JavaScript – Puzzle

Ziel der Übung:
Programmiere mit Hilfe von JavaScript ein Schiebe-Puzzle. Die genaue Spiel-Funktion des Puzzle ist mittels beiliegenden Videos visualisiert. Als Vorlage dient puzzle_vorlage.html sowie der nachfolgende Beschrieb.

Am Schluss habt ihr 2 Versionen einer HTML-Seite. Die erste mit einem anfangs sortierten Puzzle (puzzle_sortiert.html), die zweite ist am Anfang unsortiert (puzzle_unsortiert.html).

Die benötigten Bilder sind im Ordner „Resourcen“.
Bewertet wird die Funktionalität und nicht das Design .

Material:
- 8 Bilder
- 1 Blindbild
- puzzle_vorlage.html

Abgabe:
- Ordner mit “vorname_name_klasse_klassennummer_javascript” erstellen, in welchem sich die 2 HTML-Versionen sowie der Ordner “bilder” befindet.
- Im Ordner muss eine saubere Struktur vorhanden sein mit allen Dateien.
- Abgabeort: Common/Alle/Abgaben/Uebungen

Korrektur:
Mit dem Supi vor Ort. Der Supi wird dir bei erfolgreicher Bewertung eine physische Quittung aushändigen.

Abgabetermin:
Der Abgabetermin ist der Kursübersicht zu entnehmen!

Viel Spass beim Erstellen!

1. Aufbau und Struktur

Als erstes solltest du dir Gedanken zum Aufbau des Puzzles machen. Was für Funktionalitäten und welche Resourcen es braucht.

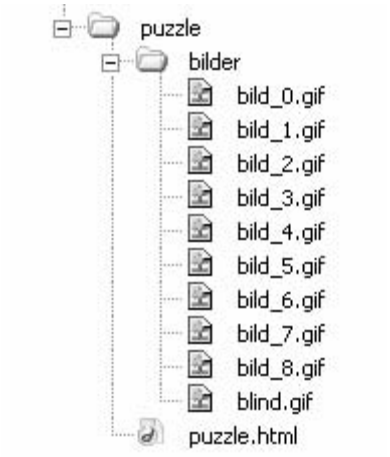
Es werden folgende Funktionalitäten benötigt:

- starten des Spiels
- verschieben der einzelnen Puzzleteile
- zählen der Klicks
- check ob Puzzle gelöst

Zudem die folgenden Resourcen:

- Html-Seite als Grundlage für Darstellung und JavaScript
- Bilder für Puzzleteile

Ebenfalls ist bei Web-Projekten die Ordnerstruktur von bedeutung. Am besten legst du die Bilder in einen Unterordner, damit nicht so ein Chaos ensteht. Das würde dann etwa so aussehen:



2. Ablauf

Bevor du mit der Programmierung der Funktionalitäten beginnen kannst, müssen die benötigten Resourcen erstellt werden.

Wenn alle Resourcen soweit wie nötig fertig sind, kannst du mit der Programmierung starten. Die Kern-Funktion des Puzzles ist das verschieben der einzelnen Teile. Aus diesem Grund ist es ratsam, sie als erstes zu schreiben, denn alle anderen Funktionalitäten sind von dieser abhängig.

Zum Schluss kann dann das Neustarten des Spiel's in Angriff genommen werden, da erst am Ende alle nötigen Daten dafür vorhanden sind.

3. Los geht's, die Bilder

Die Bilder für das Puzzle findest du im Ordner 'Resourcen'. Selbstverständlich können auch eigene Bilder verwendet werden. Du kannst ein Bild in 9 Teile zerschneiden und sie entsprechend der Vorlage benennen. Am einfachsten machst du das mit einem Slice-Tool wie Fireworks oder ImageReady. Benenne dann die einzelnen Dateien mit bild_0.gif, bild_1.gif bis bild_9.gif. Beachte, dass das erste Bild die Nummer 0 und nicht 1 bekommt. Dadurch kannst du nachher einfacher die Bilder mit einer JavaScript Schleife durchgehen.

Zudem wird ein Bild als Platzhalter für die Lücke, die zum Verschieben vorhanden sein muss, benötigt. Du kannst da das 'blind.gif' im Ordner Resourcen nehmen.

4. Die Html-Seite

Erstelle nun eine neue Html-Datei mit deinem Vor- und Nachnamen als Dateiname. Bitte kombiniere die zwei Wörter aber mit einem Unterstrich. Im Web sollten keine Leerzeichen in Dateinamen verwendet werden. Denke auch an die richtige Endung '.html' oder '.htm'.

Erstelle am Anfang der Seite eine Überschrift, z.B. 'JavaScript Puzzle'.

Jetzt können die Bilder im Body der Seite platziert werden. Am besten in einer Tabelle mit 3 Zeilen und je 3 Zellen. Wenn du beim Tabellenattribut 'cellspacing' eine 1 anstatt 0 nimmst, ensteht ein feiner Rahmen zwischen den Bildern.

Die Bilder sollten wie folgt platziert werden:

Die Bilder müssen relativ eingebunden werden, z.B. <img src="bilder/bild_5.gif" ..., da andernfalls Probleme beim Korrigieren entstehen!

Eines der Bilder sollte durch das 'blind.gif' aus dem Resourcen Ordner ausgewechselt werden, da sonst kein freier Platz zum verschieben bleibt.

Dann benötigt das Puzzle noch ein Textfeld, indem die aktuelle Anzahl Klicks angezeigt werden kann. Erstelle also unterhalb der Bilder ein Formular, und gib ihm einen Namen. Dann darin das Textfeld erstellen. Du kannst dem Feld mit 'value' schon mal den Wert 0 zuweisen. Schreibe auch noch einen Beschreibungstext dazu, damit ersichtlich ist, was dieses Feld bedeutet.

Und zum Schluss brauchst du noch einen Button um ein neues Spiel zu starten. Füge diesen ebenfalls in Formular ein. Kein Button vom Typ 'submit', sondern ein normaler 'button'.

Fertig! Zumindest der Teil des Designs...weiter geht's mit der programmierung der einzelnen Funktionalitäten.

5. Den Bildern Bewegung einhauchen

Dies ist die Kern-Funktionalität des Schiebe-Puzzles. Deshalb sollte sie auch zuerst erstellt werden. Ausserdem ist es auch die aufwendigste Sache des Spiels.

Was genau muss den Geschehen?

Beim Klick auf ein Bild muss sich dieses Verschieben, jedoch nur wenn es auch verschoben werden kann. Ein Bild kann dann verschoben werden, wenn es an die Lücke grenzt. Andernfalls ist es ja vollständig von andern Bildern umschlossen und kann nicht bewegt werden.

Das heisst in JavaScript, du brauchst eine Funktion, die beim Klick auf eines der Bilder aufgerufen wird. Die Funktion muss dann das angeklickte Bild verschieben, sofern es möglich ist. Damit diese Funktion das Bild verschieben kann, muss sie aber wissen, welches Bild denn angeklickt worden ist. Das erreichst du am einfachsten durch einen Parameter, auf den du einfach die Nummer des Bildes übergibst.

Nummer des Bildes? Genau, um die Bilder auf einer Html-Seite ansprechen zu können, brauchst du das 'image'-Objekt. Am besten schaust du gleich mal im SelfHTML nach, was das Objekt so kann. Alle Bilder können über dieses Objekt angesprochen werden. Dabei funktioniert das wie bei den Formularen oder den Formular-Elementen. Also, jedes Bild wird im Image-Objekt referenziert und kann gezielt angesprochen werden. Dies geschieht über einen Listenzugriff. Das erste Bild auf der Html-Seite von Oben nach Unten gesehen, ist das erste Bild in der Image-Liste und kann mit dem Index 0 angesprochen werden. Das zweite mit dem Index 1, das dritte mit dem Index 2 usw.

Das erste Bild auf einer Seite kann also so angesprochen werden:
document.images[0]

bzw. das zweite mit:
document.images[1]

Willst du den Pfad der Bilddatei auslesen, geht das so:

document.images[0].src

Hier der Link zum SelfHTML: http://selfhtml.teamone.de/javascript/objekte/images.htm

Erstelle also eine JavaScript Funktion, natürlich im Head-Bereich. Und gib ihr einen Parameter für die Bildnummer:

```
function gameMovePic(nummer) {
        ........
}
```

Diese Funktion soll dann beim Klick auf eines der Bilder aufgerufen werden. Du musst also auch auf allen Img-Tags, noch die Funktion anhängen. Stichwort Event's!

Der Parameter 'nummer' musst du für jedes Bild einzeln anpassen und richtig übergeben. Beim ersten Bild übergibst du den Wert 0, beim zweiten den Wert 1 und so weiter bis zum letzten Bild, das den Wert 8 bekommt.

```
<img src="bilder/bild_1.gif" onclick="gameMovePic(1)">
```

Tipp: Zum Testen machst du am besten ein alert(nummer); in die Funktion, so wird dir beim Klick auf ein Bild die Nummer, die in der Funktion ankommt, angezeigt.

Jetzt fehlt aber noch das verschieben bzw. der Inhalt der Funktion. Damit ein Bild verschoben werden kann, muss bekannt sein in welche Richtung verschoben werden soll. Nun kann jedes Bild in mehrere Richtungen verschoben werden, du musst also zuerst herausfinden ob eine Richtung überhaupt möglich ist.

Deshalb wird einfach jede Richtung nacheinander abgearbeitet und getestet. Schauen wir uns mal die Richtung nach Unten an. Nicht alle Bilder können nach Unten verschoben werden, also ist unsere erste Abfrage sicher einmal, ob das angeklickte Bild nach Unten verschoben werden könnte. Überlege dir welche Bilder nach Unten verschoben werden können. Du wirst merken, dass alle nach Unten verschoben werden können, ausser die Bilder der untersten Reihe. Das kannst du in JavaScript so umsetzen:

```
// check ob bilnummer theoretisch nach unten verschoben werden koennte
if (nummer == 0 || nummer == 1 || nummer == 2 || nummer == 3 || nummer == 4 || nummer == 5) {
        ....
}
```

Mit einer IF wird geprüft, ob die übergebene Bildnummer einer der Nummern ist, die nach unten bewegt werden könnten. (theoretisch)

Wenn eine der Bedingungen zutrifft, dann kann weiter geprüft werden, ob denn der Platz darunter auch wirklich frei ist. Denn nur dann kann das Bild wirklich verschoben werden.

Dazu muss aber wiederum die Nummer des Bildes, das darunter platziert ist, bekannt sein. Die Nummer des Bildes auf dem Platz darunter kannst du errechnen. Einfach die aktuellen Bildnummer + 3 und du hast die Nummer des Bildes darunter, denn es sind ja immer 3 in einer Reihe. Nun prüfst du ob das Bild auf dem Platz darunter das 'blind.gif' ist, wenn ja kann das Bild dann verschoben werden. Der check ob es das Blind gif ist, kannst du über die Eigenschaft 'src' des Image-Objekt's machen. Da jedoch nicht nur der Bildname, sondern der ganze Pfad im 'src' gespeichert werden, musst du es mit indexOf() prüfen.

```
// bildnummer des darunter liegenden bildes errechnen
var nummerUnten = nummer + 3;

// check ob bild darunter das blind.gif ist
if (document.images[nummerUnten].src.indexOf("blind.gif") != -1) {
	.....
}
```

Trifft diese Bedingung zu, ist der Platz darunter frei, da ja das 'blind.gif', also unsere Lücke, platziert ist. Und das angeklickte Bild kann verschoben werden.

Das verschieben des Bildes ist nun ein Leichtes, es muss nur der 'src' des angeklickten Bildes mit dem 'src' des darunter liegenden Bildes, also dem 'blind.gif' ausgetauscht werden und schon verschiebt sich das Bild.

```
// bildpfad des blind.gif in variable speichern und bild src austauschen
var blindSrc = document.images[nummerUnten].src;
document.images[nummerUnten].src = document.images[nummer].src;
document.images[nummer].src = blindSrc;
return;
```

Das 'return;' am Ende sorgt dafür, dass die Funktion nach dem verschieben sofort beendet wird und nicht noch die anderen Richtungen wie Oben, Links und Rechts ausführt.

That's it! Und so sollte es jetzt bei dir innerhalb der Funktion ausehen:

```
// check ob bilnummer theoretisch nach unten verschoben werden koennte
if (nummer == 0 || nummer == 1 || nummer == 2 || nummer == 3 || nummer == 4 || nummer == 5) {
	// bildnummer des darunter liegenden bildes errechnen
	var nummerUnten = nummer + 3;

	// check ob bild darunter das blind.gif ist
	if (document.images[nummerUnten].src.indexOf("blind.gif") != -1) {
		// bildpfad des blind.gif in variable speichern und bild src austauschen
		var blindSrc = document.images[nummerUnten].src;
		document.images[nummerUnten].src = document.images[nummer].src;
		document.images[nummer].src = blindSrc;
		return;
	}
}
```

Dasselbe muss jetzt für jede weitere Richtung gemacht werden. Ich werde die Richtung Oben noch aufzeigen, die anderen 2 Richtungen musst du dann selbst machen.

Wird also das Bild nicht nach unten verschoben, weil es nicht nach unten verschoben werden kann, dann kann die nächste Richtung getestet werden. Dies geschieht unterhalb der IF von der Richtung Unten aber immer noch innerhalb der selben Funktion.

```
// check ob bilnummer theoretisch nach oben verschoben werden koennte
if (nummer == 3 || nummer == 4 || nummer == 5 || nummer == 6 || nummer == 7 || nummer == 8) {
	// bildnummer des darueber liegenden bildes errechnen
	var nummerOben = nummer - 3;

	// check ob bild darueber das blind.gif ist
	if (document.images[nummerOben].src.indexOf("blind.gif") != -1) {
		// bildpfad des blind.gif in variable speichern und bild src austauschen
		var blindSrc = document.images[nummerOben].src;
		document.images[nummerOben].src = document.images[nummer].src;
		document.images[nummer].src = blindSrc;
		return;
	}
}
```

Wenn du bis hier hin alles richtig gemacht hast, dann solltest du die 2 Richtungen bereits testen können. Lade die Seite im Browser und klicke auf eines der Bilder.

Wenn nichts geht, Netscape öffen und die Datei laden, 1-2 Klicks machen und ‘javascript:’ eingeben.

Die anderen 2 Richtungen solltest du nun gut selbst einbauen können.

6. Klicks zählen

Der schwierigste Teil ist gemacht und nun müssen noch die Klicks beim Verschieben gezählt werden. Dazu brauchst du eine Variable, die bei jedem verschieben um 1 erhöht wird. Zudem sollte die Anzahl noch im Textfeld erscheinen.

Erstelle also eine globale Variable, am besten gleich am Anfang des Scriptbereichs.

```
// globale variable fuer die anzahl klicks
anzahlKlicks = 0;
```

Dann brauchst du noch die Funktion, die die Klicks inkrementiert und im Textfeld ausgibt.

```
// erhoeht die klicks und gibt sie im textfeld aus
function addClick() {
	anzahlKlicks += 1;
	document.deinFormularName.deinTextfeldName.value = anzahlKlicks;
}
```

Nun muss die Funktion addClick() natürlich noch irgendwo aufgerufen werden, damit die Klicks auch gezählt werden. Das machst du in der Funktion gameMovePic() von vorhin. Immer nach dem verschieben eines Bildes soll der Klick gezählt werden. Du musst also an 4 Stellen in der gameMovePic() Funktion die addClick() Funktion aufrufen.

7. Lösung prüfen

Es fehlt nun noch das Überprüfen der richtigen Anordnung der Bilder. Sind alle Bilder am richtigen Platz, ist das Puzzle gelöst. Hier kommen jetzt die Bildnamen ins Spiel. Wie du ja jetzt weisst, hat jedes Bild eine Indexnummer. Die Bilder sind auch nach diesen Nummern benannt. Also das Bild, das Oben Links steht hat die Nummer 0 und heisst demnach 'bild_0.gif'.

Anhand dieser Information kann jetzt geprüft werden, ob jedes Bild dann auch am richtigen platz sitzt.

```
// check ob puzzle geloest
function checkPuzzle() {
	// variable fuer speicherung der bildnummer des bild.gif's initialisieren
	blindIndex = -1;

	// mit schleife alle bilder durchgehen
	for (var i=0; i < document.images.length; i++) {
		// check ob das bild das blind.gif ist
		if (document.images[i].src.indexOf("blind.gif") != -1) {
			// wenn ja bildnummer des blind.gif speichern
			blindIndex = i;
		// andernfalls check ob bild am richtigen platz
		} else {
			// dem Index entsprechenden Dateinamen zusammenbauen
			var bildname = "bild_" + I + ".gif";
			if (document.images[i].src.indexOf(bildname) == -1) {
				// wenn bildname nicht gefunden an dieser index position
				// dann abbrechen, da das bild am falschen platz ist
				return;
			}
		}
	}

	// laeuft die schleife ohne fehler bis zum ende sind alle bilder am richtigen platz
	// anhand der gefundenen IndexNummer kann das blind.gif noch ersetzt werden
	if (blindIndex != -1) {
		// blind.gif tauschen und erfolgsmeldung ausgeben
		document.images[blindIndex].src = "bilder/bild_3.gif";
		alert("Bravo, du hast das Puzzle mit " + anzahlKlicks + " Klicks geloest!");
	}
}
```

Mit einer Schleife werden alle Bilder auf der Seite nacheinander geprüft. Ist das Bild das 'blind.gif' wird die Index-Nummer gespeichert, damit nachher das Bild ausgetauscht werden kann. Andernfalls wird der richtige Bildname für den jeweiligen Index zusammengesetzt und geprüft ob der 'src' des Bildes an dem Index auch wirklich den

Bildnamen enthält. Wenn ja, ist das Bild am richtigen Ort. Wenn nein wird die Funktion mit 'return;' beendet, da das Puzzle noch nicht gelöst sein kann.

Läuft die Schleife ohne Fehler bis zum Ende, sind alle Bilder am richtigen Platz und das Puzzle wurde gelöst. Nun kannst du noch das blind.gif durch das richtige Bild ersetzen und eine Alert -Meldung ausgeben, die anzeigt, wieviel Klicks es gebraucht hat.

Aufrufen kannst du diese Funktion, innerhalb der addClick() Funktion, dann musst du es nur 1 x anstatt 4 x angeben. Denn nach jedem erfolgreichem Verschieben muss die Überprüfung stattfinden.

8. Neues Spiel starten

Zum Abschluss muss noch der 'Neues Spiel starten' Button mit Funktionalität ausgerüstet werden. Beim Start eines neuen Spiels muss die Klickzahl auf 0 gesetzt werden und die Bilder müssen neu gemischt werden.

Da das Mischen aber nur mit Array's (Listen) sinnvoll machbar ist, lassen wir das mal. Das Spiel wird schon durchmischt angezeigt, wenn es geladen wird. Dies kannst du erreichen, indem du jetzt die Bilder nicht in der richtigen reihenfolge im Html-Document einfügst. Also vertausche einfach einige Bilder.

Achtung: speichere dein Html-File 2 x mal ab. 1 x mit den Bildern in der richtigen Reihenfolge und 1 x mal mit der durchmischten. Es wäre unmöglich alle Übungen mit durchmischten Bildern zu korrigieren. Gebt also auch beide Dateien ab!

Jetzt fehlt nur noch das zurücksetzen des Klickzählers. Da die Bilder ja beim laden bzw. neuladen gemischt angezeigt werden, ist das laden der Seite auch ein guter Zeitpunnkt um den Zähler auf 0 zu setzen. Also schreibst du noch folgende letzte Funktion:

```
// startet das spiel
function gameStart() {
        // anzahl klicks auf 0 setzen und ins Textfeld schreiben
        anzahlKlicks = 0;
        document.deinFormularName.deinTextfeldName.value = anzahlKlicks;
}
```

Aufrufen kannst du diese Funktion im Body-Tag mit dem Event 'onload'.

Nun musst du nur noch ein onclick="location.href=location.href" auf den Button 'Neues Spiel starten' zu machen und das Puzzle ist.....

...fertig.