This version edited: 14th September 2019

Deploying your visualisations

Documentation for the FSC Identikit

Original development funded by the Esmée Fairbairn Foundation and the Heritage Lottery Fund

1 Contents

2 Introduction

If you develop an interesting taxonomic knowledge-base and you want other people to be able to use it with the FSC Identikit, you need to share the results of your labour somehow. There are three approaches to doing this:

1. sharing locally by copying between computers;
2. deploying to your own website; or
3. hosting it on the FSC Biodiversity project website.

Each of these options is described below.

3 Sharing by copying between computers

The most straightforward way to share your knowledge-based visualisations with one or just a few people is simply to copy it to another computer. Follow the steps below.

1. **Copy the entire tombiovis folder** (e.g. tombiovis-1.2.3) to a memory stick or some other media and copy it from here onto another computer. Alternatively you could zip the folder up and email it but, this could be quite a large file – especially if you have used a lot of images – so be cautious about this.

The above step is similar to the first step you took when installing the Identikit on your computer, i.e. downloading and unzipping the Identikit, only this time it includes the knowledge-base you've created. The remaining steps are exactly the same as installing the Identikit for the first time on any computer.

2. **Install Node** on the new computer.

As you certainly know by now, the above steps are 'one-off' but the following two are required whenever you want to run your visualisations on the new computer.

3. Open a Windows 'command window' in the folder where you installed the Identikit by **double-clicking the start.bat file**.

4. In the command window, type the following command**: npm install.**

An alternative to the above steps is simply to get the owner of the computer onto which you are copying your knowledge-base to follow the Quick-start Guide to install the Identikit on their computer and then to copy just your knowledge-base folder into the kb folder.

4 Deploying to a website

4.1 Simple deployment as a standalone page

Deploying to a website isn't so very different from deploying to a computer.

1. **Make a copy your entire tombiovis folder** (e.g. tombiovis-1.2.3) on your computer and rename it, e.g. to tombiovis-web.

2. **Delete the Documentation folder** from this new folder.

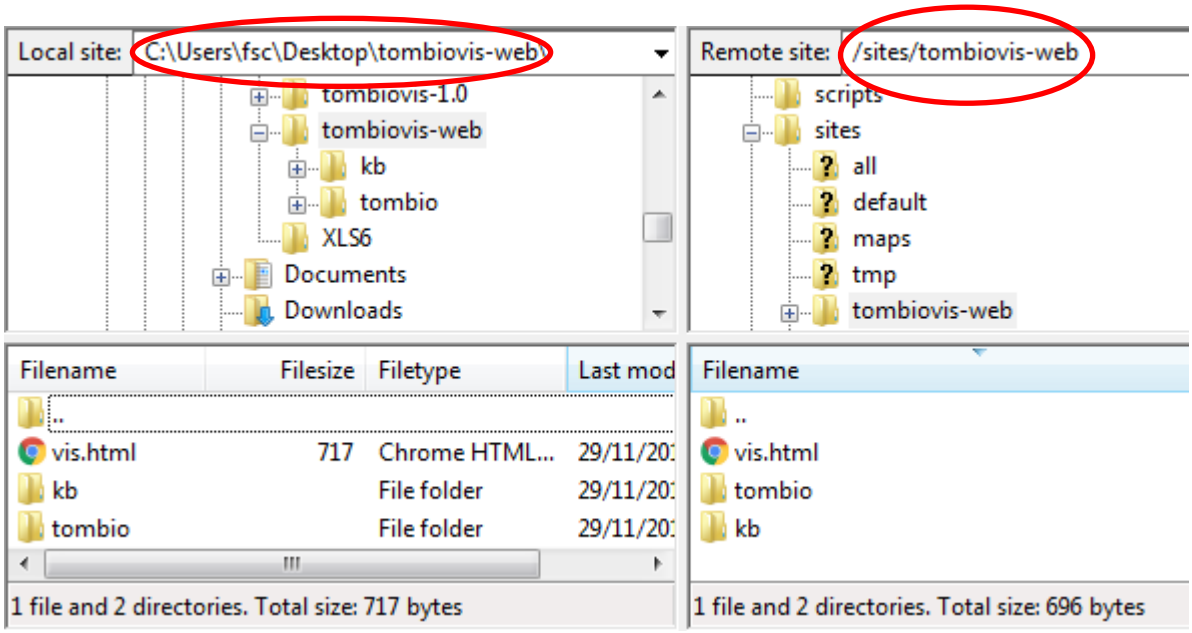
3. **Delete all files in this new folder *EXCEPT* for**:
    a. vis.html
    b. site.css
    c. vism.html *
    d. sw.js *
    e. manifest.json *

If this is *not* a deployment for a mobile key, you can also delete those files above marked with an asterisk. (If this *is* a deployment for a mobile key, read the section titled 'Implementations for mobile use'.)

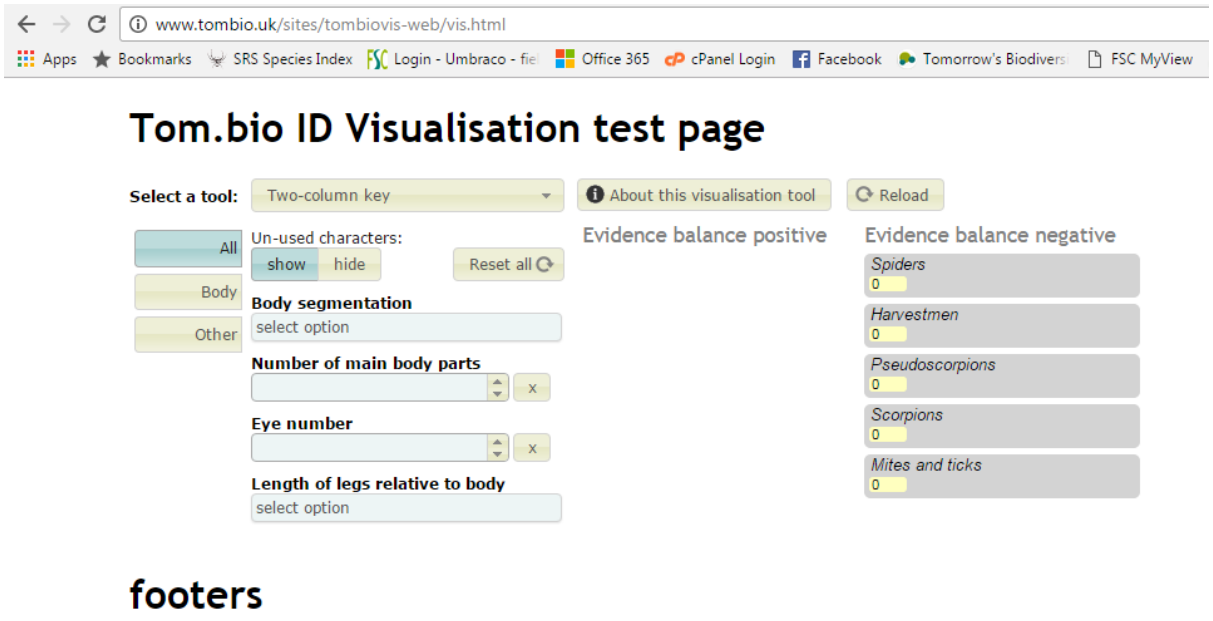
4. **Delete any unwanted knowledge-bases**, e.g. biscuits, from the kb folder.

5. **Copy the new folder** (e.g. tombiovis-web) **to your webserver** below the root directory representing your domain.

If I was doing this on a website – www.tombio.uk – for example, I could copy the tombiovis-web folder to the 'sites' folder shown below in a screenshot from FileZilla.



This would give me a URL for my visualisation of: http://www.tombio.uk/sites/tombiovis-web/vis.html (see below).



If you want to change the appearance of the page around the actual visualisation, e.g. the title **FSC Identikit test** and **footers**, you need to edit the vis.html file. You can also edit the site.css file if you want to override or add any CSS.

4.2 Deploying to a content management systems (CMS)

Deployment to a Content Management System (CMS) such as Drupal, Wordpress, Joomla or Umbraco is a little more involved and what you do will depend on your CMS and its configuration on your site. If you are even reading this section it suggest that you know something about your CMS – if not then you need to seek help from your webmaster – showing them this installation guide.

The contents of the vis.html file are shown below:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv='cache-control' content='no-cache'>
    <meta name="viewport" content="width=device-width,initial-scale=1">

    <title>FSC Identikit</title>
    <script>
        //Set options to tailor page configuration (from 1.6.0)
        var tombiovis = {
            opts: {
                hideVisDropdown: false,
                tools: ["vis1", "vis2", "vis3", "vis4", "vis5"],
                //selectedTool: "vis1",
                //lastVisualisation: "vis1",
                //gui: "guiOnsenUi",
                //ignoreNegativeScoring: false,
                pwaSupress: true,
                toolconfig: {
                    genKeyinput: {
                        selectedGroup: "Structure",
                    },
                    //vis4: {
                    //    subTitleChar: "CommonName"
```

```
                    //},
                    //vis6: {
                    //    keyinput: "keyInputOnsenUi"
                    //}
                },
                //tombiover: "refresh-1",
                checkKB: true,
                devel: true,
                tombiopath: "tombio/",
                tombiokbpath: "kb/biscuits/"
            },
        }
    </script>
    !--<link rel="manifest" href="/manifest.json">-->
    <!--ES6 polyfill - new for 1.7.0-->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/core-
js/2.5.6/core.min.js"></script>
    <!--Local site CSS-->
    <link rel="stylesheet" type="text/css" href="site.css">
    <!--Change the path to load.js & load.css to match the installation environment-->
    <link rel="stylesheet" type="text/css" href="tombio/min/css/load.min.css">
</head>
<body>
    <h1 id="tombiod3-header">FSC Identikit test</h1>
    <div id="tombiod3"></div>
    <div id="tombiod3-footer" style="width: 100%">footers</div>

    <script type="text/javascript" src="tombio/load.js"></script>
</body>
</html>
```

The most important parts are highlighted in yellow and these are the bits that need to be implemented in an appropriate place on your CMS page. The *tombiopath* and *tombiokbpath* options must be edited appropriately to reflect the installation location of the Identikit and knowledgebase folder. All the other options are optional (!) and described elsewhere in this document.

The script tag that includes the core.min.js file is necessary if your visualisation is likely to be used by people with browsers that don't support ES6 (ECMAScript 6 – also known as Javascript 6). It is a 'polyfill' library that will allow the Identikit to work correctly with these browsers. Version 1.7.0 of the Identikit started to use some ES6 features, such as promises, and therefore requires browsers that support ES6.

The 'load.js' script is responsible for loading and starting the Identikit and it replaces the contents of the `<div id="tombiod3"></div>` tag with all the dynamically created visualisation markup – so place this tag where you actually want the visualisation to appear within your page.

The `<h1 id="tombiod3-header">` and `<div id="tombiod3-footer">` tags are optional – the Identikit replaces them with the name of your knowledgebase (from the title metadata tag) and the citation for the knowledgebase respectively.

4.2.1 Other considerations when hosting on CMS pages

If you host Identikit visualisations within the context of a CMS page it is quite likely that you might have to make some adjustments to the CSS styling rules that are incorporated within the Identikit. Sometimes the CSS that creates the general look and feel of your CMS website affects the look and

feel of the Identikit visualisations in unexpected ways and you might have to adjust or add further CSS to the Identikit stylesheets to fix this.

Most of the Identikit's CSS currently resides in the file 'tombio/tombiovis.css' although some visualisations have their own CSS in their corresponding sub-folder. There is also a CSS file for the taxon selection tool that appears in a couple of the visualisations ('tombio/taxonselect.css').

For convenience, the vis.html file references a style file, site.css, in the top-level folder which is a good place to enter specific CSS for your implementation.

If you are configuring your deployment to make a mobile-first multi-access key, hosting on a CMS page may be problematic (see section below – Implementations for mobile use).

4.2.2 Caution when deploying on CMS sites or within other frameworks

Aside from the possible clashing of CSS styles as described in the previous section, a potentially more serious problem involves incompatibilities between Javascript libraries required for the Identikit and those packaged as part of the CMS or other frameworks.

For example there is a problem on Drupal 8 sites. When viewed in an iPad, Drupal 8 loads fastclick.js which interferes with the jQuery UI Dropdown lists used by the Identikit and is very difficult to avoid or workaround.

Because of such difficulties, we recommend implementing pages built with the Identikit as standalone pages, even in CMS and other framework environments.

5 Hosting on the FSC Biodiversity project website

If you don't have a website that you can deploy to, you can ask the FSC Biodiversity project team to host the visualisation for you.

FSC will consider hosting on a page that makes it clear that they are only providing a hosting service and that the knowledge-base, and responsibility for it, is yours.

FSC will only ask you to confirm that nothing they are hosting on your behalf infringes anyone's copyrights or intellectual property rights.

6 Options for tailoring your deployment

There are a number of javascript options that can be specified on the main *tombiovis* javascript object to modify the way the Identikit behaves. The *tombiovis* object is created in a script tag on your top-level html page.

In vis.html, it looks like this:

```
<script>
    var tombiovis = {
        opts: {
            hideVisDropdown: false,
            tools: ["vis1", "vis2", "vis3", "vis4", "vis5"],
            //selectedTool: "vis1",
            //lastVisualisation: "vis1",
```

```
            //gui: "guiOnsenUi",
            //ignoreNegativeScoring: false,
            pwaSupress: true,
            toolconfig: {
                genKeyinput: {
                    selectedGroup: "Structure",
                },
                //vis4: {
                //    subTitleChar: "CommonName"
                //},
                //vis6: {
                //    keyinput: "keyInputOnsenUi"
                //}
            },
            //tombiover: "refresh-1",
            checkKB: true,
            devel: true,
            tombiopath: "tombio/",
            tombiokbpath: "kb/biscuits/"
        },
    }
</script>
```

This has all the possible top-level options (though some are commented out in the default vis.html file. Each of the options is described below.

6.1 The `hideVisDropdown` option

If this option is present and set to `true`, then the drop-down visualisation selection list is not displayed on your visualisation page. This is only really useful if you either want to present only a single visualisation or you are using API features of the tombiovis object to provide other means for the user to change the visualisation (see the tombiovis API section).

6.2 The `tools` option

Use this option to specify which tools to include in the drop-down list. The currently generally available tools are:

- vis1 – the two-column key
- vis2 – the single-column key
- vis3 – the side-by-side comparison tool
- vis4 – the full details tool
- vis5 – the circle-pack key.
- vis6 – the mobile key.

These values are supplied as strings in an array, e.g. `["vis4", "vis3", "vis3"]`. If no value is supplied, then the Identikit uses the value `["vis1", "vis2", "vis3", "vis4", "vis5"]`.

6.3 The `selectedTool` option

Use this to specify which tool should be automatically selected when the visualisation starts. This can take any of the values specified in the array of visualisations (which can be specified with the tools option), e.g. `"vis2"`. *However*, the selectedTool option can also take on other values shown below:

- visInfo – displays information on the Identikit (corresponds to 'About FSC Identikit' item on the visualisations drop-down menu).
- kbInfo – displays information on the knowledge-base (corresponds to 'About the knowledge-base' item on the visualisations drop-down menu).
- currentVisInfo – displays information on the 'current visualisation' (name on the visualisations drop-down menu changes depending on the last tool selected).
- tombioCitation – displays a page that explains how to cite the Identikit, knowledge-base and last used visualisation.

If the value is not specified, then the first value from the array of all available visualisations (which can be specified with the tools option) is used. When this option is set to either of the values 'currentVisInfo' or 'tombioCitation', the lastVisualisation option must also be set (see below).

6.4 The `lastVisualisation` option

The Identikit tools that display information on the current visualisation (option 'currentVisInfo') and citation information (option 'tombioCitation'), expect a current visualisation to have been set. But when either of these is set as the first tool with the selectedTool option, these tools don't know what the current visualisation – the last used visualisation – is. In such cases, the lastVisualisation must be used to specify one of the visualisations, e.g. `"vis3"`.

6.5 The `gui` option

Use can use this option to tailor the high-level GUI for the Identikit. When not supplied, the default value for this is 'guiLargeJqueryUi'. The alternative value is 'guiOnsenUi' which switches the top-level GUI to one suitable for small-format devices. Use this option if you are configuring a deployment for a mobile key ("vis6").

6.6 The `ignoreNegativeScoring` option

The ignoreNegativeScoring option (introduced v1.8.0) to instruct the Identikit *not* to score non-matching characters negatively. The default value for this option (taken if not specified) is false. When false, the default behaviour of Identikit is to include negative scoring for taxa with character states that don't match user character state input. But if the value of this option is set to true, negative score values are not included.

6.7 The `pwaSupress` option

This should be set to 'true' unless you are configuring a deployment for a mobile key ("vis6"), in which case either comment it out, or set it to 'false'. It prevents Identikit from starting the PWA service worker.

6.8 The `toolconfig` option

Use this to specify configuration options specific to particular visualisations or the character input controls. Each tool has a configuration subsection in this structure. All those currently available are documented below.

6.8.1 The character input control options (genKeyinput)

6.8.1.1 The `selectedGroup` option

The selectedGroup option allows you to specify a character group that should be selected by default in the characters input control. The value is a string which must be a value specified in the Group column of the characters tab of the knowledge-base. If no value is specified, then the default – 'All' – tab is selected.

6.8.2 The 'Full taxon details' visualisation (vis4)

6.8.2.1 The `subTitleChar` option

This option allows you to specify the name of a character whose value will appear in parentheses after the taxon name to form the title of species account pages in the 'Full taxon details' visualisation.

6.8.3 The 'mobile key' visualisation (vis6)

6.8.3.1 The `keyinput` option

This option can be used to specify a mobile-first character input control rather than the default. If the mobile key visualisation is used, the value of this option should be set to "keyInputOnsenUi".

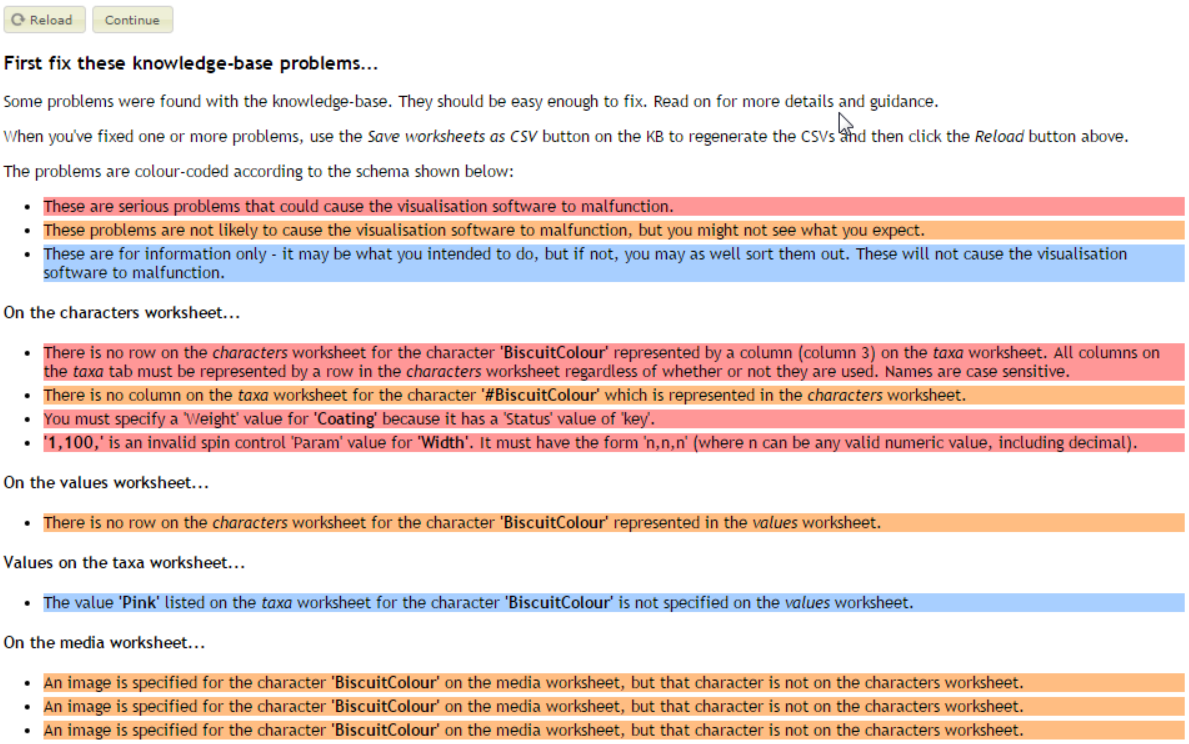
6.9 The `tombiover` option

Setting this option to a new unique string, of any value, forces that string to be used as a query string parameter when resources such as javascript, css, image or knowledge-base files are loaded. For example if the value is set to 'refresh-1' then the core javascript resource 'visP.js' will be loaded with the following relative URL: `tombio/visP.js?ver=refresh-1`

This can help overcome browser caching problems if you, as a developer, change some of these resources and want your users to use the new resources – not those from their browsers cache.

6.10 The `checkKB` option

Set this option to "true" if you want to initiate the knowledge-base validity checks and reporting. It's best to set this to true when you are developing a knowledge-base but either remove it or set it to false when you deploy your visualisation to speed up initialisation for your users. Typical output from the validity checks is shown below. Setting this flag also adds the 'Check media files' item to the 'Select a tool' drop-down list which allows you to report on whether or not media files referenced by the knowledge-base can be found.



6.11 The `devel` option

Set this option to "true" if you want to use non-minified version of javascript and css file resources. If you are developing software, have this set to true so that your debugger reports the correct line numbers in problem code. But on production sites, set it to false, or remove it, to speed up initialisation.

6.12 The `loadWait` option

Set this option to "true" if you want to instruct the load.js module to defer execution, after loading, until instructed (by the `tombiovis.startLoad()` API call). This enables hosting pages to load without the overhead of loading and initialising the Identikit until required. (This option is not illustrated in the vis.html example page.)

6.13 The `tombiopath` option

This is a mandatory option! In other words you must specify this for the Identikit to load. It specifies the path – relative to the root folder of your website – where the core software is stored.

6.14 The `tombiokbpath` option

The second mandatory option! This indicates where the knowledge-base folder is located relative to the root folder of your website.

6.15 The `loadCallback` option

You can use the loadCallback option to specify a callback function that will be called when the main tombiovis modules are loaded (but before any particular tools is loaded). (This option is not illustrated in the vis.html example page.)

7 Implementations for mobile use

Version 1.8.0 introduced a new visualisation – 'vis6' – that is a 'mobile-first' multi-access key which is designed to work well on small format devices. To use this visualisation you will need to set some of the top-level options for tailoring your deployment (see previous section). Version 1.8.0 also includes a complete change to the architecture of Identikit to enable its deployment as a 'Progressive Web App' (PWA). This means we can now do several things to improve the performance of Identikit resources on mobile devices in the field. Two major new features are:

1. The ability to cache resources, like the knowledge-base, images etc when in range of wifi and use the resource in the field where no internet connection is available.

2. Allow users who are using supported browsers (e.g. Chrome on Android) to install an Identikit resource so that it behaves much like a native app.

This section of the document discusses a few issues relating to those features.

7.1 Example HTML page for a mobile-first implementation

As well as the sample top-level page 'vis.html' which demonstrates a basic implementation of the Identikit tools through an interface optimised for large-format devices such as a laptop, there is also a sample top-level page 'vism.html' which demonstrates an implementation of the Identikit tools through an interface optimised for small-format devices like mobiles.

The previous section has explained the options required to set this up, but if you are going to be setting up an ID resource that you wish to deploy primarily as a mobile-first resource, base your top-level page on 'vism.html'.

7.2 Manifest file for mobile implementations

The manifest file – 'manifest.json' – found in the root folder of Identikit facilitates 'installing' Identikit resources as apps on mobikle devices, enabling launching from an icon on the homescreen for example.

The 'manifest.json' delivered with Identikit contains information that specifies what icon and text to use for this (among other things). Identikit is delivered with a manifest file with some general Identikit branding, but you may want to change this to suit your own resource.

The manifest file delivered with Identikit is shown below:

```
{
    "name": "FSC Identikit",
    "short_name": "Identikit",
    "icons": [
        {
            "src": "tombio/resources/icon-128x128.png",
            "sizes": "128x128",
            "type": "image/png"
        },
        {
            "src": "tombio/resources/icon-144x144.png",
            "sizes": "144x144",
            "type": "image/png"
        },
```

```
        {
            "src": "tombio/resources/icon-152x152.png",
            "sizes": "152x152",
            "type": "image/png"
        },
        {
            "src": "tombio/resources/icon-192x192.png",
            "sizes": "192x192",
            "type": "image/png"
        },
        {
            "src": "tombio/resources/icon-256x256.png",
            "sizes": "256x256",
            "type": "image/png"
        }
    ],
    "start_url": "vis.html",
    "display": "standalone",
    "background_color": "#3E4EB8",
    "theme_color": "#2F3BA2"
}
```

You must provide at least the `short_name` or `name` property. If both are provided, `short_name` is used on the user's home screen, launcher, or other places where space may be limited and `name` is used in the app install prompt.

The icons array specifies the path to icon images which can be used, in various situations, to represent your app, e.g. on the smartphone's homescreen. Normally you will use the same image represented at different sizes here. The `src` property specifies the path to the icon image and the `sizes` property indicates the size of it.

The `start_url` property tells the browser where the app should start when it is launched – you shouldn't need to change this. You can also leave the `display` property set to 'standalone' to make your app look and feel like a standalone native app.

The `background_color` property is used on the splash screen when the application is first launched and `theme_color` sets the color of the tool bar – you may wish to tailor these to your app.

7.3 Enabling the service worker

The service worker – 'sw.js' – is the key resource that enables the offline use of the Identikit. For sites that deliver an Identikit resource from a standalone html file (like 'vism.html') you should not need to rename, move or otherwise mess with this file. But to actually make use of the service worker's caching capabilities, your site needs to be served over a secure connection, i.e. a website delivered over https (rather than http). Identikit resources delivered from websites over a non-secure connection will still work, but they will not be able to make use of the service worker capabilities of caching resources for offline use.

If you are hosting an Identikit resource on a CMS hosted page, using the service worker can be problematic. You may have to deploy the 'sw.js' file in the root of your website. There may be other problems to overcome. If you can deploy via a standalone HTML page, this will be much easier.

You must also either remove the `pwaSupress` option in your top-level html page, or set its value to false, in order for the Identikit to use the service worker.

8 API features of the tombiovis object

8.1 Switching tool

As of version 1.6.0 there is an API (Application Programmers Interface) element on the tombiovis object that allows you to switch between visualisations without using the default visualisation drop-down list. This is useful if, for example, you are implementing the visualisation within the Identikit of a wider website and you want that website's GUI to be able to change the visualisation tool.

The single API call currently available is: `tombiovis.visChanged(tool, lastVis)`

The first argument to the visChanged function is to specify the tool to engage. It ca take any of the values that can be specified by the selecteTool top-level option described in the previous section. The optional second argument is required if you are using either of the values 'currentVisInfo' or 'tombioCitation' for the first parameter and it is directly equivalent to the lastVisualisation top-level parameter and can take any of the values allowed for that option.

8.2 Starting tombiovis load

When the top-level option loadWait is set to `true` the load.js module will not automatically load the rest of the core Identikit software. When this is the case, you need to start the load by making the following API call:

```
tombiovis.startLoad()
```

9 URL parameters

A number of URL parameters can be used to specify options. For example the URL parameter – *selectedTool* – can be used on the link which invokes your visualisation to select a particular tool on initialisation. The example shown below would start a visualisation with the *Circle-pack key* (vis5) selected.

```
http://www.tombio.uk/harvestmanvis?selectedTool=vis5
```

Some of the visualisations have right-click context menu options, e.g. 'Set URL for full details view', that construct URLs with parameters that take you straight to the same tool configured in exactly the same way as when you used the menu option. An example is shown below:

```
http://www.tombio.uk/vis.html?selectedTool=vis4&taxon=Milk%20Chocola
te%20Digestive&opts=image-text&imgi=2&txti=1
```

This is starting the page vis.html with the 'vis4' (full details) visualisation and the taxon 'Milk Chocolate Digestive' selected. Furthermore it is reselecting the very image and text files and display options selected when the URL was created. You can use URLs created in this way to create links to specific visualisations that are displaying exactly what you want people to see.