SUPINFO International University

Sign of Success

CSS 3

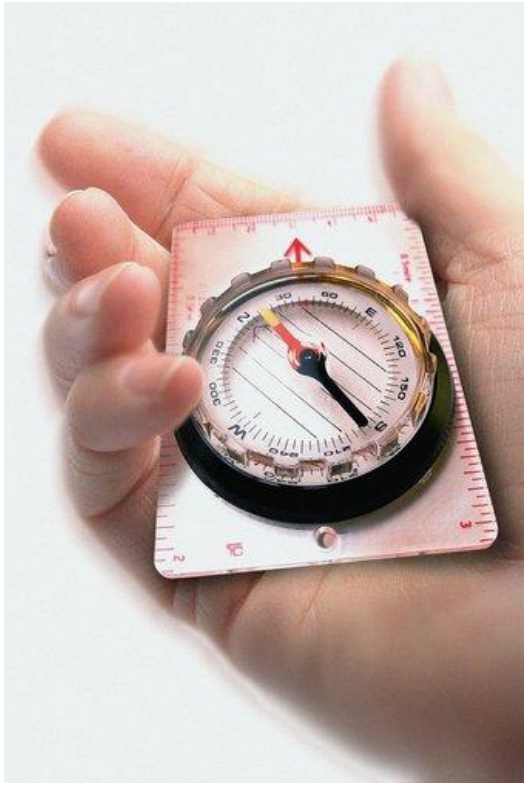
Web Design Revolution

Course objectives

By following this course, you will be able to:

- Enumerate CSS3 attributes
- Use level 3 selectors
- Do media queries
- Explain namespaces concept
- Use LESS Framework

Course plan



- Attributes
- Selectors level 3
- Media queries
- Namespaces
- LESS framework

CSS3

NEW ATTRIBUTES

Presentation

- CSS3 is part of HTML5 specification
- Composed of several specifications
- Developped by W3C and WHATWG
- At the moment, partially implemented by browsers

Presentation

- Since CSS3, new attributes are usable
- Can be sorted in four categories
  - Containers
  - Backgrounds
  - Texts
  - Transforms
- We'll discover them now!

Browser Compatibility

- Be careful: CSS3 support is still in progress!
- Depending on your browser, you may have to prefix attributes by the vendor specific code:

Browser	Property scheme
Internet Explorer	-ms-propertyName
Opera	-o-propertyName
Konqueror	-khtml-propertyName
Gecko based	-moz-propertyName
Webkit based	-webkit-propertyName

Containers

- Box-Shadow:
  - Set a shadow on blocks

border-radius: left top size color;

```
div { box-shadow: 4px 4px 2px #AAA; }
```

Sample text

Containers

- Box-sizing:
  - Allow to override default width and height calculation

box-sizing: type-box;

  - Where *type-box* can be:
    - border-box
    - padding-box
    - content-box

Containers

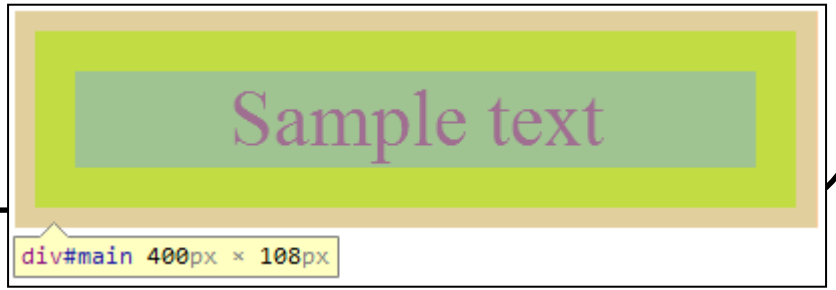
- Box-sizing:
  - Example with content-box:

```
div {
    box-sizing: content-box; padding: 20px;
    border:10px solid #AAA; width: 400px;
    /* Actual width is 460px (400 + 20*2 + 10*2) */
}
```

Containers

- Box-sizing:
  - Example with border-box:

```
div {
    box-sizing: border-box;
    padding: 20px;
    border:10px solid #AAA;
    width: 400px;
    /* Actual width is now
        400px */
}
```



Containers

- Border-Radius:
  - Round corners

border-radius: top-left top-right bottom-right bottom-left;

```
div { border-radius: 10px 5px 20px 0; }
```

Sample text

Containers

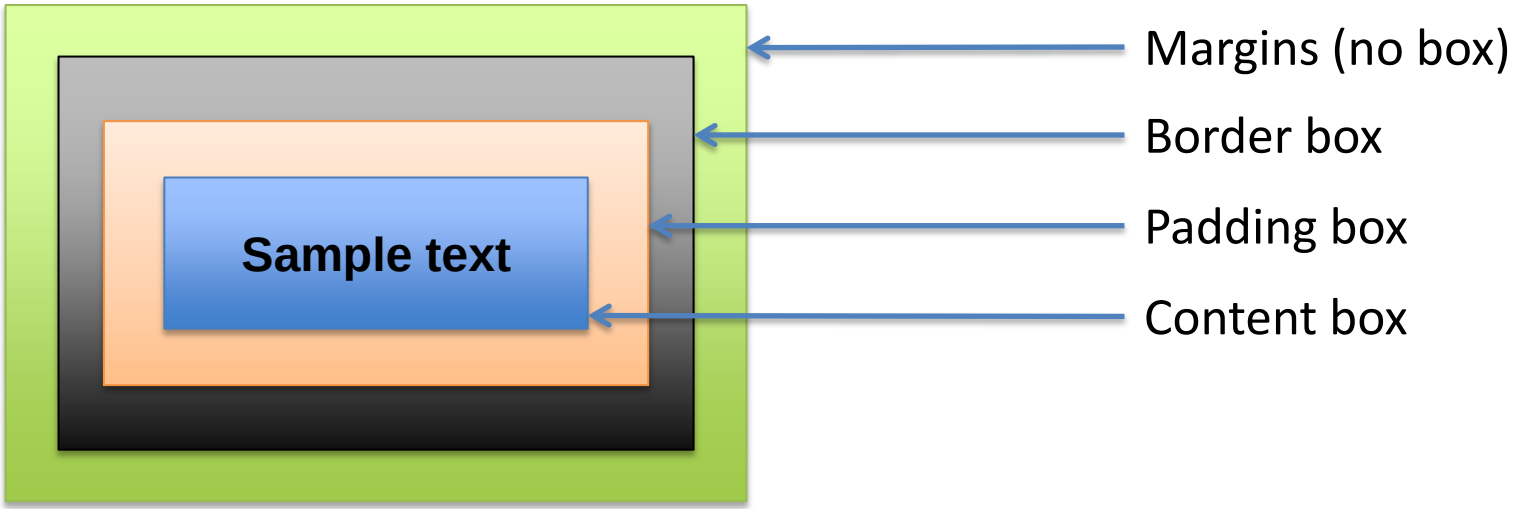
- Border-Image:
  - Renders a border based on a specific image
    - Outputs image over background definitions

border-image: url size x-mode y-mode;

```
div { border-image: url('css3.png') 20% repeat repeat; }
```

Backgrounds

- Background-Origin:
  - Change the default origin for background images



Backgrounds

- Background-Origin:
  - Change the default origin for background images

```
div {
    background-origin:
        border-box;
}
```

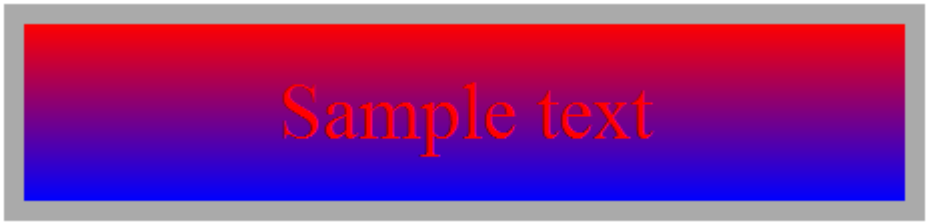
```
div {
    background-origin:
        padding-box;
}
```

Backgrounds

- Linear-Gradient:
  - Advanced gradient tool

background: linear-gradient(color1, color2 [,colorN])

```
div { background: linear-gradient(red, blue) }
```



Backgrounds

- Radial-Gradient:
  - Advanced gradient tool

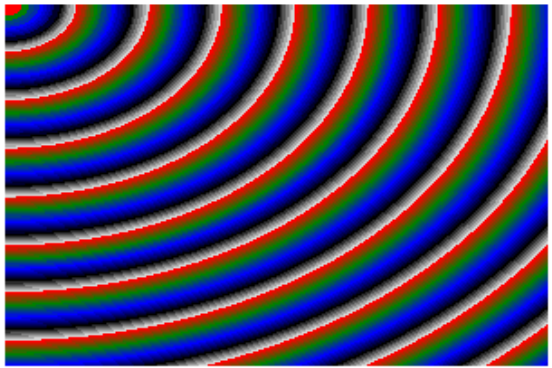
background: radial-gradient(circle, color1, color2);

```
div { background: radial-gradient(circle, red, blue) }
```

Backgrounds

- Radial-Gradient can take many parameters in order to perform complex gradients
  - Complex example:

```
div {
    background:
        repeating-radial-gradient(
            0 0, ellipse farthest-corner,
            red 0px,green 10px, blue 20px,
            black 30px, white 40px
        )
}
```



Backgrounds

- Box-Reflect:
  - Show a reflection effect on the target

box-reflect: direction margin mask-box;

```
div { box-reflect: below 5px
      linear-gradient(transparent, white); }
```

Texts

- Text-Shadow:
  - Set a shadow on texts

text-shadow: right top size color;

```
span { text-shadow: 4px 4px 2px #AAA; }
```

Sample text

Texts

- Text-Stroke:
  - Define a color wrap around texts

text-stroke: size color;

```
span { text-stroke: 2px black; }
```

Sample text

Texts

- Now you can set columns to blocks!

- Use these three properties:
  - Column-count:
    - Splits entire text by specified number
  - Column-gap
    - Sets margins between each column
  - Column-width
    - Specifies each column width

Texts

- Columns example:

column-count: number;

column-gap: gap;

column-width: width;

- Usage:

```
div {
    column-count: 5;
    column-gap: 10px;
    column-width: 40px;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam et magna in ligula condimentum interdum. Praesent ac felis ac magna semper scelerisque vel nec enim. Morbi aliquam, leo ut molestie vestibulum, massa nisi malesuada lacus, in dignissim nisl tellus non erat. Pellentesque ornare semper felis ut scelerisque. Pellentesque lacus odio, vehicula ac bibendum a, pulvinar et eros. Integer dapibus congue purus, ut molestie nunc pharetra ac.

Transforms

- Many transformations handled by the *transform* attribute
  - Scale
  - Rotate
  - Translate
  - Matrix
  - Perspective
  - Skew
- We'll see all the left ones

Transforms

- ScaleX/Y/Z:
  - Increase or reduce the element's dimensions

scaleX(number); scaleY(number); scaleZ(number);

```
div { transform: scaleX(2); }
```

Sample text

Attributes

Transforms

- RotateX/Y/Z:
  - Perform a rotation

rotateX(Ndeg); rotateY(Ndeg); rotateZ(Ndeg);

```
div { transform: rotateZ(10deg); }
```

Sample text

Transforms

- TranslateX/Y/Z:
  - Push the element in the specified direction

translateX(Ndeg); *translateY*(Ndeg); *translateZ*(Ndeg);

```
div { transform: translateX(-50px); }
```

Questions ?

CSS 3

SELECTORS LEVEL 3

Why new selectors?

- CSS 2 uses some specific selectors which targets specific HTML elements
  - Still available in CSS 3

- Several manipulations were impossible before
  - Target odd or even rows in a table
  - Find all disabled elements
  - Use regular expressions on attributes
  - …

What is new?

- New selectors (1 of 3):

Selector	Targetting
E[foo^="bar"]	An **E** element whose **"foo"** attribute value begins with **"bar"**
E[foo$="bar"]	An **E** element whose **"foo"** attribute value ends with **"bar"**
E[foo*="bar"]	An **E** element whose **"foo"** attribute value contains the substring **"bar"**
E:root	An **E** element, root of the document
E ~ F	An **F** element preceded by an **E** element

What is new?

- New selectors (2 of 3):

Selector	Targetting
E:nth-child(n) **E:nth-last-child(n)**	An **E** element, the n-th child of its parent… …counting from the last
E:nth-of-type(n) **E:nth-last-of-type(n)**	An **E** element,the n-th sibling of its type… …counting from the last
E:first-child **E:last-child**	An **E** element, first child of its parent… … counting from the last
E:empty	An **E** element that has no children (including text nodes)

What is new?

- New selectors (3 of 3):

Selector	Targetting
E:odd	An odd **E** element inside its parent container
E:even	An even **E** element inside its parent container
E:target	An **E** element targetted by the referring URI (#anchor)
E:enabled **E:disabled**	An **E** element which is enabled… …or disabled
E:checked	An UI element **E** which is checked (radio button, checkbox)
E:not(s)	An **E** element that does not match the simple selector **s**

Questions ?

CSS 3

MEDIA QUERIES

Presentation

- Handle CSS declarations depending on media used
  - Computers
  - Tablets
  - Smartphones
  - …
- Two ways to use it

Presentation

- In the stylesheet link:

```
<link rel="stylesheet"
      media="screen and (max-width: 600px)"
      href="small.css" />
```

- In CSS file directly:

```
@media screen and (max-width: 600px) {
    /* classic CSS here */
}
```

Media Types

- Ten types defined suitable for all displays:

Media type	Usage
all	Suitable for all displays
screen	Classical computer screen
speech	Speech synthesizers
tty	Terminals
tv	Television devices

Media Types

- Ten types defined suitable for all displays:

Media type	Usage
projection	Suitable for projectors
braille	Braille tactile feedback devices.
embossed	Paged braille printers
handheld	Small screen / limited bandwidth
print	Print preview mode (Ctrl + P on browser)

Min-Width / Max-Width

- Max-Width:
  - Useful for high resolution display:

```
@media screen and (max-width: 1900px) {
    #container { width: 1500px; }
}
```

- Min-Width:
  - Mostly smartphone related:

```
@media screen and (min-width: 600px) {
    #print-button { display: none; }
}
```

Max-Device-Width

- Min-width and max-width stands for resolution dedicated to browser (eg. window size)

- Max-device-width is related to screen resolution

```
@media screen and (max-device-width: 600px) {
    #footer { padding: 0; }
}
```

Combining media queries

- Operator « and » can combine them!
  - Already seen on previous slides

```
@media screen and (max-width: 23em) {}
```

- Operator « , » can associate combinations

```
@media handheld and (min-width: 20em), screen
          and (min-width: 20em){}
```

Combining media queries

- Some keywords must be placed in braces:
  - Used if the screen can render colors:

```
@media screen and (color) {}
```

- Operator « not » discard style inclusion if prerequisites are matched:
  - Not used if the media used is the print preview:

```
@media not print and (color) {}
```

Additional informations

- Most useful media queries are already available on the web, don’t reinvent the wheel !
  - Examples:
    - http://nmsdvid.com/snippets/
    - http://bit.ly/8XugAd

- Final note: Internet Explorer 9 and olders simply don’t support media queries.

Questions ?

Exercise (1/2)

- Take back the Web Worker exercise about tweets
  - Let’s do a special UI for smartphones!

- As drag and drop has a specific implementation on mobile browsers, it won’t work in our version
  - Don’t display the « Favorties » section for smartphones
  - Create a full width template for the « Latest Tweets » zone
    - Use CSS3 properties to make your page similar to a List View

Exercise (2/2)

- Possible rendering on Mobile Opera Emulator:

CSS 3

NAMESPACES

Presentation

- Remember the XML attribute « xmlns » ?
  - Allows to define namespaces
- CSS 3 can style XML elements depending on their respective namespaces
- We'll see how in this chapter ☺

XML namespaces

- An XML namespace is set by its attribute « xmlns »

```
<Enterprise xmlns="http://www.ns.com/ns/Enterprise">
```

- The namespace will be declared until the closing tag
  - Usually all namespaces are set in the first tag

```
<Enterprise xmlns="http://www.ns.com/ns/Enterprise"
    xmlns:manager="http://www.ns.com/ns/Manager"
    xmlns:employee="http://www.ns.com/ns/Employee">
```

Include CSS in XML

- Remember the doctype used for XML?

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- Right after it, include your stylesheet like this:

```
<?xml-stylesheet type="text/css" href="ns-style.css" ?>
```

XML namespace syntax

- How to include a specific tag in a namespace?

<namespace:theTag> </namespace:theTag>

- Non-prefixed tag belongs to the main namespace

```
<manager:Manager>Manager stuff</manager:Manager>
<info>Some unrelated informations</info>
```

Complete XML example

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/css"
    href="ns-style.css" ?>

<Enterprise xmlns="http://www.ns.com/ns/Enterprise"
    xmlns:manager="http://www.ns.com/ns/Manager"
    xmlns:employee="http://www.ns.com/ns/Employee">
  <manager:Manager>
    <manager:FirstName>Barney</manager:FirstName>
    <manager:LastName>Stinson</manager:LastName>
  </manager:Manager>

  <employee:Employee>
    <employee:FirstName>John</employee:FirstName>
    <employee:LastName>Doe</employee:LastName>
  </employee:Employee>
  <info>Description of the enterprise</info>
</Enterprise>
```

What about CSS?

- In CSS, define your namespaces as follows:

```
@namespace url("http://www.ns.com/ns/Enterprise");
@namespace manager url("http://www.ns.com/ns/Manager");
@namespace employee url("http://www.ns.com/ns/Employee");
```

- CSS namespaces are linked to XML namespaces thanks to the URL provided in both files:
  - In this example, root CSS namespace will concern all tags with the « Enterprise » namespace.

What about CSS?

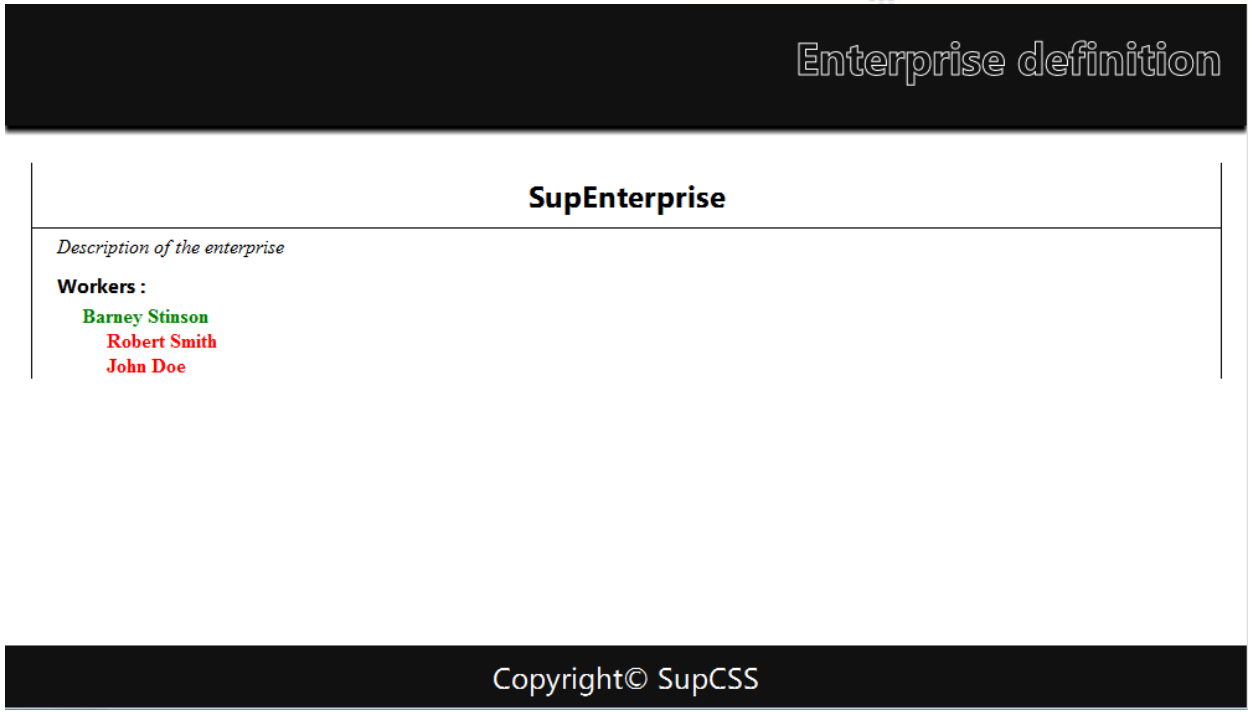
- After namespaces declaration, use it!

Selector	Target
ns|a	Represents tag **a** in the namespace **ns**
|b	Represents tag b that belongs to no namespace
***|c**	Represents tag c in any namespace
d	Represents tag d in the root namespace (as classical CSS)

```
manager|Manager { color: red; }
Enterprise employee|FirstName { color: green; }
*|LastName { font-weight: bold; }
```

Xml Rendering

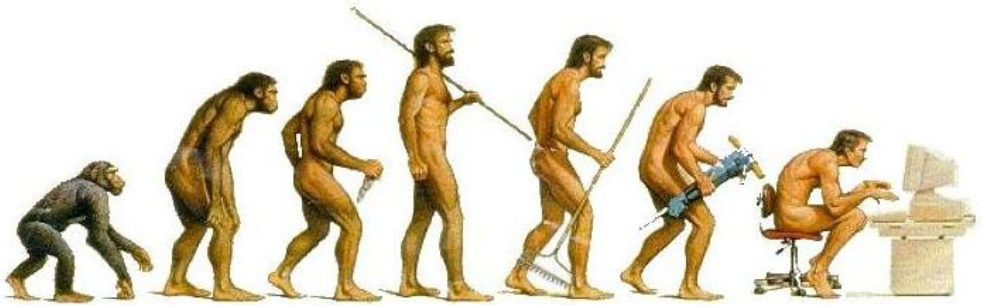
- Now you can style XML as easily as HTML!



Questions ?

CSS 3

TRANSITIONS



Why transitions?

- CSS animating was a little criticized first
  - Style has been static since the beginning!
- Think of transitions as dynamic styling
  - Allows to change CSS properties in a smooth way
- Each time a style is applied to an element, the defined transition occurs

Syntax

- CSS transitions are composed of four properties:
  - **property**: For which property this transition applies
  - **duration**: How many seconds the transition takes
  - **timing-function**: Selects a built-in transition model
  - **delay**: Time to wait before perform transition

```
transition-property: color;
transition-duration: 1s;
transition-timing-function: ease;
transition-delay: 0s;
```

Notes

- **transition-property**:
  - Target multiple properties by separating each by a space
  - Target all properties by using the keyword « all »
- **transition-timing-function**:
  - Eight methods available: linear, ease, ease-in, ease-out, ease-inout, step-start, step-end, custom*
  - Combine them by separating each by a space
  - Live demo showing six of these eight methods:

    http://www.catuhe.com/msdn/transitions/easingfunctions.htm

How to…

- …write a shortcut to CSS transitions:

```
transition: color 1s ease 0s;
```

- …define multiple transitions for each property:

```
transition: color 1s ease 0s, opacity 0.5s linear 2s [,...];
```

- …launch a transition:
  - By pseudoclasses (:hover, …), or by JavaScript

JavaScript launching example

```
<body>
    <style>
        p { color: blue; font-weight: bold;
            cursor: pointer; }
        .dark { color: black;
            transition: color 10s ease 0s; }
    </style>

    <p onclick="this.className = 'dark'">
        Join the dark side!
    </p>
</body>
```

Pros and cons

- Advantages:
  - Lightweight and easy
  - Use hardware acceleration
  - Performs smooth transitions for UI enhancement
- Drawback:
  - Executed one time per action!

- So how to write animations ? (endless transitions)

Keyframes

- @keyframes are rules you can use to create animations:

@keyframes <animation-name> { … }

- Inside it, describe the animation with keywords
  - **from** and **to**
  - **percentages** for each animation step

Keyframes

- Examples:

```
@keyframes my-animation {
    0% { color: red; }
    50% { color: purple; }
    100% { color: blue; }
}

@keyframes other-animation {
    from { font-size: 16px; }
    to { font-size: 30px; }
}
```

Assign keyframes

- CSS « animation » property
  - Specifies which element use which animation

- Most useful derivatives:
  - **name**: The name defined in keyframes
  - **duration**: How many seconds to do the process
  - **iteration-count**: Number of repetitions
    - You can use the keyword « infinite » if you want
  - **timing-function**: Same as transitions

Assign keyframes

- Example:

```
#mySuperElement {
    animation-name: my-animation;
    animation-duration: 1s;
    animation-iteration-count: 3;
}

/* Or simpler... */

.otherElements { animation: other-animation 5s infinite; }
```

Browser Compatibility

- As new CSS attributes, those properties may need to be used with **the vendor prefix**

- Example with *webkit*:

```
@-webkit-keyframes my-animation { /* ... */ }

#myWebkitElement {
    -webkit-transition: all 1s ease-inout 5s;
    -webkit-animation: my-animation 1s 3;
}
```

Questions ?

Exercise (1/2)

- Get the following XML file:

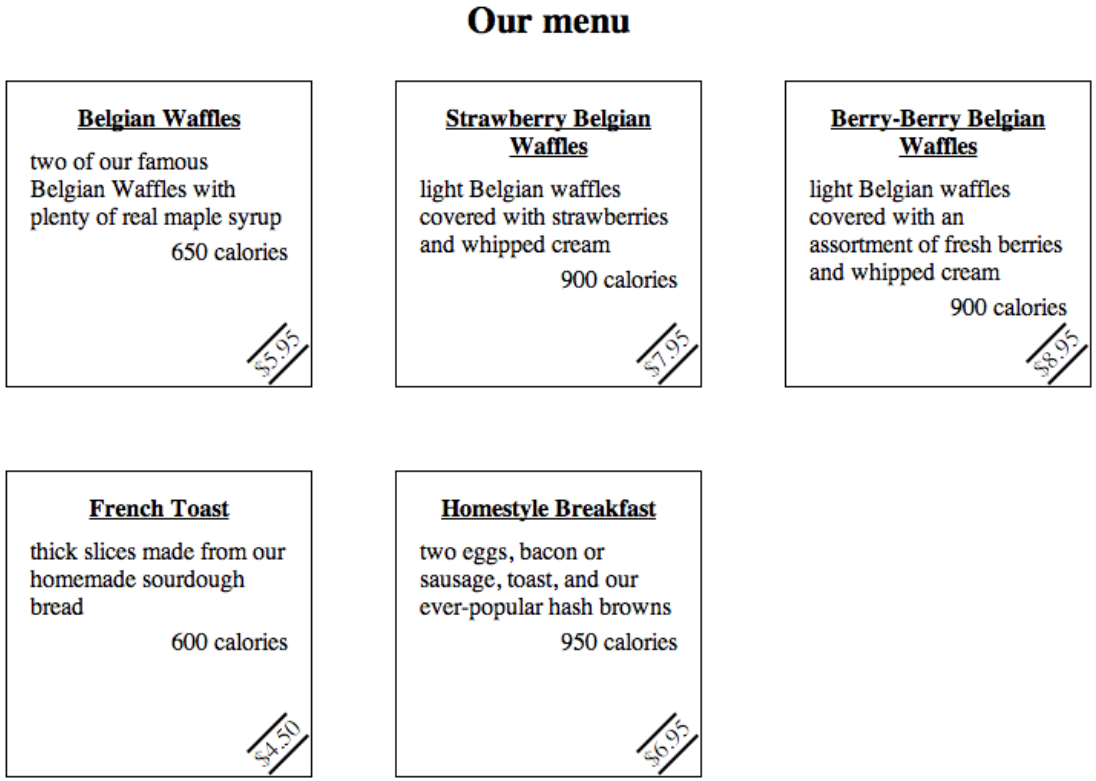
  http://www.w3schools.com/xml/simple.xml

- Enrich it with some logical namespaces

- Style it with CSS and render it as a classical HTML page

Exercise (2/2)

- Possible rendering:



Exercise (3/3)

- Now add some animations in your XML design!

- Make your elements change with at least:
  - One tranformation
  - One transition
  - One keyframe

The end

SUPINFO International University

Sign of Success

Thanks for your attention