JASPL Language Documentation

Introduction to JASPL

JASPL (Just A Simple Programming Language) is a lightweight, declarative language designed for quickly creating graphical user interfaces (GUIs) and interactive web elements. It simplifies common web development tasks by providing a straightforward syntax for defining elements, applying styles, and adding basic interactivity without requiring deep knowledge of HTML, CSS, or JavaScript.

JASPL programs are executed within a dedicated IDE environment that interprets the JASPL code and renders the corresponding GUI.

I. Program Structure

Every JASPL program must adhere to a specific structure, starting with a program definition and encapsulating its logic within start and end tags. Optional declarations for styling and module imports can precede the start tag.

1. The @jaspl Tag (Mandatory)

- **Syntax:** @jaspl
- **Purpose:** This tag is **mandatory** and must appear exactly once at the beginning of your main program file. It signifies the start of a JASPL program and cannot be modified or used as a custom program name.
- **Rule:** Only one @jaspl tag is allowed per file.

2. The @jaspl-style Tag (Optional)

- **Syntax:** @jaspl-style
- **Purpose:** This tag is **optional**. If present, it enables the use of style commands within your program. If @jaspl-style is omitted, any style command in your code will result in an error.
- **Placement:** Can appear anywhere after @jaspl and before the start tag.

3. The @source() Tag (Optional)

- **Syntax:** @source(<filePath>) [with id(<alias>)]
- **Purpose:** This tag is **optional**. It allows you to import elements defined in other JASPL program files (modules) into your current program.
    - <filePath>: The path to the JASPL file to be imported (e.g., C:/code/program2.jaspl).
    - with id(<alias>): (Optional, but **highly recommended** for add ... from id() usage). Assigns a unique alias to the imported module. This alias is crucial for

referencing elements from that specific source file later.

- **Placement:** Can appear anywhere after @jaspl and before the start tag. Multiple @source tags are allowed to import from various files.
- **Note:** Sourced files are executed recursively. If a sourced file itself contains @source declarations, those will also be processed.

4. The start and end Tags (Mandatory)

- **Syntax:**
  start
  // Your JASPL commands go here
  end

- **Purpose:** These tags delimit the executable content of your JASPL program. All GUI creation, styling, and interactivity logic must be placed between start and end.
- **Rule:** Both start and end tags are mandatory and must appear exactly once within the program, after all @ declarations and in the correct order.

II. Element Creation and Addition

JASPL allows you to create various GUI elements and add them to your application.

1. create Command

- **Syntax:** create <elementType> [with id(<id>)] [content(<text>)] [links to(<url>)] [source(<filePath>)]
- **Purpose:** Creates a new GUI element and adds it to the output.
  - <elementType>: The type of element to create (see list below).
  - with id(<id>): (Optional) Assigns a unique identifier to the element. This ID is used for styling, linking, sourcing, and when...do blocks. If omitted, a unique ID will be auto-generated.
  - content(<text>): (Optional) Sets the initial text or placeholder for the element. If not provided, the element will appear blank.
  - links to(<url>): (Optional) Makes the element clickable and opens the specified URL in a new tab/window when clicked.
  - source(<filePath>): (Optional) Makes the element clickable and executes the specified JASPL file when clicked.
- **Supported Element Types:**
  - button: A clickable button.
  - radiobutton: A radio button (part of a group, only one can be selected).
  - slider: A range input slider.

    - checkbox: A checkbox.
    - textbox: A single-line text input field.
    - uploadbutton: A file upload input.
    - canvas: A drawing area (blank by default).
    - text: A plain text display element (uses <span>).
    - url: A clickable hyperlink (uses <a>).
    - window: Opens a new blank browser window.
    - instance: Opens a new blank browser tab/instance.

2. add Command

- **Syntax 1 (Local Element):** add <elementType> [with id(<id>)] [content(<text>)] [links to(<url>)] [source(<filePath>)]
    - **Purpose:** Behaves identically to the create command when used for local element creation.
- **Syntax 2 (From Sourced Module):** add <elementType> from id(<remoteId>) source with id(<sourceAlias>)
    - **Purpose:** Adds a *clone* of an element that was defined in an imported JASPL module.
    - <elementType>: The type of element (e.g., button, checkbox). This is primarily for clarity and should match the remoteId's type.
    - from id(<remoteId>): The ID of the element *within the sourced module*.
    - source with id(<sourceAlias>): The alias assigned to the sourced module using the @source(...) with id(...) tag. This is **mandatory** when using from id().
    - **Rule:** from id() can **only** reference elements from imported modules. It cannot reference elements defined in the current program file.

3. add gap Command

- **Syntax:** add gap[(<pixels>)]
- **Purpose:** Adds a vertical spacing element (a line break or a specified number of pixels).
    - <pixels>: (Optional) The height of the gap in pixels. If omitted, a default line break height is used.

III. Styling Elements

JASPL provides a style command to customize the appearance of your GUI elements.

1. The style Command

- **Syntax 1 (Attribute-based):** style <elementid> (<attribute>=<value>, <attribute>=<value>, ...)

    - **Purpose:** Applies one or more style attributes to the element identified by <elementid>.
    - **Requirement:** The @jaspl-style tag must be present in your program's header.
- **Syntax 2 (CSS-mode):** style <elementid> css-mode (<rawCssString>)
    - **Purpose:** Allows direct application of raw CSS properties to an element. Use with caution, as it bypasses JASPL's structured styling.
    - **Requirement:** The @jaspl-style tag must be present.
- **Supported Style Attributes (Attribute-based styling):**
    - color: Sets the text color (e.g., red, #FF0000).
    - bgcolor: Sets the background color (e.g., blue, #0000FF).
    - border-thickness: Sets the border width in pixels (e.g., 2px). Requires border-color to be visible.
    - border-color: Sets the border color.
    - font: Sets the font family. Can be a standard font name (e.g., Arial) or a Google Fonts URL (e.g., https://fonts.googleapis.com/css2?family=Roboto:wght@400&display=swap).
    - font-style: Sets font characteristics. Can be bold, italic, strikethrough, monospaced, or none to reset. Multiple values can be combined (e.g., bold italic).
    - animation: Applies a predefined CSS animation.
        - **Supported Animations:** fade-in, fade-out, zoom-in, zoom-out, expand, shrink, shake.
        - **Syntax:** animation=<animation-name>[(<duration_in_seconds>)] (e.g., animation=fade-in(2)).

2. content() as a Style Attribute

- **Syntax:** style <elementid> (content=<newText>)
- **Purpose:** Dynamically changes the displayed text content of an element.
    - For text, button, url elements, it changes their textContent.
    - For radiobutton, checkbox, slider, uploadbutton, it changes the text of their associated <label>.
    - For textbox, it changes its placeholder text.

IV. Interactivity (Logic Blocks)

JASPL allows you to define actions that occur when specific events happen on an element using when...do blocks.

1. The when...do Block

- **Syntax:**
  ```
  when <elementid> <action> do (
     // JASPL commands to execute on event
  )
  ```
- **Purpose:** Attaches an event listener to an element. When the specified <action> occurs on <elementid>, the JASPL commands within the do() block are executed.
- **<elementid>:** The ID of the element to which the event listener is attached.
- **<action> (Supported Event Types):**
  - clicked: Triggers on a single mouse click or touch tap.
  - hovered: Triggers when the mouse pointer enters the element's area.
  - double-clicked: Triggers on a double mouse click.
  - hold: Triggers when a mouse button is pressed down on the element.
  - release: Triggers when a mouse button is released over the element.
  - selected: Primarily for input elements (textbox, radiobutton, checkbox, slider, uploadbutton). Triggers when the element's value changes (e.g., a checkbox is checked, a slider value is adjusted, text is entered in a textbox and focus is lost). A warning is issued if used on non-input elements.
- **do(...) Block:** Contains one or more standard JASPL commands (create, add, style, add gap, links to, source). These commands execute dynamically when the event occurs.
- **Rule:** do() blocks can only contain JASPL commands. Raw JavaScript is not supported directly within do() blocks for security and language consistency. Nested when...do blocks are not allowed.

V. Key Concepts and Rules

- **Plain by Default:** All elements created in JASPL are designed to appear plain (minimal styling) until explicitly styled using the style command. This ensures full control over aesthetics.
- **Unique IDs:** Element IDs must be unique within the entire application context (including elements from sourced modules). While auto-generated IDs prevent direct conflicts, using duplicate explicit IDs will result in warnings and potential unexpected behavior.
- **URL Validation:** URLs provided to links to() commands are validated. Invalid URLs will result in console errors.
- **Simulated File System:** The JASPL IDE operates on a simulated file system for @source imports. In a real-world scenario, these paths would resolve to actual files.
- **No alert() or confirm():** These browser functions are not supported in the JASPL

environment.
- **Error Reporting:** The console provides detailed error and warning messages, including line numbers and context, to assist with debugging.

VI. Example JASPL Code

```
@jaspl
@jaspl-style
@source(C:/code/program2.jaspl) with id(p2_module)
@source(C:/Files/program3.jaspl) with id(p3_styles)
start
create text with id(mainText) content(Welcome to JASPL Main Program!)
style mainText (color=#FF5733,
font=https://fonts.googleapis.com/css2?family=Roboto:wght@400&display=swap,
font-style=bold, animation=fade-in(2))
add gap(15)

create button with id(clickMeButton) content(Click Me!)
style clickMeButton (bgcolor=lightblue, color=blue, border-thickness=2px,
border-color=darkblue)
when clickMeButton clicked do (
   style clickMeButton (content=Clicked!, bgcolor=red, color=white)
   create text with id(dynamicText) content(Button was clicked!)
   style dynamicText (color=purple, font-style=italic)
   add gap
)
when clickMeButton hovered do (
   style clickMeButton (bgcolor=darkblue, color=yellow)
)
when clickMeButton release do (
   style clickMeButton (bgcolor=lightblue, color=blue)
)
add gap

add button from id(p2Button) source with id(p2_module)
style p2Button (bgcolor=gray, color=white, font-style=bold) // Styling a cloned
element
add gap

add checkbox from id(chkbox) source with id(p2_module)
```

```
when chkbox selected do (
   create text with id(chkStatus) content(Checkbox state changed!)
   style chkStatus (color=orange)
   add gap(5)
)
add gap

create button with id(runP3Button) content(Run Program 3)
source(C:/Files/program3.jaspl)
style runP3Button (bgcolor=teal, color=white, font-style=bold)
add gap

create checkbox with id(acceptTerms) content(I accept the terms and conditions)
add gap

create radiobutton with id(optionX) content(Option X)
create radiobutton with id(optionY) content(Option Y)
add gap(30)

create textbox with id(yourName) content(Enter your name here)
end
```