Lesson 06.02

Chinese Zodiac Animals, Pt. 2

playing Audio

reveal hidden elements

output years series via loop

Chinese Zodiac Animals, Part 2

In this Part 2 of the Chinese Zodiac Animals application, we will continue where we left off. Our START file is the FINAL file from Lesson 06.01.

Up to this point, we have outputted the 12 animal images, each with a text box beneath for entering the English or pinyin of the animal. We got the input part working too: if the spelling is correct, the box turns green; else, it turns red.

1. Preview the page with **FINAL.js** as a reminder of what the final application looks like.

- click a sound icon to play the mp3 file for that animal
- enter an animal name and hit enter to turn the box green

2. Switch to **START.js** and reload the page. Now we have just the animal pic and the input box--right where we left off last lesson. The content to add in this lesson:

- years of tha animal in the 12-year cycle
- a sound icon to click to play an mp3 audio file
- the jpg of the Chinese character for that animal
- three bits of text which are hidden on page load:
    - the English name of the animal
    - the pinyin spelling (Romanization of Chinese pronunciation)
    - the tone number from 1-4
    - the hidden elements will appear if the user spells the animal correctly in English or pinyin

3. Open **js/zodiac-animals-data.js** and have another look at the **animals** array of objects. Let's take a closer look at the properties, since we will be using all of them in this lesson:

```
    const animals = [
        {eng: "chicken", aka: 'rooster', chi: "ji", pin: "ji&#772;",
year: 2029, tone: 1},
        -- ETC --
        {eng: "tiger", chi: "hu", aka: 'laohu', pin:
"la&#780;ohu&#780;", year: 2022, tone: 3}
    ];
```

- **eng** is the English name. We use this to concatenate the animal image and audio file names, to output the English name and to check English spelling.
- **chi** is the pinyin name, without tone symbols. We use this to concatenate the character image file name and to check pinyin spelling.
- **pin** is the Chinese Pinyin name, with gibberishy symbols. We use this to output the Pinyin, complete with tone markings.
- **aka** is an alternate name: "rooster" is an accepted alternative to "chicken", while "hu" (tiger) and "laohu" ("old tiger") are both good.
- **year** is a recent or upcoming year in the 12-year cycle. We use that to run a loop that backs up 180 years and outputs 15 years at 12 year intervals
- **tone** from 1-4. In the Pinyin spelling, tone markings appear above certain vowels. In our interface, the tone number accompanies the pinyin.

Now that we have reviewed the project, let's forge ahead with new content. First up: get the sound working again.

the html audio element and JS Audio object

HTML has an **audio** tag which is used to play audio files, such as **mp3** files. The **src** attribute is set to the file to play. The JS version of this tag is the **Audio** object.

4. Open the **audio** folder and check the files. The naming format is the **eng** property followed by the **.mp3** file extension: **chicken.mp3, cow.mp3**, etc.
5. In the **START.js** file, inside the loop, make an image and set its source to **sound-icon.png**:

```
let soundIcon = new Image();
soundIcon.src = "images/sound-icon.png";
```

6. Assign the **sound-icon** class to the image. The class uses absolute positioning, which will park the icon in the upper right corner of the div:

```
soundIcon.className = "sound-icon";
```

7. Assign the **eng** proprety of the current animal object to be the **eng** property of the image. This way, when the image is clicked, the function can identify the animal as **this.eng**. This is used to concatenate the **mp3** file name:

```
soundIcon.eng = animal.eng;
```

8. Make the clicking of the sound icon call the **playSound** function:

```
soundIcon.addEventListener("click", playSound);
```

9. Output the sound icon to the div:

```
divvy.appendChild(soundIcon);
```

We need to define the **playSound** function before reloading the page in the browser, or else we will get an error. But first, we need an **Audio** object to play the sound.

10. Above the loop, in the global scope, instantiate an instance of the **Audio** object. It's the same syntax as the **Image** object, with the **new** keyword:

```
const sound = new Audio();
```

11. Below that, aka in the global scope, write the function.

- **pause()** stops any sound which may be playing
- **src** sets the source of the **mp3** file
- **play()** plays the source file

```
function playSound() {
  sound.pause();
  sound.src = `audio/${this.eng}.mp3`;
  sound.play();
}
```

12. Reload the browser. The sound icons should be back. Click one to play that animal's mp3 file.

13. Open the **chars** folder in **images**. It contains Chinese character jpgs, with the naming format: **"char-pinyin.jpg": char-gou.jpg, char-hou.jpg**, etc.

14. Still inside the loop, make an image and set its source to the character jpg for the animal. The **chi** property has the pinyin (sans tone markings) that we need to concatenate the file name:

```
let chineseChar = new Image();
chineseChar.src = `images/chars/char-${animal.chi}.jpg`;
```

15. Assign the **chinese-char** class to the image:

```
chineseChar.className = "chinese-char";
```

16. Output the Chinese character to the div:

```
divvy.appendChild(chineseChar);
```

Notice that we did not assign the Chinese character image any properties. This is because it won't be clickable or have any behavior that would require us to refer to it during runtime.

Reload the page. The application is really starting to shape up: we have the animal and character pics, as well as the sound and input box--and everything so far works.

Chinese zodiac year cycle

In the Chinese zodiac, animals repeat every 12 years. The data object has just one **year** for each animal, but that is enough, since we can use a loop to increment by 12 and make as many years as we want. We can fit 15 years down the side of the div, which is what the generous space at left is for.

17. Still in the big loop, make a **p** tag to hold the years:

```
let pTagYrs = document.createElement("p");
```

18. Assign the p tag its class and output the p tag to divvy:

```
pTagYrs.className = "zodiac-year";
divvy.appendChild(pTagYrs);
```

19. Declare a start year variable and a string to hold all the years concatenated with spaces: "1880 1892 1904 1916 1928..."

```
let startYr = animal.year - 156;
let yearsStr = "";
```

20. Run a for loop that starts 156 years before the current animal yaer and continues until the counter reaches the current animal year plus 12; counter goes up by 12 each time:

```
for (let y = startYr; y <= animal.year + 12; y += 12) {
  yearsStr += y + " ";
}
```

21. Output **yearStr** to the **pTagYrs** tag:

```
pTagYrs.textContent = yearsStr;
```

23. Reload the browser. Viola--we should have 15 years at 12-year intervals running down the left hand side of each div. The last year is the next future one in the cycle.

showing hidden text

The next and final step is to make elements to display text:

- English name centered at the top of the div
- Pinyin with tone markings next to the Chinese character
- The tone number from 1-4 underneath the pinyin

24. Still in the big loop, make a **span** tag to hold the English name:

```
    let english = document.createElement('span');
```

25. Apply its class, which has **display: none** to hide the tag on page load:

```
    english.className = 'english';
```

26. Set its text to be the English name ("cow", "dog", etc.):

```
    english.textContent = animal.eng;
```

27. Assign the tag a dynamic id that uses the current index: **eng0, eng1**, etc. The element needs an id, so that we can get -- and show -- the element in the **checkSpelling** function:

```
    english.id = 'eng' + i;
```

28. Output the span tag to the div:

```
    divvy.appendChild(english);
```

29. Still in the big loop, make a span tag to hold the pinyin and its tone number. The process is the same as for making the span to hold the English name, so we will do this all one step:

```
    let pinyin = document.createElement('span');
    pinyin.innerHTML = animal.pin + '<br>' + animal.tone;
    pinyin.pin = animal.pin;
    pinyin.className = 'pinyin';
    pinyin.id = 'pin' + i;
    divvy.appendChild(pinyin);
```

30. Reload the page. The text does not appear, nor do we have any functionality yet to make it appear.

31. In the CSS, in the **.english** and **.pinyin** classes, comment out the **display: none** properties:

```
.english {
  top: 5px;
  right: 50%;
  /* display: none; */
  font-size: 1.1rem;
  color: var(--colr);
}

.pinyin {
  top: 200px;
  right: 23%;
  font-size: 1rem;
  /* display: none; */
  letter-spacing: 1px;
}
```

32. Reload the page. Now the text appears.

33. Restore the **display: none** in the CSS so that the text is hidden again.

We want the **checkSpelling** function to show the text when the user spells an animal correctly, either in English or pinyin.

34. In the **checkSpelling** function, in the **if** part, get the English and pinyin elements by id. The first animal's English and pinyin span tags have ids of **eng0** and **pin0**. To get the elements by id, we need to first concatenate the ids using the index:

```
if (input == this.eng || input == this.chi || input == this.also) {
  // changes background and text color
  this.style.backgroundColor = "#0B0";
  this.style.color = "#fff";
  // spelling is correct so reveal English and Pinyin:
  // concat the pinyin span id:
  // get the pinyin span from the DOM:
  let pinID = "pin" + this.i;
```

```
    let pinSpan = document.getElementById(pinID);
    // change display none to inline so we can see it:
    pinSpan.style.display = "inline";
    // repeat for english span so we can see English:
    let engID = "eng" + this.i;
    let engSpan = document.getElementById(engID);
    engSpan.style.display = "inline";
  } else {
    // wrong answers just change BG color to red:
    this.style.backgroundColor = "#921";
  }
  // make the input text color white whether answer is
  // correct (green bg) or incorrect (red bg)
  this.style.color = "#fff";
```

36. Reload the page and enter an animal name in the text box.
    Hit Enter or Tab. The English and pinyin should appear.