

1st assignment

1st exercise (35%)

A – Xpath & CSS selectors (10%)

Fill in the following table:

| CSS Selector | Equivalent XPath expression |
|------------------------------------|-----------------------------|
| ol ul > li#foo | //ol//ul/li[@id = "foo"] |
| | //p/descendant-or-self::* |
| | //*[@id != "foo"] //p[1] |
| a[href*=".aueb.gr"][href\$=".pdf"] | |
| img:nth-of-type(even) | |
| h2 + p | |
| ul > li:nth-child(3n+2) | |
| ol > li:nth-last-child(3n+2) | |
| p:only-child | |
| p:empty | |

Notes

- Suppose we're in a (not necessarily valid) XHTML document, so tag names are case sensitive
- The CSS selectors or XPath expressions you provide must match the same elements as their equivalents in **any DOM tree**

B – CSS Selectors (10%)

Suppose you have to make a website that appears right in a browser we'll call "X". X doesn't support some kinds of CSS selectors. How would you rewrite the following CSS selectors without using the unsupported features?

| CSS Selector we need | X browser doesn't support... | Equivalent supported CSS selector |
|--------------------------------------|--|-----------------------------------|
| <code>.error</code> | <ul style="list-style-type: none">• class selector (<code>.error</code>) | <code>[class~="error"]</code> |
| <code>#footer</code> | <ul style="list-style-type: none">• id selector (<code>#footer</code>) | |
| <code>a:link</code> | <ul style="list-style-type: none">• The <code>:link</code> pseudo-class | |
| <code>[lang "en"]</code> | <ul style="list-style-type: none">• <code>[attr =value]</code>• The <code>:lang()</code> pseudo-class | |
| <code>:not(:first-child)</code> | <ul style="list-style-type: none">• The <code>:not()</code> pseudo-class | |
| <code>img:not(:first-of-type)</code> | <ul style="list-style-type: none">• The <code>:not()</code> pseudo-class | |
| <code>li:nth-child(-n+3)</code> | <ul style="list-style-type: none">• The <code>:nth-child()</code> pseudo-class | |

Notes

- Any other CSS3 selector is supported by browser X
- The CSS selectors you provide must match the same elements as their equivalents in **any DOM tree**
- Specificity doesn't have to be the same

C – CSS Cascade, Inheritance etc (10%)

Find the text color in every highlighted element. Justify your answer.

| Author CSS | User Agent CSS | HTML fragment |
|---|----------------|---|
| <pre>.message { color: black; } .informative.message { color: navy; } [class~="informative"][class~="message"] { color: #333; }</pre> | | <pre><p class="informative message"> You have already voted </p></pre> |
| Answer: | | |
| <pre>body p { color: red !important; } p[style] { color: orange !important; }</pre> | | <pre><body> <p style="color: blue;">I'm a paragraph</p> </body></pre> |
| Answer: | | |

| Author CSS | User Agent CSS | HTML fragment |
|---|---|--|
| <pre>h1 + p, h1 + [id="foo"]p:first-of-type { color: red; } p:nth-child(odd) { color: gray; } p:first-child { color: black; }</pre> | <pre>body { color: black; }</pre> | <pre><body> <h1>Τίτλος</h1> <p id="foo">Lorem</p> <p>Ipsum</p> <p>dolor sit amet</p> </body></pre> |
| Answer: | | |
| <pre>a[rel~="tag"] { color: gray; } body > p a { color: navy; } a:nth-of-type(5n+1) { color: inherit; }</pre> | <pre>body { color: black; } a[href] { color: blue; }</pre> | <pre><body> <p>CSS</p> </body></pre> |
| Answer: | | |

| Author CSS | User Agent CSS | HTML fragment |
|--|--|--|
| <pre>#products { color: #333; } .error { color: red; } p:not(#products + p) { color: fuchsia !important; }</pre> | <pre>body { color: black; }</pre> | <pre><body id="products"> <p class="error"> Oops! Something went wrong. </p> </body></pre> |
| <i>Answer:</i> | | |
| <pre>#products p { color: inherit; } .error { color: red; }</pre> | <pre>body { color: black; }</pre> | <pre><body id="products"> <p class="error"> Oops! Something went wrong. </p> </body></pre> |
| <i>Answer:</i> | | |
| <pre>#products { color: orange !important; }</pre> | <pre>a[href] { color: blue !important; }</pre> | <pre> Products </pre> |
| <i>Answer:</i> | | |

| Author CSS | User Agent CSS | HTML fragment |
|--|----------------|---|
| <pre>p:first-child { color: red; } p:not(a) { color: black; }</pre> | | <pre><body> <p>This can be tricky&hellip;</p> </body></pre> |
| <i>Answer:</i> | | |

Notes

- Suppose there's no other CSS that affects the page
- Your answer must state a particular color (not "initial" or "inherit" for instance)
- Answers without a justification will not get any points

D – Freeform questions (5%)

1. Why do we need the **:checked** pseudo-class? Isn't the attribute selector **[checked]** sufficient?
2. Why do we need the **:lang(en)** pseudo-class? Isn't the attribute selector **[lang="en"]** sufficient?
3. As we saw, CSS selectors include combinators for **descendants** (E F), **children** (E > F) and **next siblings** (E + F, E ~ F). Could you imagine why there are no combinators for **ancestors**, **parents** and **previous siblings**, despite their obvious usefulness?

2nd exercise (20%)

Suppose you want to conduct a survey to collect responses on some topic of your choice. Create the survey form (in HTML), which has to include at least:

- Some fields for demographic data, including:
 - Gender
 - Age

- Education level
- Income
- None of the above should allow freeform text (it makes it harder to use the data for statistical reasons)
- A field for comments regarding the survey
- A field for the respondent's email, in case they want to participate in the lottery
- A submit button
- At least 5 questions that are relevant to your topic of choice, containing:
 - A scaling question for multiple items
 - A ranking question
 - A multiple choice question (with an "Other" option and an accompanying text field)

Notes

- The key to the best grade here is:
 - Properly structured HTML
 - Valid HTML
 - Using the right controls for each question
 - Accessibility
- You may use HTML5 form controls, but in the context of progressive enhancement.

3rd exercise (45%)

A - CSS (30%)

In the assignment's zip file, you will find the folder **css-gallery**. This folder contains a subfolder with images, an XHTML file (**index.xhtml**) and a folder with screenshots (For the medium and large views, a screengrab from the whole page is also included). You have to write a CSS file (**style.css**) which will style the XHTML page into looking like the provided screenshots/screengrabs.



Notes

- Your CSS will be checked in **Firefox 3.6.x**.
- **You are not allowed to modify the XHTML page** or use extra images.
- The different views feature must be implemented **with CSS alone**. **JavaScript is not allowed in this assignment**.
- You are not allowed to use non-standard properties, values or selectors.
- Reasonable deviations from the colors or fonts presented in the screenshots are fine, it's not a graphic design course.
- Besides the HTML page looking similar to the screenshots, your grade will also depend on how well your CSS scales in changes or additions of data.

B – XSLT (15%)

Suppose we decide to move the data (title, filename, flickr URL) of every image to a separate XML file, for two reasons:

- Reducing data repetition (Eg the alt text of the images is the same as the text below them)
- To make it easier to add more pictures or delete old ones

That file is **pictures.xml** and can be found in the root of the **css-gallery** folder. You have to write an **XSL Transformation (transformation.xsl)** that will turn the XML file into XHTML like the one in **index.xhtml**.

Notes

- Your XSLT will be checked in **Firefox 3.6.x**.
- **You are not allowed to modify the XML file.**

Resources you may need

This list is here to help you. Don't assume that these links are the only information you will need.

- <http://www.w3.org/TR/css3-selectors/>
- <http://www.w3.org/TR/CSS2/cascade.html>
- <http://www.w3.org/TR/xpath/>
- <http://www.w3.org/TR/WCAG10/>
- <http://www.w3.org/TR/WCAG10-HTML-TECHS/>
- <http://joelclark.org/book/sashay/serialization/>
- <http://www.w3.org/TR/xslt>