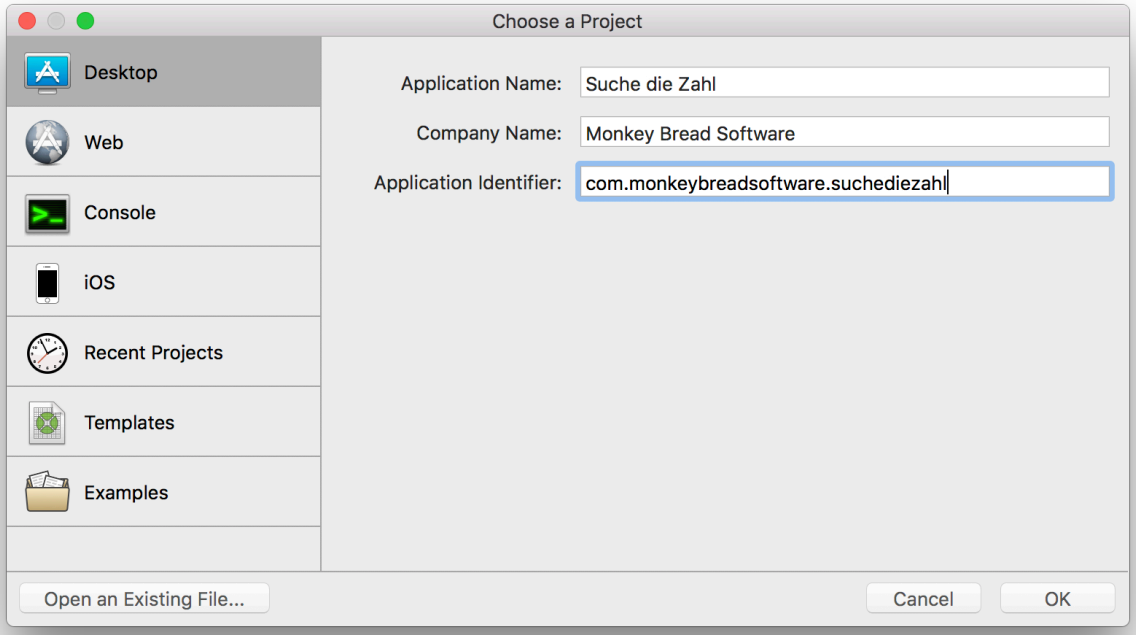
# Finde die Zahl!

**Lernen sie die Umgebung von Xojo kennen**

**von Stefanie Juchmes, Monkeybread Software**

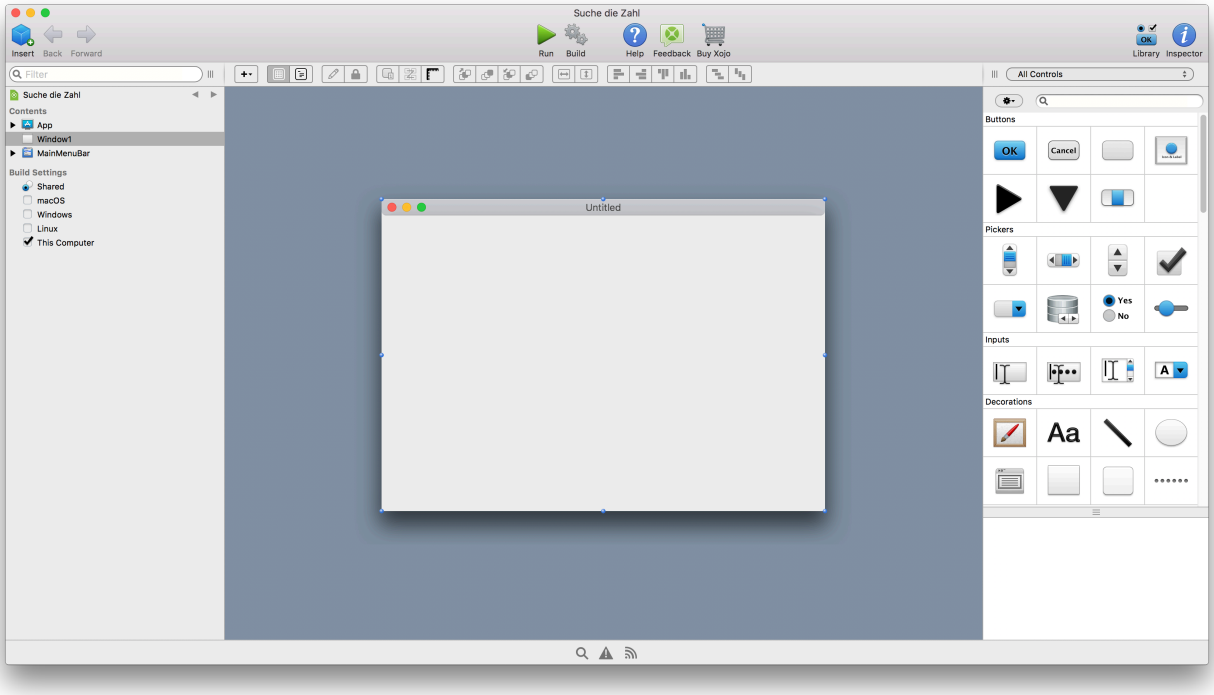
Als Einstieg in die Programmiersprache Xojo möchte ich mit ihnen gerne ein einfaches aber unterhaltsames Spiel programmieren. Bei diesem Spiel berechnet der Computer eine Zufallszahl zwischen 1 und 100 und wir müssen sie erraten.

Nun wollen wir mit dem Programmieren loslegen. Öffnen sie dazu Xojo und legen sie ein neues Desktop Projekt an.




Dazu klicken sie auf den Reiter Desktop. Unter Application Name können sie eintragen wie ihr Programm nachher heißen soll. Unter Company Name können sie den Namen ihrer Firma eintragen.

Wenn sie nun auf OK klicken, erscheint die Entwicklungsumgebung. Das Fenster ist in mehrere Teilbereiche unterteilt.




Links sehen wir die Auflistung aller Fenster, Kontrollelemente, Funktionen und Eigenschaften die in unserem Projekt vorhanden sind.

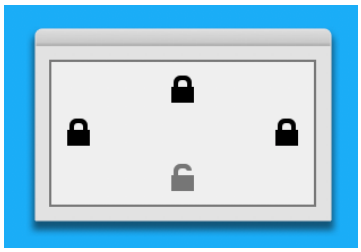
Standardmäßig ist bereits ein Startfenster, das den Namen Window1 besitzt, angelegt. In der Mitte sehen wir dieses Fenster. Momentan ist es noch leer. Bei dem mittleren Teil handelt es sich um unseren Arbeitsbereich. Wenn wir gleich programmieren werden, wird sich in diesem Bereich unser Codeeditor befinden, in dem wir den Code schreiben.

Auf der rechten Seite befindet sich ein weiteres Menü. Es enthält die Controls, die mit der Maus einfach ausgewählt, ins Fenster gezogen und dort positioniert werden können. Sollen sie diese nicht sehen, müssen sie zuvor auf die Schaltfläche „Library“ in der oberen Menüleiste klicken.
Im Standard werden die Symbole nach Kategorien sortiert. Probieren sie es doch einfach mal und ziehen einen Generic Button in das Fenster (Abteilung Buttons, 1. Reihe der 3. von Links). Wenn sie nicht wissen welches Element sie in der Bibliothek gerade vor sich haben, hilft ihnen das Fenster am rechten unteren Rand. Wenn sie mit der Maus über die einzelnen Controls in der Bibliothek fahren bekommen sie in diesem Fenster eine kurze Erläuterung zu dem Control. Sollten sie das Fenster nicht sehen, gehen sie ans Ende der Bibliothek und verschieben sie den schmalen grauen Balken ein Stück nach oben.

Sie sehen, dass der Knopf, den sie in das Fenster gezogen haben, dort erschienen ist. Sie können ihn jetzt mit der Maus beliebig im Fenster positionieren und über Griffpunkte skalieren. Wenn sie mit der Maus über den Knopf fahren, erscheint in der unteren rechten Ecke ein kleiner Stift. Wenn sie auf diesen Stift klicken, können sie den Namen des Kopfes ändern.

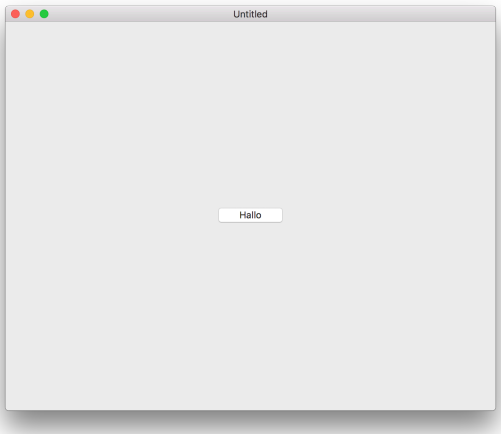
Über der Bibliothek entdecken sie im Menü zwei Knöpfe. Einmal den Knopf **Library** und einmal den Knopf **Inspector.** Wenn sie auf den Inspector klicken, sehen sie dort, wo vorher die Bibliothek mit den Controls war, die Eigenschaften zu dem Element, das wir im Hauptfenster ausgewählt haben. Die Eigenschaften können wir hier einsehen und verändern. So können wir dem Knopf in unserem Fenster zum Beispiel die Beschriftung (Caption) „Hallo“ geben. Hier können wir auch über Koordinaten bestimmen, wo sich der Knopf im Fenster befindet und wie groß er ist. Sie können zwischen der Library und dem Inspector durch einen click auf die entsprechende Schaltfläche hin und her wechseln.

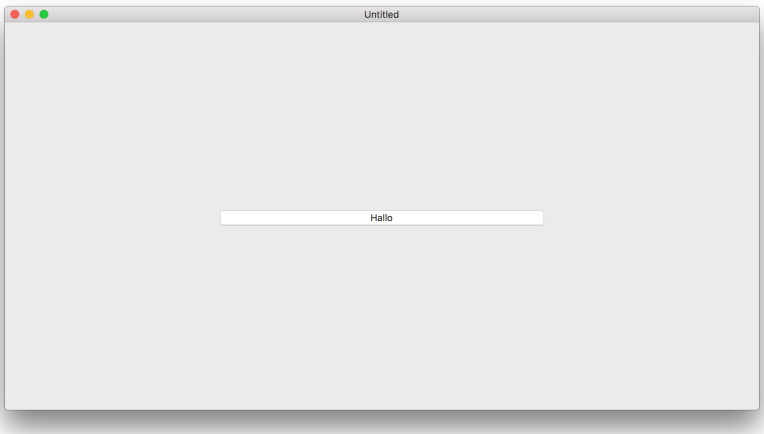
Im Locking können wir bestimmen was geschehen soll, wenn wir das Umgebungsfenster in seiner Größe variieren.
Wenn wir auf die Schlösser klicken, können wir diese schließen und öffnen. Wenn wir ein Schloss schließen, können wir uns das vorstellen, als ob wir in diesem Moment den Rand des Elements an dieser Seite festkleben würden.



Schließen wir zwei gegenüberliegende Schlösser, so wird unser Control, wenn sich die Fenstergröße verändert in dieser Achse skaliert.

Ist nur ein Schloss in dieser Achse geschlossen, so wird das Element beim Skalieren des Fensters in eine Richtung, in die gelockte Richtung gezogen.

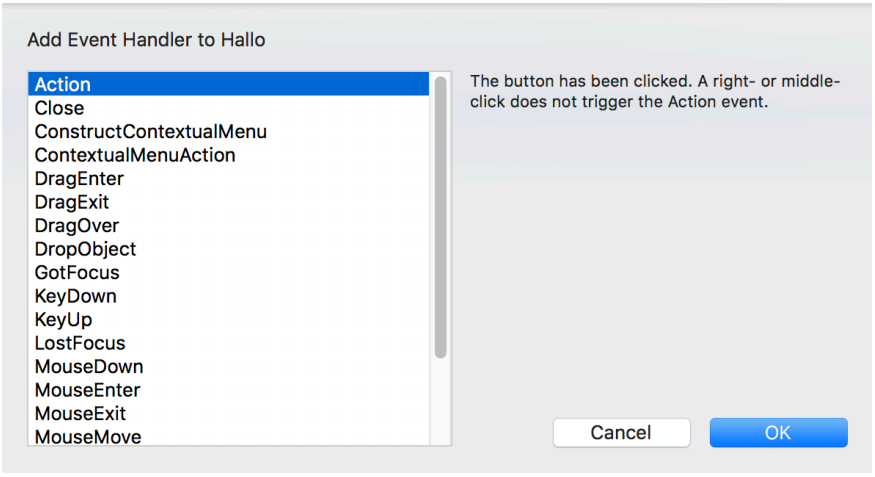



Über dem Hauptbereich der Entwicklungsumgebung befindet sich eine Menüleiste mit mehreren Icons. Wir klicken auf das kleine Plus. Dabei öffnet sich ein Untermenü. In diesem Untermenü klicken wir den ersten Eintrag Event Handler an. Es öffnet sich eine Liste mit mehreren Auswahlmöglichkeiten.

Rechts neben der Liste sehen wir eine kleine Beschreibung der Events. Diese Events werden aufgerufen wenn etwas bestimmtes mit diesem Objekt passiert z.B. wenn der Button geklickt (Action) wurde oder sich die Maus über dem Objekt befindet (MouseEnter). Dann wird der Code der vorher im ActionEvent geschrieben wurde ausgeführt.

Wir wählen das ActionEvent aus. Das bedeutet, wir legen nun fest was passieren soll, wenn der Knopf geklickt wurde.



**Hallo Welt!**

Nach dem wir das ActionEvent ausgewählt haben, öffnet sich der Code Editor. Hier schreiben wir nun den Code rein der aufgeführt werden soll. Als Test schreiben wir einmal

```
MsgBox „Hallo Welt“
```

Gehen sie jetzt einmal auf das grüne Playzeichen (Run) in der Menüleiste. Dadurch kompiliert ihr Programm und wird gestartet.

Drücken sie nun einmal den Knopf und genießen den Moment.

**Compiler:**
Kompilieren bedeutet dass ihr Programm durch den Compiler in eine Sprache umgeschrieben wird, die ihr Computer verstehen kann. Denn Ihr Computer versteht weder deutsch noch englisch. Sondern nur die Maschinensprache, Bitsequenzen aus 0 und 1. Dadurch kann der Computer ihr Programm verstehen und ausführen.

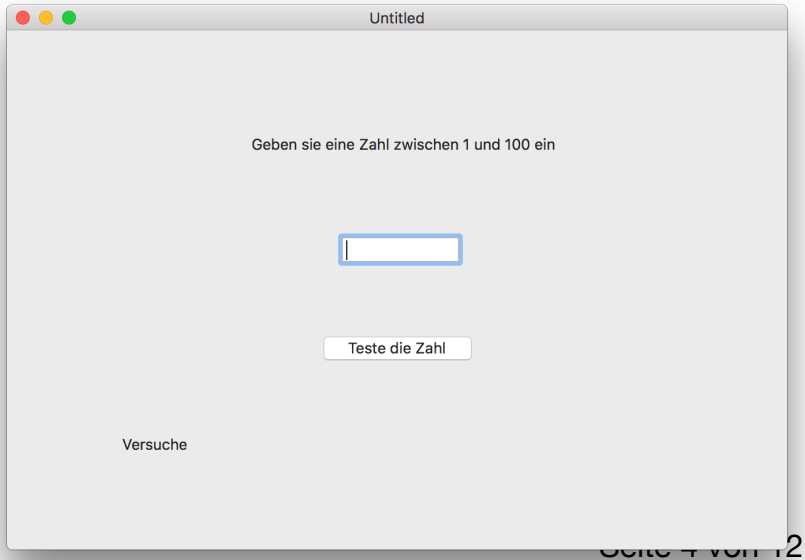

# „Herzlichen Glückwunsch!“

Sie haben ihr erstes Programm geschrieben!
Der Befehl MsgBox öffnet ein Hinweisfenster. Den Text, den dieses Hinweisfenster haben soll können wir in Anführungszeichen angeben.

**Das Interface**

Nachdem sie die Xojo Umgebung schon ein wenig erkundet haben, wollen wir nun wirklich mit dem Programm starten. Zuerst einmal bauen wir uns dafür unser Benutzer Interface.

Dazu fügen wir aus der Library ein Eingabefeld (TextField) ein. Dort wollen wir nachher die Zahlen eingeben. Wir geben ihm im Inspector den Namen „Eingabe“. Wir brauchen noch eine erklärende Überschrift, die dem Benutzer sagt, was er machen soll. Dafür nehmen wir uns ein Label aus der Library und tragen im Inspector dort den folgenden Text ein: Geben sie eine Zahl zwischen 1 und 100 ein. Den Knopf benennen wir in seinem Inspector in „Test“ um und geben ihm die Beschriftung: „Teste die Zahl“
Unter diesem Knopf werden zwei Label aus der Library positioniert. Ihr Inhalt wird gelöscht. Dem oberen Label geben wir im Inspector den Namen „Info“ und dem unteren den Namen „Versuche“. Neben dem unteren der zwei Label setzen wir ein Label mit dem Text Versuche. Ihr Fenster sollte in etwa jetzt so aussehen.



**Variablen: Hier macht ihnen nun niemand mehr ein X für ein U vor**

Die Aktion im Programm wird gestartet, wenn der Benutzer eine Zahl eingibt und auf den Knopf drückt. Das bedeutet wir brauchen den Action Event. Dieser Action Event sollte dabei zuerst einmal die Daten aus dem Fenster auslesen und in eine Variable speichern. Eine Variable ist ein Ort, an dem wir einen beliebigen Wert speichern können. Diesen Wert kann man dann an verschiedenen Stellen benutzen ohne ihn genau zu kennen. Zum Beispiel, wenn wir eine Rechnung haben x + 5 = Ergebnis, dann ändert sich das Ergebnis je nachdem welchen Wert x hat. Wenn x = 3 wäre dann würde die Rechnung als 3 + 5 interpretiert und das Ergebnis würde den Wert 8 annehmen. Genauso gut könnte X aber auch 50 sein, dann wäre das Ergebnis 55. Bei der Belegung von Variablen gibt es allerdings Einschränkungen, denn eine Variable besitzt einen Typ. Bei den meisten Programmiersprachen wird bei den Variablen der Variablentyp unterschieden. Ein Text wird aus diesem Grund mit einem anderen Typ gespeichert als eine Zahl oder ein Datum. Unsere Zahl, die wir in der Eingabe bekommen, soll in einer Variablen „EingabeZahl“ vom Typ Integer (Ganzzahl) gespeichert werden.
Dafür deklarieren wir die Variable zuerst, das bedeutet, wir legen sie an.
Wir schreiben in den Action Event (Vorher den alten Code löschen)

```
Dim EingabeZahl as integer
```

Nun wollen wir dieser Variable auch den Wert zuweisen, denn bis jetzt haben wir dem Programm nur gesagt, dass es eine geben soll. Die Wertzuweisung erfolgt anders wie gewohnt von rechts nach links. Das bedeutet, die Variable die den Wert bekommen soll steht auf der linken Seite des Gleichheitszeichens. Der Ausdruck, der den Wert liefert steht rechts.
Da wir ein Textfeld für die Eingabe benutzen, wird die eingegebene Zahl als Text interpretiert und liegt uns als Zeichenkette (String) vor. Um mit ihr richtig arbeiten zu können müssten wir sie von einem String erst in eine Zahl umwandeln. Das macht die Funktion CDbl( ).
CDbl wandet eine Text in eine Dezimalzahl (Double) um. Wir wollen zwar eine Zahl vom Typ Integer das ist aber nicht weiter schlimm, das hier einfach die Nachkommastellen abgeschnitten werden und die Typfestlegung durch die Variable „EingabeZahl“ festgelegt ist.

```
EingabeZahl = CDbl(Eingabe.text)
```

Mit dem Vorherigen Befehl weisen wir „EingabeZahl“ den richtigen Wert zu. Eingabe.text spricht dabei das Control Eingabe an und greift dort auf den Text zu.

**Irgendwann muss jeder die Entscheidung treffen**

Nun wollen wir einmal testen, ob der Benutzer überhaupt etwas eingegeben hat. Dies machen wir über eine Bedingung. Im normalen Sprachgebrauch würden wir sagen: **Wenn (If)** die Eingabe leer ist, **dann (then)** melde dich mit: „Sie müssen etwas eingeben“. So in etwa ist es auch beim Programmieren.

```
If Eingabe.Text = "" Then
        MsgBox "Sie müssen etwas eingeben"
        Return
End If
```

In diesem Fall müssen wir tatsächlich nochmal den String zu Rate ziehen, den wir aus dem Eingabefeld auslesen, denn so können wir überprüfen ob er leer ist. Die Struktur einer Bedingung sieht dann wie folgt aus:

If *Bedingung* Then
    *Code der ausgeführt wird wenn die Bedingung erfüllt ist*
End If (Bedingung abschließen)

Die Bedingung kann dabei nicht nur entscheiden ob zwei Sachen gleich sind, sondern z.B. auch ob sie größer oder kleiner sind.

Im Code kennen wir schon den Befehl für die Textausgabe in einem Dialogfeld, da wir die message box eben schon verwendet haben.

Das „Return“ sorgt dafür, dass wir wieder zu unserem Fenster zurückkehren und erneut eine Eingabe machen können. Wenn das „Return“ aufgerufen wurde, kehrt das Programm an die Stelle zurück wo es aufgerufen wurde. Alles was an Code hinter einem Return steht wird nicht mehr berücksichtigt.

**Hab ich Urlaub ODER nicht?**

Nun wollen wir noch testen ob die eingegebene Zahl überhaupt zwischen 1 und 100 liegt. Dafür brauchen wir wieder eine If-Bedingung. Wir wollen entscheiden: Wenn die eingegebene Zahl kleiner als 1 oder die eingegebene Zahl größer als eins ist, dann soll angezeigt werden, dass die Zahl außerhalb des gültigen Bereichs liegt. Dafür hängen wir die folgenden Zeilen dem Code unseres ActionEvents an:

```
If EingabeZahl < 1 OR EingabeZahl > 100 Then
        MsgBox "Geben sie eine Zahl zwischen 1 und 100 ein"
        Return
End If
```

Wie im Sprachgebrauch gibt es auch in der Programmierung ein Oder (OR). Es prüft ob mindestens eine der Aussagen der Wahrheit entspricht. Man kann sich das Oder mit einem Beispiel verdeutlichen: Ich muss entscheiden ob ich zur Arbeit muss oder nicht. Ich muss nicht zur Arbeit, wenn es ein Feiertag ist oder wenn ich Urlaub habe. In welchen Fällen habe ich frei?

1. Ich habe keinen Urlaub und es ist kein Feiertag => false OR false => false=> muss arbeiten
2. Ich habe Urlaub aber es ist kein Feiertag => true OR false => true => Ich habe frei
3. Ich habe keinen Urlaub aber es ist Feiertag => false OR true => true => Ich habe frei
4. Ich habe Urlaub und es ist Feiertag => true OR true => true => Ich habe frei

Bis auf den Fall 1 habe ich immer frei. Wenn übrigens schon die erste Aussage mit wahr bestätigt wurde, springt das Programm in den Code Teil der ausgeführt wird, wenn die Bedingung stimmt und schaut sich gar nicht mehr an, ob die anderen Bedingungen auch noch zugetroffen hätten.

In unserem Fall wird die Bedingung nur nicht durchlaufen, wenn die Zahl zwischen 1 und 100 liegt.

**Helfen wir dem Zufall auf die Sprünge**

Um zu überprüfen, ob die eigegebene Zahl mit der Zufallszahl übereinstimmt, müssen wir diese Zufallszahl erst einmal anlegen. Die Zufallszahl brauchen wir pro Spiel nur einmal. Das bedeutet sie hat im ActionEvent des Buttons nichts zu suchen, weil dieser ja mehrmals während eines Spieles ausgelöst wird. Wir könnten aber ein anderes Event benutzen. Es eignet sich das OpenEvent des Spielfensters. Wir öffnen mit ausgewähltem Fenster den Menü Handler und wählen das OpenEvent. Dieses wird gestartet sobald das Fenster geöffnet wird, in unserem Fall direkt beim Start. Nun müssen wir uns überlegen, wie wir es möglich machen, dass wir die Zahl im OpenEvent erzeugen können und dann trotzdem noch auf, vom ActionEvent des Buttons, Zugriff auf diese Zahl haben. Mit einer einfachen Variablen, wie wir sie bis jetzt kennen gelernt haben können wir das nicht, weil sie eine lokale Variable ist und man nur in dem ActionEvent oder der Methode auf sie zugreifen kann, in dem man sie angelegt hat. Deswegen muss eine andere Möglichkeit her um die Zahl zu speichern. Wir können dem Projekt über das Plus, über das wir auch den Event Handler hinzugefügt haben, eine Eigenschaft, ein sogenanntes Property, hinzufügen. Ein Property ist eine Variable die an das Fenster, in dem sie angelegt wurde gebunden ist. Das bedeutet sie von allen Events bzw. Methoden des Fensters gelesen und mit Werten belegt werden kann.

Wir fügen die Property hinzu und nennen sie „Zufallszahl“. Der Typ sollte wieder Integer sein, da wir ja nur mit ganzen Zahlen raten wollen.
Nun öffnen wir wieder auf den OpenEvent und geben dort die folgende Zeile ein:

```
Zufallszahl = rnd * 100+1
```

Die Funktion rnd liefert uns eine Zahl mit sehr vielen Nachkommastellen, die zwischen 0 und 1 liegt (0 ist ein möglicher Wert). Weil wir eine Zahl zwischen 1 und 100 haben möchten, rechnen wir mal 100 damit die Zahl werte von 0 bis 99 annehmen kann und dann +1, damit die Zahl keine 0 ist und unser Wertebereich nun zwischen 1 und 100 liegt. Da unsere Zahl vom Typ Integer ist, werden alle Nachkommastellen abgeschnitten und wir müssen uns um diese nicht kümmern. Ein Beispiel verdeutlicht das Ganze:

Rnd liefert uns die Zahl 0,0123919

rnd * 100 wäre 1,23919
rnd * 100 + 1 wäre dann 2,23919

Integer schneidet die Nachkommastellen ab, ist die Zahl 2.

Die Zufallszahl wird nach ihrer Ermittlung in der Eigenschaft Zufallszahl gespeichert. Nun können wir weitere Abfragen im ActionEvent des Knopfes tätigen.

Dort fragen wir weiter ab was passiert, wenn unsere eingegebene Zahl nicht unsere Zufallszahl ist. Das Zeichen für Ungleich ist <>. Das kann man sich gut merken weil die Zahl dann kleiner oder größer sein muss.

```
If Zufallszahl <> EingabeZahl Then
```

In dieser Bedingung müssen wir wieder eine Unterscheidung treffen, da wir im Infotext ja angegeben bekommen möchten ob unsere eingegebene Zahl größer oder kleiner als die Zufallszahl ist. Dafür bauen wir, in dem wir in den Ausführungsbereich der If Bedingung wieder eine If-Bedingung ein. Wir wollen im Label mit dem Namen „Info“, neben der Meldung ob eine Zahl größer oder kleiner ist, auch die eingegebene Zahl anzeigen lassen. Dafür können wir z.B. die folgende Zeile schreiben.

```
Info.Text = str(EingabeZahl) + " ist zu groß"
```

Wir wandeln unsere Zahl mit str() wieder zu einer Zeichenkette (String) um. Das + verbindet zwei Strings miteinander. Dabei ist darauf zu achten, dass auch das Leerzeichen ein Zeichen ist und deswegen geschrieben werden muss.

So können wir schreiben:

```
If Zufallszahl <> EingabeZahl Then
        If Zufallszahl > EingabeZahl Then
                Info.Text = str(EingabeZahl) + " ist zu klein"
        End If

        If Zufallszahl < EingabeZahl Then
                Info.Text = str(EingabeZahl) + " ist zu groß"
        End If
End If
```

**Was denn Sonst**

Im Sprachgebrauch gibt es nicht nur ein **Wenn dann** sondern auch ein **Sonst**.

Beispiel:

Wenn Ich ins Theater gehe dann
    ziehe ich ein tolles Kleid an
Sonst
    ziehe ich den Jogginganzug an

Sie merken schon an der Art wie es aufgeschrieben wurde, dass wir fast wieder eins zu eins die Idee dahinter in unserem Programm verwenden können.

Sonst heißt im Englischen Else.

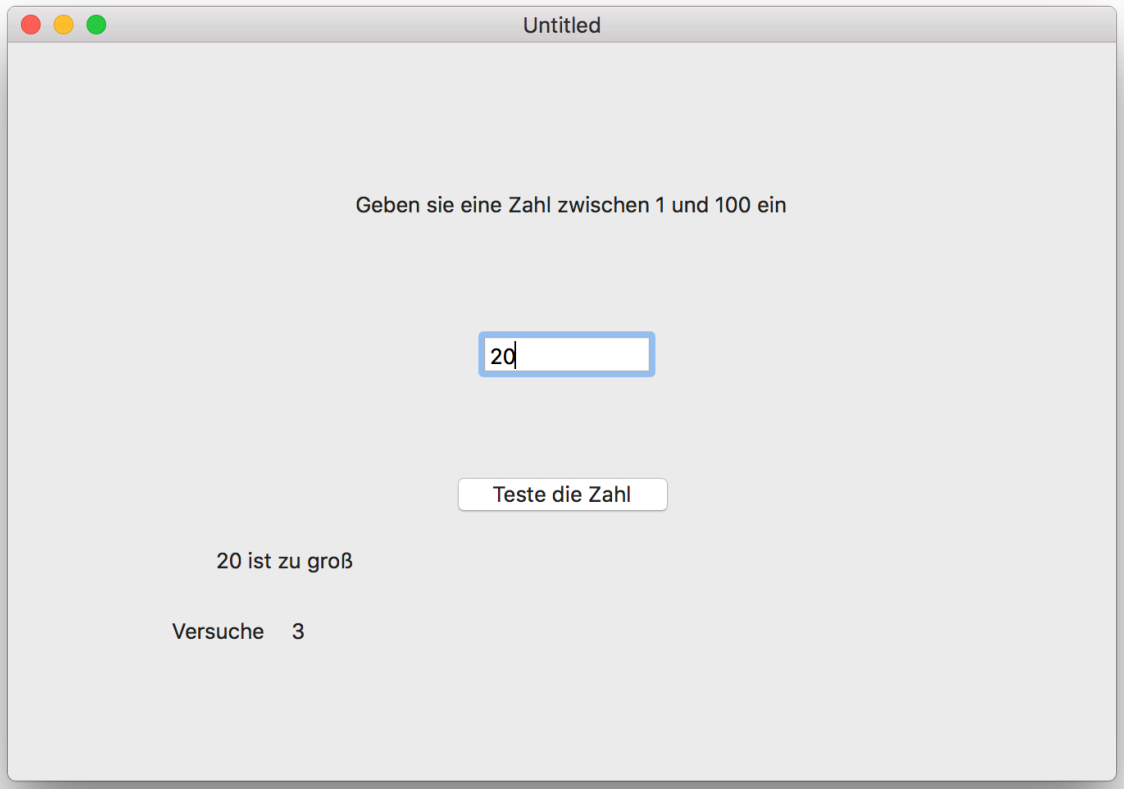
Wir haben jetzt den Fall, das die Zufallszahl und die eingegebene Zahl ungleich sind, schon bearbeitet. Es fehlt noch der Fall, das die beiden übereinstimmen und dieses können wir jetzt einfach in einen Else-Teil dieser Bedingung packen, denn wenn es nicht ungleich ist, dann muss es gleich sein.

Deswegen sieht die ganze Anweisung jetzt so aus:

```
If Zufallszahl <> EingabeZahl Then
        If Zufallszahl > EingabeZahl Then
                Info.Text = str(EingabeZahl) + " ist zu klein"
        End If

        If Zufallszahl < EingabeZahl Then
                Info.Text = str(EingabeZahl) + " ist zu groß"
        End If

Else
        Info.Text = "Herzlichen Glückwunsch! " + str(EingabeZahl) + " ist die gesuchte Zahl"

End If
```

**Schönheitskorrekturen**

Nun starten wir das Spiel




Es funktioniert herzlichen Glückwunsch :-)

Aber bevor sie nun spielsüchtig von ihrem eigenen Programm werden wollen, wir das Programm noch etwas schöner und benutzerfreundlicher machen.

**Wie viele Versuche habe ich benötigt**

Dazu geben wir zunächst einmal an wie viele Versuche wir gebraucht haben, um die Zahl zu finden. Das möchten wir in dem Label mit dem Namen Versuche ausgeben.
Da wir über die Spieldauer immer wieder die Zahl um eins erhöhen wollen, denn genau wenn eine gültige Zahl eingeben wurde, müssen wir wieder eine Property anlegen. Diese ist vom Typ Integer und wird „AnzahlVersuche“ genannt. Der Typ ist wieder Integer. Da dieses aufgeführt werden soll wenn die eingegebene Zahl gültig ist, schreiben wir

```
AnzahlVersuche = AnzahlVersuche + 1
Versuche.Text = str(AnzahlVersuche)
```

hinter die zwei If-Bedingungen die abfragen ob die Zahl ausserhalb des Definitionsbereichs liegt oder der Benutzer nichts eingegeben hat.
Durch AnzahlVersuche = AnzahlVersuche + 1 wird AnzahlVersuche immer im eins erhöht.

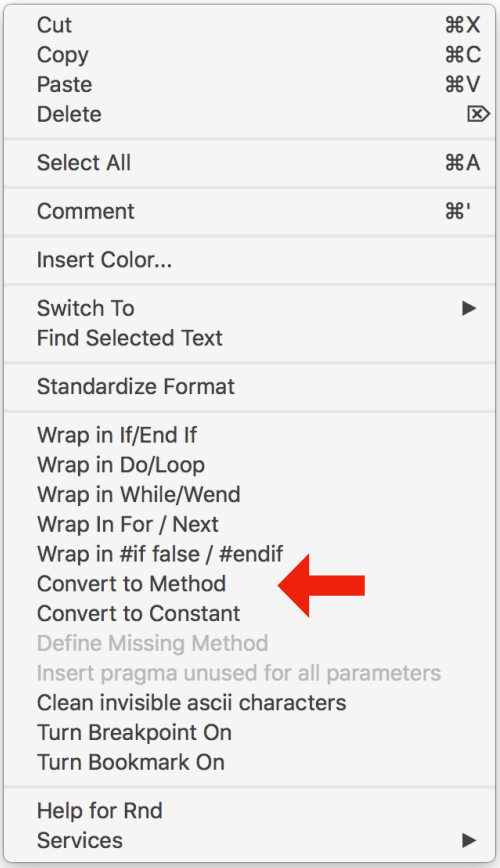
**Leere das Feld und teste die Zahl mit Return**

Nun möchten wir gerne, dass sich, wenn wir eine Zahl getestet haben, das Eingabefeld wieder leert. Deswegen schreiben wir ganz am Ende des ActionEvent

```
Eingabe.Text=""
```

Das überschreibt den Text der Eingabe mit einem leeren String.

Nun wollen wir, wenn wir eine Zahl eingegeben haben, einfach mit Enter bestätigen können ohne, dass wir den Knopf immer wieder drücken müssen. Aus diesem Grund legen wir den Knopf jetzt als Standardknopf fest, das bedeutet bei Druck auf Enter wird das ActionEvent aufgerufen. Dafür schalten wir im Inspektor des Knopfes unter Apperance die Eigenschaft Default an.
Dann funktioniert das Bestätigen der Zahl mit Enter.




**Jetzt kommt Farbe ins Spiel**

Man kann nun auch noch ein wenig am Layout feilen. Um das Aussehen eines Textes zu verändern können sie Einstellungen der einzelnen Label im Inspektor ändern. Sie könnten die Textfarbe ändern je nachdem ob die Zahl größer, kleiner oder gleich der gesuchten Zahl ist. Das können sie mit
dieser Zeile direkt vor der jeweiligen Ausgabe bezwecken.

```
Info.TextColor = rgb(0, 0, 255)
```

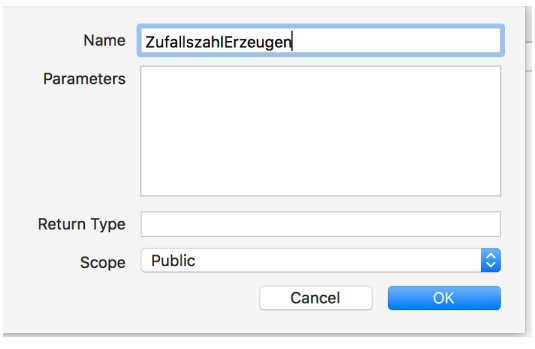
Die Farbwerte sind als RGB angegeben, das würde jetzt zum Beispiel den Text in blau einfärben.

Sie können auch die Hintergrundfarbe des Fensters ändern. Dafür können sie im Inspektor des Fensters unter Background eine Background Color auswählen. Damit die Hintergrundfarbe angezeigt wird, müssen sie Custom Color aktivieren. Im Backdrop können sie noch eine Grafik in den Hintergrund legen

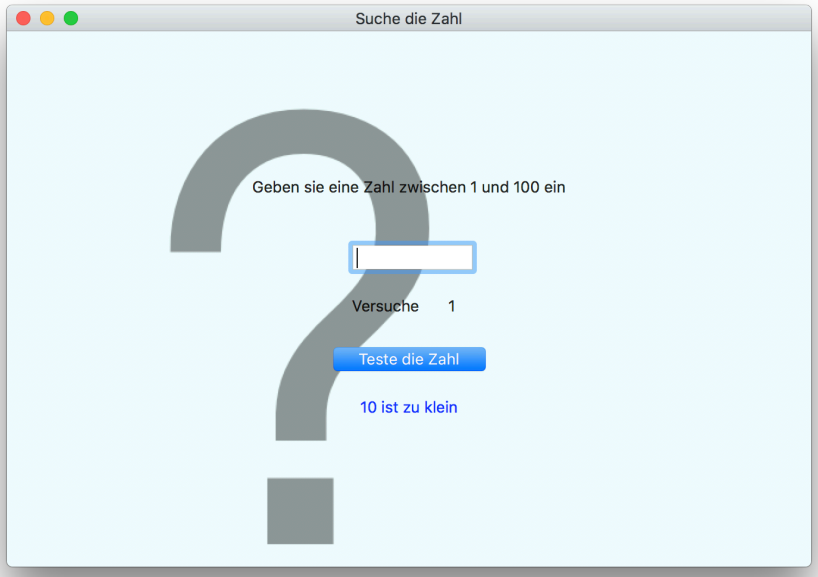
**Neues Spiel <> neues Fenster?**

Nun wollen wir aber nicht immer wieder, dass wir das Fenster schließen müssen um ein neues Spiel zu starten. Dafür fügen wir dem Layout einen Knopf hinzu. Diesen nennen wir neues Spiel. Wenn dieser gedrückt wurde soll eine neue Zufallszahl erzeugt werden, die dann in der Eigenschaft Zufallszahl abgelegt wird. Die Versuche sollen auf 0 gesetzt werden, das Infofenster wird gelehrt und die Eingabe wird geleert.
Da wir den Code um eine Zufallszahl in diesem Bereich zu erzeugen bereits geschrieben haben und wir uns beim Programmieren doppelte Arbeit sparen wollen, gehen wir in den OpenEvent vom Fenster und markieren den Code. Danach klicken wir mit der linken Maustaste darauf. Uns werden einige Optionen angeboten, was wir nun mit dem markierten Code tun können. Wir klicken Convert to Method an.

Es öffnet sich dieses Fenster und unter Name geben wir ZufallszahlErzeugen an.




Nun ist im OpenEvent die folgende Zeile zu lesen:
ZufallszahlErzeugen()
Das ist der Aufruf unserer eigenen Methode. Eine Methode ist im Endeffekt ein Codeschnipsel der an dieser Stelle auch einfach nur eingefügt werden könnte, aber da wir ihn mehrmals brauchen, haben wir ihn ausgelagert um erstens nicht so viel tippen zu müssen und um den Programmtext übersichtlicher zu gestalten.




Wir sehen die Methode jetzt am linken Rand. Wir können sie anklicken und sehen wieder den ursprünglichen Inhalt vom OpenEvent.

Nun können wir Im ActionEvent des neuen Button auch die Funktion einfügen, so dass diese jetzt so aussieht.

```
Eingabe.Text = ""
Info.Text = ""
AnzahlVersuche = 0
Versuche.Text = "0"
ZufallszahlErzeugen
```

Bei dem Aufruf ZufallszahlErzeugen können wir, die Klammern weg lassen.

Damit ist unser Spiel fertig.

**Ich wünsche ihnen viel Spaß und Erfolg beim Programmieren!**