ASSUMPTIONS FOR THIS CHECKLIST

1. Checklist aims to make producing accessible applications automatic and repairing applications painless
2. The checklist addresses only the Sakai 2.x world. Sakai 3 is uncharted territory and needs/deserves its own documentation and processes
3. There are possible contexts:
    a. Design exists, developer is carrying it out or
    b. Developer is involved in repair
4. The checklist can be very Sakai specific

GENERAL CONSIDERATIONS

1. Most designs are amenable to be expressed by standard Sakai idioms.
2. Standard Sakai idioms provide consistency and accessible design.
3. Occasionally there is no idiom or precedent - what to do then

So - to create accessible applications in the Sakai 2.x world, it is crucial to present to a developer the following:

1. an easy way to understand the standard structure of a Sakai page and a Sakai interaction
2. an easy way to translate a given design element to a given Sakai UI element (markup, style)
3. an easy way to know how to do scripted behaviors right
4. an easy and succinct set of guidelines to follow when there is no pre-existing Sakai UI idiom.

1. STANDARD SCREEN GENRES

There a re limited number of screen genres. Following the conventions makes the presentation more homogenous, more logical, and more accessible. A list of these genres will be provided.

For each genre:

- Description
- List of required and recommended elements depending on design
- Examples
- Issues regarding rendering techne s (Velocity, JSF, RSF, JSP) and workarounds

Example: Item structure

Description

When a single item is being displayed on the page – ie. An assignment, an announcement, etc.

Elements

Title

- Content “Item type” (required) + “item subject/identifier” (optional) – ie: “Announcement: Exam 2 dates moved”
- Format - <h3> Item type subject/identifier</h3>

Item metadata

Content: whatever metadata is important

Format:

```
<table class=”itemSummary” summary=””>
    <tr>
        <th>Metadata name<./th>
        <td>Metadata value</td>
    </tr>
    . . . . . .
</table>
```

Item content

Content: whatever the content is – if content is sectioned (bit of text, a set of attachments, some comments, each section has a descriptive header <h4>)

Simple Format:

```
<div class=textPanel>Content</div>
```

Complex format:

```
<h4>Subsection header</h4>
<div class=textPanel>Content</div>
<h4>Subsection header</h4>
<div class=textPanel>Content</div>
```

Item actions

Content: whatever actions the user can take on item

Format:

```
<p class=”act”>
    <input />
    <input />
    <input />
</p>
```

Rendering techs issues

JSF: cannot programmatically create a table with vertical table headers (see item metadata above). Recommend the use of the <f:verbatim> tag to create the structures needed.

2. STANDARD SAKAI IDIOMS

A given screen is composed of a set of standard UI idioms. A list of these idioms will be provided.

For each idiom:

- Format, required, recommended attributes
- Rendering techne (Velocity, JSF, RSF, JSP) and workarounds

Example: Form element, radio / checkbox group

Format:

```
<ul>
    <li class=”checkbox”>
        <input id=”id1” />
        <label for =”id1” />Descriptive label 1</label>
```

```
        </li>
        . . . . . .
</ul>
```

Notes: if this group is part of a complex form and constitutes a coherent subsection wrap construct in

```
<fieldset>
        <legend>Title for this subsection</legend>
                . . . . . .
</fieldset>
```

Rendering techs issues

All – the match between the input and label is required. Ids area alphanumeric, no spaces. If the list is being programmatically generated you can use a counter to generate the ids ("radio1, radio2") in Velocity and JSP.
RSF: the label and input (for and id) match is done via a UILabelTargetDecorator in the producer.
JSF: The standard <h:selectOneListbox /> produces a substandard result (labels wrap the inputs, the inputs have no ids, the whole thing is wrapped in a table, etc.) – seek alternatives.

A classification and treatment of these idioms would list and treat among others:

- Content presentation
    - Links
    - Lists
    - Multiple attribute lists (aka tables)
    - Graphics
    - Hierarchy (headings)
    - Etc, etc.
- Forms
    - Form elements
    - Form element combinations
    - Form structures

3. DYNAMIC PAGE COMPONENTS

- Make functions work with keyboard only/no mouse! (onclick for actions, onfocus for highlighting)
- Initiate functions by space bar or enter (not onkeypress, onfocus, or onchange)
- Move between items using tab, move within items using arrows
- Confirm when an action has been successful or failed (return focus to next location after confirmation of success or notification of failure)
- Inform user when action will cause window to open

4. GENERAL GUIDELINES TO FOLLOW WHEN PRE-EXISTING IDIOMS DO NOT EXIST

TBD