verändert wurde, und speichert die Dateien im Backup-Verzeichnis in der Form *Kopierdatum_Änderungsdatum__Name.indd*.

`_dok.save();`
14 Zuletzt muss noch das aktuelle Dokument gespeichert werden. Dies geschieht mit der Methode `save()`.

Im nächsten Kapitel zeige ich, wie man einen Menü-Eintrag für das Skript erstellt und es beim Starten von InDesign automatisch aktivieren kann.

11.13 Eigene Einträge im Menü erstellen

In InDesign kann man eigene Menü-Einträge für Skripte erstellen. So können Sie Ihre Skripte in die InDesign-Menü-Struktur einhängen und müssen sie nicht mehr über das Bedienfeld Skripten aufrufen. Wenn man dann noch die Möglichkeit nutzt, Skripte direkt beim Start von InDesign zu laden, wird die Bedienung der Skripte deutlich angenehmer.

Gerade wenn Sie Skripte für andere Benutzer entwickeln, sind diese Techniken wichtig. Es hat sich gezeigt, dass Akzeptanz und Benutzung von Skripten in InDesign deutlich zunehmen, wenn sie in der normalen Menü-Struktur verankert sind.

11.13.1 Menü-Einträge

Als Beispiel zeige ich, wie man das Skript zum Erstellen eines Backups aus dem vorherigen Unterkapitel in das Datei-Menü hinter den Eintrag Speichern einhängen kann.

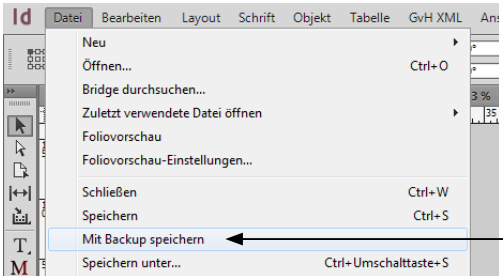

Abb. 101
Datei-Menü

Listing 124
Menü-Eintrag erstellen
11-13_MenuEintrag-1.jsx

```
1  #targetengine "saveWithBackupSession"
2  var _action = app.scriptMenuActions.add("Mit Backup speichern");
3  _action.eventListeners.add("onInvoke", saveWithBackup);
4  var _hm = app.menus.itemByName("$ID/Main");
5  var _dm = _hm.submenus.itemByName("$ID/FileDestinationPanel");
6  if (_dm.menuItems.itemByName("Mit Backup speichern") == null) {
7    _dm.menuItems.add(_action, LocationOptions.AFTER, _dm.menuItems.
     itemByName("$ID/Save"));
8  }
```

```
#targetengine "saveWithBackupSession"
```

Session anlegen

1 Mit der Festlegung einer eigenen Session (→ Unterkapitel 7.17) stellt man sicher, dass das Skript bis zum Beenden von InDesign erreichbar ist. Variablen und eigene Funktionen sind nur innerhalb einer Session gültig und können deswegen nicht von anderen Skripten beeinflusst werden. Sobald das Skript einmal ausgeführt wurde, erscheint die Session `saveWithBackupSession` in der zweiten Dropdown-Liste des Debug-Bedienfelds vom ESTK. Bei Bedarf könnte man in einem anderen Skript, das in der gleichen Session ausgeführt wird, auf die Funktion `saveWithBackup()` zugreifen.

```
var _action = app.scriptMenuActions.add("Mit Backup speichern");
_action.eventListeners.add("onInvoke", saveWithBackup);
```

Menü-Befehl erstellen

2+3 Zunächst wird ein Objekt vom Typ `ScriptMenuAction`, das zum Objekt `Application` gehört, erstellt. Der Parameter der Methode `add()` übernimmt den Namen des Menü-Eintrags.

Dann muss dem Objekt noch ein `EventListener` hinzugefügt werden. Dieser soll immer dann, wenn der Menü-Eintrag ausgewählt wird, reagieren. Bei der Erstellung werden zwei Parameter übergeben: Der erste beschreibt das Event, der zweite, welche Funktion beim Eintreten des Events ausgeführt werden soll (zu Events siehe Unterkapitel 7.18). Das Event `onInvoke` tritt ein, wenn der Menü-Eintrag ausgewählt wird. Die Funktion `saveWithBackup()` ist weiter unten im Skript enthalten und entspricht weitgehend der Funktion aus dem vorherigen Unterkapitel.

Jetzt ist ein fertiger Menü-Eintrag erstellt, der allerdings noch nicht in ein Menü eingehangen wurde.

```
var _hm = app.menus.itemByName("$ID/Main");
```

4 In der Variablen `_hm` wird eine Referenz auf das Hauptmenü gespeichert. Das Hauptmenü erreicht man über die Sammlung `menus`. Die Menüs werden über die sprachunabhängigen Namen (engl. *locale independent string*) angesprochen. Wenn das Skript nur unter der deutschen Version von InDesign laufen würde, könnte man auch die Namen aus der Benutzeroberfläche übernehmen.

Menü-Einträge ansprechen

Generell sollten Menü-Einträge immer mit den sprachunabhängigen Namen und Menü-Befehle über ihre `ID` angesprochen werden. Da es keine Liste für diese Bezeichnungen gibt, kann man mit dem Skript *11-13_AlleMenus.jsx* eine Textdatei mit den Namen und IDs aller Menüs und Befehle erstellen (die Laufzeit kann mehrere Minuten betragen). Leider werden einige wenige Namen nicht korrekt aufgelöst, andere Menü-Einträge haben mehrere Entsprechungen. Außerdem haben sich einige Bezeichnungen in den verschiedenen Versionen geändert. Hier muss man im Zweifel etwas experimentieren.

```
var _dm = _hm.submenus.itemByName("$ID/FileDestinationPanel");
```

5 Mit Hilfe der Sammlung `submenus`, die die Untermenüs DATEI, BEARBEITEN etc. enthält, wird eine Referenz auf das Datei-Menü, das auf die sprachunabhängige Bezeichnung `$ID/FileDestinationPanel` hört, in der Variablen `_dm` gespeichert werden.

Menü-Eintrag erstellen

```
if (_dm.menuItems.itemByName("Mit Backup speichern") == null) {
   _dm.menuItems.add(_action, LocationOptions.AFTER, _dm.menuItems.
   itemByName("$ID/Save"));
}
```

6–8 Zunächst wird geprüft, ob der Eintrag MIT BACKUP SPEICHERN bereits vorhanden ist. Wenn dies nicht der Fall ist, wird der Menü-Eintrag erstellt. Dazu wird dem Datei-Menü ein neues Objekt vom Typ `MenuItem` hinzugefügt. Als erster Parameter wird die `ScriptMenuAction`, die vorher in der Variablen `_action` gespeichert wurde, übergeben. Der zweite Parameter legt die Position des neuen Elements in Bezug zum dritten Parameter – einem Menü-Eintrag – fest. Der neue Menü-Eintrag erscheint unter dem Eintrag SPEICHERN.

Menü-Eintrag kontextabhängig einblenden

Das Skript aus dem vorherigen Unterkapitel prüft zu Beginn, ob es sinnvoll ausführbar ist. Diese Prüfung kann man auch in eine eigene Funktion auslagern, die je nach Situation den Menü-Eintrag ein- bzw. ausblendet.

Listing 125 Menü-Eintrag kontextabhängig *11-13_Menu Eintrag-2.jsx*

```
2  var _action = app.scriptMenuActions.add("Mit Backup speichern");
3  _action.eventListeners.add("onInvoke", saveWithBackup);
4  _action.eventListeners.add("beforeDisplay", canRun);
   //...
22 function canRun(_event) {
23   var _action = _event.parent;
24   if (app.documents.length > 0 && app.activeDocument.saved == true
     && app.activeDocument.modified == true) {
25     _action.enabled = true;
26   } else {
27     _action.enabled = false;
28   }
29 }
```

4 Für die Prüfung, ob der Eintrag angezeigt werden soll, kann man einen Eventlistener vom Typ `beforeDisplay` verwenden. Bevor der Eintrag angezeigt wird, wird dann immer die Funktion `canRun()` ausgeführt.

```
function canRun(_event) { //...
```

22–29 Der Funktion `canRun()` wird von InDesign automatisch das Event, von dem sie ausgelöst wurde, als Parameter übergeben. In Zeile 23 wird dann die eigentliche `ScriptMenuAction` ermittelt – sie ist das Elternelement des Events.

```
if (app.documents.length > 0 && app.activeDocument.saved == true
   && app.activeDocument.modified == true) {
```

24 Hier wird geprüft, ob die Funktion sinnvoll ausgeführt werden kann – die Prüfung ist im ursprünglichen Skript *11-12_SaveWith Backup-1.jsx* in der ersten Zeile durchgeführt worden. Wenn die Prüfung erfolgreich ist, wird die `ScriptMenuAction` mit der Eigenschaft `enabled` aktiviert, ansonsten deaktiviert.

Menü-Eintrag entfernen

Der Menü-Eintrag bleibt nur bis zu einem Neustart von InDesign erhalten. Wenn Sie den Eintrag vorher entfernen wollen, benötigen Sie das folgende Skript:

```
4  var _eintrag = _dm.menuItems.itemByName("Mit Backup speichern");
5  if (_eintrag != null) {
6    _eintrag.remove();
7  }
```

Listing 126
Menü-Eintrag entfernen
11-13_MenuEintrag Entfernen.jsx

Das Skript muss ebenfalls in der Session `saveWithBackupSession` laufen. In Zeile 2+3 wird wie im vorherigen Skript das Datei-Menü adressiert. Danach wird der Menü-Eintrag in `_eintrag` gespeichert und mit einer if-Abfrage geprüft, ob der Eintrag im Datei-Menü existiert. Wenn `_eintrag` dem Wert `null` entspricht, existiert kein Eintrag. Mit der Methode `remove()` kann der Eintrag gegebenenfalls entfernt werden.

Untermenüs bleiben erhalten.

Wenn Sie eigene Untermenüs (`Submenu`) erstellen, bleiben diese auch nach einem Neustart erhalten. Die folgende Zeile erstellt ein Untermenü Skripte am Ende der Menü-Leiste.

```
app.menus.itemByName("$ID/Main").submenus.add("Skripte");
```

Zum Entfernen des Menüs können Sie die folgende Zeile verwenden:

```
app.menus.itemByName("$ID/Main").submenus.itemByName("Skripte").remove();
```

11.13.2 Skripte beim Start von InDesign laden

Für eine gute Integration des Skripts in InDesign sollte das Skript automatisch beim Start geladen und der Menü-Eintrag erstellt werden.

Dazu muss das Skript in einen Ordner mit dem Namen *startup scripts* gelegt werden. Die Startskripte von InDesign, wie z. B. *URLs in Hyperlinks konvertieren*, liegen neben dem Ordner *Scripts Panel*. Ich empfehle, eigene Skripte auf der gleichen Ebene zu speichern. Für das Skript würde ich deswegen erst einen Ordner *SaveWithBackup* und in diesem dann den Unterordner *startup scripts* anlegen. Nachdem das Skript hierhin kopiert wurde, brauchen Sie nur noch InDesign neu zu starten.

Wenn man das Skript regelmäßig verwenden möchte, kann man es auf einen Tastaturbefehl legen (Bearbeiten → Tastaturbefehle … → Produktbereich Skripte).