# Codetwister

## Spelplanen

1. Skapa en tabell med 14 rader och 3 kolumner.
2. Den översta raden har (från vänster till höger) ett dolt lösningsfält och en knapp.
3. Den understa raden har ett enda textfält för meddelanden.
4. De övriga tolv raderna består av (från vänster till höger) gissningsfält, svarsfält och ett piggfält för att välja pigg (färg och form).
5. Före spelstart består alla gissnings- och svarsfält av 6 tomma pigghål.

## Spelet

1. Efter att spelplanen ritats upp slumpas en kod med 6 piggar i olika färger och former.
2. En markör placeras i det första (längst till vänster) pigghålet i det första (understa) gissningsfältet.
3. Spelaren väljer en pigg, som placeras på markörens plats. Markören flyttas till det första tomma pigghålet, räknat från vänster.
4. Om alla pigghål är fyllda händer inget när spelaren väljer en ny pigg.
5. Spelaren kan klicka på en redan utplacerad pigg, som då ersätts av en markör.
6. Om spelaren i det läget klickar på ytterligare en utplacerad pigg flyttas markören dit och dess gamla plats blir ett tomt pigghål.

## Funktioner

### placeMarker

Hämtar den första tomma cellens id (se följande metod) och ersätter dess bild med ”mark.gif”.

### getIdOfFirstEmptyPegHole

Går igenom alla tabellceller tills den hittar en tom cell (med img=”empty.gif”) och returnerar dess id.

### findCurrentGuessIndex

1. Sök fram den första tabellraden som har en tom eller markerad tabellcell.

Går igenom tabellraderna tills den hittar en tom eller markerad tabellcell och returnerar radens index. Detta räcker dock inte: den behöver också ta hänsyn till om en rad har en ”full” gissning, det vill säga har alla pigghål fyllda utan att användaren klickat på ”Check”-knappen.

För varje rad:

Är alla pigghål fyllda?

### pegSelectorClicked

Utgångspunkt är att det finns en markör