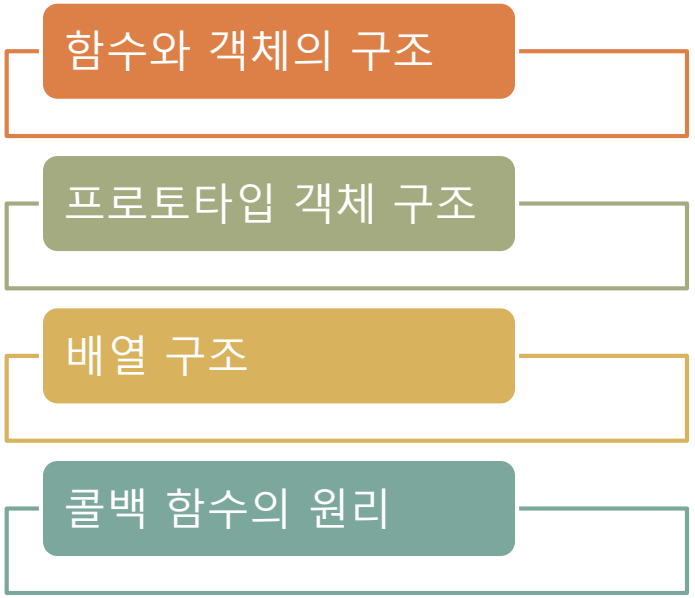
제2장 웹 클라이언트 UI 설계(4)

1. 웹 기획 설계
2. HTML5 핵심
3. CSS3 핵심
4. **자바스크립트 핵심**

자바스크립트 핵심기술

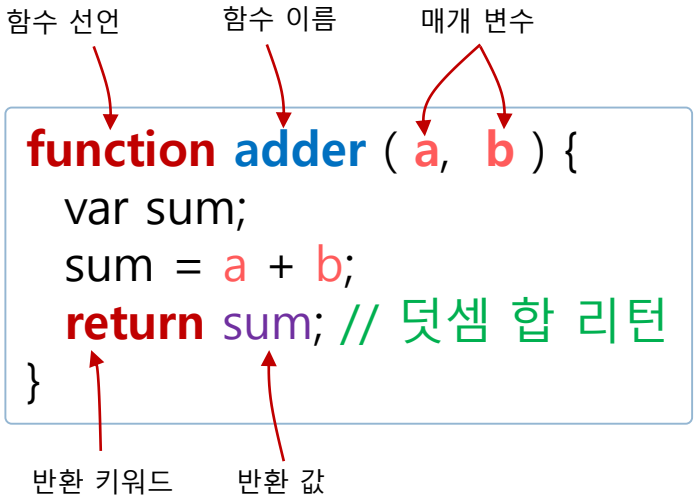
- 함수와 객체의 구조 및 사용법
  - 사용자 **함수** 정의 방법
  - 사용자 **객체** 만들기
  - **프로토타입** 객체 구조 및 사용법
  - DOM 객체의 구성과 사용법 및 동적 생성 방법
- **배열** 구조 및 사용법
  - 배열 생성 및 사용법
- **콜백** 함수의 원리



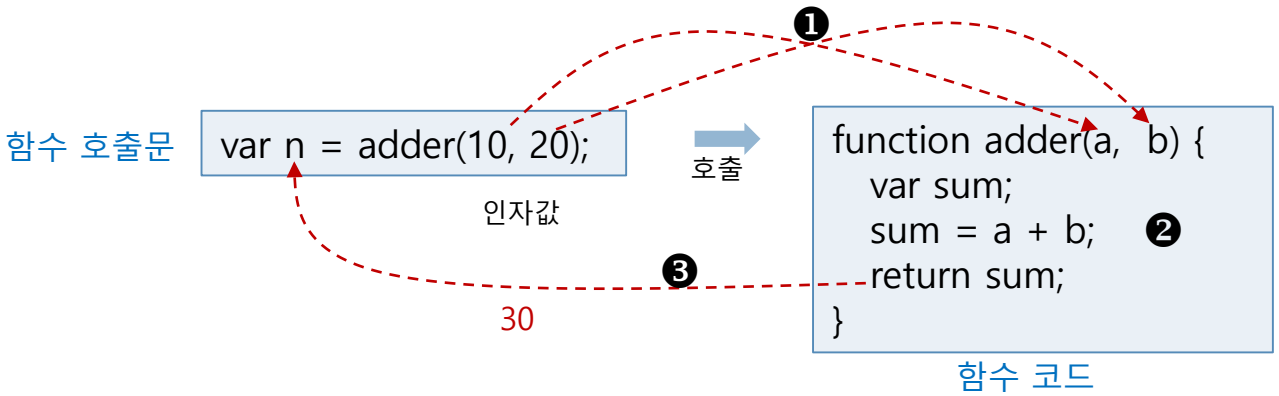
함수의 구성과 사용법

- 함수의 정의

```
function 함수이름(arg1, arg2,..., argn) {
    ...프로그램 코드...
    결과를 리턴하는 return 문
}
```

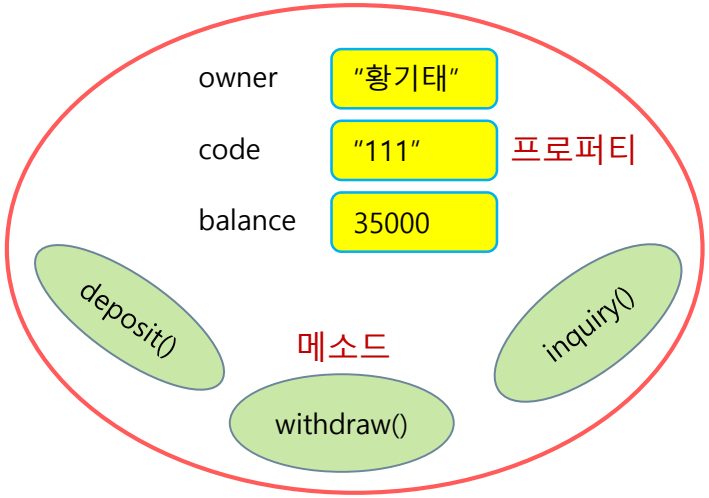


- 함수 호출
  - 함수의 코드 실행 요청

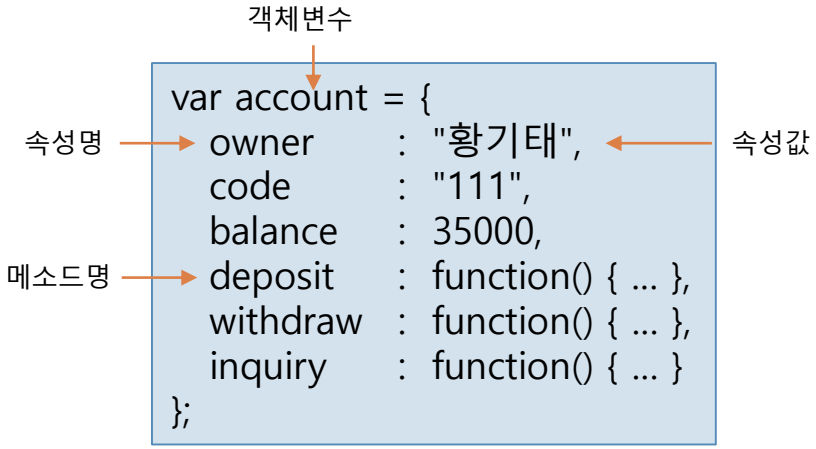


객체의 구성과 사용법

- 자바스크립트 객체 구성
  - 여러 개의 프로퍼티(property)와 메소드로 구성
    - 프로퍼티 : 객체의 고유한 속성(변수)
    - 메소드(method) : 함수



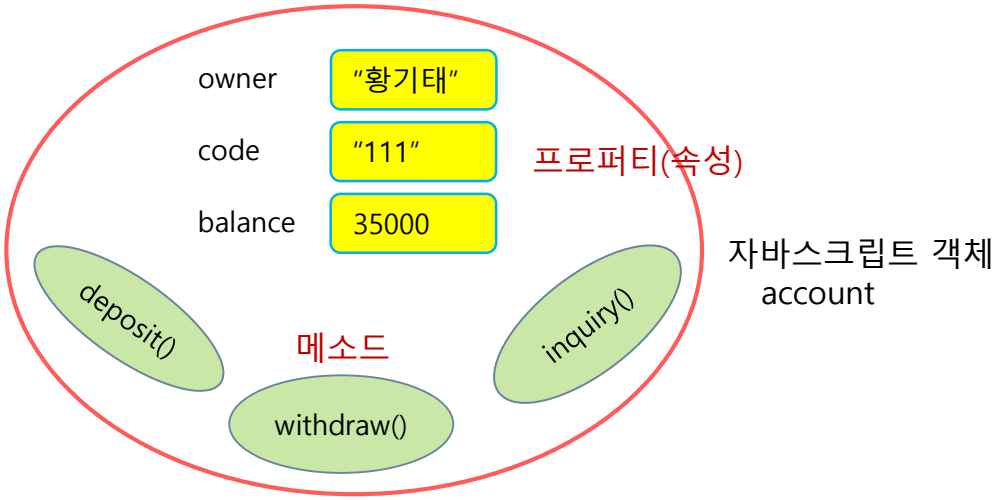
자바스크립트 객체 account



account 객체를 만드는 자바스크립트 코드

사용자 객체 만들기

- 사용자가 새로운 타입의 객체 작성 가능 : 3 가지 방법
  1. 직접 객체 만들기
     - **new Object() 이용**
     - 리터럴 표기법 이용
  2. 객체의 틀(**프로토타입**)을 만들고 객체 생성하기
- 샘플
  - 은행 계좌를 표현하는 account 객체

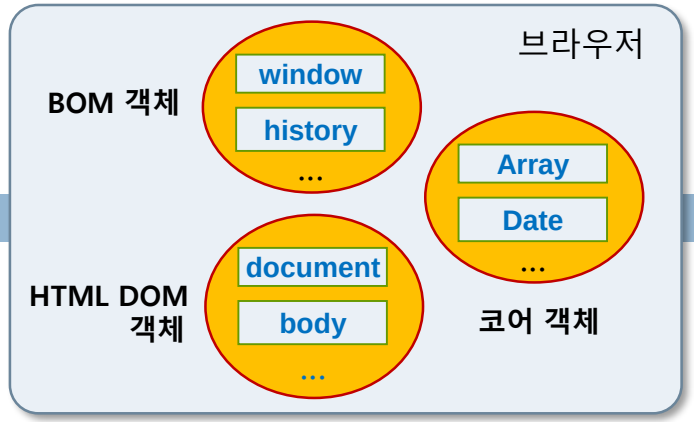


new Object()로 객체 만들기

1. new Object()로 빈 객체 생성
2. 빈 객체에 프로퍼티 추가
   - 새로운 프로퍼티 추가(프로퍼티 이름과 초기값 지정)
3. 빈 객체에 메소드 추가
   - 메소드로 사용할 함수 미리 작성("생성자 함수"라는 용어 사용)
   - 새 메소드 추가(메소드 이름에 함수 지정)

```
var account  = new Object();    // new 키워드에 의해 객체 생성
account.owner = "황기태";       // 계좌 주인 프로퍼티 생성 및 초기화
account.code = "111";           // 코드 프로퍼티 생성 및 초기화
account.balance = 35000;        // 잔액 프로퍼티 생성 및 초기화
account.inquiry = inquiry;      // 프로토타입에 메소드 선언
account.deposit = deposit;      // 프로토타입에 메소드 선언
account.withdraw = withdraw; // 프로토타입에 메소드 선언
```

DOM 객체의 구성 요소



- DOM 객체는 5 개의 요소 구성
  - 프로퍼티(property)
    - HTML 태그의 속성(attribute) 반영
  - 메소드(method)
    - DOM 객체의 멤버 함수로서, HTML 태그 제어 가능
  - 컬렉션(collection)
    - 자식 DOM 객체들의 주소를 가지는 등 배열과 비슷한 집합적 정보
  - 이벤트 리스너(event listener)
    - HTML 태그에 작성된 이벤트 리스너 반영
    - 약 70여 개의 이벤트 리스너를 가질 수 있음
  - CSS3 스타일
    - HTML 태그에 설정된 CSS3 스타일 시트 정보를 반영
    - DOM 객체의 style 프로퍼티를 통해 HTML 태그의 모양 제어 가능

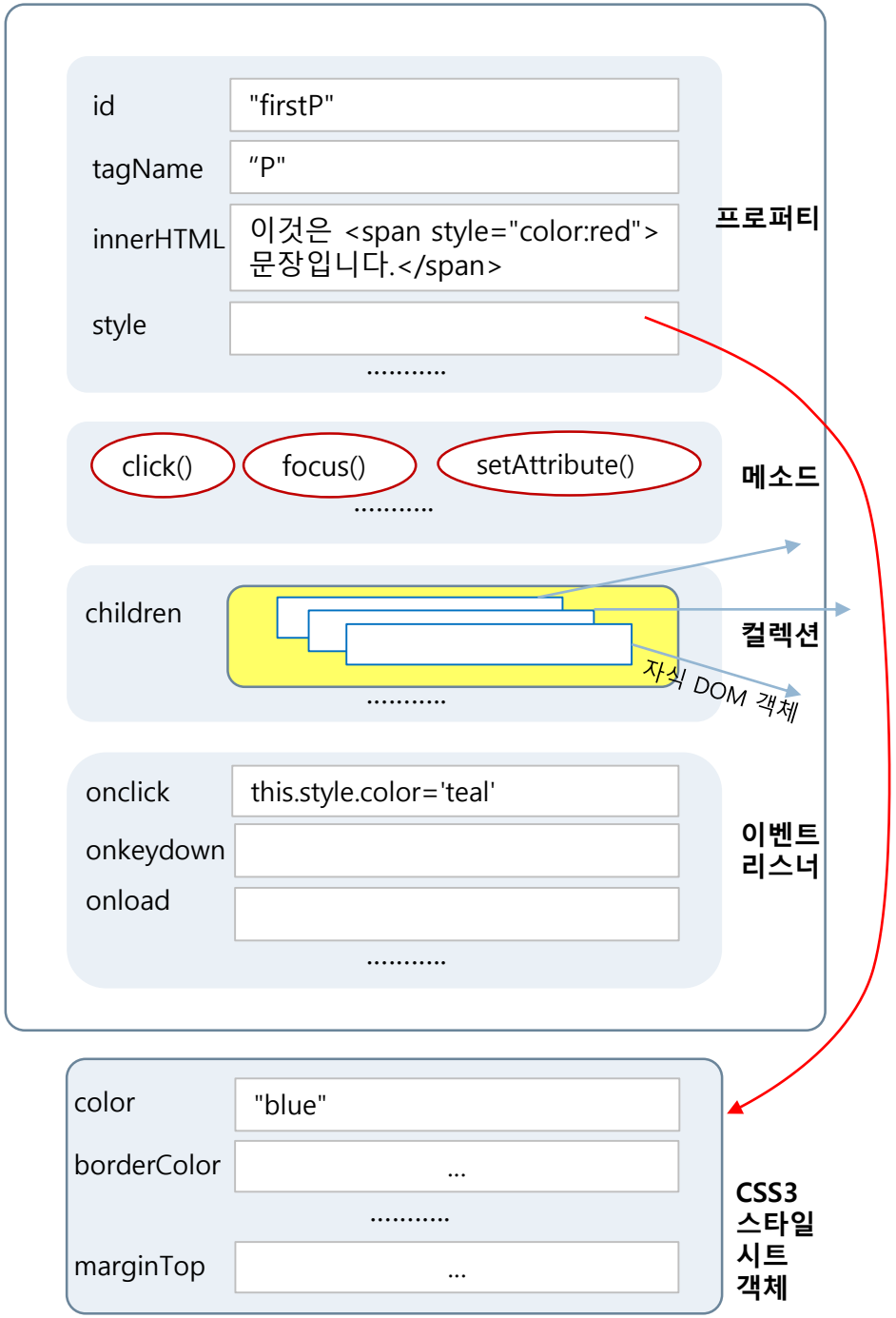
DOM 객체의 구성

- **프로퍼티(property)**
- **메소드(method)**
- **컬렉션(collection)**
- **이벤트 리스너(event listener)**
- **CSS3 스타일**

```
<p id="firstP"
    style="color:blue"
    onclick="this.style.color='teal'">
   이것은
   <span style="color:red">
          문장입니다.
   </span>
</p>
```

** ... 태그**

DOM 객체 p



DOM 객체 사용법

- **DOM 객체 구분**, id 태그 속성

```
<p id="firstP">안녕하세요</p>
```

document.getElementById()
document.getElementsByTagName()
document.getElementsByClassName()

- **DOM 객체 찾기**, document.getElementById()

```
var p = document.getElementById("firstP");  // id 값이 firstP인 DOM 객체 리턴
p.style.color = "red";                      // p 객체의 글자 색을 red로 변경
```

- **DOM 객체의 CSS3 스타일 동적 변경**
  - CSS3 스타일 프로퍼티는 다음과 같이 사용
    - background-color 스타일 프로퍼티 -> backgroundColor
    - font-size 스타일 프로퍼티 -> fontSize

```
<span id="mySpan" style="color:red">문장입니다.</span>
```

```
var span = document.getElementById("mySpan");  // id가 mySpan인 객체 찾기
span.style.color = "green";                    // '문장입니다'의 글자 색을 green으로 변경
span.style.fontSize = "30px";                  // '문장입니다'의 폰트를 30px 크기로 변경
span.style.border = "3px dotted magenta"; // 3픽셀의 magenta 점선 테두리
```

DOM 객체 사용법

- 태그 이름으로 찾기
  - **document.getElementsByTagName()**
    - 태그 이름이 같은 모든 DOM 객체들을 찾아 컬렉션 리턴
    - 예) 태그의 모든 DOM 객체 찾기

```
var divTags = document.getElementsByTagName("div");
```

```
var n = divTags.length; // 웹 페이지에 있는 <div> 태그의 개수
```

- class 속성으로 찾기
  - **document.getElementsByClassName()**
    - class 속성이 같은 모든 DOM 객체들을 찾아 컬렉션 리턴
    - 예)

```
<div class="plain">...</div>
<div class="important">...</div>
<div class="plain">...</div>
```

```
var plainClasses = document.getElementsByClassName("plain");
var n = plainClasses.length; // 웹 페이지에 class="plain" 속성을 가진 태그의 개수
```

DOM 객체 동적 생성 및 삽입과 삭제

- DOM 객체 동적 생성: **document.createElement("태그이름")**
  - 태그이름의 DOM 객체 생성
    - 예)

```
var newDIV = document.createElement("div");
```

```
newDIV.innerHTML = "새로 생성된 DIV입니다.";
```

```
newDIV.setAttribute("id", "myDiv");
newDIV.style.backgroundColor = "yellow";
```

- DOM 트리에 삽입
  - **부모.appendChild(DOM객체);**
  - 부모.insertBefore(DOM객체 [, 기준자식]);
    - 예) 생성한 태그를 <p "id=p"> 태그의 마지막 자식으로 추가

```
var p = document.getElementById("p");
p.appendChild(newDiv);
```

- DOM 객체의 삭제
  - var removedObj = **부모.removeChild(떼어내고자하는자식객체);**
    - 예)

```
var myDiv = document.getElementById("myDiv");
var parent = myDiv.parentElement;
parent.removeChild(myDiv);     // 부모에서 myDiv 객체 삭제
```

배열

- **리터럴 배열 생성**

  var 배열명 = [원소1,....., 원소n]; // var data = [10, "hi", 20.5];

- **배열 생성**

  var data=[ ]; //배열 선언 후 데이터 입력

  data[0] = 10;

  data[1] = "hi";

  data[2] = 20.5;

- **배열 객체로 생성**

  var 배열명 = new Array(원소1,....., 원소n);

- **연관 배열 생성**

  var 배열명 = { 키1:값1, ... , 키n:갑n};

  var data={"k1":10, "k2":"hi", "k3":20.5};

  data["k4"] = true;

  data.k5 = 300;

콜백 함수(Callback Function)

- 콜백 함수 : 함수의 매개변수로 전달된 함수를 다른 함수의 내부에서 호출하는 방식

```
function callTest(eprint){
        for(var i=1; i<=10; i++){
                eprint();
        }
}
callTest(function() {
        document.write("함수 호출");
});
```

- 비동기 입출력 방식(논블로킹입출력, Non-Blocking IO)
  - 하나의 요청 처리가 끝날 때까지 기다리지 않고 다른 요청을 동시에 처리하는 방식