WEB DEVELOPMENT

Practicum - Week 9.1

Exercise guidelines

🗁 Folder structure

For this exercise, **create a new folder called week 9.1**. in your web development folder. Only put the files necessary to complete the exercise in here.

DO NOT add any source materials in this folder (screenshots, pdf files, ...)

🗁 Submission deadline

The deadline for this exercise is **Sunday, December 5th, 23:59.**

🗁 Submission rules

Submission rules are listed in the week 1.1 briefing

Plan of attack

If you've been paying attention in class, you should be able to complete the exercise below. For definitions of **fancy nancy and paper trail,** see week 7.2.

Robots, assemble!

Recreate the example as demonstrated on the screencast. Don't forget to commit regularly! We expect **at least 10 commits with useful commit messages**. Commit 1, commit 2, ... are NOT useful messages. Describe what you have been doing.

🛩 PAPER TRAIL

Ensure a paper trail for this exercise and hand it in along with your source files. Post it in the root of the 9.2 directory and call it "paper-trail.png" or .txt or .md

Tip: Solve in the following order:

1) Change the robot's mood

2) Try changing his feet / arms / mohawk

3) Try changing a single square for the animation, then move on to the full animation

Some tips and trick for this exercise

1. This assignment uses **SVG tags**. You can **think of them as images**. Do not be taken aback by their verbose nature. Just replace the <sgv> tag in your mind with an <img> tag. They are exactly the same thing.

   Tip: You can collapse long parts of code by hitting the minus "-" symbol in the gutter in WebStorm.

   SVG tags give us the advantage of manipulation. One of the required changes is changing colour. You'll need the **CSS property "fill"** to change an SVG's fill colour.
   Example:
   `document.querySelector('.head .st0').style = ` `fill: #FF0000` `;`

   Several hooks have been created in the HTML, by means of CSS classes.
   The **robot's head consists out of 3 parts:**
   .head .st0
   .head .st1
   .head .st4

2. As taught in class, you can know **which element has been clicked** by implementing **e.target**. This will only work inside an event handler.

3. The animation is JavaScript based. Though you would implement this in CSS were this a live project, the object of this exercise is to **practice setting and clearing a timer**. Take a closer look at the **setInterval** function to make this work https://developer.mozilla.org/en-US/docs/Web/API/setInterval

   The elements representing the droid"s lights are bundled with the **CSS class "lights".** Adding the class **"active"** to one of those elements will change the layout. You can always try this directly in HTML to see the effects before you program it in JS.
   Colours used are **#ff1a00, #91e842 en gold**

4. When assigning an upgrade you will probably need to remove elements from the DOM. Consult the MDN documentation on how to do this https://developer.mozilla.org/en-US/docs/Web/API/Element/remove

Don’t panic if you can’t complete everything just yet, this will take some practice. Remember: **try breaking down your problem in smaller parts**, the smaller the better and then program what you *do* know.

Baby steps are okay!

🏅 FANCY NANCY

1. Every action has an upgrade score, which will grant our robot an ultimate destruction score. A scoring shortlist:
   - every action grants 100 points
   - except for angry and jealous, these deduct a 100 points
   - the mohawk adds 50 points
2. Careful: the score cannot be lower than 0
3. Do not store the score values listed above in JS. **JS only holds the formula**, so dynamically adapts to any new score values. A use case for this would be regenerated HTML (e.g. from the database) when changing the score.
   The same counts for the **power ups**