일반 함수와 arrow function의 차이점

1. 일반 함수는 함수를 선언할 때 this에 바인딩할 객체가 정적으로 결정되는 것이 아니고, 함수를 호출할 때의 함수 호출 방법에 따라 this에 바인딩할 객체가 동적으로 결정되는 반면, arrow function은 함수를 선언할 때 this에 바인딩할 객체가 정적으로 결정된다. Arrow function의 this는 항상 상위 scope의 this를 가리키며, call, apply, bind 메소드를 사용하여 this를 변경할 수 없다.
2. 일반 함수는 생성자 함수로 사용할 수 있지만, arrow function은 생성자 함수로 사용할 수 없고 new로 호출하면 TypeError가 발생한다. Arrow function은 prototype 프로퍼티를 가지고 있지 않기 때문이다.
3. 일반 함수에서는 함수가 실행될 때 암묵적으로 arguments 변수가 전달되어 사용할 수 있지만, arrow function에서는 변수가 전달되지 않는다.

예시 코드

|  |
| --- |
| function func() {  this.name = "a";  return {  name: "b",  speak: function () {  console.log(this.name);  },  };  }  function arrowFunc() {  this.name = "a";  return {  name: "b",  speak: () => {  console.log(this.name);  },  };  }  const func1 = new func();  func1.speak(); // b  const func2 = new arrowFunc();  func2.speak(); // a |
| 일반 함수에서 this는 함수가 호출될 때 결정되고, 해당 함수를 호출하는 객체를 가리킨다. 하지만 arrow function의 this는 함수가 선언될 때 결정되고, arrow function을 선언한 컨텍스트에서 this를 상속받는다. 따라서 arrow function의 this는 함수가 선언된 시점에서의 this를 그대로 가리킨다.  예시에서 func 함수는 일반 함수이고, this.name은 함수를 호출하는 객체이다. func1.speak()를 호출할 때 this는 func1 객체이고, func1 객체의 name은 “b”이므로 b가 출력된다.  arrowFunc 함수는 arrow function이고, this는 함수를 선언한 컨텍스트에서 상속받으므로 arrowFunc가 선언된 컨텍스트의 name 프로퍼티인 “a”를 가리키게 된다. 따라서 “a”가 출력된다. |