Mouin 1.0

Create native mobile apps based on your web site

1. Introduction

1.1. Introduction

The Mouin system is designed to allow you to create native mobile apps for your web site as quickly and simply as possible. Using it, you can build a fully custom native mobile app in a matter of hours. Not only that, but you will get native apps for all the major mobile platforms. Currently, iPhone, Android, Blackberry and Java/J2ME are supported, with more coming in the near future.

To apply the Mouin system to your web site, you just need to insert extra Mouin-designed CSS style tags in your web site to describe how it should be presented on a mobile phone. Then, you can use the Mouin web site to automatically build the apps based on your tags, and Mouin will also submit the apps to the corresponding app store for each mobile platform.

With the Mouin system, you don't need to learn Objective C, Java, C#, or any other software development technology. You just need to learn the extended CSS tags, insert them in your web site, and maintain them in the future as you change the web site. You don't even need to be a software developer, with some web design skills, and learning the Mouin CSS extensions, anyone is ready to build native apps for all major mobile platforms.

1.2. The Mouin System

The Mouin system consists of three parts:

- The Mouin-designed CSS extensions
- The standardized native app for each supported mobile platform
- The Mouin App Builder and Submitter (available at **mouin.com**)

The **Mouin CSS extensions** are a specification designed to add to traditional HTML/CSS content the description on how to present the content on a mobile device. They are simple CSS attributes that will allow you to tag the content on your web site. Since it's a simple CSS extension, you can easily integrate it in your existing web design, CMS or web templates. They are embodied as "vendor-specific CSS extensions", a format of standardized HTML/CSS, so if your site validates with W3C today, it will keep validating after you add the new styles. That is, if your site is one of that few percent of web sites that is fully standard compliant!

Here is some sample Mouin CSS styling:

```
.entry {
  -mouin-style: item image("/bullet.gif") yellow on(black);
}
```

The **standardized native apps** are the actual code that makes things work. These are fully native client apps that we have developed, which can read the content and the Mouin CSS

tags from your web site, and turn that into a fully native UI, using platform-specific native controls and conventions.

These apps, with the name, icon and branding changed to your web site's, will actually be the app that your end-users will download and use on their mobile phones.

Finally, the **Mouin App Builder and Submitter** is a web application available at **mouin.com**, where after signing in, you can instruct our system to customize the standardized native mobile apps, and tell it to build them and submit them to the respective app stores. Once you have added the Mouin CSS to your web site, you will go to **mouin.com** to create and manage the app creation and submission process. Other value-added services are available here too.

1.3. Compatibility

The Mouin CSS extensions are designed to be universal, describing the mobile adaptation of the web content in fully general terms, rather than being platform-specific. This makes the design well-adapted to future evolution. Currently, Mouin supports a number platforms, but this will be extended to cover all major mobile and related platforms.

If your web site uses Javascript, this will affect compatibility with the Mouin system. Some platforms are better adapted to Javascript processing than others. Mouin depends on each platform's compatibility, and this makes the Mouin system provide different levels of Javascript-compatibility across different platforms. On platforms like iPhone and Android, the Javascript on your website can be processed quite well, and even if all the content is generated on the fly with inline Javascript, it should work fine. On a more limited platform like a Blackberry or J2ME phones, only the clean HTML and CSS content will be processed. You might want to generate a Mouin-specific version of your website not using Javascript, to provide full compatibility across all platforms, and this will also help get better performance across the board.

2. General Concepts

2.1. Design principles

The main goal was to design the simplest CSS extensions so that native multiplatform mobile apps can be easily created from existing web content.

The design builds upon and “fits in” with existing HTML/CSS conventions and traditions, staying compatible, such that the whole system will be familiar and easy to learn for current web designers and developers.

The extensions are done in a way that they keep existing web content standard-compliant. To this effect, Mouin mobile styles are inserted in CSS as a vendor-specific style (similar "-moz-border-radius" in Mozilla browsers). Thus, they always look like “-mouin-style”, “-mouin-settings”, etc...

2.2. Overall design

The main way the Mouin system works is making all HTML content hidden by default, and giving the web designer the tools to mark which HTML elements to show and how to show them, using the extended CSS “-mouin-style” attribute.

Basically, you can set the “-mouin-style” CSS attribute on the HTML elements on your web page so that they will be treated especially on the mobile client. The possible values of this attribute allow full control of how the element must be shown.

“-mouin-style” is a complex CSS attribute that can contain a lot of information in a number of properties, but the main property is called “type”. Let’s see an example, simplified web page:

```
<html>
 <head><title>Online Shope</title></head>
 <body>
 <h1><img src="logo.gif">Welcome to the online shop!</h1>
 <a class="main" href="/catalogue.html">Product Catalogue</a><br>
 <a class="main" href="/support.html">Customer support</a>
 </body>
</html>
```

The goal here would be to create a mobile header from the H1 logo, and to create two direct access items for each of the links. This is how the Mouin CSS styles have to be added:

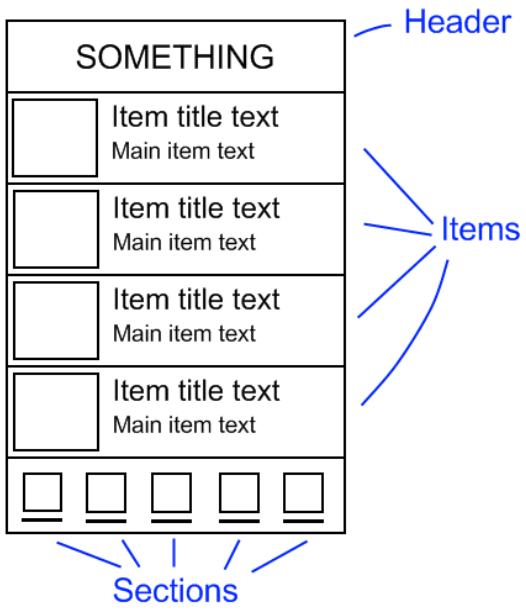
```
h1 img { -mouin-style: header white on(blue) text("Online Shop"); }
a.main { -mouin-style: item image("/bullet.gif") blue on(white); }
```

As you see, it is quite simple to define the Mouin mobile style. Mouin property values are determined by the property name followed by the value between parentheses. The first property ("header", "item") is actually the value for the "type" property, but the "item" property name can be omitted because it is inferred from the value. The color refers to the text color, and the "on" property refers to the background color.

The main Mouin element types are **header**, **item**, **section** and **article**. They are described in detail below. You can build very interesting native mobile applications using just this type of controls, but more will be coming in the future.

2.3. Mouin mobile applications

This are the main parts of a Mouin mobile application:



The application always shows a header area, a main content area, and a quick-access area to navigate to the main sections of the app. The main content area can take three main forms: a list of items as shown on the diagram, a block of rich-text and images content (called an "article"), or a form.

To build this type of content, you only need to mark HTML elements to act as the header, as sections, as items to be displayed as a list, or as an article or form.

HTML elements with no Mouin style have –mouin-style-type set to "none" by default, and they are just not displayed by the client.

2.4. Mouin styles, properties and shorthand

All Mouin items can have a variety of pre-specified properties, each one with an assigned value. Some of the properties are "composite", having further sub-properties. Each non-composite property or subproperty can be a color, a string, a URL, or one of several predefined values.

Each Mouin element type will use the specified property values to control its display, according to the element's function. Some properties are used by most elements (**image** or **text**), while other properties are particular to only one or several element types (for example, the **href** property is not used at all on the **header** element).

Element properties can be set individually with their full name, or inside the –mouin-style definition. Also, since they often go in shorthand groups, they can be set individually, or in tandem with others. See these equivalent examples on how to set the header's text:

```
/* Format A */
#header {
  -mouin-style-type: header;
  -mouin-style-text-content: "Hello";
}

/* Format B */
#header { -mouin-style: header text("Hello");

/* Format C */
#header { -mouin-style: header text-content("Hello");
```

Each of formats A, B and C mean exactly the same thing and assign exactly the same values. Formats B and C take advantage of the fact that "header" is autodected as the type property, and format B further uses the fact that a string found inside the text specification is taken to mean the content.

This is an extension of the classic CSS "shorthand" mechanism, as used in "border: 1px solid black;", expanded such that it will cover the more complex needs of full mobile-UI specification.

2.5. Mouin element types

The main Mouin element types are **header**, **section**, **item** and **article**. Here is a description of each:

HEADER

The header is always present at the top of the screen. It provides branding and context. The main elements of the header are the image, the title text and the background.

- **image**: Any GIF or PNG image can be used. The client app will resize it to the required resolution (this is necessary as mobile devices can vary greatly in resolution). We suggest you provide a high-enough resolution image to look great on any device. Images with a transparent contour are recommended for best results across the board.
- **on**: this property configures the aspect of the background. The background color can be set, and it can be rendered either flat or using a gradient that the client app creates around the provided color. The subproperties are **color** and **type**, where type can be **flat** or **gradient**.
- **text**: The main header text and color, in the **content** and **color** subproperties. Please keep the text short to allow best display on all devices.
- **border**: this property contains a subproperty **color** that allows configuring the color of the line separating the header from the main content, and a subproperty **type** which can be **solid** or **transparent**.

The property names for the text-content and text-color can be omitted, a string will be taken as the text and a color will be taken as the text-color. As always, the element type can be given directly too, as the keyword is recognized. Background-type is assumed to be gradient by default.

A header may contain either just an image or just text, in both cases the content will be centered. If both are specified, they will be shown one alongside the other, the image first, and the full pair will be centered. Please watch out for too large image-text combinations which may have to be cut.

Example header:

```
#header {
  -mouin-style: header image(h.png) "Hello" white on(gradient red);
}
```

SECTION

Sections are main areas of the mobile app or web site. Each one is identified by a name and image. The target URL and display attributes like colors can be specified separately. Direct accesses to sections are permanently displayed at the bottom of the screen on large touch-screen devices. Other device types may provide access to sections from a submenu. There is no maximum amount of sections, but most mobile platforms, and usability principles, favor having a small amount. Usually, "Home" is the first section entry, providing a clear reference point throughout the app experience.

Here are the available Mouin style properties for elements of type **section**:

- **image**: Any GIF or PNG image can be used. The client app will resize it to the required resolution. We suggest you provide a high-enough resolution image to look great on any device.
- **on**: this property configures the aspect of the background. The background color can be set, and it can be rendered either flat or using a gradient that the client app creates around the provided color. The subproperties are **color** and **type**, where type can be **flat** or **gradient**.
- **text**: The text and color, in the **content** and **color** subproperties. Please keep the text short to allow best display on all devices. Default value for text-content is the HTML text of the element.
- **href**: the href property contains the URL of the page to load and visit when the section is chosen. The **hreftype** property will control whether it is displayed using Mouin styles or using the standard browser. A fragment identifier can be used and it may switch to the target subpage if it has –mouin-style-page-type enabled. The link may just be a fragment ("#subsection"), in which case no load will be necessary and the Mouin client will jump to the given subpage.
- **image-auto**: if set to extract, it will find the first child **img** HTML element and take its contents.

Example sections spec:

```
.mainmenu a
{
  -mouin-style: section image-auto(extract) white on(gradient red);
}


<div class="mainmenu">
  <a href="/home.html"><img src="home.gif">Home</a>
  <a href="/news.html"><img src="news.gif">News</a>
  <a href="/about.html"><img src="about.gif">About</a>
</div>
```

ITEM

Items, or list items, as seen on most native mobile apps, are the main mobile interaction mechanism. Items are arranged on a list, and they serve both to directly display information and to allow clicking to get to extra information they target.

Mouin allows the following parts in items:

<table>
<tr><td rowspan="5">IMAGE</td><td>TOPNOTE</td><td rowspan="5">RIGHTIMAGE</td></tr>
<tr><td>TITLE</td></tr>
<tr><td>MIDNOTE</td></tr>
<tr><td>TEXT</td></tr>
<tr><td>BOTTOMNOTE</td></tr>
</table>

Each part corresponds to an attribute. **image** and **rightimage** are image-type properties, and **title**, **text** and the three ***note** are text-type properties.

Here are the details on the supported properties:

- **image** and **rightimage**: each property controls the image displayed on each side. It allows a URL for a supported image resource, and the image is resized to fit the device.
- **title, text**, **topnote**, **midnote**, **bottomnote:** these are text properties, each supporting a default **content** subproperty and a **color** property.
- **on:** this property allows specifying the background color and style, as in the header
- **href**: this property contains a string that designates the target URL that is visited when clicking anywhere on the item.
- **hreftype**: this property is either **internal** or **external**, controlling how a given URL should be followed: either using the Mouin display, or using the external browser. Default is **internal** for URLs in the same domain and subdomain, and **external** for URLs where either the domain or even just the subdomain is different.
- **border**: this property contains a subproperty **color** that allows configuring the color of the line separating the each item from the next.
- **text-auto**: if set to **extract**, Mouin will extract the first few words from the contained text and use this for the item text-content. If set to **full**, Mouin will extract the whole text content. Default value is **none** which leaves text empty if not specified explicitly.
- **image-auto**: if set to **extract**, Mouin will extract the first contained image from the item's children, and will use this for the item image. Default value is **none**.

Example items spec:

```
.maincontent ul li {
  -mouin-style: item image-auto(extract) white on(gradient black);
}
```

```
<ul class="maincontent">
  <li><a href="/home.html"><img src="home.gif">Home</a>
  <li><a href="/news.html"><img src="news.gif">News</a>
  <li><a href="/about.html"><img src="about.gif">About</a>
</div>
```

ARTICLE

An “article” is a piece of rich HTML content which is displayed for easy reading. It contains rich text (text with different fonts and colors), with links and images interspersed in-between.

The Mouin system automatically converts the full HTML content, extracting the main attributes, and displaying in mobile-optimized, readable way.

An “article” item doesn’t need or use any extra properties, as the conversion is automatic. The only manual tool to control this automatic conversion is removing pieces from the content. This is done by setting the Mouin property “visibility” to value “hidden”:

```
<div class="entry">
  <p>Whatever it is <a href="/is.html">happening</a>.</p>
  <img src="/image.gif">
  <p class="note">Note: stolen</p>
</div>

...

.entry { -mouin-style: article; }
.note { -mouin-style: visibility(hidden); }
```

2.6. Subpages

It is often the case that different parts of a web page may have to be shown in separate views in a mobile device. The Mouin CSS extensions allow tagging a part of a given page as a separate subpage, such that its contents don't appear when viewing the main page, and such that it is possible to "navigate" to this section as if it were actually a separate web page.

The way a part of a web page is marked to be presented this way is to set the **–mouin-style-page** property to **enable**. This can be done on Mouin elements of type **article**, but also to any HTML element with no special mouin element type (**-mouin-style-type** set to **none**). This last case makes this element just a grouping container.

The **body** element of an HTML document has the **page** property set to **enable** by default. It can't be turned off.

It is possible to create a link to this subpage from anywhere by using a classic HTML "fragment" which refers to the HTML **id** of the element that is marked as **page** (as **<a href="#subpage_id"**).

It is often the case that you would want to move a given section of the page to a separate **subpage** and create a link in place where the section was present. It is possible to do this using the **–mouin-insert** style. The Mouin client will automatically assign the **href** attribute to the created subpage's. You can use either an element of type **item** or an element of type **section** for this purpose.

The **page** property is actually a complex property having the subproperties **type** (**enable** or **disable**), **update** (enable or disable) and **title** (a string). If **update** is enabled, the client app will present a way for the user to refresh the content easily. The **title** may be used to enhance the content presentation on some platforms.

2.7. Default property extraction

When you add "-mouin-style: item;" to an HTML element, some default property values are extracted. Mainly:

- **text**: it's generated from the elements text content.
- **image**: if this is applied to an **IMG** element, its **SRC** attribute is taken. If other, the first IMG child of the current element is searched for.
- **href**: on an **A** element, its **HREF** attribute is taken. If other, the first **A** child of the current element is searched for.
- **text-color**: its text-color is taken.
- **background-color**: its background-color is taken.

Default values can easily be overridden by explicitly assigning empty values

2.8. Piecewise Mouin element composition

Especially when adding CSS to an already existing web site, you might find that you want to combine information from several sources to create the Mouin native mobile elements. The image, text and link may be in separate HTML elements, and you may need to combine them all in to build the right item. Let's see a sample case, suppose you have the following HTML code:

```
<div class="entry">
  <h2>Sales figures</h2>
  <a href="/sales.htm">
    <img src="/sales.gif">
    Details of recent sales
  </a>
</div>

<div class="entry">
  <h2>Extra info</h2>
  <a href="/extra.htm">
    <img src="/extra.gif">
    View extra information
  </a>
</div>
```

You would probably want to generate two Mouin **items**, displayed in a native list. The problem here is that, even if you can easily assign the rules to the **div** elements of class "entry", you need to extract the title, text, image and link from separate elements. The Mouin system provides a special Mouin attribute designed to do this, called **–mouin-set-prop**. You apply this attribute to the element you want to extract the information from, and configure what property you want to apply to the related Mouin element. For example, you would apply –mouin-set-prop to the **img** HTML element to extract the image, indicating that you want to set the **image** property.

Here is how you would need set the CSS styles here to build the items piecewise:

```
.entry { -mouin-style: item; }
.entry h2 { -mouin-set-prop: title; }
.entry a { -mouin-set-prop: href text; }
.entry img { -mouin-set-prop: image; }
```

2.9. Explicit element creation

When building a Mouin mobile app, you may want to insert extra content that is not present in the original HTML. Mouin provides a specific syntax to allow creating Mouin elements in a compact form. All types and properties of Mouin elements are supported by this syntax. This may be especially useful when creating a mobile application for a web site the source of which you can't or don't want to modify.

You can image this functionality as if it created new, empty and anonymous HTML elements and inserted them in the page's DOM tree. These elements only have Mouin properties. You can create separate elements, or you can create full trees of them.

The syntax is composed by the **–mouin-insert** attribute, followed by a list of elements to create, each element between parentheses. See:

```
body {
 -mouin-insert: (header  "Abc"  image("a.gif") red on(black))
                (section "Home" image("home.gif") href(/home.html))
                (section "News" image("news.gif") href(/news.html));
}
```

You can also create a tree by inserting extra elements between parentheses inside a given one. A sample:

```
body {
 -mouin-insert: (header  "Abc"  image("a.gif") red on(black))
                (subpage
                   (item "Item1"  image("1.gif") href(/1.html))
                   (item "Item2"  image("2.gif") href(/2.html))
                 );
}
```

2.10. Implementation details

For performance reasons , the Mouin client only reads CSS refered in links with media including "mouin" (not in regular CSS links). Be sure to add media “mouin” if a given CSS file contains Mouin styles.

Mouin CSS can be inserted inline in the HTML document, although not in the "style" attribute, but in the "data-mouin" one, for current platform limitations. This will hopefully change in future versions, but data-mouin will be supported indefinitely.

HTML, CSS and JS support are platform dependent. iPhone and Android are quite compatible with current web standards, due to the underlying Webkit engine. Blackberry and Java implementations of the Mouin client app are more primitive and don’t support Javascript at all.

Many attributes can be applied to any HTML element, but some global ones have to be applied to the BODY of the document.

2.11. App definition attributes

The Mouin Builder and Submitter connect to the URL to build and submit the app to the relevant app stores and markets. The general information of the app (name, icon, etc...) are read from here. There are specific Mouin attributes to describe this content. These are read just once, when the app is built and submitted. In the future, this information may be accessed more frequently, so it would be good to keep it up to date. Here is a sample of the information for a given app:

```
body {
  -mouin-app-name: "Socks";
  -mouin-app-icon: url(http://mysocks.xyz/socks.png)
}
```

It is recommended to provide a high-resolution icon (512 x 512 pixels), which Mouin will resize to the appropriate resolution for each platform and device.

3. Mouin CSS Reference

3.1. Data types

Mouin properties and styles can be of simple or complex types. These are the simple types:

- **String**: defined between double quotes. Samples: **"abc"**, **"Hello world"** or **""**.
- **Color**: standard CSS color spec: either # followd by an RGB triplet in 6 or 3 digits, or one of the standard CSS colors. Examples: **#ff0000**, **#ff8** or **white**.
- **URL**: for elements involving links, this is the type of the **href** property. Also for **image** properties, the URL from which to get the image
- **Enumerated**: several Mouin properties allow a fixed set of possible values, called enumerations. Here are the main enumeration types and their possible values:
  - **style-type**: `none, header, section, item, article`
  - **background-type**: `transparent`, `flat`, `gradient`
  - **border-type**: `transparent`, `solid`
  - **href-type**: `internal`, `external`
  - **text-auto**: `none`, `extract`, `full`
  - **image-auto**: `none`, `extract`

Complex properties are sets of subproperties (either simple or complex). There is a set of allowed subproperties for each property. Values can be indicated with the name of each subproperty followed by a value between parentheses, possibly recursively. In each property-context, there is a default subproperty for each simple type, such that if specified without a subproperty identifier, it is assigned to its default subproperty. Strings are often assigned to **content** or **text-content**, and colors are often assigned to **color** or **text-color**. These conventions allow very efficient and readable specification of Mouin elements.

3.2. Full reference

Here is the full list of Mouin CSS styles and properties.

-mouin-settings

This style is only meaningful in the BODY element. it is a set of flags and setting values that control the behavior of the Mouin client app. The settings allow you to finetune the behavior to get the best performance for the web content that is being processed. Documentation on these will be made available in future versions of the documentation.

-mouin-app-* styles

The set of app styles allows specifying how the page must be treated when converted into a mobile app. This includes the app name and icon. Here are the actual styles:

- **app-name**: this string is name under which the app will be published.
- **app-icon**: this URL refers to the icon for the app. Using a 512 x 512 pixels image or higher is recommended.

Here is a sample of applying these styles:

```
<body data-mouin="-mouin-app-name: "Socks"; -mouin-app-icon:
url(http://mysocks.xyz/socks.png);">

  ...

</body>
```

These styles have to be applied to the BODY element. Currently, they must be present in the main HTML for limitations in the the 1.0 client app.

-mouin-sytle

This is the main Mouin CSS attribute. You use this CSS attribute to set the Mouin properties of an HTML element. When you give it a type other than "none", then the element will be treated especially by the Mouin client.

Here are the supported properties:

- **type**: this is the main property, controlling any special display of an element. The possible values are *none*, *header*, *section*, *item* or *article* (the initial value is *none*)
- **text**: the main text. This has two sub-properties:
  - **text-content**: a string with the text
  - **text-color**: the color and attributes of this text
- **title**: a string with the title text
  - **title-content**: a string with the title
  - **title-color**: the color and attributes of the title
- **image**: the URL of image
- **href**: the URL of the target for a Mouin element involving a link
- **on**: the background attributes for an element
  - **on-type**: *transparent*, *flat* or *gradient*
  - **on-color**: the background color
- **border**: the color of the border or separator line. It may be applied differently, or not at all, on the different supported client platforms.
- **page**: a property that controls subpage creation (presenting part of a page as a separate page). IT has two subproperties:
  - **type**: this can be either *enable* or *disable* (initial value except for the **body** element). It makes this element or its children be displayed in a separate page.
  - **update**: this can page has to present a quick “update” UI element
  - **title**: the title or heading string for the subpage
- **visibility**: this can be *visible* or *hidden*. This only affects the translation of regular HTML nodes when processed as part of an article element. The initial (default) value is *visible*, if set to hidden, its content is removed when translating.

Property values have to be given in NAME(VALUE) pairs. When there is no ambiguity, you can give just the VALUE and Mouin will deduce the NAME.

Example:

```
#header { -mouin-style: header "Abc" white on(blue gradient); }
```

In this example, the property names **type**, **text-content** and **text-color**, and the subproperty names **on-type** and **on-color** have been elided, and are deduced and applied by default.

-mouin-set-prop

This style makes Mouin extract some information from the element it is applied to, and assign it to a Mouin property of another element.

The property value is a list of keywords, indicating what properties to add the given value to. The value of the target property is taken from either the text of the element (for **text** or **title**), from the **href** attribute (for the **href** property), or from the image **src** (for the **image** property).

Example:

```
a.note { -mouin-set-prop: title href; }
```

The properties are assigned, by default, to the previous or parent element of the one it is applied to.

-mouin-insert

This style allows inserting anonymous, empty HTML elements into the DOM tree with Mouin styles. This is especially useful when injecting Mouin CSS to a page created without Mouin in mind.

The property value is a list of element descriptors, each surrounded by parentheses. An element descriptor has the same syntax as the **-mouin-style** style. You can build full trees embedding children, by using extra element descriptors in parentheses inside their parent element.

Examples:

```
body {
  -mouin-insert: (header  "Abc"  image("a.gif") red on(black));
}
```

With children elements:

```
body {
 -mouin-insert: (subpage
                   (item "Item1"  image("1.gif") href(/1.html))
                   (item "Item2"  image("2.gif") href(/2.html))
                 );
}
```

-mouin-default

This property allows setting default property values for all elements of a given type in each page or subpage.

The syntax is similar to that of **–mouin-insert**, although each descriptor between parentheses sets the default values for one type of element.

Example:

```
body {
  -mouin-default: (item image("bullet.gif") white on(blue))
                  (section on(gray));
}
```