)
- 3 Don't be afraid of using many options
- 4 Always use Event namespaces to register events
- 5 Prefix your CSS classes with the widget name, follow the hierarchy

# What you have learned today - About the factory

---

- 1 State of your Widget is preserved

# What you have learned today - About the factory

---

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)

# What you have learned today - About the factory

---

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)
- 3 The fluent interface is taken care of if no value is returned

# What you have learned today - About the factory

---

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)
- 3 The fluent interface is taken care of if no value is returned
- 4 User options and defaults are automatically merged

# What you have learned today - About the factory

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)
- 3 The fluent interface is taken care of if no value is returned
- 4 User options and defaults are automatically merged
- 5 `_create` is the right place to manifest your widget in the DOM

# What you have learned today - About the factory

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)
- 3 The fluent interface is taken care of if no value is returned
- 4 User options and defaults are automatically merged
- 5 `_create` is the right place to manifest your widget in the DOM
- 6 `_init` is the right place to initialize the widgets state

# What you have learned today - About the factory

- 1 State of your Widget is preserved
- 2 Properties are always private, methods can be (underscore)
- 3 The fluent interface is taken care of if no value is returned
- 4 User options and defaults are automatically merged
- 5 `_create` is the right place to manifest your widget in the DOM
- 6 `_init` is the right place to initialize the widgets state
- 7 ThemeRoller can be used in conjunction with custom widgets

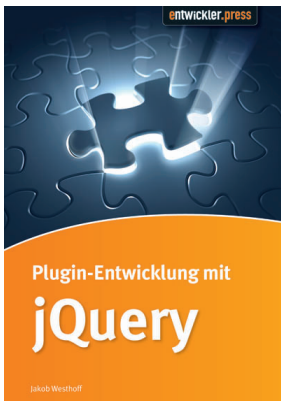
- Widgets

# jQuery-UI plugintypes

- Widgets
- Behaviors
- Effects
- Normal jQuery plugintypes
  - Set methods (`jQuery.fn`)
  - Utility functions (`jQuery`)
  - Special event handler
  - CSS-Selectors
  - Easing functions (Animation)
  - ...

# Want to learn more?

- Buy "Pluginentwicklung mit jQuery" (german)
- <http://entwicklerpress.de/jquery>



Questions, comments or annotations?

Slides: <http://westhoffswelt.de/portfolio.htm>

Contact: Jakob Westhoff <jakob@php.net>

Follow Me: @jakobwesthoff