**HTML5, CSS3, SASS, React, Node.js, Pager 공부 정리**

노수일

**1. HTML5**

Hyper Text Markup Langaue

HTML Elements : <태그> </태그> 의 형태, 시작 태그와 닫는 태그 사이에 내용이 들어김.

Attribute(속성) : 모든 Elements는 attibute를 가질 수 있음. element의 추가적인 정보들을 제공.

ex) <img src =’abc/p[ng’ widht=’500’ />img 태그의 src 속성은 이미지 파일의 이름이 지정.

**HTML Tags**

<ㅗead>: HTML 문서의 메타 데이터를 가진다. 메타 데이터는 보여지지 않음.

<h1> ~ <h6> : 제목들은 크기별로 제일큰게 <h1> 제일 작은게 <h6>로 사용.

<hr> : Horizontal Rules : 주제별 구분, 내용 분리를 할때 사용.

<p>: 문단, 단락을 정의.

<br> : 줄을 바꿀 때 사용, 새로운 단락을 사용하지 않고 줄 바꿈.

<pre> : 서식이 미리 지정된 텍스트 정의, 너비, 글꼴, 공백, 줄 바꿈 모두 유지.

<b> : 굵은 글씨.

<strong> : 굵은 글씨, 중요한 텍스트.

<i> : 기울임 꼴 텍스트

<em> : 의미가 중요한 강조 텍스트.

<small> : 더 작은 텍스트.

<mark> : 마킹으로 강조되는 텍스트.

<del> : 삭제된 텍스트임을 표시하는 텍스트.

<sup> : 윗첨자를 표시하는 텍스트.

<q> : 짧은 인용을 정의, 따옴표 삽입.

<blockquote> : 들여쓰기.

<abbr> : 약어, 머리글자 정의, title 속성에 약어 정리.

<address> : 문서, 기사의 연락처 정보를 정의, 기울임꼴로 표시.

<bdo> : 현재 텍스트 방향을 재정의.

<!-- --> : 주석.

<style> : 내부 CSS 는 style 태그 내에 정의.

<link rel=”stylesheet” href=”\*.css”> : 외부 CSS 파일 적용, 하나의 파일로 여러 HTML 적용가능.

<a href=””> : 하이퍼링크, href 속성에 정의된 경로의 다른 문서로 이동하거나, 다른 태그의 id를 이용하여 책갈피처럼 문서 내에서 이동 가능, 외부 경로에는 URL, 절대 경로, 상대 경로가 있음.

<img /> : HTML에서 이미지 정의, 닫는 태그 없음, alt 속성은 이미가 없는 경우 대체 텍스트, width, height 속성으로 너비와 높이 조절, src 속성으로 외부 url 이미지, 이미지 경로 가능, gif로 움직이는 이미지 가능.

<table> : 테이블 정의, <tr>은 테이블에서 행을 나타내고, <td>는 테이블에서 데이터/셀로 정의, colspan 속성을 사용하여 둘 이상의 열이나 행을 합해서 사용. <caption> 태그를 이용해 표에 캡션을 추가할수 있음.

<ul> : 순서가 지정되지 않은 목록 style 속성에서 list-style-type을 통해 마커의 스타일을 정할 수 있음.

<ol> : 순서가 지정되어 있는 목록 type 속성에서 마커의 유형(숫자, 알파벳 등)을 정할 수 있음.

<li> : 각 목록 항목.

<dl> : 설명할 용어 목록을 정의.

<dt> : 설명할 용어의 이름을 정의

<dd> : 영어를 설명함.

<iframe> : 웹 페이지 내에 웹 페이지를 표시하는 데 사용.

<script> : 클라이언트 측 스크립트를 정의 하는데 사용.

<noscript> : 브라우저에서 사용할 스크립트를 하거나, 클라이언트 측 스크립트를 지원하지 않는 브라우저가 사용자를 위한 대체 콘텐츠를 제공

<meta> : 페이지 설명, 키워드, 제작자 및 다른 메나 데이터를 사용하는 문제 세트를 지정, 브라우저(내용 표시 방법), 검색 엔진 및 기타 웹 서비스에서 사용.

<code> :컴퓨터 코드를 나타냄.

<kbd> : 키보드 입력 또는 음성 명령과 같은 사용자 입력을 나타냄.

<samp> : 프로그램 또는 컴퓨터 시스템의 출력

<var> : 변수를 표시함. 수학식 표현식의 변수나 프로그래밍 변수.

<form> : 사용자의 입력을 수집하는데 사용되는 양식들.

- action : form에서 내용이 제출 될 때 수행할 작업을 정의함. <form action=’/action\_page.php’> 라고 되어 있으면 form 데이터는 ‘/action\_page.php’라는 서버의 페이지로 전송함.

- target : 제출된 결과가 새 브라우저 탭, 프레임에서 열거나 현재 창에서 할 경우의 속성들을 지정함. \_self : 기본값 현재창, \_blank : 새 브라우저 탭

- method : 속성은 HTTP의 GET, POST 메소드.

GET : 제출된 form의 데이터가 페이지 주소필드에 표시됨. ex)/page.phh?name=Roh

POST : form 데이터에 중요한 정보(개인 정보 등)가 포함되어있으면 사용, 제출된 form데이터를 주소필드에 표시하지 않음

- name : 제출되어야 할 요소들은 name속성이 있어야함, name속성이 없는 데이터는 전송되지 않음.

<input> : 텍스트 필드, 체크 박스, 라디오 버튼, submit 등과 같은 입력 요소. type 속성으로 입력 형태를 정할 수 있음.

- type 속성

text : 텍스트를 입력받는 필드

password : 비밀번호 입력 필드

submit : form 내부에서 데이터를 전송할 버튼 형태.

reset : form 내부의 입력된 값들을 reset하는 버튼 형태.

radio : 여러개의 선택지중 한가지만 선택할 수 있는 버튼.

checkbox : 0~n개의 선택을 체크해서 선택할 수 있는 버튼.

button : 버튼.

color : HTML5에서 새로 추가된 type, 색상을 선택할 수 있는 버튼

date : HTML5에서 새로 추가된 type, 날짜를 선택할 수 있는 버튼.

datetime-local : HTML5에서 새로 추가된 type, 날짜와 시간을 선택할 수 있는 버튼.

email : HTML5에서 새로 추가된 type, 이메일 주소를 포함해야 입력이 가능.

month : HTML5에서 새로 추가된 type, 사용자가 월 및 연도를 선택할 수 있음.

number : HTML5에서 새로 추가된 type, 숫자를 입력받음, 숫자 제한 가능.

- vaule : 입력 필드의 초기값을 지정.

- readonly : 읽기 전용, 변경될 수없는 것으로 지정.

- disabled : 입력 필드가 불가능함으로 지정, 사용, 클릭 불가능하고 값이 전송되지 않음.

- size : 입력 문자의 크기를 지정.

- maxlength : 입력 필드의 최대 허용 길이 지정.

<fieldset> : form 요소 안에 연관된 데이터끼리 그룹을 하는데 사용.

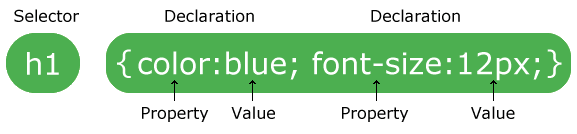
<legend> : fieldset 요소에 캡션(표제)을 정의.

**2. CSS3**

Cascading Style Sheets

화면, 페이지, 등에서 HTML 요소를 표시하는 방법을 설명.

**기본 문법 ↓**



CSS 선언은 항상 중괄호로 묶음.

각 선언에는 ;(콜론) 으로 구분 된 CSS 속성(Property) 과 값(Value)으로 표현.

Selector(선택자) : 스타일을 지정할 HTML 요소를 가리킴, 요소의 이름 (태그명), ID, 클래스, 특성 등에 따라서 찾거나 선택하는데 사용.

id Selector : HTML 요소의 id속성을 사용하는 특정 요소를 선택함. id는 페이지 내에서 고유해야 하므로 id Selector는 하나의 요소를 선택하는데에 사용. 만약 id = “kk” 라고 되어있으면 #(해시) 다음 요소의 id를 사용하므로 #kk { } 이런 방식으로 사용.

class Selector: 특정 클래스 속성이 있는 요소를 선택함. 마침표 기호 다음에 클래스 이름을 사용. 만약 class=”kkk”라고 되어있으면 .kkk { } 이런 방식으로 사용. 만약에 p.kkk { } 이런 방식으로 쓴다면 클래스 속성이 kkk인 p태그를 지정함. HTML 요소는 둘 이상의 클래스를 참조할 수 있기 때문에 <p class = “abc qwe”>content</p> 처럼 되어 있으면 abc, qwe의 스타일이 둘다 적용.

**CSS 적용 방식**

CSS를 적용하는 방식은 세가지가 있음.

1. 외부 스타일 시트 2. 내부 스타일 시트 3. 인라인 스타일

1. 외부 스타일 시트

외부 스타일 시트를 사용하면 하나의 파일로 전체 웹사이트의 모양을 변경할 수 있음. 각 페이지는 <link> 태그 안에 외부 스타일 시트 파일에 대한 참조를 포함해야 함. <link> 태그는 <head> 섹션에 있어야 함. ex) <link rel = ”stylesheet” href = ”abc.css”>

2. 내부 스타일 시트

한 페이지 내에 고유한 스타일이 있으면 내부 스타일 시트를 이용할 수 있음. 내부 스타일은 <head> 섹션에 <style> 요소 내에서 정의할 수 있음.

3. 인라인 스타일

단일 요소에 고유한 스타일을 적용하는데 사용함. 관련 요소에 스타일 속성을 추가해서 사용. ex) <h1 style = “color : red; margin : 10px”>

다중 스타일 시트 : 다른 스타일 시트에 동일한 선택자에 대해 같은 속성이 정의 된 경우 마지막으로 읽은 스타일 시트의 값을 사용.

ex) 내부 스타일 시트에 h1 { color:red; }, 외부 스타일 시트에 h1 { color:blue; }처럼 되어 있고, <head> <link rel = ”stylesheet” href = ”abc.css”> <style> h1 { color:red; } </style> </head> 와 같이 되어 있으면 내부 스타일 시트가 마지막으로 읽어지므로 h1의 글자색은 빨간색. <head> <style> h1 { color:red; } </style> <link rel = ”stylesheet” href = ”abc.css”> </head> 와 같이 되어있으면 외부 스타일 시트가 마지막으로 읽어지므로 h1의 글자색은 파란색.

우선순위

1. 인라인 스타일(HTML 요소 내부) 2. 외부,내부 스타일 시트(헤드 섹션) 3. 브라우저 기본값

**3. SASS**

[Syntactically Awesome Style Sheets](https://sass-lang.com/)

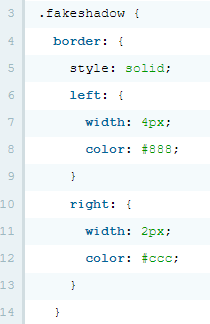
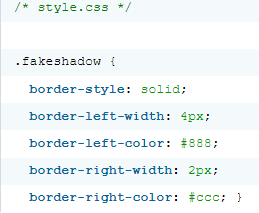
프로젝트의 규모가 커지면 CSS를 유지보수하는 것은 어렵거나 불가능한 일이 되어버림. 이러한 CSS의 단점을 보완하기 위한 기술, SASS를 직접 적용할수는 없고 SASS 문법에 맞게 파일을 만들면 컨버터를 이용해 CSS를 생성함.

**SASS 문법**

중첩(Nesting) : CSS의 특성으로 인해서 셀렉터를 중복해서 사용해야 하는 경우가 많음. SASS의 Nesting을 이용하면 코드의 양을 줄이고 연관된 코드끼리 그룹핑할 수 있음.

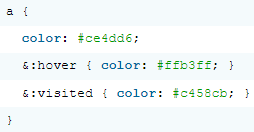
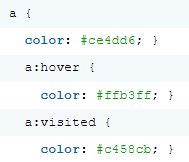
SASS CSS

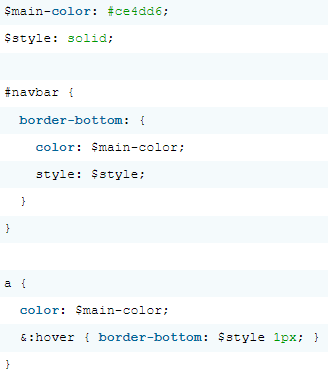
부모 엘리먼트를 참조 (Parnet References) : :hover와 같은 pseudoclasses의 경우는 특수기호 &를 이용해서 부모엘리먼트를 참조할 수 있음.

SASS CSS

변수(Variables) : CSS 내에서 변수를 사용할 수 있음. 변수 이름은 $로 시작해야 하고, 변수의 값으로 올 수 있는 것은 문자, 숫자(px, em포함), 칼라(#a0b0c0)가 있음. 변수를 이용하면 크기나 색상과 같은 값을 일괄적으로 변경 할 수 있음.

SASS CSS

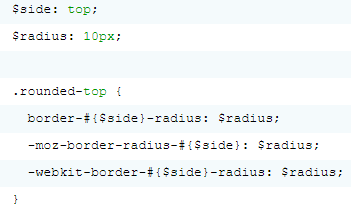
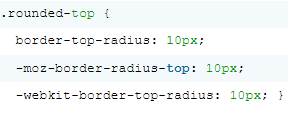
연산자와 함수(Operations and Functions) : 연산자와 함수를 이용해서 엘리먼트의 크기나 좌표 또는 색상을 동적으로 변경할 수 있음.

SASS CSS

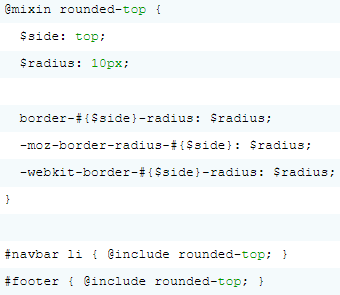
Interpolation : #{ } 를 사용해서 변수로 속성이나 선택자의 이름을 동적으로 치환할 수 있음.

SASS CSS

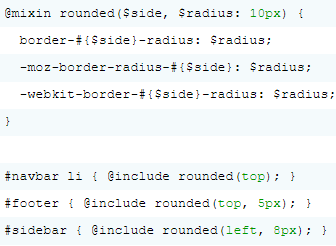
Mixins : 선택자와 속성을 재활용할 수 있도록 해주는 방법. 선언할 때는 ‘@mixin’으로 시작하고, 호출할 때는 ‘@include’를 사용함..

SASS CSS

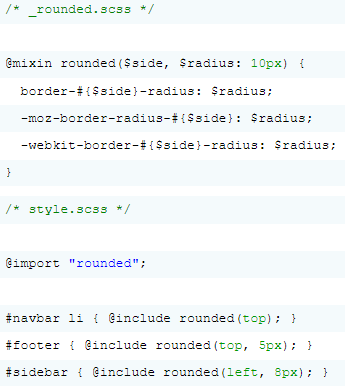
인자(Arguments) : 인자는 Mixin 안에서만 사용되는 지역변수를 의미. 인자는 기본값을 가질 수 있음.

SASS CSS

불러오기 (@import) : CSS는 @import 명령을 지원하는데, 이 명령은 다른 CSS를 불러오는 것임. 이 것은 별도의 파일을 네트워크를 통해서 로딩하는것인데, 느리고 비쌈. Sass에서 import는 그 파일의 내용을 실제로 가져와서 파일에 통합함. Sass에서 import를 위한 이름규칙이 있는데, 불러지는 파일은 partials라고 불리고 이 파일은 \_로 이름이 시작함. \_rounded.scss와 같이 붙음. 이파일을 불러올 때에는 @import’rounded’ 라고 사용함

SASS CSS

**4. React**

페이스북에서 개발한 UI 라이브러리, 웹 페이지 내부의 각 부분을 컴포턴트로 다룰 수 있게 해줌. 컴포넌트르르 HTML 태그로 작성할 수 있게 해주는 JSX를 사용하며 가상 DOM을 사용해 렌더링 성능을 극대화.

**JSX**

자바스크립트 확장 언어, 바벨을 사용하여 자바스크립트 문법의 코드로 변환하여 사용.

**Render( )**

리엑트에서 DOM을 출력할 때 ReactDOM.render( ) 메서드를 사용하여 ( ) 안에 HTML 처럼 사용 가능(JSX). DOM을 구축해서 화면에 화면에 출력하는 처리 : 렌더링

**리액트 / JSX**

- JSX 태그 내부에 변수를 넣는 방법 : <tag> { value } </tag>

- JSX 태그 속성 값에 변수를 넣는 방법 : <tag attr={ value }> </tag>

- JSX 주의 사항 :

1. 닫는 태그가 반드시 있어야 함.

2. JSX 범위를 제대로 인식하지 못 할 수 있으므로 소괄호로 범위를 명시적으로 지정해줘야 함.

3. 여러개의 태그를 연속적으로 사용할 수 없음.

div 태그로 감싸면 해결 가능.

잘못된 예 : <p> a </p> <p> b </p> 해결 방법 : <div> <p> a </p> <p> b </p> </div>

- JSX로 스타일 지정하기

style 속성을 문자열로 지정할 수 없음. 객체로 지정함

ex) const jsxstyle = { prop1:value1, prop2:value2, prop3:value3 }

const dom = <tag style = { jsxstyle } ></tag>

- 변수의 값은 자동으로 이스케이프 처리

HTML에서 특수한 의미로 사용되는 “<”, “>” 등과 같은 기호가 지동으로 “&alt;” 또는 “&gt;”로 변환됨. 특수 문자 변환을 쉽게 할 수 있고, 받은 값을 출력할 때 발생할 수 있는 보안문제도 방지 가능.

**가상 DOM**

DOM의 상태를 메모리에 저장하고, 변경 전과 변경 후의 상태를 비교한 뒤 최소한의 내용만 반영하는 기능, 리액트가 인기 있는 이유중 하나.

**컴포넌트**

특정 기능을 가진 범용적인 부품 : 버튼, 텍스트 박스 -> 하나하나 만들지 않고 재사용 가능하도록 컴포넌트로 만들어서 사용함. 기존의 <h1> 태그처럼 사용가능, 개발자가 원하는 형태로 UI 부품을 설계함. ES6에서 새로 나온 문법인 Class를 이용하여 좀 더 복잡한 컴포넌트도 구현 가능.

state 객체를 사용하여 check 박스와 같은 상태를 가지는 컴포넌트를 만들 수 있음. setState( ) 메소드는 state의 프로퍼티 값을 변경할 때 사용되며 이 때, 자동으로 render( ) 메서드가 함께 실행됨.

**이벤트**

HTML/JavaScript가 기본적으로 제공하는 이벤트를 기반으로 만든 독자적인 이벤트 사용.

ex) 일반적인 클릭 이벤트 지정 : <div onclick=’clickHandler(e)’>Click Me</div>

리엑트에서 클릭 이벤트 지정 : <div onClick=’clickHandler(e)’>Click Me</div>

- 리액트로 이벤트를 만드는 방법

1. render( ) 메서드 내부에 이벤트 핸들러 정의 - 가장 쉬운 방법

2. 클래스의 메서드로 정의하고, this 바인드 하기 - bind() 메서드 사용으로 복잡할 수도 있음

3. 클래스 메서드로 정의하고, 화살표 함수로 호출하기 - 익숙해지면 편리함.

**npm create-react-app**

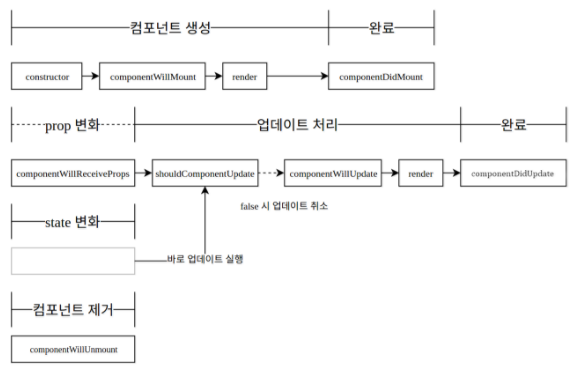
node.js 에서 간단하게 리액트를 개발할 수 있는 환경을 제공하는 애플리케이션.

**webpack**

자바스크립트 또는 CSS 등의 리소스 파일을 하나로 합쳐주거나, JSX처럼 특수한 기법으로 작성된 파일을 변환하는 도구.

**컴포넌트의 라이프 사이클**

리액트의 컴포넌트는 생성, 파괴, 상대변화 등의 시점에 자동으로 호출되는 메소드가 있음.



- 컴포넌트 생성과 DOM에 마운트

constructor(props) - 객체가 생성될 때

componentWillMount( ) - 컴포넌트가 DOM에 마운트되기 직전

render( ) - 컴포넌트가 렌더링 될 때

componentDidMount() - 컴포넌트가 DOM에 마운트 된 직후

- 컴포넌트 업데이트

componentWillReceiveProps(nextProps) - 컴포넌트의 프로퍼티가 변경될 때.

shouldComponentUpdate(nextProps, nextState) - 컴포넌트의 외관을 변경해도 좋을지 판단 할 때.

render( ) - 컴포넌트가 렌더링 될 때

componentDidUpdate( ) - 컴포넌트가 리 렌더링을 마친 후.

- DOM에서 언마운트(제거)

componentWillUnmount - 컴포넌트가 DOM에서 언마운트 될 때

**컴포넌트끼리 연동**

컴포넌트는 제품을 구성하는 부품이므로 컴포넌트 외부에 어떠한 영향을 주지 않으며, 자체적으로 모든 것이 완결 되는 것이 좋음. 리액트의 컴포넌트는 외부와의 창구를 만들고 그 창구를 통해서만 외부와 연동할 수 있음. 이때 외부와의 창구로 사용되는 것이 요소의 프로퍼티(태그의 속성). 외부에서 컴포넌트의 프로퍼티를 변경할 수 있지만, 컴포넌트 스스로는 프로퍼티의 값을 변경할 수 없음. onChange, onSubmit등의 이벤트를 통해 컴포넌트 값의 변화를 외부에 통지할 수 있음.

**컴포넌트의 3대 요소**

- 상태(state) : 컴포넌트의 상태를 나타내며, 변화할 수 있는 데이터의 집합. 상태가 변화하면 컴포넌트를 다시 렌더링 하게 됨. 상태로 사용해야 하는 것은 컴포넌트의 상태를 나타내는 값(리스트에서 선택한 값, 체크박스에서 값이 체크되었는지 여부, 텍스트박스의 텍스트 등), 외부에 공개하지 않고 컴포넌트가 스스로 관리해야 함. 상태를 변경할 때는 setState( )를 통해 값을 변경.

- 프로퍼티(props) : 외부에서 컴포넌트와 소통하는 창구 역할, 프로퍼티의 값은 부모요소에서 설정, 컴포넌트 내부에서 변경하지 않음, 초기값 설정(defaultProps), 유효성 검사(popsTypes) 등을 사용 가능, 프로퍼티는 읽기 전용이고, 프로퍼티 변경은 컴포넌트 상태 변경을 함께 발생시킴.

- 이벤트 : 리액트의 이벤트는 HTML,JavaScript의 이벤트와 다름, onXxx의 형태로 이벤트 이름을 사용, 프로퍼티를 통해 지정.

**5. Node.js**

구글 크롬의 자바스크립트 엔진(V8 Engine)에 기반해 만들어진 서버사이드 프랫폼

Asynchronous I/O (비동기식), 이벤트 위주, 서비스(서버) 분야 IoT(임베디드시스템)분야에서 활용

**전역 변수, 전역 함수, 전역 객체**

전역 변수 : 프로그램 전체에서 사용할 수 있는 변수, \_\_filename(경로명/파일명), \_\_dirname(경로명)

전역 함수 : 선언하지 않고 사용할 수 있는 함수, setTimeout(), setInterval(), clearInterval()...

전역 객체 : 프로그램 전체에서 사용할 수 있는 객체, console, export, process... 선언 없이 사용

console 객체 : 표준화면에 간단한 디버깅 기능을 제공하는 객체 log(), error(), time(), trace()...

process 객체 : 프로세스의 종료를 제어하거나, 실행환경 정보를 제공하는 객체 exit(), uptime(), argv, env, arch, version...

exports 객체(->require()) : 모듈을 자바스크립트 프로그램 안으로 불러와서 기능을 확장하는 객체 (2개 이상의 js파일), 프로그램을 개발할 때 모듈방식을 최대한 이용하여 재활용을 극대화 할 것.

**기본 모듈**

Node.js 프레임워크에서 기본적으로 제공하는 모듈들

os 모듈 : Node.js가 실행되는 서버 운영체제와 관련된 정보/기능을 제공하는 모듈

url 모듈 : URL 주소를 파싱하고 분석하는 기능을 제공하는 모듈

util 모듈 : 프로그래밍에서 필요한 유틸리티 기능을 제공하는 모듈

File System 모듈 : Directory/File I/O 기능을 제공하기 위한 모듈, 파일에서 읽기, 쓰기, 파일존재 유무, mode/소유자 변경 등 동기/비동기 메소드 제공, 비동기 메소드는 파라미터로 callback 함수가 반드시 포함됨. 기본적으로는 비동기식 I/O이며, 동기식 I/O기능도 제공하고 있음.

crypto 모듈 : 데이터 암호화 기능을 제공하는 모듈

**이벤트**

이벤트 기반의 비동기 프로그래밍

process.on() : 이벤트 등록

process.removeListener : 등록한 이벤트 핸들러를 제거

process.emit(’’) : 이벤트 강제 발생

**HTTP 모듈**

HTTP 모듈의 주요 객체 : Server 객체, Response 객체, Request 객체

- Server 객체 : 원하는 서비스 처리를 할 수 있도록, 서버 생성/처리 기능을 제공하는 객체

const server = http.createServer( );

- Response 객체 : http 서버가 Client에게 응답할때 사용하는 객체 (http response 메시지)

const server = http.createServer( (request, response)=>{ response.end(); } );

response.writeHead( 200, { ’Content-Type’ : ’text/html’ } )

숫자 의미 : Status codes for response 1XX : 조건부 응답, 2XX : 성공, 3XX: 리디렉션 완료, 4XX : 요청 오류, 5XX : 서버 오류

MIME types : text, image, audio, video, applicatioin

Cookie : ‘Set-Cookie’:[ ‘abc = kkk, Expries = ‘ + date.toUTCString() ] Expires : 쿠키 유효기간 설정

- Request 객체 : request.method : 요청방식 반환 GET, POST 등, request.url : url 문자열 반환, request.headers.cookie : 쿠키 추출

**외부 모듈 (확장 모듈)**

3rd party vendor, 개인이 개발하여 제공하는 모듈, 다수의 프레임워크 유틸리티, 함수기능을 제공

npm을 이용한 설치

템플릿 모듈 : node.js 프레임 워크에서는 \*.html 파일만 읽어서 제공 (정적인 페이지) jade(=pug), ejs 모듈은 HTML 문서에 JavaScript 코드를 넣을 수 있도록 해서, 동적인 웹 페이지를 생성하는 모듈

supervisor 모듈 : 파일의 내용이 바뀌거나 서버가 비 정상적으로 종료되면 restart하는 기능을 제공하는 모듈

**Express 모듈**

웹 애플리케이션을 http 모듈보다 더 간단하고, 더 빠르게 개발하게 해주는 외부 모듈,

express 모듈은 다양한 미들웨어들을 포함, 활용하여 쉽게 개발할 수 있음.

- use, router, morgan, static, cookie parser, body parser, express-session 등

const express = require( ’express’ ) //모듈 추출

const app = express() //웹 서버 생성

app.use(...) //웹 서버 사용, 처리

app.listen( port, function( ) ) //port 번호로 웹 서버 실행 localhost:port

- router 미들웨어 : 페이지 라우팅을 쉽게 구현해주는 모듈 -> 큰 규모의 프로그램을 모듈화

- Cookie parser 미들웨어 : request 쿠키를 추출하는 미들웨어.

- Body parser 미들웨어 : POST request 데이터를 추출하는 미들웨어, request 객체에 body 속성이 부여되어, http 메시지 body 부분의 내용을 가져올 수 있음.

- express-session 미들웨어 : 웹 애플리케이션에서 필요한 세션 정보를 서버에 저장하고 관리하는 미들웨어, 쿠키정보가 사용자 PC에 저장되므로, 유출 가능성이 높아 보안이 문제되어 세션이 이를 보완, 일반적으로 쿠키와 같이 사용됨.

app.use( session({ secret : ’secret key’, resave: false, saveUninitialized: true }));

sercet : 세션 id를 암호화 할때 사용하는 문자열

resave : 접속할때마다 세션 id를 발급할지 말지를 결정함. false = 하지말라, true = 해라

saveUninitialized : 처음 생성할때 초기화 하지 않음. (true) (세션 id를 사용 전까진 발급하지 말 것)

**Express Generator**

애플리케이션의 골격(skeleton)을 신속하게 작성하도록 도와줌.

package.json 파일에 필요한 모듈 작성.

MVC 패턴에서 Controller 부분은 bin/www 파일에서 찾을 수 있음, View 부분은 views 디렉토리에서 찾을 수 있음, route 디렉토리로 router 미들웨어 적용.

**Socket.io**

Node.js 기반의 소켓 통신

웹 소켓(Web socket) 방식 - 실시간 양방향 전송

- public 통신방식 : 자신을 포함한 broadcast 전송

- boradcast 통신방식 : 자신을 제외한 broadcast 전송

- private 통신방식 : 특정 클라이언트에 직접 전송

**6. Phaser**

Photon Storm에서 만든 HTML5 게임을 만들기 위한 2D 게임 프레임 워크. Canvas와 WebGL 렌더러를 내부적으로 사용.

**Phaser 2.6.2 API**

**Via** : 클래스가 이 항목이 있는 경우 신속하게 로컬참조를 통해 접근 가능, ex) this.camera , game.camera 게임이 전역적으로 정의 된 경우 카메라를 제어할 수 있다.

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| Game | game | 부팅, 서브 시스템 생성, 로직 실행 및 루프 렌더링을 관찰. |
| World | world | 모든 게임 객체가 있는 게임 월드. |
| Camera | camera | 게임 월드에서의 카메라 뷰 |
| Stage | stage | 루트 디스플레이 객체 |

**Game States**

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| StateManager | state | 게임 상태를 생성, 관리 및 교환. |
| State |  |  |

**Loader**

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| Cache | cache | 캐시는 로드된 모든 에셋이 저장되고 검색되는 곳 |
| Loader | load | 모든 외부 에셋 타입(이미지, 오디오, json, xml, txt)를 로드하고 이를 캐시에 추가 |
| LoaderParser |  | 복잡한 에셋 유형의 구문 분석을 처리하기 위해 Loader에서 사용하는 정적 클래스 |

**Game Scaling**

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| ScaleManager | scale | 여러 기기에서 게임의 크기 조정 및 관리 |
| FlexGrid | scale.grid | 반응형 레이아웃 그리드 |
| FlexLayer |  | 응답성 있는 그리드 레이어 |

**Signals**

|  |  |
| --- | --- |
| Class | Description |
| Signal | 신호는 Phasers 내부 이벤트 시스템 |
| SignalBinding | Signal에 바인딩 된 콜벡을 관리 |

**Plugins**

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| PluginManager | plugins | 플러그인을 설치, 업데이트 및 파괴 |
| Plugin |  | 확장할 수 있는 기본 Plugin 객체 |

**Game Objects**

|  |  |  |
| --- | --- | --- |
| Class | Via | Description |
| GameObjectFactory | add | Phaser Game Objects를 생성하고 Game World에 추가 할 수 있는 도우미 클래스 |
| GameObjectCreator | make | Phaser Game Object를 생성하고 반환할 수 있는 도우미 클래스 |
| Group |  | Group에는 여러 게임 객체가 포함될 수 있으며 검색, 정렬, 호출, 업데이트 및 필터링 기능이 있음. |
| InputHandler | object.input | Game Object가 입력이 가능한 경우 이 클래스는 클릭 및 드래그를 포함하여 모든 입력관련 이벤트 제어 |
| Events | object.events | 모든 게임 객체 레벨 이벤트 |
| Create | create | 동적 스프라이트 및 텍스처 생성 |

**Display**

|  |  |
| --- | --- |
| Class | Description |
| Sprite | 텍스처, 애니메이션, 입력 이벤트 및 물리학을 실행 할 수 있는 게임 객체 |
| Image | 질감과 입력이 있지만 물리적 또는 애니메이션 처리기가 없는 경량의 게임 객체 |
| TitleSprite | 반복 텍스처가 스크롤 되고 스케일 될 수 있는 게임 객체 |
| Button | 도우미 메서드 및 이벤트가 있는 이미지 게임객체로 UI 버튼으로 바꿈 |
| SpriteBatch | 커스텀 Sprite 배치. 동일한 텍스처를 공유하는 여러 Sprite를 훨씬 빠르게 렌더링 가능. |
| Rope | strip-based 텍스처를 가진 결합된 게임 객체 |