**Mouselab WEB 2.0, Defining the JSON file**

**Cells**

In the cell object, the basic styling and structure of the cells 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-4", "w3-margin-left"]

“**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"]

“**width**”: defines the width of all cells of this type.

“**height**”: defines the height of all cells of this type.

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 of the labels separately from the cells

{

"name":"label",

"width":"25%",

"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"

}

Defaults:

**btnTxt** ["w3-white"];

**btnClass:** ["w3-button", "w3-block", "w3-border", "w3-border-gray", "w3-round-xlarge"];

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

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

**Options**

This object defines the options, names and their attributes. In other words, the content to be used in the trials. Options can have one or more sets of attributes (for example, in a gamble, we might have 3 outcomes and 3 associated probabilities and these would be two sets of attributes with 3 instances). This object has the following properties:

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

“**optionName**”: the name of the option to display within the trials/experiment. These will be shown on the choice button.

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

“**label**:” the label describing the set of attributes. Typically this will show the name of the option, but more than 1 set of attributes it allows to label each of them (e.g. outcomes versus probabilities0.

“**txt**”: the textual content inside the box corresponding to the attributes used. This is defined as an array of html codes (e.g. [X1, X2, X3] for three attributes). Each element of the array will contain some HTML code that defines the content. 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 an array (e.g. [Y1, Y2, Y3] for three attributes). See previous “txt”description about layout.

**“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. A\_p1 would indicate the probability of the first gamble option A.

**Attributes**

“**labels**:” the labels represent the general labels show for each of the attributes of the options (e.g. price, customer rating and quality). These will be shown in the column or row headings when set by the displayLabels attribute (see Set below).

**“var”**: the names of the attributes as stored in the database when referring for example to the attribute order. This allows for shorthand names more useful for data analysis, where the labels themselves can be longer and more descriptive (e.g. “customer rating” in the label is ‘rating’ in shorthand in this **var** list (see tv example).

Note that we assume one similar set of attributes for all options…

**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.

“**styling**”: can be either set to “uniform” (all the same style), “byOpt” (by option), or “byAtt” (by attribute). Make sure cellFormat has the appropriate number of elements for this setting.

**“styleStructure”:** this array determines whether the style should ‘move’ with the options or attributes. If not, use “independent”. With “fixAttributes" it moves with the attributes (useful if styling is set to ‘byAtt’. For styling by options it is a bit more tricky as you need to pair a style with an option, using an Array. For example, {"A":"0",  "B":"1"} to couple the first style with the first option.

“**cellFormat**”: define an object with instances of a specific cellType (the name of an instance of the cell-object defined earlier) and boxType (open/closed/blur). If styling is set for “uniform”, only one style instance needs to be defined. If styling is set for “byOpt” or “byAtt”, define a style instance for each row or column respectively. When the boxType is set to blur, the box is open (no box shown), but the content (the txt property) is blurred

*Example of a style instance:*

{"cellType":"A", "boxType":"closed"}

“**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”).

“**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”.