[01-29] JS - 함수, 객체

1) 함수(function) - 교안 31page ~ , JS05

함수(function)는 아래와 같이 여러 가지 동작을 연결하여 하나의 작업 단위로 실행될 수 있도록 묶어 놓은 소스 코드의 덩어리 이다. 함수는 중괄호("{}")를 사용해 함수의 영역을 지정하고 그 안에 필요한 여러 가지 코드를 작성하는데 이 함수가 다른 곳에서 사용(호출)될 때 중괄호 안에 작성한 소스 코드가 위에서부터 순차적으로 실행된다.

익명 함수(이름 없는) 정의 : 함수 리터럴 표현으로 정의, 이름 없는 함수를 정의할 때 아래와 같이 변수에 함수를 할당해 놓고 다른 곳에서 이 변수 이름으로 함수를 여러 번 호출 할 수 있다.

- function(){ } 로 시작할수 있다. 그러나 이대로 사용하기는 어려우므로
- 변수 let docPrint = function(){ } 형태로 변수에 할당한다.
- 다른 곳에서는 변수명을 통해 함수를 원하는 만큼 호출하는 것

함수를 정의할 때 함수의 괄호는 그 함수가 실행되면서 필요한 데이터를 외부로부터 받을 수 있는 통로이다. 외부에서 데이터를 받을 필요가 없으면 괄호 안을 비워서 빈 괄호로 작성하면 되고 함수 실행에 필요한 데이터를 외부로부터 받으려면 아래와 같이 괄호 안에 매개변수를 만들면 된다.

함수를 정의할 때 괄호 안의 변수는 함수를 실행하기 위해서 외부로 부터 입력 받는 값으로 함수 외부에서 함수 안쪽으로 값을 전달하여 사용할 수 있도록 연결해 주는 매개체라는 의미로 매개변수(parameter)라고 부른다.

매개변수가 있는 함수를 호출할 때 아래와 같이 함수의 괄호 안에 넣어주는 값을 함수 안으로 전달하는 실제 값이라는 의미로 전달인수 또는 전달인자라고 하며 간단히 인수(인자, argument)라고 부른다.

매개변수와 인수를 정리하자면 매개변수는 함수를 정의할 때 함수의 괄호안에 선언하는 변수를 의미하며 함수를 호출할 때 지정하는 값은 인수라고 구분하면 된다. 참고로 자바스크립트에서는 인수를 지정하지 않고 함수를 호출할 수 있지만 함수 안의 코드가 실행되면서 오류가 발생할 수도 있다. 또한 오류는 발생하지 않더라도 함수로부터 원하는 기능을 얻을 수 없기 때문에 함수를 제대로 활용하려면 함수를 호출할 때 인수를 정확하게 지정해야 한다.

- let rectangle = function(width, height) { //width, height 가 매개변수
  return width * height;
  }
- document.write('사각형의 넓이 : ' + rectangle(10, 20) + '
'); //(10,20)이 인수

변수의 유효 범위(scope) : script 영역에 선언된 변수는 전역, 함수의 매개변수는 함수안에서 선언되었기 때문에 지역 변수가 된다.

화살표 함수(익명함수)

가변인수 : 매개변수의 개수를 정확히 맞추기 위해 arguments 객체를 사용했다. 가변인수는 함수안에서만 쓸 수 있다.

2) 객체(object) - 교안 45p, JS06

실세계의 특정 대상의 속성(property)과 기능(Method)을 정의해 추상화(Abstration)해 놓는 것, 함수와 메서드의 혼용해서 사용하는 편//함수안에 메서드가 메서드안에 함수가 가능하기 때문

객체지향 프로그래밍 : 객체들 간의 연결, 유기적인 상호작용으로 하나의 완성된 프로그램을 만드는 것
자바스크립트에는 내장객체, BOM, DOM, 사용자 정의 객체등이 있다. 사용자 정의객체는

- 객체 리터럴로 객체 생성 및 속성과 기능을 추가하거나
- 생성자함수(익명 함수, 생성자함수)를 정의, 이 함수를 통해 객체를 생성
- 클래스를 정의해 객체 생성(ES6이후)으로 총 3가지가 있다.

2-1) 객체 리터럴 : 객체가 생성된 것

객체 자체는 변함이 없으므로 const 로 주로 선언한다.
객체는 데이터(변수, Property)와 기능(함수, Method)으로 구성되며 표현된다.
다시 말해 객체는 속성과 메서드를 묶어서 하나의 데이터로 만든 것이다.

```
const Student = {
 이름: '홍길동',
 학년: 3,
 toString: function() {
  return '<p>'
       + '이름 : ' + this.이름 + '<br/>'
       + '학년 : ' + this.학년 + '<br/>'
       + '</p>';
 }};
```

2-2) 익명 함수방식으로 객체 정의 : 객체가 정의만 된것이지 생성된 것은 아님

```
const Member = function(name, id, address) {
 this.name = name;
 this.id = id;
 this.address = address;
 this.toString = function() {
  return '<p>'
       + '이 름 : ' + this.name + '<br/>'
       + '아이디 : ' + this.id + '<br/>'
       + '주 소 : ' + this.address + '<br/>'
       + '</p>';
};};
```

2-3) 생성자 함수방식으로 객체 정의 : 객체가 정의만 된것이지 생성된 것은 아님

```
function Human(name, age, gender) {
  this.name = name;
  this.age = age;
  this.gender = gender;
  this.toString = function() {
    return '<p>'
         + this.name + '(' + this.gender + '-' + this.age + ')<br/>'
        + '</p>';
  };}
```

생성자 함수와 익명 함수 방식으로 객체를 정의하면 아래와 같이 new 연산자를 사용해 서로 다른 속성을 가지는 객체를 무한정 생성할 수 있다. new 연산자를 사용해 객체를 생성하는 것을 "인스턴스를 생성한다." 라고 말 한다.

```
let member = new Member("홍길동", "midas", "서울 구로구"); // 객체가 생성되며, 인스턴스 생성된 것
let human = new Human("김유신", 25, "남성");

// 아래와 같이 출력 함수에서 참조 변수를 사용하면 toString()이 자동으로 호출된다.
document.write("<h2>객체를 정의하기</h2>");
document.write(Student);
document.write(member);
document.write(human);
```

- document.write(human[“age”]);
- 위 형태도 가능하다. 25가 출력된다.

2-4) 클래스로 객체 만들기

자바스크립트에서 사용하는 클래스는 기존의 생성자 함수 방식을 조금 더 쉽게 표현할 수 있도록 변경한 것으로 완전한 클래스 방식은 아니다. 생성자의 역할 : 객체의 초기화

```
 class Member {
//클래스 안에서 생성자와 메서드를 분리해 작성한다. 생성자는 클래스로 인스턴스를 생성할 때 호출하는 함수이다. 생성자를 통해서 인스턴스가 가지는 속성을 초기화 할 수 있다.
   constructor(name, id, address) {
     this.name = name;
     this.id = id;
     this.address = address;
  }

 // 메소드는 다음과 같이 메소드 이름만 정의해 내용을 작성하면 된다.
   toString() {
     return '<p>'
         + '이 름 : ' + this.name + '<br/>'
```

```
      + '아이디 : ' + this.id + '<br/>'
      + '주 소 : ' + this.address + '<br/>'
    + '</p>';
}}
```

3) 내장객체 : object, Math, Number, Date, Array, String,

4) 브라우저 객체 모델(BOM)

4-1) window 객체 속성 : window의 이름 폭, 너비, 좌표등
4-2) 새창 띄우기 : subwin = window.open / close
4-3) window 객체의 메소드 : 스크롤바 위치 조정
4-4) 타이머 및 시계

```
document.getElementById('result').innerText
```

. nullpointerexception

순서대로 읽으면 result 는 없는 상태므로 순서를 재조정하거나 onload를 통해 한번 더 읽어줘야 한다.
구형 이벤트 핸들러 : window.onload = function(){ } 안에 넣어줌
DomContentLoaded?

setTimeOut은 한번만 실행됨
setInterval 은 지정한 시간만큼 반복 실행
clearInterval 은 setInterval() 함수에 의해 실행된 반복 작업을 종료하는 함수

4-5) screen 객체를 이용해 모니터 정보 : 전체 화면 너비, 해상도등
4-6) location 객체를 이용한 페이지 주소(URL) 다루기 : 웹브라우저의 주소 표시줄의 속성 및 기능

- href : http://127.0.0.1:5501/JavaScriptStudy/JavaScriptStudyCh07/javascript07_06.html
- origin : http://127.0.0.1:5501
- pathname /JavaScriptStudy/JavaScriptStudyCh07/javascript07_06.html
- protocol : http:
- host : 127.0.0.1:5501
- hostname : 127.0.0.1
- port :5501
- search :
- 쿼리 스트링 없음

4-7) navigator 객체를 이용해 브라우저 정보 접근

- userAgent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
- appCodeName : Mozilla
- appName : Netscape
- appVersion : 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
- language : ko-KR
- product : Gecko
- platform : Win32
- onLine : true

4-8) history 객체를 이용해 방문한 사이트 정보 다루기 : history.back, history.forward, history.go

5) 문서 객체 모델(DOM, Document Object Model)

DOM에서 사용하는 객체를 노드**(Node)**라고 하고 상 하위 노드 간의 관계를 부모노드와 자식노드라 고 하며 같은 위치의 노드는 형제노드라고 부른다. 노드에는 12가지 노드 타입(Node Type)이 있지 만 잘 쓰이지 않는 것도 있으며 요소노드, 속성노드, 텍스트노드 정도만 기억하면 된다.

- 요소노드(element node) : <a> , , 등과 같은 태그를 의미
- 속성노드(attribute node) : id, name 등과 같은 태그 내의 속성을 의미
- 텍스트노드(text node) : 요소 안의 내용을 의미

JQuery 는 대부분의 브라우저의 호환을 해주므로 많은 개발단계에서 선호되어 왔다. 특히, 대기업들은 대부분 제이쿼리로 구성되어 있어 산적한 문제를 안고 있기에 지식이 필요한 편이다.

HTML 문서의 모든 구성 요소가 메모리에 준비되면, onload 를 해야 원할히 진행된다.

```
최신 :  document.addEventListener("DOMContentLoaded", function(){   })
구 : window.onload = function()
```

5-1) DOM에서 다양한 방법으로 문서 객체 선택

```
// DOM에서 id가 plan인 요소를 찾아 HTMLElement 객체로 읽어온다.
const plan = document.getElementById("plan");
// DOM에서 모든 li 요소를 찾아 HTMLCollection 객체로 읽어한다.
const liTags = document.getElementsByTagName('li');
// DOM에서 class가 done-list인 요소를 찾아 HTMLCollection 객체로 읽어온다.
const doneList = document.getElementsByClassName('done-list');

// DOM에서 CSS 선택자로 class가 plan-list인 첫 번째 요소를 Element 객체로 읽어온다.
const content = document.querySelector('.plan-list');
// DOM에서 CSS 선택자로 class가 plan-list인 모든 요소를 NodeList 객체로 읽어온다.
const contentAll = document.querySelectorAll('.plan-list');
```

5-2) DOM 트리 탐색

DOM 트리에서 부모, 자식, 형제간의 관계를 이용한 요소 노드 탐색 Element 객체의 children, parentElement, previousElementSibling, nextElementSibling, firstElementChild, lastElementChild 프로퍼티는 DOM 트리에서 요소 노드를 대상으로 탐색할 수 있는 프로퍼티들이다.

```
const container = document.getElementById("container");
const content1 = container.children[1];
const doneList = content1.nextElementSibling.children[1].children;
console.log(doneList[0].parentElement.firstElementChild)
```

5-3) 요소 노드의 프로퍼티 : innerHTML, outerHTML, textContent, innerText 는 요소 읽기 및 설정

5-4) 요소 노드의 추가, 수정, 삭제 : createElement(), cloneNode(), prepend(), append(), before(),after(), remove()

- append를 2개줘도 한개만 적용되므로 cloneNode 를 쓴다.
- prepend : append는 자식노드의 맨마지막에 추가되나, prepend는 맨앞으로 추가
- before, after : 특정 노드의 자식을 가져와 다른 노드의 자식으로 지정 가능

5-5) HTML 태그의 속성 다루기

input , box 같은 애들은 표준 속성이 있으나, 일부 태그들은 속성이 없어 비표준 상태이지만 유저가 직접 속성을 부여해 표준 속성으로 되는 것이다.

요소노드의 프로퍼티를 읽어올 수 없는 비표준 속성의 경우 Attribute 와 같은 특정 메서드를 사용해야 한다.

```
console.log(plan.parentElement.className); //content
console.log("plan.value : ", plan.value); //plan.value : undefined
console.log(plan.getAttribute("value")); //study-plan
console.log(plan.getAttribute("id")); //plan
```

요소 노드의 setAttribute() 메서드를 사용하면 HTML 태그에 표준과 비표준 속성 모두를 설정할 수 있으며 기존에 이미 설정된 속성도 수정할 수 있다.

```
plan.parentElement.setAttribute("id", "content"); //부모노드에 id = content 부여됨
plan.setAttribute("class", "plan");
plan.setAttribute("value", "javascript-study");
```

- HTML 에서는 `<ul id="plan" value="study-plan">` 로 확인되나,
- 개발자도구에서는 `<ul id="plan" value="javascript-study" class="plan">…</ul>` 변경이 확인된다.

CSS : 가상요소 선택자

```
<style>
 li[data-study='studying']::after {
   content :  " - studying";
   color: blue;
 }
 li[data-study='waiting']::after {
   content :  " - waiting";
   color: red;
 }
</style>
```

JavaScript 함수 - studying
JavaScript DOM - studying
JavaScript Event - studying
jQuery 함수 선택자 - waiting
jQuery Effect - waiting