[HTMLDocument|HTMLElement|Node|NodeList].getElementsByClassNames

For future implementations this DOM getter is thougth to be a native method of any **[HTMLElement]** or any **[Node]** object as well as of any **[NodeList]** object.

In addition this method should also be *kind of a public static* property of the two *namespace*s **[[Node]]** and **[[NodeList]]** since this is the only possible way to provide such funtionality in a comprehensive way to todays browsers afterwards at all.

[className] - The Methods Core Attribute - Thoughts:

[className] should remain a single all purpose attribute, where users will provide their search(pattern(s)) to ...

... **Examples:**

- **"happy, excited, erubescent"** will match any occurence of any of the given comma separeted single _class_names_. Thus such an attribute value gets treated like an arguments list/array.

  And furthermore this is the reason to label this method as **"getElementsByClassNames"**.

- **"happy excited erubescent"** will exactly match any elements **[className]** attribute that features any loose combination of the given (multiple) _class_name_.

- **"happy excited erubescent, excited coy erubescent, excited"** is a combination of the above discussed features. Here the methods returning result list would contain elements that's **[className]** attribute features any loose combination of either the given (multiple) _class_name_ **"happy excited erubescent"** or the (multiple) _class_name_ **"excited coy erubescent"** - moreover every match on **"excited"** will be added to this list.

- The second nature of **[classNames]** is not made up of the just discussed string type. This argument should definitely be allowed to be a **[RegExp]** object as well.

Note: Certainly the result list references every possible match only once.

Multiple Class Names Within An Elements **[className]** Attribute

How It Is Supposed To Work:

Sticking to *[http://www.w3.org/TR/CSS21/selector.html#class-html]* css class rules that are provided in the folowing different ways
...

```
p.happy {background-color: #00bfff;} /* [deepSkyBlue] \*/
p.excited {background-color: #ffa500;} /* [orange] \*/
p.happy.excited {background-color: #ffc0cb;} /* [pink] \*/

p.happy.excited.joking {background-color: #ff0;} /* [yellow] \*/
p.happy.excited.joking.childish {background-color: #ff0;} /* [yellow] \*/
p.excited.erubescent {background-color: #ff7f50;} /* [coral] \*/
p.excited.coy.erubescent {background-color: #f00;} /* [red] \*/ /*

p.happy.excited.joking {background-color: #ff0;}\*/ /* [yellow] \*/
```

... are supposed to be acknowledged if applied by an elements **[class]** attribute as shown in the following example ...

```
  <p class="happy">happy : [deepSkyBlue]</p>
  <p class="excited">excited : [orange]</p>
  <p class="excited happy">excited happy : [pink]</p>
  <p class="happy excited">happy excited : [pink]</p>
  <p class="happy erubescent excited">(happy) erubescent excited – [coral]</p>
  <p class="erubescent happy excited">erubescent (happy) excited – [coral]</p>
  <p class="excited erubescent coy happy">excited erubescent coy (happy) : [red]</p>
  <p class="excited coy happy erubescent">excited coy (happy) erubescent : [red]</p>
  <p class="coy happy erubescent excited">coy (happy) erubescent excited : [red]</p>
  <p class="coy happy erubescent">coy happy erubescent : [deepSkyBlue]</p>
  <p  class="excited  coy  erubescent  happy  excited">excited  coy  erubescent  happy  excited  :
[red]</p>
  <p  class="happy  excited  coy  excited  erubescent">happy  excited  coy  excited  erubescent  :
[red]</p>
  <p  class="erubescent  coy  excited  happy  joking">erubescent  coy  excited  happy  joking  :
[red]</p>
  <p  class="happy  joking  excited  erubescent  coy">happy  joking  excited  erubescent  coy  :
[red]</p>
  <p  class="erubescent  coy  happy  excited  joking">erubescent  coy  happy  excited  joking  :
[red]</p>
  <p  class="happy  excited  joking  erubescent  coy">happy  excited  joking  erubescent  coy  :
[red]</p>
  <p  class="happy  excited  joking  childish  erubescent  coy">happy  excited  joking  childish
erubescent coy : [yellow]</p>
  <p class="happy joking excited erubescent coy childish">happy joking excited erubescent coy
childish : [yellow]</p>
```

... that's practical use clearly points out that order is not an exclude/preclude criterion whether a rule gets applied or not but surely/definitely matters a rules specificity for it will take effect or not.

Thus the discussed getter has to work likewise.

Multiple Class Names Within An Elements `[className]` Attribute

How A Major Part Of The Prospective Users Might Expect It To Work As Well:

They may want to provide multiple classes as one of the many possible variants to which this methods **[classNames]** attribute can adopt/mutate. And they might think of the given order as kind of an identifier to a certain such classified element or rather element group/cluster.

Therefore it should be considered to let this method work by default within the specifictions conform *loose combination* mode. The methods last boolean type argument - maybe **[complyStrictOrder]** - hereby gets omitted and therefore gets converted to **[false]**.
But if this argument was set explicitly **[true] [getElementsByClassNames]** runs within a *strict class names order* mode.
In case the **[classNames]** attribute is a regular expression any value of **[complyStrictOrder]** will be ignored - the method just has to run this given filter.

A fully implemented API of the above discussed matter than might look like the following pseudo code tries to illustrate:

```
/*
  *static* methods of the [[Node]] / [[NodeList]] *namespace*:
*/

NodeList.getElementsByClassNames(nodeListObj, classNames[, complyStrictOrder]);

Node.getElementsByClassNames(nodeObj, classNames[, complyStrictOrder]);


/*
  before a serious implementation takes place
  the type or instance of the returned list
  needs to be discussed: [[Array]] vs [[NodeList]]

  valid arguments values/types/instances:

  - nodeListObj:[HTMLCollection|NodeList|Array]
  - nodeObj:[HTMLElement|Node]
  - classNames:[undefined|null|"*"|""|string|String|RegExp]

    [undefined], [null], [""] are all synonyms for ["*"] and will
    result to a list that references all elements that's [className]
    attribute has been set in any way (HTML coded or assigned by
    JavaScript).

  - complyStrictOrder:[undefined|boolean|Boolean]
*/


/*
  [document] getter that most of the prospective users do expect:
*/

document.getElementsByClassNames(classNames[, complyStrictOrder]);

//[object HTMLDocument].getElementsByClassNames(classNames[, complyStrictOrder]);


/*
  (prototype) methods of every [HTMLElement] and/or [Node] object
  as well as of every [NodeList] object:
*/

[object HTMLElement].getElementsByClassNames(classNames[, complyStrictOrder]);

[object NodeList].getElementsByClassNames(classNames[, complyStrictOrder]);
```