바닐라 자바스크립트

chapter05

자바스크립트 고급 문법

Contents

01

part

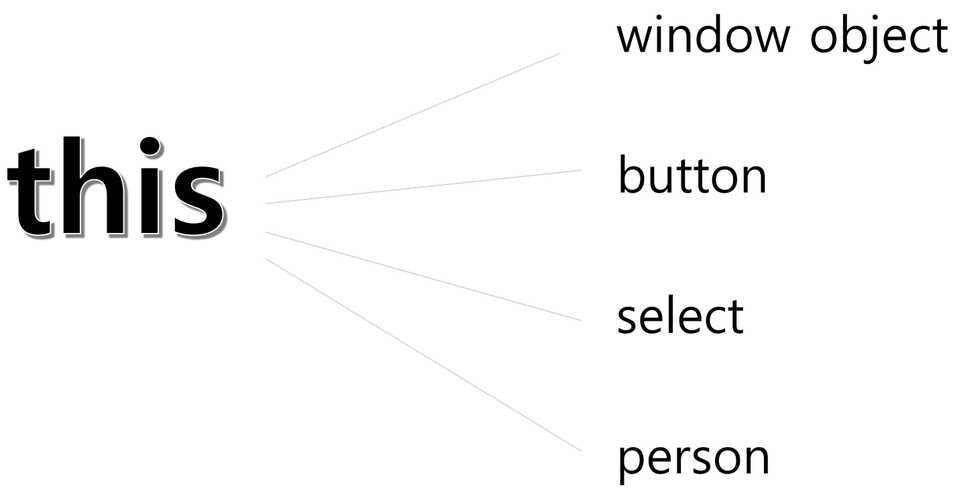
this 키워드

this 소개

this의 개념과 특징

this 키워드는 함수에서 자기 자신(객체)을 가리키는 참조변수이다.

this의 값은 함수를 호출한 방법에 따라 달라진다.



this 키워드

this에 바인딩된 객체들

전역 컨텍스트에서 this

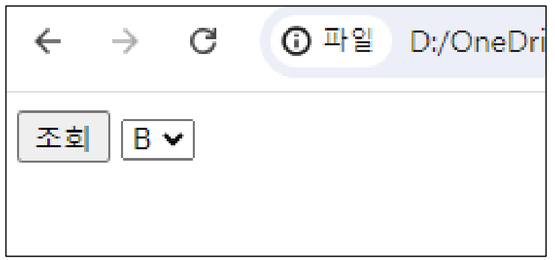
전역 컨텍스트, 함수 밖에서 this는 window 객체를 가리킨다.

```
<script>

  console.log(this); //Window
```

이벤트 핸들러에서 this

DOM 요소에 이벤트가 발생하면, this는 DOM 요소 자체를 가리킨다.

```
<body>
  <button type="button" onclick="doSearch(this);">조회</button>

function doSearch(obj) {
  console.log(obj); //Button
}
```



this 키워드

this에 바인딩된 객체들

Object객체에서 this

객체의 메소드를 호출하면, this는 메소드가 속한 객체를 가리킨다.

```
let person = {
  firstName: "John",
  lastName: "Doe",
  getFullName: function () {
    return this.firstName + " " + this.lastName; //person 객체
  },
};
```

02
part

스코프

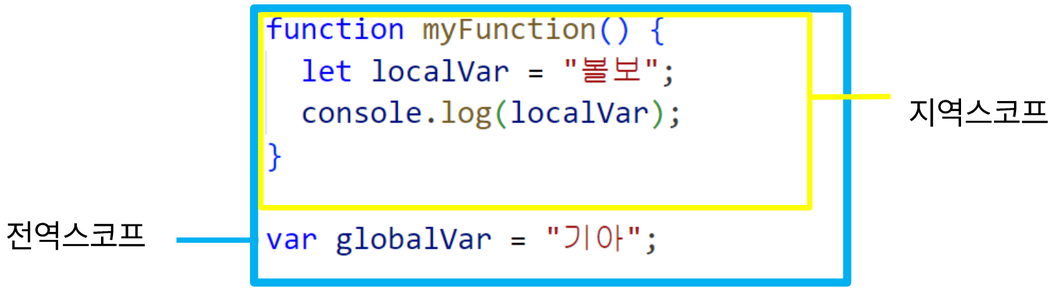
Scope

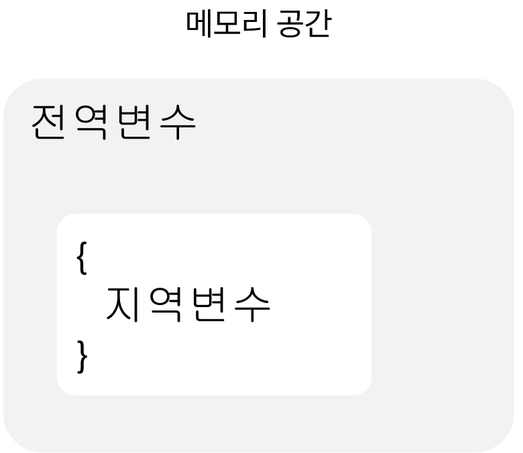
Scope는 변수에 대한 접근 범위를 의미한다.

변수는 선언된 위치에 따라 지역스코프와 전역스코프로 구분한다.

지역변수: 지역변수는 함수 안에서 선언되며, 해당 함수 안에서만 접근할 수 있다.
함수 실행이 끝나면 변수는 메모리에서 소멸되어, 다른 함수나 전역 코드에서 접근할 수 없다.

전역변수: 전역변수는 함수나 블록 밖, 전역 컨텍스트에서 선언된 변수이다.
스크립트 내 어디서나(모든 함수와 블록) 사용할 수 있다.





03

part

디폴트 파라미터 Default Function Parameter

함수의 매개변수에 값이 전달되지 않은 경우, 매개변수를 기본값을 설정할 수 있다.
기본값 매개변수를 설정하는 이유는, 사용자의 실수로 함수에 파라미터가 전달되지 않을 수 도 있기 때문이다.
파라미터가 전달되지 않으면 해당 매개변수를 undefined로 처리되어 잘못된 결과가 나올 수 있다.

예를 들어 곱셈 함수에서 b 값이 전달되지 않으면, "a * b"의 대한 결과가 "NaN"가 될 것이다.
예전에는 이런 오류를 막기 위해, 함수 안에서 값을 확인하고 기본값으로 재설정하는 방식을 사용했다.
하지만 이제 디폴트 파라미터를 사용하면, 이런 검사를 할 필요없이 미리 지정한 기본값이 자동으로 적용된다.

```
function multiply(a, b = 1) {
  console.log(a * b);
}
multiply(5, 2); // 10
multiply(5); // 5
```

디폴트 파라미터

04

part

Rest 파라미터 Rest Parameter

rest 파라미터는 함수의 매개변수로 전달된 나머지 데이터를 배열로 저장하는 방식이다.
함수의 마지막 매개변수 앞에 "..."(점 세 개)를 붙이면 rest 파라미터로 설정할 수 있다.
rest 파라미터는 항상 매개변수의 맨 마지막에 위치해야 한다.

예시를 보면, 전달된 인자 중 첫번째와 두번째는 각각 one과 two에 할당이 되고, 나머지 인자들은 rest 배열에 저장되었다.

```
function func(one, two, ...rest) {
  console.log(rest); //[3, 4, 5]
}

func(1, 2, 3, 4, 5);
```

Rest 파라미터 연습문제

Q1. 사용자의 이름과 인삿말을 입력받아 인사를 만드는 함수를 정의하세요.
함수를 호출했을 때 다음과 같은 결과를 출력하세요.

```
console.log(greet("철수", "반갑습니다")); //"반갑습니다, 철수님!" 출력
console.log(greet("훈이")); //"안녕하세요, 훈이님!" 출력
```

Q2. 학생의 영어, 수학, 국어 점수를 입력받아 평균점수를 출력하는 함수를 정의하세요.

```
calculateGrade("철수", 85, 90, 78); //"철수님의 평균 성적은 84.33점입니다." 출력
calculateGrade("훈이", 80); //"훈이님의 평균 성적은 26.67점입니다." 출력
```

Q3. 단어 여러개를 입력받아 하나의 문장으로 만드는 함수를 정의하세요.
단, 첫 번째 단어는 무시하고 나머지 단어들로만 문장을 만드세요.

```
console.log(createSentence("Hello", "this", "is", "JavaScript"));
// "this is JavaScript" 출력
```

Q4. 사람의 정보를 입력받아 출력하는 함수를 정의하세요.

```
printPerson("맹구", 5, "축구", "독서", "영화 감상");
//"맹구의 나이는 5살이고, 취미는 축구, 독서, 영화 감상입니다." 출력

printPerson("유리", 5, "소꿉놀이");
//"유리의 나이는 5살이고, 취미는 소꿉놀이입니다." 출력
```

Rest 파라미터 연습문제

Q5. 학생들의 성적을 계산하는 프로그램을 만들어주세요.

1) 여러 학생을 인자로 입력받아, 특정과목의 평균점수를 반환하는 함수를 정의하세요.

2) 만약 학생이 해당 과목의 점수를 가지고 있지 않으면 계산하지 않고, 해당 학생은 건너뜁니다.

예: 맹구와 유리의 수학과목의 평균점수: (80)/1 = 80점

```
학생 목록:
{ name: '짱구', eng: 80, math: 50, kor: 70 }
{ name: '철수', eng: 90, math: 85, kor: 88 }
{ name: '훈이', eng: 75, math: 95, kor: 92 }
{ name: '맹구', eng: 65 }
{ name: '유리', math: 80, kor: 90 }

짱구, 철수, 훈이, 맹구의 영어과목의 평균점수는 77.5점 입니다
맹구와 유리의 수학과목의 평균점수는 80.0점 입니다
국어과목의 전체 평균점수는 85.0점 입니다
```

Rest 파라미터 연습문제

Q6. 상품들의 가격을 계산하는 프로그램을 만들어주세요.

여러 상품을 인자로 입력받아, 평균 가격을 반환하는 함수를 정의하세요.

각 상품의 할인율을 적용하여 최종가격을 계산해야 합니다.

예: 새우깡의 최종가격: 1500 * 0.9 = 1350원

```
상품 목록:
{ name: '새우깡', price: 1500, discount: 10 }
{ name: '홈런볼', price: 2000, discount: 5 }
{ name: '포카칩', price: 2500 }
{ name: '칙촉', price: 3000, discount: 15 }
{ name: '오잉', price: 1800 }

새우깡, 홈런볼, 포카칩의 평균 가격은 1916.67원 입니다.
전체 평균 가격은 2020.00원 입니다.
```

05

part

화살표 함수 Arrow Function

화살표 함수 만드는 방법

익명함수에서 화살표함수로 변경할 수 있다.

화살표 함수는 기존 함수 표현식에서 “function” 키워드 대신 화살표 기호 “=>” 를 사용한다.

바디가 한 줄인 경우, 중괄호{} 와 return 키워드를 생략할 수 있다.

화살표 함수를 사용하는 이유

화살표 함수는 코드가 간단해서 짧은 함수에서 유용하다.

특히 매개변수로 함수가 전달될 때, 인자를 간단하게 작성할 수 있다.

```
const hello = function (name) {
  return "Hello" + name;
};
```

```
const hello = (name) => "Hello " + name;
```

Q7. 다음의 익명 함수를 화살표 함수로 변환하세요.

```
const add = function(a, b) {
  return a + b;
};
```

Q8. 다음의 익명 함수를 화살표 함수로 변환하세요.

```
const greet = function() {
  return "Hello, World!";
};
```

Q9. 다음의 익명 함수를 화살표 함수로 변환하세요.

```
const logSum = function(a, b) {
  console.log(a + b);
};
```

연습문제

Q10. 다음의 익명 함수를 화살표 함수로 변환하세요.

```
const multiplyAndAdd = function(a, b, c) {
  const temp = a * b;
  const result = temp + c;
  return result;
};
```

Q11. map에서 사용한 익명 함수를 화살표 함수로 변환하세요.

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(function(number) {
    return number * 2;
});
```

Q12. filter에서 사용한 익명 함수를 화살표 함수로 변환하세요.

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const evenNumbers = arr.filter(function(number) {
    return number % 2 == 0;
});
```

06

part

템플릿 리터럴 Template Literals

템플릿 리터럴 사용 방법

템플릿 리터럴은 문자열 안에 변수를 삽입하는 기능이다.

템플릿 리터럴은 문자열을 백틱(``)으로 감싸고 ${ }구문을 사용하여 변수나 표현식을 삽입할 수 있다.

템플릿 리터럴을 사용하는 이유

템플릿 리터럴을 사용하면 복잡한 문자열도 쉽게 작성할 수 있다.

```
var nm = "John";
console.log(`Hello ${nm}`); // "Hello John" 출력

var firstName = "Jane";
var lastName = "Doe";
console.log(`My name is ${firstName} ${lastName}`); // "My name is Jane Doe" 출력
```

객체 리터럴

object Literal Syntax Extension

객체 리터럴이란?

객체 리터럴은 객체를 생성할 때 사용하는 문법으로, 키와 값을 쌍으로 묶어 객체를 생성한다.

객체에 키 동적으로 추가하기

객체의 키를 고정된 문자열이 아닌 변수로 정할 수 있다.

예제에서 type 변수의 값을 객체의 키로 사용하고 있다.

키를 동적으로 추가하는 이유

객체의 속성을 미리 알 수 없거나 동적으로 설정할 때 사용한다.

```
var type = "name";
var score = {
  [type]: "John", // name: "John"
  score: 95,
};
```

연습문제

Q13. 이름과 나이를 변수에 저장하고, 변수를 사용하여 다음과 같이 출력하세요.

예: 이름=아리, 나이=17 로 설정한 후, "아리의 나이는 17 입니다.”를 출력

Q14. 숫자 두개를 변수에 저장하고, 두 숫자를 곱한 결과를 포함하여 다음과 같이 출력하세요.

예: a=10, b=20으로 설정한 후, "10과 20을 곱한 결과는 200 입니다”를 출력

Q15. 숫자 두개를 변수에 저장하고, 두 수의 평균을 포함하여 다음과 같이 출력하세요.

예: a=10, b=20으로 설정한 후, "10과 20의 평균은 15 입니다"를 출력

Q16. 도시이름과 인구수를 변수에 저장하고, 변수를 사용하여 다음과 같이 출력하세요.

예: 도시이름=서울, 인구수=9736000로 설정한 후, "서울의 인구는 9736000 명 입니다"를 출력

Q17. 학생의 이름과 점수를 변수에 저장하고, 변수를 사용하여 다음과 같이 출력하세요.

점수가 60점 이상이면 합격이고, 그렇지 않으면 불합격 입니다

예: 이름="소영", 점수=75으로 설정한 후, "소영의 점수는 75점으로 합격입니다"

연습문제

Q18. 아래와 같이 변수 세 개를 선언하세요.

key1 = "make"; key2 = "model"; key3 = "year";

그리고 이 세 개의 변수를 키로 사용하여 자동차 객체 car를 생성하세요.

객체: { make: '현대', model: '쏘나타', year: 2021 }

Q19. 아래와 같이 사람 객체 person을 생성하고, 변수 property를 선언하세요.

그리고 person 객체에 property 변수를 사용하여 동적으로 속성을 추가하세요.

property = "address";

초기 객체: { name: '철수', age: 21 }

업데이트된 객체: { name: '철수', age: 21, address: '인천 남동구 만수동' }

Q20. 아래와 같이 변수 두 개를 선언하세요.

빈 객체 product를 생성하고, 변수를 사용하여 동적으로 속성을 추가하세요.

key1 = "productName"; key2 = "productPrice";

초기 객체: { }

업데이트된 객체: { productName: "필통", productPrice: 5000}

08

part

스프레드 연산자 Spread operator

스프레드 연산자란?

스프레드 연산자는 객체이나 배열 같은 집합을 전개해주는 연산자이다.

이 연산자는 배열이나 객체의 요소를 하나씩 분해하여 개별적으로 처리할 수 있다.

스프레드 연산자 사용 방법

아래 예시에서 "...arr"을 사용하면 스프레드 연산자가 배열의 각 요소를 하나씩 분해하여 "1 2 3"이 출력된다.

배열 앞에 "..."(점 세 개)를 붙이면 스프레드 연산자로 설정할 수 있다.

```
let arr = [1, 2, 3];
console.log(arr);
console.log(...arr); // 1 2 3
```

" ... "

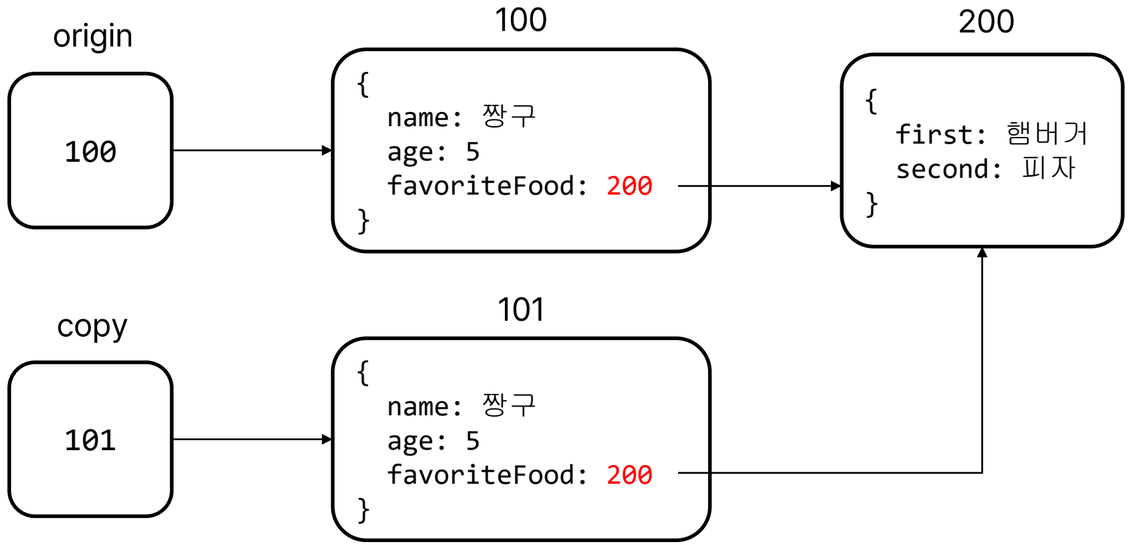
스프레드 연산자

스프레드 연산자

얕은 복사, 깊은 복사

스프레드 연산자를 사용하면 객체를 복사할 수 있지만, 얕은 복사만 이루어진다.
얕은 복사란 객체 안에 또 다른 객체가 포함되어 있는 경우에는 참조 주소만 복사되는 것이다.
이렇게 복사된 객체를 수정하면 원본 객체에 영향을 미칠 수 있다.
이런 경우, 원본 데이터를 완벽하게 복사하려면 깊은 복사를 사용해야 한다.
깊은 복사는 JSON 메소드를 사용하여 처리할 수 있다.

```
let origin = {
  name: "짱구",
  age: 5,
  favoriteFood: {
    first: "햄버거",
    second: "피자",
  }
};
let copy = { ...origin };
```



09

part

구조 분해 할당 Destructuring assignment

구조 분해 할당이란?

구조 분해 할당은 배열이나 객체같이 구조화된 데이터를 분해하여 특정 요소들만 꺼내는 방식이다.

구조 분해 할당을 사용하면 필요한 데이터만 꺼내서 사용할 수 있다.

```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 37,
  email: "john@gmail.com",
  city: "New York",
  country: "USA"
};

let { firstName, lastName } = person;
console.log(firstName); //John
console.log(lastName); //Doe
```

09

part

구조 분해 할당

객체 및 배열 분해하기

객체 분해하기

객체의 속성 이름을 중괄호 {} 안에 변수로 선언한다.

변수의 이름이 객체의 속성명과 일치해야 해당 속성의 값이 변수에 할당된다.

배열 분해하기

배열의 요소를 분해하여 추출 할 수 있다.

대괄호 []를 사용하여, 배열에서 추출한 요소를 담을 변수를 나열한다.

```
let person = {
  firstName: "John",
  lastName: "Doe",
  age: 37
};
var { firstName, lastName } = person;
```

```
let arr = [10, 20, 30];
let [a, b, c] = arr;
```

Q21. 다음과 같이 책 객체를 생성하고, 객체를 복사하세요.
그리고 복사된 객체에서 출판년도를 2023으로 변경하세요.

```
원본객체: {"title":"위대한 개츠비","author":"스콧 피츠제럴드","publishedYear":1925}
복사된객체: {"title":"위대한 개츠비","author":"스콧 피츠제럴드","publishedYear":1925}
원본객체의 출판년도: 1925
복사된객체의 출판년도: 2023
```

Q22. 다음과 같이 책 객체를 생성하고, 객체를 완벽하게 복사하세요.
복사된 객체에서 저자의 성을 "김"으로 변경하세요.

```
원본객체: {"title":"자바 프로그래밍 입문","author":{"firstName":"은종","lastName":"박"}}
복사된객체: {"title":"자바 프로그래밍 입문","author":{"firstName":"은종","lastName":"박"}}
원본객체의 성: 박
복사된객체의 성: 김
```

구조 분해 할당 연습문제

Q23. 다음 객체에서 브랜드(brand)와 가격(price) 속성을 추출하여 변수에 저장하세요.

```
객체: {"brand":"Apple","price":990000,"model":"iPhone 13"}
브랜드: Apple
가격: 990000
```

Q24. 다음 객체에서 제목(title)과 출시연도(year) 속성을 추출하여
각각 'movieTitle'과 'releaseYear'라는 이름의 변수에 저장하세요.

```
객체: {"title":"인셉션","director":"크리스토퍼 놀란","year":2010}
제목: 인셉션
출시연도: 2010
```

Q25. 함수에서 객체를 매개변수로 받아, 'brand'와 'model' 속성을 출력하세요.

```
객체: {"brand":"Tesla","model":"Model S","color":"Red"}
자동차 브랜드: Tesla, 모델명: Model S
```

구조 분해 할당 연습문제

Q26. 배열을 분해하여 모든 요소를 변수에 저장하세요.

배열: ["red", "green", "blue"];

출력: “red” “green” “blue”

Q27. 배열을 분해하여 첫번째와 네번째 요소만를 변수에 저장하세요.

배열: ["사과", "바나나", "오렌지", "키위"];

출력: “사과” “키위”

Q28. 함수에서 배열을 입력받고, 배열의 첫 번째와 세 번째 요소를 출력하세요.

배열: ["사과", "바나나", "오렌지", "키위"];

출력: “사과와 오렌지”

Fetch API

Fetch API란?

Fetch API란?

Fetch API는 웹 어플리케이션에 HTTP 요청을 보내고 응답을 받는 기능이다.

서버로부터 데이터를 비동기 방식으로 받아 온다.

사용 방법

Fetch 함수는 URL로 HTTP 요청을 보낸다.

그리고 then 함수에서 fetch함수가 반환한 응답을 처리한다.

이를 활용하여 다양한 작업을 수행할 수 있다.

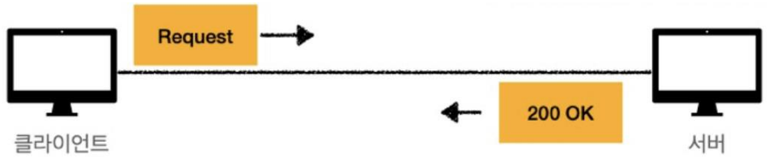
```
fetch(resource)
.then((response) => {
  // ...
});
```

Fetch API

Fetch API 예시

필요한 요소들

- URL: 요청을 보낼 서버의 주소
- 요청메소드: GET, POST, PUT, DELETE 등 (기본값은 GET)
- 바디: 본문데이터. 주로 POST 요청시 사용



예시

```
fetch("https://jsonplaceholder.typicode.com/posts/1")
  .then((response) => response.json())
  .then((json) => console.log(json));
```

결과

```
문제  출력  디버그 콘솔  터미널  포트
C:\Program Files\nodejs\node.exe .\11_fetch.js
{userId: 1, id: 1, title: 'sunt aut facere repellat provide
suscipit recusandae consequun…m rerum est autem sunt rem ev
```

Promise란?

promise는 비동기방식으로 API를 호출할때 사용하는 객체이다.

promise를 사용하면 응답이 올 때까지 기다리지 않고, 다음 코드를 실행할 수 있다.

```
function getData() {
        const xhr = new XMLHttpRequest();
        xhr.open("GET", "요청주소");
        xhr.send();
        xhr.onload = () => {
          if (xhr.status === 200) {
            console.log(xhr.response);
            return res;
          } else {
            console.error(xhr.status);
            return;
          }
        };
      }

      var res = getData();
      console.log(res); //undefined
      console.log("다음 코드를 실행합니다.");
```

응답을 기다렸다가 다음코드 처리

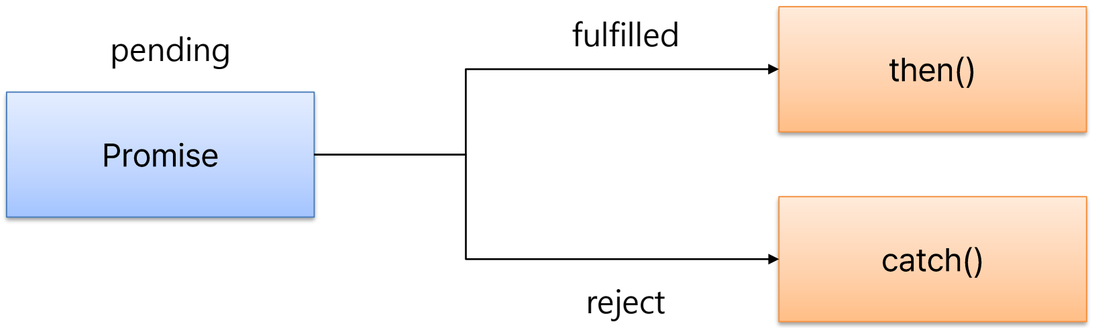
```
function getDataPromise() {
        return new Promise((resolve, reject) => {
          const xhr = new XMLHttpRequest();
          xhr.open("GET", "요청주소");
          xhr.send();
          xhr.onload = () => {
            if (xhr.status === 200) {
                  resolve(xhr.response);
            } else {
                  reject(new Error(xhr.status));
            }
          };
        });
      }

      getDataPromise().then((res) => {
        console.log(res);
        console.log("다음 코드를 실행합니다.");
      });
```

응답을 기다리지 않고 다음 코드 처리

Promise

promise는 진행상황에 따라 세가지 상태를 가진다.
promise를 만들고, 요청을 실행하면 “pending” 상태가 된다.
요청이 성공적으로 끝나면 “fulfilled” 상태가 되고 실패하면 “rejecked” 상태가 된다.

두가지 상태에 따라 처리 로직이 수행된다.
성공적으로 작업을 완료하면 resolve()를 호출하고, then()를 사용하여 남은 작업을 처리한다.
작업에 실패하면 reject()를 호출하고, catch()를 사용하여 남은 작업을 처리한다.



Promise의 상태

Promise 상태	설명
Pending	요청에 대한 응답을 기다리는 상태
Fulfilled	요청이 성공한 상태
Rejected	요청이 실패한 상태

결과 처리 메소드	설명
then	요청에 성공하면 연결됨
catch	요청에 실패하면 연결됨

Async/Await

fetch api는 promise 객체를 포함하고 있다.
그래서 fetch api를 호출한 후에, then으로 다음 코드 실행할 수 있다.
그런데 then을 많이 사용하면 코드가 복잡해진다.

```
fetch( " 요청주소")
 .then((response) => response.json())
 .then((json) => console.log(json));
```

Async/Await을 사용하면 코드를 더 깔끔하게 작성할 수 있다.
API를 호출할 때 함수 앞에 await을 선언하면, 서버에서 응답이 올 때까지 대기하고 결과를 받으면 실행된다.
그리고 await을 사용하는 함수는 반드시 async를 붙여야 한다.

```
async function myFunction() {
  const res = await fetch( " 요청주소");
  const resJson = await res.json();
  console.log(resJson);
}
```

Modules

모듈(Module)이란?

파일 하나에 많은 내용을 정의하면 코드가 복잡해진다.

그래서 함수를 별도의 파일로 분리하고, 필요한 함수만 가져와서 사용하는 것이 좋다.

함수가 정의되어 있는 파일을 모듈이라고 한다.

export: 변수나 함수 앞에 붙이면, 외부에서 사용할 수 있다

import: 외부 모듈에서 가져와서 사용할 수 있다

함수 내보내기

13_module.html

.자바스크립트 고급 문법 > 13_module.html > {} "13_module.html"

```
1  <script type="module">
2    import { log } from "./log.js";
3    log("log로 메시지 출력");
4  </script>
```

함수 가져오기

log.js M

5.자바스크립트 고급 문법 > log.js > ...

```
1  export function log(message) {
2    console.log(message);
3  }
```

Error

에러의 종류

에러	내용
EvalError	eval() 함수에서 발생하는 오류
RangeError	숫자 값이 허용된 범위를 벗어났을 때 발생하는 오류
ReferencError	존재하지 않는 변수를 참조하려고 할 때 발생하는 오류
SyntaxError	코드 구문이 잘못되었을 때 발생하는 오류
TypeError	데이터 유형이 잘못되었을 때 발생하는 오류
URIError	encodeURI 또는 decodeURI와 관련된 오류
AggregateError	여러 개의 오류를 한번에 처리할 때 사용
InternalError	JavaScript 엔진 내부에서 발생하는 오류

정규표현식이란?

정규표현식이란?

문자열에서 특정한 문자를 찾기 위해 사용되는 패턴이다.

데이터 유효성 검사에 주로 활용한다.

어디에 사용될까?

아이디 검사

정규표현식 만들기

정규식 함수

함수	내용
exec	정규표현식과 일치하는 문자열을 검색하여 배열로 반환
test	정규표현식과 문자열을 비교하여 일치 여부를 불리언으로 반환
match	문자열에서 정규표현식과 일치하는 부분을 찾아 배열로 반환
search	문자열에서 정규표현식과 일치하는 첫 번째 위치를 반환
replace	문자열에서 정규표현식과 일치하는 부분을 다른 문자열로 대체
split	정규표현식을 기준으로 문자열을 분할하여 배열로 반환

정규표현식 만들기

정규식 특수문자

특수문자	내용
₩	문자 앞에 ₩가 있으면, 특수문자가 아닌 문자 그대로 해석
^	문자열의 시작을 의미하고, 문자 클래스 내에서는 부정을 의미
$	문자열의 끝을 의미
*	앞의 패턴이 0개 이상 반복되는 것을 의미
+	앞의 패턴이 하나 이상 반복되는 것을 의미
?	앞의 패턴이 0개 또는 1개 존재하는 것을 의미
( )	하위 표현식을 그룹화하며, 일치한 부분 문자열을 추출
{ }	앞의 패턴이 정확히 지정된 횟수만큼 반복되는 것을 의미
[ ]	문자 클래스를 정의하며, 괄호 안에 있는 문자 중 하나와 일치

정규표현식 만들기

정규식 플래그

특수문자	내용
g	전역 검색 – 대응되는 문자 전부 검색
i	대소문자 구분 없는 검색
m	다중행 검색