SESSION FOURTEEN
NOVEMBER 28, 2023

REVIEW + WRAP UP

DAVIS SCHERER
SCHERERD@NEWSCHOOL.EDU

SCHEDULE

1. Final
2. Best Practices/Wrap Up
3. P2P Critique
4. Work session

FINALS FILE STRUCTURE

```
LName_FInitial_HCFinal
    index.html
    style.css
    Entry1
        entry1.html
        style.css
        images
    Entry2
    ...     ...
    Entry11
```

*please use this nomenclature for the folder

} hub

} all files for each entry, including images, scripts, etc.

RUBRIC

- ☐ Do you have 11 entries?
- ☐ Are your entries around a theme?
- ☐ Do you have a working hub page?
- ☐ Do all of your links work?
- ☐ Do all of your images/media load?
- ☐ Do you have a favicon?

RUBRIC

If you meet all of those criteria, you pass (C).
Each one you’re missing knocks you down 5%.

The remaining 30% of your grade (70—100%)
is about the quality of the design and the entries.

RUBRIC

This part is more holistic. I’ll take into account:

- ☐ Is there variety in your entries?
- ☐ Are your entries well considered/designed?
- ☐ Are your entries engaging critically with code?
- ☐ Are your entries engaging critically with your concept?
- ☐ Did you try new things?

RUBRIC

I don’t want to take points from anyone!!!!! **PLEASE** test your websites. Shift the folder around to a new location, try it on a friend’s computer. Debug your links!!!! I am available over email if things aren’t looking right.

RUBRIC

QUESTIONS?

SIGN UP FOR FINAL CRIT

SIGN UP SHEET

BEST PRACTICES

LECTURE MATERIALS BY JACK RIEGER

This should be a bit of a recap, but! Some things to remember in and out of this class when reading and writing code…

1. DON’T REPEAT YOURSELF

DRY (Don’t Repeat Yourself) is key in this and all contexts—if you find yourself including a line of code over and over, find a way to write one line and reuse it.

1. DON'T REPEAT YOURSELF

```css
.child-1 {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 20px;
  line-height: 1.5;
}

.child-2 {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 20px;
  line-height: 1.5;
}

.child-3 {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 20px;
  line-height: 1.5;
}
```

$\rightarrow$

```css
.font-md {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 20px;
  line-height: 1.5;
}
```

2. FRAGMENTS/INCLUDES/ PARTIALS/COMPONENTS

Basically: break your HTML into small, reusable pieces. A website is already composed of components—elements that are reused across the whole system.

2. FRAGMENTS/INCLUDES/ PARTIALS/COMPONENTS

```html
<!-- components/header.html -->
<header>
  <h1>My Website</h1>
  <nav>
    <a href="/">Home</a>
    <a href="/about">About</a>
    <a href="/contact">Contact</a>
  </nav>
</header>
```

```html
<!-- components/footer.html -->
<footer>
  <a href="mailto:me@gmail.com">Email me</a>
  <p>© 2021</p>
</footer>
```

```html
<!-- index.html -->
<main class="page-content">
  <p>My website is about...</p>
</main>

<script src="js/fetch-elements.js">
```

```javascript
// fetch-elements.js
const pageContent = document.querySelector(".page-content");

fetch("./components/header.html")
  .then((response) => {
    return response.text()
  })
  .then((headerHtml) => {
    pageContent.insertAdjacentHTML("beforebegin", headerHtml);
  });

fetch("./components/footer.html")
  .then((response) => {
    return response.text()
  })
  .then((footerHtml) => {
    pageContent.insertAdjacentHTML("afterend", footerHtml);
  });
```

2. FRAGMENTS/INCLUDES/ PARTIALS/COMPONENTS

This allows you to:
(1) Reuse the same header/footer/component across pages without copy and pasting
(2) Update all your component instances at once
(3) Read & maintain your HTML more easily

This is essentially the same principle as a static site generator like Jekyll or Hugo

3. CSS VARIABLES

We talked about these earlier! It’s a way to store and reassign values in CSS, especially to create consistency of colors/spacing fonts or to adjust for media queries

3. CSS VARIABLES

```css
:root {
  --blue: rgb(0, 123, 255);
  --red: rgb(255, 59, 48);
  --yellow: rgb(255, 204, 0);

  --space-sm: 1rem;
  --space-md: 2rem;
  --space-lg: 4rem;
}
```

```css
.element {
  background-color: var(--blue);
  padding: var(--space-md);
}
```

4. CSS NESTING

CSS nesting is a new feature that allows you to physically nest CSS properties rather than defining sub-selectors in the code.

4. CSS NESTING

```css
.element {
  background-color: var(--blue);
  padding: var(--space-md);
}

.element .child {
  color: var(--red);
}
```

```css
.element {
  background-color: var(--blue);
  padding: var(--space-md);

  .child {
    color: var(--red);
  }
}
```

4. CSS NESTING

The big plus for this is less about child elements or classes, and more about states. It’s easy to group how states like hover and focus will appear.

These selectors are defined with an ampersand

```
&:hover {
```

5. GRIDS!

Lastly! Don’t forget about grids! They are the backbone of the web and make creating responsive websites so much easier!

PEER-TO-PEER CRITIQUE

We’re going to do **two** rounds of peer critique. Pair up, with the person next to you. Take 10 minutes each to walk through your entries, hubs, and any questions you may have about your work. I’ll let you know when it’s been 10 minutes and you can switch.

PEER-TO-PEER CRITIQUE

Now, switch. We will do this one more time. Address any questions that the last critique may have brought up.

COURSE EVALS

Please take the rest of class to make revisions and complete your course evaluation, if you care to do so.

SESSION FOURTEEN
NOVEMBER 28, 2023

THANK YOU

DAVIS SCHERER
SCHERERD@NEWSCHOOL.EDU