JavaScript 빠른 참조

ES6+, DOM, 이벤트, Fetch API, async/await

기본

변수

```
let name = "Alice";    // 재할당 가능
const PI = 3.14;       // 상수
var old = "avoid";     // 함수 스코프 (레거시)
```

데이터 타입

string	문자열: **"hello"** 또는 **'hello'**
number	정수 또는 실수: **42**, **3.14**
boolean	**true** / **false**
null	의도적인 빈 값
undefined	선언됐지만 값이 할당되지 않음
object	키-값 쌍: **{ a: 1 }**
array	순서 있는 목록: **[1, 2, 3]**
symbol	고유 식별자

타입 확인 및 변환

```
typeof "hello"     // "string"
typeof 42          // "number"
Number("42")       // 42
String(100)        // "100"
parseInt("3.9")    // 3
parseFloat("3.14") // 3.14
```

문자열

템플릿 리터럴

```
const name = "Alice";
`Hello, ${name}!`       // Hello, Alice!
`Total: ${2 + 3}`       // Total: 5
`Multi
 line string`
```

문자열 메서드

s.length	문자 수
s.toUpperCase()	대문자로 변환된 복사본
s.toLowerCase()	소문자로 변환된 복사본
s.trim()	앞뒤 공백 제거
s.split(",")	배열로 분리
s.includes("x")	포함 여부 확인 → bool
s.indexOf("x")	첫 번째 인덱스 (-1이면 없음)
s.slice(1, 4)	인덱스로 부분 문자열 추출
s.replace(a, b)	첫 번째 매치 교체
s.replaceAll(a, b)	모든 매치 교체
s.startsWith(x)	접두사 확인 → bool
s.endsWith(x)	접미사 확인 → bool
s.padStart(n, c)	길이 n이 될 때까지 앞에 패딩

배열

생성 및 접근

```
const fruits = ["apple", "banana", "cherry"];
fruits[0]          // "apple"
fruits.length      // 3
fruits.at(-1)      // "cherry"
```

변경 메서드

arr.push(x)	끝에 추가
arr.pop()	마지막 요소 제거 및 반환
arr.unshift(x)	앞에 추가
arr.shift()	첫 번째 요소 제거 및 반환
arr.splice(i, n)	인덱스 i에서 n개 제거
arr.sort()	제자리 정렬 (사전순)
arr.reverse()	제자리 역순 정렬

비변경 메서드

arr.map(fn)	각 요소를 변환
arr.filter(fn)	fn이 true인 요소만 유지
arr.reduce(fn, init)	단일 값으로 누적
arr.find(fn)	첫 번째 매치 또는 undefined
arr.findIndex(fn)	첫 번째 매치의 인덱스 (-1)
arr.includes(x)	포함 여부 확인 → bool
arr.slice(a, b)	부분 얕은 복사
arr.join(",")	문자열로 결합
arr.forEach(fn)	순회 (반환값 없음)
[...a, ...b]	배열 연결 (스프레드)

객체

생성 및 접근

```
const user = { name: "Alice", age: 20 };
user.name          // "Alice"
user["age"]        // 20
user.gpa = 3.85;   // 추가/업데이트
```

구조 분해 및 스프레드

```
const { name, age } = user;
const copy = { ...user, age: 21 };
```

객체 메서드

Object.keys(o)	키 배열
Object.values(o)	값 배열
Object.entries(o)	[키, 값] 쌍 배열
Object.assign(t, s)	속성 복사 s → t
"k" in obj	키 존재 여부 → bool
delete obj.k	속성 제거
Object.freeze(o)	불변으로 만들기 (얕은 동결)

제어 흐름

if / else if / else

```
if (score >= 90) {
  grade = "A";
} else if (score >= 80) {
  grade = "B";
} else {
  grade = "C";
}
```

삼항 연산자 & 널 병합 연산자

```
const status = score >= 60 ? "pass" : "fail";
const name = user.name ?? "Anonymous";
```

switch

```
switch (color) {
  case "red":   stop();   break;
  case "green": go();     break;
  default:      wait();
}
```

반복문

for / for...of / for...in

```
for (let i = 0; i < 5; i++) { }

for (const item of ["a", "b"]) { }  // arrays

for (const key in obj) { }  // object keys
```

while / do...while

```
while (count < 10) { count++; }

do { count++; } while (count < 10);
```

break & continue

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break;       // stop loop
  if (i % 2 === 0) continue; // skip
}
```

함수

함수 선언 및 화살표 함수

```
function greet(name) {
  return `Hello, ${name}!`;
}
const greet = (name) => `Hello, ${name}!`;
const square = x => x * x; // single param
```

기본 매개변수 & 나머지 매개변수

```
function greet(name = "World") { }

function sum(...nums) {
  return nums.reduce((a, b) => a + b, 0);
}
```

콜백

```
[1, 2, 3].map(x => x * 2);    // [2, 4, 6]
[1, 2, 3].filter(x => x > 1); // [2, 3]
setTimeout(() => console.log("done"), 1000);
```

클래스

```
class Dog {
  constructor(name, breed) {
    this.name = name;
    this.breed = breed;
  }
  bark() { return `${this.name} says Woof!`; }
}

class Puppy extends Dog {
  constructor(name, breed, toy) {
    super(name, breed);
    this.toy = toy;
  }
}
```

에러 처리

```
try {
  JSON.parse("bad json");
} catch (err) {
  console.error(err.message);
} finally {
  console.log("always runs");
}
```

JavaScript 빠른 참조

에러 던지기

```
throw new Error("Something went wrong");
```

DOM

요소 선택

```
document.querySelector(".cls")    // first match
document.querySelectorAll("li")   // all matches
document.getElementById("main")
```

요소 수정

```
el.textContent = "new text";
el.innerHTML = "<b>bold</b>";
el.style.color = "red";
el.classList.add("active");
el.classList.toggle("hidden");
el.setAttribute("data-id", "42");
```

이벤트

```
btn.addEventListener("click", (e) => {
  console.log(e.target);
});
```

요소 생성

```
const li = document.createElement("li");
li.textContent = "New item";
ul.appendChild(li);
el.remove(); // remove element
```

Fetch API

GET 요청

```
fetch("https://api.example.com/data")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

POST 요청

```
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ key: "value" }),
});
```

Async / Await

```
async function loadData() {
  try {
    const res = await fetch(url);
    const data = await res.json();
    return data;
  } catch (err) {
    console.error(err);
  }
}
```

병렬 요청

```
const [users, posts] = await Promise.all([
  fetch("/users").then(r => r.json()),
  fetch("/posts").then(r => r.json()),
]);
```

모듈

이름 있는 내보내기

```
// math.js
export const PI = 3.14;
export function add(a, b) { return a + b; }

// main.js
import { PI, add } from "./math.js";
```

기본 내보내기

```
// logger.js
export default function log(msg) { }

// main.js
import log from "./logger.js";
```