# CSE 331
# Software Design & Implementation

James Wilcox

Autumn 2021

Modern Web UIs

## Dynamic Web Content

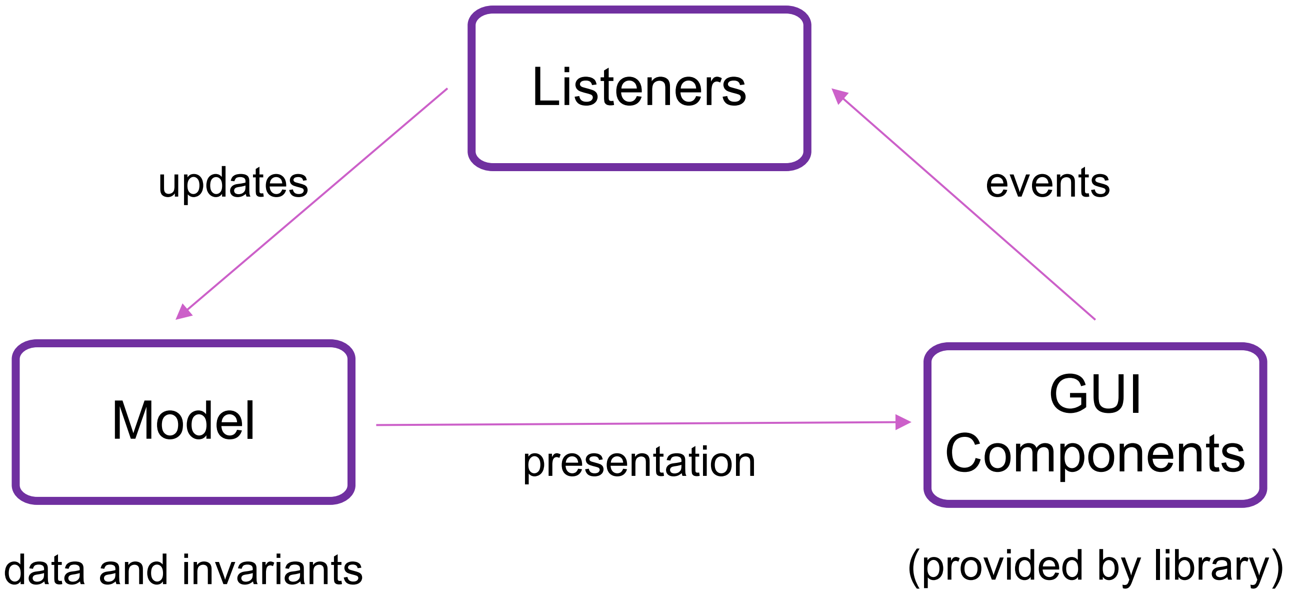
- Earlier example had a fixed set of components.
  - same for iPhone / Android apps

- More realistic apps need to change the set of components displayed on the screen dynamically
  - consider Gmail as an example
  - need the components to come from code

## JS Example

register-js/index.js

# Structure of GUI Application



## Problems

These tools can be used to write Gmail

But it has a number of problems…

1. Lack of tool support
   - no checking of types, tags, etc.
2. No support for modularity
   - all the code and UI in a single file
3. More boilerplate
   - minimized JS file would change function names
   - need to call `btn.addEventListener` by hand

## JS Modules

- EcmaScript6 (ES6) added support for modules.
- Each file is a separate unit (“namespace”)
- Only exported names are visible outside:

```
export function average(x, y) { …
```

- Others can import using:

```
import { average } from ‘./filename’;
```

## ES6 Example

register-js2/…

# JS Classes

- ES6 added new syntax for classes:

```
class Foo {
  constructor(val) {
    this.secretVal = val;
  }

  secretMethod(val) {
    return val + this.secretVal;
  }
}
```

## More from ES6 Example

register-js2/…

# JS vs Java Classes

- JS method signatures are just the name
  - JS objects are just HashMaps
  - field names are the keys

```
obj.avg(3, 5)
```

- Java methods signatures are name + arg types
  - e.g., `avg(int,int)`

- JS has only one method with a given name
  - language allows different numbers of arguments
    - missing arguments are undefined
  - can strengthen a spec by accepting a wider set of possible input types

# Problems

These tools can be used to write Gmail

But it has a number of problems…

1. Lack of tool support
   - no checking of types, tags, etc.
2. No support for modularity
   - all the code and UI in a single file
3. More boilerplate
   - minimized JS file would change function names
   - need to call `btn.addEventListener` by hand

## TypeScript

- Adds type constraints to the code:
  - arguments and variables

    ```
    let x : number = 0;
    ```

  - fields of classes

    ```
    quarter: string;
    ```

- **`tsc`** performs type checking

- Creates version has type annotations removed

# TypeScript Types

- Basics from JavaScript:
  number, string, boolean, string[], Object
- But also
  - specific classes Foo
  - tuples: `[string, int]`
  - enums (as in Java)
  - allows null to be included or excluded (unlike Java)
  - `any` type allows any value
  - …

## TypeScript Example

register-ts/…

## TypeScript

- Type system is unsound
  - can't promise to find prevent all errors
  - can be turned off at any point with any types
    - `x as Foo` is an unchecked cast to `Foo`
    - `x!` casts to non-null version of the type (useful!)

- Full description of the language at `typescriptlang.org`