Understanding the Opportunity and Impact of Immersive Technologies

Introduction

What do you think of when you think of something immersive? Do you think of Facebook? Pinterest? YouTube? Is it enough to only have your interest immersed, for something to take to you another place? Perhaps you've had a time where you've been so engrossed in a book that you've felt transported to another place. Maybe you've played a video game where you truly felt like you were a character - in another world.

If we look at the concept of immersion, we see a series of principles that make us feel transported into a deep interest or sense of belonging to an environment, or a topic, or a world. When we apply this to software, we get spatial computing.

Broadly speaking, spatial computing today refers to virtual and augmented reality technologies. These are specific types of hardware and software applications that aim to transport the user to another world, or to integrate digital content into the environment around us. It's the idea that the form factors of computers are changing, and that how we as humans observe, create, and interact with computers and data is changing with them. Through these new hardware devices, the world around us ceases to be constrained to physical dimensions and limits, and we are able to enter new areas of human-computer interactivity. When we consider that software is a set of increasingly powerful tools at our disposal, human capability and opportunity expands.

Just within the past decade, we have seen the ways that technology can exponentially grant new access and capabilities to solutions that used to cost tens of thousands of dollars. We have been granted the gift of the world's collective knowledge in our pocket, and with the advent of immersive virtual and augmented reality technologies, we are entering a new era where we aren't just consuming this knowledge, but are becoming able to collaborate and generate entirely new human experiences based in that collective knowledge.

The ability to share a space with others regardless of their geographic location is in and of itself essentially a superpower. When you imagine being in a world where that ability to exist and communicate in an embodied way outside of the limitations of geography is available to the 7.5 billion people who live on this planet, the opportunity space for immersive technologies starts to become abundantly clear. It's computers allowing us the capabilities of breaking the laws of physics and geography and going beyond what a single human mind can accomplish.

But this wouldn't be a particularly compelling keynote if I told you that was the entirety of the potential that immersive technologies offer. I’m going to tell you about how VR and AR technology is already amplifying the human experience and changing how we think about our relationships with software, data, and with each other. I'm going to challenge the assumptions that you have about these technologies and explore the ways the the web community is laying

the groundwork for scalable, accessible virtual and augmented reality, and I'm going to show you demos because nothing has ever gone wrong when you try to do it live on stage!

Because I've mentioned it a few times already, and it is more directly related to what I work on every day, I'll start of with talking about virtual reality. How many of you would consider yourself relatively familiar with today's virtual reality hardware?

Introducing VR hardware

Because there are a number of terms that we can use to describe different hardware with varying types and depth of functionality, we'll start off by getting on the same page with what some of these terms mean, and get into an overview of some of the defining characteristics of what the different devices entail. I'll add a disclaimer that these definitions may vary to differing degrees depending on who you ask, but it should provide a relatively good starting point for discussion about today's immersive computing devices. You can follow along with these and other terms on the guide I put together, which is available online at the link on this slide.

Today, the terms 'virtual reality', shortened to 'VR', generally refer to specialized pieces of hardware called 'head mounted displays', or HMDs, which are worn by the user on their head and provide a display that stereoscopically draws a 3D environment onto the device display in order to provide an illusion to the user that they are present within an entirely new environment that displaces the room they are actually in. One can think of a virtual reality context as one that aims to completely replace the user in a setting that is unrelated to the space they are presently a part of, and most of the time will not rely on the physical environment of the user to dictate or drive the experience that the user participates in from within the headset. There are quite a few hardware manufacturers working on headsets today - Facebook's Oculus, HTC, Google, and Valve are just a few.

While not as common as HMD-driven VR, another type of virtual reality experience that you may experience today is in the form of a 'CAVE', or a series of displays that surround a user and project the virtual world onto screens around the user. For today's talk, we'll focus primarily on head-mounted VR as we discuss specific headsets and implementations, but many of the principles related to creating 3D environments and spaces for HMD-driven VR could be adapted to CAVE style VR systems as well.

Within the head-mounted device ecosystem for today's VR systems, there are a few different ways of characterizing capabilities for the devices to help provide different functionality for experiences to take advantage of to create varying degrees of immersion. One such descriptor is to explain the 'degrees of freedom' that a headset might provide. At a minimum, today's headsets are capable of tracking a user's head rotation as they are wearing an HMD - this is what allows the headset to know how a user is looking around the 3D environment and drive the camera within the virtual world accordingly, to align with the viewer's motion to a new perspective. .

Headsets that track rotational elements only within a space are referred to as having ‘three degrees of freedom’ (3DoF), and virtual reality content for 3DoF headsets must be designed in such a way that it does not require the user to physically move their position. Headsets with higher tracking capabilities are able to track the position of a user within a physical space, supporting features like walking around, jumping, or squatting down, and are described as having six degrees of freedom.

Another set of descriptors that can be used to describe VR headsets today separate headsets out into those that are ‘tethered’, or connected to and powered by an external computer, and those that are ‘untethered’, which are powered by hardware present within the headset itself and don’t require a cable.

This chart shows a spectrum of tethered vs. untethered headsets mapped against their degrees of freedom. Special types of cameras, hand controllers, and wireless capabilities are all elements that play into how headsets are able to surface these different elements to a developer and the experience they create. Some headsets, like the Oculus Rift, Playstation VR, and the HTC Vive, have external cameras that track the headset through infrared sensors. Others, like the Oculus Quest, have specialized depth cameras placed on the front of the headset to provide tracking of the room by reconstructing a map of the physical environment and tracking the user by looking from the inside of the device outside into the room. Headsets like the Gear VR, which is powered by a Samsung phone placed into a specialized headset case, or the Oculus Go, which is a “standalone” headset that is entirely self-contained, do not provide external tracking and are examples of headsets with 3 degrees of freedom.

Because six degree of freedom headsets captures and supports more elements of the user's physical input, they tend to provider richer interactions for a user within an experience, but good content can be written for both types of headsets as long as the device's capabilities are taken into consideration as the application is designed. 6DoF headsets are generally more expensive than 3Dof headsets, due to the additional sensors, so cost is also a factor in headset adoption.

While virtual reality is the concept of using a device to create a digital environment that replaces the physical world, augmented reality, in its various forms, is the idea that we can use technology to **add** digital information to the physical world in order to supplement what we see around us. You can think of virtual and augmented reality as two points on a spectrum, varying in how much digital information replaces the physical world, with additional points along the way to indicate differing functionality. The concept of “mixed reality”, as an example, is a descriptor for technologies that behave as a type of AR device, but with the addition of hardware capabilities that grant the device awareness of the physical environment to create a hybrid application.

Augmented reality devices can, like VR headsets, be worn, but more recently, the concept of “magic window” AR has created an opportunity where mobile phone cameras can be used to

display content on top of the feed on a mobile device. Digital content can be applied in several different ways to create augmented reality applications: through depth maps, a virtual object can be pinned on top of specific points in the physical world via a mesh mapping of the environment or object; an image or pattern such as a QR code can act as a marker that triggers a digital element, or, as is seen in the game Pokemon Go, the virtual object can simply be laid over a camera feed to give an appearance of existing within the physical world.

Designing Content (15 min)

Within all of these modalities of viewing immersive content, there are also other capabilities that improve the immersion of experiencing 3D content. Positional audio allows applications to create realistic soundscapes. Interactivity through touch, voice, and gestures all provide natural modes of engaging with the virtual content placed in the world without using keyboards, mice, and game controllers, although all of those input methods are equally valid in their use in different experiences. Many of the six degree of freedom headsets have specialized controllers, one for each hand, that allows users to grab and manipulate virtual objects. Mixed reality headsets often rely on a combination of hand gestures and small, clicker like controllers, and an ecosystem around wearable controllers like wristbands and rings are also available for customized input mechanics.

Development across the immersive technology spectrum requires a conceptual understanding of 3D environments, which isn't as intimidating as it may seem on the surface - we live in a three dimensional world, and can apply the existing understand that we have of our physical environment to begin to think about and design for spatial computing paradigms. Instead of drawing buttons, text, and visual elements on a flat surface, with virtual and augmented reality, we place these items in the space around the viewer, and, in the case of a full virtual reality experience, alongside mesh objects that are drawn from 3D models to create a digital environment. You can prototype 3D experiences in a number of ways that don't even involve VR or AR devices - one of my favorite low Fidelity ways to prototype for VR is through sticky notes that I place around the room.

In most cases, developing for virtual reality involves first thinking about the experience from the perspective of the camera, which acts as the user's origin and is the viewpoint from which the virtual scene is rendered, and the environment. A canvas draws the contents of the virtual environment twice, once per eye, and renders at a frequency of 60 to 120 Hz, depending on the device. User interface elements can be integrated into the scene, or exist as a layer that is drawn above it.

At the device level, most of the core APIs for interacting with VR devices interface through graphics libraries that provides the mechanics for drawing visuals onto the display and transmitting information about the hardware state back to the application. In the case of VR, the input that drives the information about the state of the experience is derived from our physical

bodies, which provides the illusion that we are able to step into the application running on the device.

VR on the Web

So… why bring the browser into all of this? I'm not sure that this is an audience I'll have to try too hard to convince - you're all here because you believe in the power of the web, right? Everyone in the audience here is looking to push the web to its full potential and you know the strengths and opportunities that web technologies create. Accessibility. Cross platform capabilities. Standards. Connectivity. When you consider applying the power of the web to new computing and interaction models, these benefits push VR and AR forward to compete in the arenas of accessibility and access, and changes the narrative of who can build for and use these technologies.

Access to experiences is one core advantage that the web brings to immersive technology. Almost half of the world's population is online now, but access to data is not equally distributed, and access to high end computers that are required to run the latest, highest resolution, expensive virtual reality hardware isn't evenly distributed either. Developing virtual reality content for the browser means that we begin to chip away at the elite, inaccessible restrictions to adoption that have kept the VR industry from accelerating at a more rapid pace. Virtual and augmented reality won't succeed if everyone has to be an expert games programmer to create content.

Luckily, we're starting to see that change. I'm incredibly fortunate to work on immersive technologies at a company like Mozilla, where the web is at the core of what we do, and when it comes to virtual and augmented reality, this industry is still just getting started.

My journey with learning Javascript began back in 2014, when I first started studying the ecosystem around virtual reality development. At the time, Mozilla and Google were just beginning to explore what a standard set of browser-level APIs that surfaced virtual reality devices like the HTC Vive, Oculus Rift, Samsung’s Gear VR, and Google Cardboard headsets could look like. The industry was already differentiating between more accessible, mobile-phone powered headsets and ones that relied on desktop-grade graphics cards, and the WebVR initiative was one of the first forays into a unified development platform for this generation of immersive displays. I actually learned Javascript primarily to create webvr content, so you may find me sorely lacking in knowledge of popular frameworks and libraries outside of what exist for VR.

Back in the 90’s, VRML attempted to lay the foundation for virtual reality on the web with a file format for specifying 3D content accessible by browsers, but at the time, the ecosystem around devices and general availability of the internet was significantly different than today. The combination of today’s computing devices and access to data has changed the landscape from past attempts at bringing virtual and augmented reality to the mainstream, and we’ll explore how

this style of computing equips us to innovate across areas of education, social connection, productivity, enterprise, and more.

Browsers

Today, browsers are moving towards a unified set of APIs called the WebXR API to surface immersive device capabilities to websites across virtual, mixed, and augmented reality devices. Over the past couple of years, Firefox and Chromium have both implemented the predecessor of the WebXR API, the WebVR API (yeah, I know, we have to get some things figured out about how we're naming things in this industry) and now, the W3C has published the draft specifications for the unified device API that will cover the different capabilities of immersive devices and how they're surfaced to web developers.

Mozilla, Google, and Microsoft have all committed support to the proposed WebXR standards, and their browsers have varying degrees of support already for today's virtual and augmented reality hardware ecosystem. And while we wait for the standards to land in that ideal world where every browser supports every device responsively, there are Javascript libraries, frameworks, and polyfills that fill in the gaps.

There is also the question of what it means to be a browser in VR or AR contexts. The browsers of today have a generally well-understood mission statement: to deliver content through urls. In the past several years, we've seen a greater focus and opportunity on creating responsive applications with a two dimensional layout that looks good on screens of many sizes - what does it look like when the surface space is the entire world around a viewer? How do sites seamlessly move between a flat viewing mode and a fully immersive experience for a VR device, or become contextually aware of what surfaces are available to 3D content in an AR headset? What does a VR hyperlink look like, or tabs? These are just some of the questions that the immersive web is exploring today.

Firefox Reality, Oculus Browser, and Samsung Internet are three browsers that are at the forefront of exploring the answers to those questions, but these are the early days. The opportunity to work towards evolving standards is one that I find quite invigorating, because things are so frequently changing to respond to new advancements in the technology.

With that all laid out, let's get into some examples of VR content on the web!

Demos (30 min)

Almost four years ago to the day, I stood on stage at JSConf Budapest and presented my first WebVR project - a rainbow, swirling environment that ran only on Firefox Nightly with several polyfills holding things together to display on my Oculus Developer Kit. It ran at about ten frames per second, which is around 65 fewer frames than necessary for a smooth VR experience.

As I was preparing for this talk, I decided to revisit that experience and see how different it was to develop the exact same experience today, with all of the advancements in the WebVR space over the last four years. I actually ended up re-implementing it, it's a very "hello world" level project for developing for VR on the web, two different ways, which I talk through to give you a high level overview of just a few of the frameworks that are out there for 3D web development. These aren't the only frameworks out there, but they're the ones that I've used myself, and that said, I'll also make a point to include some resources that explore other tools out there that can be used to develop for VR or AR on the web.

I also want to pause here and give a quick disclaimer - if you're feeling intimidated at the thought of designing or programming 3D worlds, don't be! I won't say that it's painless, especially if you start to explore a lot of physically-driven interactions, but as human beings, we exist naturally in 3D environments and getting started will probably feel more intuitive than it seems as I stand up here talking about it. Immersive technology is very experiential - it makes so much more sense when you see it in action, especially when you're able to stand in the virtual space yourself.

The first thing that I'll show you is the three.js implementation of my first WebVR demo. This 3D scene is made up of a single, stationary camera and one single mesh that surrounds the entire camera. That mesh happens to be a giant, rainbow colored torus knot, and it happens to be spinning.

Let's look at the code of the three.js implementation of this first example. The project is hosted on Glitch at this url if you want to open it up on your phone and follow along there. Three.js is a 3D graphics library that uses a WebGL context to render the scene in a browser canvas, and provides a robust mechanic for creating environments that can be rendered and explored as part of your web page. Looking at the code here, you can see that there really isn't too much to it - it's sitting at just about fifty lines, including comments and our HTML.

We start off with defining a few variables that we'll use to reference different elements in our scene. The renderer is what will handle drawing our 3D objects into the browser canvas, and those 3D objects are stored in our scene. The geometry variable stores the base physical properties of the torus knot that we'll be adding to the scene, while the material stores - you guessed it, the object's material, and the mesh refers to both of those two components, the geometry and material together, to make a single object. Lastly, we store a reference to our camera, which is the area that we'll be viewing our scene from and passing information from our renderer to draw from.

In our window.onload function, we're calling two functions. The first initializes our scene so that it's set up for us to view it, and the second initializes the rendering and kicks off the animation loop.

In the function initScene(), we first create our core Three scene element, which acts as a container for everything else that we have in here. This is a particularly light scene, because it only has a single mesh in it, but we'll take a look at a more complex example in just a few minutes.

In 3D applications, objects can have a number of properties that give them certain behaviors or a specific visual appearance. In this example, we have a mesh with a material on it, but a material could also apply a texture to an object. This object is a geometric primitive with a material, but you can also have complex mesh models in a number of different formats. A widely recommended 3D model format for web based applications is glTF, which is a compact and extensible format that was created for interoperability.

The glTF format was built as a JSON-based standard for content that can be compiled into a glb file and used to distribute 3D information in a succinct yet robust manner. It is at the core of many VR and AR applications, including Hubs by Mozilla.

We'll come back to an example with a glTF object in bit, but there's quite a lot you can do just with primitives. For this particular function, we first set up our camera and position it. We then create the scene, the torus knot geometry and its material, and then create the mesh. We add the mesh and the camera to our scene, but we won't be able to see any of it until we create our renderer.

So, if we look at our initRender() function, we'll set up our renderer to draw using a WebGL context. WebGL is based on the OpenGL graphics library, and brings powerful 3D rendering capabilities to the browser in a similar way in how native applications utilize graphics frameworks.

We tell our renderer that we want to enable VR mode, set its size, and then appended the renderer and a button to our HTML body. The button allows the site to move between flat and VR mode if the browser detects that the user has an HMD, either desktop or mobile, available to them.

This demo also has a short animation, where on each animation frame, the mesh is rotated and the camera re-renders the scene.

Now, this scene, like the original one that I created back in 2014, is written with javascript - but I also have a version of the same demo that is written using A-Frame, a web framework that provides a markup-style HTML wrapper to three.js. You can see that the A-Frame demo is even shorter than the three.js version - and a big portion of that was actually just to replicate the material that I used for the mesh in the original demo. If that particular material was included as an A-Frame default, this could be rendered in just about 15 lines of HTML.

A-Frame uses a-dash notation to specify entities within a scene, so instead of storing mesh objects in a JS variable, you can refer to them by their DOM id attribute. A-Frame entities can have different components added to them to create different behaviors, as you can see with both our camera and our torus knot geometry, which has the animation built into the entity, rather than being edited separately as part of our animation loop. We also get a few bonuses a little more easily with A-Frame when compared to the three.js implementation - like implementing WASD controls for desktop mouse look through a single property - but at the tradeoff that if we want to do something that isn't handled by what A-Frame covers out of the box, we have to spend more time implementing that ourselves (and that falls back to three.js).

(40 min)

Looking at a slightly more complex (and relevant!) example, we'll take a look at a VR scene that uses three.js to create an environment for viewing photos taken by an astronaut aboard the International Space Station - from the perspective of being in space. Like the first demo, we set up our scene by creating our camera, but this time we add an audio component, our objects, which are more detailed than the torus knot because they incorporate textures, and a skybox element. To add sound to our scene, we have the ability to add an ambient sound, in this case, a .wav file that was generated from the sounds picked up near the rings of Saturn, or *positional audio*, which is when we place audio at a specific point in our scene and allow it to play back spatially. The skybox surrounds our mesh elements, and creates the base of our setting. We add in two spheres with different textures, one to represent the earth and one to represent the moon.

Like in the first demo, we set up both our scene and our renderer, and an animation cycle. We also include some shortcuts to change the photos that are presented as textures on a plane element. You can find the source code for this on Glitch as well, or view it on your phones and see how you can use the accelerometer to simulate head movement.

(45 minutes)

Now I want to move into a demo that is near and dear to my heart for a few reasons - one, it's the core of the product that I get to contribute to in my role at Mozilla, and it represents the core scenarios around virtual reality that I've been motivated by for the past five years: bringing people together.

Social virtual reality is one of the areas of immersive technology that I'm personally the most excited about. Fundamentally, it's about providing opportunities for people to connect and be co-present with one another in a shared virtual world. We've gotten a sense of what that feels like for video games, but these experiences span beyond what is possible with games and expands the benefits of avatar embodiment and 3D worlds to nongaming applications.

Hubs, the product I work on at Mozilla, is a web-based open source platform that supports virtual reality devices, but also runs on mobile and desktop browsers that support WebRTC. With Hubs, users can join into private virtual rooms of their own and invite their friends, family, community members, or coworkers into a dynamic shared space, and fill that space with content from all over the web.

Let's take a look! If I jump over to hubs.mozilla.com, I can create a new room and call it Yglf-demo. I'm going to pick a scene for the room, jump inside, and here I am! Let me take a selfie to remember the moment.

[Now, this could get messy, but for fun, you can go to this URL on your phone or laptop to join in and run around. Please keep in mind the conference code of conduct - that applies in this virtual room too!]

Breaking down the barriers of geography and building on what the internet and social media has started allows us to explore new ways to connect and collaborate on a global scale.

Hubs is built using some of the same technologies that I demoed earlier, including three.js and A-Frame, but is a much more sophisticated experience that is being developed with the immersive web in mind. It uses WebRTC to provide avatars the ability to talk to one another, and an open source physics library called ammo.js so that users in a shared room and grab and interact with objects and have those states shared between everyone who is connected. The environments are glTF models, which I mentioned earlier, and can be created with another web tool called Spoke. We can add a model to our room right now - a media browser build into the platform allows you to search and add content on the fly.

The web is an increasingly powerful platform, and being able to deliver this caliber of experience for virtual reality is already a huge advancement - and with the adoption and future development on the WebXR specification, the capabilities will only continue to grow.

My primary focus is on virtual reality, but I'd feel remiss if I didn't explain a little about augmented reality on the web, too. As I mentioned earlier in my talk, augmented reality can have different features and capabilities layered in, and the browser is able to support an increasing number of those through webcams and device APIs. Let's take a quick look at one of these demos before we start to wrap things up.

This demo uses A-Frame and a library called ar.js to allow a device’s webcam to detect a marker and display something when the marker is seen. This is one of the most basic types of augmented reality applications, but it works on the most devices as it isn’t dependent on a specific depth camera requirement. Markered AR has a lot of benefits and is one form of image-recognition based augmented reality. In this example, I have a marker that I’ve defined as my image target, and when I hold up the marker to my computer’s webcam, you can see the message that I’ve left!

Looking at the code for this, it should seem pretty familiar - it shares a lot of the same structure as the earlier A-Frame demo did, though we now have a few new components thanks to the ar.js library. First, we specify that we're doing an embedded AR scene when we declare our a-scene component. We define our target and put our digital object as a child of the target, which is how the target will be able to display something when it's detected. I put text in here, but you can also use images, 3D models, or primitive geometries, too. If I uncomment this html here, we can refresh and see a little robot model pop up instead of the text! Okay, so back to the slides.

When discussing frameworks and libraries, I tend to focus on the ones that I've personally used, but do want to take a moment to highlight a couple other frameworks for building 3D content that I haven't personally used, but have seen out in the ecosystem.

React360 is an extension of the React library that is also built on top of three.js. It allows for the creation of custom components to support immersive content on websites.

Babylon.js, like three.js, is a 3D Javascript library that allows for the creation of rich WebGL environments with support for features like physically based rendering..

(50 min) Examples of immersive technologies IRL

Now that we've had a chance to explore a few demos, let's look at other applications of how immersive technology is being used today.

Educators are beginning to use virtual reality environments as part of their curriculum - bringing students to a re-creation of an Egyptian tomb, where they can meet with an expert Egyptologist halfway around the world, showing the makeup of an organism from inside the cell walls - creating virtual environments to amplify learning is just one of the ways that social virtual reality technologies are being used today. Curated by Kai is one group focusing on creating such virtual field trips using web VR experiences.

In the medical industry, students are using applications on the Microsoft Hololens mixed reality headset to assist with learning human anatomy, and the device is even being used in surgeries. Training simulations can allow professionals to train for dangerous scenarios in a controlled manner: while simulations for training were previously prohibitively costly for many industries, the wider availability of lower cost hardware now makes this type of training available across industries ranging from medical to retail, manufacturing, and even food service - in a somewhat bizarre application of VR technologies, American fast food restaurant KFC created a comedic experience around the process of making fried chicken.

Moving away from the comedic, we can look at the power of immersive storytelling as a way of exploring the actual events going on in the world around us. Applying virtual reality technology

in journalism utilizes the capability to drive empathy and enable people to step into situations that they are unable to experience in their day to day lives. The Emblematic Group is working to create immersive journalistic experiences like Greenland Melting and After Solitary.

Representing one's work as a professional can also be supported in different ways through immersive technologies. This photographer has chosen to create a digital portfolio in the form of a virtual reality studio showcasing selected pictures. Architects can walk through models of houses and make changes on the fly, from paint colors on the walls to moving staircases around.

Virtual museums can display artwork and historical artifacts without risk of impacting the original works. This allows access to experience culture and history that is independent of geography, but also introduces the question of authenticity, ownership, and responsibilities around how 3D replicas of these artifacts should be handled. There has even been a purported case of theft of an object that was 3D scanned without permissions, and though the legitimacy of the claim has been questioned, the ethical considerations related to immersive content have yet to be widely understood.

Ethical considerations and responsibilities

But it is critical to think about the responsibilities that content creators hold. We must think about how we are trusted with incredibly intimate information around how a user navigates and looks around a virtual environment. We must consider the degrees to which we build layers of augmentation into the world, and how much of that is shared. Will we all see a shared base augmented reality, or be able to opt into those reality layers that our friends and family share with us? What kind of property implications exist for augmented reality vandalism? We've already seen an instance of this - an augmented reality advertisement for Burger King appeared to set McDonald's billboards on fire.. While some considered this a playful experiment in mobile filters, others found it to be an alarming display of disrespect and for it to have set a dangerous precedent for non-consensual digital manipulation of another's property. When does this cross a line?

We must think about how to protect a user's identity across the applications and experiences they visit and interact with - a question that today's social media sites and ad platforms are also contending with. Biometric data can be collected by headsets and track, for example, how you move around your living room when you're in VR. Eye and gaze tracking can provide application developers with heat maps of exactly where a user looked, and for how long, within an application. It's important to point out that these capabilities aren't inherently bad in and of themselves - but they do provide a vector that can be misused if developers are careless with how they use this information.

With immersive headsets, we get valuable functionality by the inclusion of tracking cameras, but device manufacturers have a responsibility to treat the massive amounts of informations

captured on their devices with the utmost care, in terms of how user data is handled and redistributed to provide the services offered.

Beyond the data layer, there are sociological considerations at play with immersive technologies. Social applications that allow users to embody avatars must take care to provide affordances to protect the users from malicious actors and psychological harm that could result from virtual harassment. Studies already show that female-appearing avatars are subject to far greater degrees of inappropriate language gestures, and simulated assault than male-appearing avatars, and it is critical for social platforms to be cognizant of the behaviors of their users and be proactive about the ways that systems can be used for harm.

Now, none of this is to say that today's immersive technologies are uniquely flawed: technology is never perfect. It's necessary to address the challenges alongside the opportunities. We're not at the inflection point where we have an 'iPhone' moment for a virtual or augmented reality headset and suddenly everyone will be buying them - the hardware that we wear on our heads is still bulky and difficult to wear for extended periods of time, we don't have a universal input language with gestures or an easy way to type. It can be easy to look at the hardware that we have today and apply the capabilities they have to the scenarios we're already familiar with, but the opportunity lies in the areas where we can reduce the distance between us and our digital information, and improvements to these issues will come.

Future advancements and integrating other web technologies into AR and VR

Advancements with wearable technologies, haptics, and computer vision enable us to make progress towards more efficient input mechanics for immersive technologies. Voice input capabilities, algorithms for creating more realistic avatar expressions, and more compact hardware are all under active development and allow us to tackle the challenges that exist for widespread adoption more readily.

It's also straightforward to build applications that integrate other web technologies into the solution - so you can imagine immersive applications that use text to speech capabilities, real time translation from one language into another, WebRTC or midi, and so much more. The ecosystem around the web is well-established, and it provides numerous capabilities to extend functionality into immersive web apps.

And if I'm being honest, we've only been able to scratch the surface here of what's happening in how the web is enabling VR and AR. Web infrastructure is enabling countless services related to ads, payments, and analytics. With web assembly, code written in different languages can be compiled into applications that run in the browser - today's game engines are capable of outputting HTML 5 web apps or generating lightweight engines that can be embedded into a webpage. Non-player agent characters can utilize chatbot backend technologies to create branching narratives. VR experiences can be seamlessly broadcast to streaming sites and allow for asymmetric communication between users in VR and those who aren't. The web is everywhere, and that includes the devices of today and those that are coming in the near future.

The thing that I'm most excited about with these technologies over the coming years is access - access to hardware at lower costs and further development of libraries, frameworks, and tools will make virtual and augmented reality applications easier than ever to create. For the first time in technological history, hardware, software, and network capabilities are at a place where we can truly begin to seriously think about the opportunities that immersive technologies grant us.

Perhaps most importantly, immersive technology gives us access to understand and educate people about some of the world's most pressing challenges of today. It is a tool that can be used to literally step into the experience that another person has and to see an event from their physical perspective. Virtual meeting solutions offers us a path towards reducing carbon emissions caused by travel and assist in granting us new capabilities around collaboration to facilitate remote working scenarios. We can more effectively visualize climate and weather patterns to illustrate the reality of our impact on the environment.

If the ideas that I talked about today excite you, I want to end with some resources that are available online that will help you get started. There isn't one specific right way to get started, so I recommend spending some time looking at what's out there and what excites you most. And you don't have to have a headset to get started. One of the simplest ways to begin is by using your mobile phone.

The site on this slide provides materials and sample projects for getting started with programming applications for virtual or augmented reality on the web. The projects that I demoed earlier on in my talk are also available online if you want to remix them and start with those. If you're interested in looking at the source code for how we're building Hubs, you can find that on GitHub too. Contributing to an open source project allows you to develop a familiarity with the types of libraries that are most commonly used for the immersive web.

There are also a number of public immersive web Discord servers that you can join if you want to connect with communities who are actively building web-based virtual and augmented reality apps.

If there are only a few things that you take away from this today, I hope that it's these: thinking about immersive technologies means thinking beyond specific decides or modalities. It's not just using your phone to place furniture or catch pokemon or a particular device with certain controls, it's about the presence of new interaction models and wearable computer displays that grant you access to contextual information with the world around you.

With all of these new considerations comes a question of access and risk. Data privacy, and recognizing how data about how we think can be combined, through immersive devices, with information about how we move throughout our environments. We need to be aware of developing content for virtual and augmented reality that changes our assumptions about how humans interact with information and one another.

We have more to draw from than video games. We need to consider elements of psychology and sociology and politics in designing these systems. Jump scares, platforms for organizing groups around hate and harassment - all of these become more emotionally amplified with head mounted displays. Already a problem with two dimensional technologies, we are faced with new contexts to consider for technologies that we step inside of.

But with these capabilities, we are presented with bringing new advancements in our understanding - our understanding of how information flows, our relationships with the world around us, and the way we interact with each other. These technologies are changing the ways that we work, learn, and connect. It is vital that as we consider the ways that the web will encompass these new capabilities, we do so with an intense focus on accessibility, inclusion, and responsibility.

I would also invite you to continue the discussion of the topics that I covered today. If you have questions or thoughts, reach out to me on Twitter or through email and I'll be happy to talk through them. I'm also always happy to meet in Hubs, and have actually started to have quite a few of my meetings at work as an avatar instead of through video conferencing!

I hope that you've enjoyed the first day of YGLF - I know I have - now let's call it a wrap and start the after party, yeah?

Thank you!