C11 - Interactive contens I (Lesson plan)

COURSE 01

- Outline presentation
- Importance of responsive design :
    - Better ranking (SEO)
    - Less time to produce/update

Responsive design (revision)

- **CSS initial scale (for mobile)**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

- **media query (CSS)**

```
@media only screen and (max-width: 600px) {
        body {
        background-color: lightblue;
        }
}
```

- **For mobile :**

```
@media only screen and (max-width: 768px) {
        [class*="col-"] {
        width: 100%;
        }
}
```

Common viewports

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

Exercise 1a : try media queries

Viewport's orientation

```
@media only screen and (orientation: landscape) {
        body {
        background-color: lightblue;
        }
}
```

Hide/show content based on viewport (toggle)

```
/* If the screen size is 600px wide or less, hide the element */
@media only screen and (max-width: 600px) {
        div.example {
                display: none;
        }
}
```

* Can be used for different elements. Ex. : font-size based on viewport

Exercise 1b : Toggle content using media query

Responsive image

```
img {
        width: 100%;
        height: auto;
}
```

- Image adapts to container's width
- Display: block, sometimes useful
- ALSO : background-image / contain / no-repeat
- max-width: 100%; (or absolute value to avoid pixelisation)

Responsive background

```
/* For width smaller than 400px: */
body {
        background-image: url('img_smallflower.jpg');
}

/* For width 400px and larger: */
@media only screen and (min-width: 400px) {
body {
        background-image: url('img_flowers.jpg');
        }
}
```

min-device-width INSTEAD OF min-width:

```
/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
        body {
                background-image: url('img_flowers.jpg');
        }
}
```

Create boxes

- Responsive container
- Responsive image content

Exercise 1c: Produce several responsive containers containing responsive images of different sizes

Positioning the boxes (Flex)

- Use max-width / min-width for the items + flex-wrap:wrap
- Items height mandatory to avoid stretching
- Rows height determined by the highest item's content

Positioning the boxes (Grid)

- Create grid + vary number of columns to view the results

```
.container {
        display: grid;
        grid-template-columns: repeat(3,1fr);
        grid-gap: 10px;
        grid-auto-rows: auto;
}
```

Positioning the boxes (columns layout) - Masonry

```
.container {
        -webkit-column-count: 3; /* Chrome, Safari, Opera */
        -moz-column-count: 3; /* Firefox */
        column-count: 3;
}
```

Possible properties:

```
column-count
column-gap
column-rule-style
column-rule-width
column-rule-color
column-rule
column-span
column-width
```

Assignment 01: Create a masonry page (eStore type)

COURSE 02

Includes

- What are inludes / Usually server-side (PHP) / Can be done client-side using JavaScript

- **Example : include text** *SEE : JAVASCRIPT_INCLUDE_1*

  HTML :
  ```
  <script src="text.js"></script>
  ```

  JS :

  ```
  document.write("This is text content");
  ```
  OR
  ```
  document.getElementById("nav").innerHTML = "Here is text content included using JavaScript.";
  ```

Exercise 2a : include text

- **Example header 1** *SEE : JAVASCRIPT_INCLUDE_2*

  HTML :
  ```
  <script src="header.js"></script>
  ```

  JS :

  ```
  document.write('<header>'
                 +'<h1>HEADER EXTERNAL</h1>'
                 +'<nav>'
                 +'<a href="#">Link 1</a>'
                 +'<a href="#">Link 2</a>'
                 +'</nav>'
                 +'</header>'
  );
  ```

- BUT there are some problems with using *document.write*

- **Example header 2**
  SEE : JAVASCRIPT_INCLUDE_3

  HTML :
  ```
  <script src="header.js"></script>
  ```

  JS :

  ```
  document.getElementById("nav").innerHTML =
  '<a href="#">Home</a>';

  document.getElementById("nav").innerHTML +=
  '<a href="#">Products</a>';

  document.getElementById("nav").innerHTML +=
  '<a href="#">Services</a>';
  ```

Exercise 2b : include header

Document object model (DOM)

SEE : C11-PRESENTATION-02

- DOM = independent form programing/coding languages
  Defines all elements as objects

- Window / Location / History / Document ... (Top elements of the hierarchy)

- CSS = Selectors define the document's object we want to change
  JS = getElementById + getElementByTagName etc. identify the object we address to

Show/hide content using :target (CSS)

- Content may be hidden based on viewport's size

- Content can be hidden or shown on events / states (:hover / :target)

HTML :

```
<a href="#toggle">Show</a>
<div id="toggle">
<a href="#" class="hide">Hide</a>
Content to show and hide.
</div>
```

CSS :

```
#toggle {
        display: none;
}

#toggle:target {
        display: block;
}

#toggle:target .hide {
        display: block;
}

.hide {
        position: absolute;
        right: 10px;
        top: 00px;
        display: none;
}
```

- BUT : Too many content elements to show/hide requires a lot of coding

  JAVASCRIPT very useful then

Assignment 02 : Create 3 titles showing hidden content using :target

COURSE 03

CSS drop-down menu

SEE 04-DROP_DOWN_MENU-CSS

HTML :

```
<header>
        <nav>
                <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">Services</a>

                        <!-- second level links -->
                        <ul>
                        <li><a href="#">Domestic</a></li>
                        <li><a href="#">Industrial</a></li>
                        <li><a href="#">Businesses</a></li>
                        </ul>

                </li>
                <li><a href="#">Portfolio</a></li>
                <li><a href="#">About us</a></li>
                <li><a href="#">Contact</a></li>
                </ul>
        </nav>
<header>
```

CSS :

```
/* First level menu */

header {
        width: 800px;
        margin: auto;
}

header nav ul {
        display: flex;
        justify-content: flex-end;
        border: solid 1px #c0c0c0;
}

header nav ul li {
        list-style-type: none;
        position: relative;
}


header nav ul li a {
        display: inline-block;
        text-decoration: none;
        color: black;
        padding: 0.5em 1em;
        background: white;
}
```

```
header nav ul li a:hover {
        background: #c0c0c0;
}

/* Second level menu */

header nav ul li ul {
        position: absolute;
        display: none;
        padding-left: 0;
}

header nav ul li:hover ul {
        display: block;
}

header nav ul li ul li {
        border-left: 1px solid #c0c0c0;
        border-right: 1px solid #c0c0c0;
}

header nav ul li ul li:last-of-type {
        border-bottom: 1px solid #c0c0c0;
}

header nav ul li ul li a {
        display: inline-block;
        width: 100%;
        box-sizing: border-box;
}
```

Exercise 2d : Create a drop down menu

Transitions css :hover

- Make it possible to gradually go from one state to another on a certain time laps

HTML :

```
<div>My container</div>
```

CSS :

```
div {
        background-color: lightblue;
        width: 30%;
        margin: auto;
        Transition: width 2s;
}

div:hover {
        width: 80%;
}
```

Transitions css :hover *SEE 05-TRANSITIONS*

HTML :

```
<div class="container">
        <h2>Container stretching using :hover</h2>
</div>
```

CSS :

```
div {
        background-color: lightyellow;
        height: 0px;
        transition: height 2s, background-color 2s;
}

div:hover {
        height: 500px;
        background-color: yellow;
}

h2 {
        background-color: white;
}
```

Exercise 2e : test transitions (size, colours, etc.)

Transitions css :target *SEE 05-TRANSITIONS*

HTML :

```
<a href="#stretch">Stretch container now!</a>
<div id="stretch">
        <h2>Container stretching using :target</h2>
</div>
```

CSS :

```
#stretch {
        background-color: lightyellow;
        height: 0px;
        transition: height 2s,background-color 2s;
}

#stretch:target {
        height: 500px;
        background-color: yellow;
}

h2 {
        background-color: white;
}
```

Exercise 2f : Test transitions using :target

Menu transitions stretch box with :hover *SEE 06-STRETCH_BOXES_CSS*

HTML :

```
<section>
		<article>
				<nav><h2>BOX</h2></nav>
				<img src="bart.jpg" />
		</article>

		<article>
				<nav><h2>BOX</h2></nav>
				<img src="bart2.jpg" />
		</article>

		<article>
				<nav><h2>BOX</h2></nav>
				<img src="bart3.jpg" />
		</article>
</section>
```

CSS :

```
section {
		display: grid;
		grid-template-columns: 1fr 1fr 1fr;
		grid-gap: 20px;
		padding: 20px;
}

article {
		position: relative;
		height: 90px;
		overflow: hidden;
		transition: height 1s;
}

nav {
		padding: 10px;
		border: solid 1px black;
		background-color: white;
}

article:hover {
		height: 100%;
}

section article img {
		width: 100%;
}
```

CSS burger menu with transition using :target *SEE 07-MENU_BURGER_CSS*

HTML :

```
<header>
    <nav>
        <a href="#show"><img src="burger.png"></a>
    </nav>

    <div class="menu" id="show">
        <a href=""#>Info</a>
        <a href=""#>Produics</a>
        <a href=""#>Services</a>
        <a href=""#>Contacts</a>
    </div>
</header>
```

CSS :

```
* {box-sizing: border-box;}

body {
    width: 100%;
    height: 100%;
    padding: 0px;
    margin: 0px;
}

header {
    position: relative;
    width: 95%;
    max-width: 800px;
    height: 100px;
    margin: auto;
    padding-top: 10px;
    background-color: white;
}

nav {
    height: 100%;
    border: solid 1px #c0c0c0;
    text-align: right;
}

nav img {
    height: 80%;
}

.menu {
    position: absolute;
    display: flex;
    flex-direction: column;
    right: 0px;
    bottom: 0px;
    width: 100%;
    height: 300px;
    background-color: #c0c0c0;
    z-index: -10;
    transition: bottom 1s;
}
```

```
#show:target {
		bottom: -300px;
}

.menu a {
		display: flex;
		justify-content: center;
		align-items: center;
		width: 100%;
		height: 100%;
		background: lightblue;
		border-style: solid;
		border-color: black;
		border-width: 0px 1px 1px 1px;
		text-decoration: none;
		color: black;
		font-family: sans-serif;
		font-weight: bold;
}

.menu a:hover {
		background: lightyellow;
}
```

Assignment 03 : Home page with responsive menu

- Create a home page using an image as backgound and a fixed header.
- The page will have a breaking point at 768px.
- The large version will show a logo on the left and a 5 links navigation on the right.
- The narrow version will replave the navigation with the burger icon.
- Clicking on the burger icon will cover the page with an 50% opacity overlay containing equally sized and spread links/buttons.

COURSE 04

Revision : transform using :hover *SEE 08-TRANSFORM*

HTML :

```
section {
        position: relative;
        top: 100px;
        width: 200px;
        height: 200px;
        margin: auto;
        background-color: pink;
}

section:hover {
        width: 200px;
        height: 200px;
        transform-origin: top left;
        transform: rotate(-15deg) skew(20deg) scale(1.5);
        box-shadow: 5px 5px 10px grey;
        /*      transform: translateX(30px, 50px); */
}
```

Animated nested container using :hover *SEE 09-ANIMATED-CAPTION*

HTML :

```
<figure><figcaption>Content of figcaption</figcaption></figure>
```

CSS :

```
figure {
        position: relative;
        top: 20px;
        width: 400px;
        height: 500px;
        margin: auto;
        background-color: lightblue;
        overflow: hidden;
}

figcaption {
        position: absolute;
        width: 100%;
        bottom: -50px;
        background-color: lightyellow;
        height: 50px;
        transition: bottom .6s;
}

figure:hover figcaption {
        bottom: 0px;
}
```

Animated nested container using :target *SEE 10-ANIMATED-CONTAINER-TARGET*

HTML:

```
<section>
		<nav>
				<a href="#one">One</a>
		</nav>

		<figure id="one">1</figure>
</section>
```

CSS:

```
section {
		position: relative;
		width: 900px;
		height: 100%;
		margin: auto;
		background-color: lightblue;
		overflow: hidden;
}

nav {
		background-color: lightpink;
		padding: 10px;
		text-align: center;
}

figure {
		display: flex;
		justify-content: center;
		align-items: center;
		position: relative;
		top: 20px;
		width: 300px;
		height: 300px;
		background-color: lightyellow;
		font-size: 40px;
		left:-340px;
		transition: all 1s ease;
}

figure:target {
		left: 0px;
}
```

Alternatively animated nested containers using :target

SEE 11-CONTAINER-ANIMATED-ALTERNATE-TARGET

HTML :

```
<section>
        <nav>
                <a href="#one">One</a>
                <a href="#two">Two</a>
        </nav>

        <figure id="one">1</figure>
        <figure id="two">1</figure>
</section>
```

CSS :

```
section {
        position: relative;
        width: 900px;
        height: 100%;
        margin: auto;
        background-color: lightblue;
        overflow: hidden;
}

nav {
        background-color: lightpink;
        padding: 10px;
        text-align: center;
}

figure {
        display: flex;
        justify-content: center;
        align-items: center;
        position: relative;
        top: 20px;
        width: 300px;
        height: 300px;
        background-color: lightyellow;
        font-size: 40px;
        left:-340px;
        transition: all 1s ease;
}

figure:target {
        left: 0px;
}
```

Assignment 04 : SLide-in animated containers using : target

- Create 3 links that will trigger 3 content's containers showing alternatively in a main container.

  SEE 12-EXAMPLE-ASSIGNMENT-04

COURSE 05

CSS basic animation principles

- Animation principles :
  - Initial keyframe : from()
  - Final keyframe : to()

HTML :

```
<div class="anim1"> </div>
```

CSS :

```
.animation {
        width: 100px;
        height: 100px;
        background-color: yellow;
        animation-name: fade;
        animation-duration: 2s;
        animation-iteration-count: infinite;        /* OR number */
}

@keyframes fade {
        from {background-color: yellow;}            /* OR 0% {all properties} */
        to {background-color: red;}                 /* OR 100% {all properties} */
}
```

TO USE INTERMEDIATE KEYFRAMES :

```
@keyframes fade {
        0% {background-color: yellow;}
        50% {background-color: green;}
        100%{background-color: red;}
}
```

OTHER ANIMATION PROPERTIES :

```
animation-timing-function
linear / ease / ease-in / ease-out / ease-in-out / step-start / step-end / steps(int / start / end) / cubic-bezier(n,n,n,n)

animation-delay                 /* Time using s OR ms */

animation-iteration-count       /* number / infinite */

animation-direction
normal / reverse / alternate / alternate-reverse

animation-fill-mode
none / forwards / backwards / both

animation-play-state
paused / running
```

Various examples of CSS animations

SEE 13-ANIMATIONS-VARIA

- Explain example

SEE 14-ANIMATION-TURN_AROUND

- Explain example

CSS animations using :hover and :target

SEE 16-ANIMATION_PHONE-OVER

- Explain example

CSS steps animations Animation using :hover and :target

SEE 16-ANIMATION-01-OVER

- Explain example

Page building itself up with slide-in elements

SEE 17-PAGE-BUILT-SLIDE-IN

- Explain example

Controling velocity using cubic-bezier

SEE 18-ANIMATION-CUBIC-BEZIER

- animation-timing-function: cubic-bezier(.1,0,.5);
- Explain example

Assignment 05: Create a self building page using slide-in elements with delays

COURSE 06

Revision

Workshop

COURSE 07

Midterm exam

COURSE 08

<picture>

```
<picture>
        <img src="grumpy_black.jpg" />
</picture>
```

Image use based on viewport

```
<picture>
        <source media="(min-width: 1200px)" srcset="grumpy_big.jpg">
        <source media="(max-width: 800px)" srcset="grumpy.jpg">
        <img src="grumpy_black.jpg" />
</picture>
```

- img tag = make sure an image shows if <picture> is not supported

<audio>

```
<audio controls autoplay loop muted>
        <source src="laugh.wav" type="audio/wav">
        Your browser does not support the audio tag.
</audio>
```

OR

```
<audio controls autoplay loop muted src=" laugh.wav" type="audio/wav">
        Your browser does not support the audio tag.
</audio>
```

- Autoplay doesn't work anymore (possible, but usage forbidden / bad practice)

Audio using :hover

SEE 21-AUDIO-HOVER

```
<div id="hoverElement">Hover this text</div>

<audio id="audio" style="visibility:hidden;">
		<source src="doorbell.mp3" type="audio/mpeg; codecs=MPEG-3">
</audio>

<script>
let hoverArea = document.getElementById('hoverElement');
let audio = document.getElementById('audio');

hoverArea.onmouseover= function(){
		audio.play();
}

hoverArea.onmouseout= function(){
		audio.pause();
}
</script>
```

<video>

```
<video width="100%" controls loop autoplay poster="grumpy_big.jpg">
		<source src="movie.mp4" type="video/mp4">
		Your browser does not support the video tag.

</video>
```

Video using :hover *SEE 22-VIDEO-HOVER*

```
<video loop onmouseover="play();" onmouseout="pause();">
		<source src="movie.mp4" type="video/mp4">
		Your browser does not support the video tag.
</video>
```

- Video background : container fixed with negative z-index

Video background *SEE 23-BGVIDEO*

```
<video class="bg" src="movie.mp4" autoplay loop> </video>

.bg {
		position: fixed;
		top: 0px;
		width: 100%;
		height: 100%;
		object-fit: cover;
		object-position: top right;
}
```

Preload

- To preload heavy element so they are available when needed

- **Preload CSS** :

  Position content with absolute = left: -9999px z-index négatif

- **Preload property and values :**
  <audio preload="auto"> </audio>
  <video preload="auto"> </video>

  - auto : file is preloaded as soon as the page has charged.
  - metadata : only metadata are preloaded when the page has charged.
  - none : file is downloaded on demand.

Preload with <link>

- Prelaod can be done using <link> to prioritize specific elements

  EXAMPLES :

  ```
  <link rel="preload" href="style.css" as="style">
  <link rel="preload" href="main.js" as="script">
  <link rel="preload" href="exemple.mp4" as="video" type="video/mp4">
  <link rel="preload" href="exemple.ttf" as="font" type="font/ttf" crossorigin="anonymous">
  <link rel="preload" href="exemple.png" as="image" media="(max-width: 600px)">
  ```

Possible preload elements (partial)

- audio : audio file.
- document : HTML within a <frame> or <iframe>.
- embed : external file integrated into a <embed> tag.
- font : Typeface.
- image : image file.
- object : external file integrated into a <object> tag.
- script : JavaScript file.
- style : stylesheet.
- track : WebVTT file.
- video : video file.

Assignment 06: Produce a multimedia interactive web page

SEE 24-EXAMPLE-ASSIGNMENT-06

- Video background

- Contents max-width : 960px;

- Fixed header with a 3 links navigation using equal width

- A heading mentionning your name

COURSE 09

SEE 25-VARIA-JS

Progression bars <meter>

```
<meter value="2" min="0" max="10">2 of 10</meter> (HTML5)
OR
<meter value="0.6">60%</meter>
OR
<progress value="22" max="100"> </progress>
```

Image-map

- Active zones cliquables in an image (clickable areas)
- Difficult to define = use software or site
  ex.: https://www.image-map.net/

```
<img src="colorofstate.jpg" usemap="#image-map">

<map name="image-map">
<area target="_blank" alt="My page" title="My page" href="mypage.html" coords="492,146,99" shape="circle">
<area target="_blank" alt="My page" title="My page" href="mypage.html" coords="0,1411,999,1223" shape="rect">
</map>
```

<object>

- Allow integration of various contents : audio, video, Java applets, pdf, activeX, flash...

```
<object width="400" height="400" data="myFlashMovie.swf">You browser doesn't support object tag</object>
```

JavaScript to interact with DOM

CHANGING HEADER'S TEXT (getElementById)

HTML :

```
<header>
        <h1 id="headerText">HEADER</h1>
</header>

<a href="#" onclick="queryHeaderText();">Change header's text</a></li>
```

JS :

```
<script>
function queryHeaderText(){
        let header = prompt("Changez le texte HEADER pour :");
        document.getElementById("headerText").innerHTML = header;
}
</script>
```

CGANING HEADER'S BACKGROUND COLOR (getElementsByTagName)

HTML :

```
<header>
        <h1 id="headerText">HEADER</h1>
</header>

<a href="#" onclick="queryHeaderColor();">Change the header's background color for red</a>
```

CSS :

```
header {
        background-color: black;
}
```

JS :

```
<script>
function queryHeaderColor(){
        document.getElementsByTagName("header")[0].style.backgroundColor = "red";
}
</script>
```

Keyword « this »

```
<header onclick="this.style.backgroundColor= 'red';" ondblclick="this.style.backgroundColor= 'black';">
```

Modifying attributes

```
myVariable.src = "http://www.domaine.com/photo.jpg";
OR
myVariable.setAttribute("src", "http://www.domaine.com/photo.jpg");
```

Modifying element's class

```
myVariable.className = "myClass";
```

Example :

```
x.title = "bigTitle";
```

Add a supplentary class to an element

```
myVariable.className += "newClassToAdd";
```

Modifying the content of a tag (text / html)

```
myVariable.innerHTML = "Hello <strong>the world!</strong>";
```

OR (without html)

```
myVariable.textContent = "Hello the world!";
```

Image-map using JavaScript

HTML :

```
<figure>
        <figcaption id="bubble"> </figcaption>
        <img class="mapped" src="biker.jpg" usemap="#image-map" />
        <map name="image-map">
                <area target="" alt="Courageous biker" title="Courageous biker" href="#" coords="289,53,332,113,354,209,321,367,293,453,230,399,223,239,182,149" shape="poly" onmouseover="bubbleOn()" onmouseout="bubbleOff()" />
        </map>
</figure>
```

JS :

```
function bubbleOn(){
        document.getElementById("bubble").style.display = "block";
        let x = document.getElementsByTagName("area")[0].getAttribute("alt");
        document.getElementById("bubble").innerHTML = x;
}


function bubbleOff(){
        document.getElementById("bubble").style.display = "none";
}
```

CSS :

```
figure {
        position: relative;
        height: auto;
        padding: 0px;
        margin: 0px;
}

figcaption {
        display: none;
        position: absolute;
        top: 50px;
        left: 330px;
        width: auto;
        height: auto;
        background: yellow;
        border: solid 3px white;
        border-radius: 30px 30px 30px 0px;
        padding: 20px;
        font-weight: bold;
}

figure img {
        position absolute;
        display: block;
        left: 0px;
}
```

JavaScript transitions SEE *26-TRANSITION-JS_BOX_IN*

HTML :

```
<div id="container" onclick="stretch()">
	<h2>Container stretching using :hover</h2>
</div>
```

CSS :

```
#container {
	background-color: lightyellow;
	height: 0px;
	transition: height 2s,background-color 2s;
}

h2 {
	background-color: white;
}
```

JavaScript :

```
function stretch(){
	document.getElementById("container").style.height = "500px";
	document.getElementById("container").style.backgroundColor = "yellow";
}
```

JavaScript transitions SEE *27-PAGES-SLIDE-IN-JS*

Hide/show content (JavaScript/CSS)

HTML :

```
<h2 class="trigger">Toggle content 1</h2>
<div style="display:none">
        <p>Content 1<p>
</div>

<h2 class="trigger">Toggle content 2</h2>
<div style="display:none">
        <p>Content 2<p>
</div>
```

CSS :

```
.trigger {
        cursor:pointer;
}

.trigger:hover {
        color:red;
}
```

JavaScript :

```
function toggle(event) {
        if (event.target && event.target.className == 'trigger') {
                let next = event.target.nextElementSibling;

                if (next.style.display == "none") {
                        next.style.display = "block";
                } else {
                next.style.display = "none";
                }
        }
}

document.addEventListener('click', toggle, true);
```

Assignment 07: Produce an interactive web page using the topics covered during this class

COURSE 10

CSS : mask-image

To mask using a linear gradient *SEE 28-CSS-MASK-IMAGE*

```
.myMask {
        -webkit-mask-image: linear-gradient(to bottom, transparent 30%, black 85%);     /* to top, to right...*/
        mask-image: linear-gradient(to bottom, transparent 30%, black 85%);
}

<section>
        <img class="myMask" src="montreal.jpg">
</section>
```

To mask using a radial gradient

```
.myMask {
        -webkit-mask-image: radial-gradient(circle at 50% 50%, black 20%, transparent 45%);
        mask-image: radial-gradient(circle at 50% 50%, black 20%, transparent 45%);
}

<section>
        <img class="myMask" src="montreal.jpg">
</section>
```

To mask using an image *SEE 29-CSS-MASK-IMAGE*

```
.myMask {
        -webkit-mask-image: url("image-mask.png");
        mask-image: url("image-mask.png");
        -webkit-mask-size: 100% 100%;
        mask-size: 100% 100%;
}

<section>
        <img class="myMask" src="montreal.jpg">
</section>
```

To mask using an anim-gif *SEE 29-CSS-MASK-IMAGE*

```
.myMask {
        -webkit-mask-image: url("image-mask-anim.gif");
        mask-image: url("image-mask-anim.gif");
        -webkit-mask-size: 100% 100%;
        mask-size: 100% 100%;
}

<section>
        <img class="myMask" src="montreal.jpg">
</section>
```

CSS : clip-path

SEE 30-CSS-MASKS

- clip-path = Selects a visible zone

```
.trapezeMask {
        -webkit-clip-path:  circle(35% at 50% 50%);
        clip-path:  circle(35% at 50% 50%);
}

<section>
        <img class="trapezeMask" src="montreal.jpg">
</section>
```

CSS : clip-path - animation

SEE 31-CSS-MASKS-ANIMATION

```
.trapezeMask {
        -webkit-clip-path: polygon(10% 10%, 90% 10%, 90% 90%, 10% 90%);
        clip-path: polygon(10% 10%, 90% 10%, 90% 90%, 10% 90%);
}

section img.animate {
        animation: myAnim 2s infinite;
        animation-direction: alternate;
        animation-play-state: paused;
}

section:hover img.animate {
        animation-play-state: running;
}

@keyframes myAnim {
        0% {
                -webkit-clip-path: polygon(10% 10%, 90% 10%, 90% 90%, 10% 90%);
                clip-path: polygon(10% 10%, 90% 10%, 90% 90%, 10% 90%);
        }

        100% {
                -webkit-clip-path: polygon(50% 10%, 50% 10%, 90% 90%, 10% 90%);
                clip-path: polygon(50% 10%, 50% 10%, 90% 90%, 10% 90%);
        }
}

<section>
        <img class="trapezeMask animate" src="montreal.jpg">
</section>
```

Path amking tool:

http://bennettfeely.com/clippy/

COURSE 11

Drawuing using <canvas>

SEE 32-CANVAS-FORMES

- Allows to draw and animate + interactivity (key clicks, mouse clicks, button clicks, finger movement...)
- Used along with JavaScript (id mandatory)

```
<canvas id="myCanvas" width="500" height="300" style="border:1px solid #000000;"></canvas>
```

To draw a line :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.moveTo(20, 20);                    /* Start coordinates */
        content.lineTo(500, 120);                  /* End coordinates */
        content.lineWidth = 10;                    /* Line width */
        content.strokeStyle = "blue";              /* Line color */
        content.stroke();                          /* Creates the line */
        content.lineCap = "round";                 /* Line endings : bevel, round, miter */
</script>
```

Drawing a line with several anchor points :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.moveTo(20, 20);
        content.lineTo(500, 100);
        content.lineTo(20, 150);
        content.lineWidth = 10;
        content.strokeStyle = "green";
        content.stroke();
</script>
```

Drawing a filled rectangle :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.fillStyle = "red";
        content.shadowBlur = 10;                   /* Creates a shadow */
        content.shadowOffsetX = 5;
        content.shadowOffsetY = 5;
        content.fillRect(10, 15, 500, 150);        /* x, y, width, height */
</script>
```

Drawing a stroked rectangle :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.strokeStyle = "black";
        content.lineWidth = 20;
        content.lineJoin = "round";                 /* Corners shape = bevel, round, miter */
        content.strokeRect(25, 25, 400, 150);

</script>
```

Drawing a filled and stroked circle :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.beginPath();                        /* Starts the path */
        content.arc(150, 150, 90, 0, 2 * Math.PI);  /* Creates the circle : x, y, r, start angle, end angle, formula */
        content.lineWidth = 10;
        content.fillStyle = 'pink';                 /* here, fill under stroke */
        content.fill();
        content.strokeStyle = "orange";
        content.stroke();
</script>
```

Making a linear gradient :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

let grd = content.createLinearGradient(50, 0, 400, 0);      /* Create linear gradient */
grd.addColorStop(0, "red");                                  /* Start color */
grd.addColorStop(1, "yellow");                               /* End color */

        content.fillStyle = grd;
        content.fillRect(0, 0, 500, 500);                    /* start x, start y, end x, end y */

</script>
```

Making a radial gradient :

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        let grd = content.createRadialGradient(100, 100, 10, 100, 100, 70);     /* x0, y0, r, x1, y1, r1 */
        grd.addColorStop(0, "red");
        grd.addColorStop(1, "white");

        content.fillStyle = grd;
        content.fillRect(10, 10, 160, 160);
</script>
```

Using text:

```
<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

        content.font = '30px Arial';                    /* Font size, typeface */
        content.fillStyle = 'red';
        content.fillText('Example of text', 50, 75);           /* text, x, y */

</script>
```

- ALSO = strokeText("Hello World", 10, 50);
- ALSO = fillStyle = "red";
- ALSO = textAlign = "center";

Using an image:

```
<canvas id="myCanvas" width="600" height="600"> </canvas>

<script>
let canvas = document.getElementById("myCanvas");
let content = canvas.getContext("2d");

let img = new Image();

img.onload = function(){
        content.drawImage(img, 0, 0, 400, 400);
}

img.src = 'bart2.jpg';
</script>
```

SVG

SEE 33-SVG

- SVG = Scalable Vector Graphics (XML)
- Can be animated, printed and scaled lossless
- Can be created with text editors
  IDEAL : Illustration software : Illustrator / Inskscape

```
<svg width="100" height="100">                                                                    /* SVG container */
        <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />          /* SVG shape */
</svg>
```

- Container can be smaller than the shape OR mask it

- x, y (coordinates) +rounded corners can be used :

```
<rect x="10" y="10" width="100" height="100" rx="20" ry="20" stroke="black" stroke-width="2" fill="pink" />
```

- CSS :

```
fill: rgb(0,0,255);
stroke-width: 3;
stroke: rgb(0,0,0);

fill-opacity: 0.1;
stroke-opacity: 0.9
```

Other shapes

```
<rect width="195" height="195" stroke="black" stroke-width="2" fill="green" />

<ellipse cx="95" cy="95" rx="80" ry="50" stroke="black" stroke-width="2" fill="yellow" />

<line x1="10" y1="0" x2="200" y2="200" stroke="red" stroke-width="4" />

<polyline points="10,10 50,75 125,85 195,195" stroke="red" stroke-width="4" fill="none" />
/* anchor points coordinates : x,y x,y x,y */

<polygon points="100,20 50,190 250,210" stroke="green" stroke-width="4" fill="pink" />
/* anchor points coordinates : x,y x,y x,y */

<polygon points="100,10 40,198 190,78 10,78 160,198" stroke="green" stroke-width="4" fill="pink" fill-rule="nonzero" />
/* ALSO : fill-rule="evenodd" */
```

Stroke types

```
stroke-width="4"
stroke-linecap="butt"            /* butt, round, square
stroke-dasharray="5,5"           /* OR : 6,10,3 */
```

SVG can contain multiple shapes:

```
<svg width="200" height="200" style="border:dashed 1px red;">

        <ellipse cx="100" cy="100" rx="80" ry="30" fill="blue" />

        <ellipse cx="100" cy="70" rx="50" ry="20" fill="pink" />

        <ellipse cx="105" cy="45" rx="40" ry="15" fill="lightblue" />

</svg>
```

SVG can pile-up:

```
<svg width="250" height="250" style="border:dashed 1px red;">
        <ellipse cx="120" cy="100" rx="110" ry="50" fill="pink" />
        <ellipse cx="110" cy="100" rx="90" ry="30" fill="lightblue" />
</svg>
```

Paths

SEE 34-SVG-PATH

Available commands

- d=" " : Attribute defining a sequence of movements
- UPPERCASE commands = based on origine points of the container (0,0)
  LOWERCASE commands = based on actual current position

M = moveto
Usually indicates starting position followed with the coordinates.
Exemple : M 10,15

L = lineto
Signifies t trace a line to a certain point (coordinates).
H and V : horizontal and vertical lineto

C = curveto
Bezier curve (d=" Q cx,cy x,y")
ALSO : S (smooth curveto), M (quadratic Bézier curve) et T (smooth quadratic Bézier curveto)

A = elliptical Arc
More complex : A rx,ry xAxisRotate LargeArcFlag,SweepFlag
Not treated in this class

Z = closepath
Placed at the end to close an open path from the ending to the starting point.

Filled shape

```
<svg width="250" height="250" style="border:dashed 1px red;">
        <path d="M100 10 L20 200 L225 220 Z" fill="orange" />
</svg>
```

Line

```
<svg width="250" height="250" style="border:dashed 1px red;">
        <path d="M 50,20 L 150,180" stroke="blue" stroke-width="3" fill="none" />
</svg>
```

Stroke

```
<svg width="250" height="250" style="border:dashed 1px red;">
        <g fill="none">
                <path stroke="red" stroke-width="4" d="M 10,20 L 230,20" />
                <path stroke="black" stroke-width="8" d="M10,50 L 230,80" />
                <path stroke="blue" stroke-width="12" d="M10,80 L 230,150" />
        </g>
</svg>
```

Curve

```
<svg width="250" height="250" style="border:dashed 1px red;">
    <path d="M 10,190 Q 100,20 220,190" stroke="red" stroke-width="5" fill="none" />
</svg>
```

Text

```
<svg width="250" height="250" style="border:dashed 1px red;">
    <text x="0" y="120" fill="black" transform="rotate(-20 0, 45)">Wow! That's a great text</text>
</svg>
```

Multi-lines text

```
<svg width="250" height="250" style="border:dashed 1px red;">
    <text x="20" y="30" fill="black">Several lines:
        <tspan x="20" y="55">First line</tspan>
        <tspan x="20" y="80">Second line</tspan>
    </text>
</svg>
```

Hyperlink

```
<svg width="250" height="250" style="border:dashed 1px red;">
    <a xlink:href="http://www.collegecdi.ca" target="_blank">
        <text x="20" y="35" fill="red">Click here!</text>
    </a>
</svg>
```

SVG transitions / animations

SEE 35-SVG-TRANSITION-CSSN (A & B)

Responsive SVG

```
svg {
    viewBox: (0% 0%, 100% 100%, 0% 100%, 100% 100%);
    transform: scale(1);
}
```

SEE 36-SVG-TRANSITION

SEE 37-SVG-ANIMATION

Assignment 08: SVG animations and transitions

COURSE 12

SVG animation (SMIL)

SEE 38-SVG-ANIMATION-SMIL

- SVG can be animated using <animate>
  initially defined by SMIL animation specifications (Synchronized Multimedia Integration Language).

- Some animation are impossible to make using CSS = JavaScript is then used

Available commands

<animate>
To animate SVG's attributes and properties.

<set>
Shorthand of *animate* tu use non-numerical value in certain attributes (example : visibility OR display).

<animateMotion>
To animate an element along a path.

<animateColor>
To animate the modification of a color (deprecated : removed from SVG2).
Better use *animate*.

<animateTransform>
Like the CSS transform attribute, animates transformation attributes.

path (attribute)
To pecify a path's attributes data.

<mpath>
Used with *animateMotion* in conjunction with a path, *mpath* is placed in an *animateMotion* before the closing tag.

keypoints (attribute)
Used to precisely control an animation's speed when using a movement path.

rotate (attribute)
animateMotion's attribute to controle the orientation of an element on a path (x axis).

<animate>

```
<rect>
        <animate attributeType="CSS" attributeName="opacity" from="1" to="0" dur="2s" repeatCount="indefinite" />
</rect>
```

OU

```
<rect id="rectangle" ... />
<animate xlink:href="#rectangle"  attributeName="opacity" from="1" to="0" dur="2s" repeatCount="indefinite" />
```

- dur = equivalent of *animation-duration* in CSS
- repeatCount = equivalent of *animation-iteration-count* in CSS
- fill = equivalent of *animation-fill-mode* in CSS
  freeze = state of the last image (equivalent of *forwards*)
  remove = state of the first image (equivalent of *backwards*)
- restart = Restarts the animation.
  always = default. Restarts the animation.
  whenNotActive = Restarts the animation only if it's inactive.
  never = Keep animation from being restarted.

attributeName

To target the attribute to animate.

Example :
To animate the center of a circle, cx would be used

In CSS animations, some attributes are CSS, others XML only.

CSS attrubutes :
font, font-family, font-size, font-size-adjust, font-stretch, font-style, font-variant, font-weight, direction, letter-spacing, text-decoration, unincode-bidi, word-spacing, visibility, text-rendering, writing-mode, clip-path, mask-opacity, filter, pointer-events, image-rendering, clip, color, cursor, display, overflow

XML attributes :
clip-rule, flood-color, flood-opacity, stop-opacity, kerning, tech-anchor, color-profile, color-rendering, fill, fill-opacity, fill-rule, marker, marker-end, marker-mid, marker-start, stroke, stroke-width, stop-color, lighting-color, enable-background, dominant-baseline, color-interpolation-filters, color-interpolation, glyph-orientation-horizontal, glyph-orientation-vertical, shape-rendering, baseline-shift, alignment-baseline, stroke-miterlimit, stroke-linejoin, stroke-linecap, stroke-dashoffset, stroke-dasharray, stroke-opacity

Animate a circle on click

```
<circle id="myCircle" r="30" cx="50" cy="50" fill="orange" />
        <animate xlink:href="#myCircle" attributeName="cx" from="50" to="350" dur="1s" begin="click" fill="freeze" />
```

- ALSO : begin="2s" OR begin="click + 2s"

Animations sequence

SEE 39-SVG-ANIMATION-SMIL-SEQUENCE

```
<circle id="orange-circle" r="30" cx="50" cy="50" fill="orange" />
<rect id="blue-rectangle" width="50" height="50" x="25" y="200" fill="#0099cc"></rect>

        <animate
                xlink:href="#orange-circle"
                attributeName="cx"
                from="50"
                to="350"
                dur="3s"
                begin="click"                           /* Can add repeatCount="2"  */
                fill="freeze"                           OR repeatCount="indefinate" repeatDur="01:30" */
                id="circ-anim" />

        <animate
                xlink:href="#blue-rectangle"
                attributeName="x"
                from="50"
                to="330"
                dur="1s"
                begin="circ-anim.begin + 1s"            /* OR circ-anim.end + 1s */
                fill="freeze"                           /* AND/OR begin="circ-anim.repeat(2)"*/
                id="rect-anim" />
```

Values and keyTimes

SEE 40-SVG-ANIMATION-SMIL-VALUES-KEYTIMES

```
<circle id="orange-circle" r="30" cx="50" cy="50" fill="orange" />
<rect id="blue-rectangle" width="50" height="50" x="25" y="200" fill="#0099cc"></rect>

<animate
        xlink:href="#orange-circle"
        attributeName="cx"
        from="50"
        to="350"
        dur="2s"
        begin="click"
        values="50; 350; 300; 350"
        keyTimes="0; 0.8; 0.9; 1"                       /* equivalent of 0%{}; 50%{}... */
        fill="freeze"
        id="circ-anim" />

 <animate
        xlink:href="#blue-rectangle"
        attributeName="x"
        from="50"
        to="330"
        dur="1s"
        begin="circ-anim.begin + 1s"
        fill="freeze"
        id="rect-anim" />
```

- *values* can be used without *keyTimes*   /* then = automatic values */

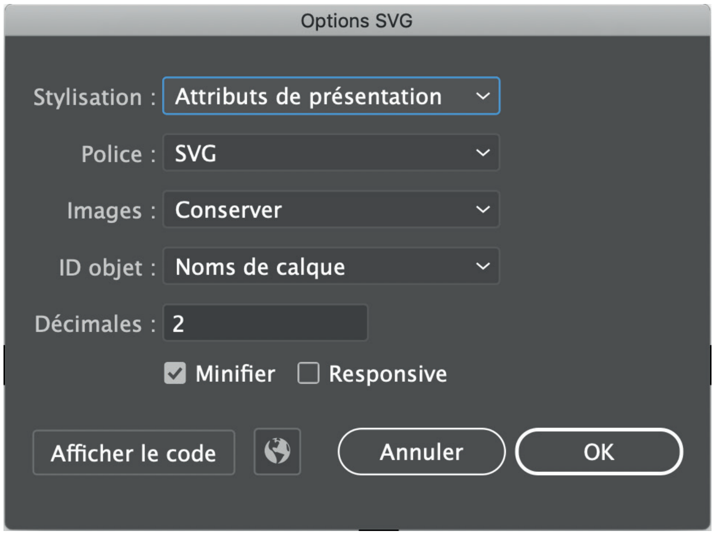
Smooth animation's speed

SEE 41-SVG-ANIMATION-SMIL-VALUES-KEYSPLINES.HTML

```
<circle id="orange-circle" r="30" cx="50" cy="50" fill="orange" />

<animate
        xlink:href="#orange-circle"
        attributeName="cy"
        dur="1s"
        begin="click"
        values="50; 100; 350"              /* position anchor points */
        keyTimes="0; 0.5; 1"               /* Percentage of total animation duration */
        keySplines=".5 0 1 1;
                   .5 0 1 1;
                    0 0 1 1;"
        fill="freeze"
        id="circ-anim" />
```

- values = Position in pixel (here on y axis)
- keyTimes = Linked to *values*, equivalent of *%{}*
- keySplines = Bézier curves (speed variations linked to *values* and *keyTimes*)



Options SVG

Stylisation : Attributs de présentation

Police : SVG

Images : Conserver

ID objet : Noms de calque

Décimales : 2

☑ Minifier ☐ Responsive

Assignment 09: Create an interactive animations sequence full screen using 3 elements

Creating a SVG in an external application

SEE 42-SVG-ILLUSTRATOR

SVG with Illustrator

Creating the illustration

- Create an element in Illustrator.
- Adjust the artboard to fit the element (not way bigger).
- Bring every parts on the same layer
  OR = Groups parts to animate together on of different layers.
- Layers names are used as ID in the code.
- Group parts adequately.
- Vectorized bitmap images if some are used.

Exporting the SVG

- Use FILE / EXPORT / EXPORT AS
- Name the file and select SVG format
- **Stylization : Presentation attributes**
  Codes the attributes into HTML tags.
- **Font : SVG**
  Keeps the textual nature of the text.
- **ID objects : layers names**
  Use layers names as ID in the code.
- Check *Minify* + Do no check *Responsive*.
- SHOW CODE : SVG codes to copy/paste in the HTML document.
- OK : Saves the file as SVG so it can be integrated just as another image in a HTML document.

Inregrate a SVG produced with an external application

SEE 43-SVG-INTEGRER-ILLUSTRATOR

<img>

```
<img src="nounours.svg" />
```

- Only for static SVG

<object>

```
<object type="image/svg+xml" data="nounours.svg"></object>
```

- For SVG animations only OR SVG into which only certain parts are to be animated.

<svg> & <image>

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
        <image xlink:href="nounours.svg" />
</svg>
```

SVG as a background image

```
<style>
        selector {
                background-image: none, url('green-circle.svg');
                background-position: 50% 50%;
                background-repeat: no-repeat;
        }
</style>
```

Assignment 10: Create a simple illustration in Illustrator an animate it in an interactive manner.

COURSE 13

SVG filters *SEE 44-SVG-FILTERS*

- All <filter> must be contained between <defs>.
- <filter> needs *id* to be identified in the element to modify.

<feGaussianBlur>

```
<svg height="200" width="200">
        <defs>
                <filter id="f1" x="0" y="0" width="200%" height="200%">
                        <feGaussianBlur in="SourceGraphic" stdDeviation="10" />
                </filter>
        </defs>

        <rect width="200" height="200" stroke="green" stroke-width="10" fill="yellow" filter="url(#f1)" />
</svg>
```

- in="SourceGraphic" = Means effect is created for the entire element
- stdDeviation = Amount of blur

<feOffset>

- Often used to create a drop shadow.

```
<svg height="300" width="300">
        <defs>
                <filter id="f2" x="0" y="0" width="200%" height="200%">
                        <feOffset result="offOut" in="SourceGraphic" dx="30" dy="30" />
                        <feBlend in="SourceGraphic" in2="offOut" mode="normal" />
                </filter>
        </defs>

        <rect width="250" height="250" stroke="green" stroke-width="3" fill="yellow" filter="url(#f2)" />
</svg>
```

<feOffset> avec <feGaussianBlur>

- Often used to create a drop shadow.

```
<svg height="300" width="300">
        <defs>
                <filter id="f3" x="0" y="0" width="200%" height="200%">
                        <feOffset result="offOut" in="SourceGraphic" dx="20" dy="20" />
                        <feGaussianBlur result="blurOut" in="offOut" stdDeviation="10" />
                        <feBlend in="SourceGraphic" in2="blurOut" mode="normal" />
                </filter>
        </defs>

        <rect width="250" height="250" stroke="green" stroke-width="3" fill="yellow" filter="url(#f3)" />
</svg>
```

- For a black shadow, change in feOffset in="SourceGraphic" for in="SourceAlpha"

<linearGradient>

```
<svg height="300" width="400">
    <defs>
        <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
            <stop offset="0%" style="stop-color:red;stop-opacity:1" />
            <stop offset="100%" style="stop-color:yellow;stop-opacity:1" />
        </linearGradient>
    </defs>

    <polygon points="0,0 400,0 400,300 0,300" fill="url(#grad1)" />
    <text fill="#ffffff" font-size="100" font-family="Verdana" x="5%" y="60%">HELLO!</text>
</svg>
```

- To add text in the shape = <text> </text>

<radialGradient>

```
<svg height="300" width="400">
    <defs>
        <radialGradient id="grad1" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
            <stop offset="0%" style="red;" />
            <stop offset="100%" style="yellow;" />
        </radialGradient>
    </defs>

    <ellipse cx="200" cy="100" rx="250" ry="100" fill="url(#grad1)" />
    <text fill="#ffffff" font-size="100" font-family="Verdana" x="5%" y="45%">HELLO!</text>
</svg>
```

<radialGradient> with variable opacity

```
<svg height="300" width="400">
    <defs>
        <radialGradient id="grad3" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
            <stop offset="0%" stop-color="red" stop-opacity="1" />
            <stop offset="100%" stop-color="red" stop-opacity="0" />
        </radialGradient>
    </defs>

    <ellipse cx="200" cy="100" rx="250" ry="100" fill="url(#grad3)" />
</svg>
```

<animationMotion>

```
<svg width="500" height="500">
    <g transform="translate(0,0)">
        <text x="-100" y="20" style="font-family:Verdana;font-size:24">
        Ceci est un exemple
            <animateMotion path="M 0 0 L 400 400" dur="3s" fill="freeze" />
        </text>
    </g>
</svg>
```

Several examples :

https://www.w3schools.com/graphics/svg_examples.asp

Clipping SVG

SEE 45-SVG-CLIPPING

- To make a clipping mask (no gradients).

```
<svg>
	<defs>
		<clipPath id="cut-bottom">
			<rect x="-100" y="-100" width="500" height="150" />
		</clipPath>
	</defs>

	<circle cx="100" cy="100" r="200" clip-path="url(#cut-bottom)" />
</svg>
```

SVG mask

- To mask (even with a gradient).

```
<svg>
	<defs>
		<linearGradient id="myGradient">
			<stop offset="0" stop-color="white" stop-opacity="0" />
			<stop offset="1" stop-color="white" stop-opacity="1" />
		</linearGradient>

		<mask id="myMask">
			<rect x="-100" y="-100" width="400" height="400" fill="url(#myGradient)"  />
		</mask>
	</defs>

	<rect x="-100" y="-100" width="400" height="400" fill="red" />
	<rect x="-100" y="-100" width="400" height="400" fill="yellow" mask="url(#myMask)" />
</svg>
```

Final project :

- Teacher assigns a static web site to be redone (for instance, an artist's web site)
- Redo the home page interface
- The page must be dynamic and react to user's activities
- Student must work on the wow effect.

COURSE 14

Revision

Workshop

COURSE 15

Final exam

Handle final project