MAIN OBJECTIVE

Design websites using html and css

SPECIFIC OBJECTIVES

- **Insert text, images and videos, work with forms and tables**
- **Make links between pages**
- **Format by modifying the color, size, background, font...**
- **Position the elements of the site: header, menus, footer...**
- **Work with flex boxes and grids**

PARTIE 1 : LES BASES DE HTML5 ET CSS 3

CHAPTER I: HTML AND CSS : TWO LANGUAGES TO CREATE A WEBSITE

General objective : to allow the student to have the basics in html and css

Specific objectives :

- **define html and css**
- **define text editor**
- **set browser**
- **define tag and attribute**
- **highlight the types of beacons**
- **structure an html page**

Keywords: computer language, html, css , website, Notepad++, Google chrome, chevrons, doctype , <head> , <body>…

Learning methods : receptive, active collaborative, active individual

Types of activity : collaborative work, presentation, practical exercises, etc.

Evaluation methods : before, during and after learning.

I.1 The role of HTML and CSS. The different versions of HTML and CSS

These are ***computer languages*** used to create websites. All websites are based on these languages, they are essential and universal today. To create a website, one must give instructions to the computer. It is not enough simply to type the text that will be on your site (as you would do in a Word word processor for example), you must also indicate where to place this text, insert images, make links between the

pages, etc To explain to the computer what you want to do, you will have to use a language that it understands.

a. HTML (Hypertext Markup) Language)

It appeared in 1991 when the Web was launched. Its role is to manage and organize the content. It is therefore in HTML that you will write what you want the page to display: text, links, images...

HTML Versions

- **HTML 1** : This is the very first version created by Tim Berners -Lee in 1991.
- **HTML 2** : the second version of HTML appears in 1994 and will end in 1996 with the appearance of HTML 3.0. It is this version that will actually lay the foundation for future versions of HTML. The rules and operation of this version are given by the W3C (while the first version was created by one man).
- **HTML 3** : appeared in 1996, this new version of HTML adds many possibilities to the language such as tables, applets, scripts, positioning of text around images, etc.
- **HTML 4** : This is the most popular version of HTML (specifically it is HTML 4.01). It appeared for the first time in 1998, and offered the use of frames (which cut a web page into several parts), more complex tables, improvements to forms, etc... But above all, this version allows for the first times the use of style sheets, our famous CSS!
- **HTML 5** : this is THE latest version. Still quite uncommon, it is getting a lot of attention because it brings many improvements such as the ability to easily include videos, better content layout, new features for forms, etc. It is this version that we are going to discover together.

b. CSS (Cascading Style Sheets , also called Style Sheets)

The role of CSS is to manage the appearance of the web page (layout, positioning, decoration, color, text size...). This language complemented HTML in 1996. HTML defines the content (as you can see, it's raw!). The CSS allows him to arrange the content and define the presentation: color, background image, margins, text size...

CSS versions

- **CSS 1** : from 1996, the first version of CSS is usable. It lays the foundations of this language which allows you to present your web page, such as colors, margins, fonts, etc.

- **CSS 2** : appeared in 1999 and then supplemented by CSS 2.1, this new version of CSS adds many options. We can now use very precise positioning techniques that allow us to display elements where we want them on the page.
- **CSS 3** : this is the latest version, which brings much-awaited features like rounded borders, gradients, shadows, etc.

➢ The text editor

The text editor is a tool that allows you to create and format text or it is software intended for the creation and editing of text files . We can classify these *website creation software* into two categories:

- WYSIWYG (What You See Is What You Get - What You See Is What You Get): these are programs that are very easy to use, they allow you to create websites without learning any particular language **.** Among the best known of them: Nvu , Microsoft Expression Web, Dreamweaver... and even Word! Their main drawback is the quality of the HTML and CSS code which is automatically generated by these tools, often of rather poor quality. A good website designer must sooner or later know HTML and CSS, that's why I don't recommend using these tools.
- **Text editors** : these are programs dedicated to writing code. They can generally be used for multiple languages, not just HTML and CSS. They turn out to be powerful allies for website builders

- **Under Windows**

There are a large number of text editors. Nevertheless, I invite you to look at Notepad++, one of the most used of them under Windows. This software is simple, in French and free .

To achieve this: go to the "Language" / "H" / "HTML" menu. This will let the software know that you are going to type HTML.

When you use the software, it will color your code which will make it easier for you to find your way around.

Other editors available under Windows exist:

- I'm saying
- padd
- CONTEXT

... and many others

➢ Browsers

The browser is the program that allows us to view websites. The browser's job is to read the HTML and CSS code to display a visual result on the screen. If your CSS code says "Headings are red", then the browser will display the headings in red. The role of the browser is therefore essential ! The main problem is that **browsers do not display all sites in exactly the same way** !

There are many different browsers. Here are the main ones to know:

Navigateur	OS	Téléchargement	Commentaires
Google Chrome 	Windows Mac linux	Download	Google's browser, easy to use and very fast. **This is the browser I use every day.**
Mozilla Firefox 	Windows Mac linux	Download	The famous and reputable Mozilla Foundation browser. I use it frequently to test my websites.
Internet Explorer 	Windows	Download (Already installed on Windows)	Microsoft's browser, which equips all Windows PCs. I use it frequently to test my websites.
Safari 	Windows Mac	Download (Already installed on Mac OS X)	Apple's browser, which equips all Macs.
Opera 	Windows Mac linux	Download	The Eternal *Outsider* . It is less used but offers many features.

NB : It is advisable to install several browsers on your computer to ensure that your site works correctly on each of them. In general, it is necessary to test your website regularly at least on Google Chrome, Mozilla Firefox and Internet Explorer.

As explained earlier, browsers don't always display websites in *exactly* the same way. This is because they don't always know the latest HTML and CSS features. For example, Internet Explorer has long lagged in some CSS features (and ironically, it has also been ahead of a few others).

Several versions of browsers coexist:

- **Firefox 2, Firefox 3.5, Firefox 3.6, Firefox 4;**
- **Internet Explorer 6, Internet Explorer 7, Internet Explorer 8, Internet Explorer 9**
- **Chromium 8, Chromium 9, Chromium 10**

Each version supports new features, but if users don't update their browsers it becomes a problem for webmasters *like* you who create websites.

Chrome largely solved the problem by setting up automatic updates, without user intervention. Firefox has users who don't think to update, and Internet Explorer has a hard time encouraging them to update because the latest versions also require Windows to be updated (Internet Explorer 9 is not available for Windows XP For example).

I.2 Tags and their attributes

a. Tags

HTML tags are the building blocks of **HTML coding** . They are used to format a text, structure and prioritize the content of a page. Tags also tell the browser how to display the page **in** question.

Tags are easy to spot. They are surrounded by " **chevrons** ", i.e. symbols < and > , like this: **<tag>** . They indicate the nature of the text around them. They mean for example: "This is the title of the page", "This is an image", "This is a paragraph of text", etc.

There are two types of tags: pair tags and orphan tags.

- **Even tags**

They open, contain text, and close later. This is what they look like:

Code: HTML

```
<title>Ceci est un titre</title>
```

There is an opening tag (**<title>)** and a closing tag (**</title>) which indicates** that the title ends **.** This means for the computer that everything that is not between these two tags... is not a title.

- **Orphan tags**

These are tags that are most often used to insert an element at a specific location (for example an image). There is no need to delimit the beginning and the end of the image, we just want to tell the computer "Insert an image here".

An orphan tag is written like this:

Code: HTML

```
<image/>
```

b. Attributes

Attributes are tag options. They complement them to provide additional information. The attribute goes after the name of the opening tag and usually has a value, like this:

Code: HTML

```
<balise attribut="valeur">
```

Example: Let's take the **<image/> tag** that we have just seen. Alone, it is of little use. We could add an attribute that indicates the name of the image to display:

Code: HTML

```
<image nom="photo.jpg" />
```

For the writing of the HTML code to be correct, the following points must be checked:

- There should be no space between the opening chevron and the tag name
- There must be a space (or more) between the tag name and the attribute
- There must imperatively be a space (or more) between the successive attributes
- The attribute value must be declared between quotes (double quote or single quote). However, browsers can tolerate the absence of quotes, but your code remains invalid from the point of view of W3 C(*).

➢ Special characters

HTML consists mainly of tags. But that's not all because there are sequences, other than tags, that the browser recognizes and replaces with their meanings. These sequences are called: **special characters** .
The HTML code for special characters begins with the symbol " **&** " and ends with " **;** ".
For example : **©** stands for " © ".
The following table lists the most used special characters in HTML:

Character	HTML code
"	& quot ;
&	& amp ;
<	& lt ;
>	>
where	& oelig ;
Space	& nbsp ;
£	£
©	©
®	®
±	& moremn ;
µ	&microphone;
½	½
VS	& Ccedil ;
æ	& aelig ;

NB : Some special characters like © or & can be written directly without going through their HTML code. HTML does not take into account the succession of white spaces, only the first is considered. To force several successive spaces you have to go through the HTML code **& nbsp ;**

II. Basic structure of an HTML5 page

An HTML page is structured like this:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

- The doctype

The very first line is called the *doctype* . It is essential because it is what indicates that it is indeed an HTML web page . It is not really a tag like the others (it begins with an exclamation point).

- The <html> tag

This is the main code tag. It encompasses all the content on your page. As you can see, the closing **</html> tag** is at the very end of the code!

- The header <head> and the body <body>

A web page consists of 2 parts:

- ✓ **The header < head >** : this section gives some general information about the page, such as its title, encoding (for handling special characters), etc. This section is usually quite short. The information that the header contains is not displayed on the page, it is simply general information intended for the computer. However, they are very important!
- ✓ **The body <body>** : This is where the main part of the page is. Everything we write here will be displayed on the screen. It is inside the body that we will write most of our code.

- *The encoding (charset)*

This tag indicates the encoding used in your .html file . , the encoding indicates how the file is saved. It is he who determines how the special characters will be displayed (accents, Chinese and Japanese ideograms, Arabic symbols, etc.).

We can differentiate between two main families of meta tags :

- The metas **name** : Affects all information about the document, author, site, tools used, etc.
- The metas **http- equiv** : Relates to metas that communicate with the browser.

While there are many different meta tags , most are unnecessary and therefore completely optional. To keep the W3C validity of your page, only the tag concerning its encoding is necessary.

A few years ago, certain tags were essential for the referencing of its website, such as the description and the keywords, however, the referencing techniques being in perpetual evolution, today their interest is minimal.

III-D-2. The meta content-type tag ▲

This meta is the most important and the only one required to pass the W3C validation.

- meta **content-type** — Specifies the MIME type of the page as well as the character set used (encoding).

Example

Select

< meta http- equiv ="content-type" content=" text /html; charset =utf-8">

III-D-3. Add info about yourself ▲

- meta **author** — Provides the first and last name of the author of the current page. If there are several authors, separate them with a comma.

Example

Select

< meta name="author" content=" Josselin WILLETTE">

- meta **publisher** — Provides the first and last name of the site publisher. If there are multiple publishers, separate them with a comma. In the case of a company, you must enter its name.

Example

Select

< meta name =" publisher " content="Josselin WILLETTE">

- meta **reply -to** — Sets the page author's email address. Beware, this tag is scanned by spambots!

Example

Select

< meta name="reply-to" content=" jwillette at monsite dot com">

- meta **contact** — Defines the email address of the person or department to contact. Beware, this tag is scanned by spambots!

Example

Select

< meta name="contact" content="contact at mycompany dot com">

- meta **contactname** — Specifies the name of the contact person.

Example

Select

< meta name =" contact name " content="Josselin WILLETTE">

- meta **contactorganization** — Specifies the name of the company that owns the site.

Example

Select

< meta name =" contactorganization " content=" MyCompany SARL">

- meta **contactstreetaddress1** — Indicates the first line of the address of the author or company that owns the site.

Example

Select

< meta name ="contactstreetaddress1" content="15 coughing doll street">

- meta **contactstreetaddress2** — Indicates the second line of the address of the author or company that owns the site.

Example

Select

< meta name ="contactstreetaddress2" content="Building A, Floor 3">

- meta **contactzip** — Indicates the postal code of the author or company owning the site.

Example

Select

< meta name =" contactzip " content="75000">

- meta **contactcity** — Indicates the city of the author or company owning the site.

Example

Select

< meta name =" contactcity " content="MAVILLE">

- meta **contactstate** — Indicates the country of the author or company owning the site.

Example

Select

< meta name =" contactstate " content="FRANCE">

III-D-4. Add info to your site ▲

- meta **description** — Short description of the site that will be displayed in the results of most search engines. Do not exceed 200 characters under penalty of seeing the description truncated.

Example

Select

< meta name ="description" content="HTML tutorial on the basics to know to start web programming well">

- meta **keywords** — Contains a list of keywords that search engines will refer to. Keywords are separated by a comma. Having too many keywords can be considered by the engines as spam, so do not exceed 500 characters.

Example

Select

< meta name ="keywords" content="html tutorial, html course, meta tag , forms, lists, tables, frames">

- meta **identifier-url** — Provides the entry URL for the site.

Example

Select

<meta name ="identifier-url" content="http://j-willette.developpez.com/">

- meta **subject** — Explains the subject of your site in a few words.

Example

Select

< meta name =" subject " content="HTML learning site">

- meta **distribution** — Defines the type of publishing network for the site.

Values	Description
overall	Visible on the Internet.
local	Visible only on a local workstation.

iu	Visible on an intranet.

Example

Select

< meta name ="distribution" content="global">

- meta **rating** — Defines the type of audience targeted by the site.

Values	Description
general	For all.
mature	Reserved for adults.
restricted	Restricted.
14years	Access prohibited under 14 years.

Example

Select

< meta name ="rating" content=" general ">

- meta **language** — Indicates the language used on the site.

Example

Select

< meta name =" language " content=" en ">

III-D-5. Add creation info ▲

- meta **copyright** — Indicates the copyright(s) to which the page is subject.

Example

Select

< meta name ="copyright" content="© 2007 Josselin WILLETTE">

- meta **generator** — Declares what software is used to make the site. The different software are separated by a comma.

Example

Select

< meta name="generator" content="Notepad++, Gimp">

- meta **date- creation - ddmmyyyy** — Specifies the creation date of the page in DDMMYYYY format.

Example

Select

< meta name ="date- creation - ddmmyyyy " content="01122007">

- meta **date- revision - ddmmyyyy** — Specifies the date the page was last modified in DDMMYYYY format.

Example

Select

< meta name ="date- revision - ddmmyyyy " content="01122007">

- meta **content-script-type** — Specifies the scripting language used in the page by its MIME type .

Example

Select

< meta http- equiv ="content-script-type" content=" text / javascript ">

- meta **content-style-type** — Indicates the use of a style sheet in the page and populates its MIME type .

Example

Select

< meta http- equiv ="content-style-type" content=" text / css ">

III-D-6. Orient the robots ▲

- meta **robots** — Tells crawlers whether the page should be indexed or not. Several values are possible, separated by a comma. By default, the value is all.

Values	Description
index	Allow robots to index the page.
noindex	Forbid robots to index the page.
follow	Allow robots to follow links on the page.
nofollow	Prohibits robots from following links on the page.

all	Matches "index, follow ".
none	Matches " noindex , nofollow ".
noarchive	Prohibits search engines from offering a cached version of the page.

Example

Select

< meta name ="robots" content="index, nofollow , noarchive ">

- meta **revisit-after** — Tells crawlers how many days to wait before their next visit to the site. Please note that many robots today no longer refer to this tag.

Example

Select

< meta name =" revisit-after " content="3 days ">

III-D-7. Orient visitors ▲

- meta **refresh** — Refreshes the page every x seconds where x is the value of content. If a URL is entered, then the browser will redirect the visitor after x seconds to the page indicated.

Example

Select

< meta http- equiv =" refresh " content="60">

< meta http- equiv =" refresh " content="0; url=http://j-willette.developpez.com/">

III-D-8. More... ▲

- meta **window-target** — Specifies the destination frame for all site links when using frames. Allows you to avoid filling in the target attribute on each of the links.

Example

Select

< meta http- equiv =" window-target " content="body">

- meta **pragma** — Disallow caching of pages with its only value: no-cache. Very unreliable tag, to be coupled with a server-side solution.

Example

Select

< meta http- equiv =" pragma " content="no-cache">

- meta **expires** — Populates the expiration date of the page, in the format "DD Month (YYYY)". Beyond this date, the previous page is reloaded by the browser. The page can never expire by specifying the value never .

Example

Select

< meta http- equiv ="expires" content=" never ">

< meta http- equiv ="expires" content="21 December 2007">

- meta **set-cookie** — Sets a cookie on the visitor's computer.

Example

Select

< meta http- equiv ="set-cookie" content=" dejavenu =yes; path =/; expires=Thursday, 14-April-08 00:00:00 GMT">

- meta **imagetoolbar** — Removes the toolbar that appears over images when the mouse hovers over them in Internet Explorer. Its only value is no.

Example

Select

< meta http- equiv =" imagetoolbar " content="no">

- meta **site-enter** — Creates a transition at the site entrance. Only works with Internet Explorer.
- meta **site-exit** — Creates a transition when exiting the site. Only works with Internet Explorer.
- meta **page-enter** — Creates a transition when entering the page. Only works with Internet Explorer.
- meta **page-exit** — Creates a transition when exiting the page. Only works with Internet Explorer.

Values	Description
blendTrans (duration=4)	Creates an input fade in and an output fade out.
revealTrans (duration=4,transition=0)	Creates a square in transition.
revealTrans (duration=4,transition=1)	Creates a square outgoing transition.
revealTrans (duration=4,transition=2)	Creates a round in transition.
revealTrans (duration=4,transition=3)	Creates a sortan round transition .
revealTrans (duration=4,transition=4)	Creates a curtain-up transition.
revealTrans (duration=4,transition=5)	Creates a curtain down transition.

revealTrans (duration=4,transition=6)	Creates a curtain transition to the right.
revealTrans (duration=4,transition=7)	Creates a curtain transition to the left.
revealTrans (duration=4,transition=8)	Creates a multi-curtain transition to the right.
revealTrans (duration=4,transition=9)	Creates a multi-curtain transition down.
revealTrans (duration=4,transition=10)	Creates a checkerboard transition to the right.
revealTrans (duration=4,transition=11)	Creates a downward checkerboard transition.
revealTrans (duration=4,transition=12)	Creates a brush transition .
revealTrans (duration=4,transition=13)	Creates a door transition that closes vertically.
revealTrans (duration=4,transition=14)	Creates a door transition that opens vertically.
revealTrans (duration=4,transition=15)	Creates a door transition that closes horizontally.
revealTrans (duration=4,transition=16)	Creates a door transition that opens horizontally.
revealTrans (duration=4,transition=17)	Creates a NE-SW diagonal transition.
revealTrans (duration=4,transition=18)	Creates a SE-NO diagonal transition.
revealTrans (duration=4,transition=19)	Creates a diagonal NW-SE transition.
revealTrans (duration=4,transition=20)	Creates a SO-NE diagonal transition.
revealTrans (duration=4,transition=21)	Creates a transition by horizontal lines.
revealTrans (duration=4,transition=22)	Creates a vertical line transition.
revealTrans (duration=4,transition=23)	Creates a random transition.

Example

Select

< meta http- equiv ="page-enter" content=" blendTrans (duration=2)">

You can modify the duration value as you wish, it informs the transition time.

And even more...

You may have already noticed meta tags in the source code of some sites that begin with "DC.", such as:

Example

Select

< meta name =" dc.keywords " content="list, of, keywords, keywords">

These tags are part of what is known as the Dublin Core . For more information, you can read the official translation of the Dublin Core User's Guide as well as the list of these metas . The idea of creating consistency in the creation of metadata started with a good intention, but it did not manage to impose itself in mores, no doubt because of too great a timidity in the propagation of this

idea.

Therefore, they are not widely used.

- The main title of the page

This is the title of your page, probably the most important element! Every page should have a title that describes what it contains. It is recommended that the title be quite short (less than 100 characters in general). The title is not displayed in your page but at the top of it (often in the browser tab). Save your web page and open it in your browser. You will see that the title is displayed in the tab:

- Comments

A **comment** in HTML is text that serves simply as a memo. It is not displayed, it is not read by the computer, it does not change the display of the page. You can use comments to leave indications of how your page is working . This will help you remember how your page is working if you come back to your source code after a long absence.

A comment is an HTML tag with a very special form:

Code: HTML

```
<!-- Ceci est un commentaire -->
```

You can put it wherever you want within your source code: it has no impact on your page, but you can use it to help you find your way around your source code (especially if it is long).

NB : **Everyone can see the HTML code of your page** once it is online on the Web. Just right-click on the page and select "View page source code" (the title may change depending on your browser):

CHAPTER II: ORGANIZING YOUR TEXT

General objective : to allow the student to write content in a web page using tags

Specific objectives :

- **Write paragraphs;**
- **Structure your page with titles;**
- **Give importance to certain words in their text;**
- **Organize information as a bulleted list;**

keywords : tags, paragraph, bulleted list, unordered lists, ordered lists, even tags, odd tags ...

learning methods : receptive, active collaborative, active individual

types of activity : collaborative work, presentation, practical exercises, etc.

assessment methods : before, during and after learning.

We will see how to write the content of a web page in this chapter. As we have seen, this should not be done just anyhow: we must not forget that an HTML page is made up of tags. These tags tell the computer what the text means: this is a paragraph, this is a title, and so on.

I. Paragraphs

Most of the time, when we write text in a web page, we do it inside paragraphs. The HTML language precisely offers the ** tag** to delimit paragraphs.

Code: HTML

```
<p>Bonjour et bienvenue sur mon site ! </p>
```

- ** ** stands for "Start of **Paragraph "**
- ** ** means "End of Paragraph"

The <p> tag **is** a block type tag, ie it creates a block and automatically generates a line break. It is used to define a paragraph. It can be equipped with the **align attribute** which allows you to align the content of the paragraph as you wish. The different values of the align attribute are:

- ✓ **left** : This is the default. It aligns the content of the paragraph to the left of the page (or to the left of the container that contains the <p> tag).
- ✓ **right** : It aligns the content of the paragraph to the right of the page (or to the right of the container that contains the tag).
- ✓ **center** : It allows to center the paragraph.
- ✓ **justify** : It allows to justify the content of the paragraph (extends the text so that it occupies the entire line).

❖ Skip a line

There is a "Go to newline" tag. It is an *orphan tag* which is just used to indicate that we must go to the next line: < **br** /> . You must put it **inside a paragraph** . Here's how to use it in code:

Code: HTML

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Sauts de ligne</title>
    </head>

    <body>
        <p>
Bonjour et bienvenue sur mon site ! <br />
            Ceci est mon premier test, alors soyez indulgents s'il
vous plaît, j'apprends petit à petit comment ça marche.
        </p>

        <p>
            Pour l'instant c'est un peu vide, mais revenez dans 2-3
jours quand j'aurai appris un peu plus de choses, je vous assure que
vous allez être surpris !
        </p>
    </body>
</html>
```

II. The titles

When the content of your page will expand with many paragraphs, it will become difficult for your visitors to find their way around. This is where headings come in handy. In HTML we're polished, we're allowed to use 6 different levels of headings. So we have 6 different title tags:

- **< h 1></h1>** : means "very important title". In general, it is used to display the title of the page at the beginning of this one.
- **< h 2></h2>** : means "important title".
- **< h 3></h3>** : the same, it's a slightly less important title (we can say a "subtitle" if you want).
- **< h 4></h4>** : even less important title.
- **< h 5></h5>** : title not important.
- **< h 6></h6>** : title really, but then there really not important at all.

Come on, I'll give you an example of using titles in a web page (you'll see that I don't use all the tags):

```
<html>
<head>
    <meta charset="utf-8" />
    <title>Présentation du Site du Zéro </title>
</head>

<body>
    <h1>Bienvenue sur le Site du Zéro ! </h1>

    <p>
        Le Site du Zéro vous propose des cours (tutoriels) destinés
aux débutants : aucune connaissance n'est requise pour lire ces
cours !
    </p>
    <p>
        Vous pourrez ainsi apprendre, sans rien y connaître
auparavant, à créer un site web, à programmer, à construire des
mondes en 3D !
    </p>

    <h2>Une communauté active </h2>

    <p>
        Vous avez un problème, un élément du cours que vous ne
comprenez pas ? Vous avez besoin d'aide pour créer votre site ?  <br
/>
        Rendez-vous sur les forums ! Vous y découvrirez que vous
n'êtes pas le seul dans ce cas, et vous trouverez très certainement
quelqu'un qui vous aidera aimablement à résoudre votre problème.
    </p>
</body>
</html>
```

a. Development

Within your paragraphs, certain words are sometimes more important than others and you would like to make them stand out. HTML gives you different ways to emphasize the text on your page. To highlight your text, you can use:

- Either the tag **< em ></ em > ** which has the effect of putting the text in *italics depending on whether you are on a particular browser*
- Either the tag **< strong ></ strong > ** which, on the other hand, displays the text in bold to mark the importance

Their use is quite simple, just surround the words to be highlighted with these tags, and that's it.

b. Mark text

The **<mark> tag** visually highlights a portion of text. The text is not necessarily considered important but we want it to stand out from the rest of the text. This can be useful for highlighting relevant text after a search on your site, for example. By default, **<mark>** has the effect of highlighting the text.

III. Types of lists

Lists often allow us to better structure our text and order our information. We will discover here three types of lists:

- Unordered lists
- Ordered or numbered lists
- Definition lists

a. Ordered list

An ordered list works the same way, only one tag changes: you have to replace **< ul ></ ul >** by **< ol ></ ol >** . Inside the list, we don't change anything: we still use ** tags** to delimit the elements

.

Code: HTML

```
<h1>Ma journée</h1>

<ol>
    <li>Je me lève</li>
    <li>Je mange et je bois </li>
    <li>Je retourne me coucher </li>
</ol>
```

It is possible to change the type of numbering to HTML or CSS. In HTML, it will be with the **type** attribute which can accept the following values:

- **decimal** : 1,2,3;
- **lower-alpha** : a, b, b;
- **upper-alpha** : A, B, C;
- **lower -roman** : i, ii, iii (Roman);
- **upper -roman** : I, II, III (Roman style).

For example :

```
< ol type = " lower -alpha" >
 <li> Germinal </li>
 <li> Of the social contract </li>
 <li> The Miserables </li>
</ol> _ _
```

And displays:

a. Germinal
b. Of the social contract
c. Wretched

Note that the default value is the use of numbers. Sometimes it is necessary to start numbering at a particular value. In this case, using the **start attribute** comes in handy.

```
< ol start = "10" >
 <li> Germinal </li>
 <li> Of the social contract </li>
 <li> The Miserables </li>
</ol> _ _
```

And displays:

10. Germinal
11. Of the social contract

12. Wretched

attribute **Start** of **ul** and **ol** allows for example to have several continuous lists in terms of numbering. However, it will then be necessary to make sure to enter the appropriate figures for the continuation of the incrementation.

Sometimes it is also necessary to nest one list inside another. This less common practice poses problems of readability of the page and also of the code. It is preferable to create two different lists except in certain situations such as navigation menus.

The principle is simple: each new list is entirely included in a parent **li** .

```
<ol> _ _
 <li> Germinal
  <ul> _ _
   <li> Available in epub </li>
   <li> Available in PDF </li>
  </ul> _ _
 </li>
 <li> Of the social contract </li>
 <li> The Miserables </li>
</ol> _ _
```

And displays:

1. Germinal
2. Of the social contract
3. Wretched

The appearance of the second-level list is then a little different from the first-level list, so as to avoid confusion when reading.

b. Unordered list

An unordered list looks like this:

- Strawberries
- Raspberries
- Cherries

It is a system that allows us to make a list of elements, without any notion of order (there is no "first" or "last"). Creating an unordered bulleted list is very simple. All you have to do is use the **< ul > tag** , which you close a little further with a **</ ul >** .

Like the <ol> tag , the <ul> tag has attributes that allow you to customize the list. The most important of these is the type attribute which defines the type of marker and which can have one of the following values:

- ✓ **disc** : the marker is a filled circle. This is the default value.
- ✓ **circle :** the marker is a hollow circle.
- ✓ **square** : the marker is a filled square.

c. definition

Definition or glossary lists are used in the particular case where the list is made on two levels:

- a main element to name;
- a secondary element that comes to describe it.

The most frequent case of use of this type of list is represented by dictionaries or glossaries. But it is usable in any context allowing to give details to a listed element such as a bibliographic list or an instruction list.

In HTML, this will look like this:

```
<dl>
 <dt> Geminal </dt> __ _ _
 <dd> Book written by Emile Zola </dd>
 < dt > Of the social contract </ dt >
 <dd> Written by Jean-Jacques Rousseau </dd>
 <dt> The Miserables </dt> __ _
 <dd> Major work of Victor Hugo </dd>
 <dd> Certainly the most representative and represented abroad </dd>
</dl>
```

Which will give in the browser:

Germinal

Book written by Emile Zola

Of the social contract

Written by Jean-Jacques Rousseau

Wretched

Major work of Victor Hugo

Certainly the most representative and represented abroad

Note that the code begins with the element **dl** which frames the whole list and that a sequence of **dt** and **dd** of the same level is used:

- **dt** defines the term to be described;
- **dd** is the description itself, to be placed just after the dt it describes. It can be used several times for the same dt . Descriptions are displayed indented from the term to be defined.

Some of these lists sometimes display primary information and secondary information on the **dt line** . It is then possible to use the **dfn element** to specifically point to the one described below:

```
<dl> _ _
 <dt> <dfn> Germinal </dfn> , 1885 </dt> _ _ _ _ _ _
 <dd> Book written by Emile Zola </dd>
 < dt > Of the social contract </ dt >
 <dd> Written by Jean-Jacques Rousseau </dd>
 <dt> The Miserables </dt> _ _ _
 <dd> Major work of Victor Hugo </dd>
 <dd> Certainly the most representative and represented abroad </dd>
</dl>
```

Which will be interpreted as:

Germinalis , 1885

Book written by Emile Zola

Of the social contract

Written by Jean-Jacques Rousseau

Wretched

Major work of Victor Hugo

Certainly the most representative and represented abroad

Apart from the graphic aspect which can be modified by the CSS, the use of **dfn** makes it possible to highlight the main element. If we add a unique **id attribute** to it, we can then refer to it directly through a link and use the system for glossaries, footnotes or endnotes.

and <sub> tags

The **<sup> tag** puts a text in superscript and the <sub> tag **puts it in** subscript.
For example, the following code:
x ²
gives :
x2 –
and code
H₂O
given
H2O _ _

<font> tag

The best-known text formatting tag is probably the famous **<font>. The <font>** tag owes its notoriety to the effects it can bring to the texts it includes.

It has three main attributes that allow you to apply significant effects to texts. These three attributes are: **face** , **size** , and **color** .

The face attribute

The **face attribute** lets you specify the font to use to display the text included in the **<font> tag** . By default, the "Times New Roman" font is applied by most browsers if no font is specified

The size attribute:

size attribute changes the font of the text. By default the browser applies the size 12 points (written 12pt) to fonts, with some exceptions like "Netscape Navigator" which applies the default size 10pt.

color attribute :

In a text, there is not only the font and the size, there is also the color. The **color attribute** is used to apply a color to the text surrounded by the **<font> tag** .

CHAPTER III: CREATING LINKS

General objective : to allow the student to create links between his pages.

Specific objectives :

- **Create a link to another site ;**
- **Create a link that displays a tooltip on hover;**
- **Create a link that opens a new window;**
- **Create a link to send an e-mail;**

keywords : link, site, anchor, tooltip , window ,

learning methods : receptive, active collaborative, active individual

types of activity : collaborative work, presentation, practical exercises, etc.

assessment methods : before, during and after learning.

A link is text that you can click on to go to another page.

You can make a link from a page a.html to a page b.html , but you can also make a link to another site (eg http://www.iaicameroun.com). In both cases, we will see that the operation is the same.

I. A link to another site

It is easy to recognize the links on a page: they are written in a different way (by default in underlined blue) and a cursor in the shape of a hand appears when you point to them.

To make a link, the tag we are going to use is very easy to remember: **<a>** . However, you must add an attribute, href , to indicate which page you want to take to.

If you want to make a link to another site, all you have to do is copy its address (we are talking about URL) in http://. Note that some links sometimes start with https:// (secure sites) or other prefixes (ftp://...). By default, the link is displayed in underlined blue. If you have already visited the page, the link is displayed in purple.

The links that we have just seen are called **absolute links** , because we indicate the complete address. We are now going to see that we can write the links in a slightly different way, which will be useful for us to make links between the pages of our site.

a. A link to another page on their site

i Two pages located in the same folder

Let's start by creating 2 files corresponding to 2 different HTML pages that we can call page1.html and page2.html . We will therefore have these 2 files on our disk **in the same folder.** Since the two files are located in the same folder, it is enough to simply write the name of the file to which we want to bring.

For example : **<a href="page2.html">** . It is said to be a **relative link** .

Here is the code we will use in our page1.html and page2.html files .

page1.html

Code: HTML

```
<p>      Votre paragraphe          <a href="page2.html" Nom de la page 2</a>  </p>
```

page2.html

Page 2 (landing page) will simply display a message to indicate that we have arrived on page 2:

Code: HTML

```
<h1>Bienvenue sur la page 2!  </h1>
```

ii ***Two pages located in different folders***

Let's say page2.html is in a subfolder called **content**

In this case, you will have to make a link like this:

```
<a href="contenu/page2.html" >
```

If there were several subfolders, we would write this:

```
<a href="contenu/autredossier/page2.html" >
```

If your target file is placed in a folder that is "before" in the tree structure, you must write a colon like this:

```
<a href="../page2.html" >
```

iii ***A link to an anchor***

An **anchor** is a kind of landmark that you can put in your big HTML pages . Indeed, if your page is very big it can be useful to make a link bringing you lower in the same page so that the visitor can jump directly to the part that interests him.

To create an anchor, simply add the id attribute to a tag which will then serve as a marker. It can be any tag, a title *for example* .

id attribute to name the anchor. This will then serve us to make a link to this anchor. For example :

Code: HTML

```
<h2 id="mon_ancre">Titre</h2>
```

Then just make a link as usual, but this time the href attribute will contain a hash mark (#) followed by the name of the anchor. Example :

Code: HTML

```
<a href="#mon_ancre">Aller vers l'ancre</a>
```

Normally, if you click on the link, it will take you further down the same page (provided the page has enough text for the scrollbars to move automatically).

iv Link to an anchor located in another page

The idea is to make a link that opens a new page AND that leads directly to an anchor located further down on this page. In practice it's quite simple to do: just type the name of the page, followed by a hash (#), followed by the name of the anchor.

For example : **<a href="anchors.html#rollers">** _ _ _ _ _ _

... will take you to the anchors.html page , directly to the anchor called "rollers".

Here is a page that contains 3 links, each leading to one of the anchors of the page of the previous example:

```
<h1>Le Mégamix</h1>
<p>
   Rendez-vous quelque part sur une autre page : <br />
   <a href="ancres.html#cuisine">La cuisine</a><br />
   <a href="ancres.html#rollers">Les rollers</a><br />
   <a href="ancres.html#arc">Le tir à l'arc</a><br />
</p>
```

II. A link that displays a tooltip on hover

title attribute which displays a tooltip when pointing to the link. This attribute is optional.
You will get a result that looks like this:

The tooltip can be useful to inform the visitor even before he has clicked on the link. Here is how to reproduce this result:

Code: HTML

```
<p>Bonjour. Souhaitez-vous visiter le  <a
href="http://www.siteduzero.com"  title="Réservé aux débutants" >Site
du Zéro</a> ?</p>
```

III. A link that opens a new window

It is possible to "force" the opening of a link in a new window. To do this, we will add target ="_ blank " to the **<a> tag** :

Code: HTML

```
<p>Bonjour. Souhaitez-vous visiter le  <a
href="http://www.siteduzero.com"  target="_blank">Site du Zéro</a>
?<br />
Le site s'affichera dans une autre fenêtre. </p>
```

NB : Depending on the browser configuration, the page will be displayed in a new window or a new tab. You can't choose between opening a new window or a new tab

IV. A link to send an email

mailto " type links . Nothing changes at the tag level, you just have to modify the value of the href attribute like this:

```
<p><a href="mailto:votrenom@bidule.com" >Envoyez-moi un e-mail
!</a></p>
```

So just start the link with "mailto:" and write the e-mail address where you can be contacted. If you click on the link, a new blank message opens, ready to be sent to your email address.

V. A link to download a file

In fact, you have to do exactly as if you were making a link to a web page, but this time indicating the name of the file to download.

For example , suppose you want to download myfile.zip . Simply place this file in the same folder as your web page (or in a subfolder) and link to this file:

```
<p><a href="monfichier.zip" >Télécharger le fichier </a></p>
```

The browser, seeing that it is not a web page to display, will launch the download procedure when the link is clicked.

CHAPTER IV: IMAGES

There are different image formats that you can use on websites, and you shouldn't choose them at random. Indeed, the images are sometimes large to download, which slows down the loading time of the page (much more than the text!).

General objective : to allow the student to insert images into a page .

Specific objectives :

- **Identify the format of an image ;**
- **Insert an image;**
- **Add a tooltip in an image;**
- **Make a thumbnail clickable;**

keywords : image formats, clickable thumbnail, tooltip , src , alt …

learning methods : receptive, active collaborative, active individual

types of activity : collaborative work, presentation, practical exercises, etc.

assessment methods : before, during and after learning.

I. The different image formats

1) JPEG

Images in the *Joint Photographic Expert Group (JPEG*) format are very common on the web. This format is designed to reduce the size of photos, which can contain more than 16 million different colors. JPEG images are saved with the extension . jpg or .jpeg .

2) PNG

The PNG (*Portable Network Graphics*) format is the newest of all. This format is suitable for most graphics (I would be tempted to say "everything that is not a photo"). PNG has two big advantages: it can be made transparent and it does not alter the quality of the image .

- PNG exists in 2 versions, depending on the number of colors that the 8-bit PNG image must contain: 256 colors
- 24-bit PNG: 16 million colors (as many as a JPEG image)

3) The GIF

It is a fairly old format, which has nevertheless been widely used (and which remains widely used out of habit).

The GIF format is limited to 256 colors (while the PNG can go up to several million colors). Nevertheless, the GIF retains a certain advantage that the PNG does not have: it can be animated.

II. Inserting an image

< img /> tag that we will use here. It is an *orphan* type tag (like **< br />**). This means that you don't have to write it twice like most other tags. Indeed, we don't need to delimit a portion of text, we just want to insert an image at a specific location.

The tag must be accompanied by 2 mandatory attributes:

- **src** : it allows you to indicate where the image you want to insert is located. You can either put a path in absolute (ex.: http://www.site.com/fleur.png), or put the path in relative (which we do most often). So, if your image is in an images sub-folder you will have to type: src ="images/flower.png"
- **alt** : this means "alternative text". You must **always** indicate an alternative text to the image, that is to say a short text which describes what the image contains. This text will be displayed instead

of the image if the image cannot be downloaded. It also helps search engine crawlers for image searches. For the flower, we would put for example: alt ="A flower".

Images must be inside a paragraph ( ). Here is an example of inserting an image:

```
<p>
     Votre paragraphe
          <br />
     <img src="images/montagne.jpg"  alt="une fleur(par exemple)"/>
</p>
```

1) Add a tooltip

The attribute used to display a tooltip is the same as for links: it is **title .** This attribute is optional (unlike **alt).**

Here's what it can do:

Code: HTML

```
<p>
     Votre paragraphe
          <br />

          <img src="…"  alt= "…"
     title="…  />
     </p>
```

Mouse over the photo to see the tooltip appear.

2) Clickable thumbnail

If your image is very large, it is advisable to display the thumbnail of it on your site. Then add a link to this thumbnail so that your visitors can view the image in its original size.

There are millions of software for creating image thumbnails. Like for example EasyThumbnails . I will thus have 2 versions of my photo: the thumbnail and the original image.

I place them both in a folder called img for example. I display the mountain_mini.jpg version on my page and I make a link to mountain.jpg so that the enlarged image is displayed when you click on the thumbnail.

Here is the HTML code I will use for this:

```
<p>
     Vous souhaitez voir l'image dans sa taille d'origine ? Cliquez
dessus !<br />
     <a href="img/montagne.jpg"><img src="img/montagne_mini.jpg"
alt="Photo de montagne"  title="Cliquez pour agrandir"  /></a>
</p>
```

Inserting a video

To add a video to your website, you must use the <video> tag provided for this purpose. **You can use the video tag in several ways** , first, by specifying the source directly on the tag like this:

```
< video controls src =" video /my-video.mp4" > The text to display if the video does
not load </video>
```

You notice that this time unlike the <img> tag , this tag is not orphan, it has an opening and closing tag . It is also between these tags that you can **specify a text that is only displayed if the video does not load** , or if the user's browser is too old to read this HTML tag.

The other way to display a video with the <video> tag is to assign **multiple sources** to it , so your website can **serve the video in multiple formats** .

do this , simply remove the src attribute from the video tag and place <source> tags inside the <video> tag , like this:

```
< video controls >
        < source src =" videos /my-video.mp4" type=" video /mp4">
        < source src =" videos /my- video.webm " type=" video / webm ">
```

```
        The text to display if the video does not load
</video> _ _
```

You may have noticed, but the video tag has a `controls attribute` . This attribute is only used to **display a player with control buttons** directly on the video. This parameter is not mandatory, you can remove it. Also note that **the interface of the video player will depend on the browser** chosen by the user.

It is also possible to enable **video autoplay with the** `autoplay` attribute , however some browsers may disable autoplay, so you won't be able to do anything about it if your video doesn't play automatically.

CHAPTER IV: TABLES

I. A simple table

The first tag to know is **<table></table>** . This tag is used to indicate the start and end of a table.

This tag is of the block type, so it must be placed outside a paragraph. Example :

Code: HTML

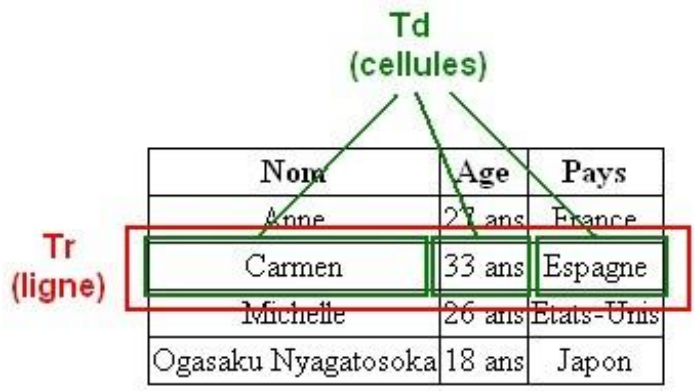
```
<p>Ceci est un paragraphe avant le tableau. </p>

<table>
   <!-- Ici, on écrira le contenu du tableau  -->
</table>

<p>Ceci est un paragraphe après le tableau. </p>
```

Then we have the following tags:

- **< tr ></tr>** : indicates the beginning and the end of a row in the table.
- **<td> </td> : Indicates the start and end of** cell content.

In HTML, your table is built line by line. In each line (**<tr>**), we indicate the contents of the different cells (**<td>**).

Schematically, our table is built like this:



We have a line tag (**<tr>**) that encompasses each of the cells (**<td>**)

For example, if I want to make a two-line table, with 3 cells per line (so 3 columns), I would type this:

Code: HTML

```
<table>
   <tr>
       <td>Carmen</td>
       <td>33ans </td>
       <td>Espagne</td>
   </tr>
   <tr>
       <td>Michelle</td>
       <td>26ans </td>
       <td>Etats-Unis</td>
   </tr>
</table>
```

The result is a little depressing:

Carmen 33 ans Espagne
Michelle 26 ans Etats-Unis

Yes, a table without CSS looks very empty. So precisely, adding borders is very simple, you already know the corresponding CSS code!

Code: CSS

```
td /* Toutes les cellules des tableaux... */
{
    border: 1px solid black; /* ... auront une bordure de 1px */
}
```

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Indeed, we would like there to be only one border between 2 cells, but this is not the case here.
Luckily, there is a table-specific CSS property: **border-collapse** , which means "glue the borders together".
This property can take 2 values:

- **collapse** : the borders will be glued together, this is the effect we are looking for.

- **separate** : the borders will be separated (default value)

Code: CSS

```
table
{
    border-collapse : collapse; /* Les bordures du tableau seront
collées (plus joli) */
}
td
{
    border: 1px solid black;
}
```

This is better!

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

1. The header line

Now, we are going to add the header row of the table. In our example, the headers are "Name", "Age" and "Country".

The header line is created with a **<tr>** as we have done so far, but the cells inside are this time <th> **and** not **<td>** !

Code: HTML

```
<table>
   <tr>
        <th>Nom</th>
        <th>Age</th>
        <th>Pays</th>
   </tr>

   <tr>
        <td>Carmen</td>
        <td>33ans </td>
        <td>Espagne</td>
   </tr>
   <tr>
        <td>Michelle</td>
        <td>26ans </td>
        <td>Etats-Unis</td>
   </tr>
</table>
```

The header line is very easy to recognize for 2 reasons:

- The cells are **<th>** instead of the usual **<td>** .

- This is the first row of the table

Since the cell names are a bit different for the header, consider updating the CSS to tell it to apply a border on the normal cells AND on the header:

Code: CSS

```css
table
{
    border-collapse : collapse;
}
td, th /* Mettre une bordure sur les td ET les th */
{
    border: 1px solid black;
}
```

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

As you can see, the text in the header cells is bold. This is what browsers generally do, but if you wish you can change it with CSS: modify the background color of the header cells, their font, their border, etc.

2. Table title

The title allows the visitor to quickly learn about the content of the table. In our example, we have a list of people... yes, but then? What does it represent? Fortunately **,** there is **<caption> !**

This tag is placed at the very beginning of the table, just before the header. It is she who indicates the title of the table:

Code: HTML

```html
    <tr>
        <td>Michelle</td>
        <td>26ans </td>
        <td>Etats-Unis</td>
    </tr>
</table>
```

```
        </tr>
        <tr>
            <td>Carmen</td
            <td>33ans </td
            <td>Espagne</t
        </tr>
```

Passagers du vol 377

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Note that you can change the position of the title with the CSS **caption-side property** , which can take four values:

- ✓ **top** : the title will be placed at the top of the table (by default).
- ✓ **bottom** : the title will be placed at the bottom of the table.
- ✓ **left** : the title will be placed on the left of the table.
- ✓ **right** : the title will be placed to the right of the table.

II. A structured table

We learned to build simple little paintings. These small tables are enough in most cases, but there will be times when you need to make more...complex tables.
We will discover 2 specific techniques:

- ✓ For large tables, it is possible to **divide them** into 3 parts:
  - On your mind
  - Table body
  - Table foot
- ✓ For some tables, you may need to **merge** cells together.

III. Split a large array

If your table is large enough, you will have every interest in cutting it into several parts. For this, there are HTML tags that allow you to define the 3 "zones" of the table:

- ✓ **The header (at the top)** : it is defined with the tags **< thead ></ thead >**
- ✓ **The body (in the center)** : it is defined with the tags **< tbody ></ tbody >**
- ✓ **The foot of the table (bottom)** : it is defined with the tags **< tfoot ></ tfoot >**

Generally, if it's a long table, you copy the header cells into it. This allows you to see even at the bottom of the table what each of the columns relates to.
Schematically, a table in 3 parts is therefore divided like this:

It's a bit confusing, but it's a good idea to write the tags in this order:

We therefore put in the code first the top part, then the bottom part, and finally the main part (**< tbody >**). The browser will take care of displaying the elements in the right place.

/85

4 here is the code to write to build the table in 3 parts:

Code: HTML

```
<table>
   <caption>Passagers du vol 377</caption>

   <thead> <!-- En-tête du tableau -->
       <tr>
           <th>Nom</th>
           <th>Age</th>
           <th>Pays</th>
       </tr>
   </thead>

   <tfoot> <!-- Pied de tableau -->
       <tr>
           <th>Nom</th>
           <th>Age</th>
           <th>Pays</th>
       </tr>
   </tfoot>

   <tbody> <!-- Corps du tableau -->
       <tr>
           <td>Carmen</td>
           <td>33 ans</td>
           <td>Espagne</td>
       </tr>
       <tr>
           <td>Michelle</td>
           <td>26 ans</td>
           <td>Etats-Unis</td>
       </tr>
       <tr>
           <td>François</td>
           <td>43 ans</td>
           <td>France</td>
       </tr>
       <tr>
           <td>Martine</td>
           <td>34ans </td>
           <td>France</td>
       </tr>
       <tr>
           <td>Jonathan</td>
           <td>13ans </td>
           <td>Australie</td>
       </tr>
       <tr>
           <td>Xu</td>
           <td>19ans </td>
           <td>Chine</td>
       </tr>
   </tbody>
</table>
```

< > ,

❖ How to merge table cells?

On some complex tables, you will need to "merge" cells together.

An example of merging? Take a look at this table listing the films and who they are for:

Titre du film	Pour enfants ?	Pour adolescents ?
Massacre à la tronçonneuse	Non, trop violent	Oui
Les bisounours font du ski	Oui, adapté	Pas assez violent...
Lucky Luke, seul contre tous	Pour toute la famille !	

For the last film, you see that the cells have been merged: they are now one. This is exactly the effect we are trying to achieve.

To perform a merge, you must add an attribute to the **<td> tag** . You should know that there are 2 types of fusion:

- ✓ **Merging columns** : this is what I just did on this example. The merge is done horizontally. We will use the **colspan** attribute .
- ✓ **The merging of lines** : here, two lines will be grouped together. The merge will take place vertically. We will use the **rowspan attribute** .

As you know, you have to give the attribute a value (be it colspan or rowspan). You must indicate the number of cells to be merged together. In our example, we have merged two cells: that of the "For children?" column, and that of "For teenagers?". We must therefore write:

Code: HTML

```
<td colspan="2">
```

... which means: " *This cell is the fusion of 2 cells* ". It is possible to merge more cells at once (3, 4, 5... as many as you want).

Here is the HTML code that allows me to perform the merge corresponding to the previous table:

Code: HTML

```
<table>
   <tr>
       <th>Titre du film</th>
       <th>Pour enfants ?</th>
       <th>Pour adolescents ?</th>
   </tr>
   <tr>
       <td>Massacre à la tronçonneuse </td>
       <td >Non, trop violent</td>
       <td>Oui</td>
   </tr>
   <tr>
       <td>Les bisounours font du ski </td>
       <td>Oui, adapté</td>
       <td>Pas assez violent... </td>
   </tr>
   <tr>
       <td>Lucky Luke, seul contre tous </td>
<td colspan="2">Pour toute la famille ! </td>
   </tr>
</table>
```

An important note: you see that the last line contains only 2 cells instead of 3 (there are only 2 **<td> tags**). This is completely normal, because I merged the last two cells together. The < **td** colspan = "2" > indicates that this cell takes the place of 2 cells at a time.

For our example, we are going to "reverse" the order of our table: instead of putting the movie titles on the left, we are going to place them at the top.

It's another way of looking at the painting: instead of building it vertically, you can build it lengthwise.

In this case, the **colspan** is no longer suitable, it is a **rowspan** that must be used:

Code: HTML

```
<table>
   <tr>
       <th>Titre du film</th>
       <td>Massacre à la tronçonneuse </td>
       <td>Les bisounours font du ski </td>
       <td>Lucky Luke, seul contre tous </td>
   </tr>
   <tr>
       <th>Pour enfants ?</th>
       <td>Non, trop violent</td>
       <td>Oui, adapté</td>
       <td rowspan="2">Pour toute la famille ! </td>
   </tr>
   <tr>
       <th>Pour adolescents ?</th>
       <td>Oui</td>
       <td>Pas assez violent... </td>
   </tr>
</table>
```

Result: the cells are merged vertically!

Titre du film	Massacre à la tronçonneuse	Les bisounours font du ski	Lucky Luke, seul contre tous
Pour enfants ?	Non, trop violent	Oui, adapté	Pour toute la famille !
Pour adolescents ?	Oui	Pas assez violent...	

NB : Note that you can modify the vertical alignment of text in table cells, with the **vertical- align property** that we discovered in the chapter on layout.

This concludes our overview of the paintings. The way to create them may not be natural, I admit, but you get used to it quickly.

I advise you above all to check that your tags open and close in the correct order, it is very important.

For example, NEVER put a **<td> tag** if it is not surrounded by a **<tr> line tag** .

Finally, I've told you before and I'll say it again: a table without CSS is... not very aesthetic

CHAPTER V: FORMS

Any HTML page can be enriched with interactive forms, which invite your visitors to fill in information: enter text, select options, validate with a button...
However, we come to the limits of HTML language, because it is then necessary to be able to analyze the information that the visitor has entered... and this cannot be done in HTML language. The processing of the results must be done in another language as we will see, such as PHP.

I. How to create a form

To insert a form into your HTML page, you must first write a **< form ></ form > tag** . It is the main tag of the form, it allows to indicate the beginning and the end.

Code: HTML

```
<p>Texte avant le formulaire</p>

<form>
   <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```

NB :Notez qu'il faut obligatoirement mettre des balises de type bloc à l'intérieur de votre formulaire si vous souhaitez écrire du texte à l'intérieur.

comment posted by a visitor in a forum via a form, you must add 2 attributes to the **<form> tag**

- **method** : this attribute indicates by which means the data will be sent. There are 2 ways to send data over the web:
  - **method = " get "** : this method is generally not very suitable, because it is limited to 255 characters. The particularity comes from the fact that the information will be sent in the address of the page (http://...);
  - **method ="post"** : this is the most used method for forms because you can send a large amount of information using it. Data entered in the form does not pass through the address bar.

- **action** : this is the address of the page or program that will *process* the information. This page will take care of sending you an email with the message if that's what you want, or save the message with all the others in a database.

This cannot be done in HTML and CSS, we will generally use another language that you may have heard of: PHP.

are now going to complete the <form> tag **with the** 2 attributes we have just seen.

For **method** , we will put the value " **post** ".

For **action** , we will type the name of a fictitious page in PHP (treatment.php). This is the page that will be called when the visitor clicks on the "Send form" button.

Code: HTML

```
<p>Texte avant le formulaire</p>

<form method="post" action="traitement.php">
   <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```

A. Basic input areas

We are going to review the different HTML tags used to enter text in a form. You should know that there are 2 different text boxes:

- **single-line text area** : as its name suggests, you can only write one line inside. It is used to enter short texts, such as: "Enter your nickname".
- **multi-line text area** : this text area allows you to write a large amount of text on several lines, such as: "Write an essay on the usefulness of HTML in the development of Southeast Asian countries"

1. Single line text box

This is what a single line text box looks like :

To insert a one-line text box, we will use the **<input/> tag** .

To create a one-line text box, you must write:

Code: HTML

```
<input type="text" />
```

This is still not enough: you must give a name to your text area. This name does not appear on the page, but you will need it later. Indeed, this will allow you (in PHP for example) to recognize where the information comes from: you will know that such text is the visitor's nickname, such text is the password he has chosen, etc.

To give a name to a form element, we use the **name attribute** . Here, we will assume that we ask the visitor to enter his nickname:

Code: HTML

```
<input type="text" name="pseudo" />
```

So let's try to create a very basic form with our text field: **Code: HTML**

```
<form method="post" action="traitement.php">
   <p><input type="text" name="pseudo" /></p>
</form>
```

a. Labels

To insert a label in a form, we will use the **<label> tag** :

Code: HTML

```
<form method="post" action="traitement.php">
    <p>
        <label>Votre pseudo</label> : <input type="text"
name="pseudo" />
    </p>
</form>
```

But that's not enough. You have to link the label with the text area.

To do this, you must give a name to the text area, not with the name attribute but with the **id attribute** (which can be used on all tags)

To bind the label to the field, it must be given an attribute **for** which has the same value as the id of the field.

Example :

Code: HTML

```
<form method="post" action="traitement.php">
    <p>
        <label for="pseudo">Votre pseudo</label> : <input type="text"
name="pseudo" id="pseudo" />
    </p>
</form>
```

❖ Some additional attributes

We can place a number of other attributes on the **<input/> tag** to customize how it works:

- ✓ You can enlarge the field with **size** .
- ✓ You can limit the number of characters you can enter with **maxlength** .
- ✓ You can pre-fill the field with a default value with **value** .
- ✓ You can give an indication of the content of the field with **placeholder** . This indication will disappear as soon as the visitor clicks inside the field.

In the following example, the text area contains an indication allowing you to understand what to enter, it is 30 characters long but you can only write a maximum of 10 characters inside:

Code: HTML

```
<form method="post" action="traitement.php">
    <p>
        <label for="pseudo">Votre pseudo </label>
       <input type="text" name="pseudo" id="pseudo" placeholder="Ex
:Zozor"  size="30" maxlength="10" />
    </p>
</form>
```

b. Password area

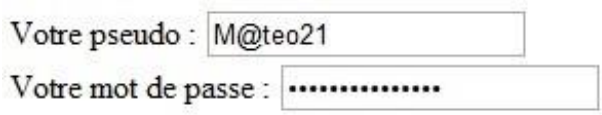
You can easily make the text area behave like a "password" area, that is, an area where the typed characters cannot be seen on the screen. To create this type of input box, use the **type=" password** " attribute .

I complete my form. It now asks the visitor for their username AND password

Code: HTML

You will see that the characters are not showing on the screen.

```
<form method="post" action="traitement.php">
   <p>
       <label for="pseudo">Votre pseudo </label>
       <input type="text" name="pseudo" id="pseudo" />

       <br />
       <label for="pass">Votre mot de passe </label>
       <input type="password" name="pass" id="pass" />

   </p>
</form>
```



2. Multiline text box

To create a multiline text area **,** we change the tag: we will use **<textarea> </textarea> .**

As with any other element of the form, you must give it a name with **name** , and use a **label** that explains what it is.

Code: HTML

```
<form method="post" action="traitement.php">
   <p>
       <label for="ameliorer">Comment pensez-vous que je pourrais
améliorer mon site </label><br />
       <textarea name="ameliorer" id="ameliorer"></textarea>
   </p>
</form>
```

Comment pensez-vous que je pourrais améliorer mon site ?

You can change the size of the < **textarea** > in 2 different ways:

- **In CSS** : just apply the **width** and **height CSS properties** to the < **textarea** > .
- **With attributes : you can** add the **rows** and **cols attributes** to the **<textarea> tag** . The first indicates the number of lines of text that can be displayed simultaneously, and the second the number of columns.

You can prefill the **<textarea> with** a default value **.** In this case, we do not use the value attribute : we simply write the default text between the opening tag and the closing tag!

Code: HTML

```
< form method = " post" action = "processing.ph ><p>
< label for = " improve " > How do you think I can improve my site? </label>< br />
< textarea name = " improve" id = " improve" rows = "10" cols= "50 " > Improve
your site?!
But finally ! He's so awesome that he's not
nécessaire de l'améliorer</textarea>
</p>
</form>
```

Comment pensez-vous que je puisse améliorer mon site ?

Améliorer ton site ?!
Mais enfin ! Il est tellement génialissime qu'il n'est pas nécessaire de l'améliorer !

B. Rich input fields

HTML5 brings many new features to forms. New types of controls have indeed appeared with this version. Just give the type attribute of the **<input/>** tag one of the new values available.

a. E-mail

You can request to enter an email address :

Code: HTML

```
<input type="email" />
```

The field will look identical to you, but your browser now knows that the user must enter an email address.
It can display an indication if the address is not an e-mail, this is what Firefox does for example:

b. A URL

With the url type , you can ask to enter an absolute address (usually starting with http://):

Code: HTML

```
<input type="url" />
```

Same principle: if the field does not look different on your computer, know that your computer does understand that the visitor is supposed to enter an address.

c. Phone number

This field is dedicated to entering telephone numbers:

Code: HTML

```
<input type="tel" />
```

d. Number

This field allows you to enter an integer:

Code: HTML

```
<input type="number" />
```

The field will usually display with small arrows to change the value:

You can customize how the field works with the following attributes:

- **Min** : minimum authorized value;
- **Max** : maximum authorized value;
- **Step** : this is the "step" of movement. If you indicate a step of 2, the control will only accept values of 2 by 2 (for example 0, 2, 4, 6...).

e. A slider

The range type allows you to select a number with a slider (also called *slider*):

Code: HTML

```
<input type="range" />
```

Here too you can use the min , max and step attributes to define the limits that can be selected.

f. Color

This field allows you to enter a color:

Code: HTML

```
<input type="color" />
```

In practice, it remains poorly implemented by browsers at present. Don't be surprised if you only see a classic text field.

g. Date

Different types of date selection fields exist:

- Date: for the date (08/07/2010)
- Time: for the time (13:50)
- Week : for the week
- Month : for the month
- Datetime -local: for the date and time (without time difference management)
- Datetime : for the date and time (with time difference management)

Example :

Code: HTML

```
<input type="date" />
```

Currently, few browsers support this type of control apart from Opera.

n. Research

You can create a search field like this:

Code: HTML

```
<input type="search" />
```

The browser then decides how to display the search field. He can add a small magnifying glass to the field to indicate that it is a search field for example, and possibly memorize the last searches carried out by the visitor.

C. Option items

HTML gives you a host of option elements to use in your form. These are elements that ask the visitor to choose from a list of possibilities. We will see :

- Checkboxes
- Options areas
- drop-down lists

i. Checkboxes

To create a checkbox, we will reuse the **<input/> tag** , this time specifying the **checkbox type** :

Add a well-placed **<label>** , and voila!

Code: HTML

```
<input type="checkbox" name="choix" />
```

Code: HTML

```
<form method="post" action="traitement.php">
   <p>
        Cochez les aliments que vous aimez manger :<br />
        <input type="checkbox" name="frites" id="frites" /> <label
for="frites">Frites</label><br />
        <input type="checkbox" name="steak" id="steak" /> <label
for="steak">Steak haché</label><br />
        <input type="checkbox" name="epinards" id="epinards" />
<label for="epinards">Epinards</label><br />
        <input type="checkbox" name="huitres" id="huitres" /> <label
for="huitres">Huitres</label>
   </p>
</form>
```

Don't forget to give a different name to each checkbox, this will allow you to identify later which boxes the visitor has checked.

Cochez les aliments que vous aimez manger :
- [x] Frites
- [x] Steak haché
- [ ] Epinards
- [ ] Huitres

Finally, be aware that you can make a box checked by default with the **checked attribute** :

Code: HTML

```
<input type="checkbox" name="choix" checked />
```

ii. Options areas

Option boxes allow you to choose (and only one) from a list of possibilities. They look a bit like checkboxes, but there's a little extra difficulty: they have to be organized into groups. The same option group has the same name (name), but each option must have a different value (value).

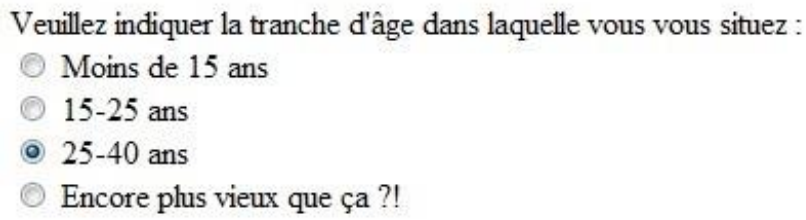
The tag to use is always an **<input/>** , this time with the radio value for the type attribute .

Example :

Code: HTML

```
< form method = " post" action = "
process.php " ><p>
the age range you fall into :
```

```
< input type = " radio" name = " age" value = "minus15" id=
"minus15"
/>< label for = "under15" > Under 15 </label>< br
/>
< input type = " radio" name = " age" value =
"medium15-25"
id = "medium15-25" />< label for = "medium15-25" > 15-25
years old </label>< br />

< input type = " radio" name = " age" value = "medium25-40"
id = "medium25-40" />< label for = "medium25-40" > 25-40
years old </label>< br />
< input type = " radio" name = " age" value = "plus40" id=
"plus40" />
< label for = "plus40" > Even older than that?! </label>
p ><
```



Why did you put the same name for each option? Quite simply so that the browser knows which "group" the button belongs to.

Try to remove the name attributes , you will see that it becomes possible to select all option elements. But that's not what we want, that's why we "link" them together by giving them an identical name.

You will notice that this time we chose a different id from name . Indeed, the names being identical, we could not have differentiated them (and you know very well that an id must be unique!). This is why we chose to set the id to the same value as value .

If you have 2 different option zones, you must give a unique name to each group like this: Code: HTML

```
< form method = " post" action = "
processing.php " ><p>
the age range you fall into :
< input type = " radio" name = " age" value = "minus15" id=
"minus15"
/>< label for = "under15" > Under 15 </label>< br
/>< input type = " radio" name = " age" value =
"medium15-25"
id = "medium15-25" />< label for = "medium15-25" > 15-25
years old </label>< br />
```

```
< input type = " radio" name = " age" value = "medium25-40"
id = "medium25-40" />< label for = "medium25-40" > 25-40
years old </label>< br />
< input type = " radio" name = " age" value = "plus40" id=
"plus40" />
< label for = "plus40" > Even older than that?! </label>
</p>
<p> _ _
What continent do you live on ? <Br /> _ _
< input type = " radio" name = " continent" value = "
europe " id= "europe" /><label for= "europe" > Europe
</label><br/><input type= "radio" name= "continent" value=
"africa" id= "africa" /><label for= "africa" > Africa
</label><br/><input type= "radio" name= "continent" value=
"asia" id = "asia"
/>< label for = " asia " > Asia </label>< br />
< input type = " radio" name = " continent" value = "
america " id= "america" /><label for= "america" > America
</label><br/><input type= "radio" name= "continent" value=
"australia" id= " australia " /> label for = " australia
" > Australia </label>
</p>
</ form >
```

`checked` attribute is again available to select a default value.

iii. drop-down lists

Drop-down lists are another elegant way to choose from several possibilities. The operation is a little different. We will use the **<select></select>** tag which indicates the beginning and the end of the drop-down list. We add the name attribute to the tag to give a name to the list.

Then, inside the **<select></select>** , we will place several **<option></option> tags** (one per possible choice). We add to each of them a value attribute to be able to identify what the visitor has chosen.

Here is an example of use:

Code: HTML

```
< form method = " post" action = " processing.php " >
<p> _ _
< label for = "country" > In which country do you live? </label><
br />
< select name = " country" id = "country" >
< option value = " france " > France </option>
< option value = " spain " > Spain </option>
< option value = " italy " > Italy </option>
< option value = " uk " > UK </option>
< option value = "canada" > Canada </option>
```

```
< option value = " usa " > USA </option>
<   option   value   =   "china"   >   China
</option><option   value=   "japan"   >   Japan
</option>
</select>
</p>
</ form >
```

selected attribute this time :

Code: HTML

```
<option value="canada" selected>Canada</option>
```

You can **also** group your options with the **<optgroup> </optgroup> tag .** In our example, why not separate the countries according to their continent?

Code: HTML

```
< form method = " post" action = "
processing.php " ><p>
< label for = "country" > In which country do you live? </label><
br />
< select name = " country" id = "country" >
< optgroup label = "Europe" >
< option value = " france " > France </option>
< option value = " spain " > Spain </option>
< option value = " italy " > Italy </option>
< option value = " uk " > UK </option> </optgroup> _
< optgroup label = "America" >
< option value = "canada" > Canada </option>
< option value = " usa " > USA </option> </optgroup> _
< optgroup label = "Asia" >
< option value = "china" > China </option><option
value= "japan" > Japan </option>
</optgroup> _ _
```

```
</select>
</p>
</ form >
```

NB : Groups cannot be selected. In our example, we cannot choose "Europe" for example, only country names are selectable.

II. Finalize and send the form

1. Group fields

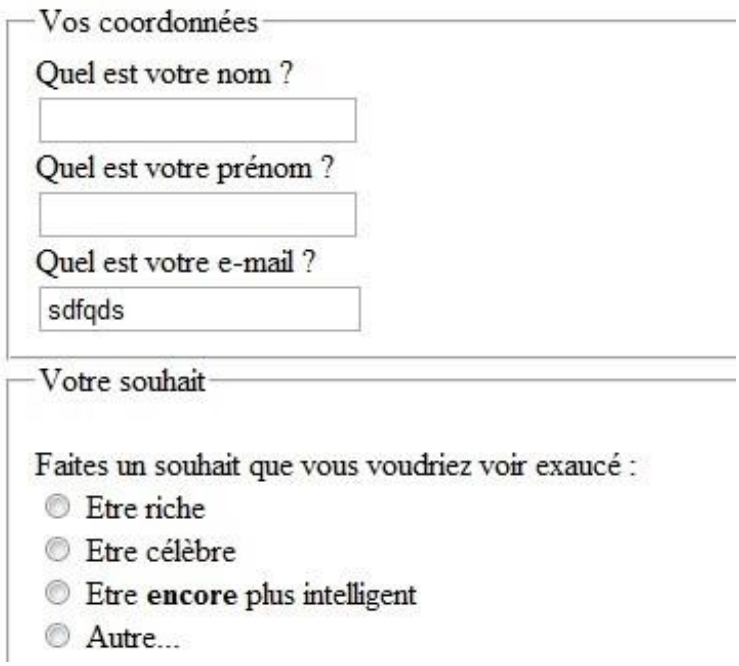
If your form is large and has a lot of fields, it may be useful to group them into several **< fieldset > tags** . Each **<fieldset> can contain a** legend with **the <legend> tag** .

Example :

Code: HTML

```
< form method = " post" action = "
processing.php " >
< fieldset >
< legend > Your contact details </ legend > <!-- Title of the fieldset
-->

< label for = "name" > What is your name? </label>
< input type = " text" name = " name" id = "name" />

< label for = " first name " > What is your first name? </label>
< input type = " text" name = " firstname" id
= " firstname " />
< label for = "email" > What is your email? </label>
< input type = " email" name = " email" id = "email" />
```

```
</fieldset> _ _

< fieldset >
< legend> Yourwish </legend> <!-- Fieldset title -- >
<p> _ _
Make a wish that you would like to see granted:

< input type = " radio" name = " wish" value = "rich" id=
"rich" />< label for = "rich" > To be rich </label>
< input type = " radio " name = " wish" value = " famous
" id = " famous " />< label for = " famous " > To be
famous </label><input type = " radio " name = " wish"
value= "smart" id= "smart" />< label for = "smart" > Being
< strong > even </ strong > smarter </label>
< input type = " radio" name = " wish" value = "other"
id="autre" /> <label for="autre">Autre...</label>
      </p>

      <p>
         <label for="precisions">Si "Autre", veuillez préciser
:</label>
         <textarea name="precisions"id="precisions"cols="40"
rows="4"></textarea>
      </p>
   </fieldset>
</form>
```



2. Automatically select a field

You can automatically place the cursor in one of your form fields with the autofocus attribute . As soon as the visitor loads the page, the cursor will move to this field.

For example, to have the cursor in the "First name" field by default :

Code: HTML

```
<input type="text" name="prenom" id="prenom" autofocus />
```

3. Make a field required

You can make a field mandatory by giving it the **required attribute .**

Code: HTML

```
<input type="text" name="prenom" id="prenom" required />
```

The browser will then indicate to the visitor that he must fill in the field if it is empty at the time of sending.

4. The send button

Now we just have to create the submit button. Here again, the **<input/> tag** comes to our rescue. It exists in 4 versions:

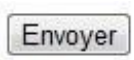
- **type =" submit "** : the main form submit button. This is the one you will use most often. The visitor will be sent to the page specified in the action attribute of the form.
- **type ="reset"** : reset of the form.
- **type ="image** " : equivalent of the " submit " button, presented this time in the form of an image. Add the src attribute to indicate the URL of the image.
- **type =" button "** : generic button, which will (by default) have no effect. In general, this button is managed in JavaScript to perform actions on the page. We will not use it here.

NB: You can change the text displayed inside the buttons with the value attribute .

To create a send button, we will therefore write for example:

Code: HTML

```
<input type="submit" value="Envoyer" />
```



When you click on the "Send" button, the form takes you to the page indicated in the action attribute . We had imagined a fictitious page (treatment.php), remember.

The problem is that you can't create this page just in HTML. It is necessary to learn a new language, such as PHP , to be able to "retrieve" the information entered and decide what to do with it.

PARTIE 2 : LE CSS

CHAPTER I: THE JOY OF FORMATTING WITH CSS

General objective : to allow the student to have the basics with regard to css .

Specific objectives :

- **Define the css ;**
- **Know where to insert the css in/on a page;**

Keywords : css , stylesheet, property, value,
Learning methods : receptive, active collaborative, active individual
Types of activity : collaborative work, presentation, practical exercises, etc.
Evaluation methods : before, during and after learning.

I. Importance of CSS

The **CSS** allows us to choose the color of your text. It is this that allows us to select the font used on our site. It is still the one that allows you to define the size of the text, the borders, the background...And also, it is the one that allows you to make the layout of your site. You can say: I want my menu to be on the left and occupy such a width, that the header of my site is wedged at the top and that it is always visible, etc.

II. Where do you write the CSS ?

You have the choice, because we can write code in CSS language in 3 different places:

- In a file . css (most recommended method);
- **In** the **<head> of** the HTML file;
- Directly in the tags of the HTML file via a style attribute (**the least recommended method**);

1) In a file . css (recommended)

As we have just said, most often we write CSS code in a special file with the extension . css (unlike HTML files which have the .html extension). This is the most convenient and flexible method. It saves us from mixing everything in the same file.

You will notice the line < **link** rel = " stylesheet" href = "style.css" /> : it is this which indicates that this HTML file will be associated with a file called style.css which will manage its formatting. This line is directly after the < meta line charset ="utf-8"/>.

2) In the header < head > of the HTML file

in your HTML files is to insert the CSS code directly into a **<style> tag** inside the **<head> header** .

3) Directly in beacons (not recommended)

Last method, to handle with care: you can add a style attribute to any tag. You will insert your CSS code directly into this attribute:

Schematically, a CSS style sheet looks like this:

```
balise1
{
   propriete : valeur;
   propriete : valeur;
   propriete : valeur;
}

balise2
{
   propriete : valeur;
   propriete : valeur;
   propriete : valeur;
   propriete : valeur;
}

balise3
{
   propriete : valeur;
}
```

In a CSS code like this, there are 3 different elements:

- ✓ **Tag names** : we write the names of the tags whose appearance we want to modify. For example, if I want to change the appearance of all **<p>** paragraphs , I have to write **p** .
- ✓ **CSS properties** : the "style effects" of the page are stored in properties. There is for example the **color property** which allows to indicate the color of the text, **font-size** which allows to indicate the size of the text, etc. There are many CSS properties.
- ✓ **Values** : each CSS property must be given a value. For example, for the color, you must indicate the name of the color. For the size, you have to indicate what size you want, etc.

Example: **Code: CSS**

```
p
{
    color: blue;
}
```

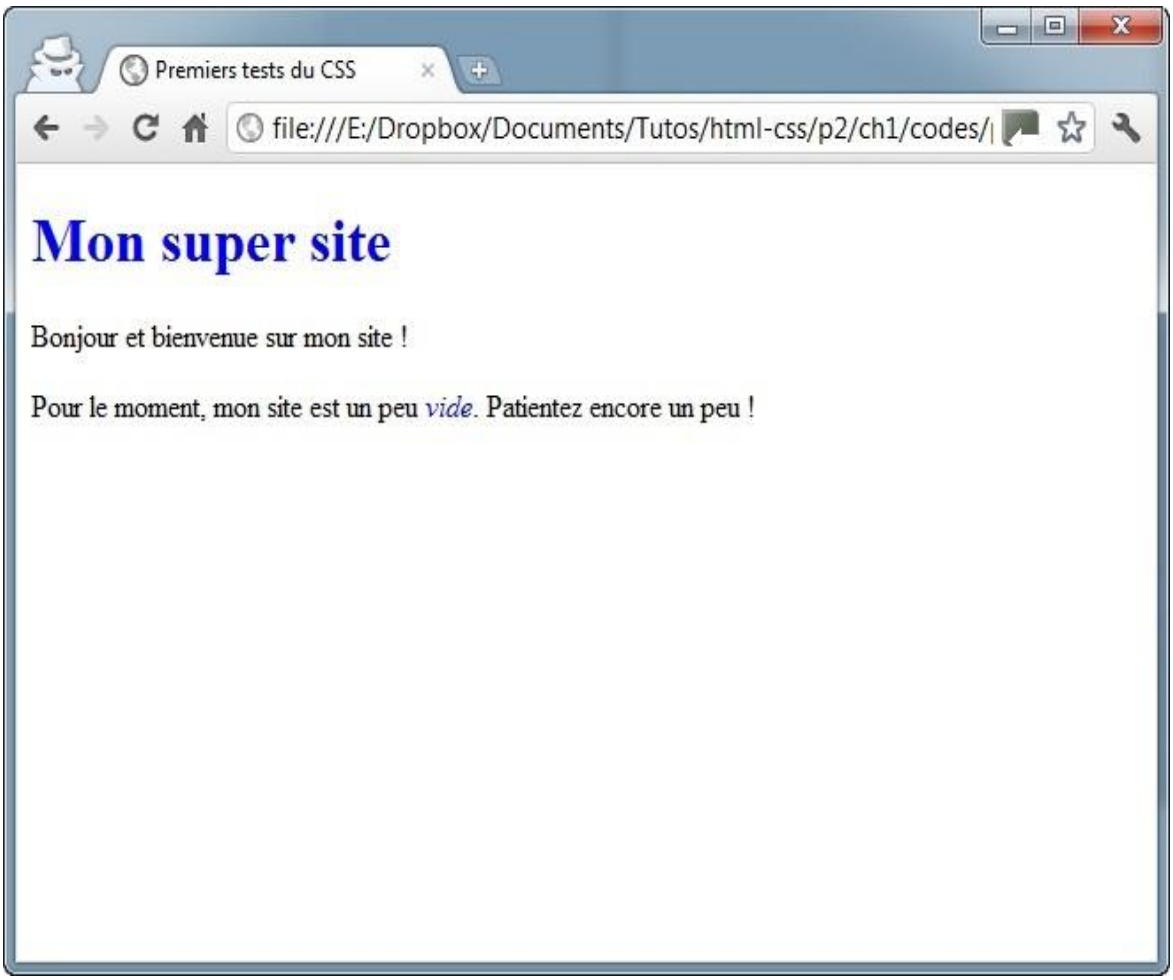
... thus means in French: " *I want all my paragraphs to be written in blue.* ".

III. Apply a style to multiple tags

There is a way in CSS to go faster if two tags should have the same layout. Just combine the declaration by separating the tag names with a comma like this:

```
h1, em
```

```
{
    color: blue;
}
```



This means : " *I want the text of my* ***<h1>*** *and* ****** *to be written in blue* ".

IV. Comments in CSS

As in HTML, it is possible to put comments. The comments will not be displayed, they are simply used to indicate information, for example to find you there in a long CSS file.

So, to make a comment, type /* , followed by the comment, then */ to end the comment. Comments can be written on one or more lines. For example :

```
style.css
---------

Par Florence LOE
*/

p
{
   color: blue; /* Les paragraphes seront bleus */
}
```

```
  /*
```

Code: CSS

V. Apply a style: class and id

The main flaw with the style seen previously is that it implies that ALL the tags change css style , hence the importance for us to identify them with the following attributes:

- The class attribute
- The id attribute

1) The class attribute

It is an attribute that can be put on any tag, title, paragraph, image, etc.

Code : HTML

```
<h1 class="nom dela classe"></ h1>
<p class="nom dela classe"></p>
<img class="nom delaclass" />
```

In fact, you must write a name which serves to identify the tag. Whatever you want, as long as the name starts with a letter.

For example, I'll give the class *introduction* to my first paragraph:

Code: HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Premiers tests du CSS</title>
    </head>

    <body>
        <h1>Mon super site</h1>

<p class="introduction">Bonjour et bienvenue sur mon site !</p>
        <p>Pour le moment, mon site est un peu <em>vide</em>.
Patientez encore un peu !</p>
    </body>
</html>
```

Now that it's done, your paragraph is identified. It has a name: *introduction* . You will be able to reuse this name in the CSS file to say: " *I want only the tags that have the introduction name to be displayed in blue* ".

To do this in CSS, specify the name of your class starting with a **dot** , like this:

Code: CSS

```
.introduction
{
   color: blue;
}
```

Only your paragraph called "introduction" will be displayed in blue!

2) The id attribute

It works exactly the same way as class , except for one thing: it can only be used *once* in the code.

In practice, we will only put ids on elements that are unique on your page, such as the logo:

If you use ids, in the CSS you will have to precede the name of the id with a **hash mark** (#):

```
<img src="images/logo.png" alt="Logo du site" id="logo" />
```

Code: CSS

```
#logo
{
    /* Indiquez les propriétés CSS ici */
}
```

VI. Universal tags

There will be times when you need to apply a class (or an id) to certain words that are not originally surrounded by tags. This would be easy to do if there was a tag around the word, unfortunately there are not any. Fortunately, we invented 2 so-called *universal tags* that have no particular meaning (they don't indicate that the word is important, for example). It is :

- ** :** This **is an inline** type tag . It is a tag that is placed within a paragraph of text, to select certain words only. The **< strong >** and **< em > tags** are from the same family. This tag is therefore used in the middle of a paragraph.
- **< div > : this is a** *block* -type tag that surrounds a block of text. The tags of the same family are **<p>** , **<h1>** , etc. These tags have something in common: they create a new "block" in the page, and therefore cause a line break. **< div >** is a frequently used tag in constructing a design.

For example, let's use the **< span > tag** . We put it around "welcome", we add a class to it (a name of your choice), and we apply the corresponding css

Code : HTML

```
<p>Bonjour et <span class="salutations">bienvenue</span>sur mon
site !</p>
```

css codes

```
.salutations
{
   color: blue;
}
```

VII. Applying a style: advanced selectors

In CSS, the hardest part is knowing how to target the text whose shape you want to change. To target (we say "select") the elements of the page to modify, we use what we call **selectors** .

1. The selectors you already know

These selectors, which we saw earlier, are by far the most commonly used. You have to know them by heart. Let's start with the basics:

Code: CSS

...

```
p
{

}
```

means "I want to affect all paragraphs". Afterwards, it's up to you to say what you do with these paragraphs (you write them in blue for example).

We have also seen

Coded :

```
h1, em
{

}
```

... which means "All important titles and texts". We selected two beacons at once.

And finally, we saw how to select specific tags to which we gave a name thanks to the class and id attributes :

Code: CSS

```
.class
{

}

#id
{

}
```

2. Advanced selectors

❖ *: universal selector

Code: CSS

```
*
{

}
```

Selects all tags without exception. This is called the **universal selector** .

❖ AB: a tag contained in another

Code: CSS

```
h3 em
{

}
```

all **** tags inside an **<h3> tag .** Note that there is no comma between the two tag names.

Example of corresponding HTML code:

Code: HTML

```
<h3>Titre avec <em>texte important</em></h3>
```

❖ A + B: a beacon that follows another

Code: CSS

```
h3 + p
{

}
```

Selects the first **<p> tag** located after an **<h3> title** .

Example :

Code: HTML

```
<h3>Titre</h3>

<p>Paragraphe</p>
```

❖ *A[attribute]: a tag that has an attribute*

Code: CSS

```
a[title]
{

}
```

<a> links that have a title attribute .

Example :

Code: HTML

```
<a href="http://site.com" title="Infobulle">
```

❖ *A[attribute="Value"]: a tag, an attribute and an exact value*

Code: CSS

```
a[title="Cliquez ici"]
{

}
```

Ditto, but the attribute must also have the exact value "Click here".

Example :

Code: HTML

```
<a href="http://site.com" title="Cliquez ici">
```

❖ *A[attribute*="Value"]: a tag, an attribute and a value*

Code: CSS

```
a[title*="ici"]
{

}
```

Ditto, the attribute must this time contain in its value the word "here" (regardless of its position).

Example :

Code: HTML

```
<a href="http://site.com"  title="Quelque part par ici" >
```

CHAPTER II: TEXT FORMATTING

General objective : to allow the student to modify the size, the font and the alignment of the text on a Web page.

Specific objectives :

- **Define the css ;**
- **Know where to insert the css in/on a page;**

Keywords : size, font, text,

Learning methods : receptive, active collaborative, active individual

Types of activity : collaborative work, presentation, practical exercises, etc.

Evaluation methods: before, during and after learning.

I. The size

To change the size of the text, we use the CSS **font-size property** . There are two possible ways to change the size of a text:

- Indicate an **absolute size** : in pixels, centimeters or millimeters. This method is very precise but it is advisable to use it only if it is absolutely necessary, because there is sometimes a risk of indicating a size that is too small for certain readers.
- Indicate a **relative size** : in percentage, " em " or "ex", this technique has the advantage of being more flexible. It adapts more easily to the size preferences of visitors.

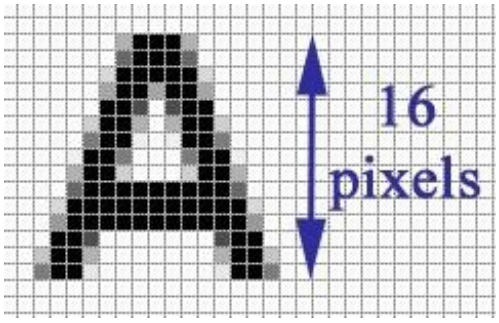
1. An absolute size

To indicate an absolute size, pixels are generally used. To have a text 16 pixels high, you must write:

Code: CSS

```
font-size : 16px;
```

The letters will be 16 pixels in size, as shown in the following image:



Here is an example of usage (put this code in your .css file) :

Code: CSS

```css
p
{
    font-size: 14px; /* Paragraphes de 14 pixels */
}
h1
{
    font-size: 40px; /* Titres de 40 pixels */
}
```

2. A relative value

This is the recommended method because the text then adapts more easily to the preferences of all visitors.

There are several ways to indicate a relative value. For example, you can write the size with English words like these:

- ✓ **xx - small** : tiny
- ✓ **x - small** : very small
- ✓ **small** : small
- ✓ **medium** : medium
- ✓ **large** : large **x-large** : extra large **xx-large** :

You can test the use of these values in your CSS code:

Code: CSS

```
p
{
    font-size: small;
}
h1
{
    font-size: large;
}
```

Well, this technique has a flaw: there are only 7 sizes available (because there are only 7 names). Fortunately there are other ways. The one that consists of indicating the size in " em ". It is a CSS-specific unit.

- If you write 1em, the text has a normal size.
- If you want to enlarge the text, you can enter a value greater than 1, such as 1.3em.
- If you want to reduce the text, enter a value less than 1, such as 0.8em.

Example :

Code: CSS

```
p
{
    font-size: 0.8em;
}
h1
{
    font-size: 1.3em;
}
```

Other units are available. You can try the " **ex** " (which works on the same principle as the **em** but which is smaller in base) and the percentage (80%, 130%...).

II. The police

Indeed, the problem is that, for a font to be displayed correctly, all Internet users must have it. If an Internet user does not have the same font as you, their browser will take a default font (a standard font) which may not have anything to do with what you expected.

The good news is that since CSS 3, it is possible to have the browser automatically download a font .

a. Change the font used

The CSS property that lets you specify the font to use is **font- family** . You must write the name of the font like this :

Code: CSS

```
balise
{
    font-family: police;
}
```

Only, to avoid there being any problem if the user does not have the same font as you, we generally specify *several* font names, separated by commas:

Code: CSS

```
balise
{
    font-family: police1, police2, police3, police4;
}
```

The browser will first try to put the **font1** . If it doesn't have it, it will try to put **font2** . If it doesn't, it will try font3 **and** so on.

In general, we indicate in very last **serif** , which corresponds to a standard font (which is only put if no other font has been found). Here is a list of fonts that work well on most browsers:

- ✓ Arial
- ✓ Arial Black
- ✓ comic without MS
- ✓ Courier New
- ✓ georgia
- ✓ Impact
- ✓ Times New Roman
- ✓ Trebuchet MS
- ✓ Verdana

So if I write:

Code: CSS

```
p
{
    font-family: Impact, "Arial Black", Arial, Verdana, sans-serif;
}
```

...that means: " *Put the font Impact, or, if it's not there, Arial Black, or else Arial, or else Verdana , or if nothing worked put a standard font (sans- serif)* " .

In general, it is good to indicate a choice of 3-4 fonts (+ serif or sans- serif) to ensure that at least one of them will be found on the visitor's computer.

b. Use a custom font with @font-face

Today, with CSS 3, there is fortunately a way to use any font on your site. It works well with most browsers.
But beware, there are flaws (it would be too good otherwise):

- Your visitors' browser will have to automatically **download** the font file, which can sometimes weigh 1 MB or more...
- Most of the fonts are copyrighted, so it's not **legal** to use them on his site. Fortunately, there are sites like fontsquirrel.com and dafont.com that offer a number of royalty-free fonts for download. I particularly recommend fontsquirrel.com because it allows you to download ready-to-use packages for CSS 3.
- Also note the Google Web Fonts service which is very well done.

There are **several** font file formats, and they don't work in all browsers.

Here are the different font file formats that exist and you should be aware of:

- ✓ . ttf : *TrueTypeFont* . Works on IE9 and all other browsers.
- ✓ . eot : *Embedded OpenType* . Works on Internet Explorer only, all versions. This format is proprietary to Microsoft.
- ✓ . otf : *OpenTypeFont* . _ Doesn't work on Internet Explorer
- ✓ . svg : *SVG Font* . The only format recognized on iPhone and iPad at the moment.

- ✓ . woff : *Web Open Font Format* . New format designed for the web that works on IE9 and all other browsers .

In CSS, to define a new font, you must declare it like this:

Code: CSS

```
@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot');
}
```

The font file (here *MySuperFont.eot*) must here be located in the same folder as the CSS file (or in a subfolder if you use a relative path).

It should be noted that the . eot only works on Internet Explorer. The ideal is to offer several font formats: the browser will download the one it can read. Here's how to specify multiple formats:

Code: CSS

```
@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot') format('eot'),
         url('MaSuperPolice.woff') format('woff'),
         url('MaSuperPolice.ttf') format('truetype'),
         url('MaSuperPolice.svg') format('svg');
}
```

To test how it works, I suggest you download a font from fontsquirrel , for example Learning Curve Pro . Click on "@font-face Kit", this will allow you to download a ready-to-use kit with all font formats.
Your CSS file will look like this in the end :

Code: CSS

```
@font-face { /* Définition d'une nouvelle police nommée
LearningCurveProRegular */
    font-family: 'LearningCurveProRegular' ;
    src: url('LearningCurve_OT-webfont.eot' );
    src: url('LearningCurve_OT-webfont.eot?#iefix' )
format('embedded-opentype' ),
         url('LearningCurve_OT-webfont.woff' ) format('woff'),
         url('LearningCurve_OT-webfont.ttf' ) format('truetype'),
         url('LearningCurve_OT-webfont.svg#LearningCurveProRegular' )
format('svg');
}

h1 /* Utilisation de la police qu'on vient de définir sur les
titres */
{
    font-family: 'LearningCurveProRegular' , Arial, serif;
}
```

@font-face section allows you to define a new font name that can be used in the CSS file. Then we use this font name with the **font- family property** we know, to change the appearance of the **<h1> headings** .

III. The style: Italic, bold, underlined...

There are a series of classic text formatting properties in CSS. We are going to discover here the setting in bold, italics, underlining... and in passing we will see that it is even possible to go so far as to make the text blink!

1. Italicize

Concretely, to italicize in CSS we use **font-style** , which can take 3 values:

- italic : the text will be italicized.
- oblique : the text will also be italicized (slanting the letters).
- normal : the text will be normal (by default). This allows you to undo italicization.

For example, if you want the texts between **< em >** not to be italicized anymore, you would have to write:

Code: CSS

```css
em
{
    font-style : normal;
}
```

On the following example, I use for example **font-style** to italicize all my titles **<h2>** :

Code: CSS

```css
h2
{
    font-style : italic;
}
```

2. Bold

Bolding in CSS allows you to bold, for example, titles, certain entire paragraphs, etc. It's for you to see. The CSS property for bolding is **font- weight** , and takes the following values:

- **bold** : the text will be in bold.
- **normal** : the text will be written normally (by default).

Here is an example of how to write titles in bold:

Code: CSS

```css
h1
{
    font-weight : bold;
}
```

3. Underlining and other decorations

This CSS property is aptly named: **text-decoration** . It allows among other things to underline the text, but not only. Here are the different values it can take:

- ✓ Underline : underlined .
- ✓ line through

crossed out .

- ✓ overline : line above.
- ✓ blink : blinking. Does not work on all browsers (Internet Explorer and Google Chrome in particular). none : normal (default).

This CSS will allow you to test the effects of text-decoration :

Code: CSS

```
h1
{
    text-decoration : blink;
}
.souligne
{
    text-decoration : underline;
}
.barre
{
    text-decoration : line-through;
}
.ligne_dessus
{
    text-decoration : overline;
}
```

4. alignment

The CSS language allows us to do all the alignments that we know: left, centered, right and justified. It's quite simple. We use the **text-align property** , and we specify the desired alignment:

- ✓ left : the text will be aligned to the left (this is the default setting).
- ✓ center : the text will be centered.
- ✓ right : the text will be aligned to the right.
- ✓ justify : the text will be "justified". Justifying the text makes it so that it takes all the possible width without leaving white space at the end of the lines. Newspaper texts, for example, are always justified .

Look at the different alignments on this example:

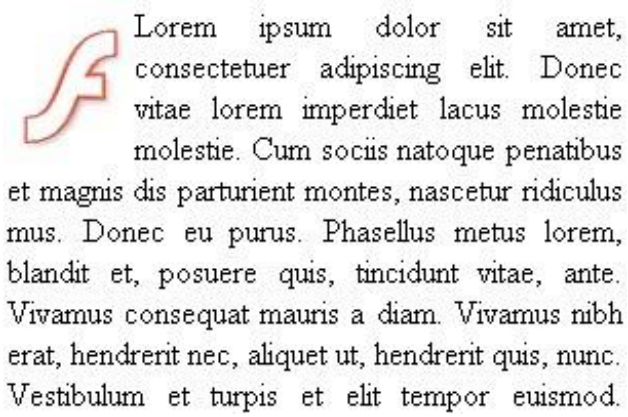
Code: CSS

```css
h1
{
    text-align : center;
}

p
{
    text-align : justify;
}

.signature
{
    text-align : right;
}
```

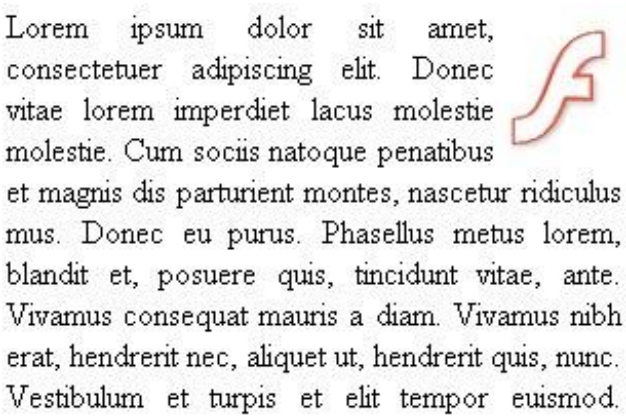
NB : You cannot modify the alignment of the text of an *inline tag* (like < **span** > , <**a**> , < **em** > , < **strong** > ...). Alignment only works on *block* type tags (<**p**> , <**div**> , <**h1**> , <**h2**> , ...), and it's kind of logical when you think about it: you can't change the alignment of a few words in the middle of a paragraph! It is therefore generally the entire paragraph that you will need to align.

5. Floats

CSS allows us to float an element around the text. We also say that we are doing a " **dressing** ".
So that you can see what we are talking about, here is what we are going to learn how to do:



An image floating left



An image floating to the right

The property we are going to use here is... **float** . This property can take 2 very simple values:

- left : the element will float to the left.
- right : the element will float... to the right! Yes well done.

Using floats is very simple:

1. You apply a **float** to a tag.
2. Then you continue to write text in sequence as normal.

NB : You can use the **float property** on block tags as well as on inline tags . It is common to float an image so that it is wrapped in text, as in the previous examples.

a. Float an image

Here we will learn how to float an image. Here is the HTML code we need to type first:

Code: HTML

```
<p><img src="flash.gif" class="imageflottante" alt="Image flottante"
/> Ceci est un texte normal de paragraphe, écrit à la suite de
l'image et qui l'habillera car l'image est flottante. </p>
```

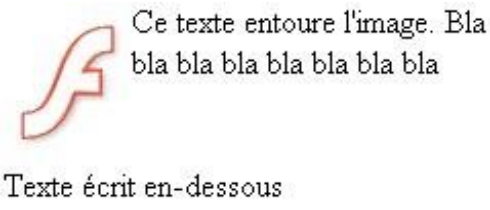
Here's the only piece of CSS code we need to type, which floats the image to the left :

Code: CSS

```
.imageflottante
{
    float: left;
}
```

b. Stop a float

When you set up a float, the text around it wraps. But what if you want the text to continue *below* the float after a while ? We could do several **< br />** in a row, but it wouldn't be elegant or very clean... Basically, we would like to be able to do this:



There is in fact a CSS property that allows you to say: " *Stop, this text must be below the float and no longer next to it* ". It is the **clear** property which can take these three values:

- ✓ **left** : the text continues below after a **float** : **left** ;
- ✓ **right** : the text continues below after a **float** : **right** ;
- ✓ **both** : the text continues below, whether after a **float** : **left** ; or after a **float** : **right** ;

To simplify, we will always use **clear** : **both** , which works after a float on the left and after a float on the right (so it always works). To illustrate how it works, we will take this HTML code :

Code: HTML

```
<p><img src="flash.gif" class="imageflottante" alt="Image flottante"
/></p>
<p>Ce texte est écrit à côté de l'image flottante. </p>
<p class="dessous">Ce texte est écrit sous l'image flottante. </p>
```

And this CSS code:

Code: CSS

```
.imageflottante
{
   float: left;
}
.dessous
{
   clear: both;
}
```

CHAPTER III: COLOR AND BACKGROUND

We will see here:

- How to change text color
- How to put a background color or image
- How to add shadows

- How to play with transparency levels

I. Text color

The property that allows you to modify the color of the text is : **color** . We are going to look at the different ways to indicate the color, because there are several.

1. Indicate the name of the color

The easiest and most convenient way to choose a color is to type its name.

The only drawback of this method is that there are only 16 so-called "standard" colors. Other unofficial colors exist, but since they will not necessarily work the same on all browsers, I will avoid showing them to you.

Here are the 16 colors you can use by just typing their name:

Couleur	Aperçu
white	
silver	
gray	
black	
red	
brown	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

Example : To change all the titles to brown, we

can write **Code: CSS**

```
h1
{
    color: maroon;
}
```

2. Hexadecimal notation

There are several ways in CSS to choose a color from all that exist. The first one I'm going to show you is the **hexadecimal notation** . It is commonly used on the web, but there is also another method that we will see later.

A color name in hexadecimal looks like this: #FF5A28. Simply put, it's a combination of letters and numbers that indicate a color.

You must always start by writing a hash mark (#), followed by 6 letters or numbers ranging from 0 to 9 and from A to F.

These letters or numbers work in pairs. The first 2 indicate an amount of red, the next 2 an amount of green, and the last 2 an amount of blue. By mixing these quantities (which are the Red - Green - Blue components of the color) we can obtain the color we want.

Thus, #000000 corresponds to the black color and #FFFFFF to the white color.

Here is for example how to apply the white color in hexadecimal to the paragraphs:

Code: CSS

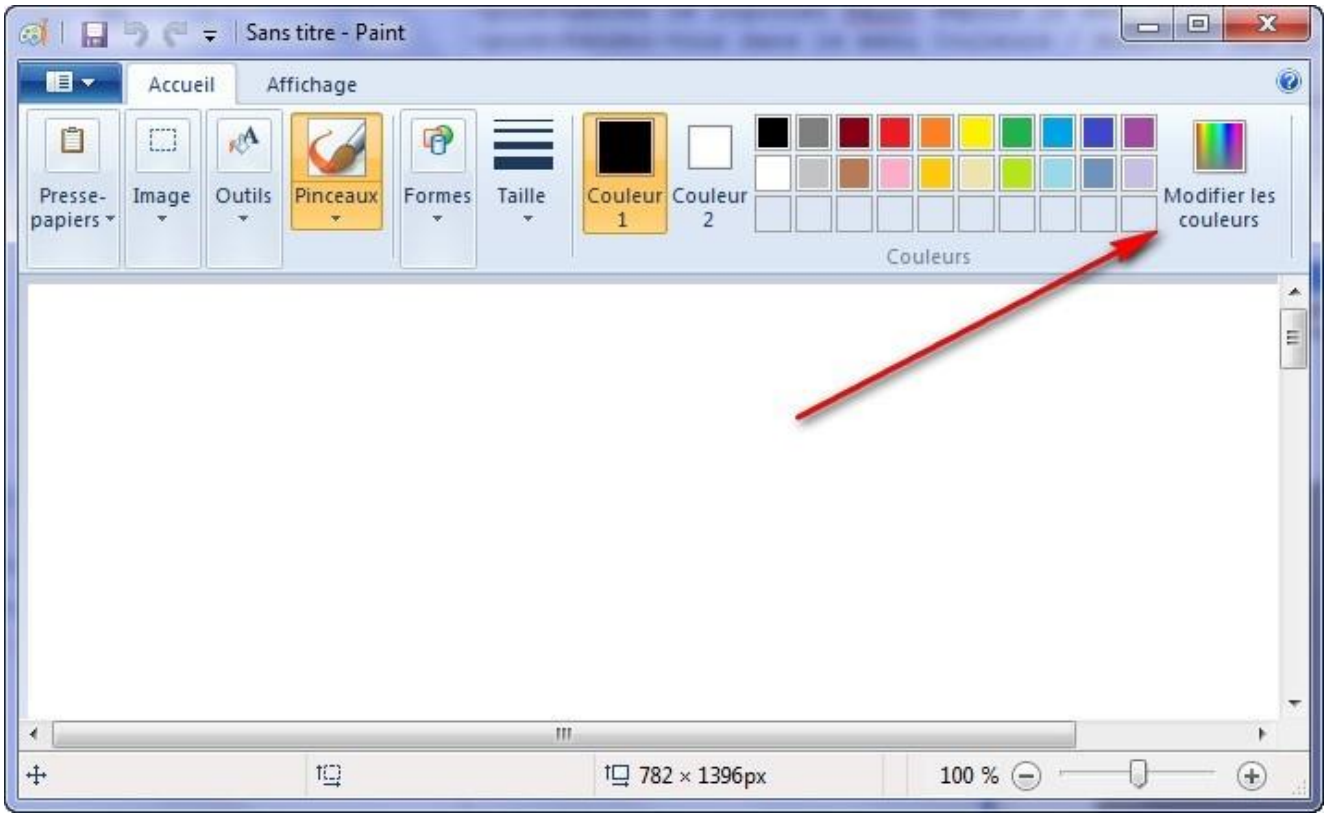
```
p
{
    color: #FFFFFF;
}
```

3. The RGB method

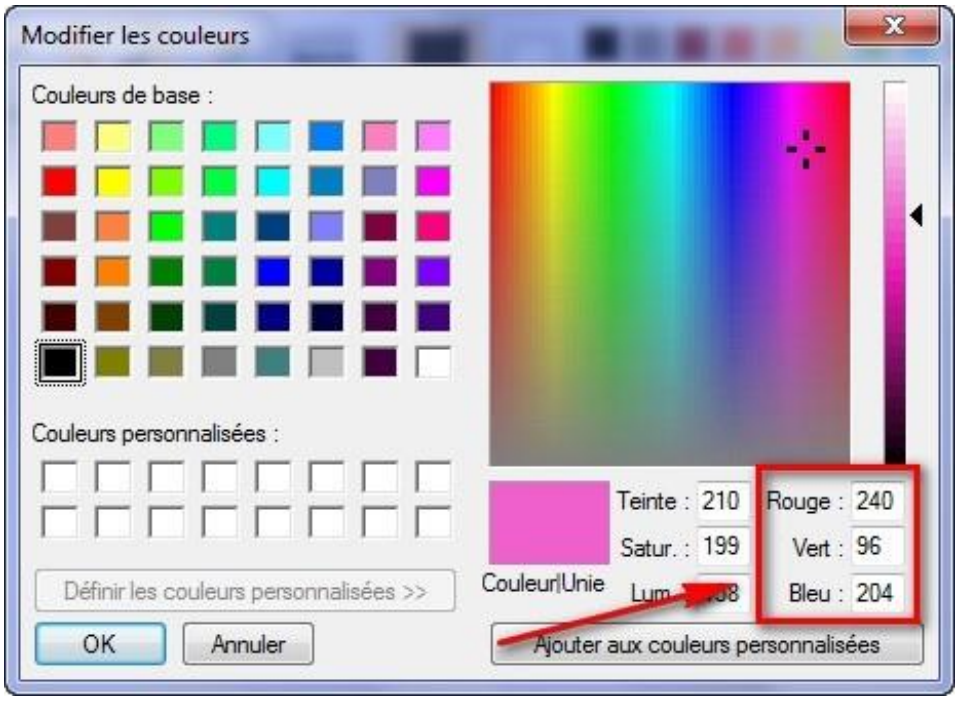
What does RGB mean? In English, Rouge-Vert-Bleu is written Red -Green-Blue, which is shortened to "RGB". As for the hexadecimal notation, one must define an amount of red, green and blue to choose a color.

With simple drawing software like Paint, you can find the color you want. Here are the steps to follow :

1. Launch the Paint software from the Start menu.
2. Go to the *Change colors section* :



3. A window opens. In the area that appears on the right, move the sliders to select the color that interests you. Suppose I get the urge to write my **<h1> headings** in barbie pink . I select the color in the window, like this:



4. Note the corresponding quantities of Red-Green-Blue indicated at the bottom right of the window (here 240-96-204). I copy these values in this order in the CSS file, like this :

Code: CSS

```
p
{
    color: rgb(240,96,204);

}
```

As you have seen in the example, to use the RGB method you must type rgb (Red, Green, Blue) replacing "Red, Green, Blue" by the corresponding numbers. For information, these quantities are always between 0 and 255.

II. Background color

To specify a background color, we use the CSS **background- color property** . It is used in the same way as the **color property** , that is to say that you can type the name of a color, write it in hexadecimal notation or use the RGB method.

To indicate the background color of the web page, you have to work on the **<body> tag** . Yes, **<body>** corresponds to the entire web page, so changing its background color will change the background color of the web page.

Example :

Code: CSS

```
body /* We are working on the body tag, so on the WHOLE page */
{ background -color : black ; /* The background of the page
will be black */ color : white ; /* The page text will be
white */
}
```

❖ CSS and inheritance

In CSS, if you apply a style to a tag, all the tags inside will take on the same style.

It's actually simple to understand and intuitive. The **<body> tag** , as you know, contains the paragraph **<p>** and title **<h1> tags, among others** .
If I apply a black background color and a white text color to the **<body> tag** , all my titles and paragraphs will also have a black background color and a white text color... This is the phenomenon that is called inheritance: tags that are inside another tag are said to " **inherit** " its properties. If you later say you want your titles in red, that style will take precedence and your titles will therefore be in red. On the other hand, if you don't indicate anything in particular (as we did earlier), then your titles will inherit the color white.

It doesn't just work for color. All CSS properties will be inherited: you can for example require bolding in the **<body> tag** , and all your headings and paragraphs will be bold.

- **Example of inheritance with the <mark> tag**

We tend to believe that we can only change the background color of the page. This is wrong: you can change the background of any element: your titles, your paragraphs, certain words... In this case, they will appear highlighted (as if we had put a marker on them).

Do you remember, for example, the **<mark> tag** that highlights certain words? Let's use it again here:

Code: HTML

```
<h1>Qui a éteint la lumière ? </h1>

<p>Brr, il fait tout noir sur ce site, c'est un peu
<mark>inquiétant</mark>comme ambiance non vous trouvez pas ?  </p>
```

By default, the text is displayed on a yellow background. You can change this behavior in CSS:

Code: CSS

```css
body
{
    background-color : black;
    color: white;
}

mark
{
     /* La couleur de fond est prioritaire à celle de toute la page
*/
     background-color : red;
}
```

On the text of the **<mark> tag** , the red background color is applied. Indeed, even if the background of the page is black, it is the CSS property of the most precise element that takes precedence .

NB : The same principle applies to all HTML tags and all CSS properties! If you say so :

- My paragraphs are 1.2em in size
- My important texts (< **strong** >) have a size of 1.4em

... one would think that there is a conflict. Important text is part of a paragraph, how big should it be? 1.2em or 1.4em? The CSS decides that the most precise declaration wins: as < **strong** > corresponds to a more precise element than paragraphs, the text will be written in 1.4em.

III. Background pictures

In the following examples, I will modify the background image of the page. However, just like the background color, remember that the background image does not necessarily apply to the entire page. You can also put a background image in titles, paragraphs, etc.

1. Apply a background image

The property for specifying a background image is **background-image** . As value, we must give it url("nom_de_l_image.png") .

For example :

Code: CSS

```
body
{
    background-image : url("neige.png") ;
}
```

Of course, your background is not necessarily in PNG, it can also be in JPEG or GIF.
The address indicating where the background image is located can be written in absolute (http://...) or in relative (fond.png).

NB : when you write a relative address in the CSS file! The address of the image must be indicated *relative* to the . css and not against the .html file . I advise you to place the background image in the same folder as the **. css** to simplify things (or in a subfolder).

❖ Options available for the background image

background-image property that we have just seen with several other properties that allow you to change the behavior of the background image.

✓ ***background - attachment : attach the background***

The CSS **background- attachment property** allows you to "fix" the background. The effect obtained is interesting, because we then see the text "slide" over the background. Two values are available:

- **fixed** : the background image remains fixed.
- **Scroll** : background image scrolls with text (default)

.

Code: CSS

```
body
{
    background-image : url("neige.png") ;
    background-attachment : fixed; /* Le fond restera fixe */
}
```

✓ ***background - repeat : background repeat***

By default, the background image is tiled. You can change this with the **background- repeat property** :

- **no - repeat** : the background will not be repeated. The image will therefore be unique on the page.
- **repeat -x** : the background will be repeated only on the first line, horizontally.
- **repeat -y** : the background will be repeated only on the first column, vertically.
- **repeat** : the background will be repeated in mosaic (by default).

Example of use :

Code: CSS

```
body
{
    background-image : url("soleil.png") ;
    background-repeat : no-repeat;
}
```

- ✓ ***background -position: background position***

You can indicate where the background image should be with **background-position** . This property is only interesting if combined with **background-repeat** : **no-repeat** ; (a non-repeating background).
You need to give **background-position** two pixel values to indicate the position of the background relative to the top left corner of the page (or paragraph if you are applying the background to a paragraph). So if you type:

Code: CSS

```
background-position : 30px 50px;
```

… your background will be placed 30px from the left and 50px from the top. It is also possible to use these values in English:

- **top** : top.
- **bottom** : bottom.
- **left** : left.
- **center** : centered.
- **right** : right.

It is possible to combine these words. For example, to align an image to the top right, you would type:

Code: CSS

```
background-position : top right;
```

Thus, if I want to display a sun as a background image, in a single copy (no- repeat), always visible (fixed) and positioned at the top right (top right), I will write this:

2. Combine Properties

If you use a lot of properties related to the background (as is the case on this last example), you can use a kind of "super-property" called **background** which can take several combined values of the properties seen previously: **background- image** , **background- repeat** , **background- attachment** and

```
body
{
    background-image url("soleil.png");
    background-attachment: fixed; /* Le fond restera fixe */
    background-repeat: no-repeat;  /* Le fond ne sera pas répété */
    background-position: top right; /* Le fond sera placé en haut à
droite */
}
```

backgroundposition .

So we can simply write:

Code: CSS

```
body
{
    background: url("soleil.png") fixed no-repeat top right;
}
```

This is the first "super-property" that I show you, there will be others. You must know that:

- The order of the values does not matter. You can combine the values in any order.
- You don't have to put all the values. So, if you don't want to write **fixed ,** you can remove it without problem.

3. Multiple background images

Since CSS3, it is possible to give several background images to an element. To do this, simply separate the declarations with a comma, like this:

Code: CSS

```
body
{
    background: url("soleil.png") fixed no-repeat top right,
url("neige.png") fixed;
}
```

The first image in this list will be placed on top of the others. Please note that the order in which the images are declared is important. It should be noted that multiple background images work on all browsers, except on older versions of Internet Explorer, which only recognizes it from version 9 (IE9).

NB : it is important to remember that you can apply a background to any element (a title, a paragraph, certain words in a paragraph, etc.).

IV. Transparency

The CSS allows us to play very easily with the levels of transparency of the elements! For this, we will use features of CSS3: the **opacity property** and the RGBa notation .

1. The property " **opacity** »

The very simple **opacity property** lets you specify the level of opacity (it's the opposite of transparency).

- With a value of **1** , the element will be completely opaque: this is the default behavior.
- With a value of **0** , the element will be completely transparent.

You must therefore choose a value between **0** and **1** . For example with 0.6, your element will be 60% opaque... and we will therefore see through!

Here is how it can be used:

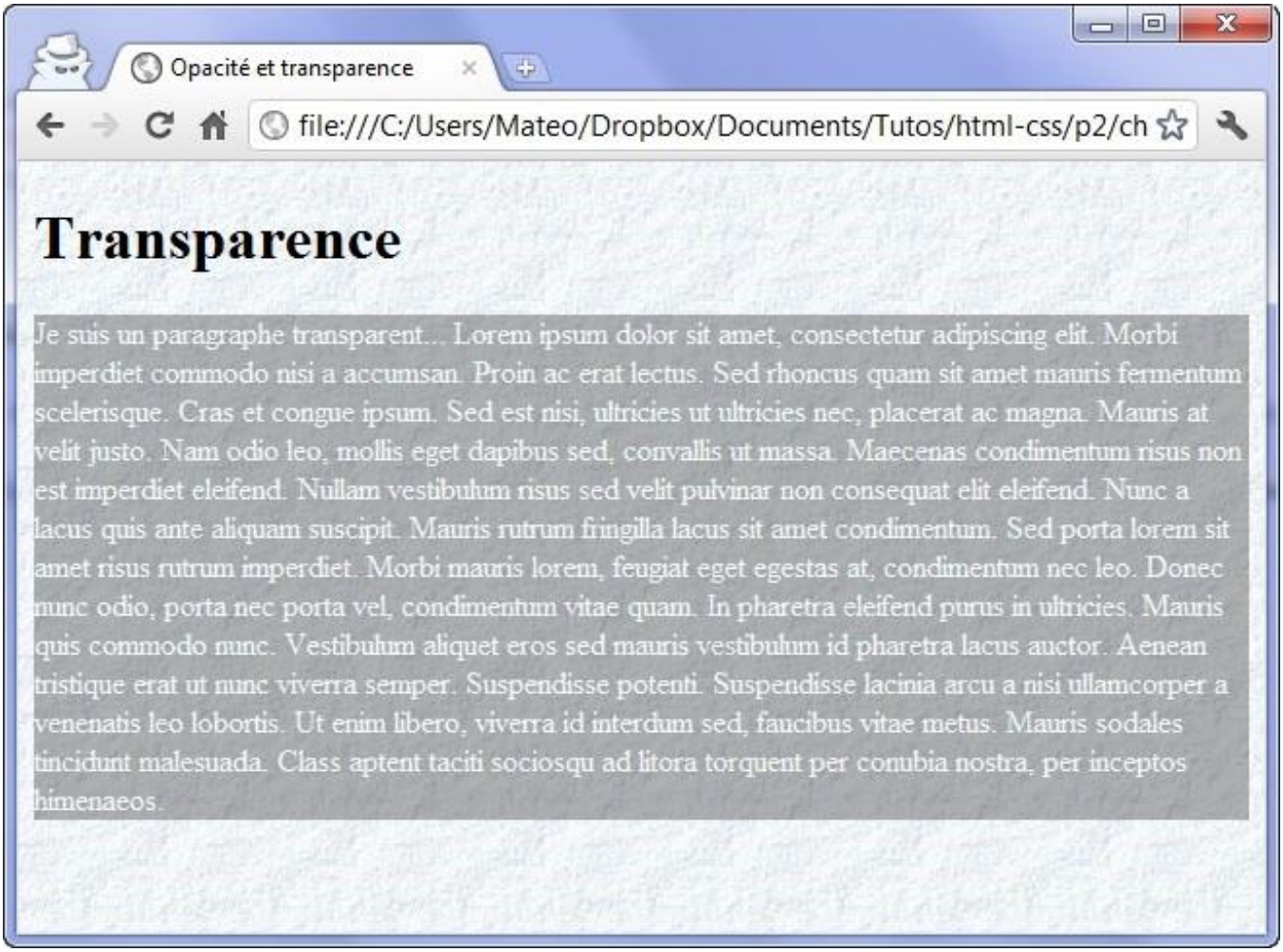
Code: CSS

```
p
{
    opacity: 0.6;
}
```

Here is an example that will allow us to appreciate the transparency:

Code: CSS

```css
body
{
   background : url('neige.png') ;
}

p
{
    background-color : black;
    color: white;
    opacity: 0.3;
}
```



NB : Note that transparency works on all recent browsers, including Internet Explorer from IE9.
If you apply the **opacity property** to an element on the page, all of that element's content will be made transparent (even images, other blocks inside, etc.). If you just want to make the background color transparent, use the RGBa notation instead .

2. The notation **RGBa**

CSS3 offers us another way to play with transparency: the RGBa notation . This is actually the RGB notation we saw earlier, but with a fourth parameter: the level of transparency (called "alpha channel"). Similarly, with a value of 1, the background is completely opaque. With a value less than 1, it is transparent.

Code: CSS

```
p
{
    background-color : rgba(255, 0, 0, 0.5); /* Fond rouge à moitié
transparent */
}
```

It's as simple as that. You can achieve the exact same effect as with **opacity** just by playing around with the RGBa notation .

This notation is known to all recent browsers, including Internet Explorer (from IE9). For older browsers, it is recommended to specify the classic RGB notation in addition to RGBa .

Code : CSS

```
p
{
    background-color : rgb(255,0,0); /* Pour les anciens navigateurs
*/
    background-color : rgba(255,0,0,0.5); /* Pour les navigateurs
plus récents */
}
```

The background will then not be transparent for these browsers, but at least there will be a background color.

CHAPTER IV: BORDERS AND SHADOWS

Here, we are going to look at borders and shadow effects that can be applied, both to the text and to the blocks that make up our page.
In particular, we will reuse our knowledge of colors to choose the color of our borders and shadows.

I. standard borders

Many CSS properties allow you to modify the appearance of your borders: **border- width** , **border-color** , **border-style** ...
To get to the point, I suggest that you use the **border super-property directly** , which groups all of these properties. Remember the **background super-property** ? It works on the same principle: we will be able to combine several values.

For **border** you can use up to 3 values to modify the appearance of the border:

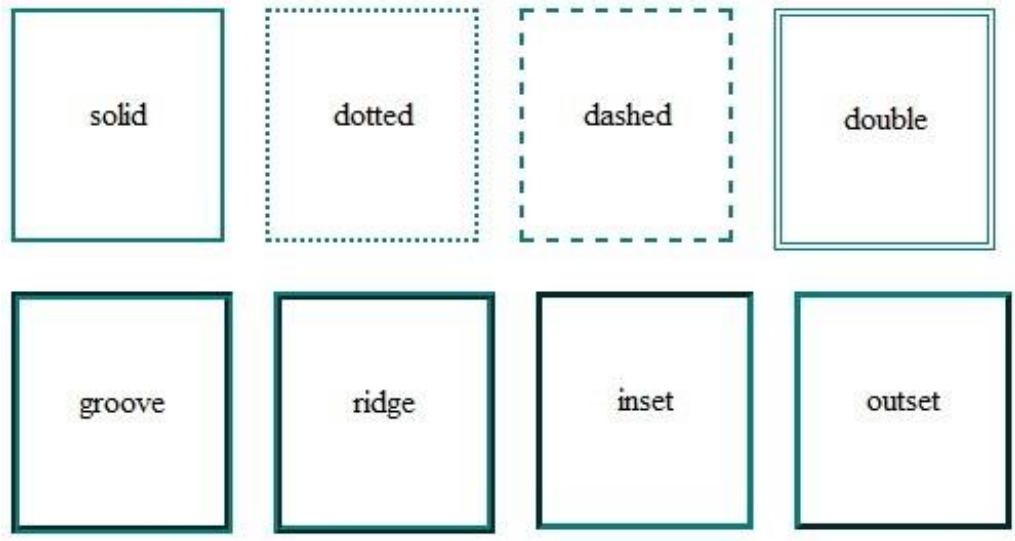
- ✓ **The width** : indicate the width of your border. Put a value in pixels (like 2px).
- ✓ **The color** : this is the color of your border. Use, as learned, either a color name ("black", " red "...), a hex value (#FF0000), or an rgb value (rgb (198, 212, 37)) .
- ✓ **The type of border** : there, you have the choice. Your border can be a simple line, or dotted lines, or dashes etc... Here are the different values available:
  - **none** : no border (default)
  - **solid** : a single line.
  - **dotted** : dotted.
  - **dashed** : dashes.
  - **double** : double border.
  - **groove** : in relief.
  - **ridge** : another relief effect.
  - **inset** : Inset 3D effect.
  - **outset** : raised 3D effect.

So, to have a blue dashed border of 3 pixels around my titles, I will write:

Code: CSS

```
h1
{
    border: 3px blue dashed;
}
```

Here are the different border styles in pictures to help you make your choice:



1. Up, right, left, down...

If you want to put different borders depending on the side (top, bottom, left or right), you can do it without problem. In this case, you will need to use these 4 properties:

- **border -top** : border at the top.
- **border - bottom** : bottom border.
- **border - left** : left border.
- **border -right** : border to the right.

NB : There are also equivalents to configure each detail of the border if you wish : **border-top- width** to modify the thickness of the top border , **border -top- color** for the top color, etc. These are also super-properties, they work like **border** but therefore only apply to one side.

To add a border only to the left and right of the paragraphs, we would therefore write:

Code: CSS

```css
p
{
    border-left: 2px solid black;
    border-right: 2px solid black;
}
```

NB :On peut modifier les bordures de n'importe quel type d'élément sur sa page. Nous l'avons fait ici sur les paragra
mais on peut aussi modifier la bordure de ses images, des textes importants comme etc.

II. Rounded edges

border-radius property will allow us to round the corners of any element easily. All you have to do is indicate the size ("the importance") of the rounding in pixels:

Code: CSS

```css
p
{
    border-radius: 10px;
}
```

Rounding can be seen in particular if:

- The element has borders;
- Or if it has a background color.

You can also specify the shape of the rounding for each corner. In this case, specify 4 values:

Code: CSS

```css
p
{
    border-radius: 10px 5px 10px 5px;
}
```

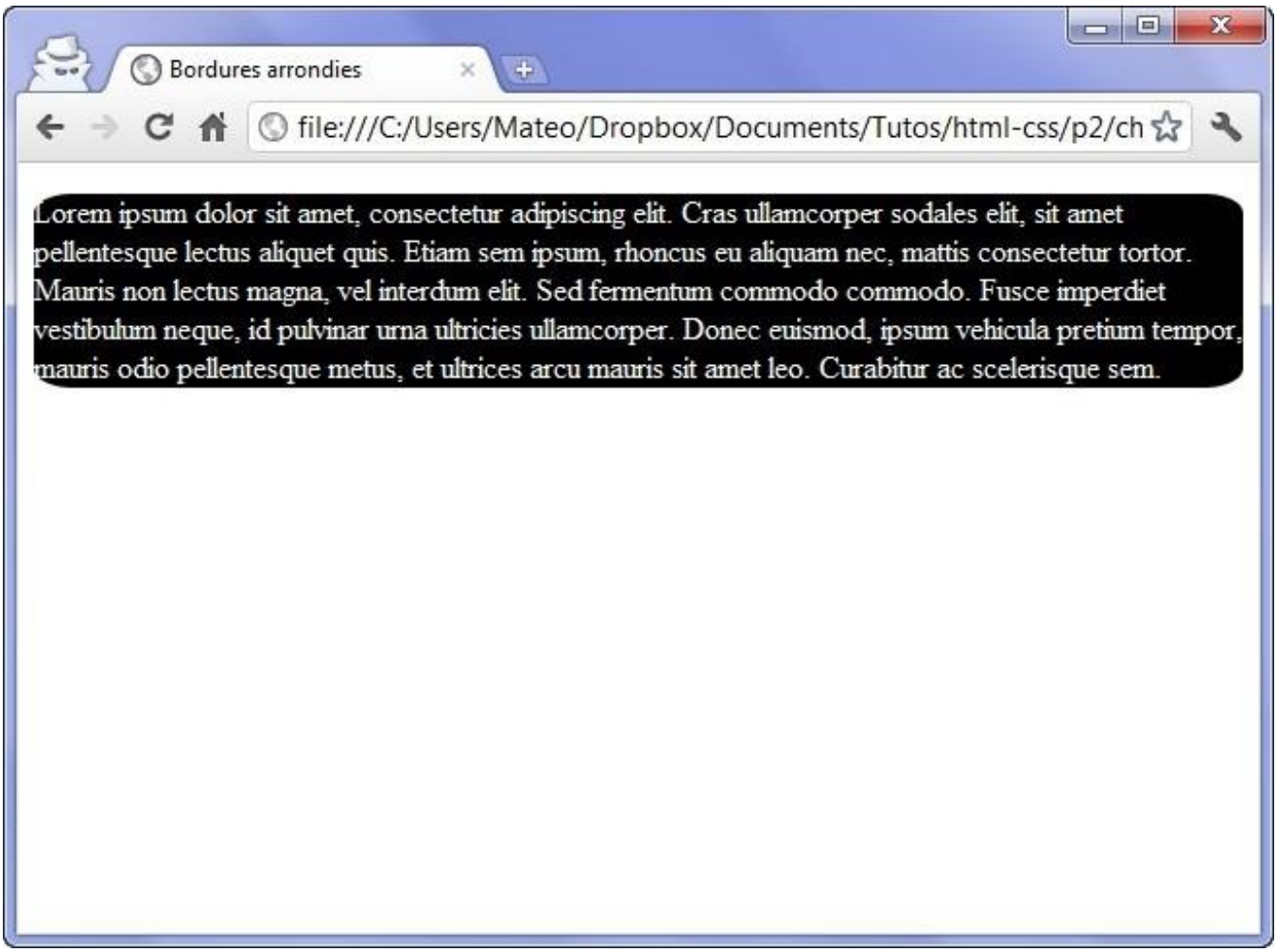
The values correspond to the following angles in this order:

1. On the top corner left
2. Top right
3. At the bottom right
4. At the bottom left

Finally, it is possible to refine the rounding of our corners by creating elliptical curves. You must specify 2 values separated by a slash (/).

Code: CSS

```
p
{
    border-radius: 20px / 10px;

}
```



III. The shadows

Shadows are one of the recent new features offered by CSS3. Today, adding shadows to your pages is done in a single line of CSS!

Here we will discover two types of shadows:

- The shadows of the boxes
- Text shadows

1. box - shadow : box shadows

box- shadow property applies to the whole block and takes 4 values in this order:

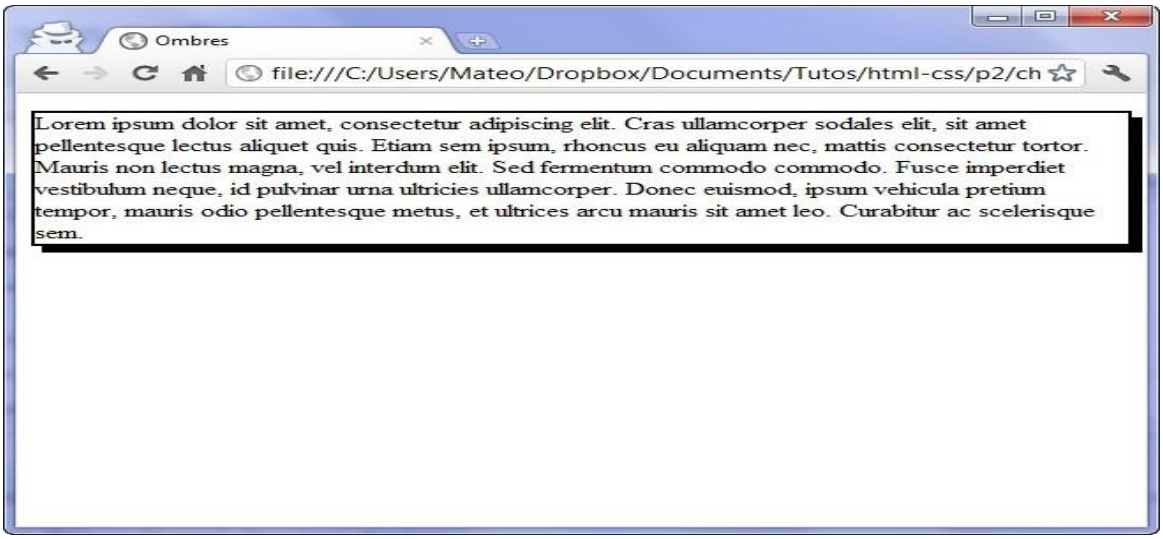
1. The horizontal offset of the shadow
2. The vertical offset of the shadow
3. The softening of the gradient
4. The color of the shadow

For example, for a black shadow of 6 pixels, without softening, we will write:

Code: CSS

```
p
{
    box-shadow: 6px 6px 0px black;
}
```

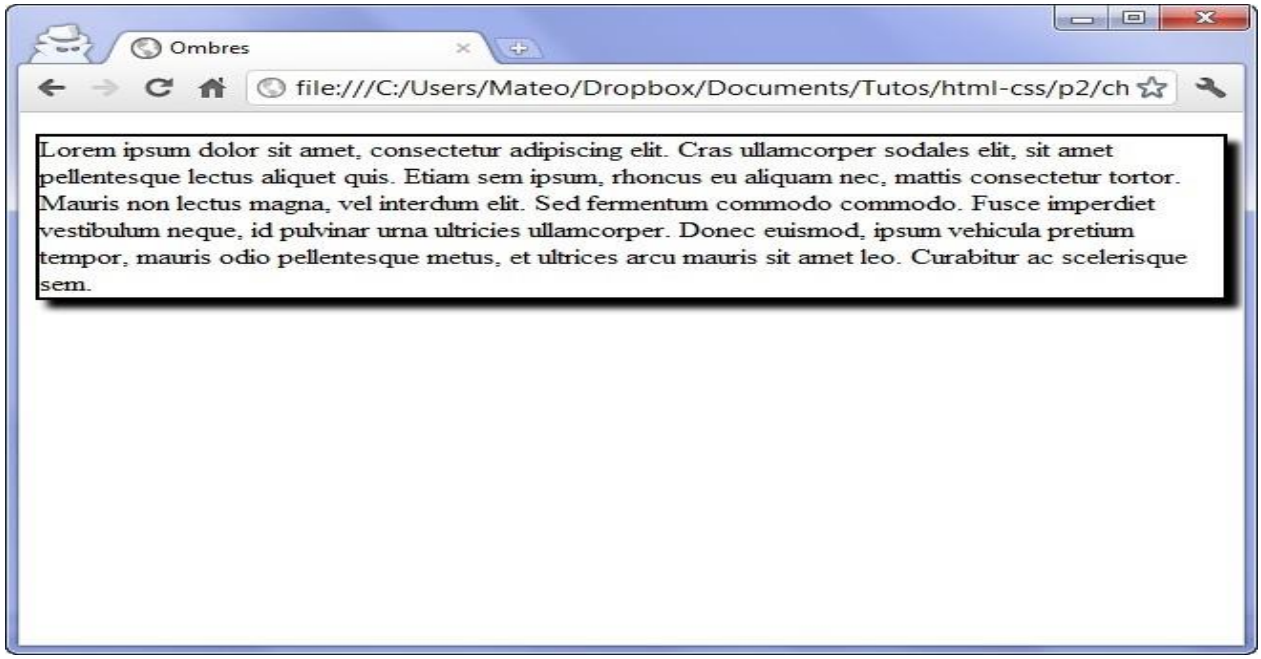
Which gives (I added a border to the paragraph to make it easier to see the effect):



Let's add softening with the third parameter. Softening can be **low** (less than offset), **normal** (equal to offset), or **high** (greater than offset). Let's try a normal offset:

Code: CSS

```
p
{
    box-shadow: 6px 6px 6px black;
}
```



You can also add a fifth optional value: **inset** . In this case, the shadow will be placed inside the block, to give a sunken effect

2. " text -shadow ": the shadow of the text

With **text-shadow** , you can add a shadow directly to the letters of your text! The values work exactly the same as **box- shadow** : offset, feathering, and color .

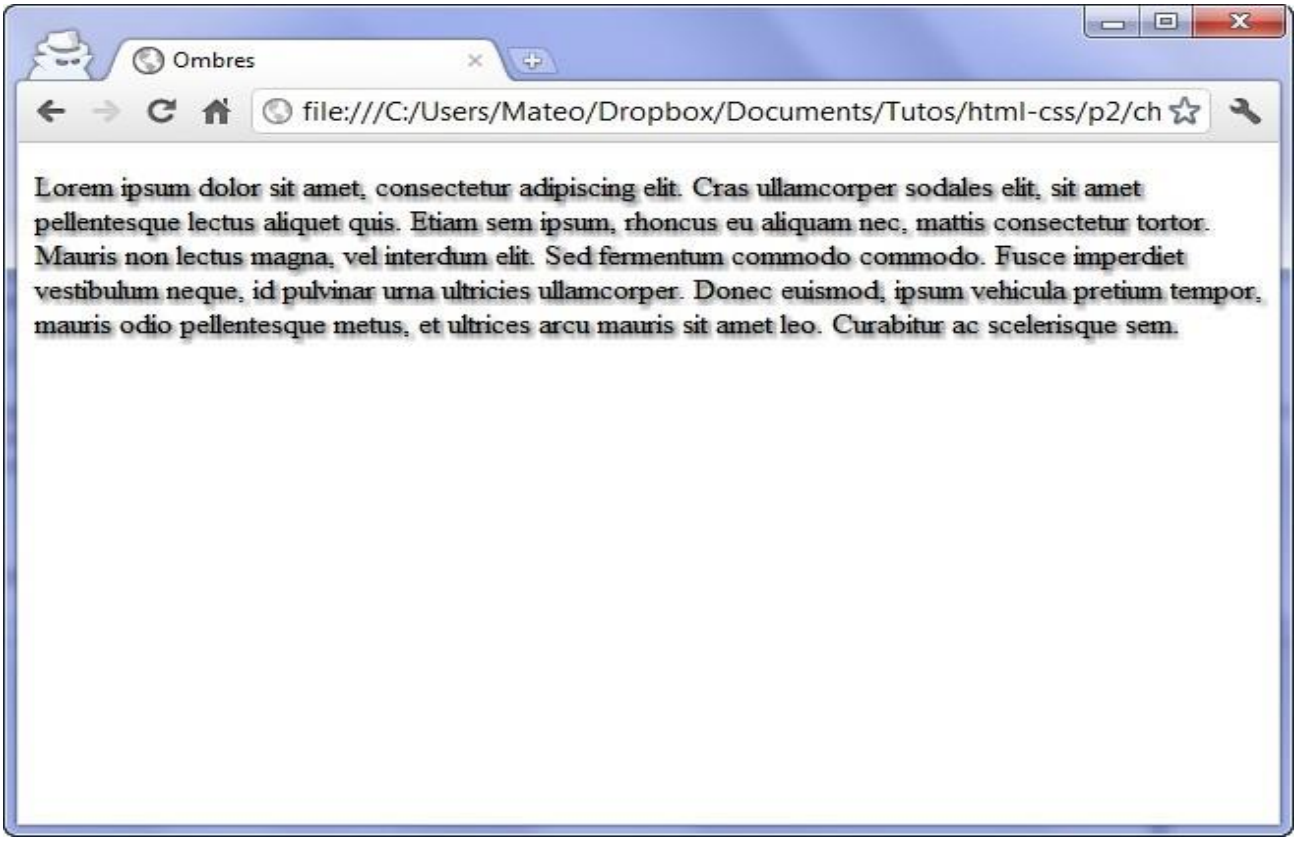
Code: CSS

```
p
{
    text-shadow: 2px 2px 4px black;
}
```

Code : CSS

```
p
{
    box-shadow: 6px 6px 6px black inset;
}
```

Here is the result



CHAPTER V: CREATING DYNAMIC APPEARANCES

CSS also allows us to modify the appearance of elements dynamically, that is, elements can change shape once the page has been loaded. We are going to use a powerful feature of CSS: **pseudo-formats** .

We will see in this chapter how to change the appearance:

- ✓ On hover
- ✓ When clicking
- ✓ When in focus (selected item) When a link has been visited

I. On hover

In this chapter, we will discover several CSS pseudo-formats. The first one I'm going to show you is called : hover . Like all the other pseudo-formats that we will see, it is information that is added after the name of the tag (or class) in the CSS, like this:

Code: CSS

```
a:hover
{

}
```

: hover means "on top". **a** :hover therefore means: "When the mouse is over the link" (when you point to it).

From there, it's up to you to define how the links should look when hovered over. Let your imagination run wild, there is no limit.

Here is an example of how links are presented:

Code: CSS

```
a /* Liens par défaut (non survolés) */
{
   text-decoration : none;
   color: red;
   font-style : italic;
}

a:hover /* Apparence au survol des liens */
{
   text-decoration : underline;
   color: green;
}
```

We have defined here 2 versions of the styles for the links:

- For default (non-hover) links
- For links on hover

Here is what it gives:

Although it is often used on links, you can change the appearance of any element. For example, you can change the appearance of paragraphs when hovering over them:

Code: CSS

```
p:hover /* Quand on pointe sur un paragraphe */
{

}
```

II. On click and on selection

You can interact even more finely in CSS. We will see here that we can change the appearance of elements when clicked on and when they are selected!

1. :active : when clicked

The pseudo-format :active allows to apply a particular style *at the moment of the click* . In practice, it is only used on links.

The link will keep this appearance for a very short time: in fact, the change appears when the mouse button is pressed. Clearly, it is not necessarily always clearly visible.

For example, you can change the background color of the link when you click on it:

Code: CSS

```css
a:active /* Quand le visiteur clique sur le lien */
{
    background-color : #FFCC66;
}
```

2. :focus: when the element is selected

The :focus pseudo- format applies a style when the element is selected . In other words, Once clicked, the link remains "selected" (there is a small dotted border around it) . This is the selection.

Let's try the links:

Code: CSS

```css
a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color : #FFCC66;
}
```

3. When the link has already been visited

It is possible to apply a style to a link to a page that has already been viewed. By default, the browser colors the link purple.

You can change this appearance with :visited (which means "visited"). In practice, you can't change a lot of things except the color on the visited links.

Code: CSS

```
a:visited /* Quand le visiteur a déjà vu la page concernée */
{
    color: #AAA; /* Appliquer une couleur grise */
}
```

If you do not want previously visited links to be colored differently, you will need to apply the same color to them as normal links.

PARTIE III : MISE EN PAGE DU SITE

We learned to build basic pages in HTML, to modify the formatting with CSS... Now let's look at the layout of our site! At the end of this part, we will end up with our first complete site, arranged as we want!

CHAPTER I: STRUCTURING YOUR PAGE

In general, a web page consists of a **header** (at the very top), **navigation menus** (at the top or on the sides), different **sections** in the center... and a **footer** (all downstairs).
We will focus in this chapter on the new HTML tags dedicated to the structuring of the site. These tags were introduced by HTML5 (they didn't exist before) and will allow us to say: "This is my header", "This is my navigation menu", etc.
We're not going to do a layout right now. We will actually prepare our HTML document to be able to discover the layout in the next chapters.

I. HTML5 structuring tags

I will present here the new tags introduced by HTML5 to structure our pages. You'll see, this won't change the look of our site much at this time, but it will be well built and ready to be formatted next.

1. < **header** >: the header

Most websites generally have a header, called " **header**" in English. There is most often a **logo** , a **banner** , the **slogan** of your site...

You will need to place this information inside the <header> tag :

Code: HTML

```
<header>
    <!-- Placez ici le contenu de l'en-tête de votre page  -->
</header>
```

The header can contain anything you want: images, links, texts...

NB : There may be several headers in your page. If it is split into several sections, each section can indeed have its own **<header>** .

2. < footer > : the footer

Unlike the header, the footer is usually found at the very bottom of the document. It contains information such as contact links, the name of the author, legal notices, etc.

Code: HTML

```
<footer>
    <!-- Placez ici le contenu du pied de page  -->
</footer>
```

3. < nav > : main navigation links

The <nav> tag should include all of **the** site's main navigation links. For example, you will place the main menu of your site there.

realized as a bulleted list inside the **<nav> tag** :

Code: HTML

```
<nav>
    <ul>
        <li><a href="index.html">Accueil</a></li>
        <li><a href="forum.html">Forum</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</nav>
```

4. <section> : a page section

The **<section> tag** is used to group content according to their theme. It generally encompasses a portion of the content in the center of the page.

Code: HTML

```
<section>
    <h1>Ma section de page</h1>
    <p>Bla blablabla  </p>
</section>
```

NB :Each section can have its level 1 title (**<h1>**), just as the header can contain a **<h1> title** too. Each of these blocks being independent of the others, it is not illogical to find several **<h1> titles** in the code of your web page. We thus have "The title **<h1>** of the **<header>** ", " The title **<h1>** of this **<section>** ", etc.

5. < aside >: additional information

The **< aside > tag** is designed to contain additional information to the document being viewed. This information is usually placed on the side (although this is not a requirement).

Code: HTML

```
<aside>
    <!-- Placez ici des informations complémentaires  -->
</aside>
```

There can be multiple **< aside > blocks** in the page.

On Wikipedia for example, it is common to see a block of additional information on the right

6. < article > : an independent article

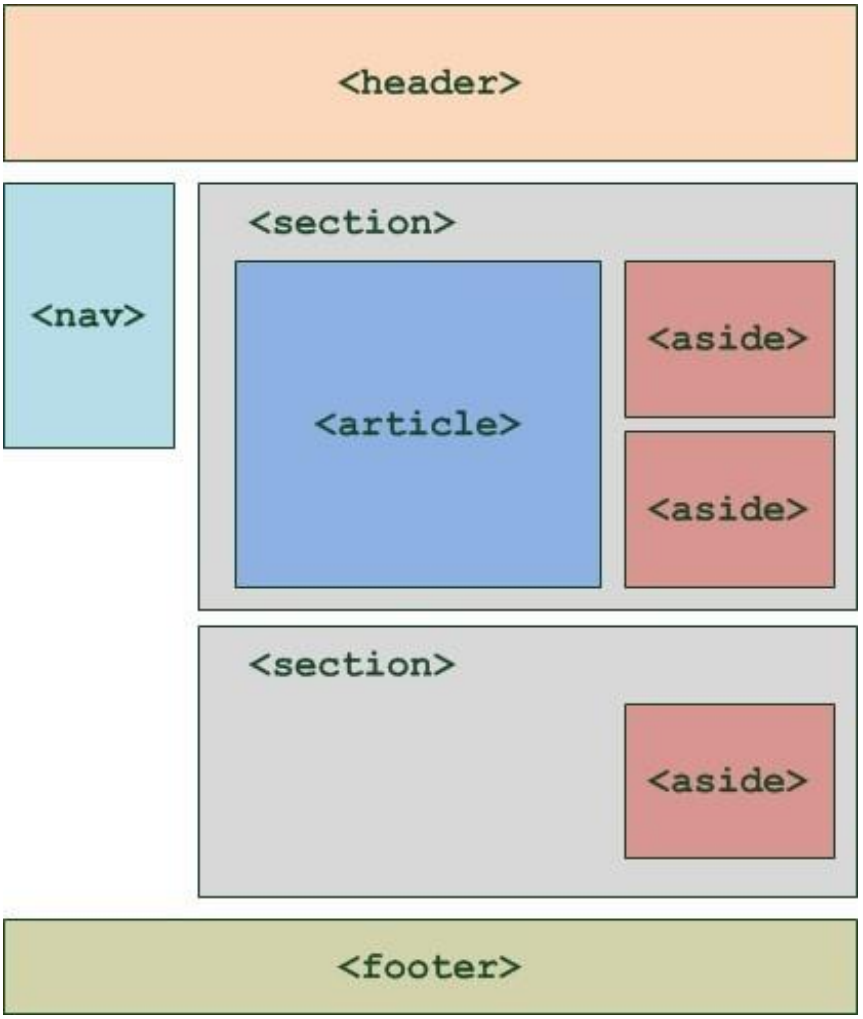
The **<article> tag** is used to encapsulate a generally self-contained portion of the page. It is a portion of the page that could (for example) be taken up elsewhere on another site. This is the case, for example, of news (newspaper or blog articles).

Code: HTML

```
<article>
    <h1>Mon article</h1>
    <p>Bla blablabla  </p>
</article>
```

Summary :

We can see here what these tags look like.



NB: This diagram offers an example of page organization. Nothing prevents you from deciding that your navigation menu should be on the right, or at the very top, that your **< aside > tags** are above, etc.

We can even imagine a second **<header> tag** , this time placed inside a **<section>** . In this case , it will be considered as the header of the section.

Finally, a section does not have to contain an **<article>** and **< aside >** . Use these tags only if you need them. Nothing prevents you from creating sections containing only paragraphs for example.

❖ Ensure compatibility with IE

The new tags that we have just seen are only recognized by Internet Explorer since version 9 (IE9). This will cause a problem because, when older versions of IE don't know a tag... they don't process the page correctly (impossible to modify their css for example)!

Fortunately, this can be fixed quite easily using a **JavaScript script** . Scripts are small pieces of code that allow you to manipulate the web page and perform certain actions **.** We are not going to focus on JavaScript here , but you should know that they are called from HTML pages, much like CSS files.

JavaScript files are usually files with the extension **. js** . In our HTML code, we generally place them in the **< head > tag** with this tag:

Code: HTML

```
<script src="monscript.js"></script>
```

CHAPTER II THE BOX MODEL

A web page can be seen as a succession and a stack of boxes, called " **blocks** ". Most of the elements seen in the previous chapter are blocks: **<header>** , **<article>** , **< nav >** ... But we already knew other blocks: paragraphs ** ** , titles **<h1>** ...

In this chapter, we will learn how to manipulate these blocks like real boxes. We are going to give them dimensions, arrange them by playing on their margins, but also learn how to manage their content... to prevent the text from protruding from these blocks!

These are fundamental notions that we will need to lay out our website...

I. inline type tags

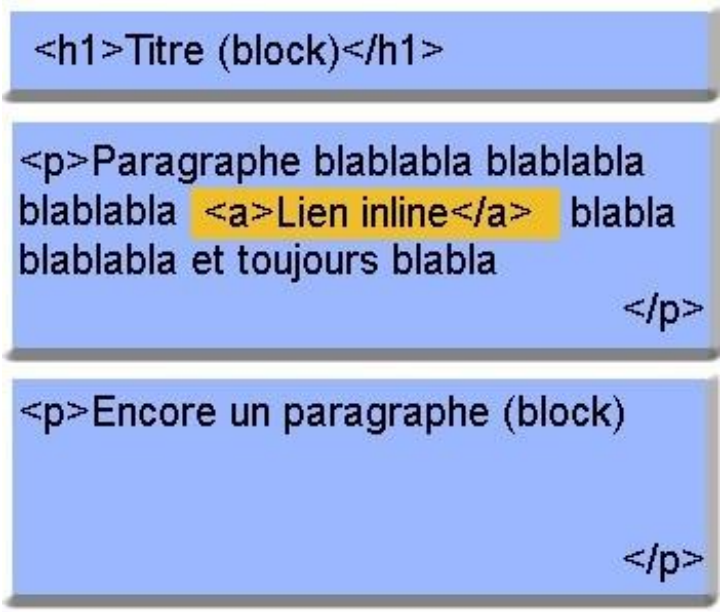
In HTML, most tags fall into two categories:

- ✓ *Inline* tags : this is the case for example of **<a> links** .
- ✓ *Block* tags : this is the case for example of ** paragraphs** .

There are in fact several other very specific categories, for example for table cells (tablecell type) or bullet points (list -item type).

- **block** : a "block" type tag on your web page automatically creates a line break before and after. Just imagine a block. Your web page will actually consist of a series of blocks one after the other. But you will see that in addition, it is possible to put one block inside another, which will considerably increase our possibilities for creating the design of our site!
- **inline : an "** inline " type tag must be inside a "block" tag. An inline tag does not create a line break, the text inside is therefore written following the previous text, on the same line (this is why we speak of a "tag" on line").

To better visualize the concept, here is a small diagram that I have concocted for you:



On a blue background, you have everything that is of type block.

inline type .

As you can see, the blocks are one below the other. You can also nest them inside each other (remember, our <section> blocks **contain <aside> blocks** , for example !). **The** inline **<a>** tag is found at inside a block tag and the text is inserted on the same line.

Some examples

To better help you understand which tags are inline and which tags are block, here is a small table listing some common tags.

block tags	Inline tags
<p>	_ _
< footer >	_ _
<h1>	<mark>
<h2>	<a>
<item>	< img />
...	...

This table is not complete.

II. Universal tags

These are tags that have no particular meaning (unlike **<p>** which means "paragraph", **< strong >** "important", etc.).

The main interest of these tags is to be able to apply a class (or an id) to them for CSS when no other tag is suitable.

There are 2 generic tags, and coincidentally the only difference between the two is that one of them is inline , the other is block:

- ✓ **** (*inline*)
- ✓ ** ** (*block*)

1. The dimensions

Here we will only work on "block" type tags. To start, let's look at block size. Unlike an inline , a block has precise dimensions. It has a width and a height.

This means that we have 2 CSS properties:

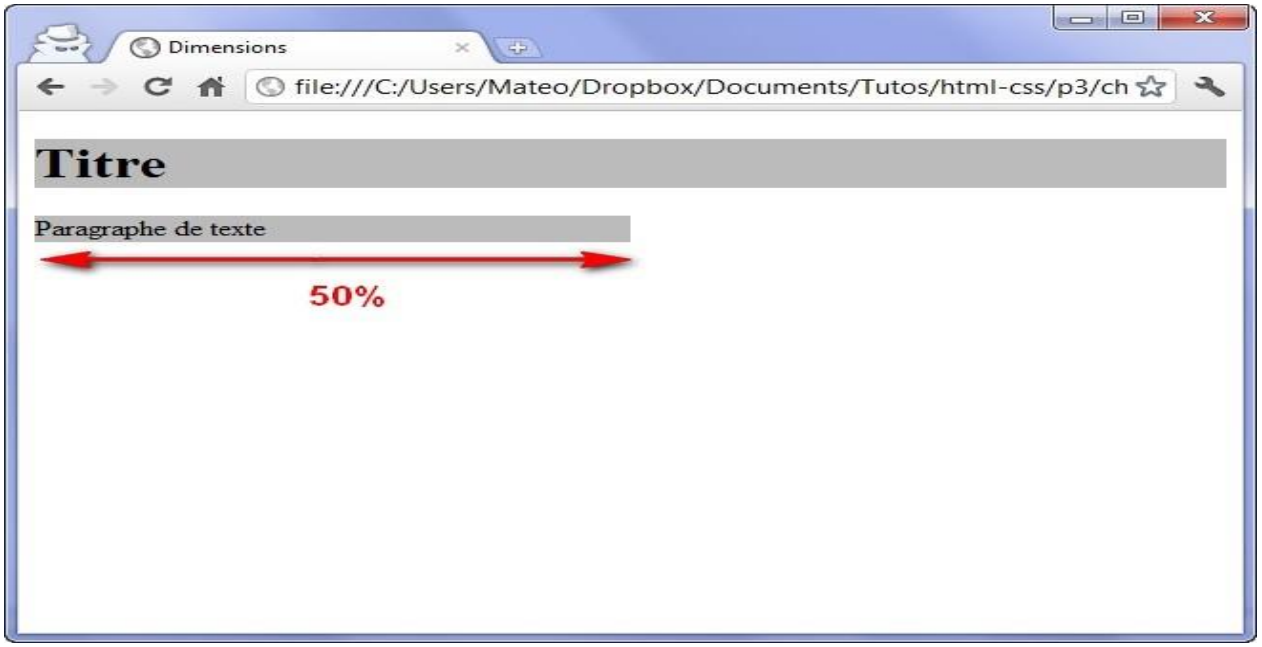
- ➢ **width** : this is the width of the block. To be expressed in pixels (px) or in percentage (%).
- ➢ **height** : this is the height of the block . Again, it is expressed either in pixels (px) or in percentage (%).

NB : **width** and **height** represent the width and height of the *content* of the blocks. If the block has margins (we will discover this principle a little later), these will be added to the width and the height. By default, a block takes 100% of the available width. But, we can change that.

Example : The following CSS says: "I want all my paragraphs to be 50% wide".

Code: CSS

```
p
{
    width: 50%;
}
```



The percentages will be useful for creating a design that automatically adapts to the resolution of the visitor. However, you may need to create blocks with a specific pixel dimension:

Code: CSS

```
p
{
    width: 250px;
}
```

a. min and max

A block can be required to have minimum and maximum dimensions. This is very convenient, as it allows us to set "limit" dimensions so that our site adapts to the different screen resolutions of our visitors.

- **min - width** : minimum width
- **min - height** : minimum height
- **max - width** : maximum width
- **max - height** : maximum height

For example, you can ask for paragraphs to be 50% of the width *and* require them to be at least 400 pixels wide in all cases:

Code: CSS

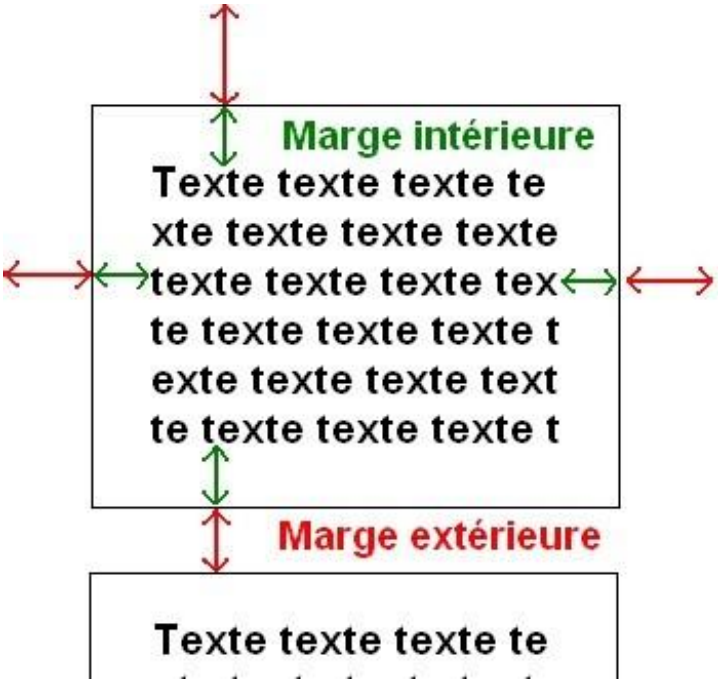
```
p
{
    width: 50%;
    min-width: 400px;
}
```

2. Margins

You should know that all blocks have margins. There are **2 types of margins** :

- The inner margins
- The outer margins

Drawing



- The space between the text and the border is the inside margin (in green).
- The space between the border and the next block is the outer margin (in red).

In CSS, you can change the size of the margins with the following 2 properties:

- ✓ **padding** : indicates the size of the inner margin . To be expressed in general in pixels (px).
- ✓ **margin** : indicates the size of the outer margin . Again, pixels are most often used.

To see the margins well, let's take 2 paragraphs to which I simply put a small border:

Code: CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
}
```

As you can see, there is no padding by default . On the other hand, there is an outer margin (margin). It is this margin that causes 2 paragraphs not to be glued and that we have the impression of "skipping a line".

Note : The default margins are not the same for all block type tags. Try to apply this CSS to ** ** tags that contain text for example, you will see that there is no inner margin or outer margin by default!

Suppose I want to add a 12px padding to paragraphs:

Code: CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
padding: 12px; /* Marge intérieure de 12px */
}
```

Now I want my paragraphs to be more spaced between them. I add the **margin property** to ask for a 50px margin between 2 paragraphs:

Code: CSS

```css
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
margin: 50px; /* Marge extérieure de 50px */
}
```

If you want to indicate a margin at the top, bottom, left and right, you will have to use more precise properties... The principle is the same as for the **border property** .

a. Up, right, left, down...

Here we will use the following properties:

- **Top** : top
- **Bottom** : bottom
- **Left** : left
- **Right** : right

I'm going to list the properties for **margin** and **padding anyway** , so you can be sure you understand the principle.

Here is the list for **margin** :

- **margin -top** : outer margin at the top.
- **margin -bottom** : bottom outer margin.
- **margin -left** : left outer margin.
- **margin -right** : outer margin on the right

And the list for **padding** :

- **padding -top** : padding at the top.
- **padding -bottom** : bottom padding.
- **padding -left** : left padding.
- **padding -right** : padding on the right.

NB : There are other ways to specify margins with the **margin** and **padding** properties . For example : **margin** : **2px03px1px** ;
... means "2px margin top, 0px right (px is optional), 3px bottom, 1px left. Other shorthand notation : **margin** : **2px1px** ;
...means "2px margin top and bottom, 1px margin left and right".

III. Center Blocks

It is quite possible to center blocks. It's even very useful for creating a centered design when you don't know the visitor's resolution.
To center, the following rules must be respected:

- Give the block a width (with the **width property**)
- Indicate that you want automatic outer margins, like this: **margin** : **auto** ;

Let's try this technique on our small paragraphs:

Code: CSS

```css
p
{
width: 350px; /* On a indiqué une largeur (obligatoire) */
margin: auto; /* On peut donc demander à ce que le bloc soit centré
avec "auto" */
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 20px;
}
```

Thus, the browser automatically centers our paragraphs!

1. When it exceeds...

When we start to define precise dimensions for our blocks, as we have just done, sometimes they become too small for the text they contain.

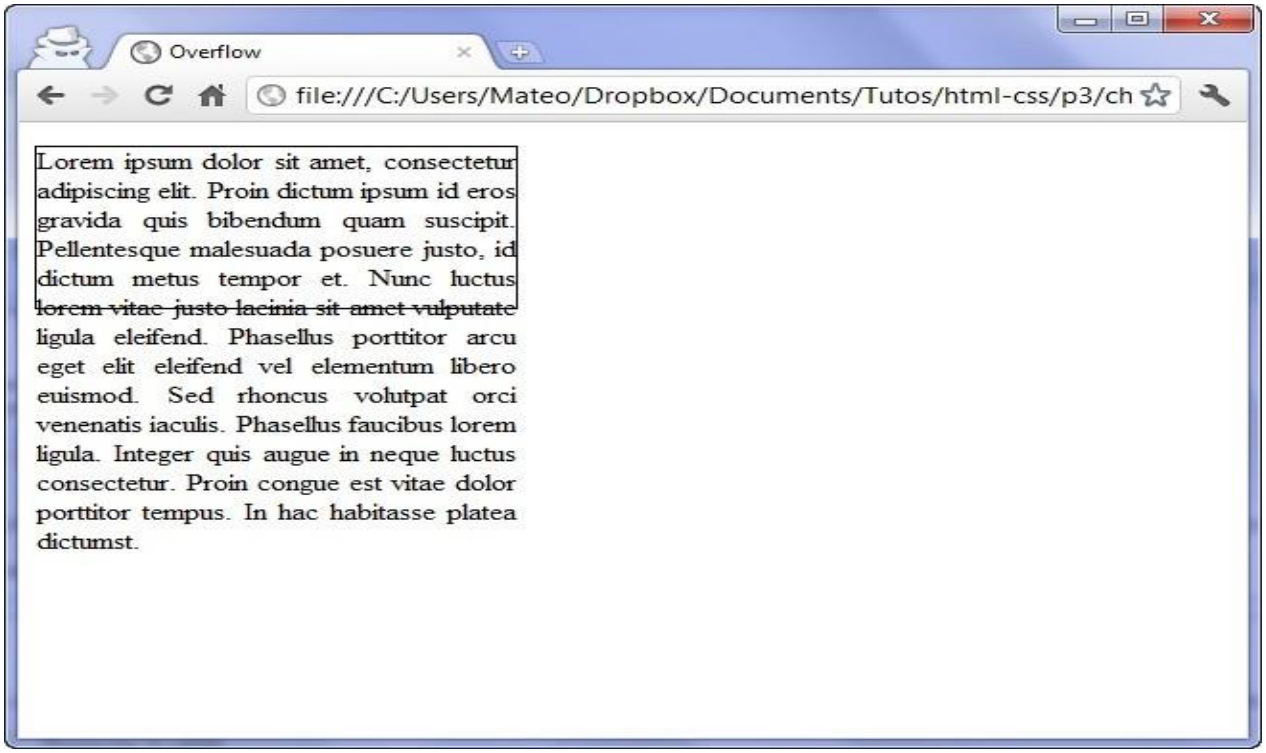
The CSS properties that we are going to see here were created precisely to control overflows... and to decide what to do if ever it should happen.

a. overflow : cut a block

Suppose you have a long paragraph and you want (for whatever reason) you want it to be 250px wide and 110px high. Let's add a border to it and fill it with text...to the brim :

Code: CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
}
```



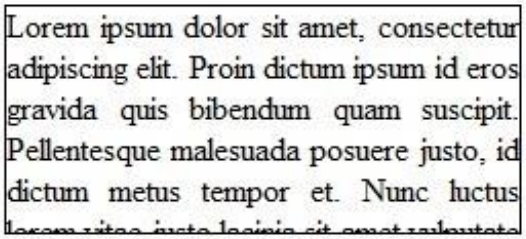
Eh yes ! You asked for specific dimensions, you got them! But... the text doesn't fit inside such a small block.

If you don't want the text to go beyond the paragraph boundaries, you'll have to use the **overflow property** . Here are the values it can accept:

- ✓ **visible** (default): if the text exceeds the size limits, it remains visible and voluntarily leaves the block.
- ✓ **hidden** : if the text exceeds the limits, it will simply be cut off. You won't be able to see all the text.

- ✓ **scroll** : again, the text will be cut if it exceeds the limits. Except that this time, the browser will set up scroll bars so that we can see all the text. It's a bit like a frame inside the page.
- ✓ **auto** : this is the "automatic pilot" mode. Basically, it's the browser that decides whether or not to put scrollbars (it will only put them if necessary). This is the value I recommend using most often.

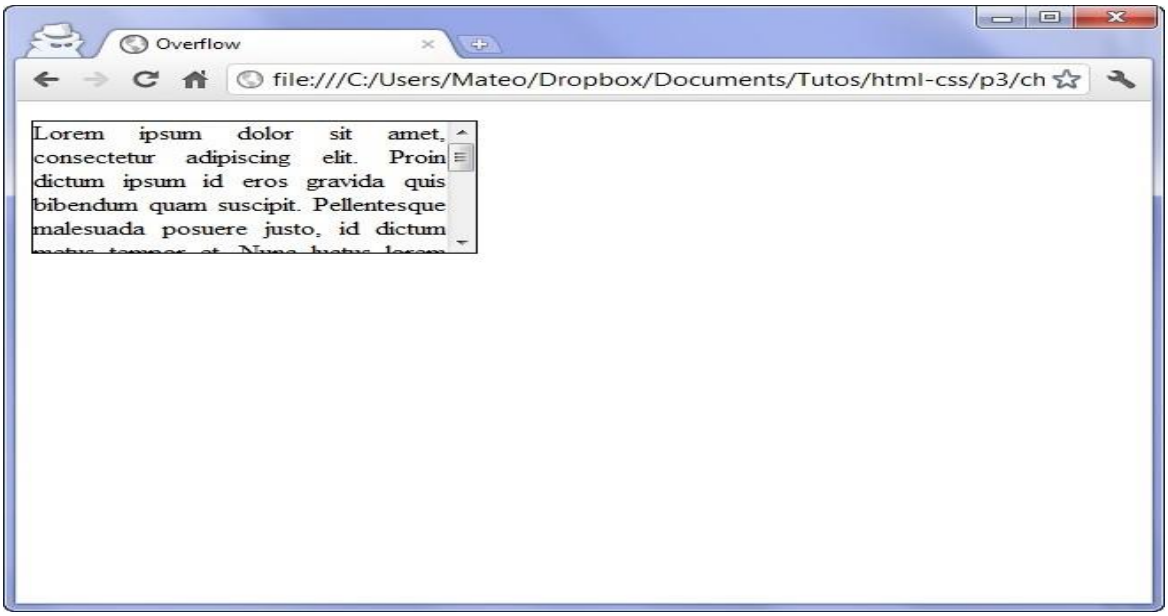
With **overflow** : **hidden** ; the text is therefore cut off (we cannot see the rest):



Now let's `try` **overflow** : `auto` ; with the following CSS code:

Code: CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
    overflow: auto;
}
```

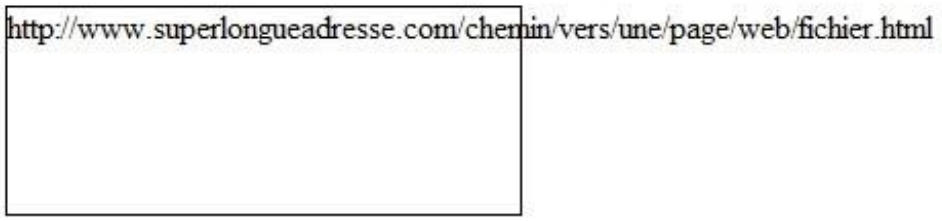


Scroll bars now allow us to view content that was not visible .

b. word -wrap: cut texts that are too wide

If you need to wrap a really long word in a block, which doesn't fit across the width, you'll love **word -wrap** . This property is used to force the hyphenation of very long words (generally slightly long addresses).

Here is for example a problem that we can have when we write a long URL in a block:

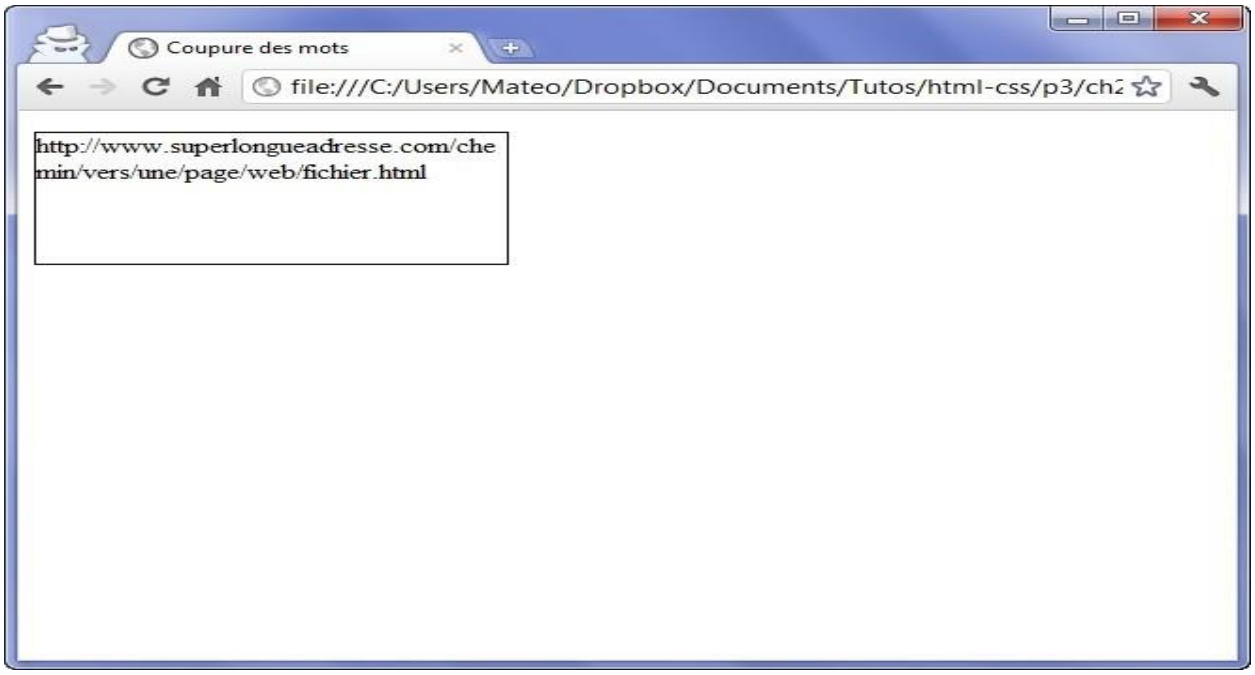


The computer does not know how to "cut" the address because there is no space or dash. He doesn't know how to make the hyphenation.

With the following code, the hyphenation will be forced as soon as the text is likely to exceed:

Code: CSS

```
p
{
    word-wrap: break-word;
}
```



NB : it is advisable to use this feature as soon as a block of text is likely to contain text entered by users (for example on the forums of your future site). Without this trick, you can easily "break" the design of a site (by writing a long series of " aaaaaaaaaa " for example).

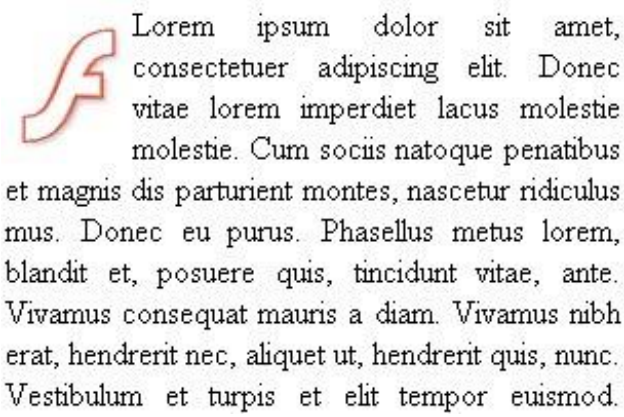
CHAPTER III: POSITIONING IN CSS

Here we will learn how to change the position of elements on our page.

You will see, there are several techniques to perform the layout of its site. Each has its advantages and disadvantages, it will be up to you to select the one that seems best to you according to your case.

I. Floating positioning

Remember the **float property** ? We used it to float an image around the text:



It turns out that this property is now used by the majority of websites to... do the layout! Indeed, if you want to place your menu on the left and the content of your page on the right, this is a good way. I say a priori, because this property was not designed to do the basic layout and we will see that it has some small flaws.

Let's go back to the structured HTML code that we created a few chapters ago:

Code: HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>A propos de l'auteur</h1>
                <p>C'est moi, Zozor ! Je suis né un 23 novembre
2005.</p>
            </aside>
            <article>
                <h1>Je suis un grand voyageur</h1>
                <p>Bla blablabla (texte de l'article)    </p>
            </article>
        </section>

        <footer>
            <p>Copyright Zozor - Tous droits réservés

            <a href="#">Me contacter !</a></p>
        </footer>

    </body>
</html>
```

I remind you that, without CSS, the layout looks like this:

We will try to place the menu on the left, and the rest of the text on the right. To do this, we'll float the menu to the left, and let the rest of the text flow to its right.

We want the menu to be 150 pixels wide. We are also going to add a black border to the menu and a blue border to the body (at the **<section>**) to distinguish them clearly:

We want the footer ("Copyright Zozor ") to be placed at the bottom under the menu, but on the other hand we would like the whole body of the page to be made up as a single block on the right.

To solve these two problems at once, we need to add an outer margin to the left of our **<section>** *greater* than the width of the menu. If our menu is 150px, we will for example give a left outer margin of 170px to our page section.

Code : CSS

```
nav
{
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
margin-left: 170px;
    border: 1px solid blue;
}
```

NB: Conversely, you may want an element to be placed under the menu. In this case, it will be necessary to use ... **clear** : **both** ; , which we had already discovered, which forces the continuation of the text to be positioned under the floating element.

❖ Transform your elements with display

There is a very powerful property in CSS: **display** . It is able to **transform** any element of your page from one type to another. With this magic property, I can for example transform my links (originally of inline type) into blocks :

Code: CSS

```
a
{
    display: block;
}
```

At this point, the links will position themselves one below the other (like normal blocks) and it becomes possible to modify their dimensions!
Here are some of the main values that the **display property can take** in CSS (there are still others):

Valeur	Exemples	Description
inline	<a> , , ... _ _ _ _	Elements of a line. Place themselves next to each other.
block	<p> , , _ _ <section> ...	Block-shaped elements. Place one below the other and can be resized.
inlineblock	<select> , _ _ <input>	Elements that are positioned next to each other (like inlines) but can be resized (like blocks).
none	< head >	Items not shown.

We can therefore decide to completely hide an element of the page with this property. For example, if I want to hide elements that have the class "secret", I will write:

Code : CSS

```
.secret
{
    display: none;
}
```

II. Inline -block positioning

If floating positioning remains, by far, the most widely used positioning mode on the Web today, other techniques exist and very few webmasters know about them. One of them, surprisingly powerful, has passed under the eyes of web designers even though it has existed since CSS 2.1, that is to say for more than 10 years! It consists of transforming your elements into inline -block with the **display property** .

inline -block type elements :

- They are positioned next to each other (exactly what we want to place our menu and the body of our page!).
- You can give them precise dimensions (again, exactly what you want!).

We are going to transform into inline -block the two elements that we want to place side by side: the navigation menu and the center section of the page.

Code : CSS

```css
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
}

section
{
    display: inline-block;
    border: 1px solid blue;
}
```

inline -block elements position themselves on a same baseline (called *baseline*), at the bottom. Fortunately, having transformed the elements into inline -block allows us to use a new property, normally reserved for tables: **vertical- align** . This property allows you to change the vertical alignment of elements. Here are some of the possible values for this property:

- **baseline** : aligns the base of the element with that of the parent element (by default)

- **top** : align at the top
- **middle :** align in the middle
- **bottom** : align at the bottom
- **(value in px or %)** : aligns at a certain distance from the baseline (baseline)

All we have to do is align our elements at the top, and voila!

Code: CSS

```
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
vertical-align: top;
}

section
{
    display: inline-block;
    border: 1px solid blue;
vertical-align: top;
}
```

Note : You will notice that the body (the **<section>)** does not take up the full width. Indeed, it is no longer a block! The section takes up only the space it needs.

If this doesn't work for your design, modify the section size with **width** .

III. Absolute, fixed and relative positioning

There are other somewhat specific techniques for precisely positioning elements on the page:

- ✓ **Absolute positioning** : it allows us to place an element anywhere on the page (top left, bottom right, center, etc.)
- ✓ **Fixed positioning** : identical to absolute positioning, but this time the element always remains visible, even if you go further down the page. It's a bit like the same principle as **background- attachment** : **fixed** ;
- ✓ **Relative positioning** : allows the element to be offset from its normal position.

As with floats, absolute, fixed and relative positioning also work on inline type tags . block tags than on inline tags . You must first choose between the 3 positioning modes available. To do this, we use the CSS **position property** to which we give one of these values:

- **absolute** : absolute positioning.
- **fixed** : fixed positioning.
- **relative** : relative positioning.

We will study each of these positions one by one.

1. Absolute positioning

Absolute positioning allows an element to be placed (really) anywhere on the page. To perform an absolute positioning, we must write:

Code: CSS

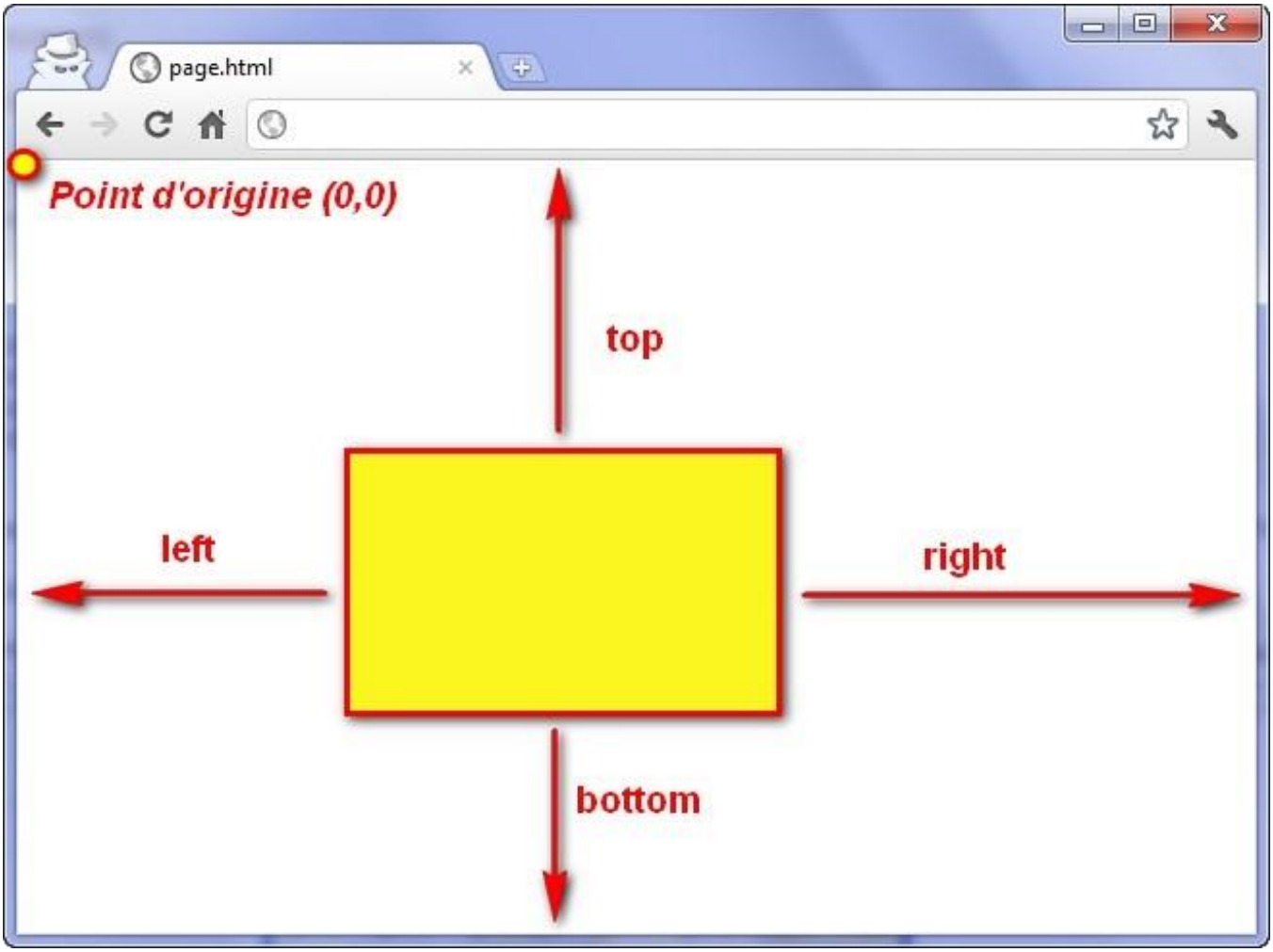
```
element
{
    position: absolute;
}
```

But that's not enough! We said we wanted absolute positioning, but we still have to say *where we want the block to be positioned on the page* .
To do this, we will use 4 CSS properties:

- ✓ **left** : position relative to the left of the page.
- ✓ **right** : position relative to the right of the page.
- ✓ **top** : position relative to the top of the page.
- ✓ **bottom** : position relative to the bottom of the page.

They can be given a pixel value, like "14px", or a percentage value, like "50%".

If it's not very clear to some of you, this diagram should help you understand:



With that, you should be able to position your block correctly.

You must therefore use the **position property** and at least one of the 4 properties above (**top** , **left** , **right** or **bottom**). If we write for example:

Code: CSS

```css
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

... this means that the block must be positioned at the very bottom right (0 pixels relative to the right of the page, 0 relative to the bottom of the page).

If we try to place our **< nav > block** at the bottom right of the page, we get this result:

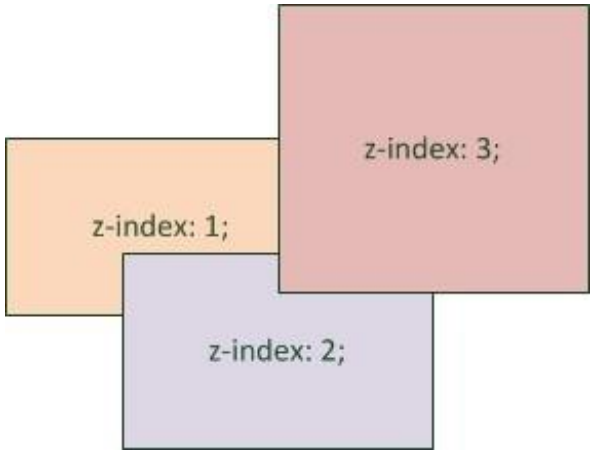
You can of course add an interior margin (padding) to the menu so that it is less stuck to its border.

Absolutely positioned elements are placed above the rest of the elements on the page! Also, if you move two absolute elements to the same location, they may overlap. In this case, use the **z-index property** to indicate which element should appear on top of the others:

Code: CSS

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
z-index: 1;
}
element2
{
    position: absolute;
    right: 30px;
    bottom: 30px;
z-index: 2;
}
```

The element with the highest z-index value will be placed on top of the others, as shown in this diagram:



NB : A small technical precision which is important: the absolute positioning is not always necessarily done in relation to the top left corner of the window! If you position in absolute terms a block A which is in another block B itself positioned in absolute (or fixed or relative), then your block A will be positioned in relation to the top left corner of block B.

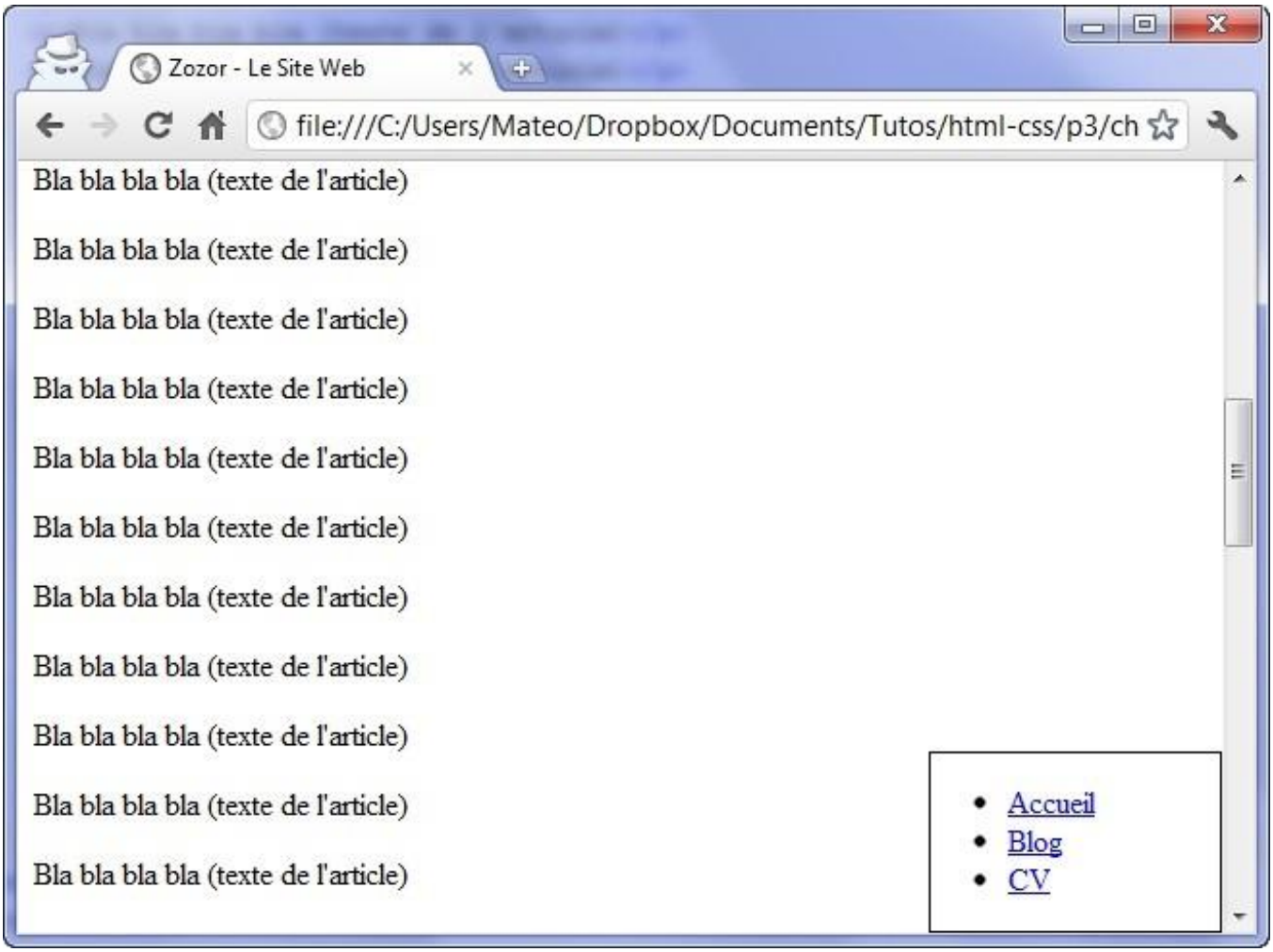
2. Fixed positioning

The principle is **exactly the same** as for absolute positioning, except that this time the block remains fixed in its position, even if you go further down the page.

Code: CSS

```css
element
{
    position: fixed;
    right: 0px;
    bottom: 0px;
}
```

Try the result, you will see that the menu remains in this case displayed at the bottom right even if you go further down the page.



3. Relative positioning

Relative positioning is a little trickier to use. This positioning allows you to make "adjustments": the element is offset from its initial position.

Take for example an important text, located between 2 **< strong > tags** . To start, I put it on a red background so that we can better identify it:

Code: CSS

```css
strong
{
   background-color : red; /* Fond rouge */
   color: yellow; /* Texte de couleur jaune */
}
```

This time, the diagram that I showed you earlier for the absolute and fixed positions no longer works. For what ? Because the origin has changed: the coordinate point (0, 0) is no longer at the top left of your window as it was earlier. No, this time the origin is at the top left... of your element's current position.

Twisted isn't it? This is the principle of relative position. This diagram should help you understand where the origin of the points is:

So if you do a **position** : **relative** ; and you apply one of the properties **top** , **left** , **right** or **bottom** , the text with a red background will move relative to the position where it is.

Let's take an example: I want my text to shift 55 pixels to the right and 10 pixels down. So I'm going to ask for it to be offset by 55 pixels from the "left edge", and 10 pixels from the "top edge":

Code: CSS

```css
strong
{
   background-color : red;
   color: yellow;

position: relative;
left: 55px;
top: 10px;
}
```

The text then shifted from its initial position, like this:

Happy reading !!!!