# Chacmool: Augmented reality in Chrome Canary


**By** Reza Ali

Reza is a software developer.

**By** Chris Wilson

Chris is a contributor to Web**Fundamentals**


When preparing for Google I/O, we wanted to highlight the exciting possibilities of augmented reality (AR) on the web. Chacmool is an educational web experience demo we built to show how easily web based AR can help users engage with AR experiences. The web makes AR convenient and accessible everywhere.

We have now enabled this demo on Chrome Canary on ARCore-compatible Android devices with Android O or later. You'll also need to install ARCore. This work relies on a new WebXR proposal (the WebXR Hit Test API), so it is under a flag and intended to stay in Canary as we test and refine the new API proposal with other members of the Immersive Web Community Group. In fact, to access the demo you'll need to enable two flags in `chrome://flags: #webxr` and `#webxr-hit-test`. Once you have these both enabled and have restarted Canary, you can check out the Chacmool demo.

The Chacmool AR experience is centered around education, leveraging AR's immersive and interactive nature to help users learn about ancient Chacmool sculptures. Users can place a life size statue in their reality and move around to see the sculpture from various different angles and distances. The immersive nature of AR allows users to freely explore, discover and play with content, just like they can in the real world. When viewing an object in AR, as opposed to seeing it on a flat 2D screen, we are able to get a deep understanding of what we are looking at because we can see it from many different angles and distances using a very intuitive interaction model: walking around the object, and getting physically closer or further away. Also, in this experience, there are annotations placed directly on the sculpture. This enables users to directly connect what is described in text and where those features are on the sculpture.

This demo was built over the course of about a month to build, leveraging some of the components from the WebXR team's first web based AR demo, WebAR-Article. Information about the sculpture was sourced from its Google's Arts & Culture page, and the 3D model was provided by Google Arts & Culture's partner, CyArk. To get the 3D model ready for the web, a combination of Meshlab, and Mesh Mixer was used to repair the model and decimate its mesh to decrease its file size. Then Draco, a library for compressing and decompressing 3D geometric meshes and point clouds was used to reduce the model's file size from 44.3 megabytes to a mere 225 kilobytes. Finally, a web worker is used to load the model on a background thread so the page remains interactive while the model is loaded and decompressed, an operation that would typically cause jank and prevent the page from being scrolled.

We can't stress enough that, since we were developing on desktop and deploying onto a phone, using Chrome's remote debugging tools to help inspect the experience creates a great fast iteration cycle between code changes, and there are amazing developer tools in Chrome for debugging and checking performance.

## Best practices for AR/VR experiences

Most design and engineering guidelines for designing for native AR experiences apply for making web based AR experiences. If you'd like to learn more about general best practices, check out the AR design guidelines we recently released.

In particular, when designing web based AR experiences, it's best to use the entire screen (i.e. go fullscreen similar to how video players going fullscreen on mobile) when using AR. This prevents users from scrolling the view or getting distracted by other elements on the web page. The transition into AR should be smooth and seamless, showing the camera view before entering AR onboarding (e.g. drawing a reticle). What is important to note about web based AR is that, unlike native, developers do not have access to the camera frame, lighting estimation, anchors, or planes (yet), so it's important that designers and developers keep these limitations in mind when designing a web based AR experience.

In addition, due to the variety of devices used for web experiences, it's important that performance is optimized to create the best user experience. So: keep poly counts low, try to get away with as few lights as possible, precompute shadows if possible and minimize draw calls. When displaying text in AR, use modern (i.e. signed distance field based) text rendering techniques to make sure the text is clear and readable from all distances and angles. When placing annotations, think about the user's gaze as another input and only show annotations when they are relevant (i.e. proximity based annotations that show up when a user is focus on an area of interest).

Lastly, because this content is web based, it's important to also apply general best design practices for the web. Be sure the site provides a good experience across devices (desktop, tablet, mobile, headset, etc) - supporting progressive enhancement means also designing for non-AR-capable devices (i.e. show a 3D model viewer on non-AR devices).

If you are interested in developing your own web-based AR experiences, we have a companion post here that will give more details about how to get started building AR on the Web yourself. (You can also check out the source to the Chacmool demo.) The WebXR Device API is actively in development and we'd love feedback so we can ensure it enables all types of applications and use cases, so please head over to GitHub and join the conversation!

---



*Last updated July 2, 2018.*