Vue.js

starting vue.js

- 자바스크립트 프레임워크
- https://vuejs.org/
- https://github.com/vuejs/awesome-vue
- Data Driven
- Template, Directive
- Data binding
- Nuxt.js, VuePress
- Vuex, Vue Router
- https://curated.vuejs.org/
- https://element.eleme.io/
- https://onsen.io/
- Vue Devtool @chrome store
- 가상 DOM

Vue.js 다운로드

i

- https://vuejs.org/js/vue.js

CDN 이용

```
<!-- development version, includes helpful console warnings -->
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
```

```
<!-- production version, optimized for size and speed -->
<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
```

Vue.js 기본 구조

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="utf-8">
  <title>Vue.js App</title>
  <link href="main.css" rel="stylesheet">
</head>
```

```
<body>
  <div id="app">
    <!-- template 출력 -->
  </div>
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <script src="main.js"></script>
</body>
</html>
```

```
var app = new Vue({
  el: '#app'
})
```

기본 기능

- 텍스트 바인딩

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue.js!'
  }
})
```

```
<p>{{ message }}</p>
```

- 반복 렌더링; v-for directive, list in data:
- 이벤트 사용; v-on directive, methods:
- 입력 양식과 동기화; v-model directive, data:
- 조건 분기; v-if directive, boolean in data:
- 트랜지션과 애니메이션; <transition>
- {디렉티브}{:매개변수}{.장식자}={값}; ex) v-bind:value.sync = “message”

Options in Vue Objects

```
var app = new Vue({
```

```
  // el
  el: '#app',
  // data
  data: {
    message: 'Vue.js'
  },
  // computed
  computed: {
    computedMessage: function() {
      return this.message + '!'
    }
  },
  // lifecycle hooks
  created: function() {
    // something to what you do
  },
  // methods
  methods: {
    myMethod: function() {
      // something to what you do
    }
  }

})
```

Lifecycles	
Methods	**Timing**
beforeCreate	인스턴스가 생성되고, 리액티브 초기화가 일어나기 전
created	인스턴스가 생성되고, 리액티브 초기화가 일어난 후
beforeMount	인스턴스가 마운트되기 전
mounted	인스턴스가 마운트된 후
beforeUpdate	데이터가 변경되어 DOM에 적용되기 전
updated	데이터가 변경되어 DOM에 적용된 후
beforeDestroy	Vue 인스턴스가 제거되기 전
destroyed	Vue 인스턴스가 제거된 후
errorCaptured	임의의 자식 컴포넌트에서 오류가 발생했을 때

Data Binding

- 템플릿에 사용하는 모든 데이터를 리액티브 데이터로 정의
- 리액티브 데이터 정의; 컴포넌트의 data 옵션에 정의
- 렌더링; mustache {{ 속성이름 }}
    - 표현식만 가능, 문장식은 불가능
    - 3항 연산자의 경우 산출 속성 computed 사용을 권장
    - 문자열이나 숫자를 변환할 때는 필터를 권장
    - 속성에는 사용 불가, 속성에 바인딩 하려면 v-bind 디렉티브 사용
    - data 상태 json으로 출력; <pre> <nowiki>
                                                data
      </pre>

- this; 콜백으로 익명 함수를 사용하거나, 다른 라이브러리와 함께 사용할 경우 this 변경되므로 사용에 주의
- 클래스 이름에 하이픈을 넣을 때는 작은 따옴표(')로 감쌈.
- 템플릿에서 조건 분기; v-if → 주석처리, v-show → 스타일로 보이지 않게만.
- v-if, v-else-if, v-else; key 설정
- 요소를 반복해서 렌더링; v-for = "<각 요소를 할당할 변수 이름> in <반복 대상 배열 또는 객체>" v-bind:key=".."
    - 값, 키, 인덱스 순
    - 반복 처리 순서; Object.keys()의 순서에 기반
    - v-for 안에 v-if
- 리스트
    - 추가; push, unshift ex) this.list.push(새로운 값)
    - 제거; splice ex) this.list.splice(index, 1)
    - push, pop, shift, unshift, splice, sort, reverse
    - Vue.set 메서드; this.$set(변경할 데이터, 인덱스 또는 키, 새로운 값)
    - this.list.filter(function(el) { ... })

v-bind 장식자	
장식자	**의미**
.prop	속성 대신에 DOM 속성으로 바인딩, DOM 속성과 직접 바인딩
.camel	케밥케이스[1] 속성 이름을 카멜 케이스로 변환
.sync	양방 바인딩

- $el을 사용해 DOM을 직접 참조 가능, 라이프사이클 중 mounted 이후부터 사용가능.
- 속성 ref와 $ref를 사용해 참조
- $el과 $ref는 일시적 변경

템플릿 제어 디렉티브	
디렉티브	**설명**
v-pre	템플릿 컴파일 생략
v-once	한 번만 바인딩
v-text	Mustache 대신 텍스트 콘텐츠로 렌더링
v-html	HTML 태그를 그대로 렌더링
v-cloak	인스턴스 준비가 끝나면 제거

Event Handling

- Event Handler, Handle
- v-on

이벤트 장식자	
장식자	**설명**
.stop	event.stopPropagation()을 호출
.prevent	event.preventDefault()를 호출
.capture	캡처 모드로 DOM 이벤트를 핸들
.self	이벤트가 해당 요소에서 직접 발생할 때만 핸들러를 호출
.native	컴포넌트의 루트 요소 위에 있는 네이티브 이벤트를 핸들
.once	한 번만 핸들

이벤트 장식자	
장식자	**설명**
.passive	{passive: true}로 DOM 이벤트를 핸들

클릭 이벤트	
장식자	**설명**
.left	마우스 왼쪽 버튼으로 눌렀을 때만 핸들러 호출
.right	마우스 오른쪽 버튼으로 눌렀을 때만 핸들러 호출
.middle	마우스 중간 버튼으로 눌렀을 때만 핸들러 호출

- 키 장식자

키 코드 별칭	
별칭	**의미**
.enter	Enter(엔터) 키를 눌렀을 때
.tab	Tab(탭) 키를 눌렀을 때
.delete	Delete(딜리트) 키를 눌렀을 때
.esc	ESC 키를 눌렀을 때
.space	Space(스페이스) 키를 눌렀을 때
.up	화살표 위 키를 눌렀을 때
.down	화살표 아래 키를 눌렀을 때
.left	화살표 왼쪽 키를 눌렀을 때
.right	화살표 오른쪽 키를 눌렀을 때

시스템 장식자	
별칭	**의미**
.ctrl	Ctrl(컨트롤) 키가 눌린 경우
.alt	Alt(안트) 키가 눌린 경우
.shift	Shift(시프트) 키가 눌린 경우
.meta	Meta(메타) 키가 눌린 경우

- 양방향 데이터 바인딩; v-model
  - 데이터 바인딩으로 요소의 value 속성 변경
  - 이벤트 핸들링으로 받은 값을 데이터에 대입
  - file 타입에는 사용 불가

v-model 장식자	
장식자	**의미**
.lazy	input 대신 change 이벤트 핸들링
.number	값을 숫자로 변환
.trim	값 양쪽에 있는 쓸데없는 공백 제거

데이터 가공 및 감시

- 산출 속성; 리앤티브 데이터를 기반으로 결과 캐시
- getter and setter; get:, set:
- 워처(watcher); 특정 데이터 또는 산출 속성의 상태를 감시해서 변화가 있을 때 등록한 처리를 자동을 실행

```
new Vue({
  // ..
  watch: {
  <감시할 데이터>: function(<새로운 값>, <이전 값>) {
      // value가 변화했을 때 하고 싶은 처리
    },
    'item.value': function(newVal, oldVal) {
  // 객체의 속성도 감시할 수 있음
    },
    list: {
      handler: function(newVal, oldVal) {
        // list가 변경될 때 하고 싶은 처리
      },
      deep: true,  // 속성: deep, 값: Boolean, 네스트된 객체도 감시할지
      immediate: true  // 속성: immediate, 값: Boolean, 처음 값을 읽어 들이는 시점에도 호출할지
    }
  }
})
```

- 인스턴스 메서드로 등록; this.$watch → 워처 제거
- 필터; 문자 수를 반올림하거나 쉼표를 넣는 등의 테스트 기반 변환 처리에 특화된 기능. v-bind 디렉티브 값 뒤에 파이프(|)로 연결해서 사용
  - 전역 필터; Vue.filter 사용해서 등록
- 사용자 정의 디렉티브; v- 프리픽스를 붙여 사용
  - directives:
  - 전역 등록; Vue.directive

사용자 정의 디렉티브에서 사용할 수 있는 훅	
메서드 이름	**시점**
bind	디렉티브가 처음 요소와 연결될 때
inserted	연결된 요소가 부모 Node에 삽입될 때
update	연결된 요소를 내포하고 있는 컴포넌트의 VNode가 변경되었을 때
componetUpdated	내포하고 있는 컴포넌트와 자식 컴포넌트의 VNode가 변경되었을 때
unbind	연결되어 있는 요소로부터 디렉티브가 제거될 때

훅 매개변수	
매개변수	**내용**
el	디렉티브가 연결되어 있는 요소
binding	배인드된 값, 매개변수, 장식자가 들어 있는 객체
vnode	요소에 대응되는 VNode
oldVnode	변경 이전의 VNode(update 또는 componentUpdated에서만 사용 가능)

binding의 속성	
속성	**설명**
arg	매개변수
modifiers	장식자 객체
value	새로운 값
oldValue	이전 값(update 또는 componentUpdated에서만 사용 가능)

- nextTick; Vue.nextTick, this.$nextTick

Components

- Vue.component; 정의 → 사용
- 컴포넌트 간 통신;
  - 부모(속성, on) ↔ 자식(props, $emit)
  - 이벤트 버스
- 슬롯(slot); 부모가 되는 컴포넌트 측에서 자식이 되는 컴포넌트의 템플릿 일부를 끼워넣는 기능.
  - 이름 있는 슬롯
  - 스코프 있는 슬롯
- 컴포넌트의 v-model, .sync로 양방향 데이터 바인딩
- 템플릿; template 옵션, inline-template, text/x-template + 선택자, 단일 파일 컴포넌트, 렌더링 함수
- 템플릿이 DOM으로 인식되는 경우; 컴포넌트 명명 규칙, HTML 내포 가능한 요소 규칙
- 함수형 컴포넌트, 동적 컴포넌트, 공통적인 처리를 등록하는 믹스인

Transition & Animation

- 트랜지션
  - 단일 요소 트랜지션

트랜지션 클래스의 시점	
Enter 계열의 클래스	**대상 요소가 DOM에서 삽입될 때의 트랜지션 페이즈**
.v-enter	대상 요소가 DOM에 삽입되기 전에 추가되며, 트랜지션이 종료될 때 사라짐. Enter 액티브 상태
.v-enter-to	트랜지션이 실제로 시작될 때 추가되며, 트랜지션이 종료될 때 사라짐. Enter의 종료 완료
.v-enter-active	대상 요소가 DOM에 삽입되기 전에 추가되며, 트랜지션이 종료될 때 사라짐. Enter의 액티브 상태
Leave 계열의 클래스	**대상 요소가 DOM에서 제거될 때의 트랜지션 페이즈**
.v-leave	트랜지션 시작 전에 추가되며, 트랜지션 개시 때는 사라짐. Leave 시작 상태
.v-leave-to	트랜지션 시작 전에 추가되어, 트랜지션이 종료되었을 때 사라짐. Leave 종료 상태
.v-leave-active	트랜지션 시작 전에 추가되어, 트랜지션이 종료되었을 때 사라짐. Leave의 액티브 상태

- 트랜지션 클래스의 기본
  - 추가할 때; .v-enter → .v-enter-to
  - 제거할 때; .v-leave → .v-leave-to

Enter와 Leave 시점 변경	
모드	**동작**
in-out	Enter 트랜지션이 종료되고 나서 Leave 트랜지션 시작
out-in	Leave 트랜지션이 종료되고 나서 Enter 트랜지션 시작

- 리스트 트랜지션; 이동 트랜지션, Leave와 Move가 동시에 적용
- SVG 트랜지션

- 트랜지션 훅

사용할 수 있는 트랜지션 훅	
Enter 계열 훅	**시점**
before-enter	DOM에 요소가 추가되기 전
enter	.v-enter가 있는 DOM에 요소가 추가된 후
after-enter	트랜지션이 끝나거나 또는 enter에서 done()을 호출한 후
enter-cancelled	enter 페이즈가 중간에 취소되었을 때
Leave 계열 훅	**시점**
before-leave	클래스가 추가되기 전
leave	.v-leave가 추가된 후
after-leave	DOM에서 요소가 제거된 후 또는 leave에서 done()을 호출한 후
leave-cancelled	leave 페이즈가 중간에 취소되었을 때

References

- 고양이도 할 수 있는 Vue.js 지원 페이지
- 基礎から学ぶ Vue.js 書籍用サポートページ 基礎から学ぶ Vue.js 書籍用サポートページ

1)

kebab-case, lisp-case, spinal-case; 하이픈으로 구분

From:
https://theta5912.net/ - **reth**

Permanent link:
https://theta5912.net/doku.php?id=public:computer:vuejs&rev=1629794392

Last update: **2021/08/24 17:39**