

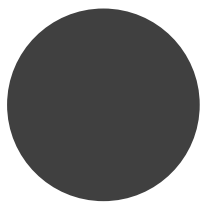

# 자바스크립트 기반의 영상 처리를 활용한 벽돌깨기 게임

대한상공회의소 서울기술교육센터  
자바기반 빅데이터 시각화 시스템 개발  
프론트엔드 프로그래밍

임선우



## 목차

- 01 프로젝트 개요
  - 02 프로젝트 소개
  - 03 화면 구성
  - 04 코드 구현
  - 05 시연 영상
- 
- 

# 프로젝트 개요

## 프론트엔드(front-end) 프로그래밍

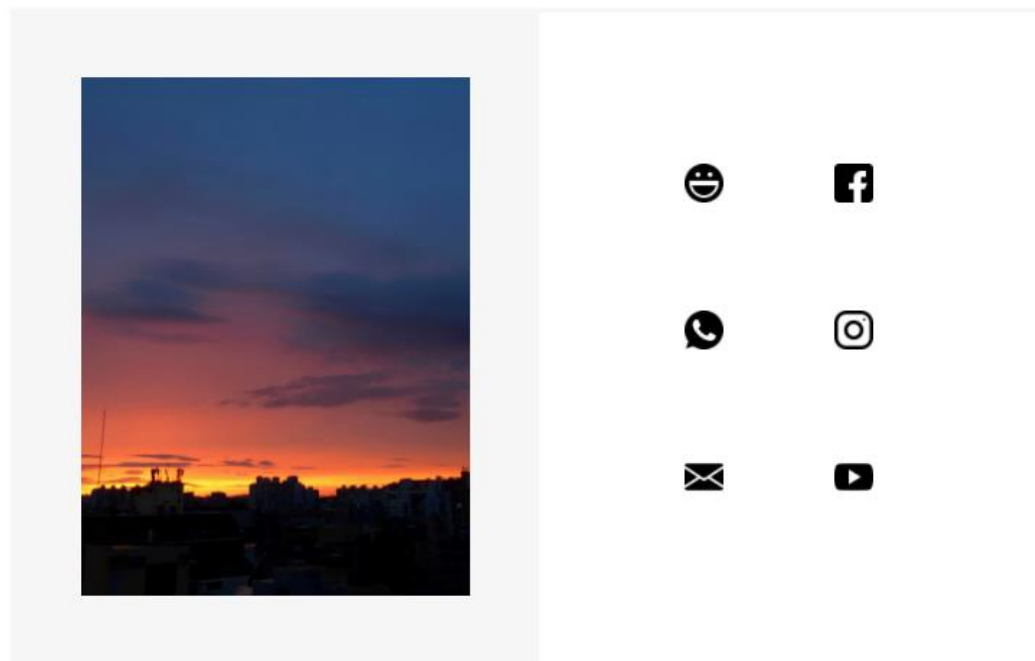
### 학습 내용

- html 태그, css 스타일 시트 작성
- javascript를 이용한 동적 구현
- 디지털 이미지에 대한 영상 처리 알고리즘

### 개인 프로젝트

주제: 영상 처리를 활용한 벽돌깨기 게임  
사용 언어: HTML5 / CSS3 / JAVASCRIPT  
개발 환경: Visual Studio Code  
제작 기간: 21.03.25 - 21.03.30

HELLO



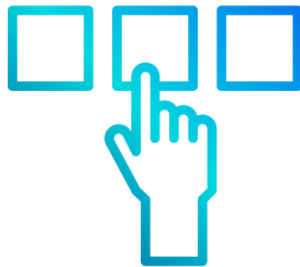
### My Project

영상처리를 활용한 벽돌깨기 게임  
html / css / javascript / jquery

GO

## 프로젝트 소개

### CHOOSE



사용자가 이미지를 가지고 있지 않아도 프로그램을 사용할 수 있도록 개발자가 제공하는 범위에서 원하는 이미지를 선택할 수 있게 하였습니다.

### EDIT



화소점 처리, 기하학 처리, 에지 검출 등 여러 영상처리 기법을 이용하여 사용자가 선택한 이미지를 편집할 수 있습니다.

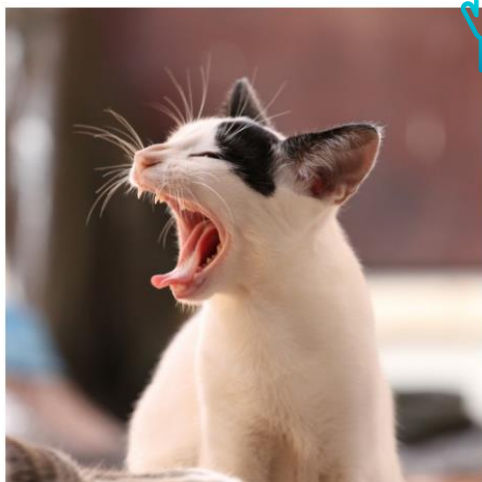
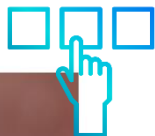
### USE



편집한 이미지를 활용하여 벽돌깨기 게임이 진행되며 이를 통해 사용자의 흥미와 관심을 더욱 이끌어냅니다.

# 화면 구성

Choose Your IMAGE



SELECT

NEXT

Edit Your IMAGE

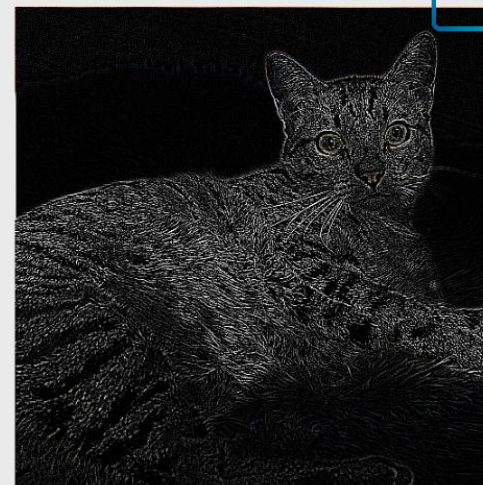


SAVE

NEXT



Break Your IMAGE



HOME

START

## 코드 구현 - 이미지 선택

Choose Your IMAGE



SELECT

NEXT

이미지 출력

```
function drawImage() {  
    // 캔버스 생성  
    inCanvas = document.getElementById('inCanvas');  
    inCtx = inCanvas.getContext('2d');  
  
    // 빈 이미지 객체 생성  
    var inImage = new Image();  
    // 미리 저장되어 있는 png 파일을 이미지 객체로 불러오기  
    inImage.src = num + ".png";  
  
    inImage.onload = function() {  
        // 캔버스 크기를 png 파일 크기로 결정  
        inCanvas.width = inImage.width;  
        inCanvas.height = inImage.height;  
        // 캔버스에 이미지 그리기  
        inCtx.drawImage(inImage,0,0,inWidth,inHeight);  
    }  
}
```

슬라이드

```
// png 파일 (1.png ~ 5.png)  
var min = 1;  
var max = 5;  
  
function slideImage(value) {  
    // 이전으로 넘기기  
    if(!value) {  
        // 1번 파일의 경우 마지막 파일로 넘겨줌  
        if(num == min)  
            num = max + 1;  
        num--;  
    }  
    // 다음으로 넘기기  
    else{  
        // 마지막 파일일 경우 1번 파일로 넘겨줌  
        if(num == max)  
            num = min - 1;  
        num++;  
    }  
}
```

## 코드 구현 - 이미지 선택

Choose Your IMAGE



이미지 파일 이름 저장

```
// 이미지 선택하기  
// 파일 이름 저장 후 다음 html파일에서 불러오기  
function selectImage() {  
    var imageNum = num;  
    // key와 value를 이용하여 스토리지에 저장  
    sessionStorage.setItem("key", imageNum);  
}
```

다음 html로 이동

```
<a type="button" onclick="location.href='editImage.html'"></a>
```

SELECT

NEXT



# 코드 구현 - 이미지 편집

## ① 저장된 파일 이름 가져오기

```
// 선택한 이미지 그리기
var inImage = new Image();
// 스토리지에 저장된 파일 이름
var imageNum = sessionStorage.getItem("key");
inImage.src = imageNum + ".png";
```

## ② 3차원 배열 생성

```
// 이미지 처리를 위한 3차원 배열 준비 (적용 / 미리보기)
// R, G, B
imageArray = new Array(3);
for(var j=0; j<3; j++) {
    // 이미지 높이
    imageArray[j] = new Array(height);
    for(var i=0; i<inHeight; i++)
        // 이미지 너비
        inImageArray[j][i] = new Array(width);
}
```

## ③ 이미지 픽셀값 저장

```
// 캔버스에서 픽셀값을 뽑아 배열에 저장
var imageData = ctx.getImageData(0,0,width,height);
var R, G, B;
for(var i=0; i<height; i++) {
    for (var k=0; k<inWidth; k++) {
        // 1pixel = 4byte
        pixel = (i*width + k)*4;
        R = imageData.data[pixel + 0];
        G = imageData.data[pixel + 1];
        B = imageData.data[pixel + 2];
        //A = imageData.data[pixel + 3];
        imageArray[0][i][k] = String.fromCharCode(R);
        imageArray[1][i][k] = String.fromCharCode(G);
        imageArray[2][i][k] = String.fromCharCode(B);
    }
}
```

Edit Your IMAGE

PREVIEW



RESET

POINT1

POINT2

GEOMETRY

HISTOGRAM

CONVOLUTION

EDGE

APPLY

SAVE

NEXT

## ④ 캔버스에 이미지 출력

```
// 배열에 저장된 픽셀값을 통해 캔버스에 이미지 그리기
function displayImage() {
    var R, G, B;
    // 캔버스에 이미지를 바로 그릴 수 없기 때문에 종이 사용
    paper = ctx.createImageData(width, height);
    for(var i=0; i<height; i++) {
        for (var k=0; k<width; k++) {
            // byte 문자를 숫자로 변경
            R = imageArray[0][i][k].charCodeAt(0);
            G = imageArray[1][i][k].charCodeAt(0);
            B = imageArray[2][i][k].charCodeAt(0);
            // 종이에 픽셀값 저장
            paper.data[(i*width + k) * 4 + 0] = R;
            paper.data[(i*width + k) * 4 + 1] = G;
            paper.data[(i*width + k) * 4 + 2] = B;
            paper.data[(i*width + k) * 4 + 3] = 255;
        }
    }
    // 종이 붙이기
    ctx.putImageData(paper, 0, 0);
}
```



# 코드 구현 - 이미지 편집

Edit Your IMAGE



서브버튼 보이기/숨기기

```
// 서브버튼 숨기기
var subBtn = document.getElementById('subBtn');
subBtn.style.visibility = 'hidden';

// 메인버튼을 누를 경우 서브버튼 보이기
function showButton(mainBtnNum) {
  switch(mainBtnNum) {
    case 1:
      if(subBtn.style.visibility == 'visible')
        subBtn.style.visibility = 'hidden';
      else {
        subBtn.style.visibility = 'visible';
        otherSubBtn.style.visibility = 'hidden';
      }
      break;
  }
}
```

이미지 저장

```
// 편집한 이미지 저장하기
function saveImage() {
  // 이미지 저장 버튼과 연결
  console.log("saveImage()");
  // 저장할 이미지 파일: myImage.png
  var imageName = "myImage";
  console.log(imageName.length);
  if (imageName.length == 0) {
    imageName = "image";
  }
  imageName += ".png";
  var savedImage = document.getElementById("saveImage");
  var image = document
    .getElementById("inCanvas")
    .toDataURL("image/png")
    .replace("image/png", "image/octet-stream");

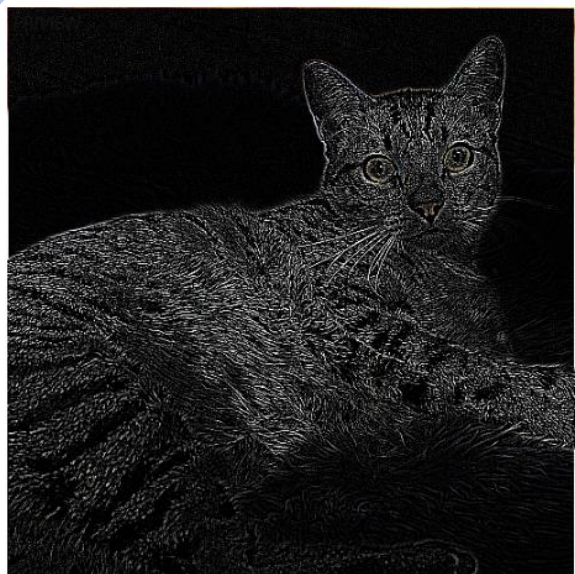
  savedImage.setAttribute("download", imageName);
  savedImage.setAttribute("href", image);
}
```

<!-- 이미지 저장 버튼 -->

<a id="saveImage" type="button" download="image.png" onclick="saveImage()"></a>

# 코드 구현 - 이미지 편집

## Edit Your IMAGE



SAVE

```
// 단순 영상처리 함수
function editImage1() {
    var R, G, B;
    for(var i=0; i<height; i++) {
        for (var k=0; k<width; k++) {
            // 배열에 저장되어 있는 픽셀값을 숫자로 변경
            R = inImageArray[0][i][k].charCodeAt(0);
            G = inImageArray[1][i][k].charCodeAt(0);
            B = inImageArray[2][i][k].charCodeAt(0);

            // ** 여러 영상처리 알고리즘 **

            // 영상처리를 거친 픽셀값을 배열에 문자로 저장
            editImageArray[0][i][k] = String.fromCharCode(R);
            editImageArray[1][i][k] = String.fromCharCode(G);
            editImageArray[2][i][k] = String.fromCharCode(B);
        }
    }
    // 이미지 출력 함수
    displayImage();
}
```

## 영상처리 함수

```
// 마스크(3*3)를 사용한 영상처리 함수
function editImage2(mask) {
    // 마스크 처리를 위한 임시 배열 생성
    tempArray = new Array(3);
    for(var j=0; j<3; j++) {
        tempArray[j] = new Array(height + 2);
        for(var i=0; i<height + 2; i++)
            tempArray[j][i] = new Array(width + 2);
    }
    // 임시 배열 초기화
    for(var j=0; j<3; j++) {
        for(var i=0; i<height + 2; i++)
            for(var k=0; k<width + 2; k++)
                tempArray[j][i][k] = String.fromCharCode(127);
    }
    for(var i=0; i<height; i++)
        for (var k=0; k<width; k++)
            for (var m=0; m<3; m++)
                for (var n=0; n<3; n++) {
                    // 마스크 처리
                    maskR += mask[m][n]*tempArray[0][i+m][k+n].charCodeAt(0);
                    maskG += mask[m][n]*tempArray[1][i+m][k+n].charCodeAt(0);
                    maskB += mask[m][n]*tempArray[2][i+m][k+n].charCodeAt(0);
                }
            // 영상처리를 거친 픽셀값을 배열에 문자로 저장
            editImageArray[0][i][k] = String.fromCharCode(maskR);
            editImageArray[1][i][k] = String.fromCharCode(maskG);
            editImageArray[2][i][k] = String.fromCharCode(maskB);
        }
    }
    // 이미지 출력 함수
    displayImage();
}
```

# 코드 구현 - 이미지 편집

## 화소점 처리



### 명도 조정

```
if(pixel + value > 255)
    pixel = 255;
else if(pixel + value < 0)
    pixel = 0;
else
    pixel += value;
```



### 그레이스케일

```
var RGB = parseInt((R+G+B) / 3);
```



### 대비 조정

```
if(pixel*value > 255)
    pixel = 255;
else if(pixel*value < 0)
    pixel = 0;
else
    pixel *= value;
```



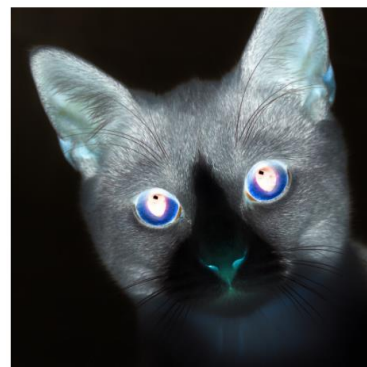
### 흑백 처리

```
var RGB = parseInt((R+G+B) / 3);
if(RGB > 127)
    RGB = 255;
else
    RGB = 0;
```



### 감마 조정

```
pixel = Math.trunc(((pixel/255)**(1/value))*255);
```



### 반전 처리

```
R = 255 - inImageArray[0][i][k].charCodeAt(0);
G = 255 - inImageArray[1][i][k].charCodeAt(0);
B = 255 - inImageArray[2][i][k].charCodeAt(0);
```



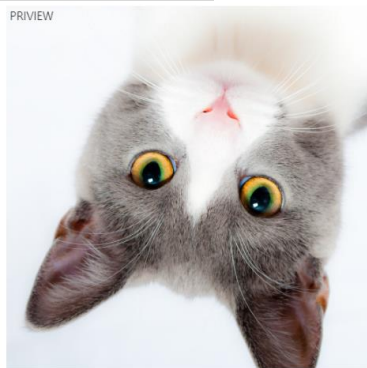
### 포스터라이징

```
for(var j=0 ; j<256 ; j+=256/value) {
    if(R >= j && R < j + 256/value)
        R = j + 256/value - 1;
    if(G >= j && G < j + 256/value)
        G = j + 256/value - 1;
    if(B >= j && B < j + 256/value)
        B = j + 256/value - 1;
}
```



# 코드 구현 - 이미지 편집

## 기하학 처리



### 상하 뒤집기

```
outImageArray[j][i][k] = inImageArray[j][inHeight-1-i][k];
```



### 좌우 뒤집기

```
outImageArray[j][i][k] = inImageArray[j][i][inWidth-1-k];
```



### 시계방향 회전

```
outImageArray[j][i][k] = inImageArray[j][inHeight-1-i][k];  
tmp = outImageArray[j][i][k];  
outImageArray[j][i][k] = outImageArray[j][k][i];  
outImageArray[j][k][i] = tmp;
```



### 반시계방향 회전

```
outImageArray[j][i][k] = inImageArray[j][i][inWidth-1-k];  
tmp = outImageArray[j][i][k];  
outImageArray[j][i][k] = outImageArray[j][k][i];  
outImageArray[j][k][i] = tmp;
```

## 히스토그램



### 히스토그램 평활화

```
histo[j][value]++;  
sumHisto[j][i] = sumHisto[j][i - 1] + histo[j][i];  
normalHisto[j][i] = sumHisto[j][i]*(1/(inWidth*inHeight))*255;  
outVal = parseInt(normalHisto[j][inVal]);
```

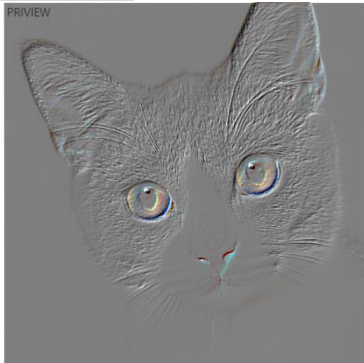


### 엔드인 탐색

```
outValR = (inValR - minR)/(maxR - minR)*255;  
outValG = (inValG - minG)/(maxG - minG)*255;  
outValB = (inValB - minB)/(maxB - minB)*255;
```

# 코드 구현 - 이미지 편집

## 마스크



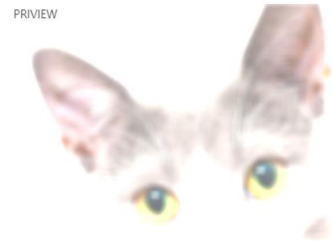
### 엠보싱

```
mask = [[-1.,0.,0.],[0.,0.,0.],[0.,0.,1.]];
outValR = SR + 127;
outValG = SG + 127;
outValB = SB + 127;
```



### 샤프닝

```
mask = [[0.,-1.,0.],[-1.,5.,-1.],[0.,-1.,0.]];
```



### 블러링

```
var mask = new Array(value);
for(var i=0; i<value; i++)
    mask[i] = new Array(value);
for(var i=0; i<value; i++)
    for(var k=0; k<value; k++)
        mask[i][k] = 1/(value*value);
```

## 에지 검출



### 수직수평에지

```
mask = [[0.,-1.,0.],[-1.,2.,0.],[0.,0.,0.]];
```



### 차연산자

```
pixel = Math.abs(tempInArray[i+0][k+0].charCodeAt(0) - tempInArray[i+2][k+2].charCodeAt(0));
pixel = Math.abs(tempInArray[i+0][k+1].charCodeAt(0) - tempInArray[i+2][k+1].charCodeAt(0));
pixel = Math.abs(tempInArray[i+0][k+2].charCodeAt(0) - tempInArray[i+2][k+0].charCodeAt(0));
pixel = Math.abs(tempInArray[i+1][k+2].charCodeAt(0) - tempInArray[i+1][k+0].charCodeAt(0));
```



### 라플라시안

```
mask = [[1.,1.,1.],[1.,-8.,1.],[1.,1.,1.]];
mask = [[0.,1.,0.],[1.,-4.,1.],[0.,1.,0.]];
```

## 코드 구현 - 이미지 활용

### 편집한 이미지 단순 출력

```
// 편집한 이미지를 캔버스에 그리기
function openImage() {
  // 빈 이미지 객체 생성
  var inImage = new Image();
  // 앞서 myImage.png로 저장한 파일을 이미지 객체로 불러오기
  inImage.src = "myImage.png";

  inImage.onload = function() {
    // png 파일 크기
    inWidth = inImage.width;
    inHeight = inImage.height;
    // 캔버스에 이미지 그리기
    ctx.drawImage(inImage, brickOffsetLeft, brickOffsetTop,
  }
}
```

## Break Your IMAGE



HOME

START

참고: MDN Web Docs  
[https://developer.mozilla.org/ko/docs/Games/Tutorials/2D\\_Breakout\\_game\\_pure\\_JavaScript](https://developer.mozilla.org/ko/docs/Games/Tutorials/2D_Breakout_game_pure_JavaScript)

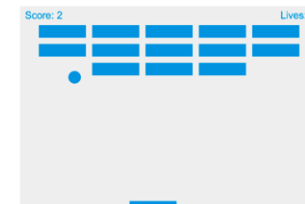
### 2D breakout game using pure JavaScript

Next »

In this step-by-step tutorial we create a simple **MDN Breakout** game written entirely in pure JavaScript and rendered on HTML5 `<canvas>`.

Every step has editable, live samples available to play with so you can see what the intermediate stages should look like. You will learn the basics of using the `<canvas>` element to implement fundamental game mechanics like rendering and moving images, collision detection, control mechanisms, and winning and losing states.

To get the most out of this series of articles you should already have basic to intermediate [JavaScript](#) knowledge. After working through this tutorial you should be able to build your own simple Web games.



### 벽돌깨기 게임 시작

```
// 카운트다운 후 게임 시작
function startCountdown() {
  // 호출 스케줄링: 함수 예약 실행
  setTimeout(function() {drawCount("3");},0);
  setTimeout(function() {drawCount("2");},1000);
  setTimeout(function() {drawCount("1");},2000);
  setTimeout(function() {drawCount("GO!");},3000);
  // 4초 후 게임 실행 함수 호출
  setTimeout(function() {startGame();},4000);
}
```



## 코드 구현 - 이미지 활용

## 이미지를 벽돌로 출력

// \*\* 벽돌 관련 변수 \*\*

```
// 벽돌 개수
var brickRowCount = 8;
var brickColumnCount = 16;
// 벽돌 크기
var brickWidth = 64;
var brickHeight = 32;
// 벽돌 사이 패딩
var brickPadding = 0;
// 벽돌과 캔버스 간격
var brickOffsetTop = 30;
var brickOffsetLeft = 24;

// 벽돌 배열 생성
var bricks = [];
for(var c=0; c<brickColumnCount; c++) {
  bricks[c] = [];
  for(var r=0; r<brickRowCount; r++) {
    // 벽돌을 그릴 위치와 상태
    bricks[c][r] = {x: 0, y: 0, status: 1};
  }
}
```

// \*\* 벽돌 그리기 \*\*

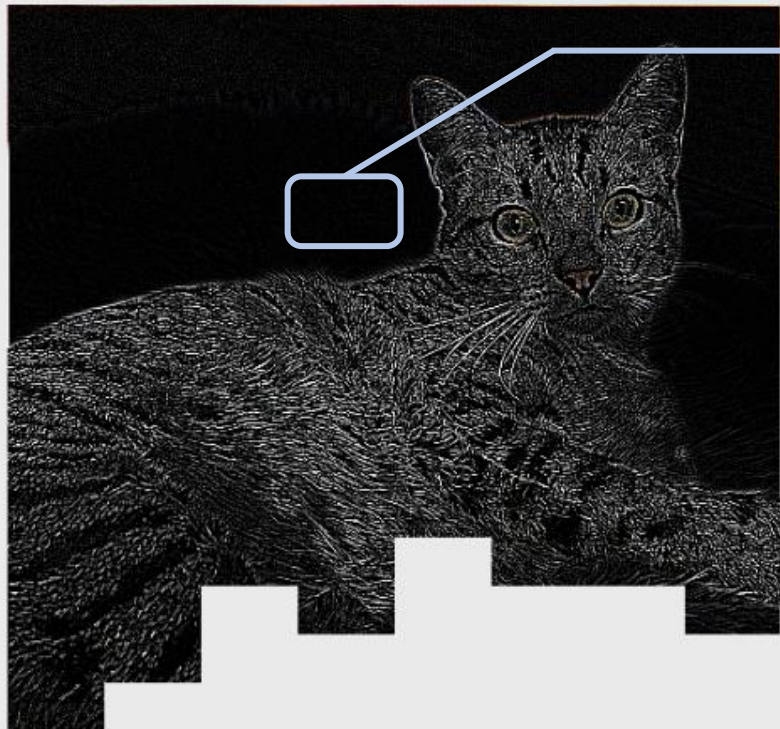
```
function drawBricks() {
  for(var c=0; c<brickColumnCount; c++) {
    for(var r=0; r<brickRowCount; r++) {
      // status == 1: 벽돌이 그려진 상태
      if(bricks[c][r].status == 1) {
        // 벽돌 그릴 위치 조정
        var brickX = (r*(brickWidth+brickPadding))+brickOffsetLeft;
        var brickY = (c*(brickHeight+brickPadding))+brickOffsetTop;
        // 각각 벽돌 변수에 위치 저장
        bricks[c][r].x = brickX;
        bricks[c][r].y = brickY;
        // 벽돌 크기만큼 종이에 픽셀값 저장
        drawImage(c, r);
        // 종이 옮기기
        ctx.putImageData(outPaper,brickX,brickY);
      }
    }
  }
}
```

// \*\* 이미지 픽셀값 저장하기 \*\*

```
function drawImage(m, n) { // 매개변수로 벽돌 그릴 위치를 받음
  // 출력을 위한 3차원 배열 준비
  outHeight = inHeight/brickColumnCount;
  outWidth = inWidth/brickRowCount;
  outImageArray = new Array(3);
  for(var j=0; j<3; j++) {
    outImageArray[j] = new Array(outHeight);
    for(var i=0; i<outHeight; i++)
      outImageArray[j][i] = new Array(outWidth);
  }
  // 벽돌 크기만큼 이미지 출력
  for(var j=0; j<3; j++) {
    for(var i=0; i<outHeight; i++)
      for (var k=0; k<outWidth; k++)
        // m, n: 벽돌 그릴 위치
        outImageArray[j][i][k] = inImageArray[j][outHeight*m+i][outWidth*n+k];
  }
  // 종이 사용
  outPaper = ctx.createImageData(outWidth, outHeight);
  // 종이에 픽셀값 저장
  var R, G, B;
  for(var i=0; i<outHeight; i++) {
    for (var k=0; k<outWidth; k++) {
      R = outImageArray[0][i][k].charCodeAt(0);
      G = outImageArray[1][i][k].charCodeAt(0);
      B = outImageArray[2][i][k].charCodeAt(0);
      outPaper.data[(i*outWidth + k) * 4 + 0] = R;
      outPaper.data[(i*outWidth + k) * 4 + 1] = G;
      outPaper.data[(i*outWidth + k) * 4 + 2] = B;
      outPaper.data[(i*outWidth + k) * 4 + 3] = 255;
    }
  }
}
```

Score: 26

Lives: 2

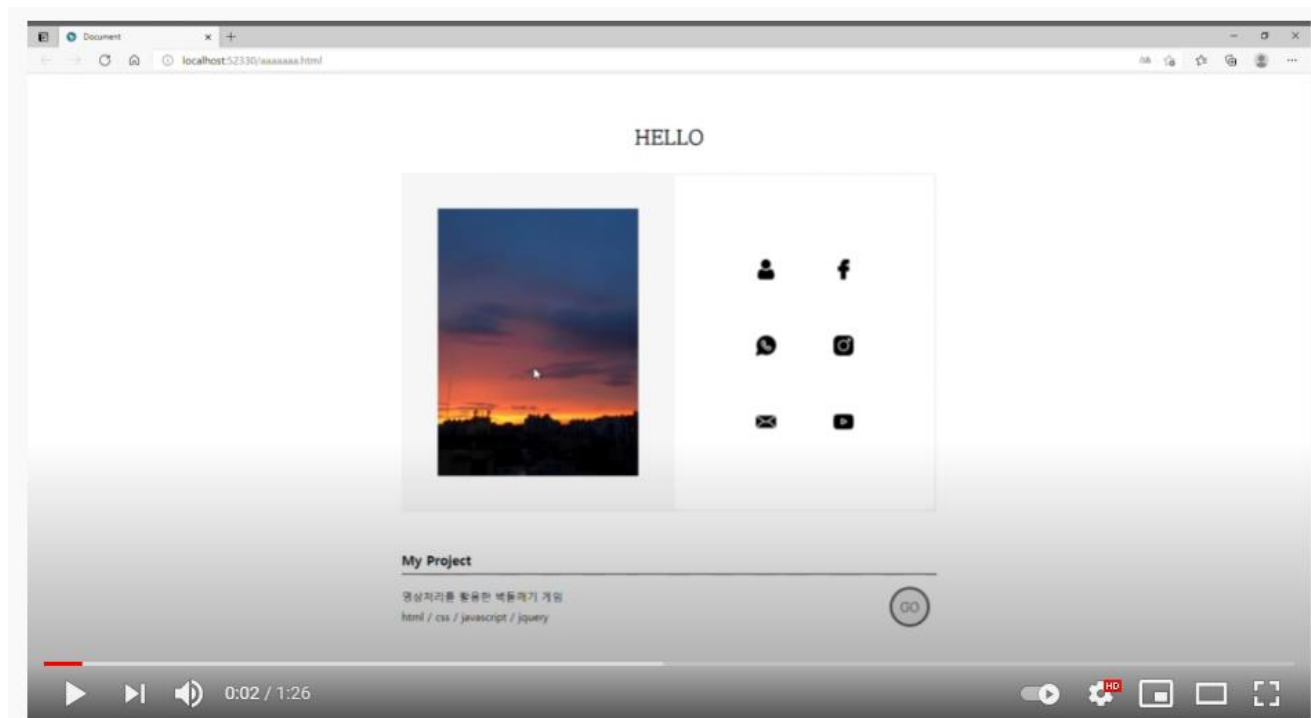


HOME

START



# 시연 영상



<https://youtu.be/gYLwj4hH6UE>

3월 마지막 주를 갈아 넣었다 html css javascript front-end programming 이렇게 제목을 지으면 검색하다가 내 영상 보겠지?!

조회수 없음 • 2021. 3. 31.

👍 0 💬 0 ➦ 공유 📌 저장 ...



선우의하루 [SUNday]  
구독자 22명

분석

동영상 수정

열심히 만들었습니다.  
소스 코드 공유 가능합니다.

를랄라뽕뽕



감사합니다

