CONTENTS

PART 03 BUBBLE.IO 버블 시작

PART 04 버블 에디터 뜯어보기

CONTENTS

PART 01

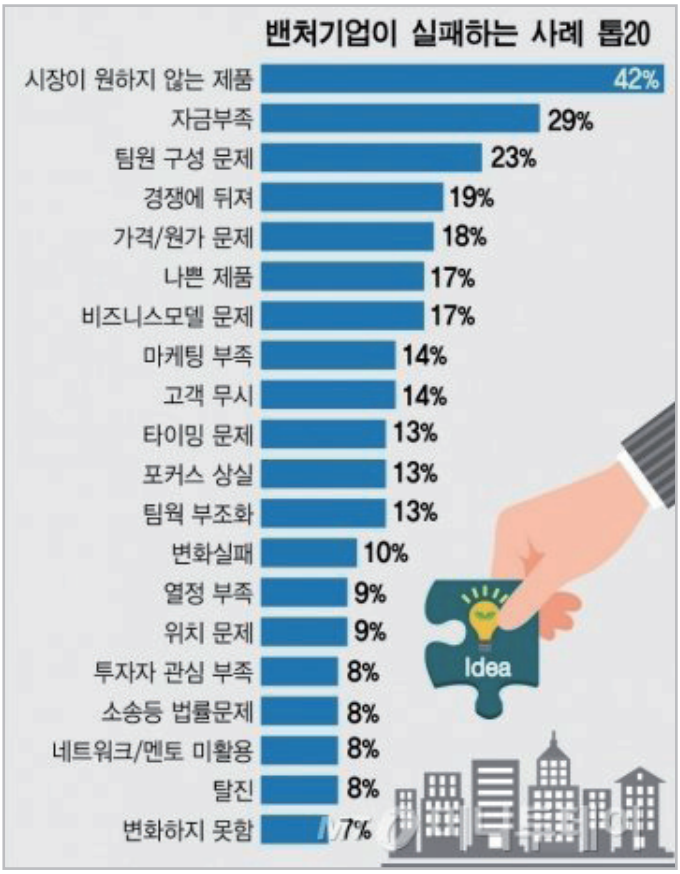
바야흐로 대 노코드 시대

01 CHAPTER 빠르게 변하는 시장에서의 노코드

서비스를 만들기로 한 여러분, 창업이 성공하는 방법은 과연 무엇일까요? 사실, 창업이 성공하는 방법은 정답이 없습니다. 각각의 아이템마다 성공한 이유는 모두 다르기 때문입니다.

하지만, 창업이 실패하는 이유들은 몇 가지로 좁혀지게 됩니다. 다시 말해, 창업이 실패하는 이유들만 잘 살피고 대응하면 망할 위험은 현저히 줄어든다는 얘기입니다.

그럼 창업이 실패하는 이유와 함께 그것들을 피할 방법을 같이 알아봅시다.



글로벌 시장조사업체인 CB INSIGHT가 조사한 바에 따르면, 벤처기업이 실패하는 가장 큰 이유는 시장이 원하지 않는 제품을 내놨거나 자금이 부족했기 때문입니다.

그리고 노코드가 점점 각광을 받는 이유는 이 실패 원인을 최소화해 줄 수 있기 때문입니다.

과연 어떻게 노코드가 이러한 원인을 해결시켜줄 수 있을까요? 일단 시장이 원하지 않는 제품을 해결하는 방법은 바로 빠른 변화입니다. 현대 시대에서 소비자들의 마음은 갈수록 빠르게 변화하고 있습니다. 이에 맞춰 제품도 시장과 소비자들에 맞춰 빠르게 변해야 합니다. IT시대에 상품을 변화시키는 방법은 바로 개발일 것입니다. 여태까지 우리는 이 개발을 "코딩"으로 진행해왔습니다. 하지만 "노코드"가 등장한 이후로 판도가 바뀌고 있고 그 흐름을 여러분들이 타야 합니다.

물론, 규모가 이미 커진 사업들은 정교한 서비스를 위해 수 많은 개발자들이 투입되어 코딩을 이용한 개발 및 유지보수가 필요할 가능성이 큽니다.

하지만 예비 창업자나 초기 창업자처럼 현재 시장에서 잘 작동하는 상품일지 아닐지 끊임없이 테스팅해야 하는 단계에서는 얘기가 다릅니다. 기능을 최대한 빠르게 개발하고 시장에 내놓은 후, 고객 반응을 참고하여 다음 개발을 계속해서 진행해야 합니다. 시장이 원하는 제품이 될 때까지 말입니다. 바로 이때, 노코드를 사용해 개발한다면 기존 코딩을 위해 배워야 하던 많은 부분을 건너뛰고 오롯이 프로덕트를 개발하고 검증하는 데 집중할 수 있게 됩니다.

개발자 출신인 필자가 감히 희망을 드리자면 코딩으로 서비스를 만드는 데 걸리는 시간보다 노코딩으로 서비스를 만드는 데 걸리는 시간이 1/10에 불과하다는 점입니다. 금쪽같은 시간을 1/10이나 아낄 수 있다는 것은 사업을 하는 데 있어 큰 경쟁력이 아닐 수 없습니다.

어떤가요? 이제 노코딩을 배워야겠다는 확신이 조금은 드나요?

아직도 코딩으로 개발을 해야겠다고 생각하시는 분이 있다면 다음 장에서 코딩과 노코딩을 조금 더 깊게 비교해 보겠습니다.

02 CHAPTER 코딩 vs. 노코딩

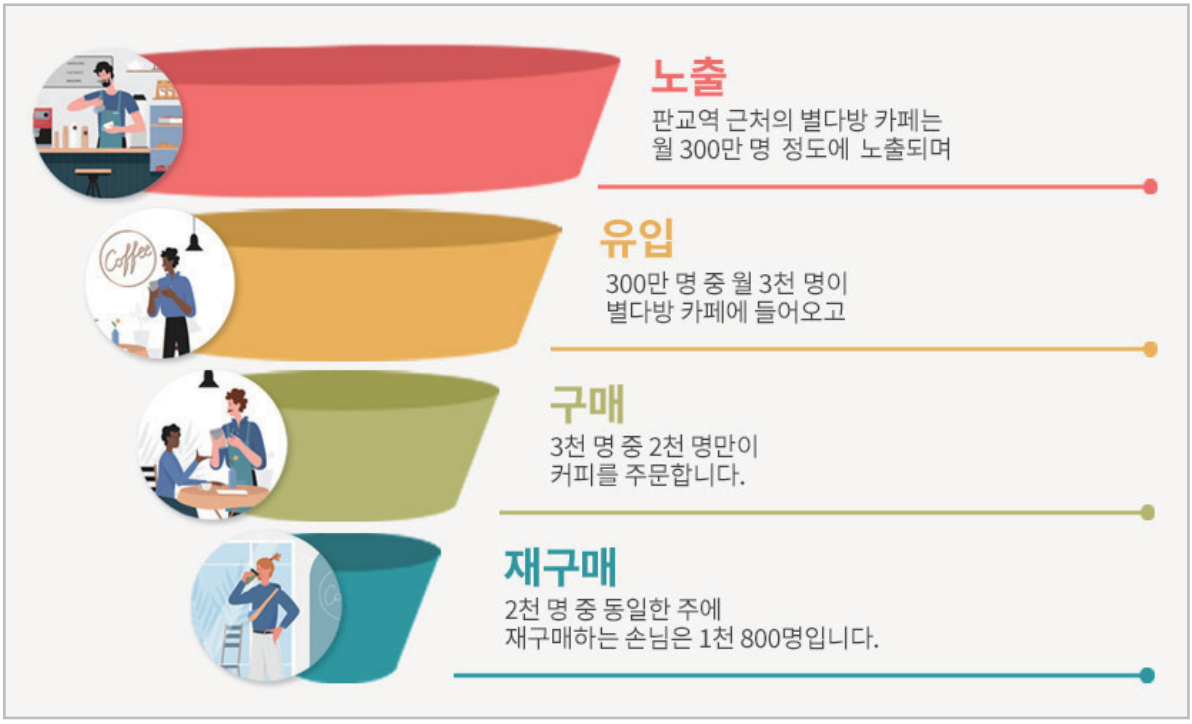
앞서 말했던 것처럼 여러분이 배워야 하는 건 코딩 능력이 아닙니다. 즉 아름다운 코드를 짜라는 게 아니라, 원하는 기능을 정확하게 구현해 내라는 말이 더 정확한 표현이라고 할 수 있겠네요.

사실 필자는 여러분들이 개발자가 되는 걸 바라지 않습니다. 개발자보다는 창업가 혹은 기업가가 되고 싶은 분들이 많았으면 좋겠습니다.

물론 개발자도 매우 흥미로운 직업입니다. 검은 화면에 영어로 알아듣지 못한 명령어들을 적다보면 번듯한 홈페이지, 번듯한 앱을 바로바로 만들어내는 마술사들 같기 때문입니다.

그렇지만 필자가 실제 서비스를 운영해보니 개발은 정말 창업이나 사업의 시작점 그 자체인 것 같습니다. 말 그대로 "창업"을 하고 싶은 사람에게 인터넷 시대에서의 개발은 최소, 기본, 조건이라는 이라는 것입니다.

진짜 게임은 그 이후에 어떻게 사람을 데려오느냐, 어떻게 그들을 설득해 구매까지 유도하느냐, 어떻게 우리 제품이 생각나서 다시 돌아오게 하느냐 등등 완수해야 할 미션들은 무궁무진합니다.



그러니 개발에만 매몰되면 시작도 못하고 끝나는 셈입니다. 여러분들은 최대한 개발을 빨리 완료하고 시장에 나가서 부딪혀 보는 게 가장 중요합니다.

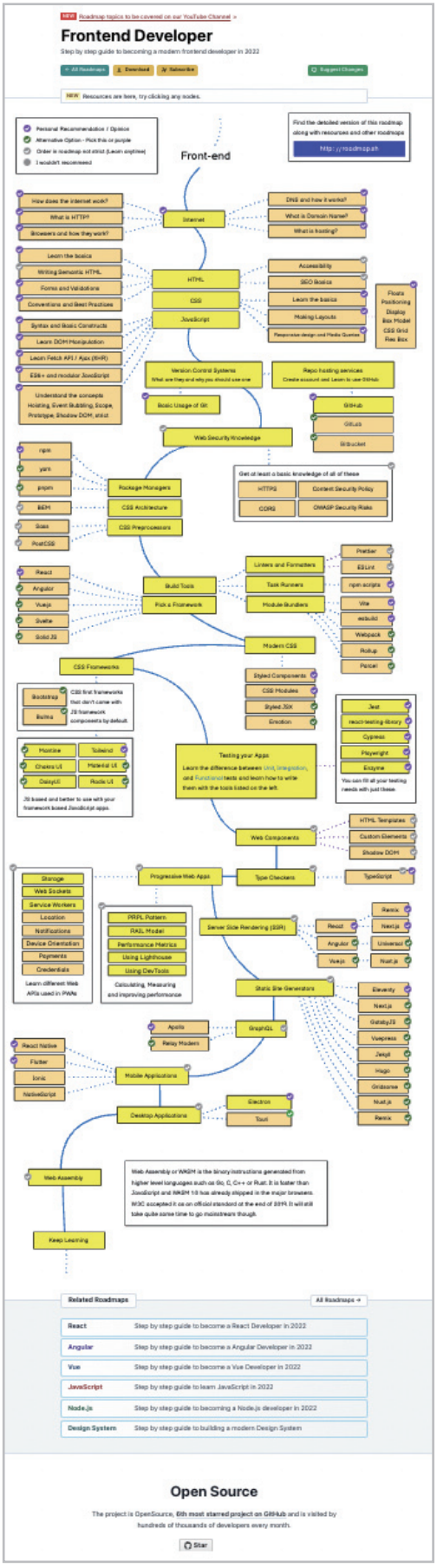
그냥 개발자를 고용해서 만들게 할 생각이신 분도 분명 있으실 겁니다. 그렇지만 아무리 개발자를 고용할지라도 개발의 'ㄱ'자를 알고 지시하는 것과 모르고 지시하는 건 천지 차이입니다.

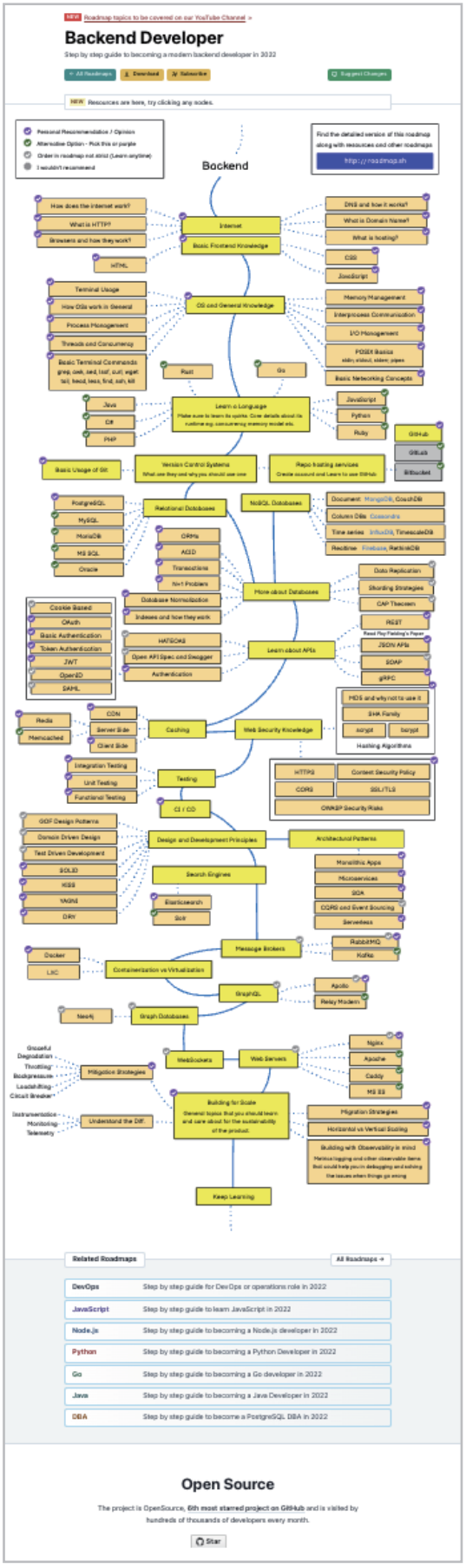
그렇다면 코딩을 이용해 개발하는 법과 노코딩을 이용해 개발하는 법의 차이를 낱낱이 파헤쳐 봅시다.

1. 코딩

인터넷상에서 서비스를 만들 땐 크게 두 가지 분야로 나뉩니다. 사용자가 마주하는 화면과 그와 상호작용하는 기능을 넣는 프론트엔드, 서비스가 사용하는 데이터들을 관리하는 백엔드가 있습니다.

코딩에 조금이라도 발을 들여 놓았다면 한 번쯤은 들어봤을, (웹 개발자가 되기 위한) 프론트엔드, 백엔드의 로드맵을 살펴보겠습니다.

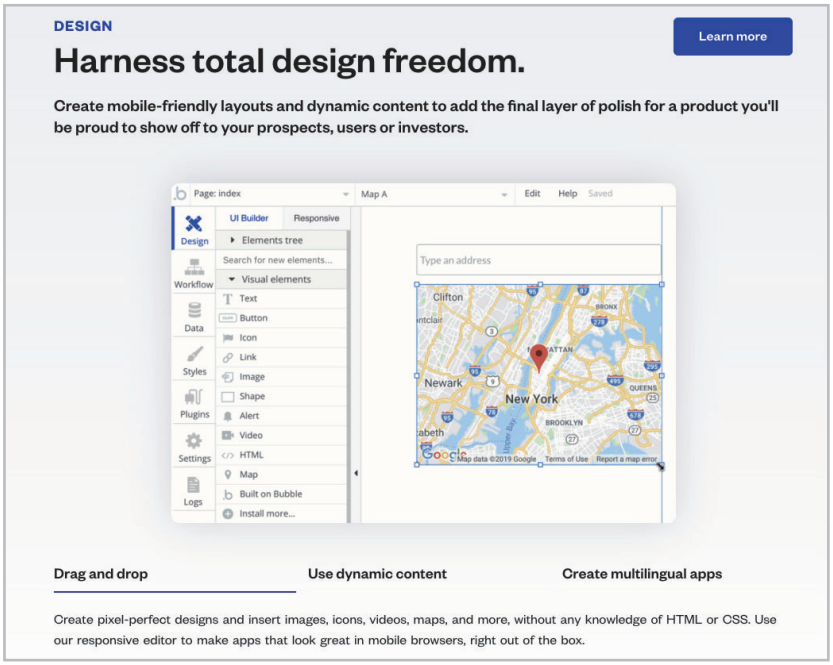

Frontend Developer
Step by step guide to becoming a modern frontend developer in 2022
Front-end
Related Roadmaps
React
Angular
Vue
JavaScript
Node.js
Design System
Open Source


Backend Developer
Step by step guide to becoming a modern backend developer in 2022
Backend
Related Roadmaps
DevOps
JavaScript
Node.js
Python
Go
Java
DBA
Open Source

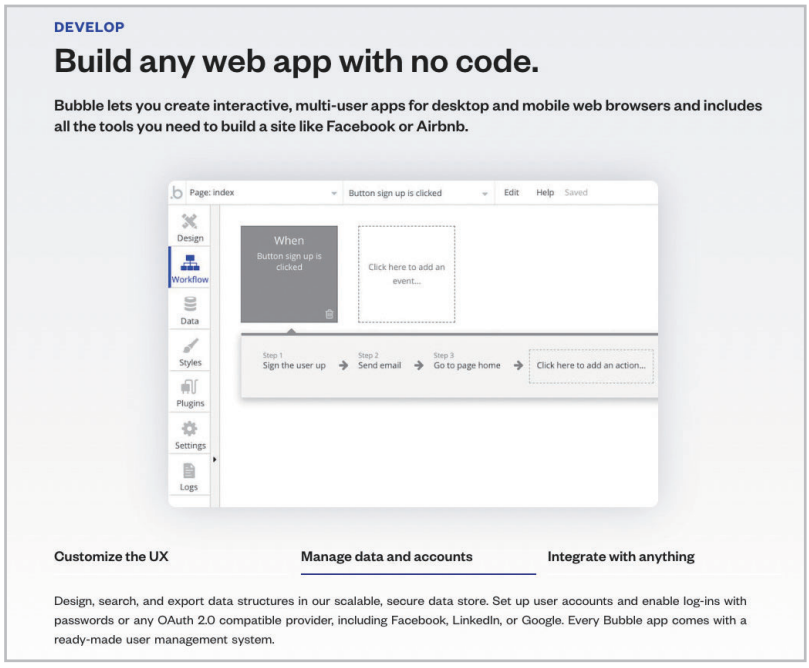
사실 더 깊게 들어가면 프론트엔드와 백엔드 외에도 데브 옵스, 보안 등 웹 개발을 위해 알아야 할 코딩 지식들은 정말 무궁무진합니다.

2. 노코딩

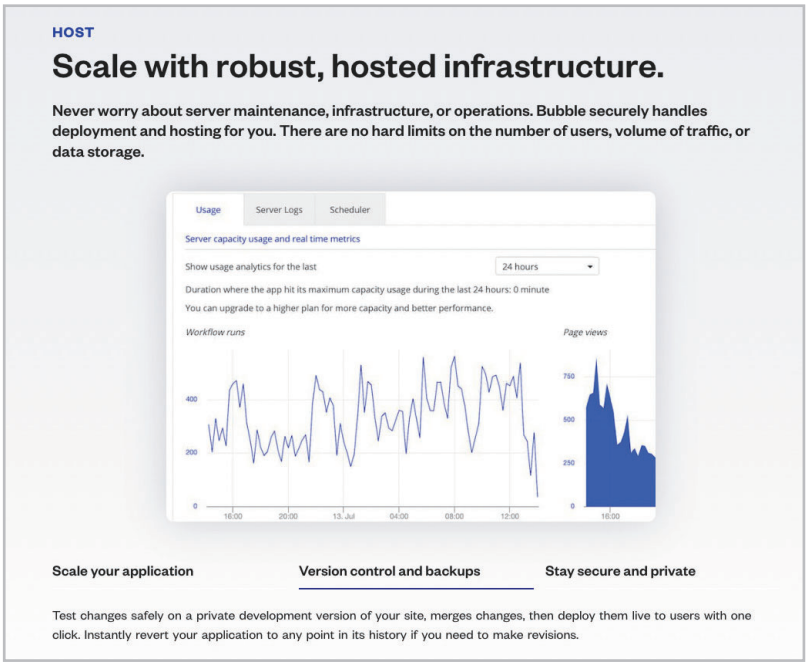
"코드 없이" 서비스할 웹 사이트나 앱의 화면을 그립니다.



"코드 없이" 각각의 요소에 알맞은 기능을 부여합니다.



“코드 없이” 배포하고 관리합니다.



어떤 방법이 더 빠르고 유연해 보이나요? 아무리 다시 봐도 노코딩이 더 쉽고 빨라 보입니다.

필자가 이렇게 말했다고 코딩이 아예 필요 없다는 말은 절대 아닙니다. 왜냐하면 추후 크게 스케일업할 경우에는 코딩으로 귀결될 수 있기 때문입니다. 노코드도 결국 코딩으로 만들어진 기술입니다. 따라서 고도화를 하려면 노코딩 밑의 단계인 코딩까지 내려가야 최적화를 시킬 수 있습니다. 하지만 그 정도 서비스면 이미 시장검증이 끝난 뒤, 유능한 개발자를 고용할 정도의 자금은 충당이 되어있을 겁니다.

그러니 걱정하지 말고 확신을 갖기 바랍니다.
“노코딩으로 사업을 시작하면 시장검증을 빠르게 할 수 있다.”
“노코딩으로 사업하면 확장성이 떨어진다.”

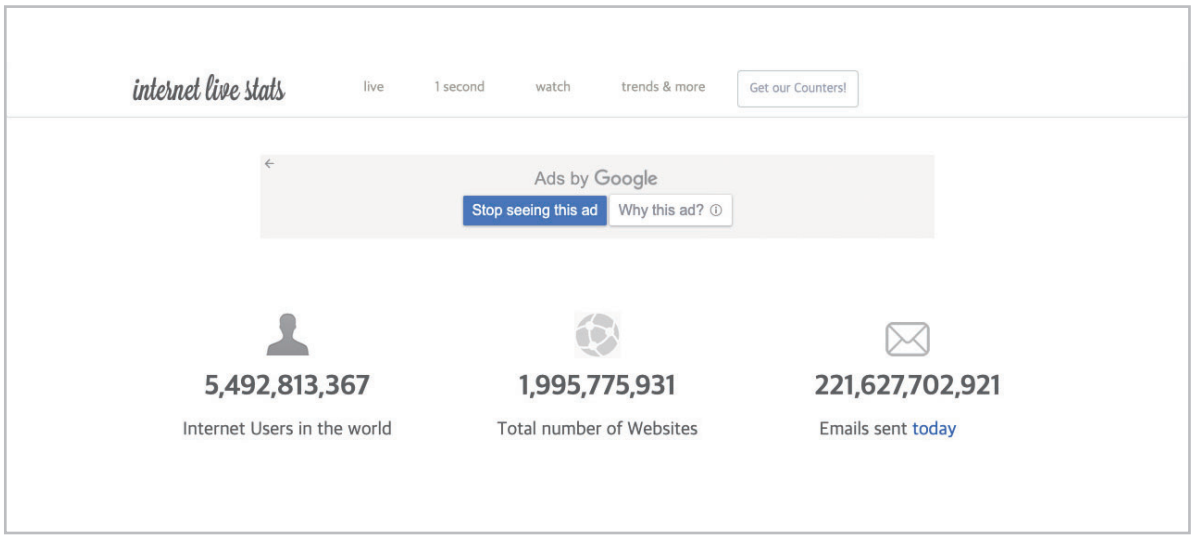
03 CHAPTER 웹을 만들어야 할 까? 앱을 만들어야 할 까?

온라인 비즈니스를 시작하려 할 경우, 크게 웹 서비스로 진행할지 앱 서비스로 진행할지 많이들 고민합니다 (간단한 랜딩 페이지 + 문의하기 형태의 MVP는 제외하겠습니다).

우선, 누구나 어렴풋이 알고 있겠지만 앱은 핸드폰에 특정 파일을 설치해서 구동시키는 형태이고, 웹은 인터넷에 특정 주소를 입력해 접속하는 형태입니다.

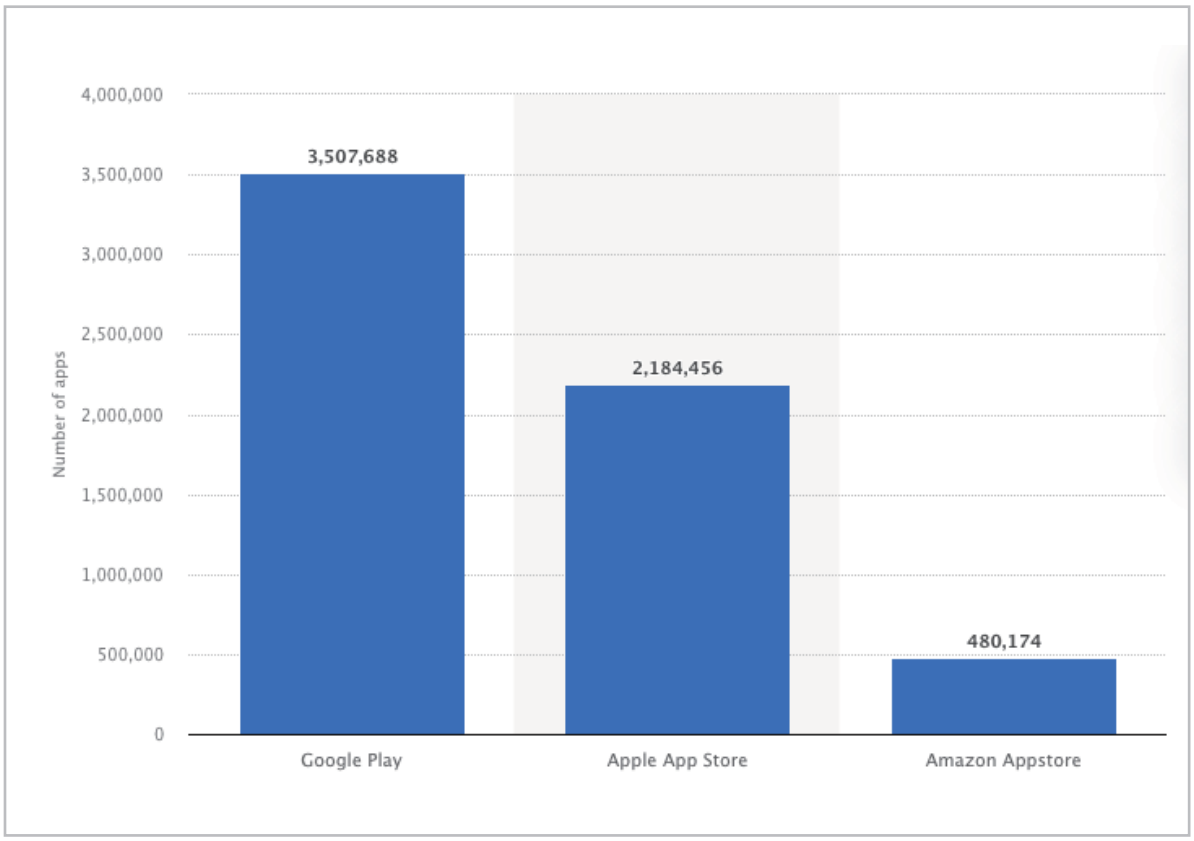
물론 서비스 특성마다 웹이 적합한지 앱이 적합한지 달라지겠지만, 개발 비용과 초기창업 측면에서 비교를 통해 개인적인 추천을 하도록 하겠습니다.

상식적으로 생각해보면 우리 주변에는 사이트가 앱보다 훨씬 많습니다. 현재 전 세계에 있는 총 사이트의 개수는 19억 9,577만 5,931개입니다(202년 11월 기준). 현재도 이 숫자는 초단위로 늘어나고 있습니다.



앱의 수도 찾아보겠습니다.

가장 보편적인 앱 마켓인 앱스토어와 플레이스토어에 배포된 앱은 총 569만 2,144개입니다.



언뜻봐도 웹 사이트의 개수가 압도적으로 많습니다. 이 말이 뜻하는 바는 앱 개발자가 웹 개발자가 상대적으로 적다는 뜻입니다. 희소한 자원은 가격이 오르기 마련입니다. 앱 개발이 웹 개발보다 비싼 이유도 개발을 공급하는 입장에서 앱 개발자가 상대적으로 희소하기 때문입니다.

하지만 웹과 앱의 비용 격차는 제작 이후에도 나타나게 됩니다. 일단 웹을 만들고 나서 불특정다수에 노출되기 위해서는 인터넷 세상에 한 공간을 부여받아 그 공간에 우리의 사이트를 올려놔야 합니다. 이를 호스팅이라고 하고 호스팅 비용은 초기 사용자가 별로 없을 경우나 MVP단계일 경우에는 월 평균 5,500원으로도 충분합니다.

상세 사양 비교

사양명	절약형	일반형	비즈니스	퍼스트클래스	자이언트	자이언트플러스
월 사용료 (VAT 포함)	500원	1,100원	5,500원	11,000원	22,000원	33,000원
설치비 (VAT 포함)	5,000원	11,000원	11,000원	11,000원	11,000원	11,000원
초과 트래픽 요금 (VAT 포함)	-	-	-	-	-	165원/GB

상품 사양

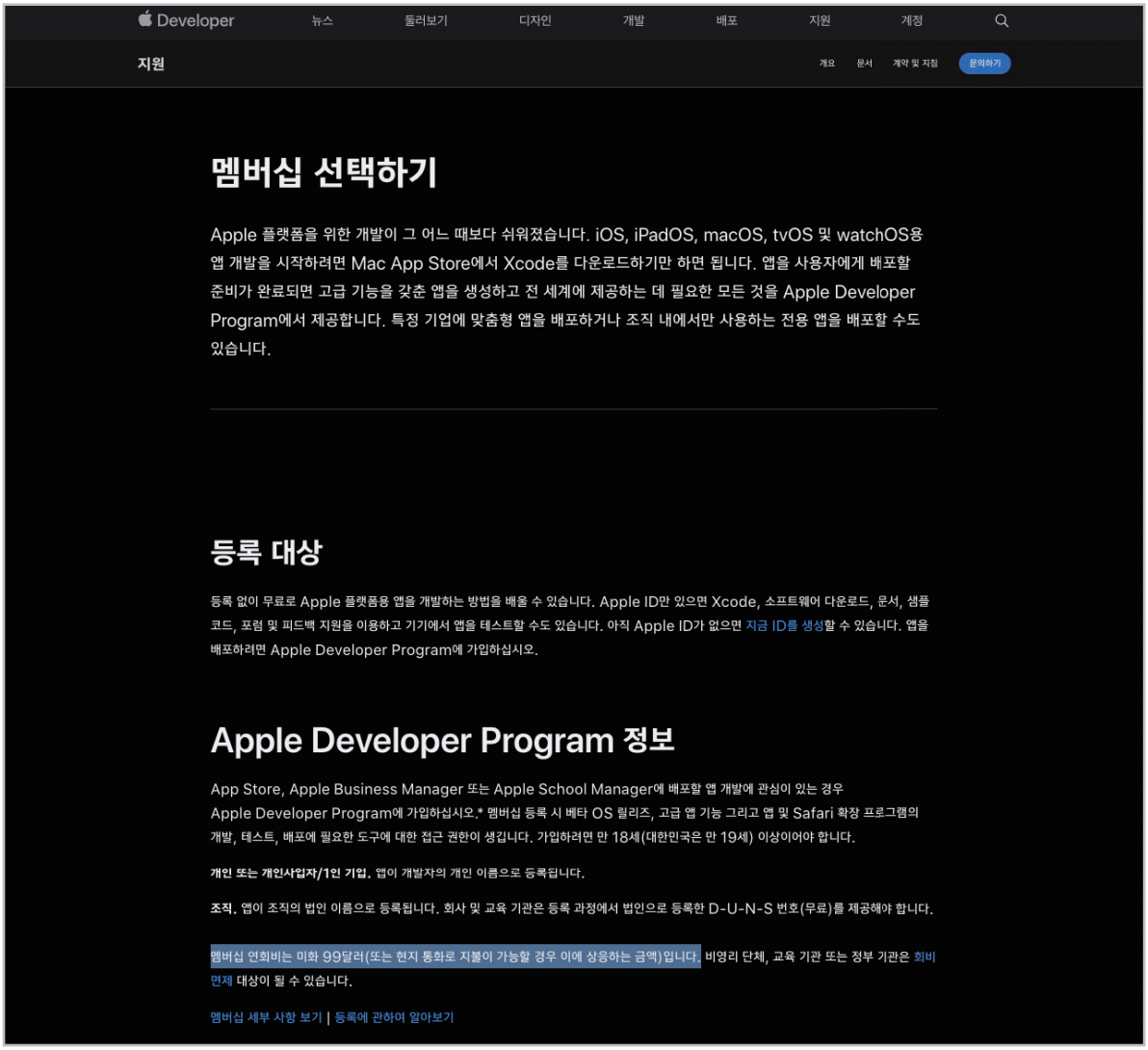
웹용량	500MB	1GB	3GB	6GB	10GB	10GB
동영상 스트리밍 웹용량	100MB	200MB	500MB	1GB	2GB	2GB
이미지 CDN 웹용량	100MB	200MB	500MB	1GB	2GB	2GB
웹 트래픽 용량	800MB	1.5GB	3.5GB	7.5GB	14GB	500GB/월
동영상 스트리밍 트래픽 용량	400MB	500MB	1.5GB	3.5GB	10GB	10GB/일
이미지 CDN 트래픽 용량	400MB	500MB	1.5GB	3.5GB	10GB	10GB/일
DB 용량	서버 공간 내 무제한	서버 공간 내 무제한	서버 공간 내 무제한	서버 공간 내 무제한	서버 공간 내 무제한	서버 공간 내 무제한

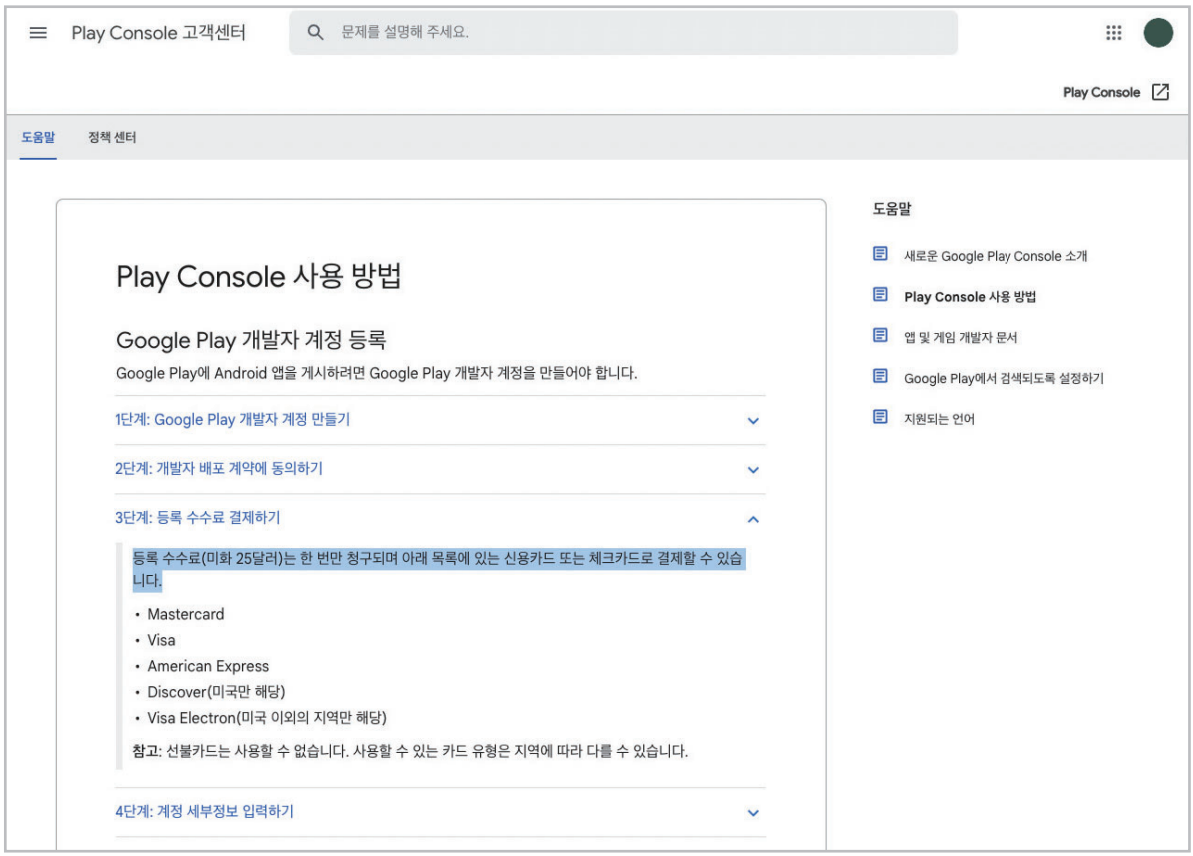
그리고 네이버를 들어가고 싶은 사람들이 “naver.com”을 입력해서 찾아가듯이 우리 사이트에도 간판과 같은 주소가 필요합니다. 이를 도메인 주소라고 하고 도메인도 평균 22,000원이면 구매 가능합니다.

가장 인기있는 도메인!

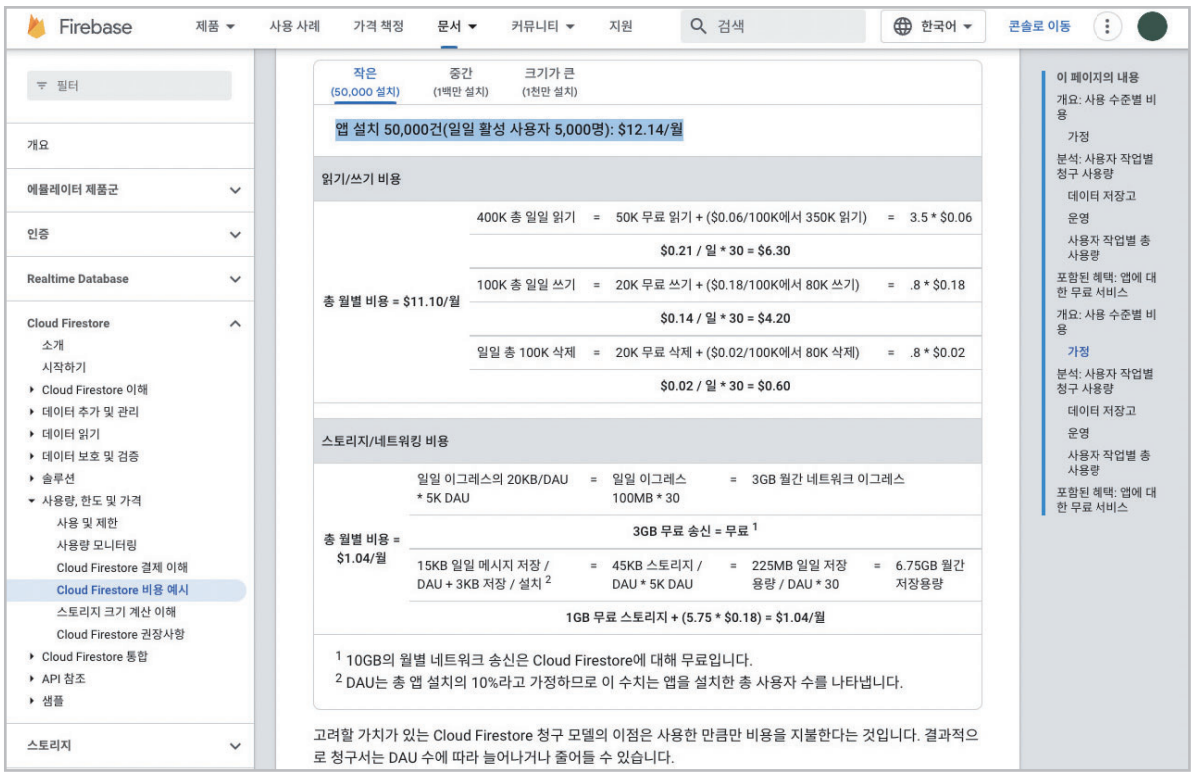
.com 23,500원/1년	EVENT .shop ~~46,750원/1년~~ 550원/1년	.net 22,000원/1년	.biz 22,000원/1년
.org 22,000원/1년	.co.kr 22,000원/1년	.kr 22,000원/1년	.or.kr 22,000원/1년

반면, 앱은 제작 후에 앱스토어와 플레이스토어에 입점하려면 각각 연간 141,376원, 35,680원이 듭니다 (2022년 11월 환율 기준). 여기서 벌써 호스팅 비용과 도메인 구매 비용을 넘어설 확률이 큽니다.

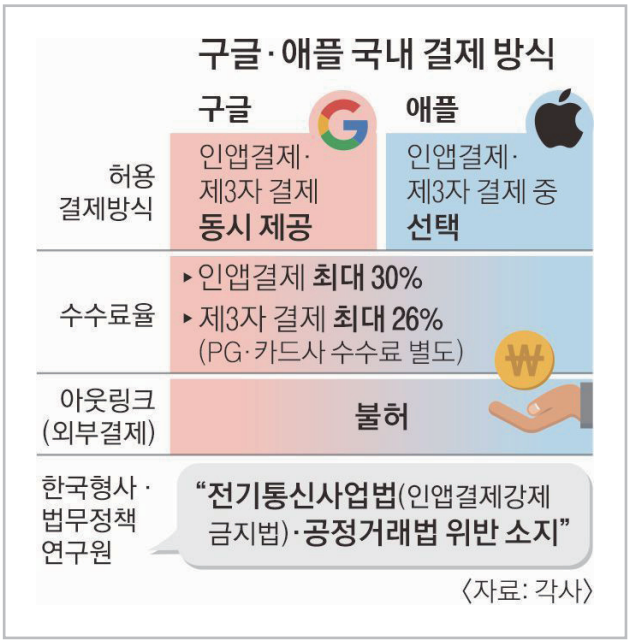




심지어 추가로 서버와 통신해야 하는 앱일 경우 따로 서버비가 발생합니다.



여기에 더해 앱에서 결제를 받으면 매출의 30%를 앱스토어에 지불해야 합니다(인앱 결제 수수료). 그래서 앱스토어들이 폭리를 취한다는 기사가 매번 나오기도 합니다.



물론 앱만이 누릴 수 있는 것들도 있습니다. 앱은 설치해놓은 (알림 설정을 동의한) 사용자들에게 푸쉬알림을 보낼 수 있습니다. 이를 통해 재방문, 재구매를 유도하기 편리합니다. 그리고 앱스토어나 플레이스토어의 광고 효율은 여전히 높다고 합니다.

하지만 최근에는 웹에서도 푸쉬알림을 보낼 수 있는 기능이 생겼고 문자, 카카오톡(한국 한정)으로 리텐션을 유도하는 메시지를 보낼 수 있기 때문에 아예 대체 불가능한 기능이 아니긴 합니다.

더욱이 우리는 검증된 아이템을 개발하는 게 아닙니다. 우리가 만든 서비스가 실제 고객들이 필요로 하는 아이템인지, 충분한 수요가 있는 시장이 있는지를 검증해야 하는 단계입니다. 만약 시장과 제품의 핏(PMF)이 맞지 않으면 사라질 가능성도 있습니다.

그래서 더더욱 위에 설명한 비용을 감당하면서까지 초기에 앱을 제작 하는 건 낭비입니다. 그러니 여러분이 온라인 창업, 온라인 서비스를 시작할 예정이라면 웹을 먼저 만들어서 사람들을 모으고, 시장 반응을 살피며 발전시켜 나가야 합니다. 앱 제작은 그 이후라도 늦지 않습니다.

웹 제작은 앞으로 배울 버블로 간편하게 개발 가능할 뿐만 아니라, 앱으로 변환시키는 것도 가능하니 끝까지 주목해주시기 바랍니다.

04 CHAPTER 노코드를 활용한 서비스 제작 로드맵

자, 그럼 노코드로 서비스를 제작하려면 어떤 절차를 밟아 나가야 하는지 로드맵을 살펴보겠습니다.

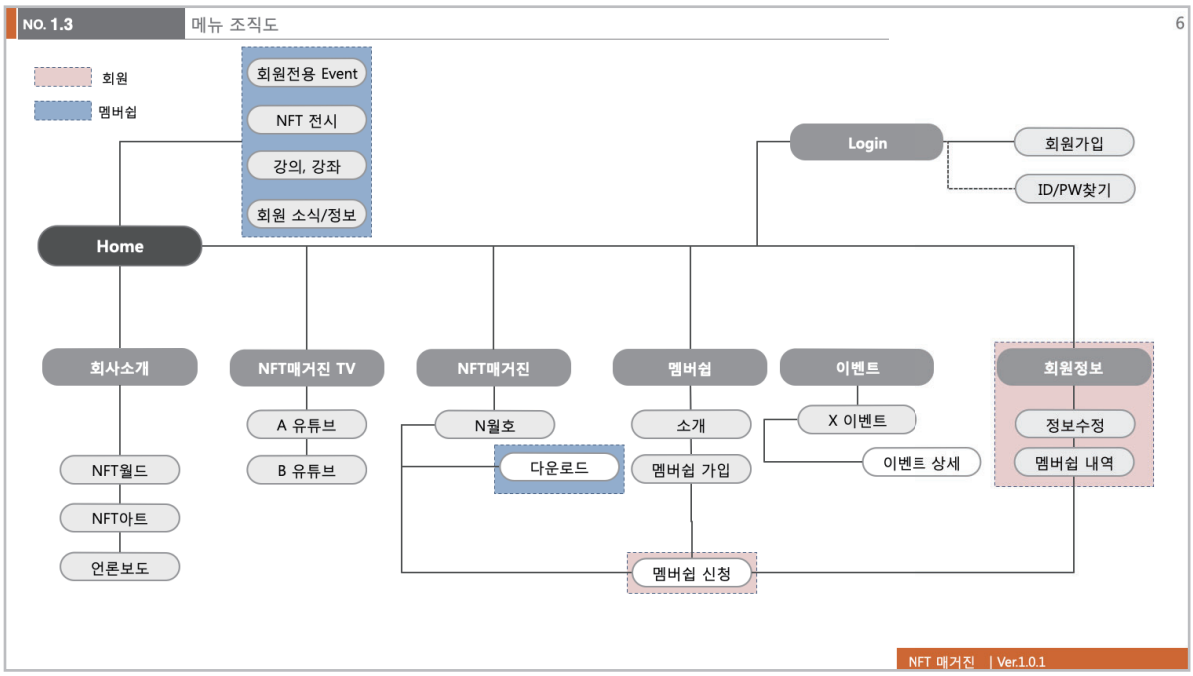
참고로 제작 이전에 어떤 서비스를 만들지 계획하는 것과 제작 이후에 서비스를 실제로 운영하는 것은 여기서 다루지는 않습니다.

하나의 서비스를 만드려면 크게 3단계 STEP이 있습니다.
기획, 디자인, 개발이 바로 그것입니다.

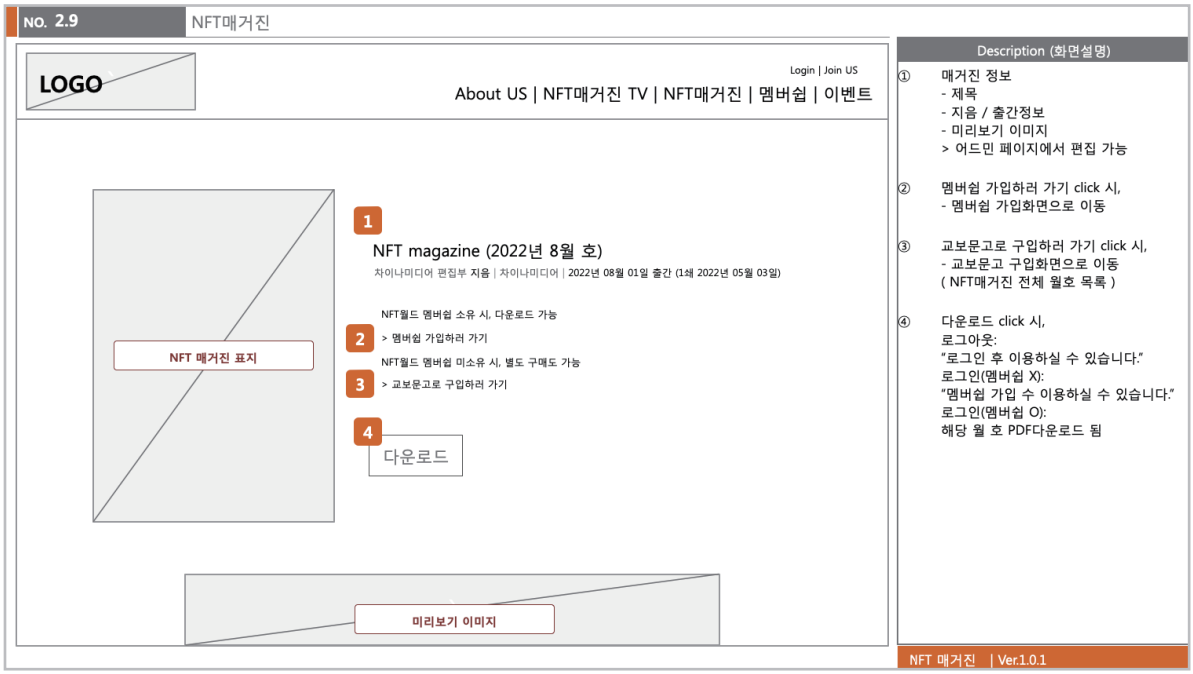
1. 기획

웹 또는 앱에 필요한 화면과 기능들을 기획하는 단계입니다. 실제 업무 프로세스에서는 이 과정도 상당히 길고 복잡합니다. 하지만 여기에서는 간단하게 필수적인 부분만 짚고 바로 제작으로 들어가는 것을 지향할 예정입니다.

우선, 서비스를 이용하기 위한 필수 기능들을 나열합니다. 기능들을 나열하다 보면 필요한 화면들이 나옵니다. 서비스의 구조를 보기 위해 화면 구조도를 정리합니다.



구조가 머리 속에 들어왔으면 해당 화면들을 하나하나 그립니다. 벤치마킹할 서비스나 유사 서비스를 참고해서 추가할 화면이 있는지 확인 작업이 추가되면 서비스의 질이 올라가기 때문에 추가로 조사해도 좋습니다.

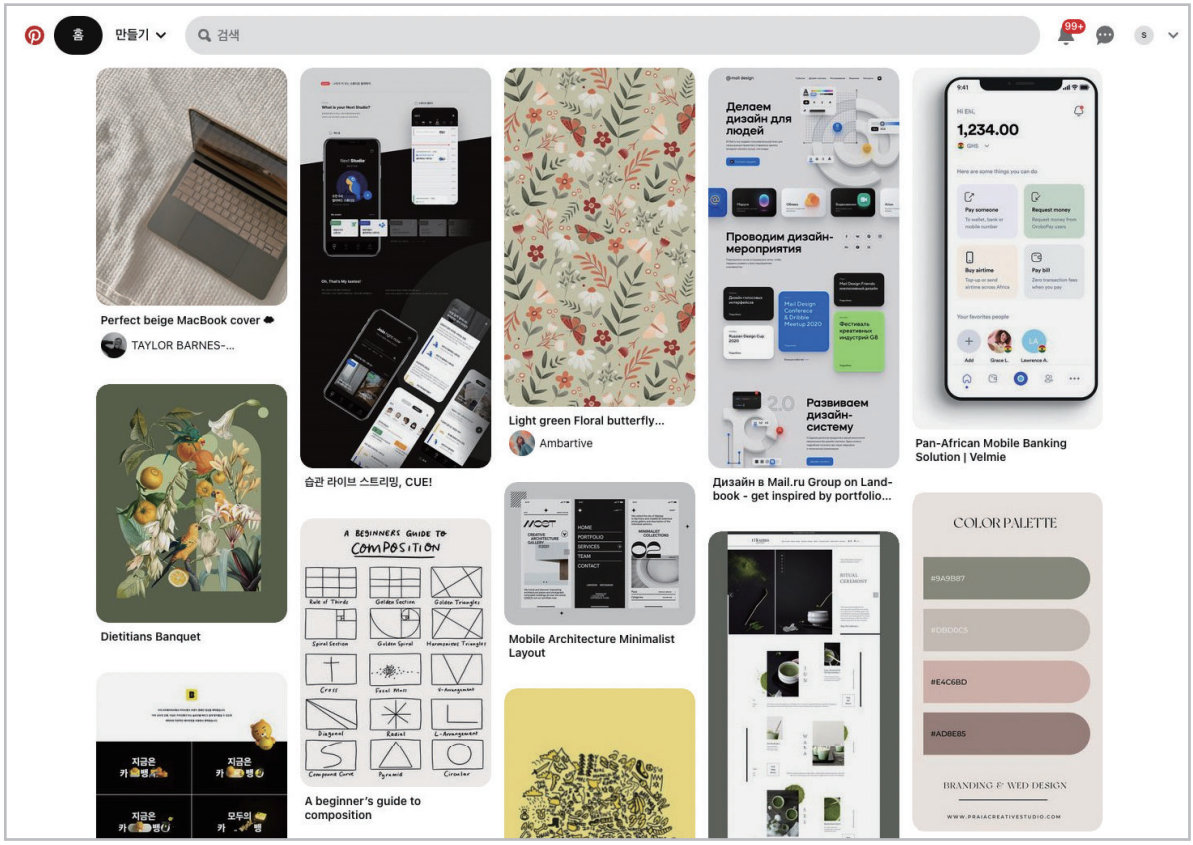


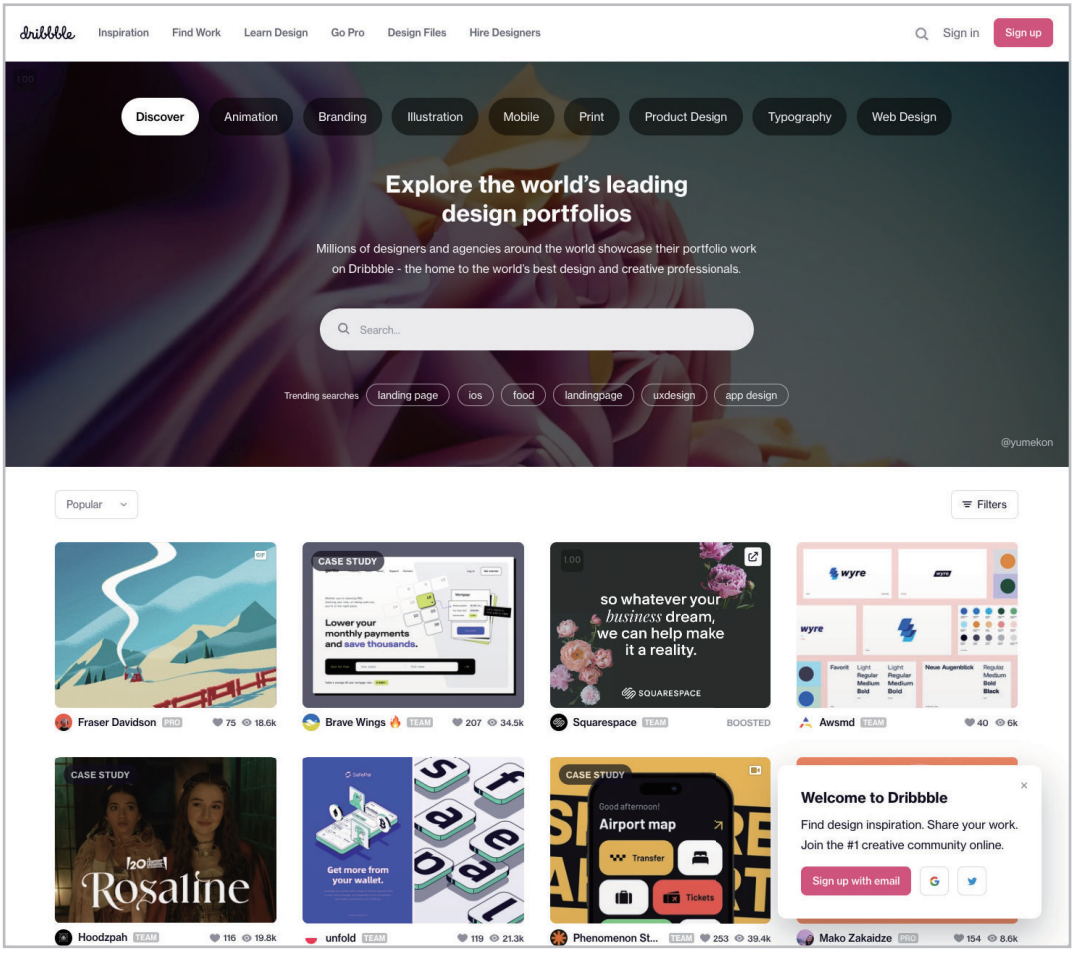
그 후에는 각 페이지별로 필요한 요소들과 그 내용에 대한 세부 사항을 정리하는 스토리보드를 작성합니다. 이때 버튼이나 입력창 같은 요소는 눌렀을 때 어떤 기능이 실행되어야 하는지 같이 기록해두어야 향후 개발을 진행할 때 자연스럽게 프로젝트가 진행될 수 있습니다.

2. 디자인

그 다음 단계는 지금까지 만든 화면 뼈대에 예쁘게 살을 붙이는 작업입니다.

화면을 예쁘고 아름답게 만들기 위해서는 전문적인 디자인 기술이 들어가야 합니다. 다양한 영감을 위해서 핀터레스트나 드리블과 같은 디자인들을 모아놓는 사이트를 참고해보시기 바랍니다. 둘러보다 보면 좋은 아이디어가 많이 떠오르게 마련입니다.





이에 더해 시간을 단축시키기 위해 손쉽게 템플릿을 가져오거나 기본 제공 스타일을 변형시켜 사용하는 방법을 배우겠습니다. 자세한 내용은 이후에 나올 '버블로 화면 그리기' 부분에서 언급됩니다.

추후 사업자금이 생기거나 리뉴얼을 할 때는 전문 UI/UX 디자이너에게 맡기는 것도 고려할 만합니다. 전문가의 손길을 거친 결과물은 언제나 배울 점이 많기 때문입니다.

3. 개발

노코드를 활용한 서비스 제작 로드맵의 마지막 단계는 만들기, 즉 개발입니다. 서비스를 만들기 전에 어떤 툴로 만들어야 할지 정한 뒤에 개발을 시작해야 합니다. 여기에서는 기존 개발자들이 서비스를 만드는 방식을 노코딩으로 재현한 버블(bubble.io)을 사용할 예정입니다.

다음 챕터에서 노코드 툴 버블이 무엇인지 소개하고, 수많은 노코드 툴 중에서 왜 콕 집어 버블(bubble.io)을 배워야 하는지 살펴본 뒤 서비스 개발을 시작해 보겠습니다.

PART 02

노코드 툴
최강자 버블

01 CHAPTER 전 세계에서 주목하는 노코드 툴 Bubble.io

개발자들이 기술을 선택할 때 가장 중요하게 생각하는 것 중 하나가 바로 "공신력"입니다.

공신력이란 말 그대로 공적인 신뢰를 받을 만한 능력을 뜻합니다. 개발 기술이나 프레임워크가 공신력을 가지기 위해서는 몇 가지 필요한 조건들이 있습니다.

공신력을 입증할 만한 하나의 중요한 요소는 해당 기술의 역사입니다. 쉽게 말하면 나온 지 오래된 기술일수록 많은 업데이트로 인한 안정성이 확보됩니다.

여기에 더해 사용하는 사람들의 수도 중요합니다. 위의 맥락에 이어서 사용자가 많을수록 버그 리포트 및 요구사항이 많아지면서 서비스가 고도화됩니다.

위의 요소들 외에도 외신들의 평가 및 투자 소식들이 해당 기술을 사용하게 할 이유를 보충해주기도 합니다.

일단 버블은 2012년에 출범한 프로덕트 빌딩 툴로 출시된 지 10여년 정도 된 서비스입니다. 최근에 디자인 툴의 점유율의 1위를 차지한 피그마도 2016년에 출범한 걸 감안하면, 노코딩이라는 도메인에서 2012년에 시작된 버블도 사용자들이 필요로 하는 필수적인 기능과 레퍼런스들은 충분히 쌓인 걸 볼 수 있습니다.

심지어 2021년에는 시리즈A로 1,300억 원 정도의 투자 유치에 성공해 세상에 버블에 대한 목표의 크기를 보여주기도 하였습니다.

마지막으로 유명한 외신들이 평가한 버블에 대해 몇 가지 나열하고 이 장을 마치겠습니다.

The New York Times	'노코드'는 대중들에게 인공지능의 파워를 가져오게 해준다.
Financial Times	이 툴은 좋은 인터넷 시민이 되기 위한 도구이다.
Bloomberg	노코드 솔루션 버블과 마이크로소프트, 창업자 허브 파트너십 개시
Entrepreneur	로우코드와 노코드는 미래의 웹 사이트 빌딩이다.
Microsoft for Startups	출시하고 측정하자: 프로토타입부터 MVP까지

02 CHAPTER 개발자가 극찬하는 노코드 툴

사실 노코드 툴이라도 프로덕트를 만들어낼 때 필요한 도구는 코딩과 동일합니다. 그리고 현재 IT서비스들을 만들 때는 대부분 코딩으로 이루어진 웹 사이트와 프로그램으로 운영하고 있습니다. 이 말은 아무리 노코드 툴로 코딩 없이 서비스를 제작할 수 있다 하더라도 서비스를 만드는 방식은 기존의 코딩과 유사할수록 검증된 개발 방법이라고 할 수 있습니다.

그렇다면 노코드 툴로 홈페이지를 만들 때 어떤 식으로 개발되는지 알아봅시다.

홈페이지는 크게 프론트엔드와 백엔드로 나뉩니다. 프론트엔드는 간단하게 설명하자면 화면과 사용자 인터렉션을 부여하는 과정입니다. 이는 디자인 탭에서 화면을 구성하고 인터렉션은 워크플로우 탭에서 모두 대응할 수 있습니다.

또한 백엔드는 서비스에 사용되는 데이터 구조와 로직을 담당하는 부분입니다. 이는 데이터 탭에서 실제 개발에서 데이터베이스를 구축하는 방법과 유사하게 설계할 수 있습니다. 또한 로직적인 부분은 워크플로우에서도 함께 적용할 수 있습니다.

그리고 버블은 실제 기업에서 프로덕트를 만들 때 필요한 방식이나 협업도 적용시킬 수 있도록 지원을 해주고 있습니다. 예를 들면 일반 기업들은 디자인 시스템을 구축해서 유지보수가 용이하게 할뿐만 아니라 디자인 팀과의 협업이 편리하도록 프로젝트를 진행하는데, 스타일 탭을 이용하면 버블에서도 이와 같은 프로세스로 프로덕트 빌딩이 가능합니다. 디자인 파트가 아니더라도 다양한 개발자들이 하나의 서비스를 개발할 때 사용하는 버전 관리 시스템도 버블에서 지원하여 여러 명이 하나의 프로덕트를 개발할 때 서로 충돌 없이 개발 가능하도록 설계되어 있습니다.

이렇듯 버블은 실제 IT 프로덕트를 만드는 데 진행되는 과정을 그대로 적용시켜도 무리 없는 툴입니다. 다만 정말 코드를 입력하는 과정만 빠져있을 뿐인 것입니다. 개발자 입장에서 보았을 때는 기획 이후 디자인, 개발, 버전 운영 등 하나의 서비스를 만드는 데 필요한 올인원 툴이라고 생각됩니다.

심지어 이렇게 만들어진 서비스가 투자를 받거나 엄청난 스케일 업이 필요해 코딩으로 전환을 해야 하는 상황까지 온다면 그 또한 용이합니다. 왜냐하면 개발팀에 넘겨주어야 하는 화면 명세서는 디자인 탭에, 기능 명세서는 워크플로우 탭에, 데이터베이스 설계 및 구조는 데이터 탭에 모두 나와 있어 추출 후 넘겨주면 됩니다. 데이터들 또한 App Data탭에서 csv형태로 뽑는 것도 가능합니다. 추가로 위에서 언급했듯이 버블은 개발팀에서 개발하는 과정과 비슷해 실제 코드 개발로 전환할 때도 개발자들과 커뮤니케이션 시 무리가 없습니다.

앞서 살펴본 바와 같이 개발자가 보기에는 버블이 초기 가설 검증을 위한 MVP 그리고 그 이후의 운영까지 가능하게 하는 노코드 올인원 툴입니다.

03 CHAPTER 버블로 만들 수 있는 것들

앞서 버블이 왜 프로덕트를 빌딩하는 노코드 툴로 적합하고 심지어 추천까지 하는지 말씀드렸습니다. 그럼 이 쯤에서 여러분들이 궁금해 하는 부분은 '도대체 버블로 어디까지 만들 수 있는지' 일 것 같습니다.

개발 관점에서 살펴보면 CRUD를 활용하는 기능이 들어가는 웹이나 앱은 모두 만들 수 있습니다. CRUD란 Create, Read, Update, Delete의 약자로 서비스에 사용되는 데이터를 생성, 조회, 수정, 삭제를 말합니다.

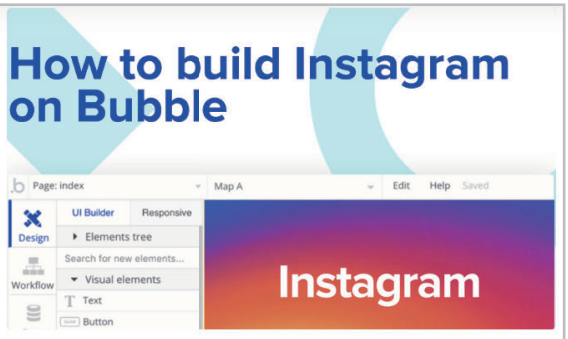
버블에서 메인 페이지에 들어가자마자 나오는 메인 문구에는 소셜 네트워크, 마켓 플레이스, SaaS, 대시보드, CRM 등을 만들 수 있다고 명시하고 있습니다. 이외에도 버블 홈페이지 내에서 다양한 서비스를 그대로 따라 만들 수 있는 가이드 라인을 제공하고 있으니 접속해서 가볍게 경험해 보면 큰 도움이 될 것입니다.

HOW TO

How To Build A Shopify Clone With No Code

Learn how to create your own e-commerce platform, using Bubble's visual web editor.

LACHLAN KIRKWOOD
11 AUG 2020 • 13 MIN READ



HOW TO

How To Build An Instagram Clone With No Code

Learn how to create your own no code photosharing app, using Bubble's visual web editor.

LACHLAN KIRKWOOD
27 JUL 2020 • 11 MIN READ

HOW TO

How To Build A Duolingo Clone With No Code

Learn how to create a language learning app without writing code, using Bubble's visual web editor.

LACHLAN KIRKWOOD

HOW TO

How To Build A News App With No Code

Learn how to create your own no-code news aggregator app, using Bubble's visual web editor.

LACHLAN KIRKWOOD

04 CHAPTER 버블로 서비스를 만드는 과정

지금까지는 노코드가 탄생한 과정에서부터 노코드 툴 버블에 대한 설명까지 다뤘습니다. 그리고 버블이라는 노코드 툴에 대한 확신을 좀 더 드렸으니 어떤 식으로 서비스 혹은 프로덕트를 만들어 낼 수 있는지 알아보겠습니다.

하나의 서비스를 만들 때 개발적인 부분에서 서비스를 만들어 가는 과정 이라고 생각하면 되고 각각의 단계에 대한 자세한 설명은 이어지는 챕터들에 하나하나 자세히 설명해 놓았습니다. 따라서 이번 챕터에서는 버블로 하나의 프로덕트가 만들어 지는 과정의 핵심만 살펴보겠습니다.

1. 데이터

대부분의 서비스는 데이터를 보유하고 있습니다. 예를 들어 회원가입을 받는 홈페이지라면 사용자의 계정 정보가 있을 것이고 게시판이 있는 홈페이지라면 게시글에 대한 데이터를 가지고 있을 것입니다. 그렇다면 서비스를 만들기 전에 어떤 정보 또는 데이터를 어떻게 사용할지 설계해야 합니다. 데이터를 설계하는 방법은 서비스가 제공할 기능을 나열하다 보면 필요한 정보들이 생기는데, 이를 관련된 묶음으로 정리하면 처음의 대략적인 구조가 나오게 될 것입니다. 나머지는 화면을 설계하거나 기능을 더 잘게 쪼갤 때 더욱 정교하게 완성됩니다.

2. 디자인

디자인은 서비스를 고객 등 사용자에게 보여주는 부분이라 매우 중요합니다. 페이지의 레이아웃, 글, 이미지들을 어떤 식으로 배치하느냐에 따라 보여지는 느낌이 180도 달라지기 때문입니다. 또 특정 기능의 작동을 위한 필연적인 화면 혹은 요소가 있을 수도 있습니다. 이렇게 하나의 서비스가 정상적으로 작동하기 위해 만들어 지는 화면을 설계하고 꾸미는 단계가 바로 디자인 단계입니다.

3. 워크플로우

데이터와 디자인을 정했다면 기능에 대한 개발이 필요합니다. 사용자가 요소를 클릭했을 때 무슨 일을 일어나게 할지, 아니면 화면을 전환 시킬지 등등 화면에 대한 액션이나 데이터의 변경이 필요할 때는 모두 이 워크플로우를 거치게 됩니다. 개발을 할 때 가장 다채롭게 만들 수 있는 부분이지만 오히려 그렇기 때문에 가장 난이도가 높은 단계라고 볼 수 있습니다. 실제 개발자들 사이에서의 실력은 이 워크플로우를 얼마나 정교하고 효율적으로 구성하는지에 따라 갈린다고 해도 과언이 아닙니다. 하지만 겁을 먼저 먹는 것보다는 서비스를 동적으로 만들어 주는, 활기를 불어넣어 주는 영역이라고 생각하고 기대감을 갖되 부담없이 접근해 봅시다.

4. 배포

위의 세 단계를 거쳐 왔다면 이제 남은 단계는 그간 만든 이 서비스를 세상에 내보이는 일입니다. 실제 개발에서는 배포라는 작업도 쉬운 일이 아닙니다.

간략히 설명 드리자면, 무한한 땅을 가지고 있는 인터넷이라는 세계의 어느 공간에 땅값을 내고 공간을 부여받습니다(개발에서는 이를 호스팅이라고 합니다). 그 이후에 사람들이 이 공간을 잘 찾아올 수 있도록 이름을 짓고 간판을 답니다(개발에서는 이를 도메인이라고 합니다). 마지막으로 우리가 만들었던 건축물을 땅에 올립니다. 이 하나하나의 작업을 모두 거친 후에야 주변에서 흔히 볼 수 있는 웹 사이트처럼 주소를 치고 들어가서 해당 홈페이지에 대한 내용을 볼 수 있는 것입니다.

하지만 버블에서는 간단하게 버튼 하나만 클릭하면 이 모든 과정들이 자동으로 완료됩니다. 이 방법에 대한 자세한 내용도 이후 챕터에서 더욱 자세히 다룰 예정입니다.

PART 03

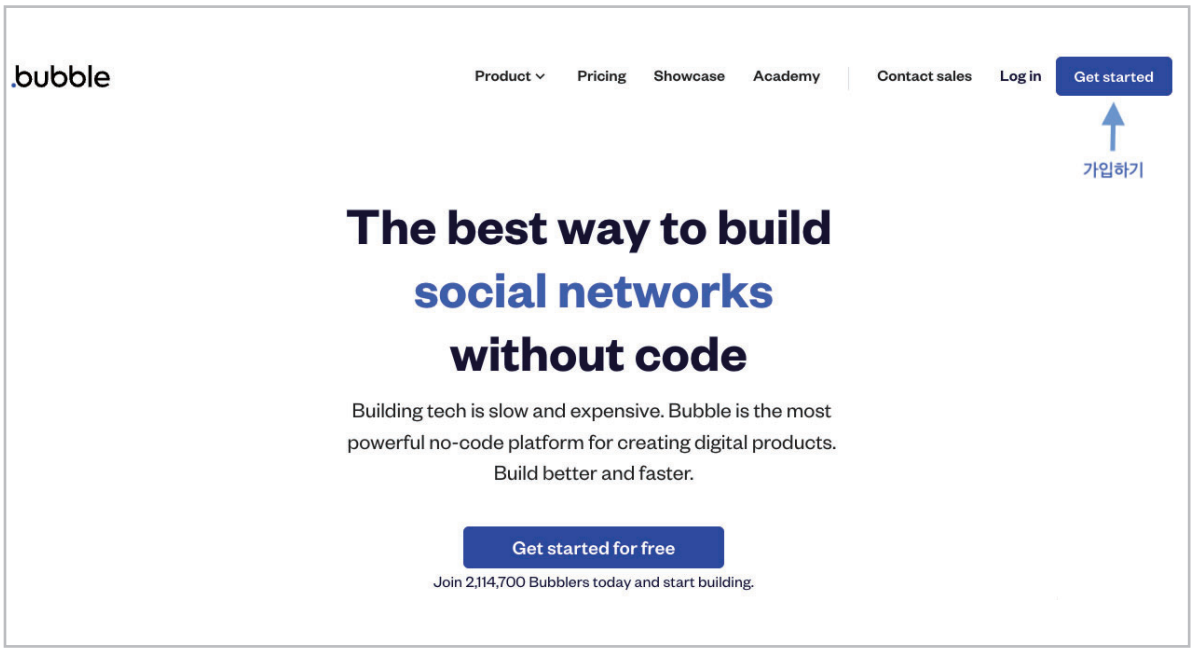
BUBBLE.IO
버블 시작

01 CHAPTER Bubble.io 계정 만들기

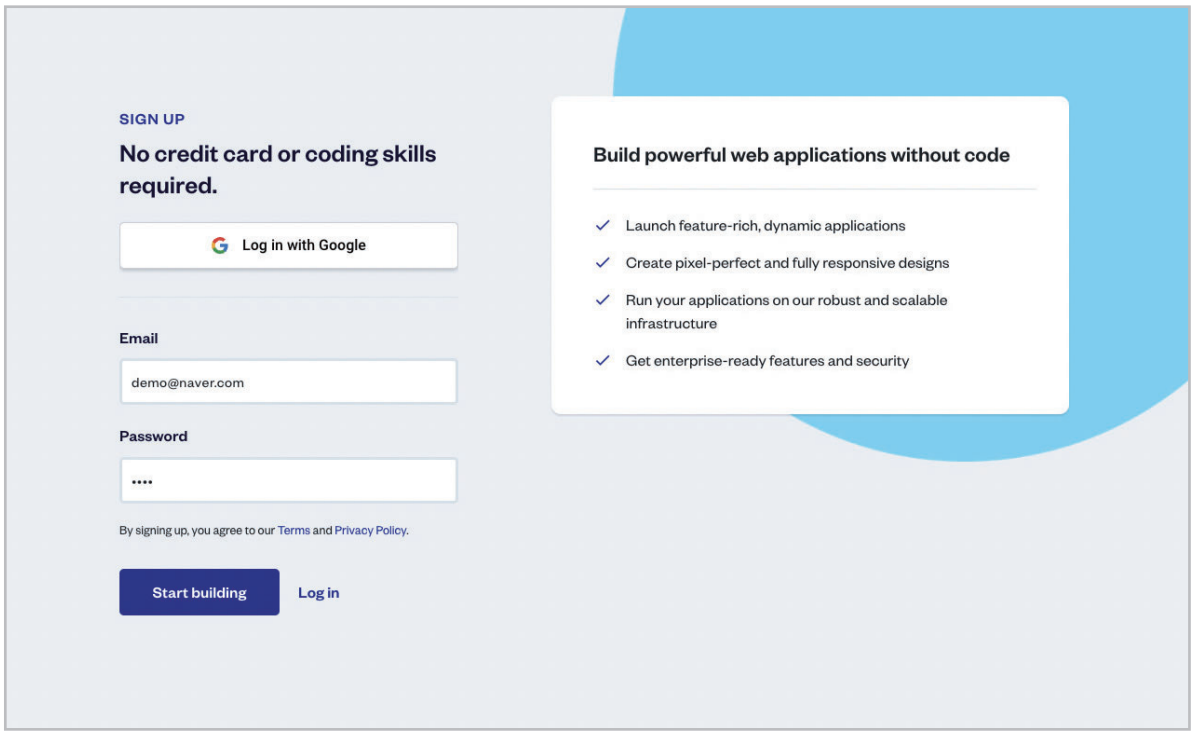
버블에서 프로덕트를 만들기 위해서는 가장 먼저 계정을 만들어야 합니다.
아래 주소를 통해 버블 홈페이지에 들어가서 회원가입을 합시다.

https://bubble.io/

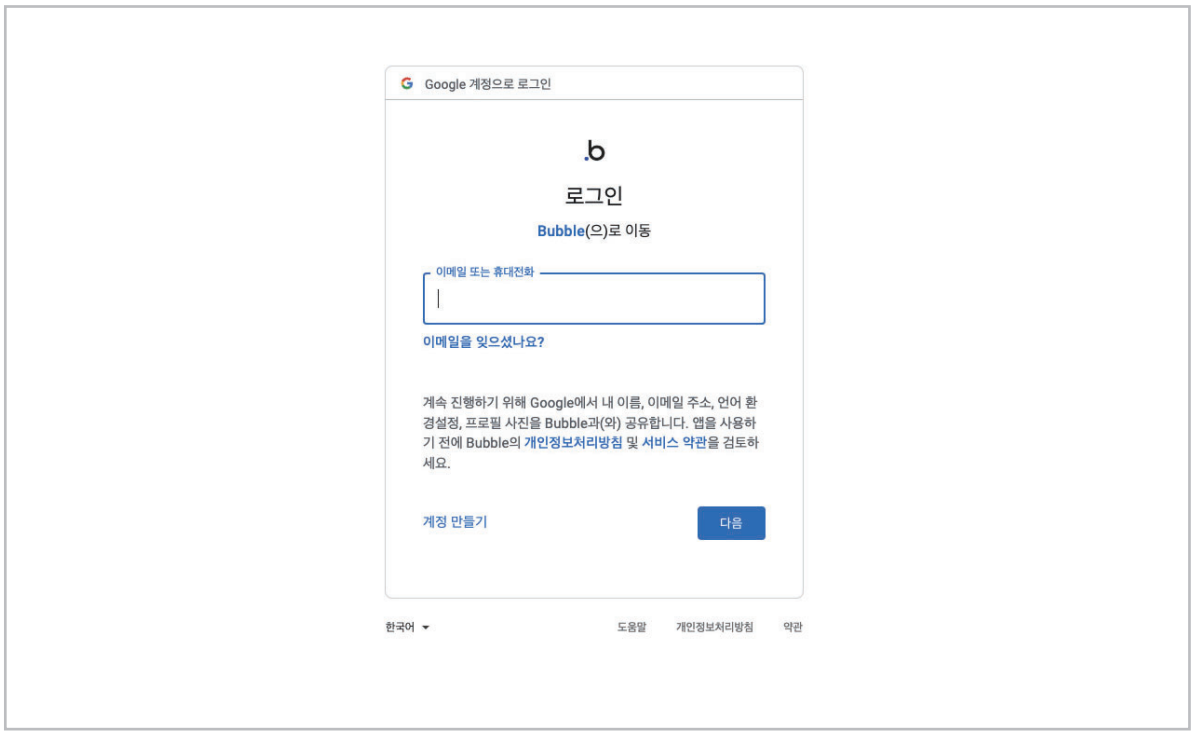
들어가자 마자 우측 상단에 "Get started" 버튼을 누르면 회원가입 페이지가 나옵니다.



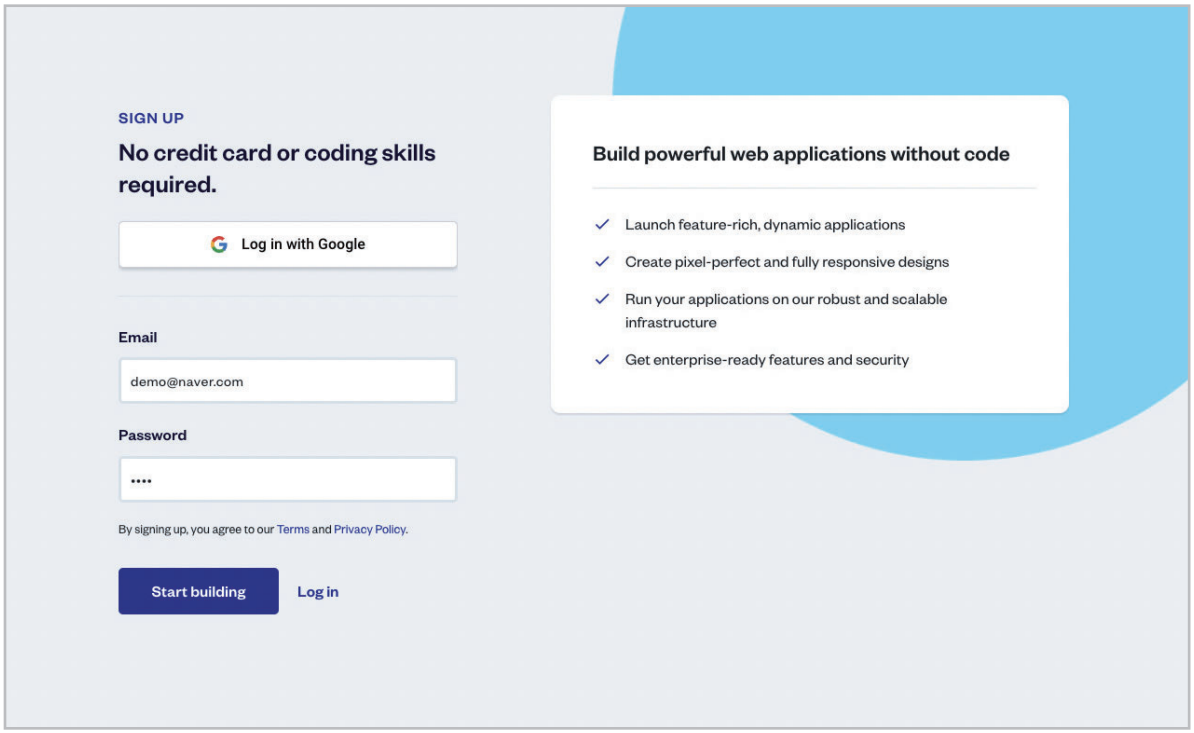
가입 방법은 두 가지가 있습니다.



1. 구글 계정으로 가입

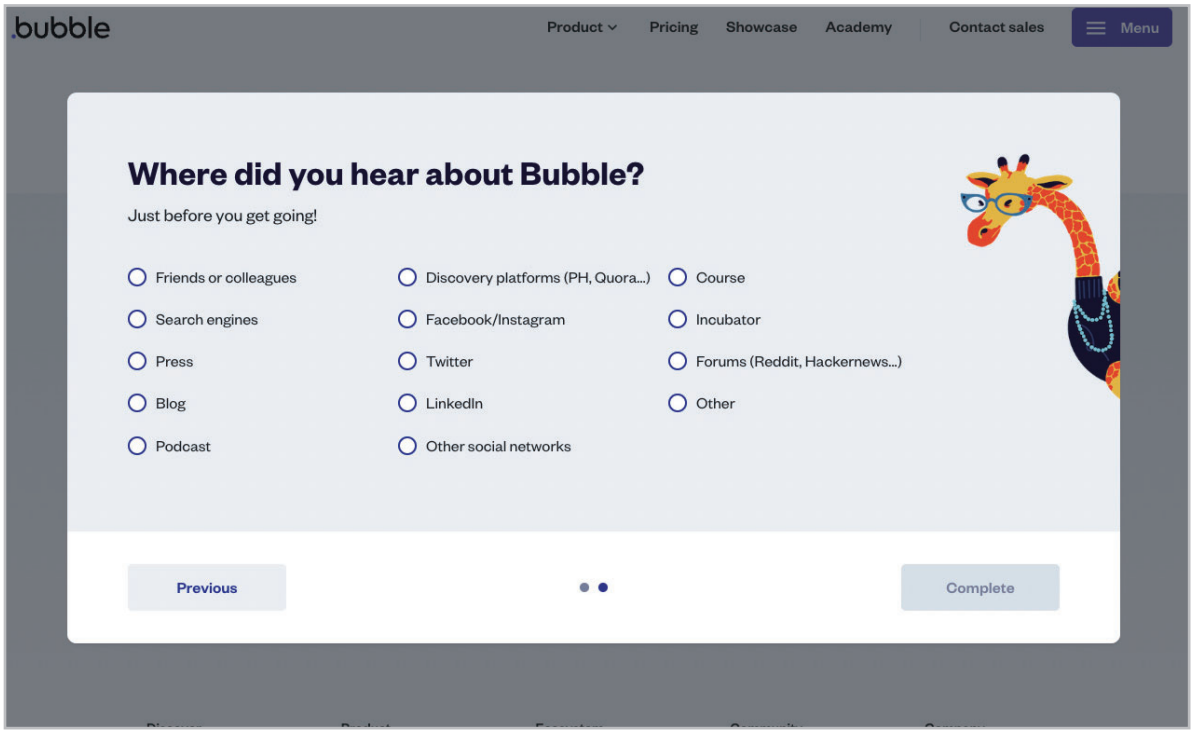


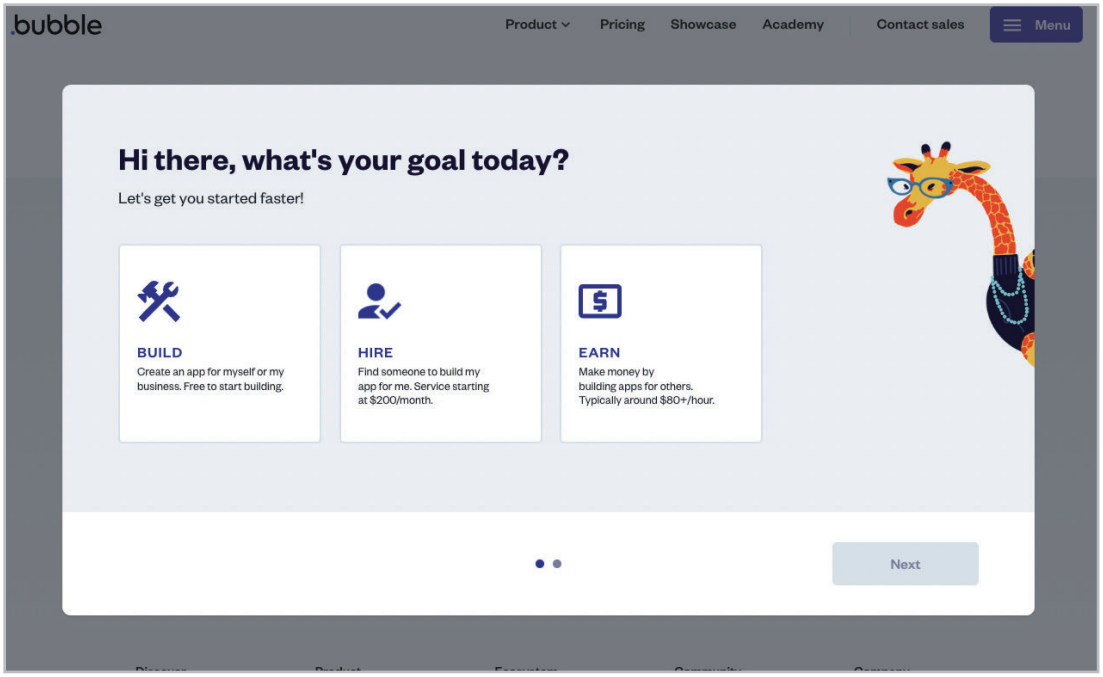
2. 이메일 주소로 가입



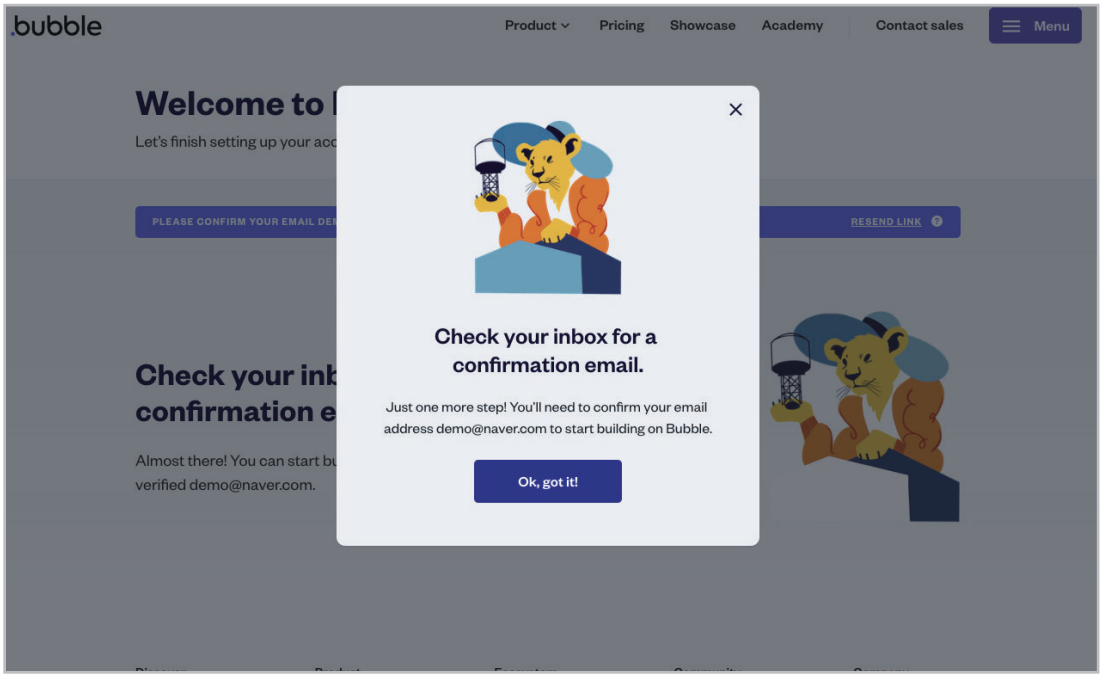
각각의 방법으로 가입을 하고 나면, 간단한 설문조사 문항이 나옵니다.

창업가로써 고객들의 솔직한 피드백이 소중하기에 가입 시 버블에 가입한 목적을 솔직하게 체크해 주는 것이 좋겠습니다.





마지막으로 이메일 인증을 해주어야 버블에서 제작을 할 수 있습니다. 각자 가입할 때 적은 이메일 수신함에 가서 승인요청을 수락합시다.



주의!

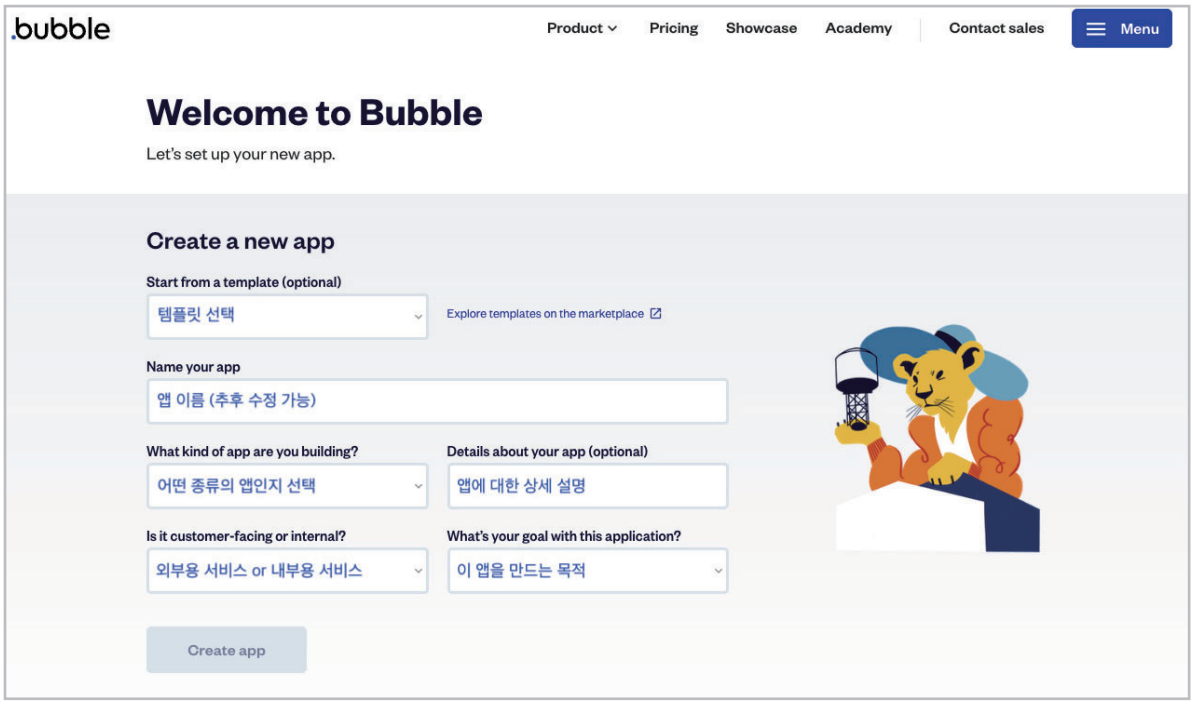
주기적인 버블 측의 A/B 테스트로 위의 첨부 화면이 달라질 수 있습니다. 하지만 이메일 인증 후에 App을 생성할 수 있다는 것은 동일합니다.

02 CHAPTER App 생성

1. App 기본 정보 입력

가입을 완료하고 나면 버블에서 첫 번째 App을 만들기 위한 페이지가 뜹니다.

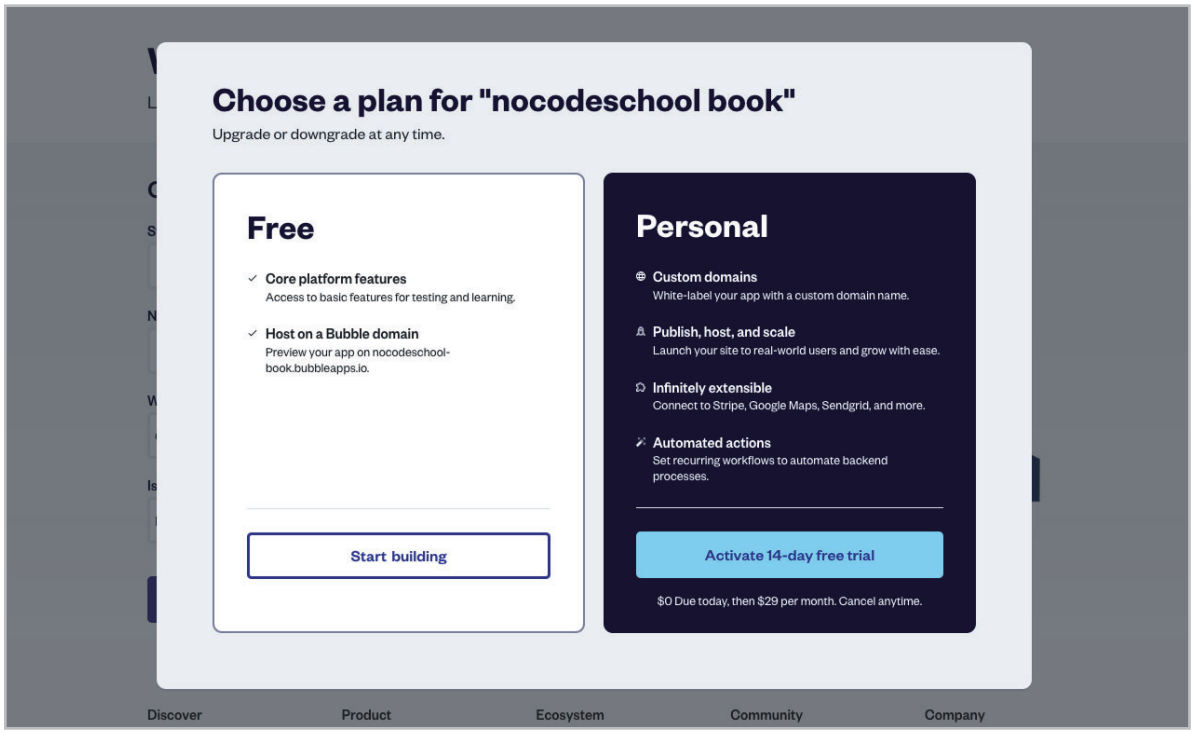
> 주의!
> 여기서 App은 우리가 흔히 생각하는 어플이 아니라 버블에서 제작할 한 프로젝트의 단위를 말합니다.
> 그러니 '나는 Web을 만들고 싶은데 왜 App을 Create하라고 하지?' 라고 헷갈리지 맙시다.



맨 위의 템플릿 선택은 템플릿을 구매했거나, 내부에서 만들었을 때 사용하는 기능인데 초기에는 잘 사용하지 않으므로 그냥 두면 됩니다. 나머지 아래 입력란을 전부 채우고 "Create app"을 누르면 첫 App이 만들어지게 됩니다.

주의사항으로는 버블은 영어 기반의 툴이고 앱 이름으로 임시 url을 생성해주기 때문에 App 이름은 영어와 기본 특수 기호들로 작성해야 합니다.

2. App 플랜 선택



"Create App"을 누르면 플랜을 선택하라고 나옵니다. 버블의 가격 정책에 대해서는 바로 다음에 설명해 드리겠습니다. 우선은 연습용으로 만들 것이기 때문에 Free 플랜으로 "Start building" 합니다.

3. Application Assistant 설정

"Start building"을 누르면 초기 설정을 5 Step으로 도와줍니다.

(1) 타이틀 설정

브라우저를 통해 탭을 열게 되면 생기는 탭에 어떤 텍스트가 나오게 할 지 정할 수 있습니다.

(2) 언어 설정

메인 언어를 선택할 수 있습니다. 우선 한국어를 선택하겠습니다.

(3) 파비콘 설정

탭 앞쪽의 아이콘을 설정할 수 있습니다.

(4) API Connector 플러그인 설치

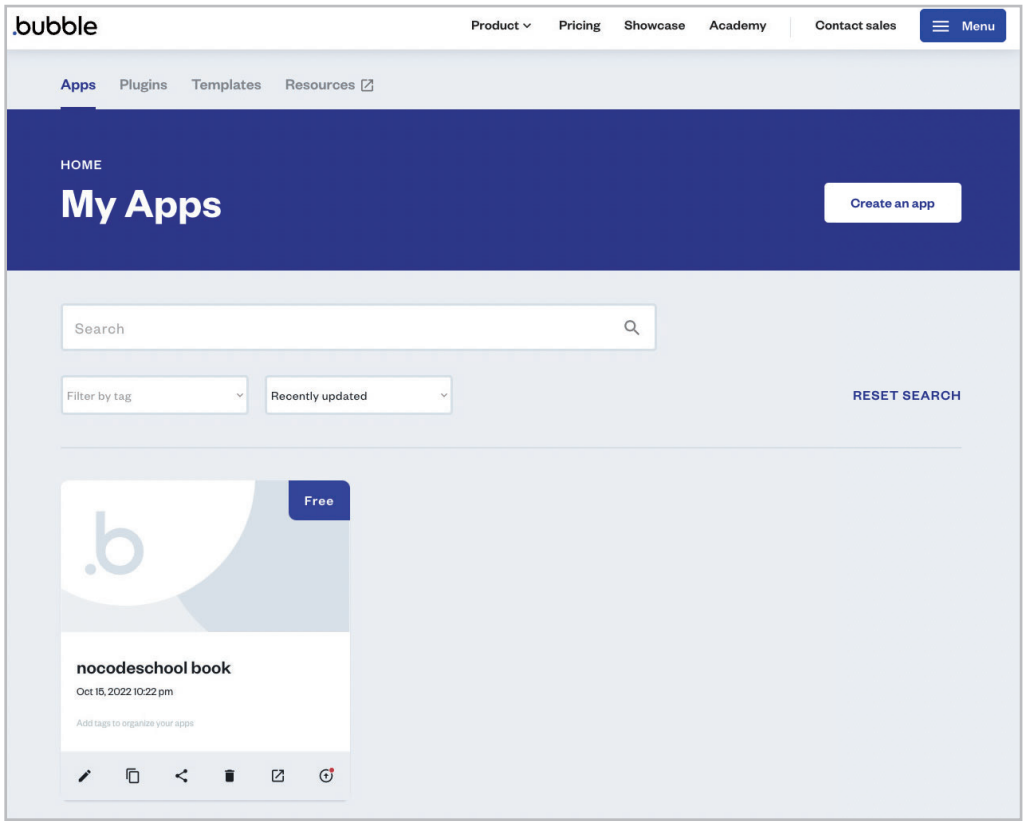
다양한 서비스를 만들다 보면 외부 API를 가져와야 할 경우가 생깁니다. 다만, 우선 버블의 기본적인 기능을 배우고 API 연결을 배우는 것도 늦지 않습니다. 일단은 넘어가도록 합시다.

(5) 그 외의 플러그인 설치

버블에서는 다양한 써드파티 플러그인을 지원합니다. 나중에 천천히 자주 쓰는 플러그인도 같이 배워 볼테니 이것도 지금은 패스하겠습니다.

이렇게 버블에서 첫 App 만들기를 성공했습니다.

> **주의!**
> 위의 5단계도 다양한 A/B테스트 혹은 업데이트로 달라질 수 있으나, 향후 변경 가능한 설정들을 미리 세팅하고 가는 것이므로 큰 신경을 쓰지 않으셔도 됩니다.



이어서 버블의 가격 정책만 빠르게 짚고 넘어간 후에 본격적으로 버블로 서비스를 만들기 위해 필요한 에디터를 뜯어보겠습니다.

03 CHAPTER Bubble의 가격 정책

버블의 가격 정책은 크게 네 가지로 나뉩니다.

가격은 변동이 될 수 있으니 최근 기준으로 설명하겠습니다(작성일: 2022.10.30.).

> **참고**
> https://bubble.io/pricing , https://bubble.io/pricing/compare

1. Free 플랜(무료)

- 버블의 핵심 기능 사용 가능
- 우측 하단에 버블 브랜드 노출
- 커뮤니티 지원

버블을 연습하는 단계에서는 무료 플랜으로도 충분히 가능합니다.

다만 서비스를 론칭하려면 Live 버전으로 배포해야 합니다. 이때부터는 Personal 플랜으로 진행해야 합니다.

2. Personal 플랜(월 25$/1년, 월 29$/1달)

- 버블의 핵심 기능 사용 가능 + API 지원
- 커스텀 도메인 설정 가능
- 이메일 지원

개발 중일 때는 Free로 유지했다가 서비스 론칭 직전에 Personal로 전환해서 배포하면 됩니다. 이 플랜부터는 커스텀 도메인 설정이 가능하니 사람들에게 전달할 사이트 주소를 지정해 보도록 합시다. 그리고 Personal 플랜부터는 우측 하단에 나오는 "Built on Bubble"의 라벨을 없앨 수 있습니다. 이외에도 파일 스토리지도 0.5GB에서 10GB로 늘어납니다.

3. Professional 플랜(월 115$/1년, 월 129$/1달)

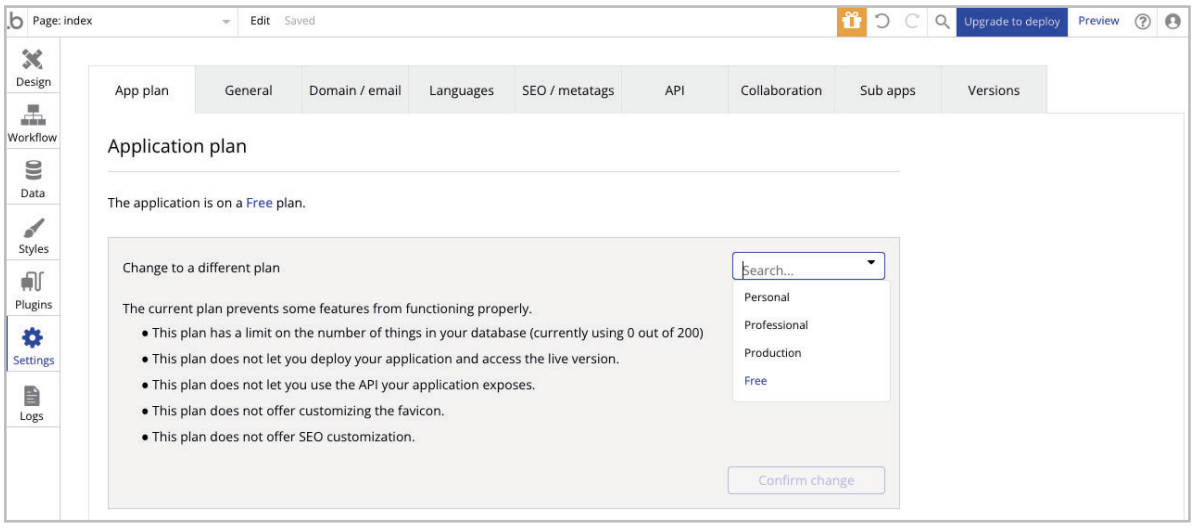
- 서버 capacity 3 유닛
- 2명 에디터 지원
- 2개 dev 버전 지원

MVP로 서비스를 론칭하고 어느 정도 사용자가 늘어나면 서버 용량 확장이 가능합니다. 수정사항을 반영할 버블러도 더 많이 필요해지기 때문에 에디터를 2명까지 추가할 수 있습니다. 개발 버전도 이때 부터는 2개씩 관리할 수 있게 됩니다.

4. Production 플랜(월 475$/1년, 월 529$/1달)

- 서버 capacity 10 유닛
- 15명 에디터 지원
- 20개 dev 버전 지원

Production 플랜부터는 용량, 버블러 제한 수, 개발 버전 모두 넉넉하게 쓸 수 있습니다. 개인적으로, 개발 중간까지는 Free플랜으로 진행하다가 검수 및 론칭 전에 Personal로 전환하는 걸 추천합니다. 그 뒤로 회원수가 증가하고 데이터가 많이 쌓이거나 트래픽이 늘어날 경우에 차차 플랜은 늘려갑시다. 위의 4개 플랜 말고도 개발 에이전시를 위한 에이전시 플랜, 학생 플랜이 있으니 해당 되는 플랜에 위에 적어 놓은 참고 URL로 들어가서 자세히 알아보면 됩니다. 마지막으로, 플랜을 바꾸고 싶을 경우에는 버블 에디터의 Seetings탭> App plan에서 원하는 플랜으로 변경하면 됩니다.



PART 04

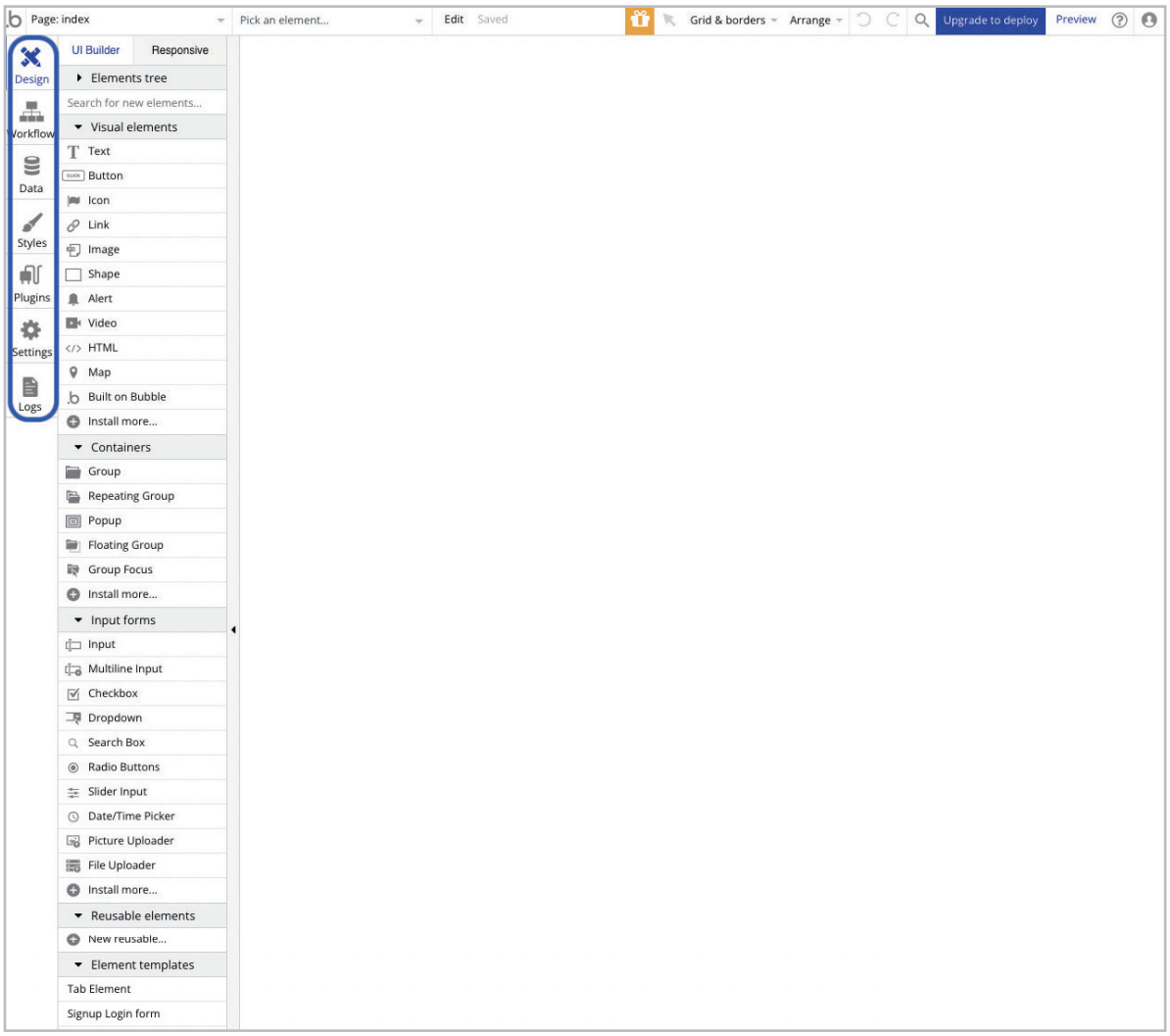
버블 에디터 뜯어보기

01 CHAPTER 메인 탭

앞선 챕터에서 앱을 만들었으니 이제 이러한 앱을 편집할 수 있는 버블 에디터에 대해 소개하도록 하겠습니다.

버블 앱을 활성화하면 아래 이미지처럼 버블 에디터가 나오게 됩니다.

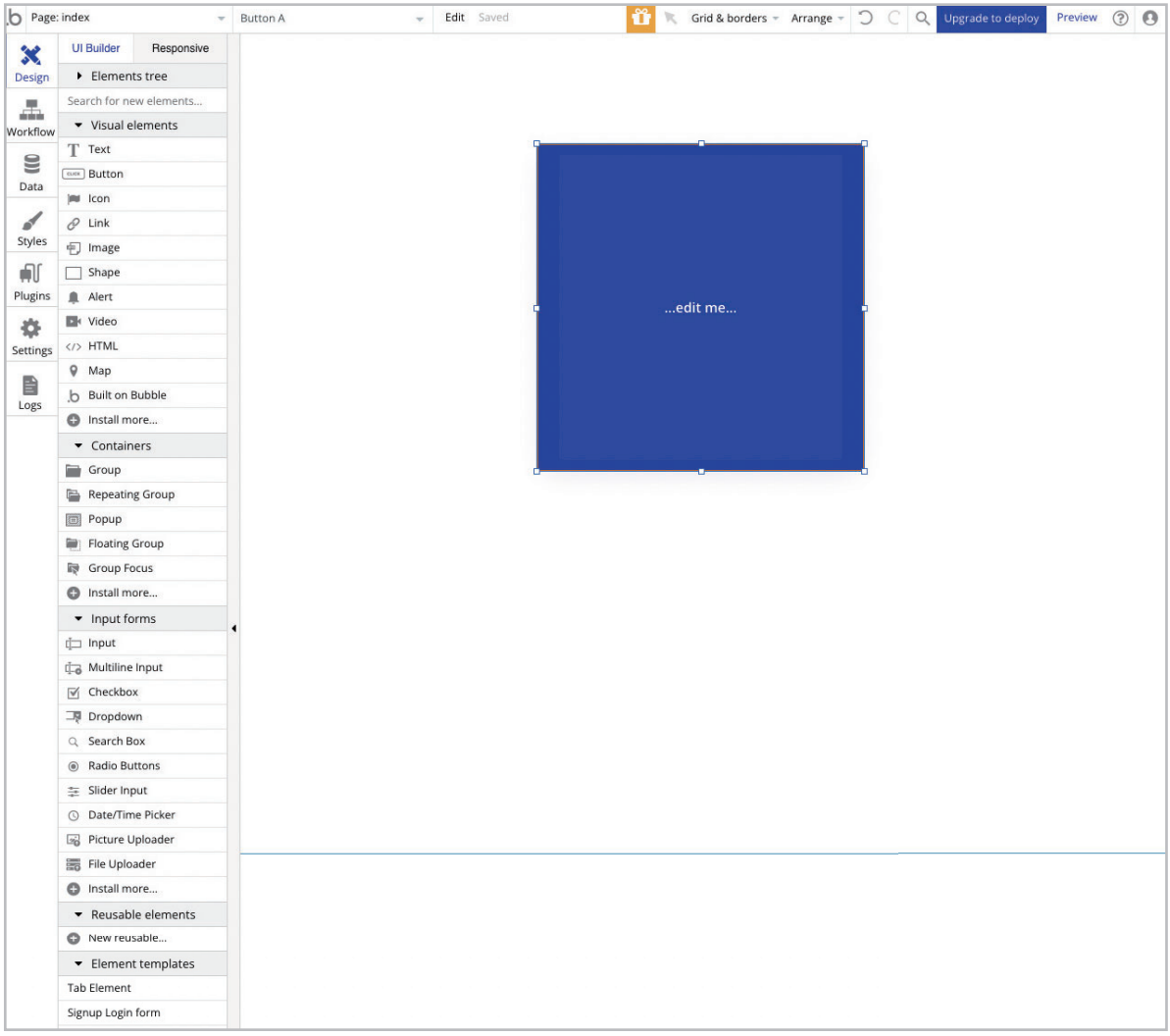
왼쪽에 보면 총 7개의 메인 탭이 있고 여기에서 탭을 바꿔가며 화면, 워크플로우, 데이터들을 만들거나 편집할 수 있습니다.



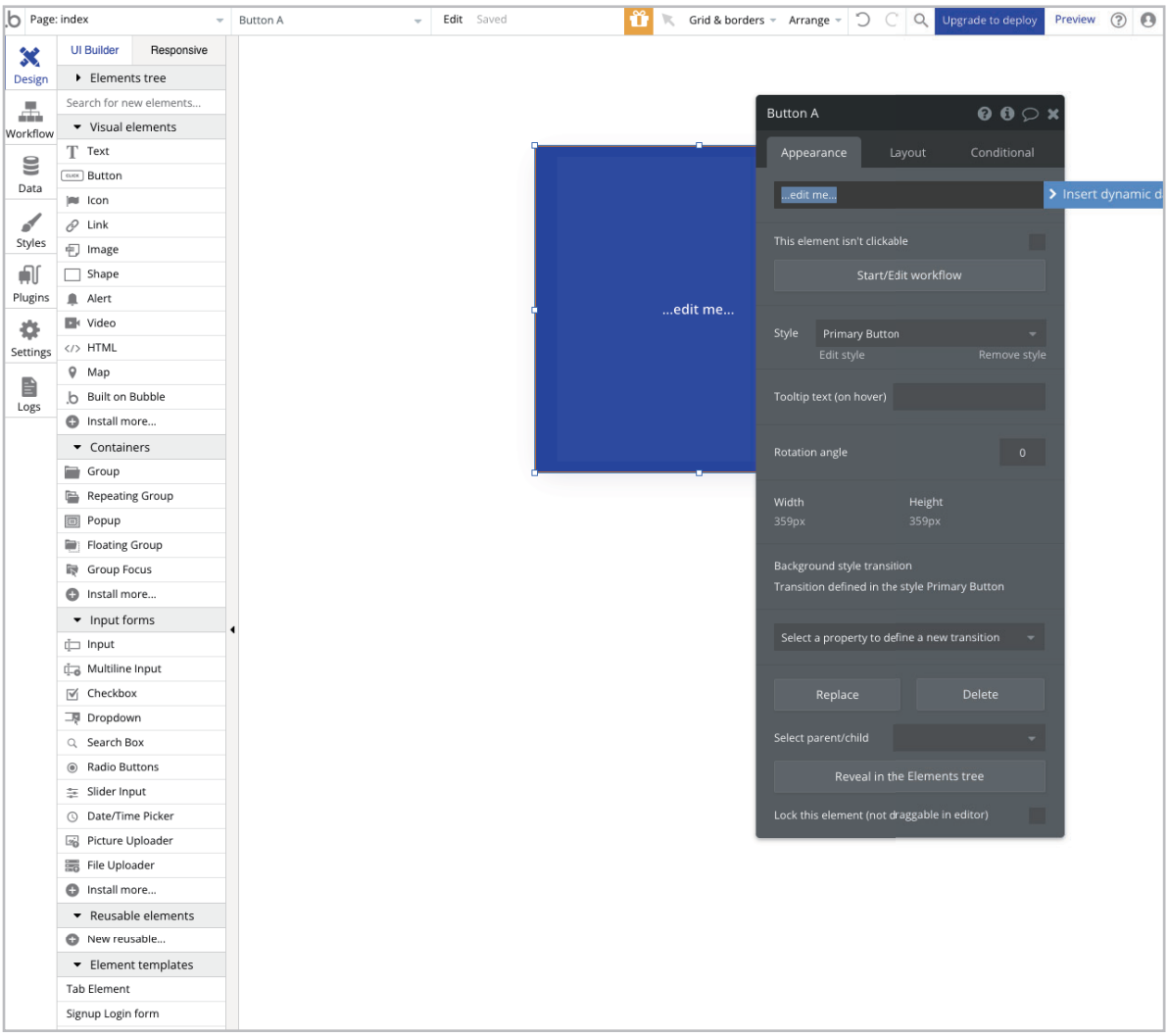
1. 디자인

이 곳은 서비스의 외형을 만들 수 있는 공간입니다. 페이지 위에 요소를 그릴 수도 있고 간편하게 드래그로 사이즈를 조절할 수도 있습니다. 그리고 각 요소들의 프로퍼티도 변경할 수 있습니다.

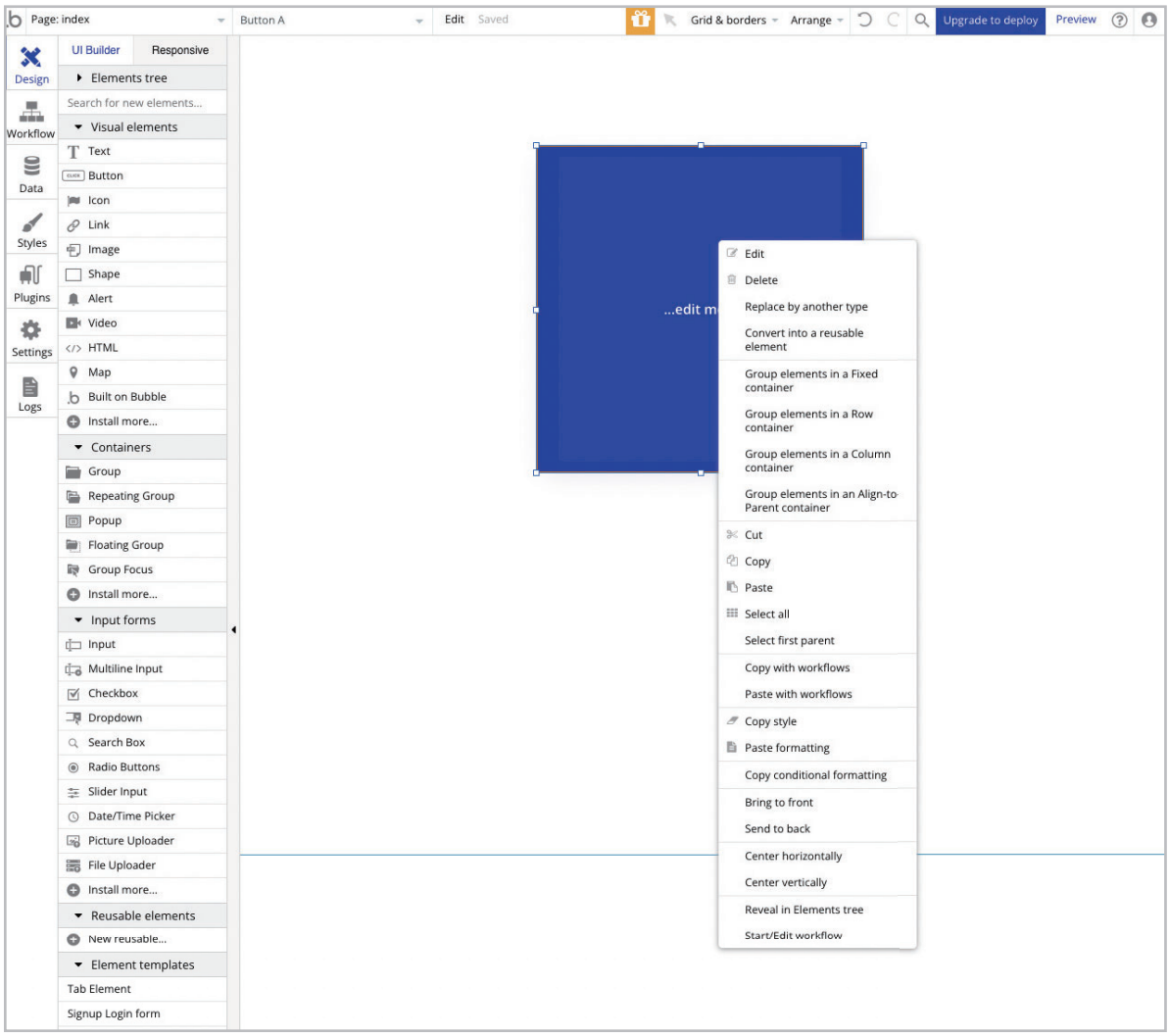
새로운 요소를 추가하기 위해서는 왼쪽의 요소 타입을 클릭해서 페이지에 드래그하면 화면에 해당 요소가 추가됩니다.



만들어진 요소를 더블 클릭하면 특정 성질이나 행동을 커스터마이징할 수 있는 프로퍼티 에디터가 나오게 됩니다(프로퍼티 에디터에 대한 내용은 뒤에서 설명해 드립니다).

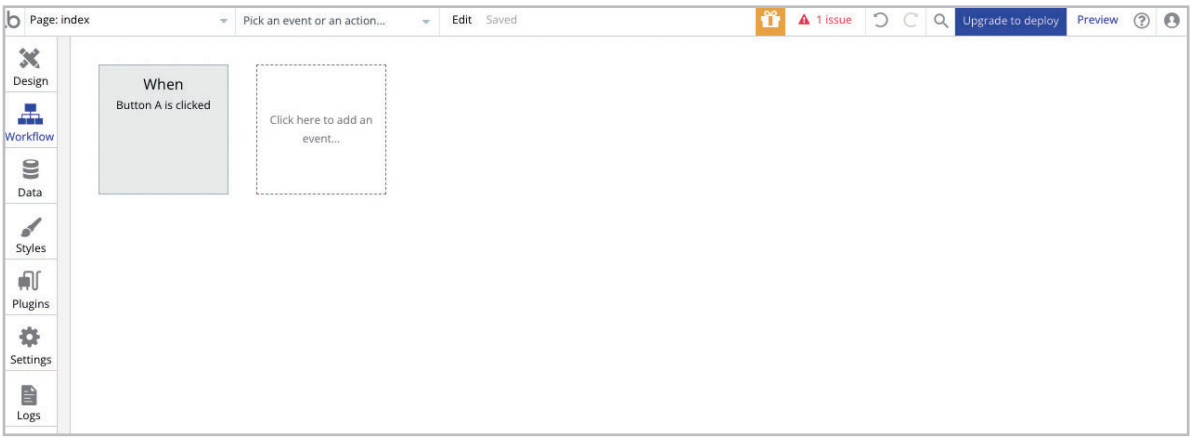


우 클릭 시, 편집을 위한 각종 추가 옵션들이 드롭다운 형태로 나타납니다.

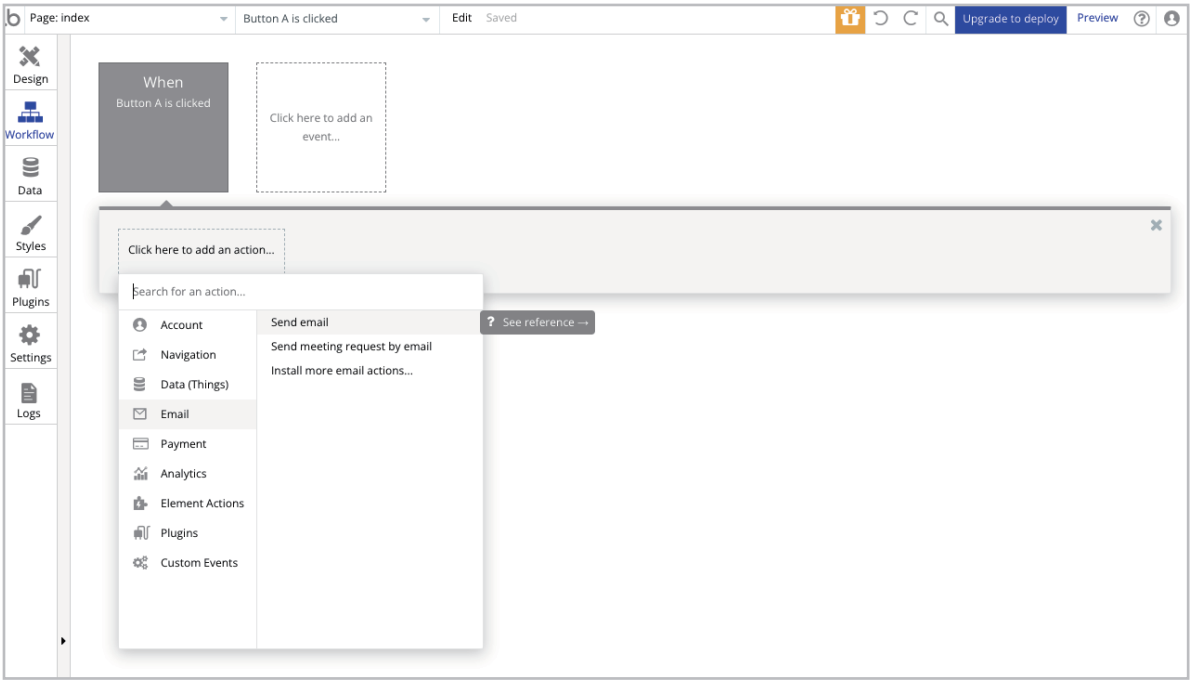

Page: index
Button A
Edit
Saved
Grid & borders
Arrange
Upgrade to deploy
Preview
Design
Workflow
Data
Styles
Plugins
Settings
Logs
UI Builder
Responsive
Elements tree
Search for new elements...
Visual elements
Text
Button
Icon
Link
Image
Shape
Alert
Video
HTML
Map
Built on Bubble
Install more...
Containers
Group
Repeating Group
Popup
Floating Group
Group Focus
Install more...
Input forms
Input
Multiline Input
Checkbox
Dropdown
Search Box
Radio Buttons
Slider Input
Date/Time Picker
Picture Uploader
File Uploader
Install more...
Reusable elements
New reusable...
Element templates
Tab Element
Signup Login form
...edit m
Edit
Delete
Replace by another type
Convert into a reusable element
Group elements in a Fixed container
Group elements in a Row container
Group elements in a Column container
Group elements in an Align-to-Parent container
Cut
Copy
Paste
Select all
Select first parent
Copy with workflows
Paste with workflows
Copy style
Paste formatting
Copy conditional formatting
Bring to front
Send to back
Center horizontally
Center vertically
Reveal in Elements tree
Start/Edit workflow

2. 워크플로우

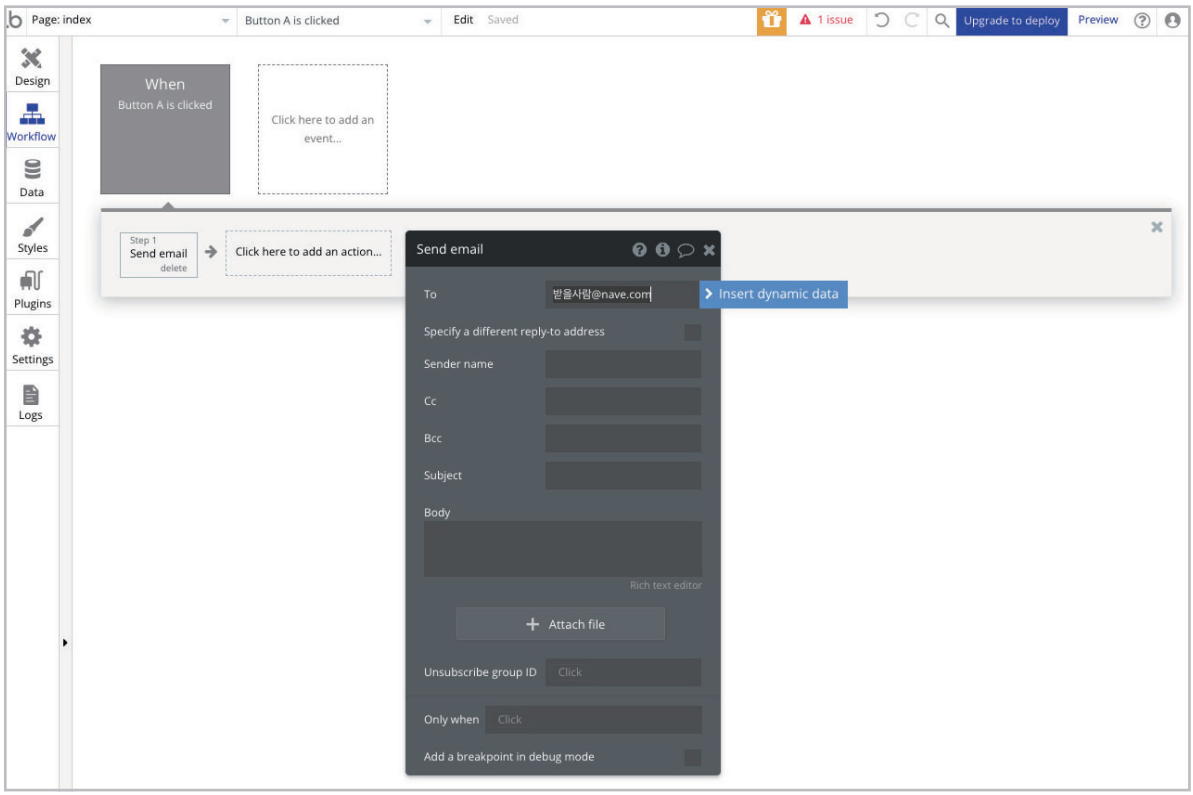
워크플로우 탭에서는 각 페이지에서 작동해야 하는 논리구조를 구축할 수 있습니다. 각각의 워크플로우는 박스 형태로 나타나고 하나의 워크플로우를 "이벤트"라고 부릅니다.



이벤트(혹은 액션)를 클릭하게 되면 프로퍼티 에디터와 비슷한 패널이 등장하게 됩니다. 여기에서 액션을 위한 다양한 필드를 채울 수 있습니다.



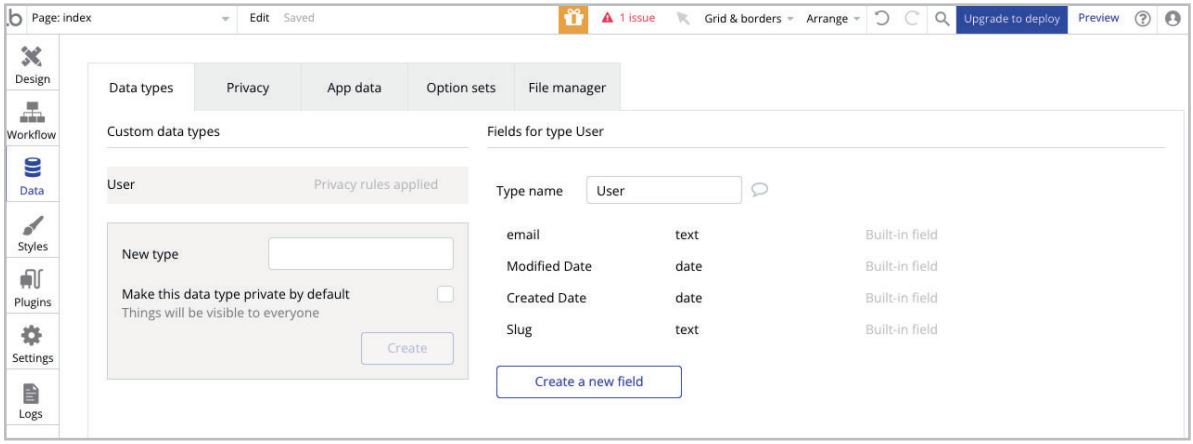
예를 들면 "이메일 전송하기" 액션은 이메일을 받을 "To" 필드를 채울 수 있습니다. 제목은 "Subject", 내용은 "Body"에 정해서 입력해 놓을 수 있습니다. 이후 해당 이벤트가 실행될 때 "To"필드에 써 있는 주소로 정해놓은 제목과 내용이 코딩 없이 전송되게 됩니다. 만약 접속한 사람마다 다른 주소로 메일을 보내야 한다면 옆에 있는 "Insert dynamic data"를 통해 동적인 데이터를 넣을 수도 있습니다.



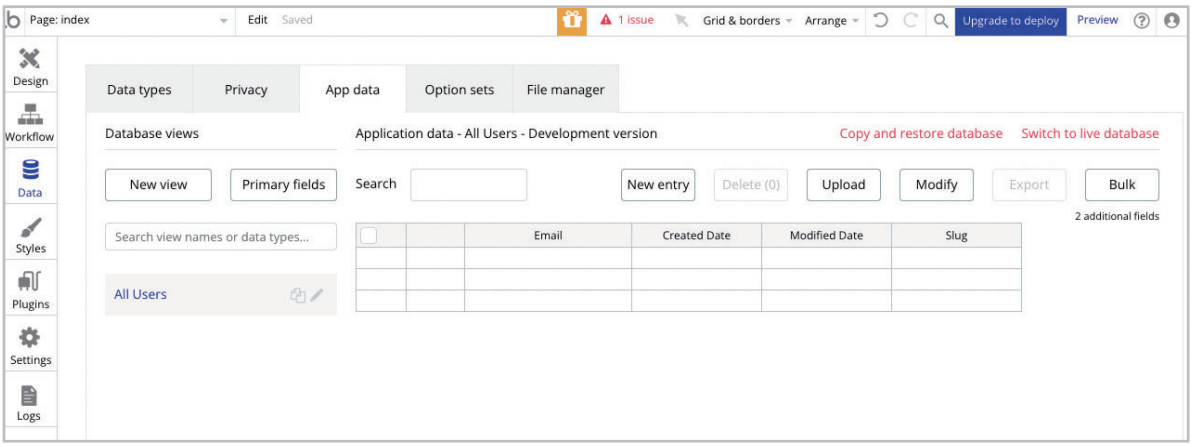
주의! – 정적 데이터 vs. 동적 데이터

정적 데이터는 어떤 경우에도 바뀌지 않는 정해져 있는 데이터지만, 동적 데이터는 특정 상황과 조건에 따라 바뀌게 되는 데이터를 말합니다.

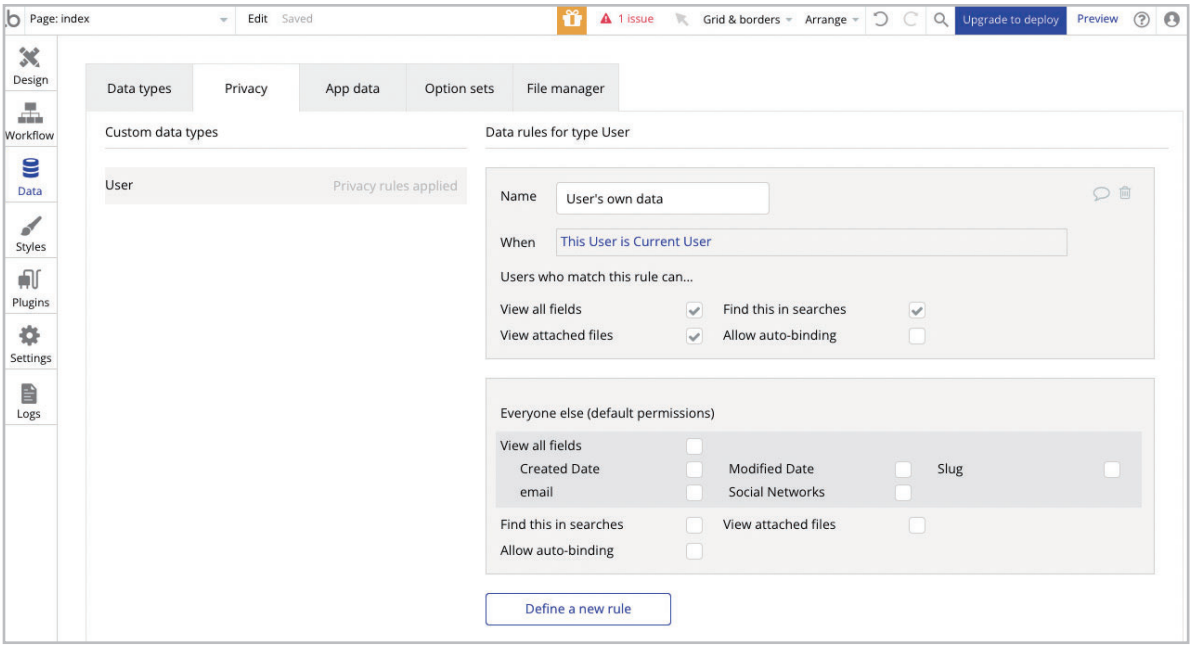
3. 데이터



데이터 탭은 우리가 만든 데이터 구조를 보거나 수정할 수 있습니다. 서비스에 사용된 다양한 데이터 타입과 그 안에 있는 필드들을 한 눈에 볼 수 있습니다.



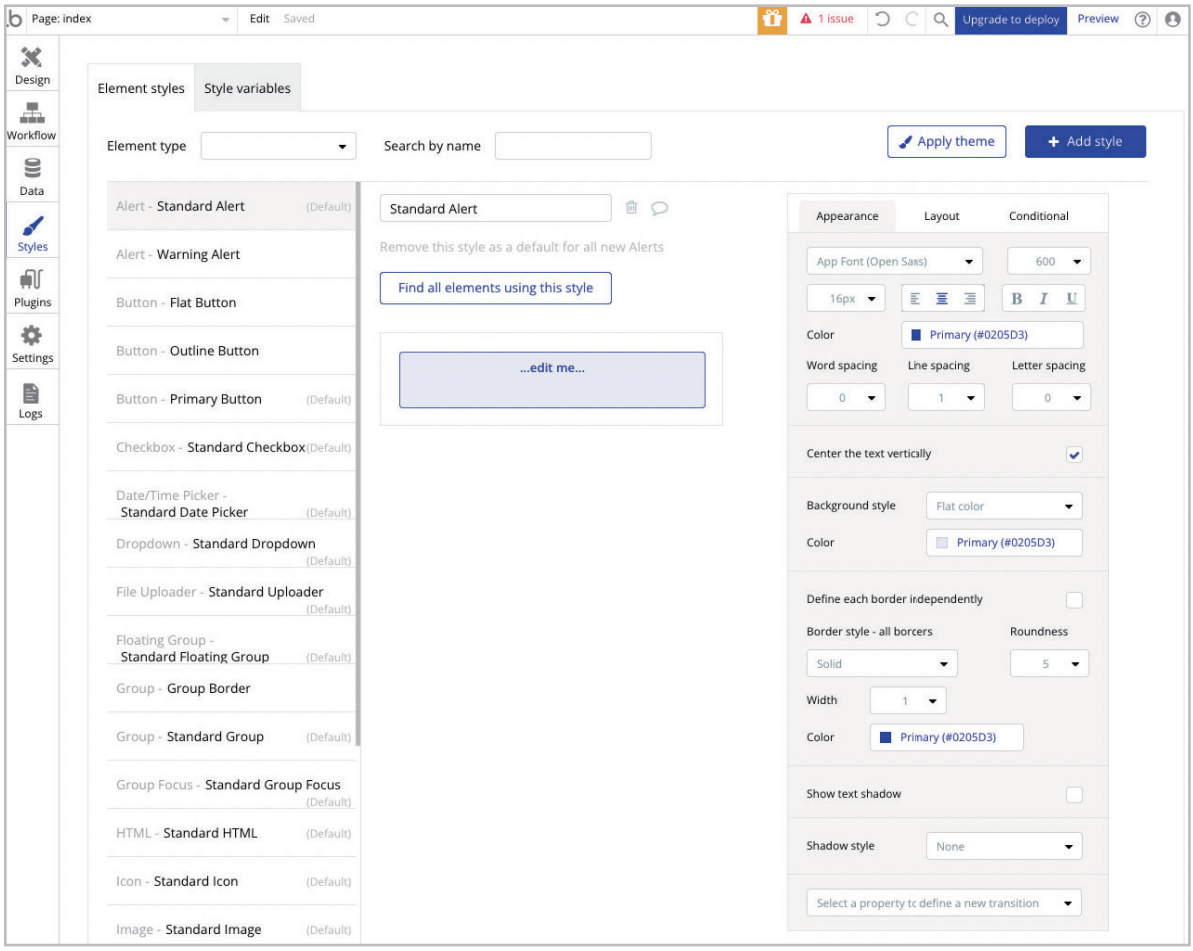
"App Data" 섹션에서는 실제 만들어진 데이터들을 테이블 형식으로 조회가 가능합니다. 이 곳에서 직접 각 데이터들을 추가, 수정, 삭제가 가능하고 엑셀 형식의 파일 형태로 export도 가능합니다.



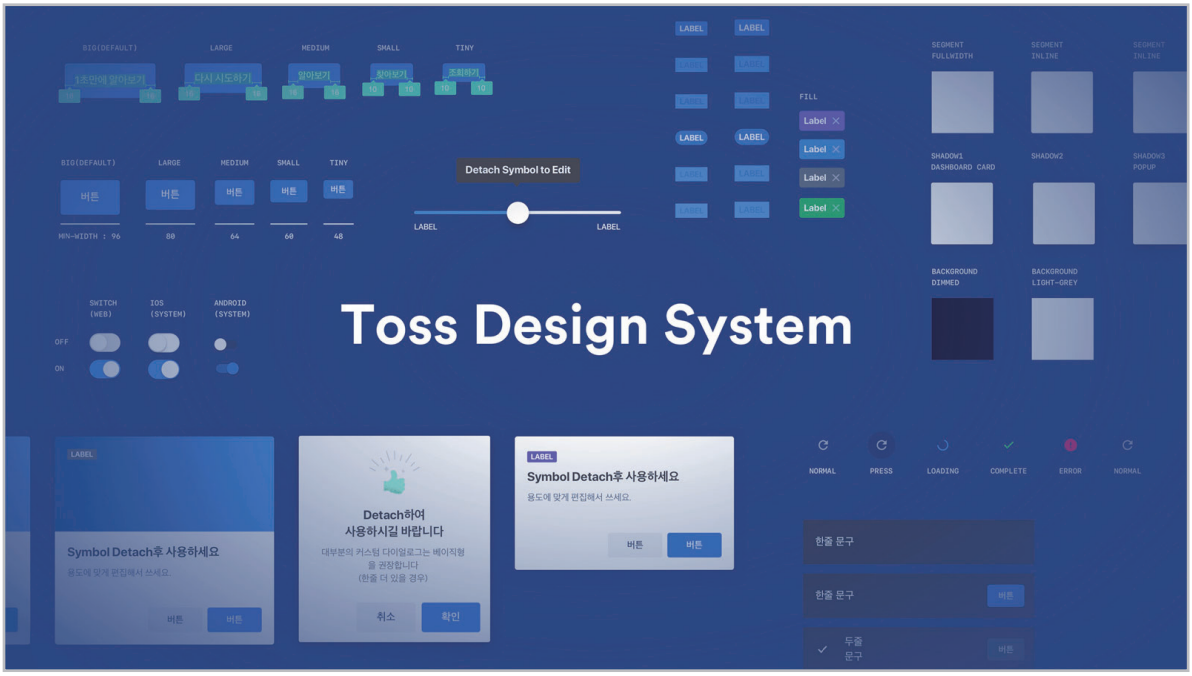
프라이버시 섹션은 데이터마다 특정 경우에만 조회, 수정, 삭제할 수 있도록 규칙을 지정해 둘 수 있는 곳입니다. 이 기능은 심화적인 내용이긴 하지만 실제 서비스 운영까지 간다면 분명 필요한 순간이 올 겁니다. 보안은 중요하기 때문입니다.

4. 스타일

스타일 탭은 다양한 요소의 디자인을 일정한 스타일로 지정해놓을 수 있게 만든 탭입니다.



스타일은 좀 더 일관된 화면 구성을 만들 수 있도록 도와줍니다. 텍스트를 쓸 경우 A 스타일, 버튼을 만들 경우 B 스타일 등으로 지정해 놓는다면 여러 명이 한 서비스를 만들어도 한 사람이 만든 것처럼 통일성을 갖추게 됩니다.



실제 현업에서는 이를 “디자인 시스템”이라고 부릅니다. 디자이너들이 서비스를 만들 때 요소마다 일관된 디자인을 지정해 놓으면 개발자들은 그걸 보고 그대로 옮기면 됩니다.

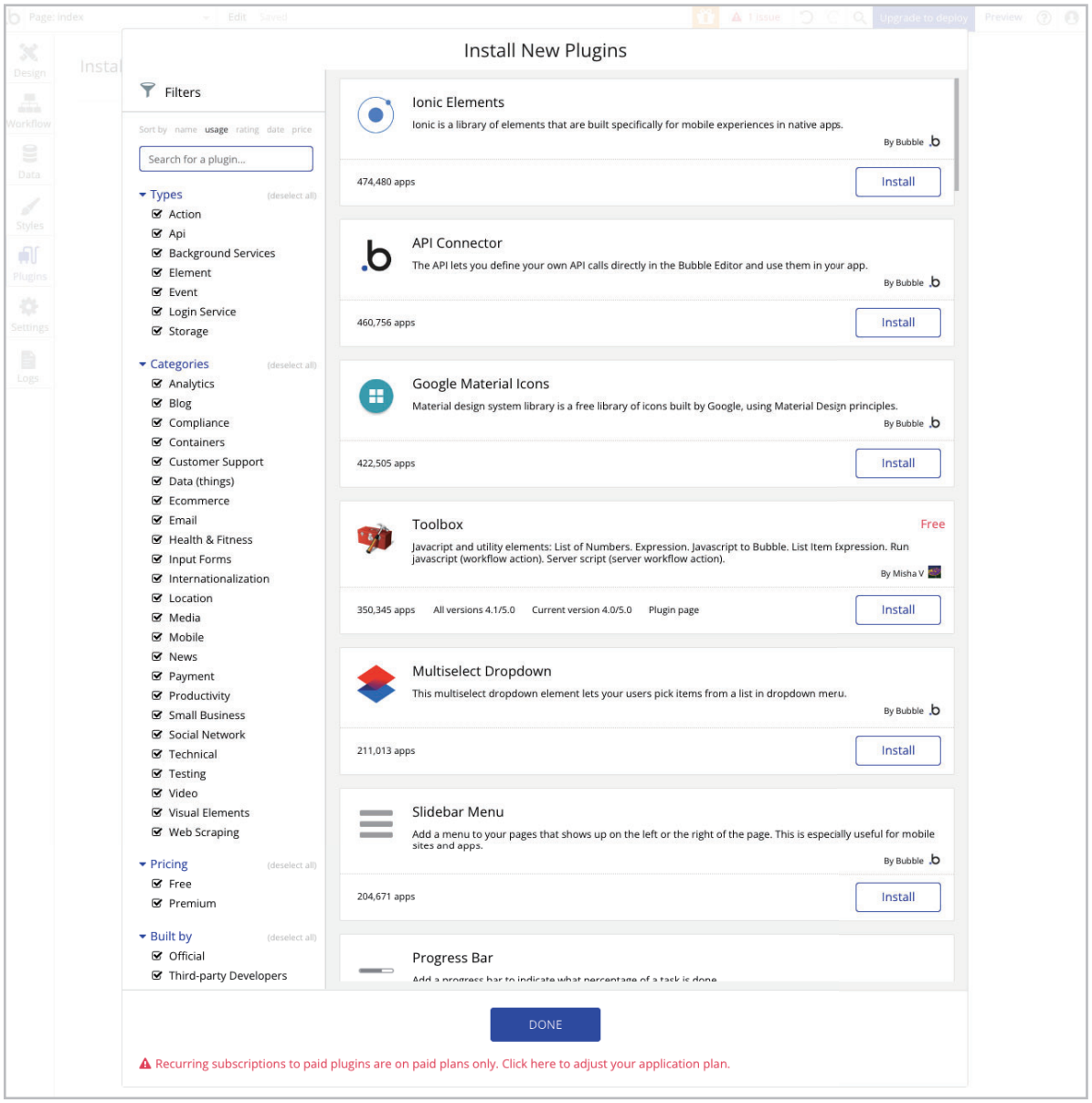
그리고 이렇게 스타일을 공통적으로 정해놓으면 나중에 테마를 바꾸거나 서비스에 쓰인 요소의 디자인을 바꿀 때 빠르게 처리할 수 있습니다.

예를 들면, 모든 버튼의 색상을 보라색에서 분홍색으로 바꾸어야 할 때 스타일 탭에서 버튼의 색상만 바꿔주면 일괄 적용되게 됩니다.

스타일은 생산성을 탁월하게 높여주므로 적극적으로 활용하길 추천합니다.

5. 플러그인

플러그인 탭은 버블의 기능을 확장해서 쓸 수 있는 아주 실용적인 플러그인들을 모아놓은 곳입니다.

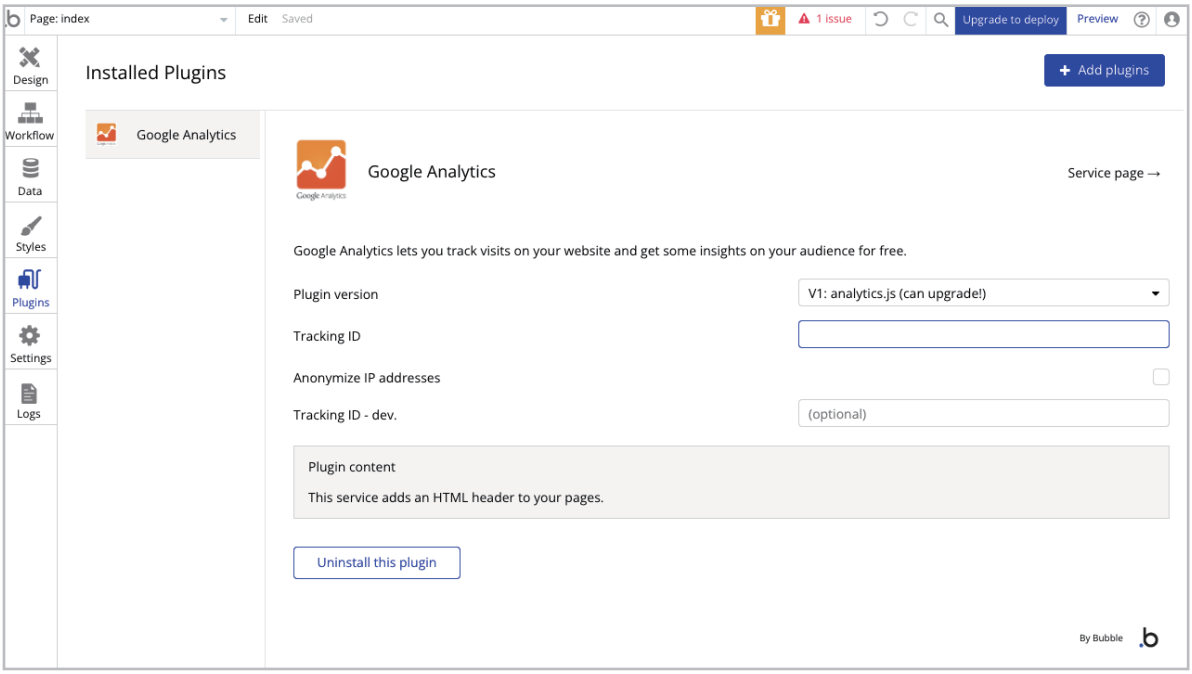


예를 들면, 구글 로그인이나 결제 기능처럼 공통적으로 많이 쓰이는 기능을 손쉽게 이용할 수 있도록 하나의 모듈로 만들어 놓은 플러그인들이 있습니다.

그 외에도 직접 버블로 만들기 오래 걸리거나 까다로운 부분들은 플러그인을 설치해 적용하면 원하는 바를 빠르게 구현할 수 있습니다.

다만 유료 플러그인도 있어 너무 많이 사용하게 되면 월 관리비가 점점 비싸질 수 있으니 유의해야 합니다. 지금은 배우는 단계이니 최대한 직접 만들어 보는 방법을 추천합니다.

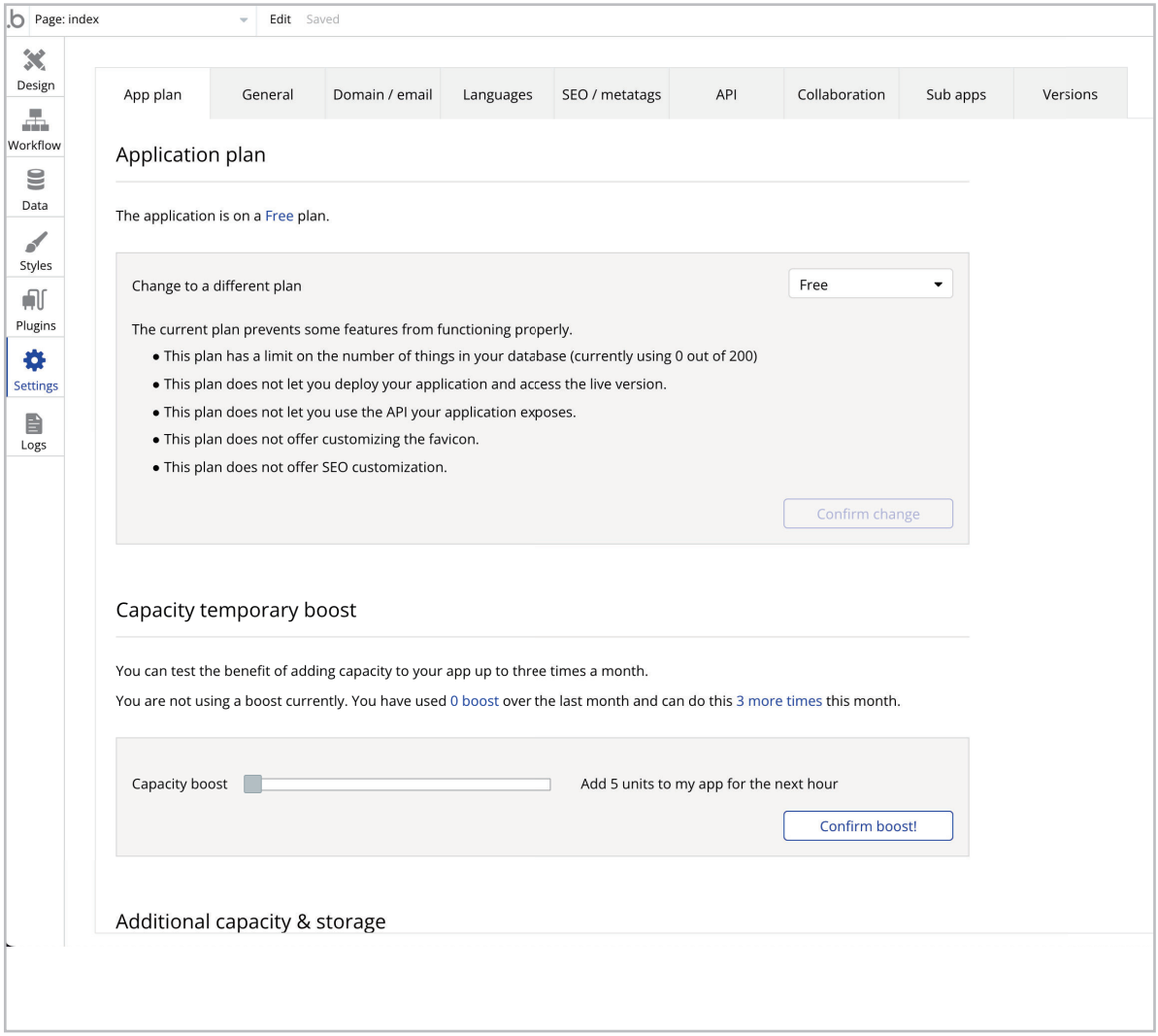
플러그인은 버튼 하나로 설치가 가능하고 설치한 뒤에는 플러그인 상세페이지에 적힌 설명대로 사용하거나 기입해야 하는 파라미터를 채워 넣으면 적용되기도 합니다.



대표적으로 사용자들을 분석해주는 구글 애널리틱스 플러그인은 ID값을 넣어주면 자동으로 적용되도록 되어 있습니다.

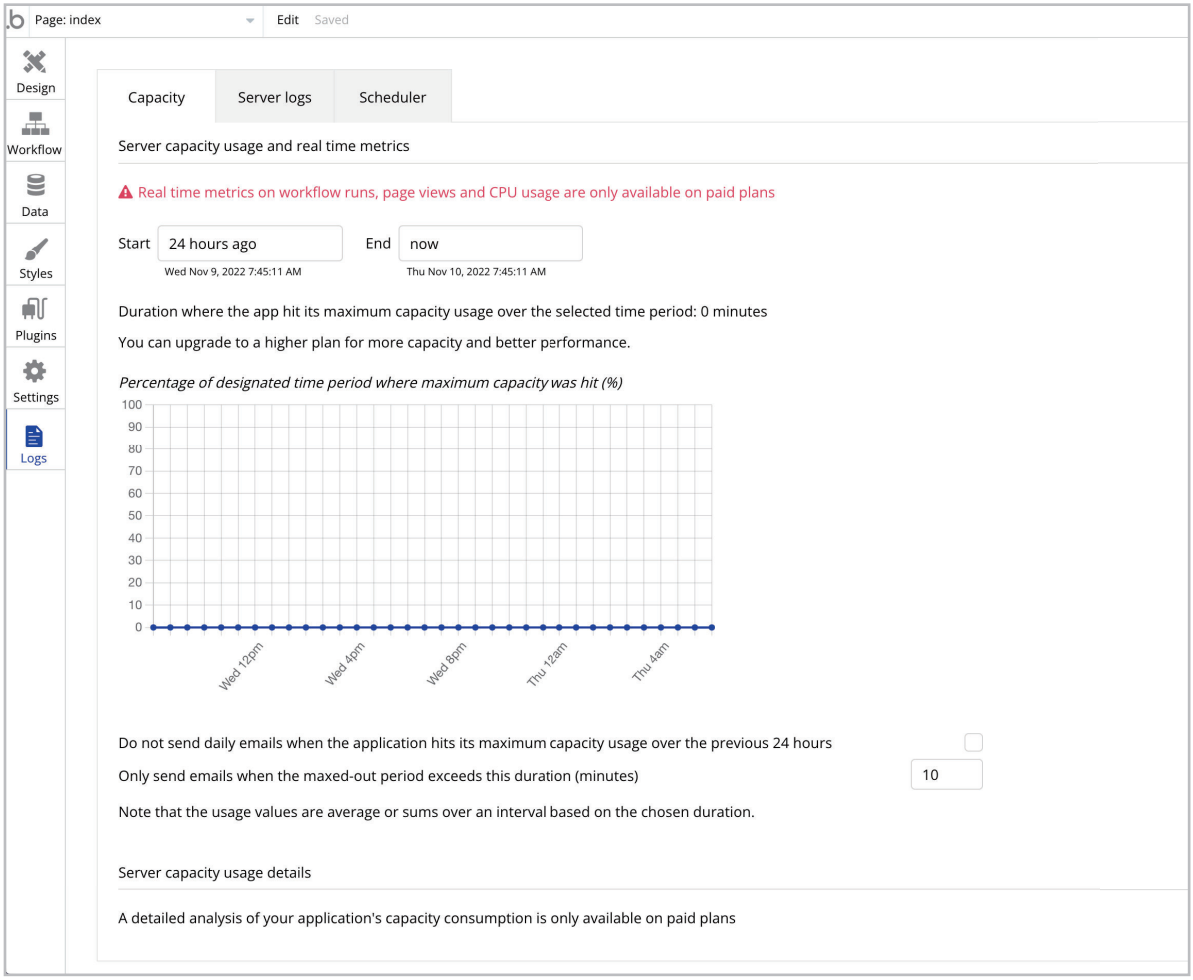
6. 세팅

세팅 탭은 서비스에 대한 전반적인 사항들을 설정할 수 있습니다. 이곳에서는 버블 요금제를 선택할 수도 있고 론칭하기 전에 필요한 세팅 값들도 설정이 가능합니다. 심지어는 컬러 팔레트부터 커스텀 폰트, 비밀번호 정책, 그리고 앱의 보안까지 관리할 수 있습니다.



7. 로그

로그 탭은 서비스가 라이브를 실제로 작동하고 있을 때 아주 유용합니다. 아니면 Run-mode로 테스트할 때에도 좋습니다. 여기에서는 실행된 이벤트나 워크플로우 이슈들을 진단할 수 있고, 예약된 워크플로우도 확인할 수 있습니다.



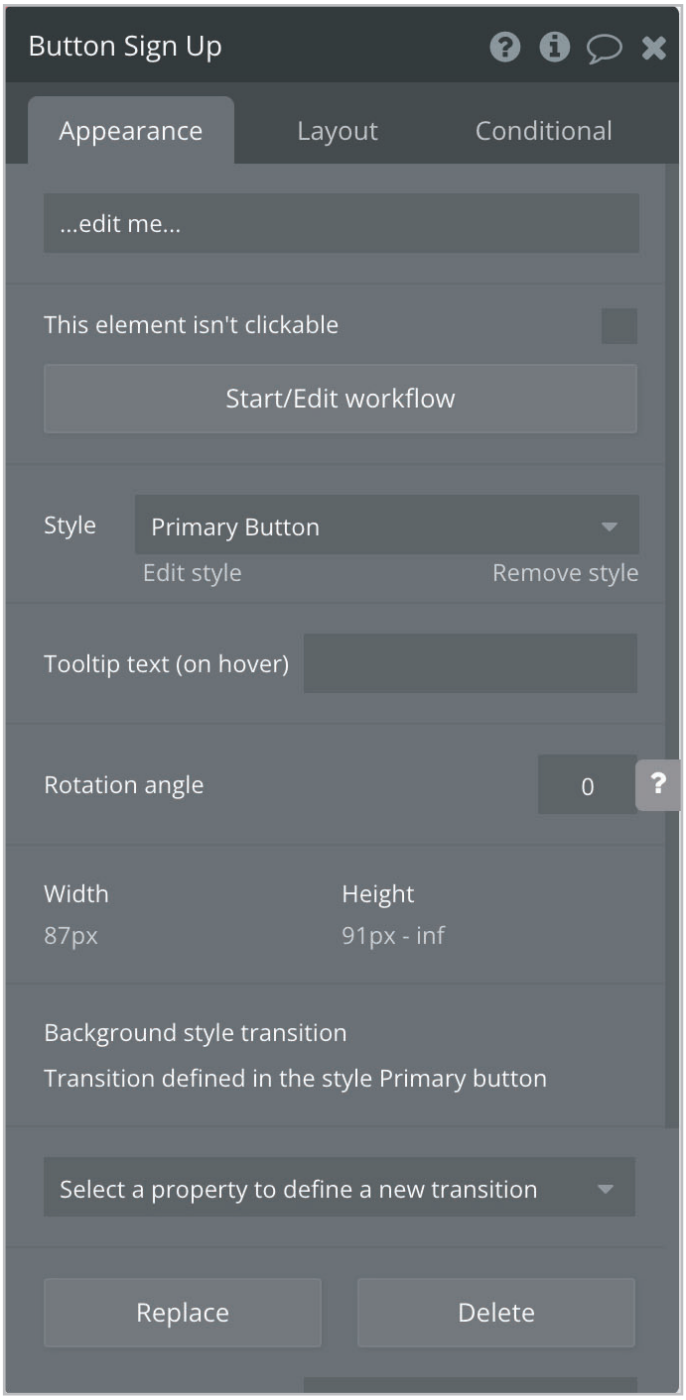
02 CHAPTER 주요 도구

1. 프로퍼티 에디터

프로퍼티 에디터는 각 페이지의 다양한 요소들을 디자인, 이벤트 커스터마이징하는데 쓰이는 메인 패널입니다. 이 에디터는 드래그로 이동 가능한 팝업이고 요소를 위해 채워 넣을 수 있는 필드들로 구성되어 있습니다. 이 에디터는 요소를 더블 클릭하거나 이벤트 또는 액션에서 한 번 클릭하면 나타납니다. 가장 위에 있는 바에서는 현재 편집하고 있는 요소/액션/이벤트의 이름을 수정할 수 있습니다. 그리고 상단 바의 오른쪽에 위치한 아이콘들은 요소 인스펙터를 보여주거나, 댓글 패널을 보여주거나, 문맥에 맞는 설명 비디오를 보여주기도 합니다.

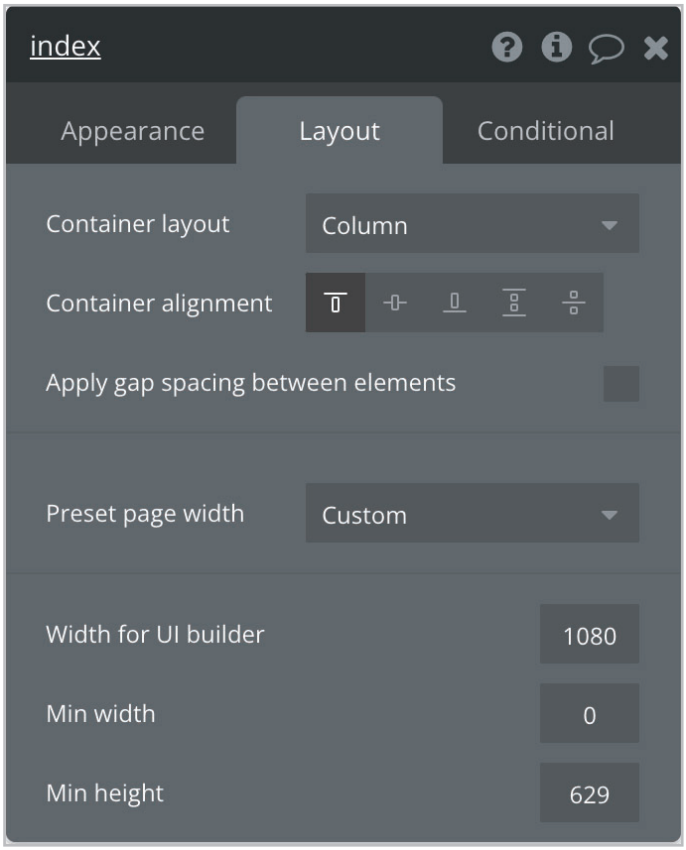
상단 바 바로 아래 3개의 탭이 있는데 각각에 대해 차례로 설명하면 다음과 같습니다.

(1) Appearance 외관



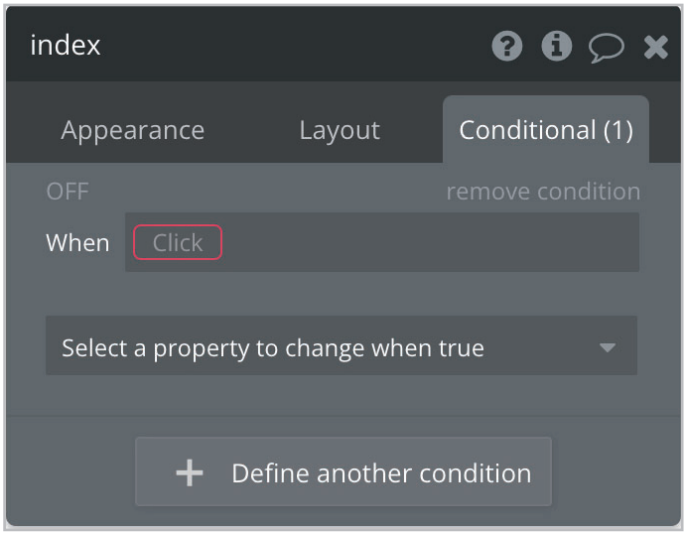
요소의 전체적인 외관과 관련된 것들을 편집할 수 있습니다. 이 탭에는 데이터 타입, 데이터 소스, 스타일, 등을 수정할 수 있게 구성되어 있습니다.

(2) Layout 레이아웃



페이지 내에서 요소의 레이아웃에 관련된 값들을 조절할 수 있는 탭입니다. 여기에서는 요소의 넓이, 높이, 컨테이너 레이아웃 타입, 보이기/숨기기 유무, 마진과 패딩 값 등을 설정할 수 있습니다.

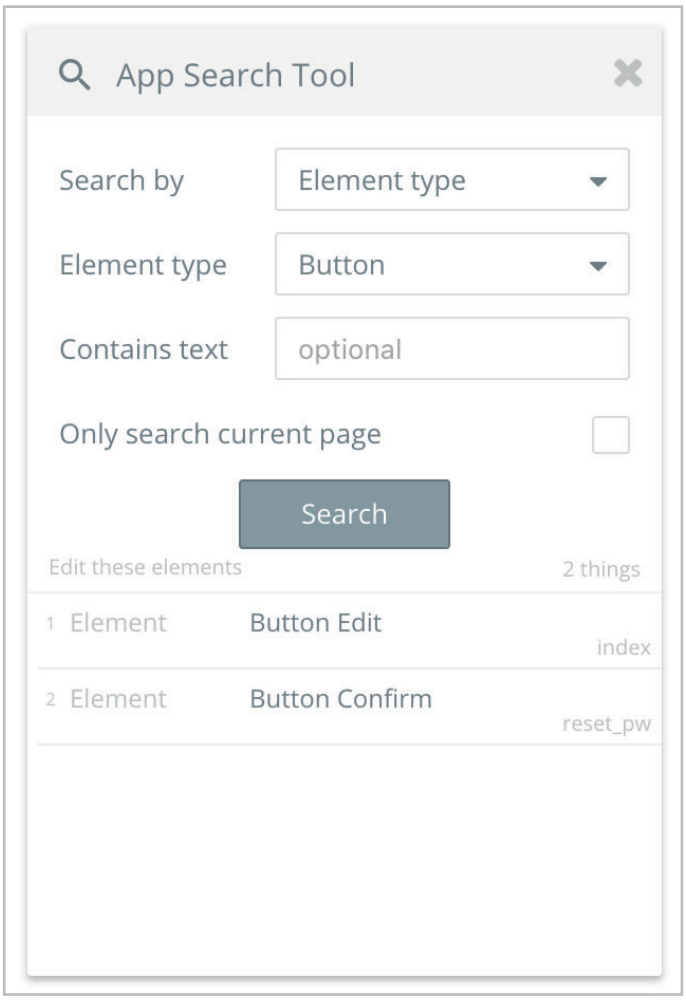
(3) Conditionals 조건



조건 탭에서는 요소가 특정 상황이나 조건에서 어떻게 행동하고 나타날지 정하는 탭입니다.

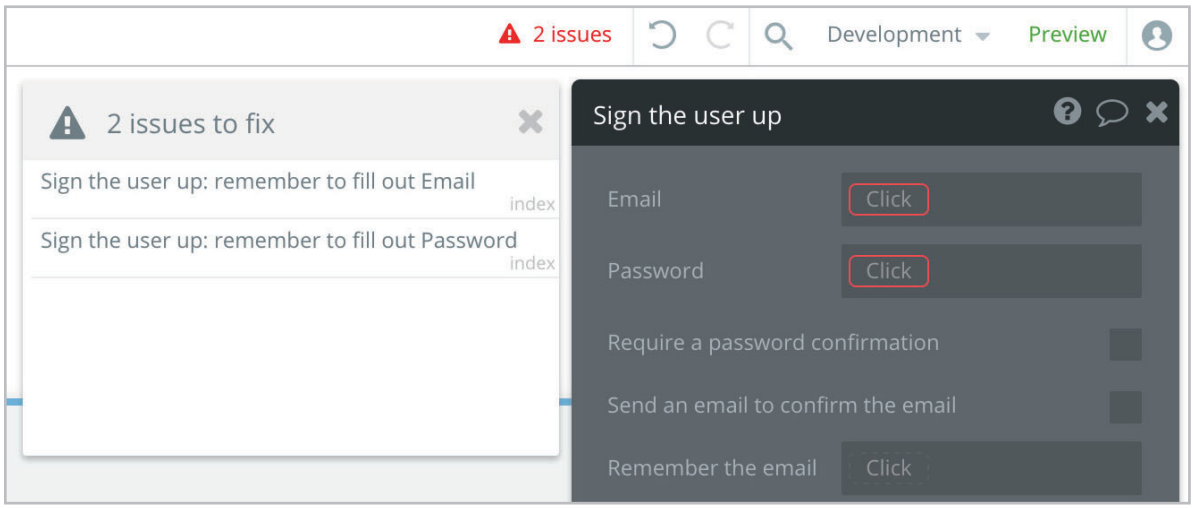
2. 앱 서치 툴

제작을 하다보면 대개 여러 개의 페이지와 요소들을 생성하게 됩니다. 예를 들어 서비스에 쓰인 모든 버튼을 찾고 싶은 경우 등에 쓰이는 도구가 바로 이 앱 서치 툴입니다. 이 도구로 요소, 액션과 같은 것들을 검색할 수 있습니다. 아래 결과로 나온 부분을 클릭하면 해당 페이지의 해당 부분으로 이동시켜 줍니다.



3. 이슈체커

버블을 사용하다보면 대부분 본의 아니게 실수를 하게 됩니다. 이는 자연스러운 과정입니다. 이 실수들은 필수 필드를 비워놓았다거나 숫자를 써야 하는 상황에 텍스트를 쓰는 것과 같이 부적절한 타입을 선택했을 수도 있습니다. 이슈체커는 여러분들이 고쳐야 하는 이슈들을 실시간으로 목록화 해줍니다. 우측 상단에 빨간 경고의 아이콘이 보인다면, 클릭하여 목록을 확인합시다. 서치 툴과 비슷하게 목록을 클릭하면 고쳐야 하는 이슈로 이동시켜 줍니다.



항상 이 이슈목록을 최대한 적거나 없게 만들어야 합니다. 이슈들이 있는 상태로는 run-mode 즉 배포를 할 수 없기 때문입니다.

4. 미리보기

버블에서 서비스 제작을 했다면 실행을 해봐야 합니다. 우측 상단의 Preview를 누르게 되면 실제 작동할 서비스 화면을 보여주게 됩니다. 만들면서 언제든지 눌러 실제 서비스될 화면을 확인할 수 있습니다.

5. 버전 콘트롤

만들고 있는 과정에서 여러분들은 어떤 특정 시점으로 시간을 돌리고 싶을 수도 있습니다. 만약 지우면 안 되는 페이지를 지웠는데 뒤로 가기로 복구가 안 되는 상황 같은 경우에는 지워지기 전 시점으로 돌리게 되면 없어진 페이지를 살릴 수 있습니다.

03 CHAPTER 주요 기능

1. 실시간 저장

버블의 주요 기중 중 하나는 제작 중인 앱을 자동으로 실시간 저장을 지원합니다. 요소, 액션, 세팅 등을 변경하자마자 그 수정사항들도 저장됩니다. 상단에 작은 글씨로 'saving' 알림 메세지를 볼 수 있고 만약 저장이 끝난다면 'saved'로 문구가 바뀐 모습을 볼 수 있을 겁니다.

가끔 이 글씨가 빨간색으로 변할 때가 있는데, 이러한 이유는 인터넷 연결이 불안정하여 저장을 제대로 실행시킬 수 없을 때 나타납니다. 이럴 때는 작업을 멈추고 인터넷이 안정적으로 돌아올 때까지 기다리면 됩니다. 새로고침을 눌러서 정상 상태로 되돌려줄 수도 있습니다. 하지만 이때 마지막으로 한 작업이 날아 갈 수 있습니다. 그렇기 때문에 저장 경고 상태가 나오면 작업을 일단 멈추는 게 좋습니다. 경고가 뜬 상태 이후에 작업된 내용들이 소실될 수도 있기 때문입니다.

2. 뒤로 가기, 앞으로 가기

모든 수정사항들은 뒤로 혹은 앞으로 갈 수 있습니다. 상단에 있는 화살표 모양의 아이콘을 클릭하면 뒤로 혹은 앞으로 가는 것을 적용할 수 있습니다. 다만, 보여 지는 수정사항이 아니라면 이 아이콘을 클릭했을 때 적용되었는지 안 되었는지 확인하기 어려울 수 있으니 주의해야 합니다.

3. 다중 편집 기능

현업에서도 하나의 서비스를 여러 명이 만들 듯이 버블에서도 여러 명이 동시에 제작을 할 수 있습니다(이 기능은 유료 플랜에서만 적용 가능합니다). 다른 사람이 앱을 편집하고 있으면 여러분은 다른 편집자의 마우스 포인터의 이동을 볼 수 있습니다. 이 기능은 여러 사람이 같은 요소를 동시 편집을 피하도록 도와주는 역할을 합니다.

4. 숏컷

숏컷은 말 그대로 지름길을 의미합니다. 개발에서는 어떠한 작업을 빠르게 해주는 용도로 사용한다고 생각하면 됩니다. 그때그때 개발을 빠르게 하는 방법이 모두 다르기 때문에 우측 상단의 물음표 아이콘 내에서 확인해도 되고, 필요한 상황에서 패널에 적절히 뜨기 때문에 굳이 우리가 찾아내려고 하지 않아도 됩니다.

5. 레퍼런스 페이지 이동

개발을 하다보면 특정 필드나 요소의 기능이 궁금할 수 있습니다. 이때 아이콘이나 필드를 수 초간 마우스에 대고 있으면 나오는 툴팁 'See reference'를 눌러봅시다. 그러면 새 탭에서 해당 부분에 대한 세부사항을 볼 수 있는 페이지가 나오게 될 겁니다.

PART 05

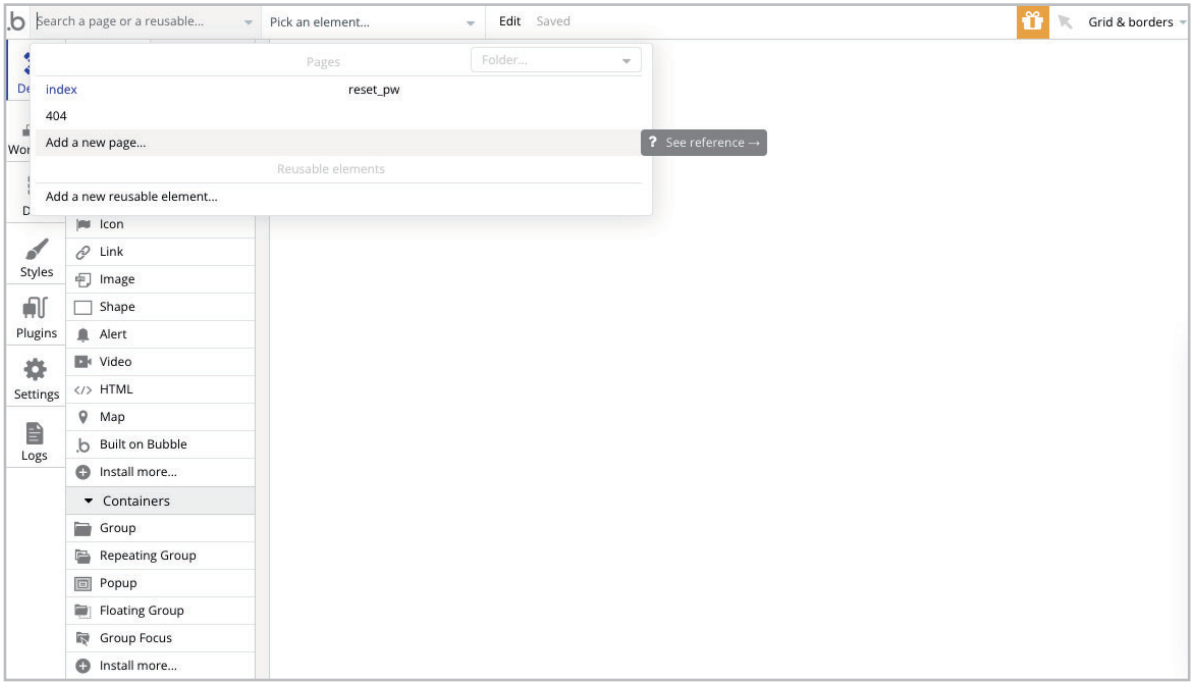
노코드 서비스를 만들기 위해 필요한 버블 기능들

01 CHAPTER 페이지 구축

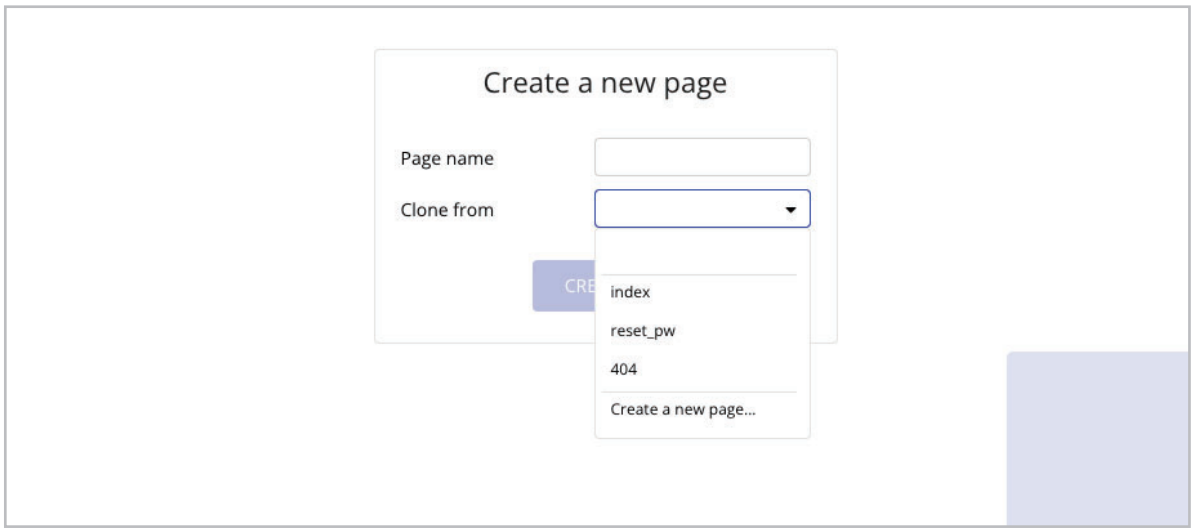
하나의 서비스를 만들기 위해서는 로그인/회원가입, 메인페이지 등 많은 페이지가 필요합니다. 이러한 페이지들을 버블에서 만들고 서로 연결시키는 방법을 살펴 보겠습니다.

1. 페이지 추가

페이지를 추가하는 방법은 에디터 상단에서 할 수 있습니다.



상단에 페이지 이름이 적혀있는 드롭박스를 클릭한 뒤 “Add a new page…”를 클릭합니다.



Page name에 페이지 이름을 적습니다. 이 부분은 추후에 수정할 수 있습니다. 이외에도 기존 페이지에서 복제된 새로운 페이지를 만들고 싶다면 아래 Clone from에서 복제하고 싶은 페이지를 선택해주면 됩니다. 이 경우 복제할 페이지의 기능까지 동일하게 복사됩니다.

2. 페이지 간 연결

사용자가 여러분들이 만든 서비스의 페이지들을 이동하려면 두 가지 방법이 있습니다. 링크 요소를 사용하거나, 워크플로우의 'Go to page action'을 사용하는 것입니다. 참고로 버블에서는 워크플로우 진행 중에 페이지 이동이 필요한 게 아니라면, 링크 요소를 추천한다고 합니다. 왜냐하면 화면 전환이 약간이라도 더 빠르고 유저가 우 클릭을 통해 새로운 창에서 열게 해줄 가능성도 제공하기 때문입니다. 필자도 동일한 의견인데 실제 서비스를 운영하다 보면 매 순간 버블이 수용할 수 있는 워크플로우의 수가 정해져 있습니다. 따라서 링크 요소로 페이지 이동의 워크플로우를 줄일 수 있다면 다른 워크플로우가 실행될 수 있는 횟수를 확보할 수 있습니다.

페이지 이동에서 주의할 점은 페이지가 전환되고 나면 워크플로우는 종료됩니다. 그러므로 화면을 전환하는 액션은 워크플로우에서 가장 마지막 단계에 이루어져야 합니다. 그렇지 않으면 이슈체커가 이슈 목록에 에러를 남깁니다. 하지만 페이지 이동 액션에 조건을 걸어놓으면 이슈체커가 이슈 목록에 남기지 않게 되니 주의하여 사용해야 합니다.

이외에도 도착 페이지가 콘텐츠 타입을 가지고 있다면 링크 요소와 이동 워크플로우를 작성할 때 해당 타입도 같이 전달해주어야 합니다.

3. 멀티 페이지 서비스 vs. 싱글 페이지 서비스

서비스를 개발할 때는 페이지에 관해서 크게 두 가지의 옵션이 있습니다. 바로 멀티 페이지 서비스이거나 싱글 페이지 서비스입니다. 우선 멀티 페이지 서비스는 다양한 페이지들을 만들어 놓고 사용자가 페이지 간 이동하며 상호작용하는 서비스입니다. 반면에 싱글 페이지 서비스는 하나 혹은 몇 개의 페이지 내에서 그룹들을 만들어 특정 조건 및 상황에서만 보여지고 숨겨지게 만드는 서비스입니다.

만약 여러분이 싱글 페이지 서비스를 만든다면, 조건부 핸들링과 커스텀 상태를 쓰는게 요소를 보이거나 숨기게 할 때 유용하게 작용할 것입니다. 퍼포먼스적인 관점에서 두 옵션은 각각 트레이드 오프가 발생합니다. 멀티 페이지 서비스는 페이지가 여러 개로 나누어져 있기 때문에 각 페이지의 로딩시간이 짧을 수 있습니다. 하지만 페이지의 전환이 자주 이루어져야 합니다. 반대로 싱글 페이지 서비스의 경우 첫 로딩이 길 수 있지만, 페이지 전환이 필요 없게 됩니다. 사실 서비스를 만들다 보면 결과적으로 이 두 경우의 어느 중간을 채택하게 될 확률이 높습니다.

서비스를 기획하는 많은 상황에서 원하는 방식을 구현해 내는 데는 하나 이상의 해결책이 있게 마련입니다. 이때는 경험이나 팀원들의 선호도에 따라 방법을 결정하게 될 것입니다.

4. [심화 Q&A] 페이지 이동

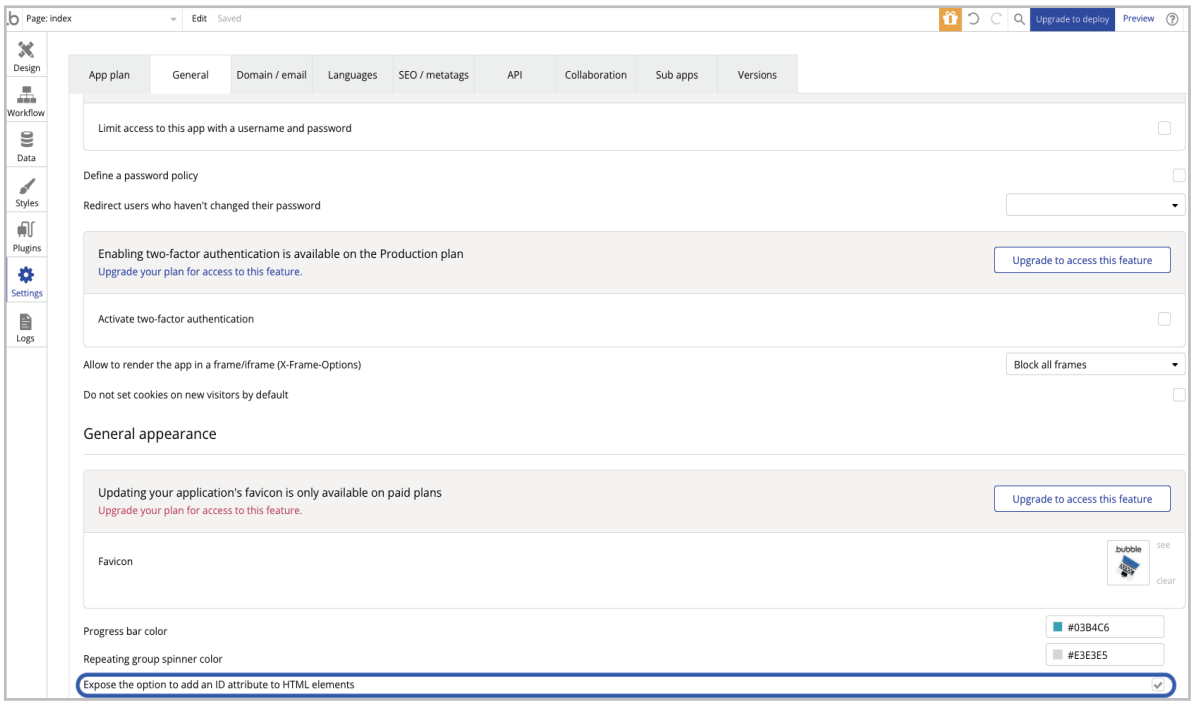
아래의 질문들은 서비스를 만들다 보면 생길 수 있는 심화 질문들이지만 해당 상황에 닥쳐본 적이 없으면 도움이 안 될 수도 있습니다. 따라서 위치만 기억해 놓았다가 추후에 동일한 질문이 개발하다 떠오른다면 찾아와서 해결해도 좋습니다.

새로고침이나 새로운 URL의 이동 없이 파라미터나 페이지 패스를 바꿀 수 있나요?

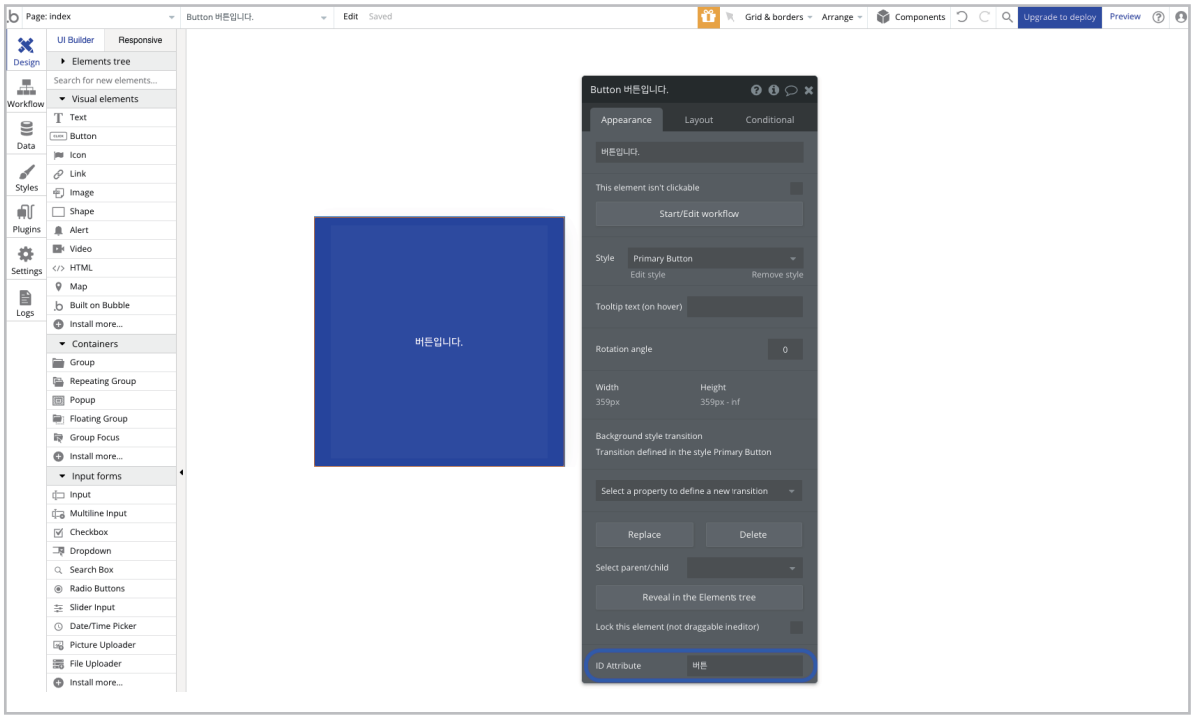
이 질문은 싱글 페이지 서비스를 만들 때 많이 궁금해 하는 질문입니다. 지금으로써는 버블 자체에서 화면을 바꾸는 선택지 외에는 URL path를 바꿀 수는 없습니다. 다만, 버블 생태계에 있는 플러그인 중에서는 해당 요청을 기능적으로 지원하는 플러그인이 있습니다.

앵커 링크를 실행할 수 있나요? (예: URL에 #오브젝트가 있으면 #오프젝트가 있는 곳으로 스크롤 되는 기능)

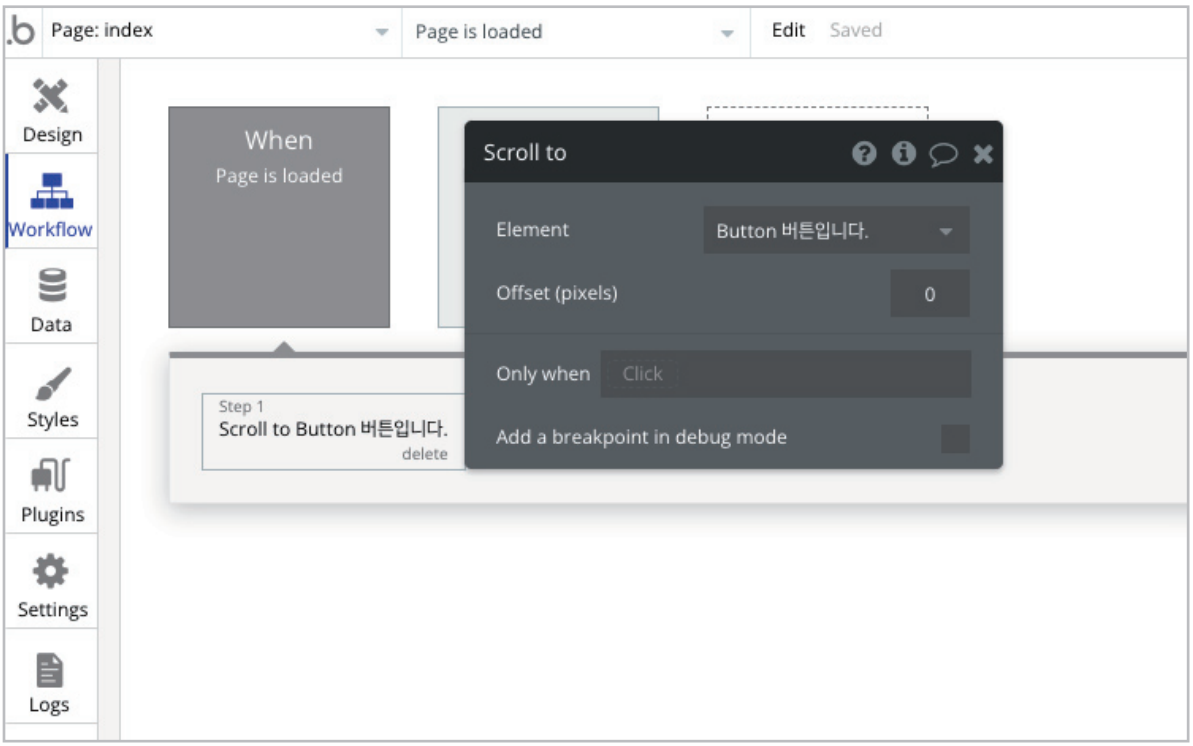
네 가능합니다. 해당 기능을 위해서는 일단 Settings > General > General Appearance에서 add HTML IDs to elements의 설정을 활성화합니다.



활성화가 되면 각각의 요소들의 Appearance 부분의 맨 아래에 ID Attribute값을 부여할 수 있습니다.



이제 페이지의 URL 맨 뒤에 #{부여한 id값}을 추가하게 되면 해당 요소로 스크롤 되게 됩니다. 이외에도 워크플로우 액션 중에 “Scroll to”액션을 사용하면 비슷하게 작동하도록 만들 수 있습니다.

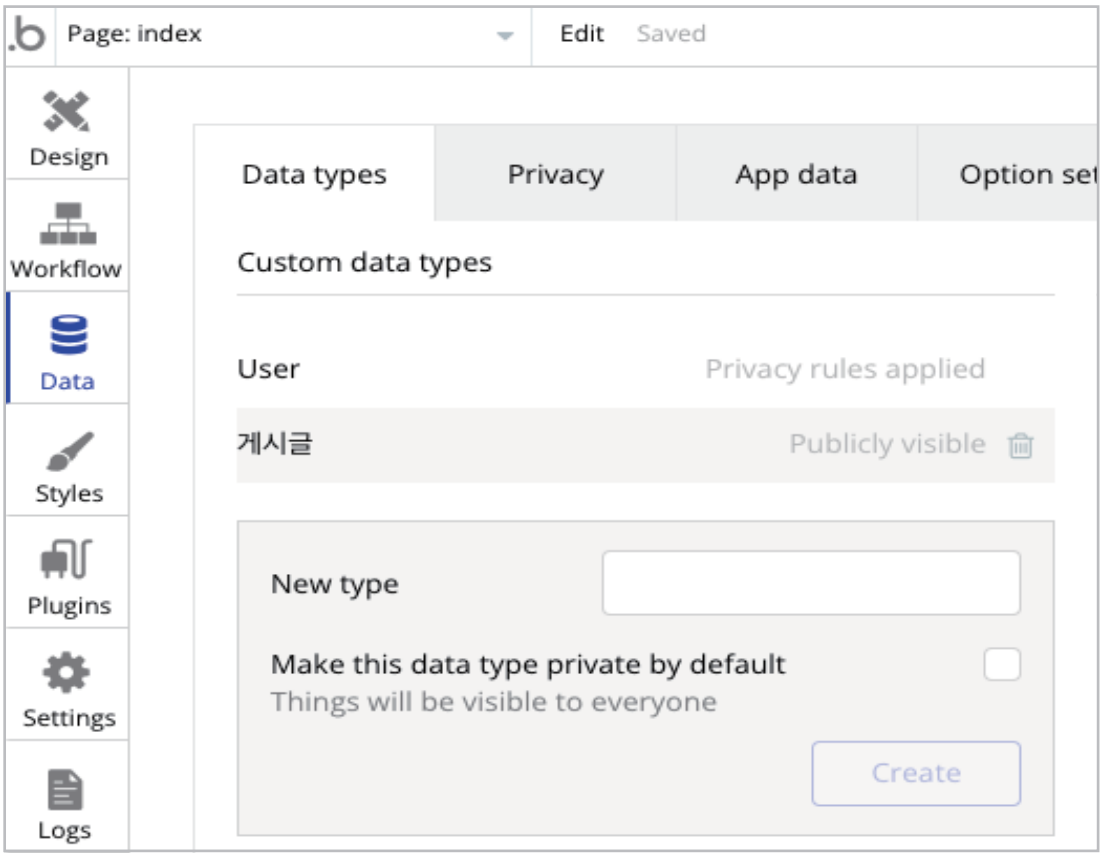

Page: index
Page is loaded
Edit
Saved
Design
Workflow
Data
Styles
Plugins
Settings
Logs
When
Page is loaded
Scroll to
Element
Button 버튼입니다.
Offset (pixels)
0
Only when
Click
Add a breakpoint in debug mode
Step 1
Scroll to Button 버튼입니다.
delete

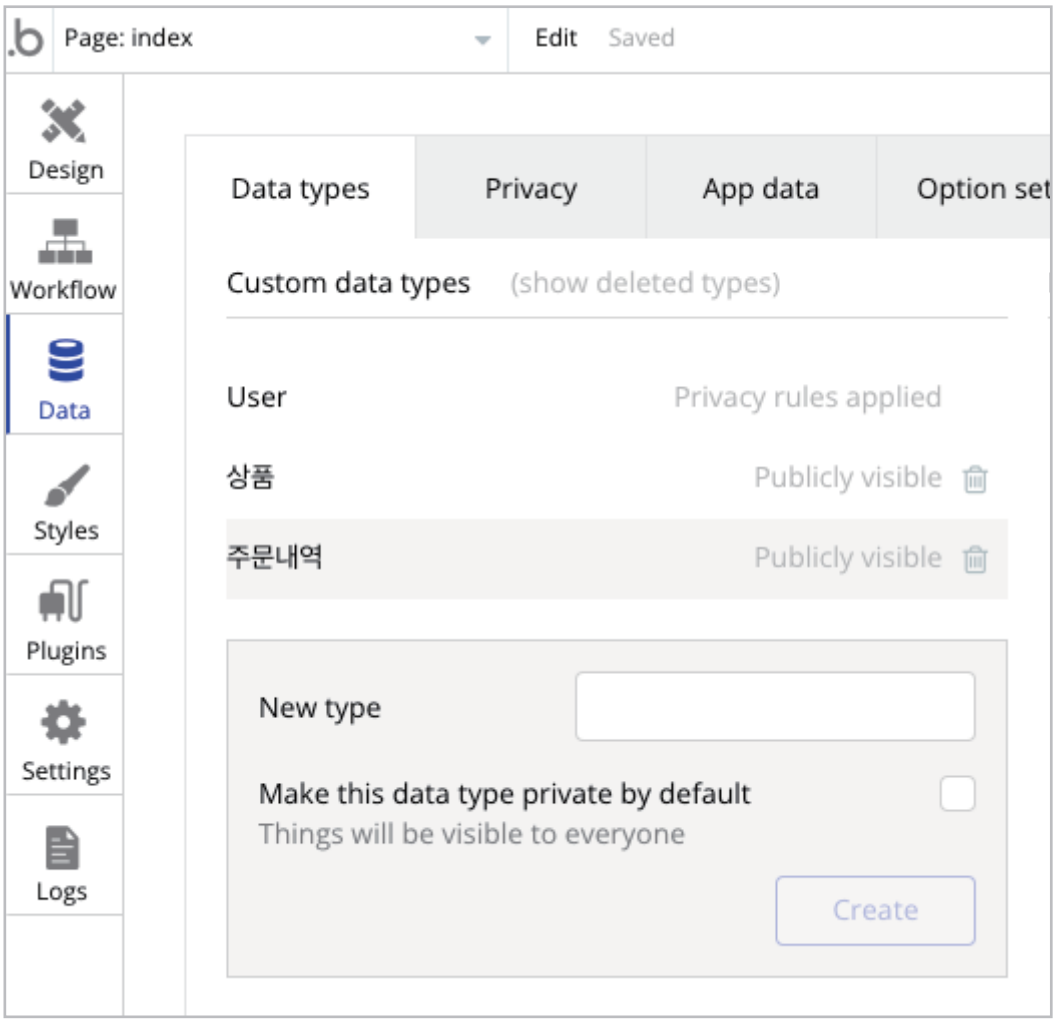
02 CHAPTER 데이터 설계

서비스를 만들 때 데이터 구조를 설계하는 것은 아주 중요합니다. 이러한 과정은 데이터의 타입과 필드를 정의하는 과정으로 진행됩니다. 천천히 학습하면서 서비스에 필요한 데이터를 어떻게 정의하는지 알아봅시다.

1. Type 타입

데이터가 필요한 서비스를 만들 때 가장 상위 개념에 해당되는 것이 바로 데이터 타입입니다. 서비스가 작동할 때 상호작용해야 하는 데이터의 주체라고 생각하면 편합니다. 예를 들면 우리가 인스타그램을 만든다고 했을 때, '사용자' 타입과 '게시글' 타입이 필요합니다. 다른 예시로 쇼핑몰을 만든다고 했을 때, '사용자' 타입과 '상품' 타입이 필요하고 구매를 한 기록을 남길 수 있도록 '주문내역'타입도 필요합니다. 이렇게 데이터의 타입을 정의하는 일은 데이터베이스가 어떤 정보들을 포함하고 있을지 정하는 첫 번째 단계입니다.

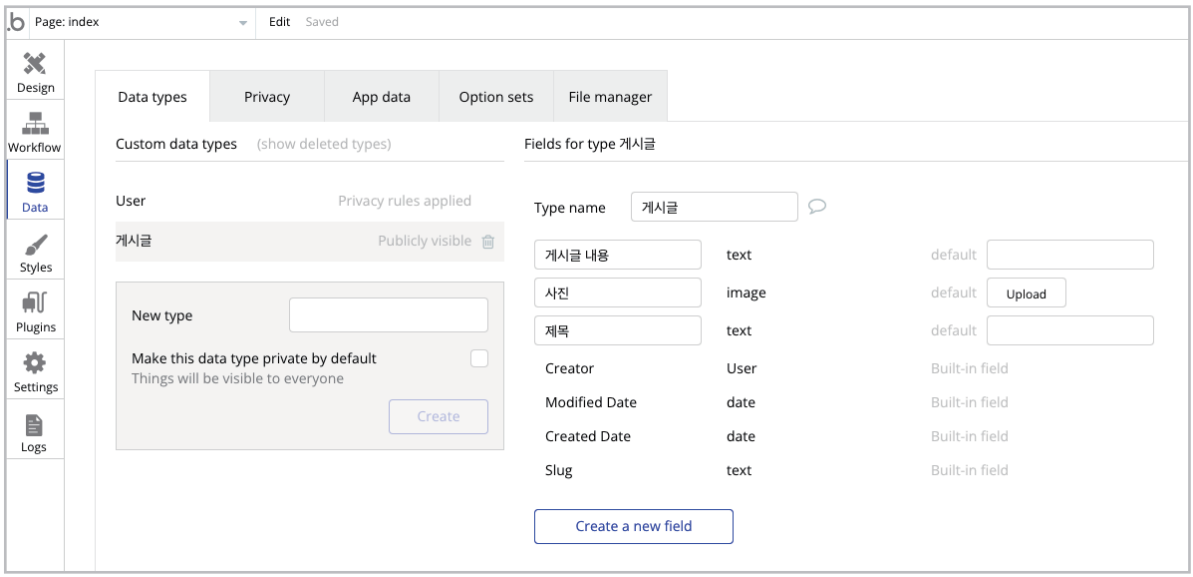




2. Field 필드

필드는 앞서 정의한 데이터 타입 안에 구체적으로 어떤 정보들이 들어있어야 하는지에 해당되는 부분입니다. 앞의 타입의 예시를 가져와 이어서 설명해 보겠습니다. 첫 번째 예시에서 인스타그램의 '게시글' 타입 안에는 '사진' 필드와 '제목' 필드, 그리고 '게시글 내용' 필드가 필요합니다. 두 번째 예시인 쇼핑몰에서 '상품' 타입 안에는 '상품명', '가격', '상품 상세설명' 필드가 필요하게 됩니다. 이렇게 데이터 타입 안에 필요한 필드들을 정의해 놓아야 특정 데이터를 원하는 형태로 저장할 수 있게 됩니다. 원하는 형태로 저장한다는 의미는 반대로 원하는 형태로 불러올 수 있다는 의미와 동일합니다.

이어 필드를 만들 때 알아두어야 할 사항은 다음과 같습니다. 필드를 정의할 때는 우선 필드의 타입을 선택해야 합니다. 위의 예시를 끌고 와 설명해보면, '게시글' 타입 안에 있는 '사진' 필드는 이미지여야 하고, '제목' 필드와 '게시글 내용' 필드는 텍스트여야 합니다.

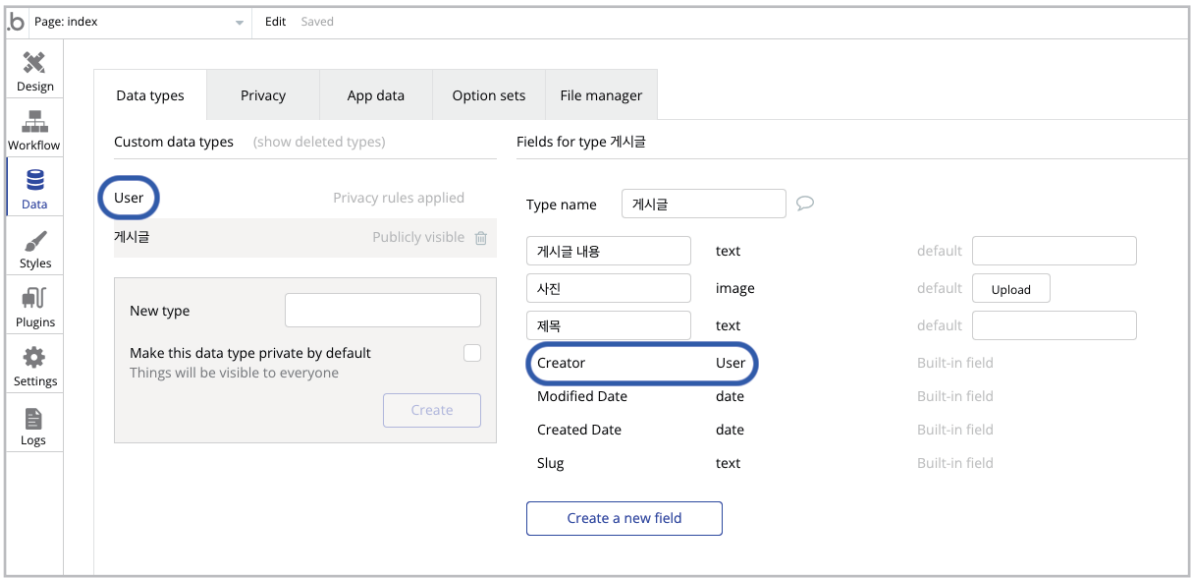


필드를 정의할 때 알맞은 필드 타입으로 지정해 줍니다. 아래는 버블에서 제공하는 필드 타입입니다.

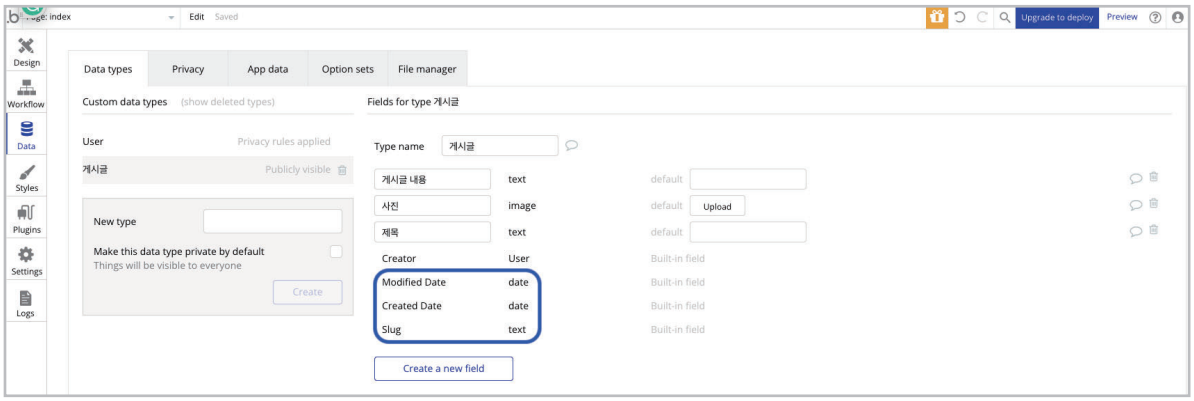
- text : 텍스트 형태 데이터
- number : 숫자 형태의 데이터
- yes/no : 예, 아니요로 나뉠 수 있는 데이터 (컴퓨터공학에서는 bool/boolean 타입이라고 함)
- date : 날짜 형태의 데이터
- geographic address : 장소 형태의 데이터
- image, file : 이미지나 파일 형태의 데이터
- number interval : 숫자 간격 형태의 데이터
- date interval : 날짜 간격 형태의 데이터

필드 타입을 결정하는 것은 후에 서비스를 만들 때 데이터를 어떤 식으로 사용할 수 있는지를 결정하는 것이기 때문에 중요합니다. 예를 들자면 text 타입은 서로 연결되거나 잘라낼 수 있고, number 타입은 서로 더하거나 뺄 수 있으며, address 타입은 지도상에 표현할 수 있습니다.

추가로 빌트인 필드 타입이란, 만든 데이터의 타입 Type이 필드의 타입이 될 수 있다는 것입니다. 이는 다소 어려운 말로 "합성 타입"이라고 하며, 데이터베이스 상에서 서로 다른 두 요소를 연결하는 방법으로 쓰입니다. 예를 들어, '게시글' 타입 안에는 '작성자'라는 필드가 필요합니다. 이 '작성자' 필드는 초반에 정의했던 '사용자' 타입으로 정의될 수 있습니다.



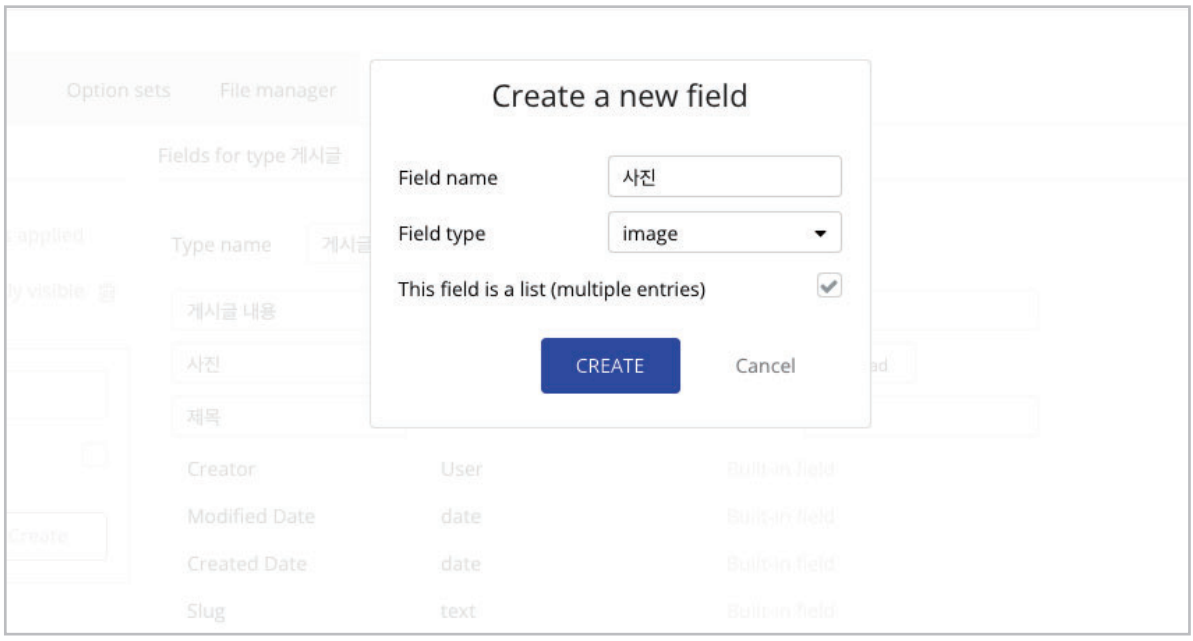
여기에 추가로 버블에서 데이터 관리의 편의를 위해 각각의 타입이 만들어질 때마다 생성되도록 한 필드 타입이 있습니다. 그것은 데이터가 생성된 날짜를 기록하는 'Created Date'와 수정된 날짜를 기록하는 'Modified Date'입니다. 이외에도 User 타입을 제외한 데이터들은 모두 생성자 타입인 User를 가지고 있습니다. 그러므로 서비스 상에서 데이터가 생성될 때 자동으로 생성한 User 타입도 같이 기록됩니다. 모든 타입은 'Slug'라는 필드도 가지고 있는데 데이터를 참조할 수 있는 또 다른 방법이며 보통 URL에 데이터의 정보를 나타낼 때 쓰입니다.



버블에서 기본으로 제공하는 이러한 빌트인 필드들을 사용하면 데이터의 생성과 수정에 관련된 좀 더 정확한 정보를 얻을 수 있으니 적극적으로 활용하도록 합시다.

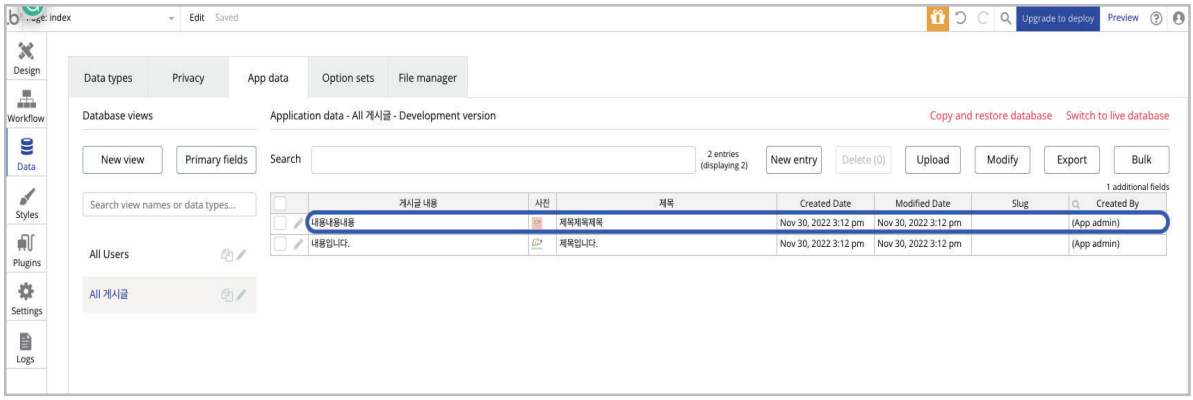
마지막으로 필드의 타입을 정의할 때 해당 필드에 값이 하나가 들어가게 될지, 여러 개가 들어가게 될지 정해주어야 합니다. '게시글' 타입 안에 있던 '사진'이 하나 밖에 들어갈 수 없다면 '사용자'들은 '게시글'당 무조건 하나의 사진 밖에 올리지 못할 것입니다. 하지만 '게시글'의 '사진'을 List of images로 만들어 놓는다

면 사람들은 게시물에 여러 개의 사진을 올릴 수 있게 됩니다. 이렇게 필드 타입을 리스트로 만드는 방법은 필드 타입을 정의할 때 아래의 'This field is a list (multiple entries)'를 체크하면 됩니다.

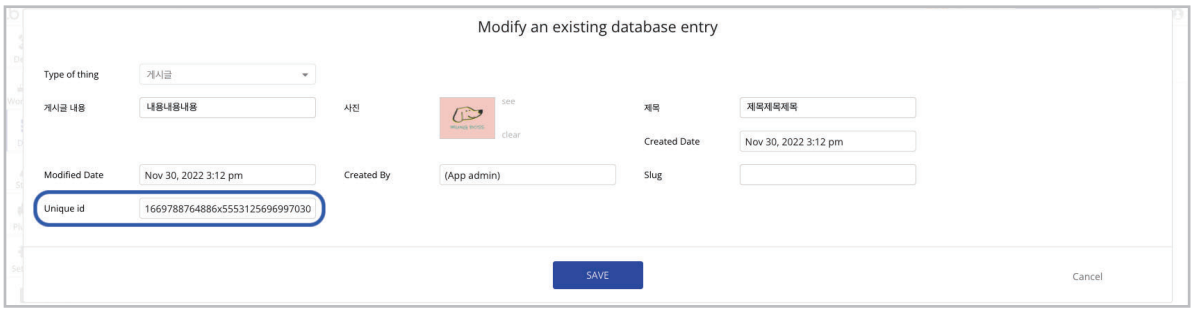


3. Things 레코드

버블에서는 데이터 타입별로 저장된 하나의 데이터를 Thing이라고 부릅니다. Thing을 국문으로 번역하면 '것'이기 때문에 편의상 '레코드'라고 하겠습니다. 실제 컴퓨터공학에서 데이터베이스에 저장된 하나의 데이터를 레코드라고 합니다.



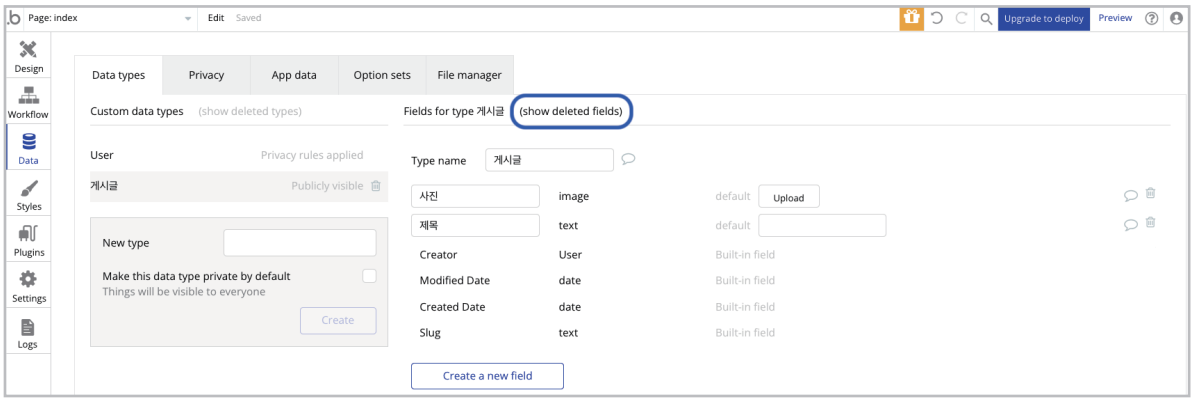
각각의 레코드들은 생성될 때 unique ID 값을 부여받게 됩니다. 이 값은 중간에 'x'가 포함되어있는 랜덤 문자열입니다. 이 필드를 쓸 일은 보통은 없지만 특정 레코드를 식별할 때 사용되곤 합니다.

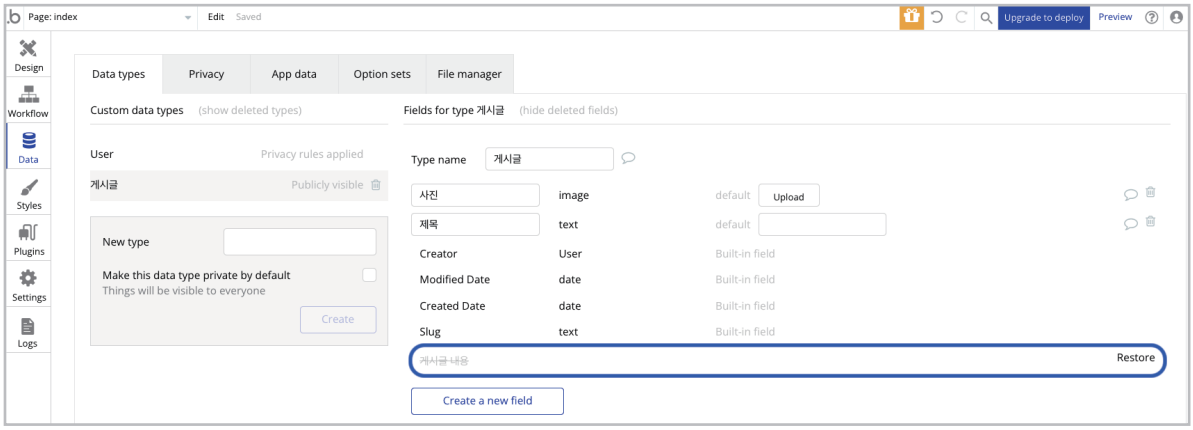


4. 타입과 필드 생성

버블에서 데이터의 타입과 필드를 생성하는 방법은 크게 두 가지로 구분됩니다. Data 탭에서 만들거나 화면 혹은 워크플로우를 구성하는 동안에도 만들 수 있습니다.

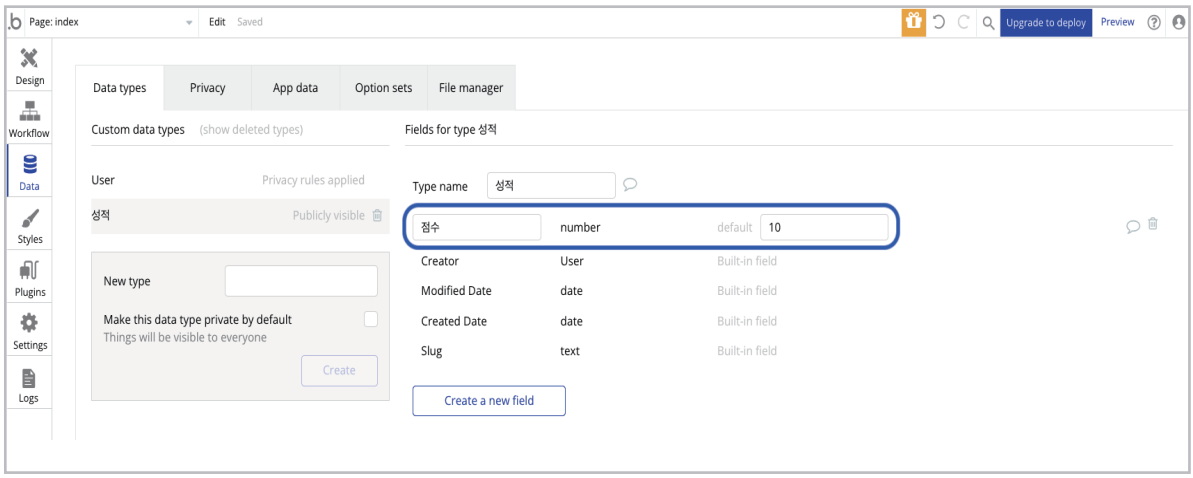
간혹 타입 혹은 필드를 잘못 만들었을 때, 삭제하고 싶은 순간이 있을 수 있습니다. 이때 타입 혹은 필드를 삭제하더라도 버블에서는 실제로 삭제된 게 아닙니다. 왜냐하면 이미 저장된 데이터들의 필드를 지울 수 없으니 일단 숨겨 놓는 방식이기 때문입니다. 그 말인즉슨, 복구하고 싶으면 복구가 가능하다는 말입니다. 복구하는 방법은 Data 탭에서 'show deleted types' 나 show deleted fields'를 누르게 되면 삭제했던 것이 보여지고 'restore'를 누르게 되면 마침내 복구가 진행됩니다.





5. 디폴트 값

데이터 탭에서는 추가로 각 필드의 기본 값을 설정해 놓을 수도 있습니다. 이 기본 값은 새로운 레코드가 하나 생길 때 사용됩니다. 예를 들어 레코드를 하나 생성할 때 '점수'라는 필드가 10부터 시작했으면 좋겠다면 default값에 10을 적어 놓으면 됩니다.



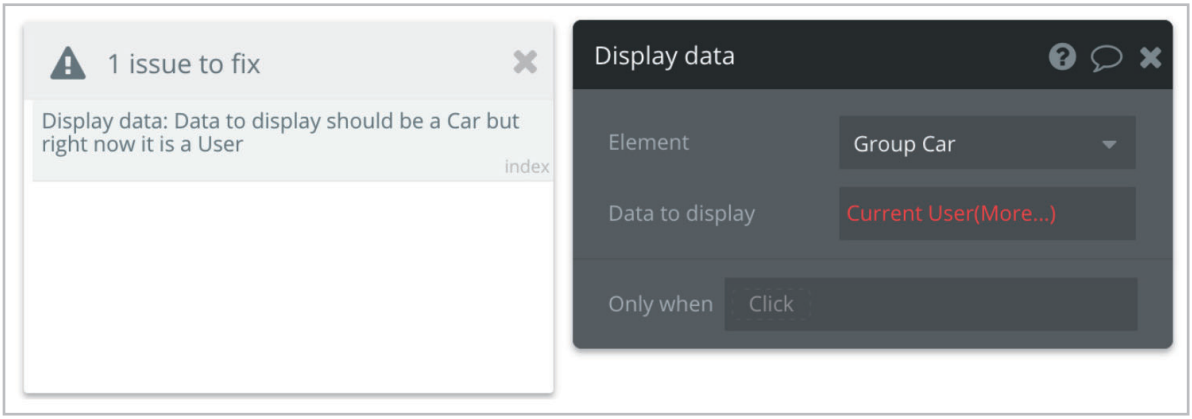
6. User 타입

버블은 서비스가 유저 기반으로 돌아간다고 가정하고 만든 툴입니다. 유저들은 회원가입, 로그인, 로그아웃 등의 액션을 취할 수 있게 됩니다. 그러므로 버블에서 만든 각각의 앱들은 유저 타입인 'User' 타입을 기본으로 지원합니다. 'User' 타입은 다른 타입과 비슷하게 작동되지만 가장 중요한 차이점은 서비스가 누구에 의해 실행되고 있는지를 알 수 있게 해준다는 점입니다.

또한 이 'User'타입은 몇 개의 빌트인 필드를 가지고 있습니다. 가장 중요하지만 기본적인 'email' 필드가 있습니다. 이 필드는 회원가입을 할 때 활용될 수 있습니다. 그 외에도 유효 링크를 보내 소유권을 확인할 수도 있습니다.

7. [주의] 타입 모순

앞서 말했듯이, 이슈체커는 여러분들이 서비스를 개발하는 도중에 만나는 오류들을 잡아줍니다. 그 중에서도 가장 많이 만나게 될 이슈는 바로 "타입 모순"입니다. 요소나 액션들은 각자 원하는 데이터 타입이 있는데 다른 타입을 대입하도록 개발했다면 이슈체커가 이를 체크해 고치기를 요구할 것입니다. 에러가 하나도 없어야 배포가 가능하므로 이러한 에러를 바로 잡는 것은 굉장히 중요합니다.



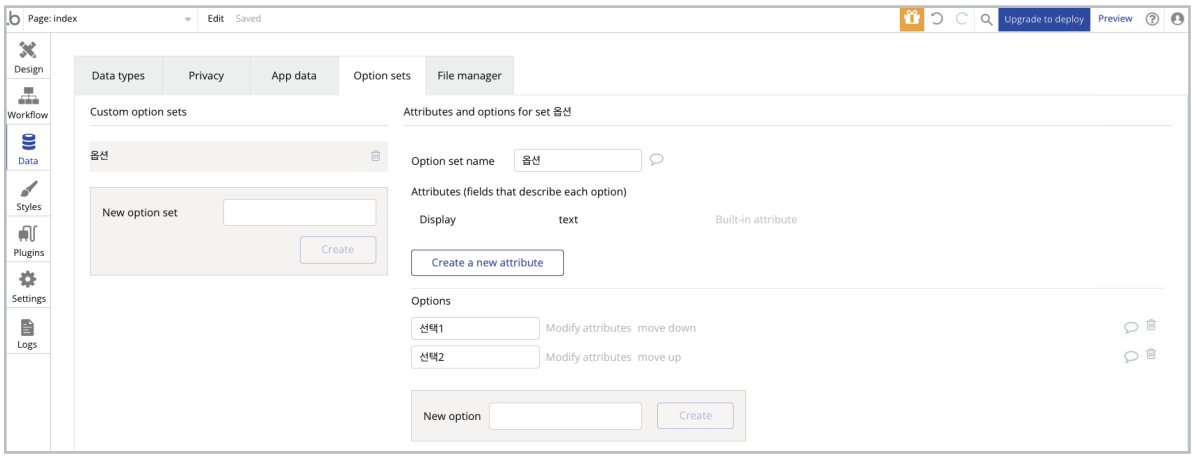
예시로, '자동차' 타입을 그룹에 나타내려고 했는데 Display data를 User로 했다면 Type끼리 맞지 않으니 에러를 띄우게 됩니다. 이러한 에러는 필드에서도 동일하게 적용됩니다. '자동차'타입의 가격을 저장해야 하는데 text를 저장하려고 하게 되면 에러가 나게 됩니다. 그러니 데이터의 타입끼리 그리고 필드끼리 같은 종류로 대입되도록 만드는 것을 주의해주시기 바랍니다.

03 CHAPTER 옵션 셋

옵션 셋은 정해진 옵션 리스트를 사용할 수 있게 해줍니다. 이 옵션들은 버블 앱 어디에서나 접근 가능합니다. 이 옵션 셋은 서비스에서 어떤 변수가 와야 하는지 바로 나오는 경우에 많이 쓰입니다. 예를 들면, 일주일의 요일, 상태, 팀 이름 등이 있습니다.

1. 옵션 셋 생성

옵션 셋은 커스텀 타입을 만드는 것 과 같이 동일하게 생성, 수정, 삭제할 수 있습니다.
Data탭의 Option set에서 설정이 가능합니다.



주의!

옵션 셋의 이름을 특별하게 짓는 것을 추천합니다. 만약 옵션 셋이 커스텀 타입과 이름이 동일하게 되면 버블이 두 데이터를 혼돈하여 예상치 못한 상황이 생길 수 있습니다.

주의!

옵션 셋은 앱 구조 안에 포함되어 있는 요소입니다. 이 말은 어떤 뛰어난 개발자들은 앱에 있는 옵션 셋을 모두 조회할 수 있다는 얘기입니다. 그러니 민감한 정보는 옵션 셋에 기재하지 않는 것이 좋습니다! 민감한 정보는 Privacy Rule을 적용한 데이터로 관리합시다.

2. 옵션의 속성 정의

옵션을 생성하고 나면, 그 옵션의 속성도 추가할 수 있습니다. 속성은 타입에서의 필드와 같습니다. 속성은 이름과 필드 타입으로 이루어져 있습니다. 이 속성들은 각각의 옵션에서 접근할 수 있습니다. Display라는 속성은 옵션을 만들 때 기본 생성되는 속성입니다. 이 Display 속성은 옵션의 이름이며 특히 드롭다운과 같은 곳에 옵션을 나열해서 보여줄 때 사용됩니다.

3. 옵션 추가

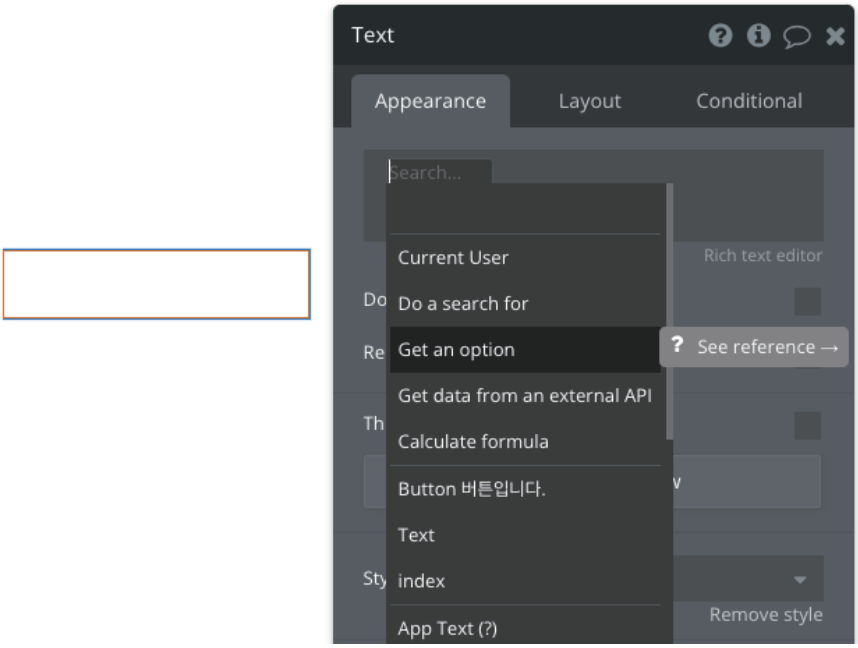
옵션은 생성된 옵션 셋 우측의 New option에 생성할 옵션의 이름을 적고 Create 버튼을 누르면 만들어 집니다. 옵션의 속성은 만들어진 옵션 옆에 Modify attributes로 편집 가능합니다. 그 옆의 move down / move up으로 정렬 순서도 바꿀 수 있습니다.

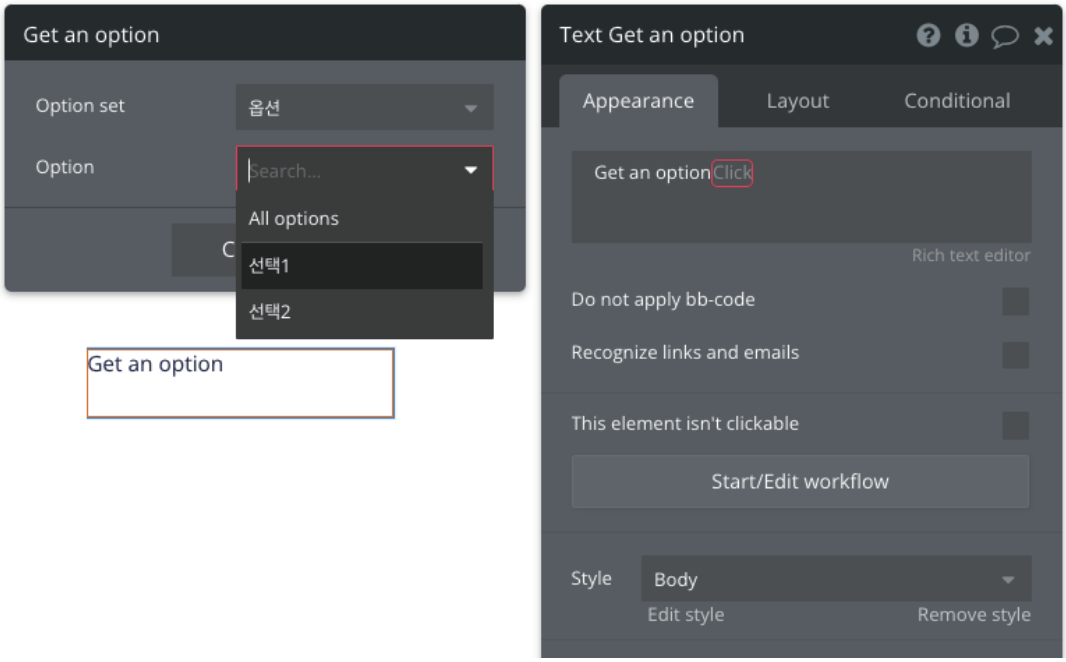
4. 옵션 사용

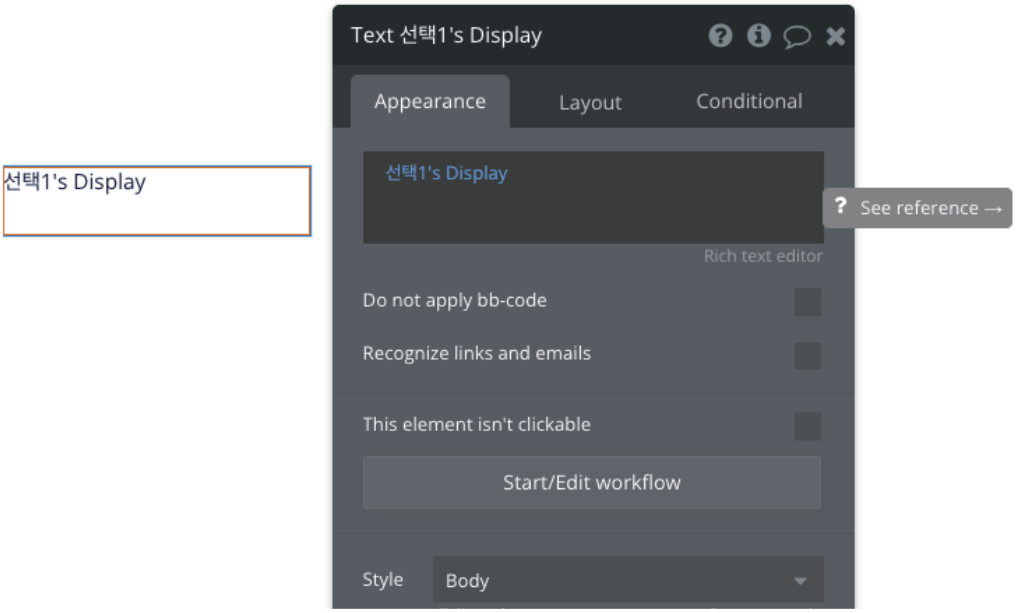
옵션은 보통의 커스텀 타입이 쓰이는 곳에는 어디에나 쓰일 수 있습니다(예: 레코드의 하나의 필드, 반복 그룹의 타입, 드롭 다운의 옵션 등). 특히 드롭다운의 선택지에 많이 쓰이긴 하지만 서비스 전반에 걸쳐 쓸 수도 있습니다. 이미 정해진 데이터를 빠르게 불러오는 곳에는 모두 쓰일 수 있고 데이터베이스에서 불러오는 것 보다 훨씬 빠르게 가져올 수 있습니다.

옵션을 사용하는 방법은 크게 두 가지가 있습니다.

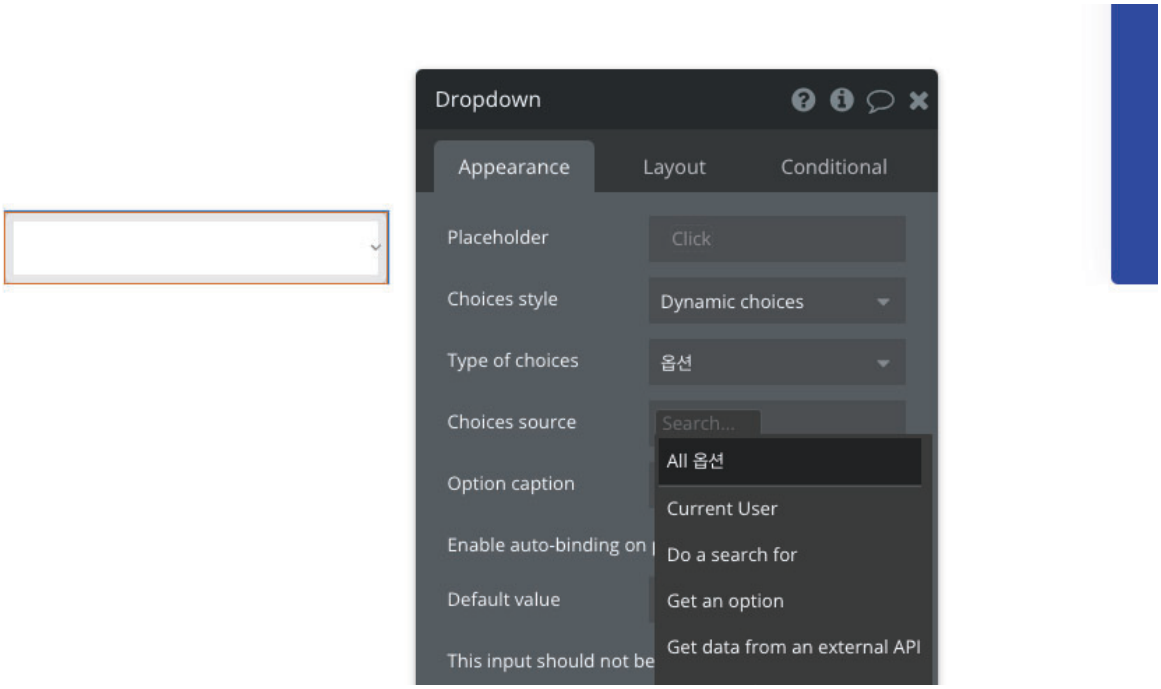
첫 번째 방법은 “Get an option”을 사용하는 방법입니다. 버블 앱 어디에서나 Insert Dynamic Data를 선택한 뒤 Get an option을 이용해 특정 옵션 셋의 전체 옵션 리스트, 혹은 하나의 특정 옵션을 사용할 수 있습니다. 이때 선택한 옵션의 Display 혹은 속성들에 접근할 수 있습니다.

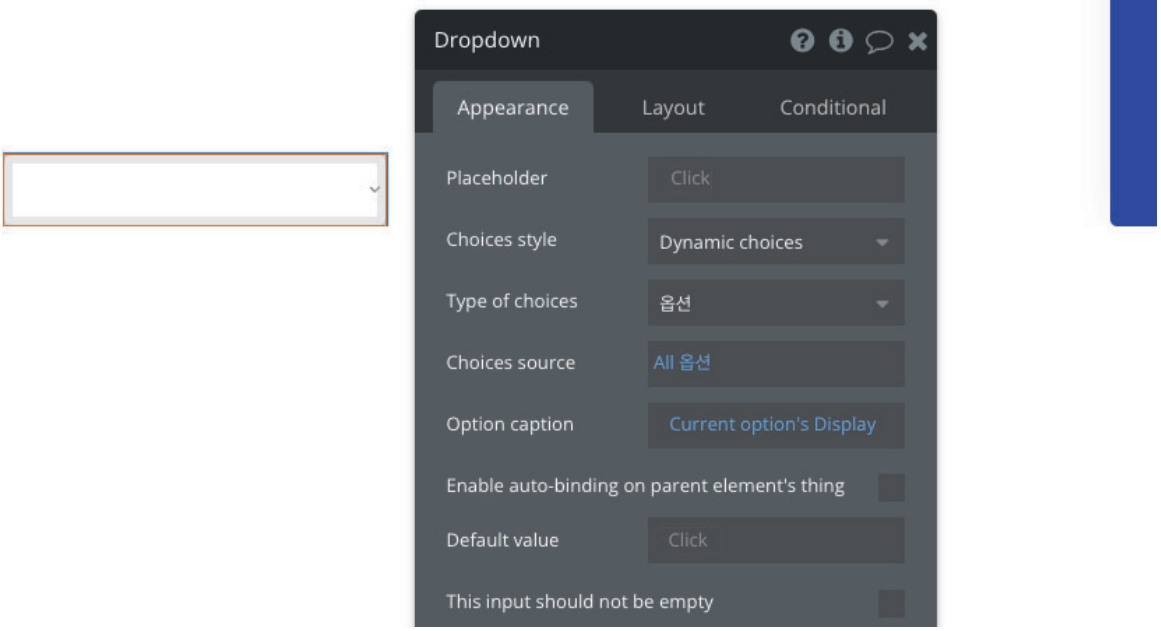






두 번째 방법은 옵션을 필요로 하는 요소에 자동으로 매칭해서 사용하는 방법입니다. 버블은 요소마다 필요한 조건들은 추천하는 데 능합니다. 만약 요소에 옵션이 필요하다면 해당되는 옵션 셋을 선택하고 모든 옵션 리스트를 선택지로 넣고 싶을 때 All {옵션 셋}을 사용하면 됩니다.





5. 옵션 vs. 커스텀 타입

앞에서 잠깐 언급했지만 옵션 셋을 쓰는 것이 커스텀 타입을 쓰는 것보다 기술적으로 많은 이점을 가지고 있습니다. 아래 나열된 이유를 보고 옵션 셋을 사용하는 것을 고려해 보는 것도 좋은 방법입니다.

- 옵션은 데이터베이스를 읽어올 필요가 없기 때문에 더욱 빠르게 접근 가능합니다.
- 옵션 셋은 옵션을 확장하거나 변경하는 게 쉽습니다.
- 버블 앱의 어느 곳에서나 접근할 수 있고 데이터베이스를 검색해서 쓰는 것보다 더 쉽게 가져올 수 있습니다.

그러나 커스텀 타입에 비해 몇 가지 단점도 존재합니다.

- privacy setting이 존재하지 않아 민감한 정보를 담기에는 부적절합니다.
- 일반 데이터는 앱의 라이브 버전에서 추가되고 변경될 수 있지만, 옵션은 변경한 후 새로운 버전으로 배포하여 반영해야 합니다.

04 CHAPTER 커스텀 상태

각각의 요소들은 그들에게 종속된 데이터를 가질 수 있습니다. 이를 바로 커스텀 상태라고 합니다. 이 개념은 고급 기능이긴 하지만, 요소에 데이터를 저장하고 싶을 때 매우 편리하게 사용될 수 있습니다.

1. 커스텀 상태 사용 예시

(1) 현재 페이지에서 보여 주어야 할 뷰를 저장할 때 사용할 수 있습니다. 예를 들면 페이지에 두 가지의 탭이 있는 상태에서 첫 번째 탭과 두 번째 탭을 전환시킬 때 사용됩니다. 커스텀 타입이 1일 경우 첫 번째 탭을, 2일 경우 두 번째 탭을 보여주면 됩니다. 유저가 페이지를 새로고침 하게 되면 마지막 상태는 초기화 됩니다.

커스텀 상태는 페이지 안에 특정 정보를 저장하고 싶지만 데이터베이스까지는 사용하고 싶지 않을 때 유용합니다. 아래에 두 가지 예시로 살펴보겠습니다.

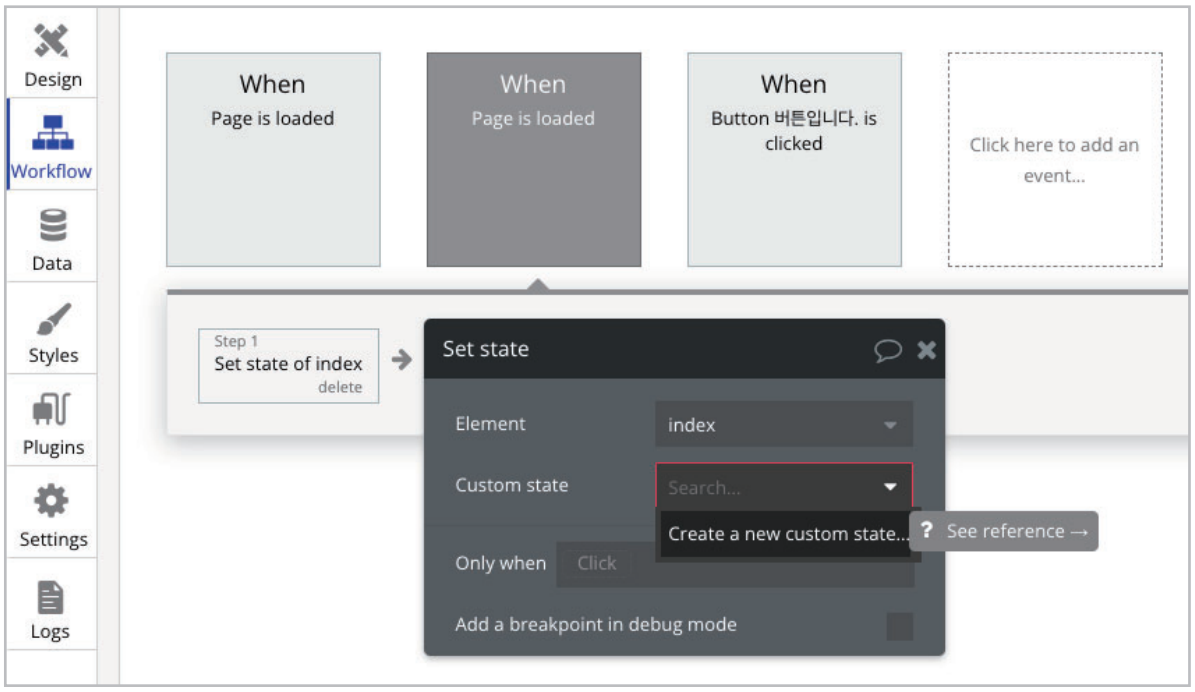
(2) 워크플로우 내에서 임시 값을 저장할 때 사용할 수 있습니다. 만약 어떤 연산을 수행하면서 중간의 결과값을 저장하고 싶다면 커스텀 상태에 저장해 놓았다가 다음 액션에 사용할 수 있습니다.

2. 커스텀 상태 만들기

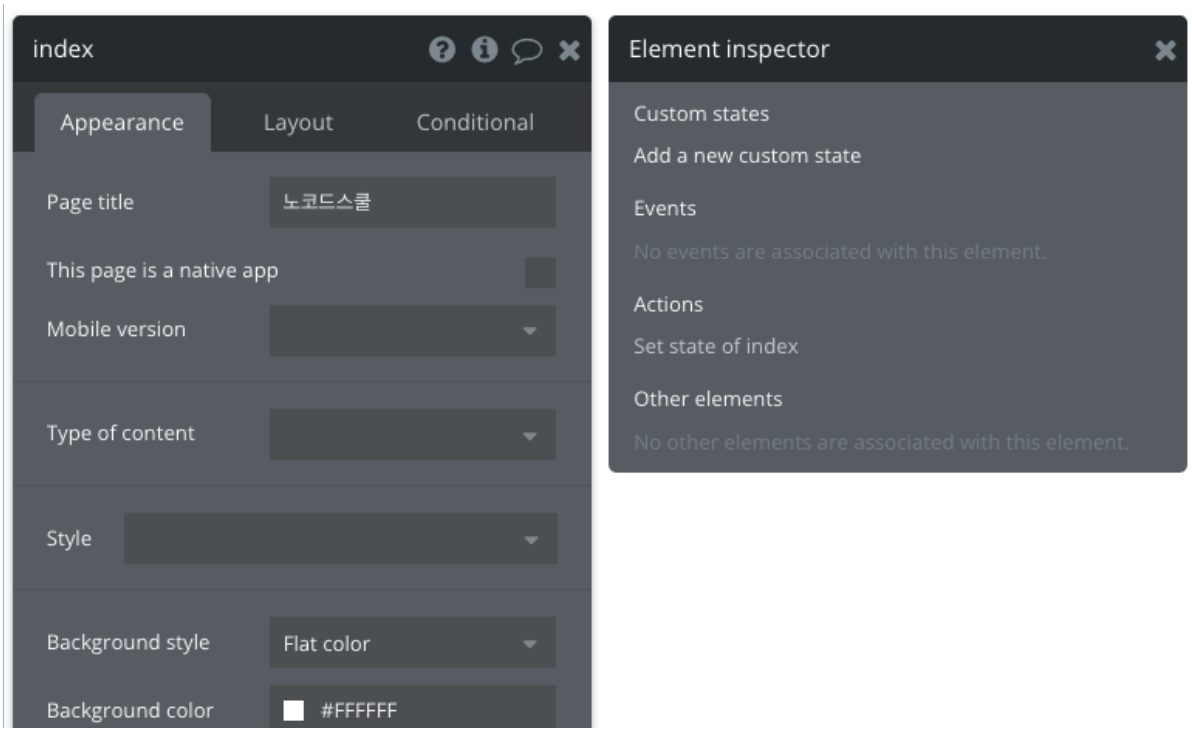
어느 요소나 커스텀 상태를 가질 수 있습니다. 그러니 여러분들은 어떤 요소에 커스텀 상태를 추가해야 적절할 지 판단하면 됩니다. 하나의 요소는 여러 커스텀 상태를 소유할 수 있습니다.

커스텀 상태도 만드는 방법이 두 가지가 있습니다.

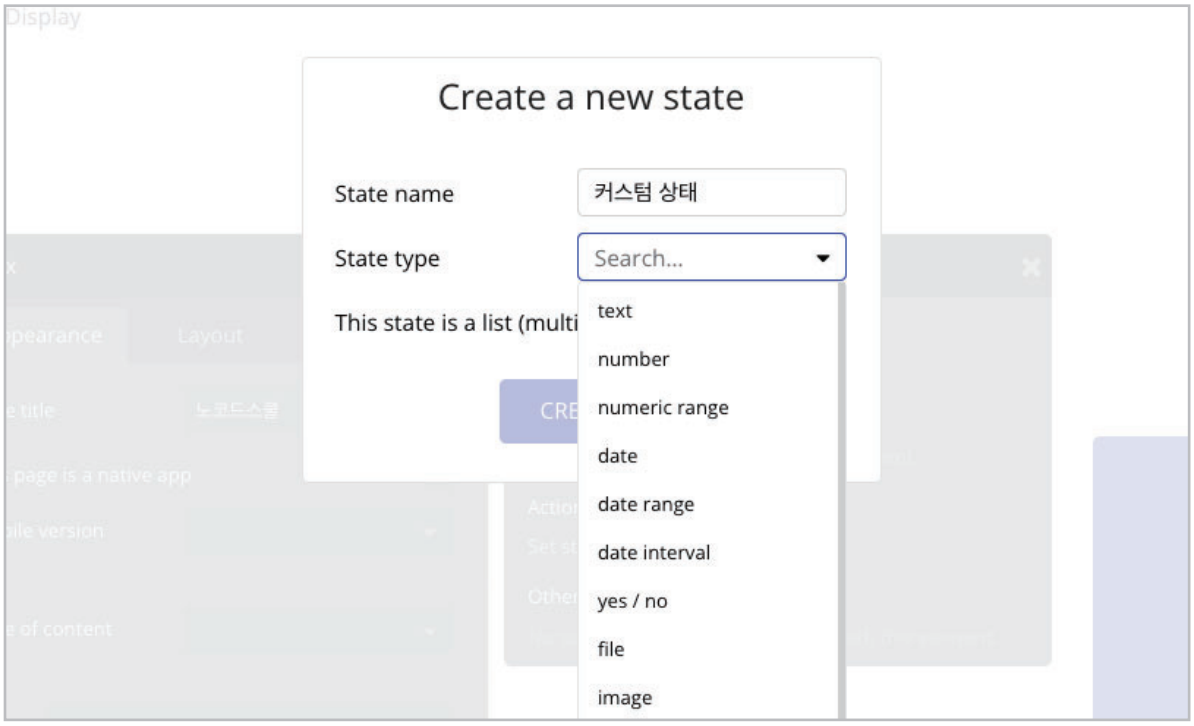
첫 번째 방법은 'Set custom state'를 워크플로우에서 사용하는 방법입니다. 커스텀 상태를 선택하는 옵션의 맨 마지막에 생성할 수 있는 옵션이 있습니다.



두 번째 방법은 요소 우측 상단의 'i' 아이콘을 클릭하면 나오는 Element Inspector에서 만들 수 있습니다.

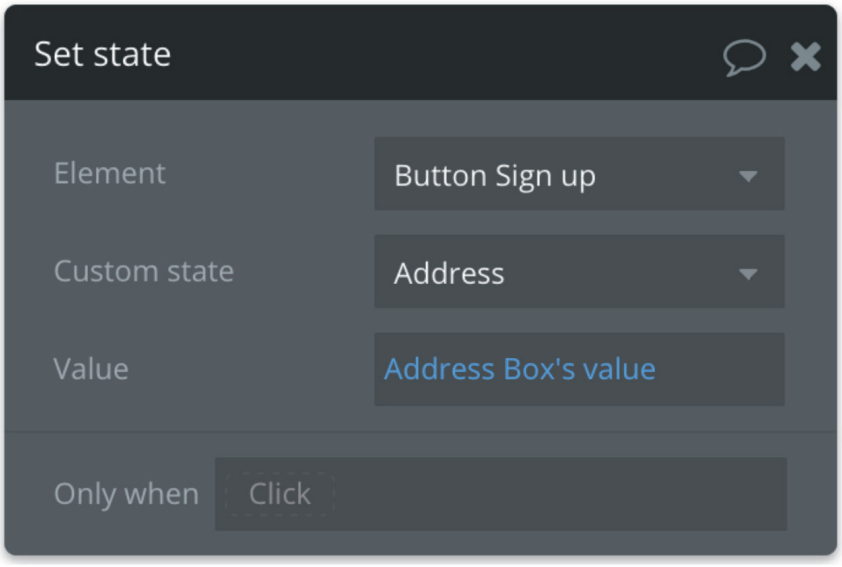


커스텀 상태도 필드처럼 타입을 가지고 있으니 만들 때 이름과 타입을 지정해주어야 합니다.



3. 커스텀 상태 변경하기

커스텀 상태를 변경하는 것은 워크플로우의 'Set custom state' 액션에서 가능합니다. 우선 변경할 커스텀 상태를 소유하고 있는 요소를 선택한 뒤에 적절한 값을 지정해 줍니다. 이때 커스텀 상태의 타입과 값의 타입이 일치하지 않을 경우 에러가 나게 됩니다.

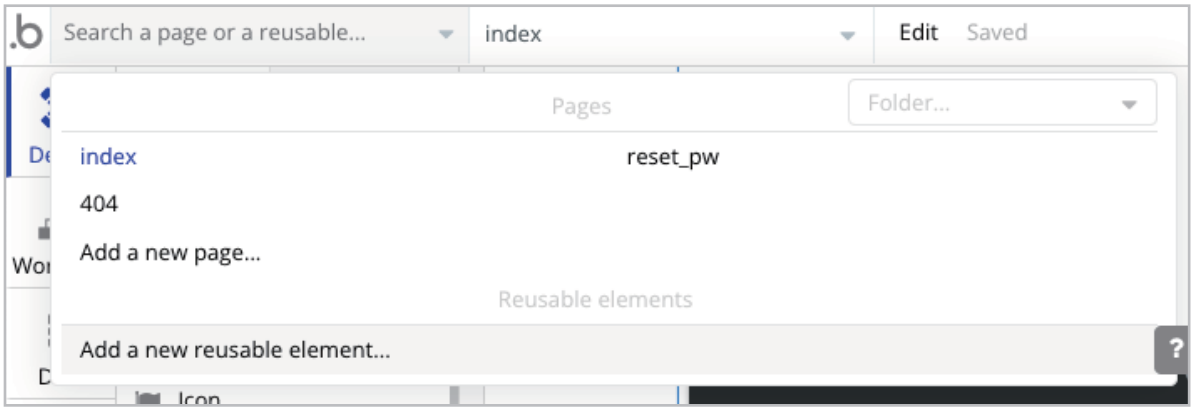


05 CHAPTER 재사용 요소

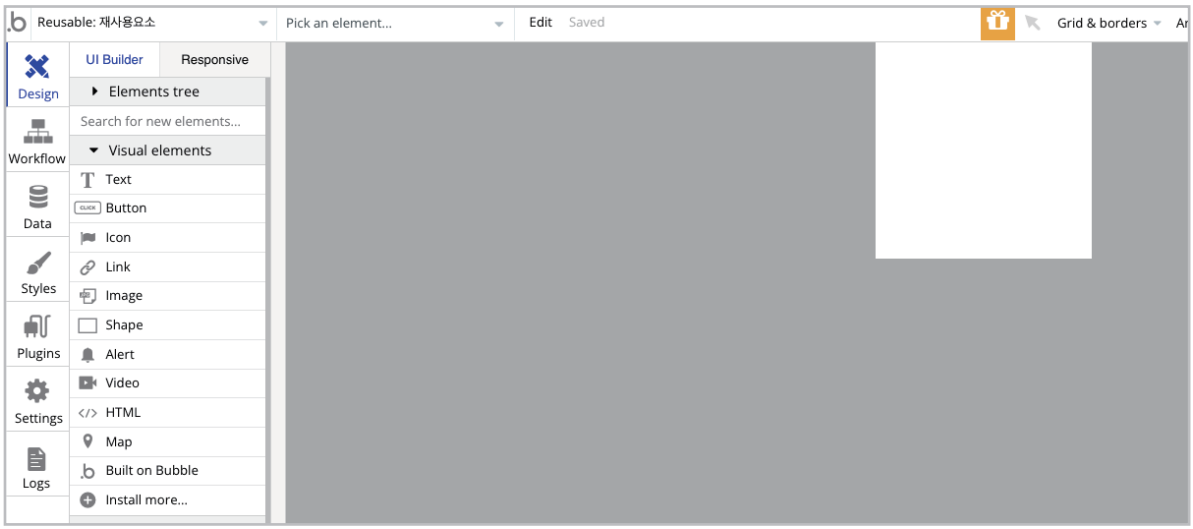
재사용 요소는 하나의 페이지 이상에 사용되는 그룹을 만들 때 사용됩니다. 만약 로그인 팝업을 많은 페이지에서 사용하고 싶다면 한 번 만들어 놓고 재사용하면 됩니다. 이와 유사하게, 푸터와 헤더도 모든 페이지들 마다 사용되니 한 번 만들어 놓은 뒤 재사용하면 됩니다. 재사용 요소를 사용하게 되면 여러분의 서비스가 한 층 가벼워질 뿐만 아니라 유지보수 또한 더욱 쉬워질 것입니다. 재사용 요소를 만들었다면 그 요소는 하나의 요소 타입이 되어 페이지에서 추가할 수 있습니다. 추가로 재사용 요소는 재사용되는 워크플로우를 만들 때도 유용합니다.

1. 재사용 요소 생성하기

재사용 요소를 만들려면 페이지를 선택하는 메뉴를 연 뒤, 'Add a new reusable element'를 선택합니다.



선택하고 나면 재사용 요소를 만들 수 있는 편집 페이지로 이동하게 됩니다. 이때 요소를 디자인하거나 관련된 워크플로우를 정의할 수 있습니다.

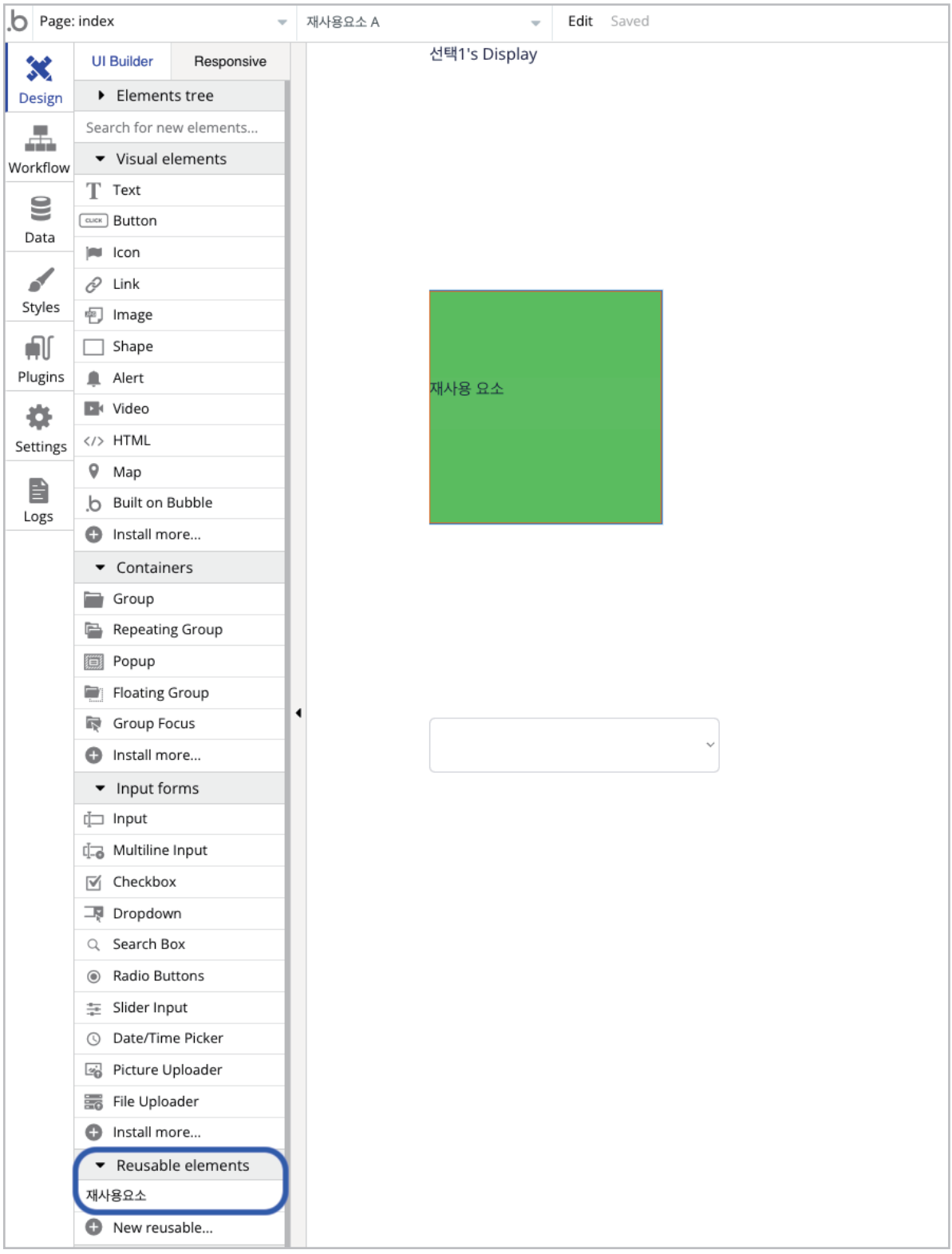


2. 재사용 요소 특징

재사용 요소는 컨테이너 요소이기 때문에 그룹이나 팝업과 같이 작동됩니다. 그렇기 때문에 이 요소의 콘텐츠 타입을 바꿀 수도 있고 데이터를 보낼 수도 있으며 보여주거나 숨길 수 있습니다. 일반 그룹, 팝업과 다른 점이라면 팝업인 재사용 요소가 모달 동작을 한다는 것입니다. 즉, 팝업이 닫히지 않으면 사용자는 팝업 외부의 다른 곳을 클릭할 수 없습니다.

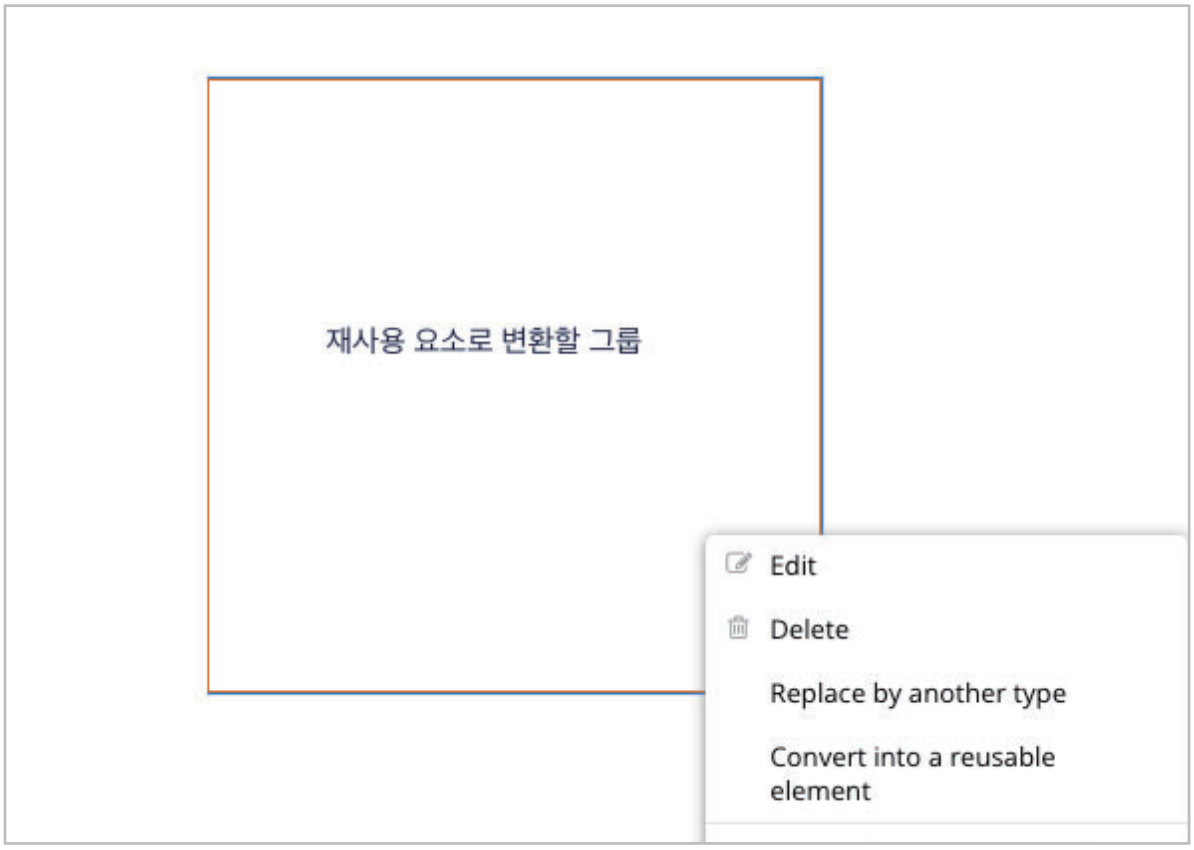
3. 재사용 요소 페이지에 추가하기

재사용 요소를 페이지에 추가하는 방법은 다른 요소들과 동일합니다. 좌측의 요소 목록에서 재사용 요소를 선택한 위 페이지에 추가하면 됩니다.



4. 기존 요소를 재사용 요소로 전환하기

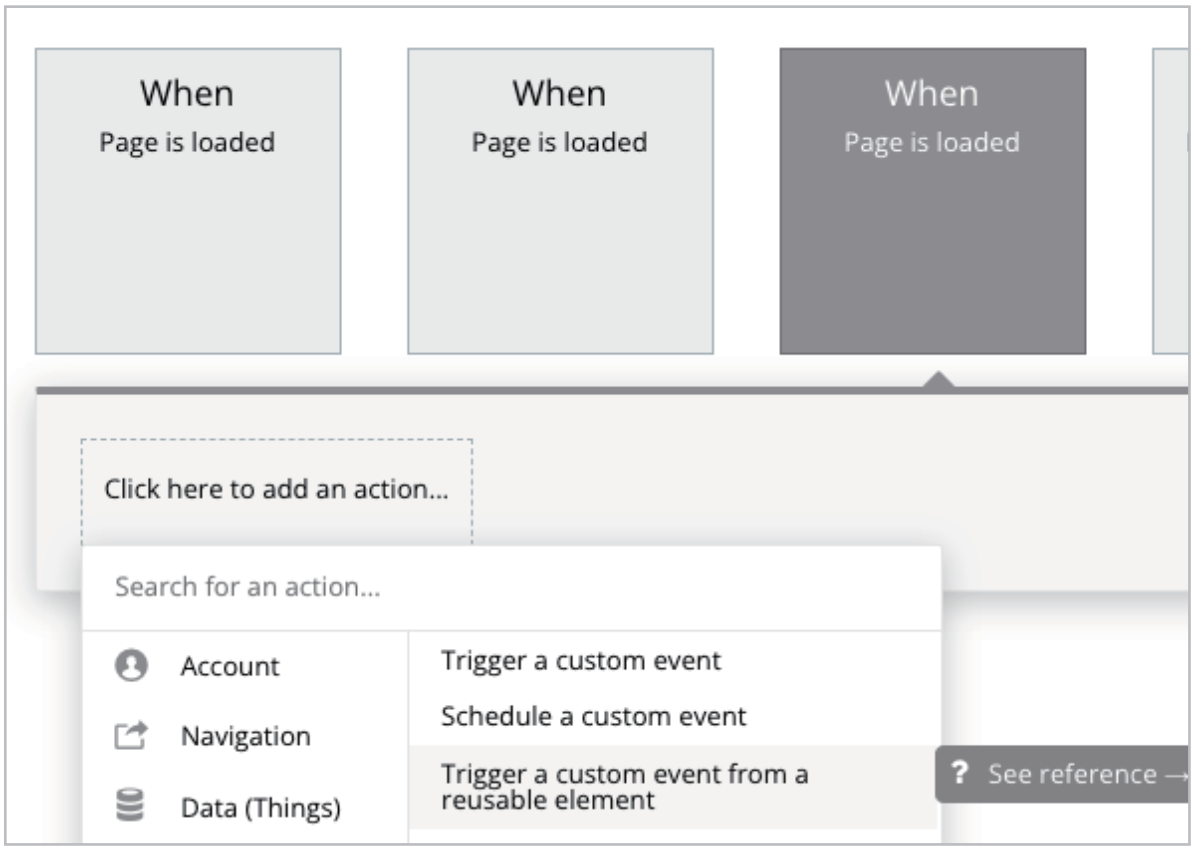
요소 그룹을 재사용 가능한 요소로 변환할 수 있습니다. 두 번 이상 사용할 요소 그룹을 만들었을 경우, 이 옵션을 사용하면 시간을 절약하고 다양한 페이지에서 일관된 설계를 수행할 수 있습니다.



5. 재사용 요소에서의 워크플로우

재사용 가능한 요소의 특별한 사용 사례는 둘 이상의 페이지에서 재사용하려는 워크플로우를 모으는 것입니다. 이렇게 하면 동일한 작업 흐름이 두 곳 이상에서 사용되므로 실수할 가능성이 최소화되고 앱 디버깅 및 수정 속도가 훨씬 빨라집니다.

이렇게 하려면 내부 요소 없이 재사용 가능한 요소를 만들고 페이지 간에 공유할 워크플로우를 추가합니다. 이 재사용 가능한 요소를 페이지에 추가한 후(원하는 경우 보이지 않도록 설정) Trigger custom event from reusable element 액션을 사용하여 재사용 가능한 요소에 추가한 워크플로우에 접근할 수 있습니다.



06 CHAPTER [심화] 페이지 슬러그 관리하기

슬러그는 데이터베이스의 모든 항목(레코드)에 URL 역할을 하도록 할당하는 새로운 빌트인 필드입니다. 슬러그가 지정되기 전까지는 버블 앱의 URL은 항상 고유 ID를 가리킵니다.

1. 슬러그와 URL

슬러그가 없을 경우 레코드가 페이지로 전달되었을 때의 URL은 아래 사진처럼 지정됩니다.

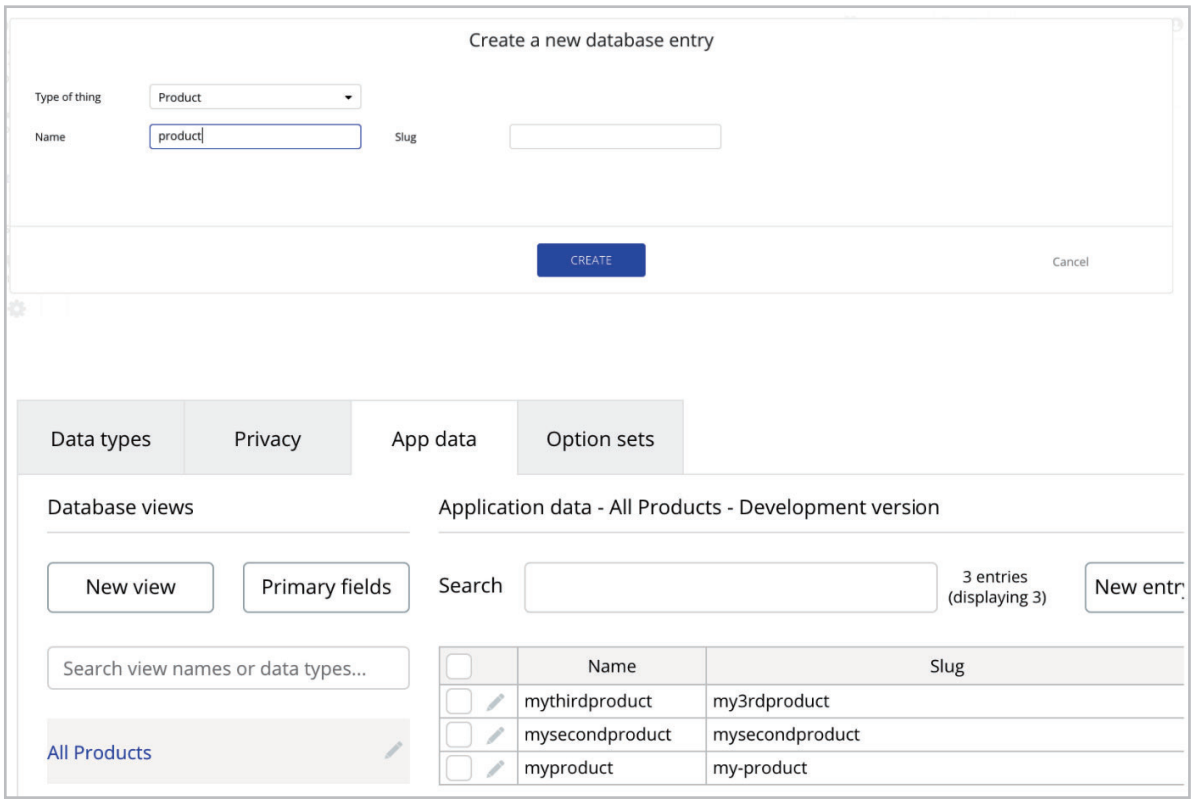
슬러그가 있을 경우 'Scone' 레코드는 아래와 같이 표현됩니다.

슬러그를 사용하면 사용자가 원하는 대로 슬러그를 작성할 수 있기 때문에 URL을 더 짧게 혹은 검색이 용이하게 만들 수 있습니다.

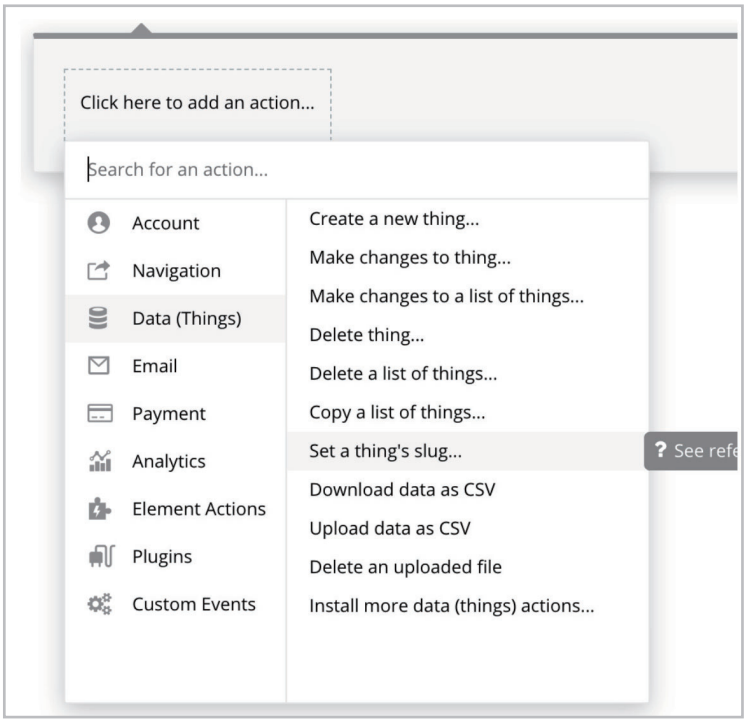
슬러그가 URL에 나타날 때의 순서는 {domain: ~.com}/{page: index}/[slug]입니다.

2. 슬러그 세팅하기

슬러그는 데이터베이스의 레코드를 추가할 때 만들 거나 수정할 때 편집할 수 있습니다.



그리고 워크플로우에서도 편집이 가능합니다.



슬러그는 실시간으로 업데이트할 수 있고, 슬러그를 사용하는 페이지에서 슬러그를 업데이트하면 URL이 자동으로 변경됩니다.

> **주의!**
> – 슬러그는 소문자, 숫자, 하이픈을 포함한 250자 미만으로 작성 가능합니다.
> – 슬러그는 모든 언어를 지원합니다.
> – 워크플로우를 통해 특수 문자가 포함된 슬러그를 설정하면 슬러그에 그대로 저장되어 URL에 표시되고 올바르게 작동합니다.

> **주의!**
> 슬러그는 'Make changes to thing' 액션에서는 바꿀 수 없고 'Set a things slug'로 따로 설정해야 합니다.

3. 슬러그 순서

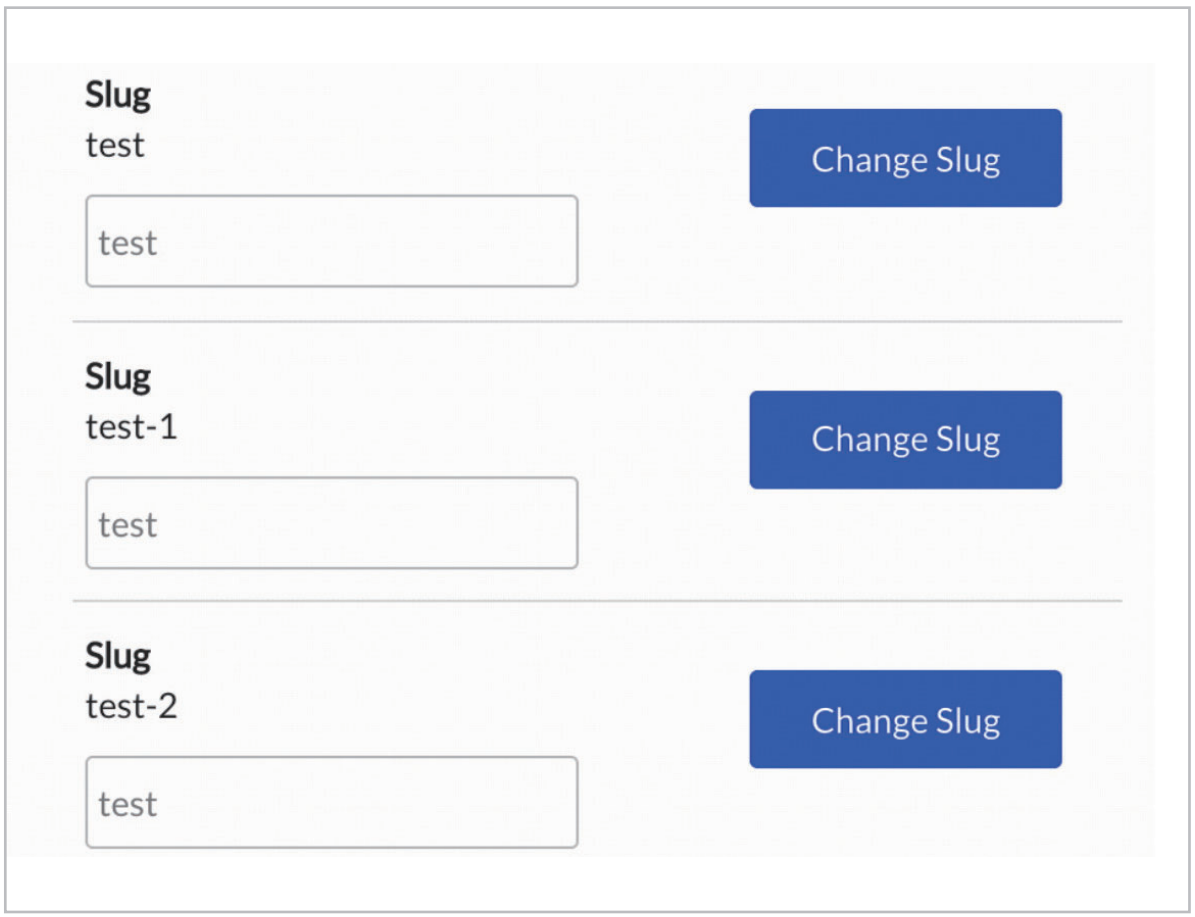
버블이 URL을 정할 때 우선순위가 존재합니다.

1. **슬러그**
2. **URL의 백업 필드(페이지 Inspector에서 설정 가능)**
3. **Unique ID**

만약 슬러그 값이 있다면 버블은 슬러그를 URL에 사용할 것입니다. 슬러그가 없다면 버블은 백업 필드가 있는지 볼 것이고 그 이후에 Unique ID를 마지막으로 URL에 사용합니다.

4. 슬러그 중복

데이터 타입이 다른 경우 동일한 슬러그 값을 공유하도록 할 수 있습니다. 그러나 동일한 슬러그 값을 가진 동일한 데이터 유형은 가질 수 없습니다. 만약 같은 슬러그 값으로 편집을 시도하면 슬러그-1, 슬러그-2, 슬러그-3 와 같이 추가됩니다.

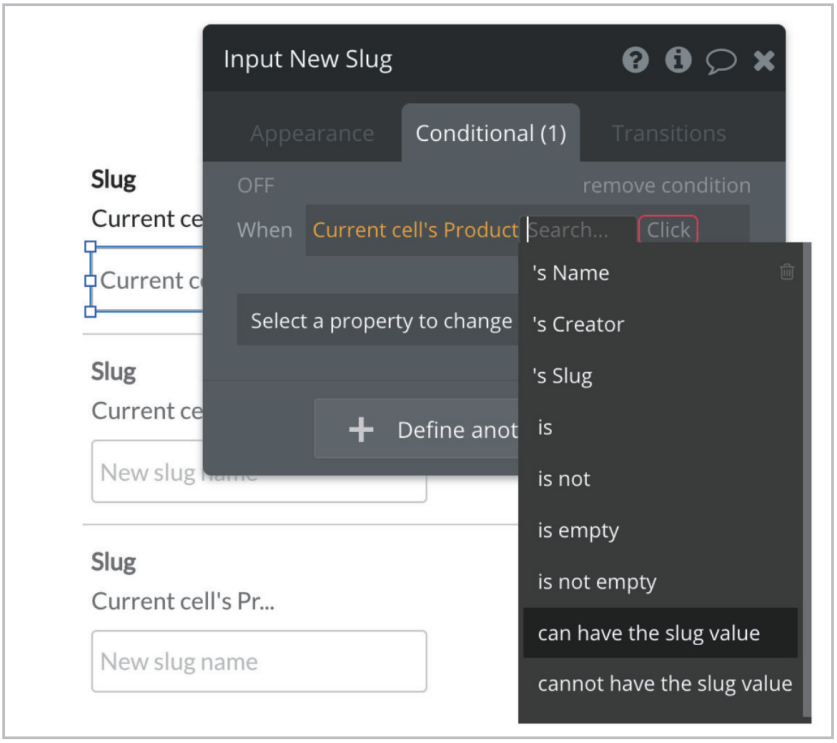


5. 슬러그와 조건부 속성

슬러그를 요소나 워크플로우의 조건으로 활용할 때 두 가지의 경우로 나뉩니다.

Can Have The Slug Value - 값이 해당 항목에 대한 유효한 슬러그 값인지 테스트합니다. 유효한 슬러그 값은 고유하며 소문자, 숫자 및 하이픈으로만 올바르게 형식이 지정됩니다.

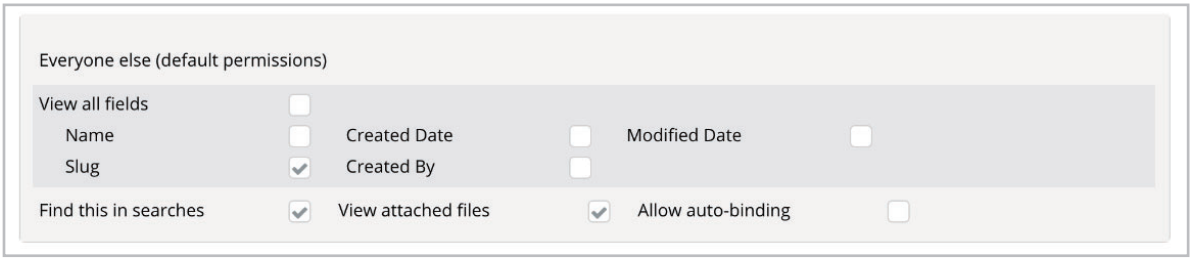
Cannot Have The Slug Value - 값이 Thing에 대한 잘못된 슬러그 값인지 테스트합니다. 잘못된 슬러그 값은 소문자, 숫자 및 하이픈만으로 구성되지 않거나 고유하지 않은 값입니다.



이러한 조건은 슬러그를 사용자 프로필과 같이 이미 존재하는 것과 비교해야 할 때 적합합니다. 이 조건을 사용하여 사용자 이름이 이미 사용되었는지 여부를 확인할 수 있습니다.

6. 슬러그와 프라이버시 규칙

프라이버시 규칙이 있는 데이터의 슬러그를 세팅할 경우에는 'View all fields'에 대한 기본 권한에서 '슬러그' 필드를 선택했는지 확인합니다. 이렇게 하면 모든 사용자가 페이지의 URL에 설정한 슬러그 값을 볼 수 있습니다. 이 옵션을 선택하지 않고 사용자가 슬러그 필드를 볼 수 있는 프라이버시 규칙 조건을 충족하지 않으면 URL에서 URL의 백업 필드나 고유 ID를 대신 보게될 수 있습니다.



주의!

슬러그 필드가 프라이버시 규칙에 의해 숨겨져 있는 경우, 버블 앱에서는 슬러그의 원래 용도를 찾을 수 없기 때문에 중복된 슬러그를 추가할 수 있습니다.

PART 06

버블로 화면 그리기

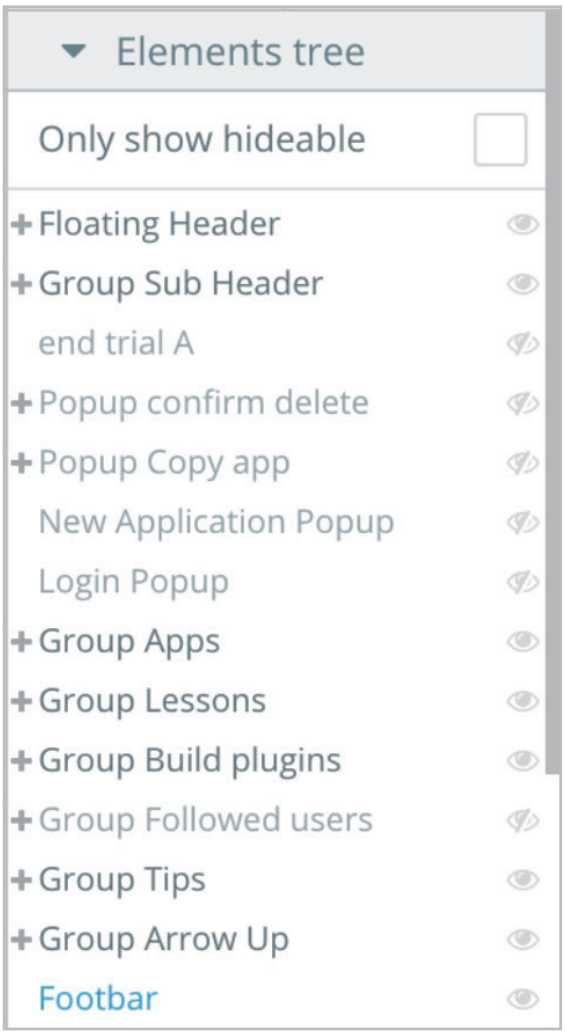
01 CHAPTER 화면 설계 기초

버블 화면 편집기는 What You See Is What You Get(WYSIWYG) 원칙에 기반을 두고 있습니다. 고정(Fixed) 레이아웃에서 원하는 위치에 픽셀까지 요소를 배치할 수 있으며 실행 모드에서는 앱이 에디터와 동일하게 표시됩니다. 화면 크기가 변경될 때 크기가 조정되는 반응형 페이지를 만들려면 컨테이너 레이아웃 유형을 혼합하고 일치시켜서 상상할 수 있는 모든 레이아웃을 만들 수 있습니다. 아래는 인터페이스를 만들 때 염두에 두어야 할 기초이니 먼저 짚고 넘어가도록 하겠습니다.

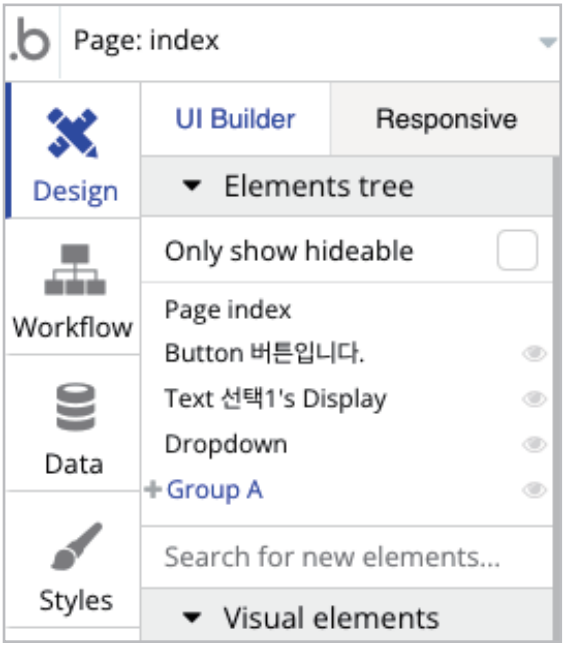
1. 요소 간의 부모-자식 관계

어떤 요소들은 컨테이너 타입일 수 있고 모든 요소들은 부모에 속해 있습니다. 페이지 자체가 최고 상위 항목이며, 페이지의 모든 구성요소는 페이지를 상위 항목으로 가집니다. 컨테이너 요소 안에 요소를 그리려면 컨테이너 위로 마우스를 이동하면 되며 이때 테두리가 빨간색으로 바뀝니다. 요소가 컨테이너 안에 있으면 요소의 동작은 편집 모드와 실행 모드 모두에서 부모의 동작을 따릅니다. 예를 들어, 편집기에서 요소를 이동할 경우 자식은 부모와 비교하여 동일한 위치에 머무르게 됩니다. 실행 모드에서 부모를 숨기면 컨테이너 내부의 모든 요소도 숨겨지고 부모를 표시하면 컨테이너 내부의 모든 요소도 표시됩니다.

왼쪽의 요소 트리를 사용하면 페이지 구조를 볼 수 있으며, 요소를 표시/숨김으로써 편집 및 구성을 개선할 수 있습니다. 기본적으로 많은 요소가 숨겨져 있으며, 눈 아이콘을 클릭하여 보거나 숨길 수 있습니다(필요한 경우 숨겨진 모든 상위 요소도 표시됨). 요소 트리에서 끌어서 놓는 방법으로 요소의 상위 관계를 바꿀 수 있습니다.



요소 트리에는 두 가지 모드가 있습니다.



첫 번째 모드는 페이지의 모든 요소를 트리 보기에서 부모 및 자식과 함께 표시하는 모드입니다. 또 다른 모드는 숨겨진 요소만 표시하는 모드입니다. 숨겨진 요소란, 페이지 로드 시 숨겨진 요소입니다(해당 요소의 속성 편집기에서 'This is element is visible on page load'를 비활성화한 요소). 즉, only show hideable elements 옵션을 선택한 경우에만 목록에 이러한 요소가 표시됩니다. 이 기능을 사용하면 편집할 때 보이지 않는 요소를 빠르게 표시할 수 있습니다. 반면 첫 번째 모드는 페이지의 전체 보기를 가져오는 데 유용합니다.

2. 절대적 위치

버블이 처음 나왔을 때는 요소들의 위치를 좌표(X, Y)를 이용하여 지정했습니다. 이 방법을 사용하면 여러분들은 요소를 원하는 위치에 원하는 대로 놓을 수 있습니다. 이는 많은 자유를 제공하지만 편집기에서의 절대적 위치를 웹 브라우저에서 그대로 사용한다는 뜻이기도 합니다. 만약 스크린의 화면이 편집기보다 작거나 커지게 되면 그에 따라 페이지의 레이아웃이 바뀌지 않기 때문에 결국은 반응형 디자인을 위해 레이아웃 세팅이 필요하게 되었습니다.

이에 따른 반응형 디자인 방법은 다음 장에서 다루겠습니다.

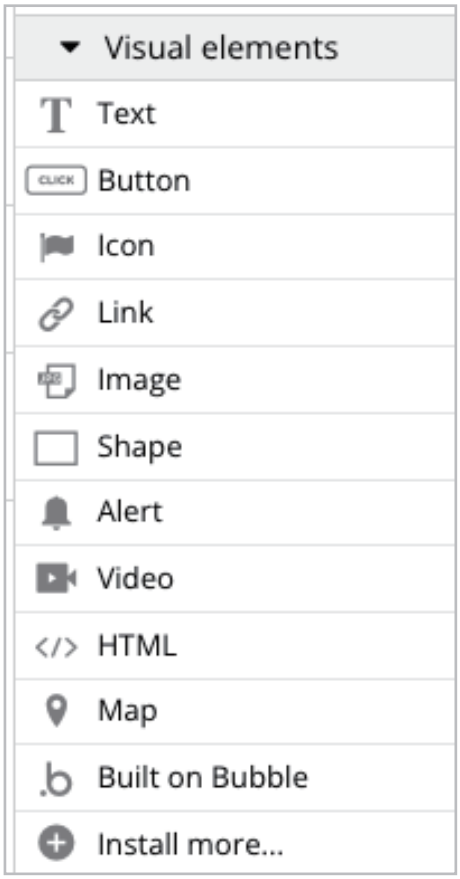
3. 요소 편집

여러분은 에디터 왼쪽의 요소 팔레트에서 원하는 요소를 클릭하여 화면에 가져다 놓으면서 요소를 추가할 수 있습니다. 요소가 페이지에 추가되고 나면 드래그 또는 프로퍼티 에디터에서 이동 혹은 사이즈 조절을 할 수 있습니다. 몇 개의 예외를 제외하고는 대부분의 요소들이 드래그가 가능하고 사이즈 조절도 가능합니다. 팝업이나 모달 컨테이너 요소는 페이지의 맨 위, 가운데에 위치하기 때문에 드래그될 수 없습니다.

프로퍼티 에디터 맨 위의 섹션에서는 요소의 이름을 바꿀 수 있습니다. 현재 이름을 선택한 뒤 변경할 이름으로 바꾸면 됩니다. 이름의 규칙은 개인 혹은 팀마다 다르게 정할 수 있습니다. 예를 들어 버튼은 btn이라고 적는다든지, 첫 글자는 대문자를 쓴다거나 밑줄을 그을 수도 있습니다. 일단 규칙을 한 번 정했다면 그 규칙을 모든 페이지 모든 요소에 적용해야 유지보수가 편하게 됩니다.

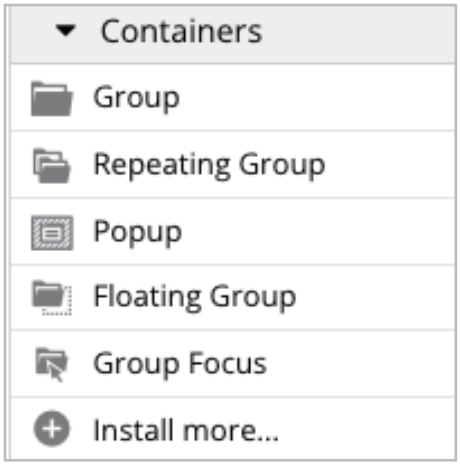
4. 요소 종류

(1) 비주얼 요소



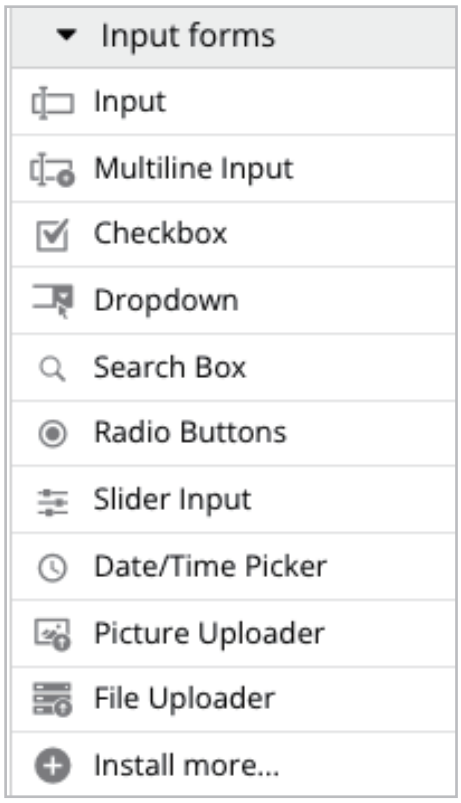
비주얼 요소는 정보를 나타내거나 사용자가 클릭으로 상호작용할 수 있는 요소입니다. 그러나 유저가 정보를 입력하는 것은 불가능합니다.

(2) 컨테이너 요소



컨테이너 요소는 다른 요소들을 포함시키는 요소입니다. 이 요소의 노출 유무는 자식들의 노출 유무에 영향을 미칩니다. 추가로 컨테이너 요소에는 데이터 타입을 지정할 수 있고 레이아웃의 종류도 부여할 수 있습니다.

(3) 입력 요소



입력 요소는 사용자가 정보를 입력할 수 있게 하는 요소입니다. 가장 일반적인 입력 요소는 입력창입니다.

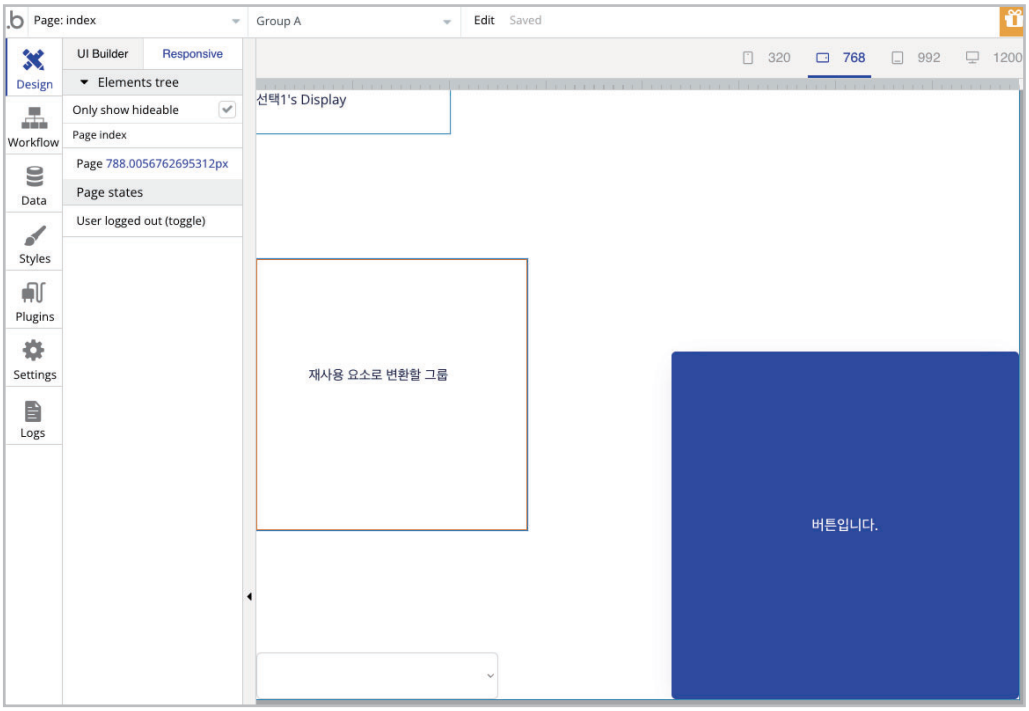
각각의 요소들은 모두 자신만의 필드를 가지고 있어 기능과 외형에 필요한 값들을 편집할 수 있습니다(예를 들어, 입력창의 플레이스 홀더 혹은 색상 등).

02 CHAPTER 반응형 디자인

버블의 페이지들은 반응형도 가능합니다. 컨테이너 레이아웃을 조합하면 페이지 레이아웃을 모바일 장치에서 보기 좋게 화면 너비에 맞게 조정할 수 있습니다. 이때 사용하는 컨테이너 레이아웃은 Row, Column, Align to Parent가 있습니다. 이러한 레이아웃은 자식 요소들의 정렬이나 크기 핸들링이 가능하도록 설계되어 있습니다.

일단은 노트북이나 데스크톱 화면에서 작업을 한 뒤에, 에디터의 사이즈를 조정해가며 반응형을 조절하면 됩니다. 이어 Design탭에 Responsive 뷰어를 통해 페이지의 크기에 따라 어떻게 요소들이 바뀌는지 확인할 수 있습니다.

Responsive 뷰어에서 페이지 영역의 맨 위에 있는 눈금자는 현재 페이지 너비를 정의합니다. 눈금자를 클릭하거나 끌면 페이지 크기가 조정되고 페이지가 동적으로 작동하는 방식을 볼 수 있습니다. 화면 위쪽에 있는 미리 설정된 너비 아이콘을 사용하여 모바일 세로 모드, 모바일 가로 모드, 태블릿, 랩톱 또는 데스크톱에서 페이지가 어떻게 표시되는지 확인할 수도 있습니다.



요소를 클릭하면 특성 편집기의 레이아웃 탭에 해당 동작에 영향을 미치도록 수정할 수 있는 여러 매개 변수가 표시됩니다. 다양한 설정을 수정하면서 페이지에 어떤 영향을 미치는지 실시간으로 확인할 수 있습니다. UI Builder 뷰에서도 동일하게 레이아웃 설정을 조정할 수 있습니다.

1. 반응형 작동방식의 이해

컨테이너 레이아웃은 하위 요소의 동작 및 위치를 정의합니다. 컨테이너 레이아웃 유형은 페이지 자체를 포함하여 모든 컨테이너 요소(그룹, 플로팅 그룹, 반복 그룹 등)에서 사용할 수 있습니다. 화면 크기 변경에 대응하려면 페이지가 반응형 컨테이너 유형(즉, "Fixed" 아님)이어야 합니다. 각 컨테이너 레이아웃 유형에는 해당 레이아웃 유형별로 고유한 콘트롤 세트가 있습니다. 또한 하위 요소는 상위 컨테이너의 선택된 레이아웃 유형에 따라 고유한 콘트롤을 상속합니다.

아래에 컨테이너 요소가 가질 수 있는 레이아웃 유형을 차례로 소개하겠습니다.

(1) Fixed

고정 레이아웃 유형은 고정 너비 및 높이 컨테이너를 하위 요소의 절대 위치와 함께 정의합니다. 하위 요소는 사용자가 익숙한 대로 드래그 앤 드롭하여 위치를 지정하고 크기를 조정합니다. 이렇게 고정된 컨테이너는 화면 크기나 콘텐츠 크기의 변경에 응답하지 않습니다.

- 부모 레이아웃 Controls
  - 없음
- 자식 레이아웃 Controls
  - 너비 및 높이

(2) Align to Parent

하위 요소는 상위 컨테이너의 3×3 배열에 정렬됩니다. 내부에 그려지거나 상위에 정렬로 끌어넣은 새 하위 요소는 가장 가까운 부분에 위치됩니다. 하위 요소의 위치와 크기는 속성 편집기에서 제어되지만 요소를 끌어서 놓기를 통해 위치를 변경할 수 있습니다. 상위 컨테이너의 크기가 조정됨에 따라 하위 요소는 각각의 배열에 정렬된 상태를 유지하며 서로 겹칠 수 있습니다.

- 부모 레이아웃 Controls
  - 없음
- 자식 레이아웃 Controls
  - 선택된 부분
  - 너비 및 높이

(3) Row

행 컨테이너의 하위 요소는 수평으로 정렬됩니다. 컨테이너 내부에 그려진 새 하위 요소는 기본적으로 목록 끝에 추가되지만 순서 제어를 사용하거나 끌어서 놓기를 통해 다시 정렬할 수 있습니다. 하위 요소의 위치 및 크기는 속성 편집기에서 제어됩니다.

- 부모 레이아웃 Controls
  - 컨테이너 정렬
  - Row 갭
  - Column 갭
- 자식 레이아웃 Controls
  - 수직 정렬
  - 순서 선택
  - 너비 및 높이

(4) Column

열 컨테이너의 하위 요소는 수직으로 정렬됩니다. 하위 요소는 화면 또는 콘텐츠 크기가 조정될 때 다른 요소를 확장하거나 아래로 밀어 넣습니다. 컨테이너 내부에 그려진 새 하위 요소는 목록 끝에 추가되지만 순서 제어를 사용하거나 끌어서 놓기를 통해 다시 정렬할 수 있습니다. 하위 요소의 위치 및 크기는 속성 편집기에서 제어됩니다.

- 부모 레이아웃 Controls
  - Row 갭
- 자식 레이아웃 Controls
  - 수평 정렬
  - 순서 선택
  - 너비 및 높이

주의!

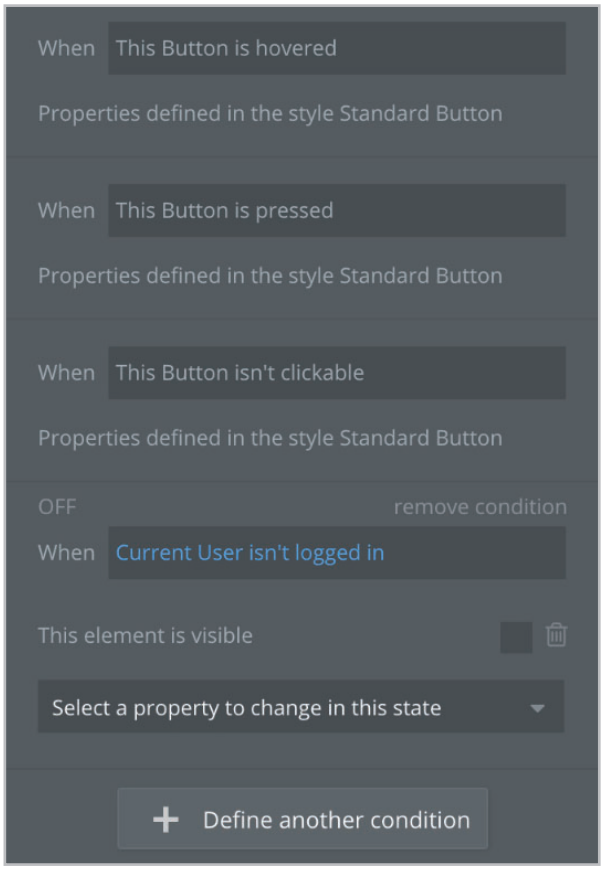
페이지 너비보다 넓은 요소는 반응형 엔진에서 예기치 않은 동작을 발생시킬 수 있습니다. 모든 요소를 페이지 너비보다 넓히지 않도록 주의하세요.

03 CHAPTER 조건부 핸들링

특정 상황에서 요소들이 다르게 작동하거나 보여지게 할 수 있는데 이 방법을 조건부 핸들링이라고 합니다. 예를 들어, '로그아웃' 버튼을 로그인이 되었을 때만 보이게 하거나, 버튼에 마우스를 올리면 색상이 바뀌게 하는 것을 말합니다. 이러한 기능은 프로퍼티 에디터의 세 번째 탭인 Conditional 탭에서 정의할 수 있습니다.

1. 조건부 정의

프로퍼티 에디터의 세 번째 Conditional 탭에서 조건을 정의할 수 있습니다. 그리고 이러한 조건은 두 가지로 구성되어 있습니다.

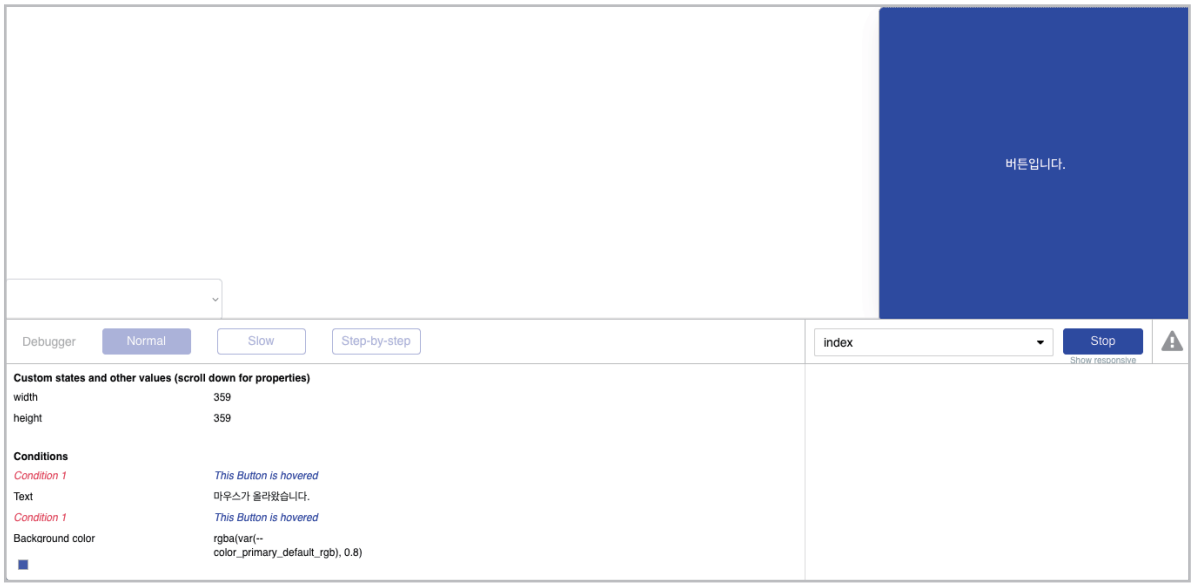


1. 언제 해당 조건이 실행될 지에 대한 경우
2. 1번 조건일 때 변경될 프로퍼티와 그 값

조건은 어떠한 데이터로도 구성할 수는 있지만 결과가 '예/아니요'로 나와야 합니다. 그렇지 않으면 에러가 발생합니다. 또한 여러 조건이 한 번에 만족하는 경우가 있을 수 있는데 이러한 경우에는 가장 나중에 만들어진 조건으로 적용됩니다. 그러므로 우선 순위가 필요할 때는 move up / down을 활용하면 됩니다.

각 조건의 왼쪽 상단에 ON / OFF 로 해당 조건이 적용 될지 안 될지 설정할 수도 있습니다.

조건이 적용되었는지는 Run-mode에서 디버깅으로 확인할 수 있으니, 조건이 중첩되거나 예상과 다르게 동작할 경우에는 디버거를 통해 확인해 봐야 합니다.

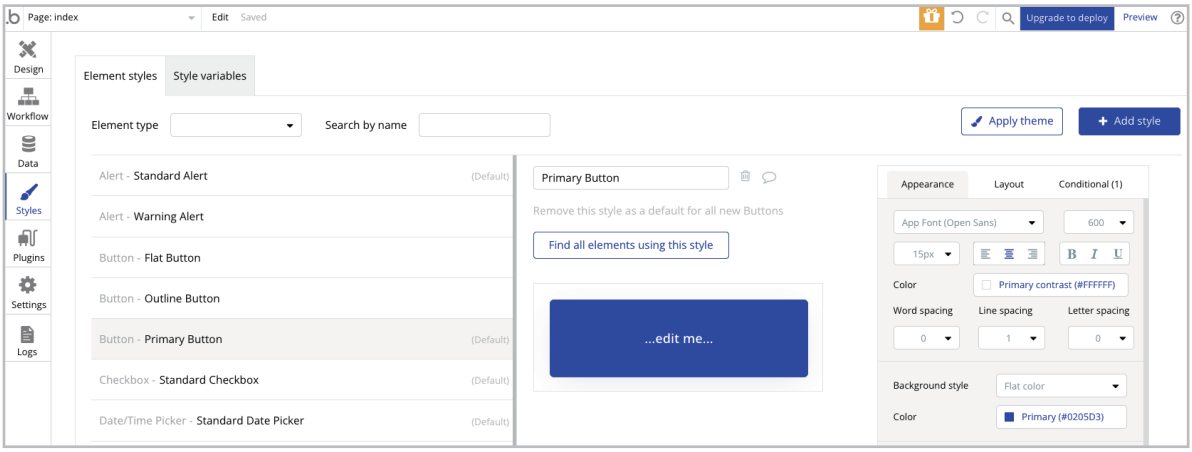


04 CHAPTER 스타일 적용

버블에서는 페이지에 쓰이는 요소들에 공통의 스타일을 적용할 수 있는 시스템을 보유하고 있습니다. 이 스타일을 잘 사용하면 디자인을 한 층 통일감 있게 작업할 수 있습니다. 하나의 스타일만 변경시키면 해당 스타일을 사용하는 요소들 모두에 적용되기 때문입니다. 또한 스타일을 사용하면 버블 앱의 공간이 절약되고 스타일 데이터가 유사한 스타일을 사용하는 요소끼리 공유되므로 로딩 속도도 빨라집니다.

1. 스타일 정의하기

스타일은 스타일 탭에서 생성할 수 있습니다. 스타일을 편집하는 것은 프로퍼티 에디터에서 요소를 편집하는 것과 유사합니다.

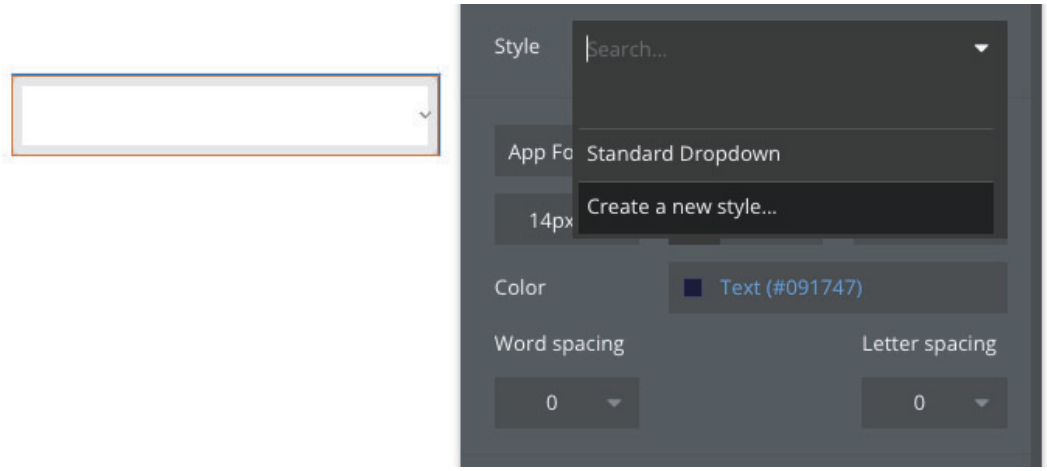


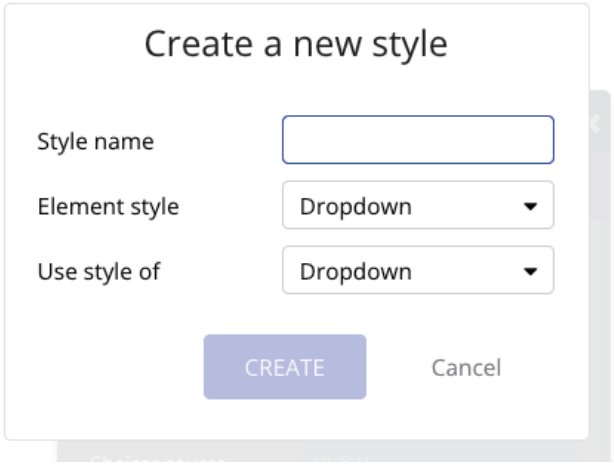
스타일에서 조건부 핸들링도 적용할 수 있지만 기본적인 상태만 조건에 포함할 수 있습니다. 데이터를 사용하는 조건이나 유저의 프로퍼티에 의한 조건들은 스타일 탭 내의 조건부 핸들링에서는 적용할 수 없습니다.

스타일은 원하는 한 많이 생성이 가능합니다. 버튼도 여러 스타일의 버튼이 있을 수 있기 때문에 이름을 구별 가능하게 만들어 놓는 것을 추천합니다. 그리고 스타일의 이름을 적는 란 아래에는 해당 요소의 기본 스타일로 지정할 수 있는 설정 기능도 있습니다. 버튼 요소를 만든다면 기본적으로 지정될 스타일을 이 속성으로 기본 지정해 놓으면 됩니다.

스타일을 생성하는 방법은 스타일 탭에서 'New Style'을 클릭하여 만들 수 있습니다.

또 다른 방법으로는 요소 자체에서 현재의 스타일을 공통 스타일로 생성할 수도 있습니다.





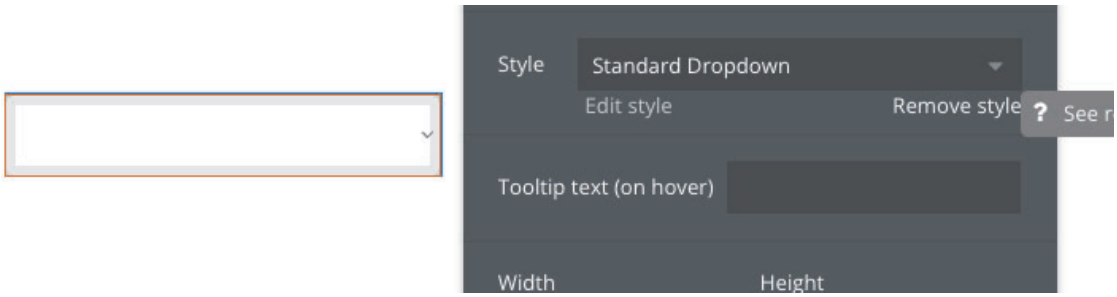
이 경우는 비슷한 스타일은 다른 요소에서도 자주 사용하는 데 아직 공통 스타일로 지정이 안 되어 있을 경우에 유용하게 쓰입니다.

주의!

스타일을 만들 때 특정되어 있는 라벨(H1, headers)은 사용하지 마세요.

2. 스타일 적용 & 삭제

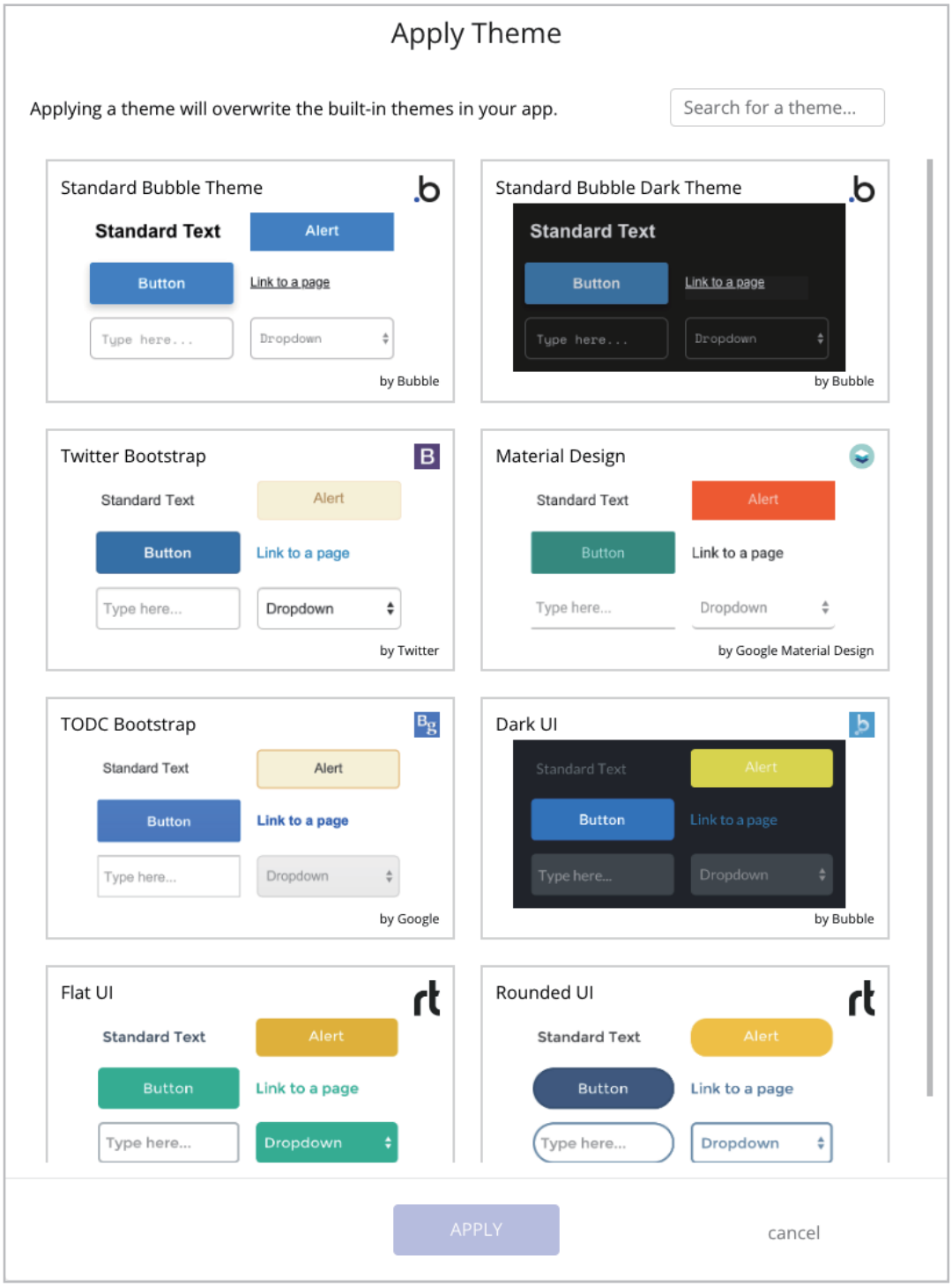
요소에 스타일을 적용하면 해당 스타일에 포함되어 있는 현재 프로퍼티들은 삭제됩니다. 반면에 요소에 있던 스타일을 삭제하면 스타일에 있던 프로퍼티들이 요소에 복사됩니다. 그러므로 버튼의 스타일을 지우더라도 기존 스타일에서 가지고 있던 값들은 유지되므로 변경되지 않은 상태에서 요소의 외형을 수정할 수 있습니다.



3. 조건부 충돌

공통 스타일에 있던 조건과 해당 요소에 있는 조건이 충돌하게 되면 우선 순위는 해당 요소에 있는 조건으로 적용되게 됩니다.

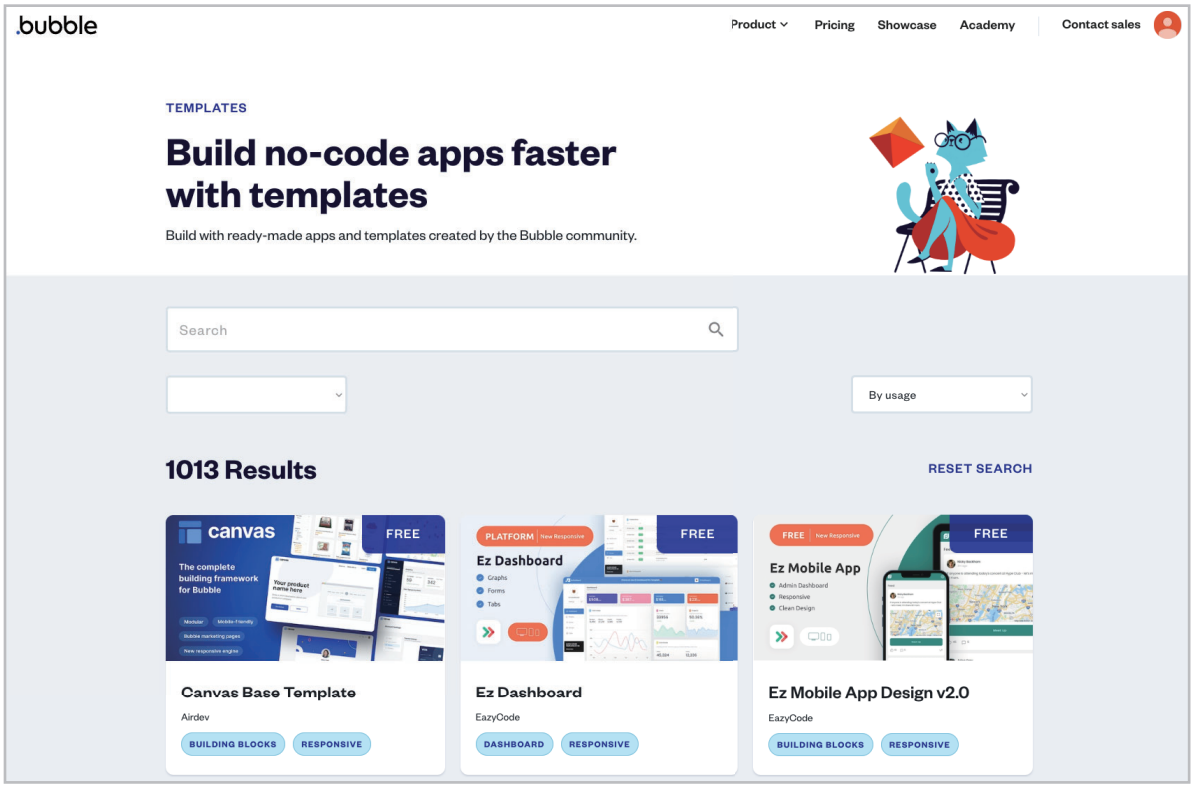
4. 테마 사용



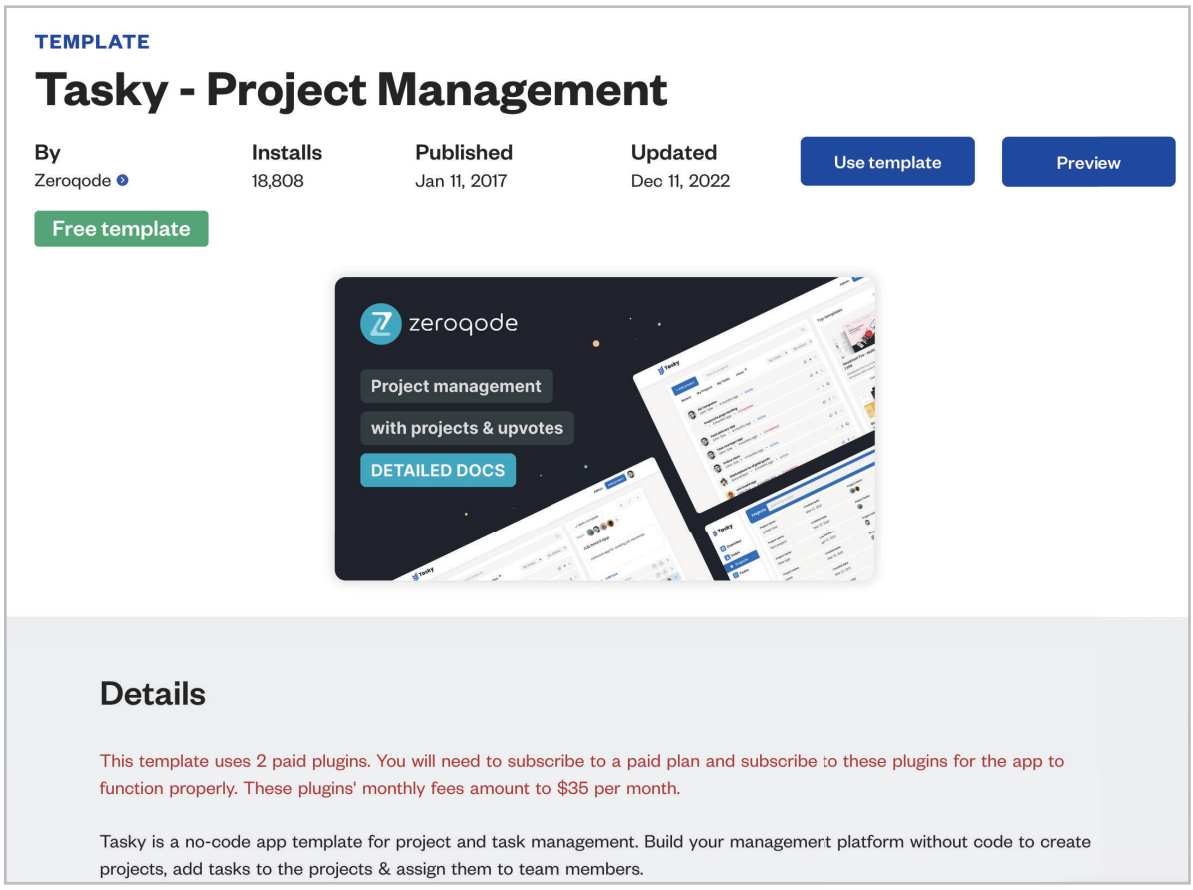
버블은 서비스에 다양한 테마를 제공하고 있습니다. 테마를 적용하면 기존 스타일에 덮어씌우게 됩니다. 버블은 이미 부트스트랩과 머티리얼 디자인과 같은 다양한 인기 있는 디자인 라이브러리에서 영감을 받은 테마를 만들어 놓았습니다. 참고로 테마를 적용할 때 기존에 추가된 스타일은 수정되지 않습니다.

05 CHAPTER 템플릿 활용

버블로 서비스를 만들 경우에 빈 페이지에서 시작할 수도 있지만 템플릿 페이지를 활용할 수도 있습니다. 템플릿을 사용할 경우 버블 앱을 생성할 때 적용할 템플릿을 선택하면 됩니다.

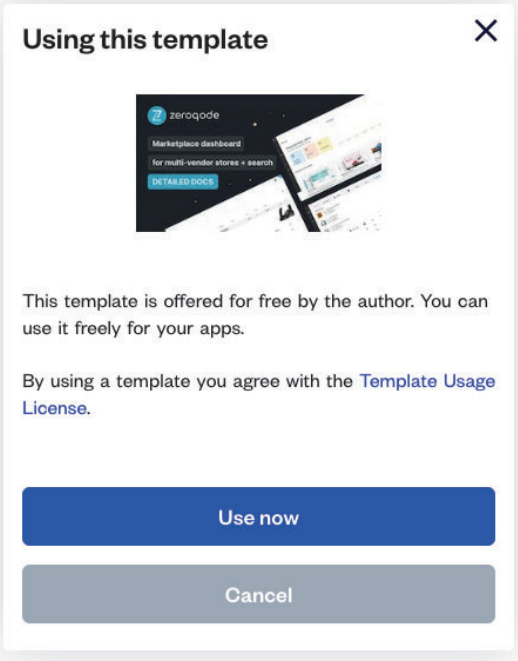


템플릿은 우측 상단의 Product 메뉴에서 Market place에서 찾을 수 있습니다.

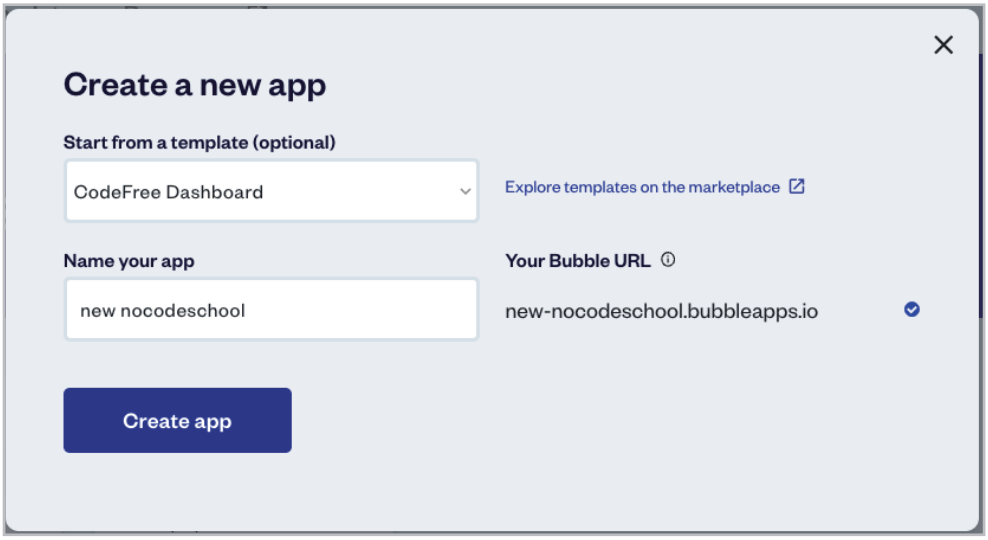


원하는 템플릿을 선택하면 상세 정보를 볼 수 있고 템플릿을 적용한 버전의 미리보기를 볼 수 있습니다.

템플릿을 구매 혹은 사용을 원한다면 'Use template'를 클릭하여 이용하면 됩니다.



또는 앱을 새로 만들 때 템플릿을 직접 선택하는 방법도 있습니다.



주의!

앱을 생성한 후에는 템플릿을 추가할 수 없습니다.

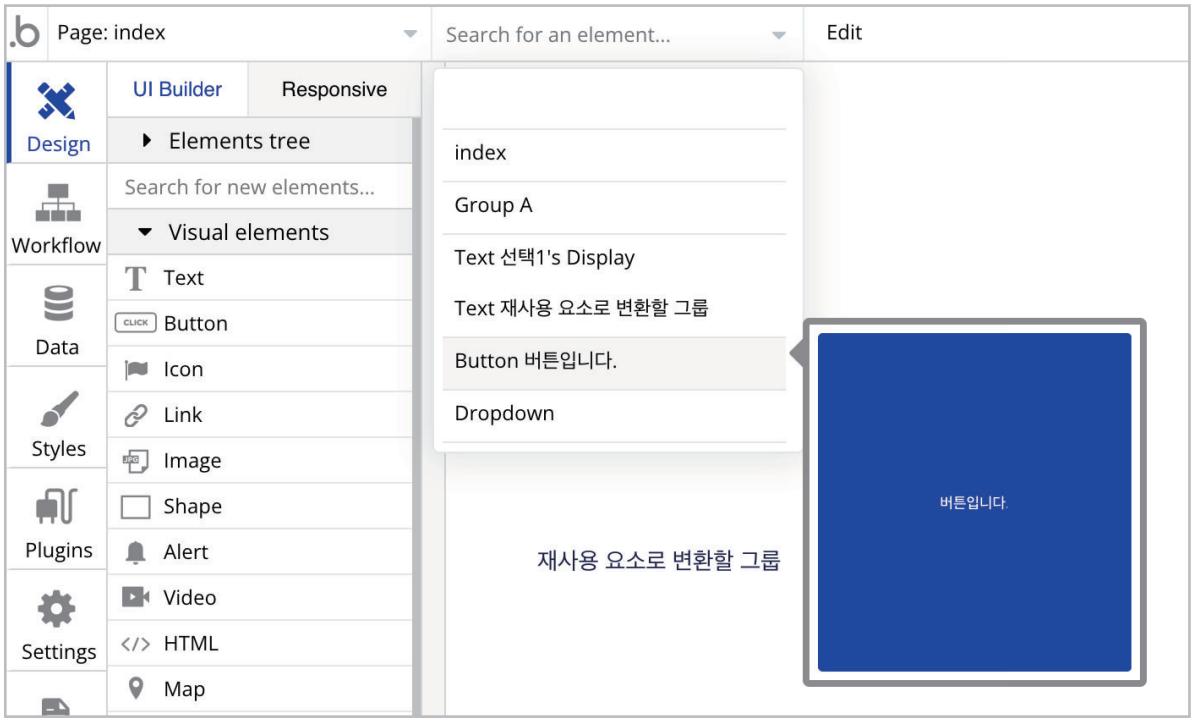
06 CHAPTER [Tips] 버블 디자인 팁

이번 챕터의 마무리로 버블에서 제안하는 디자인 팁을 알아보도록 하겠습니다.

1. 손쉽게 요소 찾기

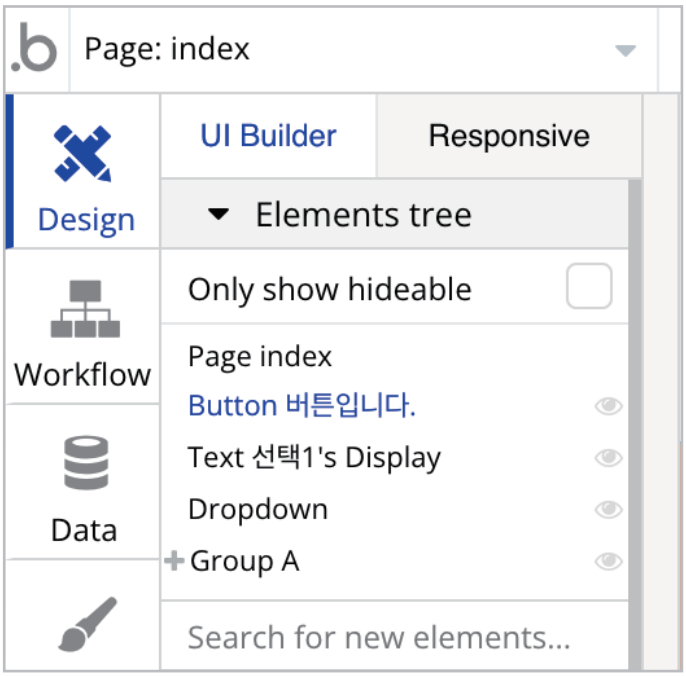
페이지에 많은 요소가 있으면 편집할 요소를 찾기 어려울 수 있습니다. 이런 경우 유용한 몇 가지 방법을 소개합니다.

- 상단 바에 있는 요소 선택기를 사용합니다. 모든 페이지 요소들은 알파벳순으로 정렬되어 있습니다. 여러분들이 찾는 요소와 맞는지 확인하기 위해서는 해당 요소에 마우스를 가져가면 썸네일로 보여지게 됩니다. 이 방법을 유용하게 사용하려면 이름을 규칙적으로 짓는 것이 습관화 되어있어야 합니다.

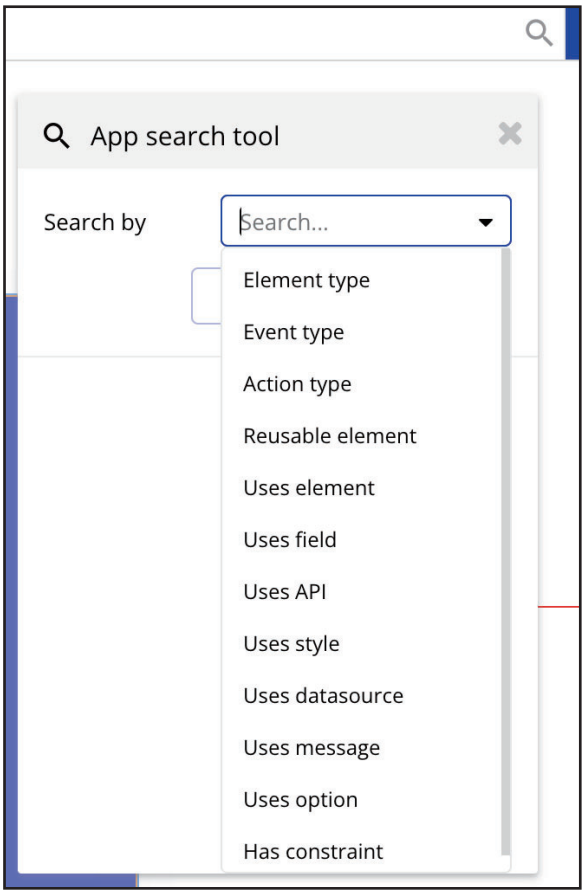


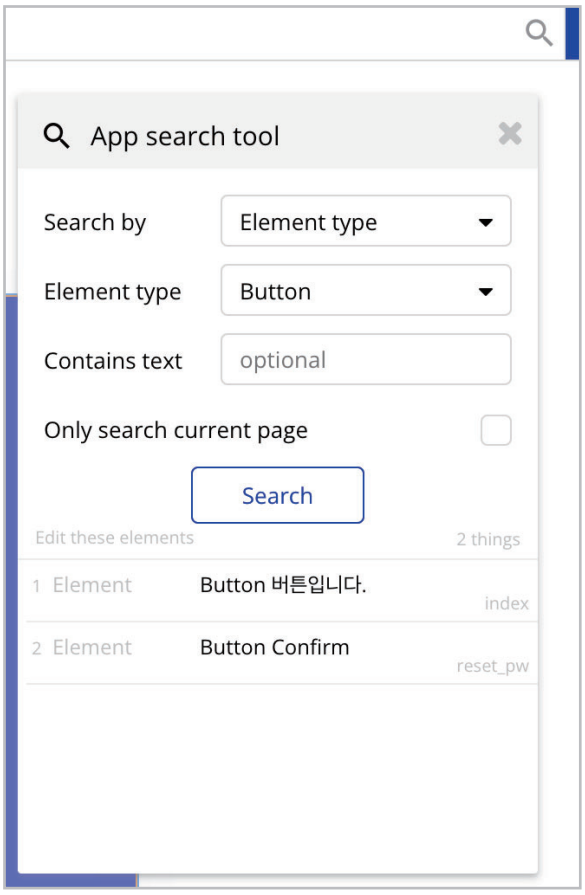
- 두 개의 요소가 겹쳐져 있을 때 cmd키를 누른 채로 클릭하게 되면 각각의 요소를 하나씩 선택하게 됩니다. 이것을 사용하게 되면 페이지 레이아웃을 변형하지 않고도 요소의 아래 부분들까지 선택할 수 있습니다.
- 'X-Ray'아이콘을 클릭하면 요소를 반투명 상태로 만들어 주어 위에 언급했던 cmd+클릭을 보다 더 편리하게 적용할 수 있습니다.

- 페이지의 구조를 정갈하게 보고 싶다면 좌측의 요소 트리를 사용하세요.

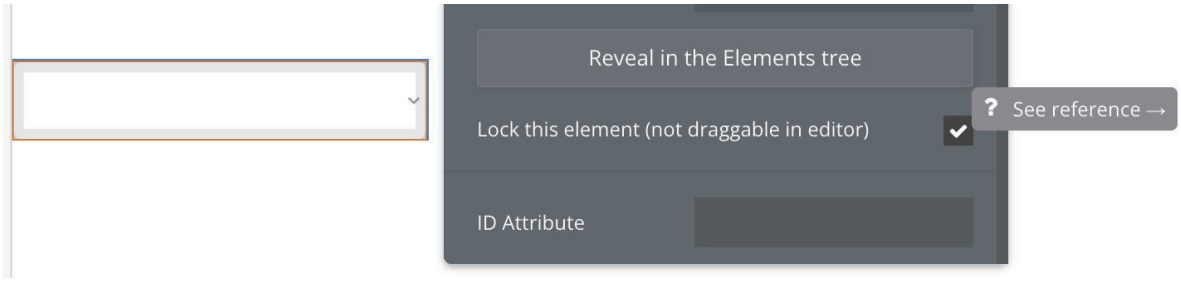


- 앱 서치 툴은 현재 페이지에서 특정 타입의 요소를 찾거나 특정 문구를 포함한 요소를 찾는 데 유용합니다.

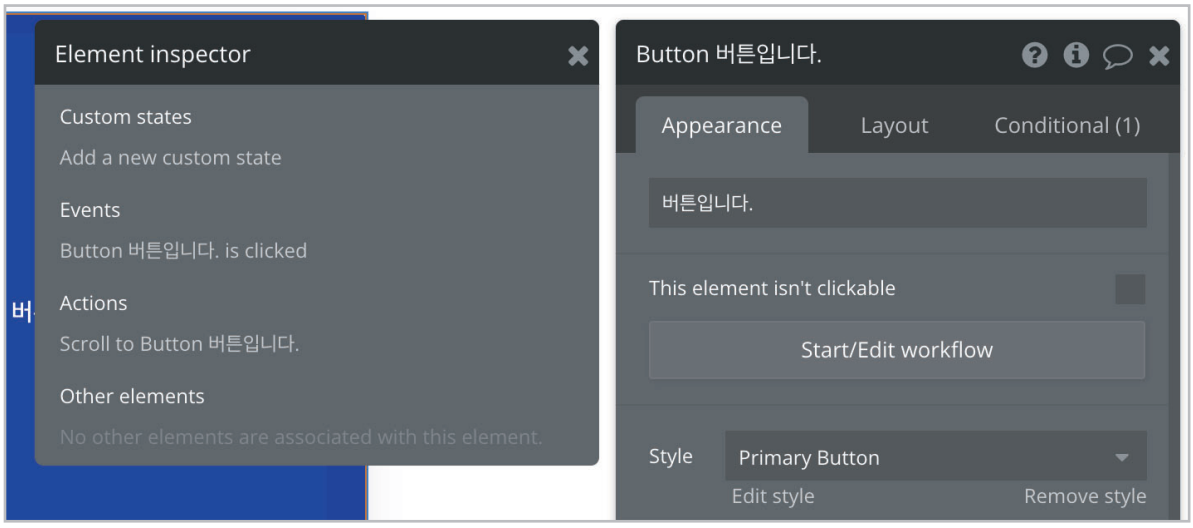

App search tool
Search by
Search...
Element type
Event type
Action type
Reusable element
Uses element
Uses field
Uses API
Uses style
Uses datasource
Uses message
Uses option
Has constraint


App search tool
Search by
Element type
Element type
Button
Contains text
optional
Only search current page
Search
Edit these elements
2 things
1 Element
Button 버튼입니다.
index
2 Element
Button Confirm
reset_pw

- 만약 움직이면 안 되는 요소들을 계속 이동시킬 때, 'Lock this element'를 활성화하여 이동을 방지할 수 있습니다. 이는 순수하게 편집 목적으로 사용되며 복잡한 페이지를 작업할 때 유용합니다.



2. Inspector 활용하기



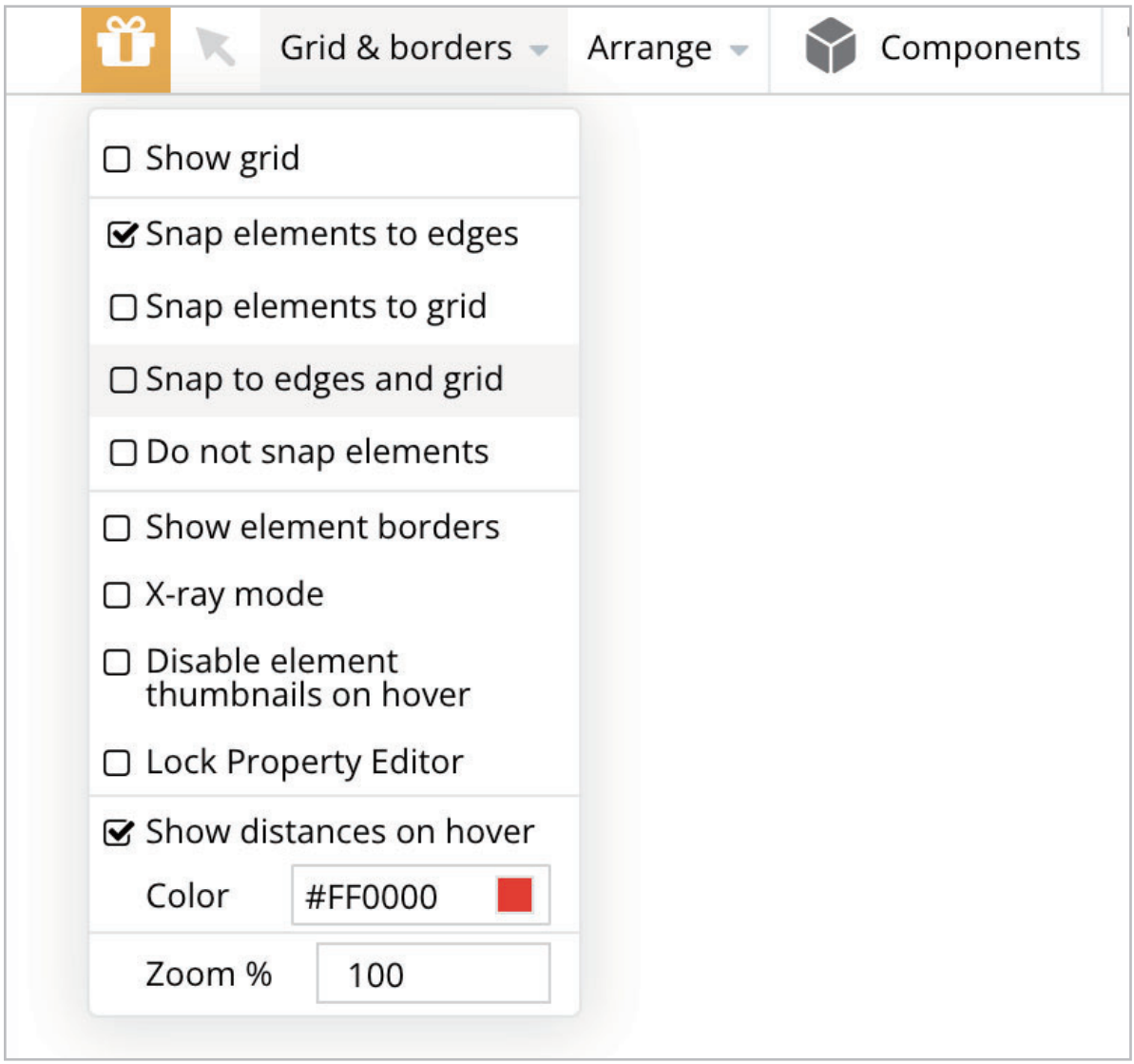
복잡한 페이지를 제작하다 보면 하나의 요소가 이벤트, 액션, 다른 요소 등 여러 곳에서 사용되게 됩니다. 이 때 Inspector에서 하나의 요소가 어느 곳에 쓰여 있는지 모아줄 수 있습니다. 이 뷰는 요소의 'i'아이콘을 클릭하면 볼 수 있습니다. 아래는 Inspector에서 보여주는 정보입니다.

- 현재 요소가 가지고 있는 커스텀 상태를 보여주고 수정, 삭제, 추가가 가능합니다.
- 현재 요소가 사용되는 이벤트들을 보여줍니다.
- 현재 요소가 사용되는 액션들을 보여줍니다.
- 현재 요소가 다른 페이지에서 참고된 것을 보여줍니다.

3. 정렬 선택하기

요소들의 정렬은 보기 좋고 깔끔한 디자인의 키 포인트입니다. 버블은 이를 위해 몇 개의 툴을 지원합니다.

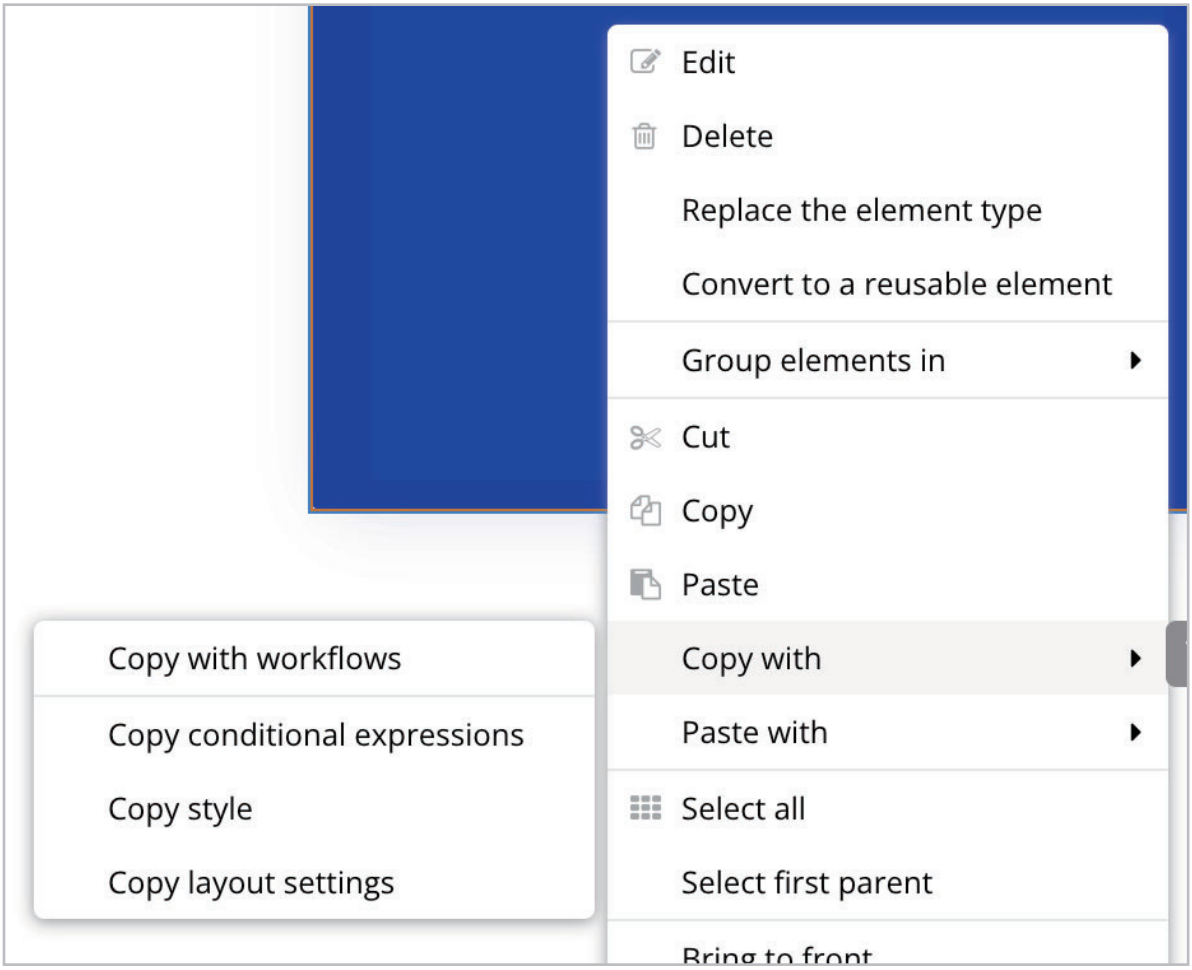
1. 드래그나 사이즈를 재조정할 때, 요소가 모서리로 스냅됩니다. 버블은 기본적으로 페이지에 있는 선에 요소들은 스냅해주려고 노력합니다. 만약 이러한 기능을 그리드로 확장하고 싶다면 상단 바의 메뉴에서 설정할 수 있습니다.



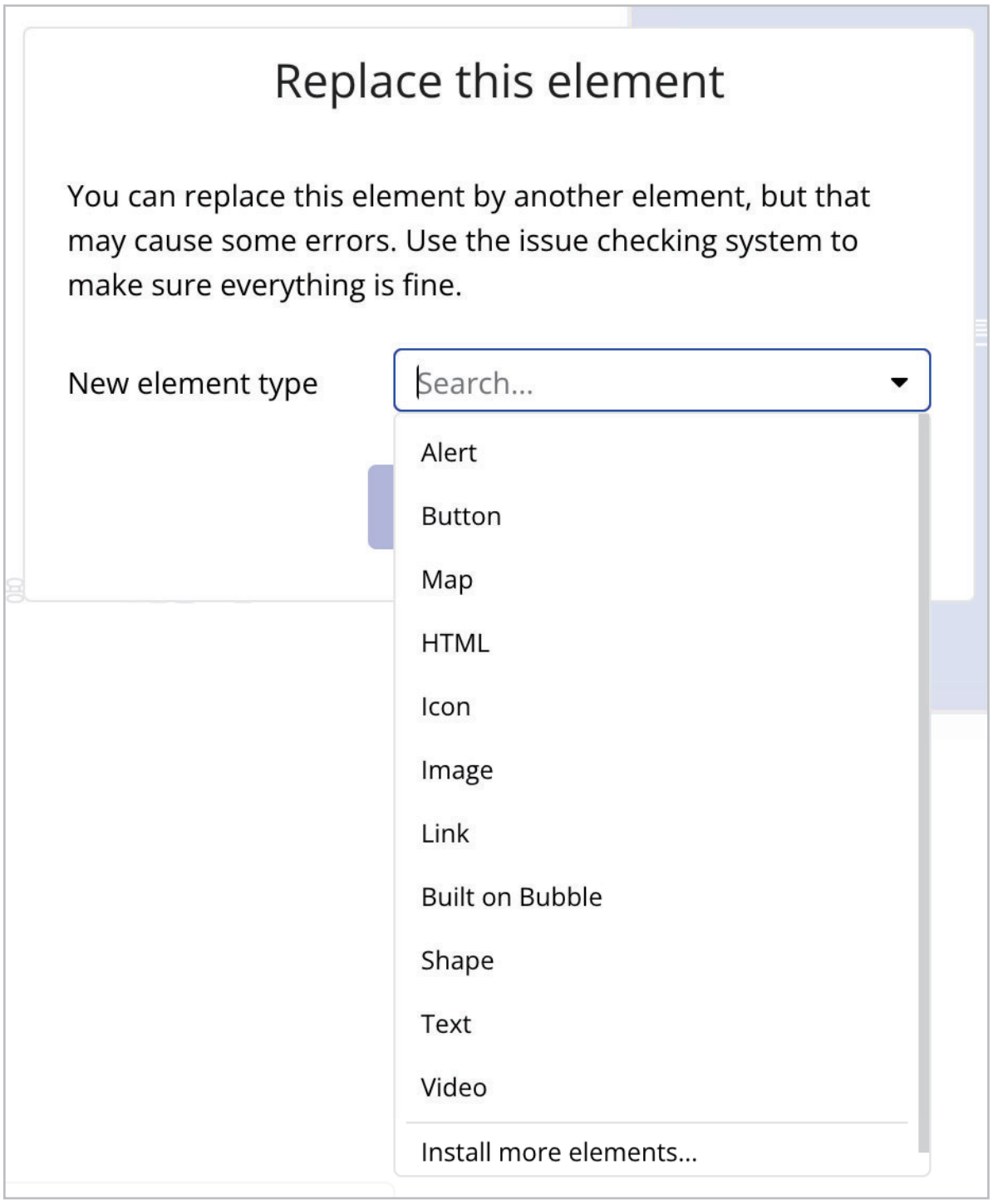
2. 그리드를 보여주고 간격을 색상이 있는 선으로 표시해줍니다.
3. 배열 메뉴는 유용한 분배 기능을 지원합니다. 예를 들면 세 요소 사이의 공간을 동일하게 하려는 경우에 쓰입니다. 또한 요소를 부모의 중심에 위치하도록 사용될 수도 있습니다(이는 Ctrl + E를 통해서도 가능합니다).

4. 문맥상의 메뉴(Contextual Menu) 활용하기

요소에서 우 클릭을 하면 이 메뉴가 나타납니다. 특정 복사 붙여넣기를 사용하면 많은 시간을 아낄 수 있습니다. 이 기능에는 스타일 혹은 조건부 핸들링을 함께 복사할 수 있고, 더 나아가서 워크플로우와 함께 복사도 가능합니다.



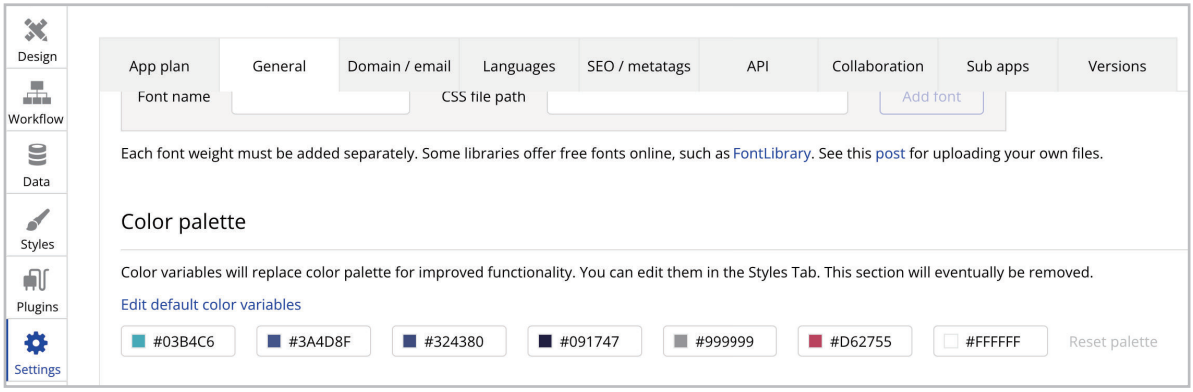
다른 요소 타입으로 전환시킬 수도 있습니다. 예를 들어 체크박스 요소를 아이오닉 토글로 대신하고 싶다면 체크박스 요소를 지우고 다시 아이오닉 토글을 추가할 필요가 없습니다. 다만 요소가 반환하는 데이터 유형이 다를 수 있기 때문에 몇 가지 예상외의 결과를 초래할 수 있습니다. 다만, 체크 박스를 날짜 입력으로 바꾸려고 할 때 에러가 발생할 수 있습니다.

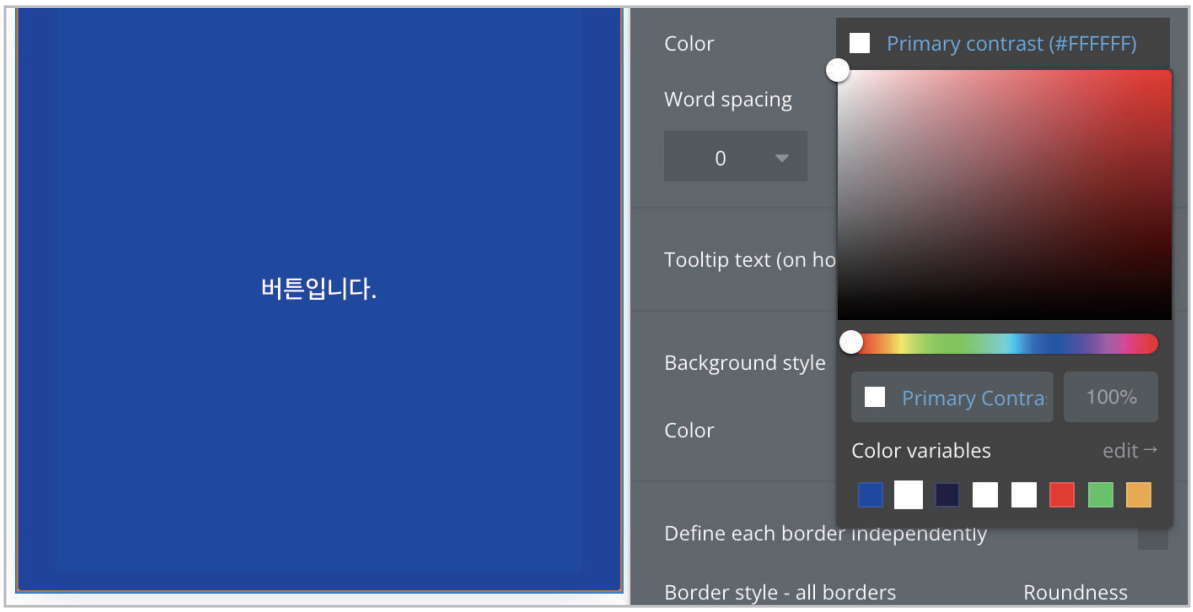


문맥상의 메뉴는 또한 선택된 요소들을 그룹으로 만들어주는 기능도 있습니다. 마지막으로 요소나 그룹을 재사용 요소로 전환시킬 수도 있습니다. 이는 제작 중에 자주 사용될 요소라는 걸 알았을 때 유용하게 쓰입니다.

5. 컬러 팔레트 커스터마이징

디자인의 일관성을 위하여 동일한 색상을 사용하는 것이 중요합니다. 이 과정을 쉽게 하기 위해 세팅 탭에서 미리 프리셋 색상들을 정의해 놓을 수 있습니다. 그리고 나면 이 색상들은 컬러 피커를 쓰는 곳 어디에서나 바로 접근이 가능합니다.

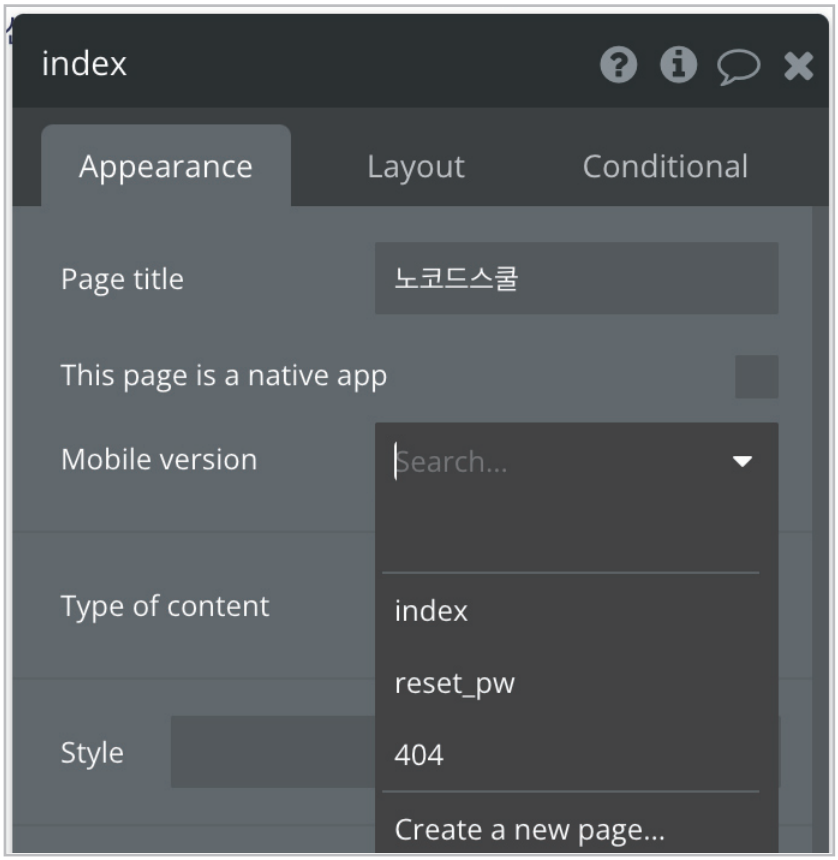




6. 반응형 페이지

반응형으로 페이지를 제작하는 게 처음에는 어색할 수 있습니다. 지정된 레이아웃대로 요소들이 배치되기 때문입니다. 하지만, 일단 반응형 레이아웃에 적응하면 원하는 배치를 자유롭게 할 수 있습니다. 버블에서는 설정 값을 바꾸자마자 피드백이 오기 때문에 여러 시도와 동시에 시작할 수 있습니다. 아래에는 반응형 디자인을 위한 몇 가지 유용한 기술입니다.

1. 화면 넓이를 채우는 경우가 필요하다면 그룹(또는 플로팅 그룹)을 사용하여 에디터에서 꽉 채워지게 설정합니다. 그리고 최대 너비를 설정하지 않으면 화면 가장자리까지 자동으로 확장됩니다.
2. 화면의 너비에 따라 폰트를 바꿀 수 있습니다. 현재 페이지의 너비와 부모 요소의 너비 / 높이는 조건부 탭에서 사용가능하기 때문에 너비가 줄어들면 폰트도 작게 설정할 수 있습니다. 모든 요소가 동일한 규칙을 따르도록 스타일에 적용하는 것이 좋습니다.
3. 모바일용으로 근본적으로 다른 디자인을 원한다면 아예 다른 페이지를 만들면 됩니다. 페이지를 만든 뒤 'mobile version'을 지정해주면 모바일 기기로 접속했을 때 'mobile version' 화면으로 로드됩니다.



PART 07

버블로 기능 넣기

01 CHAPTER 워크플로우

이번 챕터에서는 기능을 담당하는 워크플로우의 생성과 편집에 대해 알아보겠습니다.
워크플로우는 서비스가 수행하는 작업을 표현해주는 매체라고 생각하면 좋습니다.

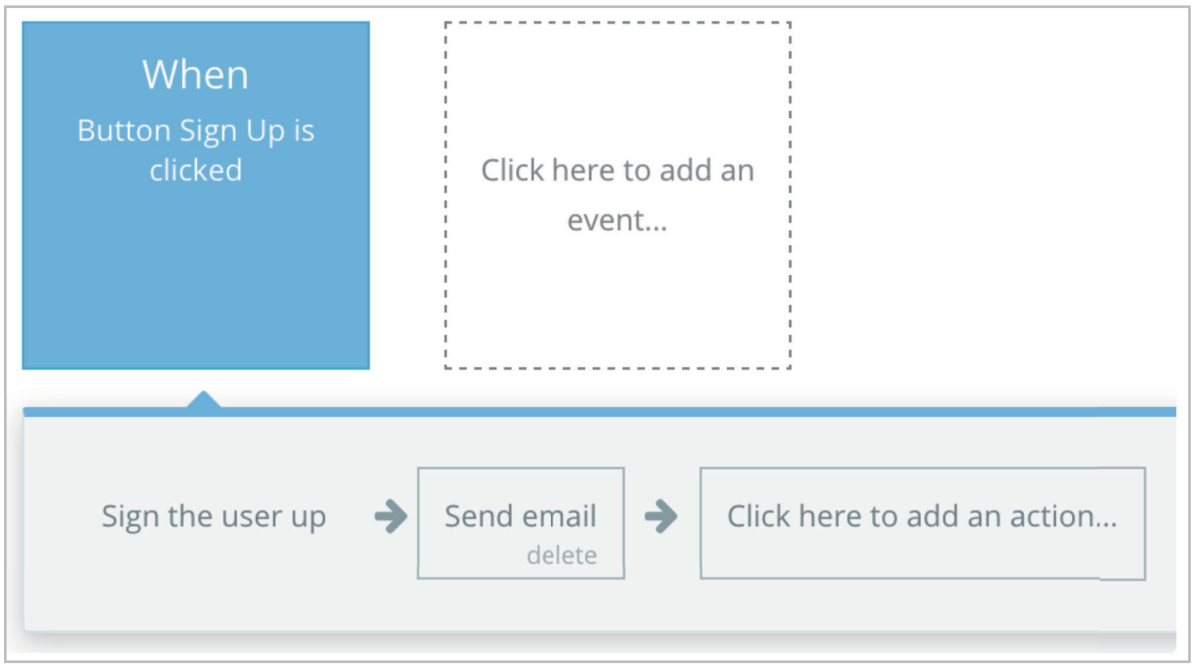
버블은 워크플로우를 통해 기능이 활성화합니다. 워크플로우란 사용자가 어떠한 일을 트리거 시켰을 때 실행되는 일련의 액션들을 뜻합니다. 예를 들어 사용자가 '저장' 버튼을 눌렀다면 '어떠한' 행동을 하도록 하는 것과 동일합니다.

워크플로우는 액션들을 모아놓은 이벤트로 구성이 되어 있습니다. 이벤트의 조합과 액션들의 묶음이 워크플로우를 정하게 됩니다. 워크플로우의 예시로 "'회원가입'버튼을 클릭하면, 회원가입을 시킨 뒤 이메일을 보내고 메인 페이지로 이동한다." 가 있습니다. 워크플로우들은 페이지에 특정되어 있고 좌측의 워크플로우 탭에서 편집할 수 있습니다. 만약 액션이 정보를 반환하게 된다면, 'Result of previous step'으로 해당 정보에 접근할 수 있습니다.

따라서 처음에 어떤 이벤트가 새로운 워크플로우를 시작할지 정해야 합니다. 대부분의 경우 버튼을 클릭하거나 이벤트를 적용시킬 어떠한 요소를 지정할 것입니다. 만약 워크플로우가 특정 상황에서만 실행되어야 한다면 조건을 추가해도 됩니다.

이벤트가 만들어 지면 워크플로우 패널이 보여지게 되고 그 안에서 각각의 액션들을 지정하면 됩니다. 각각의 실행을 위해서는 필수적으로 필요한 필드들이 있습니다. 예시로 회원가입을 할 때 이메일이 필요합니다.

워크플로우와 액션은 마우스 오른쪽 클릭을 통해 복사 혹은 붙여넣기 또한 가능합니다. 만약 이벤트를 복사 혹은 붙여넣기 했다면 워크플로우의 전체가 복사됩니다. 이외에도 각각의 액션 하나마다 복사가 가능합니다. 액션을 복사한 뒤 붙여 넣게 되면 액션은 선택된 액션 뒤에 붙여 넣어지게 됩니다.



주의!

만약 리피팅 그룹 안의 요소에 워크플로우를 달고 싶다면, 리피팅 그룹 안의 요소가 워크플로우를 실행시켜야 합니다.

1. 워크플로우의 실행 규칙

효율성을 위해 워크플로우는 서버와 프론드엔드에서 병렬로 실행됩니다. "Step 1", "Step 2"라는 이름에도 불구하고, 주어진 단계가 반드시 다음 단계를 트리거하기 전에 이전 단계가 완료될 때까지 기다리지는 않습니다. 다음은 버블 워크플로우에 대한 논리구조와 일반적인 추천에 대한 정보입니다. 그러니 각각의 스텝과 액션은 순서가 바뀔 수 있지만 순서가 중요한 순간에 참고하는 걸 추천합니다.

▣ 워크플로우가 실행되는 일반적인 규칙

- 프론트엔드 워크플로우 액션은 순서대로 실행되지만 다음 액션은 이전 액션이 끝나는 걸 기다려주지 않습니다.
- 백엔드 워크플로우는 순서에 상관없이 실행되자마자 작동하게 됩니다. 예를 들어, "Schedule API Workflow" 액션은 워크플로우의 마지막에 배치된 경우에도 워크플로우가 트리거 되는 즉시 실행됩니다.
- 커스텀 이벤트는 병렬이 아닌 순차적으로 실행됩니다. 워크플로우1이 워크플로우2를 시작하는 커스텀 이벤트를 작성하는 경우 워크플로우2는 워크플로우1의 나머지 작업이 실행되기 전에 완료됩니다.

- 탐색은 항상 새로운 데이터로 업데이트 되어있는 상태에서 실행되지 않습니다. 그러므로 만약 새로운 아이템을 생성한 다음 검색을 통해 검색하려고 하면 작동하거나 작동하지 않을 수 있습니다. 고로 이 항목에 의존해서는 안 됩니다.
- "result of step X"에서 가져올 때 step X는 "Create…" 에서 미리 저장된 것입니다.

▣ 워크플로우의 일관성을 위해서 추천하는 방법

- 워크플로우 트리거(예: 버튼)가 조건에 따라 다양한 결과를 가져올 수 있다면, 워크플로우를 여러 개 만든 뒤 워크플로우의 레벨에서 조건을 다는 것이 하나의 워크플로우 안에 조건이 달린 액션들을 넣는 것 보다 더 안전합니다.
- 두 개의 액션을 가진 워크플로우에서 만약 Step2가 Step1에서 처리된 데이터를 검색하는 조건이 있을 경우 Step1이 끝난 뒤에 실행되는 것을 확정할 수 있도록 커스텀 이벤트로 실행되어야 합니다.
- 백엔드 워크플로우가 다른 워크플로우의 단계 이후에 실행되어야 한다면, 먼저 필요한 단계 이후에 배치되는 커스텀 이벤트에서 실행되어야 합니다.
- 어떠한 단계에서 데이터를 사용하기 가장 안전한 방법은 검색보다는 "Result of step X"를 사용하는 것 입니다.
- 버블은 다음 단계로 넘어가기 전에 워크플로우를 대기시키는 액션을 지원하지 않습니다. 그러나 "add a pause before next action" 액션은 효과적인 대안책입니다.

주의!

만약 사용자가 같은 워크플로우를 빠른 시간 내에 너무 많이 작동시키면, 버블은 융통성 있게 그 워크플로우를 실행시킵니다.
데이터 수정이 필요한 워크플로우는 반복적으로 실행시키되 다른 페이지로 이동하는 것과 같은 일회성 액션은 반복하지 않습니다.

2. 이벤트 & 액션 타입

이벤트와 액션은 메뉴에서 카테고리별로 정렬됩니다. 보통은 세 가지의 주요 이벤트 타입이 있습니다.

요소 이벤트

이 이벤트들은 버블로 서비스를 만들 때 가장 기본이 됩니다. 이들은 사용자가 요소와 상호작용할 때 시작됩니다. 예를 들어 지도의 마커가 클릭 되었을 때 혹은 버튼이 클릭 되었을 때와 같습니다. 각 이벤트에서 여러분들은 어떤 요소에 적용되는 지 지정해주어야 합니다. 요소 이벤트를 만들 때 페이지에 관련된 요소들만 선택하도록 보여 집니다.

일반 이벤트

이 이벤트들은 버블 앱에서 일반적인 프로퍼티들이 바뀌었을 때 시작됩니다. 예를 들어 '현재 사용자가 로그인 되어있을 때' 혹은 '어떤 조건이 충족되었을 때..' 등이 있습니다. 이것은 사용자의 특정 액션과는 관계가 없습니다. 이러한 일반 이벤트는 어떤 것이 변경되었을 때 발생하지만, 사용자가 수행한 것이 아닐 수 있기 때문에 디버깅하는데 더 복잡합니다. 이때 디버거는 이러한 상황을 분석하는 데 유용합니다(이후 챕터에서 다루도록 하겠습니다).

커스텀 이벤트

이 이벤트는 다른 워크플로우에서 사용할 수 있는 재사용 가능한 일련의 액션들을 정의할 수 있습니다.
이벤트가 아닌 액션 또한 몇 개의 카테고리로 분류가 가능합니다.

계정 관리

이 액션들은 회원가입, 로그인, 로그아웃과 같이 계정에 관련된 액션이 있습니다.

네비게이션

사용자가 다른 페이지로 이동할 수 있게 하는 액션입니다.

데이터 (레코드)

데이터를 읽고 쓰는 액션입니다.

이메일

사용자에게 이메일을 보내는 것과 관련된 액션들이 존재합니다.

결제 & 분석

결제와 분석에 관련해서 버블에 내장된 액션들입니다(빌트 인 플러그인의 설치가 필요합니다).

플러그인

대부분의 플러그인에서 사용하는 액션들이 모여 있습니다.

요소

요소에 특정된 액션들입니다. 예를 들어 반복 그룹을 보여주거나, 커스텀 상태를 설정하거나, 특정 요소로 스크롤시키는 액션이 있습니다. 각 액션은 해당 페이지에서 선택할 수 있는 요소들이 나열됩니다.

3. 클라이언트 vs. 서버

어떤 액션은 서버에서만 실행될 수 있고, 어떤 액션은 클라이언트에서만 실행될 수 있으며, 둘 다 인 경우도 있습니다. 일반적으로 데이터를 편집하거나 외부의 서비스를 연결할 때는 두 환경에서 모두 실행되어야 합니다. 페이지를 이동하거나, 요소를 토글 혹은 스크롤 하거나, 커스텀 상태를 변경하는 것은 클라이언트 측에서만 실행되는 액션들입니다. 추가로 클라이언트의 워크플로우는 버블 앱에서 볼 수 있는 서버 로그에는 기록되지 않습니다.

4. 에러 핸들링

워크플로우가 에러를 발생시키면 그 즉시 워크플로우는 멈추게 됩니다. 예를 들어 회원가입을 할 때 비밀번호 확인이 틀린다든지 카드 결제가 실패했을 때 발생합니다. 이때 에러가 나기 전에 실행된 액션, 수정된 데이터, 전송된 이메일들은 다시 돌아오지 않습니다. 그 뒤에 실행될 액션들은 실행되지 않습니다. 오류가 발생했을 때 메세지를 변경하거나 사용자 경험을 달리 하고 싶다면 커스텀 핸들링을 통해 처리할 수 있습니다.

워크플로우는 타임아웃도 가능합니다. 만약 워크플로우가 5분 이상 실행되고 있다면 에러를 발생시키고 멈추게 됩니다. 일반적으로 워크플로우는 이러한 타임아웃이 발생했을 경우 워크플로우를 다시 설계해야 합니다. 예를 들어, “Make changes to a list of things” 액션에 복잡한 쿼리를 사용했을 경우 데이터가 적을 때는 잘 작동하지만 데이터 수가 많아지면 타임아웃이 오게 됩니다. 이 경우를 다시 설계 하자면, 버블은 API 워크플로우를 만들어 “Schedule API Workflow on a list”를 사용하거나 다른 방법으로 워크플로우를 쪼갤 수 있습니다. 타임아웃이 왔는지는 서버 로그에서 확인할 수 있습니다.

워크플로우는 용량이 충분하지 않은 상황에서 실행했을 때 발생하기도 합니다. 이 상황은 같은 시간에 많은 양의 워크플로우가 실행되어야 할 때 발생할 수 있습니다. 이 경우에는 용량에 관련한 메세지를 서버 로그에서 볼 수 있습니다. 이런 상황을 완화시키기 위해서는 워크플로우 간격을 가능한 좁히거나, 플랜을 업그레이드 하거나, 더 많은 양의 용량을 구매하면 됩니다.

주의! – 5분 단위 작업에 대한 시간 초과 오류

작업 단위가 오래 걸리게 되면 예상치 않게 시간 초과 오류로 종료될 가능성이 높아집니다. 따라서 모든 작업 단위의 제한 시간은 300초입니다. 작업 단위가 5분 이내에 완료되지 않으면 오류가 발생하여 작업이 중지되기 때문입니다. 이때 말하는 가장 일반적인 작업 단위는 http요청 및 워크플로우입니다.

워크플로우에서는 시간 초과 오류 이전에 완료된 작업은 각 작업 단위가 새 타이머를 시작하기 때문에 영향을 받지 않습니다. 따라서 워크플로우 A가 워크플로우 B를 성공적으로 예약했다면 워크플로우 A가 강제 종료 되더라도 워크플로우 B는 예약된 대로 작동하게 됩니다. 만약 워크플로우 B가 5분 이내에 완료될 수 있다면, 모든 워크플로우는 완료될 것입니다. 각각의 워크플로우는 비동기적으로 발생합니다.

만약 시간 초과 오류가 워크플로우에서 발생한다면 해당 워크플로우는 재설계할 필요성이 있습니다. 이러한 경우의 사례는 아래와 같습니다.

- 한 시간마다 어떤 것을 실행하고 싶다면 워크플로우 A가 한 시간 후에 워크플로우 B를 예약해놓게 하고 싶을 경우
- 많은 수의 레코드에 대해 동일한 작업을 수행하고 싶을 때, 워크플로우 A는 적은 수의 레코드에서 먼저 실행된 후에 다음 워크플로우 A에서는 그 뒤의 레코드부터 이어서 실행 되도록 하고 싶을 경우

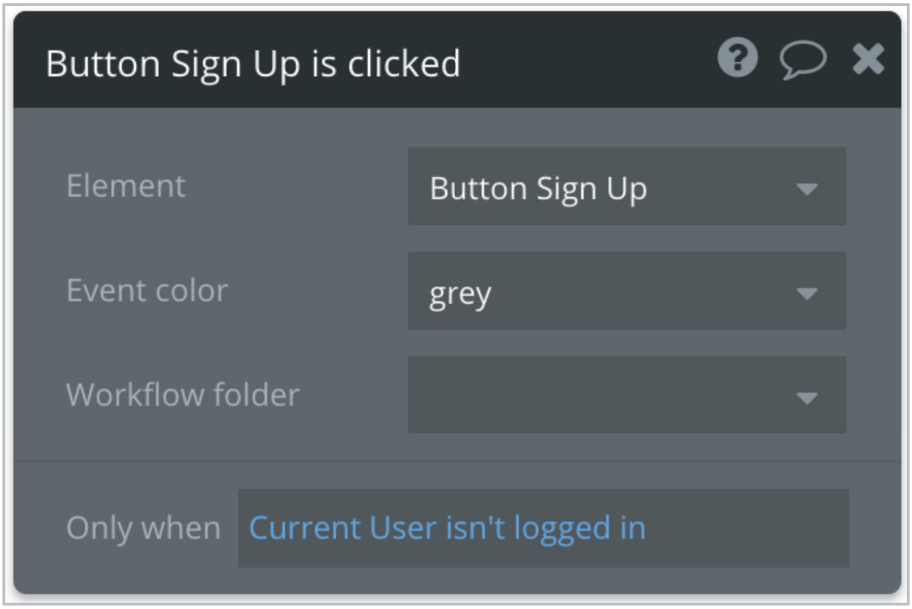
02 CHAPTER 조건부 핸들링

서비스를 만들다 보면 꽤 자주 특정 상황에서만 워크플로우가 실행되어야 하는 경우가 있습니다. 사용자가 로그인이 되었을 때 주문이나 충전이 가능하게 하거나, 동의를 체크한 경우에만 회원가입이 가능한 경우를 사례로 들 수 있습니다. 이런 것을 가능하게 하기 위해 버블은 이벤트와 액션에 조건을 달 수 있도록 해놓았습니다.

1. 이벤트 조건

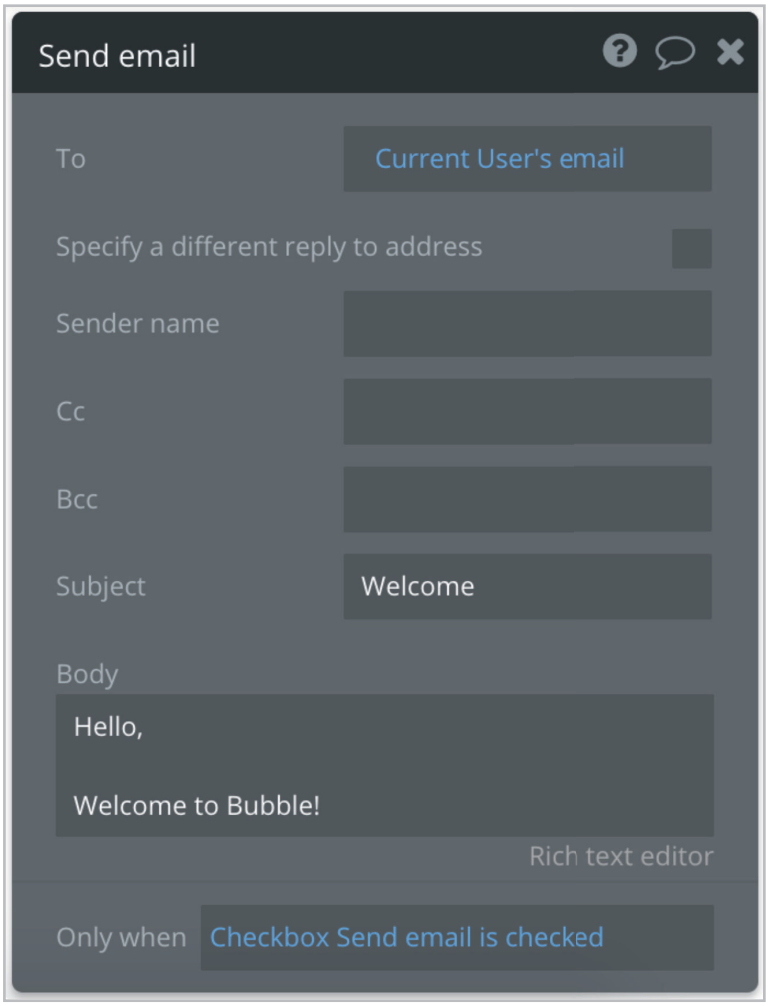
조건은 이벤트 레벨에서 적용될 수 있습니다. 이 경우에는 조건이 no로 계산되면 실행되지 않고 yes로 계산되면 실행됩니다. 조건이 따로 붙지 않으면 모든 경우에 실행됩니다.

액션에 조건을 달 경우 요소의 조건부 핸들링과 비슷하게 정의할 수 있습니다. 이때 주의해야 할 점은 조건은 항상 yes/no로 계산되어야 하고 아닐 경우에는 이슈체커가 에러를 기록합니다.



2. 액션 조건

더 작은 단위에서의 조건부 핸들링이 필요하다면 특정 액션에 조건을 부여할 수도 있습니다. 이때 적용하는 조건은 yes/no로 계산되어야 하는 이벤트 조건과 유사합니다. 조건이 yes로 끝나거나 조건이 없다면 액션은 실행될 것이고, 조건의 결과가 no로 계산될 경우에는 해당 액션이 생략되고 다음 액션으로 넘어가게 됩니다.



3. 조건 디버깅

실행 모드에서 조건에 따라 어떤 이벤트는 동작하고 어떤 이벤트는 동작하지 않는지를 분석하려면 디버깅이 자주 필요합니다. 버블은 디버거에서 step-by-step 모드를 통해 어떤 조건이 yes로 계산되어 실행되는지 그리고 그것이 워크플로우에 어떤 영향을 주는지 이해할 것을 권유합니다.

다음 액션에서 조건으로 인해 건너뛴 이전 액션의 결과를 사용하는 경우 이전 작업의 결과는 비어 있는 상태입니다. 이러한 문제를 발생시키지 않도록 하는 것은 서비스 빌더로서 여러분의 몫입니다. 이러한 경우를 방지하기 위해 디버거를 잘 활용해야 합니다.

03 CHAPTER 커스텀 이벤트

커스텀 이벤트는 여러 번 재사용 될 수 있는 워크플로우를 정의할 수 있는 방법이며 코드 프로그래밍에서는 이를 "함수"라고 합니다.

1. 커스텀 이벤트를 사용하는 Case

전통 프로그래밍이든지 버블에서든지 사람들은 반복되는 실행, 워크플로우, 코드들을 줄이려고 합니다. 이는 서비스를 보다 더 간결하게 만들고 디버깅, 수정, 유지보수를 더 쉽게 만들어 줍니다.

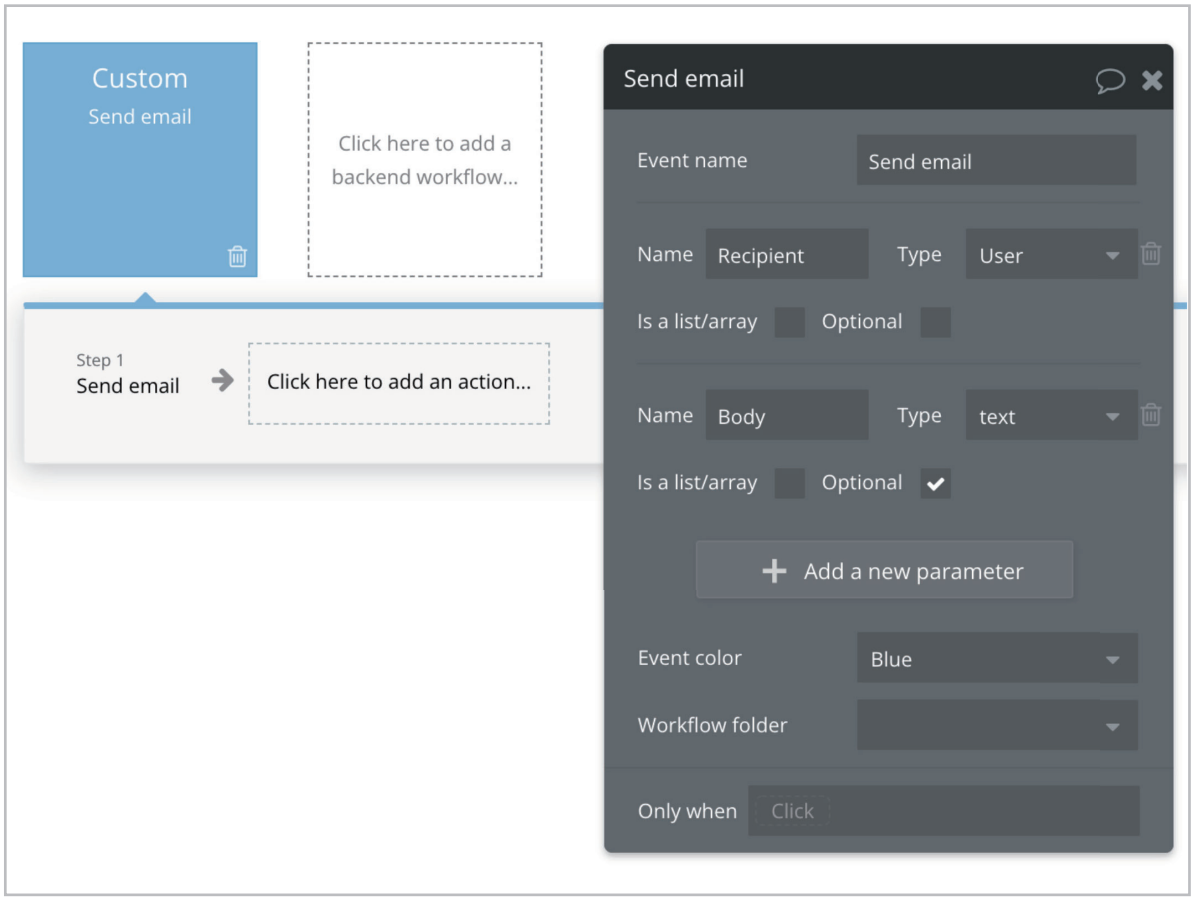
만약 서비스를 개발하다 같은 로직이 반복되는 것을 발견했다면, 다른 말로 같은 일련의 액션들이 하나 이상의 워크플로우에서 생겼다면, 워크플로우에서 실행할 "커스텀 이벤트"로 최적화를 진행할 수 있습니다.

커스텀 이벤트를 사용하는 방법에는 여러가지가 있습니다. 그 가운데 사용자가 회원가입을 하는 두 가지의 사례를 들어보겠습니다. 단순 회원가입을 할 때와 회원가입 후 이메일을 보내는 경우가 있다고 합시다. 이 경우 반복되는 일련의 액션들을 커스텀 이벤트로 옮긴 뒤에 회원가입을 하는 두 워크플로우에 모두 커스텀 이벤트를 실행시키면 됩니다.

2. 커스텀 이벤트 정의하기

커스텀 이벤트를 만드는 방법은 일반 워크플로우를 만드는 방법과 유사합니다. 이벤트를 선택한 뒤에 액션을 추가하면 됩니다. 여기에서 이벤트 타입은 'Custom Event'입니다. 다른 이벤트 타입과 비슷하게 조건을 붙일 수도 있습니다. 하나의 특정 필드는 이 워크플로우에 대해 하나 이상의 파라미터를 정의할 수 있게 해줍니다. 전통적인 프로그래머들에게는 커스텀 이벤트가 함수로 더 익숙하기 때문에, 이 특정 필드는 파라미터로 더 익숙할 것 입니다. 커스텀 이벤트의 파라미터로 가능한 것은 여러분이 정의한 데이터 타입이나 텍스트, 숫자와 같은 단순 타입들이 있습니다. 게다가 이 파라미터가 배열인지 옵션인지 설정할 수도 있습니다.

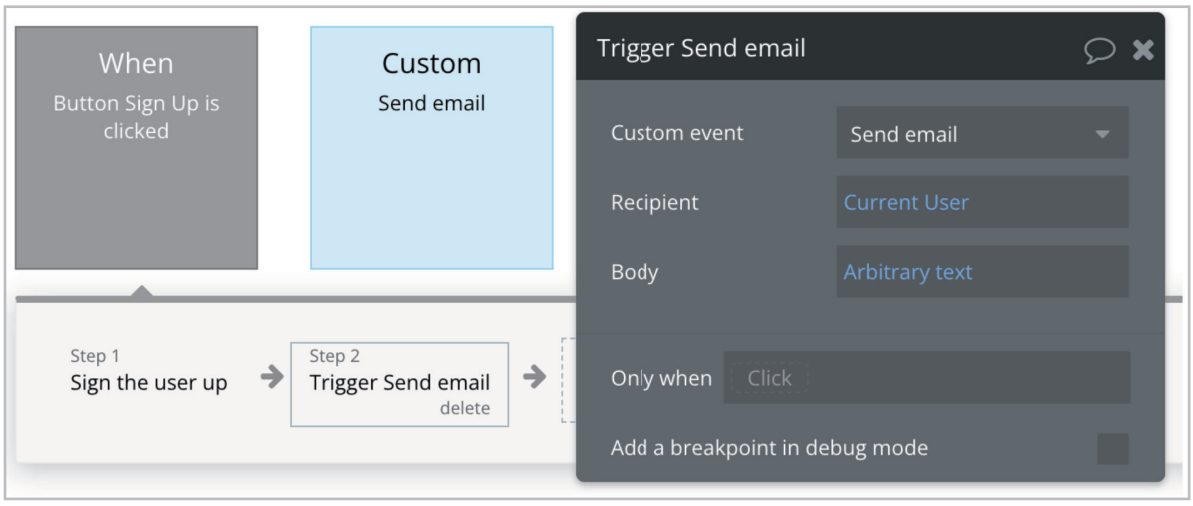
커스텀 이벤트에 적어도 하나 이상의 파라미터를 정의했을 때, 이후 액션에서 접근 가능하게 됩니다. 예를 들어 커스텀 이벤트에서 사용자를 받았다면 This Workflow Users email'를 통해 이메일을 보낼 수 있습니다. 이 표현은 액션을 실행할 때 전달된 사용자로 인식됩니다. 만약 같은 커스텀 이벤트를 하나 이상의 페이지에서 재사용하고 싶다면, 커스텀 이벤트를 재사용 요소에 추가하면 됩니다. 이럴 경우 어떤 페이지에서라도 이 재사용 요소가 있다면 커스텀 이벤트에 접근할 수 있습니다.



3. 커스텀 이벤트 트리거 시키기

커스텀 이벤트가 정의되면 이제 워크플로우에 커스텀 이벤트의 워크플로우를 추가할 수 있습니다. 일단 Trigger custom event action'(또는 Trigger custom event action from a reusable element')을 선택하면 됩니다.

만약 하나 이상의 파라미터가 정의되어 있다면 액션에 보내줄 데이터를 같이 넘겨주어야 합니다. 당연히 전송될 데이터는 받아야 하는 파라미터의 타입과 일치해야 합니다. 불일치할 경우 이슈체커가 에러 목록에 추가합니다.



4. 커스텀 이벤트 스케줄링

커스텀 이벤트가 나중에 실행되도록 예약할 수도 있습니다.

5. [심화] 트리거 vs. 스케줄링

커스텀 이벤트를 실행시키는 것과 커스텀 이벤트를 0초의 딜레이로 예약하는 것은 다르다고 생각할 것입니다. 그렇게 생각했다면 맞는 이야기입니다.

액션들을 가지고 있는 부모 워크플로우가 있다고 합시다. 그리고 "Trigger a custom event"에 또 다른 액션들이 있습니다. 부모 워크플로우가 커스텀 이벤트를 실행하면 부모 워크플로우는 멈추게 되고 다음 액션을 실행하기 전에 커스텀 이벤트가 완료되길 기다립니다. 이 부분은 커스텀 이벤트가 부모 워크플로우에서 이전에 실행된 문맥을 볼 수 있기 때문에 부분적으로 편리합니다.

반면, 커스텀 이벤트를 0초의 딜레이로 예약했다면 해당 커스텀 이벤트는 부모 워크플로우의 나머지 액션과 병렬로 실행될 수 있습니다.

04 CHAPTER [Tips] 버블 프로그래밍 팁

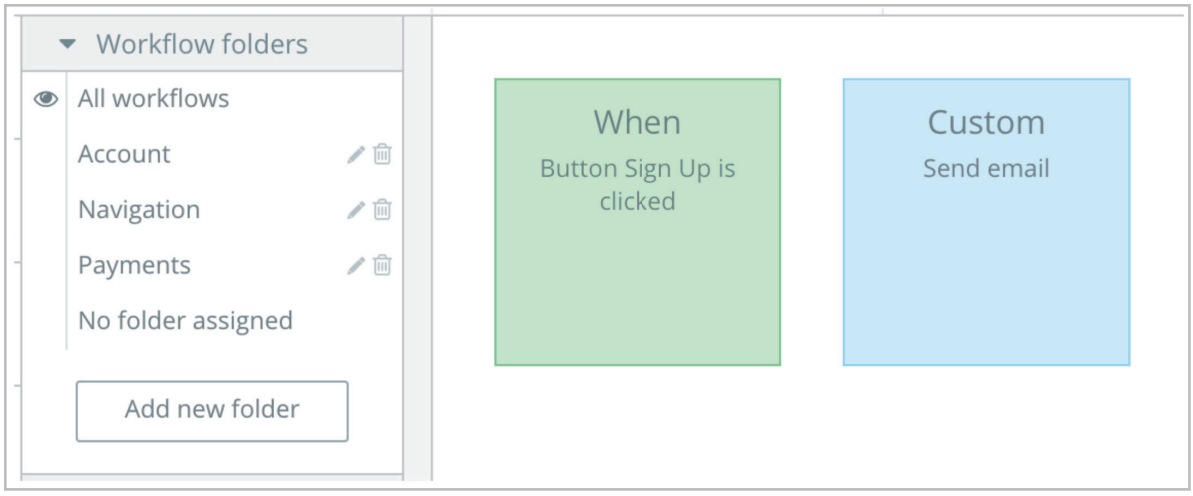
서비스가 점점 복잡해지게 되면 어떤 페이지들은 많은 워크플로우들을 가지고 있게 됩니다. 버블의 워크플로우 탭에는 워크플로우를 구성하는 데 도움이 되는 몇 가지 도구가 추가되었습니다.

1. 정돈된 워크플로우로 유지하기

이동 워크플로우, 데이터 조작 워크플로우와 같이 워크플로우들을 폴더로 구성하여 관리할 수 있습니다. 워크플로우 탭 바의 화살표 모양을 클릭하면 폴더 뷰를 보여주게 됩니다. 여기서는 여러분이 만든 서로 다른 폴더를 볼 수 있고 새로운 폴더를 만들거나 삭제할 수 있습니다.

폴더가 생성되면 각 워크플로우를 폴더에 넣을 수 있습니다. 폴더를 클릭하면 해당 폴더에 있는 워크플로우들을 볼 수 있습니다. 워크플로우에 색깔도 지정할 수 있습니다. 이 색상은 순전히 보기를 위한 기능이고 워크플로우의 실행에는 전혀 상관이 없습니다. 워크플로우를 정돈할 때 특정 워크플로우에 특정 색상을 부여하면 더 편리합니다.

버블은 이벤트나 액션에 자동으로 이름을 부여합니다. 하지만 그것은 클릭해서 나오는 프로퍼티 에디터를 통해 쉽게 바꿀 수 있습니다. 요소와 유사하게 워크플로우도 이름을 규칙적으로 바꾸면 유지보수에 편리합니다.



2. 단축키 사용

버블 앱을 만들다 보면 페이지 요소와 워크플로우가 연관되어 있을 경우, 디자인 탭과 워크플로우 탭을 왔다갔다 하는 여러분을 발견하게 될 것입니다. 이때 요소나 워크플로우에서 마우스 우 클릭을 사용하여 'Reveal workflow' 나 'Reveal element'를 선택하면 됩니다. 이 기능은 여러분이 편집하고 있는 요소나 워크플로우를 하이라이트 처리해 줄 것입니다.

요소의 프로퍼티 에디터에 있는 'Start/edit workflow'버튼도 해당 요소와 사용자가 상호작용 했을 때 실행될 워크플로우로 데려다 주는 또 다른 단축키입니다. 워크플로우가 없을 경우 해당 워크플로우가 생성되기도 합니다.

PART 08

서비스 테스팅

01 CHAPTER 테스팅 기본

이번 챕터에서는 서비스 테스팅에 대한 기본을 알려드리겠습니다.

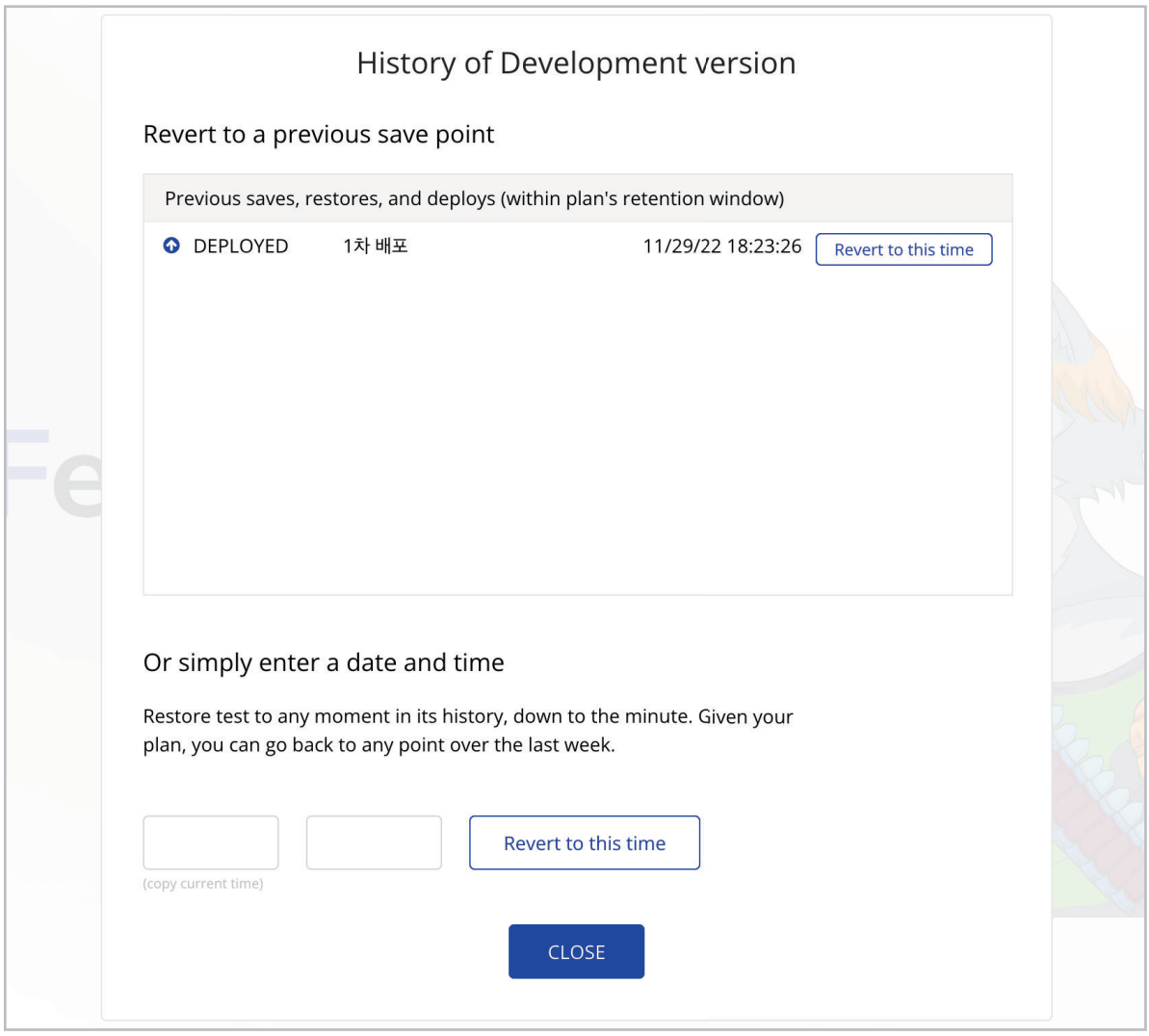
테스팅은 단순하게 말해 서로 다른 스텝들을 시도해 보는 것입니다. 디버깅은 예상하지 않은 행동에 대해 분석하거나 풀리기 전에 어떤 일이 일어났는지 이해하는 것입니다. 위의 두 케이스는 정의로는 서로 다르지만, 둘 다 여러분 서비스의 과정에 가이드를 해주는 방법입니다.

1. 개발 버전 사용

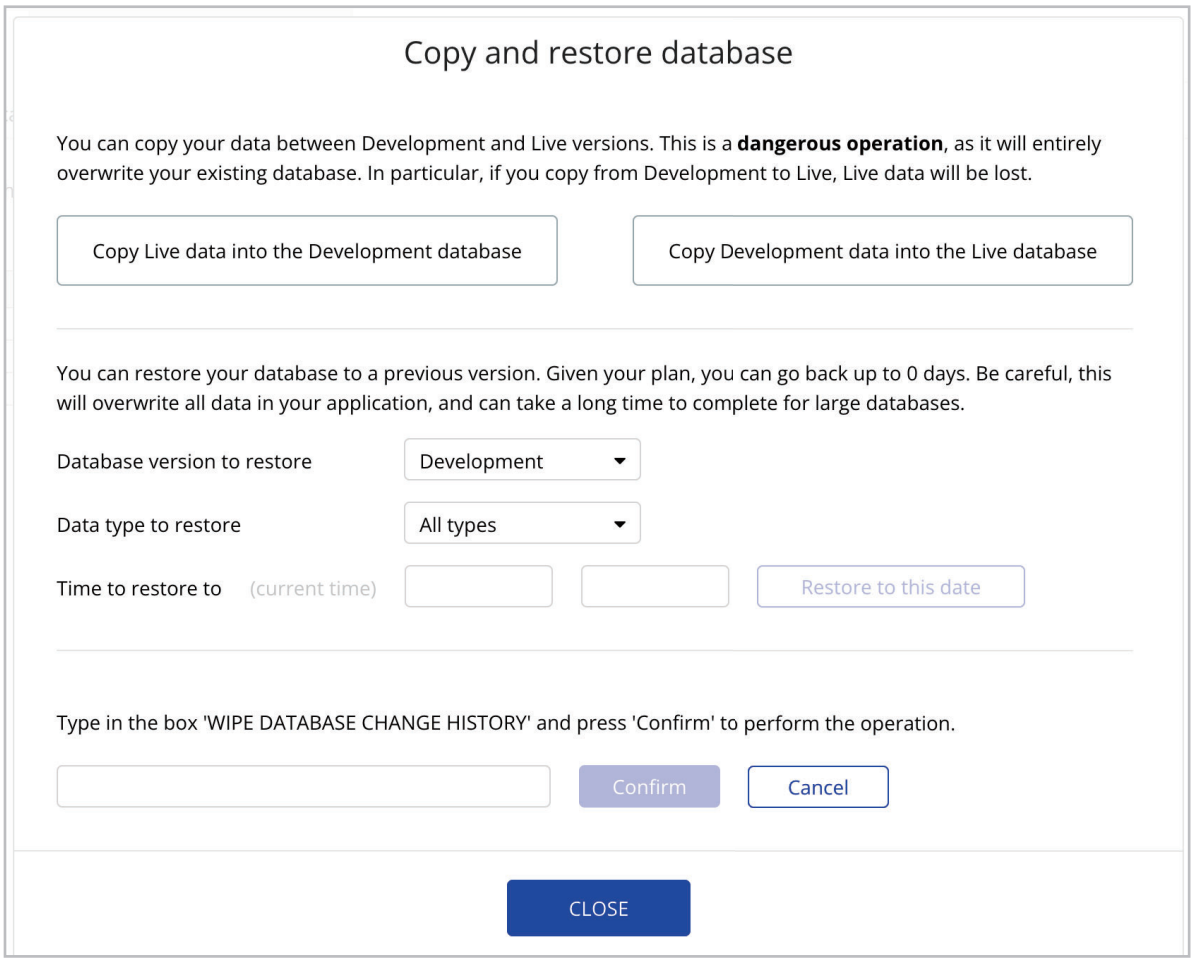
버블 앱은 두 개의 버전이 있기 때문에 개발 버전에서 테스트와 디버깅을 하길 추천합니다. 첫 번째로, 개발 버전은 디자인과 워크플로우를 수정할 수 있는 버전이기 때문입니다. 두 번째로는 워크플로우에 관련된 데이터베이스를 테스팅 할 수 있습니다. 만약 여러분이 만든 변경사항에 확신이 들지 않는다면, 라이브 데이터는 실제 유저에게 영향을 주기 때문에 개발 버전의 데이터를 시험 삼아 변경해보는 게 좋습니다.

그러나 사용자에 의해 작동하는 경우에서 라이브 버전에서만 에러가 발생되는 상황이 있을 수 있습니다. 이러한 경우는 아래의 두 가지 방법을 추천합니다.

(1) 배포하기 안정적인 상태일 때 최신버전으로 배포를 해놓습니다. 만약 이슈가 개발 모드가 아닌 라이브 모드에서만 발생할 경우, 버그가 이미 개발에서 수정되었을 수 있으며 배포를 통해 문제가 해결되었는지 확인해야 합니다.



(2) 만약 이슈가 특정 유저에게서만 발생한다면 특정 데이터가 라이브 모드에서 저장되어 나타나는 현상일 수 있습니다. 버블은 여러분의 데이터베이스를 버전을 넘나들며 복사할 수 있게 지원합니다. 이러한 경우에는, 라이브 버전의 데이터베이스를 개발 버전으로 복사하길 추천합니다. 그렇게 하면 여러분의 개발 버전 데이터베이스가 라이브 모드와 동일해지고, 배포모드와 동일한 환경에 가까이 디버깅할 수 있게 됩니다. 그 후 여러분들이 그 특정 유저로 서비스를 실행시켜 이슈를 점차 좁혀나가며 해결할 수 있게 됩니다.

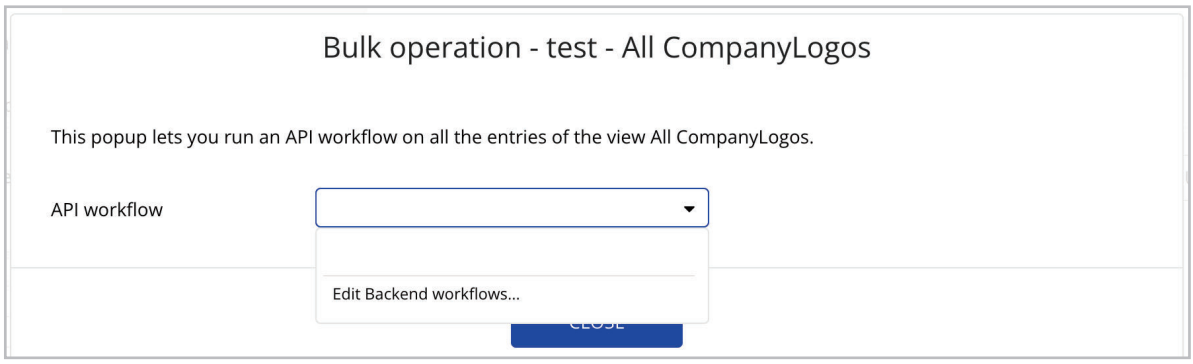


2. 이슈를 확인할 때의 팁

이슈를 만났을 때와 원하지 않는 행동이 발생되었을 때, 이러한 문제를 해결하기 가장 좋은 방법은 문제 상황을 재현할 수 있는 신뢰 가능한 결정적인 방법을 찾는 것입니다. 이는 100% 가능하지 않을 수도 있지만, 서비스를 디버깅할 때 첫 번째 단계가 되어야 합니다.

이상적으로는 항상 문제를 재현하면서 가능한 한 단순하고 짧은 일련의 액션 및 상호작용을 서비스에서 찾고자 할 것입니다.

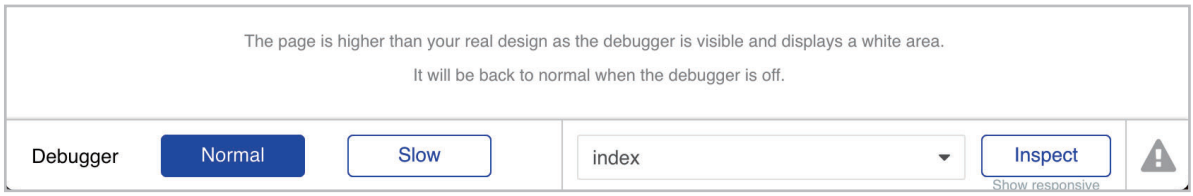
좋은 접근으로는 새로운 데이터 셋을 이용해 이슈 상황을 재실행하는 방법입니다. 예를 들면 새로운 사용자나 새로운 레코드와 함께 말입니다. 데이터 레거시 상황으로 인해 문제가 발생할 수 있으며, 새로운 데이터 셋으로 문제를 재현할 수 없다면 이는 위의 경우에 해당한다는 강력한 힌트입니다. 이때는 전체 워크플로우를 변경하는 대신 이슈를 발행시키는 항목을 추출하여 대량 작업을 사용해 항목들을 수정하는 것이 해결방법이 될 수 있습니다.



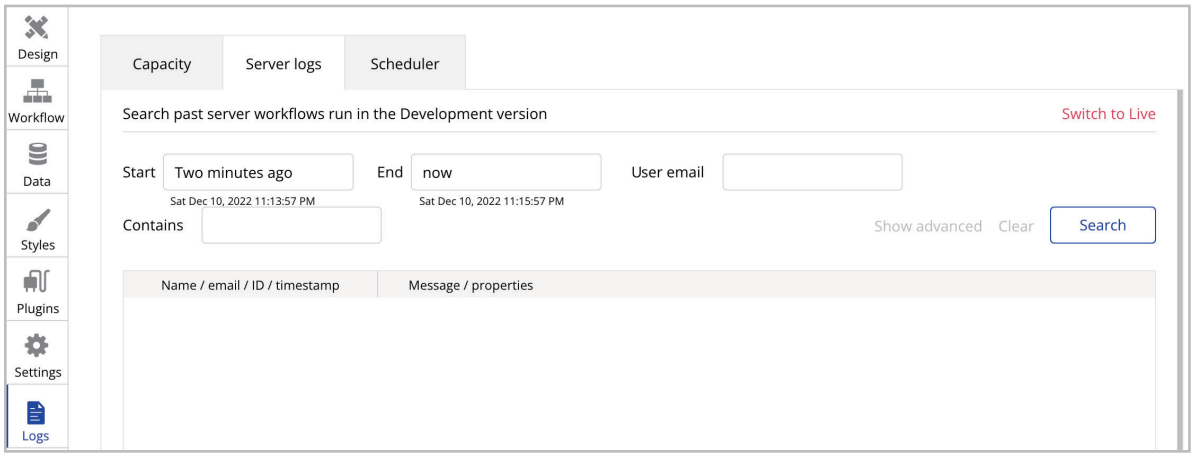
3. 테스팅 & 디버깅 툴

버블은 상황에 따라 이슈를 디버깅할 수 있는 두 가지의 방법을 제공합니다.

가장 기본적인 방법은 디버거로, 여러분의 워크플로우를 각각 액션마다 실행시켜볼 수 있고 요소의 필드값들의 변화를 지켜볼 수 있습니다.



두 번째 방법으로는 과거 발생한 이슈를 진단하는 서버 로그입니다. 서버 로그에서는 워크플로우에서 어떤 일이 일어났는지 분석할 수 있게 해줍니다.

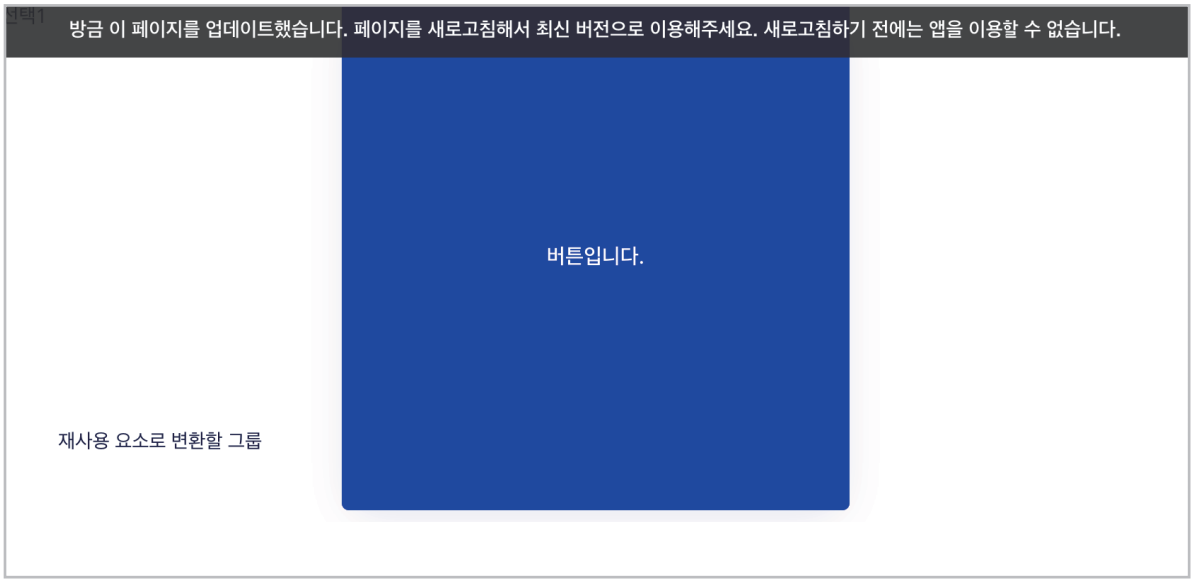


다른 주요 툴은 다른 유저로 실행시켜보는 방법입니다. 이는 여러분들의 계정이나 테스트 계정이 이슈를 재생산해내지 않는 경우 특정 유저가 이를 이슈로 발생시킬 때 사용하기 적합합니다.

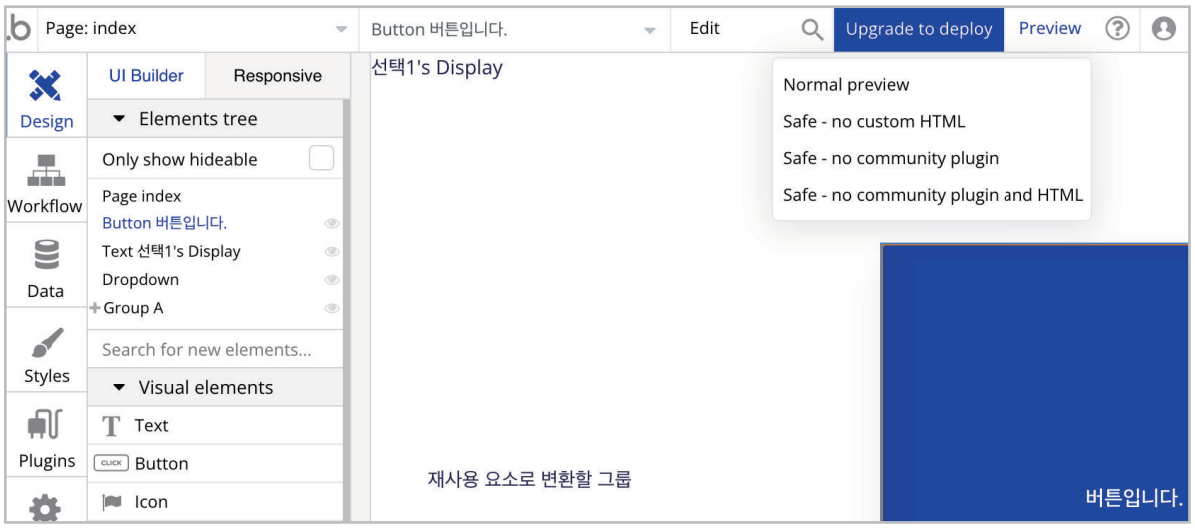
4. 안전 모드 사용

여러분은 언제라도 미리보기를 눌러서 현재 편집하고 있는 페이지를 실행시켜 볼 수 있습니다. 변경사항들이 실시간으로 저장되므로 실행 모드 탭에 표시되는 버전은 항상 최신 버전입니다(미리보기 버튼을 클릭한 후의 경우). 기본적으로 이 버튼을 통해 서비스를 실행할 때 디버거가 켜집니다.

실행 모드 탭을 열고 앱을 수정하다 보면 실행 중인 버전이 에디터의 버전보다 오래된 버전인 즉시 상단에 알림이 표시됩니다. 해당 메시지를 클릭하면 페이지가 새로 고쳐지고 실행 모드 탭에서 최신 버전이 실행됩니다.



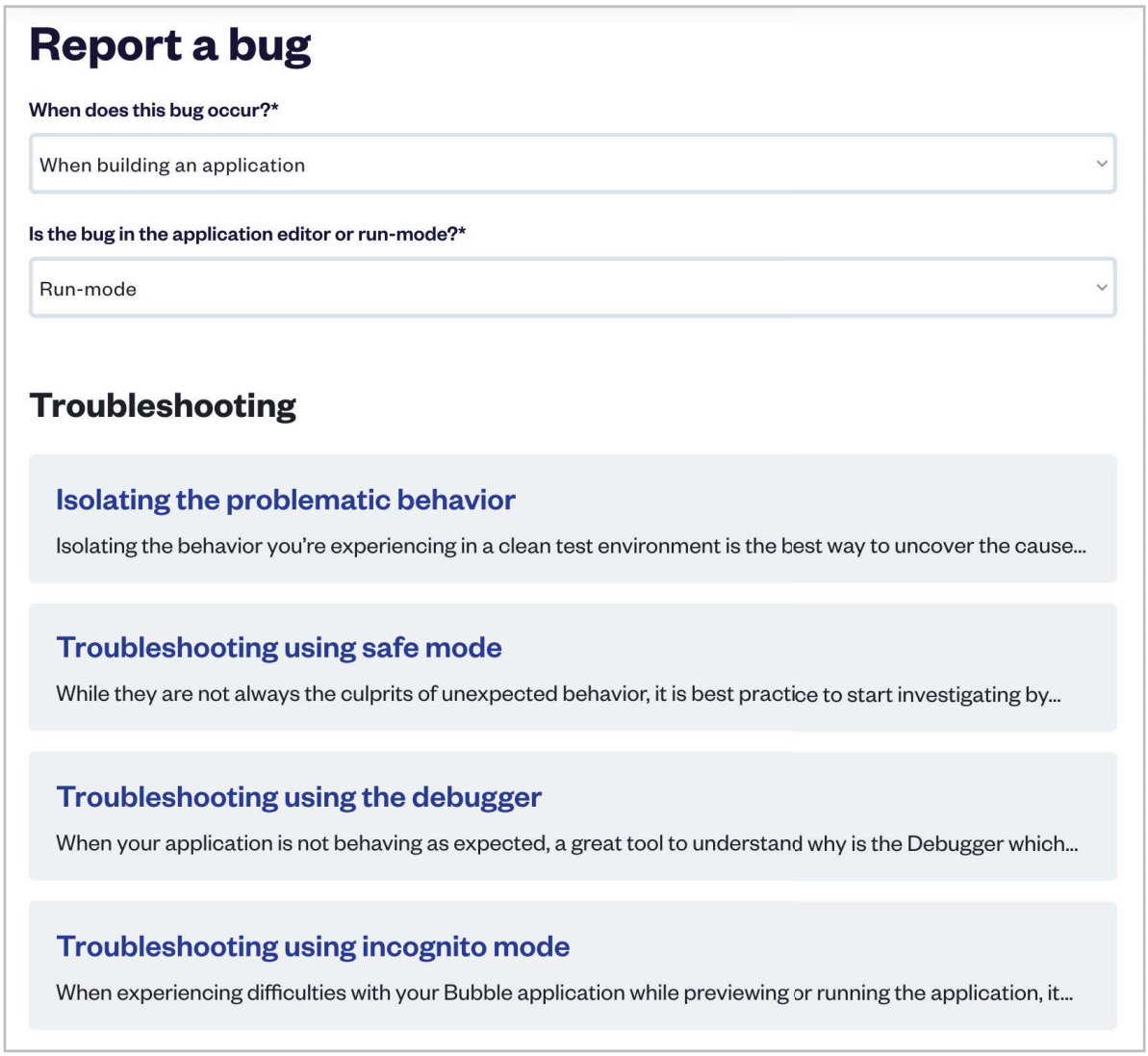
미리보기 버튼은 누르고 있으면 더 많은 미리보기 옵션이 표시됩니다. 페이지에 추가한 커스텀 HTML 없이, 커뮤니티에서 빌드 된 플러그인 없이, 혹은 둘 다 없이 실행해 볼 수 있습니다. 이러한 모드는 서비스를 디버깅할 때 유용합니다. 일부 커스텀 코드가 버블의 기본 동작과 충돌할 수 있기 때문입니다. 문제가 안전 모드에서 자동으로 해결되는 경우, 플러그인 혹은 커스텀 코드에 의해 발생한 문제라는 것을 알 수 있습니다.



5. 버블 자체 버그라고 생각될 경우

버블 개발팀은 테스트에 많은 시간을 할애하고 버그를 피하기 위해 자동화된 테스트를 광범위하게 사용하고 있습니다. 그러나 여러분이 만든 버블 앱과 워크플로우가 아니라 버블 핵심 기능의 예기치 않은 동작으로 인해 문제가 발생했다고 생각하는 경우는 버블에 연락하여 더 깊이 조사해 볼 필요가 있습니다. 참고로 버블에서 만들어지지 않은 플러그인에는 해당되지 않으니, 이런 경우 해당 플러그인을 만든 이에게 문의하는 것이 좋습니다.

버블팀에 버그를 보고하려면 버그 리포트를 사용해야 합니다. 버그 리포트를 제출하면 팀이 버그를 조사하는 데 필요한 다양한 정보를 입력할 수 있습니다.



버그를 제출하기 전에 실제 버그인지 확인하기 위해 몇 가지 작업을 수행해주시기 바랍니다.

(1) 보고서를 제출하기 전에 연결이 완전히 작동하고 브라우저가 최신 버전인지 확인합니다.

(2) 광고 차단기 및 브라우저 확장 응용 프로그램이 방해한 것 일 수 있으므로 먼저 시크릿 모드에서 테스트 해봅니다.

(3) HTML 요소, 헤더 등에서 서비스에 추가한 커스텀 코드를 제거합니다.

버그 리포트를 작성할 때 신속한 해결을 위해 몇 가지 사항을 염두에 두시기 바랍니다.

(1) 문제를 설명할 때 가능한 한 구체적으로 설명합니다. "작동하지 않는다. 그래서 버그라고 생각한다."와 같이 말하면 구체적이지 않습니다. 대신 "어떠한 조건에 따라, 이 요소는 빨간색이어야 하지만, 녹색입니다." 라고 말하는 것이 훨씬 더 효과적입니다.

(2) 자체 디버깅 프로세스에 대해 위에서 설명한 것과 마찬가지로, 문제가 분리될수록 더 빨리 조사할 수 있습니다. 실제로 설계 및 핵심 워크플로우의 맥락을 벗어나 빈 페이지에서 문제를 재현할 수 있는 경우가 가장 좋습니다.

(3) 서비스에 대한 지식이 없는 사용자가 이해할 수 있을 정도로 설명해야 합니다. 가장 좋은 설명은 '버튼 A 클릭', '입력창에 xx입력', '버튼 B 클릭', '문제를 확인해보세요.'입니다.

(4) 양식의 메시지에 따라 스크린 샷을 공유해 보시기 바랍니다.

(5) 비디오는 유용할 수 있지만 작성된 파일을 대체해서는 안 됩니다. 즉, 옵션인 경우라고 생각하면 됩니다.

버블에서의 버그는 개발 작업을 지연시킬 수 있습니다. 하지만 올바른 버그 리포트 제출을 통해 문제를 신속하게 발견하고 해결할 수 있으며, 버블 팀이 문제를 해결하면 커뮤니티 전체에 대해 해결될 수 있습니다.

02 CHAPTER 디버거 활용 방법

실행 모드에서 서비스가 예상된 행동을 하지 않을 때, 이 이슈를 분석하고 해결하는 가장 좋은 방법은 각 워크플로우의 액션을 하나씩 실행하며 잘못된 데이터가 계산되는 위치를 찾거나 조건이 다른 방법으로 계산되는 것을 확인하는 것입니다. 바로 이것이 디버거의 목적입니다.

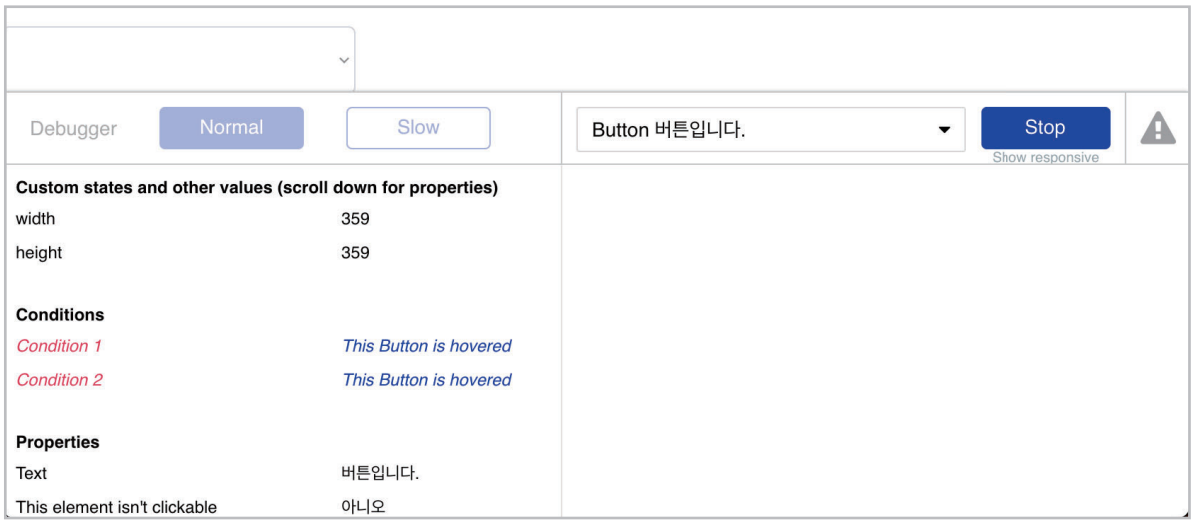
1. 디버거 활성화

디버거를 활성시키기 위해서는 URL 끝에 'debug_mode=true'를 추가하면 됩니다. 만약 URL에 다른 파라미터가 없을 경우에는 'https://app.bubbleapps.io/index?debug_mode=true' 와 같이 입력되고, 파라미터가 있을 경우에는 'https://app.bubbleapps.io/index?param=value&debug_mode=true' 와 같이 입력됩니다.

이외에도 에디터 우측 상단의 미리보기 버튼을 클릭해도 디버거가 자동으로 활성화 될 것입니다.

여러분의 서비스와 작업 내용을 보호하기 위해, 디버거는 에디터에 접근하여 수정할 수 있을 때만 디버거에 접근 가능합니다. 만약 에디터에 대한 접근 권한이 없다면 URL의 뒤에 'debug_mode=true'를 추가해도 아무런 영향이 없습니다. 추가로 현재 버블은 터치 디바이스에서는 디버거를 지원하지 않습니다. 그러니 디버거에 접근하려면 데스크톱이나 노트북을 이용하는 게 좋습니다.

디버거가 활성화 되었을 때, 페이지의 아래에 볼 수 있습니다. 이 곳에는 워크플로우 디버거와 요소 점검기가 있습니다. 디버거는 필요에 따라 아래쪽으로 자동으로 늘어나 공간을 차지하게 됩니다. 이는 디버거 모드에서만 적용되고 실제 배포된 서비스에서는 보여지지 않는 영역입니다.



2. 워크플로우 디버깅

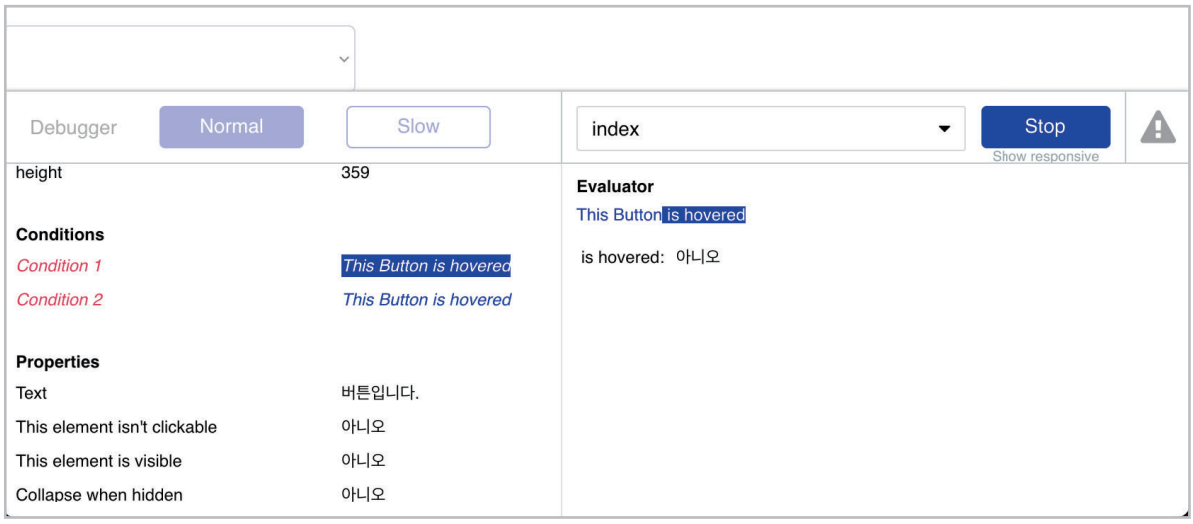
디버거의 왼쪽 편에는 워크플로우 디버거가 있습니다. 여러분은 워크플로우가 실행될 때 어떻게 디버거가 행동할지 정할 수 있는 세 개의 버튼을 볼 수 있습니다. 총 세가지 모드가 가능합니다.

(1) '보통' 모드로 워크플로우가 아무런 방해 없이 작동됩니다.

(2) '천천히' 모드로 액션들 사이마다 1초의 정지가 적용됩니다.

(3) '단계별' 모드는 워크플로우의 실행을 통제할 수 있습니다. 이 모드가 디버깅할 때 가장 자주 쓰게 될 것입니다.

워크플로우를 '느리게' 혹은 '단계별' 모드로 실행하면 워크플로우의 표현과 각 액션마다 계산된 필드 값들을 볼 수 있습니다. 파란색으로 나타나는 프로퍼티는 동적이라는 뜻입니다(에디터에 값이 나타나있음). 이 파란색 프로퍼티를 클릭하면 어떤 식으로 계산되었는지 볼 수 있습니다.



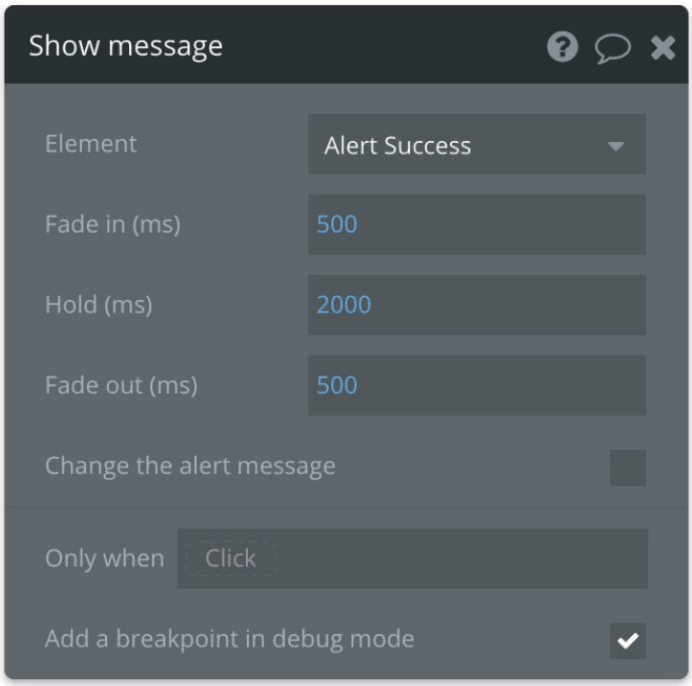
하나 이상의 워크플로우가 실행되어야 할 때는, 각각의 워크플로우가 선택된 디버그 워크플로우의 종류로 실행됩니다. 이는 커스텀 워크플로우에서도 동일하며 커스텀 워크플로우에 들어가서 완료되어 나올 때까지 계속 진행하게 됩니다.

디버거의 상태는 페이지가 새로고침 되어도 저장되어 있습니다. 만약 워크플로우가 페이지를 전환하게 되거나 페이지를 새로고침 할 경우에도 기존의 디버거의 상태로 나머지 액션들을 수행합니다.

만약 어떤 포인트에서는 워크플로우가 계속 실행되길 바란다면, '보통' 모드를 클릭하여 일반 속도로 나머지 액션이 실행되도록 하면 됩니다.

3. 브레이크 포인트 추가

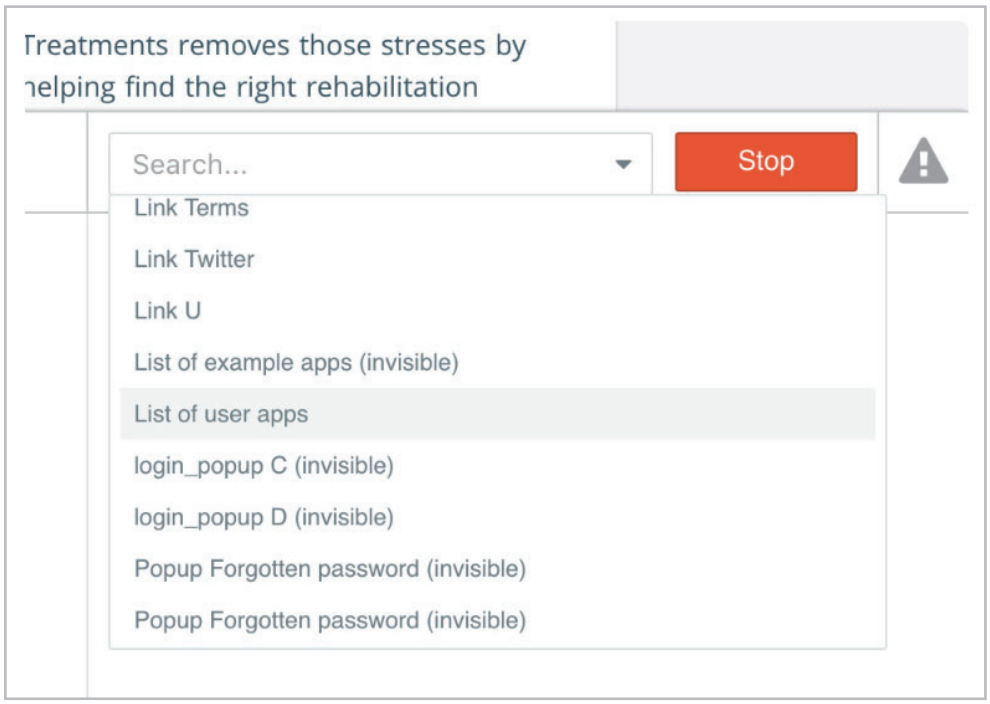
복잡한 페이지에서 작업하게 되면 많은 워크플로우가 있을 것입니다. '단계별' 디버거 모드는 너무 자주 중단되기 때문에 이상적이지는 않습니다. 만약 조사하고 싶은 특정 워크플로우, 이벤트 혹은 액션이 있을 경우에는 브레이크 포인트를 추가하면 됩니다. 그러면 디버거는 이 이벤트나 액션이 실행될 때 '단계별' 모드로 실행됩니다. 이러한 설정은 워크플로우 에디터에서 가능합니다. 참고로 이는 디버깅 모드에서만 유효하며, 실제 서비스 사용자들에게는 영향을 미치지 않습니다.



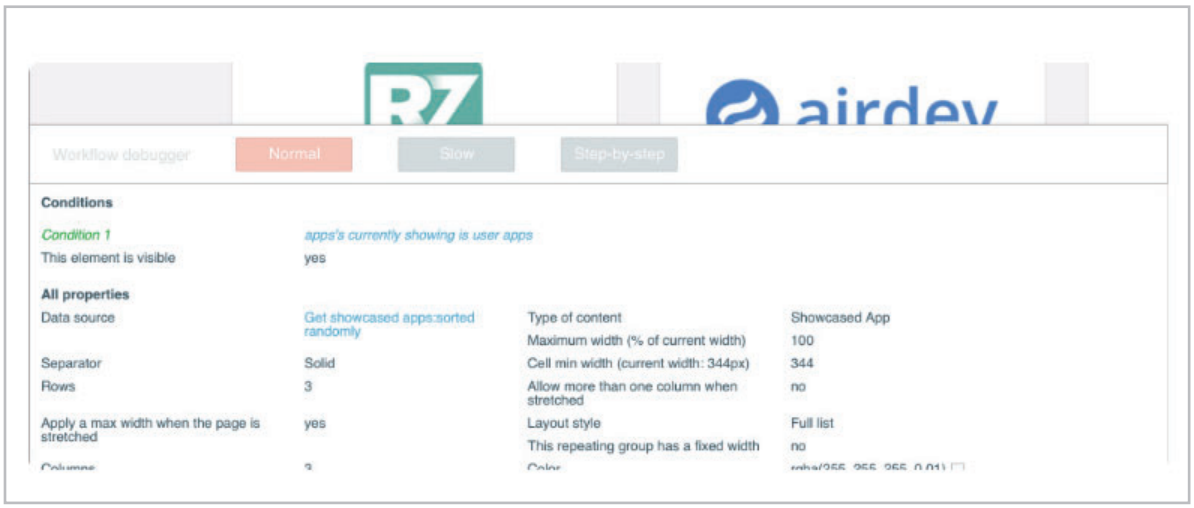
4. 요소 점검

때때로 왜 요소가 특정 방식으로 보여지는 지 알아야 합니다. 이러한 경우는 조건을 사용하거나 데이터를 보이게 할 때 적용되는 상황입니다. 디버거는 페이지에서 한 요소를 지정해 현재 상황에서의 조건과 필드의 값들을 볼 수 있게 해줍니다.

요소 인스펙터를 활성화시키기 위해 디버거 우측 하단의 '점검' 버튼을 누릅니다. 점검기가 활성화 되었으면 디버거 영역이 확장되며 점검을 원하는 요소를 선택할 수 있습니다. 원하는 요소를 클릭하거나 드롭다운 메뉴에서 찾는 방식으로 선택이 가능합니다. 이 과정에서 특정 요소가 보이는지 혹은 안 보이는지 쉽게 알 수 있습니다. 이외에도 입력창에 원하는 요소의 이름을 검색하여 선택할 수도 있습니다.



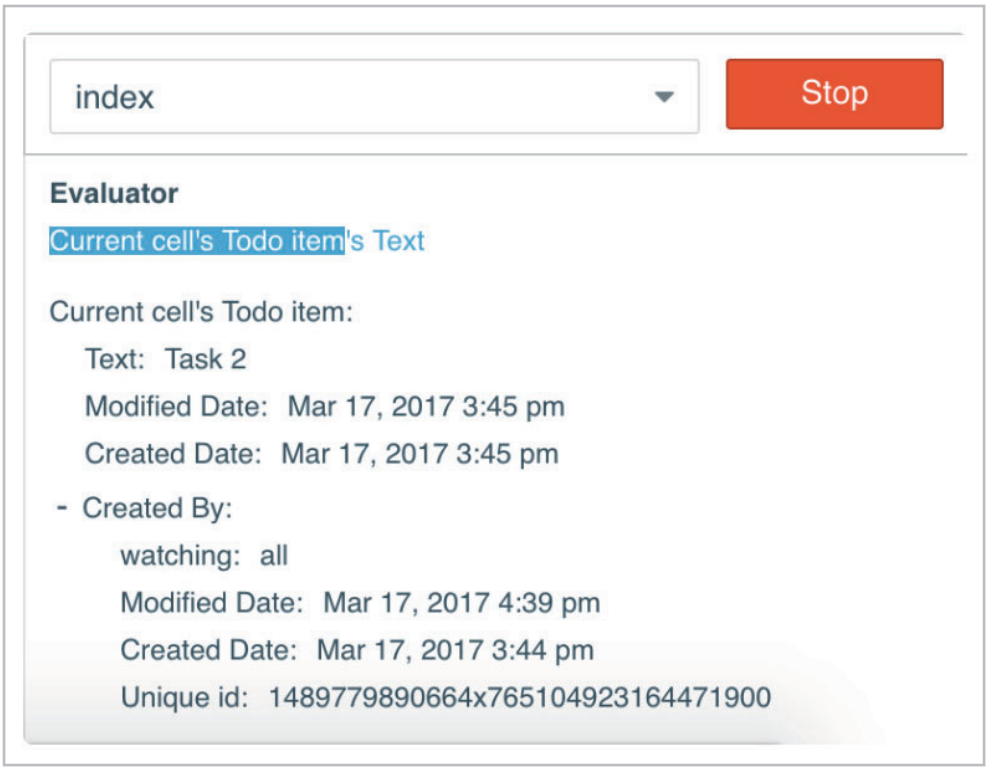
요소가 선택되었으면, 다양한 프로퍼티들과 조건, 그리고 어떻게 그들이 계산되어 졌는지가 나옵니다. 또한 인스펙터 섹션의 맨 아래에서 구성요소의 컨텍스트에 포함된 커스텀 상태 및 기타 값(예: 그룹의 내용) 목록을 볼 수 있습니다. 아래 이미지는 디버거에서 쇼케이스의 앱 목록을 보여주는 이미지입니다. 다양한 필드와 값을 볼 수 있습니다. 값이 파란색이면 동적 표현식이며, 값을 클릭하면 값을 하나씩 평가할 수 있습니다.



조건을 사용하는 경우, 디버거는 yes로 계산된 조건을 녹색으로 표시하고 그렇지 않은 조건은 빨간색으로 표시합니다. 여기서도 식을 클릭하게 되면 각 항목의 계산을 볼 수 있습니다.

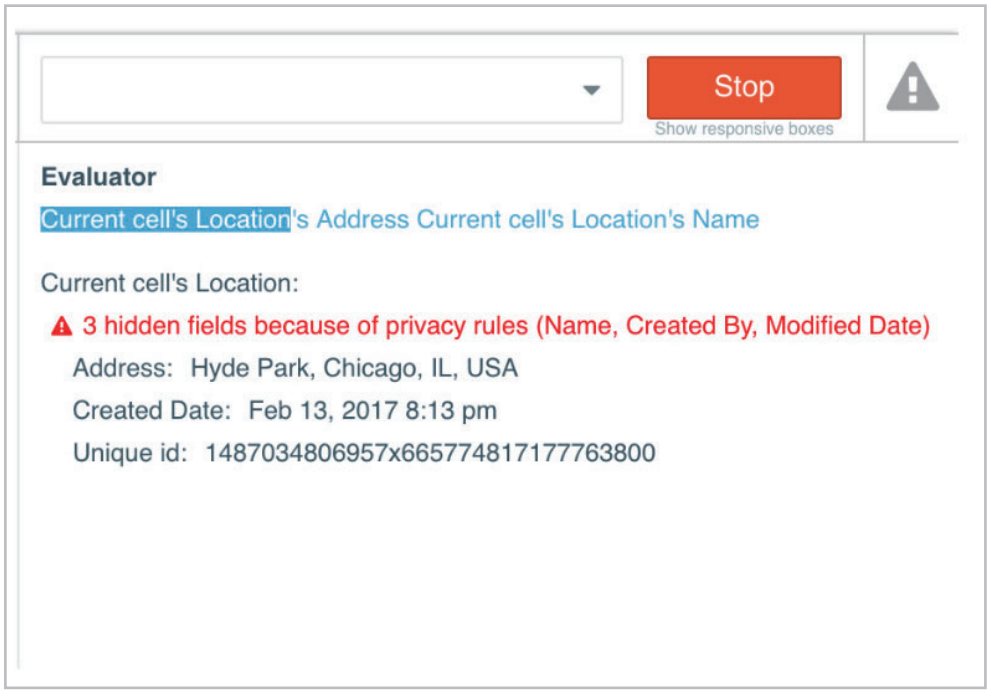
5. 표현식 값 이해

디버거를 사용하면 실행 모드에서 동적인 식을 계산하는 방법을 이해할 수 있습니다. 디버거 오른쪽에 있는 evaluator를 사용하면 식의 각 항목을 클릭할 수 있으며 값이 아래에 표시됩니다. 데이터베이스 검색 결과, API 호출 결과 등이 표시됩니다.



표현식이 다른 표현식을 매개변수로 사용하는 경우에는 각 표현식 내부에 하나씩 차례로 들어가 데이터의 출처를 분석할 수 있습니다. 식에서 일부 요소를 참조하는 경우, 식 evaluator에서 요소를 마우스로 가리키면 페이지에서 요소가 강조 표시됩니다.

디버거는 개인 정보 문제를 디버그 하는 데에도 유용합니다. 프라이버시 규칙은 일부 필드를 숨길 수 있기 때문에 복잡한 디버깅 상황을 초래할 수 있습니다. 검열된 항목에서 evaluator를 클릭하면 빨간색 언급이 표시됩니다. 프라이버시 규칙이 적용되지 않았을 때 현재 서비스 사용자에게 보이는 값과 원래의 값을 비교하는 방식입니다. 두 개가 다른 경우 프라이버시 규칙이 적용되고 설명이 표시됩니다.



6. 오류 실행

디버거는 실행 모드에서 실행 오류를 볼 수 있는 곳이기도 합니다. 예를 들어 서비스 API를 호출할 때 해당 서비스에서 매개 변수가 누락되어 오류를 반환하는 경우 오류가 표시됩니다. 워크플로우 또는 요소에서 오류가 발생하면 아이콘이 빨간색으로 바뀌고 클릭할 수 있게 됩니다. 이 아이콘을 클릭하면 오류 목록이 표시됩니다.

특히 플러그인을 통해 외부 서비스를 사용할 경우 문제가 발생할 때 가장 먼저 해야 할 일 중 하나는 실행 오류가 발생하고 있는지 확인하는 것입니다.

03 CHAPTER 서버 로그

디버거를 사용하면 현재 상황을 테스트하고 디버그 할 수 있지만 서버 로그를 사용하면 과거의 문제에 대해 탐색할 수 있습니다. 이것이 서버 로그 기능의 목적입니다. 서버에서 발생한 워크플로우 실행의 과거 실행을 서버 로그에서 볼 수 있습니다.

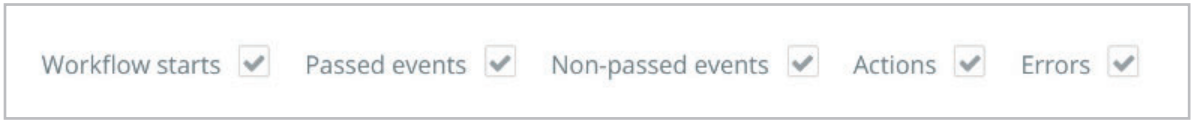
1. 로그 검색

로그 탭의 서버 로그 섹션에는 사용자가 앱과 상호 작용할 때 실행되는 이메일 보내기 혹은 데이터 변경과 같은 서버 측 작업 로그를 검색할 수 있습니다. 특정 날짜 또는 특정 키워드 내에서 이메일 또는 ID로 특정 사용자를 검색합니다.

서버 로그는 버전에 따라 다르므로 문제가 보고된 버전(라이브 vs. 개발)에 집중해야 합니다.

로그를 검색하려면 검색의 시작 날짜와 종료 날짜를 정의해야 합니다. 로그 검색에는 시간이 걸릴 수 있으며, 에디터는 로그가 검색 완료되는 대로 항목을 표시합니다. 이미 일부 결과를 보고 아래로 스크롤하면 더 많은 항목을 가져옵니다(검색 버튼의 캡션은 상황에 따라 변경됩니다).

버블 앱의 트래픽이 많으면 로그가 많이 표시됩니다. 일부 검색 기준을 사용하여 검색 범위를 좁히는 것이 유용합니다. 로그를 검색할 때 가장 일반적으로 사용할 수 있는 기준은 관심 있는 이벤트 유형을 선택하는 것입니다.



워크플로우 시작

조건이 충족되고 워크플로우가 실행되는지 여부에 관계없이 서버에서 실행되는 모든 시작된 워크플로우를 표시합니다.

통과된 이벤트

조건이 '예'로 계산된 후 실행되는 모든 워크플로우를 표시합니다.

전달되지 않은 이벤트

조건이 아니요로 평가된 후 실행되지 않은 모든 워크플로우를 표시합니다. 발생하지 않은 항목을 디버깅할 때 유용합니다.

액션

액션만 표시하고 이 액션의 실행으로 이어진 이벤트는 표시하지 않습니다.

오류

워크플로우를 실행할 때 발생한 서버 측 오류를 보여줍니다. 예를 들어 신용카드 오류 또는 이메일 보내기 오류가 있습니다. 이 기능은 문제를 진단하는 데 특히 유용합니다.

[팁]

프로그램에 많은 액션이 있는 경우 로그가 많을 수 있습니다. 로그를 검색할 때 쿼리 속도가 느리거나 시간이 초과되면 검색 시간대를 좁혀 보기 바랍니다.

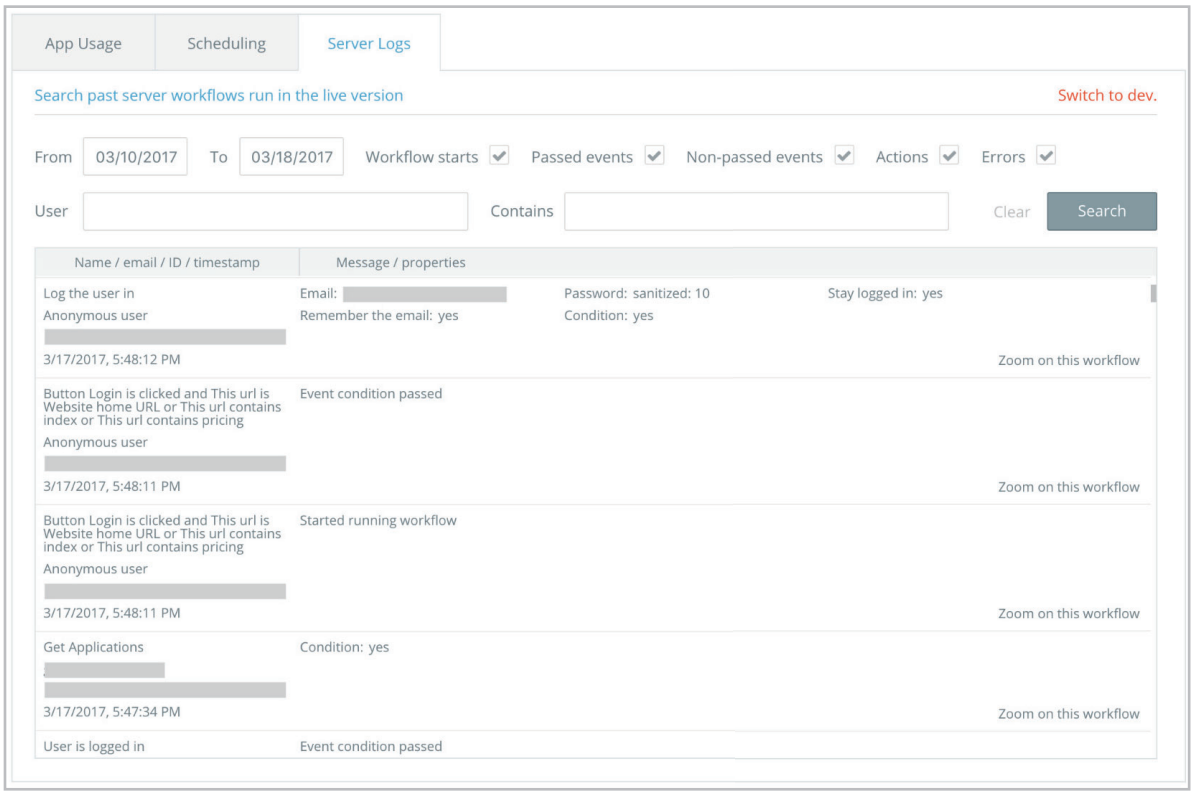
로그를 검색할 때 문제가 있는 워크플로우에 대한 자세한 정보가 있으면 특정 사용자 및 용어를 검색하여 검색 범위를 더욱 좁힐 수 있습니다. 첫 번째 입력을 통해 사용자의 이메일 또는 사용자의 고유의 ID를 입력할 수 있습니다. 이 값이 채워지면 이 사용자가 시작한 워크플로우만 반환됩니다.

마지막 상자에서 검색할 문자열을 입력할 수 있습니다. 작업에 일부 텍스트로 평가된 속성이 있고 텍스트를 검색하면 워크플로우가 검색됩니다. 예를 들어 'Boston' 텍스트가 포함된 이메일이 전송된 것을 알고 있는 경우 'Boston'을 검색하면 이메일 보내기 액션이 반환됩니다.

2. 결과 분석

결과가 검색되면 결과 영역에 최근 이벤트부터 시작하여 표시됩니다. 각 항목에 대해 액션/이벤트 이름, 이메일 및 사용자의 ID(사용자가 등록되지 않은 경우 이메일은 '익명 사용자'가 됨)가 표시되고 오른쪽 부분에는 이 작업, 이벤트 또는 오류 메시지에 대한 속성의 결과가 표시됩니다.

액션 이름(또는 이벤트)을 클릭하면 에디터에서 이 작업이 정의된 위치도 이동합니다. 'Zoom on this workflow'를 클릭하면 현재 워크플로우에 대한 이벤트, 액션 및 오류만 표시할 수 있습니다. 일련의 액션을 보는 것은 무슨 일이 일어났는지 이해하고 패턴을 식별하는 좋은 방법이 될 수 있습니다.



PART 09

서비스 유지보수

01 CHAPTER 테스팅 버전 관리

서비스 유지보수 챕터에서는 확장에 따라 애플리케이션을 유지 관리할 수 있는 다양한 개념과 기능에 대해 배울 수 있습니다. 여기에는 백업, 대량 작업, 협업 등이 포함됩니다.

모든 Bubble 앱에는 라이브(배포된) 버전과 변경할 수 있는 개발 버전으로 최소 두 개의 변경할 수 있는 버전이 있습니다. 기본적으로 모든 앱에는 'test'라는 단일 개발 버전이 제공되지만, Professional, Production 또는 Dedicated 플랜을 사용하는 경우에는 더 많은 버전을 만들 수 있습니다. 이러한 플랜에서 개발 버전을 삭제하더라도 라이브 앱에서는 어떠한 영향도 미치지 않습니다.

버전 내에서 버블을 사용하면 문제가 발생할 경우 제작의 변경사항을 되돌릴 수 있습니다. 저장 지점을 지정해놓으면 특정 지점으로 쉽게 되돌릴 수 있지만 변경 사항이 지속적으로 저장되므로 특정 시간으로 되돌릴 수도 있습니다. 얼마 만큼의 시간까지 되돌릴 수 있는지는 플랜에 따라 결정됩니다.

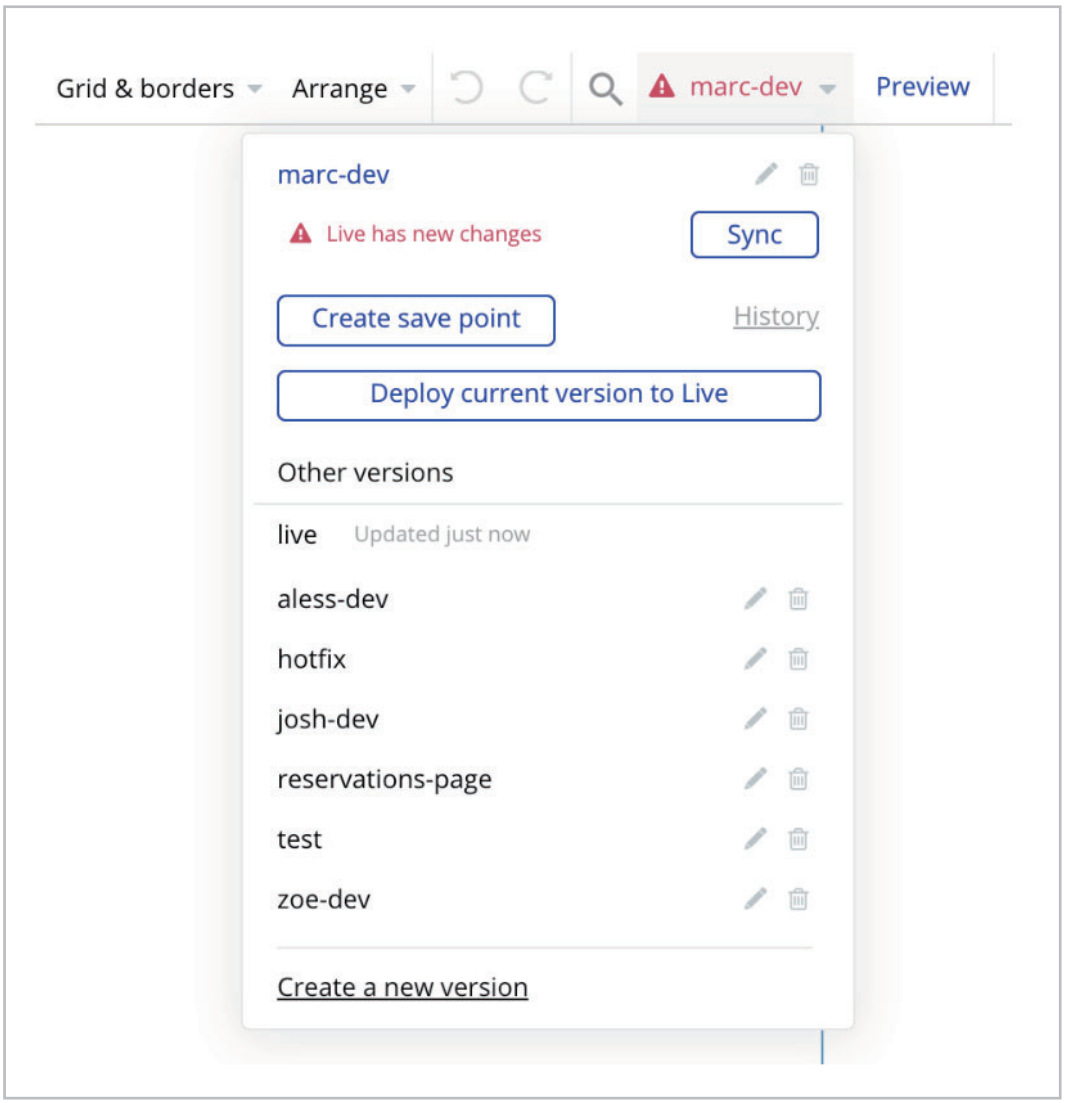
1. 버전

버블 앱에는 최소 라이브와 테스트 두 가지 이상의 버전이 있습니다. 라이브는 배포하여 공개적으로 접근할 수 있는 앱 버전이고 테스트 버전은 서비스를 편집할 수 있는 버전입니다. 배포 준비가 되었다면, 테스트 버전에서 실시간 변경 내용을 라이브 버전으로 배포할 수 있습니다. 팀, 프로덕션, 전용 플랜을 사용하는 분들을 여러 개발 버전을 추가로 가질 수 있습니다.

여러 개발 버전으로 서비스를 구축할 때는 특정 버전에서 라이브 버전으로 배포할 수 있습니다. 또한 각 버전에는 자체 저장 지점과 복구 기록이 있습니다. 모든 개발 버전의 변경 내용을 현재 개발 버전에 추가할 수 있습니다.

2. 버전 콘트롤

버전 콘트롤과 관련된 기능은 미리보기 버튼의 바로 왼쪽에 있는 버전 콘트롤 드롭다운 메뉴에서 확인할 수 있습니다.

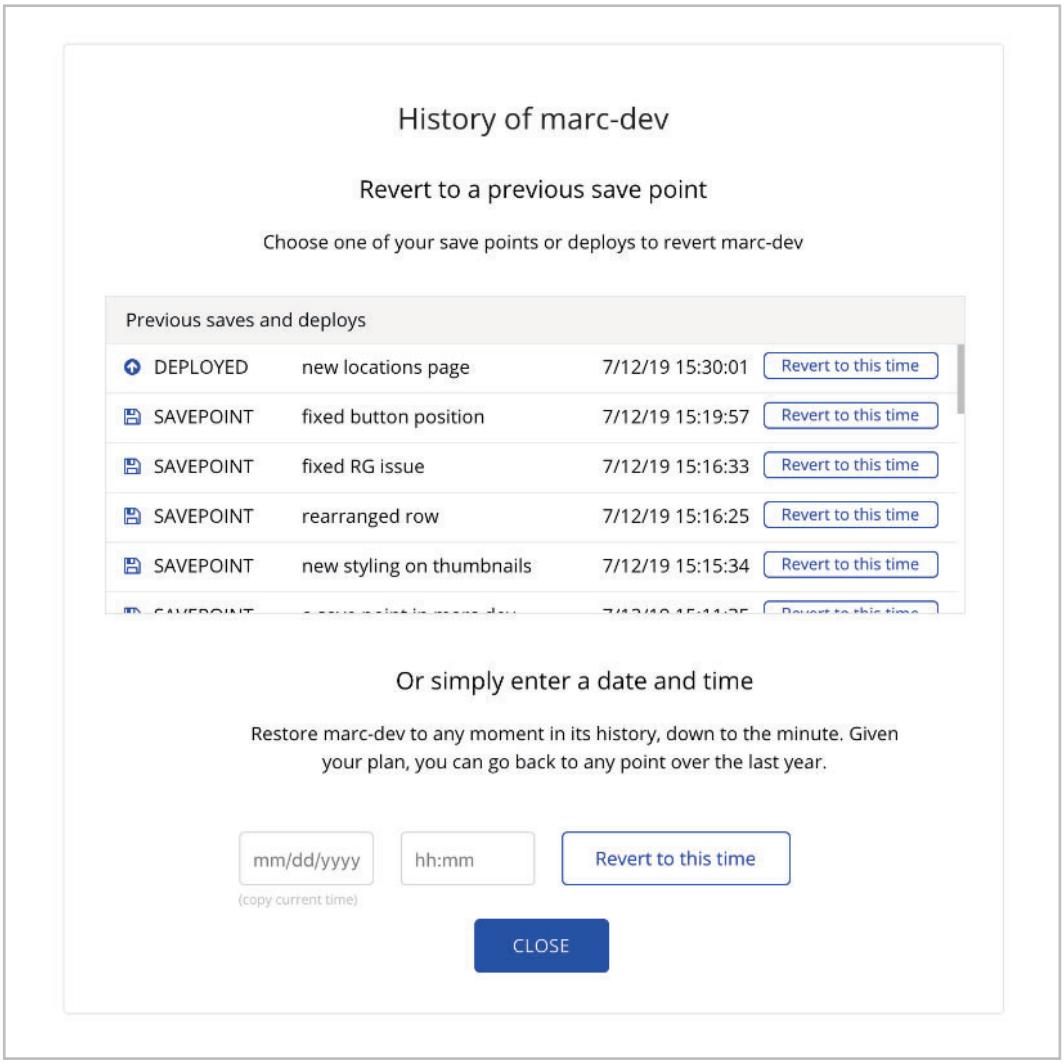


현재 활성 분기의 이름이 메뉴 상단뿐만 아니라 도구 모음에도 표시됩니다. 아래 메뉴에서 다른 버전을 선택하거나 "새 버전 만들기"를 클릭하여 새 버전을 만들 수 있습니다. 메뉴의 맨 위 부분에서는 현재 버전과 관련된 작업을 수행할 수 있습니다. 라이브로 배포하거나, 저장 지점을 만들거나, 이전 시간 또는 저장 지점으로 되돌리거나("히스토리" 클릭), 최근 변경 내용을 동기화하여 라이브에서 사용할 수 있습니다.

참고

특히 큰 서비스에서 버전 간의 차이가 큰 두 버전을 병합하게 되면, 버블이 백그라운드에서 많은 계산을 하기 때문에 병합 후 에디터가 눈에 띄게 느려집니다. 이는 잠시 기다리면 저절로 해결되는 문제입니다.

3. 히스토리



히스토리 버튼을 클릭하면 현재 버전을 이전 시점으로 되돌릴 수 있는 팝업이 표시됩니다. 버블 앱의 변경 사항은 무한 실행 취소 방식으로 세분화된 방식으로 저장됩니다. 따라서 버블의 버전을 임의의 시점으로 되돌릴 수 있게 되는 것입니다. 또한 앱을 특정 상태로 쉽게 되돌릴 수 있도록 버전 드롭다운에서 직접 저장 시점을 생성할 수 있습니다. 이 저장 시점을 주석, 특정 시간 및 날짜처럼 편의상 적용할 수 있는 라벨로 사용할 수 있습니다.

4. 라이브 버전과 동기화

다중 버전의 앱을 사용자가 특정 버전을 라이브로 배포할 때 다른 버전은 라이브 버전과 동기화 되지 않으므로 동기화 과정이 필요합니다. 라이브와 버전을 동기화하면 새로 배포된 변경 사항이 통합되어 배포가 가능합니다. 배포된 변경 사항 중 하나가 직접 충돌하는 경우 사용자에게 두 가지 옵션이 모두 표시되고 변경 사항을 선택하라는 메시지가 표시됩니다. 앱이 동기화되면 배포가 가능해 집니다.

개발 버전은 사용자가 작업하는 버전이고 라이브 버전은 읽기 전용이며 실제 사용자들과의 상호작용을 보여주는 버전입니다. 서비스를 개발 모드로 복원하고 운영 환경에 이전 버전을 배포해야 하는 경우에는 배포(하여) 실행하면 됩니다.

5. 버전 콘트롤 실습

(1) Merge 병합

병합은 하나의 버전에서 다른 버전으로 변경 내용을 추가하는 경우입니다. 예를 들어 A버전에서 서비스를 업데이트했다고 가정해 봅시다. 현재 개발 버전에 이러한 변경 사항을 포함하고 싶다면 A버전을 현재 개발 버전에 병합하면 됩니다.

(2) Conflict 충돌

충돌은 병합과 관련된 두 분기에서 동일한 요소 또는 워크플로우가 서로 다른 방식으로 변경된 경우에 발생합니다. 예를 들어 버전 A에 파란색 로그인 버튼이 있고 버전 A에서 버전 B를 만든다고 가정해 봅시다. 그런 다음 이 로그인 버튼의 색상을 버전 A에서는 빨간색으로, 버전 B에서는 녹색으로 변경합니다. 만약 버전 A를 버전 B에 병합하려고 한다면, 버블은 버튼이 어떤 색이어야 하는지 모르기 때문에 충돌을 일으킵니다. 충돌을 해결하려면 버전 A에서 빨간색으로 변경할지 아니면 버전 B에서 녹색으로 변경할지 선택해주어야 합니다.

(3) 라이브 배포

버전의 변경 사항을 웹에 게시하는 프로세스입니다. 사용 중인 버전에 없는 변경 사항이 라이브에 있는 경우 라이브와 동기화되지 않은 버전에 대해 경고를 줍니다. 이 문제를 해결하려면 라이브 버전에 병합하여 사용하면 됩니다.

(4) 개발 버전

개발 버전은 버블 앱의 '핵심 환경'입니다. 여기서 서비스를 변경하고 변경 내용을 미리 볼 수 있습니다. 앱 URL에 /version-test가 표시되므로 앱을 미리 볼 때 개발 중임을 알 수 있습니다. 개발은 기본 '버전'이기도 합니다.

(5) 라이브 버전

라이브 버전 또한 버블 앱의 '핵심 환경'입니다. 이 읽기 전용 버전은 웹에서 보여지는 그대로 나타납니다. 따라서 서비스를 실시간으로 직접 변경할 수 없고 개발 환경에서 실시간으로 변경 내용을 배포하여 변경 사항을 적용이 가능합니다.

(6) 커스텀 버전

커스텀 버전은 앱의 'branch 가지/분기' 혹은 복사본입니다. 이 버전을 사용하면 다른 버전의 서비스와 분리하여 서비스를 개발할 수 있습니다. 한 버전에서 변경한 내용이 만족스러울 경우 이 버전을 개발 버전에 병합하여 실시간으로 배포할 수 있습니다.

6. 추천 프로세스

일반적으로 개발 버전을 서비스의 '메인' 버전으로 유지하기를 강력하게 추천합니다. 개발 버전은 다른 버전을 병합하고 다른 버전을 생성하는 분기여야 하며, 라이브에 배포되는 유일한 버전이어야 합니다.

새 기능을 생성하거나 빠른 수정을 배포하려면 아래의 방법을 추천합니다.

(1) 개발 버전에서 작업 중인 기능의 이름을 다른 새 버전으로 만듭니다. 이 작업은 버전 콘트롤 드롭다운에서 'create a new version'을 사용하여 진행합니다.
(2) 새로운 버전에서 작업을 진행하고 준비가 되면 개발 버전에 병합합니다.
(3) 병합된 버전이 잘 실행되고 성공적으로 추가되었는지 확인합니다. 모든 페이지나 탭을 접속해서 확인해 보는 방법이 가장 정확한 확인 방법입니다.
(4) 개발 버전을 라이브 버전에 배포합니다.
(5) 만들었던 기능 버전을 삭제합니다.
(6) 다음 기능 작업을 위해 업데이트 된 개발 버전에서 새 버전을 다시 만들어 작업을 이어 나갑니다.

버그 수정을 빠르게 배포하여야 하지만 아직 배포할 준비가 되지 않은 개발 작업이 있는 경우에는 아래의 방법을 이용합니다.

(1) 현재의 라이브 버전을 수정할 새 버전을 만듭니다. 이 작업은 라이브 버전의 버전 콘트롤 드롭다운에서 'create a new version'을 사용하여 진행합니다.
(2) 새로 생성된 버전에서 작업합니다.
(3) 배포가 준비가 되었으면 다시 페이지나 탭을 확인하여 문제없이 잘 변경되었나 확인합니다.

(4) 이 작업 버전을 라이브에 바로 배포합니다.

(5) 배포된 라이브 버전이 성공적으로 작동하는지 확인합니다.

(6) 이 버전을 개발 버전에 병합하여 향후 개발 버전이 라이브 버전으로 배포될 때, 버그 수정내용이 함께 포함되도록 합니다.

(7) 버그 수정을 위해 만들었던 개발 버전을 삭제합니다.

참고

각 버전이 개발 버전에 병합될 때마다 다른 모든 활성 버전을 개발 버전과 동기화하는 것이 좋습니다. 이렇게 하려면 활성 버전으로 이동한 뒤 개발을 현재 버전에 병합하도록 합니다. 이 작업을 통해 활성 버전을 개발 버전에 병합할 때 충돌을 줄여줄 수 있습니다.

일반적으로 더 많은 변경사항이 이루어짐에 따라 서로 다른 버전이 병렬적으로 존재할수록 상황이 더 복잡해지기 때문에 개발자별 버전보다는 기능별 버전을 사용하는 것이 좋습니다.

위와 비슷한 이유로 개발 버전 및 라이브 버전을 제외한 다른 버전의 수명은 가능한 한 짧게 유지하는 것이 좋습니다.

또한 한 버전에서 수정 또는 기능을 작업하는 경우 다른 버전에서 동일한 항목(요소, 워크플로우, 페이지 등)을 편집하지 않는 것이 좋습니다. 이렇게 하면 두 버전을 병합할 때 발생하는 충돌 수를 줄이는 데 도움이 됩니다.

7. 개발 버전에서의 데이터베이스

모든 개발 버전은 공통의 '테스트' 데이터베이스를 공유하지만, 사용자가 만든 커스텀 타입과 관련된 데이터는 특정 버전과 연결된 상태로 남아 있습니다. 이것은 사용자 B가 참조하는 동일한 데이터베이스에 '사용자 A유형'의 레코드가 존재하지만, 사용자 B가 동기화할 때까지 사용자 B의 버전에 나타나지 않는다는 것을 의미합니다.

위에서 언급한 바와 같이 모든 앱에는 라이브 버전과 적어도 하나의 개발 버전이 있습니다. 라이브 데이터베이스의 레코드는 개발 데이터베이스의 레코드와 독립적입니다. 라이브 데이터베이스를 복사하여 개발 데이터베이스를 대체하거나 그 반대로 대체할 수 있지만, 라이브 버전에서 등록하는 사용자는 개발 버전에 자동으로 레코드를 생성하지 않습니다.

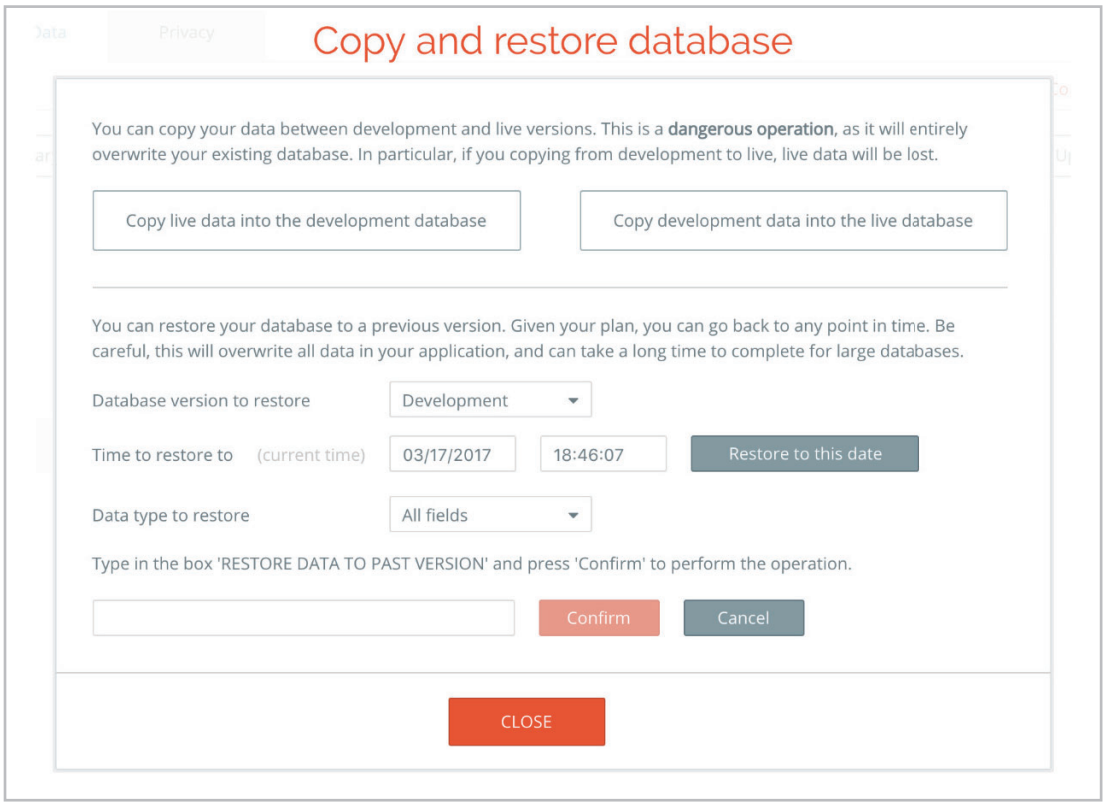
02 CHAPTER 데이터베이스 복원 & 백업

버블은 데이터 손실을 방지하기 위해 몇 가지 중복된 방법으로 서비스의 데이터를 백업합니다. 얼마나 이전 시점까지 돌아갈 수 있는지는 선택한 플랜에 따라 다릅니다. 데이터베이스는 사용자가 일부 워크플로우 또는 데이터 탭을 통해 만든 모든 '레코드' 입니다. 유지관리 측면에서 데이터베이스로 두 가지 작업을 수행할 수 있습니다. 그것은 이전 시점으로 복원하거나 여러 버전으로 데이터를 복사하는 방법입니다.

1. 데이터베이스 복원

워크플로우가 잘못 설정되어 일부 항목을 삭제하는 등 문제가 발생했을 경우 데이터를 이전 시점으로 복원할 수 있습니다. 이것은 위험한 작업이므로 절대적으로 필요한 경우에만 수행하는 것이 좋습니다. 데이터베이스를 오늘부터 일주일로 되돌리면 지난 한 주 동안 서비스에 가입한 모든 사용자가 계정을 잃어 로그인할 수 없습니다.

데이터를 복원하는 것은 데이터 탭의 App Data 섹션에 있습니다.

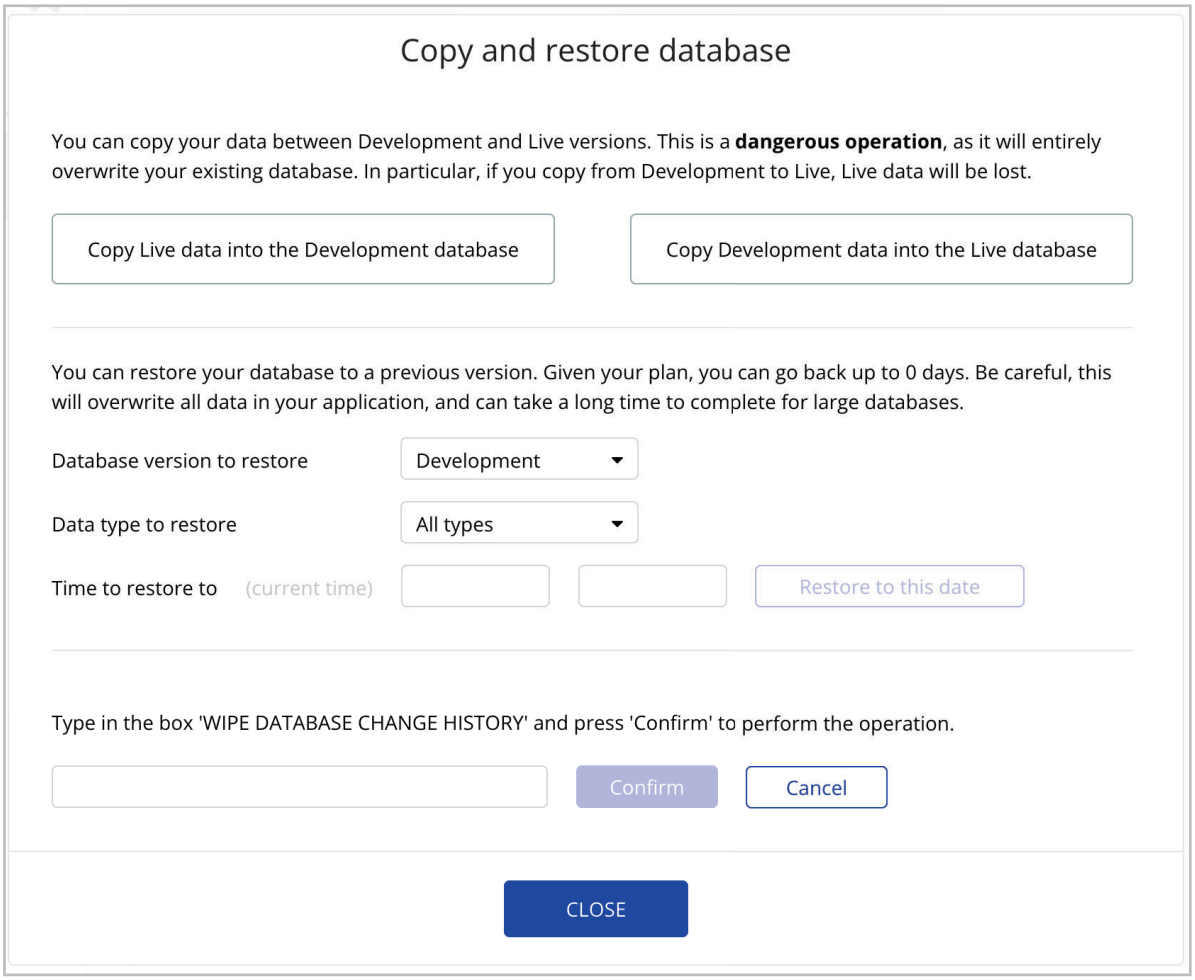


먼저 복원할 버전을 선택한 다음 시간을 지정하고 확인합니다. 한 가지 타입만 복원하도록 결정할 수 있지만 다른 타입이 관련되어 있으면 데이터 불일치가 발생할 수 있으므로 주의하여 복원해야 합니다. 서비스의 버전 콘트롤과 마찬가지로 이전 복구 전으로도 다시 복구하여 예상 결과대로 진행되지 않은 복원을 다시 되돌릴 수도 있습니다.

데이터베이스가 큰 경우에는 복원 작업을 실행하는 데 몇 분 정도 걸릴 수 있습니다. 복원을 시작하면 진행률 표시줄이 나타나며 팝업을 닫고 계속 작업할 수 있습니다. 복원을 시작한 후 에디터를 새로 고치거나 닫는 것도 가능합니다.

2. 버전 간의 데이터베이스 복사

이 옵션을 사용하면 전체 데이터베이스 또는 하나의 타입만 개발 버전에서 라이브 또는 그 반대로 적용할 수 있습니다. 이 옵션은 일반적으로 서비스를 라이브 버전으로 실행하고 데이터베이스를 먼저 개발하거나 라이브 데이터로 최신 버전의 테스트(또는 디버깅)할 때 사용됩니다.



03 CHAPTER 대량 실행

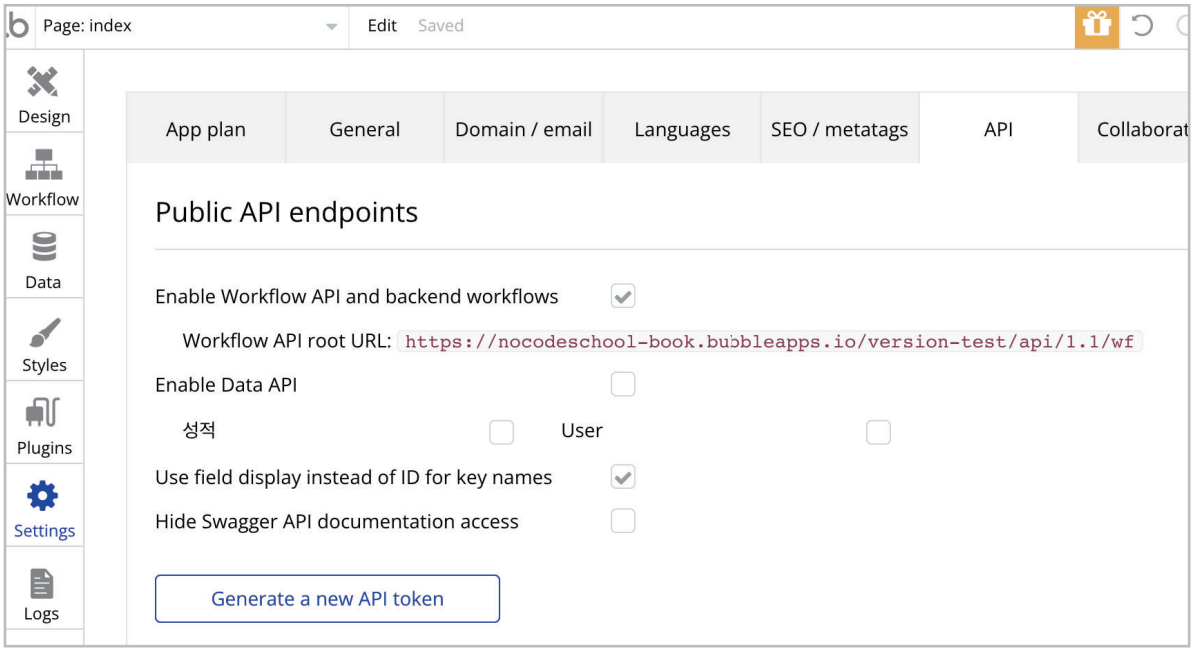
대량 작업을 사용하면 에디터에서 데이터 청크를 한 번에 수정할 수 있습니다. 이 작업은 백업을 통해 실행 취소할 수 있지만 위험한 작업이므로 배포된 데이터를 한 번에 수정해야 하는 경우에만 수행해야 합니다.

주의!

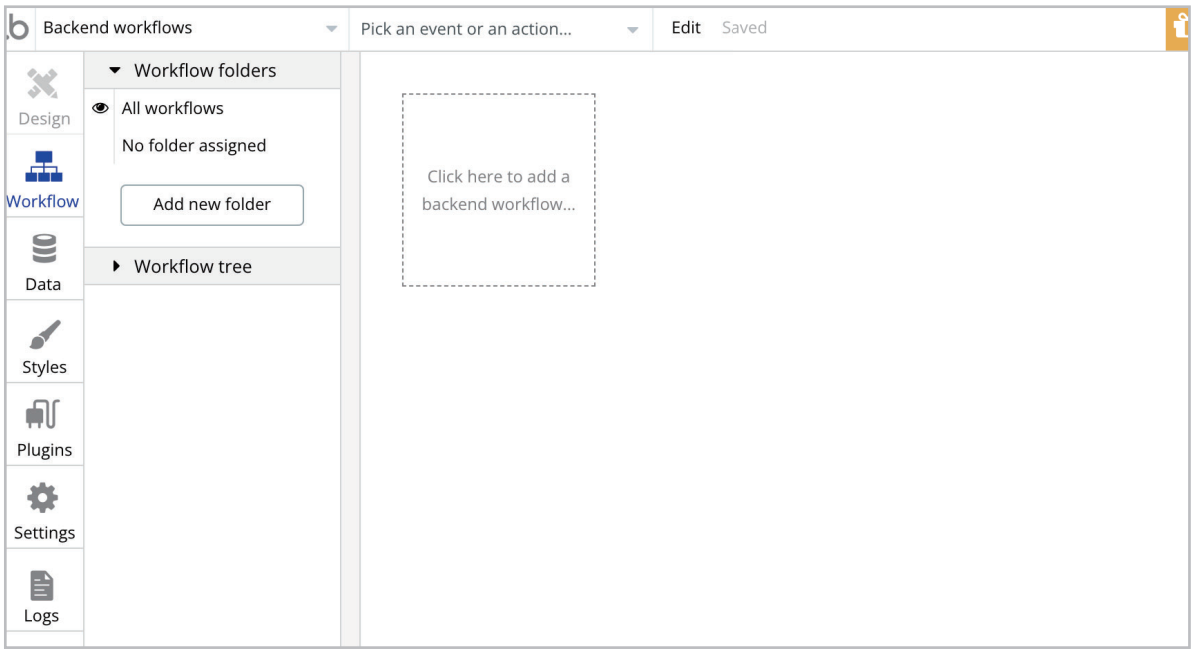
대량 작업은 클라이언트 측에서 실행되므로 브라우저 창이 열려 있고 활성 상태로 유지되는지에 따라 달라집니다. 즉, 계속해서 작업을 진행하려면 팝업을 열어 두어야 합니다. 대량의 데이터를 수정하려는 경우 반복 예약 워크플로우를 대신 사용하는 것이 가장 효과적이고 안정적인 솔루션입니다.

1. API 워크플로우 정의

데이터에 대한 대량 작업을 수행하려면 에디터의 데이터 탭에서 호출할 API 워크플로우를 정의해야 합니다. 일단 Settings > API에서 백엔드 워크플로우를 활성화한 뒤 버블 앱의 "Backend workflows" 페이지로 이동합니다.



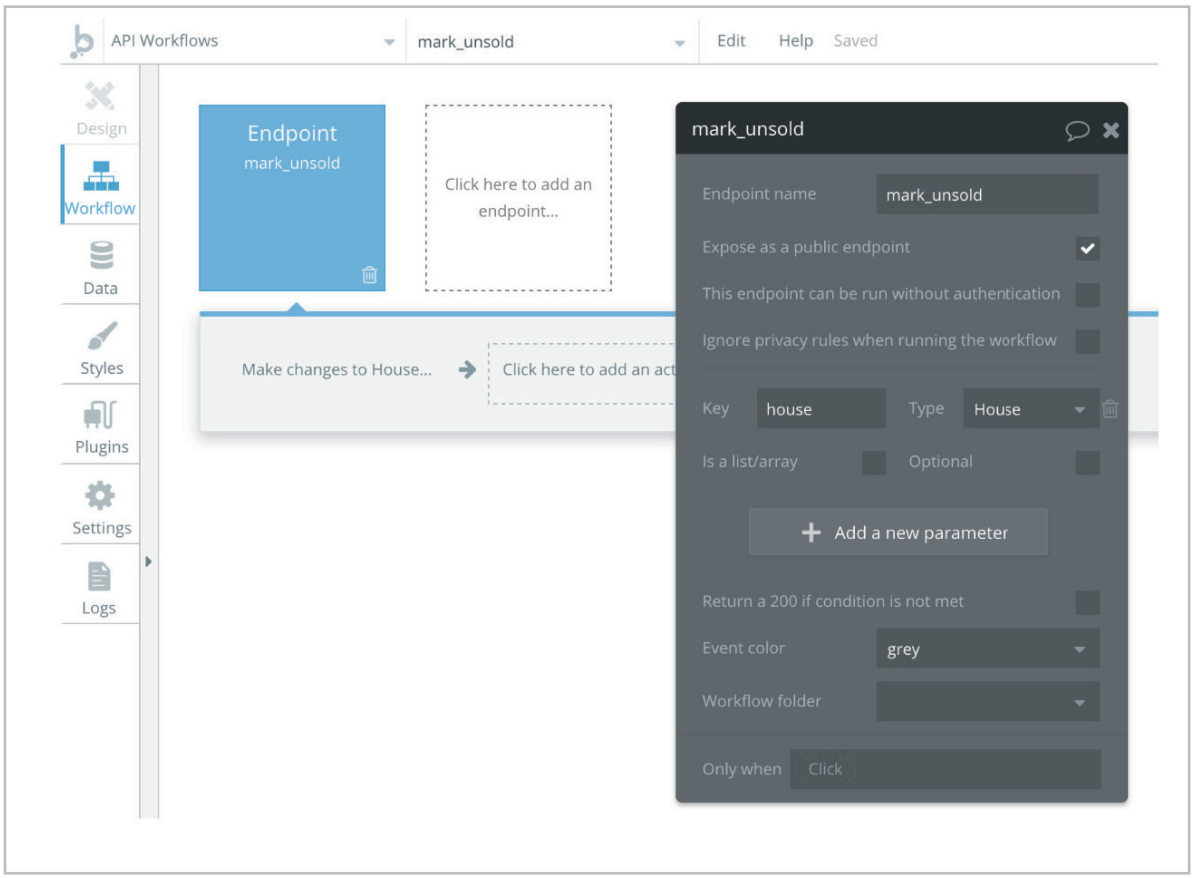
이제 하나의 매개변수를 사용하여 일부 작업을 수행하는 API 워크플로우를 생성하는 것으로 API 워크플로우의 정의가 시작됩니다.



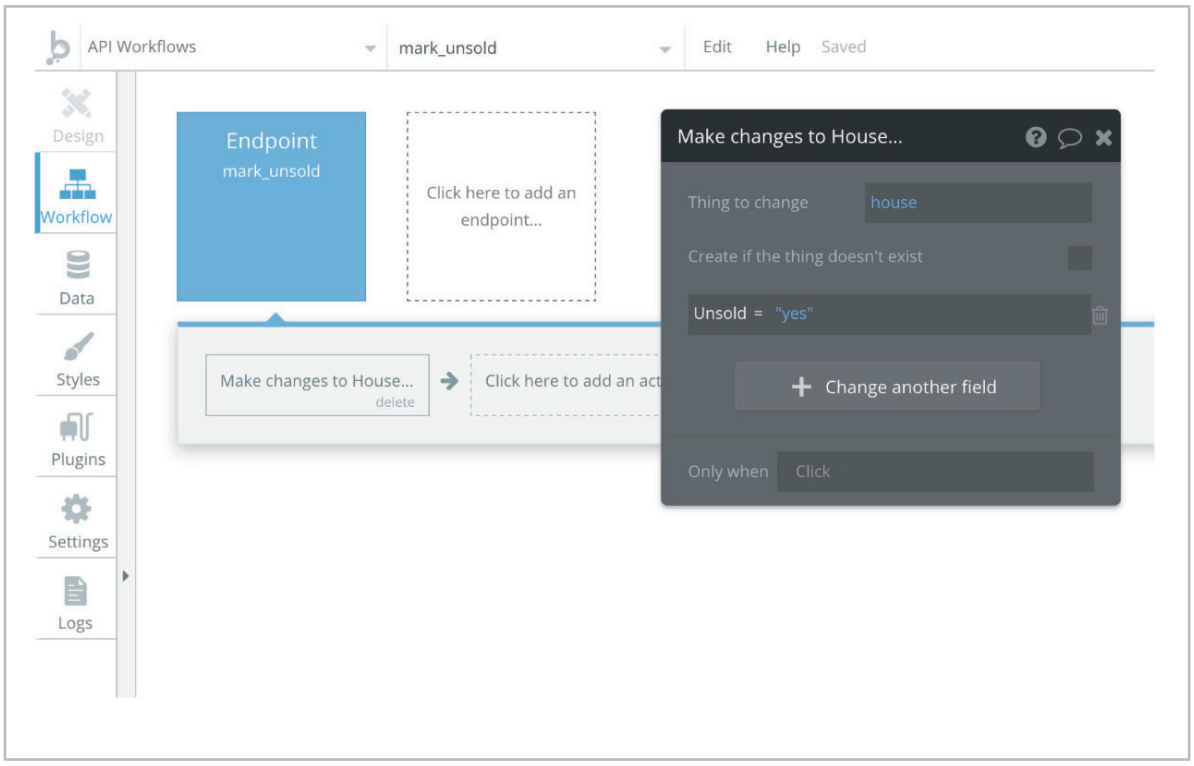
이 API 워크플로우를 내부적으로 사용하려는 경우(예: 대량 작업)에는 공개할 필요가 없습니다. 매개 변수의 타입은 처리할 데이터 타입이어야 합니다. 이러한 API 워크플로우는 대량 작업이 실행 중일 때처럼 다른 매개변수를 사용해서는 안 됩니다. 즉, 이 워크플로우를 호출할 때 해당 항목을 유일한 매개변수로 사용해야 한다는 말입니다.

해당 워크플로우를 반복시킬 수도 있습니다. 반복할 레코드에 있는 하나의 아이템에 수행될 액션을 만듭니다. 데이터 탭에서 대량 작업을 시작하면 특정 레코드 배열에서 작업을 수행하게 되면, 버블은 해당 배열의 모든 아이템에 대해 개별적으로 API 워크플로우를 수행하게 됩니다.

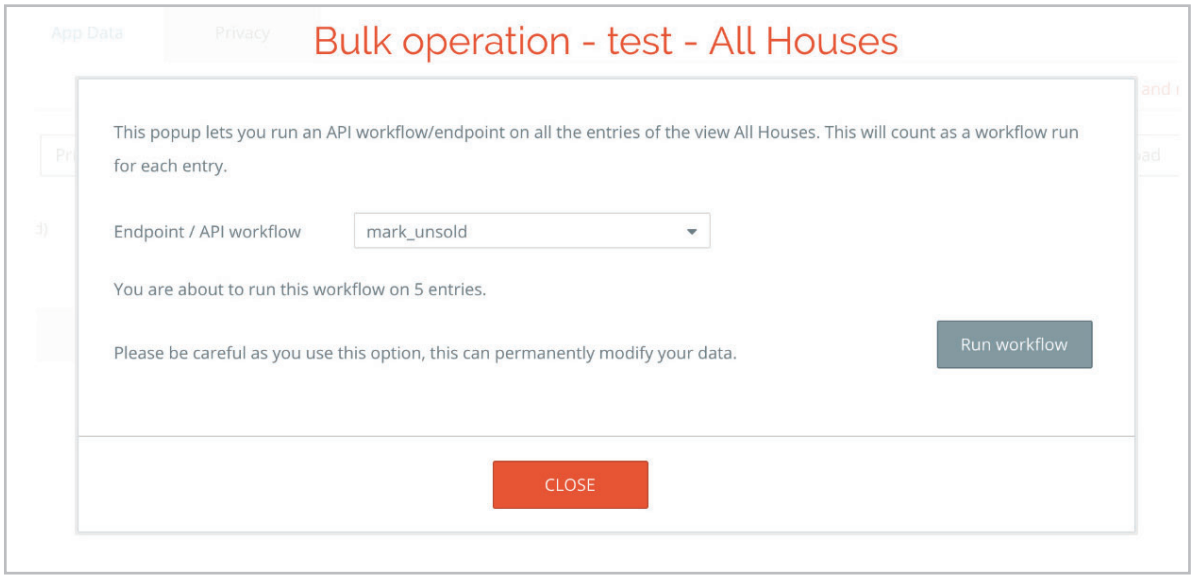
타입이 'House'이고 필드가 'unsold'인 서비스가 있다고 가정해 보겠습니다. 나중에 데이터베이스에 이미 작성된 항목을 포함하여 모든 주택에 대해 'unsold'인 필드를 '예'로 설정해야 한다는 것을 알게 되었다고 합시다. 그때 필요한 워크플로우는 다음과 같습니다.



그리고 아래에는 파라미터로 전달된 'House'가 어떤 액션을 해야 하는지 나와 있습니다.



이 파트를 마치면 데이터 탭의 App Data섹션으로 이동하여 'House'를 보여주는 뷰를 선택할 수 있습니다. 대량 작업을 클릭하면 처리할 항목과 사용할 워크플로우를 정의할 수 있는 대량 작업 팝업이 표시됩니다. 전체보기를 처리하거나 표에서 일부 항목만 선택하고 선택한 항목에 대해 워크플로우를 실행할 수 있습니다.



이 옵션은 영구적으로 데이터를 수정하므로 수천 개의 항목을 처리하는 경우 시간이 걸릴 수 있습니다.

04 CHAPTER 스케줄러

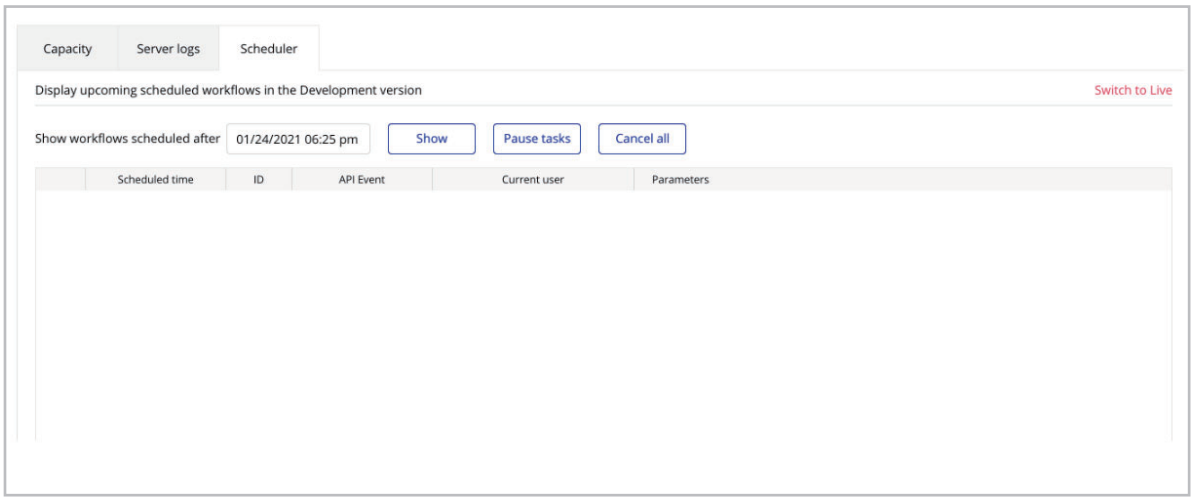
에디터의 로그 탭에 있는 스케줄러에서는 예정된 모든 워크플로우를 볼 수 있습니다.

스케줄러는 대기열 형식입니다. 쉽게 깔때기로 생각하면 됩니다. 만약 깔때기의 배수 속도보다 더 빨리 물을 채우면, 물은 깔때기를 꽉 채우게 됩니다. 마찬가지로 많은 워크플로우를 스케줄링하여 스케줄러를 채우면 이 채워지는 작업을 볼 수 있을 뿐만 아니라 지연되는 것 또한 경험할 수 있습니다.

현재 버블은 스케줄러에 부하가 걸렸을 때 서비스가 다운되는 것을 방지하기 위해 실행 속도를 CPU의 약 60%로 제한하고 있습니다. 작업이 가볍거나 설정된 시간 범위 내에 작업 일정을 적게 잡으면 처리 속도가 향상될 것입니다.

1. 조회

시간을 선택하고 “Show”버튼을 누르면 그 시간에 예정된 워크플로우들을 볼 수 있습니다.



2. 작업 중단

예정된 워크플로우를 일시 중지하려면 “Pause tasks”를 클릭하면 됩니다. 워크플로우가 실시간 또는 개발 버전에서 예상대로 실행되지 않는 경우, 워크플로우가 일시 중지되지 않았는지 확인하도록 합니다. 작업이 이미 중단된 경우 “Resume tasks”로 표시되며 이를 눌러 예약된 워크플로우를 다시 실행할 수 있습니다.

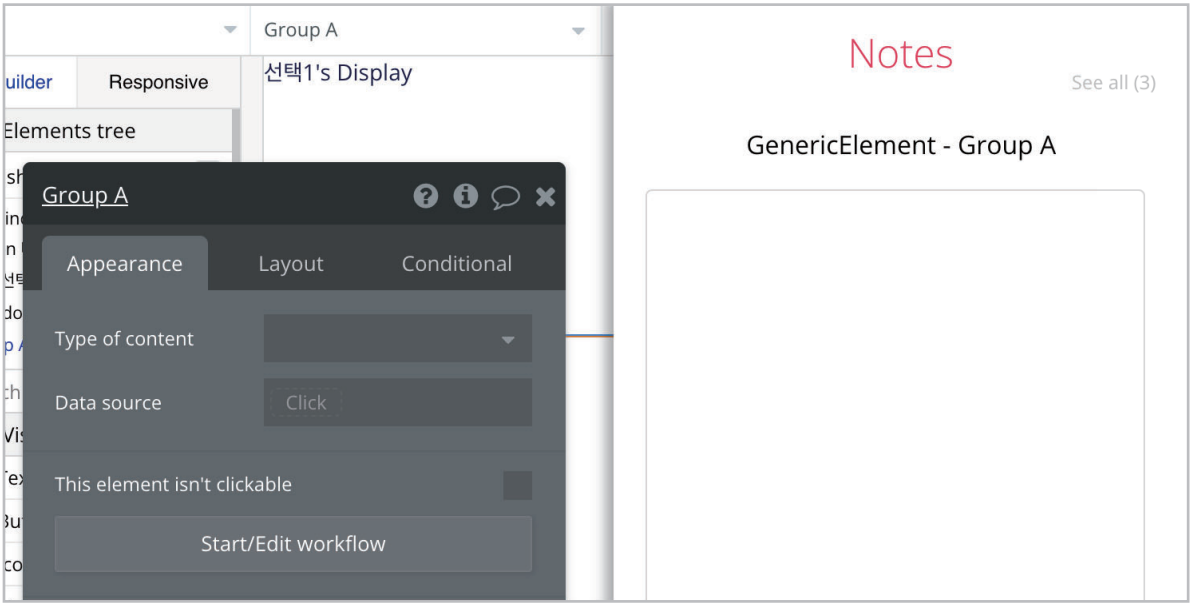
3. 전체 취소

아직 실행되지 않은 모든 과거 워크플로우 또는 예약된 워크플로우 전체를 삭제할 수 있습니다.

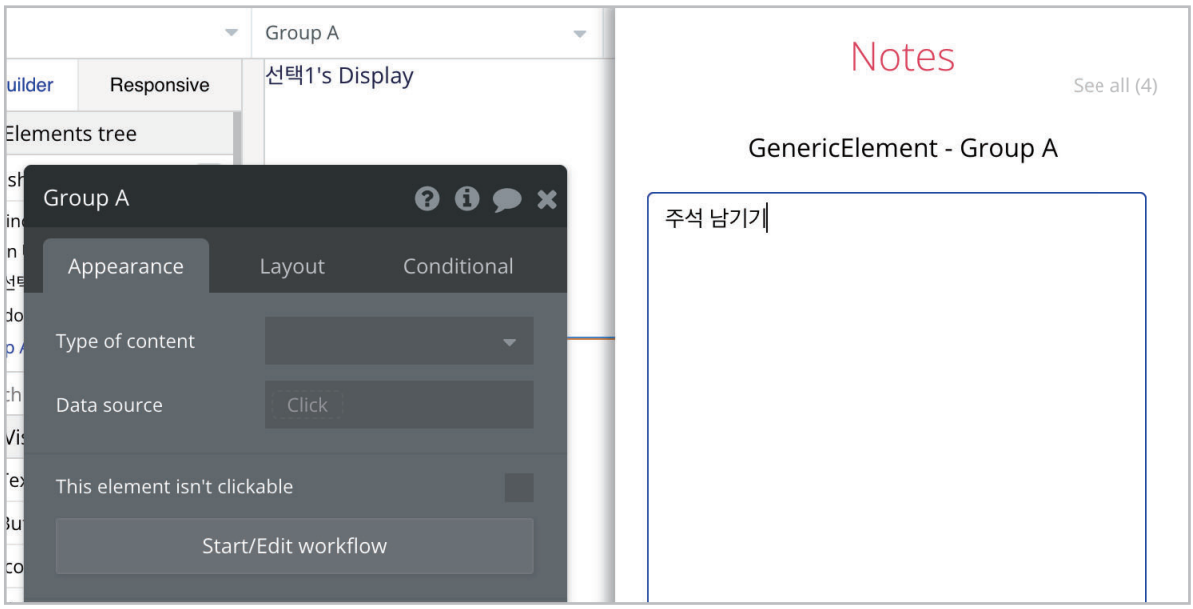
05 CHAPTER 주석

서비스가 커짐에 따라 다양한 페이지, 주요 액션, 스타일 및 데이터 유형이 무엇인지 추적하는 것이 중요합니다. 이를 위해 주석을 달아놓으면 서비스를 보다 쉽게 수정할 수 있고, 다른 사람들이 서비스 제작에 합류했을 때도 쉽게 협업할 수 있습니다.

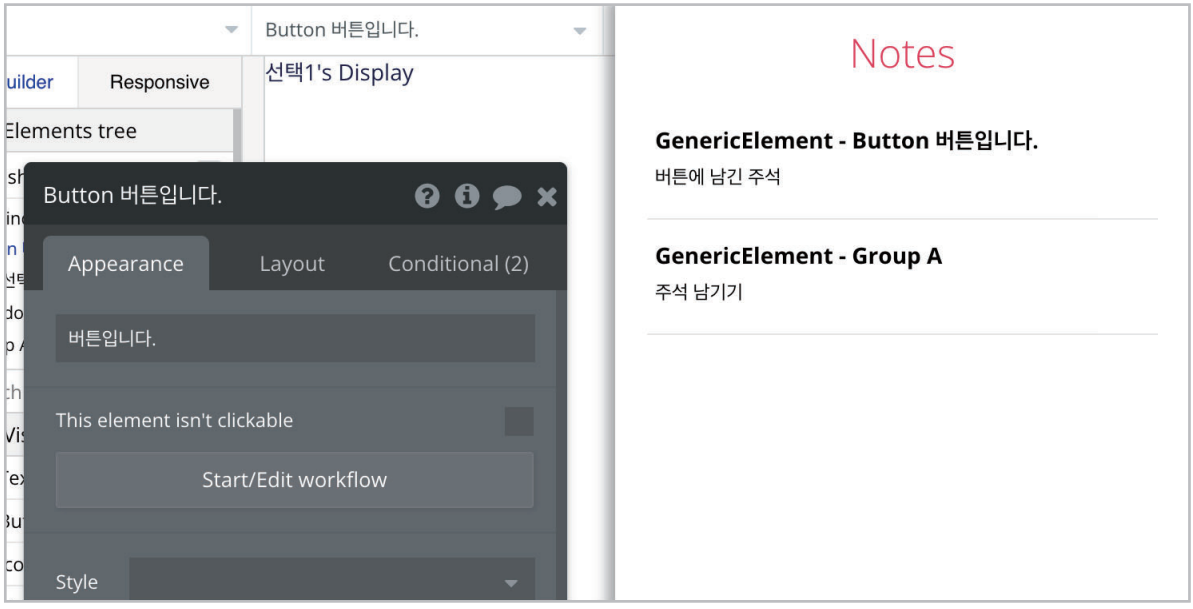
버블은 대부분의 요소에 주석을 달 수 있는 방법이 있습니다. 프로퍼티 에디터 상단에 있는 말풍선 아이콘을 클릭하면 주석을 달 수 있는 패널이 표시됩니다.



이 아이콘이 채워져 있다면 해당 요소는 주석이 존재한다는 것을 뜻합니다.



주석 패널을 사용하여 모든 주석을 하나로 볼 수 있으며, 항목을 클릭하면 주석이 표시되고 에디터에서 해당 요소로 이동합니다. 많은 사용자들은 이것을 자신의 서비스에서 협업하고 작업관리 목록을 설정하는 방법으로 사용합니다.



06 CHAPTER 협업

버블을 사용하면 두 명 이상의 에디터가 서비스를 수정할 수 있습니다. 여러 사용자가 동시에 앱을 수정할 수도 있습니다. 다른 사용자를 초대하여 서비스를 수정하려면 상위 요금제를 결제해야 합니다.

1. 에디터 권한

버블 앱의 소유자는 플랜에 대한 비용을 지불하는 사람이지만 관리자만 편집 권한을 부여하거나 취소할 수 있습니다. 사용자에게 적용할 수 있는 다양한 권한을 선택할 수도 있습니다. 아래에는 편집할 수 있는 여러 권한들입니다.

(1) View – Edit the application

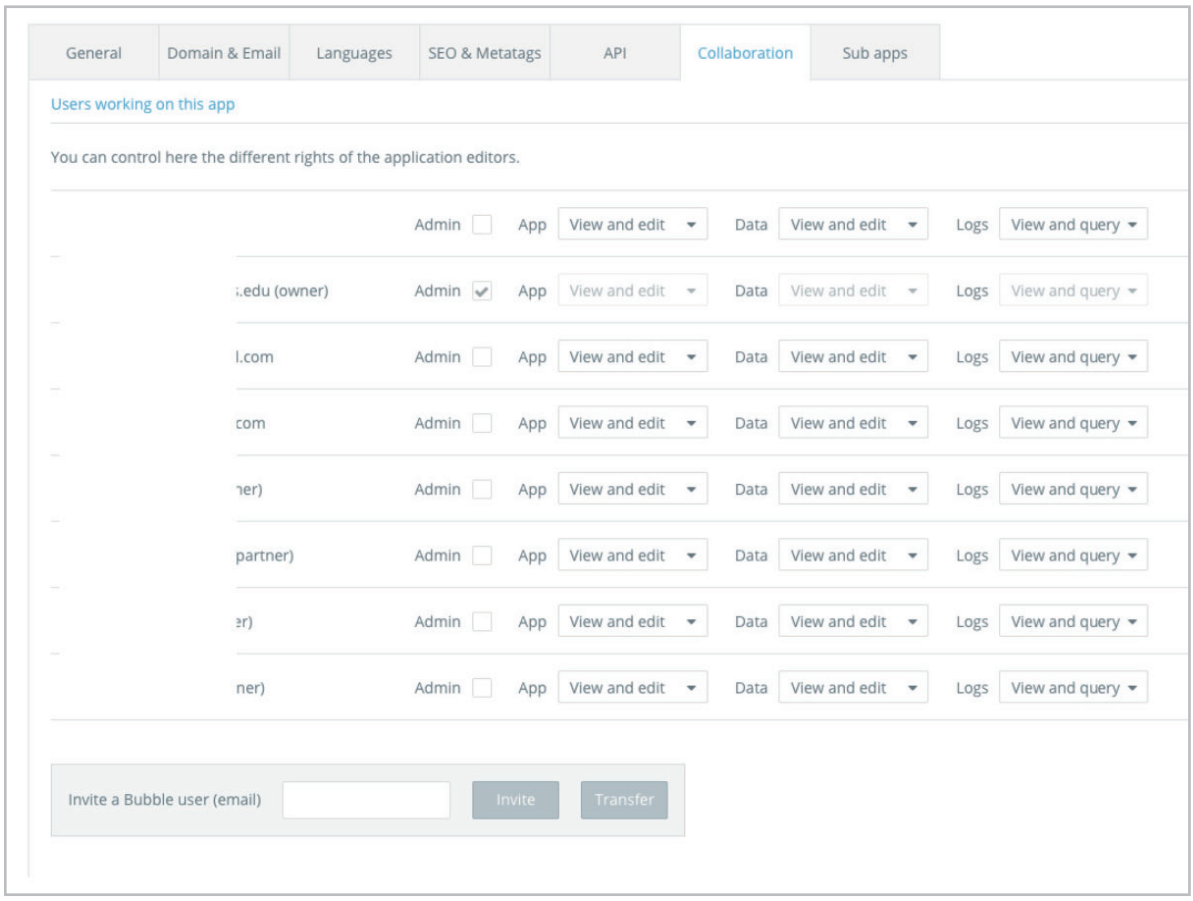
기본적으로 서비스에 대한 액세스 권한이 있는 사용자는 버블 앱을 볼 수 있습니다.

(2) View – Edit the application's database

조회는 사용자가 App Data 뷰어에서 데이터베이스를 쿼리할 수 있음을 의미하며, 편집은 사용자가 항목을 생성, 수정 및 삭제하고 대량 작업을 실행할 수 있음을 의미합니다.

(3) View & Query logs

이 옵션은 사용자가 서버 로그나 예약된 워크플로우를 볼 수 있음을 의미합니다.



2. 다중 편집

두 명 이상의 사용자가 동시에 버블 앱을 수정하면 다른 사용자의 마우스가 표시되므로 두 사용자가 동시에 동일한 요소를 수정하지 못하도록 방지할 수 있습니다.

PART 10

서비스 최적화

01 CHAPTER 퍼포먼스 & 스케일링

이번 챕터에서는 버블로 만든 서비스를 어떻게 최적화하고 퍼포먼스를 극대화 할 수 있는지 알려드립니다.

버블 팀은 확장성과 성능을 최적화하기 위해 끊임없이 노력하고 있습니다. 그리고 이러한 성능과 확장성은 어떻게 버블 앱을 구축하는 지에 따라 크게 영향을 받습니다. 이 장에서는 앱의 성능과 확장성에 대한 개요를 제공하고 몇 가지 구체적인 팁을 제공합니다.

1. 성능에 대한 일반적인 원칙 & 팁

(1) 데이터를 적게 가져올수록 성능이 빨라집니다.

페이지 로드 시 일부 데이터를 가져와야 하는 경우가 많습니다. 페이지 로드 시 100개를 가져오는 페이지는 100만 개의 데이터를 가져오는 페이지보다 훨씬 더 빨리 로드됩니다. 이와 유사하게 숫자와 같은 간단한 데이터 타입을 가져오는 것이 큰 단위의 데이터를 가져오는 것보다 더 빠릅니다.

(2) 마찬가지로 작고 단순한 페이지를 많이 갖는 것이 더 적은 수의 복잡한 페이지를 갖는 것보다 더 빠릅니다.

(3) 모든 정렬 또는 필터링을 원래의 검색과 최대한 가깝게 유지해야 합니다.

버블은 이미 여러 가지 방법으로 데이터베이스 쿼리를 최적화합니다. 하지만 데이터베이스 수준에서 정렬 또는 필터를 수행하는 것이 매우 효율적입니다. 즉, :sort 또는 :filter를 적용하는 쿼리는 결과를 다른 종류로 조작한 후 정렬 또는 필터링 하는 쿼리보다 효율적입니다(예: search:count를 수행하는 것이 search:group by:count보다 효율적입니다).

(4) 고급 필터를 사용하면 쿼리가 느려질 수 있습니다.

기본 원리는 위에 말했듯이, 필터(또는 정렬)이 "데이터베이스에서" 수행될 수 있는 경우가 데이터베이스에서 초기 데이터 집합을 검색한 후 필터링(또는 정렬)하는 것 보다 빠르다는 것입니다. 데이터베이스에서 그렇다면 "데이터베이스에서" 수행되는 필터(또는 정렬)은 검색 팔레트에 표시되는 필터("Do a search for"를 누르면 나타나는 추가 사이트바)는 데이터베이스에서 수행되므로 일반적으로 빠릅니다. :filter에 적용되는 필터는 일반적으로 속도가 느린 "고급" 필터입니다.

(5) 연결된 쿼리들은 병렬이 아닌 직렬로 실행됩니다.

버블을 사용하면 한 검색의 결과를 다른 검색의 제약 조건으로 사용할 수 있습니다. 이러한 검색은 병렬이 아닌 직렬로 실행되므로 첫 번째 검색에서 많은 데이터를 반환하면 두 번째 검색 속도가 느려집니다.

(6) 버블은 이미 많은 성능 최적화를 수행하고 있습니다.

버블은 데이터베이스에서 쿼리를 실행하고, 서버에서 이미지 크기를 조정하고, 브라우저에 javascript를 캐시하도록 지시합니다. 만약 비교적 단순하고 적은 양의 데이터를 가져올 때 느리다고 생각이 든다면, 이 장에서 제안한 최적화 방법을 적용할 수 있는지 검토해봐야 합니다.

(7) 일반적으로 간단하게 표현되는 쿼리가 더 빠릅니다.

항상 그렇다는 것은 아니지만 경험에 비추어 볼 때 좋은 방법입니다. 버블은 가장 일반적인 패턴에 대한 데이터베이스 최적화 작업을 지속적으로 수행하고 있습니다.

(8) 모든 페이지 로딩에서 데이터의 수정을 피하도록 합니다.

동일한 동작을 수행하기 위해 데이터베이스를 추가로 호출하는 것 보다 요소의 상태(커스텀 상태)를 변경하는 것이 더 효율적입니다.

(9) 많은 양의 계산은 예약된 워크플로우로 이동시킵니다.

예약된 워크플로우는 무거운 쿼리를 실행한 후 나중에 사용할 수 있도록 결과를 저장해 놓을 수 있습니다. 이는 페이지 로딩에서 무거운 쿼리를 실행하는 것 보다 성능이 뛰어납니다.

(10) “Make changes to a list of X” 워크플로우 작업을 주의합니다.

이 액션은 짧은 항목을 빠르게 변경할 때는 유용하지만 목록의 길이가 길어지면 워크플로우의 시간 초과 위험이 빠르게 증가합니다. 이 액션으로 시간 초과를 발생시키는 대신 “Schedule API Workflow on a list”를 고려해보는 것이 좋습니다. 이 액션을 사용하면 수정할 목록을 따로 가져가서 각각의 아이템에 대해 실행되기 때문에 시간초과의 위험이 감소합니다.

주의!

버블 앱의 데이터베이스는 매우 유연하고 강력합니다. 많은 데이터를 저장할 수 있지만 한 필드에 너무 많은 데이터를 저장하려고 하면 해당 필드와 관련된 성능 문제가 발생할 수 있습니다. 예를 들어 내용 필드가 있는 블로그 데이터 타입인 경우, 블로그 게시물을 제대로 처리할 수 있어야 합니다. 하지만 만약 이 내용 필드에 위키피디아의 모든 내용을 집어넣으려고 한다면 잘 작동하지 않을 것입니다. 보다 더 현실적으로 텍스

트 필드에 64로 인코딩된 이미지(많은 텍스트와 동일)를 저장하려고 하면 성능이 저하되고 예기치 않은 동작이 발생할 수 있습니다.

2. 페이지 로드 시 일어나는 일

아래는 버블이 페이지를 로드 할 때 발생하는 대략적인 이벤트 순서입니다.

(1) 버블이 모든 요소(보이는 것과 보이지 않는 것 모두)에 대한 코드를 보냅니다.
(2) 버블은 페이지에 표시되는 모든 요소를 그립니다.
(3) 버블이 표시되는 요소에 필요한 모든 동적 데이터를 가져옵니다.

이 말은 즉슨, 보이지 않는 요소는 나중에 표시될 때까지 그려지지 않습니다. 위의 명제는 보이는 요소가 보이지 않는 아이템을 참조하지 않는 한 유효합니다. 페이지 로드의 속도는 요소의 유형보다 요소의 수에 더 큰 영향을 받습니다. 모든 요소 유형은 두 가지 예외를 제외하고 성능 면에서 서로 상당히 유사합니다.

(1) 반복 그룹은 레이아웃 스타일 속성에 따라 다른 양의 데이터를 로드 합니다. 또한 반복 그룹의 각 셀에 요소가 많을수록 페이지를 렌더링하는 데 더 많은 시간이 걸립니다.

각각 10개의 셀과 2개의 요소가 있는 반복 그룹은 20개의 개별 요소보다 빠르지만 3개의 요소가 있는 반복 그룹보다는 느립니다.

중첩된 반복 그룹은 요소의 수에 곱셈만큼 로딩이 필요합니다.

(2) 플러그인은 사용 여부에 관계없이 각 페이지 로드에 코드가 포함되어 있습니다. 플러그인을 사용하지 않으면 버블이 렌더링하지 않기 때문에 성능에 큰 영향을 미치지는 않지만, 일반적으로 서비스에서 사용하지 않는 플러그인은 제거하는 것이 좋습니다.

3. 성능에 대한 추가적인 팁

(1) 재사용의 효율성

페이지에 동일한 검색이 두 개 이상 있는 경우, 버블은 자동으로 결합하여 쿼리를 한 번만 실행합니다. 스타일을 활용하여 성능을 향상 시킬 수 있습니다.

특히 무거운 검색을 실행할 때, 처음 몇 번은 향후 실행 속도보다 다소 느릴 수 있습니다. 왜냐하면 버블이 무거운 쿼리를 몇 번 실행하다 보면 그 실행 자체를 인덱싱해놓아 나중에 검색 속도를 높일 수 있기 때문입니다.

(2) X vs. Y

① 12개의 필드를 변경하는 작업이 한 개의 필드를 변경하는 작업보다 효율적입니다.

② 상대적으로 작은 목록의 경우에는 항목 목록을 변경하는 것이 빠르지만, 큰 목록의 경우에는 API워크플로우가 워크플로우 시간 초과 위험을 방지하기 때문에 확장성이 향상됩니다.

③ 큰 목록을 변경할 때 전체 목록에 대해 API 워크플로우를 한 번에 실행시키는 것 보다 목록의 아래 항목에 대해 API 워크플로우를 재귀적으로 호출하는 것이 더 느리긴 하지만 확장성이 좋습니다.

④ 일반적으로 링크 요소를 통해 새 페이지로 이동하는 것이 조금 더 빠릅니다. 페이지 이동 워크플로우는 페이지를 변경하기 전에 데이터를 저장하기 위한 다른 워크플로우를 기다려야 하기 때문입니다.

⑤ 데이터 타입 A가 여러 B에 연결되어 있는 경우(예: 게시글 당 하나의 카테고리를 가지는 경우, A = 카테고리, B = 게시물), A를 참조하는 필드가 B에 있는 것이 좋습니다. A에 포함하고 있는 B들을 필드로 가지게 되면 해당 목록이 매우 길어지면 잘 작동하지 않을 수 있기 때문입니다.

⑥ API 워크플로우의 경우 워크플로우가 수행해야 하는 항목의 수가 각 항목의 크기보다 성능에 더 큰 영향을 미칩니다.

4. 용량

기술적이지 않은 용어로, “용량”은 특정 기간 동안 앱이 할 수 있는 “기능”을 측정합니다.

웹 사이트에 방문하는 한 사용자는 약간의 용량을 사용합니다. 웹 사이에 방문하는 많은 무리의 사람들은 많은 용량을 사용합니다.

데이터베이스를 호출하면 용량이 사용됩니다. 이때 대량의 데이터베이스 쿼리를 수행하면 훨씬 더 많은 용량이 사용됩니다. 특정 워크플로우(서버에서 발생하는 워크플로우)를 실행하면 용량이 사용되며, 마찬가지로 앱의 API를 호출하면 용량이 사용됩니다.

버블을 통틀어서 이 용량의 "단위"에 대한 설명을 해보겠습니다. "유닛"은 버블의 시스템이 사용하는 다양한 자원의 가치 측정입니다. 이는 서버 CPU시간, 데이터베이스 CPU시간, 기타 백엔드 시스템 등의 요소를 포함합니다. "유닛"에 대한 정확한 공식은 버블 팀이 백엔드 시스템을 추가, 제거 또는 개선함에 따라 변경됩니다. 버블의 목표 중 하나는 용량 단위가 제공하는 성능을 계속해서 개선하는 것입니다.

특정 버블 가격 요금(예: 무료 및 Personal 요금제)에서 버블 앱은 "기본" 서버 용량을 갖습니다. 즉, 같은 요금제를 사용하는 버블 앱들은 서로 동일한 컴퓨팅 리소스를 공유합니다.

버블 앱을 "Professional" 및 "Production" 요금제로 업그레이드하게 되면 해당 앱에 할당된 "확보된" 단위의 용량이 제공됩니다. 용량이 초과되면 앱의 속도가 제한됩니다. 기술적이지 않은 용어로 말하자면 앱이 주어진 시간 내에 많은 "일"을 할 수 없다는 것을 의미하며, 앱에 대한 사용자의 요청은 확연히 느려집니다. 따라서 일반적으로 더 많은 용량을 갖는다는 것은 앱이 많은 "일"을 수행할 수 있다는 것을 의미합니다.

조금 더 쉽게 풀어 이야기하면 용량은 마트에 계산대가 얼마나 많은 지와 비교하면 이해하기 편합니다. 매장에 계산대를 더 추가하면 동시에 구매하는 고객들을 더 많이 처리할 수 있습니다. 그러나 고객이 수백 개의 상품을 카트에 싣고 오는 경우에는 한동안 해당 계산대를 차지하게 됩니다. 또한 계산대가 더 많다고 해서 상품을 많이 싣고 온 고객이 더 빨리 결제가 완료되는 것은 아닙니다. 마찬가지로 용량을 늘린다고 해서 매우 복잡한 데이터베이스 쿼리가 더 빨리 실행되는 것은 아닙니다. 이는 한 고객이 카트에 많은 상품을 넣고 계산대에 온 것과 동일하기 때문입니다.

주의!

만약 버블에서 큰 쿼리가 앱의 용량을 모두 소모할 것을 감지했다면, 버블은 나머지 일에 대한 합리적인 사용자 경험을 유지하기 위해 대용량 쿼리의 속도를 늦출 것입니다. 따라서 특정 상황에서 용량을 추가하면 대용량 쿼리가 더 빨리 실행될 수 있습니다.

버블 에디터의 왼쪽에서 로그 탭으로 이동하면 여러분들의 앱이 얼마나 많은 용량을 이용하고 있는지 볼 수 있습니다.

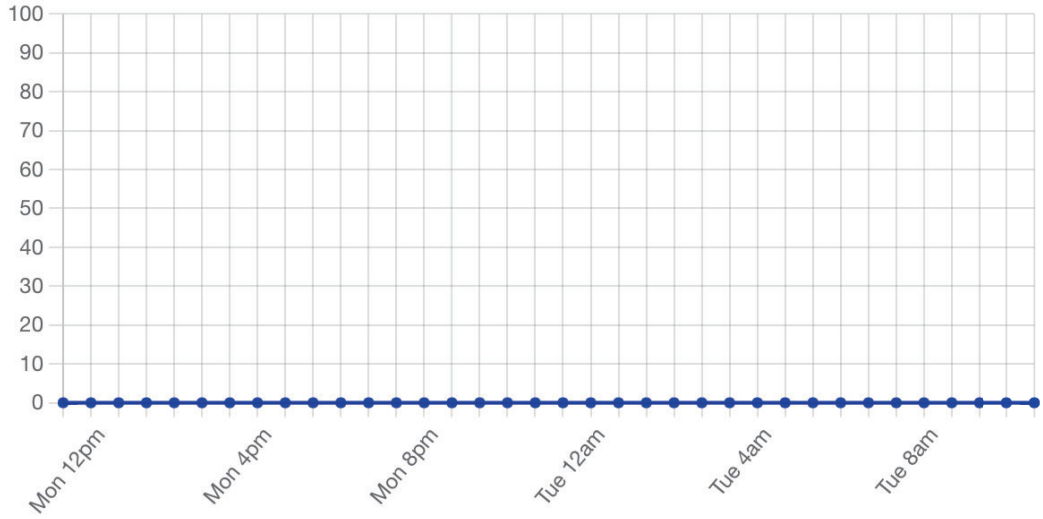
첫 번째 차트는 앱이 최대 용량에 도달한 시간을 보여줍니다.

Duration where the app hit its maximum capacity usage over the selected time period: 0 minutes

Your app could have been 1 minute faster with more capacity over the past 24 hours.

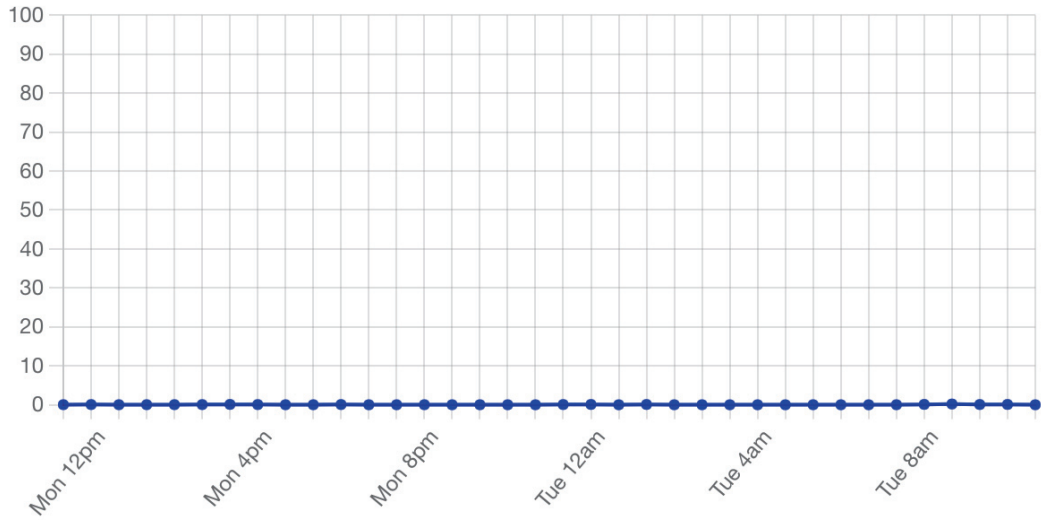
You can upgrade to a higher plan for more capacity and better performance.

Percentage of designated time period where maximum capacity was hit (%)

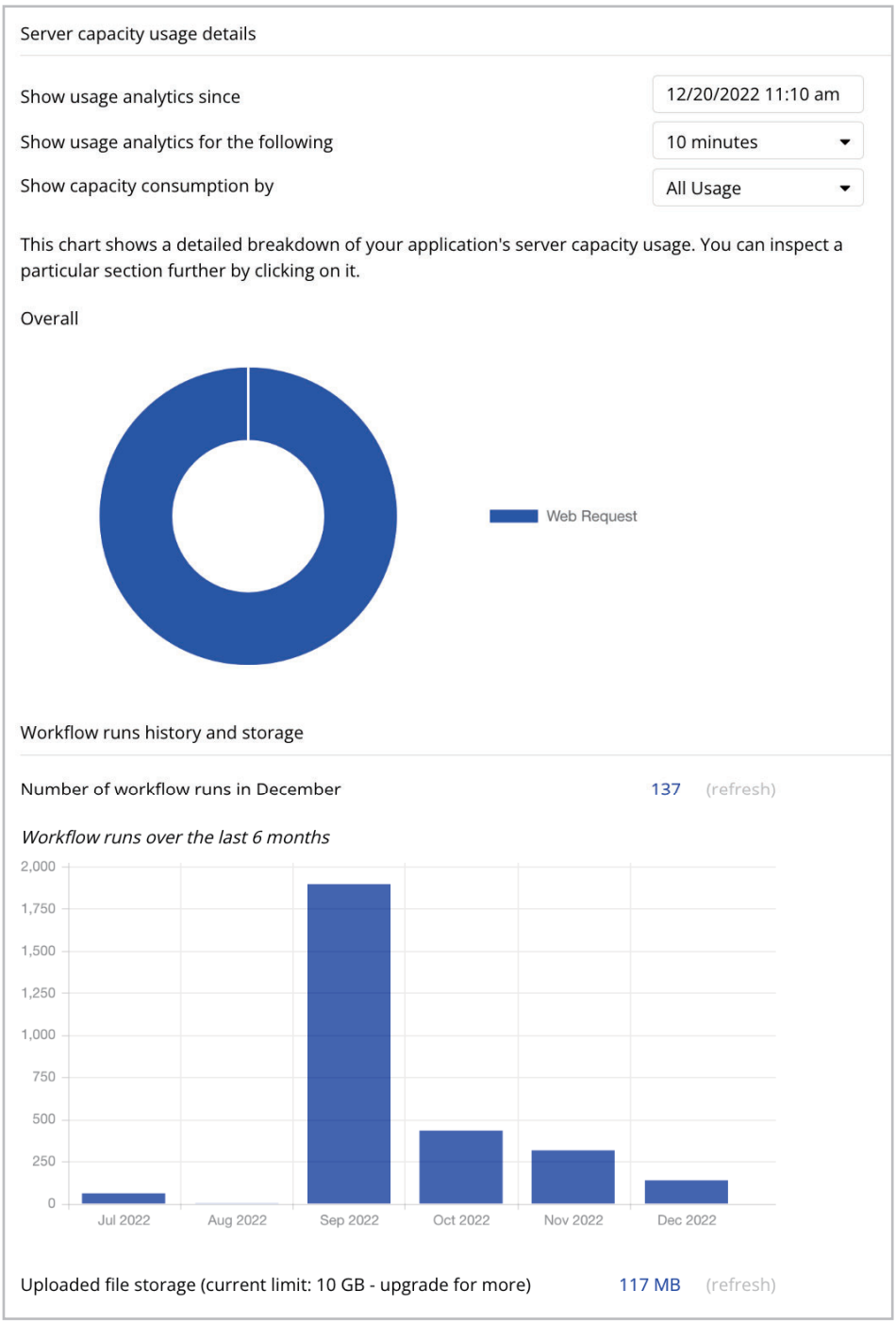


두 번째 차트는 앱이 최대 용량을 기준으로 사용한 용량을 보여줍니다.

Average CPU usage against the available capacity (%)

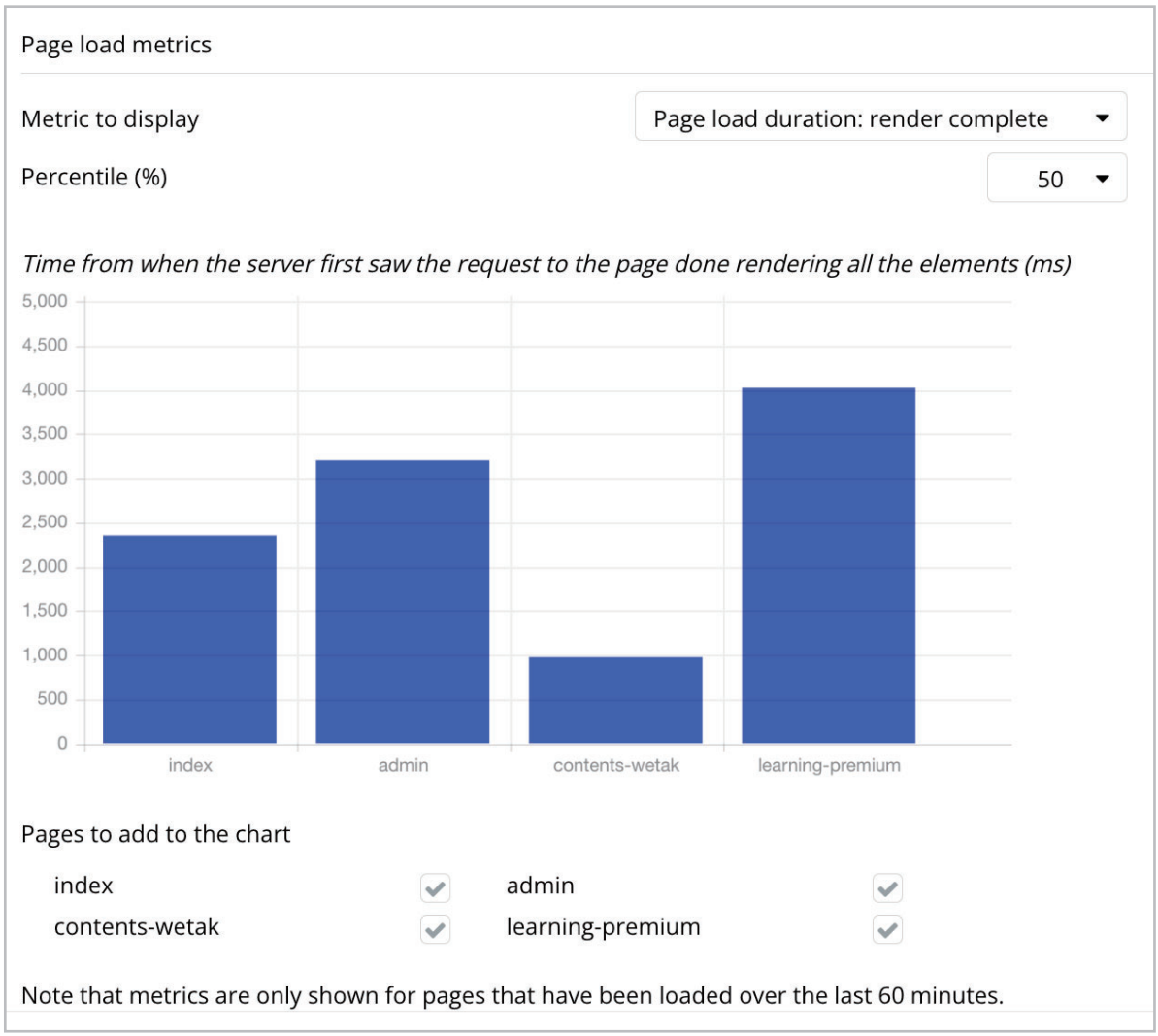


페이지의 더 아래에는 지난 24시간 동안 앱의 여러 부분에서 사용한 용량을 보여주는 서버 용량 사용에 대한 세부 정보를 보여줍니다. 서비스의 속도가 느리고 용량 제한에 도달하는 경우 추가 용량을 구입하는 것이 도움이 될 수 있습니다.



02 CHAPTER 용량

에디터의 로그 탭에서 앱의 용량 사용량과 실시간 메트릭을 볼 수 있습니다.



1. 표시할 메트릭

아래의 세 가지 종류의 메트릭을 볼 수 있습니다.

(1) Page load duration: render complete

버블 서버가 페이지에 대한 요청을 처음 수신할 수 페이지의 모든 요소 렌더링이 완료될 때까지의 시간(밀리 초)입니다. 여기에는 서버가 요청을 처리하는 데 걸리는 시간, 서버와 웹 브라우저 사이의 지연 시간, 모든 클라이언트 측 자바스크립트와 CSS를 로드하는 데 걸리는 시간, 브라우저가 페이지에 요소를 그리는 데 걸리는 시간이 포함됩니다.

(2) Page load duration: data loaded

버블 서버가 페이지에 대한 요청을 처음 수신한 후 페이지의 모든 요소가 렌더링되고 위의 요소에 필요한 모든 동적 데이터를 가져와서 표시하는 데까지 걸린 시간(밀리 초)입니다. 여기에는 "Page load duration: render complete"를 포함한 모든 시간과 데이터를 로드하는 시간(반복 그룹 목록들도 포함)이 포함됩니다.

(3) Page load: count of data items required

"Page load duration: data loaded"가 추적되는 동안 로드된 데이터 레코드의 수 입니다. 위의 요소가 로드를 완료하기 위해 페이지에 필요한 총 데이터의 추정치입니다. 이 기능은 데이터 로드가 페이지 로드 시간에 미치는 영향(예: 반복 그룹)을 조사하는 데 유용합니다.

2. 백분위

최상의 X 퍼센트 이상의 평균을 표시하도록 선택할 수 있습니다. 최상위 1%는 최상의 성과를 보여주고 99%는 모든 상황을 보여줄 것입니다.

03 CHAPTER 쿼리

이 장에서는 데이터베이스 쿼리 작동 방법에 대한 특별한 사항을 소개하겠습니다. 데이터베이스 작동 방식에 대한 기술적인 세부 사항을 설명하는 내용이지만, 데이터의 쿼리가 예상대로 작동하지 않는 상황을 디버깅하는 데 도움이 될 수 있습니다.

1. 멈춤 단어 Stop Words

"멈춤 단어"로 알려진 특별한 단어들이 있습니다. 이 단어들은 짧고 일반적인 단어들로 구성되어 있으며, 구문의 내용을 제외하고 데이터베이스가 이 단어들을 쿼리할 때 해당 단어들은 무시하게 됩니다. 몇 가지 예로는 "the", "I" 또는 "do"가 있습니다.

이는 이러한 멈춤 단어 중 하나가 쿼리에 속해 있거나 멈춤 단어와 관련된 쿼리가 있는 경우 결과가 예상과 완전히 다를 수 있음을 의미합니다.

버블은 postgres 방식을 따르기 때문에 이러한 앱 쿼리에 영향을 미치는 중지 단어 목록은 postgres를 참고하시기 바랍니다.

2. 스테밍 Stemming

"스테밍"은 데이터베이스가 검색 쿼리를 일반화하여 동일한 줄기의 다른 단어를 찾는 경우를 말합니다. 예를 들어 "test"와 "testing"은 동일한 줄기를 가지고 있으므로 "testing"을 검색할 때 "testing"이라는 단어가 포함된 결과를 찾을 수 있습니다.

Stemming은 의미론적으로 유사한 의미를 가진 단어들과 일치하기 때문에 "temp"와 "temps"는 일치하지만, "temp"와 "tempo"은 동일한 단어의 다른 시제가 아니기 때문에 일치하지 않습니다. 이와 비슷하게 "test"와 "intestine"도 그렇습니다.

3. 버블 쿼리 최적화

버블은 데이터베이스를 쿼리할 때 매핑 작업에 최적화를 적용하여 작업 속도를 향상시키려고 합니다.

아래처럼 쉼표로 구분된 동일한 숫자 목록을 반환하는 다음 두 검색을 비교해봅시다.

검색 A → "MyTypesList's count"
검색 B → "MyTypesList's: item 0's count, MyTypesList's: item 1's count, MyTypesList's: item 2's count, MyTypesLists: item 3's count, MyTypesLists: item 4's count, MyTypesList's: item 5's count, MyTypesList's: item 6's count

A 검색은 매핑된 작업이며 핵심에서 단일 데이터베이스 쿼리로 전환되어 상당히 성능이 뛰어납니다. 반면, B 검색은 7개의 개별 데이터베이스 쿼리를 생성하며, 이 쿼리는 연속적으로 실행되고 사후에 결합됩니다. 이는 상당한 성능 저하로 이어지므로 가능한 한 피해야 합니다.

04 CHAPTER 버블의 한계

다음은 앱을 빌드하는 동안 드물게 발생할 수 있는 버블에 알려진 문제들입니다. 여기에 적힌 버그는 정기적으로 검토되고 해결되는 대로 제거되나 작성하고 있는 시점에서 보고된 버그들을 기록해 놓겠습니다(여러분들이 읽는 시점에는 해결이 되었을 수도 있고 다른 버그가 생겼을 수 있습니다).

1. 일반

(1) 드롭다운 요소

버블 "드롭다운" 요소는 네이티브 HTML "〈select〉" 태그를 사용하기 때문에 일반적으로 확장된 옵션 목록에 스타일(예: 옵션 정렬, 테두리 원형도 등)을 적용할 수 없습니다(이는 일반적인 CSS의 제한 사항입니다). 대안으로 플러그인을 고려해 보는 것도 한 방법입니다.

(2) 스크롤 막대

스크롤 막대는 특히 반복 그룹에 있을 때 다른 브라우저 및 장치에서 설계된 것과 미묘하게 다르게 보일 수 있습니다. 이것은 버블이 스크롤바가 어떻게 보여야 하는지에 대해 최종 사용자의 운영 체제와 브라우저에 의존하기 때문입니다.

2. 오래된 하드웨어

버블 편집기를 신속하게 사용하기 위해 노력하고 있지만, 만약 오래된 하드웨어(예: 2012년 이전의 미드레인지 노트북 또는 2010년 이전의 데스크톱)를 사용하는 경우 편집기를 사용할 때 현저한 느림이 발생할 수 있습니다.

3. 플랫폼/브라우저

플랫폼/브라우저 구성에서 완전히 지원되지 않는 것도 있습니다. 사용자에 대해 대응할 때 다음 사항에 유의해야 합니다.

(1) 안드로이드(모바일)

- 통화 형식에 대한 입력이 지원되지 않습니다.

(2) 사파리(모바일)

- 페이지를 다시 로드하거나 장치를 기울일 때까지 팝업이 표시되지 않을 수 있습니다.
- 입력하는 동안 입력 커서가 몇 줄 간격 띄우기 될 수 있습니다.
- 열려 있는 팝업 뒤의 페이지는 여전히 스크롤 할 수 있습니다.
- Twilio Plugin 비디오 채팅은 해결하기 어려운 Safari 모바일의 일부 문제로 인해 제대로 작동하지 않습니다. 통화가 시작되면 통화 중인 사용자는 자신의 목소리만 듣고 볼 수 있지만 Twilio는 이 통화를 성공적인 통화로 인식합니다.

(3) 사파리 13+

Safari 13은 타사 쿠키가 iframe 요청으로 이동하는 것을 방지합니다. 이때문에 버블 앱이 iframe에서 실행 중이면 버블 클라이언트는 자신에게 귀속된 사용자의 uid로 워크플로우를 시작하지만 쿠키가 없기 때문에 no_user 서버 측으로 등록됩니다. 이러한 정보의 불일치로 워크플로우가 성공적으로 실행되지 않습니다. 해결 방법으로 서버와 클라이언트 모두가 익명 사용자임을 알 수 있도록 [설정] > [일반] > [개인 정보 보호 및 보안]에서 “새로운 방문자에게 쿠키를 기본적으로 설정하지 않음”을 설정하면 됩니다.

(4) 마이크로소프트 에지

- 브라우저 호환성 문제로 인해 탐색기에서 브라우저로 파일 드롭이 예상대로 작동하지 않습니다.

(5) 파이어폭스

브라우저 설정 “Firefox가 닫혔을 때 쿠키 및 사이트 데이터 삭제”가 활성화된 경우 Firefox의 localStorage 지속성에 버그가 있습니다. 버블 앱에 미치는 영향은 특히 쿠키 및 iframe과 관련된 일부 동작이 이 설정을 사용하는 Firefox의 최종 사용자에게 예상대로 작동하지 않을 수 있다는 것입니다. 가장 잘 알려진 오류 원인은 Stripe 트랜잭션으로, 앱으로 올바르게 리디렉션되지 않을 수 있습니다(불행히도, 파이어폭스 수준의 버그입니다).

주의!

브라우저마다 고유한 특성이 있습니다. 버블은 버블 앱 동작이 브라우저 전체에서 일관되도록 노력하지만, 이것을 버블이 통제할 수 없는 경우도 있습니다.

예를 들어, 서비스의 사용자가 "뒤로"를 클릭하면 일반적으로 "페이지가 로드될 때" 이벤트가 발생하지만, Safari는 이 동작을 허용하지 않습니다. 이것은 사파리가 다른 브라우저와 비교하여 "뒤로가기"가 작동하는 방식에 대해 다른 패러다임을 따르기 때문입니다.

브라우저 버전은 버블 앱의 전반적인 안정성에 중요한 역할을 합니다. 일부 버블 기능은 브라우저 버전이 구식이 되면 깨집니다. 브라우저에는 후속 업데이트로 수정되는 버그가 포함되어 있으므로 브라우저를 최신 버전으로 업데이트하는 것이 좋습니다.

일부 사용자 지정 글꼴은 운영 체제와 브라우저에 따라 약간 다르게 렌더링되기도 합니다. 이는 요소 레이아웃에 영향을 미칠 수 있는 높이 차이에서 가장 일반적으로 나타납니다.

4. 플러그인 호환성

플러그인은 앱의 기능을 확장하는 커스텀 코드의 "블록"이기 때문에 설치된 플러그인은 앱이 로드될 때 동시에 모두 컴파일됩니다. 따라서 앱에 설치된 모든 플러그인이 동일한 환경에서 공존할 수 있어야 합니다. 때때로 플러그인이 서로 충돌하여 특히 유사한 리소스 이름을 공유할 때 예기치 않은 동작이 발생할 수 있습니다. 버블은 플러그인 개발자들이 브라우저 코드의 일부에 일반적인 부분을 덮어쓰지 않도록 최선을 다할 것을 강력히 권고하지만, 항상 지켜질 수 없기 때문입니다.

예: AddToAny(버블)와 Selectize 드롭다운(서드 파티)

PART 11

버블 Web을 App으로 전환하기

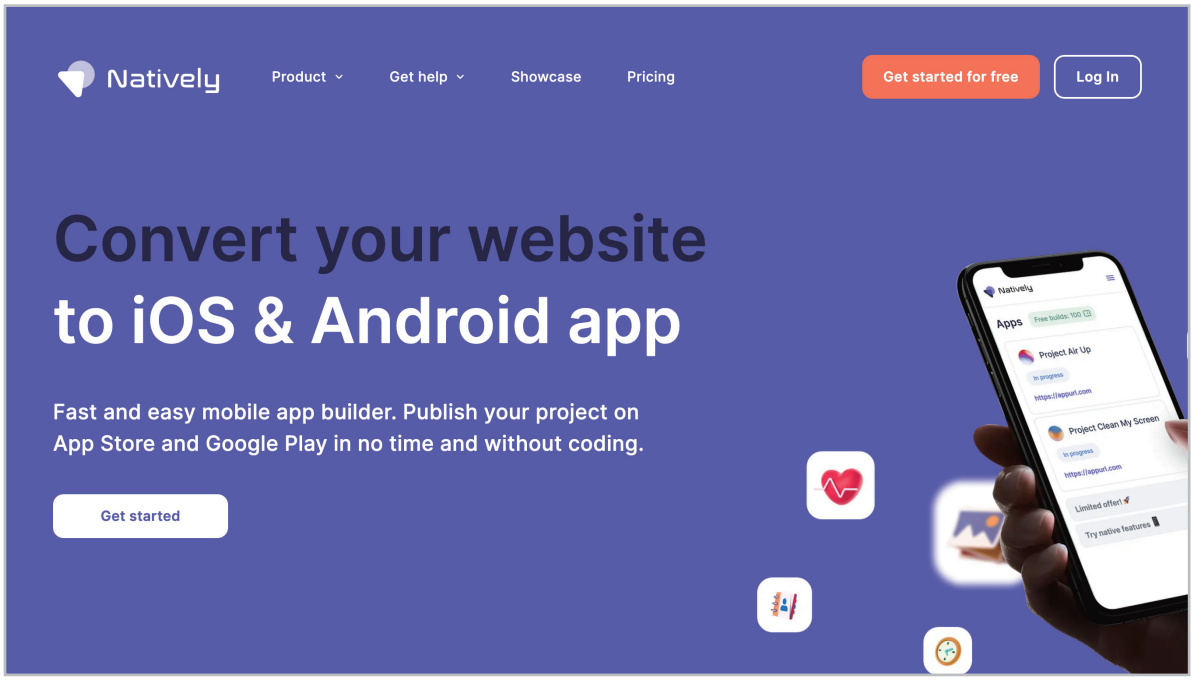
여기까지 노코드 툴 버블로 웹사이트를 만드는 방법과 그에 대한 심화내용을 살펴보았습니다. 추가로 버블은 웹사이트뿐만 아니라 앱도 만들 수 있다고 초반에 언급했습니다. 버블을 앱으로 만드는 방법은 몇 가지 방법을 이용해 웹 사이트를 앱으로 변환시키는 방법을 주로 사용합니다.

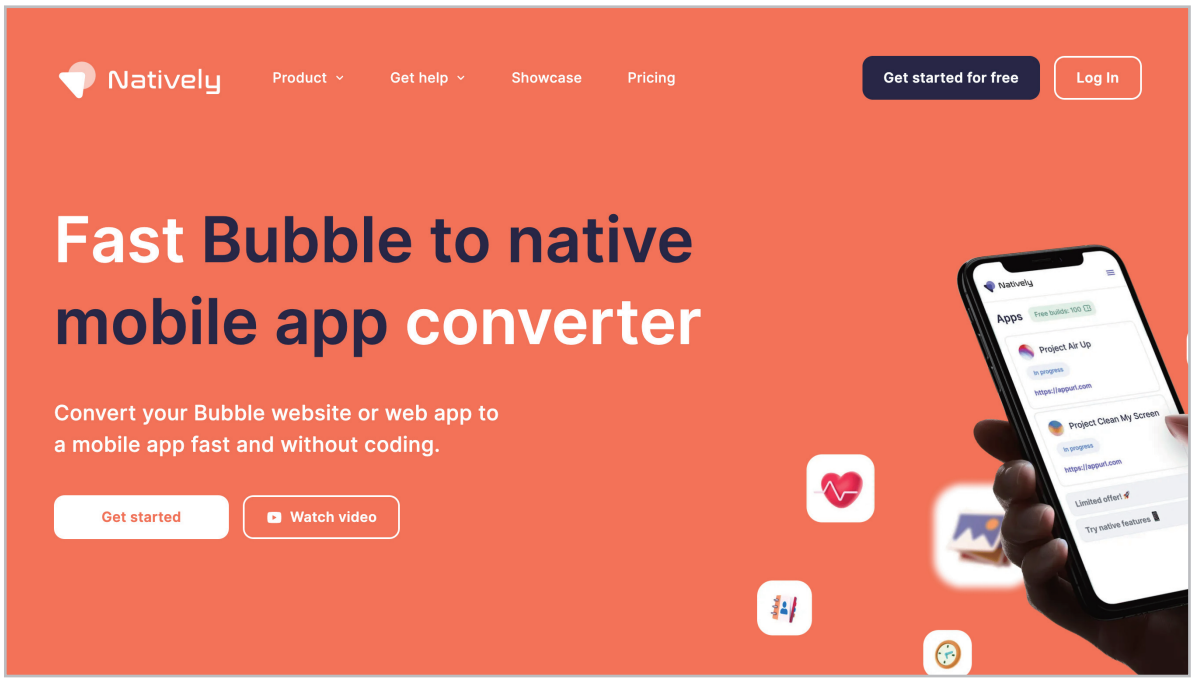
이번 장에서는 버블을 앱으로 전환하기 위해 쓰이는 몇 가지 보편적인 방법들을 홈페이지 및 진행 단계와 함께 알아보겠습니다. 그 중에서 대표적으로 이용되는 Natively에 대해 아래에 자세히 설명하겠습니다.

1. Natively

Natively는 일정 요금을 내면 여러분들이 만든 웹사이트를 앱으로 변환시켜주는 서비스입니다. 웹사이트를 앱으로 만들게 되면 인앱결제, 푸시알림, 주소록 접근, 카메라 및 마이크 접근, 지리정보가 필요한 기능을 추가로 구현할 수 있게 됩니다.

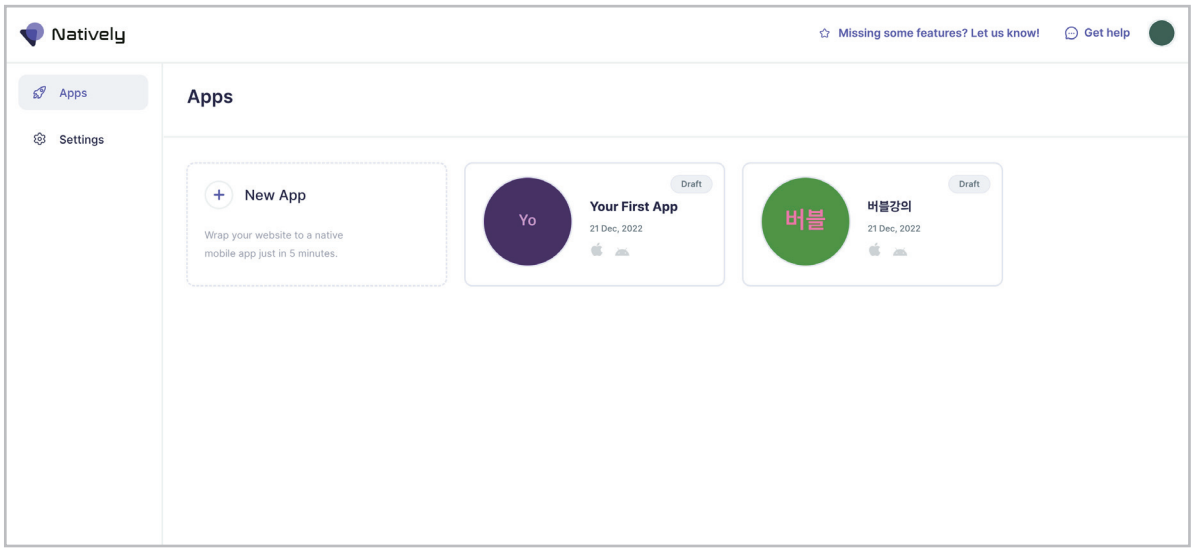
https://www.buildnatively.com/ Natively에 회원가입을 한 뒤, 버블 전용 서비스 페이지로 들어옵니다.



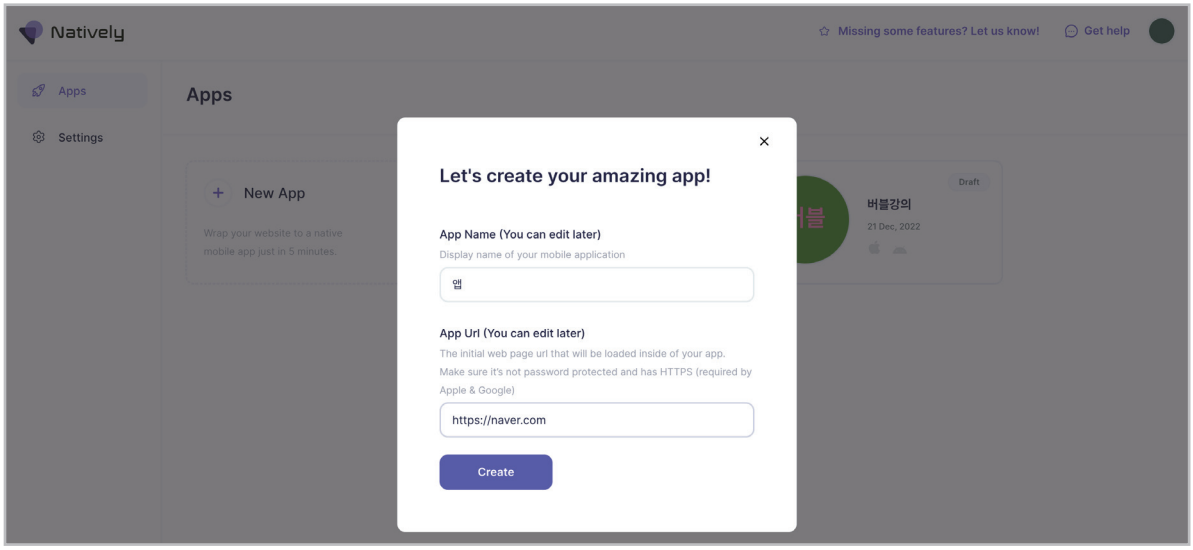


2. 대시보드

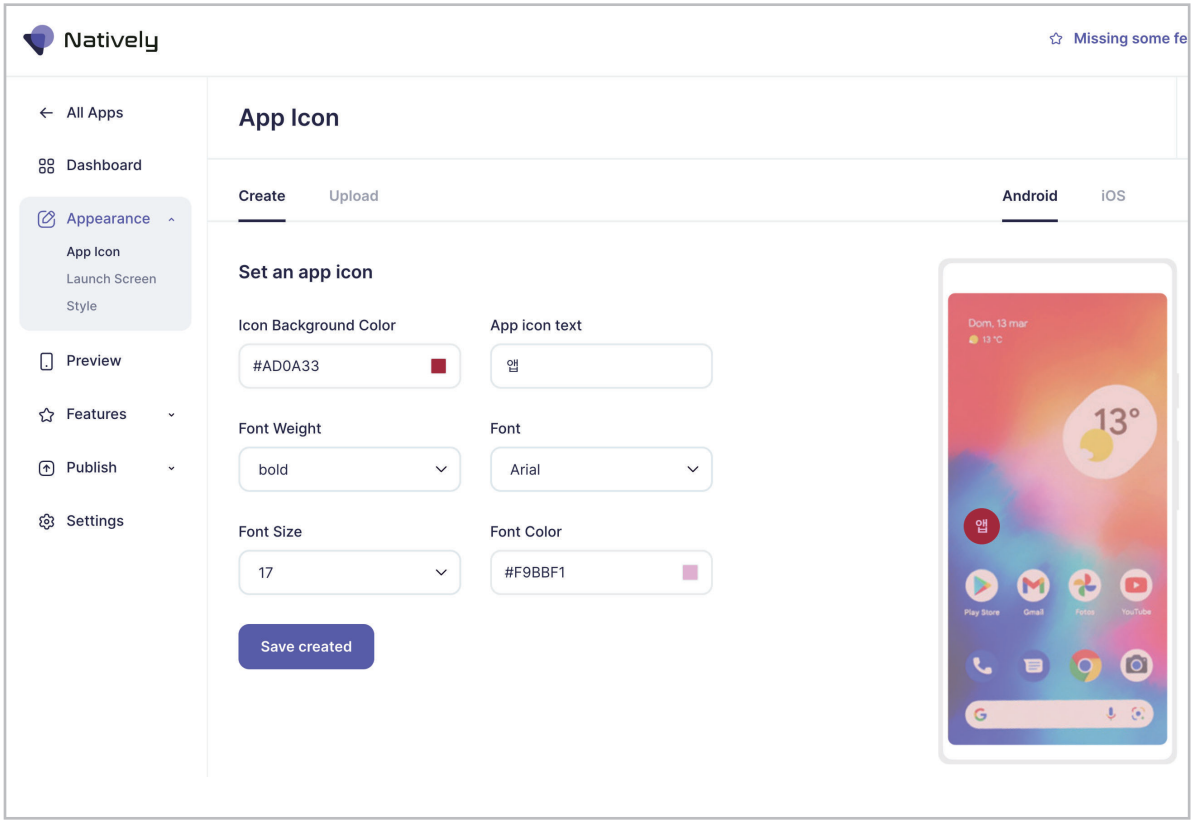
대시보드에서는 여러분의 웹사이트들을 변환시킨 앱을 모아볼 수 있습니다.

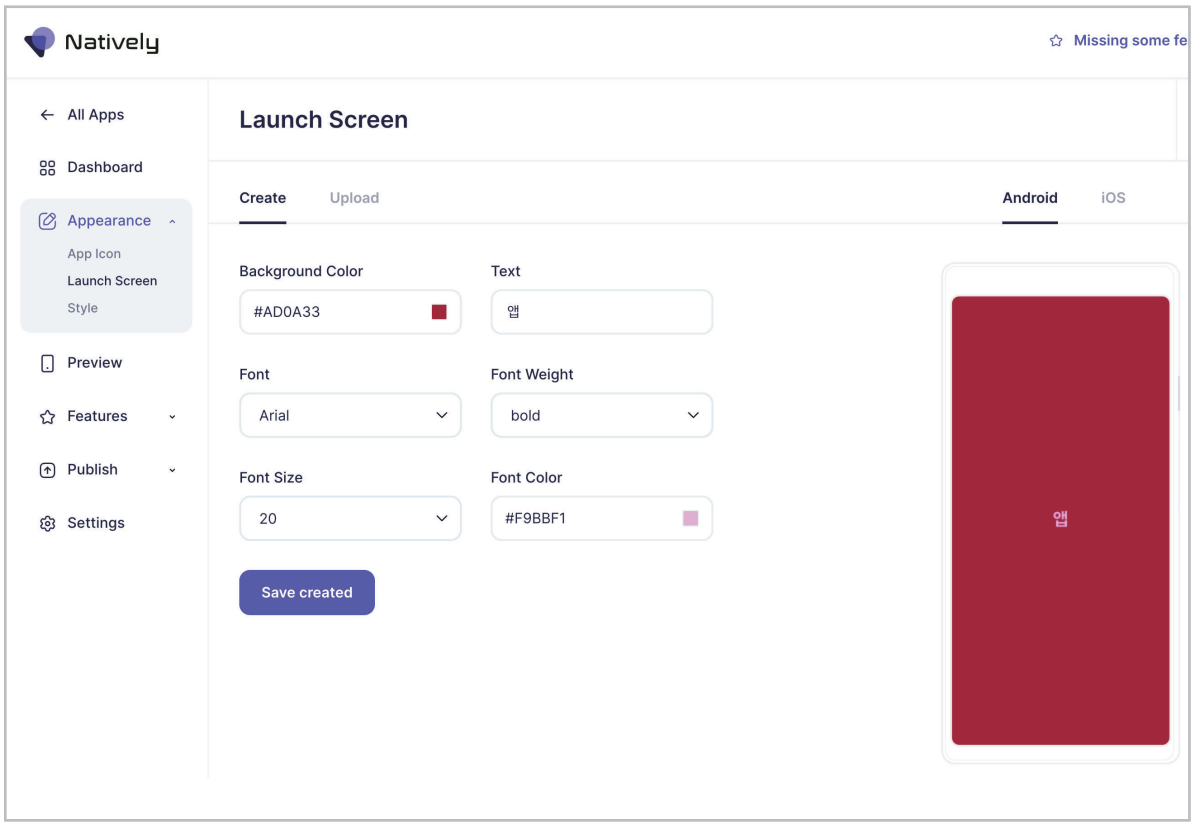


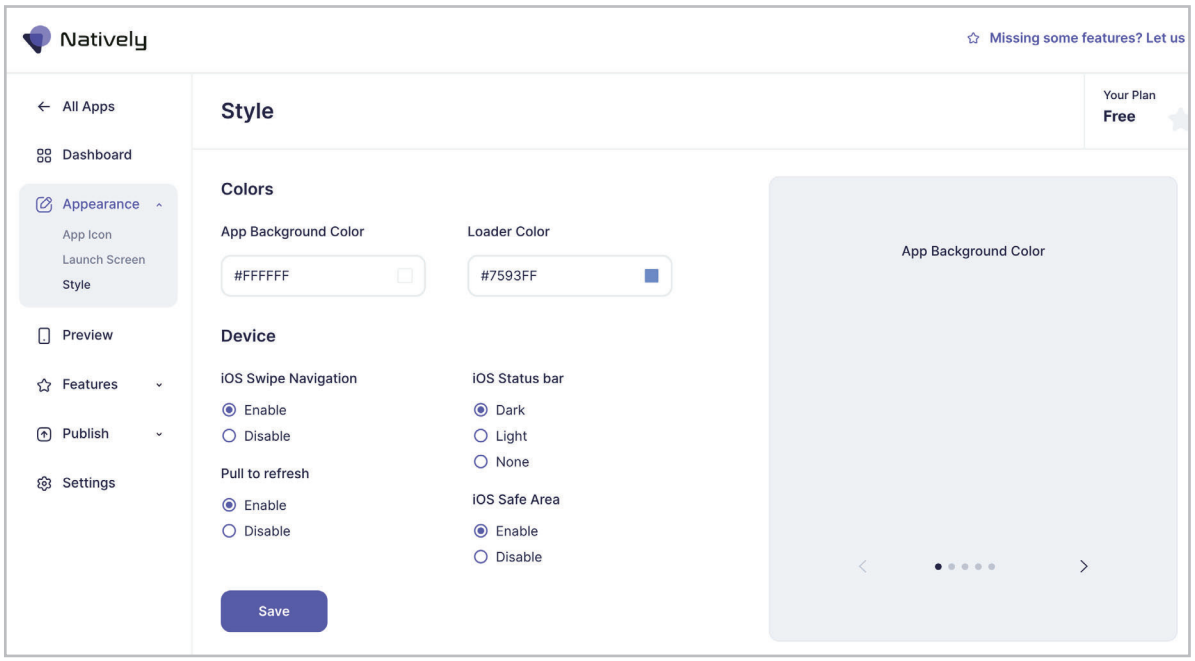
우선 New App을 클릭하여 새로운 앱을 생성합니다.



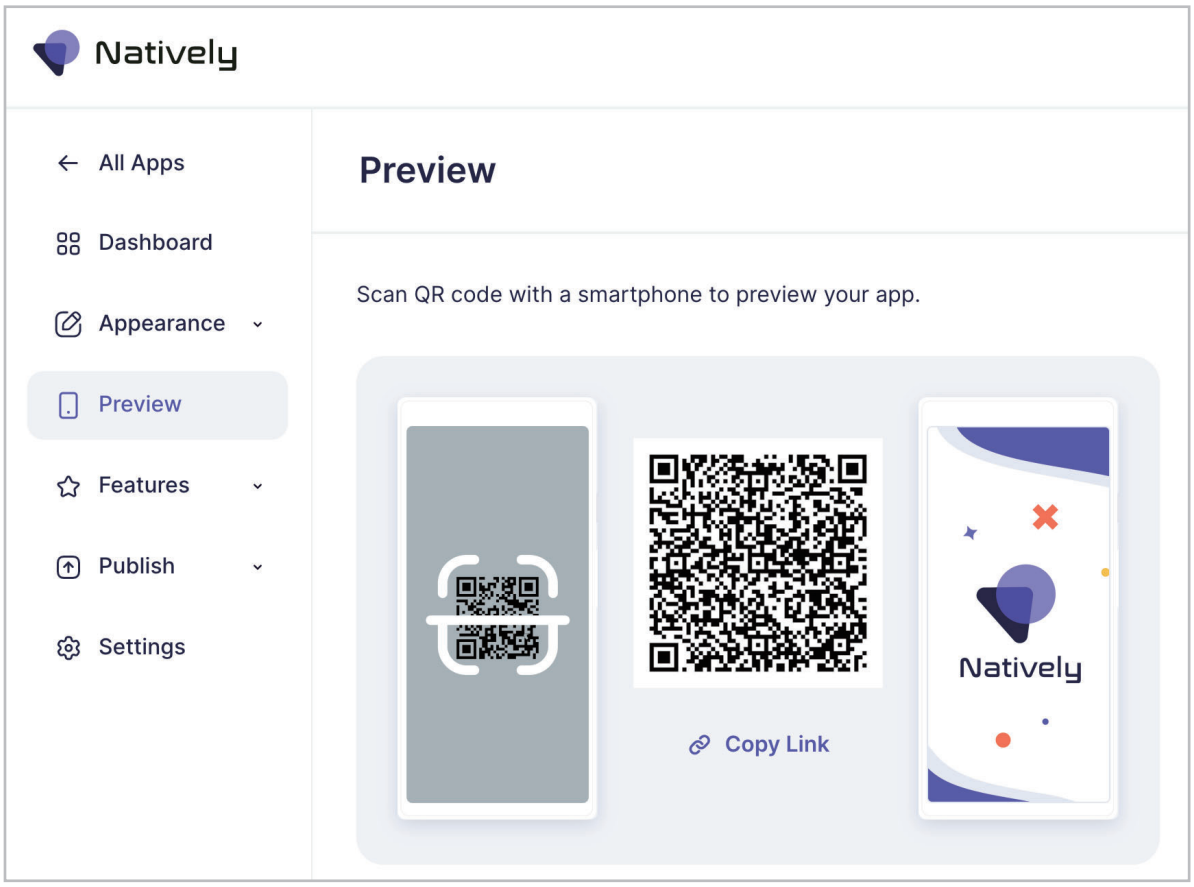
생성된 이후에는 앱 아이콘, 런치 스크린, 앱 스타일을 차례로 설정할 수 있습니다.





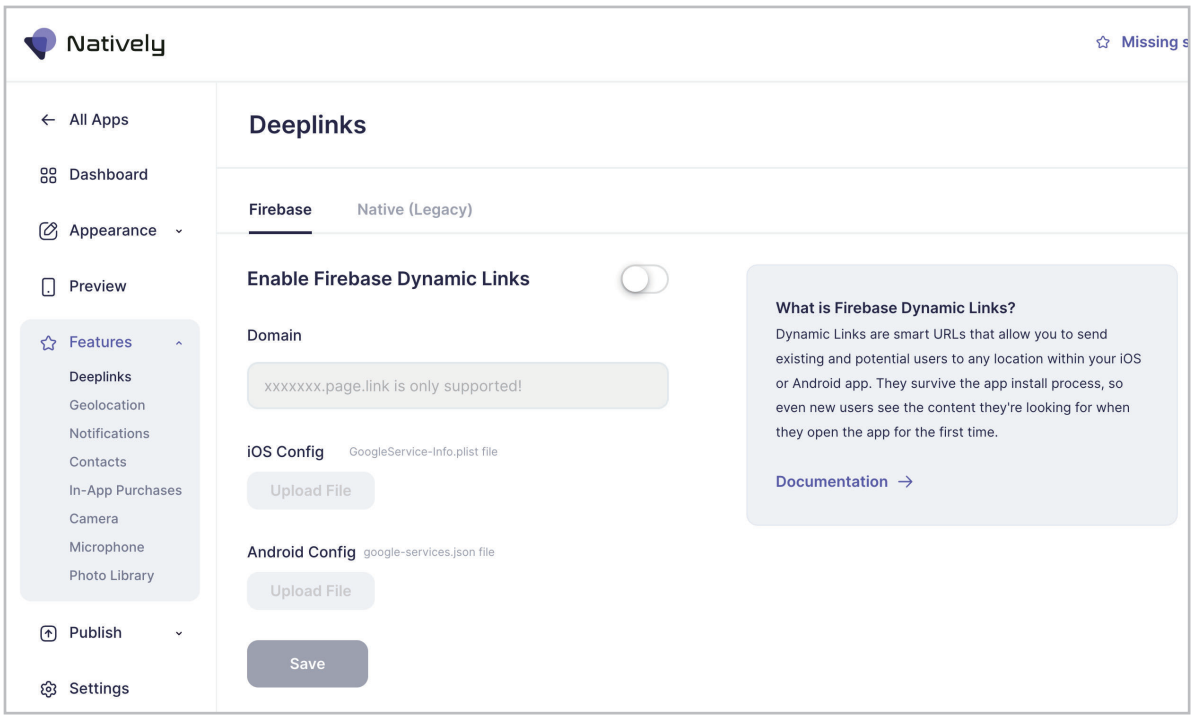


기본적인 설정이 다 끝난 후에는 Preview를 통해 변환 시킨 앱을 배포 전에 동작시켜볼 수 있습니다.

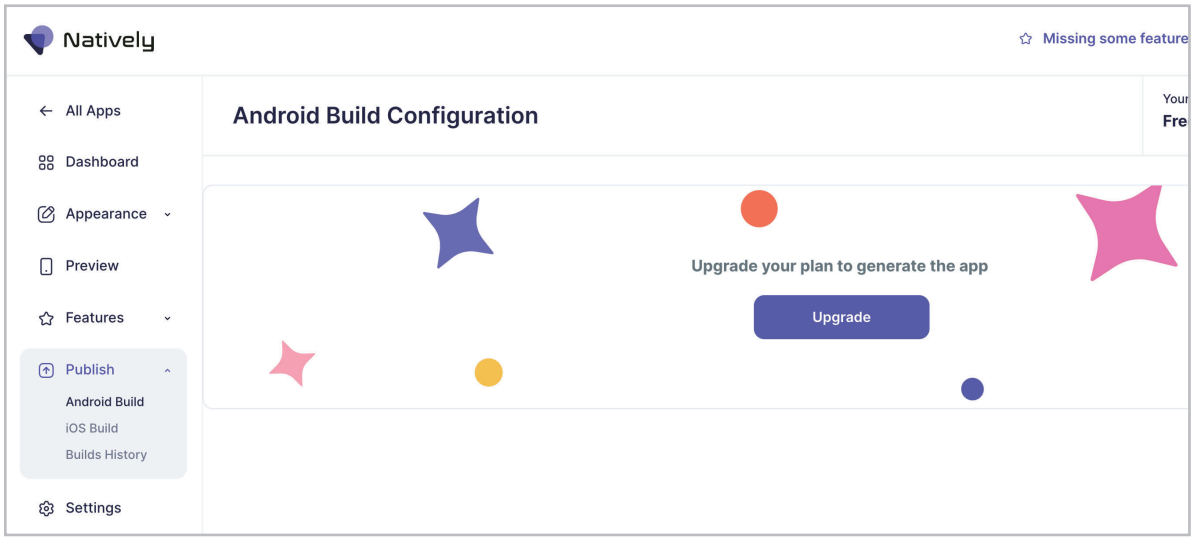


이외에도 네이티브 앱으로써 사용할 수 있는 모바일의 기능과 접근성에 대해서도 설정이 가능합니다.

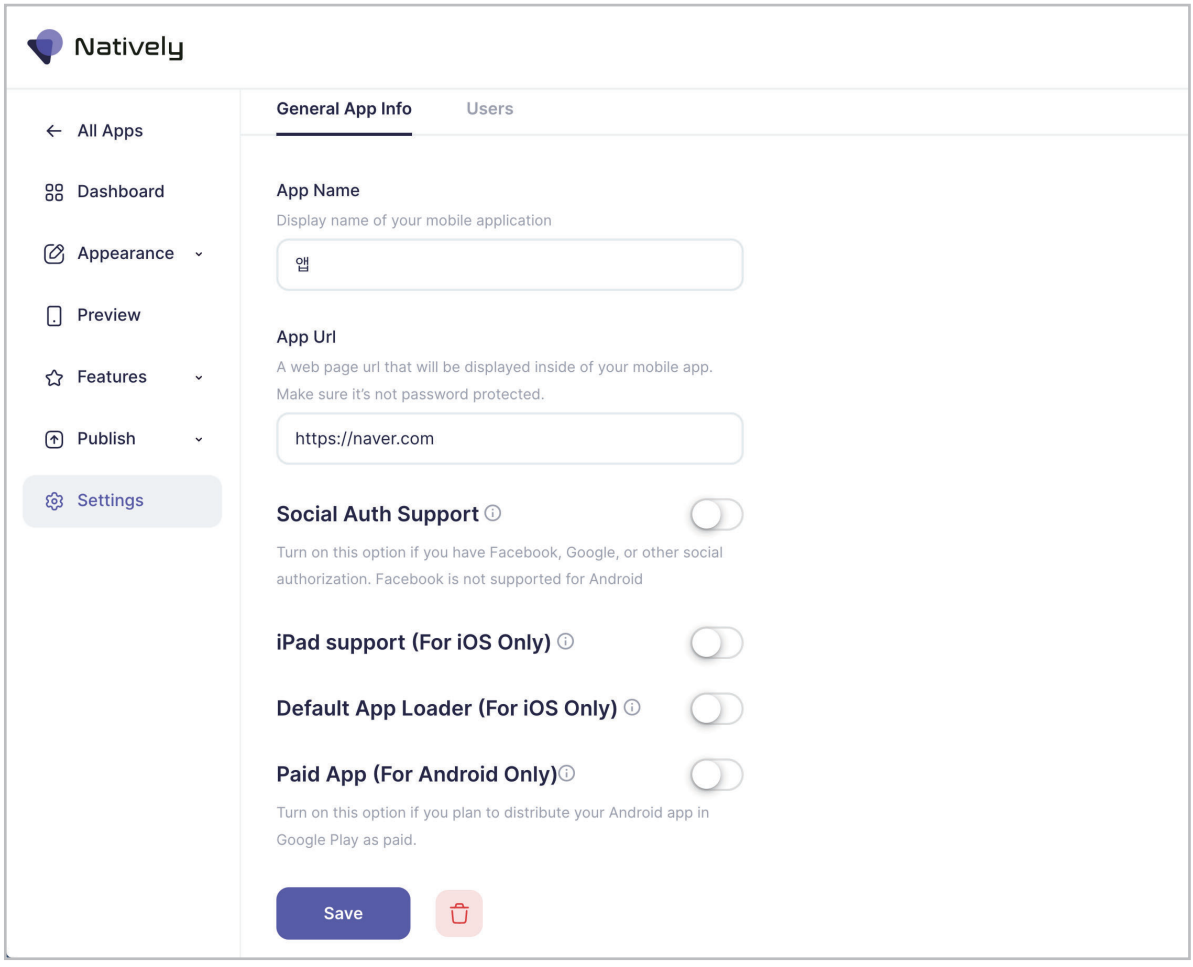
예를 들어 딥링크, 지도, 푸쉬알림, 주소록, 인앱결제, 카메라, 마이크, 사진 등을 커스터마이징할 수 있습니다.



마지막으로 배포는 안드로이드, 애플 각각의 개발자 계정이 필요합니다. 계정을 생성한 뒤에 발급받은 키를 Publish에 넣고 배포를 하면 됩니다. 이때 유료 요금제가 필요합니다.

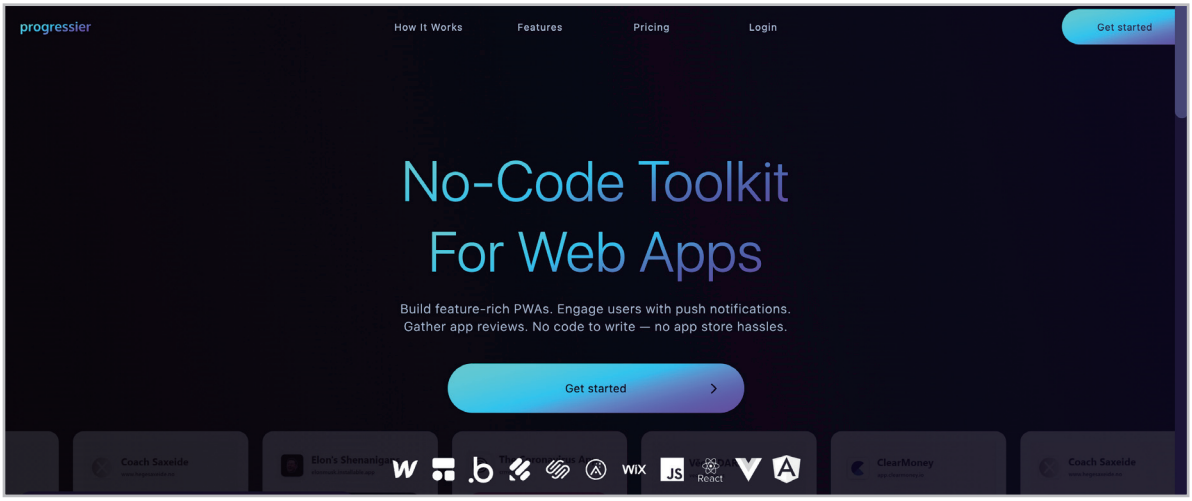


배포 후 빌드 히스토리도 통합해서 볼 수 있습니다.



마지막으로 세팅에는 앱의 기본 설정과 대시보드에 초대할 협업자를 초대할 수 있는 메뉴가 있습니다.

여기까지 버블앱을 네이티브 앱으로 변환시키는 방법에 대해 알아보았습니다. 하지만 버블앱으로 PWA도 생성할 수 있습니다. PWA는 웹과 네이티브 앱의 기능 모두의 이점을 갖도록 개발된 웹앱입니다. 버블 또한 progressior.com이라는 홈페이지를 이용해서 웹앱으로의 변환이 가능합니다.



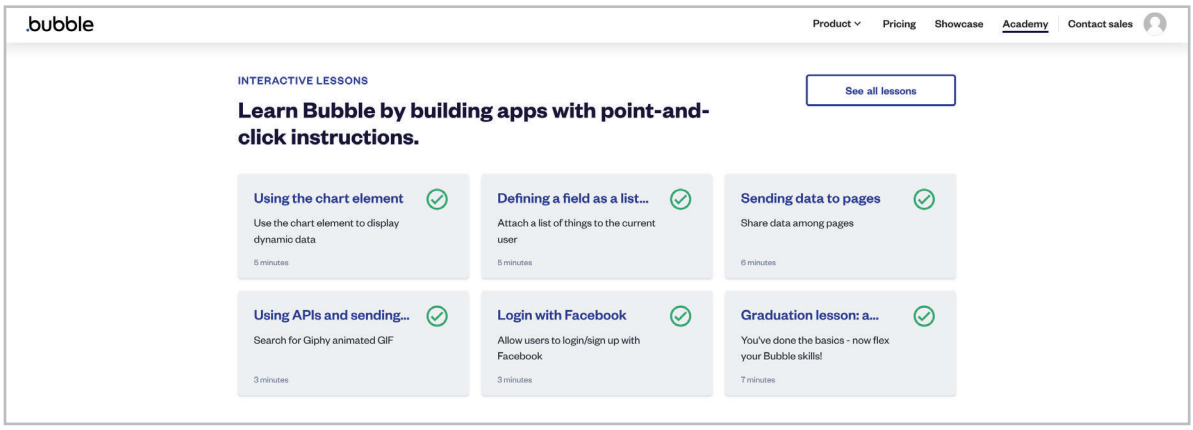
서비스 이용법은 Natively와 유사하니 네이티브 앱이 아닌 PWA 웹앱에 관심이 있다면 시도해 보는 것도 좋습니다.

PART 12

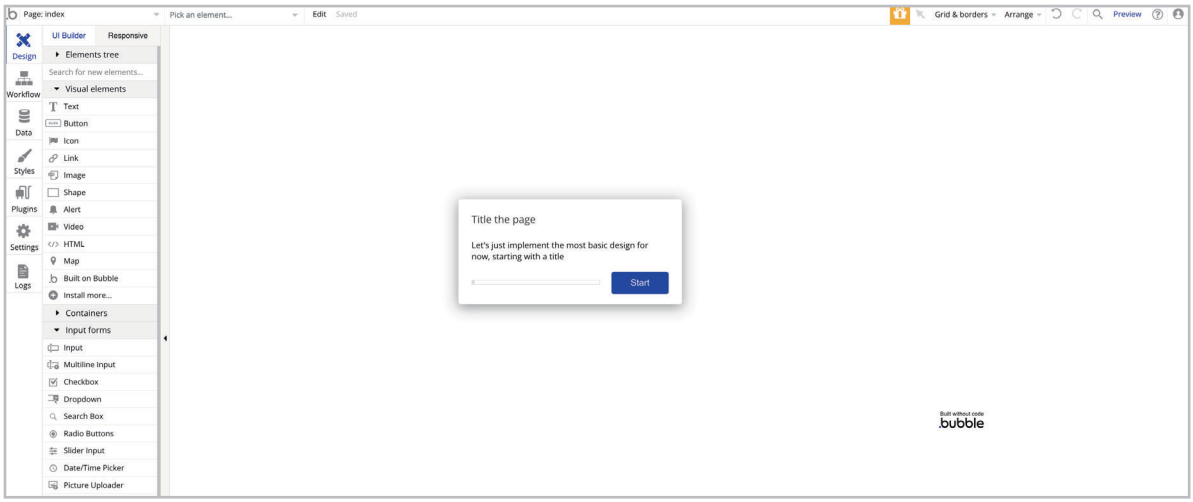
나만의 To do 리스트 직접 만들어보기

마지막으로 지금까지 배웠던 버블을 To do 앱을 함께 만들어 보면서 정리 및 실습을 진행해 보겠습니다.

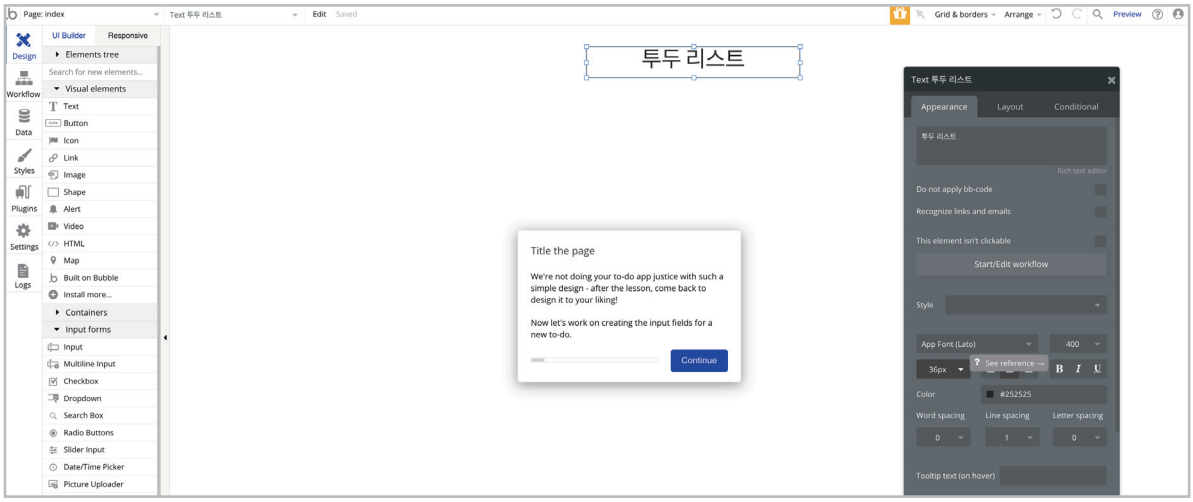
아래의 To do 리스트 실습은 버블 홈페이지의 우측 상단의 Academy > Interactive Lesson에 접속하면 같이 따라해 볼 수 있습니다.



추가로 나만의 To do 리스트 만들기는 Interactive Lesson의 마지막 lesson입니다.

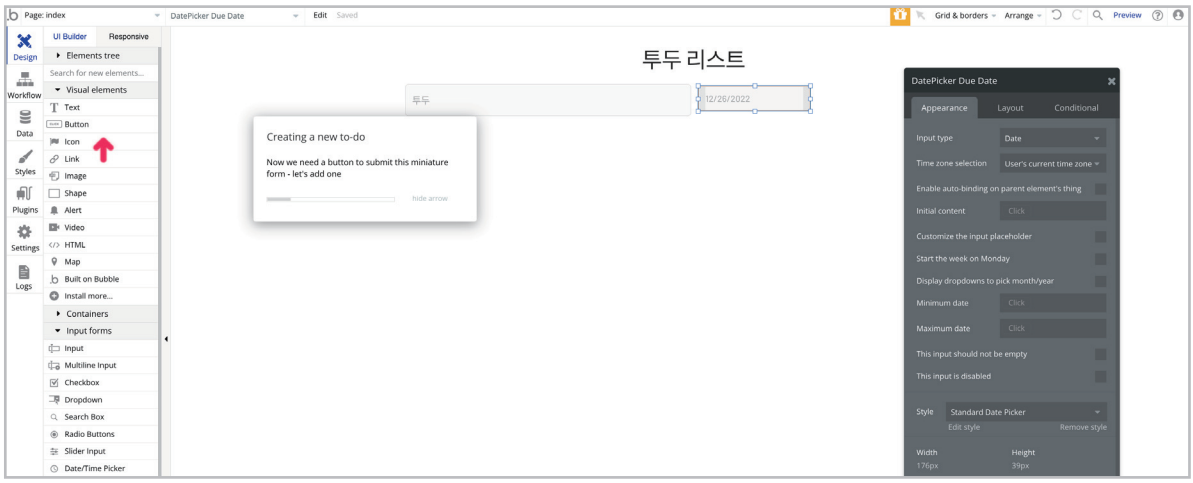


처음에는 To do 리스트의 제목을 위해 타이틀을 추가해 줍니다.

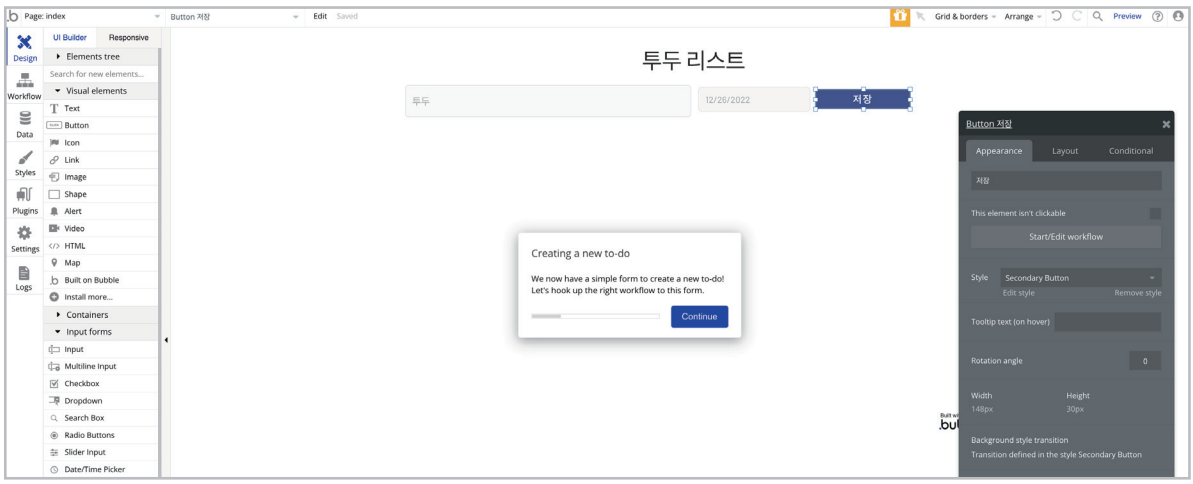


To do 리스트에는 해야 할 일을 기록할 수 있어야 합니다. 그러므로 할 일을 적는 입력란을 추가합니다.

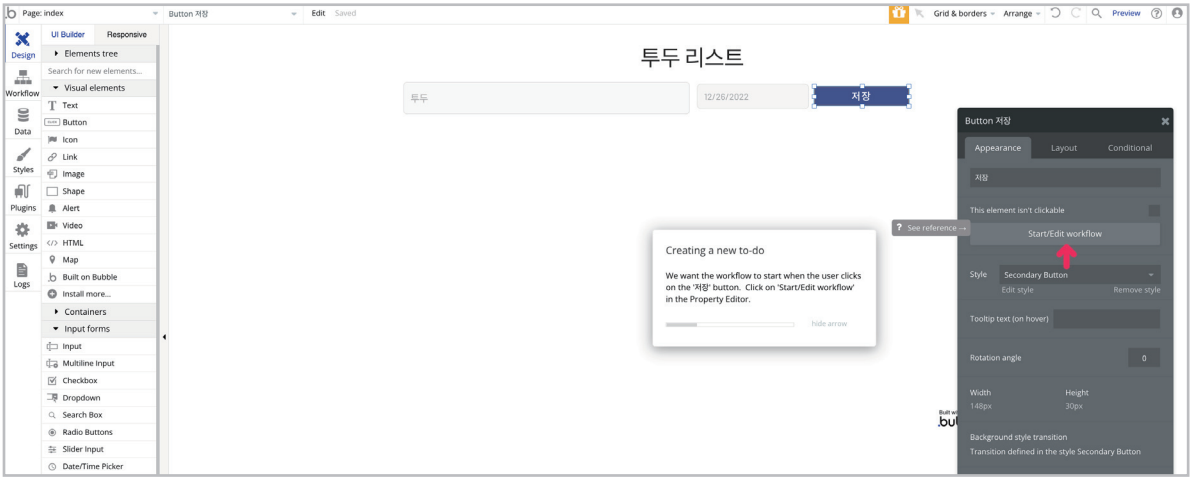
할 일을 언제까지 마쳐야 하는지에 대한 정보도 입력할 수 있게 해줍니다. 이를 위해 할 일의 마감일을 지정해 줄 날짜 입력기도 옆에 추가합니다.



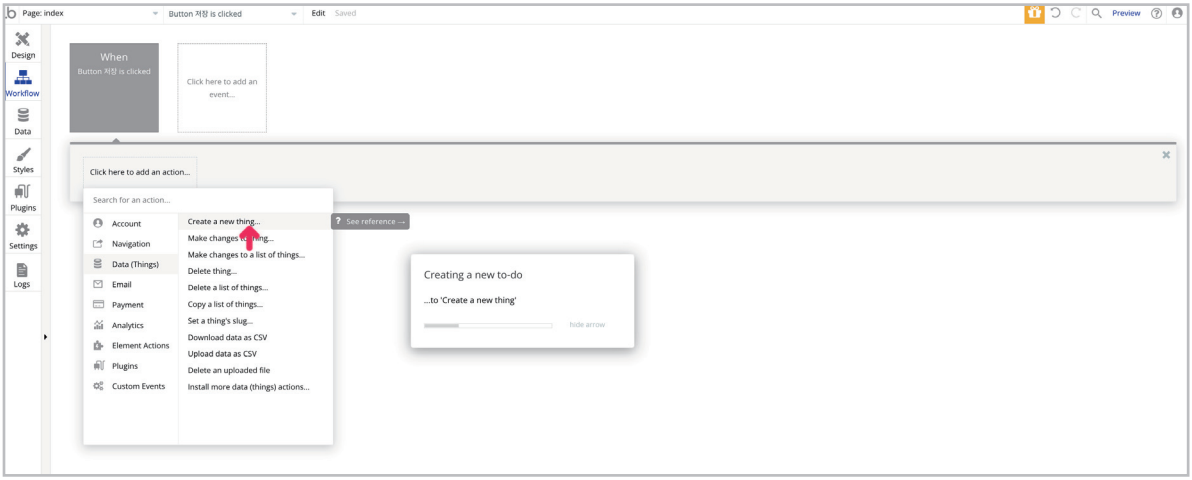
이제 작성한 할 일을 저장할 수 있도록 저장 버튼을 만듭니다.



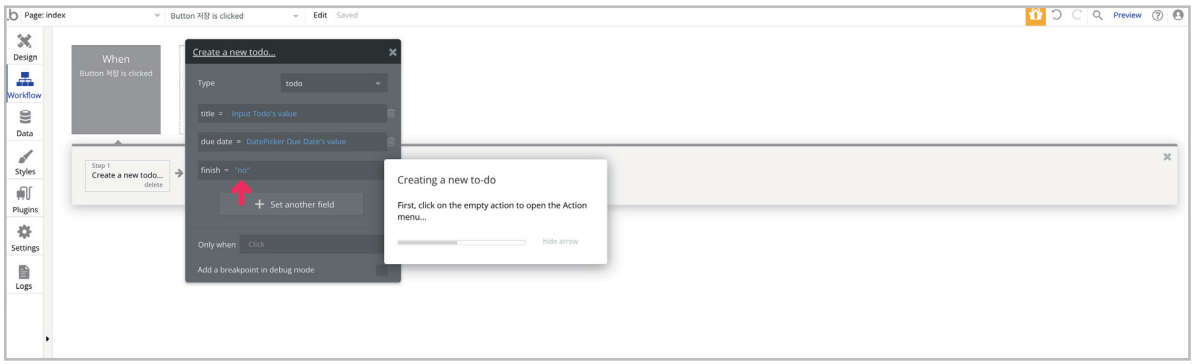
화면을 그렸으면 이제 기능을 넣어야 할 차례입니다. 저장 버튼을 눌렀을 때 할 일이 만들어지는 워크플로우를 생성합니다.



Start/Edit workflow를 눌러 저장에 대한 워크플로우를 생성해줍니다.

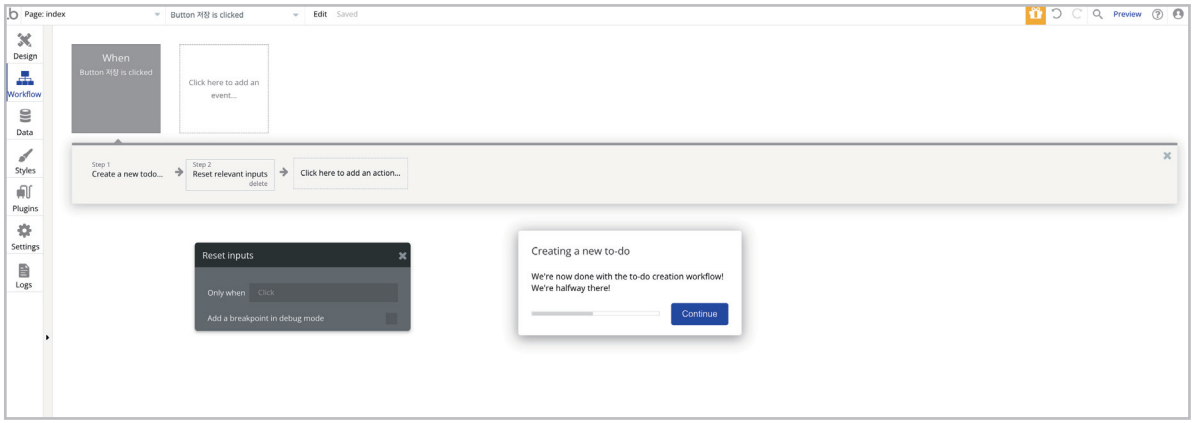


데이터 타입은 To do, 안의 필드로는 내용을 담는 title / 마감일의 due date / 완료됨을 표시하는 finish를 생성하고 각각의 필드에 알맞은 입력창의 값을 넣어줍니다. 이때 완료됨은 입력으로 받지는 않았지만, 투두 리스트의 특성상, 미완료된 할 일을 완료될 일로 만드는 게 목적이기 때문에 넣어줍니다. 추가로 처음 할 일이 생성되었을 때 미완료 상태여야 하기 때문에 필드의 데이터 타입을 yes/no로 정의하고 초기값을 no로 지정해줍니다.

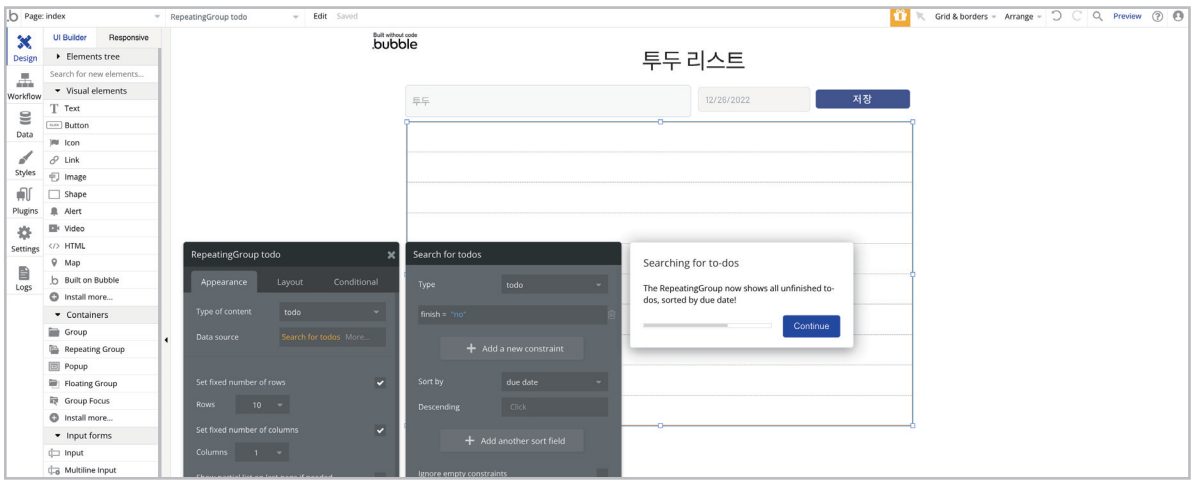


마지막으로 입력된 뒤에 입력창의 값들을 초기화해 줍니다.

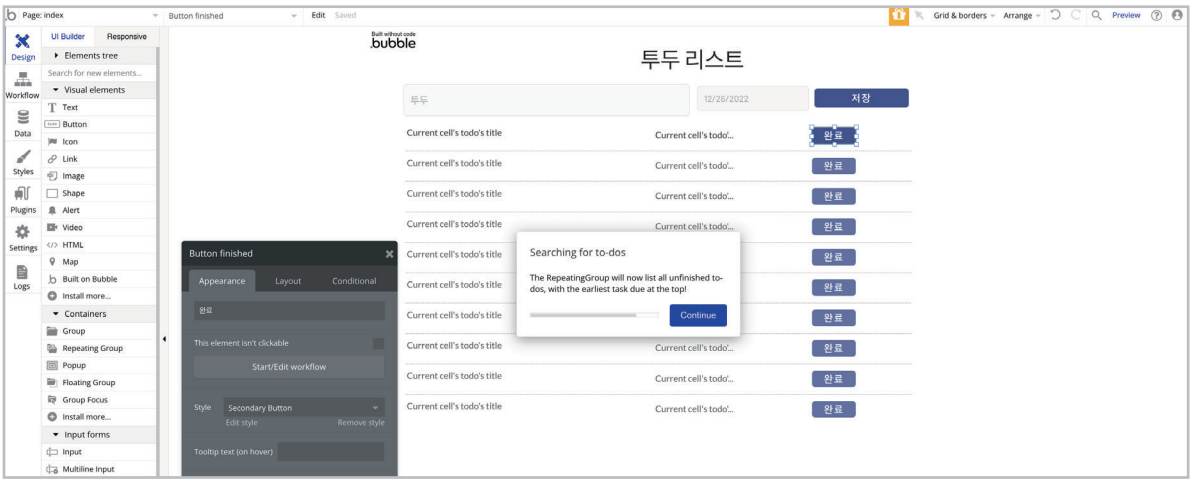
이 과정은 사용자가 버튼을 눌렀을 때 입력한 창들이 초기화가 안 된다면 정상적으로 제출이 되었는지 혼란스럽기 때문에 추가하는 단계입니다. 기능적으로 보면 큰 중요성이 없는 것 같지만, 사용자 경험에 있어서는 중요한 세부사항이므로 놓치지 않게 주의해야 합니다.



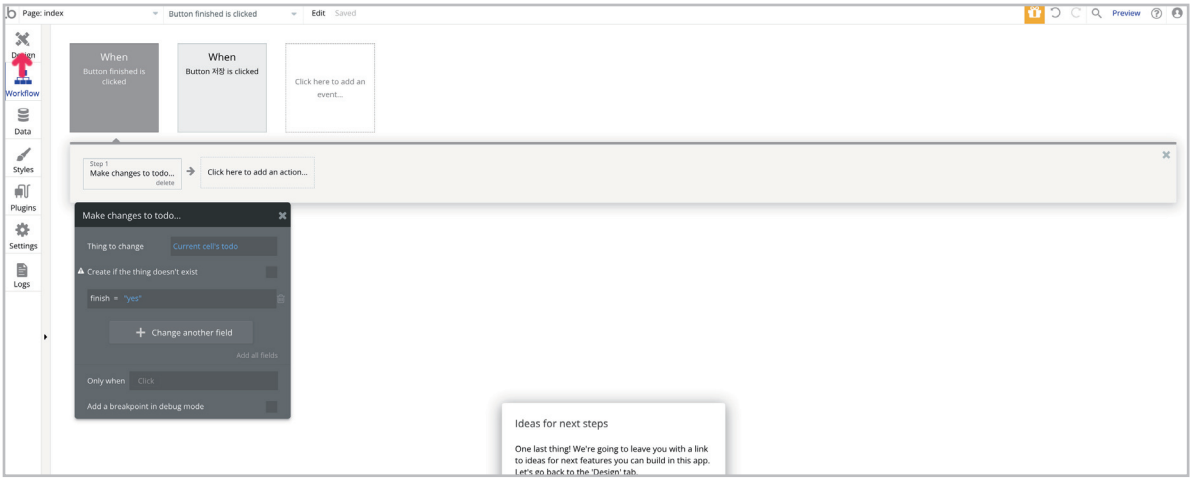
할 일을 생성하는 기능을 만들었으므로 이제는 생성한 할 일을 목록으로 볼 수 있도록 해보겠습니다. 할 일 목록을 화면에 반복그룹을 이용해서 그려줍니다.



그리고 그 안에 할 일 제목, 마감일, 완료 버튼을 차례로 놓아줍니다.



완료 버튼도 눌렀을 때 기능을 넣어봅니다. 버튼을 눌렀을 때 해당 할 일의 완료 상태가 yes로 변경되는 액션을 추가해줍니다.



이제 To do 리스트가 모두 만들어졌습니다.

이렇게 버블로 간단한 To do 리스트를 만들어 보았습니다. 이 실습을 기반으로 기능을 넓혀 가다보면 다양한 서비스들을 만드는 데 어려움이 없을 것입니다.

이제부터 여러분은 코딩 없이 원하는 서비스를 만들 수 있는 능력을 갖추게 되었습니다. 축하드립니다.