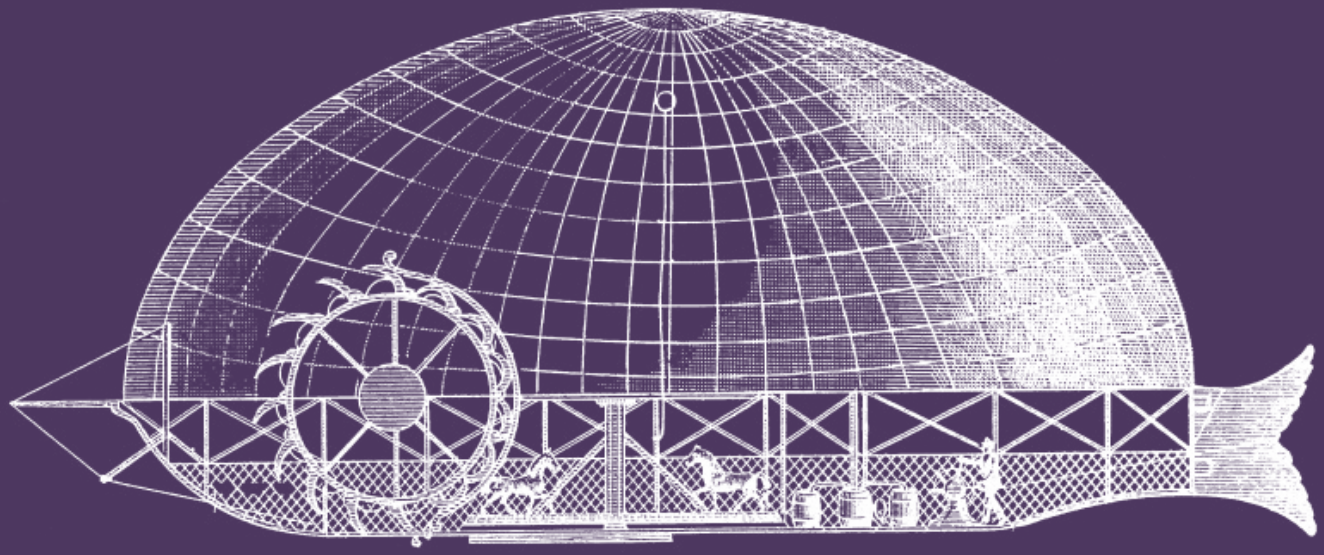
Methodology for writing (X)HTML code



*W*orking philosophy

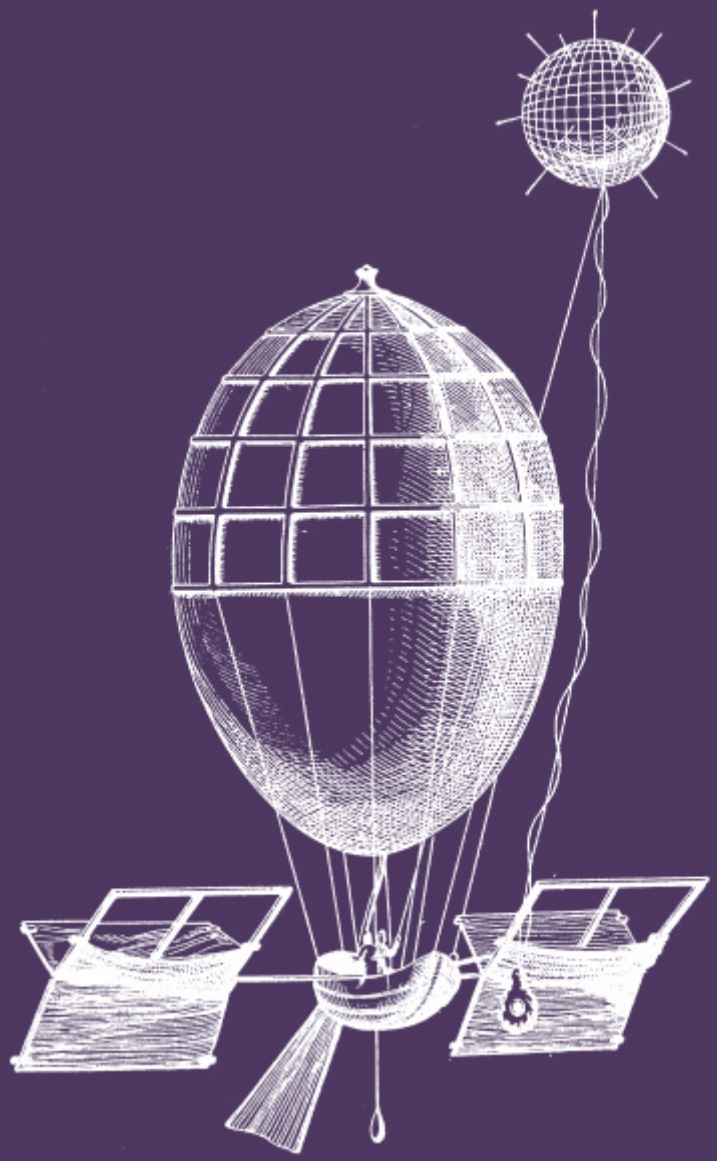
When we talk about quality, we especially mean all the characteristics inherent in a site that make it possible to **upgrade, maintain and port it** to other media **in an efficient, economical and controlled way.**

- Evolution and future compatibility
- Maintenance
- Portability

The way the code is written **is the most important point** *of all the "form/content separation" philosophy*

Think of your (X)HTML code as the foundations of your house

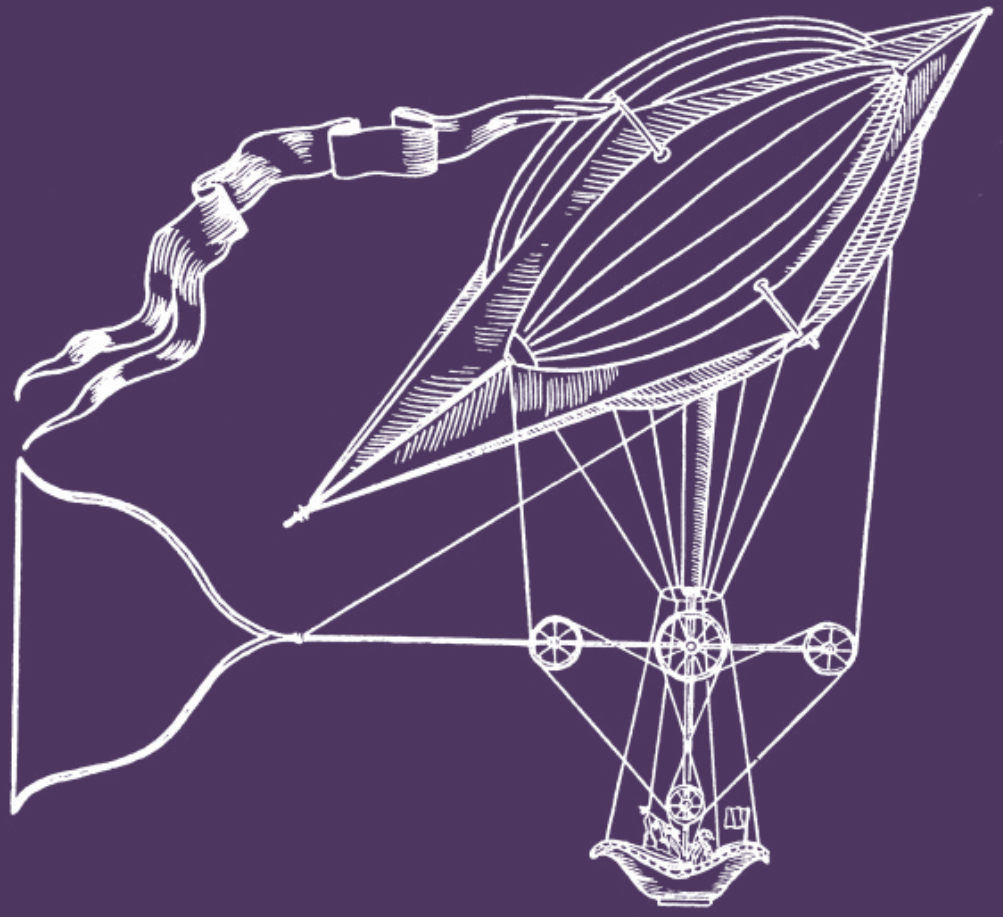
*R*ules



These rules must *always* be complied with:

3.1 DOCTYPE declaration
3.2 calling stylesheets
3.3 "*namespace*" declaration
3.4 main language declaration
3.5 "*content-type*" definition
3.6 document title
3.7 XHTML backwards compatibility rules
3.8 comply with the syntax of the language and its elements
3.9 use tabulations and carriage returns to make the code clear
3.10 do not "force" the addition of spaces or carriage returns
3.11 validate your code

Methods *and* Conventions



Editor:

choose it and know **why.**

- **Have control**: Check to see if the code is not automatically modified by the editor

- **Be safe**: Check if your editor respects the *charset* (some editors cannot save files in UTF format)

Always call the **development** CSS stylesheet

- Preserve the integrity of the original Studio stylesheet
- Track updates better
- Check bugs better

The (X)HTML code first and then the **CSS** design …

…but they are not mutually exclusive.

- Well-considered, consistent, accurate coding
- "View" the page without the CSS stylesheet.

Comply with the naming rules

- Semantic names
- No special characters or spaces
- Not starting with a digit
- Not too long
- In English
- Prefer *microformats* conventions and names conforming to HTML 5 tags

Conventions *applicable to* comments

- Always add an opening and a closing comment to a **component block**

- Remember to add a simple comment next to the closing tag of the main containers to specify the block which this closing refers to

The code has to be **flexible** and **upgradeable**

Remember to test your pages with "extreme" content and with most of the tags.

It is not recommended to start the (X)HTML editing by **the homepage**

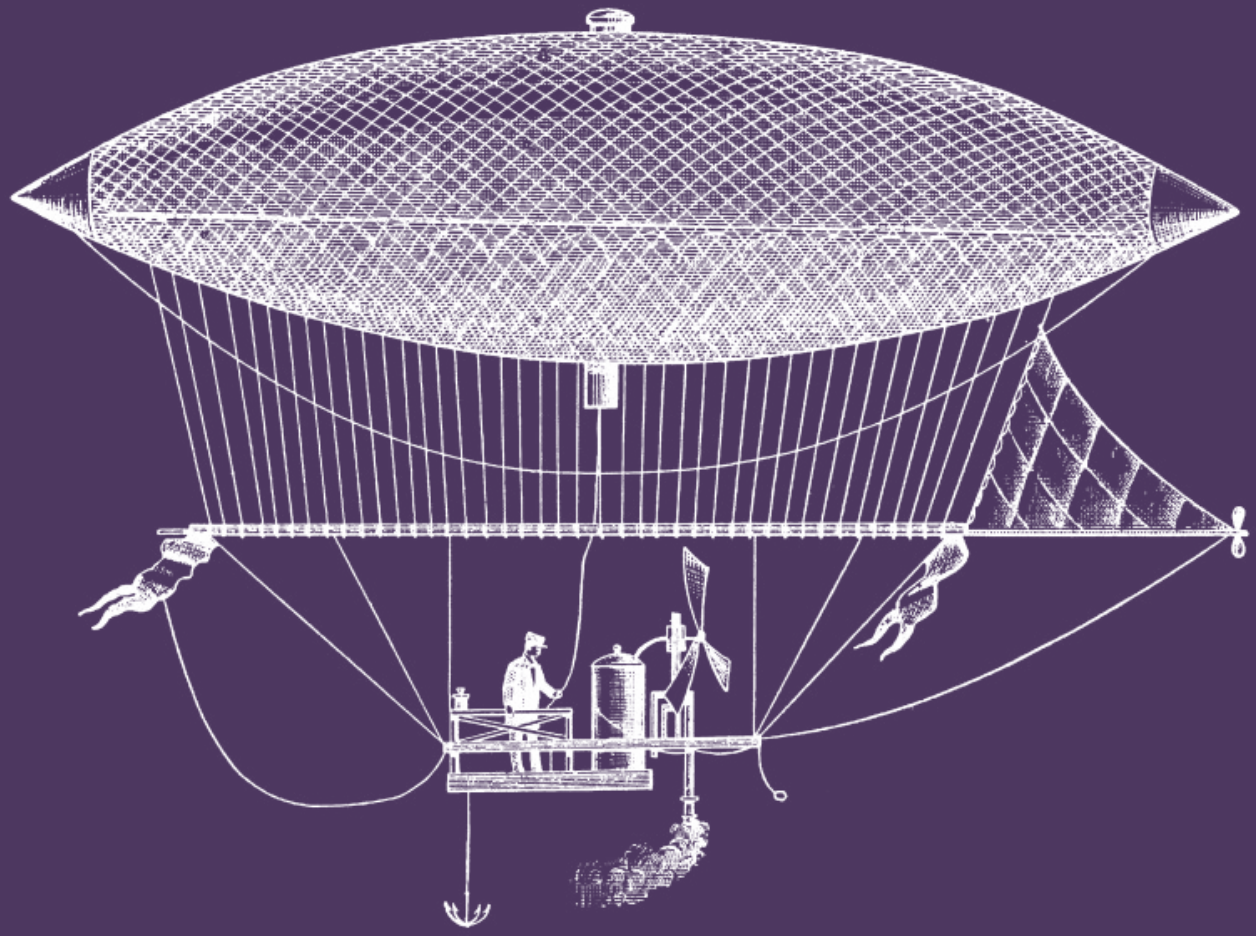
Have a very good view of the various templates and elements of internal pages, and start with them.

Think in terms of "**components**"

- Study the graphical model and/or the wireframe to identify the various component blocks.

- Group component blocks according to their differences and/or similarities.

- Put components into a separate, commented (X)HTML file.



Good practices

Avoid "*div-ities*"⋆

Every tag has its function, every bit of content has a role.

⋆ For HTML5 « writers »: avoid « section-ities » also!

Carefully consider the attribution of classes or ids

- Classes are used for **exceptions**
- Classes can be used in **combination with other classes**
- Give ids to elements that are **unique (structuring and/or dynamic)** or which have to be **manipulated through javascript.**

There must be no "anonymous" content

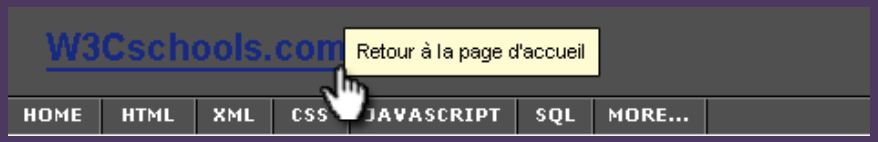
```
<body>
	<h1>Mon site</h1>
	<p><img src= "imgs/logo.gif" alt= " " /></p>
	<p>Lorem ipsum sit amet.</p>
	<p><a href= "doc2.html ">page suivante</a></p>
</body>
```

Do not add properties directly into the page using the "**style**" attribute

When testing your CSS design, preferably use **Firebug** to do tests with various values

Use the" *title*" attribute wisely

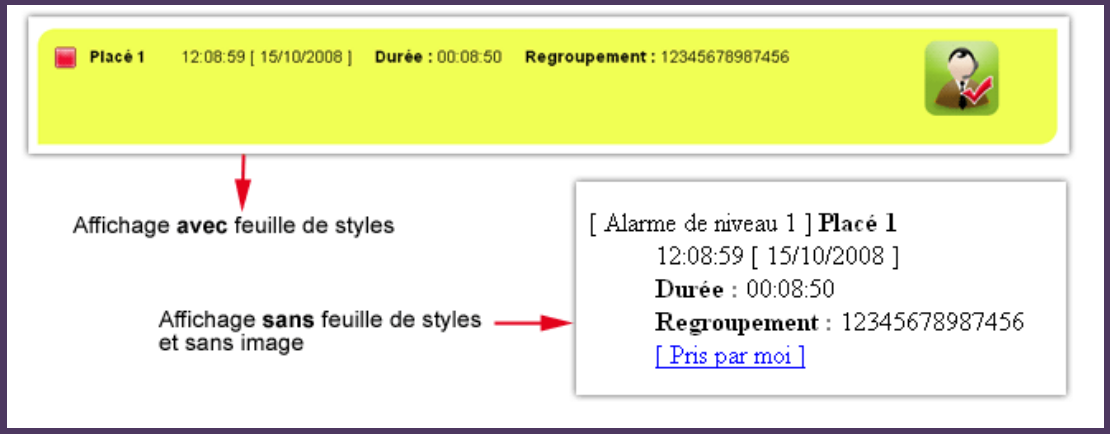
Images on



Images off

- "*title*" explains a link, context or bit of content.
- Preferably assign it to the a tag.

Know how to use the lists of definitions




```
<dl>
  <dt>
    <img src="imgs/alert1.gif" alt="[ Alarme de niveau 1 ]" />
    <strong>Placé 1</strong>
  </dt>
  <dd>12:08:59 [ 15/10/2008 ]</dd>
  <dd><strong>Durée :</strong> 00:08:50</dd>
  <dd><strong>Regroupement :</strong> 12345678987456</dd>
  <dd class="alertAc">
    <a href="#" title="Libérer main">
        <img src="imgs/suivi_pris.gif" alt="[ Pris par moi ]"/></a>
  </dd>
</dl>
```

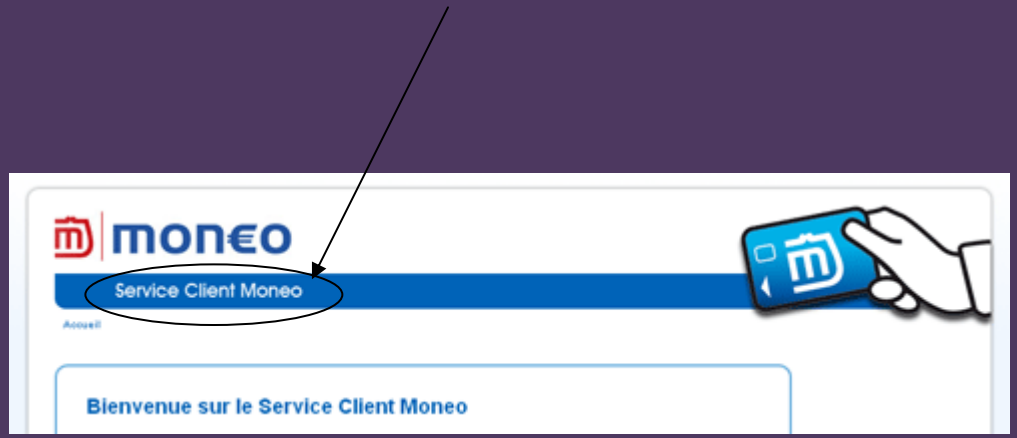
<*strong*> and **<*em*>** are pieces of semantic information:

define them in the code.

Use lines (<hr />) to separate the main *content groups*

To make the page easier to read, view it without its stylesheet and insert a <hr /> tag in order to separate the bits of content which you think should be separated.

Prefer the <h1> tag for the *title of the site*



The title of the site must take the user to the homepage also

Always add a **hidden link** for skipping menus

```
<p class="skip"><a href="#wrapper">Skip this menu</a></p>
<ul id="nav">
   <li><a href="#">Rubrique 1</a></li>
   <li><a href="#">Rubrique 2</a></li>
   <li><a href="#">Rubrique 3</a></li>
</ul>
```

Create your **menus** *carefully*

- A section menu must always be structured using (***<ul>)*** list tags
- Process and identify the current section (in all of its states: active link or not)
- A navigation line (breadcrumbs) is not a list

Images

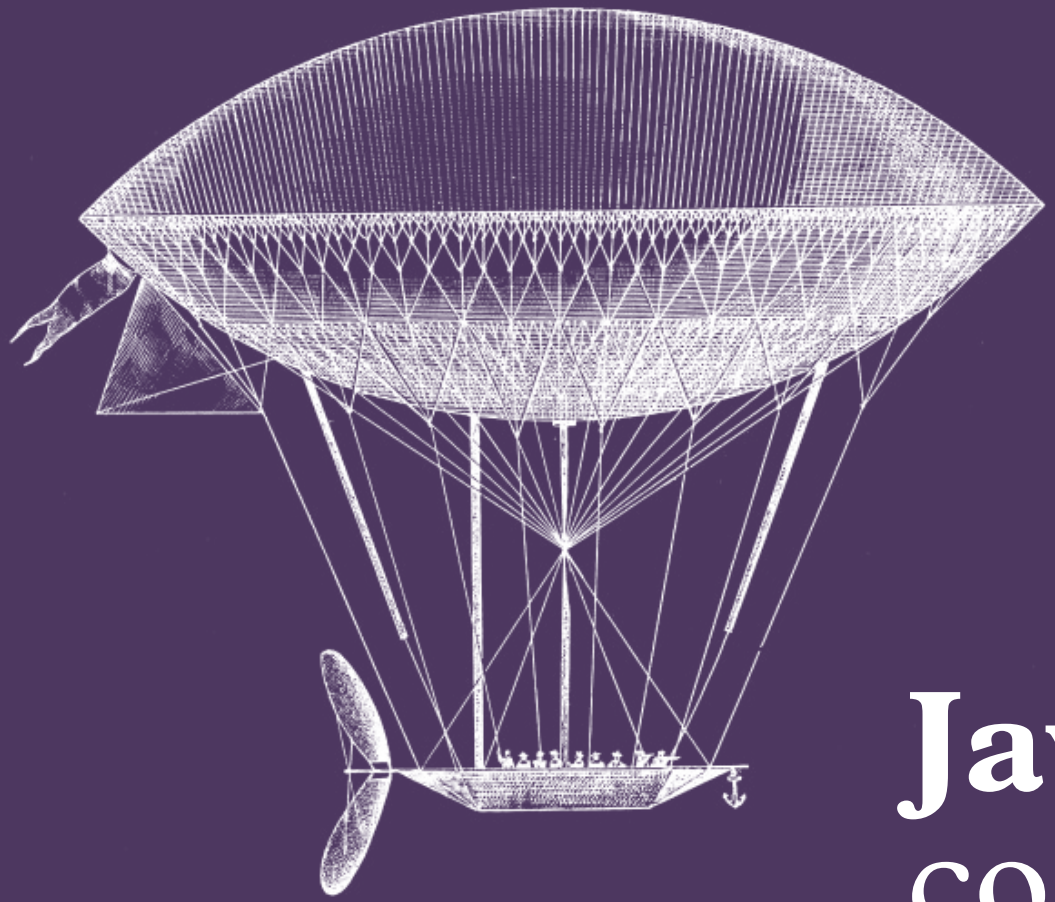
- Always provide text alternatives to images
- Consider when to give a size to pictures or not
- Decorative images (images that don't have an information) should preferably be defined as background images

Forms

- Always assign an ID to a form.
- Carefully consider the attributes of the <*form*> tag.
- Group the fields of the same type using the <*fieldset*> tag.
- Always use the <*legend*> tag for the title of a fieldset.
- Use labels to associate a description with its field.
- Always group the label and its field into a <p> tag.
- Always assign an id and a "*name*" to a form field.
- Make provision for the construction of link and input buttons.
- Create a structure for form buttons.
- Code the *Checkboxes* and/or *Radios* correctly.
- Make provision for the information elements that are frequently found in forms.

Tables

- Use CAPTION for the title of a table.
- Add a summary of the data of the table.
- Use the correct tags for column or row headers.
- Specify the logical order in which the table should be read using the SCOPE attribute.
- Consider the possible situations (cell alignment, alternate row colours, links for headers, sorting arrows, etc.).



Javascript coding

Repeat after me:
I am unobtrusive

- Do not define a function directly in the page.
- Separate behaviour from content.
- Always provide an alternative to javascript.
- Think out functions so that they do not depend on a single type of input or browser: test your code using the keyboard
- Use the correct syntax to pass variables through a URL (“&” is a reserved character)