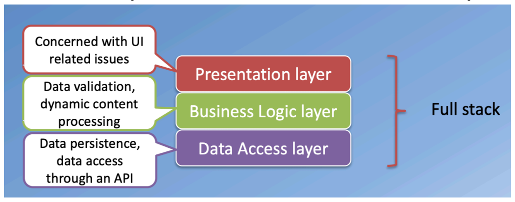
REVIEW of Front-End Web Development with React

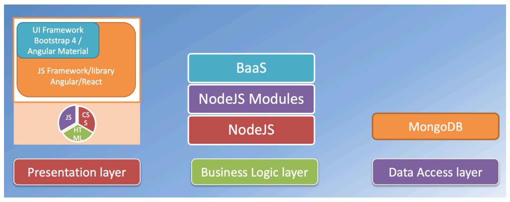
6/23/2019 Rui Li

Week1:

1. Introduction to Full Stack Web Development

Front end/ Client-side VS Back end/Server-side


Concerned with UI related issues
Data validation, dynamic content processing
Data persistence, data access through an API
Presentation layer
Business Logic layer
Data Access layer
Full stack



2. Setting up your Development Environment: Git and Node

<u>Git</u>: Version Control 把所有写的程序版本都保存起来

<u>NodeJS</u>: JavaScript runtime （翻译软件，翻译 JS）可以让 JS 这种前端语言控制 server-side

<u>NPM</u>: 网上资源库的一个集合站，帮助从网上下载资源包

3. Introduction to React

<u>Frameworks</u>: 插 code 进去 [FLUX/REDUX]

<u>Libraries</u>: 调用现有的成熟的 behavior [React]

<u>React</u>: element, component, virtual DOM

<u>JSX</u>: extension to JavaScript, kind of a combination of JS & html in our case

4. React Components

<u>Component</u>: split UI into independent, reusable pieces

<u>State</u>: components used to store its own local information passed as <u>PROPS</u>

-declared in constructor

-use setState() to change state

<u>Lifecycle</u>: Mounting, updating, unmounting

Week2:

1. React component Types

<u>Presentational</u>: concerned with rendering view [do not maintain their own local state]

<u>Container</u>: data fetching & state updates [communicate with data sources]

<u>Class component</u>: extend react component/render()/can have local state

<u>Functional component</u>: simplest/receives props as param/no local state or lifecycle/return()

2. React Router

<u>Virtual Dom</u>: created completely from scratch on every setState [React Object/ light weighted/tree data structure/fast/use diffing algorithm to update DOM] state 改变和 DOM 改变的中间一个处理层

<u>Router</u>: collection of navigational components [use URL to enable navigation among views]

Link vs NavLink: NavLink 感觉是用来做链接的，导航栏里用 NavLink

3. Single Page Application

<u>Single Page Application</u>: web application or web site fits in a single page [No need to reload entire page] 一次性下载好大部分数据，后面和 server 只需要传递部分信息

<u>React Router Parameters</u>: [建立子网页 a list] 子网页的定义 <Link to{`/menu/${dish.id}`>

Path="/menu/:dishId

Week3:

1. Controlled Forms

Form: entry (e.g.: login/ submit information)

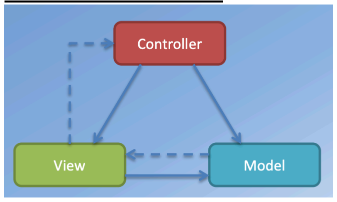
Controlled Forms: react component control the form (single source of truth/state tying with component state/**every state mutation have an associated handler function**)

2. Uncontrolled Forms

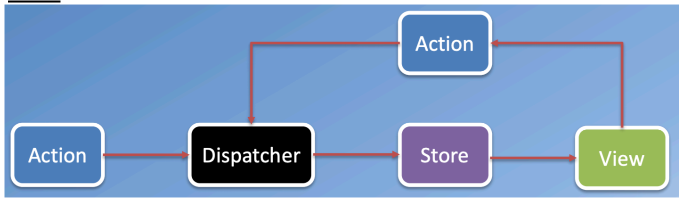
Uncontrolled Forms: easier, use a ref to get form values from DOM, instead of writing handler function (innerRef={(input)=>this.username.input})

3. Introduction to Redux

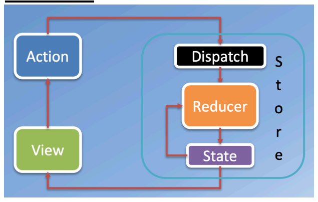
MVC Framework:



Flux: unidirectional data flow



Redux:



Redux: Framework/take previous state and action and return next state

4. React Redux Form

LocalForm: Map form model to local state of the component (when no need to form data persistence across component mounting/unmounting) (LocalForm/control.text)

Week4:

1. Redux Actions

Redux Action: type+payload

Actions: payloads of information (send data from your application to the store through dispatch())

Action Creators: Functions that create actions

Reducer: take previous state and action and return next state (splitting & combining redusers)

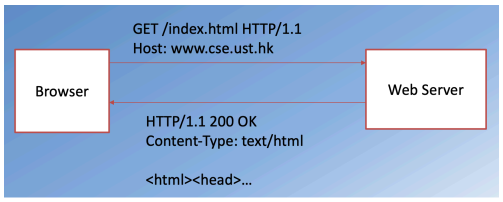
2. Redux Thunk

Redux Middleware: provides capability to run code after an action is dispatched, but before it reaches the reducer

Thunk: a subroutine used to inject an additional calculation in another subroutine

Redux Thunk: return **function** instead of an action **in action creators**, inner function receives dispatch() and getState() store methods

3. Client-Server Communication [Write applications recognizing the unexpected delays]



```
GET /index.html HTTP/1.1
Host: www.cse.ust.hk
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html><head>...
```

HTTP (Hypertext Transfer Protocol): a client-server communications protocol [response in XML or JSON]

JSON (JavaScript Object Notation): lightweight data interchange format

Web Services: a system designed to support interoperability of systems connected over a network (SOAP & REST)

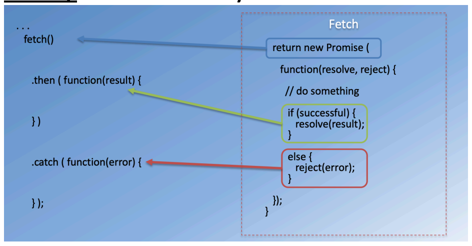
REST (Representational State Transfer): a style of software architecture for distributed hypermedia systems, outline how resources are defined and addressed

-HTTP methods explicitly
-stateless
-expose directory structure-like URIs
-transfer using XML/JSON

4. Fetch

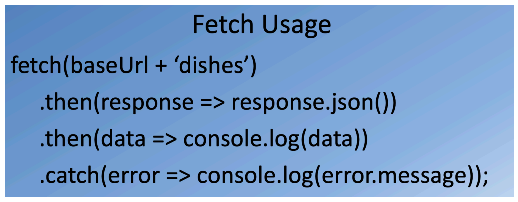
Promise: Mechanism that supports asynchronous computation [先调用方法，但允许执行不成功]

Proxy: a value may be available now, or in the future, or never



```
. . .
 fetch()

   .then ( function(result) {

   } )

   .catch ( function(error) {

   } );
```

```
return new Promise (
   function(resolve, reject) {
    // do something
    if (successful) {
      resolve(result);
    }
    else {
      reject(error);
    }
  });
}
```



Fetch Usage

```
fetch(baseUrl + ‘dishes’)
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.log(error.message));
```

Fetch: interface for fetching resources, promise based (modern replacement for XMLHttpRequest())

5. React Animations