

RocketCake Help

Overview



Welcome to the RocketCake documentation

RocketCake is an editor for designing responsive websites with no HTML knowledge, no programming is necessary. Just click together your website. A good place to start on how to use RocketCake is here:

Tutorials:

- [How to create a Responsive Website](#)
- [How to create a Contact Form](#)
- [How to use Master Pages](#)
- [How to use BreakPoints](#)
- [How to password protect a website](#)

Tutorials

Tutorials:

[How to create a responsive website](#)

[How to create a contact form](#)

[How to use Master Pages](#)

[How to use BreakPoints](#)

Creating a responsive website

The following page will show you step by step how to create a simple website using RocketCake.

Download and install RocketCake

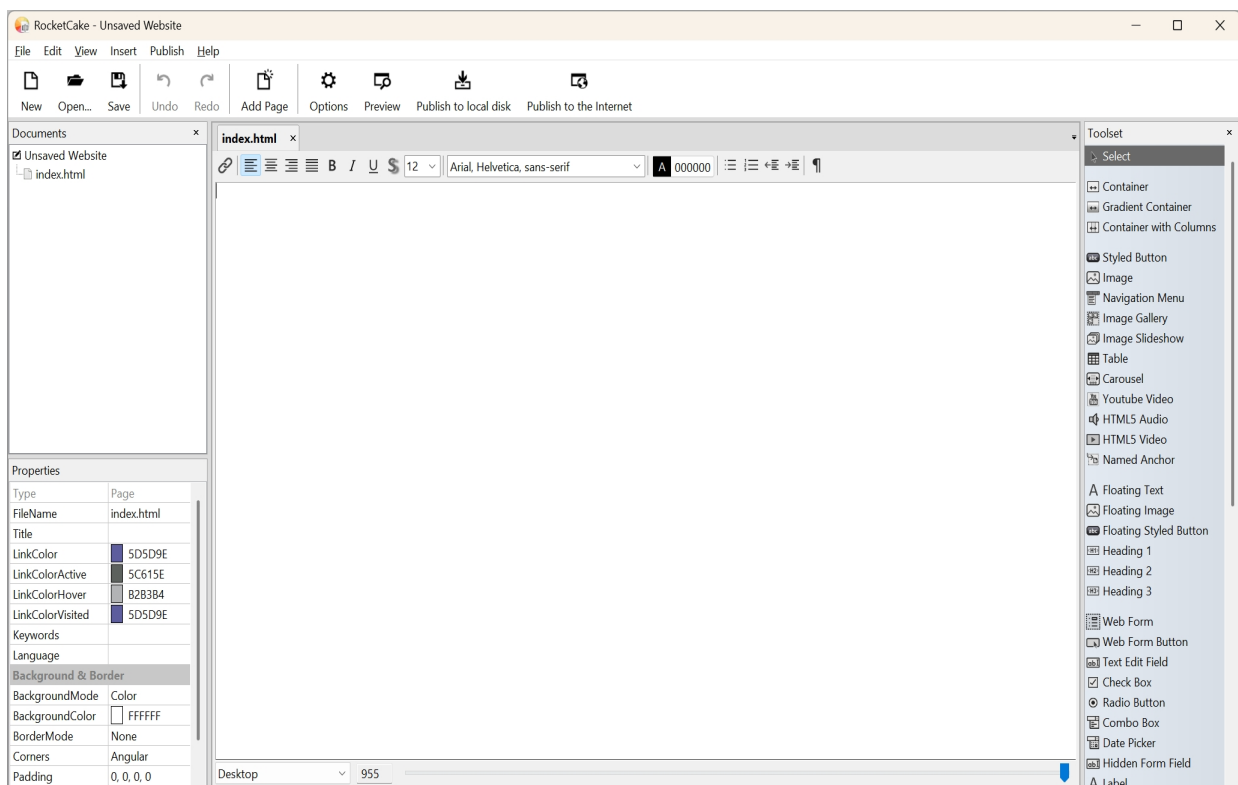
If you haven't done this yet, you need to download and install RocketCake. You can download it from here: <http://www.ambiera.com/rocketcake/>

What is a responsive Website?

"Responsive" is just a fancy word for "adjusting to the screen size". Since people are viewing websites with different devices - mobiles, tablets, PCs, notebooks - it has become very important for a website to be easily readable on all the different screen sizes: A responsive website works automatically nicely on the tiny 320x600 screen of a small smartphone, but also on a full screen 2048x1024 browser of a desktop PC. This is done for example by adjusting font sizes, rearranging and hiding elements, and making it easily scrollable on mobiles. All this is easily doable using RocketCake.

Create a new empty website

After you started the RocketCake program, it will ask you to create a new website from a template. Choose the first entry 'Empty page' to start with a new, empty website. The editor will now look like this:

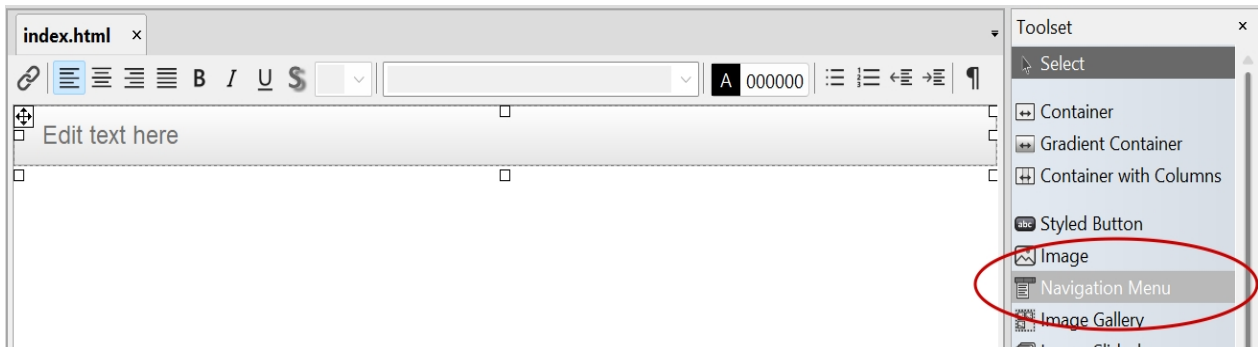


Take a look at the 'Properties' window on the left. Here, you can quickly change the appearance of the page (or whatever element currently is selected). You can enter a Title for your page here (this is the

text which usually appears for example in google if it displays your page as a search result) or the default colors for your links. Also, you can set a background color, gradient, or image if you like.

Creating a responsive website

To start, add a [Navigation Menu](#) to the page. Select the *Navigation Menu* element in the toolset on the right, and click into the page. This creates a menu element:



You can click it and directly type text into it in order to create some menu items. Do this, and create the menu items named for example "Company" and "About". Also, you can change the background color, to make your website look fancier, if you like. By default, the menu will have a width of 100%, spanning the full width of the page. This is quite useful, because it will then adjust automatically to different screen sizes.

Try it out: See the horizontal slider on the bottom of the page? Move it to the left and back again: With it, you can preview how the page will look like on devices with different screen widths.

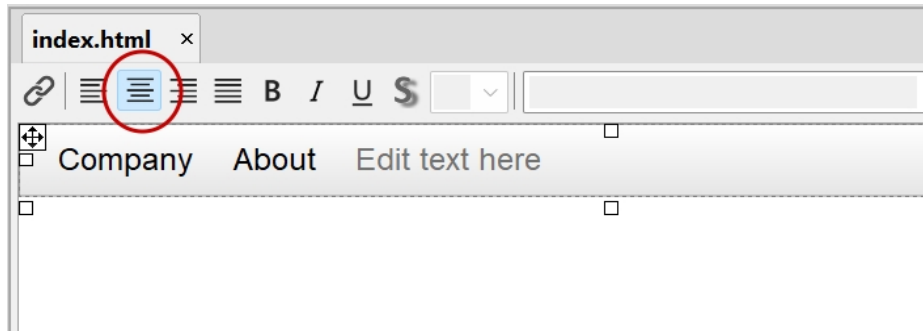


Put the slider back to the right, and we adjust the layout of the menu a bit: First, we set the menu to have a maximum width of 800. That way, the website will look nicer on huge resolutions.

While the menu is selected, in the property window on the left, search for "MaxWidth" entry, and change it "800", so it won't get wider than 800 pixels:

Properties	
Size	100%, 38
MinWidth	none
MaxWidth	800
VerticalAlignment	Bottom
Float	None
Margin	0, 0, 0, 0
Hover Colors	
UseHoverStyles	<input checked="" type="checkbox"/>
HoverBackgroundC	282828

Also, we want the menu to be centered in the container, so click on the "Center Text" button while the menu is still selected:



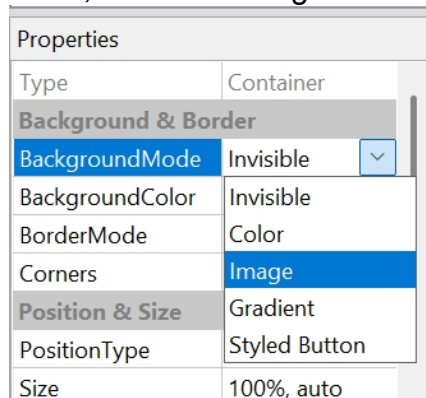
Add a logo image to your page

Great! Next, we want an image directly below the menu. We could use the [Image](#) element for this, but because we want to add some text on top of the image later, we use a [Container](#) instead and simply use its background-image property. So:

Select the *Container* element in the toolset on the right, and click into the page somewhere below the menu you created.



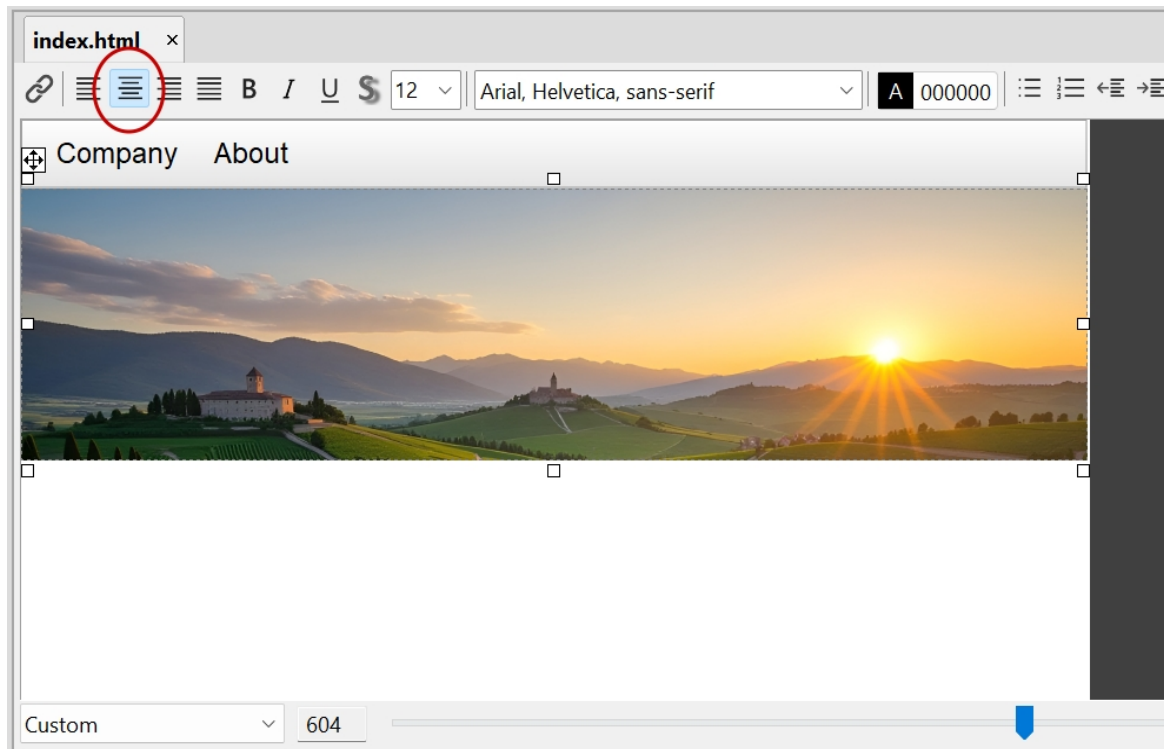
Drag the white square of its lower border a bit down to make it bigger. Then, in the property window on the left, find the "BackgroundMode" entry, and set it to "Image".



A new entry "BackgroundImage" will appear, in which you can select some image file from your disk.

As you did with the menu, search for "MaxWidth" entry, and change it "800". Also, click the 'Center Text' button while the container is selected.

The result should look like this (depending on the image you selected):



Add a website body

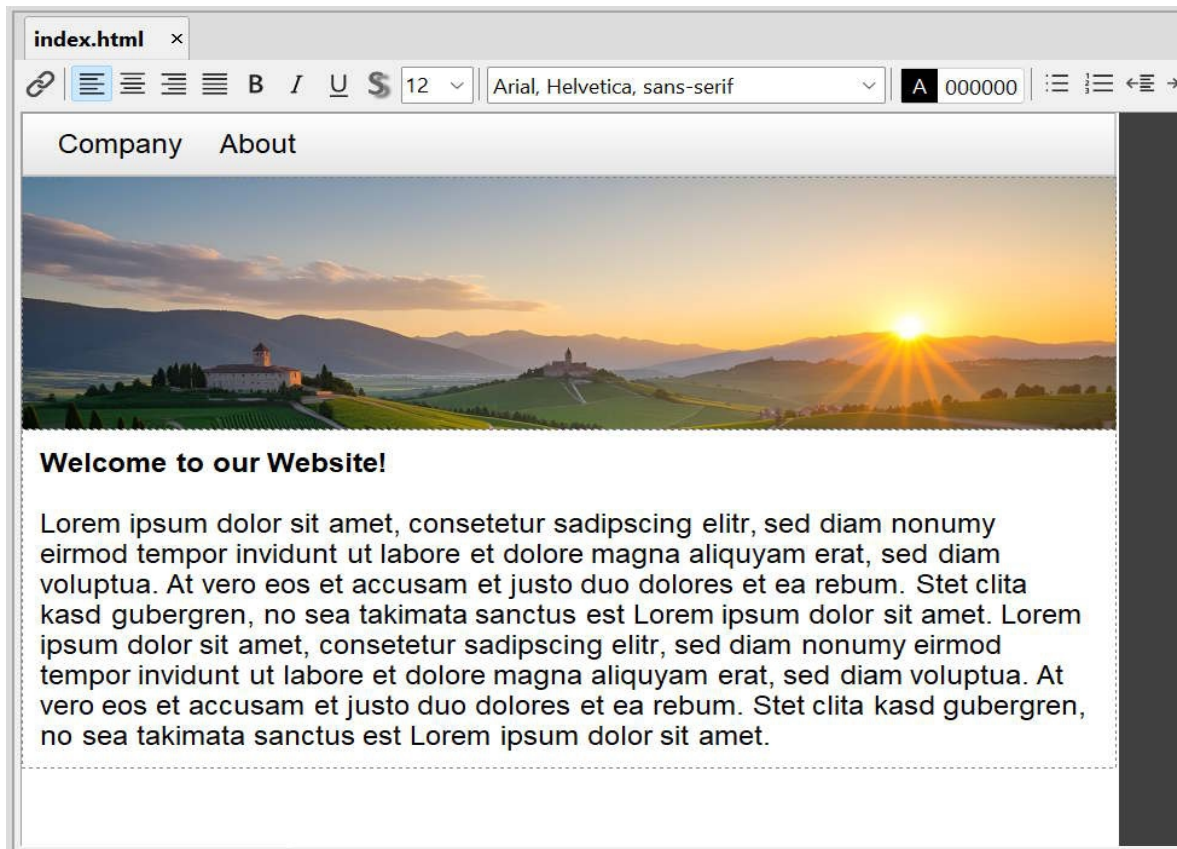
Now we have a menu, a nice looking image as logo, but we still need to add some real content, like text describing your website. In order to do this, we again add a [Container](#):

Select the *Container* element in the toolset on the right, and click into the page, somewhere below the image you added last time.

In order to make it look the same size as the menu we added on top, do the same as we did with the menu: While the new Container is still selected, in the property window on the left, search for "MaxWidth" entry, and change it "800", so it won't get wider than 800 pixels.

Also, click the 'Center Text' button while the container is selected.

Then, click into the middle of the container, and start typing some text. You can format the text anyhow you like:



Insert an image

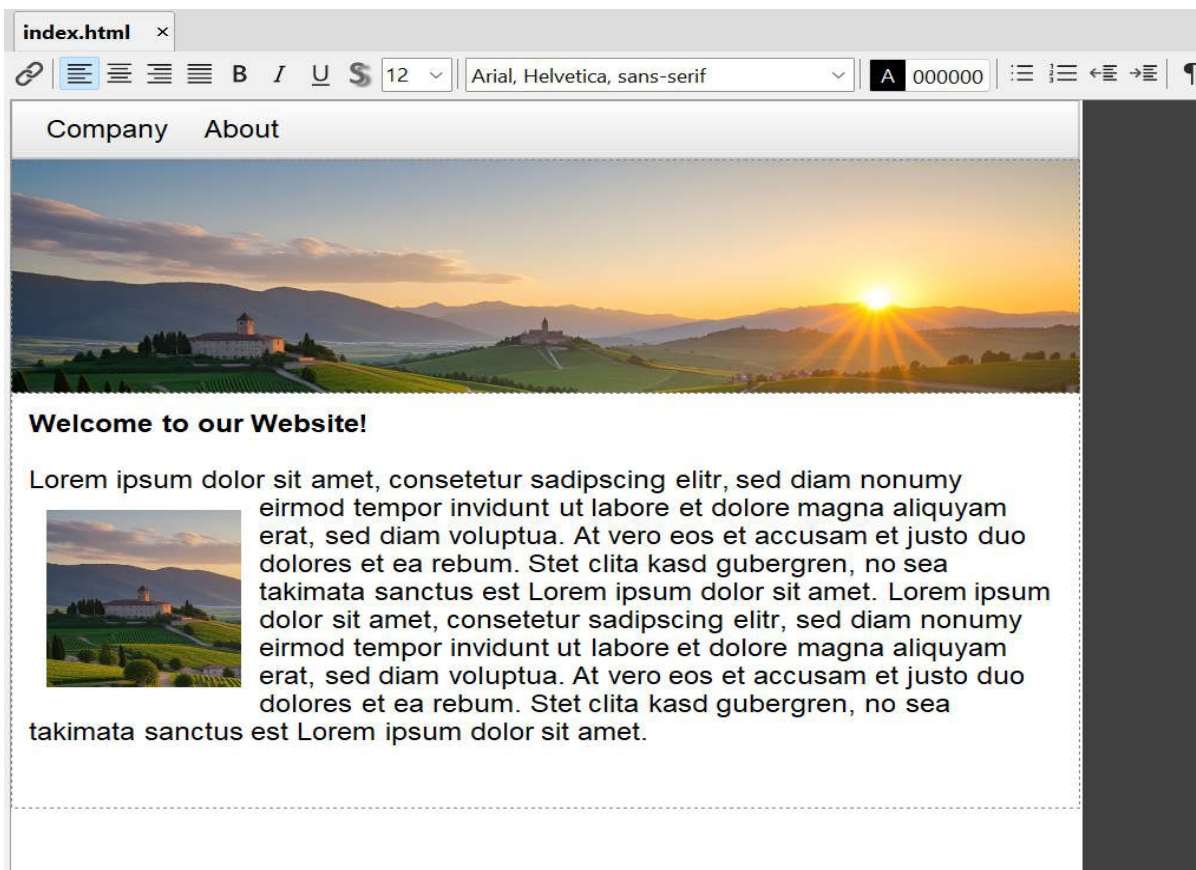
To make the website text look fancier, you can add an image into the text easily:

Select the [Image](#) element in the toolset, and click into the text, where the image should appear. Resize the image to fit your needs.

Right-Click the image, and in the menu, select "Text Float -> Left", in order to make the text float around the image.

When the image is selected, in the Property Window, search the entry "Margin", and change it from "0,0,0,0" to "10, 10, 10, 10". This will create a small margin of the text around the image, which then looks much nicer.

The result will look like this:



In the same way like how you added the image into the text, you can also add a Container directly into the text, for creating for example a box for news, and similar.

Your website is already responsive. You can preview it in any browser or in RocketCake itself, and resize it there, and see how it will adjust itself automatically.

Add other pages

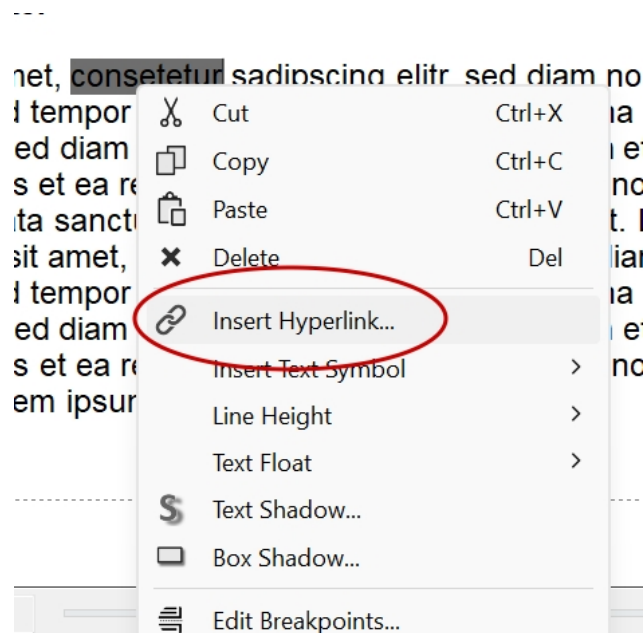
If you are finished with that page of your website, you might want to create a second page. Take a look at the top left 'Documents' window, where your only, initially named page 'index.html' is shown. Just right-click on the root element (probably named 'unsaved website' if you haven't saved it yet), and choose 'Add Page'. Alternatively, you can also use the menu command 'Insert -> Add to Project -> Add Page'.



A new page will open, which you can again fill with content. You can edit its name in the 'Properties' window.

Add hyperlinks

To add links between your pages or to other websites on the internet, you can create [Hyperlinks](#). To do this, mark the part of the text you want to be the hyperlink, right-click on it and select 'Insert Hyperlink...!'



Alternatively, you can also use the hyperlink icon in the toolbar of the editor. This also works for images and styled buttons.

A dialog will now open where you can enter the URL of the hyperlink. You can also choose 'Page in this project' as Link type and then select another page in this website.

If you are creating a text link, there will also be a 'Style' section in that dialog. Here you can define and reuse global named styles for your links, if you want more than one or some special styles. Defining different hover colors, disabling underlined links and more is possible here.

Tips for improving the website

Of course, the website isn't finished yet. Here are some tips on how to improve it:

If you want to add some text on top of the logo image, select the "Floating Text" element and click onto the image. That's it.

On small screens, the menu will automatically collapse to a smaller 'mobile' menu, which you can also change in the editor, if you make the screen size a bit smaller using the slider on the bottom. This behavior can be influenced in the property window.

For adjusting an element dynamically based on a smaller screen width, just right-click that element, and select "Edit Breakpoints". This will open the breakpoint editor, where you can easily specify rules in order to resize, hide or adjust elements based on the screen size.

Saving and Previewing the Website

To save your website, use the menu command 'File -> Save', so you can continue your work later on this page. You can also preview your website in your browser by clicking 'Publish -> Preview', or simply pressing the shortcut key F5.

Publish the website

Once you are finished with your website, you might want to publish it to a web server, so that other people can read it. You can simply use the "Publish -> Publish to the Internet" command for this, and enter the user name of your FTP server, where you want to upload the page.

Alternatively, you can use the command "Publish -> Publish to local disk". A dialog will appear to select a target directory. When you press OK, all HTML and image files will be generated on your disk, in the directory you selected. You now only need to upload these to your webserver or FTP server. For this you can use any FTP program. Ambiera recommends the free FTP client 'Filezilla'

(<http://filezilla-project.org/>) or WinSCP (<http://winscp.net/>).

How to create a contact form

Creating responsive websites with the help of RocketCake is quite simple. But adding more complex elements to a website, like PHP code might be a bit of a challenge if you don't know how to program PHP. This tutorial shows you how to create a mail contact form which sends mails by itself by using PHP.

Step 1: Get RocketCake and create a basic website

If you haven't done this yet, you need to download and install RocketCake. You can download it from here: <http://www.ambiera.com/rocketcake/index.html>

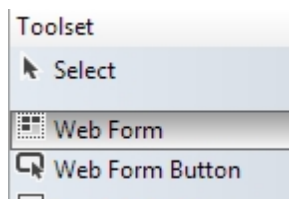
Step 2: Create the contact page

Once you have your website ready, a contact page still is missing, which we are going to add now. Click on the 'Add page' button in the toolbar, or choose the menu 'Insert' -> 'Add Page'.



The newly created page will have the name 'UntitledWebpage.html' or similar, so rename it to something which fits more, for example 'contact.php'. Notice that the new name needs to end with '.php' instead of '.html', otherwise your contact form won't work. You can change the name of the page in the 'Properties' window on the left, there should be an entry 'Filename' for this.

Now, design a contact form in that page. Use the webform element to create an area where your form will be. It is in the toolset, after you clicked the button 'more':



And use the elements 'text edit field' and 'Web Form button' to create a form which looks about like this:

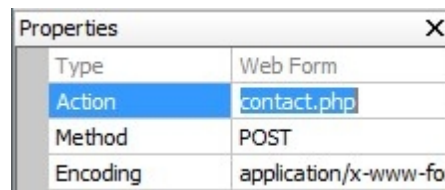
A screenshot of a contact form layout. The form is enclosed in a dashed border with small square handles at the corners. It contains the following elements: a label 'Email adress:' followed by a text input field; a label 'Text:' followed by a larger text area; and a 'Send' button at the bottom right.

(note: I used "Text Align -> Right" on the text fields and the button to make them align nicely with the text labels, but you can also use a table or similar to move your form to look nicely)

Notice that all these elements really need to be inside the 'Web Form' element, otherwise this won't work.

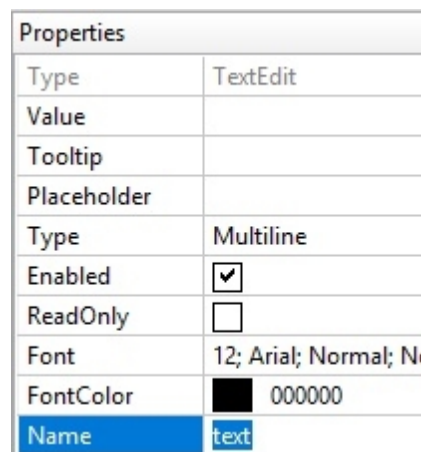
Note: Since RocketCake 5.2, you can just download an existing contact form component. To do this, goto to the component download page, download and double-click it, and place the component on your page. But if you want to manually create a contact form, do it like described in the following:

Now, we only need to set some of the properties of the created elements in order to make them work. Click the background of the form, the 'Web Form' element, and change the 'Action' property of it in the property window to the name of the contact page you created, to 'contact.php' in our example. Also, ensure the 'Method' is set to 'POST' and the 'Encoding' is set to 'application/x-www-form-urlencoded':



Properties	
Type	Web Form
Action	contact.php
Method	POST
Encoding	application/x-www-form-urlencoded

Next, set the names of the text fields to something describing their content. For example 'email' and 'text':



Properties	
Type	TextEdit
Value	
Tooltip	
Placeholder	
Type	Multiline
Enabled	<input checked="" type="checkbox"/>
ReadOnly	<input type="checkbox"/>
Font	12; Arial; Normal; N
FontColor	000000
Name	text

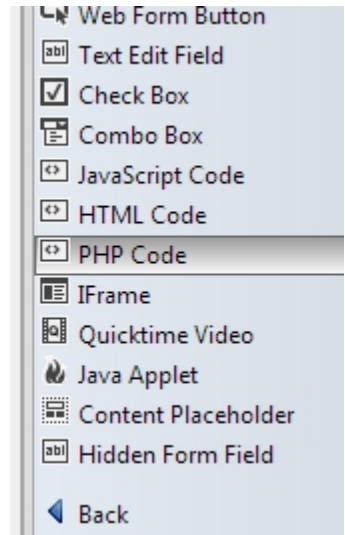
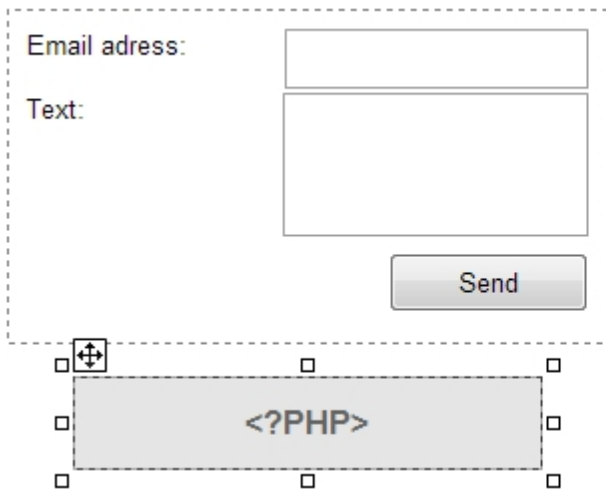
Also, as you can see above, you might want to select the 'multiline' option for the bigger text, so that your visitors are able to enter more than just one line of text.

For the 'Send' button, you can set the 'label' property to any text you want, but be sure that the 'ButtonType' is set to 'Submit'.

Step 3: Create the PHP code

So, that's it, the contact page is now designed. The only thing now missing is some PHP code which should be triggered once the user clicks the 'Send Button'. In our case, we want the text simply to be emailed to us.

Now, simply create a PHP code element and place it somewhere you your page:

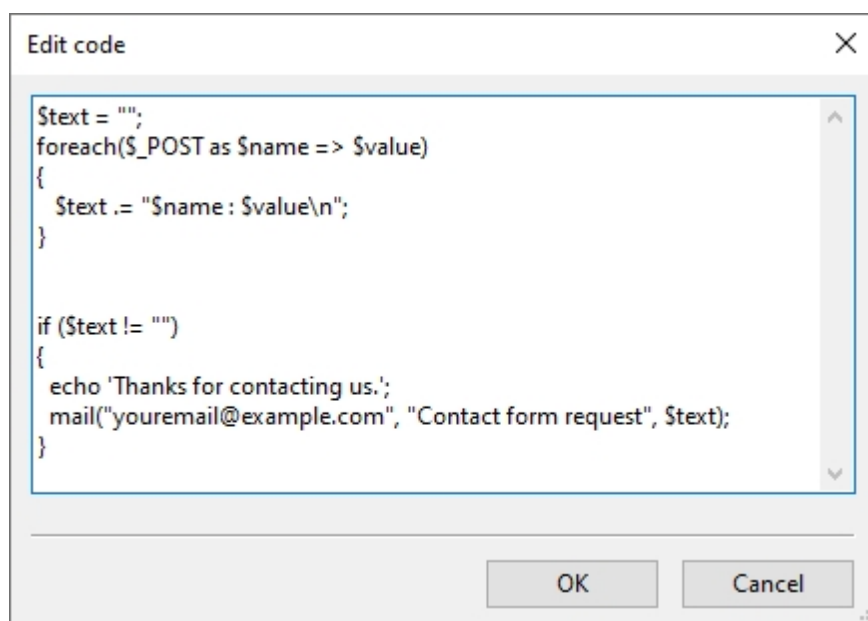


Of course you can change the background color and border mode of it in order to make it look more like you want it to be, in this example, we simply set the background and border to 'none'. Double-Click the new PHP element or right-click it and choose 'Edit Code...', and paste the following code into the dialog:

```
$text = "";
foreach($_POST as $name => $value)
{
    $text .= "$name : $value\n";
}

if ($text != "")
{
    echo 'Thanks for contacting us.';
    mail("youremail@example.com", "Contact form request", $text);
}
```

This is some PHP code which simply sends the text per mail once the visitor clicks the 'send' button. Notice, in the end of the code above, there is the text **youremail@example.com**. You need to replace this email address with your email address, so that the email then is sent to you.



Basically, that's it now, your contact form is ready now.

Step 4: Test the form

If you preview your contact form now, 'Publish -> Preview' you will notice that the PHP code which you just wrote will appear on the page. Don't worry about that, this only happens in the preview and is designed to work like this. In order to make the contact form run as you want it to be, you need to run it from a webserver which supports PHP. You probably already have a webserver with PHP support, otherwise you wouldn't have gone through this tutorial. Simply upload your website to this webserver now, and if you did everything as described on this tutorial, your contact form should then work.

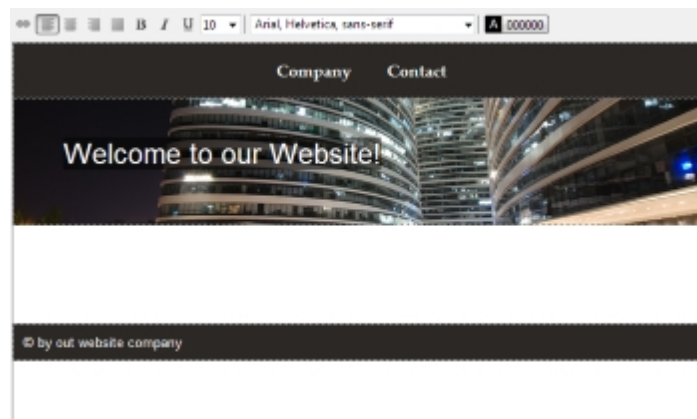
If you have any questions or comments on this, don't hesitate to visit the RocketCake forum.

How to use Master Pages

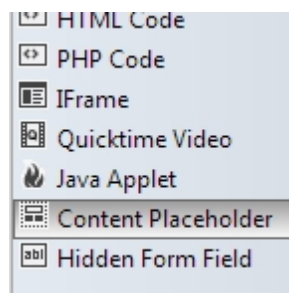
RocketCake has a built-in feature named *Master Pages*, a simple mechanism to share parts of the website between different pages. If you have a lot of web pages with repeating design, it might be easier to just design the layout once and reuse it on all pages which have actual content. This section will show how to do this.

Getting started

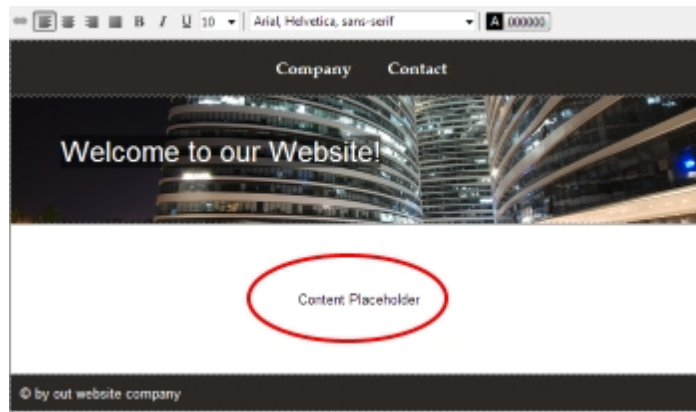
First, create some design layout page for your website. Name it for example 'masterpage.html' to be sure you don't confuse it with a real page. As example, create a menu and some caption on it, like this:



This will now be the base for all pages, and we only will change the text content on every page. To let RocketCake know which part of the website should change, add a 'Content Placeholder' on the page. You can find it in the menu item 'Insert -> More -> Content Placeholder' or on the second page of the toolset on the right:



Place it where your website content should be displayed:



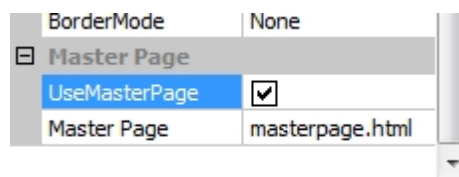
Note: Without the placeholder, Master Pages won't work.

Add content

Now, we only need to add one or two pages with content to use this master page. Add a new, empty page to the project, and fill it with some text and content:



The only thing missing is now to tell RocketCake to use the Master Page for this page. In the Properties window, go to the 'Master Page' option, and check it. A new entry will appear where you will be able to select a page as master page. Select the masterpage.html we created before:



And basically, that's it. If you now publish or preview the page, it should appear as part of the master page, exactly where you placed the content placeholder:



You can create as many pages as you want which all can use the Master Page. In this way, you only have one page containing the layout, which you only need to edit once.

How to use Breakpoints

In order to create dynamic responsive websites, RocketCake has a built-in editor for creating and modifying breakpoints.

What is a BreakPoint?

A breakpoint is simply a size of the screen which triggers certain elements on your website to change: They can appear, go away, change their font size or similar. This is very useful for controlling exactly how the website looks on different devices.

For example: When your website is viewed from a computer with a big screen, you show a lot of images. But those images are not very informative and are only in the way when viewed from a device with a very small screen. So it would be nice to hide those images when the screen is small. With a breakpoint, you can say for example: "If the screen is smaller than 480 pixels, then don't show these images". That's it.

How to create a BreakPoint

Using RocketCake, creating a Breakpoint is very easy. As example, we take a website, and make an image disappear if the website is shown on a device with a screen width smaller than 320 pixels. We take an example website like this here:

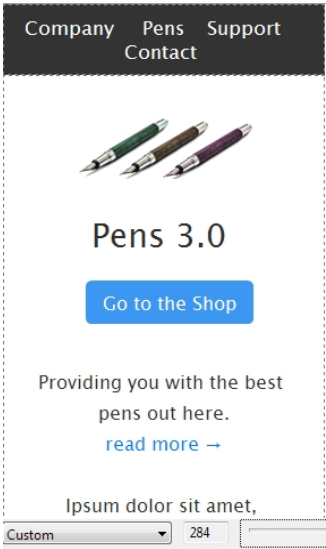


Pens 3.0

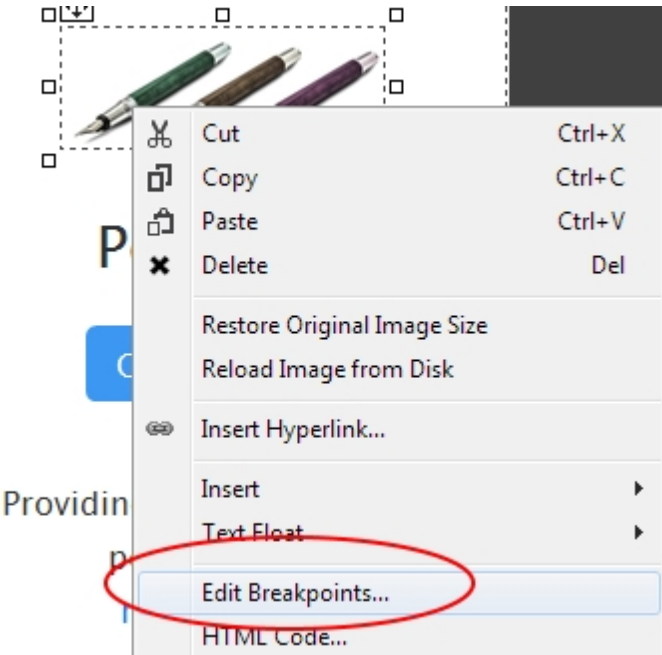
Go to the Shop

Providing you with the best pens out here.
[read more →](#)

It looks ok on a normal screen size. When the size is reduced to a smaller width, like here:



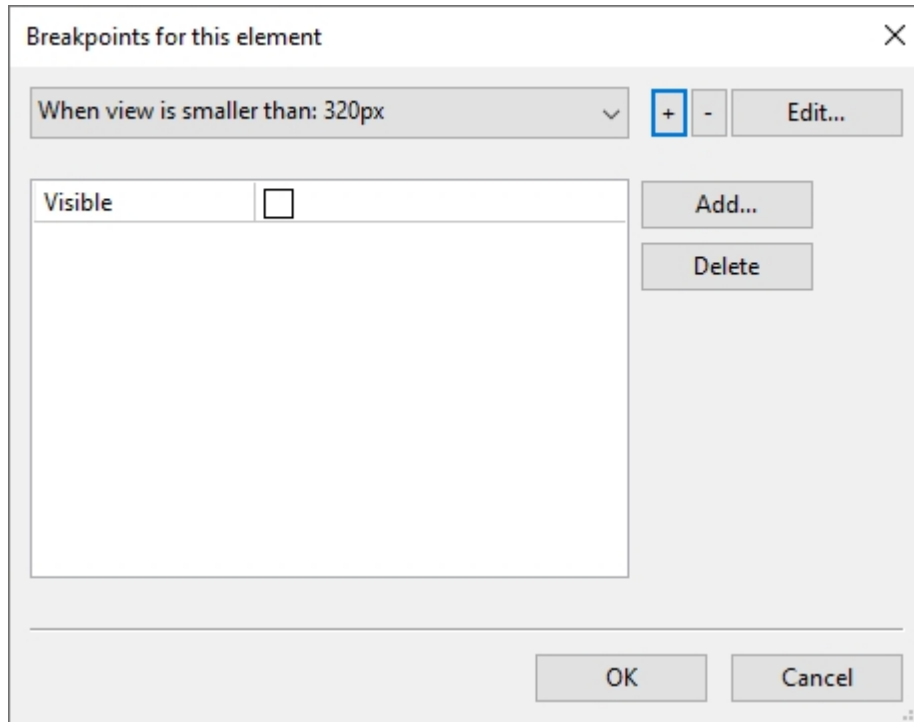
It would be nice to hide the image with the pens on it, to have more space. To do this, just right-click on the image with the pens, and select "Edit Breakpoints".



Then the breakpoint editor will appear. Do the following:

- Add a new breakpoint by clicking the '+' sign.
- Enter '320' as width
- Click the "Add..." button to add a new rule.
- Select "Visible"
- Uncheck the checkbox, to make the element invisible.

The result should look like this:



If you now view the website with a screen width smaller than 320 pixels, the image will not be visible. You can also see this in RocketCake. If you move the slider on the bottom of the page to a bigger size, the image will appear again:

Company Pens Support
Contact

Pens 3.0

[Go to the Shop](#)

Providing you with the best
pens out here.
[read more →](#)

Ipsum dolor sit amet,

You can have as many breakpoints per element as you like. You can change the font size, position,

width, and other properties using that.

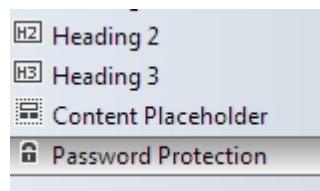
If you want to use Breakpoints in your website, keep in mind that it works like this: Design the website with the biggest supported resolution in mind. With the breakpoints, you can then make changes to your website for smaller screen sizes. However, if you don't want to use Breakpoints at all, this is fine as well. If it works without, then this is even better.

How password protect a website

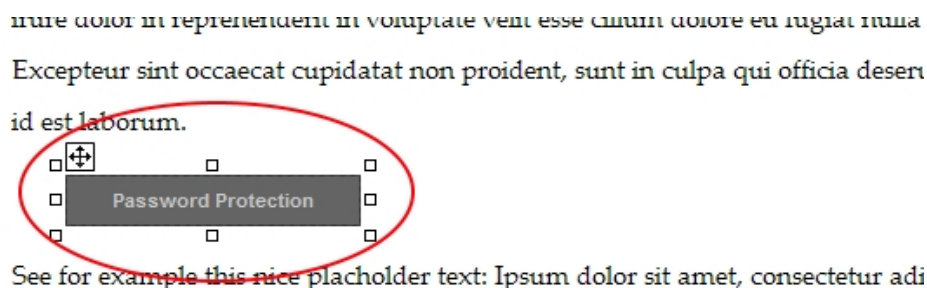
In order to prevent everyone on the internet have access to specific pages of your website, you can password protect them. Using RocketCake as editor, this is quite easy.

Password Protecting a Web Page

In order to password protect a web page, just select the "Password Protection" element in the right toolbar in RocketCake and add it somewhere onto the page:

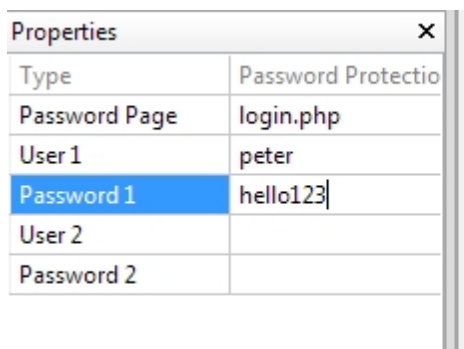


You can place it anywhere on the page, wherever you like, it is only visible in the editor and won't appear in the final website:



After you placed that element, RocketCake will likely tell you that it changed the extension of the website's name to .php (from for example .html) and that it created a login page for you.

When you select that "Password Protection" element, you will see that you can enter a number of users and passwords to be used. Do this now:



Now when you upload your website to a server, a user will need to enter the correct user name and password in order to see that page. You can put that password protection element onto any page you like. Passwords and user names are shared between these pages, so you don't need to enter the users/passwords multiple times.

That's it, now you know how to protect your website!

Useful things to know about Password Protection

These are some useful things to know about the password protection:

You won't see the password protection working when previewing your website on your local browser. It will only work when uploaded to a server.

You can customize the login page as you like, it is a page (usually named login.php) in your project. It will only work on web servers which support PHP. But mostly every webserver does that today.

When you update the passwords or user names, don't forget to re-upload your website to the server.

How to log out a user

If you want a way to log a user out again, do this: Create a new page, and add a PHP Code element into that. As code, paste the following in there:

```
ini_set('session.use_trans_sid', false);
ini_set('url_rewriter.tags', '');
session_name('rocketcakelogin');
session_start();
session_unset();
session_destroy();
```

Now, if this page is loaded, the user will be logged out again. You can just link the page from a hyperlink with the text "log out" or similar, and place a nice text onto that page, like for example "you are now logged out". That's it.

Components

RocketCake supports a lot of Components to create Websites:

[Containers](#)

[Images](#)

[Styled Buttons](#)

[Named Anchor](#)

[Image Slideshow](#)

[Image Gallery](#)

[Navigation Menu](#)

[Layout Container](#)

[Flash](#)

[Java](#)

[WebForm](#)

Label

[HTML Code](#)

[PHP Code](#)

[JavaScript Code](#)

[Table](#)

[Floating Texts](#)

[Quicktime Video](#)

[Youtube Video](#)

[HTML5 Audio](#)

[IFrame](#)

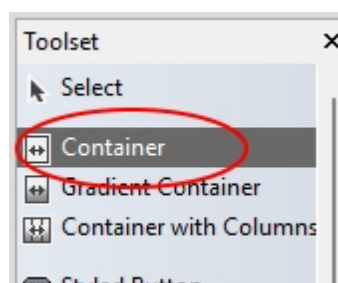
[Content Placeholder](#)

[Password Protection](#)

[PDF Document](#)

Container

The *Container* component is one of the most important components in RocketCake: it is used to organize the layout of the website. It can be filled with text, images, and any element, even other elements. Like most other elements, its can adjust its border, background and edges to fit your design.



By default, it has a width of 100%, but this can be changed if you like to. It doesn't have a specific height, this is always set to 'auto', so that it adjusts itself to the height of its content, which is important for responsive websites. However, when you drag the height in the editor, it will set its MinHeight to the value you dragged it to in the editor. So if your container doesn't get smaller when you think it should to, simply set the MinHeight to a smaller value, or even 0, to disable this feature.

By using the 'Image' entry of the Toolset or the menu command 'Insert -> Image', you can place images on the website. Once you placed your image, a file selection dialog will appear and ask you to select an image file on your disk. You can select any file of the format .jpg, .png, .gif and .bmp here.

In the 'Properties' window on the left, you can change the image file name which is shown, and other settings for its appearance, as for all other items as well.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image file cannot be loaded or the image is transparent.

Other supported properties are:

Alternate Text: Shown by most browsers before the image can be loaded

Tooltip: Text shown when the mouse cursor is above the image

Use Hover Styles: When checked, you can set a brightness (between 0 and 100) which the image should get when the mouse cursor hovers over it.

Hyperlinks

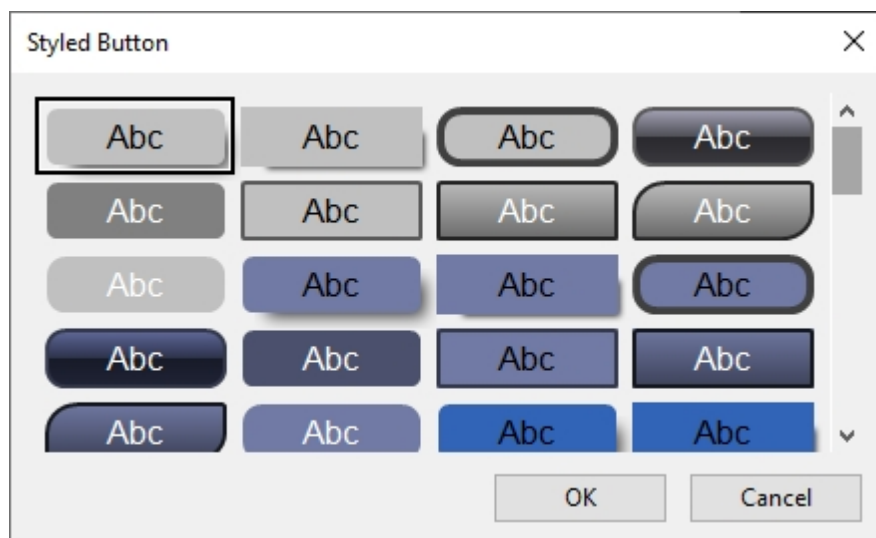
You can create a hyperlink for the image, so that the user is able to click on the image and open a new page. Simply select the image, right-click and select 'Insert Hyperlink'.

Text-Float

If you are using the image floating inside text, you can set it's text-float to make the text float around it more nicely: Just right-click the image, and use the command "Text Float -> Left" or "Text Float -> Right". If the text then is floating "to closely" around the image, change the "Margin" property of the image from the default '0,0,0,0' to something bigger, like '10, 10, 10, 10'.

Styled Button

Styled Buttons are elements in RocketCake which can be used to create a nice looking button without the need to use a graphics program to do this for you. Also, the buttons are usually created in the HTML without images if possible - using CSS style sheets alone - reducing the needed download size and speed of your website greatly.



Creating a Styled Button

By using the 'Styled Button' entry of the Toolset or the menu command 'Insert -> Styled Button', you can place buttons on the website.

You can customize the appearance of the shape using the property window.

Button with Text or Hyperlinks

You can click into a button and start typing. The text will be displayed centered in the shape, which is useful to create buttons. Additionally, it is possible to create a hyperlink for the button, so that the user is able to click on the button and for example open a new page. Simply select the shape, right->click and select 'Insert Hyperlink'.

If you want the button have a different color when the mouse cursor moves over it (which is desirable sometimes for hyperlinked shapes), simply select the 'UseHoverStyles' checkbox in the property window of that shape and choose some colors. Hover styles will not be shown in the editor, simply click 'Publish -> Preview' to test them out.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image file cannot be loaded or the image is transparent.

Other supported properties are:

BackgroundMode: How to fill the shape, either via Gradient, Color, Styled Button style or an Image file

GradientStyle: If the fillmode is 'Gradient', you can select the style of the gradient here. Supports 'horizontally' and 'vertically'

DarkeningFactor: When the BackgroundMode is 'Styled Button', it is possible to change the brightness of the color here. Choose any value between 0 and 100, 0 for very bright.

Transparency: A value between 0 for opaque and 100 for 100% transparent

UseHoverStyles: If you want the shape have a different color when the mouse cursor moves over it (which is desirable sometimes for hyperlinked shapes), simply select the 'UseHoverStyles' checkbox in the property window of that shape and choose some colors. Hover styles will not be shown in the editor, simply click 'Publish -> Preview' to test them out.

Named Anchor

Named Anchors are a kind of 'Bookmark' in the page which can be linked by hyperlinks.



If you create a [hyperlink](#), you can then select a named anchor as target for the hyperlink, and the page will scroll to the position of the named anchor when the link is clicked.

Creating a Named Anchor

By using the 'Named Anchor' entry of the Toolset or the menu command 'Insert -> Named Anchor', you can place named anchors on the website. They will only be visible in the RocketCake editor, not in the final Website at all.

In the 'Properties' window on the left, you can change name of the named anchor.

Slideshow

The Image SlideShow component gives you the possibility to create an interactive image slide show to your Website.



Creating a Slideshow

By using the 'Image Slideshow' entry of the Toolset or the menu command 'Insert -> Image Slideshow', you can place the slideshow on the website. Once you placed your image, you can select images for the slideshow in the property window. You can select any file of the format .jpg, .png, .gif and .bmp here.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image files cannot be loaded or the currently shown image is transparent. Other supported properties are:

TimeForAnImage: Time in milliseconds how long an image in the slideshow is shown before switching to the next image. 1000 is one second.

TimeForFading: Time in milliseconds how long an image in the slideshow will be faded out and the next is fading in. The default is 250 which is a quarter of one second.

Interactive Slideshows

To make the slideshow interactive, you can create links for selecting the next or the previous image. Create a Hyperlink (see [Hyperlinks](#) to see how to do this) and choose 'Invoke a special RocketCake action' as link type for this. A dialog will then pop up where you can select the action 'switch to next image in slideshow' or 'switch to previous image in slideshow' and the target slideshow element for that action.

Hyperlinks

Each image in the slide show can also have a link which opens a website when the image is clicked. For this, click the checkbox named "WithHyperlinks" in the property window when the slide show is selected. Then, the property window will show an entry named "ImageLink" for each image you selected, where you can write a URL like "http://www.example.com" or "about.html".

Fixed Aspect Ratio

Slideshows have the option "Fixed Aspect Ratio", it is checked by default. This means they automatically adjust their height based on the screen size and images inside them. Without this option enabled, their height is fixed and won't change automatically.

Image Gallery

The Image Gallery component gives you the possibility to create a gallery of small images, which show a bigger version of the image when you click on them.



Creating an Image Gallery

By using the 'Image Gallery' entry of the Toolset or the menu command 'Insert -> Image Gallery', you can place the gallery on the website. Once you placed your image, you can select images for the slideshow in the property window. You can select any file of the format .jpg, .png, .gif and .bmp here. By selecting the amount of columns in the property window and resizing the area of the gallery, you can influence how the images will be arranged.

Each image are then automatically resized to fit into the rows and columns of the other images and to keep its aspect ratio. Note that it is recommended to use images with a similar aspect ratio in order to make the array of images look more nicely.

Properties

This component supports several background modes (Color, Image, Gradient) for the images, changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image files cannot be loaded.

Other supported properties are:

Padding: Distance in pixels between each image

ColumnCount: Amount of columns used for showing the images.

Thumbnail: You can choose an aspect ratio for the small preview images here. The default is 16:9 which looks nice on most websites, but you can also choose something like 3:2 or 4:3, or square. Select 'original' to have each preview image a different size based on the actual size and format of the full size image.

OpenMode: This sets how the image will be opened when clicked on. By default, the image will open in a new layer on top of the website, but it is also possible to make it open in a new Tab or Window, in a window pop up or to replace any existing image on the current website with it.

UseHoverStyles: If you check this, it is possible to select another border color for the small images when the mouse cursor is over the image.

NavigationButtons: If you check this, the image gallery will show navigation buttons (left and right) when an image was opened, so that users can browse the gallery more easily. Also, this enables keyboard buttons (cursor left and right) to do this as well. You will see these buttons only in the browser, when clicking 'Preview', not directly in RocketCake.

ShowText: enables the possibility to write some text below each image. When this is checked, each entry for every image gets a "ImageText" entry and you can enter text there to be shown below the image

AlternateText: enables the possibility to write some "alt text" for each image. When this is checked, each entry for every image gets a "ImageAltText" entry and you can enter text there when then will be added as alt attribute, which provides alternative information for an image for SEO purposes.

Carousel

Carousels, also known as sliders, let you display text, graphics, images, by animated automatic or manual horizontal scrolling. It has built-in and configurable navigation buttons and page indicators.



Creating a Carousel

By using the 'Carousel' entry of the Toolset or the menu command 'Insert -> Carousel', you can place the carousel on the website. Then you can edit the content of each page for the carousel: Edit the text directly on the page, add buttons, images, hyperlinks, whatever you like. Select another page using the entry "Active Page" in the property window to edit other pages. You can select a background image for each page using the "BackgroundImage" property in the property window.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image files cannot be loaded or the currently shown image is transparent.

Other supported properties are:

PageCount: Enter a number how many pages there should be

Active Page: The currently selected page. Change this to the page you want to edit, and with which page the carousel should start on the website.

AnimationMode: How the transition between the pages should happen.

TimeForAnImage: Time in milliseconds how long an image or page is shown before switching to the next page. 1000 is one second.

TimeForFading: Time in milliseconds how long a page in the carousel will be faded out and the next is fading in. The default is 250 which is a quarter of one second.

ImageAdjusting: How the background image is adjusted to the size of the component. You can tell the image to fit in width, in height or to fill the full size of the component as best as possible ('cover').

FixedAspectRatio: When checked, keeps the component aspect ratio the same no matter how large it is (depending on the responsive size of the website)

There is also the option to show navigation Buttons (the buttons to the left and right) and page indicators (small circles on the bottom of the component). Use these properties for that:

NavButtons: Enables the navigation buttons

PageIndicator: Enables the page indicator controls.

Colors and appearance of both of these can be adjusted using the properties which appear when these are checked.

Interactive Carousel

If you don't want to use the integrated navigation buttons, you can use your own buttons with links for selecting the next or the previous image. Create a Hyperlink (see [Hyperlinks](#) to see how to do this) and choose 'Invoke a special RocketCake action' as link type for this. A dialog will then pop up where you can select the action 'switch to next image in slideshow' or 'switch to previous image in slideshow' and the target slideshow element for that action.

Navigation Menu

The Navigation Menu component gives you the possibility to create a drop down menu with links on your website. It looks like this:



Creating a Menu

By using the 'Navigation Menu' entry of the Toolset or the menu command 'Insert -> Navigation Menu', you can place the menu on the website. Once you placed your menu, you can click the 'Edit text here' text and enter new menu entries directly in the editor. It is possible to select a font, color and font size for each menu entry individually. If you want to change the font for multiple menu entries at once, simply select one menu entry, hold down the 'Shift' key on your keyboard and click other menu entries to select them as well. Then, you can choose a new font as usual.

Special Entries

The menu supports horizontal lines as menu entries. Simply set the text of a sub menu to "-" (=a minus symbol), and if you click 'preview' to view your website in the browser, this menu entry will appear as a horizontal line.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image files cannot be loaded or the currently shown image is transparent. If you want even different backgrounds, simply set the background and border to transparent and invisible and place any other element behind the menu bar.

Other supported properties are:

Padding: Space in pixels between menu entries

SubMenuSpacing: Vertical space in pixels between sub menu entries

Pane Background color: Color for the background of the dropdown menu pane.

Pane Border color: Color for the border of the dropdown menu pane.

Use CSS Shadow: Draws a shadow around the dropdown menu pane. This looks very nice, but only is visible on newer browsers which support this feature.

Use Hover Styles: If checked, the main menu entries in the horizontal bar show hover colors when the mouse moves over them.

Hover Background Color: Background color of a main menu entry when the mouse is over it.

Hover Text Color: Text color of a main menu entry when the mouse is over it.

Hover Pane Background Color: Background color of a sub menu entry when the mouse is over it.

Hover Pane Text Color: Text color of a sub menu entry when the mouse is over it.

Mobile Menu: See the section below

MenuAlignment: Where the top menus are aligned to: Left, Right or Center. Make the menu a full width of 100% and set this to align right, then your menu will always look nicely aligned to the right, no matter the screen size.

MenuAnimations: Enable this to select from a few animation options below which are used when the menu is opened in the website.

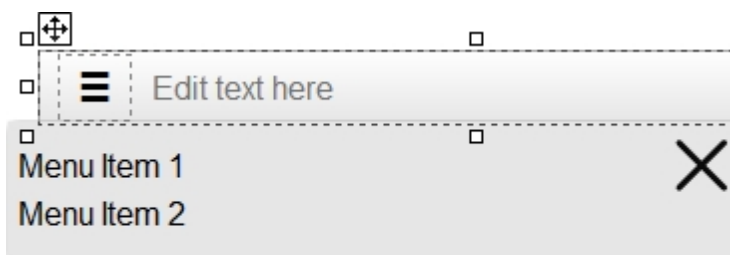
Advanced Settings: Click the "..." button here for more options, like:

- OpenCloseWhenClicked: You can select if the menu will close when clicked on it again, or open via mouse hovering on desktop or not. Some people find it more natural with the second mode, so that you can easily close the menu again by clicking on the top menu.
- GenerateAriaLabels: When selected, this will generate aria-label attributes from the text in the menu for each menu item, which is useful for accessibility.
- GenerateUsefulTabIndex: By default, this is on and will generate tabIndex attributes for the menu, so that you can cycle through all menu items easily using the Tab key. However, some accessibility auditing tools prefer every tabIndex attribute to be set to "0" instead, so you can disable this feature and all attributes will be set to 0 then.
- CloseWhenMouseOut: When the mouse cursor moves out of the menu and this setting is enabled, the menu will automatically close again. Alternatively, the menu will only close when the user clicks somewhere outside of the menu.

Mobile Menu

Each navigation menu comes with a second menu, which is shown on smaller screens, if the property "MobileMenu" is set to 'Automatically Generated' or 'Manually Filled'. You can see this mobile menu in Rocketcake if you drag the view slider on the bottom of the page to a small area, or if you select a mobile device on the bottom combo box, like for example "iPhone 7". By default, this menu is generated from your 'Desktop' menu, but when set to the 'Manually Filled' mode, you can edit this mobile menu just like the normal menu.

The advantage of this second menu is that you can arrange the menu items in a more compact way in that menu, so that everything fits on the screen also on mobile devices.



This mobile menu is automatically generated.
Change this setting in the property window.

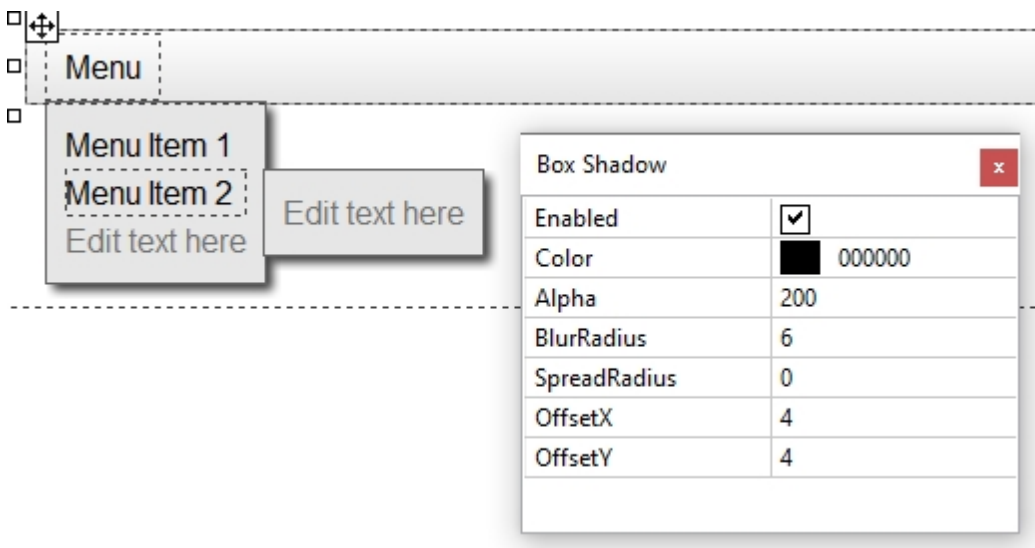
In the advanced settings of the menu, you can specify if you want the menu on mobile devices to appear fullscreen like above - which is friendlier for touch based mobile devices or more compact like this:



This mobile menu is automatically generated. Change this setting in the property window.

Shadow For the Menu

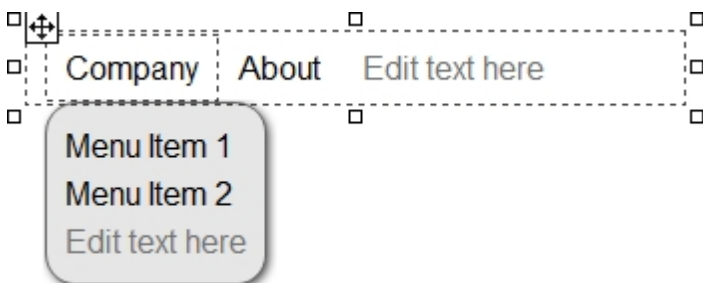
You can make the dropdown menu have a shadow. For that, just open the menu, right-click onto it, and select "Box Shadow". There you can enable and edit shadows for the dropdown menu.



Round Edges for Navigation Panes

It's possible to create round navigation menu panels. For that, simply set "edges" of your navigation menu to "round", and check the option "RoundEdgesAlsoForPanes". Then your navigation menus are also round.

If you don't want the top menu bar to have round edges as well, simply make its background transparent and place it into a container of which you then apply the background you want to have for your top menu.

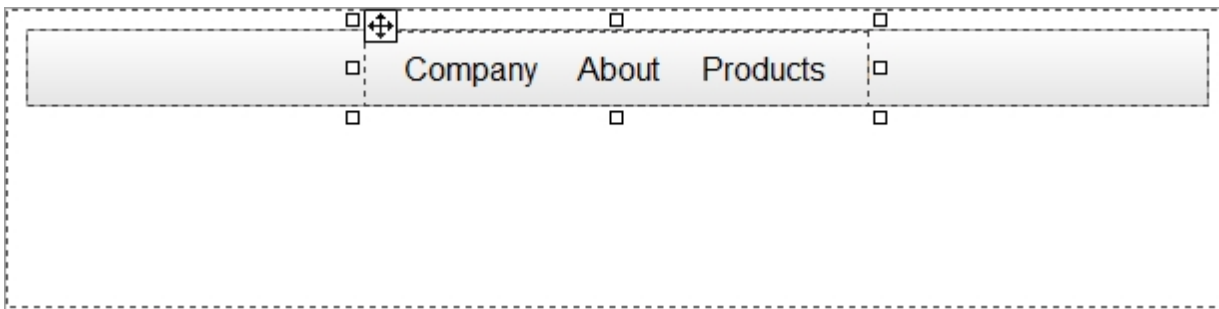


Centering the Menu

The menu itself is always right aligned. But in some cases, you might want it to be centered.



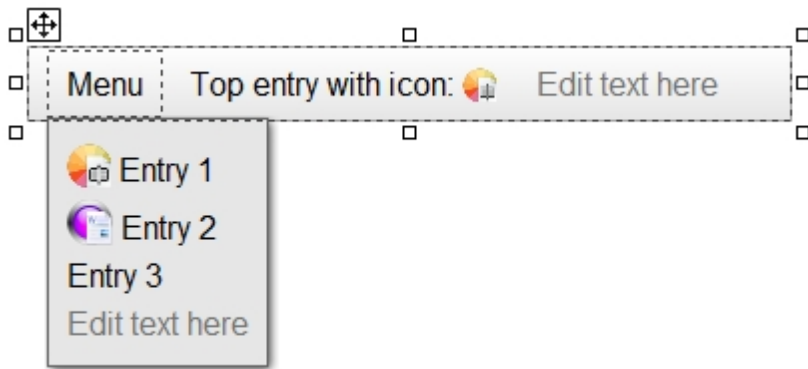
To do this, you can simply set the property "MenuAlignment" to "Center":



Done: You now have a centered menu.

Icons in the Menu

You use icons in the menu and the menu entries. Simply drag an image element into the menu at the place you want it to have, or alternatively create an image somewhere on the page, and then copy and paste it into the right position in the menu:



Be sure to have the icons at about the same size as your text so that it looks correctly in the website.

Flash

The Flash component allows to include an Adobe Flash movie clip into the website. Flash apps are an old technology and not used that much anymore. Most major browsers have stopped supporting Flash today.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Other supported Properties are:

FlashFilename: The .swf file to be shown

Quality: Flash quality setting

AllowScriptAccess: Flash script access setting

Play: If the flash movie should start playing when the document is loaded

Loop: If the flash movie should play looped

AllowFullScreen: If the flash movie is allowed to switch to fullscreen mode

WindowMode: If the background of the movie is transparent or opaque

FlashVars: The encoded list of flash vars

Java

The Java component allows to include Java applets into the website.

Java Applets are an old technology and not used that much anymore. Most major browsers have stopped supporting Java Applets today.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Other supported Properties are:

AppletFilename: The .jar or .class filename of the applet

NoJavaSupportText: Text displayed if the browser doesn't support Java

Parameter/Value: Values and Parameters for the Applet

WebForm

You can create contact, login, email and other forms with a webform.



A Webform is the container for WebForm elements such as Buttons, TextEditFields, CheckBoxes, ComboBoxes and [Labels](#). All these elements must be inside a WebForm in order to work.

Make the Form work

There is [a tutorial describing how to use a Web Form](#) in detail. To use the form, edit the properties of it:

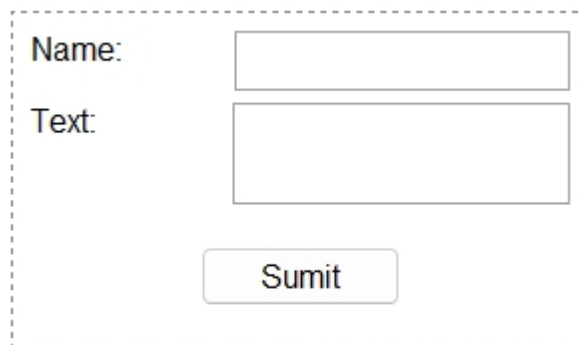
Action: This contains the action of the form to do. Usually, this contains the URL of the webpage, PHP script or CGI which will receive the data in the form. You can also enter a mailto: action with an email address, for example: *mailto:someone@example.com*. When the submit button is clicked, this will then open the email client of the user and try to send a mail, but it is not very reliable to work on all systems.

Method: select Post or Get. This is the mode of transportation of the data sent. For huge data, Post

is usually preferred.

Encoding: usually this should be set to `text/plain` for sending text.

Add a text field and a button with the `buttonType` set to 'submit' to make the form work. You usually also have to write the PHP, CGI or whatever backend for the server of your form to process the form data, that is sending it per mail or entering it into a database.



Properties

The web form supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Label

Labels are elements to be used within [Web Forms](#) and show text associated with a form element:



In the example above, you have a Web Form and some text fields and a button. The texts next to the text fields are Labels, and associated with the text fields, so that screen readers know which text field is for what.

You can associate a web form element such as a text field with a label in two ways:

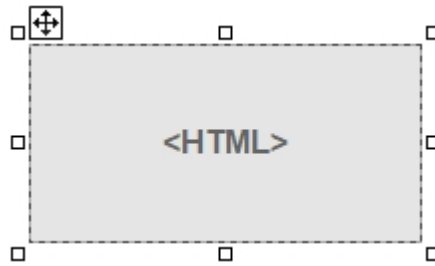
- Select the "for" element in the property window of the Label element to link another element
- Or: drag the form element into the label.

HTML Code

The HTML Code component allows to include HTML code directly into the website.

This is useful for example to include code you received from other pages to include on your website, for example to add a Website statistics counter on it, or to embed some video like from YouTube or a

'like' button from Facebook on the site.



Additionally, you can add elements not directly supported by RocketCake on your site with it.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. These are only available when the checkbox 'CreateDivAroundCode' is checked.

By double-clicking the element or by selecting the button '...' on the 'Code' entry in the property window, you can directly enter the HTML code.

PHP Code

The PHP Code component allows to include PHP code directly into the website.



This is useful for example to extend the website with more features, like guestbooks, blogs, and similar. Note: You need to run your website on a web server which supports PHP in order to make this work. Also, you should ensure that the file name extension of the web page which includes a PHP code element usually should end with '.php' instead of '.html', so you might want to rename your page for that.

There is a [detailed tutorial showing how to use the PHP code element](#).

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. These are only available when the checkbox 'CreateDivAroundCode' is checked.

By double-clicking the element or by selecting the button '...' on the 'Code' entry in the property window, you can directly enter the PHP code.

Alternative: Inline PHP Code

You don't need to use that PHP Code element. As alternative, you can place also very easily inline PHP code at any place, if you have the professional edition of RocketCake.

You only need to enclose it in the PHP tags "<?PHP" and ">?". For example just write something like

```
<?PHP echo ('hello world'); ?>
```

And RocketCake will add this code at the place where you typed it. This will only work if the name of the generated webpage has the extension .php. And it will only run if started from a webserver with PHP support.

JavaScript Code

The JavaScript Code component allows to include JavaScript code directly into the website. This is useful for example to extend the website with new dynamic features, like animations and dynamic notifications.



Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. These are only available when the checkbox 'CreateDivAroundCode' is checked.

By double-clicking the element or by selecting the button '...' on the 'Code' entry in the property window, you can directly enter the PHP code.

Table

You can create simple text based tables using the Table element:

Table with background color cell		
Data 1	some other text	This is a cell spanning multiple rows.
Data 2		
Data 3		

A Table can contain multiple rows and columns, and it is possible to add or remove columns and rows dynamically in the editor, using the menu Edit -> Table, when a table cell is selected.

Don't use the table element for layouting: It is intended to show data and might not work on responsive pages as you expect it to do. For layouting, use the [containers](#) instead.

Properties

Note that the table cells have different properties than the cell itself. When you select a table cell, only a few entries show up in the property window, like TextVAlignment and the Background. To show and edit the properties of the whole table, click on the border of the table to select it. Then there are several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes. The background will only be visible if the image file cannot be loaded or the image is transparent.

Other supported properties are:

CellPadding: The distance between the content of a table cell and its border.

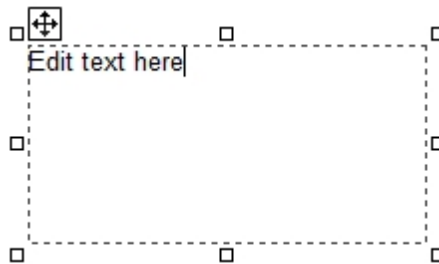
CellSpacing: The distance between each cell

Floating Text

The text component can be used to freely position text anywhere on the website or within containers.

Creating Floating Text

Click on the Entry in the Toolset on the right named 'Floating Text' or alternatively, use the Menu 'Insert -> Floating Text'. Then, click somewhere on the empty page. A text will appear, and clicking on it, you can now type in the text:



With the white rectangles around the area, you can resize the text, and by clicking and dragging the white rectangle with the arrows in it, you can move the text area anywhere you want. This works for every page.

As long as the text element is selected, you can see its Properties in the 'Properties' window on the left. There, you can change the background color, border and other settings for this text if you want.

Note: The position of a floating text is depending on the size of the parent component it is located in. Like for example the website. So if you use floating texts, the position of these in the final page is likely not what you want it to be. When the browser window has a different size, the position moves - because it is set to for example 40% of the size of the page. Instead of using floating texts, it is always a good idea to just use [containers](#) and type the text directly into them. A floating text is likely not what you want to use.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Quicktime Video

The Quicktime Video element makes it possible to embed a video directly on your website. Usually it is used to play back videos with the extension '.mov'.

Note: This will only work if the QuickTime plugin is installed on the computer of the user viewing the website.

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Other supported Properties are:

MovFilename: The .mov video file to be shown on the website.

Play: Whether to start playing the video when the website is shown

Loop: If looping the video is enabled or not

Controller: Show the control buttons (Play, Pause, Stop) to the user or not.

Youtube Video

The Youtube Video element makes it possible to quickly embed a video from Youtube.com on your website.



Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Other supported Properties are:

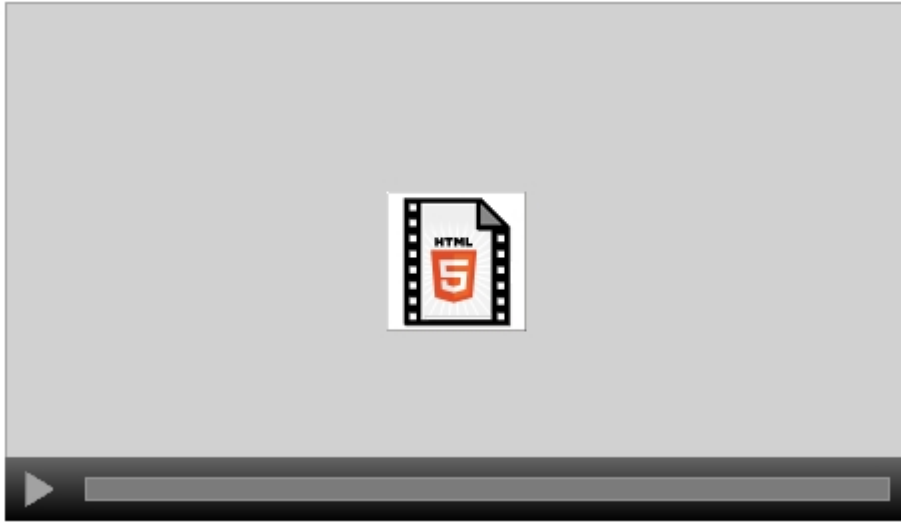
YoutubeVideo: The URL of the Youtube Video to be played on the website. Usually, it looks like something like this: <http://www.youtube.com/watch?v=dQw4w9WgXcQ>

Error 153

In case the video player shows an "Error 153" when previewing the video: The developers on the side of youtube something break it, and it will show this error when viewed from a .html page on your disk, and not from a web server. It will usually work without error when you view your page on the web, from a web server.

HTML5 Audio

Starting with HTML in version 5, it is now possible to include an audio and video playback control on websites without any plugins. This element creates such a control on your website.



Note:

You cannot play all audio and video file formats in all browsers. Some might for example support the .mp3 file format, while other's don't.

The control will look different depending on the operating system and browser of the user

Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

Other supported Properties are:

FileName: Select an audio/video file to be played on your website here.

Play: Whether to start playing when the website is shown or not. This currently works only on specific websites and is ignored by most browsers.

Loop: If the sound is played looped.

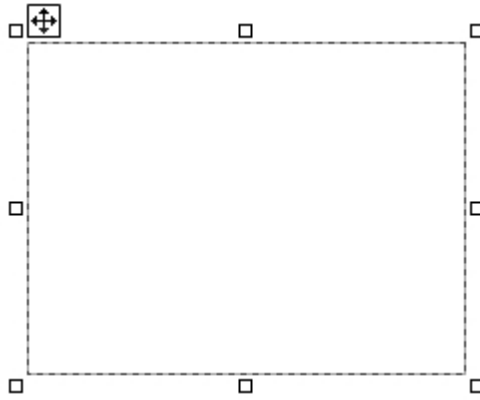
Controls: Whether to show the audio controls (Play, Pause, Stop) to the user or not. If this is not selected, the audio control will be invisible.

PosterImage: You can select an image for HTML 5 Video elements which is shown when the video has not been started or loaded yet.

Tracks: If checked, you can select track files (*.vtt) for subtitles, captions, chapters and similar. Note that these only usually work when the video is played from a webserver and not directly in preview mode from your disk.

IFrame

The IFrame element allows to embed other websites directly into your website, you only need to specify the URL of that website.



Properties

This component supports several background modes (Color, Image, Gradient), changeable by selecting the property 'BackgroundMode'. It also supports several border modes.

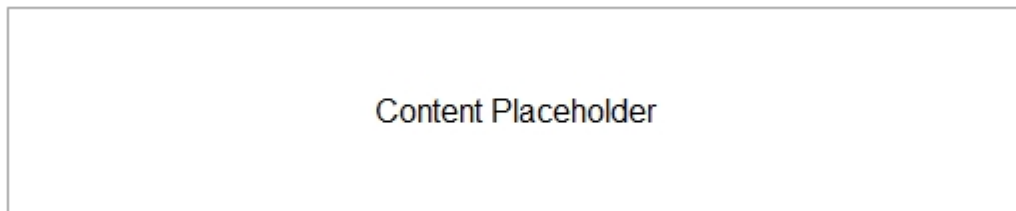
Other supported Properties are:

Web Address: Select an URL to be shown, for example <http://www.ambiera.com>

Scrollbars: Whether to show scrollbars around that frame or not.

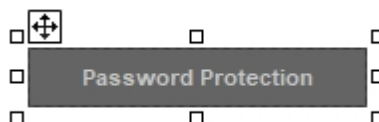
Content Placeholder

The Content Placeholder element allows to define a place where other pages are embedded, when this page is used as master page. See [Master Pages](#) for a detailed description.



Content Placeholder

RocketCake includes a built-in password protection mechanism, with which you can password protect all or specific pages of your website. You only need to put the "password protection" element on each website you want to protect.



See the [How to password protect a website tutorial](#) on how to do this.

Note: Only the websites themselves are protected by the password protection. If you include for example images and the user can obtain the URL of an included image on that page, he will be able to access that image directly.

PDF Document

You can select PDF documents in RocketCake which then get shown directly in the browser on the generated website. For that, use the PDF Document element and select a file from your disk.



The file will automatically be uploaded by RocketCake as well when you use the "Publish to the Internet" feature, so you don't need to keep track of all the files.

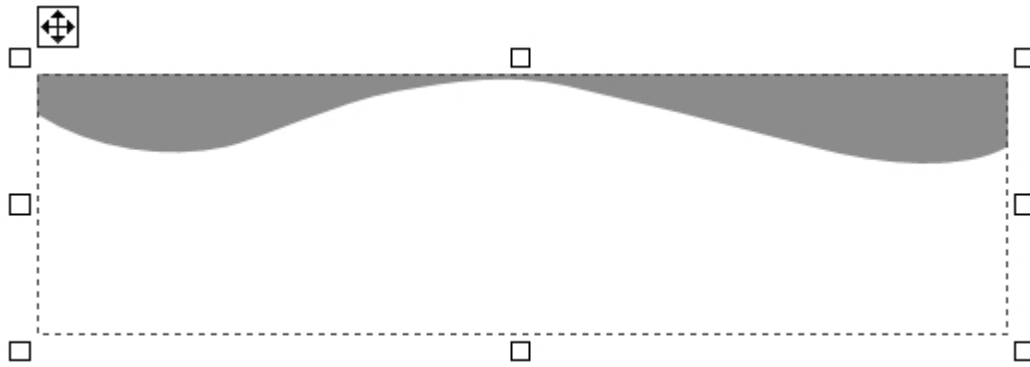
Note: Safari on iOS will only show the first page of the pdf document directly in the page, and it cannot be scrolled. This is a limitation of this browser of iOS. If you want to also have an embedded, scrollable PDF on that operating system, you can do this by embedding a PDF viewer script, in your page, which does the work for you. You can for example use Google's PDF Viewer for this. For this, simply create a JavaScript Code element on the page where your PDF Document element is, and paste this code in there:

```
var isIOS = /iPad|iPhone|iPod/.test(navigator.platform);
if (isIOS)
{
    var ifd = document.getElementsByTagName("IFRAME")[0];
    if (ifd)
    {
        var currentURL = ifd.src;
        var newURL =
'https://drive.google.com/viewerng/viewer?embedded=true&url='+currentURL+'';
        ifd.src = newURL;
        alert(ifd.src);
    }
}
```

This will then replace your embedded PDF document with the Google PDF viewer, if your page is opened on an iOS device like iPad or iPhone.

ShapeDivider

Shape Dividers work like a [containers](#) (you can place text, images etc into it) but include a generated SVG element to make a visually appealing section divider:



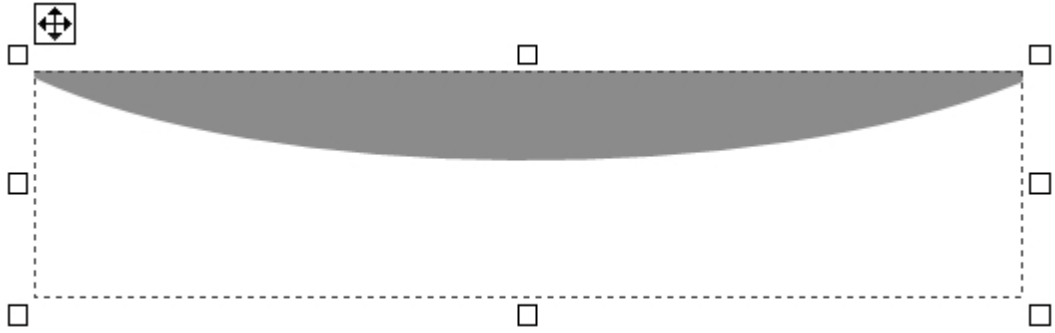
They are used to divide the sections on a website, and can be used to create a nice looking transition between pieces of content.



Shape Dividers in RocketCake have a set of built-in shapes (waves, curves, triangles, inclinations and more).

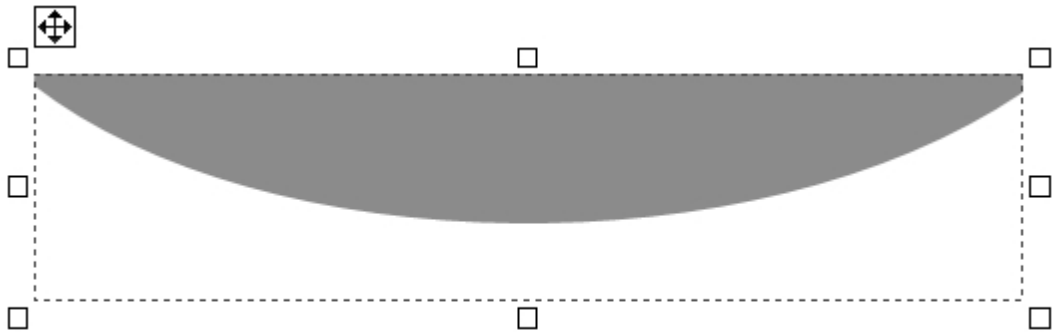
Using the "ShapeSize" property in the property window, it is possible to additionally change the layout of the shape. By default, the size is set to

100%, 30



Meaning the shape will cover 100% of the width of the container, and will have a height of 30 pixels.
By setting it to something like

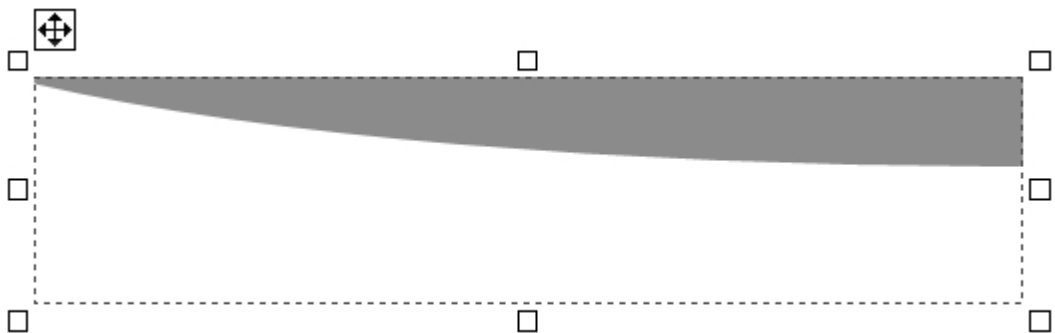
100%, 50



this will make the shape be larger.

With certain shapes (like the curve or the triangle), it is also a good idea to make the width of the shape wider, like 200%:

200%, 30

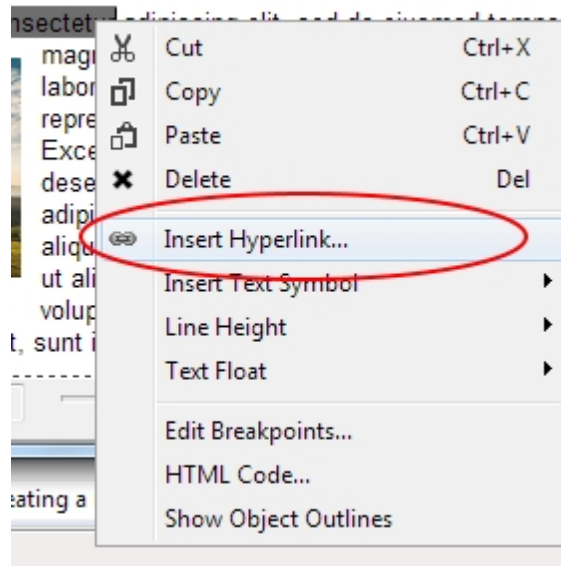


to only show part of the shape and make it look completely different.

Hyperlinks

Hyperlinks connect pages of your website with each other, and external websites and files with your site.

To add links between your pages or to other websites on the internet, mark the part of the text you want to be the hyperlink, right-click on it and select 'Insert Hyperlink...'



Alternatively, you can also use the green hyperlink icon in the toolbar of the editor. This also works for images and shapes (be it transparent, web2.0 or gradient shapes). A dialog will now open where you can enter the URL of the hyperlink.

Linking internal websites

To link two pages in this website you can choose 'Page in this project' as Link type and then select another page in this website.

Named Anchors

To link [Named Anchors](#), bookmarks in a page, simply select the 'Page in this project' and choose the named anchor in a page. It will be shown directly below the page if it exists.

Styles

If you are creating a text link, there will also be a 'Style' section in that dialog. Here you can define and reuse global named styles for your links, if you want more than one or some special styles. Defining different hover colors, disabling underlined links and more is possible here.

Special Actions

It is possible to trigger a special action for a hyperlink instead of linking a website with it. Choose 'Invoke a special RocketCake action' as link type for this. A dialog will then pop up where you can select the action and the target item for that action, for example to show the next image in a created [SlideShow](#).

Advanced Settings

Hyperlinks can have advanced settings, on the Advanced settings tab:

- rel: For adding attributes like rel="nofollow"
- title: For adding tooltips for the link when hovering it
- aria-label: Similar as "title" but is used by screen readers and not shown as tool tip, useful for accessibility
- id: For adding ids to all links

- download: For marking a link to let the browser download the target instead of opening it

PHP and ASP pages

RocketCake not only is able to create static HTML websites, but also dynamic websites created by PHP or ASP. This pages shows a short overview about how to do that, and it will focus on PHP, but it is basically the same with ASP websites.

In order to use PHP, you need to know HTML as well as how to program PHP. It is a programming language after all. It could be a good idea to read through some real PHP tutorials on the web (there are a lot of them), because explaining it here is quite out of scope.

Getting started

Using PHP, ASP and JavaScript is quite simple in RocketCake, and there are several ways to do this. For PHP, simply name your pages/files with a .php extension, not .html which is the default. To do this, change the 'FileName' property in the property window of the page when selected in RocketCake to something ending with '.php', instead of '.html'.

By using either the 'PHP Code' element (Insert -> PHP code), you can enter php code anywhere on the page. Double click it and enter for example this code to see if it works:

```
echo ('hello world');
```

Additionally, you can use the menu 'View -> HTML Code of Page' to insert PHP code before and inside the header of the website.

Inline PHP Code

At any place on the website you can also very easily add inline PHP code without using the code element, if you have the professional edition of RocketCake.

You only need to enclose it in the PHP tags "<?PHP" and "?>". For example just write something like

```
<?PHP echo ('hello world'); ?>
```

And RocketCake will add this code at the place where you typed it. This will only work if the name of the generated webpage has the extension .php. And it will only run if started from a webserver with PHP support.

Making it run

To test your PHP code, you need to run the page from a (local) webserver. Use the 'publish to local disk' feature of RocketCake (Menu: Publish -> 'to local disk') to write out the final .php file and then copy it to a webserver to test it out. If you only do the 'preview' instead, RocketCake will add a .html to your website and show it as it wasn't php.

Publishing the Website

Once you are finished with your website, you might want to publish it to an internet server, so that other people can read it.

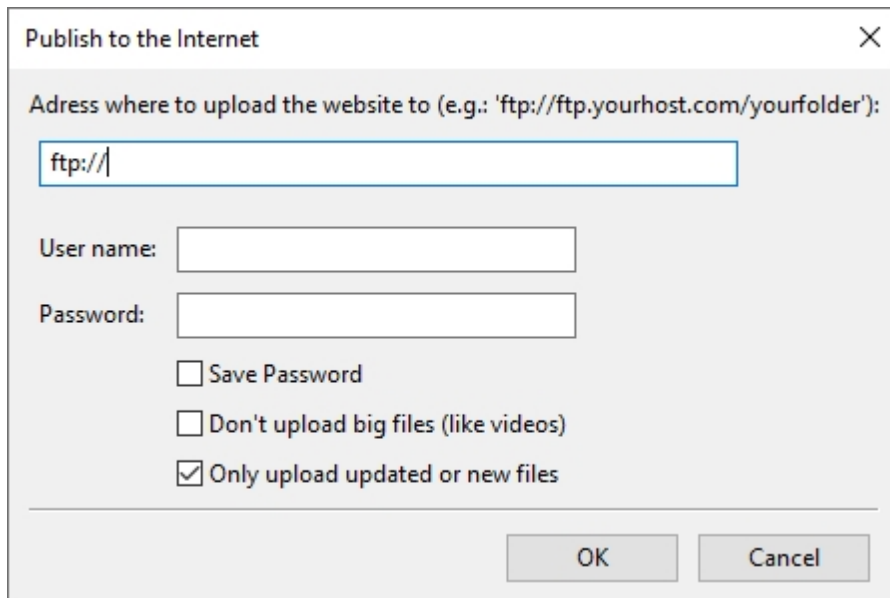
Using Publish to local disk

For this, choose the menu command 'Publish -> Publish to local disk'. A dialog will appear to select a target directory. When you press OK, all HTML and image files will be generated on your disk, in the directory you selected.

You now only need to upload these to your webserver or FTP server. For this you can use any FTP program. Ambiera recommends the free FTP client 'Filezilla' (<http://filezilla-project.org/>) or WinSCP (<http://winscp.net/>).

Using RocketCake's built-in FTP client

RocketCake also has a built-in FTP client, which can upload all the files automatically for you. For this, choose the menu command 'Publish -> Publish to the Internet'. A dialog will appear, asking you for the address and login data for the server:



Publish to the Internet

Address where to upload the website to (e.g.: 'ftp://ftp.yourhost.com/yourfolder'):

ftp://

User name:

Password:

Save Password

Don't upload big files (like videos)

Only upload updated or new files

OK Cancel

As address, you can specify an FTP server, such as ftp://ftp.yourhost.com. When pressing 'ok', RocketCake will try to upload all the files to this server.

If you want to specify a folder on your webspace where the website should be uploaded to, simply append it to the address like this: Assuming your ftp server address is ftp://ftp.yourhost.com/, and you want to upload it to a folder named 'myfolder', then the address would be:

ftp://ftp.yourhost.com/yourfolder

If your file server uses an port other than the default port 21 (this is usually not the case), you can specify another port by adding it after the host name, like this, for example for port 42:

ftp://ftp.yourhost.com:42/yourfolder

You can also enter username and password into the address line, like this:

ftp://yourUserName:YourPassword@ftp.yourhost.com/

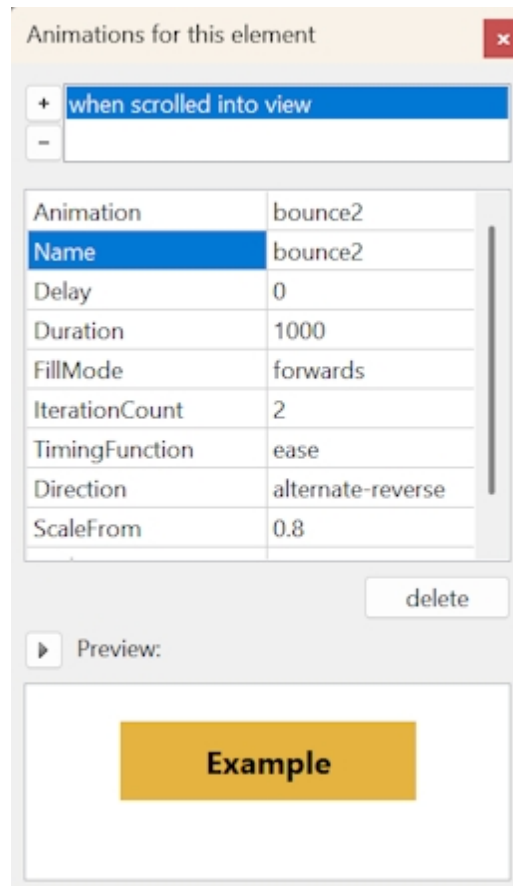
Note that in this case, leave the 'User name' and 'Password' field empty.

RocketCake also works with SFTP and FTPS, so if your server supports this, you can replace that 'ftp' part with 'sftp' or 'ftps' as well.

Animations

RocketCake supports adding animations to any element on a website - by using CSS Animations. This is possible through a very simple interface which still makes it possible to set complex and modern looking animations: You can right-click any element, select "Animations..." and add an animation.

The animation can be set to be started when the element scrolls into the view, when the user clicks or hovers an element, it can be simply shown always and more.



Animation Overview

Animation settings can be named and shared between elements. If you adjust the animation, it will be automatically adjusted on all elements using this animation

There are a couple of built-in animations like fading, bouncing, rotating, blurring, etc which can be easily adjusted.

You can set custom animations with your own keyframes: For that, add a new "custom animation" and set the keyframes as CSS manually.

Animations automatically honor the 'prefers-reduced-motion' setting of users browsers, for example not to trigger discomfort for users with vestibular motion disorders.

It is possible to set the "IterationCount" of animations not only to a number but also to the text "infinite" for infinite animations.

Automatic element hiding

Animations which only trigger when the element comes into view automatically keep track of the

element state so you don't have to manually fiddle around with any CSS or element states.

For example, if you set an animation to blend an element in when it comes into view, the element must be set to invisible initially, of course in order to look good. You don't have to do this, this is automatically done by the editor, but only if the element animation has "Fillmode" set to "forwards".

SEO friendly Animations

It's not certain if animated and especially initially invisible elements have a negative impact on SEO. But in that case, if the animation starts with an invisible opacity, RocketCake sets the initial opacity to a value slightly higher than 0, so that the text can still barely be read. In the hope that even non modern crawlers still work with your website nicely.

Additionally animations are implemented so that they are highly backwards compatible and even work with very old browsers - if possible. For example most animations will even work with the ancient Internet Explorer (!).

Generator Tag

RocketCake inserts a generator Meta Tag into your website when generating the website. The header of your website usually looks like this (note the bold generator meta tag):

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="generator" content="RocketCake">
  <title>Title of your website</title>
  <link rel="stylesheet" type="text/css" href="index.html.css">
</head>
```

If you want, you can now remove the 'generator=rocketcake' meta tag for every page: Click the menu View -> HTML Code of the page -> 'Additional code in the header', then just write the keyword

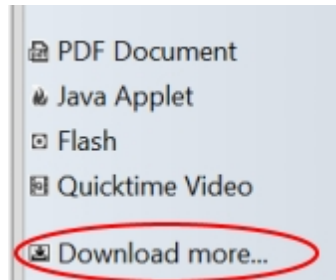
```
disableRocketCakeGeneratorTag
```

This keyword will be filtered out in the final output of the HTML code, and cause the generator tag to disappear for that page.

Extension components

RocketCake supports extension components which add more functionality. You can download these from the website or even create them on your own.

To download them, click the "Download more" button on the component list:



Once you downloaded a component from the website, double-click it and it will be installed on your disk, and can be used in RocketCake.

Note that these components will only work in the Professional Edition of RocketCake.

Creating your own extension components

You can create your own extension components for RocketCake easily. Follow these steps:

Create a text file in Your Documents\RocketCake\extensions and name it **test.rcjs**

Open it with a text editor like for example Notepad++ or VisualStudio Code. Set the language of this file to Javascript so that you get the correct syntax highlighting. (In Notepad++ click the Menu Language -> J -> Javascript, in VisualStudio code click the button on the bottom right which is named "Plain Text" and select "JavaScript")

Paste this code into the file:

```
// This is a scripted rocketcake component
//
// The following embedded xml is for the editor and describes how the component can be
edited:
// Supported types are: int, float, string, bool, color, rect
/*
    <component jsname="test" minRocketCakeVersion="5.2" version="1.0">
        <name lang="en" text="test component" />
        <description lang="en" text="A test component for trying out this feature"
/>
        <property name="text" type="string" default="text to be shown" />
    </component>
*/

test = function()
{
}

// called at the start, so all default settings can be set here
test.prototype.onInit = function()
```

```

{
    var me = rcGetThisElement();

    rcSetElementProperty(me, "Size", "50%, 22");
    rcSetElementProperty(me, "BackgroundMode", 0); // 0 = none, 1 = color, 2 = image,
3 = gradient
}

// called when HTML Code needs to be generated
test.prototype.onWriteHTML = function(isPreviewMode)
{
    var me = rcGetThisElement();

    // write HTML opening tag with id

    rcWriteCode("<div id=\"");
    rcWriteCode(rcGetHTMLElementId(me));
    rcWriteCode("\ " ");

    // write CSS style for user selected position and background

    rcWriteCode(" style=\"");
    rcWriteCode(rcGetCSSStyleForPosition(me));
    rcWriteCode(rcGetCSSStyleForBackground(me));
    rcWriteCode("\ " ");

    rcWriteCode(">");

    // content

    rcWriteCode(rcHTMLEncode(rcGetElementProperty(me, "text")));

    // closing tag
    rcWriteCode("</div>");
}

```

When you then start RocketCake, a new component named "test" should appear at the bottom of the components.

You can create it in the editor, and in preview mode, it should generate the text which you select in the property window under "text". Also, you are able to change the background color and style and position of the component in the editor easily.

In the code, you can see the onWriteHTML function. You can adjust it to write out any code you want.

Try it out:

Add another line into the onWriteHTML like this:

```
rcWriteCode("<h1>Hello World!</h1>");
```

Then let RocketCake reload the component:
Click View -> Show Log Window.

Right-Click into this Window and select "Reload All Extensions".

Then your new updated code was loaded into RocketCake.

When you now click "Preview" in RocketCake, your additional code should be written out and it should additionally spell "Hello World".

How components work

Extension components in RocketCake are just a JavaScript file, as you can see. In the beginning, it starts with a small XML section at the top, describing the components: A name, a description and the available properties are defined:

```
// The following embedded xml is for the editor and describes how the component can be
edited:
// Supported types are: int, float, string, bool, color, rect
/*
    <component jsname="test" minRocketCakeVersion="5.2" version="1.0">

        <name lang="en" text="test component" />
        <description lang="en" text="A test component for trying out this feature"
/>

        <property name="text" type="string" default="text to be shown" />
    </component>
*/
```

Note that the attribute "jsname" is set to "test" in this example. If you want to set another name for your component, like for example "JohnsComponent", then also rename the file to "JohnsComponent" and also rename the functions from "test.prototype.onInit" and "test.prototype.onWriteHTML" to "JohnsComponent.prototype.onInit" and "JohnsComponent.prototype.onWriteHTML".

The entries for the properties in the xml are for the property window in RocketCake. This line:

```
<property name="text" type="string" default="text to be shown" />
```

Shows a text (=string) property with the text set by default to "text to be shown".

Later in the code, we can access the value which the user has set it to in this way:

```
var me = rcGetThisElement();
rcGetElementProperty(me, "text");
```

You can easily add other properties of different types, like here:

```
<property name="scrollamount" type="int" default="6" />
<property name="open" type="bool" default="true" />
```

Developing components

RocketCake extension components are written in EcmaScript 5 - an older JavaScript version which didn't support "let" or "classes" yet. Use 'var' instead.

To reload and verify your modifications of your components, in RocketCake, click View -> Show Log Window.

Right-Click into this Window and select "Reload All Extensions".

Then your new updated code will be loaded and verified by RocketCake.

RocketCake components can have the following functions:

onInit - called at the start, so all default settings can be set here. Set position, background colors etc, but you can also create child elements. See [rcCreateElement\(\)](#) below.

onWriteHTML - called when HTML Code needs to be generated. Use rcWriteCode to write out your HTML and call rcWriteHTMLOfChildElements() to let child elements do the same

onWriteGlobalCSS - called when the global CSS styles are written out. Can be used to write out global styles like animations for this element, and similar. See [rcWriteGlobalCSSOfChildElements](#) out for an example.

onPaintInEditor - when you want to draw custom things in the editor, do it here. Use functions like rcDrawText and rcDrawElementBorder here. The summaryAndDetails component uses this feature for example.

onLayoutElement - called when the element gets layouted in the editor - when it calculates its position and similar. Use this to change positions of child elements if they need to be adjusted to for example a setting in the property window. The accordion component uses this for example.

The easiest way to create your own components is to download a few from the website, open them with a text editor and take a look at how they are doing it.

A list of all available functions can be found in the [extension components API documentation](#)

Extension components API

Extension components API overview

For an overview on how to write extension components yourself, see [this document](#).

The following API functions are available for creating your own components:

Element and Text Modifications

[rcSetElementProperty](#)
[rcGetElementProperty](#)
[rcSetElementText](#)
[rcInsertElementIntoText](#)
[rcSetTextHyperlink](#)
[rcSetTextAlignment](#)
[rcSetTextFontSize](#)
[rcSetFloatWhenAnchoredInText](#)
[rcSetAllowFontEditing](#)
[rcSetHyperlink](#)
[rcSetTextEditType](#)

Utility Functions

[print](#)
[color class](#)
[rect class](#)

Code and Styles

[rcWriteCode](#)
[rcHTMLEncode](#)
[rcGetHTMLElementId](#)
[rcGetCSSStyleForPosition](#)
[rcGetCSSStyleForBackground](#)
[rcWriteHTMLOfChildElements](#)
[rcWriteGlobalCSSOfChildElements](#)
[rcAddFileWithTextContent](#)

Drawing in Editor

[rcGetLayoutRect](#)
[rcDrawElementBackgroundBoxShadow](#)
[rcDrawElementBackground](#)

Navigating and creating elements

[rcGetThisElement](#)
[rcGetElementType](#)
[rcGetDocumentNode](#)
[rcCreateElement](#)
[rcGetElementFromId](#)
[rcGetElementChildCount](#)
[rcGetElementChild](#)

[d](#)
[rcDrawElementBorder](#)
[rcMeasureText](#)
[rcDrawText](#)
[rcDrawRectangle](#)
[rcDrawLine](#)

print(...)

Prints a message to the console. Makes it possible to debug your scripts. You can see the console in RocketCake by clicking View -> Log window.

Parameters:

- obj0 - obj1: A list of JavaScript objects to output
-

rcGetThisElement()

Returns a reference to the element represented by this component. The reference returned can be used as parameter in many other functions like [rcSetElementProperty](#) and similar.

rcGetElementType(elem)

Returns a string describing the type of the element, like "button" or "container"

Parameters:

- elem: Reference of the element

Example: To search for the page where the current element is located in, do it like this:

```
var currentNode = rcGetThisElement();
var page = null;

while(currentNode)
{
    if (rcGetElementType(currentNode) == "page")
    {
        page = currentNode; // found!
        break;
    }
    currentNode = rcGetElementParent(currentNode);
}
```

rcSetElementProperty(elem, propertyname, propertyvalue)

Sets the property of an element to a new value

Parameters:

- elem: Reference of the element
- propertyname: Name of the property

- `propertyvalue`: New value of the property

All elements have properties in RocketCake. You can see the properties in the property window when the element is selected. When you set the language of RocketCake to "english", then you mostly see the exact name of the property displayed there as you can also use it as the "propertyname" parameter here. Common properties are for example "BackgroundMode", "BackgroundColor", "BorderMode", "Size", "PositionType", "Position", "UseHoverStyles", "HoverFillColor", etc.

When a property is a string, for example like "Size", then you can set it like this:

```
rcSetElementProperty(button, "Size", "60%, 30");
```

When a property is a color, for example like "BackgroundColor", then you can set a color value in there using the color class (see below), like this:

```
rcSetElementProperty(button, "BackgroundColor", new color(0.5, 0.5, 0.5));
```

Some properties have special values:

`PositionType`: is an integer property with the following values: '0' for free position and '9' for "anchor in text".

`BackgroundMode`: is an integer property with the following values: 0 = none, 1 = color, 2 = image 3 = gradient, 4 = button

rcGetProperty(elem, propertyname)

Returns the property of an element.

Parameters:

- `elem`: Reference of the element
- `propertyname`: Name of the property

See [rcSetElementProperty\(\)](#) for a description of the properties.

Example:

```
var isOpen = rcGetProperty(rcGetThisElement(), "open");
```

If the element has a property named "open", then the value the user has set this property to is stored in `isOpen`.

rcGetDocumentNode()

Returns the document element reference. This is basically the root node. Can be used to search for a specific element.

Parameters:

- none

rcCreateElement(parent, type)

Creates a new element.

Parameters:

- `parent`: Where the element is to be created in
- `type`: type of the element. Can be: "text", "image", "table", "iframe", "form", "button",

"textedit", "datepicker", "shape", "container", "label", "slideshow", "carousel".

Example:

In the `onInit()` function of extension components, you can create child elements. For example when you create a contact form, you want a few input elements and a send button. For this, you need to create a container child element as root element for these and then place the child elements in there. In this example, we create a container child element and a button with the text "Press Me" and a hyperlink:

```
// called at the start, so all default settings can be set here
myComponent.prototype.onInit = function()
{
    var me = rcGetThisElement();

    rcSetElementProperty(me, "Size", "220, 50");
    rcSetElementProperty(me, "BackgroundMode", 0); // 0 = none, 1 = color, 2 = image,
3 = gradient

    // create container child element

    var textContainerElem = rcCreateElement(me, "text");
    rcSetElementProperty(textContainerElem, "Size", "100%, 100%");
    rcSetElementProperty(textContainerElem, "Position", "0%, 0%");

    // create button

    var button = rcCreateElement(textContainerElem, "shape");
    rcSetElementProperty(button, "BackgroundMode", 1); // 0 = none, 1 = color, 2 =
image, 3 = gradient
    rcSetElementProperty(button, "BackgroundColor", new color(0.5, 0.5, 0.5));
    rcSetElementProperty(button, "BorderMode", 0); // 0 = none
    rcSetElementProperty(button, "Size", "200, 28");
    rcSetElementProperty(button, "PositionType", 9); // 0 = free, 9 = anchored in
text
    rcSetElementProperty(button, "Position", "0%, 80");
    rcSetElementProperty(button, "UseHoverStyles", true);
    rcSetElementProperty(button, "HoverFillColor", new color(111 / 255.0, 111 / 255.0,
111 / 255.0));
    rcSetElementProperty(button, "Margin", "10 10 10 10");
    rcSetElementText(button, "Press Me");

    // add hyperlink
    rcSetHyperlink(button, "https://www.ambiera.com");
}
```

rcSetElementText(elem, txt)

Sets the whole text of an element.

Parameters:

- o elem: Reference of the element to change
- o txt: Text to set

Example:

Place this in the `onInit` function to create a text element with the text "Note: See our policy for details."

```
var textElem = rcCreateElement(rcGetThisElement(), "text");
rcSetElementProperty(textElem, "Size", "80%, 80%");
rcSetElementProperty(textElem, "Position", "10%, 10%");
rcSetElementText(textElem, "Note: See our policy for details.");
```

rcInsertElementIntoText(elem, elemToInsert, position)

Inserts an element at a specific position into the text of an element

Parameters:

- elem: Reference of the element to change
- elemToInsert: Reference of the element to insert
- position: Position in the text

rcSetTextHyperlink(elem, textPosFrom, textPosTo, url)

Sets the hyperlink of a part of a text. If you want to set the hyperlink for a whole element like an image, use [rcSetHyperlink](#) instead.

Parameters:

- elem: Reference of the element to change
- textPosFrom: position in the text where the hyperlink should start
- textPosTo: position in the text where the hyperlink should end
- url: A url like "https://www.ambiera.com"

Example:

Place this in the `onInit` function to create a text element with the text "Note: See our policy for details." and the 'policy' text will then be a hyperlink to 'example.com'.

```
var textElem = rcCreateElement(rcGetThisElement(), "text");

rcSetElementProperty(textElem, "Size", "80%, 80%");
rcSetElementProperty(textElem, "Position", "10%, 10%");
rcSetElementText(textElem, "Note: See our policy for details.");

rcSetTextHyperlink(textElem, 15, 21, "https://www.example.com");
```

rcSetTextAlignment(elem, textPosFrom, textPosTo, alignment)

Sets the alignment of a part of a text

Parameters:

- elem: Reference of the element to change
- textPosFrom: position in the text where the hyperlink should start
- textPosTo: position in the text where the hyperlink should end
- alignment: A string, either "left", "right", "center" or "justify"

Example:

The child text element will have the text "Hello Here" and it will be centered.

```
test.prototype.onInit = function()
{
    var me = rcGetThisElement();

    rcSetElementProperty(me, "PositionType", 9); // 0 = free, 9 = anchored in text
    rcSetElementProperty(me, "Size", "50%, 320");
    rcSetElementProperty(me, "MinWidth", 350);
```

```

    rcSetElementProperty(me, "BackgroundMode", 1); // 0 = none, 1 = color, 2 = image,
3 = gradient
    rcSetElementProperty(me, "BackgroundColor", new color(0.9, 0.9, 0.9));

    // create container child element

    var textContainerElem = rcCreateElement(me, "text");
    rcSetElementProperty(textContainerElem, "Size", "100%, 100%");
    rcSetElementProperty(textContainerElem, "Position", "0%, 0%");

    rcSetElementText(textContainerElem, "\nHello There\n\n\n\n");

    rcSetTextAlignment(textContainerElem, 0, 14, "center");
}

```

rcSetTextFontSize(elem, textPosFrom, textPosTo, size)

Sets the font size of a part of a text

Parameters:

- elem: Reference of the element to change
- textPosFrom: position in the text where the hyperlink should start
- textPosTo: position in the text where the hyperlink should end
- size: Font size like 12

rcSetFloatWhenAnchoredInText(elem, float)

Sets the float of an element when it is anchored in the text. Same as when you right-click onto an element in the editor and select "Text Float -> float"

Parameters:

- elem: Reference of the element to change
- float: Type of float: 0=none, 1=left, 3=right

rcSetAllowFontEditing(elem)

To be called in the onInit function, this enables font properties to be set for this whole element when it is selected in the editor. Properties of the font can be accessed then via properties, see the example below.

Parameters:

- elem: Reference of the element to change

Example:

An example onInit() method which calls rcSetAllowFontEditing(), so that fonts for that element can be selected:

```

test.prototype.onInit = function()
{
    var me = rcGetThisElement();

    rcSetElementProperty(me, "Size", "50%, 22");
    rcSetElementProperty(me, "BackgroundMode", 0); // 0 = none, 1 = color, 2 = image,

```

```
3 = gradient
```

```
    rcSetAllowFontEditing(me); // enables the font selection bar when this element is
selected,                               // and provides the selected font in the properties
"FontName",                               // "FontSize", "FontItalic", "FontColor" etc
}
```

Later, for example in the `.onWriteHTML`, these can be accessed via `rcGetElementProperty()` and written out as CSS styles like this:

```
// get font style

var fontName = rcGetElementProperty(me, "FontName");
if (fontName == "") fontName = "Arial, Helvetica, sans-serif"; // default font
var fontSize = rcGetElementProperty(me, "FontSize");
var fontBold = rcGetElementProperty(me, "FontBold");
var fontItalic = rcGetElementProperty(me, "FontItalic");
var fontUnderline = rcGetElementProperty(me, "FontUnderline");
var fontColor = rcGetElementProperty(me, "FontColor").getHTMLColor();

var fontStyle = "";
fontStyle += "font-size:" + fontSize + "pt; ";
fontStyle += "font-family:" + fontName + "; ";
fontStyle += "color:" + fontColor + "; ";
if (fontItalic) fontStyle += "font-style:italic; ";
if (fontUnderline) fontStyle += "text-decoration:underline; ";
if (fontBold) fontStyle += "font-weight:bold; ";

// write CSS style for font style

rcWriteCode(" style=\"");
rcWriteCode(fontStyle);
rcWriteCode("\" ");
```

In case you want to draw something with this font in the editor, in the `.onPaintInEditor` function, you can do it like this:

```
// get font style

var me = rcGetThisElement();

var fontName = rcGetElementProperty(me, "FontName");
var fontSize = rcGetElementProperty(me, "FontSize");
var fontBold = rcGetElementProperty(me, "FontBold");
var fontItalic = rcGetElementProperty(me, "FontItalic");
var fontUnderline = rcGetElementProperty(me, "FontUnderline");
var fontColor = rcGetElementProperty(me, "FontColor");

// finally draw

var text = "Hello World";
var rect = rcGetLayoutRect(me);

rcDrawText(rect.x1, rect.y1, text, fontName, fontSize, fontColor, fontBold,
fontItalic, fontUnderline);
```

rcWriteCode(text, noHTMLencode)

Writes out HTML or CSS code to the target file.

Parameters:

- text: Code to write
- noHTMLEncode: optional. Set to 'true' to disable HTML encoding when writing out code.

Example:

In the `onWriteHTML` function, you can write "Hello World" like this:

```
rcWriteCode("<div>Hello World!</div");
```

rcHTMLEncode(text)

Creates a HTML encoded string from an input string. For example for the text "<this & that>", it writes out "<this & that>".

Parameters:

- text: Code to write

Returns:

- html encoded string

rcGetHTMLElementId(elem)

In RocketCake, every element has it's own HTML Id. It can also be modified by the user. You can return the Id as string using this function.

Parameters:

- elem: Element to get the HTML Id from

Returns:

- html id

Example:

In the `onWriteHTML` function, you can write the div tag for your component like this, with the correct HTML id:

```
// write HTML opening tag
rcWriteCode("<div id=\"");
rcWriteCode(rcGetHTMLElementId(rcGetThisElement()));
rcWriteCode("\" >");

// closing tag
rcWriteCode("</div>");
```

rcGetCSSStyleForPosition(elem)

Returns the user selected CSS styles for the position of the element.

Parameters:

- o elem: Element to get the styles from

Returns:

- o string with css styles

Example:

```
// write div with CSS style for user selected position and background

rcWriteCode("<div style=\"");
rcWriteCode(rcGetCSSStyleForPosition(me));
rcWriteCode(rcGetCSSStyleForBackground(me));
rcWriteCode("\"> </div> ");
```

rcGetCSSStyleForBackground(elem)

Returns the user selected CSS styles for the background of the element.

Parameters:

- o elem: Element to get the styles from

Returns:

- o string with css styles

Example:

```
// write div with CSS style for user selected position and background

rcWriteCode("<div style=\"");
rcWriteCode(rcGetCSSStyleForPosition(me));
rcWriteCode(rcGetCSSStyleForBackground(me));
rcWriteCode("\"> </div> ");
```

rcWriteHTMLOfChildElements(elem)

In the .OnWriteHTML function, this will cause child elements to write out their HTML code and css. This is useful when your element has child elements such as web form elements in a web form.

Parameters:

- o elem: Parent element

Example:

```
// called when HTML Code needs to be generated
test.prototype.onWriteHTML = function(isPreviewMode)
{
    var me = rcGetThisElement();

    // write HTML opening tag
```

```

rcWriteCode("<div id=\"");
rcWriteCode(rcGetHTMLIdElementId(me));
rcWriteCode("\" >");

// let child elements like the text and the button write out their html

rcWriteHTMLOfChildElements(me);

// closing tag
rcWriteCode("</div>");
}

```

rcWriteGlobalCSSOfChildElements(elem)

In the `.OnWriteHTML` function, this will cause child elements to write out their HTML code and css. This is useful when your element has child elements such as web form elements in a web form.

Parameters:

- o elem: Parent element

Example:

```

// called when HTML Code needs to be generated
test.prototype.onWriteHTML = function(isPreviewMode)
{
    var me = rcGetThisElement();

    // write HTML opening tag

    rcWriteCode("<div id=\"");
    rcWriteCode(rcGetHTMLIdElementId(me));
    rcWriteCode("\" >");

    // let child elements like the text and the button write out their html

    rcWriteHTMLOfChildElements(me);

    // closing tag
    rcWriteCode("</div>");
}

```

rcAddFileWithTextContent(filename, text)

In the `.OnWriteHTML` function, this will cause a text file to be written out with the text content supplied. The function will return the web root relative filename used to create the file if successful.

Parameters:

- o filename: A filename to be used, like "sitemap.xml"
- o text: Some text to be written out into the file

Example:

```
// called when HTML Code needs to be generated
test.prototype.onWriteHTML = function(isPreviewMode)
{
    rcAddFileWithTextContent("hello.txt", "hello world!");
}
```

rcSetHyperlink(elem, url)

Sets the hyperlink of an element. Unlike [rcSetTextHyperlink](#), which sets the hyperlink in a part of the text.

Parameters:

- elem: Reference of the element to change
- url: A url like "https://www.ambiera.com"

rcSetTextEditType(elem, type)

Sets the type of an text edit web form element.

Parameters:

- elem: Reference of the element
- type: String, must be one of these: "password", "multiline", "email", "numeric", "date", "file"

rcGetElementFromId(id)

Returns a reference to the element represented by the specified id.

Parameters:

- Id of the element to search for

rcGetElementChildCount(elem)

Returns the amount of children this element has.

Parameters:

- elem: Reference of the element

rcGetElementChild(elem, idx)

Returns the nth child of this element

Parameters:

- elem: Reference of the element

- idx: Index of the child, value ≥ 0 and $< rcGetElementChildCount$

rcGetElementParent(elem)

Returns the parent element of this element.

Parameters:

- elem: Reference of the element

rcGetLayoutRect(elem)

Returns the visible rectangle coordinates of this element. Can be used to draw the rectangle of this element.

Parameters:

- elem: Reference of the element

Returns:

- rectangle with the coordinates of type [rect](#)

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    rcDrawElementBorder(me, rect);
}
```

rcDrawElementBackgroundBoxShadow(elem, rect)

Draws the element's background box shadow, if the element was set in the editor to have a box shadow. Should only be called in a `onPaintInEditor()` function.

Parameters:

- elem: Reference of the element to draw
- rect: (optional) area to draw

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    rcDrawElementBackgroundBoxShadow(me, rect);
    rcDrawElementBackground(me, rect);
    rcDrawElementBorder(me, rect);
}
```

rcDrawElementBackground(elem, rect)

Draws the element's background as set in the editor by the user. Should only be called in a `onPaintInEditor()` function.

Parameters:

- `elem`: Reference of the element to draw
- `rect`: (optional) area to draw

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    rcDrawElementBackgroundBoxShadow(me, rect);
    rcDrawElementBackground(me, rect);
    rcDrawElementBorder(me, rect);
}
```

rcDrawElementBorder(elem, rect)

Draws the element's border as set in the editor by the user. Should only be called in a `onPaintInEditor()` function.

Parameters:

- `elem`: Reference of the element to draw
- `rect`: (optional) area to draw

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    rcDrawElementBackgroundBoxShadow(me, rect);
    rcDrawElementBackground(me, rect);
    rcDrawElementBorder(me, rect);
}
```

rcMeasureText(text, fontfamily, fontsize, fontbold, fontitalic, fontunderline)

returns the size of a text when drawn with the specified font settings. Should only be called in a `onPaintInEditor()` function.

Parameters:

- `text`: Text to measure. Like "Hello world"
- `fontfamily`: (optional) Font family like "Arial"
- `fontsize`: (optional) Size of font like 12
- `fontbold`: (optional) If font should be bold. Example: true.

- fontitalic: (optional) If font should be italic. Example: true.
- fontunderline: (optional) If font should be underlined. Example: true.

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    var fontName = rcGetElementProperty(me, "FontName");
    var fontSize = rcGetElementProperty(me, "FontSize");
    var fontBold = rcGetElementProperty(me, "FontBold");
    var fontItalic = rcGetElementProperty(me, "FontItalic");
    var fontUnderline = rcGetElementProperty(me, "FontUnderline");
    var fontColor = rcGetElementProperty(me, "FontColor");

    // calculate position to draw it centered in the middle of the rectangle

    var text = "Hello World";

    // measure text size in order to draw it centered in element

    var size = rcMeasureText(text, fontName, fontSize, fontBold, fontItalic,
fontUnderline);
    var posx = rect.x1 + ((rect.x2 - rect.x1) - size.x1) / 2;
    var posy = rect.y1;

    // finally draw

    rcDrawText(posx, posy, text, fontName, fontSize, fontColor, fontBold, fontItalic,
fontUnderline);
}
```

rcDrawText(x,y, text, fontfamily, fontsize, fontColor, fontbold, fontitalic, fontunderline)

draws a text. Should only be called in a `onPaintInEditor()` function.

Parameters:

- x: x position where to draw the text
- y: y position where to draw the text
- text: Text to draw. Like "Hello world"
- fontfamily: (optional) Font family like "Arial"
- fontsize: (optional) Size of font like 12
- fontColor: (optional) Color to use for drawing. Example: `new color(1.0, 0, 0)`;
- fontbold: (optional) If font should be bold. Example: true.
- fontitalic: (optional) If font should be italic. Example: true.
- fontunderline: (optional) If font should be underlined. Example: true.

Example:

```
test.prototype.onPaintInEditor = function()
{
    var me = rcGetThisElement();
    var rect = rcGetLayoutRect(me);

    rcDrawText(rect.x1, rect.x2, "Hello World", "Arial", 12, new color(1.0, 0, 0),
```

```
false);  
}
```

rcDrawRectangle(color, x1, y1, x2, y2)

draws a rectangle. Should only be called in a `onPaintInEditor()` function.

Parameters:

- color: color of the rectangle. Example: `new color(1.0, 0, 0)`;
- x1: x1 coordinate of the rectangle
- y1: y1 coordinate of the rectangle
- x2: x2 coordinate of the rectangle
- y2: y2 coordinate of the rectangle

Example:

```
test.prototype.onPaintInEditor = function()  
{  
    var me = rcGetThisElement();  
    var rect = rcGetLayoutRect(me);  
  
    rcDrawRectangle(new color(1.0, 0, 0), rect.x1, rect.y1, rect.x2, rect.y2);  
}
```

rcDrawLine(color, x1, y1, x2, y2)

draws a line. Should only be called in a `onPaintInEditor()` function.

Parameters:

- color: color of the rectangle. Example: `new color(1.0, 0, 0)`;
- x1: x1 coordinate of the line
- y1: y1 coordinate of the line
- x2: x2 coordinate of the line
- y2: y2 coordinate of the line

Example:

```
test.prototype.onPaintInEditor = function()  
{  
    var me = rcGetThisElement();  
    var rect = rcGetLayoutRect(me);  
  
    rcDrawLine(new color(1.0, 0, 0), rect.x1, rect.y1, rect.x2, rect.y2);  
}
```

color class

The RocketCake API has a built-in color class, for getting and setting properties, mostly via **rcSetElementProperty()** and **rcGetElementProperty()**.

You can create a new color by specifying float RGB values from 0.0 to 1.0:

```
var clr = new color(0.5, 0.5, 0.5);
```

It has a method **getHTMLColor()** to return a HTML color string like "#ff00ab"

```
clr.getHTMLColor();
```

And a **toString()** function to create a string like "(1.0, 0.6, 0.5)" from its float values.

You can access the r, g, and b values using the properties **.r**, **.g**, and **.b**:

```
print(clr.r);
```

rect class

The RocketCake API has a built-in rect class, for getting and setting properties, mostly via **rcSetElementProperty()** and **rcGetElementProperty()**.

You can create a new rect by specifying float position values:

```
var rct = new rect(10, 10, 200, 200);
```

You can access the position values using the properties **.x1**, **y1**, **x2**, and **.y2**:

```
print(rct.x1);
```

Advanced Settings

Here are some further advanced settings available in RocketCake:

Change generated HTML ID

You can change the generated HTML Id of every element on any page in the professional version of RocketCake. To do this, right-click any element, select "HTML Code...", click the "..." button and select "Change HTML ID...".

You will see the change in the code immediately.

Setting HTML page language attribute

You can set the language of a page using the "Language" setting for each page in the professional version of RocketCake. This is useful when creating multi-language websites.

For this, click the page in the document explorer, and enter the wanted language attribute in the "Language" entry in the property window. This results in the top html tag being written like for example

```
<html lang="en">
```

if you specify "en" as language.

Installing RocketCake

Some advanced features when installing RocketCake:

Install RocketCake silently

You can install RocketCake silently using the RocketCake installer (RocketCake-Setup.exe). Use the /S option for that. Run RocketCake-Setup.exe like this:

```
RocketCake-Setup.exe /S
```

Alternatively, you can also simply copy the content of an installed RocketCake (which can be found usually in the RocketCake installation directory under "C:\Program Files (x86)\Ambiera\RocketCakeXX", and copy this to another PC. It will work without installation.

Installing RocketCake on many PCs in organizations or Schools

There is a .msi installer available on the [official download page](#) which makes this easily possible.

Using the same RocketCake license for all Users on the same PC

If you want to use RocketCake on a Windows PC with the same license key used by many users, this is possible, too:

Simply start RocketCake once as administrator and register your key using the menu Help -> Register. Then the key then is stored per machine (as previously only per user) and RocketCake can be used by any user using that machine.

Distributing RocketCake licenses to multiple PCs

If you are using your RocketCake license on many Windows PCs, you can deploy its license to multiple PCs easily, too:

Just install RocketCake locally and enter your license under Help -> Register. Then open the Help -> Register dialog again, and see that there is now a new button labeled "...". Clicking on this offers a new menu item to export the license as .reg file. This is a standard registry file which can be double-clicked on any PC to save the license key there.

It is also possible to distribute this .reg file easily via GPO group policy, on workstations on your whole domain.

Portable version of RocketCake

There is a portable version of RocketCake as .zip archive available as download from the [official download page](#), scroll a bit down to the bottom.

Online Help

There are several online websites which might help you with RocketCake

The official RocketCake website: <https://www.ambiera.com/rocketcake/index.html>

The forum: <https://www.ambiera.com/forum.php>

License and Copyright

END-USER LICENSE AGREEMENT FOR RocketCake

This Agreement is a legal document between you and Ambiera e.U / N.Gebhardt ("Ambiera"). Ambiera is willing to license the software to you as an individual, the company or the legal entity that will be utilizing the software (referenced below as "you" or "your") only on the condition that you accept all of the terms of this end user license agreement ("Agreement"). Read the terms and conditions of this agreement carefully before using the software. If you click the "OK" or "I agree" Button, and by installing or using any part of the Software, you agree to the terms and conditions of this agreement. If you do not agree to these terms and conditions, click on the "No" or "Cancel" button, then you are not licensed to use the software and you must uninstall or destroy all copies of it.

License

The software is the property of Ambiera and is protected by the copyright law. The software is licensed and not sold. The software is owned by Ambiera. Ambiera retains all rights, including all copyrights and intellectual property rights in and to the software, its documentation, title, logos, data files and all copies thereof. All rights not explicitly granted to you in this license are reserved by the Ambiera. Upon your acceptance of this software license agreement Ambiera grants you a non-exclusive, non-transferable, limited license to install and use a copy of the software on your computer, up to the permitted number of computers. As between you and Ambiera, files, applications, generated program code and flash files that are authored or created by you via your utilization of the Software, in accordance with the terms of this software license agreement, are your property.

You may not transfer, sublicense, network, loan, lend, lease, distribute, rent, modify, translate, disassemble, decompile, reverse engineer, translate the software into another computer language, otherwise reduce the software to human perceivable form, create derivative works based upon the software other than as otherwise provided herein, or copy or use the Software and/or the Software documentation in violation of this Agreement.

License Key

You may use and install the free version of the software on your computer. You can purchase a license key code for the professional edition which will enable you to activate advanced features of the software. You may not re-license, sell, reproduce or distribute any key code except with the express written permission of Ambiera.

Disclaimer of Warranty

THE SOFTWARE IS PROVIDED BY AMBIERA "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL AMBIERA OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED, BUT IT MAY BE LIMITED, AMBIERA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF TEN DOLLARS (USD \$10) IN TOTAL.

Furthermore, you acknowledge that you were able to test and use the demo version of the Software freely for a limited amount of time before purchasing a full license. You are thus conscious of the exact functionality provided by the software and will not claim any missing or wrongly advertised functionality by Ambiera.

Update Mechanism

This software may contain an automatic online update checking and registration feature through which

the software connects itself with Ambiera, on Ambiera's servers. The software may automatically connect and register itself with Ambiera when there is an internet connection. This process may be done entirely in the background and no dialogs or other data is displayed on your computer screen during this process. Only information concerning the identity of the user, his computer and informations about the installed software and used software license is collected for this process.

Termination

If you fail to comply with the terms and conditions of this license, Ambiera will terminate the License agreement and you immediately have to delete all copies of the Software.

Support

This license does not include support services, although Ambiera can decide to offer you prioritized support via email. Repeated voluntary provision of support services by Ambiera does not constitute a claim for future provision of support services for you.

Jurisdiction and Choice of Law

Insofar as not otherwise agreed, this Agreement shall be governed exclusively by Austrian law. Austrian law shall also be applied in case of you being located outside of Austria. You expressly agree that exclusive jurisdiction for any claim or dispute with Ambiera linked in any way to your use of the Software and/or Documentation resides in the competent court in the town of Vienna, Austria.

Severability

Should individual terms of this agreement be or become inoperative, this will not affect the remaining terms of this agreement. Ambiera and you will work in a spirit of partnership to find an arrangement that approximates the inoperative terms as closely as possible.

Licenses of enclosed software:

wxWidgets

wxWindows Library Licence, Version 3.1

=====

Copyright (c) 1998-2005 Julian Smart, Robert Roebing et al

Everyone is permitted to copy and distribute verbatim copies of this licence document, but changing it is not allowed.

WXWINDOWS LIBRARY LICENCE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public Licence as published by the Free Software Foundation; either version 2 of the Licence, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public Licence for more details.

You should have received a copy of the GNU Library General Public Licence along with this software, usually in a file named COPYING.LIB. If not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth

Floor, Boston, MA 02110-1301 USA.

EXCEPTION NOTICE

1. As a special exception, the copyright holders of this library give permission for additional uses of the text contained in this release of the library as licenced under the wxWindows Library Licence, applying either version 3.1 of the Licence, or (at your option) any later version of the Licence as published by the copyright holders of version 3.1 of the Licence document.
2. The exception is that you may use, copy, link, modify and distribute under your own terms, binary object code versions of works based on the Library.
3. If you copy code from files distributed under the terms of the GNU General Public Licence or the GNU Library General Public Licence into a copy of this library, as this licence permits, the exception does not apply to the code that you add in this way. To avoid misleading anyone as to the status of such modified files, you must delete this exception notice from such code and/or adjust the licensing conditions notice accordingly.
4. If you write modifications of your own for this library, it is your choice whether to permit this exception to apply to your modifications. If you do not wish that, you must delete the exception notice from such code and/or adjust the licensing conditions notice accordingly.

Curl

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2018, Daniel Stenberg, daniel@haxx.se, and many contributors, see the THANKS file.

All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

```
/* Copyright (c) 2004-2007 Sara Golemon <sarag@libssh2.org>
 * Copyright (c) 2005,2006 Mikhail Gusarov <dottedmag@dottedmag.net>
 * Copyright (c) 2006-2007 The Written Word, Inc.
 * Copyright (c) 2007 Eli Fant <elifantu@mail.ru>
 * Copyright (c) 2009-2014 Daniel Stenberg
 * Copyright (C) 2008, 2009 Simon Josefsson
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms,
 * with or without modification, are permitted provided
 * that the following conditions are met:
 *
 * Redistributions of source code must retain the above
 * copyright notice, this list of conditions and the
 * following disclaimer.
 *
 * Redistributions in binary form must reproduce the above
 * copyright notice, this list of conditions and the following
 * disclaimer in the documentation and/or other materials
 * provided with the distribution.
 *
 * Neither the name of the copyright holder nor the names
 * of any other contributors may be used to endorse or
 * promote products derived from this software without
 * specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
 * CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 * BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
 * USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 */
```

OpenSSL (not included in all versions of RocketCake)

The OpenSSL toolkit stays under a double license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts.

OpenSSL License

```
/* =====
 * Copyright (c) 1998-2018 The OpenSSL Project. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
```

* are met:

*

* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.

*

* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.

*

* 3. All advertising materials mentioning features or use of this
* software must display the following acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"

*

* 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
* endorse or promote products derived from this software without
* prior written permission. For written permission, please contact
* openssl-core@openssl.org.

*

* 5. Products derived from this software may not be called "OpenSSL"
* nor may "OpenSSL" appear in their names without prior written
* permission of the OpenSSL Project.

*

* 6. Redistributions of any form whatsoever must retain the following
* acknowledgment:
* "This product includes software developed by the OpenSSL Project
* for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

*

* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.

* =====

*

* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com). This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).

*

*/

Original SSLeay License

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)

* All rights reserved.

*

* This package is an SSL implementation written

* by Eric Young (eay@cryptsoft.com).

* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are adhered to. The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code. The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
* notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in the
* documentation and/or other materials provided with the distribution.
* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the routines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

WebView2 (not included in all versions of RocketCake)

Copyright (C) Microsoft Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- * The name of Microsoft Corporation, or the names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.