

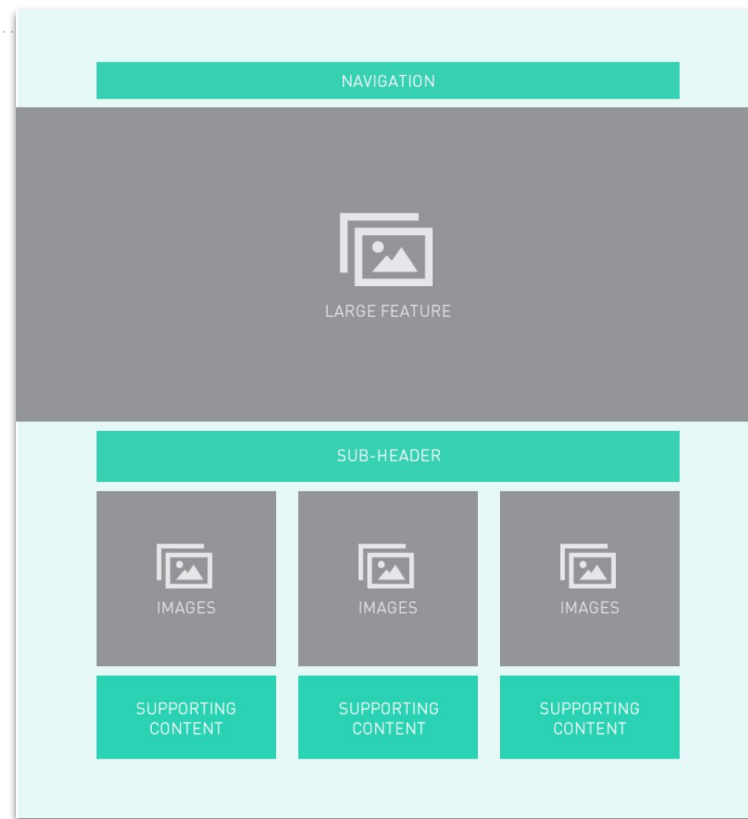
# INTRODUCTION TO WEB PROGRAMMING

## Chap. 2 / HTML5 & CSS3, part 2

Anne Jeannin-Girardon, PhD | [anne.jeannin@unistra.fr](mailto:anne.jeannin@unistra.fr)  
Associate Professor, University of Strasbourg

# Today's topics

- Metadata (data about the data)
- Character encoding
- Building forms
- Including media
- CSS : displaying elements



# Tags for metadata

Not displayed ;  
Used by search engines

index.html

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Sandbox</title>
    <meta name="description" content="Demonstration page"/>
    <meta name="keywords" content="lorem ipsum, html, css"/>
    <meta name="author" content="WebProg Team"/>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <style></style>
  </head>
  <body>
    </body>
</html>
```

General information  
about the page

Encoding

Style definition (w/  
stylesheet or direct)

# HTML responsive

- Using the meta tag for the viewport nicely scale things without much effort
- It's the first step towards responsive design
- `<meta name="viewport" content="width=device-width, initial-scale=1">`
- (more on the subject later)

## Page heading

- [Home](#)
- [Pusheen comic strips](#)
- [Contact](#)

## A big paragraph



Chase laser sniff sniff and leave dead animals as gifts howl on top of tall thing. Poop in litter box, scratch the walls i just saw other cats inside the house and nobody ask me before using my litter box or lick the plastic bag relentlessly pursues moth but purr while eating, but that box? i can fit in that box inspect anything brought into the house. Eat half my food and ask for more. Dont wait for the storm to pass, dance in the rain. Unwrap toilet paper lay on arms while you're using the keyboard. Being gorgeous with belly side up. More napping, more napping all the napping is exhausting being gorgeous with belly side up, claw drapes, so inspect anything brought into the house.

## Some miscellaneous stuff

Some text in a paragraph

This is another paragraph

Some important stuff

## Page heading

- [Home](#)
- [Pusheen comic strips](#)
- [Contact](#)

## A big paragraph



Chase laser sniff sniff and leave dead animals as gifts howl on top of tall thing. Poop in litter box, scratch the walls i just saw other cats inside the house and nobody ask me before using my litter box or lick the plastic bag relentlessly pursues moth but purr while eating, but that box? i can fit in that box inspect anything brought into the house. Eat half my food and ask for more. Dont wait for the storm to pass, dance in the rain. Unwrap toilet paper lay on arms while you're using the keyboard. Being gorgeous with belly side up. More napping, more napping all the napping is exhausting being gorgeous with belly side up, claw

# Character encoding

- Different ways of encoding alphanumeric characters:
  - ASCII, ANSI, ISO-8859-1, UTF-8, ...
- ASCII: 7 bits so  $2^7$  possible characters
- UTF-8: 64 bits (8 bytes),  $2^{64}$  characters -- sounds better?
- Chrome: `<meta charset="utf-8"/>`
- Firefox:  
`<meta content="text/html; charset=UTF-8" http-equiv="content-type"/>`



# HTML symbols

---

- UTF-8 allows us to directly write “é” instead of “&eacute;”
- All possible symbols are not necessarily on one’s keyboard
  - E.g. I don’t have a “euro” symbol on my Qwerty keyboard
  - No math symbols either
  - So we can use **HTML entities** (basically a name or a number) :  
`<p>This cost &euro; 20</p> <!-- eq. number: &#8364; -->`
- HTML entities are **interpreted** by the browser
- Other HTML entity example : non-breakable space `&nbsp;`;

# Creating forms

---

- Forms are used by user to **interact** with the web page
- You can define text fields, radio buttons, checkboxes, submit buttons, ...
- Data are gathered by the browser, sent to the server and treated by the server => this is out of the scope of this course
- But ! Javascript allows us to ~~fool around~~ do things with data from forms (e.g. check text fields format)

# Forms and HTTP requests

---

- Two major requests are GET and POST
- **POST requests** send data on the server (results in a change on the server)
  - not kept in the browser's cache nor in the history ; they are not displayed in the URL, you cannot bookmark them
  - They also are re-submitted if you frantically click on the submit button
- **GET requests** look like this :  
[https://scholar.google.fr/scholar?hl=en&as\\_sdt=0%2C5&q=deep+learning&btnG=](https://scholar.google.fr/scholar?hl=en&as_sdt=0%2C5&q=deep+learning&btnG=)
  - Used to access resources (without changing anything), e.g. search a page
- `<form action="/path/to/server/script" method="post"></form>`



# Form example

- In text fields, you can specify
  - the size
  - max/min nb of characters (maxlength)
  - a default value, ...



The screenshot shows a web form with the title "Testing forms". It contains two text input fields: "Username:" and "Password:". Below these fields are two buttons: "Submit" and "Reset form".

index.html

```
<h2>Testing forms</h2>
<form>
  <label for="user">Username:</label>
  <input type="text" name="user"/><br/>
  <label for="pwd">Password:</label>
  <input type="password" name="pwd"/><br/>
  <input type="button" name="submit"
value="Submit"/>
  <input type="reset" value="Reset form"/>
</form>
```

# Making choices: radio buttons & checkboxes

index.html

```
<form>
  <!-- ... -->
  <label for="topping">Topping:</label><br>
    Strawberry <input type="checkbox" name="topping" value="Strawberry"/>
    Anchovy <input type="checkbox" name="topping" value="Anchovy"/>
    French fries <input type="checkbox" name="topping" value="French fries"/><br>

  <label for="beverage">Choose a drink:</label><br>
    Pepsi <input type="radio" name="beverage" value="Pepsi"/>
    Coca <input type="radio" name="beverage" value="Coca"/>
    Sprite <input type="radio" name="beverage" value="Sprite"/><br>

  <textarea name="comment" rows="10" cols="50">A textarea</textarea><br>
  <!-- ... -->
</form>
```

## Testing forms

Username:

Password:

Topping:

Strawberry ☐ Anchovy ☐ French fries ☐

Choose a drink:

Pepsi ☐ Coca ☐ Sprite ☐

A textarea

# Forms: what does HTML5 add

- New input types: `url`, `tel`, `email`, `date`, ...
  - See <https://www.html5rocks.com/en/tutorials/forms/html5forms/>
- Easy form validation: `pattern`, `required` attributes, `placeholder`, ...
- Why?
  - Easier to implement some functionalities
  - Better access for user

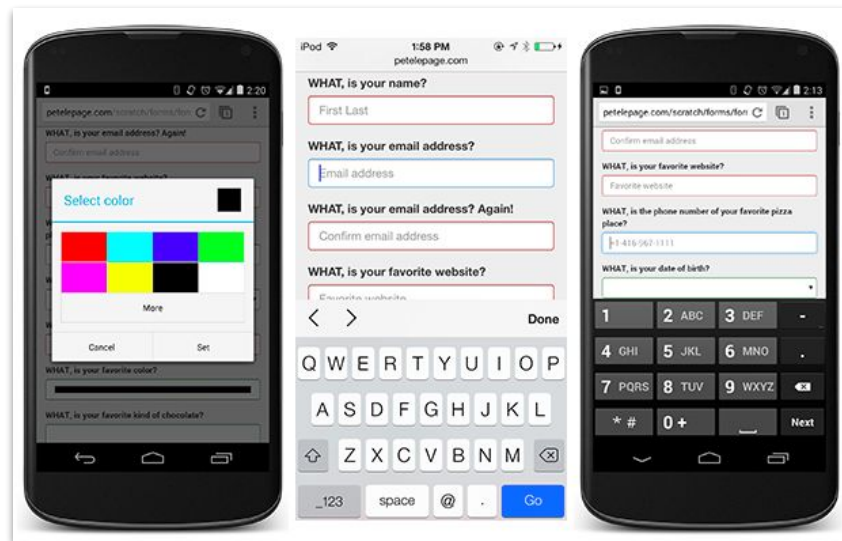


Illustration: [html5rocks.com](https://www.html5rocks.com)

# Including media in a web page

---

- Web pages can include audio, video, ...
- The supported format are varied (mp3, wav, ogg, mp4, webM, ogg video,..)
- Several sources can be specified ! The browser will load the first source it can read (the format is known)

```
<video width="640" height="480" controls autoplay>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogg" type="video/ogg">  
  Video tag not handled by the browser  
</video>
```

- Works the same way for audio !
- You can still include flash media (but flash kinda is deprecated nowadays)

# MORE CSS

# Combining selectors

---

- Adjacent elements at the same level (siblings) :  $a + b$ 
  - $h2 + p$  : select all `<p>` directly following a `<h2>`
- Elements at the same level (siblings) :  $a \sim b$ 
  - $p \sim span$  : select all `<span>` following (directly or not) a `<p>`
- Direct child :  $a > b$ 
  - $p > span$  : select the first `<span>` that is a child of `<p>`
- Children :  $a b$ 
  - $p span$  : select all `<span>` that are children of `<p>`

# CSS and layout

---

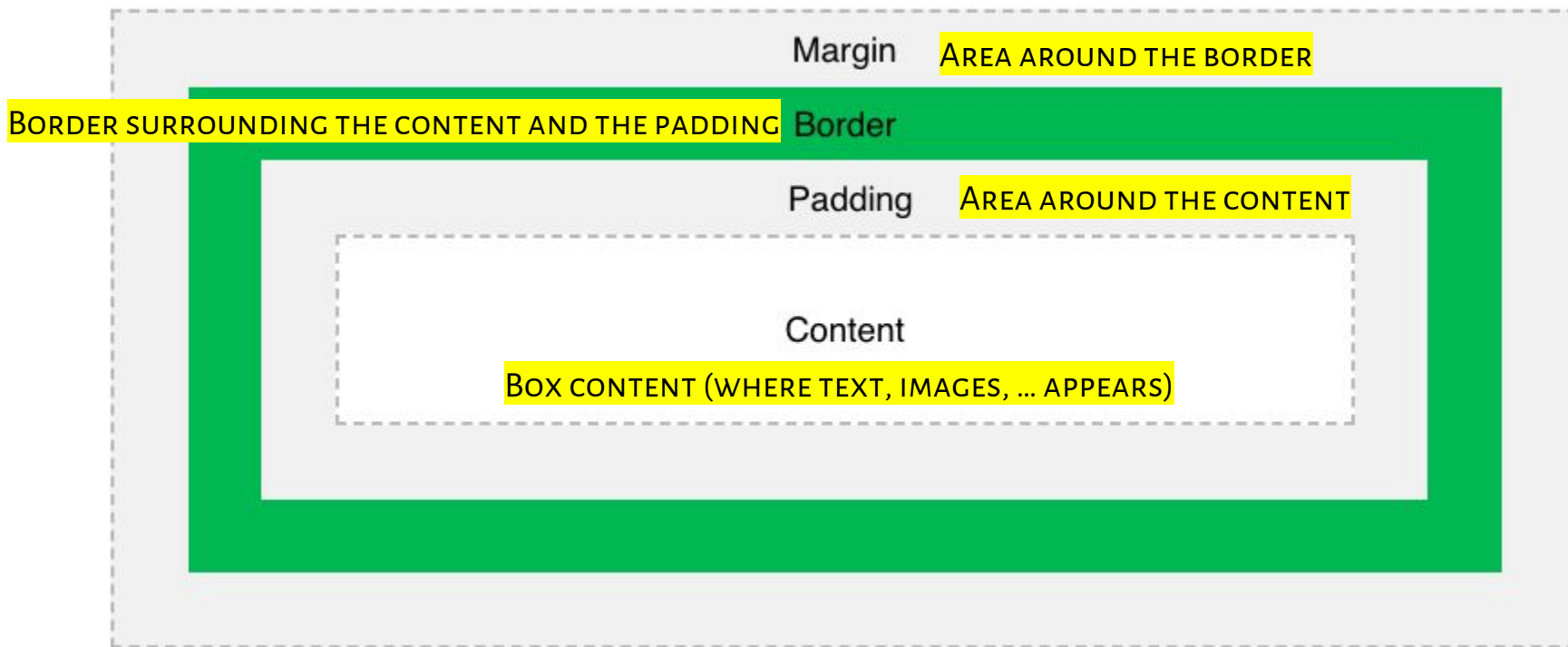
- You'll learn about the CSS box model
- And how to play with this model to build a layout for your page (border, dimensions, positioning and display)

All those elements are what will allow you to build a proper layout for your page

The prerequisite is to have a properly built HTML document

# CSS box model

---





# Dimensions

Beware that setting dimensions to an elements only sets the dimensions of the content!

- Properties : width and height
- Units : absolute and relative (best)

Total width of div  
elements ?

$320 + 10 * 2 +$   
 $5 * 2 = 350 \text{ px !}$

style.css

```
div {  
  width: 320px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```

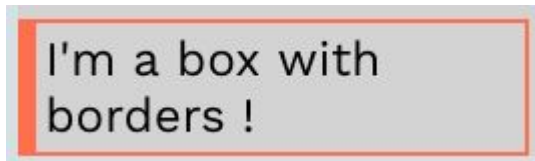
# Border and margin

---

- `border-style: value;`
  - dotted, solid, dashed, double, ...
- `border-width: 2px;`
- `border-color: #ee795b;`

But we can also be more specific (top, right, left, bottom)

- `border-left-width: 2px;`
- Selectors margin and padding
  - e.g. `margin: 2px 3px 2px 3px;`




style.css

```
.test_box {  
  border: 1px solid #ee795b;  
  border-left: 6px solid #ee795b;  
  padding: 3px;  
  width: 20%;  
}
```

# Element positions

---

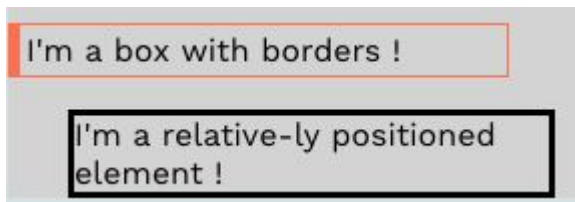
Possible positions for elements : `static`, `relative`, `absolute`, `fixed`, `sticky`

 Normal flow : block elements are disposed from top to bottom (with a newline in-between) and inline element are disposed from left to right

- Static elements (default) : normal flow
- Non-static elements : use properties `top`, `left`, `bottom`, `right` and a distance to place the element

# Relative position

 Relative : element position is relative to its position in the normal flow



style.css

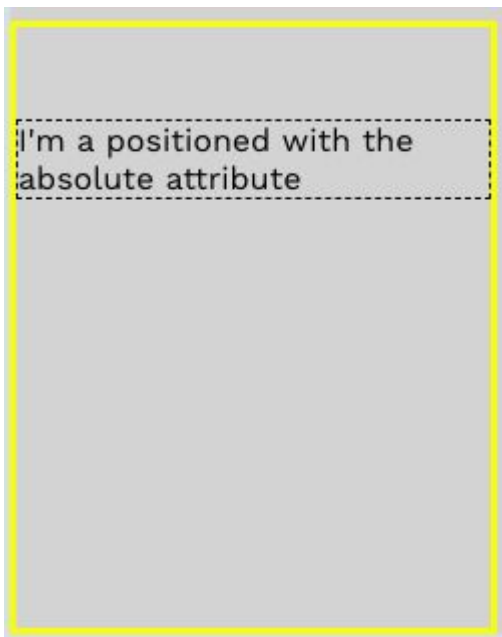
```
.rel_elt {  
  position: relative;  
  left: 30px;  
  border: 3px solid black;  
  width: 20%;  
}
```

Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_position\\_relative](https://www.w3schools.com/css/tryit.asp?filename=trycss_position_relative)

# Absolute position

**absolute** elements are no longer in the normal flow

 Absolute : element positioned regarding its nearest positioned ancestor




style.css


```
.abs_elt_parent{  
  position: relative;  
  border: 3px solid yellow;  
  height: 300px;  
  width: 20%;  
}  
  
.abs_elt {  
  position: absolute;  
  top: 30px;  
  border: 1px dashed black;  
}
```

Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_position\\_absolute](https://www.w3schools.com/css/tryit.asp?filename=trycss_position_absolute)

# Sticky and fixed positions

---

 Sticky : element positioned regarding the user cursor position (element always visible)

 Fixed : element positioned regarding the viewport (always at the same position)

Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_position\\_sticky](https://www.w3schools.com/css/tryit.asp?filename=trycss_position_sticky)

# Floating elements

---

- The `float` property is used to position content
- The possible values for float are `left`, `right`, `none` (default), `inherit`
- One classic use case is to wrap text around an image

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...



Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_layout\\_float](https://www.w3schools.com/css/tryit.asp?filename=trycss_layout_float)

# Element overlap

If you want elements to overlap (i.e. this is not an accident because you don't know how positions work)...

... use the property `z-index` to determine what elements will be positioned under/on top.

CSS Property:

z-index:

☒ auto

☐ 1

☐ 2

☐ 3

☐ 4

☐ initial

Result:



CSS Code:

```
div#myBox {  
  position: absolute;  
  background-color: red;  
  z-index: auto;  
}
```

Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_zindex](https://www.w3schools.com/css/tryit.asp?filename=trycss_zindex)



# Changing the nature of elements: property **display**

---

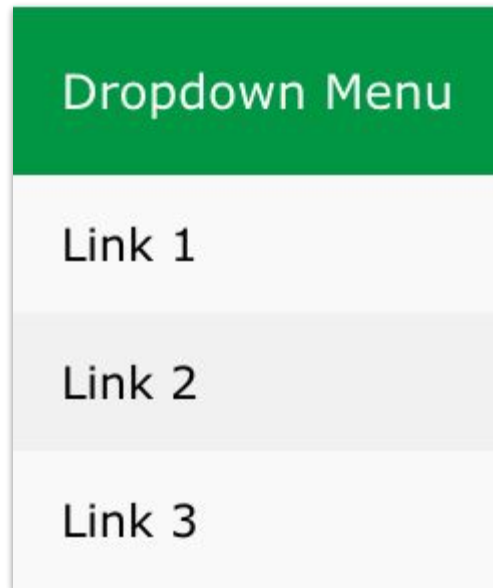
- Adds a lot of control on how we want to lay out the elements
- We can change the nature of the element (block as inline, inline as block or a block that behaves as an inline)
- Possible values: **none**, **block**, **inline**, **inline-block**
- **inline-block** is great : it displays a block as a inline, but it still has the properties of a block ! (especially top and bottom margin/padding)

Try it : [https://www.w3schools.com/css/tryit.asp?filename=trycss\\_inline-block\\_span1](https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1)

# Application: drop-down menu

---

- We want to display elements when the cursor hovers over an element
- You must use **position**, **display** and **pseudo-classes** to implement this one
- But before writing the CSS code, you must define a proper HTML structure !



# drop-down : the code

## HTML structure

```
<div class="dropdown">  
  <span>Mouse over me</span>  
  <div class="dropdown-content">  
    <p>Hello World!</p>  
  </div>  
</div>
```

## CSS properties

```
.dropdown {  
  position: relative;  
  display: inline-block;  
}  
  
.dropdown-content {  
  display: none;  
  position: absolute;  
  background-color: #f9f9f9;  
  min-width: 160px;  
  padding: 12px 16px;  
  z-index: 1;  
}  
  
.dropdown:hover .dropdown-content {  
  display: block;  
}
```

# Chapter recap

---

- Character encoding and HTML symbols
- HTML forms
- Media
- CSS box model, displaying and positioning elements