2022-05-19

당신은 UI 프레임워크가 필요하지 않습니다.

> 원문: https://www.smashingmagazine.com/2022/05/you-dont-need-ui-framework/ 번역을 기꺼이 허락해준 Josh Comeau님과 Smashingmagazine에 감사드립니다.

빠른 요약 개발자들은 종종 부트스트랩 또는 머터리얼 UI와 같은 UI 프레임워크에 손을 뻗어 많은 시간을 절약하고 전문적인 앱을 빠르게 만들 수 있기를 희망합니다. 하지만 기대와 달리, 대부분 이런 방법으로 해결되지 않습니다. 이것에 대해 함께 이야기 해봅시다.

때때로 사람들은 제가 추천하는 UI 프레임워크가 뭔지 물어보곤 합니다. 여기서 "UI 프레임워크"는 스타일의 UI 컴포넌트를 제공하는 데 중점을 둔 써드-파티 패키지를 의미합니다. 여기에는 부트스트랩 같은 CSS 프레임워크뿐만 아니라 머터리얼 UI 또는 앤트 디자인과 같은 JS 컴포넌트 라이브러리가 포함됩니다.

이 질문에 대한 나의 대답은 사람들의 허를 찌르는 경향이 있습니다. 저는 그것들을 사용하지 않으며, 대부분의 사용자를 마주하는 제품에 사용되지 않아도 된다고 생각합니다. 😅

분명히 말씀드리면, 저는 이러한 도구에 대해 반대하지 않으며, 이러한 도구에 대한 몇가지 유효한 사용 사례가 있다고 생각합니다. 하지만 저는 많은 개발자들이 그들이 해결할 문제에 대해, 혹은 얼마나 쉽게 애플리케이션을 만들 수 있는지에 대해 비현실적인 기대를 가지고 이러한 도구에 손을 뻗치는 것을 보아왔습니다.

이 글에서는 왜 이러한 도구가 필요하지 않은지 이유를 설명하려고 합니다. 또한 디자인 배경 없이 전문적으로 보이는 애플리케이션을 구축하기 위한 몇 가지 전략을 공유하겠습니다.

UI 프레임워크의 매력

개발자들이 UI 프레임워크에 손을 뻗는 이유는 많습니다. 제가 본 가장 일반적인 세 가지 이유는 다음과 같습니다.

1. 앱/사이트가 세련되고 전문적으로 보이기를 원하며, 이러한 도구들은 멋지게 설계된 UI 컴포넌트를 제공합니다.
2. 처음부터 모든 것을 구축하는 데 많은 시간을 들이지 않고도 신속하게 무언가를 시작하고 실행하기를 원합니다.
3. 많은 UI 컴포넌트(모달, 드롭다운, 툴팁 등)를 올바르게 동작하도록 만드는것이 정말 어렵다는 것을 알고 있으며, 특히 접근성 측면에서 어렵다는 것을 알고 있기 때문에 올바르게 동작한다고 보장되는 라이브러리를 사용합니다.

이런 것들은 전적으로 합리적인 것들이고, 저는 이 문제들에 대한 해결책을 찾는 매력을 확실하게 알고 있습니다. 하지만 어떤 경우에는 기대와 현실이 다르다고 생각합니다. 다른 사람들은, 그 일을 위한 더 나은 도구들이 있다고 생각합니다.

시작해봅시다.

전문적인 디자인

이 첫 번째 이유가 가장 일반적일 수 있습니다. 무언가를 만들고 싶지만 디자인 경력이 없는 개발자들이 많습니다. 디자인 방법에 대해 배우는 데 몇 년이 걸리는 대신, 아름답게 디자인된 컴포넌트를 즉시 제공하는 써드-파티 패키지를 사용하면 어떨까요?

제 생각에 문제는 디자인은 멋지게 보이는 것 이상의 의미가 있다는 것입니다.

조금 전에 레고 닌텐도 엔터테인먼트 시스템을 선물로 받았습니다.

> 역주: 아래 닌텐도 키트로 상단 TV 모델을 만든 모습

닌텐도 엔터테인먼트 시스템을 만드는 것은 **정말로** 재밌었습니다.

하지만 중요한 건, 제가 이 모델을 만들 수 있었던 이유는 200페이지 분량의 책자와 함께 나왔기 때문입니다. 각각의 블록을 어디에 놓을지 정확히 말해주는 책이죠.

모든 재료을 제공받았지만 지침이 없다면 NES는 이보다 **훨씬 더 나빠** 보일 것입니다. 고급 벽돌을 가지고 있는 것만으로는 충분하지 않고, 어떻게 사용하는지도 알아야 합니다.

컴포넌트 라이브러리는 멋진 단추, 날짜 선택기 및 페이지 지정 위젯을 제공할 수 있지만 *그것들을 조립하는 것은 여전히 여러분의 일입니다.*

머터리얼 디자인과 같은 설계 시스템의 블록은 재능 있는 설계 팀에 의해 만들어졌습니다. 그 팀은 그 프레임워크를 이해하고, 그 요소들을 아름다운 인터페이스로 조립할 수 있는 기술을 가지고 있습니다. 동일한 요소에 접근할 수는 있지만, 그렇다고 자동적으로 동일한 결과를 얻을 수 있는 것은 아닙니다.

한 디자이너가 머티리얼 디자인 앱을 멋지게 만들 수 있는 것은 구글뿐이라고 말한 것이 기억납니다. Android 앱 스토어를 보면 전문적으로 설계된 동일한 컴포넌트를 사용하지만 전혀 전문적으로 보이지 않는 써드-파티 앱으로 가득 차 있습니다.

> (역주) 안드로이드 앱들은 공식적으로 구글에서 머터리얼 디자인을 제공하고 있음 (https://material.io/develop/android)

좋은 디자인에는 균형, 간격, 일관성과 같은 무형과 같은 많은 측면이 있습니다. 컴포넌트 라이브러리를 효과적으로 사용하려면 컴포넌트를 만든 디자이너의 입장에서 생각해 보고 어떻게 컴포넌트를 배치해야 하는지 이해해야 합니다.

또한 아무리 포괄적인 라이브러리라도 필요한 모든 요소를 갖추지는 못할 것입니다. 모든 앱과 웹사이트는 독특하며, 항상 특별한 요구사항이 있습니다. 기존 타사 설계 시스템과 "혼합"하는 새로운 컴포넌트를 만드는 것은 *정말 어렵습니다*.

*불가능*하다고 생각하지 않습니다. 써드-파티 스타일을 가진 전문적으로 보이는 앱도 있을 것입니다. 하지만 만약 여러분이 멋지게 표현할 수 있었다면, 여러분은 아마도 꽤 중요한 디자인 감각을 가지고 있는 것이고, 애초에 이러한 도구가 필요하지 않을 것입니다.

저는 어떠한 디자인 직관도 없이 전문적으로 보이는 프로젝트를 시작하고 싶어하는 개발자들을 이해합니다... 하지만 내가 본 바로는 보통 그렇게 되지 않습니다.

시간을 절약한다는 점

다음 이유는 UI 프레임워크가 시간을 절약하는 데 도움이 된다는 점이었습니다. 전체 컴포넌트 라이브러리를 처음부터 구축하는 것은 중요한 작업이며, UI 프레임워크에 의존한다면 건너뛸 수 있습니다.

일부는 사실이지만, 제가 본 바로는 이는 종종 토끼와 거북이 상황이었습니다.

저는 몇 년 동안 콘코디아 대학에서 부트캠프 학생들을 대상으로 웹 개발 기초를 가르쳤습니다. 그 프로그램은 2주간의 개인 프로젝트로 마무리됩니다. 학생들은 무엇을 만들지 결정하고, 그것을 하는 것은 그들에게 달려있다. 강사로서, 저는 질문에 대답하고 혼란스럽지 않게 도와주곤 했습니다.

부트스트랩이나 머티리얼 UI와 같은 UI 프레임워크를 선택하는 학생들은 처음 며칠 동안은 빠르게 작업을 시작하고 진행합니다. 하지만 시간이 지날수록, 그들은 수렁에 빠집니다. 그들이 필요로 하는 것과 컴포넌트 라이브러리가 제공하는 것 사이의 간극이 커집니다. 그리고 컴포넌트를 올바른 모양으로 변경하는 데 *너무 많은 시간*을 소비하게 됩니다.

한 학생이 오후 내내 그들의 내비게이션을 지원하기 위해 CSS 프레임워크의 원본을 수정하려고 했던 것이 기억납니다. 결국 학생은 써드-파티 컴포넌트를 폐기하기로 하고 10분 만에 스스로 대안을 마련했습니다.

자신만의 스타일을 쓰는 것은 저에게 쓰기 시험처럼 느껴집니다. 처음에는 조금 느리지만, 초기 노력이 결실을 맺습니다. 결국, 여러분은 많은 시간, 에너지를 절약하고 좌절감을 피할 수 있을 것입니다.

사용성과 접근성

제가 언급한 마지막 이유는 매우 타당합니다. 웹은 모달, 드롭다운 및 툴팁과 같은 것에 대해 매우 강력한 "표준 라이브러리"를 가지고 있지 않습니다. 마우스 사용자, 키보드 사용자 및 화면 판독기 사용자 모두에게 잘 작동하는 모달을 만들기는 믿기 어려울 정도로 어렵습니다.

UI 프레임워크는 사용성 및 접근성과 관련하여 모 아니면 도의 기록이 있습니다. 몇몇 라이브러리는 이런 점에서 꽤 좋은 면을 갖기도 하지만 대부분의 경우 후순위 입니다.

고맙게도 *여러 가지 스타일을 규정하지 않고* 사용성과 접근성에만 초점을 맞춘 또 다른 도구들이 있습니다.

다음은 가장 좋아하는 도구들입니다.

- **Reach UI** 리액트에 대한 접근성에 중점을 둔 기본 요소 집합입니다. 리액트 라우터와 리믹스의 공동 제작자인 라이언 플로렌스에 의해 제작되었습니다.
- **Headless UI** 리액트 및 뷰에 대한 스타일 없이 완벽하게 액세스할 수 있는 UI 구성 요소 집합입니다. 테일윈드 팀이 제작 및 유지보수합니다.
- **Radix Primitives** 리액트를 위한 스타일링되지 않은 접근성 중심 컴포넌트 집합입니다. 이 라이브러리에는 아주 다양한 컴포넌트가 포함되어 있고, 정말 멋진 것들이 많이 있습니다!
- **React ARIA** 리액트 훅 라이브러리는 처음부터 만드는 접근성을 고려한 컴포넌트를 만드는 데 사용할 수 있습니다.

참고: *이 목록은 리액트 중점으로 한 것임을 알고 있습니다. 앵귤러, 스벨트 및 기타 프레임워크에 대해 유사한 도구가 있을 수 있지만, 저는 이러한 커뮤니티에서 그렇게 적극적이지 않기 때문에 잘 모르겠습니다. 다른 것을알고 있는 경우 얼마든지 트위터에서 알려주세요)*

아무도 2022년에 처음부터 시작할 때 모달을 만들지 말아야 합니다. 하지만 그렇다고 해서 거대한 스타일이 포함된 UI 프레임워크가 필요한 것은 아닙니다! 가장 중요한 접근성 문제를 정확하게 해결하면서도 겉모습과 스타일에 관해서는 전혀 무관심할 수 있는 도구들이 있습니다.

반론

저는 이 주제에 대해 몇 년 동안 개발자들과 이야기를 해왔고, 꽤 설득력 있는 반론을 들었습니다.

익숙함

먼저, Daniel Schutzmith는 부트스트랩과 같은 "산업 표준" 도구들이 하나의 큰 장점, 즉 익숙함을 가지고 있다고 지적했습니다.

널리 알려진 도구를 사용할 때 새로운 개발자와 디자이너를 참여시키는 것이 더 쉽습니다. 새로운 팀원들은 커스텀 프레임워크의 세부 사항을 학습하는 데 많은 시간을 할애할 필요가 없으며, 처음부터 끝까지 해나갈 수 있습니다.

많은 단기/중기 프로젝트를 맡는 기관의 관점에서 보면 충분히 말이 됩니다. 모든 새로운 프로젝트를 처음부터 시작할 필요는 없습니다. 그리고 팀이 UI 프레임워크에 익숙해질수록 더 효과적으로 사용하는 법을 배웁니다.

에어전시 일을 많이 해본 적이 없어서 말하기가 어렵지만, 저는 제 경력의 대부분을 제품 회사에서 일하며 보냈습니다. 제가 일해본 곳 중 어느 곳에서도 서드파티 UI 프레임워크를 사용하지 않았습니다. 저희는 항상 내부에서 무언가를 만들어 사용했습니다. (예를 들어 Khan Academy에서 WonderBlocks 또는 Digital Ocean에서 Walrus).

내부 도구

내부 도구 또는 기타 비공유 프로젝트(예: 프로토타입)를 구축할 때 UI 프레임워크를 사용하는 것이 합리적일 수 있다고 생각합니다.

목표를 신속하게 설정하고 실행하는 데 있고 UI가 100% 프로페셔널할 필요가 없다면 여러 써드-파티 컴포넌트를 신속하게 배치하는 것이 시간 절약이 될 수 있다고 생각합니다.

테일윈드와 차크라 UI는 어떨까요?

따라서, 저는 테일윈드나 차크라 UI가 이와 같은 "UI 프레임워크" 범주에 속한다고 생각하지 *않습니다*.

테일윈드는 즉시 사용할 수 있는 컴포넌트를 제공하지 않지만 디자인 토큰을 제공합니다. Max Stoiber는 테일윈드가 개발자들에게 일련의 가드레일을 제공한다고 말하기도 했습니다. 효과적으로 사용하기 위해 여전히 디자인 직관이 필요하지만, 처음부터 무언가를 디자인하는 것만큼 위압적이지 않습니다.

차크라 UI는 즉시 사용할 수 있는 스타일 컴포넌트를 제공하지만, 그것들은 매우 작고 낮은 수준입니다. 대부분 플랫폼 기본 UI의 더 좋은 버전처럼 보일 뿐입니다.

저의 좋은 친구 에미는 챠크라 UI를 사용하는 것이 체크박스와 라디오 버튼과 같은 컴포넌트의 합리적인 기본값을 제공하기 때문에 사용하는 것을 좋아한다고 말했다. 그녀는 맞춤 제작의 함정을 피할 만큼 디자인에 능숙하지만 처음부터 전체 디자인 시스템을 만드는 데 자신이 없습니다. 이 도구는 그녀의 상황에 있는 누군가에게 완벽한 타협점입니다.

제 생각에 차이점은 이러한 솔루션들이 여러분을 위한 디자인을 해결한다고 주장하지 않는다는 것입니다. 올바른 방향으로 유도하는 데 도움이 되지만, 모든 것을 사용자 정의할 수 있고 특정 디자인 미학에 얽매이지 않도록 합니다.

제안하는 대안

전문적으로 보이는 웹 사이트와 응용 프로그램을 만들고 싶지만 디자인 배경이 없는 개인 개발자가 있다면 어떻게 해야 할까요?

몇 가지 제안이 있습니다.

디자인 직관을 키웁니다.

자, 나쁜 소식이 있습니다. 저는 여러분이 디자인 기초를 배우는데 어느정도 시간을 보내야 한다고 생각합니다.

이것은 아주 조금이나마 효과가 있는 것들 중 하나입니다. 여러분은 예술 학교에 갈 필요도 없고 새로운 기술을 배우는 데 몇 년을 바칠 필요도 없습니다. 디자인은 어렵지만, 우리는 세계적인 디자이너가 되기 위해 노력하지 않습니다. 우리의 목표는 훨씬 더 겸손하고, 여러분은 그것들이 얼마나 빨리 달성될 수 있는지 혹은 여러분이 이미 얼마나 멀리 갔는지에 놀랄지도 모릅니다!

디자인에 별로 관심이 없더라도 디자인 직관을 구축하는 것은 프런트엔드 개발자들에게 중요한 기술이라고 생각합니다. 믿기 힘들겠지만, *저희는 작업 중에 끊임없이 디자인을 결정하고 있습니다*. 심지어 가장 상세하고 정확한 목업도 여전히 중요한 맥락의 *톤*이 빠져 있습니다.

예를 들어 다음과 같습니다.

- 운이 좋으면 3개의 화면 크기가 주어질 수도 있지만, UI가 해당 화면 크기 *사이에서* 어떻게 동작해야 하는지 결정하는 것은 우리에게 달려 있습니다. - 데이터는 모의실험에서 보이는 것처럼 깨끗하지 않으며 긴 이름, 누락된 데이터 등을 어떻게 처리할지를 결정해야 합니다. - 로딩, 비어 있음 및 오류 상태는 목업에서 누락되는 경우가 많습니다.

개발자로서 저의 초능력 중 하나는 디자인에 명확하게 명시되지 않은 상황에 부딪혔을 때 무엇을 해야 하는지 알아낼 수 있는 충분한 디자인 감각을 가지고 있다는 것입니다. 저는 막히는 대신 디자이너가 제 질문에 대답하기를 기다리는 동안 제 직관에 의존할 수 있습니다. 항상 정답을 맞추지는 못하지만, 대개는 그럴 것입니다(그리고 그렇지 않을 때는 디자인 직관력을 향상시킬 수 있는 또 다른 기회입니다!).

설계 직관을 어떻게 키우나요?

제품/디자인 팀과 함께 일하면 엄청난 리소스를 사용할 수 있습니다! 그들이 생산하는 디자인에 대해 비판적으로 생각해 보세요. 많은 질문을 하십시오. 대부분의 디자이너는 사물이 어떻게 구성되고 왜 이러한 결정을 내렸는지 이해하는 데 도움을 줄 것입니다. 엔지니어링 과제로 취급하십시오. 좋은 설계로 이어지는 시스템과 프로세스를 배울 수 있습니다.

저는 얼마 전에 "제품 및 디자인과의 효과적인 협업"이라는 블로그 글을 썼습니다. 이러한 아이디어들 중 일부에 대해 좀 더 깊이 파고듭니다.

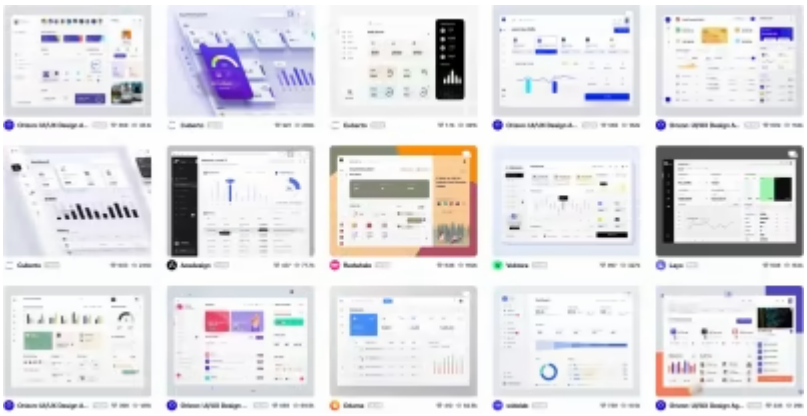
디자이너와 함께 작업하지 않거나(또는 디자이너 친구가 있는 경우), 여러분이 매일 사용하는 제품을 리버스 엔지니어링할 수 있습니다. 사물의 간격과 사용되는 글꼴 크기를 메모합니다. 비판적인 안목으로 보면, 패턴이 보이기 시작할 것입니다.

훔치기

좋습니다. 그래서 예민한 디자인 본능을 가지고도 처음부터 디자인을 생각해 내기는 여전히 *정말* 어렵습니다. 그러니까 그러지 맙시다.

대신, 우리가 만들려고 하는 것과 비슷한 전문적인 디자인을 찾아봅시다. 디자이너 커뮤니티 사이트(https://dribbble.com/) 또는 behance)에서 검색하거나 awwards)와 같은 아카이브를 사용할 수 있습니다.

예를 들어, Uber-for-dogs 스타트업을 구축하고 드라이버 대시보드를 설계한다고 가정해 보겠습니다. Dribble에서 "대시보드"를 검색하면 다음과 같은 흥미로운 디자인이 나타납니다.



드리블은 매우 "designery"를 왜곡하는 경향이 있으므로 영감을 얻기 위해 실제 제품을 사용하는 것이 좋습니다. 그것도 돼요!

> (역주) designery: 제품 디자인과 같은 전통적인 디자인 맥락

비결은 *여러 개의* 소스를 사용하는 것입니다. 만약 당신이 디자인을 100% 도용한다면, 그것은 표절이고 나쁜 형태입니다. 사람들은 눈치채고, 문제를 일으킬 것입니다.

대신, 우리는 독특한 것을 만들기 위해 3개 또는 4개의 디자인을 함께 섞을 수 있습니다. 예를 들어, 한 사이트에서는 색 구성표를, 다른 사이트에서는 일반 레이아웃과 간격, 그리고 세 번째 사이트에서는 타이포그래피 스타일을 선택할 수 있습니다!

제가 실제 디자이너들에게 이 전략을 언급했을 때, 웃으면서 모두가 하는 일이라고 말합니다. 프로그래머들이 절반의 시간을 구글링으로 보내는 "농담"과 같은 그들만의 버전이라고 생각합니다.

이 전략은 마치 인생 해킹처럼 느껴집니다. *노력하지 않은 것*도 아니고 디자인도 필요합니다. 영감을 얻기 위해 사용하는 디자인은 현재 구축 중인 것과 100% 일치하지 않으며, 부족한 부분을 채우기 위해 직관을 사용해야 합니다. 하지만 이 방법은 지금까지 디자인 배경 없이 전문적으로 보이는 디자인을 고안해낸 방법 중 *가장* 빠릅니다.

> (역주) 인생 해킹: 생산성과 효율성을 높이는 트릭, 지름길, 기술 또는 참신한 방법

모든 것을 하나로 합치다

개발자로서, UI 프레임워크가 디자인에 대한 어떤 것도 배울 필요가 없다고 믿는 것은 솔깃할 수 있습니다. 불행하게도 보통 그렇게 되지 않으며 적어도 제가 본 바로는 아니었습니다.

좋은 소식이 있습니다. 디자이너 없이도 전문적으로 보이는 제품을 만들 수 있습니다! 몇 가지 고품질 디자인 레퍼런스와 약간의 디자인 직관력을 통해 제품이 합법적이고 "실제"로 느껴지는 "충분히" 임계치에 도달하는 것을 구축할 수 있습니다.

아직 많이 이야기하지 않은 측면이 하나 더 있습니다. **CSS**입니다.

많은 프런트엔드 개발자들이 CSS와 씨름하고 있습니다. 저도 고생했습니다. CSS는 기만적으로 복잡한 언어이고, 심지어 당신이 그 언어에 대해 수년간의 경험을 한 후에도 종종 일관성이 없고 좌절감을 느낄 수 있습니다.

이것은 제가 매우 열정적으로 느끼는 문제입니다. 저는 개발자들이 언어에 대한 자신감을 가질 수 있도록 돕기 위해 CSS 과정을 만들고 개발하는 데 작년 내내 집중했습니다.

JavaScript 개발자를 위한 CSS라고 합니다. 리액트 또는 앵귤러와 같은 JS 프레임워크를 사용하는 사람들을 위해 특별히 제작되었습니다. 이 과정은 CSS가 어떻게 작동하는지 직관적으로 이해할 수 있도록 강력한 정신 모델을 제공하는 데 중점을 둡니다.

만약 당신이 CSS가 예측불허라고 느낀다면, 꼭 확인해주길 바랍니다. 9000명 이상의 개발자가 이 과정을 거쳤으며 반응은 압도적으로 긍정적입니다.

자세한 내용은 css-for-js.dev를 참고하세요.