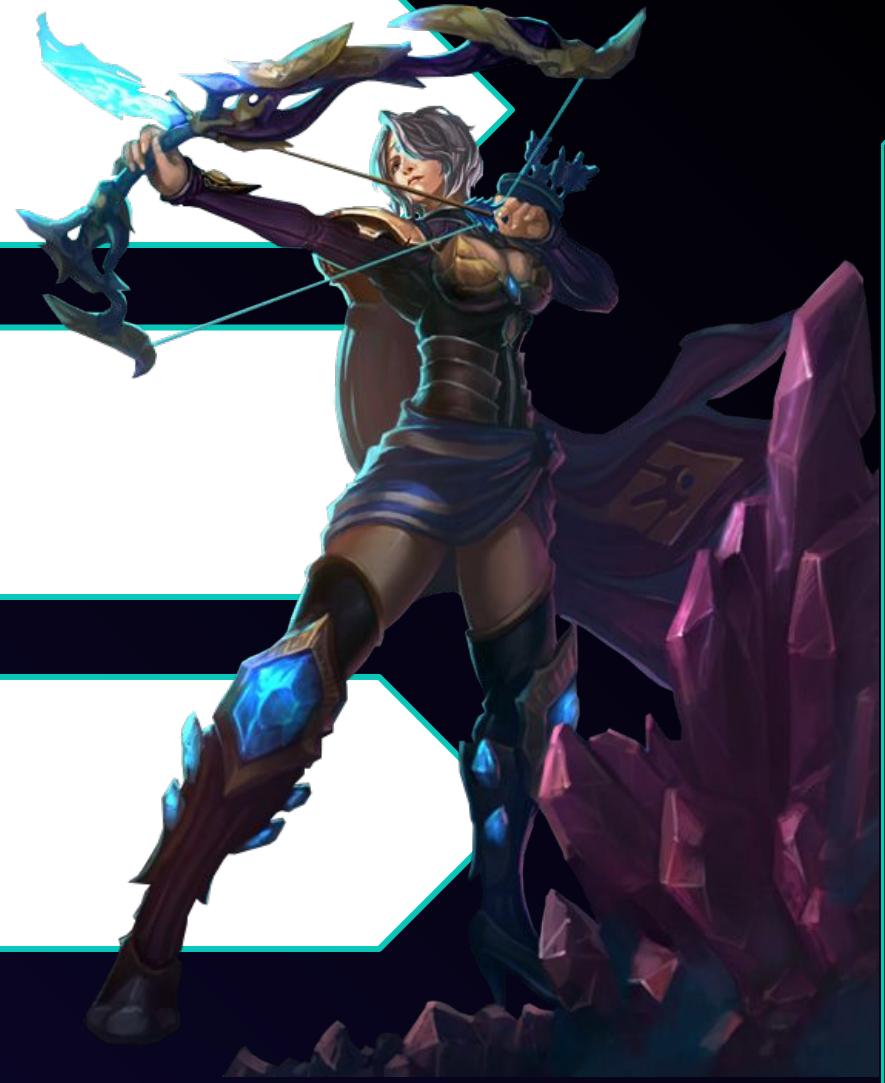


INDEX

step
01

기획

진행과정, 주제선정, 페르소나, 와이어프레임



step
02

구현 및 코드

구현화면, 소스코드

step
03

평가 및 후기

평가, 후기

01 기획



진행 과정



I Main 개발

2022.07.07~07.
15

2



Sub 개발

2022.07.29~08.
07

3



Main 수정

2022.08.08~08.09

4

문서작업

2022.08.10



주제 선정



전세계적으로 인터넷 게임은 많은 사람들이 해보지 않은 경우가 없을 정도로 대부분의 사람들이 즐깁니다. 또한, e-스포츠라는 장르가 생길 정도로 게임 경기도 인기가 많습니다.

많은 게임들 중에서도 인기가 높은 게임인 리그 오브 레전드는 라이엇 게임즈가 개발한 AOS게임 입니다. 전 세계적으로 많은 유저들을 보유 중이며, 세계에서 많이 플레이 하는 게임 중 하나입니다. 세계 E스포츠 대회인 리그 오브 레전드 월드 챔피언십 및 각 지역 리그 등 수많은 e스포츠 대회가 많이 열리며, 2018년 자카르타 아시안 게임에서는 공식 시범 종목으로 채택되기도 했습니다.

이렇게 많은 유저들을 보유하고 있는 게임은 공식 사이트를 구현했는지 찾아보게 되었고, 이를 리뉴얼하게 되면, 어떤 방식으로 사이트를 구상할 지에 대해서 고민했고 구현하게 되었습니다.



페르소나



“단순히 게임하는 것도 좋지만, 세계관도 탄탄히 잡혀 있었으면 좋겠어!”

이름: 이홍기

나이: 33세

사는곳: 경기도 성남

직업: 가수



라이프 스타일

15년차에 접어든 가수로 평소에 활동적이고 다양한 스포츠를 하는 것을 즐기지만, 집에 있을 때에는 혼자 게임을 하는 것을 즐긴다. 감성이 풍부하고, 연기나 뮤지컬 등 다양한 분야를 탄탄하게 가지고 가면서, 스토리에 대한 흥분에 대해서 관심이 많이 생겼고, 틈나는 시간에 게임을 하면서 캐릭터에 대한 세계관을 눈여겨 보는 것을 즐기는 편이다.



와이어 프레임 - Main



The wireframe diagram illustrates the layout of the League of Legends Main menu. It features a dark blue header bar at the top with five tabs: "로고" (Logo), "게임소개" (Game Overview), "챔피언" (Champions), "유니버스" (Universes), and "게임시작" (Game Start). Below the header, there is a large central section labeled "리그 오브 레전드" (League of Legends) with a descriptive paragraph about the game's objective. To the left of this central section is a placeholder for a character image labeled "캐릭터 png". At the bottom center is a large, white, diamond-shaped placeholder labeled "큰 로고" (Large Logo). The right side of the screen contains four large, light gray rectangular placeholders, each with a dark gray horizontal bar across its middle containing the text "그들에게" (To them).

로고

게임소개

챔피언

유니버스

게임시작

리그 오브 레전드

캐릭터 png

큰 로고

그들에게

그들에게

그들에게

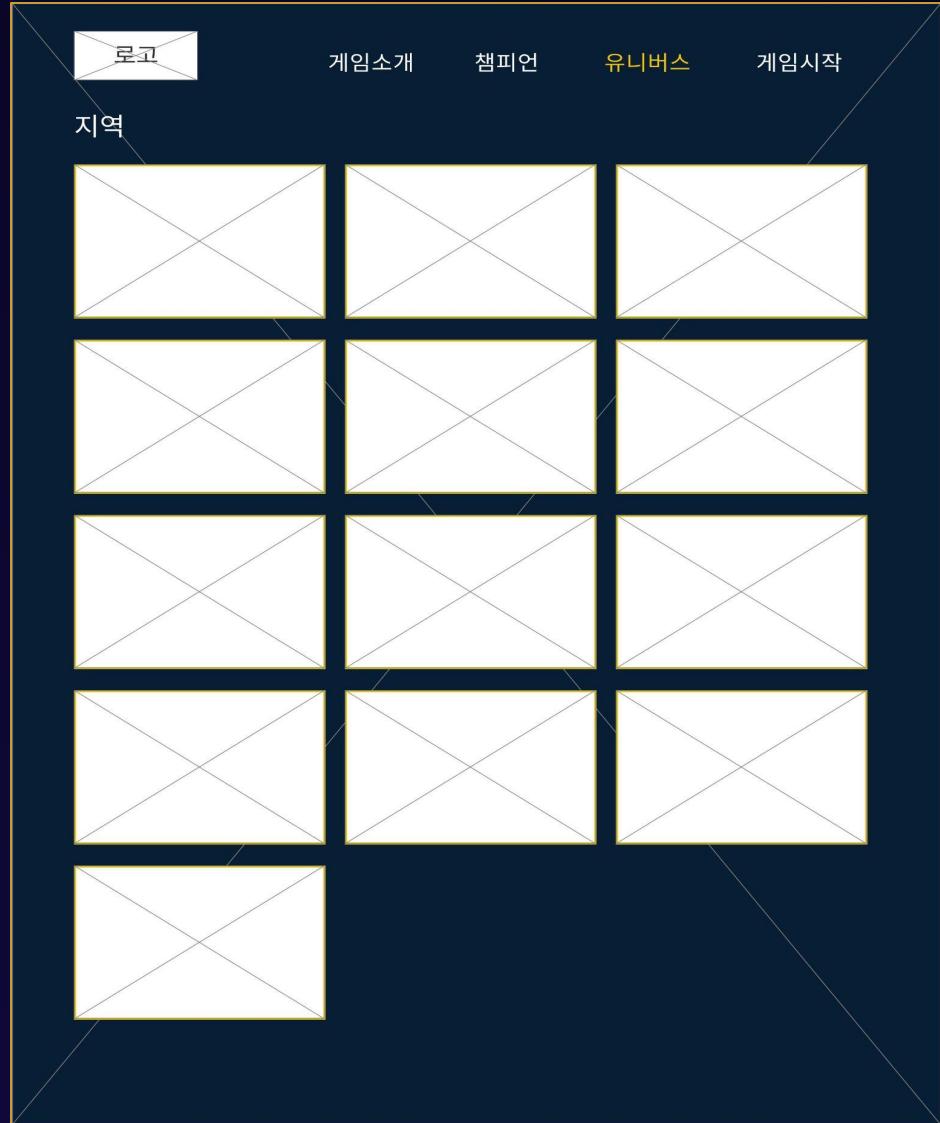
그들에게

그들에게

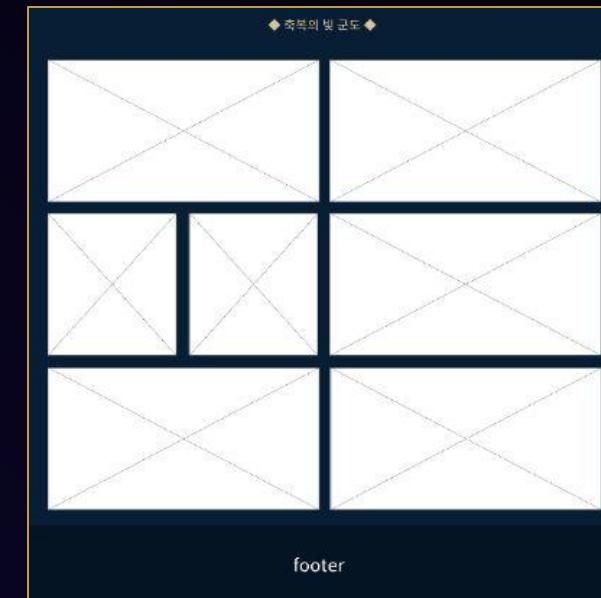
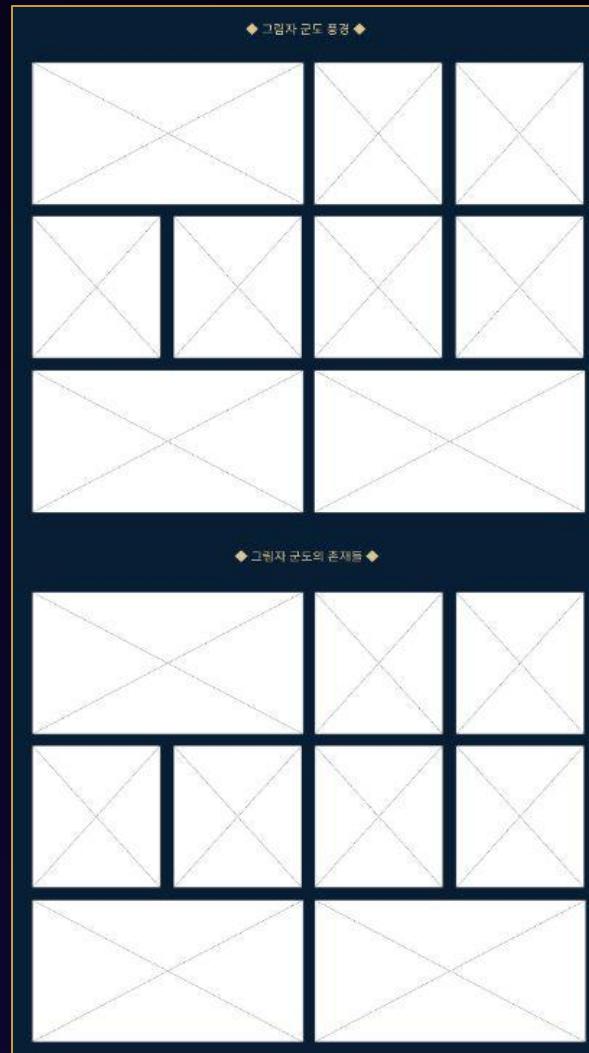
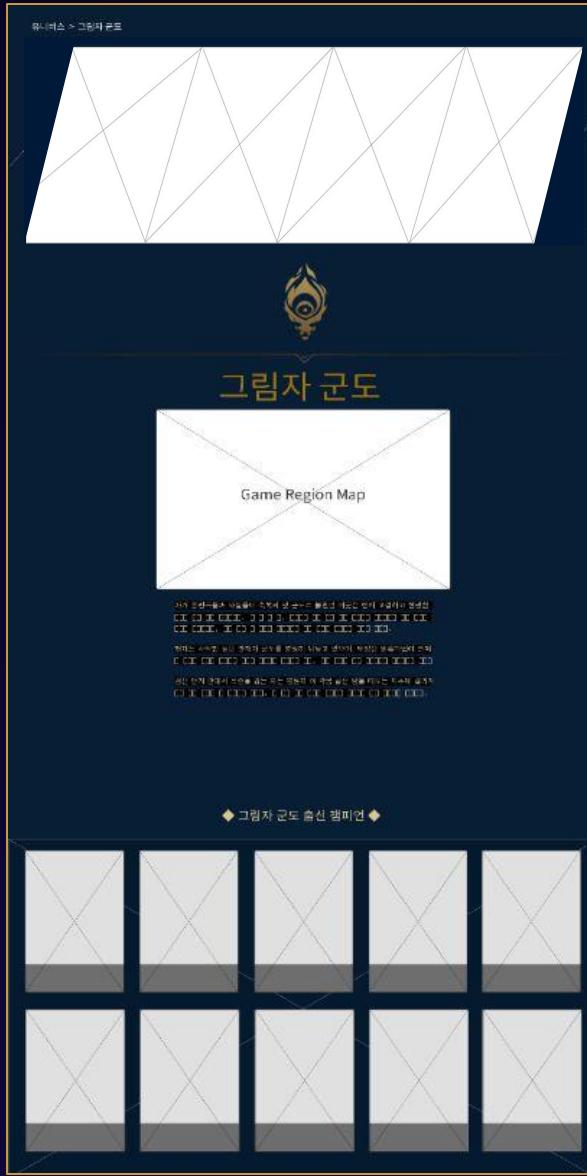
그들에게

그들에게

와이어 프레임 - Main



와이어 프레임 - Sub

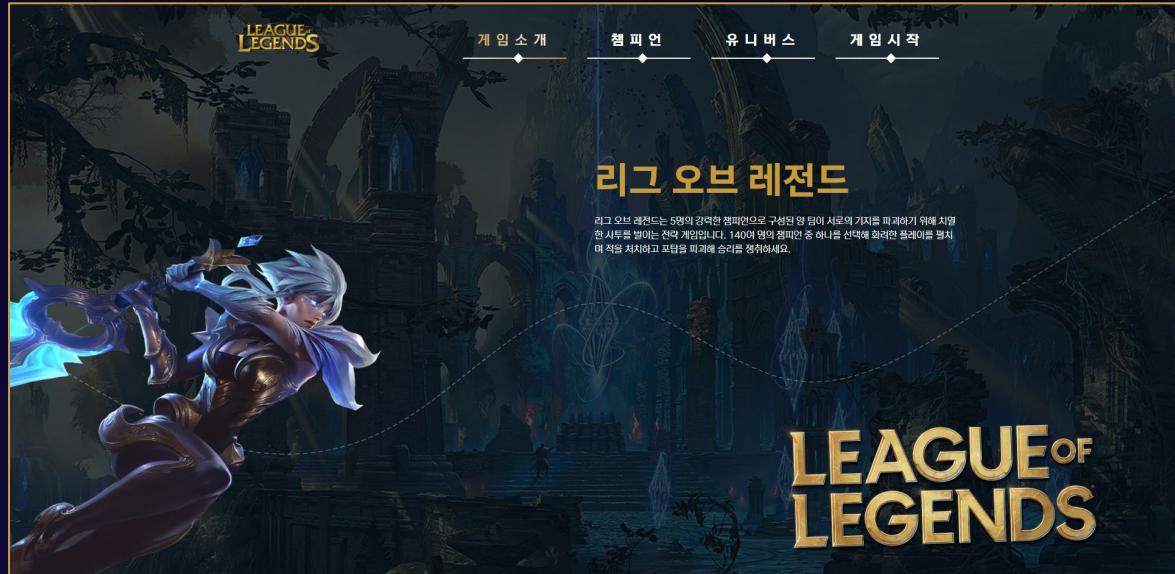


02

구현 및 코드



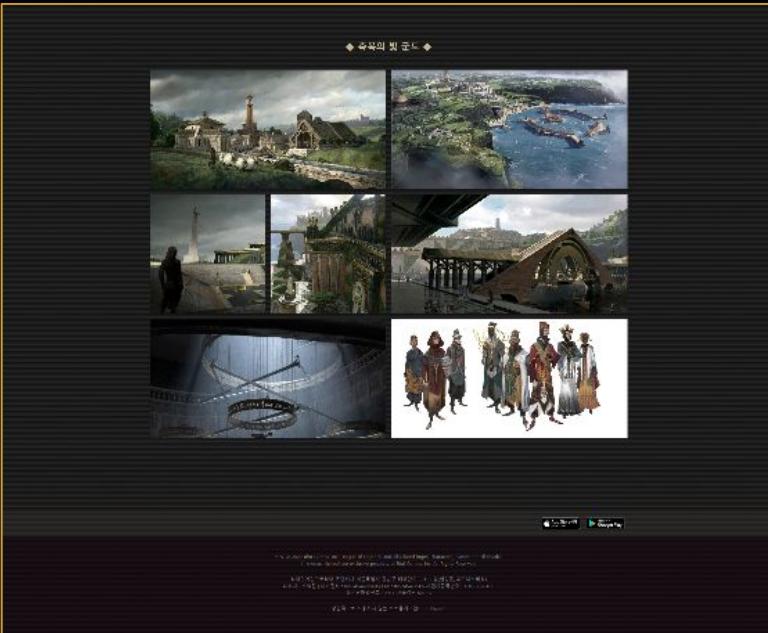
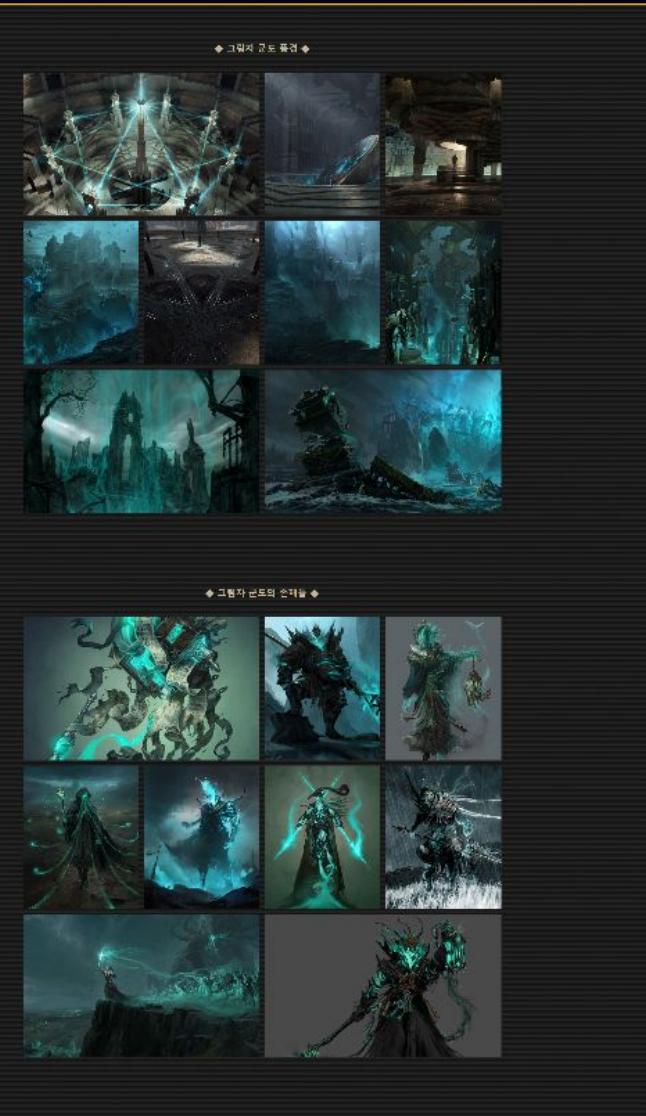
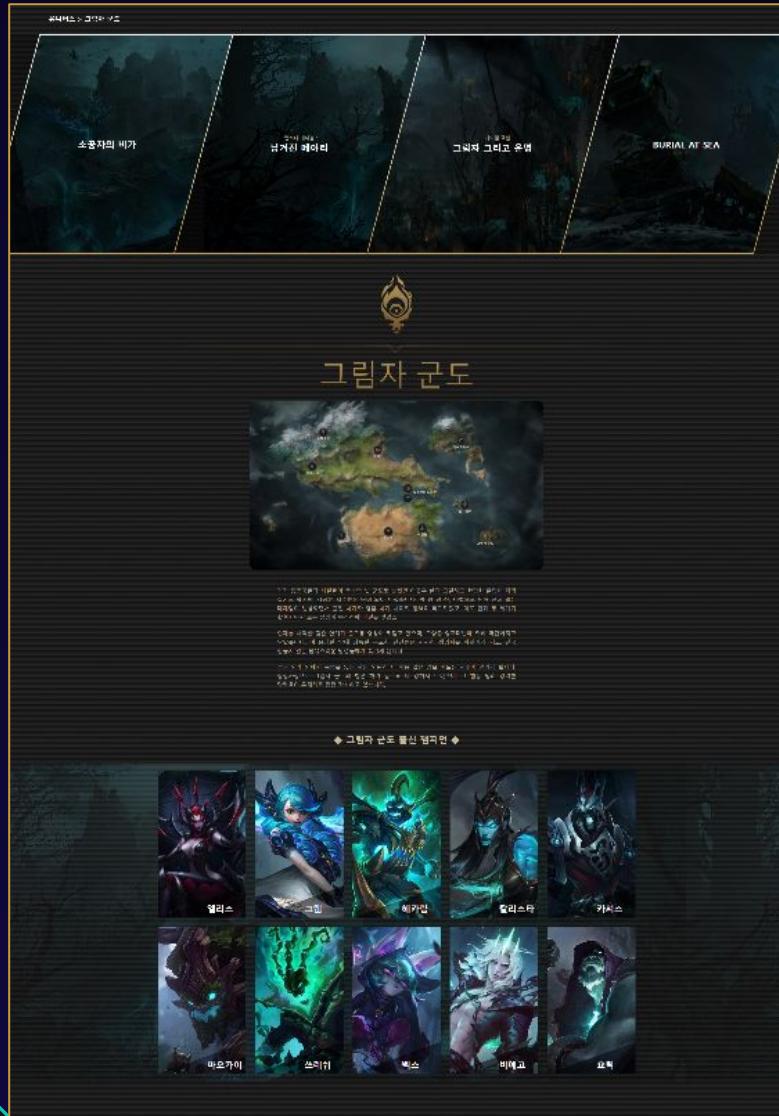
구현 화면 - Main



구현 화면 - Main



구현 화면 - *Sub*



소스코드 - *svg(path)*



```
<svg id="visual" preserveAspectRatio="none" viewBox="0 0 900 600" width="100%" height="calc(100vh - 100px)" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
  <!-- <path class="lineBg_b" -->
  d="M0 342L18.8 361.8C37.7 381.7 75.3 421.3 112.8 423.5C150.3 425.7 187.7 390.3 225.2 367.7C262.7 345
  fill="none" stroke-linecap="round" stroke-linejoin="miter" stroke="#A8A89E" stroke-width="2"></path>
  <!-- <path class="lineBg_a" -->
  d="M0 296L18.8 299.2C37.7 302.3 75.3 308.7 112.8 332C150.3 355.3 187.7 395.7 225.2 399.3C262.7 403.3
  fill="none" stroke-linecap="round" stroke-linejoin="miter" stroke="#C89B3C" stroke-width="1"></path>
  <path class="lineBg_b" -->
  d="M0 388L25 412.5C50 437 100 486 150 510.3C200 534.7 250 534.3 300 489.3C350 444.3 400 354.7 450 349
  fill="none" stroke-linecap="round" stroke-linejoin="miter" stroke="rgba(255,255,255,0.3)"
  stroke-width="1"></path>
  <path class="lineBg_c" -->
  d="M0 437L25 410C50 383 100 329 150 333C200 337 250 399 300 427.2C350 455.3 400 449.7 450 433.7C500 449
  fill="none" stroke-linecap="round" stroke-linejoin="miter" stroke="rgba(10, 200, 185, 0.3)"
  stroke-width="2"></path>
</svg>
```

svg 중 path를 사용하여 animation 효과를 부여했습니다. stroke-dasharray는 값이 작을 수록 간격이 좁아져 점선으로 표시되고, stroke-dashoffset은 선의 시작간격을 조절합니다. 점선으로 표시된 lineBg_b의 경우에는 array의 간격이 3으로 작고, 실선으로 보여지는 lineBg_c는 간격이 1000으로 크게 잡았습니다. 각각에 stroke-dashoffset을 변경하여 시작 간격을 변경하여 애니메이션 효과를 적용하였습니다.

```
.lineBg_b {
  stroke-miterlimit: 10;
  stroke-dasharray: 3;
  stroke-dashoffset: 1000;
  animation: dash 40s linear infinite;
}

.lineBg_c {
  stroke-miterlimit: 10;
  stroke-dasharray: 1000;
  stroke-dashoffset: 1000;
  animation: dash 5s linear infinite;
}

@keyframes dash {
  from {
    stroke-dashoffset: 1000;
  }
  to {
    stroke-dashoffset: 0;
  }
}
```

소스코드 - *svg(circle)*



```
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" width="600px" height="600px"
  viewBox="0 0 600 600" style="enable-background:new 0 0 600 600;" xml:space="preserve">
  <circle class="circle_strokea" style="fill:none;stroke:#c89b3c;stroke-width:2;stroke-miterlimit:10; stroke-dasharray: 100; stroke-dashoffset: 100px; "
    cx="308.973" cy="321.514" r="258.109" />
  <circle class="circle_strokeb" style="fill:none;stroke:rgba(255,255,255,0.2);stroke-width:8;stroke-miterlimit:10; stroke-dasharray: 100; stroke-dashoffset: 200px; "
    cx="308.973" cy="321.514" r="258.109" />
</svg>
```

svg 중 circle을 사용하여 animation 효과를 부여했습니다.
circle은 cx, cy, r를 설정해주어야 합니다. cx는 svg안에서의 원의 좌표 x를 의미하고, cy는 svg 안에서의 좌표 y를 의미합니다. 즉 (cx, cy)라고 볼 수 있습니다. r은 원의 반지름을 나타냅니다.

circle 앞의 path와 유사하게, stroke-dasharray는 값이 작을 수록 간격이 좁아져 점선으로 표시되고, stroke-dashoffset은 선의 시작간격을 조절했습니다.

svg circle animation에서는 각각의 애니메이션의 속도와 크기를 다르게 부여하기 위해서, 각각 다르게 작성했고, 효과도 ease-in-out, linear로 다르게 적용하였습니다.

```
.circle_strokea {
  stroke-miterlimit: 10;
  animation: circle_delaya 10s ease-in-out infinite alternate;
}

.circle_strokeb {
  stroke-miterlimit: 10;
  animation: circle_delayb 15s linear infinite alternate;
}

@keyframes circle_delaya {
  0% {
    stroke-dasharray: 300px;
  }

  100% {
    stroke-dasharray: 250px;
  }
}

@keyframes circle_delayb {
  from {
    stroke-dasharray: 250px;
  }

  to {
    stroke-dasharray: 100px;
  }
}
```



소스코드 - 타이핑 효과



리그 오브 레전드

리그 오브 레전드는 5명의 강력한 챔피언

리그 오브 레전드

리그 오브 레전드는 5명의 강력한 챔피언으로 구성된 양 팀이 서로의 기지를 파괴하기 위해 치열한 사투를 벌이는 전략 게임입니다. 140여 명의 챔피언 중 하나를 선택해 화려한 플레이를 펼치며 적을 치치하고 포탑을 파괴해 승리를 쟁취

```
// 메인 페이지 타이핑 효과
const text = "리그 오브 레전드는 5명의 강력한 챔피언으로 구성된 양 팀이 서로의 기지를 파괴하기 위해 치열한 사투를 벌이는 전략 게임입니다. 140여 명의 챔피언 중 하나를 선택해 화려한 플레이를 펼치며 적을 치치하고 포탑을 파괴해 승리를 쟁취"
const content = document.querySelector(".game_text_p")

content.innerText = "";

//sleep 함수 --> 잘못쓰면 모든 작동이 다 멈출 비동기식으로 Promise써서 해결해야함
function sleep(t) {
    return new Promise(resolve => setTimeout(resolve, t));
}

(async function () {
    while (true) {
        for (var i = 0; i < text.length; i++) {
            content.innerText = text.slice(0, i + 1);
            await sleep(100);
        }
        await sleep(2000);
        for (var i = text.length - 1; i >= 0; i--) {
            content.innerText = text.slice(0, i);
            await sleep(20);
        }
        await sleep(100);
    }
})();
```

게임에 대한 설명을 하는 글에 타이핑 효과를 적용했습니다. 작성되어야 하는 글을 상수 text에 값을 대입하고, .game_text_p 클래스에 넣어줍니다. 처음에는 sleep를 date를 gettime()를 사용해서 받아서 일정한 시간만큼 멈추고 실행되게 구현했지만, 잠시 멈춘 시간 동안 사이트의 모든 작동이 멈추는 문제가 있었습니다. 이를 promise를 사용하여 비동기식으로 구현해서 잠시 멈췄다가 지워지고, 다시 타이핑 되는 과정을 표현할 수 있었습니다.

소스코드 - *clip-path*



```
.region {  
    width: 100%;  
    height: 100%;  
    /* border: 1px solid #C89B3C; */  
    border: 2px solid transparent;  
    border-image: linear-gradient(to right bottom, #c89b3c 0%, #fff 100%);  
    border-image-slice: 1;  
    background-color: rgba(0, 0, 0, 0.8);  
    position: relative;  
    overflow: hidden;  
    transition: all 0.9s ease-in-out;  
    clip-path: polygon(0% 0%, 100% 0, 100% 0, 100% 100%, 0% 100%);  
    /* clip-path: polygon(50% 0, 100% 0, 100% 100%, 0 100%, 0 0); */  
}
```



```
.overlay {  
    position: absolute;  
    width: 100%;  
    height: 100%;  
    background-color: rgba(0, 0, 0, 0.7);  
    transition: all 0.9s ease-in-out;  
    clip-path: polygon(50% 10%, 100% 0, 100% 100%, 0 100%, 0 0);  
}  
  
.region:hover .overlay {  
    background-color: rgba(0, 0, 0, 0);  
    opacity: 0;  
    clip-path: polygon(50% 10%, 100% 0, 100% 100%, 0 100%, 0 0);  
    transition: all 0.9s ease-in-out;  
}  
  
.region:hover {  
    clip-path: polygon(50% 10%, 100% 0, 100% 100%, 0 100%, 0 0);  
    transition: all 0.6s ease-in-out;  
    border: none;  
}
```

clip-path를 사용하여 직사각형 모양을 수정했습니다. clip-path는 요소를 원하는 모양으로 자를 수 있도록 해줍니다.

clip-path에 hover 효과를 주면 같은 모양이라도 꼭지점의 위치에 대한 값에 따라서 눈에 보여지는 효과가 생길 수 있습니다. 또한 clip-path를 hover를 사용하기 위해서는 hover 효과를 사용하기 전의 원래의 클래스에도 clip-path에 대한 값을 지정해 주어야 합니다. 이때, 꼭지점의 개수를 서로 다르게 지정했다면, 원하는 결과 값이 나오지 않을 수 있기에, 꼭지점의 개수를 같게 부여해야 합니다.



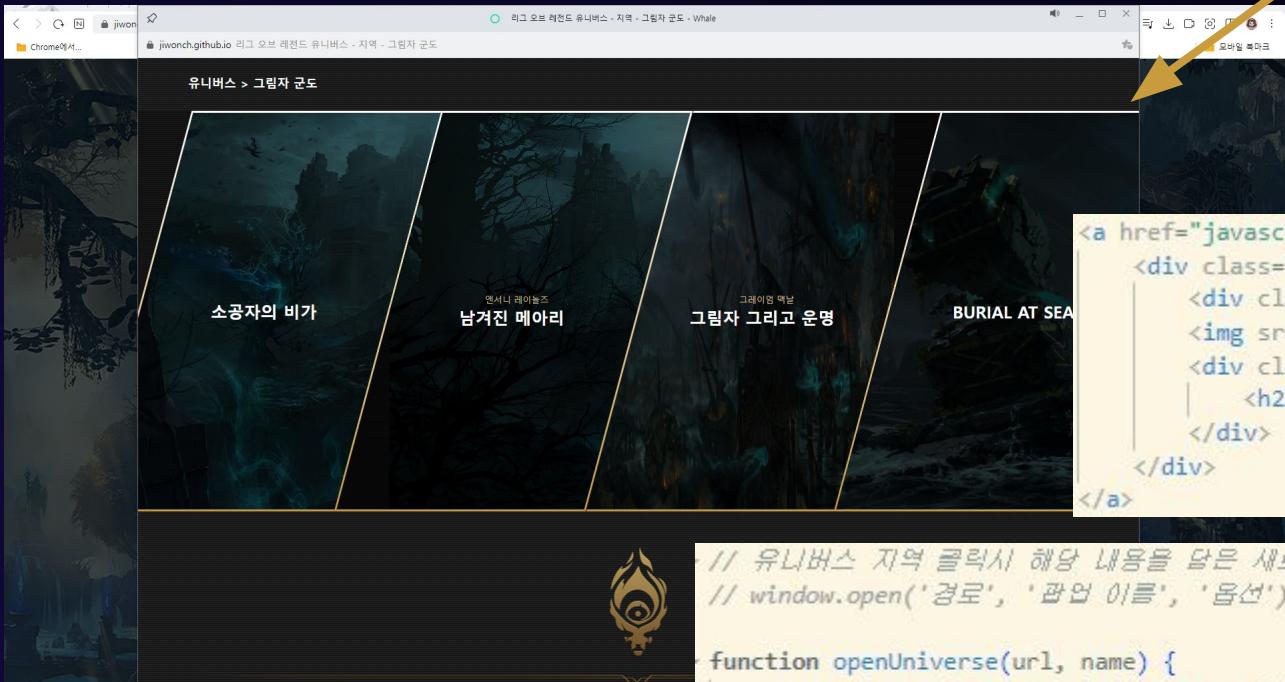
소스코드 - *window.open*



CLICK!



*window.open*을 사용하여, *a* 태그로 묶여 있는 박스 클릭 시에, 새로운 창이 열리도록 구현했습니다.

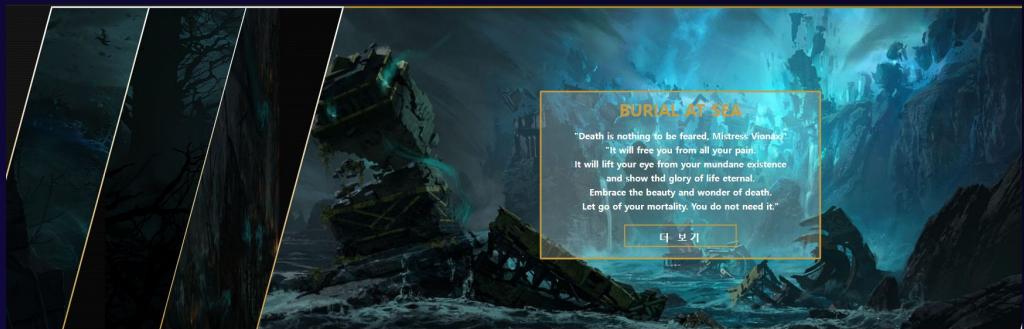
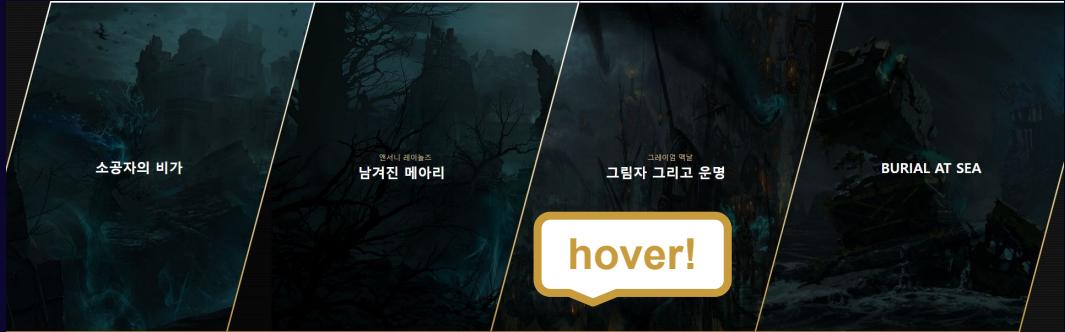


```
<a href="javascript:openUniverse('./universe_popup_shadow-isles.html', '그림자 군도');">
  <div class="region shadow-isles">
    <div class="overlay"></div>
    
    <div class="regionName">
      <h2>그림자군도</h2>
    </div>
  </div>
</a>
```

```
// 유니버스 지역 클릭시 해당 내용을 담은 새로운 팝업창 뜰
// window.open('경로', '팝업 이름', '옵션');

function openUniverse(url, name) {
  var option = 'top=100, left = 200, width=1500, height = 900, status=no, menubar=no, toolbar=no, resizable=yes';
  window.open(url, name, option);
}
```

소스코드 - skew-flex

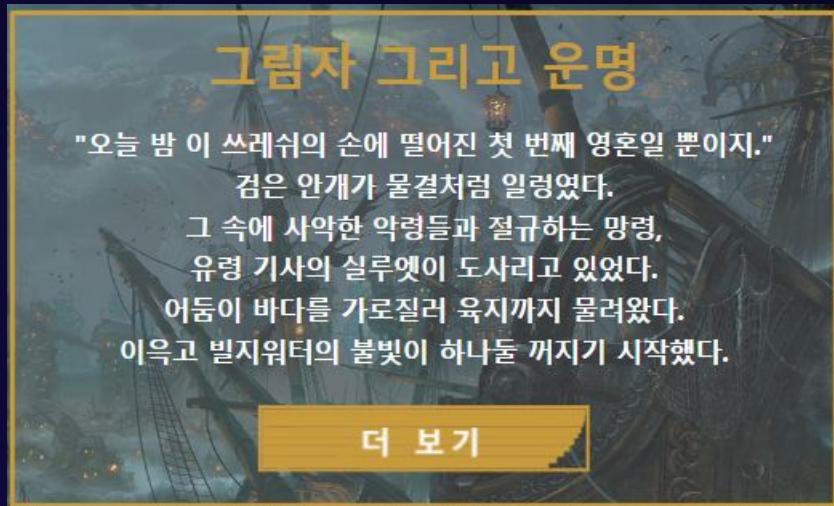
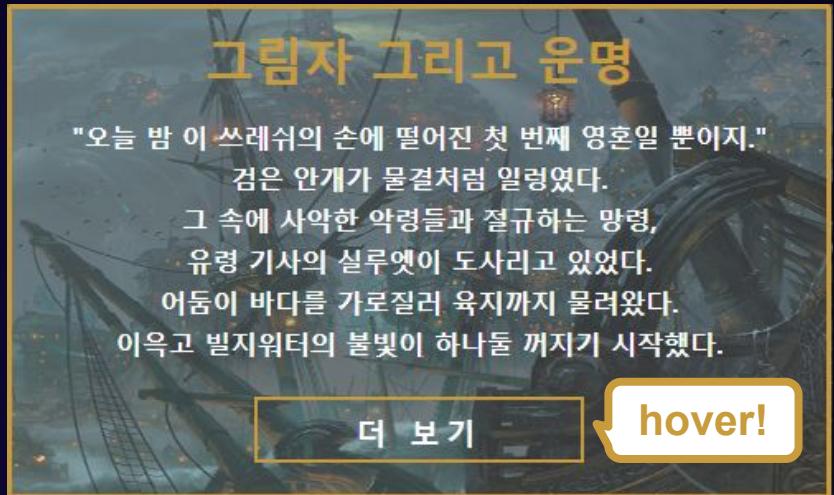


```
.skew_box {  
    border: 1px solid #fff;  
    height: 100%;  
    width: 25%;  
    transform: skewX(-15deg);  
    flex: 1;  
    border: 3px solid transparent;  
    border-image: linear-gradient(to top, #c89b3c 0%, #fff 100%);  
    border-image-slice: 1;  
    overflow: hidden;  
    transition: all .4s;  
    position: relative;  
}  
  
.skew_box:hover {  
    flex-grow: 8;  
    border: 3px solid transparent;  
    border-image: linear-gradient(to bottom, #c89b3c 0%, #fff 100%);  
    border-image-slice: 1;  
}
```

박스를 skewX를 사용해 세로에 대한 기울기를 변경했습니다. 일반 상태에서는 flex grow를 1로 주어 4개의 박스가 같은 비율로 보여지게 했습니다. hover 했을 시에, hover한 박스만 flex-grow를 8로 변경하여, 커지게 구현했습니다.



소스코드 - button effect



```
i {  
    content: '';  
    width: 100%;  
    height: 100%;  
    transition: all .5s;  
    background: #c89b3c;  
    transform: scaleX(0);  
}  
  
i:nth-child(1) {  
    transition-delay: .02s;  
}  
  
i:nth-child(2) {  
    transition-delay: .04s;  
}  
  
i:nth-child(3) {  
    transition-delay: .06s;  
}  
  
i:nth-child(4) {  
    transition-delay: .08s;  
}  
  
i:nth-child(5) {  
    transition-delay: .10s;  
}  
  
i:nth-child(6) {  
    transition-delay: .12s;  
}  
  
i:nth-child(7) {  
    transition-delay: .14s;  
}  
  
i:nth-child(8) {  
    transition-delay: .16s;  
}  
  
i:nth-child(9) {  
    transition-delay: .18s;  
}  
  
i:nth-child(10) {  
    transition-delay: .2s;  
}  
  
.over_btn i {  
    transform-origin: top left;  
}  
  
.over_btn:hover i {  
    transform: scaleX(1);  
}
```



button에 hover할 시, 세로로 i를 10개 배치해서, 0.02초 간격으로 delay되어 배경이 transition 되도록 구현했습니다.

가장 처음의 i는 scaleX가 0이기 때문에 보여지지 않고, hover 할 시에, 1로 변경 되어 보여 지기 시작합니다.

소스코드 - map blink



```
.map_marker {  
    position: absolute;  
    width: 22px;  
    height: 22px;  
    border-radius: 50%;  
    opacity: 0.4;  
    right: 148px;  
    bottom: 84px;  
}  
  
.map_marker::before {  
    content: "";  
    width: 100%;  
    height: 100%;  
    border-radius: 50%;  
    background-color: #rgba(200, 155, 60, 0.7);  
    position: absolute;  
    /* left: 50%;  
    top: 50%;  
    margin-top: -3px;  
    margin-left: -3px; */  
    animation: dot 1.2s infinite;  
}  
  
@keyframes dot {  
    100% {  
        opacity: 1;  
        transform: scale(3.5);  
        opacity: 0;  

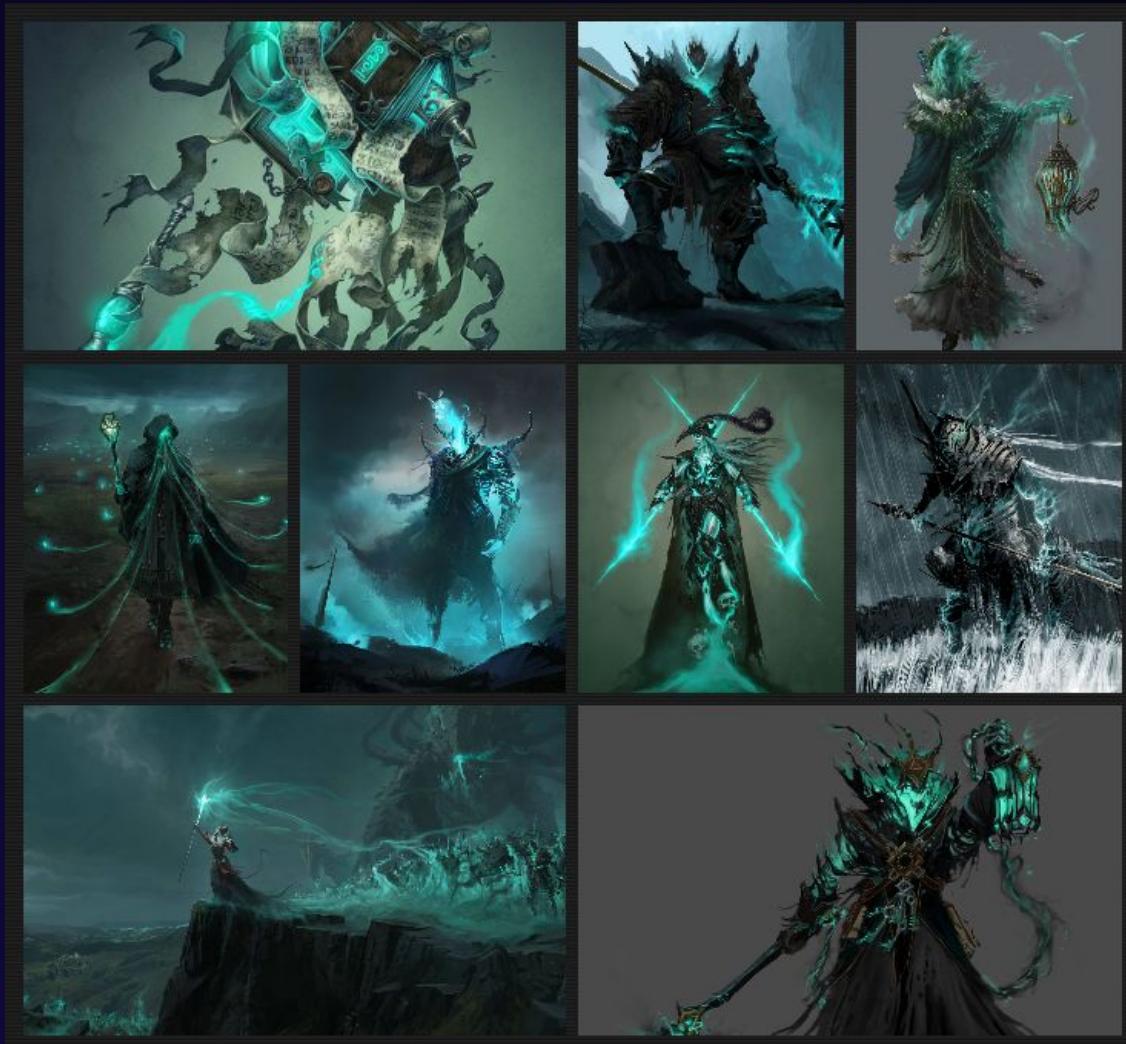
```

map에서 해당 위치를 maker 해주기 위해서, 원형 박스를 만들어 주고, before를 사용해서 애니메이션 효과가 작동할 박스를 위에 얹어 주었습니다.

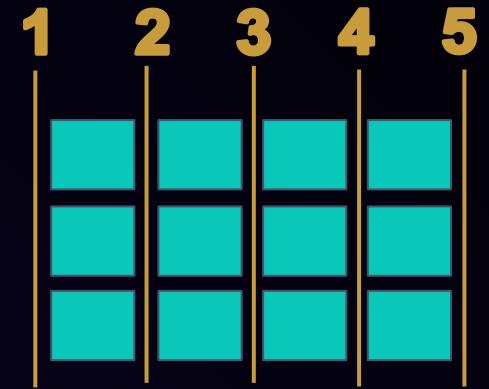
애니메이션 효과는 opacity가 1이어서 보여지면서 scale이 3.5만큼 될 때까지 커지다가 다시 opacity가 0이 되면서 보여지지 않는 상태가 되는 것을 반복합니다.



소스코드 - grid



```
.uni_scene {  
    width: 1300px;  
    height: 1200px;  
    margin: 0 auto;  
    display: grid;  
    grid-template-columns: repeat(4, 1fr);  
    gap: 15px;  
    margin-bottom: 100px;  
}  
  
.scene_item {  
    transition: all 0.3s ease-in-out;  
    position: relative;  
}  
  
.scene_item:nth-child(1),  
.scene_item:nth-child(8) {  
    grid-column: 1/3;  
}  
  
.scene_item:nth-child(9) {  
    grid-column: 3/5;  
}
```



이미지에 대한 박스를 grid를 사용해서 배치해보았습니다.
grid-template-columns: repeat(4, 1fr);을 사용하여, 박스가 열을
4개씩 같은 비율로 반복되게 설정했습니다.
열을 4개식 배치하게 되면 1박스2박스3박스4박스5의 형식으로
구성되게 되는데, 박스가 가지는 가로 범위를 2배로 변경하고 싶은
경우에는, 위에서의 첫 번째 박스 처럼, grid-column: 1/3; 을
사용하여 2배로 늘려주었습니다.



03

평가 및 후기



평가 및 후기

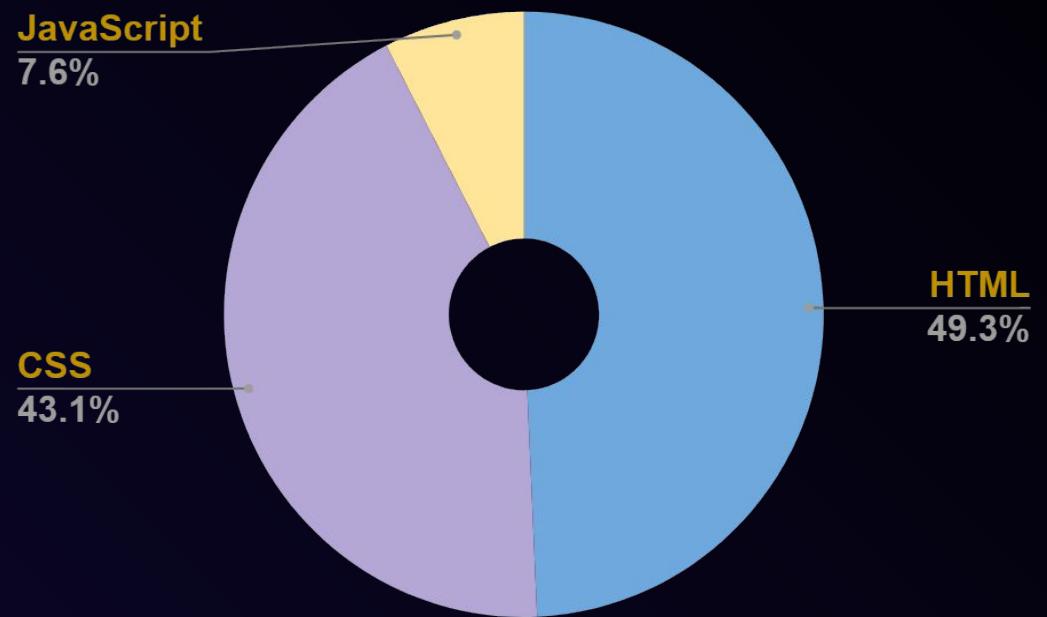


게임 사이트는 대부분 3D효과나 다양한 애니메이션 효과로 보는 눈을 재미있게 만들어주는 경우가 많다고 생각합니다.

분위기에 맞는 애니메이션 효과나 svg를 사용한 이미지 애니메이션을 구현하고 싶었지만, 원래 기획할 때 가지고 있던 생각만큼은 구현이 되지 않은 것 같습니다.

전의 포트폴리오에서는 쓰지 않았던, svg나, clip-path, grid, typing효과 등을 사용해보았습니다. 몇 가지는 쉽게 이해되서 금방 적용한 것도 있었고, 이해하는데 오래 걸려서 고민에 고민을 거쳐서 구현한 것도 있습니다. 아직 부족한 것이 많다고 생각도 들고, 알던 것들도 개념을 더 잘 정리해서 잘 사용하는 것도 중요하다는 생각이 많이 들었습니다.

다음 포트폴리오에서는 조금 더 자연스러운 애니메이션 효과나, 지금까지 쓰던 코드들을 더 깔끔하고 효율적으로 쓰기 위해서 구상 과정부터 지금까지 보다도 꼼꼼하게 확인하고 고민해야 하겠다는 생각이 들었습니다. 새로운 방법도 익히고, 적용을 위해 고민하는 시간을 가졌으면 하는 바람입니다.





게임을 종료하시겠습니까?

감사합니다.

최지원 - 리그 오브 레전드 포트폴리오