CSS design *methodology*

Philosophy


Ideality
Sublimity
Cautiousness
Secretiveness
Destructiveness
Amativeness

Like the (X)HTML source,
the CSS stylesheet is part of our *deliverables*

Our deliverables reflect our competence.

The stylesheet is a living document

- Several "editors"
- Considerable lifetime
- Properties and values are not important...
- ...but selectors are.

Ref.: Methodology for writing (X)HTML code, paragraphs 5.1 and 5.2

CSS design has to be **flexible** and **upgradeable**

- Do tests with "extreme" content
- Test your pages with text zoom
- Consider translations issues

Ref.: *Methodology* for writing (X)HTML code, paragraph 4.6

Whereas CSS design is the "decoration" of a house, (X)HTML constitutes its foundations

The CSS design cannot live without the source code, but the source code lives without the CSS design

Ref.: Methodology for writing (X)HTML code, paragraph 2.1

Always go *from the most* **generic** *to the most* **specific**

- Always have as many graphic design elements as possible before you start
- It is recommended not to start the production by the homepage or specific blocks (e.g., navigation)
- Think in terms of components

Ref.: Methodology for writing (X)HTML code, paragraphs 4.7 and 4.8

Project scoping is **crucial**

- Page behaviour
- Accessibility needs
- Customer's requirements
- Graphic design particularities
- Versions
- CMS
- Print stylesheet

Conventions

Fill in the stylesheet correctly

- Name of the project, customer, role of the CSS stylesheet, author, date;
- Relationship with the other CSS stylesheets;
- Index of style groups;
- Index of colour codes may be a plus

```
/* ----------------------------------------------------------------------
          Worldline PlayLive / Atos Worldline
              Main styles definitions
            By Le Studio, Atos Worldline, 2009
---------------------------------------------------------------------- */

/* NOTES: Used in combination with "ie6hacks.css" and "ie7hacks.css"
style-sheets, called by conditional comments by each HTML file --
check those CSS for corrections for IE6 and IE7 browsers

Check out the "dev.css" style-sheet also for modifications or
add-ons done by the dev team
*/

/* MAIN COLORS REFERENCE:
    main color/red:     #C00;
    light grey:         #CCC;
    dark grey:          #666;
*/

/* CSS index (!):
  DEFAULT VALUES
  PAGE BODY STRUCTURE & MAIN ELEMENTS
  NAVIGATION
  TABLE SPECIFICS
  FORMS SPECIFICS
  SEARCH
  GENERAL BLOCKS
  SPECIFIC PAGES
  EXCEPTIONS & SPECIFICS (used sitewide)
*/
```

Consider both *the* order and *the* **structure**

- Group the styles of the same nature and comment each group
- Use spaces and tabulations for easy viewing and editing
- Declare properties in the same order each time you define a style

```
/* ------------------------------ ! PAGE BODY STRUCTURE & MAIN ELEMENTS ------------------------------ */

/* --- MAIN BLOCKS --- */

/* Main container */
#container {
    width: 980px;
    margin: 0 auto;
}

/* Columns wrapper */
#wrapper {
    float: left;
    width: 980px;
    margin-bottom: 20px;
}

/* Columns */
#mainCt {
    float: left;
    width: 940px;
    min-height: 300px;
    color: #333;
    background: #E3E3E3;
    padding: 20px;
}

#aside {
    float: left;
    width: 956px;
    min-height: 300px;
    color: #333;
    background: #FFF url(../imgs/bg_aside.png) bottom left no-repeat;
    padding: 20px 12px;
    border-top: 3px solid #333;
}
```

Rational grouping

Redefinition of the default values;

Page structure and main elements;

Navigation;

Table-specific styles;

Form-specific styles;

General blocks;

Styles specific to a page;

Others (pop-ups, modal windows, interactive elements, etc.);

Specific exceptions and general styles used on the entire site.

Naming styles

- Use *Microformats* classes where possible
- Define classes and Ids in english
- Name based on function, not style
- Use classes and Ids based on the new HTML5 tags where possible

Ref.: Methodology for writing (X)HTML code, paragraph 4.4
Ref.: http://microformats.org/wiki/semantic-class-names

*I*mperatives

Preferably use **short-hand** declarations

For instance:

```
margin: top/bottom right/left;
color: #FFF;
```

Always define **relative font sizes and…**

- Define a generic size of **100.1%** for the BODY
- Define sizes using ***em***
- Adjust them using **percentages** where needed

…define them for **block elements**

- Sizes can be controlled more accurately if they are defined for "**text blocks**" (block elements related to textual content):
  - P, DT, DD, LI, TH, TD, CAPTION, etc.
  - **and not**: DIV, TABLE, FORM, DL, UL, etc.
- **Inline elements** (SPAN, A, SELECT, INPUT…) must inherit their parent's text size or size must be adjusted using percentage

List the **pseudo-selectors** for the **a tag in the correct order**

LoVeHAte

a

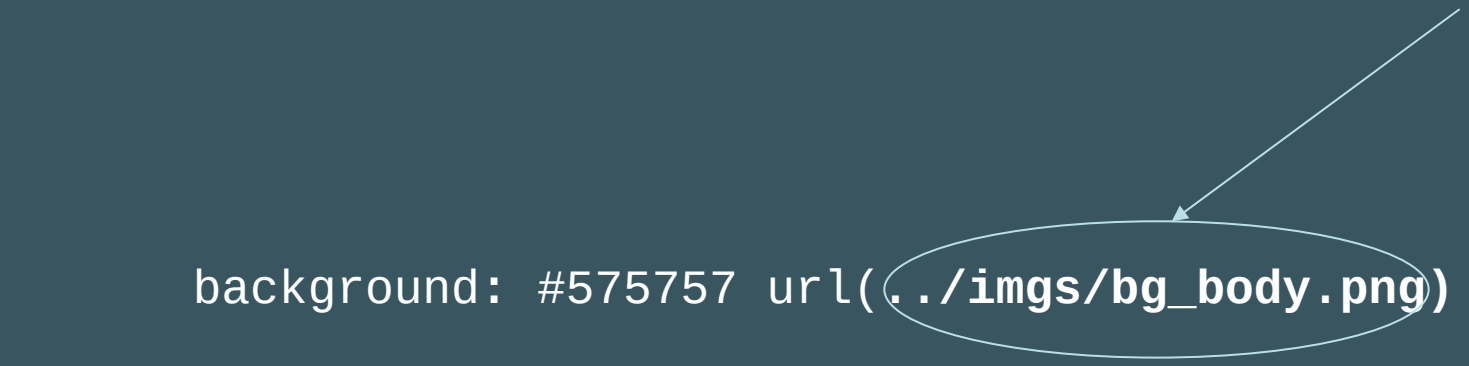
a:link

a:visited

a:hover

a:active

No **quotation marks** *for* URLs



```
background: #575757 url(../imgs/bg_body.png) repeat-x;
```

Always assign a **width** to a block that is positioned as "*absolute*“!

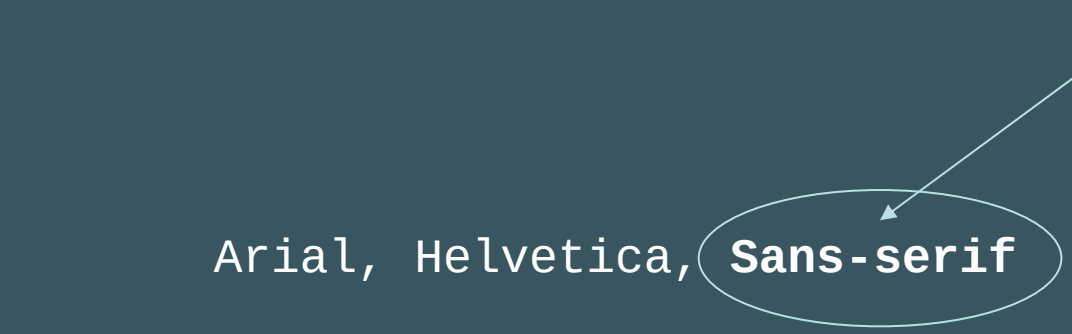
Always assign an **explicit width** to a "*float*" block!

- "auto" is an acceptable value
- Image widths are implicit

Do not assign any unit to the value 0

This is simply useless.

Always define a *generic* **font family**



Consider IE 6's **limited** support for **selectors**

- No `classe1.classe2 {…}`
- No `input[type=submit] {…}`

Good Practices

Use a **reliable*** browser for CSS design

…then do tests with all the other browsers.

* Not Internet Explorer

First, *equalize*

Equalize the default values of tag properties:

```
body, div, dl, dt,
  dd, ul, ol, li,
  h1, h2, h3, h4, h5, h6,
  pre, code, form, fieldset,
  legend, input, textarea,
  p, blockquote, th, td {
    padding: 0; margin: 0;
} /* reset */
```

Think before grouping *selectors* into the same definition

- This helps save Kbs but causes a loss of flexibility and many misunderstandings
- This has to be avoided if selectors are used in different contexts
- Good analysis of HTML components before the CSS design may prove much more useful to improve the weight and the structure of the stylesheet

Ref.: Methodology for writing (X)HTML code, paragraphs 4.7 and 4.8

Do not leave any **useless** definitions or properties

- Test properties have to be deleted
- Check for properties that "do nothing" or are redundant:

```
strong {font-weight: bold;}
```

Specify widths using *percentages* as much as possible

Flexibility and upgradeability:

if the width of the parent changes, the width of the children changes too.

Hide **navigation aid links** *correctly*

- Certain speech synthesizers will read `display: none`;
- Preferably use a negative top/left margin or text-indent

Ref.: *Methodology* for writing (X)HTML code, paragraph 5.11

Be careful before *aligning* an element **vertically**

- `line-height` has to have the same height as the parent element
- Always use *em* or *ex* to define the height in order not to suppress the ability to zoom in

Stick to the same **logic** to define *spacing*

Always define the bottom and right margins, for example, for all blocks (following the page assembly logic)

For "**liquid**" design, always define a *maximum* and a minimum width

And consider *media queries* for different sizes of devices

DEBUGGING

No, you cannot do without FireBug!

Only *good* reflexes…

- The DOCTYPE is present in the right place and is the correct one, and it is complete
- Take IE's bugs into account
- Add a background colour to the block that may cause the problem to better understand its behaviour

Validate the
F*%!?$¤£
page!

"Hey, kid! Have you thought about the double-margin bug?"

For **IE6** *and* **IE7**, consider the **HASLAYOUT** property

Elements with “layout” by default:

- `html`
- `body`
- `table`
- `tr`
- `th`
- `td`
- `img`
- `hr`
- `input`
- `button`
- `select`
- `textarea`
- `fieldset`
- `legend`
- `iframe`
- `embed`
- `object`

CSS Properties that "give layout" to an element:

IE6	IE7
position: absolute	position: absolute
float (right / left)	float (right / left)
display: inline-block	display: inline-block
width (all values besides " auto ")	width (all values besides " auto ")
height (all values besides " auto " – a value of 1% is valid)	height (all values besides " auto " – a value of 1% is valid)
zoom (IE property – avoid it: page won't validate)	zoom (IE property – avoid it: page won't validate)
	position: fixed
	min-width/max-width (all values)
	min-height/max-height (all values)
	overflow: hidden|scroll|auto