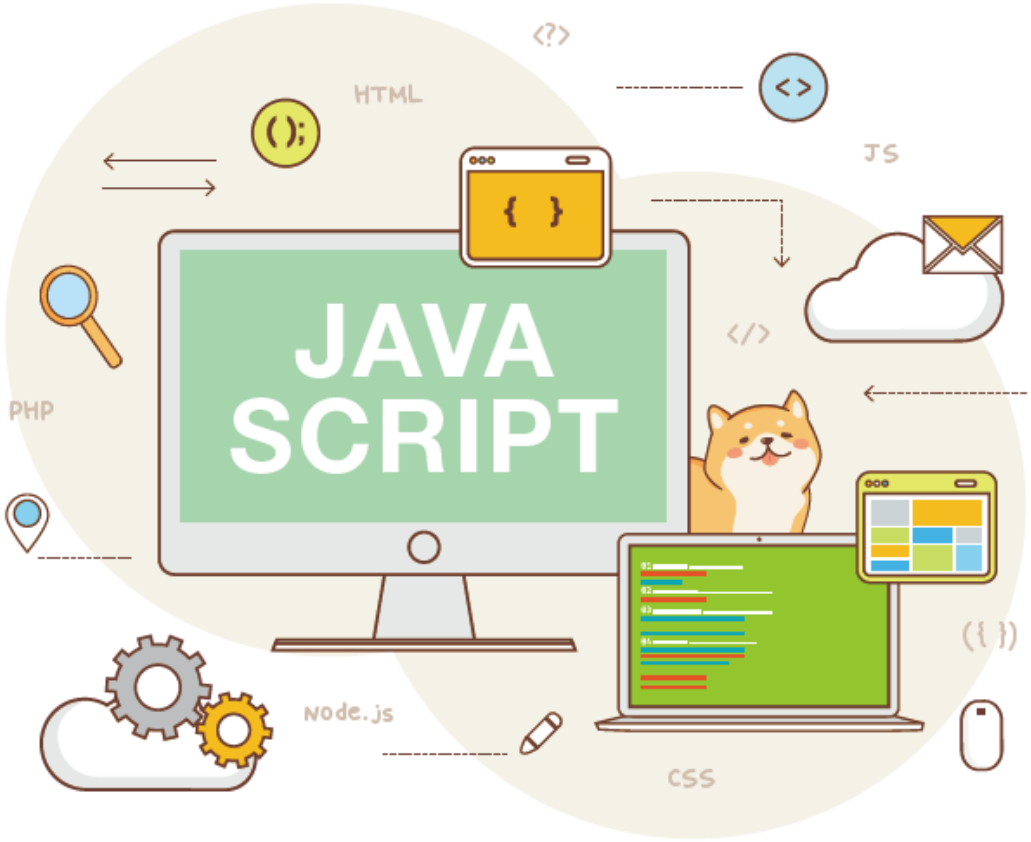
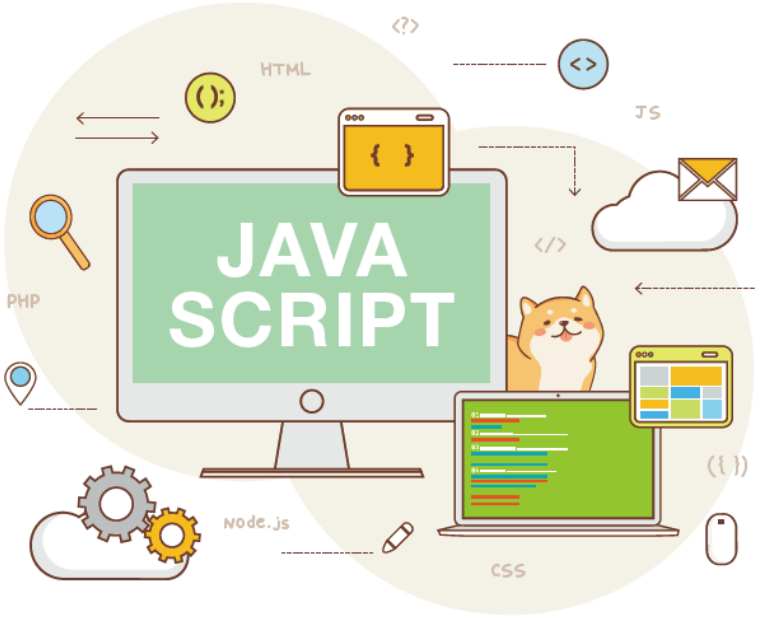



# 자바스크립트 심화
# (Chapter 5, 8)

**스마트헬스 케어 – 14주차**

# 학습 내용

1. 함수 심화
2. 콜백 함수
3. 타이머
4. 예외처리



# 이론 및 예제 실습

# 1_함수심화 - 객체식 정의

- 익명 함수로 정의

const [함수명] = function (가인자:인자정의) { 함수 로직 };

- 람다식(화살표 함수) 정의

const [함수명] = (가인자:인자정의) => { 함수 로직 };

```
let num = 10;

const plusNum = function (value) {
  console.log("value의 값은 " + value + "입니다.");
  return value + 10;
};

const plusNum2 = (value, plusnum = 20) => {
  console.log("value의 값은 " + value + "입니다.");
  return value + plusnum;
};

console.log("함수1 호출 결과 값은 " + plusNum(num) + "입니다.");
console.log("함수2 호출 결과 값은 " + plusNum2(num) + "입니다.");
```

# 1_함수심화 - 함수의 특성

- 자바스크립트에서 함수는 일급객체다

1) 변수에 할당 할 수 있다.

2) 함수를 다른 함수의 인자로 전달 할 수 있다.

3) 함수의 리턴값으로 전달 할 수 있다.

```
let num = 10;


const plusNum = function (value) {   // 1) 함수를 변수로 받음
  console.log("value의 값은 " + value + "입니다.");
  return value + 10;
};
```

# 1_함수심화 - 함수의 특성

- 자바스크립트에서 함수는 일급객체다

1) 변수에 할당 할 수 있다.

2) 함수를 다른 함수의 인자로 전달 할 수 있다.

3) 함수의 리턴값으로 전달 할 수 있다.

```
let num = 10;
const plusNum = function (value) {
  console.log("value의 값은 " + value + "입니다.");
  return value + 10;
};

const calculator = (func, num) => {   // 2) 함수를 인자로 전달
  console.log("value의 값은 " + num + "입니다.");
  let result = func(num);
  return result;
};

console.log("1결과 값은 " + calculator(plusNum, num) + "입니다.");
```

# 1_함수심화 - 함수의 특성

- 자바스크립트에서 함수는 일급객체다

1) 변수에 할당 할 수 있다.

2) 함수를 다른 함수의 인자로 전달 할 수 있다.

3) 함수의 리턴값으로 전달 할 수 있다. → 함수의 호출 결과로 함수를 받는다.

```
let num = 10;
const minusNum = function () {  // 함수를 호출해야 함수를 받음
  return function (value) {     // 3) 리턴값으로 함수 전달
    return value - 10;
  };
};

const calculator = (func, num) => {
  console.log("value의 값은 " + num + "입니다.");
  let result = func(num);
  return result;
};

console.log("2결과 값은 " + calculator(minusNum(), num) + "입니다.");
```

# 2_콜백(Callback)

- **함수의 매개변수로 전달**되며 이벤트 발생시 호출되는 함수
- 이벤트 : 타이머, UI 조작 이벤트, 네트워크 통신 이벤트

```
function callTenTimes(callback) {
   for (let i = 0; i < 10; i++) {
        // 매개 변수로 전달된 함수를 호출합니다.
        callback();
    }
}

callTenTimes(function () {   // 익명함수 정의
    console.log('함수 호출');
});
```

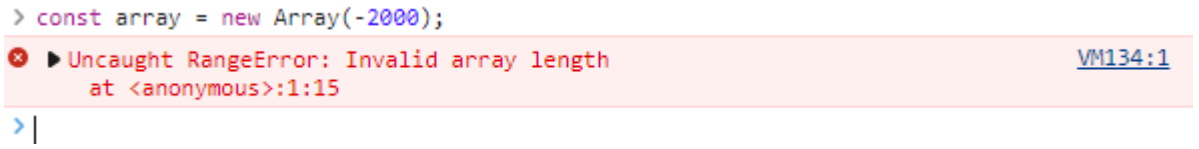
# 3_타이머 - 반복 실행, 시차를 두고 실행

- 반복 실행 : **setInterval([함수], [반복간격 : 밀리세컨], <옵션 : 호출시 실인자> )**
  . 실행을 멈추려면 clearInterval() 호출
- 지연 실행 : **setTimeout([함수], [지연시간 : 밀리세컨] , <옵션 : 호출시 실인자> )**
- 리턴값이 있는 함수 실행시 리턴값을 받을 수 없음

```
let num = 10;

function plusNum(value, plusnum = 20) {
  let num = value + 10;
  console.log("함수 호출 결과 값은 " + num + "입니다.");
}

setTimeout(plusNum, 2000, 10);   // 2초후 실행

let timer = setInterval(plusNum, 2000, 10);   // 2초 마다 실행
setTimeout(() => {
  clearInterval(timer);
  console.log("타이머 종료");
}, 6000);
```

# 4_예외 처리

- 예외 : 구문상의 오류가 아닌 로직상의 오류




- 자바스크립트는 실행에 문제가 발생하면(예외가 발생하면) 자동 중단됨
- 예외 처리 : 문제가 생겨도 다음 내용을 수행할 수 있도록 오류에 대한 처리가 필요
- try [예외발생 구문]  catch [예외발생시 처리 로직] finally [예외가 발생하든 안 하든 실행하는 로직]

```
try {
        const array = new Array(-2000);

} catch (exception) {
        console.log(`${exception.name} 예외 발생`);

} finally {
        console.log("finally 구문 실행");

}
```

```
try {
    // 예외가 발생하면
} catch (exception) {
    // 여기서 처리합니다.
} finally {
    // 여기는 무조건 실행합니다.
}
```

# 응용 예제

# 실습-1

**실습 문제** : 1초에 한번씩 브라우저에 시간을 출력하는 프로그램

document.write() 함수로 출력을 내보낼 것

# 실습-2

**실습 문제** : 타이머를 이용하여 달리는 이미지를 0.1초마다 다음 장면으로 바꾸는 애니메이션으로 구현하시오.

images.zip 파일을 깃허브에서 다운로드 하시오.

# 실습-2

**실습 문제** : 타이머를 이용하여 달리는 이미지를 0.1초마다 다음 장면으로 바꾸는 애니메이션으로 구현하시오.

```
<img src="./images/0.png" id="img1" />

<script>
  let idx = 0;
  let imgCom = document.querySelector("#img1");

  setInterval(changeRunner, 100, 1);

  function changeRunner() {
    let imgIndex = idx % 15;
    imgCom.setAttribute("src", `./images/${imgIndex}.png`)
    idx++;
  }

</script>
```