Typescript

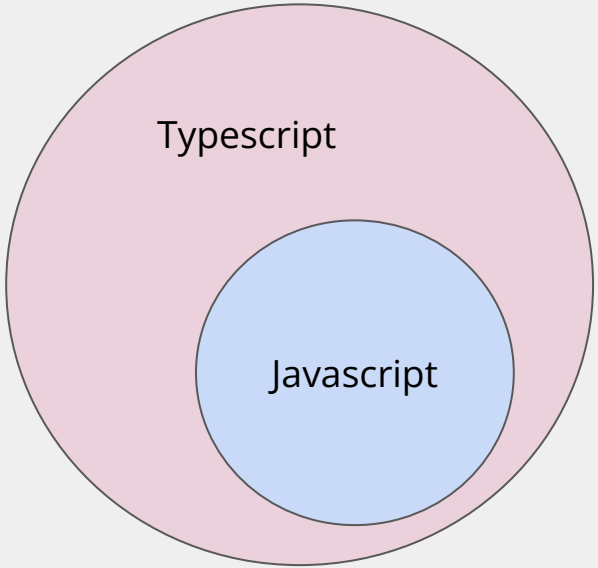
Vincent Huang

Reminders

- Hackathon tomorrow night from 7PM - 1AM (ET)
  - last chance to get live help from staff before milestone 2 is due
- Milestone 2 (MVP) is due **THIS COMING TUESDAY**, Jan 18 at 6PM sharp.
  - https://weblab.mit.edu/about/#milestones
  - ALL milestones are required to get credit or compete
- Read the pinned Piazza posts
  - skeleton code (typescript version also available)
  - lots of other useful resources!
- Tomorrow: How to deploy & more
  - Essential lecture, your MVP must be deployed!

What is Typescript?

- Language built on top of Javascript that enforces static typing
- Validates that your code works at compile-time
- Will save your life when debugging



Why do we care about static typing?

Here are some examples of issues that static typing can catch:

- Missing or unnecessary prop values
- Similarly named variables or functions
- Null value behavior - the “billion dollar mistake”
- Overloaded operators (eg. addition, comparison)

These problems may seem “obvious”, but more complex codebases = higher chance of introducing bugs!

Works with primitive types

```
1   let message: string = "Hello world!";
2   message = 1;
```

```
let message: string

Type 'number' is not assignable to type 'string'. (2322)

Peek Problem (⌥F8)    No quick fixes available

message = 1;
```

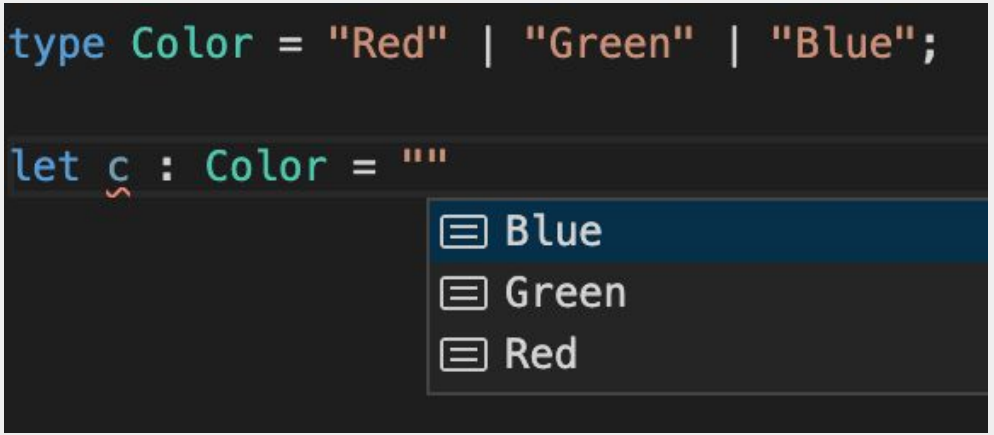
- string, boolean, number

More types

Arrays

```
1    let message: string[] = ["1", "2", "3"];
```

Enums



Types in general

- define each property and its type
- denote optional params with ?

```
type User = {
  _id: string;
  name: string;
  is_admin?: boolean;
}

const user : User = {
  _id: "555",
  name: "Vincent"
}
```

Use types with other types

- define an array of users

```
type User = {
  _id: string;
  name: string;
  is_admin?: boolean;
}

const user : User = {
  _id: "555",
  name: "Vincent"
}

const users : User[] = [
  {
    _id: "556",
    name: "Akshaj"
  },
  user
]
```

Extend types

```
type UserLogin = User & { password: string }

const userLogin : UserLogin = {
  _id: "555",
  name: "Vincent",
  password: "password"
}
```

Typed functions

- type parameters and output

```
65    const getComments = (id: string): Comment[] => {
66        return [];
67    }
```

```
type Props = {
  _id: string;
  creator_name: string;
  content: string;
  handleAdd: () => void;
}
```

Typed functions

- Use with built in React events as well

```
const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
  setValue(event.target.value);
}
```

Use with async functions too

```
65    const getComments = (id: string): Promise<Comment[]> => {
```

Easily integratable with your projects!

- Works well with defining React prop/state types
- Can integrate slowly/partially into your projects; you don't need to write entirely in Javascript or entirely in Typescript

Typed props and state

```
type ProfileProps = {
  userId: string;
};

const Profile = (props: ProfileProps) => {
  const [user, setUser] = useState<User | undefined>(undefined);
  const [catHappiness, setCatHappiness] = useState(0);
```

Define types for chats

- export to reuse in other files

```
type User = {
  _id: string;
  name: string;
}

type Message = {
  sender: User;
  content: string;
}

type ChatData = {
  messages: Message[];
  recipient: User;
}

export {User, Message, ChatData};
```

Typescript Setup Info

How to add typescript to an existing React project:

- https://www.sitepoint.com/how-to-migrate-a-react-app-to-typescript/
- (or just use the Typescript skeleton we provided for you)

Typescript Config Settings:

- Can be changed in `tsconfig.json`
- Describe “how strict” Typescript should be

git checkout typescript-functional
npm install

use the typescript skeleton if you want!
(typescript branch in the weblab-skeleton repo)

demo with Catbook!