Javascript 30

1. Javascript Drum Kit

Use `data-` prefix for made-up/customized attributes, such as `data-key`

Add event listener: `window.addEventListener('event type', callbackFunc)`

Query Selector:

- Select an element: `document.querySelector('')`
- Select all elements of a type: `document.querySelectorAll('')`

With variable in an expression, use backtick `` ` `` to enclose it, and use ES6 template string. Example:

```
document.querySelector(`audio[data-key="${varname}"]`);
```

Audio element `<audio src="sound.wave"></audio>` can be:

- played using `audio.play()`
- rewinded by using `audio.currentTijme = 0`

Change CSS style to an element `elem.classList.add("newClassName")`, similarly, there are `remove`, `toggle` methods.

Check whether a class is part of a list of classes: `elem.classList.contains('customClassName')`

Tip:

Rather than using `setTimeout` to switch style back, use a `transitionend` event. There are a lot of the same events fired. Only care about one of them.

There is also `animationend` event.

Tip:

When we have an NodeList (not the same as Array, has less methods available), we cannot directly use `addEventListner` on it. We have to use loop (ES6: `.forEach` method) to go through every element to add the event listener.

2. CSS + JS Clock

CSS `transform:rotate(20deg)`.

- Default is to rotate around the center of the element, equivalent to `transform-origin: 50%`.
- `transform-origin: 100%` will rotate around right point.
- `transform-origin: 0%` will rotate around far left point.

CSS `transition: all .5s`

CSS `transition-timing-function: ease-in-out`

JS to change the style of an element. We can use the ES6 template string here too.

```
element.style.transform = `${varname}`;
```

3. Playing with CSS Variables and JS

CSS variables are new, and can be manipulated with JS.

Define and use variables:

```
<style>
    :root {
        --base: #ffc600;
        --spacing: 10px;
        --blur: 10px;
    }

    img {
        padding: var(--spacing);
        background: var(--base);
        filter: blur(var(--blur));
    }
</style>
```

Update a variable using JS:

```
    document.documentElement
        .style.setProperty(`--${this.name}`, this.value);
```

When an element has `data-xxx` attribute, they can be grabbed by JS using `element.dataset`, which gives list of all customized attributes, without `data-` prefix.

4. Array Cardio Day 1

Filter: Filter the list of inventors for those who were born in the 1500's

```
    const fifteen = inventors.filter(inventor => {
        if (inventor.year >=1500 && inventor.year < 1600) {
            return true; // keep it.
        } else {
            return false;
        }
```

```
    );
    console.table(fifteen);
```

or a one liner:

```
    // if both conditions are true, return a boolean of true;
    const fifteen = inventors.filter(inventor => inventor.year >=1500 &&
inventor.year < 1600);
    console.table(fifteen);
```

Filter result ususally has different length from the original array.

Map: Give us an array of the inventors' first and last names Mapped results have the same length as the original array. It simply manufacture a result from input.

```
    const FullNames = inventors.map(inventor => `${inventor.last},
${inventor.first}`)
```

Sort: Sort birthday from oldest to youngest. Return is 1 or -1;

```
  const sortedInventors = inventors.sort((a, b) => a.year > b.year ? 1 : -1);
```

Reduce: How many years did all the inventors live? Allow you to build on every single one. More documentation

It's **important** to `return total` as the last statement in the reduce function. Otherwise, no correct result.

Another point is that variable initialization is the second or third part of the reduce function, depending on whether it's a function or an arrow funciton. In this case, `total` is initialized to `0`.

```
    const totalYear = inventors.reduce((total, inventor) => {
      return total + (inventor.passed - inventor.year);
    }, 0);
```

Convert `NodeList` to an `Array`, there are two ways:

- `Array.from`, example `Array.from(document.querySelectorAll('a'))`
- Spread operator `...` in ES6. Spread every single elements into an array: `[...document.querySelectorAll('a')]`

Tip: To check whether a string includes another one, use `longstring.include('shortstring')`. The return value is a booleen;

5 Flex Panels Image Gallery

Make a panels `flex`:

- add `display:flex` to the parent container.
- Add `flex: 1` to child elements, evenly distribute elements
- Children can be `flex` as well, to make them nested flex parent.

Unsplash Free beautiful pictures to use for flash screen

Read more details about CSS `transform`: 2D or 3D transformation, `transition`: The transition property is a shorthand property for the four transition properties: `transitionProperty`, `transitionDuration`, `transitionTimingFunction`, and `transitionDelay`.

`transform` can be one of the transition property to create animation.

6. Ajax Type Ahead

AJAX request can use libraries such as jQuery or axios. But browser has a new one built-in nowadays: `fetch`, which returns something called 'promise', which can be called with `.then` to return a blog of data. Calling `.json()` will convert it to proper format for this case. Other methods are listed in `_proto_`, which can be inspected in the dev mode.

```
let cities = [];
const data = fetch(url)
                .then(raw => raw.json())
                .then(data => cities.push(...data));
```

In ES6, use `let` instead of `var` for variable; use `const` for constant that will not change over time.

Create new RegExp, useful for variable matching

```
const regex = new RegExp(wordVariable, 'gi');
```

Check for matches:

```
...
return place.city.match(regex) || place.state.match(regex);
```

General Pattern for type-ahead search:

- search element define 'change' and 'keyup' listener
- function `displayMatches`, which calls `findMatches`
- function `findMatches`

When generating results from an array, it's a great time to use `.map` and `.join` to form an HTML string ready for update.

To highlight terms, build a RegExp first and then use replace method on a string

```
    let placeString = `${place.city}, ${place.state}`;
    placeString = placeString.replace(regex, `<span class="hl">${this.value}
</span>`);
```

There is a neat function to format numbers with commas.

7. Array Cardio Day 2

- Array.prototype.`some()` // is at least one person 19 or older?
- Array.prototype.`every()` // is everyone 19 or older?
- Array.prototype.`find()`, Find is like filter, but instead returns just the one you are looking for. Eg.: find the comment with the ID of 823423
- Array.prototype.`findIndex()` // Find the comment with this ID // delete the comment with the ID of 823423

Two ways to delete an array element:

- .splice

```
const index = comments.findIndex(comment => comment.id === 823423);
console.table(comments.splice(index, 1));
```

- spread

```
const index = comments.findIndex(comment => comment.id === 823423);

const newComments = [
    ...comments.slice(0, index),
    ...comments.slice(index+1),
];
console.table(newComments);
```

8. Fun with HTML5 Canvas

Canvas can be used for both 2D and 3D, which are contexts that the drawing is on.

```
  const canvas = document.querySelector('#draw');
  const ctx = canvas.getContext('2d');
```

***WATCH AGAIN**

9. 14 Must Know Dev Tools Tricks

Inpsect element, right click to:

- *Break on: attribute change*

Console

- console.log('Hello, I am a %s string!', 'abc'); or just used ES6 backtick template literal with variable ${varName}
- Style text: `console.log('%c I am some great text', 'font-size: 50px; background:red; text-shadow: 10px 10px 0 blue');`. With `%c` start, the second parameter can be a style definition
- console.`warn`('oh no');
- console.`error`('oh shit!');
- check things are true: `console.assert(1 === 1, 'that is wrong')`. The second part will only run if the first part is false.
- console.`clear()`
- console.`log(aDomElement)` will output an element's html
- console.`dir(aDomElement)` will output the element and all associated dataset, methods and other goodies
- `console.group('groupName')` and `console.groupEnd('groupName')` show messages in group.
- `console.groupCollapsed('groupName')` and `console.groupEnd('groupName')` show groups of messages in collapsed group.
- `console.count('Wes')` shows the count of a variable usage;
- `console.time('fetching data')` and `console.timeEnd('fetching data')` to check timer. Another method is `performance.now()`. Check details on MDN
- `console.table(arrayOfObjects)`

10. Hold shift to check multiple checkboxes

Use the first and last checked element to mark the range for the checked box. This method removes the need to track the item index as I previously planned. Simple and elegant.

```
  // record first click item index
  // record second click item index.
  // if shift is held, do extra work to select items between first and second
click
  const checkboxes = document.querySelectorAll('.inbox input[type="checkbox"]');
  let prevCheck

  function checkItem (e) {
    if (e.shiftKey && this.checked) {
```

```
        console.log('do something');
        let inBetween = false;
        checkboxes.forEach(checkbox => {
          if (checkbox === this || checkbox === prevCheck) {
            inBetween = !inBetween;
          }
          if (inBetween) {
            checkbox.checked = true;
          }
        });
      }
      prevCheck = this;
    }
    checkboxes.forEach(checkbox => {
      checkbox.addEventListener('click', checkItem);
    });
```

11. Custom HTML5 Video Player

For HTML5 videos, properties to control:

Grab the video element as `video`, then control use the following often used methods:

- video.`play()`
- video.`pause()`
- video.`requestFullscreen()`. This method is not listed in W3School's tutorial.

and attributes:

- paused
- volume
- playbackRate
- duration
- currentTime

and events:

- play
- pause
- timeupdate: use it to set progress bar, instead of using a timer or interval

Audio/Video can have event listener such as `play`, `pause`, similar to `click`. Use those to update interface after event is triggered.

When we use addEventListener on an element, we can access element's attribute directly with `this`. For example: `this.paused` for video here.

More details at W3School's audio/video page

For element position relative to parent, use `e.offsetX`, `e.offsetY`. To get element width, use `element.offsetWidth`.

Shortcut to combine condition check and function call in one line:

```
let mousedown = false;
// set mousedown variable somewhere else
progress.addEventListener('mousedown', (e) => mousedown && callFunction(e));
```

12. Key Sequence Detection (KONAIMI CODE)

// TODO: come back later.

Learn to use array method `splice` and `trim`.

13. Slide In on Scroll

Debounce

Use a `debounce` function to slow execution of functions on completer, scrolling, etc.

- David Walsh's ES6 debounce
- Another link

```
// Returns a function, that, as long as it
// continues to be invoked, will not
// be triggered. The function will be called
// after it stops being called for
// N milliseconds. If `immediate` is passed,
// trigger the function on the leading edge,
// instead of the trailing.
function debounce(func, wait, immediate) {
    var timeout;
    return function() {
        var context = this, args = arguments;
        var later = function() {
            timeout = null;
            if (!immediate) func.apply(context, args);
        };
        var callNow = immediate && !timeout;
        clearTimeout(timeout);
        timeout = setTimeout(later, wait);
        if (callNow) func.apply(context, args);
    };
};
```

Element position and dimensions

window.scrollY window.scrollBy

window.`scrollY` + window.`innerHeight` is the current bottom position of view port.

Determin element dimension

- .offsetTop How far an element is from the top of the window or parent element.
- `.offsetWidth` & `.offsetHeight`: total amount of space an element occupies, including width of the visible content, scrollbars, padding, and border. This is used **most often**

// TODO: to work on the excercise later.

14. Object and Arrays - Reference VS Copy

Unlike strings, numbers, booleans, the array and object is by reference.

```
const players = ['Wes', 'Sarah', 'Ryan', 'Poppy'];
const team = players;

team[3] = 'lux'
// what is players[3]?
console.log(players);
```

Make a copy of an array

There are a few ways to do it:

- `.slice()`;

```
const team = players.slice(); // pass no argument to `slice`, we get a copy of the
original array;
```

- `[].concat(array)`
- `[...players]`, use ES6's spread operator.
- `Array.from(players)`

Make a copy of an object

- `Object.assign({}, originalObject, {number: 99});` This will take an empty object, fold the originalObject's attributes in, then give the new object's number attribute a value of `99`. This won't change the originalObject's number attribute.

- `{...person}`

Both `Object.assign` and `...` operator only do one level deep. This is a shallow copy.

If we need an exact copy down to every object level, we need a 'cloneDeep' function. Google this function.

A `poor man`'s deep clone is to use `JSON.parse(JSON.stringify(object))`. Performance issue might be a problem. For simple object, it shouldn't be a problem.

15. LocalStorage and Event Delegation

On form, listen to `submit` event, instead of `click` event. This will cover click, submit, enter etc. add `e.preventDefault()` to stop page from reloading and avoid sending data to server `onSubmit`.

The backtick template string can use terniary operation like this:

```
<input type="checkbox" data-index=${i} id="item${i}" ${plate.done ? 'checked' :
''} />
```

Localstorage

MDN

Only use strings in localStorage. Remember to stringify the value first.

- localStorage.`setItem("keyname", "keyvalue")`
- localStorage.`getItem("keyname")`
- localStorage.`removeItem("keyname")`
- localStorage.`clear()`

Create customized input checkbox display

```
<form class='plates'>
    ...
    <input type="checkbox" data-index=${i} id="item${i}" />
    <label for="item${i}">${plate.text}</label>
    ...
</form>
```

```
.plates input {
    display: none;
}

.plates input + label:before {
    content: '☐';
    margin-right: 10px;
}

.plates input:checked + label:before {
    content: '🏀';
}
```

Event Delegation

For dynamically added elements, rather than listening to elements themselves, we add event listener to the parent.

16. CSS Text Shadow Mouse Move Effects

Destructuring assignment

This is useful to assign object property to new variable names quickly. documentation

```
    var o = {p: 42, q: true};
    var {p: foo, q: bar} = o;

    console.log(foo); // 42
    console.log(bar); // true
```

Finding element position II

```
  function shadow(e) {
    const {offsetWidth: width, offsetHeight: height} = hero;

    // use 'let' here instead of 'const', because value might change. Similar to
'var'.
    let {offsetX: x, offsetY: y} = e;

    // this refers to 'hero'. When the child element is the one triggering the
event and we need to get absolute position, we need to add the offsetLeft and
offsetTop. Otherwise, the x and y refers to the position of the child elemtn
relevant to the parent, not the whole document.
    if (this !== e.target) {
      x = x + e.target.offsetLeft;
      y = y + e.target.offsetTop;
    }

    console.log(x, y);
  }

  hero.addEventListener('mousemove', shadow);
```

CSS's text-shadow attribute can take multiple values separated by comma, to show multiple shadows.

17. Sorting Band Names without Articles

Use RegExp to strip strings, and use map to generate output. Content is covered mostly in previous notes.

18. Tally String Times with Reduce

Review of .map, .reduce for array.

To recap, when using `reduce` on array. Remember to `return total` as the last statement for correct behavior.

After split an array, we can pass a map function to convert elements to desired format. It saves a conversion step later.

```
    "6:10".split(":").map(parseFloat);
```

19. Unreal Webcam Fun

Video / Audio

navigator.mediaDevices

- `.getUserMedia()` returns a promise, which can use 'then' to process. The commented line is not longer working. Using the updated `.srcObject` with the stream object directly.

```
function getVideo() {
    navigator.mediaDevices.getUserMedia({video: true, audio: false})
        .then(localMediaStream => {
            console.log(localMediaStream);
            //video.src = window.URL.createObjectURL(localMediaStream);
            video.srcObject = localMediaStream;
            video.play();
        });
}
```

// come back later

20. Native Speech Recognition

Progression of the program:

- Listen for interim results
- create p element and add to the div.
- recognition will use an event listener for `result` (similar to `click`), to add this p element. This will only listen once, after result is done, the event is unbound.
- for continuous listening, we need to add another event listener `end`, then call `.start()` on the speech recognition.
- **make sure no other browser window is using the microphone**, otherwise, no feedback on the current window.

```
window.SpeechRecognition = window.SpeechRecognition ||
window.webkitSpeechRecognition;
```

```
const recognition = new SpeechRecognition();
recognition.interimResults = true;
recognition.lang = 'zh-CN';

let p = document.createElement('p');
const words = document.querySelector('.words');
words.appendChild(p);

recognition.addEventListener('result', e => {
const transcript = Array.from(e.results)
    .map(result => result[0])
    .map(result => result.transcript)
    .join('');
    p.textContent = transcript;
});

recognition.addEventListener('end', e => {
    recognition.start();
    p = document.createElement('p');
    words.appendChild(p);
});

recognition.start();
```

21. Geolocation based Speedometer and Compass

// come back later, need mac and iphone simulator.

22. Follow Along Links

Events available to listen on HTML tags:

- mouseenter
- mouseleave

```
const linkCoords = this.getBoundingClientRect();
```

which yields a rectangular object with dimensions, without scrolling:

- top
- left
- width
- height
- right

- bottom

For absolute location, add `window.scrollX` and `window.scrollY` to the left and top value.

Rather than define top and left attribute, we can use `transform` to move the element location:

```
highlight.style.transform = `translate(${linkCoords.left}px,
${linkCoords.top}px)`;
```

`px` at the end of the attribute is important. Without it, the value doesn't put the element in the correct position.

23. Speech Synthesis

speechSynthesis is a global variable, and can be passed with a SpeechSynthesisUtterance to `speak(utterance)` to speak the text.

```
const msg = new SpeechSynthesisUtterance();
let voices = [];
const voicesDropdown = document.querySelector('[name="voice"]');
const options = document.querySelectorAll('[type="range"], [name="text"]');
const speakButton = document.querySelector('#speak');
const stopButton = document.querySelector('#stop');

msg.text = document.querySelector('[name="text"]').value;

function populateVoices() {
  voices = this.getVoices();
  const voiceOptions = voices.map(voice =>
    `<option value="${voice.name}">${voice.name}</option>`)
    .join('');
  voicesDropdown.innerHTML = voiceOptions;
}

function setVoice() {
  msg.voice = voices.find(voice => voice.name === this.value);
  toggle();
}

function toggle(speak = true) {
  speechSynthesis.cancel();
  if (speak) {
    speechSynthesis.speak(msg);
  }
}

function setOptions() {
  msg[this.name] = this.value;
  toggle();
}
```

```
  // pay attention to how bind is used here. It didn't evoke the function, but
allow parameters to be passed in.
  stopButton.addEventListener('click', toggle.bind(null, false));
  speakButton.addEventListener('click', toggle);
  options.forEach(option =>
    option.addEventListener('change', setOptions));
  voicesDropdown.addEventListener('change', setVoice);
  speechSynthesis.addEventListener('voiceschanged', populateVoices);
```

How to bind an event listener and pass in argument? use `bind`

```
// use the context of `this` and pass in argument `false`, or whatever the value
is
stopButton.addEventListener('click', toggle.bind(null, false));
```

24. Sticky Nav

When scroll past nav bar's top, add a `fixed-nav` class to the body, not the nav bar itself. This is more flexible because other things can change based on the body's class change, such as content, side bar, etc., not just nav bar itself.

The change can be done on CSS with additional rule:

```
nav {
    position: relative;
    top: 0;
}

.fixed-nav {
    position: fixed;
}
```

When nav bar is switched from 'relative' to 'fixed', the element is taken out of the document flow, and cause jittery to the next element. Using the original element's offsetHeight to remove jittery.

```
  const nav = document.querySelector('nav');
  const topOfNav = nav.offsetTop;
  function fixNav() {
    if (topOfNav - window.scrollY < 0) {
      document.body.style.paddingTop = nav.offsetHeight + 'px';
      document.body.classList.add('fixed-nav');
    } else {
      document.body.style.paddingTop = 0;
      document.body.classList.remove('fixed-nav');
    }
  }
```

```
  window.addEventListener('scroll', fixNav);
```

25. Event Capture, Propagation, Bubbling and Once

In modern browser, when a click happens, event `capture` goes from parent to inner most child, then event `bubble` goes from child to parent.

```
  <div class="one">
    <div class="two">
      <div class="three">
      </div>
    </div>
  </div>
```

In `addEventListener`, we can add a third option to set `capture: true` to capture event on the way down, instead of the default `bubble` stage.

```
  divs.forEach(div => div.addEventListener('click', eventHandlerFunc, {capture:
true}));
```

To stop bubbling up and avoid trigger parent's event listener, add:

```
e.stopPropagation(); // stop bubbling.
```

Another new option is `once`, which will listen for event once, and then unbind itself. This is useful in store checkout, which disable user from submitting multiple times.

```
  divs.forEach(div => div.addEventListener('click', eventHandlerFunc, {once:
true}));
```

26. Stripe Follow Along Dropdown

// opacity and display:block work together to generate the animation with javascript. come back later. // Also need to do a setTimeout on the second class in order to get animation working.

27. Click and Drag to Scroll

Events to listen

```
slider.addEventListener('mousedown', () => {});
slider.addEventListener('mouseleave', () => {});
```

```
slider.addEventListener('mouseup', () => {});
slider.addEventListener('mousemove', () => {});
```

Using CSS to scale content

not active:

```
.elm {
  transition: all 0.2s;
  transform: scale(0.98);
}

.elm.active {
  transform: scale(1);
}
```

Finished code with comments

```
  const slider = document.querySelector('.items');
  // 1. add global is mouse down variable;
  let isDown = false;
  // need to know where to start on the whole div
  let startX;
  // also need to know where the scroll is
  let scrollLeft;

  slider.addEventListener('mousedown', (e) => {
    isDown = true;
    startX = e.pageX - slider.offsetLeft;
    scrollLeft = slider.scrollLeft;
    slider.classList.add('active');
  });

  // 2. make sure isDown is marked as false when mouseleave is triggered.
  //  Some of the 'sticky' drag problem I encountered before might be related to
this one.
  slider.addEventListener('mouseleave', () => {
    isDown = false;
    slider.classList.remove('active');
  });
  slider.addEventListener('mouseup', () => {
    isDown = false;
    slider.classList.remove('active');
  });
  slider.addEventListener('mousemove', (e) => {
    if (!isDown) return; // stop the function from running.
    e.preventDefault(); // stop other weird stuff from happening.
    const x = e.pageX - slider.offsetLeft;
    const walk = (x - startX)*3;
```

```
    console.log({x, startX, walk}); //a quick way to show multiple variables is to
generate an object.
    // we can 'set' scrollLeft value, not just 'get'.
    slider.scrollLeft = scrollLeft - walk;
    console.log('do work');
  });
```

28. Video Speed Controller UI

arrow function and this

> You don't just want to go willy-nilly using arrow functions everywhere, because it's just less to type. You need to know what the benefits and the drawbacks of them are. In this case I don't want an arrow function, because I need the keyword to reference the actual box that got clicked. That would be even more important if I had a whole bunch of them.

> We can't use an arrow function there. I'm going to bring that back to regular function.

When we add a `setTimeout` function inside another function, we can use `this` in arrow function because:

> when you have an arrow function, it does not change the value of `this`. It inherits the value of this from the parent. We don't have to worry about the scope changing or anything like that.

Read `Kyle Simpson`'s comment at the end. It clarifies/muddies some of the concept above.

Math related to number rounding

- Rounding number: `Math.round(num);`
- Floor: `Math.floor(seconds / 60);`
- Number to decimal points of 2: `num.toFixed(2);`

29. Countdown Clock

setInterval does not always work for countdown. So absolute values need to be stored in order to get accurate results.

This episodes talked about setInterval and how to clear it. Not much going on.

Form

Instead of getting form element one by one, we can use the `name` attribute to get it directly.

```
<form name="customForm" id="custom">
<input type="text" name="minutes" placeholder="Enter Minutes">
</form>
```

```
var form = document.customForm;
var min = form.minutes;
```

For forms, we listen to `submit` event and use `e.preventDefault()` to stop form from submitting. Also use `.reset()` to clear out form afterwards, instead of setting one value after another.

```
function customTimer(e) {
    e.preventDefault();
    console.log(this.minutes.value);
    this.reset();
}
```

30. Whack A Mole Game

CSS Trick: To hide a child element by shifting it downwards in `top: 100%; position: absolute;`, and parent element's position as `relative;`

Although most of this class is reviewing old content, it ties everything together. It's good to do it again to figure out the game flow and where to control the game properly.

isTrusted in event

If an event is faked (such as using javascript to trigger 'click', etc) `isTrusted` attribute of the event is `false`. For event coming from user's actual mouse input, `isTrusted` is true;