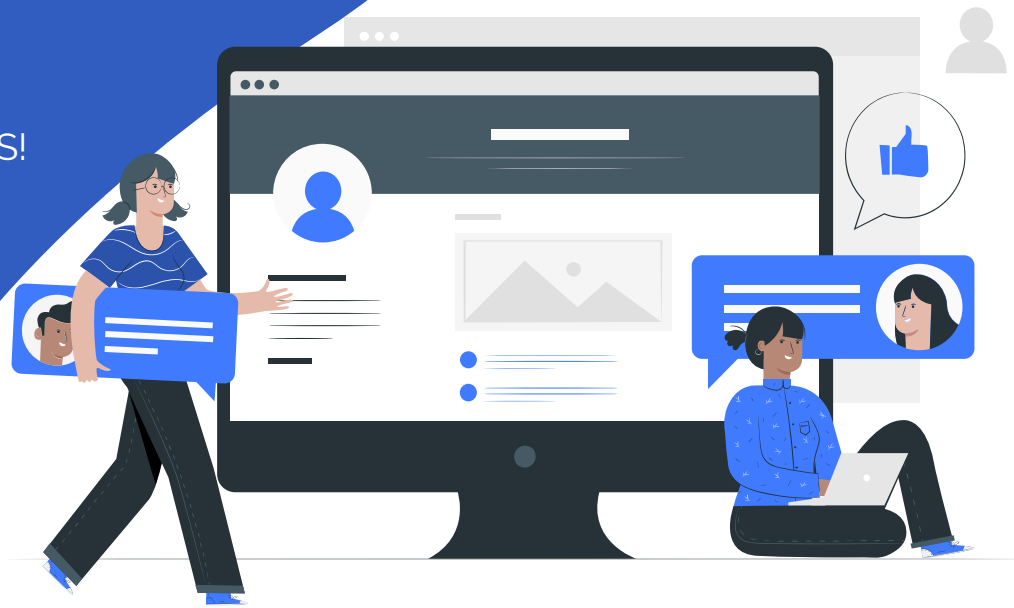
React

With an intro to UI/UX, HTML/CSS & JS!



Giang & Brian

× UTM CSSC

What We'll Cover

01

UI/UX

Why does a good user experience matter?

02

HTML/CSS

Refresh on basic HTML and DOM structures!

03

Javascript

Learn about basic Javascript

04

React Basics

Learn about components and more!

05

Dynamic Rendering

How do I keep my frontend flexible?

06

React Hooks

"Hook" into our component states!

and most importantly...

We'll Create a React App!

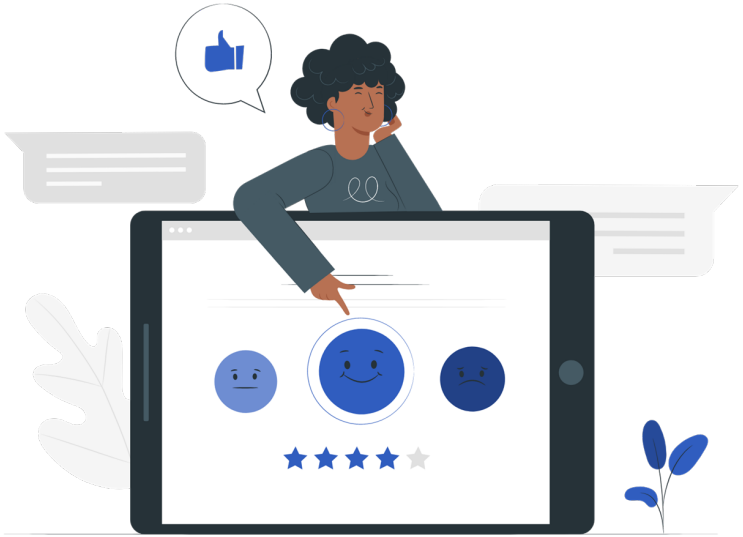
What is UI/UX?

UX (User Experience)

All aspects of **end-user interactions** with a service or product.

UI (User Interface)

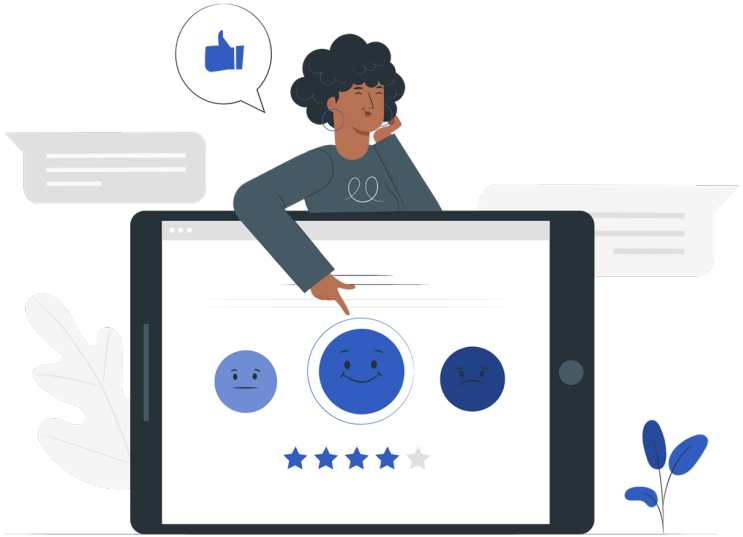
The **look**, **feel**, and **interactivity** of an experience.



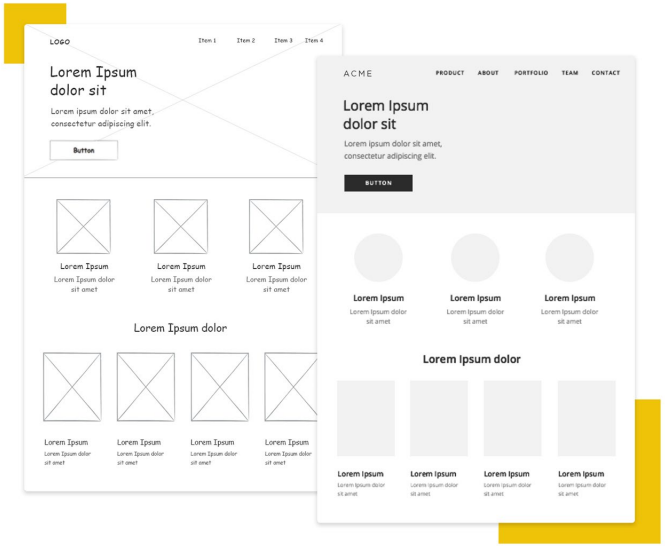
What is UI/UX?

- A product always begins with its UI/UX design.
- Consider user needs and wants
- Compose layouts and prototypes
- Requires fundamental design skills

A bad design is very noticeable!



Wireframing



Wireframing is a crucial part of the UI/UX process of frontend design.

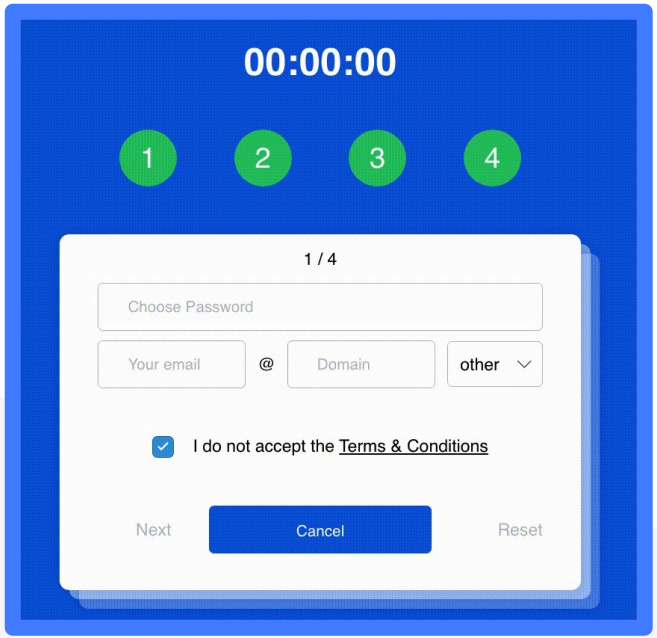
It could be thought of as a **prototyping** stage, creating a mockup that is either:

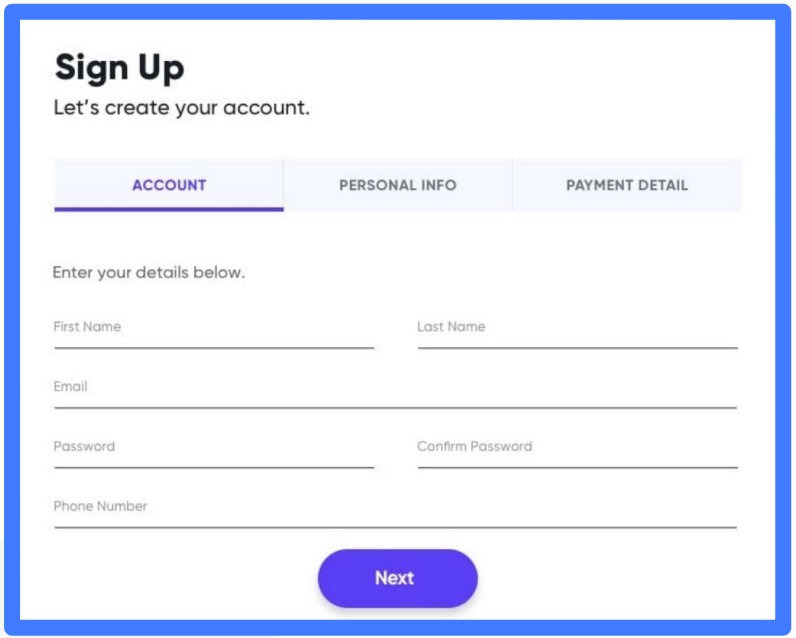
- **Low-fidelity:** used to create the layout of the the design.
- **High-fidelity:** builds off of the low-fidelity mockup, adds style elements.

A wireframe should always be made before writing any frontend code!

Good UI/UX Vs. Bad UI/UX

Spot which one's which!





Taking this into account...

Let's work towards building something ourselves!

HTML: A Refresher

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div class="container" id="workshop-box">
        <h1>Here's a...</h1>
        <img src="images/cat.jpg">
        <p>Cute cat!</p>
    </div>
</body>
</html>
```

HTML (HyperText Markup Language) is a **standard markup language** used to create websites.

An HTML document consists of a series of **elements** (**tags**).

Every HTML document uses a **DOM model**..

It's the most fundamental language for web development!

CSS: A Refresher

```
.container {
    width: 200px;
}

#workshop-box {
    font-family: 'Inter', Arial, sans-serif;
    border: 10px solid #305dbf;
    border-radius: 5px;
    padding: 10px 20px;
}

img {
    width: 100px;
    height: 100px;
}
```

CSS is used to **style HTML elements** that are displayed on the screen.

A CSS document consists of **selectors**.

The Result!

Here's a...

Cute cat!

JavaScript Overview

- Created to program web apps
- Designed to be easy to learn
- One of the world's most popular programming languages

We won't cover everything, just enough to program in React!

Variables in JavaScript

Variables are containers for **storing data**.

A variable is declared with:

- **const** if you would like the data to be **immutable**.
- **var** if you would like the data to be **mutable** and **function-scoped**
  - Nowadays, you don't usually use var!
- **let** if you would like the data to be **mutable** and **block-scoped**

```
1   var num1 = 1;
2   let num2 = 2;
3   const num3 = 3;
```

Data types in JavaScript

Data types define the **type** of a **variable**.

JavaScript is not strongly-typed language, meaning the same variable can hold different values.

```
let length = 16;                                  // Int
let lastName = "Giang";                           // String
lastName = 1;
let isHappy = true;                               // Boolean
let x = {firstName:"John", lastName:"Doe"};       // Object
let names = ["Brian", "Giang"];                   // Array
```

Functions (Normal vs. Arrow)

A function—like in other programming languages—is a piece of code that is executed when something calls it.

Arrow functions provide **closure** to the environment around it. It allows us to shorten function syntax, writing “const hello = () => {}” rather than “const hello = function() {}”!

```
1   function myFunction(p1, p2) {
2       return p1 * p2;
3   }
```

```
1   const hello = () => {
2     return "Hello World!";
3   }
```

JSX

- JSX is a **syntax extension** of JavaScript
- Used in React, but isn't required
- Allows us to write HTML elements in JavaScript
- Intertwines the markup and logic of web apps

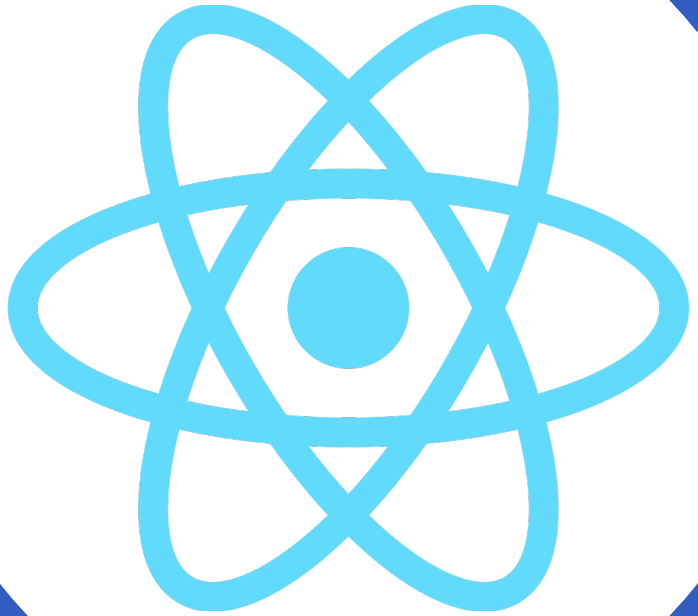
```
const name = 'Giang';
const element = <h1>Hello, {name}</h1>;
```

Finally...

Let's learn React!

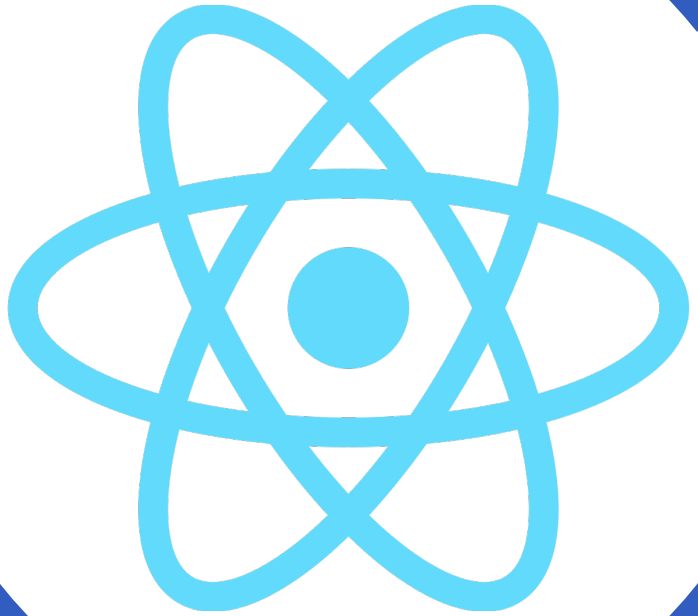
What is React?

- A JavaScript framework created by Meta
- Allows devs to quickly build user-friendly web apps with a modular infrastructure
- The most popular JavaScript framework



Why React?

- Easily organize your code into reusable **components**
- Can **dynamically render** websites without refreshing
- Due to its popularity, there is a lot of **community support**



Components

A component is a JavaScript class/function that **returns a React element**.

When creating a React component, keep **reusability** in mind! This is very important.

```
function NavBar () {
    return (
        <div>
            <div>
                <h1>Dialog</h1>
            </div>
            <div>
                <button>
                    Sign Up
                </button>
                <button>
                    Login
                </button>
            </div>
        </div>
    );
}
```

More About Components

Components can be rendered in two ways: with or without children.

With children:

```
<button>
        Login
</button>
```

Without children:

```
<NavBar />
```

```
function HeadingText (props) {
    return (
        <h1>
            {props.children}
        </h1>
    );
}

function NavBar () {
    return (
        <div>
            <div>
                <HeadingText>
                    Dialog
                </HeadingText>
            </div>
            <div>
                <button>
                    Sign Up
                </button>
                <button>
                    Login
                </button>
            </div>
        </div>
    );
}
```

More About Components

When creating a component, you can reuse it inside another component!

When you write code between tags, it is exposed to a component called **children**.

```jsx
export const Card = ({ url, title, desc}) => {
    return (
        <a href={url}>
            <h1>{title}</h1>
            <p>{desc}</p>
        </a>
    );
};
```

```jsx
<Card
    url="www.google.com"
    title="React 2022"
    desc="Welcome to the 2022 React Workshop!"
/>
```

More About Components

Finally, you can also create components with multiple properties!

The ({ url, title, desc }) seen is a process called **object destructuring**. This pulls the values out of the properties object (the {url, title, desc}!) and represents them as “url”, “title”, and “desc” respectively!

Wait but...

How do you dynamically render data?

Dynamic Rendering

Dynamic render allows us to **render components based on data**!

In the code below, we:

1. Insert JavaScript into the JSX (wrap it in { … })
2. Call map on the array of users, map:
    a. Iterates over the array
    b. Applies a function to each element
    c. Returns an array of results
3. Render a new Avatar object with each contributor's information

```
<HStack spacing="12px">
    {users && users.map((user, index) => {
        return (
            <Avatar key={index} name={user.login} src={user.avatar_url} />
        );
    })}
</HStack>
```

Hooks

- “Hook” into a component’s lifecycle and call methods for any state!
- Useful when a component needs to fetch data on load or perform actions on a page
- Examples:
  - useState
  - useEffect
  - useRef
  - useContext

Due to time, we won’t be covering the component lifecycle but it’s useful to know!

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "c
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```

Using useState

useState is used to **save the internal state** of a component. The hook has a state and setter function!

This data persists after each re-render, which is useful if a component needs to display different information!

It causes a re-render whenever the state is different from its previous state.

Let’s start coding!

Important Resources

Starter Code (this includes instructions for our project!):
https://github.com/utm-cssc/frontend-workshop-2022

React Resources:
- https://reactjs.org/

HTML/CSS Resources:
- https://www.w3schools.com/html/
- https://www.w3schools.com/css/
- https://www.w3schools.com/js/

UI/UX:
- https://grow.google/certificates/ux-design/
- any design course!

Resources

- https://reactjs.org/ (Official React Docs)
- https://www.w3schools.com/REACT/DEFAULT.ASP (W3S for react)
- https://www.w3schools.com/ (You can also use W3S to learn JS, HTML and CSS)
- Reach out to us!

Thanks for Coming!

This presentation template was created by **Slidesgo,** including icons by **Flaticon**, infographics & images by **Freepik** and illustrations by **Stories**
Credit to **Jarrod Servilla** for portions of the content in this workshop!