**Mouselab WEB 2.1, Defining the JSON file**

**Styles**

In the styles object, the basic styling and structure of the cells labels and buttons is defined for use in subsequent experiments. Multiple instances of the cell object can be used at the same time. The cell object consists of the properties:

“**name**”: the name of the instance (later to be used as identifier).

“**mainClass**”: defines the main css class(es) to use for the cell (background, border etc).   
Default: ["w3-white", "w3-center", "w3-padding-small"]

“**txtClass**”: defines the class(es) of the txt box (the box containing the text).   
Default: ["w3-light-blue"]  
  
“**boxClass**”: defines the class(es) of the box (mask overlaying the txt box).   
Default: ["w3-indigo"]

“**labelClass**”: this parameter defines the class(es) of the labels (both top and side).   
Default: ["w3-white"]

If you set the the classes to “default”, the default CSS will be used. Otherwise, an array of w3-properties should be entered. So for example replace “default” with [“w3-red”,”w3-right”] (don’t put quotes around this array…)

A special class **(“name”:”label”)** is present to set the label style, to control the height and width and style of the labels separately from the cells

{

"name":"label",

"width":"25%",

"labelClass":"default",

"height":"30px"

}

Another special class **(“name”:”button”)** is used to change the default layout of the buttons:

{

"name":"button",

"btnClass":"default",

"btnTxt": "default",

"btnSel":"default",

"btnNotSel":"default"  
"height": "30px",

"width":"10%"

}

The defaults class defines the Default values for the classes used elsewhere:

{

"name":"defaults",

"mainClass":["w3-white", "w3-center","w3-padding-small"],

"txtClass":["w3-light-blue"],

"boxClass":["w3-indigo"],

"labelClass":["w3-white"],

"btnClass":["w3-button", "w3-block", "w3-border", "w3-border-gray", "w3-round-xlarge", "w3-display-middle"],

"btnTxt": ["w3-white"],

"btnSel":["w3-blue", "w3-hover-blue"],

"btnNotSel":["w3-light-blue"]

}

**Opt**

This object defines the options in the data. Each option is typically a column or row and can have several attributes. Each option has the following properties:

“**name**”: the name of the instance (later to be used as identifier).

“**label**”: the label of the option to display within the trials/experiment. These will be shown on the choice button and in the headers.

**“width”:** The width of each cell belonging to this option

**Attr**

Defines the different attributes belonging to each option

“**name**”: the name of the attribute (later to be used as identifier).

“**label**”: the labek of the attribute to display within the trials/experiment. These will be shown on the choice button and in the headers.

**“height”:** The heightof each cell belonging to this attribute

**Cell**

Defines each individual cell. This is an array which elements contain the values for each attribute and within element the values for this attribute for each option, as associated by the label.

“cell” : [

{"A":{"var":"priceA",

"txt":"<b>5 euro</b>",

"box":"Price",

"style":"A"

}

“**attributes**”: this is an object in itself defining the attributes of this specific option. This object uses the following properties:

“A”: is the label indicating for which option this is.

**“var”**: the names of the cells as stored as variable names in the database. Make them descriptive such that you know which option and cell this is. E.g. priceA indicates this is the price of option A.

“**txt**”: the textual content inside the box corresponding to the attributes used. This is defined as html code. The JSON parser will center the content in the middle of the box, using the w3-display-middle class. However, this class takes up quite some padding space and does not work well with displaying full size images, or with multiple elements inside the box. However, if the content of a cell starts with a DIV, SPAN or IMG, the content will be parsed as is without the w3-display-middle class, so in this way you have more control over the content.

“**box**”: the textual content corresponding to the labels on the boxes that hide the txt. This is defined as a HTML. See previous “txt”description about layout.

**“style”** indicates the style to use for this particular cell, allowing single cell styling.

**Delays**

This variable object allows to set a delay on the opening time of the box, and is defined by a NxN matrix with N the total number of attribute values. The **var** list defines the order of the attributes in the matrix and should contain all attribute labels as defined by the **var** objects for each option. The Matrix defines the delay from row to column: so first row defines the delays going from the first attribute to itself (first column) and the other available attributes. In the Matrix below, there would be a 700ms delay going from A\_quality to B\_quality, and a 500ms delay going from B\_quality to A\_quality and no delays otherwise.

"delay":

{

"var" : [

"A\_price",

"A\_quality",

"B\_price",

"B\_quality"

],

"delays" : [

[0,0,0,0 ],

[0,0,0,700],

[0,0,0,0 ],

[0,500,0,0]

]

}

**OptOrder**

This object defines orders of instances of the options. The order object consists of:

“**name**”: later to be used as instance-identifier.

“**items**”: the option-instances to be included in this order. These are defined as an array (e.g. [“A”, “B”, “C”] for an order using the three option instances corresponding to these names). This also defines which options from the set to use in the experiment.

**Sets**

“**name**”: the name later used to identify the set/configuration to be used in this trial.

“**OptOrder**”: define which order(s) are to be used. Select all options order that are being used within this configuration with the name of the instance of the optOrder object as identifier (e.g. [“order1”, “order5”]).

“**attOrder**”: define in which order the attributes (earlier defined under the txt-object) appear. This parameter can either be set to “standard” (as originally coded in the txt-instance), “reverse” (the reverse order compared to standard) or “random”.

“**layout**”: set the options either to run over the horizontal axis and the attributes over the vertical axis (“optionCol”) or vice versa (“attributeCol”). Note that if options or attributes have different weights of heights, rendering probably will break.

“**buttons**”: either “on” or “off”. Buttons will appear with the options on the columns or rows, determined by the layout property.

“**displayLabels**”: set the display of labels on or off. Select “all” for showing all labels, for only the option names select “optOnly”, for only the attribute names select “attOnly”. For no labels, select “none”.

“**addedVars**”: define additional variables to include. The variables to include should be of the format “name of the variable”=”value of the variable”.