Part 3
자바스크립트로 웹 동작 구현하기

길벗

코딩 자율학습

HTML+CSS+자바스크립트

8장
자바스크립트 시작하기

8.1 자바스크립트 코드 작성 방법

8.2 프로그래밍 시작 전 알아 두기

8.1 자바스크립트 코드 작성 방법

1. HTML 파일과 자바스크립트 연결하기

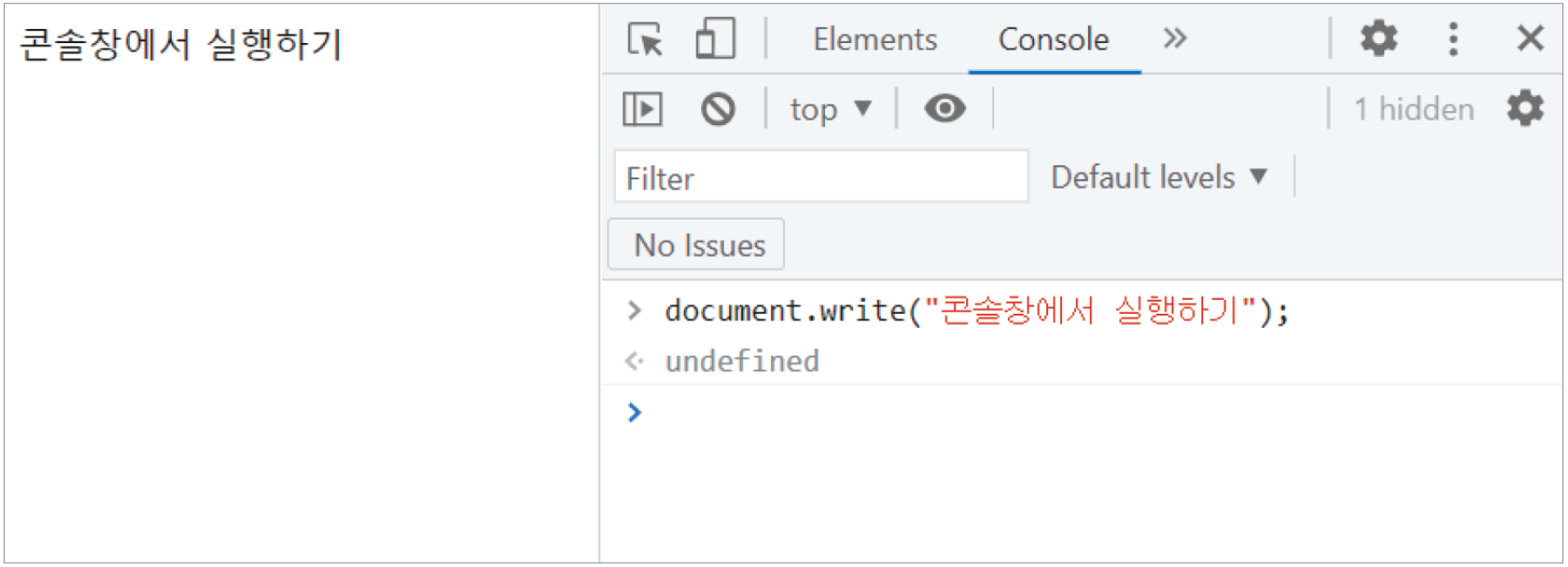
- 내부 스크립트 방법(internal script) : HTML 문서 안에서 script 태그의 콘텐츠 영역에 자바스크립트 코드를 작성
- 외부 스크립트 방법(external script) : 별도의 js 확장자 파일을 만들어 자바스크립트 코드를 작성하고 이 파일을 HTML 문서에서 script 태그로 연결
- script 태그의 사용 위치 : script 태그는 웹 브라우저에 화면이 표시되는 것에 영향을 미치지 않도록 body 태그가 끝나기 전에 사용

8.1 자바스크립트 코드 작성 방법

2. 자바스크립트 코드 실행하기

- 웹 브라우저의 개발자 도구에서 지원하는 콘솔창 활용하기

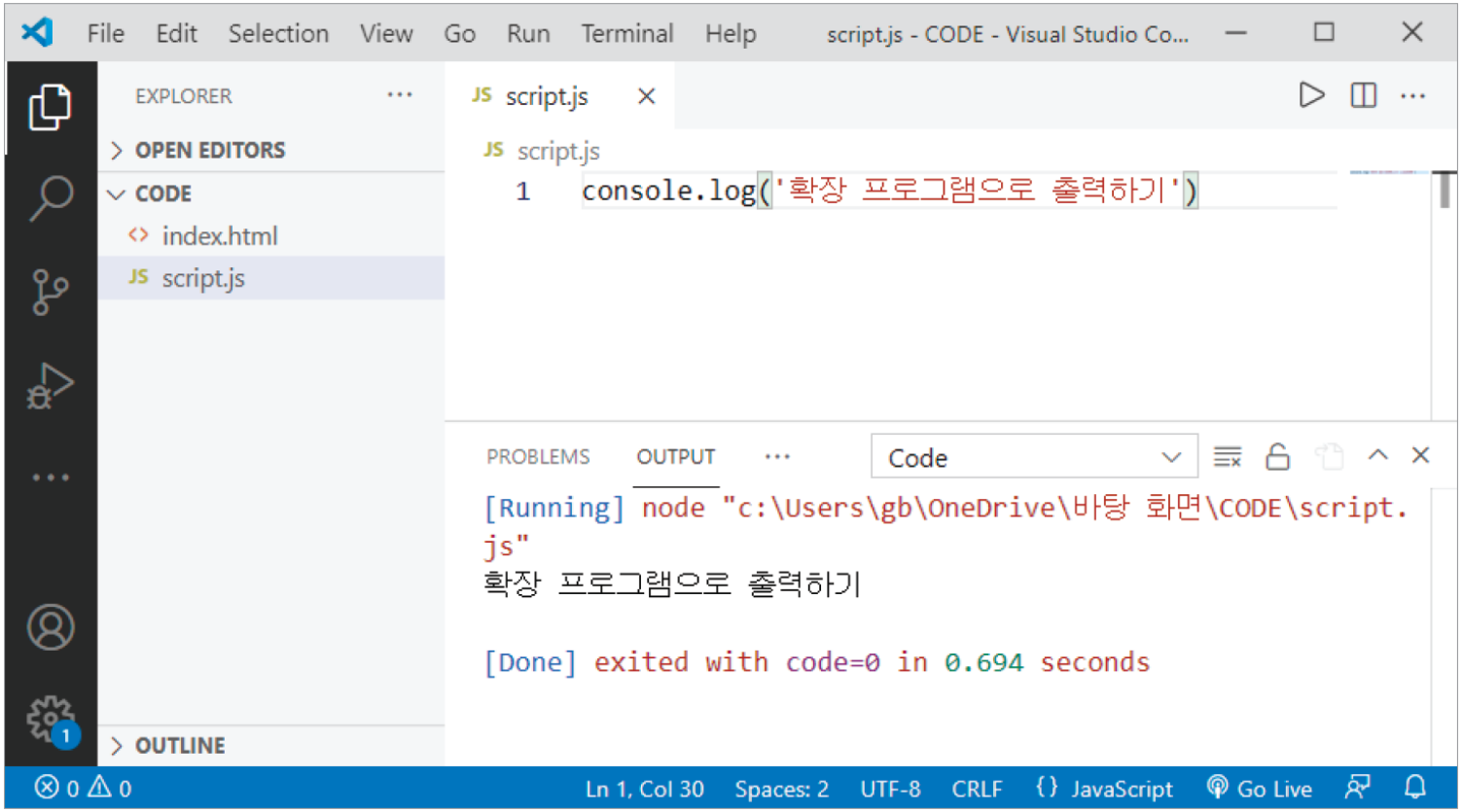
그림 8-4 콘솔창에서 자바스크립트 코드 실행하기



8.1 자바스크립트 코드 작성 방법

- VSCode의 Code Runner 확장 프로그램 활용하기

그림 8-7 Code Runner 실행결과



8.2 프로그래밍 시작 전 알아 두기

1. 주석

- 한 줄 주석 : // 기호(슬래시 2개)로 작성
- 여러 줄 주석 : /* 기호와 */ 기호 사이에 작성

2. 자바스크립트 오류 확인 방법

- 프로그래밍 언어의 실행 방법
  - 컴파일 방식 : 모든 코드를 기계어로 변환 후 실행
  - 인터프리터 방식 : 코드를 한 번에 한 줄씩 실행
- 오류가 발생하면 그 즉시 실행을 멈추고 오류 메시지와 오류가 발생한 줄 번호를 웹 브라우저의 콘솔창에 출력
- 모든 오류 관련 메시지는 웹 브라우저의 콘솔창에서 확인 가능

9장
자바스크립트 기초 문법 살펴보기

9.1 변수와 상수

1. 변수

- 변수(variant) : 변하는 수. 값이 변하는 데이터를 저장하고 관리하기 위한 공간
- 키워드(keyword) : 자바스크립트 프로그래밍 언어에서 어떤 역할이나 기능이 정해진 특별한 단어. 예약어(reserved word)라고도 함
- 식별자(identifier) : 자바스크립트 내부에서 변수, 함수 등에 부여되는 이름
- 연산자(operator) : 어떠한 연산 작업을 하는 데 사용하는 기호
- 표현식(expression) : 평가되어 하나의 값을 반환하는 식 또는 코드
- 값(value) : 더 이상 평가할 수 없는 데이터

9.1 변수와 상수

- 선언, 할당, 초기화
  - 변수 선언 : 변수를 생성하고 값을 저장하는 문법에서 var, let, const 키워드를 사용해 변수의 식별자를 지정하는 것
  - 값 할당 : 할당 연산자인 = 기호로 우변에 있는 값을 변수 공간에 대입(저장)하는 것
  - 변수 초기화 : 선언과 할당을 같이(한 번에) 하는 것

9.1 변수와 상수

2. 새로운 변수 선언 키워드 let

- 변수명 중복이 불가능
- 호이스팅되지 않음
  - 호이스팅(hoisting) : var 키워드로 변수를 선언하고 할당했을 때, 변수 선언을 자바스크립트의 스코프(scope) 맨 위로 올려 실행하는 것
- 스코프의 범위가 다름

9.1 변수와 상수

3. 상수

- 상수(constant) : 변하지 않는 수
- const 키워드는 재할당이 안 되는 특징 때문에 상수 변수(constant variable)를 선언할 때 사용하는 키워드라고 하기도 함

4. 식별자 명명 규칙

- 강제적 식별자 명명 규칙
  - 식별자에 키워드 사용 불가 예 var, let, const
  - 식별자에 공백 포함 불가 예 my School, like food
  - 식별자의 첫 글자는 영문 소문자, _(언더스코어), $ 기호만 사용

9.1 변수와 상수

- 관용적 식별자 명명 규칙
  - 식별자는 영문으로만 작성 **예** name, age
  - 식별자는 의미 있는 단어로 작성 **예** name, age(이름과 나이 저장 시)
- 식별자 표기법
  - 카멜 표기법 : 변수명과 함수명 작성 시 사용
    예 firstName, lastName
  - 언더스코어 표기법 : 상수명 작성 시 사용
    예 FIRST_NAME, last_name
  - 파스칼 표기법 : 생성자 함수명 작성 시 사용
    예 FirstName, LastName

9.2 자료형

1. 문자열

- 큰따옴표나 작은따옴표로 둘러싸인 값
- 문자열에 따옴표가 포함된 경우
  - 문자열에 포함되지 않은 따옴표로 감싸서 정의
  - 문자열 연결 연산자(+) 또는 이스케이프 문자 사용
- 문자열 연결 연산자(+)
- 이스케이프 문자(\, 역슬래시)
- 템플릿 문자열(`, 백틱)

9.2 자료형

2. 숫자형

- 정수, 실수를 포함한 모든 숫자

3. 논리형

- 논리 값(true, false)

4. undefined

- 변수에 아무런 값도 할당되지 않는 상태를 나타내는 값

9.2 자료형

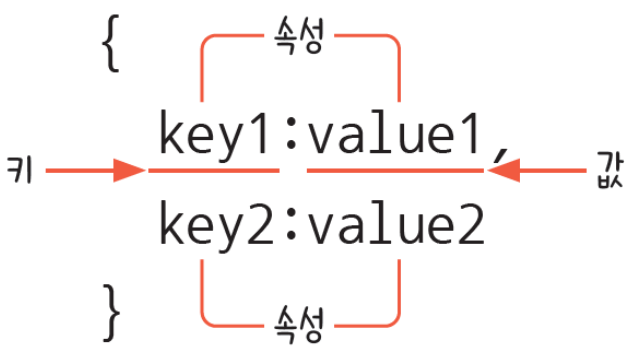
5. null

- 변수를 의도적으로 비워 두기 위해 사용하는 값

6. 객체

- 배열, 함수, 객체 리터럴 등으로 파생되는 상위 자료형
  - 배열(array) : 복수의 데이터를 정의할 수 있는 자료형
  - 객체 리터럴 : 중괄호({})를 사용하며, 키(key)와 값(value)의 한쌍으로 이루어짐

그림 9-3 객체의 구성 요소 명칭



9.3 연산자

1. 산술 연산자

- 이항 산술 연산자 : 연산을 수행하는 데 피연산자가 2개 필요한 연산자
  - x + y x에 y를 더함
  - x – y x에 y를 뺌
  - x * y x에 y를 곱함
  - x / y x를 y로 나눔
  - x % y x를 y로 나누어 나머지를 구함
  - x ** y x의 y 거듭제곱

9.3 연산자

- 단항 산술 연산자 : 연산을 수행하는 데 피연산자가 1개만 필요한 연산자
  - 연산자를 앞에 사용하면 전치 연산, 뒤에 사용하면 후치 연산
  - x++(후치 연산), ++x(전치 연산) : x를 1 증가시킴
  - x—(후치 연산), --x(전치 연산) : x를 1 감소시킴
- 단항 부정 연산자 : 항상 피연산자 앞에 위치하며 피연산자의 부호를 부정하는 연산자
  - -x : x의 부호를 부정(음수 → 양수, 양수 → 음수)

9.3 연산자

2. 대입 연산자와 복합 대입 연산자

- 대입 연산자 : 데이터를 대입(할당)하는 연산을 수행하는 연산자
  - x = y : x에 y를 대입
- 복합 대입 연산자 : 산술 연산자와 대입 연산자를 함께 사용해 산술과 할당을 한 번에 수행하는 연산자
  - x += y : x에 x + y를 대입
  - x -= y : x에 x - y를 대입
  - x *= y : x에 x * y를 대입
  - x /= y : x에 x / y를 대입
  - x %= y : x에 x % y를 대입
  - x **= y : x에 x ** y를 대입

9.3 연산자

3. 비교 연산자

- 피연산자를 비교한 뒤, 논리형 값인 참(true), 거짓(false)을 반환
  - x == y : x와 y의 값이 같으면 true를 반환
  - x === y : x와 y의 값과 자료형이 같으면 true를 반환
  - x != y : x와 y의 값이 다르면 true를 반환
  - x !== y : x와 y의 값과 자료형이 다르면 true를 반환
  - x < y : x가 y보다 작으면 true를 반환
  - x <= y : x가 y보다 작거나 같으면 true를 반환
  - x > y : x가 y보다 크면 true를 반환
  - x >= y : x가 y보다 크거나 같으면 true를 반환

9.3 연산자

4. 논리 연산자

- 연산자를 논리적으로 평가한 뒤, 조건에 맞는 피연산자를 반환
  - x && y → x가 참이면 y를 반환하고, 거짓이면 x를 반환
  - x || y → x가 참이면 x를 반환하고, 거짓이면 y를 반환
  - !x → x가 참이면 false를 반환하고, 거짓이면 true를 반환

5. 삼항 연산자

- 세 항 중 가장 왼쪽에 있는 피연산자의 참, 거짓에 따라 나머지 두 항에 있는 피연산자를 선택적으로 반환
  - x ? y : z → x가 참이면 y를 반환하고, x가 거짓이면 z를 반환

9.3 연산자

6. 연산자 우선순위

- 연산자를 여러 개 사용했을 때 어떤 연산자를 먼저 연산할지를 결정하는 기준
- 가능한 한 우선순위가 가장 높은 그룹 연산자를 사용해 식의 우선순위를 단순하게 정리하는 것이 좋음

그림 9-5 연산자의 우선순위

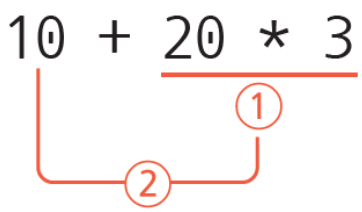


그림 9-6 우선순위 변경

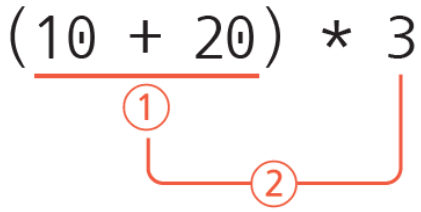
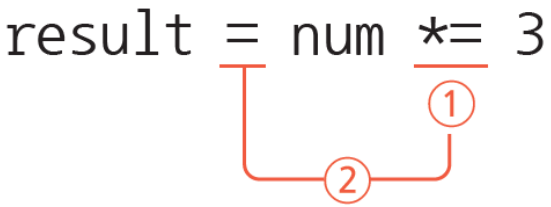


그림 9-7 결합 순서에 의한 연산 순서



9.3 연산자

7. 형 변환

- 데이터의 자료형이 다른 자료형으로 바뀌는 것
- 암시적 형 변환
  - 사용자가 형 변환을 의도하지 않았지만, 자바스크립트에서 자체적으로 형 변환하는 것
  - 개발자가 놓친 부분이라는 의미이므로 암시적 형 변환이 발생하지 않도록 코드에 형 변환을 명확하게 표시하는 것이 좋음
- 명시적 형 변환
  - 드러나게 형 변환을 처리하는 것

9.4 조건문 다루기

1. if, else, else if 문

- if 문 : if 뒤에 오는 소괄호(()) 안의 조건식이 참으로 평가되면 중괄호 안의 코드를 실행하는 조건문

```
형식 if(조건식){
        // 조건식이 참이면 실행
    }
```

- 블록문(block statement) : 한 개 이상의 자바스크립트 코드를 중괄호로 묶은 것. 블록 또는 코드 블록

9.4 조건문 다루기

- else 문 : if 문의 조건식이 거짓일 때 실행되는 블록문

형식

```
if(조건식){
    // 조건식이 참이면 블록문 실행
}else{
    // 조건식이 거짓이면 블록문 실행
}
```

9.4 조건문 다루기

- else if 문 : if 문에 조건을 추가하고 싶을 때 사용하는 블록문

  형식
```
if(조건식1){
    // 조건식1이 참이면 블록문 실행
}else if(조건식2){
    // 조건식2가 참이면 블록문 실행
}else{
    // 조건식이 모두 거짓이면 블록문 실행
}
```

- 분기 처리 : 어떤 조건식을 만족할 때 어떤 블록문을 실행할지 결정하는 것. if 문은 작성하려는 코드의 분기 처리에 따라 중첩해서 사용할 수 있음

9.4 조건문 다루기

2. switch 문

- switch 뒤에 오는 소괄호 안의 값과 일치하는 case 문이 있을 때 해당 코드를 실행하는 조건문

형식

```
switch(key){
    case value1:
      // key가 value1일 때 실행할 블록문
    break;
    case value2:
      // key가 value2일 때 실행할 블록문
    break;
    default:
      // 아무것도 일치하지 않을 때 실행할 블록문
    break;
}
```

9.4 조건문 다루기

3. if 문과 조건식

- if 문은 조건에 식을 사용
- 논리 연산자나 비교 연산자를 식에 이용할 수 있음

4. if 문 vs switch 문

- if 문 : 조건에 식(statement)을 사용, 범위를 이용한 조건을 작성할 때
- switch 문 : 조건에 값(value)을 사용, 값이 하나일 때

9.5 반복문 다루기

- 반복문(loop) : 지정한 조건이 참(true)으로 평가되는 동안 지정한 블록문을 반복해서 실행

1. while 문

- 특정 조건을 만족하는 동안 블록문을 실행

형식
```
while(조건식){
    // 조건식이 참이면 실행
}
```

9.5 반복문 다루기

2. 무한 반복문

- 반복문의 조건이 계속 참으로 평가되어 반복문이 끝나지 않고 무한히 실행되는 것

3. do...while 문

- 특정 조건이 참으로 평가되는 동안 do 다음에 오는 블록문을 반복 실행

형식
```
do{
    // 블록문
}while(조건식);
```

9.5 반복문 다루기

4. for 문

- 지정한 횟수가 끝날 때까지 블록문을 반복 실행하는 반복문

```
형식 for(초깃값; 조건식; 증감식){
        // 블록문
    }
```

- 초깃값 → 조건식 → 블록문(조건식이 참일 경우) → 증감식 → 조건식 순서로 실행
- 중첩해서 사용할 수 있음

9.5 반복문 다루기

5. for 문과 배열

- 배열과 같은 자료형을 반복 횟수 용도로 사용할 수 있음

6. for...in

형식
```
for(가변수 in 배열/객체 리터럴){
    // 블록문
}
```

- 객체 리터럴을 반복할 경우 : 탐색 결과로 가변수에 객체 리터럴의 키가 할당되어 객체 리터럴의 키와 값을 출력할 수 있음
- 배열을 반복할 경우 : 배열의 순서대로 접근하는 것을 보장하지 않으므로 코드를 작성할 때 주의할 것!

9.5 반복문 다루기

7. break 문

- 종료 조건을 만족하지 않아도 인위적으로 반복문을 종료하게 할 때

8. continue 문

- 반복문을 건너뛰게 할 때

10장
자바스크립트 함수 다루기

10.1 함수란

- 함수(function) : 어떤 목적을 가지고 작성한 코드를 모아 둔 블록문
- 함수를 정의한다 : 블록문을 function 키워드, 식별자, 소괄호와 함께 묶어 함수를 생성하는 것
- 함수를 정의하면,
  - 코드를 새로 작성할 필요 없이 정의한 함수를 호출하면 됨
  - .

예

```
function gugudan(){ // 함수 시작
  for(let i = 1; i <= 9; i++){
    console.log(`3 * ${i} = ${3 * i}`);
  }
} // 함수 끝
```

10.2 함수를 정의하는 방법

1. 함수 선언문으로 함수 정의하기

- function 키워드로 함수를 정의하는 방법

  형식 `function 식별자(){}`

2. 함수 표현식으로 함수 정의하기

- 함수도 변수에 할당해 함수를 정의하는 방법
  - 네이밍 함수 : 변수에 할당하는 함수에 식별자가 있을 때
  - 익명 함수 : 변수에 할당하는 함수에 식별자가 없을 때

  형식 `const 변수명 = function(){}; // 익명 함수`

  `const 변수명 = function 식별자(){}; // 네이밍 함수`

10.2 함수를 정의하는 방법

3. 화살표 함수로 함수 정의하기

- ES6에서 추가된 함수 정의 방법
- 화살표를 사용해 함수를 정의하는 방법
- 익명 함수로만 정의할 수 있음

형식 `() => {};`

예

```
const gugudan = () => {
  for(let i = 1; i <= 9; i++){
    console.log(`3 * ${i} = ${3 * i}`);
  }
};
gugudan();
```

10.3 함수 기능 확장하기

1. 매개변수와 인수

- 매개변수 : 함수가 호출될 때 전달받은 데이터를 할당하기 위해 함수에서 선언하는 변수
- 인수 : 정의한 함수를 호출할 때 전달하는 데이터

형식 // 함수 선언문

```
function 함수명(매개변수1, 매개변수2, ..., 매개변수N){}
// 함수 표현식
const 함수명 = function 식별자(매개변수1, ..., 매개변수N){};
// 화살표 함수
const 함수명 = (매개변수1, 매개변수2, ..., 매개변수N) => {};
// 함수 호출
함수명(인수1, 인수2, ..., 인수N);
```

10.3 함수 기능 확장하기

2. 매개변수의 특징

- 데이터 전달
  - 함수를 호출하며 데이터를 전달해도 매개변수를 정의하지 않으면 데이터를 전달받지 못함(단, 오류가 발생하지는 않음)
  - 함수를 호출할 때 전달한 데이터와 매개변수는 일대일 매칭 관계가 형성
- 매개변수의 기본값 : undefined

10.3 함수 기능 확장하기

3. return 문

- 함수 내부에서 함수를 호출한 곳으로 데이터를 전달할 때

형식 return 식(또는 값)

예

```
function sum(num1, num2){
  let result = num1 + num2;
  return result;
}
const result = sum(10, 20);
console.log("out: " + result); // out: 30
```

- 반환한다 : 함수 내부 변수인 result에 할당된 값, 즉 데이터가 sum() 함수를 호출한 곳으로 전달한다
- 반환값 : 반환된 데이터

- 화살표 함수에서 {}를 생략하면 화살표 다음에 오는 코드는 return 문으로 처리됨

10.4 함수의 특징 이해하기

1. 스코프

- 변수나 함수와 같은 참조 대상 식별자를 찾아내기 위한 규칙
- 함수 스코프
  - 함수에서 정의한 블록문만 스코프의 유효 범위로 인정하는 방식
  - 함수 내부는 지역 스코프, 함수 외부는 전역 스코프 영역이 됨
- 블록 스코프
  - {}로 구성된 블록문 기준으로 스코프의 유효 범위를 나누는 방식
  - let과 const 키워드로 선언한 변수에 한해서만 적용

10.4 함수의 특징 이해하기

- 전역 스코프
  - 스코프와 상관없이 모두 참조 가능
  - 전역 변수 : 전역 스코프에 선언한 변수
- 지역 스코프
  - 함수 내부에 선언한 변수 a는 함수 내부에서 참조 가능
  - 지역 변수 : 지역 스코프에 선언한 변수
- 참조 우선순위
  - let, const 키워드는 같은 스코프 영역에서 중복 선언이 불가능
  - 전역 스코프와 지역 스코프에 같은 식별자를 가지는 참조 대상이 있다면,
    ✓ 먼저 같은 지역 스코프의 식별자를 참조
    ✓ 같은 지역 스코프에서 참조할 식별자를 찾지 못할 때만 전역 스코프

10.4 함수의 특징 이해하기

2. 함수 호이스팅

- 호이스팅 : 코드를 선언과 할당으로 나누었을 때, 선언부를 스코프 최상위로 끌어올리는 것
- 호이스팅의 대상
  - 함수 선언문
  - var 키워드를 사용한 함수 표현식
  - 화살표 함수 방식
- let이나 const 키워드로 선언했다면 호이스팅 자체가 되지 않음

10.5 즉시 실행 함수 사용하기

- 전역 스코프가 오염됐다 : 한 번만 사용할 함수인데, 식별자를 더 이상 사용할 수 없게 되었을 때
- 즉시 실행 함수
  - 함수를 정의하면서 동시에 실행까지 하는 함수
  - 한 번 실행되고 나면 메모리에 데이터가 남아 있지 않음
  - 해당 식별자를 한 번도 사용되지 않은 것처럼 인식

형식 `(function(){})();`

예

```
(function init(){
  console.log("initialized!");
})(); // initialized!
init(); // ReferenceError: init is not define
```

11장
자바스크립트 객체 다루기

11.1 객체란

11.2 객체 속성 다루기

11.3 표준 내장 객체 사용하기

11.4 브라우저 객체 모델 사용하기

11.1 객체란

- 객체
  - 키와 값으로 구성된 속성들의 집합을 의미하는 자료형
  - {}를 이용해 생성
  - 키는 문자열로 작성(키에 공백이 들어갈 경우에는 반드시 따옴표 사용)
  - 모든 자료형의 데이터를 값으로 가질 수 있음
- 빈 객체 : 속성이 한 개도 없는 객체

11.2 객체 속성 다루기

1. 객체 속성에 접근하기

- 대괄호 연산자로 접근하기
  - []를 사용해 객체의 속성에 접근하는 방법
  - 배열에서도 사용 가능
  - 객체의 속성에 접근하려면 객체명 뒤에 []를 붙이고 [] 안에 키를 넣음(이때 키는 반드시 문자열 형태로 작성)
- 마침표 연산자로 접근하기
  - .를 이용해 객체 속성에 접근하는 방법
  - 객체 속성에 접근하려면 접근할 객체명과 객체 속성의 키를 마침표 연산자로 연결(이때 키를 큰따옴표나 작은따옴표로 감싸면 오류 발생)
  - 식별자에 공백이 있다면 마침표 연산자는 사용할 수 없음

11.2 객체 속성 다루기

2. 객체 속성 값 변경하기

- 객체로 정의된 값을 바꾸고 싶다면 키로 속성에 접근해서 값을 재할당

3. 객체 속성 동적으로 추가하기

- 속성을 동적으로 추가 : 이미 만들어진 객체에 나중에 속성을 추가하는 것
- 객체 속성에 값을 할당해 접근하면 해당 속성이 존재하는지 확인하고, 없는 속성이면 해당 키와 값으로 구성된 새로운 속성을 객체에 추가

4. 객체 속성 동적으로 삭제하기

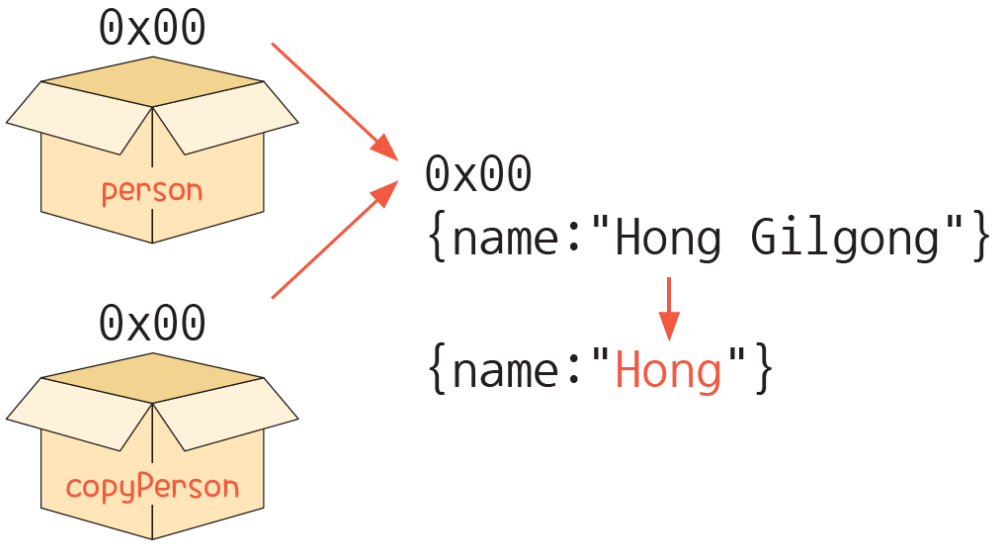
11.2 객체 속성 다루기

5. 객체의 데이터 관리 방법 이해하기

- 깊은 복사
  - 기본 자료형의 데이터 관리 방법
  - 복사한 값을 재할당할 때 한쪽 데이터가 변경되어도 서로 영향을 미치지 않게 복사되는 것

그림 11-7 참조 데이터의 복사

- 얕은 복사
  - 참조 자료형의 데이터 관리 방법
  - 한쪽 데이터가 변경되면 다른 쪽 데이터도 변경되어 서로 영향을 받는 것



11.3 표준 내장 객체 사용하기

- 표준 내장 객체 : 자바스크립트에 기본으로 내장된 객체

1. 문자열을 다루는 String 객체

- 기본 자료형 중 문자열과 관련 있는 속성과 메서드가 정의된 객체

2. 배열을 다루는 Array 객체

- 기본 자료형 중 배열과 관련 있는 속성과 메서드가 정의된 객체
- 파괴적 메서드 : 메서드를 사용했을 때 원본 데이터를 변경하는 메서드
- 비파괴적 메서드 : 원본을 변경하지 않는 메서드

11.3 표준 내장 객체 사용하기

3. 날짜와 시간을 다루는 Date 객체

- 날짜 및 시간과 관련 있는 속성과 메서드가 정의된 객체
- get 메서드 : 날짜와 시간 정보를 가져오는 메서드
- set 메서드 : 날짜와 시간 정보를 설정하는 메서드

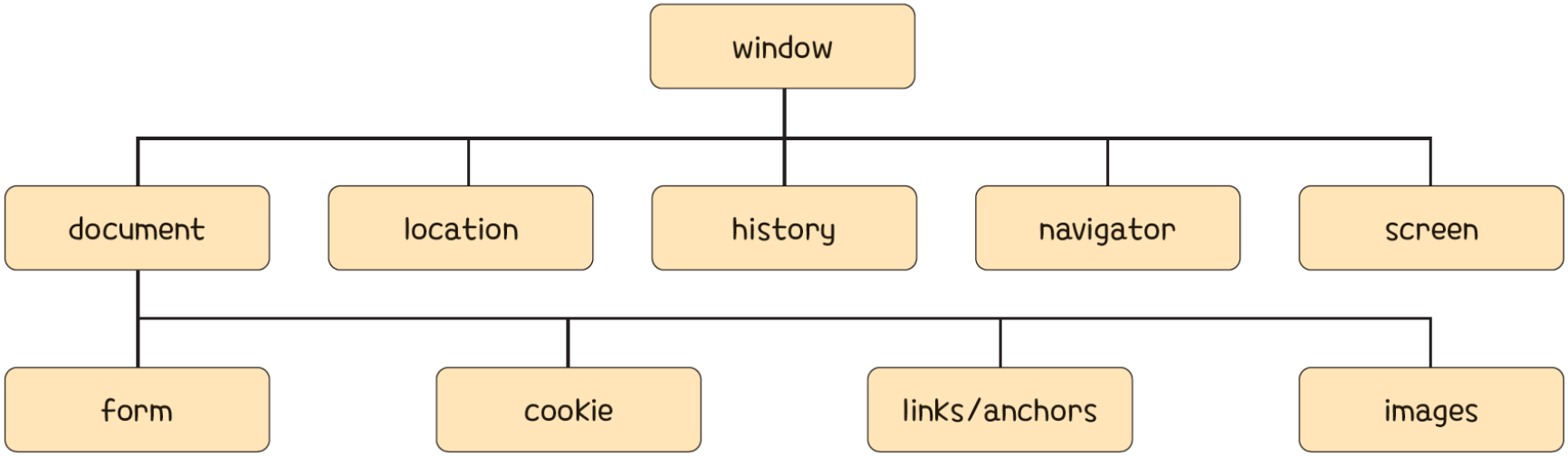
4. 수학 연산을 다루는 Math 객체

- 수학 연산과 관련 있는 속성과 메서드가 정의된 객체
- random() 메서드 : 0 이상 1 미만의 난수를 반환하는 메서드

11.4 브라우저 객체 모델 사용하기

- 브라우저 객체 모델 : 자바스크립트 언어 사양에 포함되지 않고 웹 브라우저에서 제공하는 객체

그림 11-8 브라우저 객체 모델의 계층도



11.4 브라우저 객체 모델 사용하기

- 브라우저 객체 모델의 종류
  - window : 웹 브라우저가 열릴 때마다 생성되는 최상위 관리 객체
  - document : 웹 브라우저에 표시되는 HTML 문서 정보가 포함된 객체
  - location : 웹 브라우저에 현재 표시된 페이지에 대한 URL 정보가 포함된 객체
  - history : 웹 브라우저에 저장된 방문 기록이 포함된 객체
  - navigator : 웹 브라우저 정보가 포함된 객체
  - screen : 웹 브라우저의 화면 정보가 포함된 객체

11.4 브라우저 객체 모델 사용하기

1. window 객체의 속성과 메서드

- 주요 속성(px 단위)
  - innerWidth : 웹 브라우저 화면의 너비
  - innerHeight : 웹 브라우저 화면의 높이
  - outerWidth : 웹 브라우저 창의 너비
  - outerHeight : 웹 브라우저 창의 높이
  - screenTop/screenY : 웹 브라우저 위쪽 면과 모니터의 간격
  - screenLeft/screenX : 웹 브라우저 왼쪽 면과 모니터의 간격
  - pageXOffset/scrollX : 웹 브라우저의 수평 스크롤 위치
  - pageYOffset/scrollY : 웹 브라우저의 수직 스크롤 위치

11.4 브라우저 객체 모델 사용하기

- 주요 메서드
  - alert() : 알림창 표시
  - confirm() : 확인창 표시
  - prompt() : 입력창 표시
  - open() : 새로운 웹 브라우저 창 열기
  - close() : 웹 브라우저 창 닫기
  - setTimeout() : 일정 시간(ms) 뒤에 콜백 함수를 한 번만 실행
  - setInterval() : 일정 시간(ms)마다 콜백 함수를 반복 실행
  - clearInterval : setInterval() 메서드로 반복 실행되는 함수를 중지
  - scrollTo() : 웹 브라우저의 스크롤을 특정 위치만큼 이동
  - scrollBy() : 웹 브라우저의 스크롤을 현재 위치에서 상대 위치로

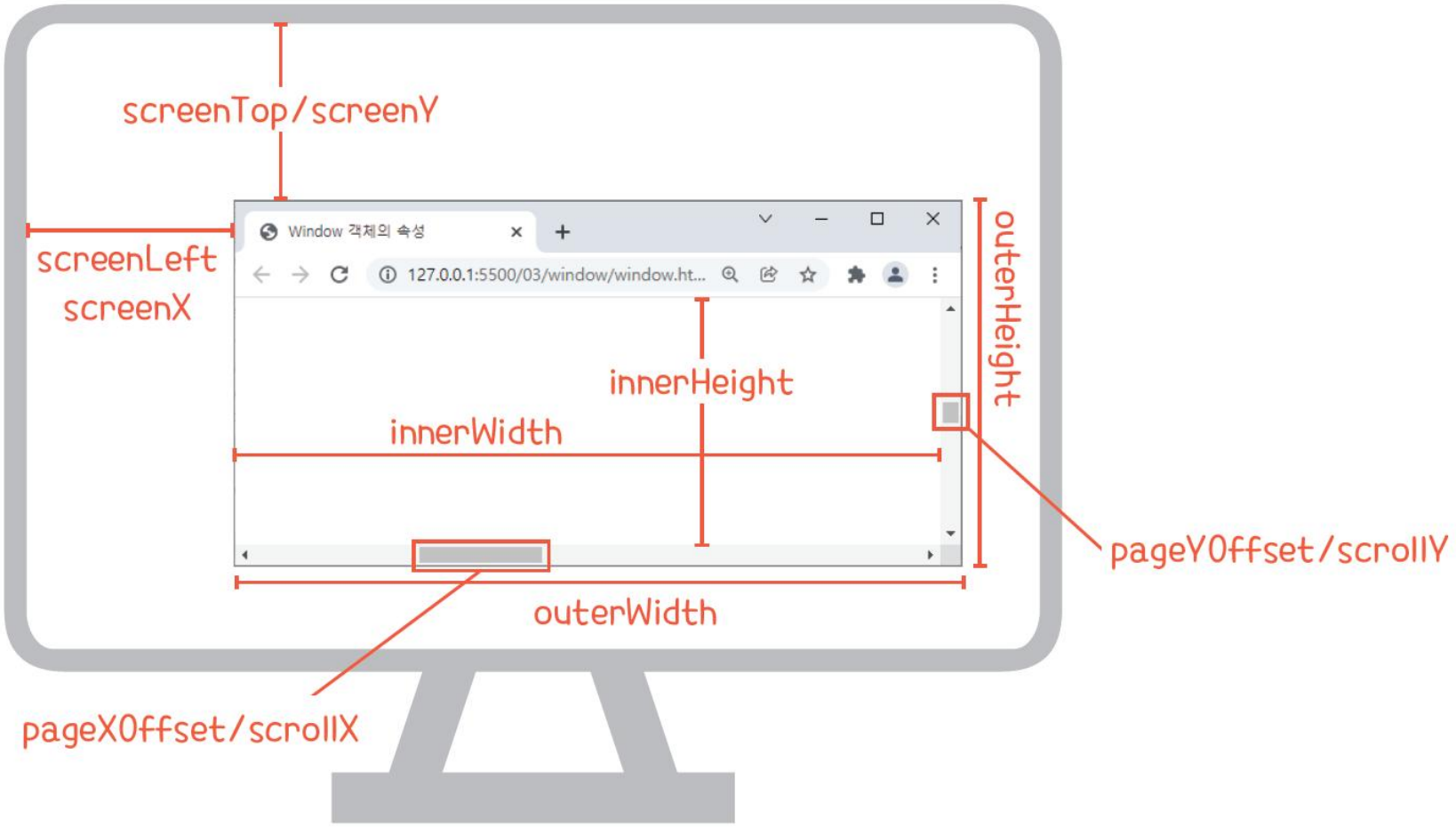
11.4 브라우저 객체 모델 사용하기

2. window 객체의 기본 속성 사용하기

- innerWidth : 웹 브라우저의 너비
- innerHeight : 웹 브라우저의 높이
- outerWidth : 웹 브라우저 창의 너비
- outerHeight : 웹 브라우저 창의 높이
- screenTop/screenY : 웹 브라우저 창 위쪽 면과 모니터 사이의 간격
- screenLeft/screenX : 웹 브라우저 창 왼쪽 면과 모니터 사이의 간격
- scroll : 웹 브라우저 창의 스크롤 가로 위치
- scrollY : 웹 브라우저 창의 스크롤 세로 위치

11.4 브라우저 객체 모델 사용하기

그림 11-9 window 객체 속성



11.4 브라우저 객체 모델 사용하기

3. 웹 브라우저에서 새 창 제어하기

- open() : 웹 브라우저에서 새로운 창을 여는 데 사용
- 팝업창 : window.open() 메서드로 열리는 새 창

  형식 `window.open(경로, 이름, 속성);`

- 창 제어 속성
  - width : 웹 브라우저의 너비를 px 단위로 지정
  - height : 웹 브라우저의 높이를 px 단위로 지정
  - left : 웹 브라우저 왼쪽에서의 위치를 px 단위로 지정
  - top : 웹 브라우저 위쪽에서의 위치를 px 단위로 지정

11.4 브라우저 객체 모델 사용하기

4. 웹 브라우저의 스크롤 이동하기

- scrollTo() : 웹 브라우저의 스크롤 위치를 특정 좌표로 이동
- scrollBy() : 웹 브라우저의 스크롤을 현재 위치에서 상대 위치로 이동

형식 `window.scrollTo(x좌표, y좌표);`

`window.scrollBy(x좌표, y좌표);`

- behavior 속성
  - scrollTo() 메서드나 scrollBy() 메서드의 매개변수에 객체 리터럴을 전달할 때, behavior 속성을 전달할 수 있음
  - smooth 값 : 웹 브라우저 스크롤이 해당 위치로 마우스 휠을 굴리듯이 부드럽게 이동(IE, 사파리 웹 브라우저에서는 지원하

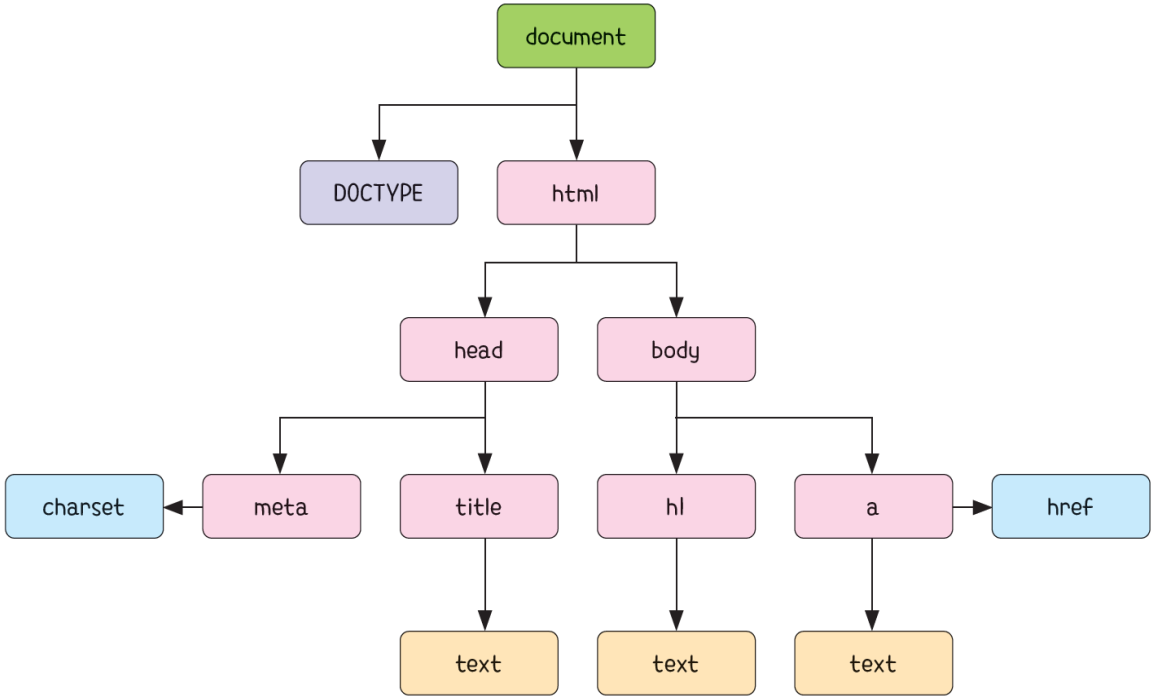
12장
문서 객체 모델과 이벤트 다루기

12.1 문서 객체 모델 이해하기

1. 문서 객체 모델이 생성되는 방식

- 문서 객체 모델(DOM, Document Object Model) : 웹 브라우저에 표시되는 문서를 자바스크립트가 이해할 수 있도록 객체화한 모델 구조
- DOM 트리 :
  문서 객체 모델이
  가지는 트리(tree) 구조

그림 12-2 DOM 트리



12.1 문서 객체 모델 이해하기

2. 노드 타입 살펴보기

- 문서 노드(Node.DOCUMENT_NODE) : 최상위 document 객체의 노드 타입
- 요소 노드(Node.ELEMENT_NODE) : 요소의 노드 타입

  예 h1, p 태그
- 속성 노드(Node.ATTRIBUTE_NODE) : 속성의 노드 타입
  예 href, src 속성
- 텍스트 노드(Node.TEXT_NODE) : 텍스트에 해당하는 노드 타입
- 주석 노드(Node.COMMENT_NODE) : 주석에 해당하는 노드 타입

12.2 노드 선택하기

1. 속성으로 노드 선택하기

- 모든 요소 탐색 속성
  - parentNode : 부모 노드를 반환
  - childNodes : 모든 자식 노드를 반환
  - firstChild : 첫 번째 자식 노드를 반환
  - lastChild : 마지막 자식 노드를 반환
  - previousSibling : 이전 형제 노드를 반환
  - nextSibling : 다음 형제 노드를 반환

12.2 노드 선택하기

- 요소 노드 탐색 속성
  - parentElement : 부모 요소 노드를 반환
  - children : 자식 요소 노드를 반환
  - firstElementChild : 첫 번째 자식 요소 노드를 반환
  - lastElementChild : 마지막 자식 요소 노드를 반환
  - previousElementSibling : 이전 요소 노드를 반환
  - nextElementSibling : 다음 요소 노드를 반환

12.2 노드 선택하기

2. 메서드로 노드 선택하기

- 속성값과 태그명 사용하기 - get 메서드
  - getElementById(<id 속성값>) : id 속성값과 일치하는 요소 노드를 1개만 선택
  - getElementsByClassName(<class 속성값>) : class 속성값과 일치하는 요소 노드를 모두 선택
  - getElemenetsByTagName(<태그명>) : 태그명과 일치하는 요소 노드를 모두 선택
- CSS 선택자 사용하기 - query 메서드
  - querySelector(<CSS 선택자>) : 매개변수로 넘어오는 CSS 선택자에 해당하는 노드를 1개만 선택
  - querySelectorAll(<CSS 선택자>) : 매개변수로 넘어오는 CSS 선택자에 해당하는 노드를 모두 선택

12.3 노드 조작하기

1. 콘텐츠 조작하기

- textContent : 노드 요소의 모든 텍스트에 접근
- innerText : 노드 요소의 텍스트 중 웹 브라우저에 표시되는 텍스트에만 접근
- innerHTML : 노드 요소의 텍스트 중 HTML 태그를 포함한 텍스트에만 접근

2. 스타일 조작하기

형식 `<노드>.style.<css 속성명> = <속성값>;`

12.3 노드 조작하기

3. 클래스 속성 조작하기

형식 <노드>.classList.add("class 속성값"); // 추가

<노드>.classList.remove("class 속성값"); // 삭제

<노드>.classList.toggle("class 속성값"); // 추가와 삭제 반복

4. 데이터 속성 조작하기

- data-* 속성
  - 사용자 정의(custom) 속성
  - dataset 속성을 사용해 조작(dataset 속성은 HTML 문서에서 data-* 속성을 가져오거나 지정)

12.3 노드 조작하기

5. 메서드로 속성 조작하기

형식 <노드>.getAttribute("속성명"); // 속성값 가져오기

<노드>.setAttribute("속성명", "속성값"); // 속성값 설정

<노드>.removeAttribute("속성명"); // 속성 삭제

12.4 노드 추가/삭제하기

1. 노드 추가하기

- 노드 생성
  - createElement() : 요소 노드를 생성
  - createTextNode() : 텍스트 노드를 생성
  - createAttribute() : 속성 노드를 생성
- 노드 연결
  - <기준 노드>.appendChild(<자식 노드>) : 기준 노드에 자식 노드를 연결
  - <기준 노드>.setAttributeNode(<속성 노드>) : 기준 노드에 속성 노드를 연결

12.4 노드 추가/삭제하기

그림 12-15 예제 코드의 DOM 트리

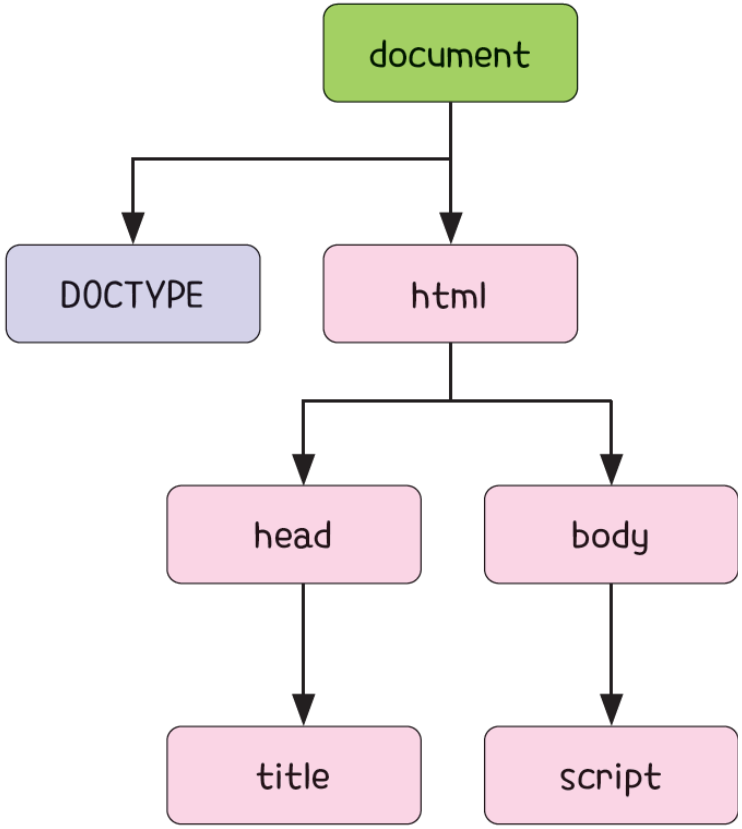
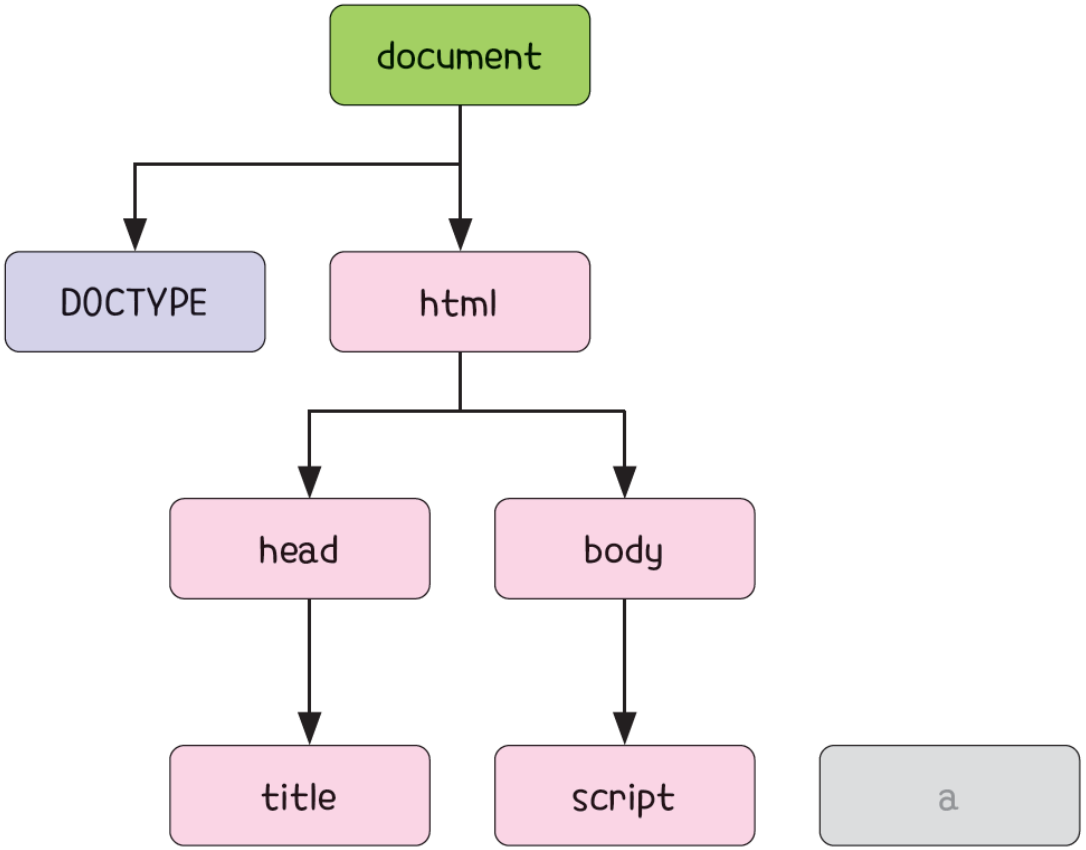


그림 12-16 a 요소 노드 생성



12.4 노드 추가/삭제하기

그림 12-17 a 요소 노드 연결

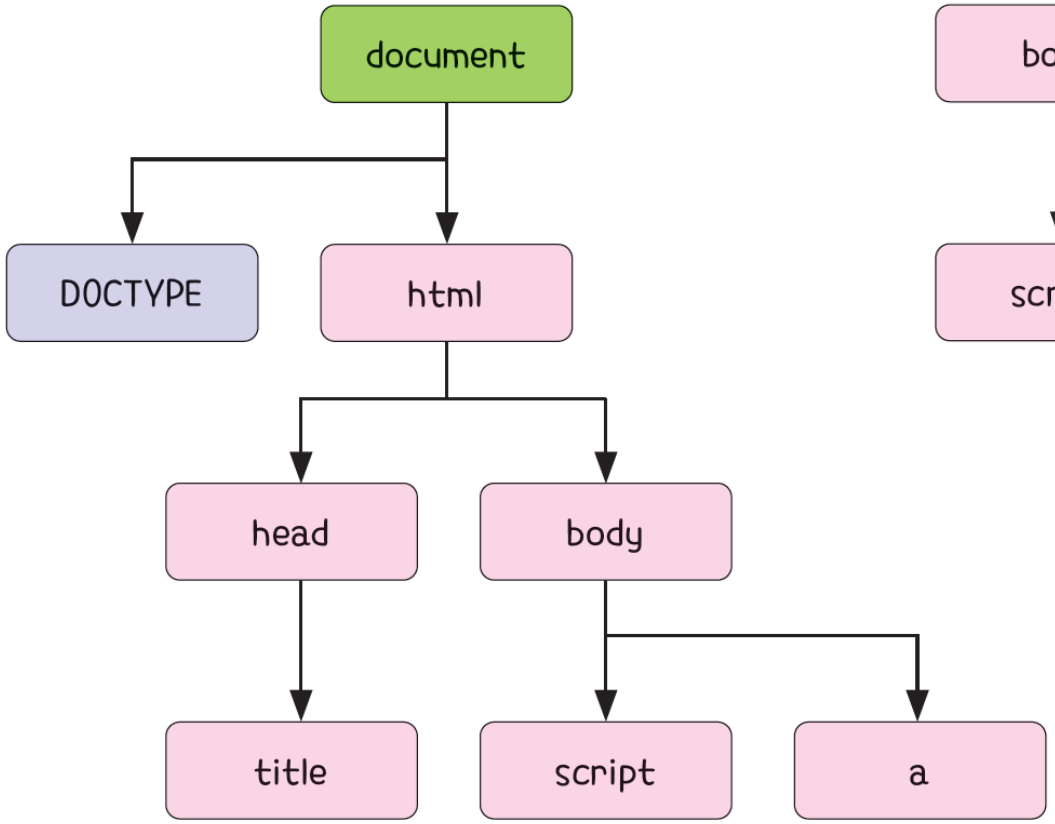
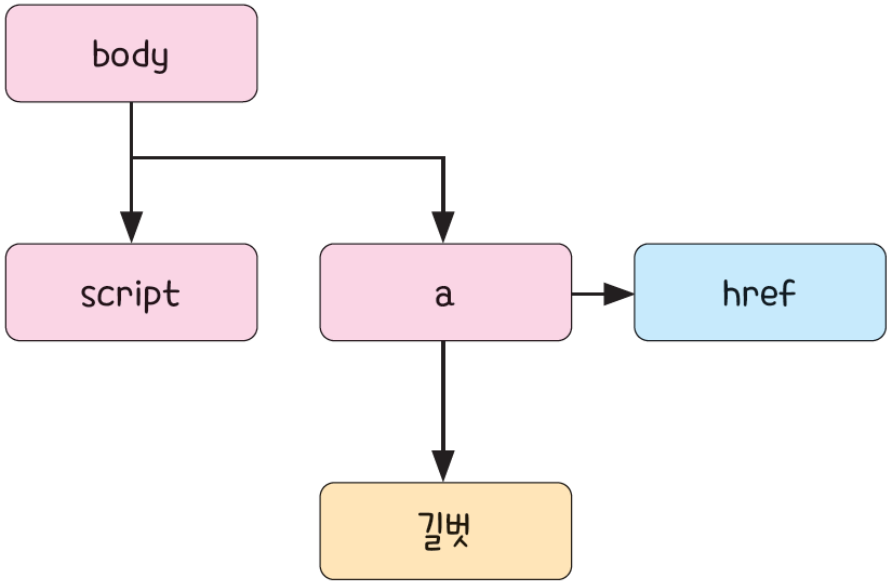


그림 12-21 속성 노드 연결



12.4 노드 추가/삭제하기

2. 노드 삭제하기

- 부모 노드에 연결된 자식 노드를 삭제

  형식 <부모 노드>.removeChild(<자식 노드>)

12.5 폼 조작하기

1. form 태그 선택하기

- forms 속성 사용하기
  - 모든 form 태그를 참조하는 HTMLCollection 객체가 반환
  - 반환된 HTMLCollection 객체를 사용해 화면에 있는 form 요소 노드를 선택
- name 속성 사용하기
  - form 태그의 name 속성값으로 form 요소 노드를 선택

12.5 폼 조작하기

2. 폼 요소 선택하기

- elements 속성을 사용하는 방법
  - elements 속성 : form 요소 노드의 하위 노드 중 폼 요소를 반환
  - 반환된 객체를 사용해 개별 폼 요소 노드에 인덱스로 접근
- name 속성을 사용하는 방법
  - 폼 요소의 name 속성값으로 폼 요소 노드에 바로 접근

12.5 폼 조작하기

3. 폼 요소의 입력값 다루기

- 한 줄 입력 요소 / 여러 줄 입력 요소
  - value 속성 참조 : 입력한 값을 가져오기
  - value 속성에 값 할당 : 새로운 값을 설정
- 체크박스 요소 / 라디오버튼 요소
  - checked 속성 : 체크 또는 선택 상태를 확인
  - checked 속성에 true 할당 : 체크 또는 선택 상태를 기본으로 설정

12.5 폼 조작하기

- 콤보박스 요소
  - selected 속성 : 항목의 선택 상태를 확인
  - selected 속성에 true 할당 : 항목을 선택 상태로 설정
- 파일 업로드 요소
  - FileList 객체 : 요소의 값 가져오기
- 폼 요소 관련 기타 메서드
  - submit() : 폼 요소의 값 전송(submit)
  - focus() : 폼 요소에 포커스(커서) 이동

12.6 이벤트 다루기

1. 이벤트 종류

- 마우스 이벤트
  - onclick : 마우스로 클릭하면 발생
  - ondblclick : 마우스로 두 번 빠르게 클릭하면 발생
  - onmouseover : 마우스 포인터를 올리면 발생
  - onmouseout : 마우스 포인터가 빠져나가면 발생
  - onmousemove : 마우스 포인터가 움직이면 발생
  - onwheel : 마우스 휠(wheel)을 움직이면 발생

12.6 이벤트 다루기

- 키보드 이벤트
  - onkeypress : 키보드 버튼을 누르고 있는 동안 발생
  - onkeydown : 키보드 버튼을 누른 순간 발생
  - onkeyup : 키보드 버튼을 눌렀다가 뗀 순간 발생
- 포커스 이벤트
  - onfocus : 요소에 포커스가 되면 발생
  - onblur : 요소가 포커스를 잃으면 발생
- 폼 이벤트
  - onsubmit : 폼이 전송될 때 발생
- 리소스 이벤트
  - onload : 웹 브라우저의 리소스 로드가 끝나면 발생

12.6 이벤트 다루기

2. 이벤트 등록하기

- 이벤트 등록 : 이벤트가 발생할 때 어떤 작업을 할지 자바스크립트 코드로 작성하는 것
- 이벤트 등록 방법
  - 인라인 : HTML 태그에 속성으로 이벤트 등록
  - 프로퍼티 리스너 : 요소 노드에 직접 속성으로 이벤트 등록
  - 이벤트 등록 메서드 : addEventListener() 메서드로 이벤트 등록

12.7 이벤트 객체와 this

1. 이벤트 객체 사용하기

- 이벤트 객체 : 이벤트 함수가 호출될 때 내부적으로 전달되는 이벤트 정보가 담긴 객체
- PointerEvent 객체의 주요 속성
  - clientX : 마우스가 클릭된 x좌표(수평 스크롤 포함 X)
  - clientY : 마우스가 클릭된 y좌표(수직 스크롤 포함 X)
  - pageX : 마우스가 클릭된 x좌표(수평 스크롤 포함 O)
  - pageY : 마우스가 클릭된 y좌표(수직 스크롤 포함 O)
  - screenX : 모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭된 x좌표
  - screenY : 모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭

12.7 이벤트 객체와 this

- KeyboardEvent 객체의 주요 속성
  - keyCode : 키보드에서 눌린 키의 유니코드 값 반환
  - ctrlKey : Ctrl 키가 눌렸으면 true, 그렇지 않으면 false 반환
  - altKey : Alt 키가 눌렸으면 true, 그렇지 않으면 false 반환
  - shiftKey : Shift 키가 눌렸으면 true, 그렇지 않으면 false 반환

12.7 이벤트 객체와 this

2. 이벤트 취소하기

- preventDefault() 메서드 : 태그에 적용된 기본 이벤트를 취소

3. this 키워드 사용하기

- this 키워드 : 이벤트를 발생시킨 요소 노드를 가리킴
- 화살표 함수일 때는 이벤트 객체의 target 속성으로 참조해야 함