모바일 프로그래밍

Vue.JS Tutorial

Phase I: Overview

- HTML5
- CSS
- Javascript
  - Vue.js2

Main Technologies

- Javascript + Vue.JS2
  - Programming language for HTML/Web
  - Vue.JS: Progressive web framework
- Frontend of your application
  - Front-end behavior
- Learning material
  - Vue.JS Guide: https://kr.vuejs.org/

Vue.js Guide

- 공식 한글 가이드
  - https://kr.vuejs.org/v2/guide/

- Webpack 가이드
  - https://vuejs-templates.github.io/webpack/

Vue.js 미니 튜토리얼

- 강의자료 내 tutorial.html 이용
  - Vue instance 구조
    - Data, method, etc
  - Template 문법
    - v-if, v-for, v-model, etc.
    - Computed property
    - Watch, filter
    - Lifecycle hook

Vue.js 미니 튜토리얼

Vue 인스턴스

- Vue App 생성을 위한 기본 단위
- 속성 및 메소드 포함
- 기타 추가 기능 포함
- 라이프사이클 존재
  - created() 콜백에 내용 추가해보기
  - https://vuejs.org/v2/guide/instance.html#Lifecycle-Diagram

참고

- https://kr.vuejs.org/v2/guide/instance.html

Vue.js 미니 튜토리얼

Template 문법

- 문자열 보간법({{ msg }})
  - <h1> 태그에 문자열 추가해보기
- 속성 바인딩(v-bind)
  - 속성과 데이터를 서로 연결
  - 약식 표현: “:attribute이름”
  - <h1> 태그의 title 속성 추가해보기

```
<h1 v-bind:title=msg>
```

참고

- https://kr.vuejs.org/v2/guide/syntax.html

Vue.js 미니 튜토리얼

클래스/스타일 바인딩(v-bind)

- 클래스와 데이터를 서로 연결
- 약식 표현: ":class, :style"
- <h1> 태그에 클래스/스타일 바인딩해보기

```
<h1 … :class=theme :style={fontSize:fontSize}> … </h1>
```

참고

- https://kr.vuejs.org/v2/guide/class-and-style.html

Vue.js 미니 튜토리얼

입력폼에 대한 양방향 바인딩 (v-model)

- 입력폼의 입력값과 데이터를 양방향 바인딩
- 예1) 텍스트 입력폼과 msg 변수의 양방향 바인딩

```
<input type="text" v-model="msg">
```

- 예2) 라디오 버튼과 currentBranch 변수의 양방향 바인딩

```
<input type="radio" :id="branch" :value="branch" name="branch"
 v-model="currentBranch">
```

참고

- https://kr.vuejs.org/v2/guide/forms.html

Vue.js 미니 튜토리얼

리스트 렌더링 (v-for)

- 배열 기반의 리스트 렌더링
- 예) 튜토리얼 내 브랜치 리스트

```
<template v-for="branch in branches">
            <input ...>

            <label>...</label>
</template>
```

- 예) feature 브랜치 추가해보기
  - 코드 내에서 정적으로 Branches 배열에 ‘feature’ 추가

참고

- https://kr.vuejs.org/v2/guide/list.html

Vue.js 미니 튜토리얼

리스트 렌더링 (v-for)

- 예) feature 브랜치 추가해보기
  - 동적으로 Branches 배열에 ‘feature’ 추가

배열 변경 감지

- push(), pop(), shift(), unshift(), splice(), sort(), reverse()
- 위 메소드들을 사용해야 동적으로 배열 변경을 감지함

참고

- https://kr.vuejs.org/v2/guide/list.html

Vue.js 미니 튜토리얼

Watch (감시자)

- 특정 데이터의 변경 감지 및 그에 따른 행위 설정
- 예) methods 내 fetchData function의 xhr.send() 주석 해제
  - 브랜치 변경 시 특정 git repo의 해당 branch의 커밋을 불러와 출력!

```
watch: {
    currentBranch: 'fetchData'
},
```

  - currentBranch의 값이 변경되면? fetchData 함수를 실행!

참고

- https://vuejs.org/v2/guide/computed.html#Watchers

Vue.js 미니 튜토리얼

조건부 렌더링 (v-if)

- v-else, v-else-if 등도 존재
- commits이 존재하면 “clear commits” 버튼을 추가해보기

```
<h1 v-if="ok">Yes</h1>
<h1 v-else>No</h1>
```

```
<button v-if="commits.length" > Clear Commits </button>
```

참고

- https://kr.vuejs.org/v2/guide/conditional.html

Vue.js 미니 튜토리얼

이벤트 핸들링 (v-on)

- 특정 이벤트 발생 시 실행할 코드 또는 메소드 설정 가능
- 약식 표현: “@이벤트”
- 예) “clear commits” 버튼을 클릭하면 commits을 []로 갱신하기

```
<button … @click=clearCommits> Clear Commits </button>
```

- clearCommits function 추가

```
clearCommits: function(){
    this.commits=[];
}
```

참고

- https://kr.vuejs.org/v2/guide/events.html

Vue.js 미니 튜토리얼

계산된 속성

- Computed 키워드 이용
- 일종의 캐시 데이터
  - 의존성이 있는 데이터의 변경시에만 호출됨
- Branch의 개수를 화면에 뿌려보자!
  - Method

```
branchCount: function(){
    console.log("method called");
    return this.branches.length;
}
```

  - computed

```
computed: {
    computedBranchCount(){
        console.log("computed called");
        return this.branches.length;
    }
}
```

Vue.js 미니 튜토리얼

컴포넌트

- 재사용 가능한 일종의 Vue instance
- HTML 태그처럼 이름을 가질 수 있음
- 버튼 컴포넌트 등록 및 사용해보기

```
var buttonComponent = {
    data: function () {
        return {
            count: 0
        }
    },
    template: '<div><button v-on:click="count++"><p>You
    clicked me {{ count }} times.<p></button></div>'
}
```

참고

- https://kr.vuejs.org/v2/guide/components.html

Vue.js 미니 튜토리얼

컴포넌트

- Single root element 여야만 함!
  - Template의 최상위 태그는 한 개여야 한다
- Data는 function으로 존재해야 함

컴포넌트 등록

```
var buttonComponent = {
        …
}
var demo = new Vue({

    el: '#demo', // demo 영역에 대한 Vue instance
    components: {
        'bt-comp':buttonComponent
    },
    …
}
```

Vue.js 미니 튜토리얼

컴포넌트 간 메시지 전달

- 상위 컴포넌트 → 하위 컴포넌트: Props 메커니즘 이용
- 하위 컴포넌트가 할 일
  - Props 속성으로 받고 싶은 데이터의 이름 설정

```
var buttonComponent = {
    data: function () {
        …
    },
    props:['branch’],
    template: '<div><button v-
    on:click="count++"><p> You clicked me {{ count }} times
    .</p></button><p> Current branch is {{ branch }}. </div
    >'

}
```

Vue.js 미니 튜토리얼

컴포넌트 간 메시지 전달

- 상위 컴포넌트 → 하위 컴포넌트: Props 메커니즘 이용
- 상위 컴포넌트가 할 일
  - 하위 컴포넌트의 props에 정의된 속성명으로 메시지 전달

```
<bt-comp :branch=currentBranch></bt-comp>
```

  - <bt-cpmp> 컴포넌트에게 현재 currentBranch의 값을 branch라는 속성명으로 전달

Vue.js 미니 튜토리얼

컴포넌트 간 메시지 전달

- 하위 컴포넌트 → 상위 컴포넌트: emit 메커니즘 이용
- 하위 컴포넌트: $emit(event 이름, 전달할 메시지) function 사용

```
var buttonComponent = {
        data: function () {
            return {
                count: 0
            }
        },
        watch:{
            count: function(){
                this.$emit('msg',this.count);
            }
        },
        props: ['branch'],
        template: '<div><button @click="count++"><p> You clicked me
{{ count }} times.</p></button><p> Current branch is {{ branch }}.</
p> </div>'
    }
```

Vue.js 미니 튜토리얼

컴포넌트 간 메시지 전달

- 하위 컴포넌트 → 상위 컴포넌트: emit 메커니즘 이용
- 상위 컴포넌트: 하위 컴포넌트에 대한 이벤트 핸들링 메커니즘 사용
  - $event: 이벤트를 통해 함께 하위 컴포넌트로부터 전달된 메시지

```
<bt-comp
  :branch=currentBranch
  @msg="this.alert($event);">
</bt-comp>
```

- <bt-comp> 컴포넌트에서 “msg”라는 이벤트 발생 시 특정 행위 수행
  - 여기서는 msg 이벤트를 통해 전달된 메시지를 alert

Vue.js 미니 튜토리얼

- 싱글 파일 컴포넌트
  - 템플릿 안의 내용이 길어지면…?
  - 사용해야 할 컴포넌트들이 많아지면…?
  - 스타일은 또 어떻게 해야 하나…?

- 각 컴포넌트를 단일 파일로 관리
  - Webpack에 의하여 모듈화
  - .vue 파일로 컴포넌트를 관리하며 Webpack에 의하여 다시 빌드됨

Vue.js 미니 튜토리얼

- 싱글 파일 컴포넌트 구조
  - <template>
    - 주로 View/Presentation 구성
  - <script>
    - Vue component 로직 구성
  - <style>
    - CSS 구성
  - 그 외 나머지는 모두 동일

- 기본과제 템플랫 앱 코드를 통해 더 알아봅시다.