MouselabWEB 2.0, Defining the JSON file

MouselabWEB 2.0 uses a json definition file to define how options and attributes should be displayed. The basic rationale of the system is this:

Any mouselabWEB table consists of a set of options, with each option consisting of multiple attributes. The basic layout thus will be a grid, and mouselabWEB assumes options are by default in columns and attributes are in rows.

The core part of the json file is the **Set** definition. A set (indicated by its name) defines the layout (what options, attributes to show in which order). A json file can contain multiple sets, and which set is show can be defined by the particular webpage that determines which set is shown. This mean that one json file can be used to show a multitude of different configurations without having to redefine the content. Sets contain a set of references to OptOrders definitions (see below) that define the order of options and attributes, and further determines if options and attributes should be transposed, what labels to show, whether to show buttons etc. Each Set can use different OptOrders and which of the OptOrders is shown is determined by a variable (condnum) or random.

The order of options and attributes in a set is defined by **OptOrders.** It defines which options are shown from the ones defined, their order (randomized, reversed or a specifically defined order). It also defines which attributes to show and in what order. Each OptOrders definition defines one possible configuration.

The core of json file is the definition of potential options (Opt) with their attributes (Attr) and the cells that contain attribute information for each option. At the highest level, Opt defines a set of possible options (and sets some basic features like name, their widths and the labeling), Attr defines the name, labels and row height, and Cell defines for each option and each attribute what is shown and in which formatting.

The Cell definition is very flexible: in the core it will contain just one peace of information (a specific attribute value of a specific option) and the Cell definition sets the variable name, the text inside the box, the text on the outside of the box and the style used for this box (that also defines if the box is open, closed or blurred). However, a cell can also contain several pieces of content, and then a Cell element is an array or matrix of cells, each with their own content and style. For example, a gamble has an outcome and probability, and we might want to keep these together when counterbalancing a layout. Or a single cell contains a list of things (reviews of an option, see the TV example) that we want to keep together. Using partial widths/heights (pwidth/pheight) for each element of the cell you can define a very flexible layout (see the flex example).

Each element within a Cell has its own name (using the var attribute). These names will show up in the data for the mouselabWEB events (mouseover/out) and can thus be made meaningful for later coding. The same names can be used to define potential delays in opening when a particular order of acquisitions is used by the participant. These delays are defined in the Delay definition. Delays are animated which makes them less noticeable and more fluent.

The Styles definition allows you to define specific styles to be used for cells, including their appearance, their type (open: means always open, closed: opens when hovered, blurred: unblurs when hovered) etc.

Appendix: references

Styles

In the styles object, the basic styling and structure of the cells labels and buttons is defined for use in subsequent experiments. The styles are defined by CSS classes. These classes would be defined in a css file or based on existing classes from the W3.CSS styles that MouselabWEB uses. The style object consists of the following 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 if transposed) 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 (if txt_button not set) and in the headers (if header is shown).

"width": The width of each cell belonging to this option

"txt_button": txt as it should appear on a choice button (optional: without this the text of the label attribute is used). Note this will prevail over fixedOptLabels (if set).

Attr

Defines the different attributes belonging to each option

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

"**label**": the label 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. The style attribute determines the layout.

```
"cell" : [
            {"A":{"var":"priceA",
                  "txt":"<b>5 euro</b>",
                  "box":"Price",
                  "style":"A"
                  }
```

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 set of cells that needs a delay as defined by the **var** list: this list only needs to contain all var names of cells that need some delay with other cells. The Matrix defines the delay from row to column: so first row defines the delays going from the first cell 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]
                ]

}
```

OptOrders

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

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

"**opt**": the option-instances to be included in this order. This can be a setting ("standard", "reversed","random") or can be defined as an array (e.g. ["A", "B", "C"] for an order using the three option instances corresponding to these names). In this latter case this also defines which options from the set to use for this order.

"**attr**": the attr-orders to be included in this order. This can be a setting ("standard", "reversed","random") or can be defined as an array (e.g. ["attr1", "attr2"] for an order using the three attribute instances corresponding to these names). In this latter case this also defines which attr from the set to use for this order.

"fixedOptLabels": these additional labels can specific alternative display labels to show that are fixed in their position in the display, despite the actual option order being different. See gamble3.php for a demo: The first gamble is labeled Gamble X, the second Gamble Y, irrespective if the actual underlying gamble is option A or option B. **Of course, this only affects the option and button labeling, not the actual data that goes into the datalyser.** Note: for backwards compatibility, fixedOptLabels only has to be defined on if on the set level "fixedOptLabels":"on" is defined as on.

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"]).

"**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 widths or 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".

“fixedOptLabels”: set fixed labels on the options on or off. Only when defined as “on” it will trigger the code to look for fixedOptLabels as defined in the **OptOrders** definition. Otherwise, no fixedOptLabels need to be defined in **OptOrders**

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