* Fix the HTML / JS /CSS boilerplate entirely
  + Have it read "visualizer.js" from current directory
  + When we export the visualizer, we make a folder, dump the HTML into it and create visualizer.js
    - Look at this: [https://stackoverflow.com/questions/5892845/how-to-load-one-javascript-file-from-another](https://gcc02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F5892845%2Fhow-to-load-one-javascript-file-from-another&data=04%7C01%7Cjeremy.raw%40dot.gov%7C6ebe9826f94842d34ef008d99bab9a34%7Cc4cd245b44f04395a1aa3848d258f78b%7C0%7C0%7C637711983612857496%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0&sdata=L0X9Nr0ahIWmu2sncZx7FRUIzKRXqLrKDdNBCMBXxI0%3D&reserved=0)
    - Look at this: [https://stackoverflow.com/questions/15521343/conditionally-load-javascript-file](https://gcc02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F15521343%2Fconditionally-load-javascript-file&data=04%7C01%7Cjeremy.raw%40dot.gov%7C6ebe9826f94842d34ef008d99bab9a34%7Cc4cd245b44f04395a1aa3848d258f78b%7C0%7C0%7C637711983612867454%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0&sdata=9xEVROkZdOKgijP9CsgWDcT8yO60tiexH7vjgou6%2FVo%3D&reserved=0)
    - Essentially, up in the <head> section we have inline Javascript to execute that injects a Script tag at the end of the <body> if "Visualizer.js" exists in the .js directory.
  + Easier: when the Visualizer HTML runs, it tries to load "js/visualizer.js"
    - If not found, it does nothing and we get the empty template (plus a console message saying "no visualizer.js"
    - Look at this: [https://javascript.info/onload-onerror](https://gcc02.safelinks.protection.outlook.com/?url=https%3A%2F%2Fjavascript.info%2Fonload-onerror&data=04%7C01%7Cjeremy.raw%40dot.gov%7C6ebe9826f94842d34ef008d99bab9a34%7Cc4cd245b44f04395a1aa3848d258f78b%7C0%7C0%7C637711983612867454%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0&sdata=TiLIseGFIeYSpbXdckHYmtGB8J9%2BCjINvW%2BSsgSOfAQ%3D&reserved=0)
  + The VisualVE script can be the same for both Category and Manual scenarios
    - See below - just configure one ScenarioGroup with each scenario a Level
    - Metrics still show the histogram of levels (and can be selected from there by range)
    - So it always gets loaded with the other JS
    - We load visualizer.js at the bottom of the page (with an onerror in case it doesn't exist)
  + Visualizer.js includes the following (and is placed in the js subdirectory of the visualizer HTML)
    - VEdata
    - scenarioconfig
    - categoryconfig
    - outputconfig
    - Suitable function call for VisualVE
      * (**$(document).ready** in the file-based version)
    - JRC version generates the same stuff, but a direct function call instead of document ready
  + If we run the visualizer live (using JRC):
    - It just pushes the HTML/CSS/JS to the web browser (straight from the package boilerplate)
      * That's just one static root page load (open page in JRC structure)
    - We use a single SendCommand to send a pile of Javascript to the browser
      * Contains the JSON guts of "visualizer.js"
      * Includes suitable VisualVE function
      * Calls the VisualVE function as its final step (which will fill up the DOM with the controls etc)
    - The Javascript is just the stuff we would have written into visualizer.js
      * But the final function call will be a direct invocation of VisualVE
      * In the "physical file" version we wrap the function call in **$(document).ready**
    - So that should be an extremely simple JRC implementation!
    - Key thing is to have on.exit called at the end to clean up any visualizer
      * Have a stack/list of visualizers in the ve.runtime environment
      * The list is visited when the visioneval session ends (how to trap that in R?)
      * The list is garbage-collected when a new visualizier is created
      * Each visualizer in the list is closed (ClosePage) when the visioneval session ends
* For Scenarios defined by Category, if there is a StartFrom stage
  + That stage is treated as Level 0 for every category
    - So there will always be  Level 0 in each defined Category
    - That is the level where no change is made to the StartFrom scenario
    - I.e. the Files defined for the Category will be unaltered from the StartFrom
  + A Category is ignored if it has no levels beyond the StartFrom
  + If the Category is fixed for the visualizer
    - Only its Level 0 is used if there is a StartFrom for the Scenarios
    - Otherwise only its Level 1 is used
* Visualizing Manual Scenarios
  + There is only one ScenarioGroup input chart (and it takes up the width of the page)
  + Each Scenario is a Level (in the order of ModelStages)
    - The info button shows the Description for the Scenario
  + We just show the standard measure charts below as histograms
    - Some of that may be dissatisfying if the scenarios don't vary
  + Does not require different javascript
* QuerySpecification for a metric can also include a "Visualizer" section
  + Specifically, the Value-Axis Ticks, TickFormat or Y-Ticks
  + Also numeric format for the Value-Axis
  + Also various tweaks to how the metric shows up in the grid
  + Applied to X-axis if used on Category/Level scenarios
  + Applied to Y-axis if used on Manual/Folder-based scenarios
* Add a Visualizer "tag" to the scenario configuration file
  + Only used for scenarios (but could be defined at the top level of the model!)
  + Includes ScenarioGroup/Levels
* "Visualizer" tag for the scenario configuration
  + Within "CategorySettings" tag (along with "StartFrom") with "Scenarios"
  + If "Visualizer" tag is not present we default when "visualize" is called:
    - Category/Level from scenario definition used for top box
    - Maximum number of Categories allowed (modest number like 8)
      * Will use first N Categories defined for Scenario
      * Subsequent Categories will be fixed (nailed to their first level)
    - Maximum number of Metrics allowed (modes number like 8)
      * Only the first N metrics will be used
  + Warnings
    - Report warning if any Category is fixed
      * Regardless of whether it is fixed by Setting or Default
    - Report warning if any Metric is left out due to MaxMetrics
    - Each Category maps to an equivalent ScenarioGroup
    - ScenarioGroup defined but not used because it's incomplete
  + Tags within Visualizer:
    - MaxScenarioGroups = 8
    - MaxMetrics = 8
    - OutputMetrics = list of metric names to select from the query
      * They will be displayed in this orderds
      * Default is all metrics, displayed in order of definition in the QuerySpec
    - FixedCategories = list of Category/Level (exactly one entry per Category)
      * If defined, only scenarios with that Category/Level will be included in the Visualizer
      * Any ScenarioGroup picking levels from that Category will be dropped
    - ScenarioGroups = list of ScenarioGroups to include; others will be fixed at their first level
      * If no Scenario Groups explicitly defined, each Category becomes a ScenarioGroup
      * Definition includes Name/Label, description header (see original visualizer)
  + ScenarioGroups validation
    - Any Category defined but not appearing at all will be fixed at its first level
    - Any Category with missing levels will ignore those levels (category treated as only having two levels
    - Only ScenarioGroups without fixed Categories will be visualized