Web design and development II (CW2)

Lesson plan

Class 01

Revision HTML + CSS

HTML document structure

<!DOCTYPE html>
<html lang="en"> + <meta charset="UTF-8">

Container CSS :

width + height + background-color
margin + padding + box-sizing : border-box

Margin and padding shorthand :
margin: 10px 5px 10px 5px;

border-style (solid, dotted, dashed, double, groove, ridge, inset, outset, none, hidden)
border-width + border-color
border-radius
outline-offset

Border shorthand :
border: 1px solid #000;

background :

background-image: url("image.jpg");
background: url(img_tree.gif), url(mountain.jpg);
background-repeat: no-repeat;
background-size: contain, cover, px, %;
background-position
background-attachment (fixed)

Shorthand :
background : #ffffff url("image.jpg") no-repeat right top;

position :

static
relative
absolute
fixed

class:

Simple classes:
.class

Multiple classes:
class="class1 class2"

Dedicated classes:
selector.class

id:

Difference between class and id

id useful with JavaScript

id useful to navigate one-pager

float + clear:

Semantic structural tags

<header> + <footer>

<main>

<nav>

<section>

<article>

<aside>

<figure>

<figcapture>

Semantic styling tags

<h1>...

<b>

<i> <cite>

Text styling CSS

font-family

font-size (auto, px, %)

font-weight (normal, bold, bolder, lighter, number, initial, inherit)

font-style (normal, italic, oblique)

font-variant (normal, small-caps)

color + HEX + rgb() + rgba()

line-height

Text styling shorthand :
font: italic bold .8em/1.2 Arial, sans-serif;

text-align (left, right, center, justify)

text-indent

text-transform (lowercase, uppercase, capitalize)

letter-spacing

word-spacing

direction (rtl, ltr)

text-shadow: 3px 2px red;

Font styling shorthand :
font: italic bold .8em/1.2 Arial, sans-serif;
font: italic small-caps bold .9em/1.1 arial,helvetica,sans-serif;

List styling shorthand :
list-style: url("dot.gif") disc inside;

hyperlinks :

External links :
(http://)

Internal links (local) :
file.html

Internal links (id / #) :
<a href="#my_id">

text-decoration: (none, underline, overline, line-through)

pseudo class :

:hover
:link
:active
:visited

pseudo elements :

::first-letter
::first-line
::before
::after
::selection

WARNING :

a:hover
Must be placed after a:link and a:visited

a:active
Must be placed after a:hover

Assignment 1 : One-pager integration

(PSD interface supplied by teacher)

Class 02 :

Columns layout

Number of columns:
column-count: 3;

Columns width:
column-width: 100px;

Gutter:
column-gap: 40px;

Rules (vertical separators) :
column-rule-style: solid;
column-rule-width: 1px;
column-rule-color: lightblue;

column-rule shorthand:
column-rule: 1px solid lightblue;

Column span:
column-span: all;

Exercise : Columns layout

Flexbox

Container display

- Container is defined as flex
  items will flow according to containers parameters
- Main axis / cross axis

Items orientation

flex-direction: row / column

Items horizontal alignment

justify-content: flex-start / center / flex-end / space-between / space-around

Items vertical alignment

align-items: stretch / flex-start / center / flex-end

align-content: flex-start / center / flex-end

Content wrap

flex-wrap: nowrap / wrap / wrap-reverse

Items auto-align

align-self: flex-start / center / flex-end

justify-self: flex-start / center / flex-end

Items dimensions

flex-grow: / flex-shrink:

flex-basis: 0 / auto / mesure abs. ou rel.

Assignment 1 : One-pager integration

(PSD interface supplied by teacher : responsive desktop/mobile)

Class 03

Grids

Simple grid

- Define container as grid (display)
- Add items to container
- grid-template-columns: 1fr 1fr 1fr;
- Measurement : fr
- grid-template-rows: 1fr 1fr 1fr;
- grid-gap
- repeat function :
  grid-template-columns: repeat(3, 33.33%)
- To indicate position of items in grid
  grid-column / grid-row

Positioning

- Cells order
  grid-column / grid-row
- repeat() function
  repeat(3, 33.33%);
- grid-area
  grid-area: 2 / 2 / 3 / 3;
- Column/row span
  grid-column: 1 / 3;
  grid-column: 2 / span 2;

```
div.container {
        width: 100%;
        height: 100%;
        display: grid;
        grid-template-columns: 150px 1fr 150px;
        grid-template-rowss: 100px 1fr;
        grid-gap: 10px 20px;

}

div.item {
        background-color : lightblue ;
}

<div class="container">
        <div class="item">1</div>
        <div class="item" >2</div>
        <div class="item">3</div>
        <div class="item">4</div>
        <div class="item">5</div>
        <div class="item-">6</div>
</div>
```

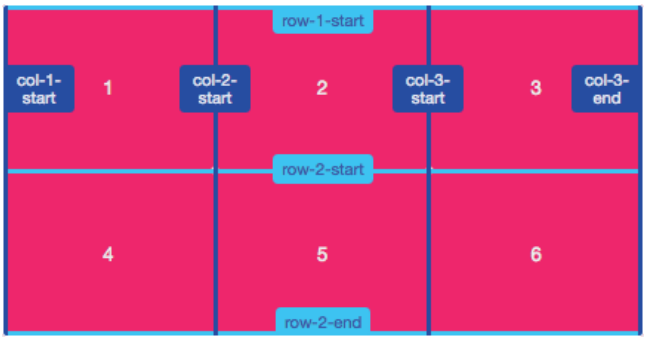
```
grid-row-start:    2;
grid-row-end:      3;
grid-column-start: 2;
grid-column-end:   3;

ou

grid-row:    2;
grid-column: 3 / 4;
```

Grids (suite)

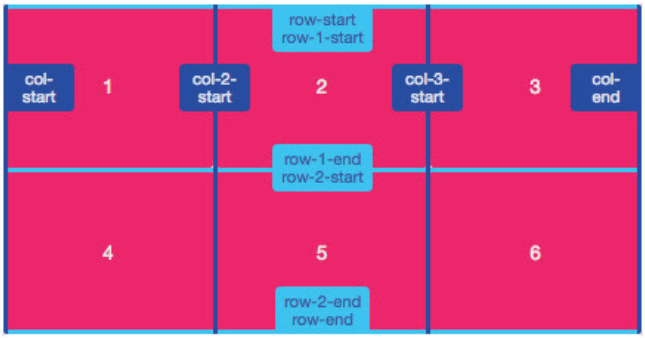
Naming grid lines

(Do not cover if group has difficulties)

- Associate names to lines
- Use names for positioning



- Use of multiple names possible



Items positioning using grid lines names

- repeat() may be used
- Implicit numbers

grid-template-rows:
[row-1-start] 1fr [row-2-start] 1fr [row-2-end];

grid-template-columns:
[col-1-start] 1fr [col-2-start] 1fr [col-3-start] 1fr [col-3-end];

[row-start row-1-start] 1fr [row-1-end row-2-start] 1fr [row-2-end row-end];

grid-template-columns:
[col-start] 1fr [col-2-start] 1fr [col-3-start] 1fr [col-end];

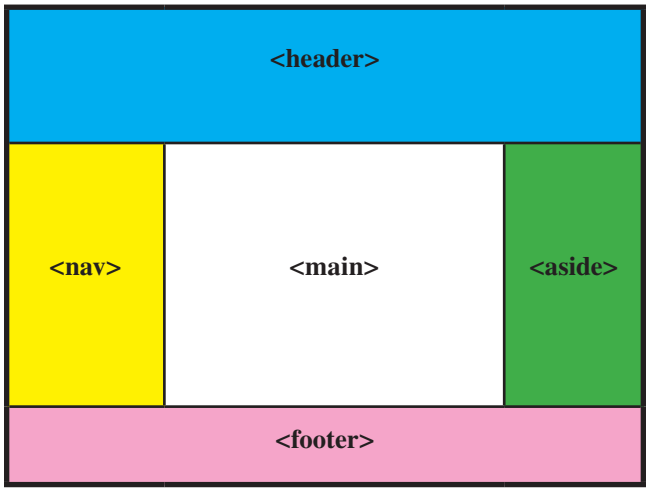
grid-row-start: row-2-start;
grid-row-end: row-end;
grid-column-start: col-2-start;
grid-column-end: col-end;

repeat(3, [row-start] 1fr [row-end]);

grid-template-columns:
repeat(3, [col-start] 1fr [col-end]);

grid-row: row-start 2 / row-end 3;
grid-column: col-start / col-start 3;

Positioning items using grid-area




```
grid-template-areas: "top top top"
                     "left center right"
                     "bottom bottom bottom";

grid-template-rows:    100px 1fr 50px;
grid-template-columns: 300px 1fr 200px;



header {
        grid-area: top;
}

nav {
        grid-area: left;
}

main {
        grid-area: center;
}

aside {
        grid-area: right;
}

footer {
        grid-area: bottom;
```

- We may also use :
  grid-row-start / grid-row-end
  grid-row / grid-rcolumn

```
grid-row-start: top;
grid-row-end: top;
grid-column-start: top;
grid-column-end: top;

grid-row: bottom;
grid-column: bottom;
```

- Automated extra columns/row creation
  «grid-auto-rows »
  «grid-auto-columns »

```
grid-template-rows: 70px;
grid-template-columns: repeat(2, 1fr);
grid-auto-rows: 140px;
grid-auto-columns: 1fr;
```

Items alignment in grids

- Horizontal :
  justify-items
- Vertical :
  align-items
- Supported values :
  - auto
  - normal
  - start
  - end
  - center
  - stretch
  - baseline
  - first baseline
  - last baseline

```
div.container {
        width: 100%;
        height: 100%;
        display: grid;
        grid-template-columns: 150px 1fr 250px;
        grid-template-rows: 100px 1fr;
        grid-gap: 10px 20px;
}

div.item {
        background-color : lightblue ;
}

<div class="container">
        <div class="item">1</div>
        <div class="item" >2</div>
        <div class="item">3</div>
        <div class="item">4</div>
        <div class="item">5</div>
        <div class="item-">6</div>
</div>
```

Assignment 2 : fluid grid page

- Create a container with nine items
- Each item contains a background color and a centered title.
- On rollover, an image replaces the background color and the title may change color to insure readability.

Class 04

Absolute measurement units (invariable)

Demeure toujours la même et ne varie pas selon le contexte.

Can be used, but not always stable (depending of screen resolution)

- Pixels (px)
- Point (pt)
- Pica
- Inch (in)
- Centimetre (cm)
- Millimetre (mm)

Relative measurement units (variable)

Varies depending on context.

- Em (em)
- Percentage (%)
- Rem (rem)
- vh / vw (1% of device)

Web design approaches

- Many different web design approaches.
- Design got more complexed since new devices : tablets, mobile, etc. (+50% of users)
- Helps us : new languages, technologies and approaches
    - **Dedicated design**
      Different version of site (with different sub-domain) for each device : ordinateurs / tablettes / mobiles
    - **Fluid design (liquid)**
      All dimensions and size in variable measurement units
      (%, em, vw, etc.).
    - **Adaptive design**
      Improvement of static design.
      Fixed dimensions (depending on viewport) + media queries
      (Braking points : 320px, 480px, 768px, 1024px, etc.)

- **Responsive design**
  Improvement of fluid design.
  Use of media queries + Content and structure reorganized depending on viewport

- **Mobile first**
  Creation of mobile version first THEN other version (progressive enhancement)
  Breaking points (minimal) : mobile, tablet, desktop

- **Progressive enhancement**
  More content as space become available (depending on viewport)

- **Progressive degradation**
  Inverse of mobile first = Desktop first
  Les content as space become limited (depending on viewport)

- Examples : http://www.liquidapsive.com/

Comon components of responsive web page (RWD)

- Fluid grid (variable measurement units)
- Flexible images and contents (do not overflow from their containers).
- Media queries
  Styles (or CSS file) vary depending on viewport.
- Mobile first / progressive enhancement
  (for accessibility, compatibility and performance).
- Use of server-side technology to optimize performance when possible

Media queries

- <link rel="stylesheet" media="screen" href="screen.css">
  <link rel="stylesheet" media="print" href="print.css">
- <link rel='stylesheet' media='screen and (min-width:600px)' href='medium.css'>
- @media screen {...}
  @media screen and (max-width:600px) {...}

Assignment 3 : media queries

Create a web page using media queries that will have a breaking point from desktop to mobile

Ex.: http://2011.uxlondon.com/

Class 05

- Workshop
- Continue media query assignment

Class 06

Assignment 4 : responsive design (progressive enhancement)

- Workshop : mobile first / progressive enhancement
- Responsive (mobile / tablet / desktop)
  Improve or redo an existing web site (teacher provides link to chosen web site)

Class 07

- Revision
- Continue working on assignment 4

Class 08

Intra exam

Class 09

Create a custom pop-up window (JavaScript)

- Create web page / interface
- Create and position pop-up window (CSS):
    - Position: absolute
      In an overlay (if desired)
      z-index (higher)
      display: none
    - Create an id in the main window's container (to use later with JavaScript event)
- Create a function to show the window (changing display to block):

```
function show(){
        document.getElementById("myPopup").style = "display:block";
}
```

- Create a function to hide the window (changing display to none):

```
function hide(){
        document.getElementById("myPopup").style = "display:none";
}
```

- Create the events :
  Example : onclick, onmouseover, onmouseout, etc.

Exercise: Create a pop-up window

Use the same procedure as the pop-up window.

Create a custom pop-up window using pseudo-class :target

See course note Class 9 page 4

- Defines a style to targeted selector
  Simple example changing style on a container

Creating the pop-up window

- Create CSS class for custom window
  Add an id to the window container for your future link (id="link")
  THEN display none

- Create the class:target (display: block)

- Create the link (#link)

Create contents under tabs using pseudo class :target

See course note Class 9 page 5

- Create the navigation

- Create the content containers
  Style with a class
  THEN display: none to hide them

- Create class:target (display: block)
  to show container upon cliking the proper link

- Add id to each container to be used as links

- Use the id as anchors in navigation links

CSS drop-down menu

- Two levels of links: first and second (nav/subnav)
- make group of nav + subnav
- Within <ul>: another <ul> for subnav
- for each link
- Display:inline-block (to show <ul> horizontally)
- z-index to show subnav on top
- display: none/block to show and hide

HTML:

```
<nav>
        <ul>
                <a href="#">Link 1</a>
                <ul class="submenu">
                        <a href="#">A</a>
                        <a href="#">B</a>
                        <a href="#">C</a>
                </ul>
        </ul>
</nav>
```

CSS:

```
ul {
        position: relative;
        display: inline-block;
        margin: 0px;
        padding: 0px;
        background: lightyellow;
}

ul ul {
        display: none;
        position: absolute;
        z-index: 1;
}

ul:hover .submenu {
        display: block;
        width: 100%;
}

ul a {
        display: inline-block;
        padding: 10px 20px;
}

ul ul a {
        display: inline-block;
        padding: 10px 20px;
        width: 100%;
        border: solid black 1px;
}
```

Class 10

Styliser les images

- **Full width image**s
    - width / height
      width: 100% + display: block (in a container) = full width of the container
      height: auto
      To avoid scaling over the original size of the image = max-width: 100%
    - To adjust image size like for background-image = object-fit: cover
      (fill / contain / cover / none / scale-down)
    - If content is bigger than container = overflow: hidden/scroll etc.
      overflow-y: hidden/scroll etc.
      overflow-x: hidden/scroll etc.
    - borders (border, radius, round images)
    - Images transparency
      opacity: 0.5;
    - Images/ container with drop shadows
      box-shadow: 0 0 5px 5px rgba(0, 140, 186, 0.5);
- **Filter effects**
    - filter: blur(4px);
    - filter: brightness(250%);
    - filter: contrast(180%);
    - filter: grayscale(100%);
    - filter: hue-rotate(180deg);
    - filter: invert(100%);
    - filter: opacity(50%); ALSO = opacity: .5;
    - filter: saturate(7);
    - filter: sepia(100%);
    - filter: drop-shadow(8px 8px 10px green); (x, y, blur, color)
- Transform
    - transform: rotate(20deg);
    - transform: skewY(20deg);
    - transform: scaleY(1.5);
    - transform: translate(50px, 100px);
    - transform: matrix(1, -0.3, 0, 1, 0, 0); (scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

Shadows

- Text shadow
  text-shadow: 2px 2px 5px #dddddd;
  text-shadow: 0 0 3px rouge, 0 0 5px blue;

- Box shadowt
  box-shadow: 10px 10px 5px grey;

- Internal shadow (offset)
  box-shadow: inset 0 0 10px #000000;

Gradients

- Linear gradient
  background-image: linear-gradient(red, yellow);

- Orientation of the gradient
  background-image: linear-gradient(to right, red, yellow);

- Diagonal gradient
  background-image: linear-gradient(to bottom right, red, yellow);

- Multiple colors gradient
  background-image: linear-gradient(red, green, yellow);

- Radial gradient
  background-image: radial-gradient(red, green, yellow);

- Define the form of radial gradient (circle/ellipse)
  background-image: radial-gradient(circle, red, yellow);

- Gradient's spans
  background-image: radial-gradient(red 5%, green 15%, yellow 55%);

Final project : Redo an existing web site

Teacher provides the web site to be redone (breaking points : desktop + mobile)

- Flow chart + schema
- Produce the interface
- Plan and slice the interface
- Integrate interface + develop navigation + tests (alpha version)
- Integrate content

Class 11

Interface integration workshop

Demonstration

- Slicing interface :
    - Planning the integration process
    - Define general structure :
        - Horizontal positioning
        - Content max-width and position
        - Header, main, footer and various rows
- Integration of interface's elements
- Developing navigation

Exercise :
Finish interface
THEN produce mobile breaking point using media queries

Class 12

Audio and video

- Integrating and managing audio
- Integrating and managing video
- Creating a video background

Workshop: final project

Class 13

Workshop: final project

Class 14

Revision

Workshop: final project

Class 15

Final exam