Supplementary notes for WEBDEV1-FEB23

LINKS FROM:

HTML/CSS

https://tinyurl.com/WEBDEV1-FEB23

https://www.photopea.com/

https://jakearchibald.com/2020/avif-has-landed/

https://caniuse.com/?search=display%3A%20grid

https://web.dev/one-line-layouts/

https://1linelayouts.glitch.me/

JS1

https://tinyurl.com/WEBDEV1-FEB23-JS1

Q: input type button vs button element

Note: While <input> elements of type button are still perfectly valid HTML, the newer <button> element is now the favored way to create buttons. Given that a <button>'s label text is inserted between the opening and closing tags, you can include HTML in the label, even images.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/button

look for references - level of detail

MDN - full depth
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/button

W3 - quick lookup

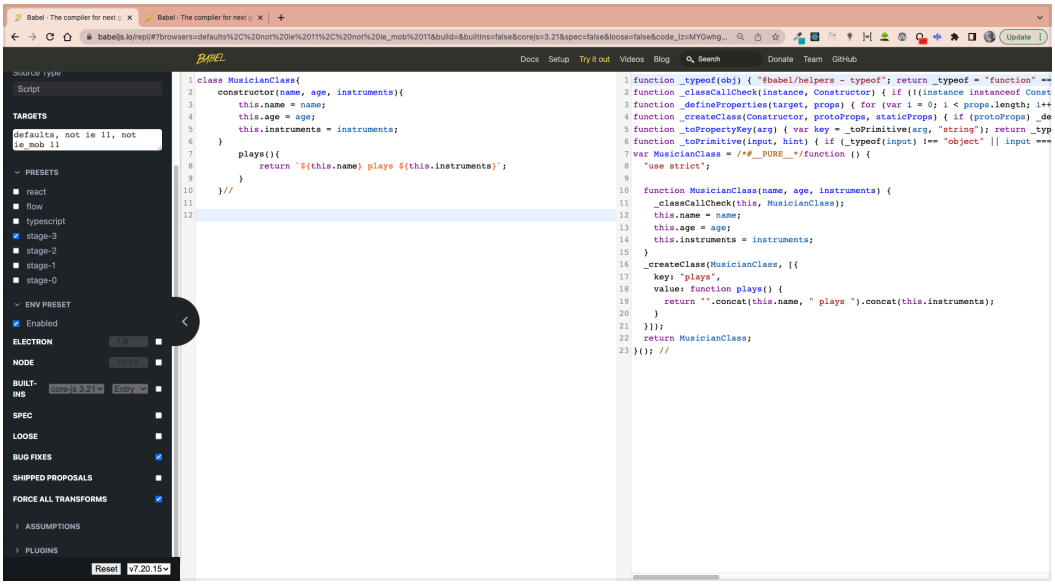
https://www.w3schools.com/tags/att_input_type_button.asp

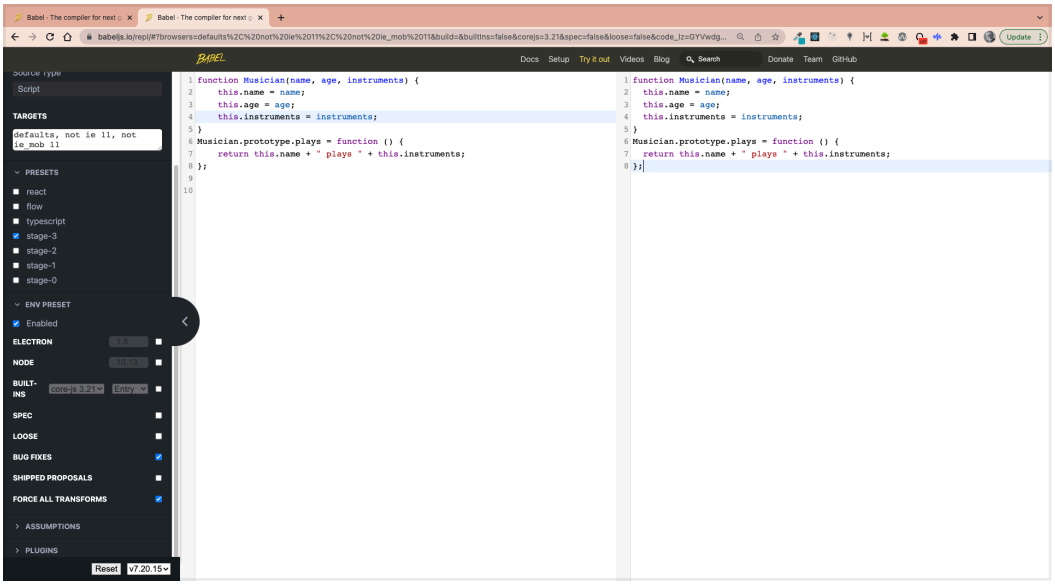
GeeksForGeeks - in between, nice level of granularity
https://www.geeksforgeeks.org/html-input-type-button/

It’s match case in Python (not case select, I was thinking VB script!). Introduced in Python 3.10: https://learnpython.com/blog/python-match-case-statement/

https://babeljs.io/repl/

ENV PRESET
✓ ENABLED
...
✓ FORCE ALL TRANSFORMS





CH9-BEFORE
https://codepen.io/lavenderlens/details/rNVrpNv

CH9-AFTER
https://codepen.io/lavenderlens/pen/PvRJZa

MODULAR JS AND MJS
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules

TAILWIND CHEATSHEET V3
https://tailwindcomponents.com/cheatsheet/

POST-COURSE RESOURCE

WHEN SHOULD WE USE FUNCTION EXPRESSIONS OR FUNCTION DECLARATIONS?

//FUNCTION DECLARATION

```
function myFunc(){

    //code to execute
}
```

//FUNCTION EXPRESSION

```
var myFuncExpression = function(){

    //code to execute
}
```

- unlike function declarations they are not loaded into memory during the first pass of the code
- so they will not be “hoisted” - they have to be declared nearer to where they are used in the code
- they may be assigned to **const** and so made final
- they may be assigned to the prototype property of a function constructor

clear online blog:
https://gomakethings.com/function-expressions-vs-function-declarations/

in super-depth:
https://github.com/getify/Functional-Light-JS/blob/master/manuscript/ch2.md/#chapter-2-the-nature-of-functions

video tutorials:
https://frontendmasters.com/courses/advanced-javascript/function-declarations-function-expressions-and-block-scope/
[course preview, paywall]

this is not the whole story - also concerns binding the context of ‘this’, but ES6 arrow functions have largely taken over, and do this out-of-the-box.

Arrow functions DO NOT HAVE their own context of ‘this’ and so inherit that of their parent which, if it’s an ES6 class, is usually what we want to happen.

Function expressions are not the only answer - Generator functions for example are written as function declarations. The original generator code must be loaded at the start and always be available in its starting state when you later assign it multiple times to make

iterators and call their next( ) method.

Core JS:

1. **EVENT LOOP**
2. **SCOPE**
3. **FUNCTION**

"Closure is when a function remembers and accesses variables from outside of its own scope, even when that function is executed in a different scope." Kyle Simpson, from Functional-Light JS

Get ALL of Kyle's book series in pdf on **GitHub.com/getify**

- if you read nothing else, these books will unlock ninja levels of mastery

---

TIMING YOUR SCRIPT LOADING
https://hacks.mozilla.org/2017/09/building-the-dom-faster-speculative-parsing-async-defer-and-preload/
including (this is very cool) use link tag as script (or other) tag(s) for preload -
All you need to write is:
<link rel="preload" href="very_important.js" as="script">
You can link pretty much anything and the as attribute tells the browser what it will be downloading. Some of the possible values are:

- script
- style
- image
- font
- audio
- video

EVENT LOOP VIDEO - A MUST WATCH!
Jake Archibald talk on the event loop, from JSConf Asia 2018
https://youtu.be/cCOL7MC4Pl0

---

how to find and kill running process IDs on any given port numbers:

find and kill running servers:
sudo lsof -i :4000
// this will print out stuff including PID
kill -9 <PID> *// without angle brackets*

ON WINDOWS:
netstat -ano | findstr :<PORT>
(Replace <PORT> with the port number you want, but keep the colon)
netstat -ano | findstr :*
(ALL ports)
taskkill /PID <PID> /F
(No colon this time)

ON LINUX:
same:
sudo lsof -i :4000
// this will print out stuff including PID
kill -9 <PID>
https://www.baeldung.com/linux/kill-process-on-port