# Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-js-and-css-challenge/grade/yc73

IT202-008-S2024 - [IT202] JS and CSS Challenge

Submissions:

Submission Selection

1 Submission [active] 2/11/2024 12:57:27 PM

Instructions

COLLAPSE

- Reminder: Make sure you start in dev and it's up to date

```
1 .git checkout dev
2 .git pull origin dev
3 .git checkout -b M3-Challenge-HW
```

1 .Create a copy of the template given here: https://gist.github.com/MattToegel/77e4b66e3c73c074ea215562ebce717c
2 .Implement the changes defined in the body of the code
    1 .Hint: You may want to use your browser's developer tools to see the script that's pulled in, this may help with a few challenges
3 .**Do not** edit anything where the comments tell you not to edit, you will lose points for not following directions
4 .Make changes where the comments tell you (via TODO's or just above the lines that tell you not to edit below)
    1 .**Hint**: Just change things in the designated `<style>` and `<script>` tags
    2 .**Important**: The function that drives one of the challenges is `updateCurrentPage(str)` which takes 1 parameter, a string of the word to display as the current page. This function is not included in the code of the page, along with a few other things, are linked via an external js file. Make sure you do not delete this line.
5 .Create a branch called M3-Challenge-HW if you haven't yet
6 .Add this template to that branch (git add/git commit)
7 .Make a pull request for this branch once you push it
8 .You may manually deploy the HW branch to dev to get the evidence for the below prompts
9 .Once done, generate the output/submission file
10Add, commit, and push the submission file
11Close the pull request by merging it to dev (double-check all looks good on dev)
12Manually create a new pull request from dev to prod (i.e., base: prod <- compare: dev)
13Complete the merge to deploy to production
14Upload the same submission file to Canvas
15Checkout dev and pull latest changes to prepare for future work

**Branch name:** M3-Challenge-HW

Tasks: 15 Points: 10.00

## Screenshots (4 pts.)

COLLAPSE

COLLAPSE

### Task #1 - Points: 1

**Text: 5 Screenshots based on checklist items**

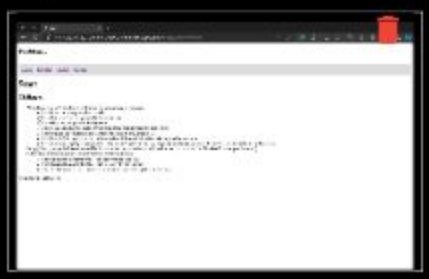
Checklist

*The checkboxes are for your own tracking

| # | Points | Details |
|---|---|---|
| #1 | 1 | A screenshot showing the Primary page with the checklist items completed (the view that initially loads) |
| #2 | 1 | A screenshot showing the page after the login link is clicked with URL shown |
| #3 | 1 | A screenshot showing the page after the register link is clicked with URL shown |
| #4 | 1 | A screenshot showing the page after the profile link is clicked with URL shown |
| #5 | 1 | A screenshot showing the page after the logout link is clicked with URL shown |
| #6 | 1 | Screenshots should be from heroku dev |

Task Screenshots:

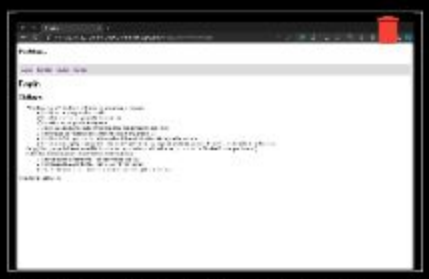
Large Gallery



Checklist Items (1)

#1 A screenshot showing the Primary page with the checklist items completed (the view that initially loads)

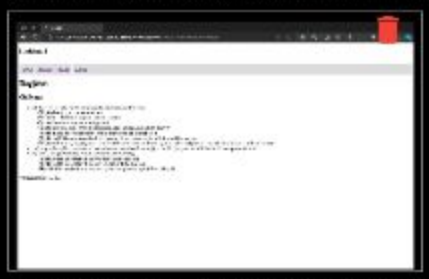
Showing the primary page with checklist items completed



Checklist Items (1)

#2 A screenshot showing the page after the login link is clicked with URL shown

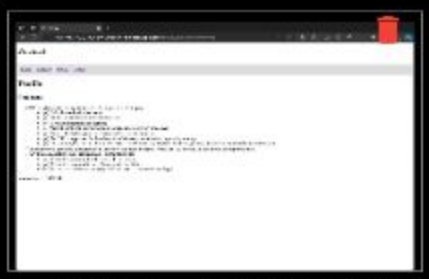
Showing page after Login link is clicked



Checklist Items (1)

#3 A screenshot showing the page after the register link is clicked with URL shown

Showing page after Register link is clicked



Checklist Items (1)

#4 A screenshot showing the page after the profile link is clicked with URL shown

Showing page after Profile link is clicked

Checklist Items (1)

#5 A screenshot showing the page

after the logout link is clicked with URL shown

Showing page after Logout link is clicked

COLLAPSE

Explanations (4 pts.)

COLLAPSE

Task #1 - Points: 1

**Text: Briefly explain how you made the navigation horizontal**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

To make the navigation horizontal, I needed to pull the list and list items out of the normal flow of the document, which I did with float (in "nav ul" and "nav li"). This allows me to position the list and lists items by floating the items next to each other (in this case to the left) so they can be pulled into a horizontal row. Since I made the list and list items float left, this caused the list and h1 element to bump into each other, so to tell the h1 to not touch that element, I added a "clear" to h1.

COLLAPSE

Task #2 - Points: 1

**Text: Briefly explain how you remove the navigation list item markers**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

To remove the navigation list item markers, I needed to apply styling to change the default settings of the list items, which I did using "list-style-type". List-style-type is a list property that allows me to make changes to the style of a list, specifically to the type of marker. So, by setting the property value to none it overrides the default styling by removing the markers/bullets. I used a descendant combinator to specifically select the <li> elements that are descendants of <nav> and apply the styling to only those <li> elements (even if they're the only <li> elements in this assignment).

COLLAPSE

Task #3 - Points: 1

Text: Briefly explain how you gave the navigation a background

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

To give the navigation a background, I used a background property because it allows me to specify and set (background related) styles to elements. I particularly used "background-color" to set the background to a light grey. However, to style just the navigation, I used the selector "nav ul" which specifically grabs the <ul> element within <nav> and only styles that element (in <nav>) instead of every <ul> element. Selecting "nav ul" also lets me add the background color to the entire list to get a full background instead of just the individual list elements.

COLLAPSE

Task #4 - Points: 1

**Text: Briefly explain how you made the links (or their surrounding area) change color on mouseover/hover**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

I made the links change color on mouseover/hover by using the ":hover" selector because not only does it know when a user interacts with an element with their cursor, it also allows me to style elements and trigger/apply that style when they mouseover it. To apply this only to the links, I added ":hover" to the "nav li a" selector, which grabs the anchors within the list items <li> that descend from <nav>. To apply the styling, I used the color property to change the color of the text in the "nav li a:hover", so it defines the style for ":hover".

COLLAPSE

Task #5 - Points: 1

**Text: Briefly explain how you changed the challenge list bullet points to checkmarks (✓)**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

To make the challenge list checkmarks, I needed to apply styling to change the default settings of the list items, which I did by using a list property. I used "list-style-type" because this property can style/make changes to the type of marker

a list is using. Since there isn't a property value for checkmarks, I set it to my own by putting the checkmark in double quotes, which means it's a custom market for the list items. The property "list-style-type" is added to the <ul> selector to target all the <ul> elements within the challenge list. Thats why, to not affect the navigation list items, I put its list-style property in a more specific selector (nav li) so it overrides the checkmark style in "ul".

COLLAPSE

Task #6 - Points: 1

**Text: Briefly explain how you made the first character of the h1 tags and anchor tags uppercased**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

I made the first character of the h1 tags and anchor tags uppercase by using a pseudo-element which allows to add style to a specific part of an element. I used the selector "::first-letter" to specifically target and apply a style to the first letter of an element. I added it to a "h1" selector and an "a" selector, which tells "::first-letter" to select and style every first letter of the <h1> and <a> element. To actually change the first letters to uppercase, I used the property "text-transform' which lets me control the capitalization of text. Then, I set its value to "capitalize" because that one changes the first character to uppercase.

COLLAPSE

Task #7 - Points: 1

**Text: Briefly explain/describe your custom styling of your choice**

Details:

Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

For my custom styling, I chose to add dividers (borders) between the list items in navigation to make it easier to read as the options would have their clear sections. To style the navigation list items, I used the "nav li" selector and added a border property. More specifically I used "border-right" which lets me add a border on the right side of each (navigation) list item. I also set values to display what kind of border I want; I set its width to 2px, chose a solid line, set the color to a white-ish color (using rgb). To avoid a border on the last list item, I created another "nav li" selector and added a ":last-child" which selects the last item in the element <li> (in navigation). Then, I set its border-right value to none to remove the styling (border) on its right side.

Task #8 - Points: 1

COLLAPSE

Text: Briefly explain how the styling for the challenge list doesn't impact the navigation list

Details:
Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

I was able to get styling for the challenge list to not impact the navigation list by leveraging specificity. I made sure to use the lowest degree of specificity to apply the checkmark style to the challenge list, which was a "ul" selector. For the navigation list items, I used a higher degree of specificity, "nav li", when applying the style of no markers. I did this because lower degrees of specificity are easy to override with higher degrees. So, although the "ul" type selector applies the checkmark style to all unordered lists, the descendant selector (nav li) is more specific than the type selector (ul), so it overrides it. The no marker style (in "nav li") displays instead of the checkmark style (in "ul").

COLLAPSE

Task #9 - Points: 1

Text: Briefly explain how you updated the content of the h1 tag with the link text

Details:
Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

To update the content of the h1 tag with the link text, I had to grab the <a> elements within <nav>, added an event listener (to each <a>) that extracts the link text (when a link is clicked), and then update the content of the h1 tag with updateCurrentPage(). First, I used a "querySelectorAll()" selector because it lets me select all the <a> elements (the links) that are children of <nav> and it puts it into an "array". I accessed all those <a> elements (the navigation links) with for loop that iterates though each <a>. For each iteration, a click event listener is added to each <a> element. When a link is clicked, I made it get the link text by using the ".innerText" property which returns the text of an element (the <a> element in this case). Then to actually update the page with that link text, I passed on text to the function "updateCurrentPage()" which handles changing the h1 tag's content with the link text I extracted.

COLLAPSE

Task #10 - Points: 1

Text: Briefly explain how you updated the content of the title tag with the link text

Details:
Explanation clearly describes solution and why it works (explain the code, don't use the code as an explanation)

Response:

Similar to updating the h1 tag, to update the content of the title tag with the link text, I had to: grab the <a> elements within <nav>, add an event listener (to each <a>) that extracts the link text (when a link is clicked), and then pass the text as a parament to updateCurrentPage() to update the title tag's content. First, I used a "querySelectorAll()" selector because it lets me select all the <a> elements (the links) that are children of <nav> and it puts it into an "array". I accessed all those <a> elements (the navigation links) with for loop that iterates though each <a>. For each iteration, a click event listener is added to each <a> element. When a link is clicked, I made it get the link text by using the ".innerText" property which returns the text of an element (the <a> element in this case). Then to actually update the page with that link text, I passed on text to the function "updateCurrentPage()" which handles changing the content of the title tag with the link text I extracted.

COLLAPSE

## Misc (2 pts.)

COLLAPSE

### Task #1 - Points: 1

**Text: Comment briefly talking about what you learned and/or any difficulties you encountered and how you resolved them (or attempted to)**

Details:
At least a few sentences

Response:

This assignment helped me learn how to use languages like CSS, JavaScript, and HTML to work together on website. I feel like this has helped me gain an understanding of the different properties in CSS and JavaScript and how to use them. The most difficulty I had was trying to make working links (JavaScript event handling) and accessing/extracting text from HTML using JavaScript (working with DOM). I spent some time researching online resources and watching the professor's videos to gain a better understanding and it ended up being pretty straightforward.

COLLAPSE

### Task #2 - Points: 1

**Text: Add a link to your pull request (hw branch to dev only)**

URL #1
https://github.com/yaneliii/yc73-it202-008/pull/7

COLLAPSE

### Task #3 - Points: 1

**Text: Add a link to your production file**

URL #1
https://yc73-it202-008-prod-35e9bd30f553.herokuapp.com/Module3/challenge.html

COLLAPSE

## Task #4 - Points: 1

**Text: Waka Time (or related) Screenshot**

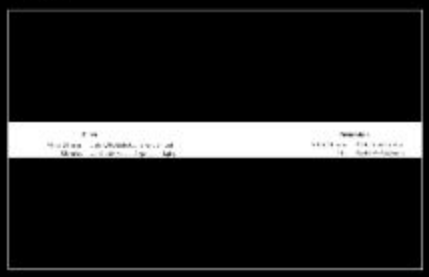
Checklist

*The checkboxes are for your own tracking

| # | Points | Details |
|---|---|---|
| #1 | 1 | Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item) |

Task Screenshots:

Large Gallery



Checklist Items (1)

#1 Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Showing waka time

End of Assignment