2021-22 CAWD “Web Design” side of our Advisory Board meeting minutes.

1/26/2022

Faculty at meeting:

- **Will Bohmann**
- **Matt Cronin**

Web Design Advisory Board Members:

- **Dave Gibson** - Propeller Media Works
- **Matt Onyon** - Keene State College
- **Mark Dubois** - Web Professionals
- **Adam Fehnel** - Putnam Investments
- **Matt Gonzalez** - Winooski School District
- **Chris Goodwin** - Dockyard
- **Steve Herr** - Rovers North
- **Tim Basiliere** - Putnam Investments

Focus 1) Github. I’ve been working Github in and out of the web development unit for the past several years. Students grasp to understand the workflow and management of code assets and repositories. If students pursue education and career exploration in the software field, what kind of fundamentals should students have regarding Github? Is this an area that we should be exploring in year two of the CAWD program?

> **Matt O** - I'm lacking on this front; I played with Github but never really mastered it so maybe my comments are not completely valid but it's everywhere. Github is built into IDEs now such as VisualStudio Enterprise, AndroidStudio and is a plugin for others such as NetBeans. I currently have several freshmen who are using Github on their own and I'll have one of them give a class presentation on the subject. It's a good thing, it's helpful and our students should know about it even though I don't yet.
>
> **Mark** - Github - students should understand workflow, forks, updates, commits and related fundamentals. They should likely have an account and examples of their work (a Github portfolio) for potential employers. They may consider contributing to projects to gain greater insights into overall workflow. This assumes they have solid coding "chops." If they are pursuing a career in web development, they should certainly have a Github presence.
>
> **Adam** - Github is definitely important when we're looking to hire new developers. It's a good way to confirm they have the skills they talk about during our interviews. We don't

really like quizzing applicants in person (some places do), so besides getting a feel for their knowledge by talking to them, looking at their Github or portfolio is an important step to confirm they aren't pulling a fast one on us. Students should know how to clone repos, make changes, commit and push them and potentially fork existing projects. Tim yells at me for not putting enough details in my commit messages, so practice making those helpful for future people looking at your code and trying to understand it.

Matt G - If you see an opportunity to introduce Github, I'd take recommend it. I am seeing so many businesses move towards AWS and their versioning software instead of Github but it's a useful tool to understand. Even if it is just to learn how to pull projects and set them up in your own environment, it will be a great stepping stone to future learning.

Chris - I assume year 1 is seen as an intro year (let's get the basics out of the way and if you like it we'll ramp up in year 2). With that in mind, I see year 2 as a good time to start teaching GH. It could be helpful to teach both the GH Desktop app and GH from the command line. A major blind spot I had when joining DockYard was my command line and GH knowledge. Even having some fundamental knowledge would have helped me.

Things I do regularly in GH:

- Set up new repos
- Clone repos
- Set up SSH keys for GH
- Push/Pull
- Commit code
- Write meaningful commit messages
- Rebase/Merge local branches with remote branches
- Undo previous commits
- Reset files that I don't want to commit
- Add files and directories to a gitignore file

As with most things, the key is to teach students how to find answers when they get stuck and give them the confidence working with GH to feel ok about the solution they find when they go searching.

Steve - Very important. I don't really get to use it at my office as I'm the only developer but I wish some of the contractors we use, used github. It would be wonderful for them to know how to create a new repository, push, pull, review, comment, and create a fork of an existing repository.

Tim - While I think understanding Git is essential to the programming workflow, I could understand holding off until a year two student as long as there is some type of backup/versioning control on the network (oh the woes of autobak at CAWD). They will be fine as long as they know how to checkout, commit/push, branch, and merge (conflict resolve) if they plan on working with software as a year two. There's still a lot about git I don't fully understand, but have been managing fine.

Focus 2) Augmented Reality is a mixture of 3D modeling, web and mobile programming. There are a lot of jobs in the field. Are there industry resources we should be looking at, credentials to consider, areas to explore?

Matt O - My son in Los Angeles who is job-hunting commented on this recently as it comes into most any job search in the "arts" in that part of the country (at least). Again, this is a subject I want to know more about and will eventually, but it is now a big part of the job market.

Mark - AR - yes, students should understand the fundamentals. Adobe has an entry into the market and students may want to investigate. I don't know how long this will remain free - https://www.adobe.com/products/aero.html
There are alternatives (and students may well wish to explore some of these. One example of a listing of alternatives -
https://www.producthunt.com/alternatives/adobe-aero

Adam - As Facebook, Google and other big players (potentially Apple?) focus on AR and VR, I can imagine this field booming in demand for developers. I'm not specifically versed in coding for it, but I'd expect a basic understanding of platforms like Unity or other 3d modeling/programming software and the fundamentals of software design patterns like MVC. Otherwise the development of VR/AR is just a matter of different hardware/software interfaces.

Matt G - I'd have no experience with a practical use of AR in my current job other than to support learning tools in the school. With the school district's multimillion-dollar construction project, I was exposed to just how common these large construction projects rely on AR to visual construction plans. I don't know where to guide you on this but I could reach out to the architecture firm that scanned our entire campus to do a day visit with you.

Chris - I know Apple has plenty of developer resources on this that are available if you have a Developer account. I'm sure other platforms have similar resources. I would encourage the class to work together through some of those tutorials together.

I feel it may be valuable to teach the basics of these technologies and to work through them as a team. I'm sure there is a lot to learn and, realistically, I'm doubtful there are many 1 person "teams" building every part of these experiences. Developers would be expected to work together with a larger team to build out a full experience.

Steve - I would like to get into this more but every time I look into a setup privacy concerns torpedo that idea. Also, I refuse to give zuck a cent of my money or data.

Tim - Unfortunately, this isn't something I can be too helpful with since I'm not in this field. Because these are mostly programming and/or modeling software based, a portfolio with a degree in computer science would probably be best.

Focus 3) How do you handle contracts when doing freelance work? Do you have a boilerplate contract to go back to client after client? Do you write up a custom contract for each job? Or do you not use contracts at all, and give and take over email and other communications?

Dave: I would definitely recommend always having a contract for any project more than a few hundred bucks. There are many sources for templates. For us, we use a custom Master Service Agreement - an umbrella agreement that includes all the common/global items that would apply to all future Statements of Work.

Matt O - In my distant past and limited experience hiring graphic designers, the designer held an initial meeting where she pulled out a pad of printed sheets, tore one off and filled it out with the parameters of the job and our contact information. All contracts have 3 required elements: Dates, task and consideration$$$. A boilerplate template is a great starting point but it's very difficult composing a form for every occasion.

Mark - I recommend starting with a boilerplate and modifying for each client. Obviously, I don't have one to share as Illinois laws differ from other states. If students are serious about this as a career path, they should definitely be doing contracts (including change orders). Who is responsible for what (and deadlines/ due dates) is a key component of any business relationship. Keep the contract simple. It is always helpful if you can have someone with a legal background review the boilerplate. Don't rely on examples found online (did I mention laws vary from state to state).

Adam - When I started, I had a very detailed freelance contract including how to litigate disagreements, other boilerplate "cover-your-ass" disclosure as well as the project details and expected breakdown of payments/milestones/etc. As I've not been sued, I've loosened my contracts to basically just include milestones and expected payment structure (I usually do 1/3, 1/3, 1/3 and extra work orders when needed). If I had a nightmare client I might still want that earlier template's disclosures, but I haven't needed them and understand I accept a certain level of risk by not including it now. I've become more trusting, which in turn makes me more personable, which in turn makes my clients happier. If I ever freelanced for an unfeeling, faceless giant corporation, I'd probably bring more legalese to the contract again.

Matt G - It depends on the client. I'd rarely do no contract at all unless you'd be willing to do the work for free in the first place. It's the same advice as not leading money to someone where you'd actually expect the money back. A boilerplate contract is just fine to start out but if you contract with a larger client, that would most likely require a personalized contract if they didn't propose one on their own. Teach the boilerplate as a baseline.

Chris - I haven't really ever freelanced so I reached out to a colleague for some of their thoughts. Here is their unedited response:

It's good to always have a contract when working with clients, and sometimes per client project. How detailed the contract is depends on a couple factors:

- How complex the project is
- How complex the client's business/team is
- What level of trust you have in the client

For example, I might have a very detailed and formal contract even for a small, simple project if the client is new to me and I don't have any references to go off of.

On the other hand, I've been able to have year-long, full-time engagements with clients based on a single page contract, because they were very trustworthy and had a good reputation.

Common sections I include in a contract are:

1. Scope of work
2. Liability
3. Effective date
4. Timeline and deliverables
5. Pricing and payment

Other sections, depending on the type of engagement, might be:

- Relationship
- Confidentiality and intellectual property
- Termination clauses

Some clients create the contract, and other times I do. Either way, it's best when both parties have equal rights and protection to go separate ways without blame if things don't work out, ideally with pre-defined terms around how much notice should be given to the other party out of respect.

Ultimately, if you're in doubt, you can find a fee-only attorney who specializes in freelance and consulting contracts to either review or create them. You can also engage an attorney to create a boilerplate contract that you can tailor to each client. It shouldn't cost that much and offers a lot more reassurance.

This last part is not advice, just my approach: I use my own judgment to determine how formal and intensive this process is. I only work with clients that I've thoroughly researched and talked with a good bit, and turn down many clients who seem cheap, untrustworthy, aggressive, etc. If you work with good people, you hopefully shouldn't ever need to enforce contract clauses, but obviously it's a good safety net in case things go very wrong. I'd say that I play things a little loose because I'm not a fan of exhaustive agreements and legal protection, but I wouldn't recommend that to others.

Related to all this, I have a sole proprietorship LLC. This keeps my personal liability and assets out of legal consideration.

You can also get affordable freelancer insurance to provide some safety in case you end up in a legal dispute. I currently don't have an insurance provider for this and "self-insure", but it's not a bad idea. All these things depend on your personal comfort with risk.

Steve - To get around all the legal garbage that comes with using freelancers I just use friends I trust and venmo/cash. If I don't know anyone that can deliver what I need, I almost exclusively use Upwork. It costs a bit more but it's worth the convenience and the pool of people are larger than anywhere else I've found.

Bridge for photo management and batch processing.

I've automated 70% of what the last employee did in my position. Making me look like a better employee because I'm able to focus on the Dev part of my job in that extra time and also gives me more time to waste on youtube. Win-win for everyone. Half of the automation I've done is through bridge and custom-built actions. Can't recommend it enough. I also use bash/applescripts/windows task scheduler/SQL jobs to do the other half of the automation.

Adobe Camera Raw for image editing and post-processing

Tim - This has always been a fairly informal process for me as far as how a project starts. It's been more akin to hiring contractors for house projects. I've always followed the lead of my client and written up a preliminary invoice upon request with bulleted details of the project summary and estimated cost (this is rare). I have my own boilerplate invoice where I keep track of the hours I've worked and any extra work orders if the scope extended over our initial agreement. I try to keep this as succinct as possible. With that said, the majority of my clients have been through larger companies that handle the majority of the contract work (financial and advertising companies) who want to pay you, so I've never felt like I had to protect myself. In my experience, advertising companies have preferred paying on an agreed upon project fee, and have been good about simply paying a larger sum or rush-based hourly fee addons depending on the situation. After every project, I've always emailed my final invoice over to request payment. I suppose I use this invoice as a contract. Hope this answers your question.

Focus 4) What is next / what should we be keeping our eye on for 2022?

Dave: Of course I'm naturally going to plug digital accessibility. The number of digital ADA cases rose 15% last year with 74% targeting Ecom. And our Accessibility.Works consultancy outpaced our web development revenue for the past two years. Doubled this past year.

Most RFPs we get now require web accessibility of "ADA compliance" although few clients are sophisticated yet to know exactly what to ask.

Matt O - I have a group of students who are working on a "fact-checking service" where algorithms are used to check the validity of text. Lies are rapidly becoming part of culture and sadly, some cultures are accepting lies, accepting liars and are considering lies as fact. The need for non-partisan, non-political truth is growing.

- People will bore of social networking so we should watch for its replacement. I just looked at Facebook to find those most boisterous are not those I would never socialize with. Watch people to bore with now commonplace hostility. Look for current social media platforms to become the new trailer parks.
- Watch for personal analytics with impending fascism in mind. We use analytics to monitor all sorts of things online, but there's not much that will give us individual-level insight about forms of surveillance we currently don't know about or know exists.

Mark - Things to focus on in 2022 - remote working (full disclosure, we offer remote working professional certification). This is something which is here to stay. Audio user experience (yes this interface is also here to stay). That also means web sites may need to consider some audio components (been saying this for a few years now - it will happen, just don't know when). Foundational knowledge is always important (know when it is best to use ES6 or newer and when a JS library makes sense, for example). Know when to apply CSS and what CSS is capable of (along with HTML - for example grid and flexbox). Also user experience design is definitely something students should be familiar with.And, students should have solid business knowledge these days (for example, why contracts are important).

Adam - Work from home is obviously becoming a standard, despite many places itching to get employees back in the office to make use of their multi-million dollar property leases. Putnam plans to go with a hybrid model some time in the spring with Omicron passes, but I bet most people will remain WFH most of the time. I might go in on Mondays and Fridays because I like breaking up my weekend and resetting my flow. I imagine most people will come in Tues, Weds, Thurs and take a "long weekend" even though they're supposed to be working Monday and Friday. As far as the industry, Vue.js or Svelte are getting some traction as far as preferred frameworks. React and Angular are losing some market cap to them. SCSS and precompilers are the norm now. We're ditching support for IE11 this summer, which I'm giddy about! Flexbox seems better than CSS Grid, but Grid is coming along in browser support and features.

Matt G - Working remotely is a new norm for many people. Learning how to separate work from home distractions is going to be a huge piece for young professionals to overcome. I just wanted to highlight this as it's going to be a silent issue that won't be easy for some, much like the universal frustration that conventional high school does not teach financial literacy.

If you want a technology answer to this, I don't have specific recommendations since no companies are alike. Single-page web applications seem to be trending around here for businesses like New Breed or Fluency, and any other relying on Vue, React, or Angular.

Chris -

Svelte/Frameworks

People seem to be very excited about Svelte. I'm not sure I believe any heavy front-end frameworks are a great solution for small/medium sized websites but when you start getting into larger scale web applications these have a lot to offer. With that in mind, there's certainly value in helping students explore beyond vanilla JS. Most Front-End Developer jobs are going to be looking for experience with these frameworks and Svelte seems like a (relatively) easy framework to start learning.

Scoped Components

With many of these new frameworks comes the possibility of writing styles that are scoped to their respective components. This opens whole new possibilities and questions for styling (Do you still use classes? Is BEM valuable anymore? How do you pass context between components?). It may be valuable to expose students to what is possible and help them to start to form their own opinions about how to approach building their websites.

Working with Design

Just about every colleague of mine is not a "do it all" developer. They rely on others with expertise in specific areas. With this in mind, it may be valuable to encourage students who are leaning into building skills in different areas to work together on projects. A student more interested in Design may be tasked with creating the visuals and building an HTML/CSS base. A student with heavier JS interest may be tasked with cleaning up and componentizing the code. Requiring students to work together through a project would be a great way to emulate a working environment. Half of my job is working with other team members to find answers, align on solutions, and talk through problems. Helping students build these skills early should make their transition into their careers easier and help them build confidence in their ability to initiate those conversations.

Note: Love that you're teaching Scrum and Agile methodologies!

Design Tooling

I've found most of my design colleagues have moved away from Adobe. There is definitely value in teaching Adobe software and I think many of those fundamentals transfer to other design tools but I wanted to note that I've seen many transition away from Adobe and towards Figma or Sketch.

Interpersonal Skills

As mentioned above, much of my job is working with colleagues to get things done. I am working remotely to do all of this so I am almost exclusively doing this in Slack. Things I do every day:

- Hop into impromptu meeting
- Talk things through
- Send clarifying questions to a coworker
- Work on code and build async
- Reference sources of truth (like design spec files)
- Communicate status in team syncs

These skills take time to learn and build. Helping students to learn how to use Slack and how to communicate with each other effectively is really important.

Note: I know remote work might still be seen as a bit of an outlier but, as far as I'm aware, most companies (remote or office based) rely on some version of online communication to communicate more efficiently.

Staying the Course

I have seen CAWD improve in leaps since I attended. I've loved seeing the curriculum grow to be as helpful to students as possible. Of course it is important to make sure what is being taught isn't outdated but the core concepts behind what you teach should continue to stay the same. Start from scratch, learn the basics, memorize tags and selectors, build understanding of the foundational parts of the web. Then, once those are in place, build on top of them by teaching tools the industry is using, more complex approaches, and develop opinions about tooling and approach. This "recipe" prepared me well for my continued education and eventual career and it's what makes CAWD so special.

Tim - It looks like AR/VR is coming together into these metaverses. Primarily via Meta and Microsoft's XR platform. Keep an eye on the OpenXR platform for native development and the plugins/programs that support OpenXR. I'm expecting a pretty rapid development of these tools.

Most importantly for web developers, IE will officially come to an end as IE11 end-of-life is on June 15! This means full SVG support, CSS variables and filters, no more ES6 IE polyfills, and so much more. I'm certainly ready for a new type of headache.