[00:00:03] **Christine Pinto** Hello and good attack. Thanks for joining me. My name is Christine, and I am going to talk about how I hacked Cypress, especially how I hacked Cypress for the visual test. I will tell you about how I stabilize the test suite and make it more efficient. And I'm going into three points.

[00:00:19] First, highlight text. How did I highlight texts? And why is it so important to highlight text, especially when we look into what storytelling platform or any editing options for text?

[00:00:30] Second, I'm going to talk about multiple elements. How do you handle in an efficient way multiple elements to avoid writing hundreds of lines of code and instead write like five lines of code? And I will show you what amazing ways I've figured out there to handle them.

[00:00:47] And third, the biggest topic is all about images. When you speak about the visual test, images are the most important, but awesome is also the most problematic part to ensure that images are loaded. I'm going to talk about three different ways how to handle that.

[00:01:03] First, how to handle them in a way that you load in image to wait for loading it. Second, how to wait for resources, and third, but not the least important. Actually, the most important, I'm using slow scrolling to load every part in the image and every part of the page. And after seeing all these different points, you're going to have a stable, workable, and really efficient test suite and going to see how to use it and to use your visual tests in an amazing and especially easy way. Let's jump into the code. All my tests scenarios today are written in Cypress.

[00:01:41] Cypress is a JavaScript end-to-end testing framework that's a really easy-to-use and pretty cool testing framework with a growing community. If you want to see how my setup is, I would say, look at my repository at GitHub.com/Crispint/visualtesting as I won't go into so much details of so many details how I set it up or which packages I'm using. You can see it in that repository quite well. If you have any other questions about how to use Cypress or how do I set it up? I definitely suggest Google it because there are tons of

different tutorials. Or look in some of my other speakers today. I'm pretty sure Cypress has multiple time mentions as well. The visual testing tool I'm using is Applitools. Applitools is a cloud-based visual testing solution, which I'm using in which I use now multiple times. It is a really cool tool. It has tons of different functions. I won't go into too many details, but I will show you what I did with my solutions and what kind of different results I got in my visual tests. They have a for free account, which I'm using here as well. If you want to know more details, just reach out to customer support.

[00:02:52] They're really, really helpful and they really help you also to set it up if you need it for your company or for yourself or whatever, and it's really good. My first scenario, what I mentioned, I will show you, how do I highlight text? Why is it so important to know how to highlight text? If you work in a storytelling platform or in a block or in general any initial tools, you want to ensure that your formatting functions or your editing functions are working from aligning a text or making a text in different text color, or whatever, and especially if you're looking into visual testing. Do you want to test with your test automation framework?

[00:03:33] Are these functions all working so can I click on all button and then the visual test you want to see? Do they look how they actually gonna look like? And for that, you want to know how to highlight a text. What my test is going to do, it will add a new note. It will write down visual tests and then I will highlight this word test and underline it.

[00:03:55] That looks not pretty easy, but finding a solution with Cypress was challenging in the beginning because I thought I could just easily use a function or anything. But it's not built-in into Cypress, that's why I'm going to show you how I did it with my code. It's quite the easy tool, all the code that you record, you visit the website. I'm using the wait until package is as a third party package, which can help you to wait on specific elements.

[00:04:25] There are different other solutions, how you can wait for something. And I will show you that in my other examples as well. But left can tell you the wait until package is really, really good and then I'm adding the note. I'm writing down the words and the visual test and what's then important. Then I'm going to highlight the text. What I'll do? I'm using real press. That is also a third-party package, which is called Cypress real events, really good, especially for having and other things that can really only recommend this tool. It's really, really good. What you're going to do is, you say real press and then you do exactly what you would do on a keyboard, you click shift and then arrow left, arrow left, arrow left, arrow left, and then you have your text highlighted really easy and simple solution. And then I'm just clicking on the buttons on the line and then it's there.

[00:05:17] I'm going to show you that now and real that you don't think I'm going to tell you anything differently, but it looks really easy. And once you get it, and once you find that solution, it is really easy to use. And you see here, it's exactly going to do what it's supposed to do. It's opening up the website and then it's clicking on the little plus button.

[00:05:37] What it's doing now, writes visual test, mocks the test, and then, you cannot. It's not so easy to see here because it happens in like really, you can't like what you see here, it's not really easy to show you. You saw it in the test that it actually highlighted it and it also went through it.

[00:05:54] And that is all it is the easiest solution when you want to highlight the text and I can only tell you that any editing tool or storytelling platform or blocker anything and you want to check also comes down sometimes different other tools may have different free

text, maybe and you want to give your user or your customer different options off designing maybe something, maybe a postcard and inviting card or anything you should test if all formatting options are working. And for this, it's really that easy solution, easy and really, it's really workable and nice.

[00:06:30] My other solution, which is also important, is multiple elements, and this and also all the other scenarios I'm going to show you at Testguild.com. I took this website, which I think everybody of you guys knows and I will show you on this website different scenarios and different strategies to optimize your visual tests. The first thing I'm going to do is, you see here in this part, there are multiple articles, and especially when it comes to visual tests, you want to ensure before you do the visual test that specific things are actually displayed not completely in if it's the white background color, but just generally functionally, all these elements there.

[00:07:13] And in this case, we have here the last four articles which were published on Test Guild. And this is quite the easy way of doing this. First, I'm going to get all the latest articles, then a really neat way of checking. Are there really four elements because we want to see four elements unless it changed something in the website, then I have to change to that as well.

[00:07:36] But I'm checking here. Do I see actually four elements? And then, really pretty cool tool. You can go through each of these elements. Instead of writing now four times, give me the first the second child, and the last child. I'm just getting myself all the articles. Check if it's four articles and then I go through each of them. On each of them, I'm checking is the headline visible, and then I'm also checking if the link is visible and if the link hasn't href.

[00:08:12] And I will show you that now. And it's really good because you don't want your visual tests to be false positives because maybe some functions didn't work you want. Before you execute a visual test, you want to check functionally, and also in general, maybe you don't have a visual test, you want to ensure that your elements are visible.

[00:08:33] This was quite fast. What you see here, I'm first checking, I found four elements. Is it really four elements? And then it goes through. Does this article have the H2? Yes, the H2 is visible. You can see it a little bit up here. It's the H2 is visible and recheck. Is it a link?

[00:08:52] Yes, there's a link, and it has the divide attribute. And instead of writing that four times, five times, or six times through the each. I really have it easily visible here and it's quite dynamic. If this would change, I want to display eight articles or maybe 12. I don't need to change anything on the code itself because it can just stay like this and like this.

[00:09:16] You can really go through different scenarios from a table where you maybe want to go or you maybe have multiple navigation points as well. I use that, for example, at one point that I got all the different navigation points and click through them actually and check if everything is fine and this is really the easiest way with each to go through them all.

[00:09:35] It's so easy. It's a native in Cypress, I think since somewhere version six or something and it's really, really cool. It just saves you so much. Look how small my code is and how much it actually can test. And it's I can only say like if you work with visual tests,

you have to ensure that all your functions work because before you go to visual test because what you want to avoid is a lot of false positives for you, but also for your team.

[00:10:01] You don't want multiple false positives because maybe one element is not loaded completely, especially now when we go on to the topics. All images it's even more important to ensure that all images are loaded or visible, that it's not a loading mistake and the website actually works. No, that it's really. If the image is not displayed, there has to be a mistake in the code and that is important that you see the difference or that you also don't have so many false positives because otherwise if your build pipeline read and read and read and read and read over and over again, you just won't look anymore. And for example, in this case, when you work with Applitools.

[00:10:39] The error message you get in your build pipeline is not so clear it just tells you it found the difference because it cannot tell you in that small little connection it has to your pipeline which actually was wrong? It just tells you which of your tests and which of the screenshots had a difference. And that's sometimes really difficult to decide a case. And actually now, because then you have to look into Applitools website, you have to look in, you have to look at the baseline and the checkpoints, and you want to avoid that for simple things like maybe an image wasn't uploaded completely on this case.

[00:11:11] Maybe one of the articles wasn't completely loaded, and you want to ensure that there is no functional problems actually before that. And when it comes down to the visual test. Maybe there were some CSS assignment problems or something because you cannot check that you cannot check if something was five pixels from the left of 10 pixels from the right. If somebody changes CSS classes. It's not so easy to test that over any of the functional testing. That's why you have visual tests. Let's go to the biggest topic images.

[00:11:43] I think everybody knows they see my role in, they hate it. When you see the little loading circle, it is really not so easy. And you know that especially in visual tests, when you have that check the baseline, what it's called where the image is then you have the checkpoint and you just see that spinner. It's not nice. You just can't. It's not clear. Was there really a problem with your third-party connection or is that just a loading problem? And here I'm going to show you today three different ways how to check that. The first is just check the image itself if it was loaded, if you have just one big image that is easy to check, you can just visit the website, get the image. And what you do is you check the natural risk if this is greater than one, because this is set when the image looks and you will see in our example, it will wait.

[00:12:41] If you have really a problem in the image, the natural width will not be greater than one and it will be zero because then the image is just not loaded at all. And that is really an easy and simple check. It doesn't take that long, and it works if you have just one image you will see in the solution later will show you the visual test. It's not the best solution for everything, but it is a possible solution if you maybe have just one image which makes you problems all the time.

[00:13:10] The second one is you can actually check specific resources if they were loaded. I found through quite a long time looking through the community and different solutions in which I compared. I found a really cool function, which is called all the people who invented that called wait for resources. I'm going to switch to my commands, wait for resources, it's quite the big command.

[00:13:38] And it's really cool because you can check for different styles and different resources if they were loaded, because especially if you saw different styles which are really complex and you really want to wait until they are loaded because they if you different in alignments, then you can wait for them. It's also, again, not the best solution, because if you have a lot of JavaScript towards a lot of CSS, you can't always wait for everything because it just also makes your code quite bulky.

[00:14:05] But what is does? I will only go a little bit in detail, it goes basically to the performance object and checks if that resource is there or not. It's really easy, really simple. When I found that I was really odd how simple and easy it was and how easy it could set up. And it's really nice and it works quite well. You can, like I said, see this whole code in my GitHub repo and just check it out. I didn't write that for myself. I just copied it as well, and it works really, really good. And it really helped a lot with a lot of my solutions.

[00:14:41] The last solution, which is the solution I prefer the most. It takes more time, but it's the solution which ensures most of my visual tests to work quite well. It's slow scrolling. What do I mean with that? If we enter a website and we have, for example, lazy loading images, then this lazy doing images will only be load when the user actually scrolls over these pages. To do that, you have to slow scroll. You have to imitate the user and actually slow scroll over the page. And that's exactly what I do. What I going to get, I get the body scroll height and I start at zero and I always go 100 pixel. I wait a little bit as sometimes. Otherwise, it is too fast and then I slow scroll through the whole page and you will see that in a sec. And then all the images are loaded to that. And that's spin so far. The best solution for all my problems I had with different image loading because through that, you avoid all the lazy loading, all the slow scrolling most of the time. I would say 95%, some things are still not working where you definitely should use one of the other solutions, but that is by far the best solution. You see here, I will not go any further details, but that's the way over I check when know how I do my screenshot, my checking, my visual test. Like I said, there's a lot of documentation how to do this, and you should definitely check it out or just check out my code and I will explain it a little bit there as well.

[00:16:13] Now, I'm going to show you the Cypress test itself, we are going to start this now. I put some pauses in that we really can see that one by one. Otherwise, the test might be too fast. This is the first one. Like I said, we check if the image itself looks, we expect that the image is visible and then what you already saw. We check that the image width, which is 700 in this case, is above one, which means the image was loaded.

[00:16:44] The next one we check is the styles CSS is loaded. Going to start this. You see here we're waiting that the styles CSS is loading. And immediately we see as well, it's loaded and it worked. The last one now, and that's why we have to see in this, we also put the pause. We again load the page and now we are going slowly through the page. You see on the left side slow scrolling slowly, slowly, slowly, 100 by 100.

[00:17:15] But you will see in the solution. And also how I will see it, in the end, is that this is the best solution of them all. Now I'm going to go into the Applitools tool to show you how it actually is, I will first show you the image. Load it. If you do. On the left side, you will always see the baseline. The baseline mean I just took a screenshot without any of my solution. In this case, when the website is actually loaded, you don't see much of the websites.

[00:17:44] It's only the upper part you see here and nothing more because all the other elements are loaded later. In this case, I'm checking the image load, which is this image,

I'm just turning off the differences because we have a lot. I'm checking if this image is loaded. The moment this image is loaded, I'm taking the screenshot. But what you now maybe see still not all like especially what I talked before.

[00:18:13] The articles are not shown, so we have the image and that is nice. I could do now for all different images could do the same image load, for example. Is it each anything else? But it takes a little longer. The second one which you talked about is wait resources. Also the same start result we just check and nothing happens. In this case, which is interesting. I'm checking the style CSS and the image is actually loaded. But again, all the other pictures.

[00:18:43] We have a big hand here. None of them are loaded. That's why I'm showing you my last solution. It's less slow scrolling, and when you see here, I will scroll a little bit. I will only show you the checkpoint as it's easier. All the images, the image up here, the image here, all the images to the articles, the video, and the other articles. All of them are shown because the slow scrolling really ensures for you. Yes, it adds seconds to your test depending on how long your page is. It will add a couple of seconds to the test, but it's worth it because what you saw in the comparison especially draw the comparison with the image loading. You just see so many elements, all the picture here and all the pictures here, they are just missing.

[00:19:33] And that's just so sad to miss out on this space and to miss out just to have a false positive because you see the moment you go through a slow scrolling, everything is there. The visual test would have told you there is a problem, but there isn't actually one. And it's really like I said, it is. You have to wait it what it's more important, but if you really want to ensure that your visual tests, if you really want, ensure that your website is on a high-quality level, also with a visual test and you don't want to have so many false-positive tests, you at this low scrolling and yes, it add some seconds, but it's really, really worth it.

[00:20:09] And it's like you saw it's really easy. It just takes the body height. I can go back to that. It's really so easy. It takes the body height and you just scroll. You can even change. You don't have to have the 100 here. Make it lower. I just had a hundred because I had some problems with really long loading images in other projects. That's why I added that here. But it can be less. And then you just could use the time off your test. But you ensure like you increase your quality, you increase your stability of your test and so so many ways. And I can only tell you that images are the worse to handle when it comes down to a visual test and you want to ensure that if you have a false or if you have an error with your visual test, it's because of really a functional problem and not because of any lazy loading images or anything. And that's all for today.

[00:21:02] Let's recap what I talked about. First, how to highlight text. I showed you in an amazing one liner how to easily and efficiently highlight text and talked about why it's so important to do this, especially when you have different editing options of formatting options for text. Second, I spoke about how to handle multiple elements that each function is the best function I ever discovered, and it's really amazing.

[00:21:26] It saves you hundreds of lines of code, and it makes it so easy to go through multiple elements, which are the same structure which just look the same. And it's really important to functionally test these elements, these multiple elements before you go into a visual test. And last but not least, I talked about all about images. I explained to you the three different ways how to handle images.

[00:21:50] First, wait for an image, especially if you just have one image, which is problematic. Just use that function. Just wait for that one image, and the moment it's loaded, you are ready to go for a visual test. Second, wait for specific resource, especially if you maybe have to engage with a special style button or anything. Use this because you can specifically wait for that one style for this button to load and then can just use it and engage in that. It doesn't always need to be for visual test.

[00:22:18] Maybe just function-wise that all the elements are loaded and last is slow scrolling. I showed you and I even showed you on my example how easy it is to add slow scrolling to your page and how important it is because you don't want to have any spinners in your visual test. You know that if 90 percent of your test fail because of your images are not loaded, nobody will take it serious anymore. And these tests are this adding to the test might add a couple of seconds, but you gain so much quality for your tests and you gain so much confidence, which is the most important in especially visual tests because you need to be confident to trust them.

[00:22:57] Because if 4 out of 10 visual tests or 4 out of 10 tests always failing at some point. You but also your whole team will not look at them anymore and will not take them seriously anymore. And you want to avoid that for any cost.

[00:23:10] I hope you guys could take something with you, learn something new, and can use all the amazing things, as it took quite a while to figure them out. And after I figured it out, a whole new world opened for me. Let's jump into the QA.