**[Node 기초2]**

**■ npm의 버전 업데이트 하기**

npm install –g npm

**■ ES6**

var(함수 스코프)은 const와 let(블록 스코프)으로 대체

const와 let의 차이

const는 한 번 대입하면 다른 값을 대입할 수 없다. let은 가능.

기본적으로 변수 선언 시에는 const를 사용하고, 다른 값을 대입해야 하는 상황이 생겼을 때는 let을 사용

const 안의 배열의 값은 바꿀 수 있다

* 탬플릿 문자열
* var num1 = 1;
* var num2 =2;
* var result = 3;
* var string1 = num1 + '더하기' + num2 + '는 \'' + result + '\'';
* console.log(string1);
* // VM271:1 1더하기2는 '3'
* const num3 = 1;
* const num4 = 2;
* const result2 = 3;
* const string2 = `${num3} 더하기 ${num4}는 '${result2}'`; // 백틱은 IE와 사파리에선 지원 안함
* console.log(string2);
* // VM435:1 1 더하기 2는 '3'
* 객체 리터럴
* var sayNode = function(){
* console.log('Node');
* }
* var es = 'ES';
* var oldObject = {
* sayJS : function(){
* console.log('JS');
* },
* sayNode : sayNode,
* };
* oldObject[es+6] = 'Fantastic';
* oldObject.sayNode(); //Node
* oldObject.sayJS(); // JS
* console.log(oldObject.ES6); //Fantastic
* const newObject = {
* sayJS(){
* console.log('JS');
* },
* sayNode,
* [es + 6] : 'Fantastic',
* };
* newObject.sayNode(); //Node
* newObject.sayJS(); //JS
* console.log(newObject.ES6); //Fantastic
* /\*
* {name:name, age:age} // ES5
* {name, age} // ES6
* \*/
* 화살표 함수
* function add1(x,y){
* return x+y;
* }
* const add2 = (x, y) => {
* return x+y;
* }
* const add3 = (x,y) => x+y;
* const add4 = (x,y) => (x+y);
* function not1(x){
* return x;
* }
* const not2 = x => !x;
* // function 선언 대신 => 기호 함수 선언
* var relationship1 = {
* name : 'zero',
* friends : ['nero','hero','xero'],
* **logFriends : function(){**
* **var that = this;**
* **this.friends.forEach(function(friends){**
* **console.log(that.name, friends);**
* **});**
* },
* };
* relationship1.logFriends();
* const relationship2 = {
* name : 'zero',
* friends : ['nero','hero','xero'],
* **logFriends(){**
* **this.friends.forEach(friends => {**
* **console.log(this.name, friends);**
* **});**
* **},**
* };
* relationship2.logFriends();
* 비구조화 할당
* var candyMachine = {
* status : {
* name : 'node',
* count : 5,
* },
* getCandy : function(){
* this.status.count--;
* return this.status.count;
* }
* };
* var getCandy = candyMachine.getCandy;
* var count = candyMachine.status.count;
* console.log(count);
* const candyMachine = {
* status : {
* name : 'node',
* count : 5,
* },
* getCandy(){
* this.status.count--;
* return this.status.count;
* }
* };
* const {getCandy, status:{count}} = candyMachine;
* console.log(count);
* // 배열 비 구조화
* var array = ['node.js', {}, 10, true];
* var node = array[0];
* var obj = array[1];
* var bool = array[array.length-1];
* const array = ['node.js', {}, 10, true];
* const [node, obj, , bool] = array;
* 프로미스
* const condition = true;
* const promise = new Promise((resolve, reject)=>{
* if(condition){
* resolve('성공');
* }else{
* reject('실패');
* }
* });
* promise
* .then((message)=>{
* console.log(message);
* })
* .catch((error) => {
* console.error(error);
* });
* promise
* .then((message)=>{
* return new Promise((resolve, reject) => {
* resolve(message);
* });
* })
* .then((message2) =>{
* console.log(message2);
* return new Promise((resolve, reject) => {
* resolve(message2);
* });
* })
* .then((message3) => {
* console.log(message3);
* })
* .catch((error) => {
* console.error(error);
* });
* function findAndSaveUser(Users){
* Users.findOne({},(err,user)=>{ // 첫 번째 콜백
* if(err){
* return console.error(err);
* }
* user.name='zero';
* user.save((err)=>{ // 두 번째 콜백
* if(err){
* return console.error(err);
* }
* Users.findOne({gender:'m'},(err,user)=>{ // 세 번 째 콜백
* // 생략
* })
* })
* })
* }
* function findAndSaveUser(Users){
* Users.findOne({})
* .then((user) => {
* user.name = 'zero';
* return user.save();
* })
* .then((user)=>{
* return Users.findOne({gender:'m'});
* })
* .then((user)=>{
* // 생략
* })
* .catch((error) => {
* console.error(error);
* });
* }
* const promise1 = Promise('성공1');
* const promise2 = Promise('성공2');
* Promise.all([promise1, promise2])
* .then((result) => {
* console.log(result);
* })
* .catch((error) => {
* console.error(error);
* })
* async/await
* function findAndSaveUser(Users){
* Users.findOne({})
* .then((user) => {
* user.name= 'zero';
* return user.save();
* })
* .then((user)=>{
* return Users.findOne({gender:'m'});
* })
* .then((user)=>{
* // 생략
* })
* .catch(err => {
* console.error(err);
* });
* }
* async function findAndSaveUser(Users){
* let user = await Users.findOne({});
* user.name = 'zero';
* user = await user.save();
* user = await Users.findOne({gender:'m'});
* }
* async function findAndSaveUser(Users){
* try{
* let user = await Users.findOne({});
* user.name = 'zero';
* user = await user.save();
* user = await Users.findOne({gender : 'm'});
* }catch(error){
* console.error(error);
* }
* }
* const findAndSaveUser = async (Users) => {
* try{
* let user = await Users.findOne({});
* user.name = 'zero';
* user = await user.save();
* user = await Users.findOne({gender:'m'});
* }catch(error){
* console.log(error);
* }
* }
* const promise1 = Promise.resolve('성공1');
* const promise2 = Promise.resolve('성공2');
* (async()=>{
* for await (promise of [promise1, promise2]){ // Promise.all 대신에 for await 문을 사용해서 프로미스를 반복하는 모습
* console.log(promise);
* }
* })();
* ajax
* <script type="text/javascript">
* var xhr = new XMLHttpRequest();
* xhr.onreadystatechange = function(){ // 요청에 대한 콜백
* if(xhr.readyState == xhr.DONE){ // 요청이 완료되면
* if(xhr.status == 200 || xhr.status == 201){ // 응답코드가 200이나 201이면
* console.log(xhr.responseText); //서버에서 보내주는 값
* }else{
* console.error(xhr.responseText);
* }
* }
* };
* xhr.open('GET','https://www.zerocho.com/api/get'); // 메서드와 주소 설정
* xhr.send(); // 요청 전송
* /\* onreadystatechange 대신에 onload와 onerror로 성공과 실패를 구분해도 됩 \*/
* var xhr = new XMLHttpRequest();
* xhr.onload = function(){
* if(xhr.status == 200 || xhr.status == 201){
* console.log(xhr.responseText);
* }
* };
* xhr.onerror = function(){
* console.error(xhr.responseText);
* };
* xhr.open('GET','https://www.zerocho.com/api/get'); // 메서드와 주소 설정
* xhr.send(); // 요청 전송
* /\* 서버로 데이터를 같이 보내는 POST 요청의 경우, JOSN 테이터를 보냄 \*/
* var xhr = new XMLHttpRequest();
* var data = {
* name : 'zerocho',
* birth : 1994,
* };
* xhr.onreadystatechange = function(){
* if(xhr.readyState == 200 || xhr.readyState == 201){
* console.log(xhr.responseText);
* }else{
* console.error(xhr.responseText);
* }
* };
* xhr.open('POST','https://www.zerocho.com/api/post/json'); // 메서드와 주소 설정
* xhr.setRequestHeader('Content-Type','application/json'); // 콘텐츠 타입을 json으로
* xhr.send(JSON.stringify(data)); // 데이터를 동봉해 전송
* </script>
* FormData
* <script>
* var formData = new FormData();
* formData.append('name','zerocho');
* formData.append('item','orange');
* formData.append('item','melon');
* formData.has('item'); // true
* formData.has('money'); // false
* formData.get('item'); // orange
* formData.getAll('item'); // ['orange','melon']
* formData.append('test',['hi','zero']);
* formData.get('test'); // hi, zero
* formData.delete('test');
* formData.get('test'); // null
* formData.set('item','apple');
* formData.getAll('item'); // [apple]
* </script>
* var xhr = new XMLHttpRequest();
* var formData = new FormData();
* formData.append('name','zerocho');
* formData.append('birth',1994);
* xhr.onreadystatechange = function(){
* if(xhr.readyState == xhr.DONE){
* if(xhr.status == 200 || xhr.status == 201){
* console.log(xhr.responseText);
* }else{
* console.log(xhr.responseText);
* }
* }
* };
* xhr.open('POST','https://www.zerocho.com/api/post/formdata'); // 메서드와 주소 설정
* xhr.send(formData); // 데이터를 동봉해 전송
* encodeURLComponent, decodeURLComponent
* <script>
* var xhr = new XMLHttpRequest();
* xhr.onreadystatechange = function(){
* if(xhr.readyState == xhr.DONE){
* if(xhr.status == 200 || xhr.status == 201){
* console.log(xhr.responseText);
* }else{
* console.error(xhr.responseText);
* }
* }
* };
* xhr.open('GET','https://www.zerocho.com/api/search/'+encodeURIComponent('노드'));
* xhr.send();
* decodeURIComponent('%EB%85%B8%EB%93%9C') // 노드
* </script>
* dataAtrribute, dataset

<ul>

<li data-id="1" data-user-job="programmer">Zero</li>

<li data-id="2" data-user-job="designer">Nero</li>

<li data-id="3" data-user-job="programmer">Hero</li>

<li data-id="4" data-user-job="ceo">Kero</li>

</ul>

<script>

console.log(document.querySelector('li').dataset); // {id : '1', userJob : 'programmer'}

</script>