[2022-2분기 OKR] JAVASCRIPT 기초 part2 강의록

▼ 섹션 0. 함수와 이벤트

▼ 이론 - 함수와 이벤트

💡 함수란? (22.03.12)

> 함수를 사용하면 좋은 것
>
> - 각 명령의 시작과 끝을 명확하게 구별할 수 있음
> - 함수에 별도의 이름을 붙이면 같은 기능이 필요할 때마다 해당 함수 실행 가능

> 매개변수(parameter), 인자값
>
> - function number(a,b){ } > a, b
> - 함수를 실행하기 위해 필요하다고 지정하는 값

> 인수
>
> - 함수를 실행하기 위해 필요하다고 지정하는 값
> - 함수를 실행할 때 매개변수로 넘겨주는 값
> - number(2, 3) > 2, 3

> return 문
>
> - 함수를 실행한 결과 값을 함수 밖으로 넘기는 문
> - 반환된 값은 함수를 호출한 곳으로 넘어감

💡 변수의 적용 범위

> 스코프 (scope)
>
> - 변수를 선언하고 사용할 때 변수가 적용되는 범위

> 지역 변수
>
> - 변수를 선언한 함수에서만 사용할 수 있는 변수
> - var 예약어 사용

> 전역변수
>
> - 스크립트 소스 전체에서 사용할 수 있는 변수
> - 함수 밖 : var 예약어 사용하여 변수 선언
> - 함수 안 : 전역 변수를 선언하려면 var 예약어 없이 선언

💡 익명 함수와 즉시 실행 함수

> 익명 함수 (무명 함수)
>
> - 이름이 없는 함수
> - 함수 자체가 식이기 때문에 함수를 변수에 할당하거나 다른 함수의 매개변수로 사용할 수 있음
> - 예시 : var add = function(a, b){ return a + b; }
> - 예시 : var sum = add(10, 20);

> 즉시 실행 함수
>
> - 함수 정의함과 동시에 실행하는 함수
> - 즉시 실행 함수를 변수에 할당할 수 있고, 함수에서 반환하는 값을 변수에 할당도 가능
> - 예시 : var result = function(a, b) { return a + b; } (10, 20);

> 이벤트
>
> - 웹 문서 영역을 벗어나 하는 동작은 이벤트가 아님
>
> 이벤트 X 예시 : 브라우저 창의 제목 표시줄 클릭

▼ 함수의 종류 (22. 04. 20)

💡 함수의 분류

> 사용자 정의 함수
>
> - 사용자가 필요한 기능을 직접 만든 함수

> 자바스크립트 코어 함수
>
> - 기본적으로 제공하는 함수 (parseInt(), Math.random())

💡 사용 방법에 따른 함수 종류

> 일반 함수
>
> - function add(x,y){ ... }

> 중첩 함수
>
> - 함수 안에 함수가 있는 경우
> - function A (){ function B(){...} }
> - A : 외부 함수 / B : 내부 함수, 중첩 함수

> 콜백 함수 (★)
>
> - 함수 실행결과값을 리턴이 아닌 매개변수로 넘어온 함수(외부)를 호출해서 넘겨주는 방식
> - 이때 매개변수로 넘어온 함수(또 다른 함수)를 콜백함수라고 함
> - function f(x,y, 콜백) { var result = x + y ; callback(result); }
> - 함수의 결과 값을 외부(다른 함수)로 보낼 때 주로 사용
> - 체인처럼 연결, 연결, 연결해서 사용 가능

> 클로저 함수
>
> - 함수 호출에 의해 함수 내부의 실행 구문을 모두 실행하면, 함수 내부에서 만든 지역 변수가 자동으로 사라지는데, 사라지지 않고 남은 경우

💡 중첩 함수

> 중첩 함수란?
>
> - 함수 내부에 새로운 함수 구문을 넣은 것

> 중첩 함수의 용도
>
> - 내부 전용 함수
> - 함수 내부의 지역변수처럼 함수 내부에서만 사용 가능

- 이름 없는 이벤트 리스너로 많이 활용
  - $(function(){ $('.btn').click(function(){ ... } })
- 중복 코드 또는 그룹화
  - 함수 내부의 커다란 기능, 중복 코드를 내부 함수로 만들어 재사용할 때 사용
  - 중첩 함수는 외부 함수와 내부 함수의 아주 밀접한 관계일 때 사용하는 것이 좋음
- 중첩함수와 중첩 함수를 포함한 함수와의 관계
  - 중첩 함수에서 중첩 함수를 포함하고 있는 함수의 지역변수에 접근해서 사용 가능

💡 콜백 함수 (22. 04. 26)

콜백 함수란?

- 함수 내부의 처리 결과값을 함수 외부로 내보낼 때 사용
- 일종의 return문과 비슷한 기능
- 로직 구현 부분은 동일, 로직 처리 부분을 다양하게 처리해야 하는 경우에 유용
- 유지보수에 용이
- 동기, 비동기 관련

콜백 함수와 return문

- return문
  - 콜백 함수와 결과는 동일
  - 단순한 처리에 적합
- 콜백 함수
  - 로직 처리 부분이 복잡한 경우, 구현과 처리가 분리되어야 하는 경우, 구현은 하나로 했지만 처리를 다양하게 만든 경우 적합

동기와 비동기

- 콜백 함수를 이해하기 위해 동기, 비동기 개념 이해 필요

  ★콜백 함수가 주로 비동기 함수의 결과 값을 처리하기 위한 방법으로 사용되기 때문에 중요★
- 동기
  - 일반적으로 함수가 호출된 후 끝날 때까지 다음 구문을 실행하지 않고 대기하는 것 (직렬적)
  - 예) alert창의 확인/닫기를 눌러야 다음 명령(예: document.write)이 실행됨
  - 단점: 반드시 사용자의 손길을 거쳐야만 프로그램 실행 가능
- 비동기
  - 동기의 반대 개념. 함수가 호출된 후 끝날 때까지 기다리지 않고 다음 구문을 바로 실행 (병렬적)
  - 예) setInterval 함수 호출된 후 끝날 때까지 기다리지 않고 다음 명령(예: document.write) 실행
  - 장점: 동기에 비해 시간이 효율적

콜백 함수의 실무적 형태 (비동기)

- 콜백 함수는 로직 구현, 로직 처리 부분을 나눠 작업할 때 주로 사용
- 이벤트 리스너 (버튼 클릭 등)
- 타이머 실행 함수로 사용 (특정 시간마다 실행하는 경우)
- 서버와 데이터를 주고 받을 때 사용하는 Ajax 기능들에서 결과물을 받을 때 (서버 접속, DB 접속 등)
- jQuery 애니메이션 기능에서 애니메이션이 모두 끝났을 때, 실행하고자 하는 함수

💡 클로저 함수

클로저란?

> - 함수 내부에 만든 지역변수가 사라지지 않고 계속해서 값을 유지하고 있는 상태
> - 예) function start() { var count = 0; setInterval function(){ count++; alert(count); }, 300 }
>
> start()가 실행되며 생성된 count 지역변수가 setInterval 함수에서도 사용되며 값이 계속 증가됨. 이때, 익명함수를 클로저 함수라고 함
> - 클로저 변수가 사라지지 않고 계속해서 값을 유지하는 상태를 전부다 클로저라고 함

> 클로저의 장점
> - 연관 있는 변수와 기능(중첩함수)을 하나의 함수로 묶어서 독립적으로 실행 가능
> - 함수 내부에 데이터가 만들어지기 때문에 함수 외부에서 수정 할 수 없는 보호된 데이터를 만들 수 있음
> - = 객체지향 프로그래밍(자바, C언어)에서는 PRIVATE 데이터 = 캡슐화 된 데이터라고 부름

▼ 섹션 1. 코어 라이브러리 (1) (22. 05. 07)

▼ 코어 라이브러리란?

- 자바스크립트가 개발자를 위해 기본적으로 제공해주는 기능
- 개발자만의 고유 라이브러리를 만들 수 있음

▼ 타이머 함수

💡 타이머함수란?

- 일정한 시간마다 특정 구문을 실행하고자 할 때 사용하는 기능
- 슬라이더, 배너, 자동으로 변경되는 실시간 검색어 기능, 게임에서 플레이 시간 나타내는 기능

💡 타이머함수의 종류

- setInterval() : 일정 시간마다 주기적으로 특정 구문 실행
- setTIimeout() : 일정 시간 지난 후 특정 구문 딱 한번 실행
- clearInterval() : 실행 중인 타이머 함수 멈추는 기능
- 타이머 함수는 모두 전역 객체인 window에 포함되어 있음

💡 일정 시간 마다 특정 구문 실행하는 타이머

- setInterval() 함수 사용
- var timerID = setinterval(function (func, duration));
- 매개변수
  - func : 지연되는 시간마다 타이머 함수에 의해 호출되는 일종의 콜백 함수
  - duration : 지연 시간, 단위는 밀리초
- 리턴값 : 생성된 timerID , 이 timerID는 함수를 멈출 때도 사용

💡 일정 시간 지난 후 딱 한번 실행되는 타이머

- setTimeout() 함수 사용
- var timerID = setTimeout(function (func, duration))
- 매개변수
  - func : 지연되는 시간마다 타이머 함수에 의해 호출되는 일종의 콜백 함수
  - duration : 지연 시간, 단위는 밀리초

💡 타이머 멈추기

- clearInterval(timerID)
- 매개변수
  - timerID : 제거할 타이머

▼ Math 클래스

💡 Math 클래스의 용도

- 숫자를 랜덤으로 생성, 사인, 코싸인 같은 수학 관련 기능이 담겨 있음
- 배너, 슬라이더의 컨텐츠를 랜덤으로 보여줄 때 사용
- 컨텐츠 위치를 무작위로 설정할 때도 Math.random() 사용
- 게시판의 페이지 수 구할 때 Math.ceil() 사용
- 이미지를 곡선에 따라 나열하고 싶을 때 Math.sin() 사용

💡 프로퍼티

- PI : 원주율 값

💡 함수 목록

- ceil() : 올림값
- floor : 버림값
- log : 자연 로그 값 반환
- max : 두 수 중 큰 값 반환
- min : 두 수 중 작은 값 반환
- random : 0과 1 사이의 난수 반환
- round : 가장 가까운 정수로 반올림하거나 반내림한 값 반환
- sqrt : 제곱근 반환

💡 장점

- Math 클래스는 다른 코어 클래스와 달리, 대부분의 기능이 클래스 메서드(정적 메서드)로 구현되어 있어서 인스턴스 생성 없이 바로 사용 가능

💡 랜덤 숫자 만들기

- var num = (Math.random() * 범위값) + 시작값

💡 작은 값, 큰 값 알아내기

- var max = Math.max(num1, num2)
- var min = Math.min(num1, num2)
- 리턴값 : 두개 중 작은/큰 값 리턴

💡 숫자 내림값, 올림값 구하기

- var result - Math.floor(num)
- var result - Math.ceil(num)
- 리턴값 : 입력 값이 실수인 경우 내림/올림한 정수값

▼ 섹션 3. 코어 라이브러리 (2) (22. 05. 29)

▼ string 클래스

💡 **string 클래스란?**

- 문자열을 생성하는 기능, 문자열과 관련된 유효한 기능이 있음

💡 **string 클래스의 실무 활용**

- 아이디와 패스워드의 좌우 공백 없애주는 기능

- 게시물의 특정 문자열을 다른 문자열로 치환/삭제

> 💡 **string 클래스의 주요 기능**

> 프로퍼티
>
> - length : 문자열 개수

> 함수 목록
>
> - 용어 설명
>   - *자바스크립트는 **객체**를 통해 현실의 모델 구현*
>   - *클래스 : 비슷한 객체들을 빠르게 만들도록 도움*
>   - *인스턴스 : 클래스의 속성과 메서드를 담고 있는 객체*
>
>   *클래스로 만든 객체 = 인스턴트*
>   - *정규 표현식(패턴) : 예) 전화번호는 무조건 010으로 시작*
>
> ▼ 메소드 종류

메소드	설명
indexOf()	String 인스턴스에서 특정 문자나 문자열이 처음으로 등장하는 위치의 인덱스를 반환함.
lastIndexOf()	String 인스턴스에서 특정 문자나 문자열이 마지막으로 등장하는 위치의 인덱스를 반환함.
charAt()	String 인스턴스에서 전달받은 인덱스에 위치한 문자를 반환함.
charCodeAt()	String 인스턴스에서 전달받은 인덱스에 위치한 문자의 UTF-16 코드를 반환함. (0 ~ 65535)
charPointAt()	String 인스턴스에서 전달받은 인덱스에 위치한 문자의 유니코드 코드 포인트(unicode code point)를 반환함.
slice()	String 인스턴스에서 전달받은 시작 인덱스부터 종료 인덱스 바로 앞까지의 문자열을 추출한 새 문자열을 반환함.
substring()	String 인스턴스에서 전달받은 시작 인덱스부터 종료 인덱스 바로 앞까지의 문자열을 추출한 새 문자열을 반환함.
substr()	String 인스턴스에서 전달받은 시작 인덱스부터 길이만큼의 문자열을 추출한 새로운 문자열을 반환함.
split()	String 인스턴스에서 구분자(separator)를 기준으로 나눈 후, 나뉜 문자열을 하나의 배열로 반환함.
concat()	String 인스턴스에 전달받은 문자열을 결합한 새로운 문자열을 반환함.
toUpperCase()	String 인스턴스의 모든 문자를 대문자로 변환한 새로운 문자열을 반환함.
toLowerCase()	String 인스턴스의 모든 문자를 소문자로 변환한 새로운 문자열을 반환함.
trim()	String 인스턴스의 양 끝에 존재하는 공백과 모든 줄 바꿈 문자(LF, CR 등)를 제거한 새로운 문자열을 반환함.
search()	인수로 전달받은 정규 표현식에 맞는 문자나 문자열이 처음으로 등장하는 위치의 인덱스를 반환함.
replace()	인수로 전달받은 패턴에 맞는 문자열을 대체 문자열로 변환한 새 문자열을 반환함.
match()	인수로 전달받은 정규 표현식에 맞는 문자열을 찾아서 하나의 배열로 반환함.
includes()	인수로 전달받은 문자나 문자열이 포함되어 있는지를 검사한 후 그 결과를 불리언 값으로 반환함.
startsWith()	인수로 전달받은 문자나 문자열로 시작되는지를 검사한 후 그 결과를 불리언 값으로 반환함.
endsWith()	인수로 전달받은 문자나 문자열로 끝나는지를 검사한 후 그 결과를 불리언 값으로 반환함.
toLocaleUpperCase()	영문자뿐만 아니라 모든 언어의 문자를 대문자로 변환한 새로운 문자열을 반환함.
toLocaleLowerCase()	영문자뿐만 아니라 모든 언어의 문자를 소문자로 변환한 새로운 문자열을 반환함.
localeCompare()	인수로 전달받은 문자열과 정렬 순서로 비교하여 그 결과를 정수 값으로 반환함.

normalize()	해당 문자열의 유니코드 표준화 양식(Unicode Normalization Form)을 반환함.
repeat()	해당 문자열을 인수로 전달받은 횟수만큼 반복하여 결합한 새로운 문자열을 반환함.
toString()	String 인스턴스의 값을 문자열로 반환함.
valueOf()	String 인스턴스의 값을 문자열로 반환함.

💡 **string 클래스의 핵심 내용**

문자열 만들기

1. **리터럴** 방식 : var str = “hi”;
2. **stirng 클래스의 객체**를 생성해 이용 : var str = new String(”hi”);

- “hi”.length = 2 / “hi”.charAt(0) = h
- 위의 구문이 정상적 실행되는 이유

  : **리터럴 방식**(1) 구문은 자바스크립트에 의해 해석될 때, **string 클래스의 객체**를 생성해서 이용하는 방식(2)으로 변환되어 실행되기 때문
- “hi”.length ⇒ new String(”hi”).length;

문자열 길이 알아내기

- length

특정 위치의 문자 구하기

- charAt

문자열 위치 찾기

- indexOf(찾는 문자열, 시작 위치)
- 리턴값 : 찾는 문자열의 위치 값 (찾지 못하면 -1 반환)

특정 위치에 문자 추가/제거

- slice + substr 이용
- slice (문자열 시작 위치, 문자열 끝 위치)
  - 리턴값 : 지정한 문자열
  - 주의 : 시작 인덱스부터 마지막 인덱스 “전”까지 문자열 “복사” (잘라내기 아님)
- substr (문자열 시작 위치, 문자열 개수)
  - 리턴값 : 지정한 문자열

특정 위치의 문자를 다른 문자로 변경

- replace (찾는 문자, 대체 문자)

▼ Date 클래스

💡 date 클래스란?

- 날짜/시간과 관련된 유용한 기능

💡 주요 기능

- get * : 데이터 받아옴
- set * : 데이터 설정함

> 주요 기능
>
> `Date.prototype.getDate()`
>
> `Date`에서 현지 시간 기준 일(`1`–`31`)을 반환합니다.
>
> `Date.prototype.getDay()`
>
> `Date`에서 현지 시간 기준 요일(`0`–`6`)을 반환합니다.
>
> `Date.prototype.getFullYear()`
>
> `Date`에서 현지 시간 기준 연도(네 자리 연도면 네 자리로)를 반환합니다.
>
> `Date.prototype.getHours()`
>
> `Date`에서 현지 시간 기준 시(`0`–`23`)를 반환합니다.
>
> `Date.prototype.getMilliseconds()`
>
> `Date`에서 현지 시간 기준 밀리초(`0`–`999`)를 반환합니다.
>
> `Date.prototype.getMinutes()`
>
> `Date`에서 현지 시간 기준 분(`0`–`59`)을 반환합니다.
>
> `Date.prototype.getMonth()`
>
> `Date`에서 현지 시간 기준 월(`0`–`11`)을 반환합니다.
>
> `Date.prototype.getSeconds()`
>
> `Date`에서 현지 시간 기준 초(`0`–`59`)를 반환합니다.
>
> `Date.prototype.getTime()`
>
> 1970년 1월 1일 00:00:00 UTC로부터의 경과시간을 밀리초 단위로 반환합니다. `Date`가 기준 시간 이전을 나타낼 경우 음수 값을 반환합니다.
>
> `Date.prototype.getTimezoneOffset()`
>
> 현지 시간대와 UTC의 차이를 분 단위로 반환합니다.
>
> `Date.prototype.getUTCDate()`
>
> `Date`에서 국제 시간 기준 일(`1`–`31`)을 반환합니다.
>
> `Date.prototype.getUTCDay()`
>
> `Date`에서 국제 시간 기준 요일(`0`–`6`)을 반환합니다.
>
> `Date.prototype.getUTCFullYear()`
>
> `Date`에서 국제 시간 기준 연도(네 자리 연도면 네 자리로)를 반환합니다.
>
> `Date.prototype.getUTCHours()`
>
> `Date`에서 국제 시간 기준 시(`0`–`23`)를 반환합니다.
>
> `Date.prototype.getUTCMilliseconds()`
>
> `Date`에서 국제 시간 기준 밀리초(`0`–`999`)를 반환합니다.
>
> `Date.prototype.getUTCMinutes()`
>
> `Date`에서 국제 시간 기준 분(`0`–`59`)을 반환합니다.
>
> `Date.prototype.getUTCMonth()`
>
> `Date`에서 국제 시간 기준 월(`0`–`11`)을 반환합니다.
>
> `Date.prototype.getUTCSeconds()`
>
> `Date`에서 국제 시간 기준 초(`0`–`59`)를 반환합니다.
>
> `Date.prototype.setDate()`
>
> 현지 시간 기준으로 일을 설정합니다.
>
> `Date.prototype.setFullYear()`
>
> 현지 시간 기준으로 연도(네 자리 연도면 네 자리로)를 설정합니다.
>
> `Date.prototype.setHours()`
>
> 현지 시간 기준으로 시를 설정합니다.
>
> `Date.prototype.setMilliseconds()`
>
> 현지 시간 기준으로 밀리초를 설정합니다.
>
> `Date.prototype.setMinutes()`
>
> 현지 시간 기준으로 분을 설정합니다.
>
> `Date.prototype.setMonth()`
>
> 현지 시간 기준으로 월을 설정합니다.
>
> `Date.prototype.setSeconds()`
>
> 현지 시간 기준으로 초를 설정합니다.
>
> `Date.prototype.setTime()`
>
> `Date`가 나타낼 시간을 1970년 1월 1일 00:00:00 UTC로부터의 경과시간(밀리초)으로 설정합니다. 기준 이전의 시간은 음수 값을 사용해 설정할 수 있습니다.
>
> `Date.prototype.setUTCDate()`

국제 시간 기준으로 일을 설정합니다.

`Date.prototype.setUTCFullYear()`

국제 시간 기준으로 연도(네 자리 연도면 네 자리로)를 설정합니다.

`Date.prototype.setUTCHours()`

국제 시간 기준으로 시를 설정합니다.

`Date.prototype.setUTCMilliseconds()`

국제 시간 기준으로 밀리초를 설정합니다.

`Date.prototype.setUTCMinutes()`

국제 시간 기준으로 분을 설정합니다.

`Date.prototype.setUTCMonth()`

국제 시간 기준으로 월을 설정합니다.

`Date.prototype.setUTCSeconds()`

국제 시간 기준으로 초를 설정합니다.

`Date.prototype.toDateString()`

`Date`의 날짜 부분만 나타내는, 사람이 읽을 수 있는 문자열을 반환합니다.

`Date.prototype.toISOString()`

`Date`를 나타내는 문자열을 ISO 8601 확장 형식에 맞춰 반환합니다.

`Date.prototype.toJSON()`

`toISOString()`을 사용해서 `Date`를 나타내는 문자열을 반환합니다. `JSON.stringify()`에서 사용합니다.

`Date.prototype.toLocaleDateString()` (en-US)

`Date`의 날짜 부분을 나타내는 문자열을 시스템에 설정된 현재 지역의 형식으로 반환합니다.

`Date.prototype.toLocaleFormat()`

형식 문자열을 사용해서 `Date`를 나타내는 문자열을 생성합니다.

`Date.prototype.toLocaleString()` (en-US)

`Date`를 나타내는 문자열을 현재 지역의 형식으로 반환합니다. `Object.prototype.toLocaleString()` 메서드를 재정의합니다.

`Date.prototype.toLocaleTimeString()` (en-US)

`Date`의 시간 부분을 나타내는 문자열을 시스템에 설정된 현재 지역의 형식으로 반환합니다.

`Date.prototype.toString()`

`Date`를 나타내는 시간 문자열을 반환합니다. `Object.prototype.toString()` 메서드를 재정의합니다.

`Date.prototype.toTimeString()` (en-US)

`Date`의 시간 부분만 나타내는, 사람이 읽을 수 있는 문자열을 반환합니다.

`Date.prototype.toUTCString()` (en-US)

`Date`를 나타내는 문자열을 UTC 기준으로 반환합니다.

`Date.prototype.valueOf()`

`Date` 객체의 원시 값을 반환합니다. `Object.prototype.valueOf()` 메서드를 재정의합니다.

💡 핵심 내용

시간, 분, 초, 밀리초

① getHours()

var hours = Date인스턴스.getHours(); //리턴값은 0~23까지의 정수

② getMinutes()

var minutes = Date인스턴스. getMinutes(); //리턴값은 0~59까지의 정수

③ getSeconds()

var seconds = Date인스턴스. getSeconds(); //리턴값은 0~59까지의 정수

④ getMilliseconds()

var mSeconds = Date인스턴스. getMilliseconds(); //리턴값은 0~999까지의 정수

년, 월, 일, 요일

① getFullYear()

var year = Date인스턴스. getFullYear(); //리턴값은 네 자리로 된 연도 값

② getMonth()

var month = Date인스턴스. getMonth(); //리턴값은 0(1월)~11(12월)까지의 정수

③ getDate()

var date = Date인스턴스. getDate(); //리턴값은 1~31까지의 정수

④ getDay()

var day = Date인스턴스. getDay(); //리턴값은 0(일요일)~6(토요일)까지의 정수

▼ Array 클래스

💡 **array 클래스란?**

- 배열을 만드는 기능부터 추가, 삭제, 찾기 등 유용한 기능

💡 **주요 기능**

프로퍼티

- length : 배열의 크기(갯수)

메서드 목록

concat(array)	두 개의 배열을 하나의 배열로 합침
push(data)	배열의 끝에 데이터를 추가 후 배열의 길이를 리턴
pop()	배열의 마지막 항목을 제거하고 제거된 데이터 반환
shift()	배열의 첫 번째 데이터 제거 후 해당 값을 리턴
unshift(data)	배열의 맨 앞에 데이터를 추가 후 배열의 길이 반환
join(구분자)	지정한 구분자 이용 배열을 문자열로 변환 후 반환
splice(index, 제거수, [data1, .. dataN])	지정 인덱스 번호부터 제거할 개수만큼 데이터 제거. data를 전달하여 대체하는 것도 가능.
slice(시작인덱스[,종료인덱스])	시작 인덱스부터 종료 인덱스 사이의 데이터를 배열로 리턴
reverse()	배열의 데이터 순서를 거꾸로 변경
sort()	배열의 데이터를 문자열로 변환 후 사전식으로 정렬

💡 **핵심 내용**

배열 만들기

- 배열 리터럴 방식
  - var arr = [num1, num2, num3, num4]
  - 내부적으로 배열 클래스 방식으로 변환되어 실행
  - 실무에서 간결한 리터럴 방식 이용
- 배열 클래스 방식
  - var arr = new Array(num1, num2, num3, num4)

특정 위치의 배열 요소 접근

- 배열의 n번째 요소에 접근하는 기능은 [ ] 안에 인덱스 값 넣기
- var 변수 = 배열명[인덱스]

배열 문자열로 만들기

- join 메서드 사용
- var arr = 배열명.join([seperator]);

문자열 배열로 만들기

- split 메서드 사용
- var arr = 문자열.split(seperator);
- 매개변수 : 구분자로 사용할 문자열
- 리턴값 : 구분자로 나눠 만들어진 배열

특정 위치에 배열 요소 추가

1. 배열 마지막 위치에 배열 요소 추가
    - push 메서드 사용
    - var result = 배열.push(element, [element])
    - 매개변수 : 배열 마지막 위치에 추가할 배열 요소
    - 리턴값 : 신규 배열 요소를 추가한 배열 리턴
2. 배열 첫번째 위치에 배열 요소 추가
    - unShift 메서드 사용
3. 배열 n번째 위치에 배열 요소 추가
    - splice 메서드 사용
    - var result = 배열.splice(start, deletCount, [element])
    - start : 추가/삭제할 배열 요소의 시작 위치
    - deletCount : start 부터 시작하여 삭제할 배열 요소의 개수 (추가할 땐 0)
    - element : 추가 요소
    - 리턴값 : 삭제한 배열 요소를 추가한 배열 리턴 (추가할 경우엔 리턴값 없음)

배열 정렬하기

1. sort 메서드
    - var result = 배열.([sompareFunction])
    - 매개변수 : compareFunction은 정렬 순서를 정의하는 함수
      * 생략 시 문자를 오름차순으로 정렬
    - 리턴값 : 정렬이 완료된 결과값, 정렬에 사용된 원본도 변경
2. 문자 오름차순 정렬
    - 작은 것 > 큰 것
    - 배열.sort(function(a,b){
        return a - b;
      })
3. 문자 내림차순 정렬
    - 큰 것 > 작은 것
    - 배열.sort(function(a,b){
        return b > a;
      })
4. 숫자 정렬
    - sort는 문자열 정렬 기능이기 때문에 숫자도 문자열로 처리
    - compareFunction 활용해야 함