M102: Μηχανική Λογισμικού για Διαδικτυακές Εφαρμογές
Project: Διαδικτυακή Εφαρμογή *iWeatherCast*
Ομάδα Εργασίας: Π. Βασιλειάδης, Α. Καϊμακάμη, Α. Κωτσιαρή
Ημερομηνία: 01.2020

1. Html / CSS Validation

An HTML validator checks to make sure the HTML code on your web page complies with the standards set by the W3 Consortium, the organization that issues the HTML standards. There are various types of HTML validators: some only check for errors, while others also make suggestions about your code, telling you when it might lead to, unexpected results.

The W3 Consortium has its own online validator which you can use for free. It may be found at: https://validator.23.org/

A CSS validator checks your Cascading Style Sheet in the same manner. That is, it will check that it complies with the CSS standards set by the W3 Consortium. There are a few which will also tell you which CSS features are supported by which browsers (since not all browsers are equal in their CSS implementation).

You can get free validation for your style sheets from the W3 Consortium: https://jigsaw.w3.org/css-validator/

There are many other validator around, free and commercial, focusing on different aspects of your web page. You can find a list of free ones (including specialized validators like those that check your code for accessibility) from the Free HTML Validators, CSS Validators, Accessibility Validators page at https://www.thefreecountry.com/webmaster/htmlvalidators.shtml.

A1 Website Analyzer 4.0.4 - HTML and CSS validates all pages in a website using TIDY or W3C validators

Source: https://www.microsystools.com/products/website-analyzer/videos/website-html-css-validation/

Why Validate Your HTML and CSS Code?

There are a number of reasons why you should validate your page.

1. It Helps Cross-Browser, Cross-Platform and Future Compatibility

Although you may be able to create a web page that appears to work on your favorite browser (whatever that is), your page may contain HTML or CSS errors that do not show up with that browser due to an existing quirk or bug. Another person using a different browser that does not share that particular bug will end up viewing a page that does not show up correctly. It is also possible that later versions of your browser will fix that bug, and your page will be broken when people use its latest incarnation.

For Example: You can have a beautiful web page, that is aesthetically pleasing, An analogy is a new Mercedes Benz Cl65 AMG Coupe; beautiful car, in anyone's view. But what if you pull up the hood of that Mercedes and find the engine of a 1950 Ford Pick-up truck? This is the perfect analogy for a nice looking website, but it's coding doesn't work well in all browsers or devices.

Coding your pages so that it is correct without errors will result in pages that are more likely to work across browsers and platforms (for example, different systems.) It is also a form of insurance against future versions of browsers, since all browsers aim towards compliance with the existing HTML and CSS standards.

2. Search Engine Visibility

Browsers typically try to compensate when there are errors in a web page in different ways. Many ignore the broken elements while others make assumptions about what the web designer was trying to do. The problem is that when search engines obtain your page and try to parse them for keywords, they will also have to make certain decisions about what to do with the errors. Like browsers, different search engines will probably make different decisions about thos errors, resulting in certain parts of your web page (or perhaps even the entire page) not being indexed.

The best way to make sure the search engines see the page you want them to see is to present them an error free page. That way, there is no dispute about which part of your page comprises the content and which the formatting code.

Limitations: What Validation Does Not Do

Validating your web page does not ensure that it will appear the way you want it to. It merely ensures that your code is without HTML or CSS errors.

If you are wondering what the difference is, an analogy from normal human language will probably make it clear. If you look at this sentence, "Sarah a sandwich ate" which is grammatically incorrect when used in a non-poetic context. It can be fixed by simply reversing the order of the last two words so that the sentence reads "Chris ate a sandwich."

But lets see what happens if you write that same sentence and it says "Chris at a pie" which you means that he a sandwich? Syntactically, the sentence is correct, since all the elements of the sentence, subject, "Chris", ver "ate" and object "a pie" are in the right order. Semantically, however, the sentence describes a different thing from what you meant.

HTML and CSS validators are designed to catch the first type of error, exemplified by the grammatical error of my first sentence. So if you write HTML code that has, let's say, the wrong order, the HTML validator will spot it and tell you. However; it cannot catch errors of the second kind, where you get the spelling and order and all other technical aspects correct, but the code you used does not match the meaning you intended.

Ensuring that your code does what you want it to do requires you to actually test it in a web browser. Depending on the complexity of your code, you may even want to test it with different browsers to make sure that your site looks the same in all of them, since its possible that you are using features of HTML and CSS that are only implemented in some browsers but not others.

What to Do If You Don't Know HTML and CSS

If you used a visual web editor to design your site and are not familiar with HTML and CSS, you will face an additional problem.

While running the validator and getting it to validate your page itself will not be an issue, since the W3 Consortium's validator is not only free, it doesn't even have to be installed to be used), the problem comes when the validator checks your page and tells you that there are errors.

If you have no knowledge of HTML and CSS, you will probably have some difficulty figuring out what those errors mean, whether they are serious, and how to fix them.

Although there is no perfect solution to this, you are not completely without resources.

1. If you are using an editor like Dreamweaver, Microsoft's Expression Web, KompoZer or BlueGriffon, you can usually assume that the code they produce on their own is valid. From my experience, these four editors seem to create correct HTML and CSS code.

This means if you get errors when you validate your page, the problems must come from elsewhere. If you have inserted code that you obtained from a website (like if you added a Youtube video to your page), it's possible that the code is the source of the error message.

Conversely, if you have modified the code on the page manually, the error may have crept in there.

Because of that, basically, sometimes the error is benign. Let's use this example, if you added XHTML code to a page that has HTML, you may or may not get validation errors since you are mixing 2 different HTML families that have slightly different conventions. As far as I can tell, for the most part, this kind of error does not cause any problem for either browsers or search engines.

2. Another way is to search the Internet for the solution. You can copy and paste the error message given by the validator into a search engine and see if there are any websites out there are talk about this particular error. This may not be as good an idea as it if seems at first, since their solution may be too general to be helpful for your specific problem. Except if the error message is the result of your pasting code from some popular source (like Youtube or something of that level of popularity).

3. A third way is of course to ask someone, whether it's someone you know personally, or someone on the Internet. This solution also has its own issues, since you may get a solution that creates a bigger mess of your page than it had in the first place. It all boils down to their competence and willingness to spend enough time figuring out the problem.

4. Finally, you can ignore the problem. If you want to do this, you should test your web page in as many web browsers as you can to make sure the error message does not diagnose a problem that causes visible issue.s If you find

that your site seems to work fine in spite of the error, you may decided to just ignore it and hope for the best.

While this solution may not be ideal, you may be forced to take it if you can't find another way. It's not ideal because the error may bite you later when you least expect if. Like when there's a new version of some web browser that chokes on the bad code. It may also caus problems in a non-visible manner, such as in the way the search engines index your page.

How Often Should I Validate?

Some people validate every time they make a modification to their pages on the grounds that careless mistakes can occur any time. Others validate only when they make a major design change.

It's good to validate the template for pages when you make a major design change. Try to validate your pages each time you make modifications. Admittedly, sometimes you may forget to do so with the occasional disastrous consequences: Murphy's Law doesn't spare webmasters.

Having an offline validator helps to make sure that you remember to validate: having to go online just to validate your pages tends to make people put off validation until later, ending up that oftentimes they simply forget. For those who are not familliar with the terminology "offline validator", it simply means a validator that can be dowloaded and installed on your computer so that you can run it on your pages without having to go to the W3 Consortium's website. You can find offline validators on the free validator page mentioned earlier. https://www.thefreecountry.com/webmaster/htmlvalidators.shtml

The HTML Tidy validator (listed on that page) is available for numerous platforms (including Linux, Mac, Windows) and has proven itself to webmasters worldwide.

Conclusion

It's generally a good idea to validate your web page. It will point you to errors that may affect how your website is understood by web browsers and search engines. Even if you ae not familiar with HTML and CSS, there are still some ways you can deal with the errors that you discover from validating your page.

Our clients, allsportdesigns.com and sailradios.com have just had their templates completely recoded by MakDigitalDesign.com. Recoding their templates made them a lot more competitive and viable. Their websites were designed with the very poor standards. They are smart entrepreneurs and realized they must evolve with technology to stay ahead in the ever-changing world of both business and technology.

Do you due diligence when it comes to your website. In today's marketplace, most sales are made using computers, tablets and cellphones. Your website should be able to adapt and evolve also. And of course, VALIDATE, where your site is parked!

Source: https://www.makdigitaldesign.com/bigcommerce-store-development/validate-web-page-html-css-validation/

What is HTML and CSS Validation? Should You Validate Your Web Page?

Whether you design your web page using a visual web editor like Expression Web, Dreamweaver or BlueGriffon, or you code HTML directly with a simple text editor, the generally recommended practice is to validate it after you finish designing it.
This article discusses what validation means, points you to some of the free tools that you can use, and deals with its limitations and the problems that a new webmaster may face.

What does Validating HTML or CSS Mean?

For those unfamiliar with the term, "validating" a page is just a jargon-filled way of referring to the use of a computer program to check that a web page is free of errors.
In particular, an **HTML validator** checks to make sure the HTML code on your web page complies with the standards set by the W3 Consortium, the organisation ("organization" if you use a different variant of English) that issues the HTML standards. There are various types of HTML validators: some only check for errors, while others also make suggestions about your code, telling you when it might lead to (say) unexpected results.
The W3 Consortium has its own online validator which you can use for free. It may be found at: https://validator.w3.org/
A **CSS validator** checks your Cascading Style Sheet in the same manner. That is, it will check that it complies with the CSS standards set by the W3 Consortium. There are a few which will also tell you which CSS features are

supported by which browsers (since not all browsers are equal in their CSS implementation).
Again, you can get free validation for your style sheets from the W3 Consortium: https://jigsaw.w3.org/css-validator/
There are numerous other validators around, both free and commercial, focusing on different aspects of your web page. You can find a list of free ones (including specialised validators like those that check your code for accessibility) from the Free HTML Validators, CSS Validators, Accessibility Validators page at https://www.thefreecountry.com/webmaster/htmlvalidators.shtml

Why Validate Your HTML and CSS Code?
There are a number of reasons why you should validate your page.

1. **It Helps Cross-Browser, Cross-Platform and Future Compatibility**

Although you may be able to create a web page that appears to work on your favourite browser (whatever that may be), your page may contain HTML or CSS errors that do not show up with that browser due to an existing quirk or bug. Another person using a different browser that does not share that particular bug will end up viewing a page that does not show up correctly. It is also possible that later versions of your browser will fix that bug, and your page will be broken when people use its latest incarnation.
Coding your pages so that it is correct without errors will result in pages that are more likely to work across browsers and platforms (ie, different systems). It is also a form of insurance against future versions of browsers, since all browsers aim towards compliance with the existing HTML and CSS standards.

2. **Search Engine Visibility**

When there are errors in a web page, browsers typically try to compensate in different ways. Some may ignore the broken elements while others make assumptions about what the web designer was trying to achieve. The problem is that when search engines obtain your page and try to parse them for keywords, they will also have to make certain decisions about what to do with the errors. Like browsers, different search engines will probably make different decisions about those errors, resulting in certain parts of your web page (or perhaps even the entire page) not being indexed.
The safest way to make sure the search engines see the page you want them to see is to present them an error-free page. That way, there is no dispute about which part of your page comprises the content and which the formatting code.
Limitations: What Validation Does Not Do
Validating your web page does not ensure that it will appear the way you want it to. It merely ensures that your code is without HTML or CSS errors.

If you are wondering what the difference is, an analogy from normal human language will hopefully make it clear. Let's take this sentence "Chris a sandwich ate" which is grammatically incorrect when used in a non-poetic context. It can be fixed by simply reversing the order of the last two words so that the sentence reads "Chris ate a sandwich".

But what happens if you write a sentence that says "Chris ate a pie" when you meant that he ate a sandwich? Syntactically, the sentence is correct, since all the elements of the sentence, subject ("Chris"), verb ("ate") and object ("a pie") are in the right order. Semantically, however, the sentence describes a different thing from what you meant.

HTML and CSS validators are designed to catch the first type of error, exemplified by the grammatical error of my first sentence. So if you write HTML code that has (say) the wrong order, the HTML validator will spot it and tell you. However, it cannot catch errors of the second kind, where you get the spelling and order and all other technical aspects correct, but the code you used does not match the meaning you intended.

Ensuring that your code does what you want it to do requires you to actually test it in a web browser. Depending on the complexity of your code, you may even want to test it with different browsers to make sure that your site looks the same in all of them, since it's possible that you are using features of HTML and CSS that are only implemented in some browsers but not others.

What to Do If You Don't Know HTML and CSS

If you have designed your site using a visual web editor, and are not familiar with HTML and CSS, you will face an additional problem.

While running the validator and getting it to validate your page itself will not be an issue (since the W3 Consortium's validator is not only free, it doesn't even have to be installed to be used), the problem comes when the validator checks your page and tells you that there are errors.

If you have no knowledge of HTML and CSS, you will probably have some difficulty figuring out what those errors mean, whether they are serious, and how to fix them.

Although there is no perfect solution to this, you are not completely without resources.

1. If you are using an editor like Microsoft Expression Web, Dreamweaver, BlueGriffon or KompoZer, you can usually assume that the code they produce on their own is valid. From my limited experience (mainly creating demo sites for the purpose of writing tutorials or reviews for thesitewizard.com), these four editors seem to create correct HTML and CSS code.

This means that if you get errors when you validate your page, the problems must come from elsewhere. If you have inserted code that you obtained from

a website (such as if you have added a Youtube video to your page), it's possible that the code is the source of the error message.
Alternatively, if you have modified the code on the page manually, the error may have crept in there.
Having said that, sometimes the error is benign. For example, if you have added XHTML code to a page that has HTML, you may or may not get validation errors since you are mixing 2 different HTML families that have slightly different conventions. As far as I can tell, for the most part, this kind of error does not cause any problem for either browsers or search engines.

2. Another way is to search the Internet for the solution. For example, you can copy and paste the error message given by the validator into a search engine, and see if there are any websites out there that talk about this particular error. This may not be as fantastic an idea as it first appears, since their solution may be too general to be helpful for your specific problem, unless the error message is the result of your pasting code from some popular source (like Youtube or something of that level of popularity).
3. A third way is of course to ask someone, whether it's someone you know personally, or someone on the Internet. This solution also has its own issues, since you may get a solution that creates a bigger mess of your page than it had in the first place. It all boils down to their competence and willingness to spend enough time figuring out the problem.
4. Finally, you can also ignore the problem. If you want to do this, you should test your web page in as many web browsers you can to make sure the error message does not diagnose a problem that causes visible issues. If you find that your site seems to work fine in spite of the error, you may decide to just ignore it and hope for the best.

Although this solution is not ideal, you may be forced to take it if you can't find an alternative. It's not ideal because the error may bite you later when you least expect it, for example, when there's a new version of some web browser that chokes on the bad code. It may also cause problems in a non-visible manner, such as in the way the search engines index your page.

How Often Should I Validate?

Some people validate every time they make a modification to their pages on the grounds that careless mistakes can occur any time. Others validate only when they make a major design change.
I always validate the template for my pages when I make a major design change. I try to validate my pages each time I make modifications, although I must admit that I sometimes forget to do so (with the occasional disastrous consequence; Murphy's Law doesn't spare webmasters).

I find that having an offline validator helps to make sure that I remember to validate: having to go online just to validate my pages tends to make me put off validation till later, with the result that it'll occasionally get overlooked. For those not familiar with the terminology I use, when I say "offline validator" I simply mean a validator that I can download and install in my own computer so that I can run it on my pages without having to go to the W3 Consortium's website. You can find offline validators on the free validators page I mentioned earlier.
The HTML Tidy validator (listed on that page) is available for numerous platforms (including Linux, Mac, Windows, etc) and has proven helpful to many webmasters the world over.

Conclusion
As I mentioned above, it's generally a good idea to validate your web page. It will point you to errors that may affect how your website is understood by web browsers and search engines. Even if you are not familiar with HTML and CSS, there are still some ways you can deal with the errors that you discover from validating your page.

Source: https://www.thesitewizard.com/webdesign/htmlvalidation.shtml

index.html

```
<html>
<head>
        <title>Welcome Page - iWeatherCast</title>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <link rel="icon" href="Resources\Old\haze.png" />

        <script type="text/javascript" src="JS/script.js"></script>
        <link rel="stylesheet" type="text/css"  href="CSS/style.css">


        <link                                                          rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

        <link                                                          rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">

        <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>

        <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>

</head>
<body onload="getLocation();">
<ul>
  <li><a href="..\">Logout <img src="Resources\log-out.png"></a></li>
</ul>
        <div class="bg-img">
        <div class="Container" id="parse1">
                <div     class="row"><div     class="col-md-12     col-sm-6">     <input
style="visibility: collapse;" type="text" name="city" id="city" placeholder="Enter City
Name" class="form-control"> </div></div>
                <div     class="row"><div     class="col-md-12     col-sm-6">     <button
style="visibility:  collapse;"  name="getWeather"  id="getWeather"  onclick="parseJson()"
class="btn btn-primary btn-block">Get Weather</button> </div></div>
                <div class="row"><div class="col-md-12 col-sm-6"> <button name="Search"
onclick="makeVisible()" class="btn btn-primary btn-block" id="SearchVisibility">Search
another city</button> </div></div>
                <!--   <div   class="row"><div   class="col-md-12   col-sm-6">   <button
name="getForcaste"   onclick="parseJson2()"   class="btn   btn-primary   btn-block">Get
Forcaste</button> </div></div> -->
                <!-- </div> -->
                <br>
                <h2 id="nointernet"></h2>
                <div class="board" id="blur-bg" style="visibility: collapse;">
                        <div   class="row"><div   class="col-md-12"><p   id="last_update"
style="visibility: collapse;"></p></div>
                        <div class="row"><div class="col-md-4"></div><div class="col-md-
2  col-sm-1"><p  id="Temperature"  style="font-style:  bold;font-size:23pt"></div><div
class="col-md-3 col-sm-2"><h2 id="City"></h2></div><div class="col-md-2 col-sm-2"><img
class="img-responsive" id="flag" style="visibility: hidden;"></div><div class="col-md-
2 col-sm-1"></div></div>

                        <div   class="row"><div   class="col-md-4   col-sm-2"></div><div
class="col-md-4 col-sm-8"></p></div><div class="col-md-4 col-sm-2"></div></div>

                        <div   class="row"><div   class="col-md-4   col-sm-2"></div><div
class="col-md-4  col-sm-8"><img  id="image"  style="visibility:  hidden;"  class="img-
responsive"></img></div><div class="col-md-5 col-sm-2"></div></div>

                        <div   class="row"><div   class="col-md-5   col-sm-2"></div><div
class="col-md-4  col-sm-8"><h4  id="Climate"></h4></div><div  class="col-md-5  col-sm-
2"></div></div>
```

```
                      <div   class="row"><div   class="col-sm-4   col-md-5"></div><div
class="col-sm-1    col-md-1"><img    id="temp_h"    src="Resources/low-temperature.svg"
class="img-responsive med-img" style="visibility: hidden;"><p id="htemp"></p></div><div
class="col-sm-2    col-md-2"><img    id="temp_l"    src="Resources/high-temperature.svg"
class="img-responsive med-img" style="visibility: hidden;"><p id="ltemp"></p></div><div
class="col-sm-4 col-md-6"></div></div>
                      <!-- <hr> -->
                      <div   class="row"><div   class="col-md-5   col-sm-6"></div><div
class="col-md-4        col-sm-4"><h4        id="wind-title"        style="visibility:
hidden;">Wind</h4></div><div class="col-md-4 col-sm-4"></div></div>
                      <div   class="row"><div   class="col-md-5   col-sm-2"></div><div
class="col-md-4  col-sm-8"><img  src="Resources/navigation.svg"  class="img-responsive
med-img" id="wind-img" style="visibility: hidden;"></div><div class="col-md-2 col-sm-
2"></div></div>
                      <div   class="row"><div   class="col-md-5   col-sm-2"></div><div
class="col-md-2   col-sm-8"><p   id="wind"></p></div><div   class="col-md-4   col-sm-
4"></div></div>

                      <!-- Trend Starts -->
                      <div class="row">
                      <div      class="col-md-12"><h3      style="padding:      30px;"
id="forecast">Forecast</h3>
                      </div></div>

                      <div class="row">
                              <div class="col-md-2"></div>
                              <div class="col-md-8">
                              <table align="center">
                                      <th><p
align="center"><b>Time</b></p></th><th>&ensp;</th><th><p><b>Temperature</b></p></th><t
h>&ensp;&ensp;</th><th><p><b>Weather</b></p></th><th>&ensp;&ensp;</th><th><p><b>Descri
ption</b></p></th>
                                      <tr><td><p
id="date1"></p></td><td>&ensp;&ensp;</td><td><p
id="temp1"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im1"/></td><td>&ensp;</td><td><p id="we1"></p></td></tr>
                                      <tr><td><p
id="date2"></p></td><td>&ensp;&ensp;</td><td><p
id="temp2"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im2"/></td><td>&ensp;</td><td><p id="we2"></p></td></tr>
                                      <tr><td><p
id="date3"></p></td><td>&ensp;&ensp;</td><td><p
id="temp3"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im3"/></td><td>&ensp;</td><td><p id="we3"></p></td></tr>
                                      <tr><td><p
id="date4"></p></td><td>&ensp;&ensp;</td><td><p
id="temp4"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im4"/></td><td>&ensp;</td><td><p id="we4"></p></td></tr>
                                      <tr><td><p
id="date5"></p></td><td>&ensp;&ensp;</td><td><p
id="temp5"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im5"/></td><td>&ensp;</td><td><p id="we5"></p></td></tr>
                                      <tr><td><p
id="date6"></p></td><td>&ensp;&ensp;</td><td><p
id="temp6"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im6"/></td><td>&ensp;</td><td><p id="we6"></p></td></tr>
                                      <tr><td><p
id="date7"></p></td><td>&ensp;&ensp;</td><td><p
id="temp7"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im7"/></td><td>&ensp;</td><td><p id="we7"></p></td></tr>
                                      <tr><td><p
id="date8"></p></td><td>&ensp;&ensp;</td><td><p
id="temp8"></p></td><td>&ensp;</td><td><img                      class="img-responsive"
id="im8"/></td><td>&ensp;</td><td><p id="we8"></p></td></tr>
                              </table>
                      </div>
              </div>


              </div>

      <!-- </div> -->
      </div>
</body>

</html>
```

HTML validation results using Nu Html Checker
https://validator.w3.org/nu/

1. **Warning:** Consider adding a lang attribute to the html start tag to declare the language of this document.

   From line 1, column 1; to line 1, column 6

   <html>↩<head

   For further guidance, consult Declaring the overall language of a page and Choosing language tags.

   If the HTML checker has misidentified the language of this document, please file an issue report or send e-mail to report the problem.

2. **Error:** Start tag seen without seeing a doctype first. Expected <!DOCTYPE html>.

   From line 1, column 1; to line 1, column 6

   <html>↩<head

3. **Error:** Bad value Resources\Old\haze.png for attribute href on element link: Backslash ("\") used as path segment delimiter.

   From line 7, column 2; to line 7, column 50

   ale=1">↩↩ **<link rel="icon" href="Resources\Old\haze.png" />**↩↩<sc

4. **Warning:** The type attribute is unnecessary for JavaScript resources.

   From line 9, column 2; to line 9, column 51

   png" />↩↩ **<script type="text/javascript" src="JS/script.js">**</scri

5. **Error:** Bad value ..\ for attribute href on element a: Backslash ("\") used as path segment delimiter.

   From line 24, column 7; to line 24, column 20

   ul>↩ <li>**<a href="..\">**Logout

6. **Error:** Bad value Resources\log-out.png for attribute src on element img: Backslash ("\") used as path segment delimiter.

   From line 24, column 28; to line 24, column 60

   \">Logout **<img src="Resources\log-out.png">**</a></

7. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 24, column 28; to line 24, column 60

\">Logout <img src="Resources\log-out.png"></a></
```

8. **Warning:** Empty heading.

```
From line 34, column 3; to line 34, column 22

↩ <br>↩ <h2 id="nointernet"></h2>↩
```

9. **Error:** CSS: font-style: bold is not a font-style value.

```
From line 37, column 80; to line 37, column 139

col-sm-1"><p id="Temperature" style="font-style: bold;font-size:23pt"></div>
```

10. **Warning:** Empty heading.

```
From line 37, column 177; to line 37, column 190

col-sm-2"><h2 id="City"></h2><
```

11. **Error:** Element img is missing required attribute src.

```
From line 37, column 233; to line 37, column 298

col-sm-2"><img class="img-responsive" id="flag" style="visibility: hidden;"></div>
```

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the element
decoding — Decoding hint to use when processing this image for presentation

12. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 37, column 233; to line 37, column 298
```

```
col-sm-2"><img          class="img-responsive"          id="flag"
style="visibility: hidden;"></div>
```

13. **Error**: No p element in scope but a p end tag seen.

```
From line 39, column 89; to line 39, column 92

col-sm-8"></p></div>
```

14. **Error**: Element img is missing required attribute src.

```
From line 41, column 89; to line 41, column 155

col-sm-8"><img    id="image"    style="visibility:    hidden;"
class="img-responsive"></img>

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

15. **Error**: An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 41, column 89; to line 41, column 155

col-sm-8"><img    id="image"    style="visibility:    hidden;"
class="img-responsive"></img>
```

16. **Error**: Stray end tag img.

```
From line 41, column 156; to line 41, column 161

sponsive"></img></div>
```

17. **Warning**: Empty heading.

```
From line 43, column 89; to line 43, column 105

col-sm-8"><h4 id="Climate"></h4><
```

18. **Error**: An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 45, column 89; to line 45, column 200

col-md-1">**<img id="temp_h" src="Resources/low-temperature.svg" class="img-responsive med-img" style="visibility: hidden;">**<p id=

19. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 45, column 256; to line 45, column 368

col-md-2">**<img id="temp_l" src="Resources/high-temperature.svg" class="img-responsive med-img" style="visibility: hidden;">**<p id=

20. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 48, column 89; to line 48, column 197

col-sm-8">**<img src="Resources/navigation.svg" class="img-responsive med-img" id="wind-img" style="visibility: hidden;">**</div>

21. **Error:** The align attribute on the table element is obsolete. Use CSS instead.

From line 59, column 5; to line 59, column 26

d-8">↩ **<table align="center">**↩

22. **Error:** th start tag in table body.

From line 59, column 27; to line 60, column 9

="center">↩ **<th>**<p ali

23. **Error:** The align attribute on the p element is obsolete. Use CSS instead.

From line 60, column 10; to line 60, column 27

↩ <th>**<p align="center">**<b>Tim

24. **Error:** Element img is missing required attribute src.

From line 61, column 104; to line 61, column 141

;</td><td>**<img class="img-responsive" id="im1"/>**</td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource

srcset — Images to use in different situations (e.g., high-resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the element
decoding — Decoding hint to use when processing this image for presentation

25. **Error**: An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 61, column 104; to line 61, column 141

```
;</td><td><img class="img-responsive" id="im1"/></td><
```

26. **Error**: Element img is missing required attribute src.

From line 62, column 104; to line 62, column 141

```
;</td><td><img class="img-responsive" id="im2"/></td><
```

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the element
decoding — Decoding hint to use when processing this image for presentation

27. **Error**: An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 62, column 104; to line 62, column 141

```
;</td><td><img class="img-responsive" id="im2"/></td><
```

28. **Error**: Element img is missing required attribute src.

From line 63, column 104; to line 63, column 141

```
;</td><td><img class="img-responsive" id="im3"/></td><
```

```
Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

29. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 63, column 104; to line 63, column 141

;</td><td><img class="img-responsive" id="im3"/></td><
```

30. **Error:** Element img is missing required attribute src.

```
From line 64, column 104; to line 64, column 141

;</td><td><img class="img-responsive" id="im4"/></td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

31. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 64, column 104; to line 64, column 141

;</td><td><img class="img-responsive" id="im4"/></td><
```

32. **Error:** Element img is missing required attribute src.

```
From line 65, column 104; to line 65, column 141

;</td><td><img class="img-responsive" id="im5"/></td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

33. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 65, column 104; to line 65, column 141

;</td><td><img class="img-responsive" id="im5"/></td><
```

34. **Error:** Element img is missing required attribute src.

```
From line 66, column 104; to line 66, column 141

;</td><td><img class="img-responsive" id="im6"/></td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

35. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 66, column 104; to line 66, column 141
```

```
;</td><td><img class="img-responsive" id="im6"/></td><
```

36. **Error:** Element img is missing required attribute src.

```
From line 67, column 104; to line 67, column 141

;</td><td><img class="img-responsive" id="im7"/></td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

37. **Error:** An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

```
From line 67, column 104; to line 67, column 141

;</td><td><img class="img-responsive" id="im7"/></td><
```

38. **Error:** Element img is missing required attribute src.

```
From line 68, column 104; to line 68, column 141

;</td><td><img class="img-responsive" id="im8"/></td><

Attributes for element img:
Global attributes
alt — Replacement text for use when images are not available
src — Address of the resource
srcset — Images to use in different situations (e.g., high-
resolution displays, small monitors, etc.)
sizes — Image sizes for different page layouts
crossorigin — How the element handles crossorigin requests
usemap — Name of image map to use
ismap — Whether the image is a server-side image map
width — Horizontal dimension
height — Vertical dimension
referrerpolicy — Referrer policy for fetches initiated by the
element
decoding — Decoding hint to use when processing this image for
presentation
```

39. **Error**: An img element must have an alt attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.

From line 68, column 104; to line 68, column 141

;</td><td>**<img class="img-responsive" id="im8"/>**</td><

40. **Error**: End tag for body seen, but there were unclosed elements.

From line 78, column 1; to line 78, column 7

>↩ </div>↩**</body>**↩↩</ht

41. **Error**: Unclosed element div.

From line 27, column 2; to line 27, column 36

bg-img">↩ **<div class="Container" id="parse1">**↩ <di

42. **Error**: Unclosed element div.

From line 26, column 2; to line 26, column 21

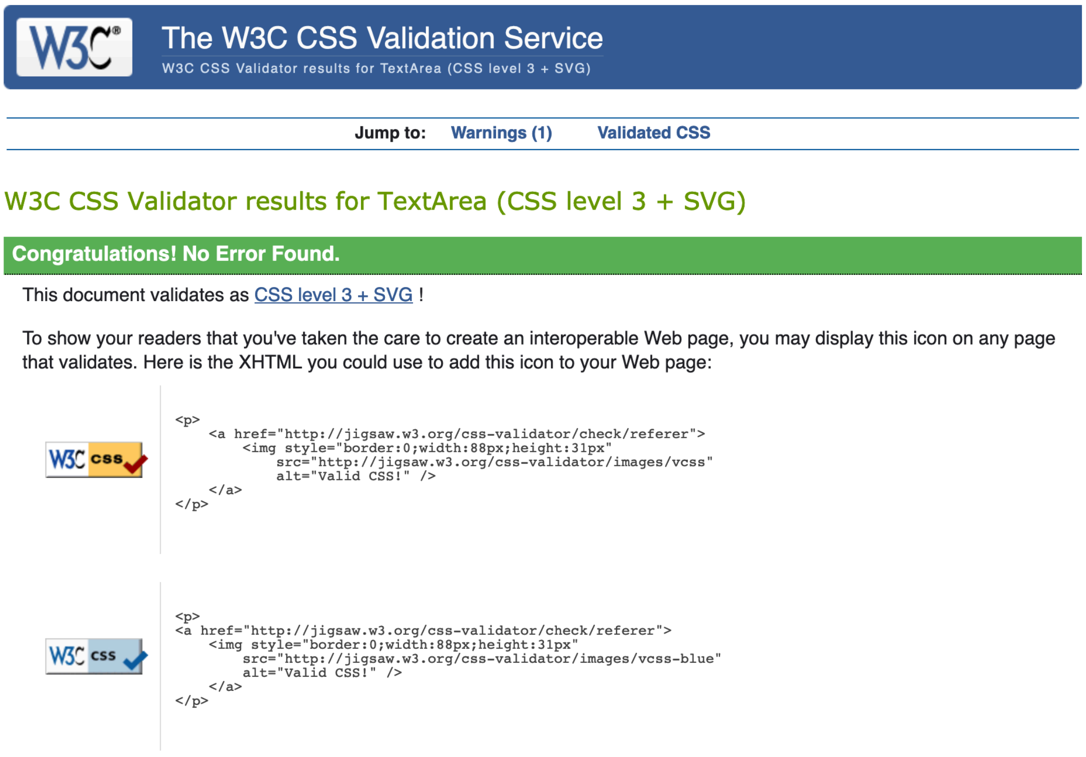
i>↩</ul>↩ **<div class="bg-img">**↩ <div

style.css

```
/*@font-face{
        font-family: 'B612';
        src: url('../Font/B612/B612-Regular.ttf')format('truetype');
}
*/

@import
url('https://fonts.googleapis.com/css?family=Anton|Gayathri:100,400,700|Lexend+Deca|Mo
ntserrat&display=swap');

*{
        /*font-family: 'Lexend Deca',sans-serif; Anton*//*Lexend Deca*/
}

.bg-img{
        background-image: url("../Resources/background.png");
        background-repeat: no-repeat;
        background-size: 100% 100%;

}


body{
        padding: 10px;
        height: 100%;
        color: #ffffff !important;
        font-family: 'Montserrat','sans-serif' !important;
}
.cool {
  height: 75px;
  width: 75px;
  background-color: #006eff;
  border-radius: 50%;
}

.mild {
  height: 75px;
  width: 75px;
  background-color: #00ddff;
  border-radius: 50%;
}

.warm {
  height: 75px;
  width: 75px;
  background-color: #ffae00;
  border-radius: 50%;
}

.hot {
  height: 75px;
  width: 75px;
  background-color: #ff5000;
  border-radius: 50%;
}

.vhot {
  height: 75px;
  width: 75px;
  background-color: #ff0000;
  border-radius: 50%;
}

#temperature,#temp1,#temp2,#temp3,#temp4,#temp5,#temp6,#temp7,#temp8{
        padding: 20px;
        color: #ffffff;
        /*margin-left: auto;*/
}

#image{
        height: 250px;
        width: 250px;
        margin-left: auto;
        margin-right: auto;
}

footer{
        font-family: 'Anton','sans-serif' !important;
        /*position: fixed;*/
        left: 0;
        bottom: 0;
        width: 100%;
        background-color: rgba(255, 255, 255, 0.62);
        padding: 10px;
        color: black;
}


.med-img{
        height: 25px;
        width: 25px;
}
#wind-img,#wind{
        height: 50px;
        width: 50px;
        /*margin-left: auto;*/
        /*margin-right: auto;  */
}


.board{
        background-color: rgba(0, 0, 0, 0.28);
        padding: 5px;
}

.container{
        margin: 10px;
}
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
  text-decoration: none;
}

li {
  float: right;
  border-right:1px solid #bbb;
  text-decoration: none;
}

li:last-child {
  border-right: none;
  text-decoration: none;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color: #4CAF50;
}
```

CSS validation results

http://jigsaw.w3.org/css-validator/



The W3C CSS Validation Service

W3C CSS Validator results for TextArea (CSS level 3 + SVG)

Jump to: Warnings (1) Validated CSS

W3C CSS Validator results for TextArea (CSS level 3 + SVG)

Congratulations! No Error Found.

This document validates as CSS level 3 + SVG !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page:

```
<p>
    <a href="http://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="http://jigsaw.w3.org/css-validator/images/vcss"
            alt="Valid CSS!" />
    </a>
</p>
```

```
<p>
<a href="http://jigsaw.w3.org/css-validator/check/referer">
    <img style="border:0;width:88px;height:31px"
        src="http://jigsaw.w3.org/css-validator/images/vcss-blue"
        alt="Valid CSS!" />
    </a>
</p>
```

2. UI / Functionality Testing

UI Testing is a process of testing the application's graphical user interface to ensure proper functionality as per the specifications. It involves checking the application components like buttons, icons, checkboxes, color, menu, windows etc.
Visual dynamics of a web application play a pivotal role in the acceptance of an application with the user.

User Interface Testing

To ensure visual aesthetic of web application is well accepted, UI and Usability testing become a key aspect of the overall QA practice. Any application which can be accessed through an URL is a Web-based application. In such applications, we mainly test the front end of the application which is to be used by the end user.
Each browser displays web pages differently, so it is important that page should look same on different browsers. If a webpage is displayed distorted and unmanaged then it will lead viewers to exit the webpage. So a website should undergo UI testing for better results.

Browser testing comprises of below two types:

Functionality testing

Testing of different functions throughout the application. It involves validating all the navigations as well as all field values which are present in the front end pages using all positive as well as negative scenarios.

UI Testing

Testing the look and feel factor of the web page. Look and feel factor includes display type, font, alignment, radio button, checkbox etc.

- Areas covered in UI testing are Usability, Look & feel, Navigation controls/navigation bars, instructions, and technical information style, images, tables, accessibility etc.
- For testing for accessibility, we have to check with W3C-Web content accessibility guidelines.

Approach for UI testing

We select a subset of test cases from functional test cases which covers all functionalities of the application.
The second step is to modify those test cases according to UI testing requirements.
Next step will be executing those test cases; comparing the result with expected results, and if there is any difference then raising the issue for the same. It is not feasible to test in all browsers. Normally the client decides in which browser there is the requirement to test.

As we know each browser display web page differently, so we cannot expect all browsers to display web page exactly similar.
For example, drop-down in windows-firefox will be different that mac-firefox. Such issues are acceptable, as these are operating system utilities and we need to accept them as such.
Base Browser: Normally application is developed targeting a browser which is expected to be used mostly by end users, it is termed as the base browser.

Commonly Occurring UI Defects

- Button alignment issues
- Inconsistent space between labels or text boxes
- Broken labels i.e. single line label getting displayed in two lines
- Misalignment among text boxes, info icons, labels or dropdowns
- Overlapping of fields
- Incomplete fields
- Data on the page is misaligned; some time-shifted upwards or downwards
- In any browser, while selecting some action, the corresponding action is not happening
- Resizing not working as expected
- Session expiry time either very short or very long for some browsers
- Browser specific issues – Few fields are not editable after entering data in one browser but editable in another browser

Key UI and Usability Test Requirements

The key UI testing requirements of the web application are:

1. Availability of various components in a UI
2. Various states of the UI component

Component:
A component is a building block, which can be used with the combination of several other components to form an application. The Components can be reused across the application.
Examples of a component include Button, Text Field, Autosuggest, Checkbox, Dropdown etc.

GUI Testing Tools
Several tools are available in the technology world that would help testers in UI testing.
Ranorex Studio is a complete test automation solution for desktop, web, and mobile apps. Write your tests in the full IDE, or automate fast with the friendly capture-and-replay tool.

Get reliable GUI object recognition (even for dynamic web elements), and built-in test reporting. Use CSV, Excel files, or a SQL database to drive your tests. Includes a customizable test report and Selenium Webdriver built-in. Integrates with your CI/CD/DevOps pipeline.

- Selenium
- HP Unified Functional Testing
- Cucumber
- Coded UI
- Sahi

Sample GUI Test Cases

1) Verify the working of carousel arrows and way finders
2) Verify that the password field is accepting values only in a masked state
3) Verify that the 'save' button remains inactive until all required fields are entered
4) Verify that the user is allowed to navigate to the top of the page using 'Top' bar
5) Verify that proper message is displayed when the applied filters do not retrieve any results
6) Verify the navigation from links available in Headers and footers
7) Verify the alignment of radio buttons is accurate
8) Verify that multiple options in checkboxes can be selected at a time
9) Verify that the title of each section is in bold letters
10) Verify the color change of hyperlinks on clicking

Conclusion

A website is the soul of many businesses. It is very necessary to make sure that it looks fine and functions similarly on different browsers & platforms. Thus UI testing is very important and it will ensure a large customer base and addition in business value.

Source: https://www.softwaretestinghelp.com/gui-testing/

3. Usability Testing

Usability definition is “the ease of use of an object”. In the web realm, it can be defined as “the ease with which your users navigate and reach their goals within your website”.

Users want to be able to find the information they need as fast and as effortless as possible. They have a low tolerance for sites that are complicated and slow and it takes them 10 seconds to leave one website to find a better alternative one. For all these reasons, usability is a core concept in interface designing.

It is aiming to evaluate interfaces’ usability. Such a practice is called “website usability testing”.

Usability is generally measured by selecting potential users and making them try the product by giving them specific tasks. Let’s say you are measuring the usability of a traveling website, then you may ask your testers to try to follow your instructions and book a trip. Let’s now dig into website usability testing knows and hows.

- **Moderated Testing.** The test is done under specific conditions and location with an “observer” also called moderator, guiding the users, making questions, and taking notes. The participant and the observer are usually in a lab or a quiet place where dialogue can happen without interferences.
- **Moderated Remote Testing.** Quite the same as above but done remotely. This means that users and observers are not in the same place but on a video call sharing screens. There are many software that enable this kind of testing.
- **Unmoderated Remote Testing**. In this case, the users can test the website thanks to online tools whenever suits them best. The results are only later analyzed, so there’s no need for a moderator. This last method gives, of course, great flexibility to usability testing.

The most diffused ones:

- **Heatmaps** (click tracking) allow you to track users’ mouse movements on the screen.
- **A/B testing** consists instead of comparing the users’ experience of two different versions of the same website.
- **Surveys** are a quick way to get direct feedback from users on your website usability.

- **Paper prototyping** conveniently allows developers to test websites in the development phase and can spare them lots of time and efforts.
- **Eye tracking** devices allow you to follow the motion of the eyes of users navigating the website.

Say you want to test your e-commerce website (loveshoes.com) as you noticed your site has a lower number of visitors compared to your competitors. You've noticed that people often visit the homepage, they start choosing the shoe size but often drop out. You want to investigate this issue. Once you have your participant ready you give her a task and then make a series of questions.

Task:
Please go on lovetshoes.com and a buy the red shoes on sale, size 5.

Questions:
1. On a scale from 1 to 10 how **easy** was buying the shoes?
2. If you found any, **when** did you experience **difficulties**?
3. Do you find the website **trustworthy**?
4. Do you like the design of the **interface**?

Thanks to this test you then realize that users struggle a lot when they have to select the size for the shoes. You realize that this function is not usable at all and it plays a very important role in the user experience. Now you know what you have to work on to make the website great!

Website usability checklist
Here is a brief bullet point list to guide you through the usability testing process.
1. **Identify** which website pages/areas/features you really want to investigate: what **objectives** you're aiming for with the test.
2. **Identify** potential **participants**. What kind of users does your website target, where can you find them?
3. Write a list of **tasks and questions** you will present to the participants.
4. **Test** your website usability! Remember to test website usability with different screen resolutions and on different devices.
5. **Analyze** the results by always keeping in mind your objectives.

Mind that this list can also be considered a mobile website usability testing checklist (mobile app testing).

What to do after website usability testing is complete: the web usability report

So you made your best to set up your usability test. You collected data, results, and feedback. What's next? What should be your next move? Here are 5 steps that can guide you through the post-testing phase.

1. Take the results you collected and **organize** them. Find similarities between the answers you collected, this way you'll be able to reduce the data. Then intuitively display them and draw conclusions.
2. Make a list where you rank top **priority** issues and less urgent ones. This way you will have a clearer **plan of action.**
3. Compare your **objectives** with the outcome of your test. Do the areas you wanted to investigate need improvements or are they good enough? Did you find some unexpected areas and features that need improvements instead?
4. Create a **website usability report** to communicate your findings to the people involved in the development and design of your website.
5. Take **action**, start from top priority issue and make your website as good and usable as possible. Creating a website testing checklist excel sheet can be great at this stage.

Like in most things, being organized and keeping track of all data is vital. Something as easy as an excel sheet checklist can be useful in guiding you into website improvements. An example of website usability checklist can feature as columns these properties:

- The **issues** resulting from the test
- A brief **description** of each issue
- The website pages and **areas** affected by the issues
- **Possible solutions** to the problem

Source: https://www.userreport.com/blog/website-usability-testing/

4. Performance Testing

Performance Testing checks the speed, response time, reliability, resource usage, scalability of a software program under their expected workload. The purpose of Performance Testing is not to find functional defects but to eliminate performance bottlenecks in the software or device.
The focus of Performance Testing is checking a software program's

- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads

Performance Testing is popularly called "Perf Testing" and is a subset of performance engineering.

Why do Performance Testing?

Performance Testing

Features and Functionality supported by a software system is not the only concern. A software application's performance like its response time, reliability, resource usage and scalability do matter. The goal of Performance Testing is not to find bugs but to eliminate performance bottlenecks.

Performance Testing is done to provide stakeholders with information about their application regarding speed, stability, and scalability. More importantly, Performance Testing uncovers what needs to be improved before the product goes to market. Without Performance Testing, software is likely to suffer from issues such as: running slow while several users use it simultaneously, inconsistencies across different operating systems and poor usability.

Performance testing will determine whether their software meets speed, scalability and stability requirements under expected workloads. Applications sent to market with poor performance metrics due to nonexistent or poor performance testing are likely to gain a bad reputation and fail to meet expected sales goals.

Also, mission-critical applications like space launch programs or life-saving medical equipment should be performance tested to ensure that they run for a long period without deviations.

According to Dunn & Bradstreet, 59% of Fortune 500 companies experience an estimated 1.6 hours of downtime every week. Considering the average Fortune 500 company with a minimum of 10,000 employees is paying $56 per hour, the labor part of downtime costs for such an organization would be $896,000 weekly, translating into more than $46 million per year. Only a 5-minute downtime of Google.com (19-Aug-13) is estimated to cost the search giant as much as $545,000. It's estimated that companies lost sales worth $1100 per second due to a recent Amazon Web Service Outage. Hence, performance testing is important.

Types of Performance Testing

- **Load testing -** checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- **Stress testing -** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- **Endurance testing -** is done to make sure the software can handle the expected load over a long period of time.
- **Spike testing -** tests the software's reaction to sudden large spikes in the load generated by users.
- **Volume testing** - Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
- **Scalability testing** - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Common Performance Problems

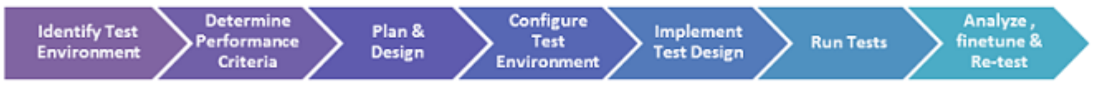
Most performance problems revolve around speed, response time, load time and poor scalability. Speed is often one of the most important attributes of an application. A slow running application will lose potential users. Performance testing is done to make sure an app runs fast enough to keep a user's attention and interest. Take a look at the following list of common performance problems and notice how speed is a common factor in many of them:

- **Long Load time -** Load time is normally the initial time it takes an application to start. This should generally be kept to a minimum. While some applications are impossible to make load in under a minute, Load time should be kept under a few seconds if possible.

- **Poor response time -** Response time is the time it takes from when a user inputs data into the application until the application outputs a response to that input. Generally, this should be very quick. Again if a user has to wait too long, they lose interest.
- **Poor scalability -** A software product suffers from poor scalability when it cannot handle the expected number of users or when it does not accommodate a wide enough range of users. Load Testing should be done to be certain the application can handle the anticipated number of users.
- **Bottlenecking -** Bottlenecks are obstructions in a system which degrade overall system performance. Bottlenecking is when either coding errors or hardware issues cause a decrease of throughput under certain loads. Bottlenecking is often caused by one faulty section of code. The key to fixing a bottlenecking issue is to find the section of code that is causing the slowdown and try to fix it there. Bottlenecking is generally fixed by either fixing poor running processes or adding additional Hardware. Some **common performance bottlenecks** are
    - CPU utilization
    - Memory utilization
    - Network utilization
    - Operating System limitations
    - Disk usage

Performance Testing Process

The methodology adopted for performance testing can vary widely but the objective for performance tests remain the same. It can help demonstrate that your software system meets certain pre-defined performance criteria. Or it can help compare the performance of two software systems. It can also help identify parts of your software system which degrade its performance.
Below is a generic process on how to perform performance testing


Identify Test Environment
Determine Performance Criteria
Plan & Design
Configure Test Environment
Implement Test Design
Run Tests
Analyze , finetune & Re-test

1. **Identify your testing environment -** Know your physical test environment, production environment and what testing tools are available. Understand details of the hardware, software and network configurations used during testing before you begin the testing process. It will help testers create more efficient tests. It will also help identify possible challenges that testers may encounter during the performance testing procedures.
2. **Identify the performance acceptance criteria -** This includes goals and constraints for throughput, response times and resource

allocation. It is also necessary to identify project success criteria outside of these goals and constraints. Testers should be empowered to set performance criteria and goals because often the project specifications will not include a wide enough variety of performance benchmarks. Sometimes there may be none at all. When possible finding a similar application to compare to is a good way to set performance goals.

3. **Plan & design performance tests -** Determine how usage is likely to vary amongst end users and identify key scenarios to test for all possible use cases. It is necessary to simulate a variety of end users, plan performance test data and outline what metrics will be gathered.
4. **Configuring the test environment -** Prepare the testing environment before execution. Also, arrange tools and other resources.
5. **Implement test design -** Create the performance tests according to your test design.
6. **Run the tests -** Execute and monitor the tests.
7. **Analyze, tune and retest** - Consolidate, analyze and share test results. Then fine tune and test again to see if there is an improvement or decrease in performance. Since improvements generally grow smaller with each retest, stop when bottlenecking is caused by the CPU. Then you may have the consider option of increasing CPU power.

Performance Testing Metrics: Parameters Monitored
The basic parameters monitored during performance testing include:

- **Processor Usage -** an amount of time processor spends executing non-idle threads.
- **Memory use -** amount of physical memory available to processes on a computer.
- **Disk time -** amount of time disk is busy executing a read or write request.
- **Bandwidth -** shows the bits per second used by a network interface.
- **Private bytes -** number of bytes a process has allocated that can't be shared amongst other processes. These are used to measure memory leaks and usage.
- **Committed memory -** amount of virtual memory used.
- **Memory pages/second -** number of pages written to or read from the disk in order to resolve hard page faults. Hard page faults are when code not from the current working set is called up from elsewhere and retrieved from a disk.

- **Page faults/second -** the overall rate in which fault pages are processed by the processor. This again occurs when a process requires code from outside its working set.
- **CPU interrupts per second -** is the avg. number of hardware interrupts a processor is receiving and processing each second.
- **Disk queue length -** is the avg. no. of read and write requests queued for the selected disk during a sample interval.
- **Network output queue length -** length of the output packet queue in packets. Anything more than two means a delay and bottlenecking needs to be stopped.
- **Network bytes total per second -** rate which bytes are sent and received on the interface including framing characters.
- **Response time -** time from when a user enters a request until the first character of the response is received.
- **Throughput -** rate a computer or network receives requests per second.
- **Amount of connection pooling -** the number of user requests that are met by pooled connections. The more requests met by connections in the pool, the better the performance will be.
- **Maximum active sessions -** the maximum number of sessions that can be active at once.
- **Hit ratios -** This has to do with the number of SQL statements that are handled by cached data instead of expensive I/O operations. This is a good place to start for solving bottlenecking issues.
- **Hits per second -** the no. of hits on a web server during each second of a load test.
- **Rollback segment -** the amount of data that can rollback at any point in time.
- **Database locks -** locking of tables and databases needs to be monitored and carefully tuned.
- **Top waits -** are monitored to determine what wait times can be cut down when dealing with the how fast data is retrieved from memory
- **Thread counts -** An applications health can be measured by the no. of threads that are running and currently active.
- **Garbage collection -** It has to do with returning unused memory back to the system. Garbage collection needs to be monitored for efficiency.

Performance Test Tools

There are a wide variety of performance testing tools available in the market. The tool you choose for testing will depend on many factors such as types of the protocol supported, license cost, hardware requirements, platform support etc. Below is a list of popularly used testing tools.

- LoadNinja – is revolutionizing the way we load test. This cloud-based load testing tool empowers teams to record & instantly playback comprehensive load tests, without complex dynamic correlation & run these load tests in real browsers at scale. Teams are able to increase test coverage. & cut load testing time by over 60%.
- NeoLoad **-** is the performance testing platform designed for DevOps that seamlessly integrates into your existing Continuous Delivery pipeline. With NeoLoad, teams test 10x faster than with traditional tools to meet the new level of requirements across the full Agile software development lifecycle - from component to full system-wide load tests.
- HP LoadRunner **-** is the most popular performance testing tools on the market today. This tool is capable of simulating hundreds of thousands of users, putting applications under real-life loads to determine their behavior under expected loads. Loadrunner features a virtual user generator which simulates the actions of live human users.
- Jmeter **-** one of the leading tools used for load testing of web and application servers.

Source: https://www.guru99.com/performance-testing.html

5. Compatibility Testing

Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or Mobile devices.
Compatibility Testing is a type of Non-functional testing

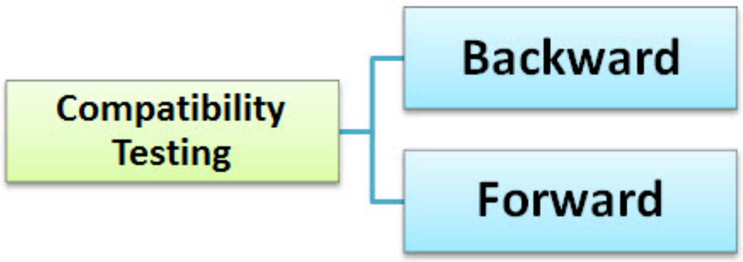
Types of Compatibility Tests

- Hardware
- Operating Systems
- Software
- Network
- Browser
- Devices
- Mobile
- Versions

Compatibility testing types issues:

- **Hardware**: It checks software to be compatible with different hardware configurations.
- **Operating Systems**: It checks your software to be compatible with different Operating Systems like Windows, Unix, Mac OS etc.
- **Software**: It checks your developed software to be compatible with other software. For example, MS Word application should be compatible with other software like MS Outlook, MS Excel, VBA etc.
- **Network:** Evaluation of performance of a system in a network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application in different networks with all parameters mentioned earlier.
- **Browser**: It checks the compatibility of your website with different browsers like Firefox, Google Chrome, Internet Explorer etc.
- **Devices**: It checks compatibility of your software with different devices like USB port Devices, Printers and Scanners, Other media devices and Blue tooth.
- **Mobile**: Checking your software is compatible with mobile platforms like Android, iOS etc.
- **Versions of the software:** It is verifying your software application to be compatible with different versions of the software. For instance checking your Microsoft Word to be compatible with Windows 7, Windows 7 SP1, Windows 7 SP2, Windows 7 SP3.

There are two types of version checking:



Backward compatibility Testing is to verify the behavior of the developed hardware/software with the **older versions** of the hardware/software.
Forward compatibility Testing is to verify the behavior of the developed hardware/software with the **newer versions** of the hardware/software.

Tools for Compatibility Testing

1. BrowserStack - Browser Compatibility Testing: This tool helps a Software engineer to check application in different browsers.
2. Virtual Desktops - Operating System Compatibility: This is used to run the applications in multiple operating systems as virtual machines. n Number of systems can be connected and compare the results.

How to do Compatibility Testing

1. The initial phase of compatibility testing is to define the set of environments or platforms the application is expected to work on.
2. The tester should have enough knowledge of the platforms/software/hardware to understand the expected application behavior under different configurations.
3. The environment needs to be set-up for testing with different platforms, devices, networks to check whether your application runs well under different configurations.
4. Report the bugs. Fix the defects. Re-test to confirm Defect fixing.

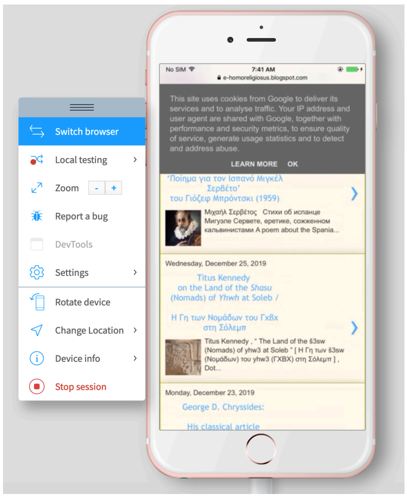
Conclusion:
The most important use of compatibility testing is to ensure whether developed software works under different configurations (as stated in requirements documentation). This testing is necessary to check whether the application is compatible with the client's environment.

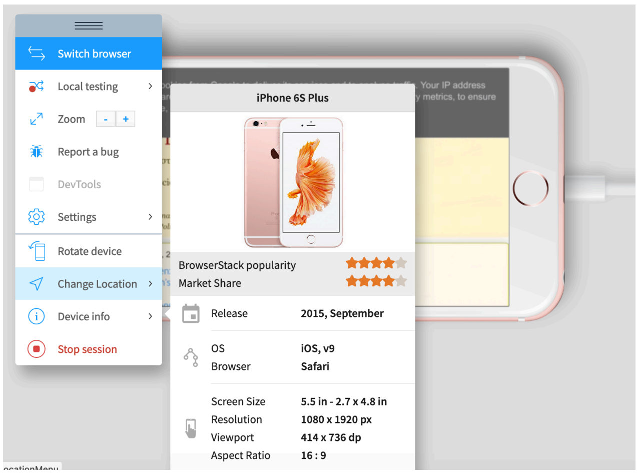
Source: https://www.guru99.com/compatibility-testing.html

Case study:

Δημιουργία λογαριασμού στο www.browserstack.com για τον έλεγχο σε διάφορα λειτουργικά συστήματα, φυλλομετρητές, είδη συσκευών, κλπ.

On BrowserStack Live, you can instantly test your website on 2,000+ desktop browsers and real mobile devices. Say goodbye to your internal lab of devices and VMs, forever (yes, we mean it!).

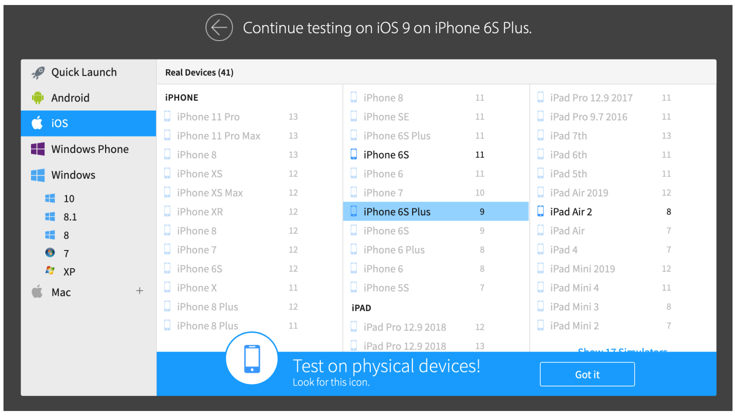




iPhone 6S Plus

BrowserStack popularity ★★★★☆
Market Share ★★★★☆

Release	2015, September
OS	iOS, v9
Browser	Safari
Screen Size	5.5 in - 2.7 x 4.8 in
Resolution	1080 x 1920 px
Viewport	414 x 736 dp
Aspect Ratio	16 : 9



Continue testing on iOS 9 on iPhone 6S Plus.

Real Devices (41)

iPHONE		iPhone 8	11	iPad Pro 12.9 2017	11
iPhone 11 Pro	13	iPhone SE	11	iPad Pro 9.7 2016	11
iPhone 11 Pro Max	13	iPhone 6S Plus	11	iPad 7th	13
iPhone 8	13	iPhone 6S	11	iPad 6th	11
iPhone XS	12	iPhone 6	11	iPad 5th	11
iPhone XS Max	12	iPhone 7	10	iPad Air 2019	12
iPhone XR	12	iPhone 6S Plus	9	iPad Air 2	8
iPhone 8	12	iPhone 6S	9	iPad Air	7
iPhone 7	12	iPhone 6 Plus	8	iPad 4	7
iPhone 6S	12	iPhone 6	8	iPad Mini 2019	12
iPhone X	11	iPhone 5S	7	iPad Mini 4	11
iPhone 8 Plus	12	iPAD		iPad Mini 3	8
iPhone 8 Plus	11	iPad Pro 12.9 2018	12	iPad Mini 2	7
		iPad Pro 12.9 2018	13		

6. Security Testing

Security Testing is a type of software testing that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders.

Focus Areas

There are four main focus areas to be considered in security testing (Especially for web sites/applications):

- **Network security:** This involves looking for vulnerabilities in the network infrastructure (resources and policies).
- **System software security:** This involves assessing weaknesses in the various software (operating system, database system, and other software) the application depends on.
- **Client-side application security:** This deals with ensuring that the client (browser or any such tool) cannot be manipulated.
- **Server-side application security:** This involves making sure that the server code and its technologies are robust enough to fend off any intrusion.

Example

This is an example of a very basic security test which anyone can perform on a web site/application:

- Log into the web application.
- Log out of the web application.
- Click the BACK button of the browser (Check if you are asked to log in again or if you are provided the logged-in application.)

Most types of security testing involve complex steps and out-of-the-box thinking but, sometimes, it is simple tests like the one above that help expose the most severe security risks.

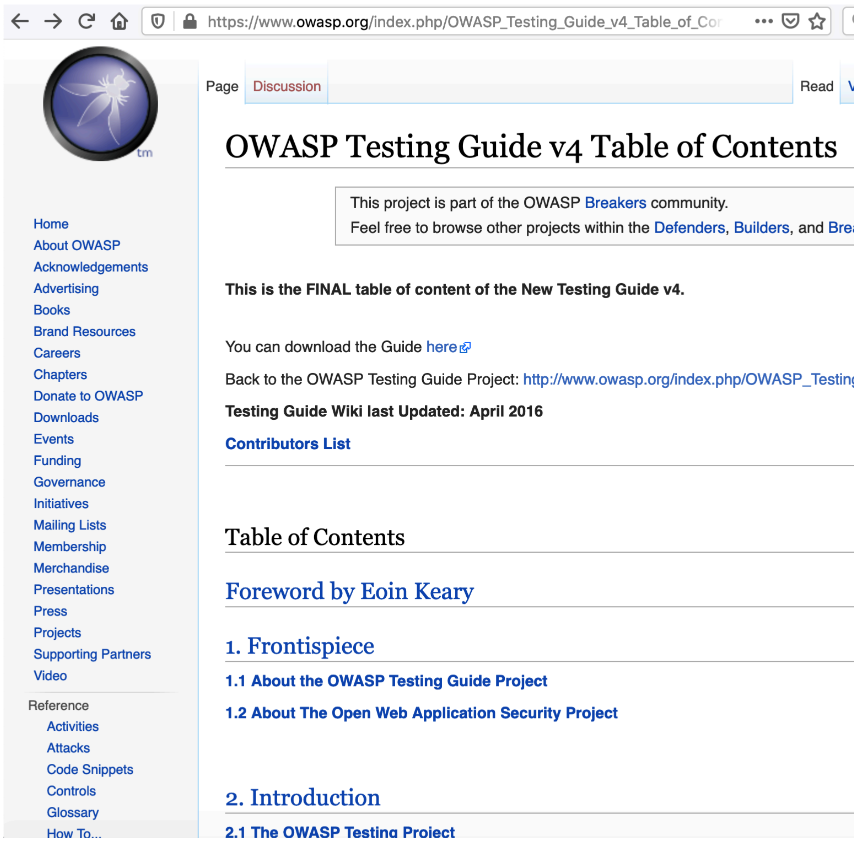
OWASP

The Open Web Application Security Project (OWASP) is a great resource for software security professionals. Be sure to check out the Testing Guide: https://www.owasp.org/index.php/Category:OWASP_Testing_Project

Source: http://softwaretestingfundamentals.com/security-testing/

OWASP Testing Guide v4 Table of Contents

https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents



https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Co

Page Discussion Read

Home
About OWASP
Acknowledgements
Advertising
Books
Brand Resources
Careers
Chapters
Donate to OWASP
Downloads
Events
Funding
Governance
Initiatives
Mailing Lists
Membership
Merchandise
Presentations
Press
Projects
Supporting Partners
Video

Reference
Activities
Attacks
Code Snippets
Controls
Glossary
How To...

OWASP Testing Guide v4 Table of Contents

This project is part of the OWASP Breakers community.
Feel free to browse other projects within the Defenders, Builders, and Bre

This is the FINAL table of content of the New Testing Guide v4.

You can download the Guide here

Back to the OWASP Testing Guide Project: http://www.owasp.org/index.php/OWASP_Testin

Testing Guide Wiki last Updated: April 2016

Contributors List

Table of Contents

Foreword by Eoin Keary

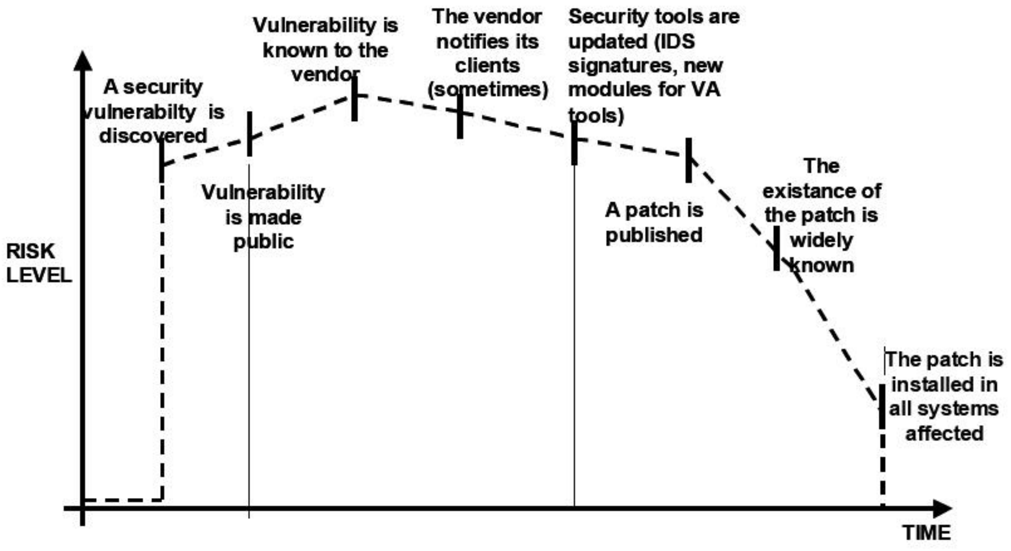
1. Frontispiece

1.1 About the OWASP Testing Guide Project

1.2 About The Open Web Application Security Project

2. Introduction

2.1 The OWASP Testing Project


RISK LEVEL
TIME
A security vulnerabilty is discovered
Vulnerability is made public
Vulnerability is known to the vendor
The vendor notifies its clients (sometimes)
Security tools are updated (IDS signatures, new modules for VA tools)
A patch is published
The existance of the patch is widely known
The patch is installed in all systems affected

Window of Vulnerability

Προτεινόμενη βιβλιογραφία:

1. Ιστότοπος *Software Testing Fundamentals*: http://softwaretestingfundamentals.com/
2. A. Mathur, *Foundations of Software Testing: Fundamental Algorithms and Techniques*. Pearson Education India, 2007.
3. M. L. Hutcheson, *Software Testing Fundamentals Methods and Metrics*. Indianapolis, IN: Wiley, 2003.
4. H. Bernard, *Fundamentals of Software Testing*. Wiley-iSTE, 2012.