ExtJS Theming Workshop

Zusammenfassung

Inhaltsverzeichnis

1. Neues Theme anlegen und zuweisen

Im ersten Schritt erstellen wir ein neues Theme, wählen ein Basis-Theme aus, auf dessen Grundlage wir arbeiten und weisen der Anwendung das neue Theme zu. Anschließend starten wir die Anwendung.

Dazu gehe bitte folgendermaßen vor:

- Öffne im Root-Verzeichnis der Anwendung eine Konsole
- Erstellen ein neues Theme mit dem Befehl

```
sencha generate theme name-of-your-theme
```

- In der Datei „app.json“ im Eintrag „builds“ musst Du das neue Theme der Anwendung zuweisen
- Die Anwendung mit dem „classic“ Toolkit startest Du über

```
sencha app watch –fashion classic
```

- Für das „modern“ Toolkit benutze

```
sencha app watch –fashion modern
```

- Öffne die Anwendung unter: http://localhost:1841/
- Hinweis: um das Basis-Theme zu ändern, auf dem dein Theme aufbaut, gehe bitte in dein Theme-Verzeichnis unter „/package/local/ *name-of-your-theme*“ und ändern den Eintrag „extend“ in der Datei „package.json“. Welche Themes Dir zu Verfügung stehen, entnimmst Du bitte der Dokumentation Configuring theme inheritance
- Hinweis: Um getrennte Style für das „classic toolkit“ und das „modern toolkit“ zu erstellen, musst Du in der „package.json“ den Eintrag „sass“ um folgende Zeilen erweitern:

```
"sass": {
    "etc": [
        …
        "${package.dir}/${toolkit.name}/sass/etc/all.scss"
    ],
    "var": [
        …
        "${package.dir}/${toolkit.name}/sass/var"
    ],
    "src": [
        …
        "${package.dir}/${toolkit.name}/sass/src"
    ]
}
```

Anschließend musst Du noch folgende Ordner in dem Basis-Verzeichnis deines Themes anlegen.:

- classic
  - sass
    - etc
    - src
    - var
- modern
  - sass
    - etc
    - src
    - var

2. Styling über Variablen

Im zweiten Schritt definieren wir Variablen, um das globale Aussehen und das Aussehen der Komponenten anzupassen. Alle Variablen liegen im Verzeichnis: „sass/var/“.

- Globale und eigene Variablen werden in der Datei „var/Component.scss“ definiert. Eine Auflistung aller verfügbaren Variablen findest Du unter:
  - Classic Toolkit: https://docs.sencha.com/extjs/6.7.0/classic/Global_CSS.html
  - Modern Toolkit: https://docs.sencha.com/extjs/6.7.0/modern/Global_CSS.html
- Beim Anlegen eigener Variablen muss „dynamic()“ verwendet werden.
  - Beispiel: $custom-color: dynamic(#00ff00);
- Variablen einer Komponente müssen in einer Ordnerstruktur abgelegt werden, die der Klassenstruktur der Komponente entspricht. So müssen zum Beispiel Variablen für ein Panel in „var/panel/Panel.scss“ abgelegt werden.
- Eine Auflistung der verfügbaren Variablen steht in der Dokumentation der jeweiligen Komponente unter „theme variables“. (Für Panels zum Beispiel unter: https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#vars)

3. Styling über UIs

Im dritten Schritt setzen wir UIs ein, um identischen Komponenten ein unterschiedliches Aussehen zu geben. Alle UIs liegen im Verzeichnis: „sass/src/“.

- UI-Mixins müssen in einer Ordnerstruktur abgelegt werden, die der Klassenstruktur der Komponente entspricht. So müssen zum Beispiel UIs für ein Panel in „src/panel/Panel.scss“ abgelegt werden.
- Eine Auflistung der verfügbaren UI-Mixins steht in der Dokumentation der jeweiligen Komponente unter „theme mixins“. (Für Panels zum Beispiel unter: https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#sass-mixins)
- Der Komponente die Eigenschaft „ui: ‚ui-name‘ “ zuweisen

4. Styling über eigene CSS-Klassen

Im letzten Schritt verwenden wir klassisches CSS um vorhandene Stile zu überschreiben oder zu ergänzen. Alle CSS-Stile und Mixins, die keinen Bezug zu ExtJs-Komponenten haben liegen im Verzeichnis: „sass/etc/“.

- Der Komponente einen eigenen CSS-Klassennamen zuweisen
  - Zum Beispiel über die Eigenschaft „cls: ‚cls-name‘“
  - Weitere Eigenschaften wären zum Beispiel „bodyCls“ oder „overCls“
  - Eine vollständige Liste der Eigenschaften zum Setzen von CSS-Klassen entnimmst Du bitte der Dokumentation. (Für Panels zum Beispiel unter: https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#configs)
- Diese CSS-Klasse kannst Du direkt in der Datei „etc/all.scss“ ansprechen und ihr über herkömmliches CSS das gewünschtes Aussehen zuweisen
- Hinweis: Um die Übersichtlichkeit der zu erhöhen, kann man eigene SCSS-Datei im Verzeichnis „etc“ anlegen und in die Datei „all.scss“ importieren

5. Styling über Sencha Themer

Eine Alternative zum manuellen Stylen der Anwendung ist der „Sencha Themer“. Er ist ein eigenständiges Produkt von Sencha zum Stylen von ExtJS 6 Anwendungen (ab 6.0.1).

Was wird benötigt:

- Windows (Win 7+) -oder- Mac OS X -oder- Linux 32-bit/64-bit
- Sencha Cmd 6.7 oder höher (https://www.sencha.com/products/sencha-cmd/)
- Sencha Themer Lizenz (enthalten im Enterprise, Pro und Premium Package oder 30-Tage Testversion https://www.sencha.com/products/themer/)
- ExtJS 6.0.1+ Applikation

Über den Themer lässt sich ein neues Theme anlegen oder ein bereits bestehendes öffnen. Der Themer bietet die Möglichkeit eine Live-Ansicht der Applikation über Fashion im Browser auszugeben.

- Dazu muss in der app.json folgender Code-Block unter „js“ einfügt werden:

```
{
    "path": "http://localhost:8900/resources/themer/js/themerInspect.js",
    "remote": true,
    "bootstrap": true
}
```

- Du startest die Anwendung über:

```
sencha app watch -fashion
```

- Die Anwendung kannst Du über den Standard-Port aufrufen: http://localhost:1841
- Der Themer verbindet sich automatisch mit der Applikation. Jede Änderung im Themer wird jetzt live im Browser angezeigt.

Es können Standard-Komponenten ausgewählt und die SASS-Variablen über die UI angepasst werden. Über das „inspect tool“ () kann in der Beispiel-Ansicht, sowie in der Applikation im Browser eine Komponente selektiert und direkt bearbeitet werden. Zusätzlich können UI’s für bestimmte Komponenten erstellt werden.

Der Themer deckt das Styling über Variablen und UI’s ab. Über eigene CSS-Klassen kann im Themer **nicht** gestylt werden. Diese Styles werden in der Beispiel-Ansicht auch nicht berücksichtigt, können aber wie gewohnt genutzt werden.

6. BEST PRACTICE

Da Sencha ExtJS alle Komponenten über die CSS-Eigenschaft „position: absolute“ positioniert, ist es sehr schwer nachträglich noch CSS-Eigenschaften wie Abstände, Höhe und Weite zu beeinflussen.

Daher hat es sich als sinnvoll herausgestellt bestimmte Eigenschaften direkt beim Schreiben der Komponente zu setzen und nicht den Weg über CSS zu gehen.

Dies betrifft vor allem folgende Eigenschaften der Komponente (am Beispiel eines Panels):

- Marging (Der Außenabstand)
  https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#cfg-margin
- Padding (Der Innenabstand)
  https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#cfg-padding
- Border (Der Rahmen)
  https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#cfg-border
- Width (Die Weite)
  https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#cfg-width
- Height (Die Höhe)
  https://docs.sencha.com/extjs/6.7.0/classic/Ext.panel.Panel.html#cfg-height

7. Eigene Schriften/Icons einbinden

1. Font erstellen und in das Theme einbinden

- Icon-Fonts können z.B. über icomoon.io erstellt und heruntergeladen werden. Schrift-Fonts finden Sie zum Beispiel hier: fontsquirrel.com oder fonts.google.com
- Font-Dateien in das Verzeichnis „resourcen“ des Themes kopieren. (Sie können Unterordner anlegen, z.B. „resources/font“ oder „resources/icon-font“)
- Datei „all.scss“ im Verzeichnis „sass/etc“ öffnen und Font über die CSS-Regel „@font-face“ einbinden (Pfadangaben sind relativ zum Verzeichnis „resources“)
- Beispiel-Code für den Icon-Font:

```
@font-face {
  font-family: 'name-des-icon-fonts;
  src:
    url('pfad/name-des-icon-fonts.woff') format('woff'),
    … weitere Schriftformate
  font-weight: normal;
  font-style: normal;
}
[class^="icon-"], [class*=" icon-"] {
  font-family: 'name-des-icon-fonts' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: 1;
}
.icon-1:before {
  content: "\e9df";
}
.icon-2:before {
  content: "\e9e0";
}
… weitere Icons
```

- Beispiel für den Schrift-Font:

```
@font-face {
  font-family: 'name-des-schrift-fonts';
  src:
    url('pfad/name-des-schrift-fonts.ttf') format('truetype'),
    url('pfad/name-des-schrift-fonts.woff') format('woff'),
    url('pfad/name-des-schrift-fonts.svg') format('svg');
    … weitere Schriftformate
  font-weight: normal;
  font-style: normal;
}
```

2. Icon-Font anwenden

- Der ExtJS-Komponente die Config „iconCls“ geben
- Als Wert die CSS-Klasse des gewünschten Icons eintragen
- Beispiel:

```
iconCls: ' icon-1'
```

3. Schrift-Font anwenden

- Schrift-Font kann jedem Element über eine CSS-Klasse hinzugefügt werden
- Beispiel:

```
.foo{
  font-family: 'name-des-schrift-fonts';
}
```

- Alternativ kann der Schrift-Font auch über ExtJs Variablen hinzugefügt werden
- Beispiele:

```
// Globale Variablen:
$font-family: 'name-des-schrift-fonts'

// Ext.panel.Panel Variablen
$panel-body-font-family: 'name-des-schrift-fonts'
```

8. Nützliche Links

- https://docs.sencha.com/extjs/6.7.0/guides/core_concepts/theming.html
- https://docs.sencha.com/extjs/6.7.0/classic/Ext.html
- https://docs.sencha.com/themer/1.3.5/guides/installation_setup.html
- https://docs.sencha.com/extjs/6.7.0/index.html