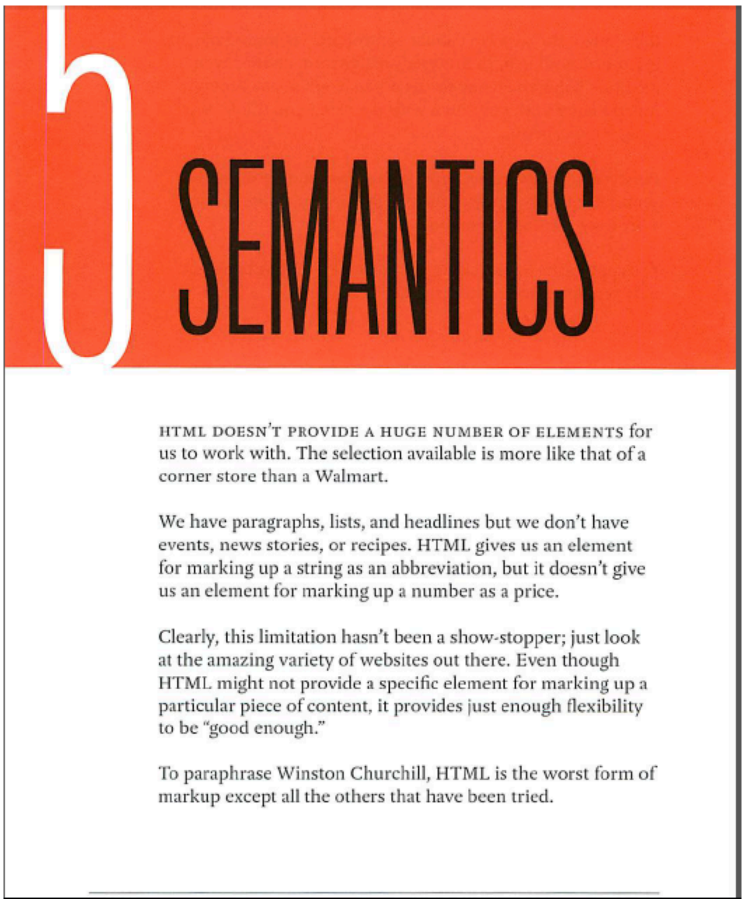
Exercises Frontend day 2

1. Markup exercise

Make a html5 document were you markup the follow text from chapter 5 in the HTML5 book with propper semantic tags (h1,h2,h3,p,q,code…) .



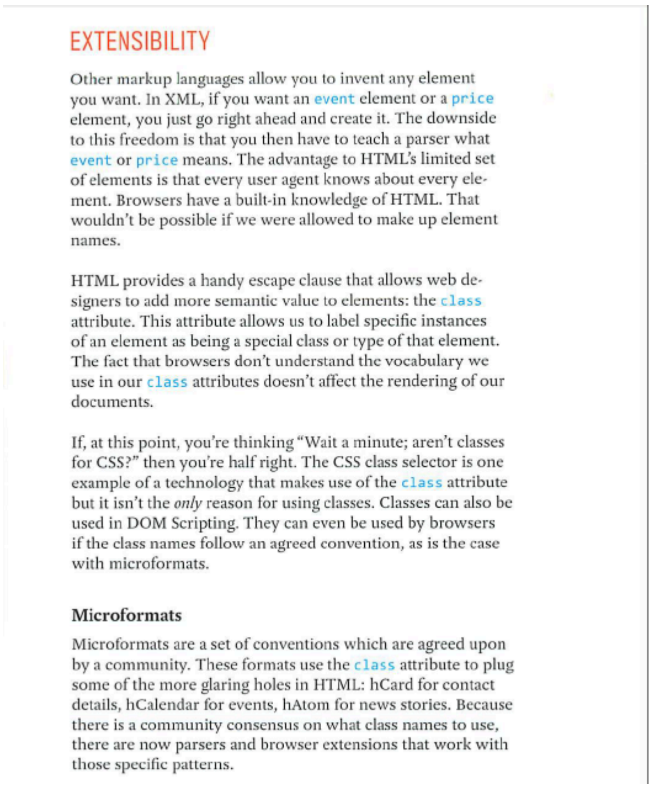
5 SEMANTICS

HTML DOESN'T PROVIDE A HUGE NUMBER OF ELEMENTS for us to work with. The selection available is more like that of a corner store than a Walmart.

We have paragraphs, lists, and headlines but we don't have events, news stories, or recipes. HTML gives us an element for marking up a string as an abbreviation, but it doesn't give us an element for marking up a number as a price.

Clearly, this limitation hasn't been a show-stopper; just look at the amazing variety of websites out there. Even though HTML might not provide a specific element for marking up a particular piece of content, it provides just enough flexibility to be "good enough."

To paraphrase Winston Churchill, HTML is the worst form of markup except all the others that have been tried.



EXTENSIBILITY

Other markup languages allow you to invent any element you want. In XML, if you want an `event` element or a `price` element, you just go right ahead and create it. The downside to this freedom is that you then have to teach a parser what `event` or `price` means. The advantage to HTML's limited set of elements is that every user agent knows about every element. Browsers have a built-in knowledge of HTML. That wouldn't be possible if we were allowed to make up element names.

HTML provides a handy escape clause that allows web designers to add more semantic value to elements: the `class` attribute. This attribute allows us to label specific instances of an element as being a special class or type of that element. The fact that browsers don't understand the vocabulary we use in our `class` attributes doesn't affect the rendering of our documents.

If, at this point, you're thinking "Wait a minute; aren't classes for CSS?" then you're half right. The CSS class selector is one example of a technology that makes use of the `class` attribute but it isn't the *only* reason for using classes. Classes can also be used in DOM Scripting. They can even be used by browsers if the class names follow an agreed convention, as is the case with microformats.

Microformats

Microformats are a set of conventions which are agreed upon by a community. These formats use the `class` attribute to plug some of the more glaring holes in HTML: hCard for contact details, hCalendar for events, hAtom for news stories. Because there is a community consensus on what class names to use, there are now parsers and browser extensions that work with those specific patterns.

The text without styling and markup:

SEMANTICS
HTML DOESN'T PROVIDE A HUGE NUMBER OF ELEMENTS for us to work with. The selection available is more like that of a corner store than a Walmart.
We have paragraphs, lists, and headlines but we don't have events, news stories, or recipes. HTML gives us an element for marking up a string as an abbreviation, but it doesn't give us an element for marking up a number as a price.
Clearly, this limitation hasn't been a show-stopper; just look at the amazing variety of websites out there. Even though HTML might not provide a specific element for marking up a particular piece of content, it provides just enough flexibility to be "good enough."

EXTENSIBILITY
Other markup languages allow you to invent any element you want. In XML, if you want an event element or a price element, you just go right ahead and create it. The downside to this freedom is that you then have to teach a parser what event or price means. The advantage to HTML's limited set of elements is that every user agent knows about every ele ment. Browsers have a built-in knowledge of HTML. That wouldn't be possible if we were allowed to make up element names.
HTML provides a handy escape clause that allows web de signers to add more semantic value to elements: the class attribute. This attribute allows us to label specific instances of an element as being a special class or type of that element. The fact that browsers don't understand the vocabulary we use in our class attributes doesn't affect the rendering of our documents.
If, at this point, you're thinking "Wait a minute; aren't classes for CSS?" then you're half right. The CSS class selector is one example of a technology that makes use of the class attribute but it isn't the onlyreason for using classes. Classes can also be used in DOM Scripting. They can even be used by browsers if the class names follow an agreed convention, as is the case with microformats.

Microformats
Microformats are a set ofconventions which are agreed upon by a community. These formats use the class attribute to plug some of the more glaring holes in HTML: hCard for contact details, hCalendar for events, hAtom for news stories. Because there is a community consensus on what class names to use, there are now parsers and browser extensions that work with those specific patterns.

Try also to style the text with internal css so it almost looks like the text in the book.

<style>

</style>

Exercise 2: XHTML 1.0 Strict with centered layout

Make an absolute centered layout (#allcontent) that validates in
xhtml 1.0 strict

1. Make a webpage with an absolute (960px) layout that is centered.

Hint:

```
<div id="allcontent">........</div>

#allcontent{

        margin: 0 auto;

        width: 960px;

        height: 600px;

        background-color: tan;

 }
```

Hints:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="da">
<head>

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

<title>Untitled Document</title>
</head>
```

Link to an external css:

<link type="text/css" rel="stylesheet" href="../css/mystyle1.css" />

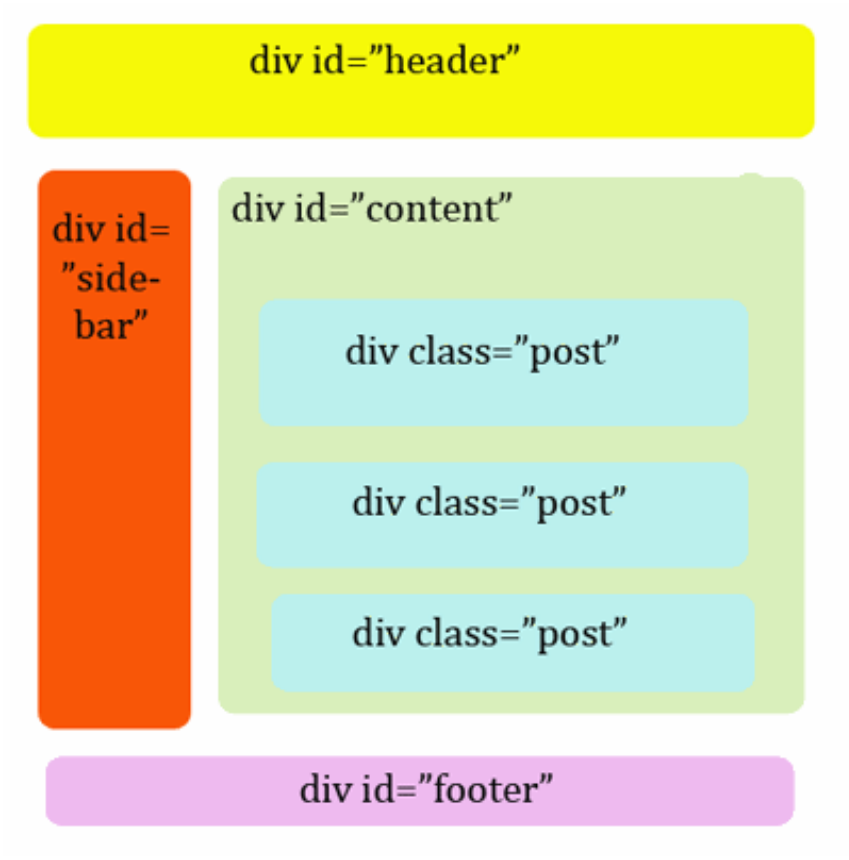
Hint 2. **Basic css elements**

Exercise 3. Make a 2 column layout with CSS

Try to implement the following wireframe on the webpage using

div tags, id and class attributes in the html code and e.g.

Selectors: #header and .post
Properties: float: left ; margin-left: 100px; background-color: yellow;



Save as exercise3.html

Exercise 4: Tools for viewing source code and testing a webpage

1. **Browser developer tools**: Try to e.g. use the Firefox addon Webdeveloper or Chrome view/developer (ctrl + I / alt+cmd+ I)tool to examine the html and css of webpage (prototype)

2. Try to test if your exercise1.html page is **valid XHTML1.0 Strict**

http://validator.w3.org/

3. Try to **cross browser test** the webpage (prototype) e.g. with

http://netrenderer.com/

Which versions of IE gives problems with the webpage?

More about cross browser testing tools
http://www.smashingmagazine.com/2011/08/07/a-dozen-cross-browser-testing-tools/

Exercise 5: Navigation

Menus with list elements: Try to e.g. make a list based menu inside the header by looking the menu tutorial .

Hint: Examine this webpage webpage

Save as exercise5_menu.html

Exercise 6 Day2 exercise blog in HTML5

Content is King

6. Try to make a html5 version of the blog page (exercise 3) with a blogpost for each exercise 1-5, with a short text about the exercise and a link to your solution.

1. Markup exercise : solution (open in a new window)

   Make a html5 document were you markup the follow text from chapter 5 in the HTML5 book with propper semantic tags (h1,h2,h3,p,q,code…)

Try to style it nicher then exercise 3
Hint:
http://code.tutsplus.com/tutorials/html-5-and-css-3-the-techniques-youll-soon-be-using--net-5708 (with so new html5 semantic tags)

Save as day2_exercises.html

Upload Frontendexercise folder to your webserver

Test if it works in the browser!

Handin a URL to **day2_exercises.html**

Fronter-Handin two days before next Frontend dev class.