INVITATION TO WEB PROGRAMMING

José del Carmen Rodríguez Santamaría
https://ejristos.com
Bogotá, Colombia

March 3, 2025

INTRODUCTION

In the same way as there are too many human languages for our communication, many languages exist for our communication with digital machines and computers. It is impossible to speak of the best option, but it is easier to speak of the most popular solutions in the last years. At present, and for the basic web document to be run on the Internet, the dominant language by preference is **HTML** (Hyper Text Mark Language). This is what we will learn. Moreover, it is mandatory to do programs that produce nice pages. To that aim, we will use **CSS** (Cascading Style Sheets). And to make dynamic adjustments and to run our own calculations, we will use **JS** (JavaScript).

Programming is difficult but here we promise to make things accessible.

We start the machine in chapter 1, *My first HTML program*. We see there the basic structure of a program in HTML, and the appropriate locations for CSS and JS **scripts**, or pieces of code.

In the second chapter, *a web-app for a real need*, we build a ladder of programs of growing complexity to fill in a specific task: to serve as a web tutor for the learning of articles and plurals in the German language. This is a program that is currently used by the Author and serves as an illustrator of all interactive applications that are self-contained and that allow a conversation between the User and the browser:

The program challenges the User with a german noun. Her or his task is to choose the right article among three options,

masculine, feminine and neutral. If the User chooses the right article, the program responds with a congratulating message and a new challenge is given. But if the User responds in the wrong way, he or she receives a red message and more opportunities are given.

In the third chapter, *My Web Site*, we succinctly show the path to make a web site.

Bogotá, Colombia, *José Rodríguez*
February 2025

Chapter 1

MY FIRST HTML PROGRAM

In our first program we express a simple message. After understanding the code, we will learn to run it.

1 Example. *The following html program prints a welcome message to the screen.*

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>MyFirstWebProgram.html</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
    <h1>MY FIRST WEB PROGRAM</h1>
    <p>Welcome to a learning adventure!</p>
    <p>We will learn step by step
        how to make a tutor for learning
        articles and plurals of the German language.
    </p>
</body>

</html>
```

2 Exercise. *Verify that in the code above:*

- For every <, there is a >. Text within <> is called a **tag**. A tag contains signals for the HTML interpreter that is used by the web browser. A **compiler** is a program that receives as input a program in a language that human beings can readily understand and translates it to a language that the computer circuity can execute.
- The tag <!doctype html> is single and denotes a global directive: the document is written in HTML. This must be specified because HTML is the dominant language for the web, but not the unique.
- The <html lang="en"> marks **the beginning of the html document**, and that the used char-set is that of english.
- After the declaration of HTML as the official language, the document has two parts: the head and the body. Global directives for the browser and classifying agencies, such as Google, come in the **head**. They may contain information to localize your document by title, author, date, topic, keywords plus instructions for styling the document. The **body** contains the instructions specific to the given document.
- The tag <meta charset="utf-8" /> is single and specifies that we publish a document in the ordinary encoding of the web. The 8 does not refer to the size of the symbols on a screen but to the memory size in bits that are allocated for each symbol.
- There are paired tags, marking a starting point that is followed by some code and that must necessarily end at some point.
- When a tag is paired, it is open with < > and closed with < />. For instance, <html> and </html>, <head> and

</head>, <body> and </body>. The tag <html> is the beginning of the overall document while </html> denotes the end of the html document. Similar rules for <head> and <body> that define separated sections.

Programs may be difficult to understand. To ease this task, one can do two things: write a documentation as a separate document, like this, and insert clarifications directly into the page as comments. Some experts consider code clearer when comments and external documentation become superfluous and unnecessary. They exhort everybody to write clear code that needs no explanations. This is difficult.

3 Example. *The following is the commented version of our first program:*

```
<!---This is a comment in html, an explanation for the human reader.  -->
<!---The next line contains a declaration for the Browser. It says:
    This is a document in HTML language. -->
<!doctype html>

<!---The next is an instruction for the web browser. It says:
    This is the beginning of the  HTML document
    and the used  alphabet is that of English-->
<html lang="en">

<!---After the initial declaration,
    an HTML program includes two sections:
    1. The head, with global directives for the browser
        and classifying agencies.
    2. The body, with specific instructions.  -->


<!---Beginning of the head with diverse directions -->

<head>
    <meta charset="utf-8"> <!---Type of encoding for the used chars -->
    <!---Title displayed in the bar of the browser -->
    <title>MyFirstWebProgramComments.html</title>
    <!--Used width of display-->
    <meta name="viewport" content="width=device-width, initial-scale=1">

</head>
<!---End of the head -->



<!---Beginning of the body that contains diverse programming -->

<body>
    <!---A title is printed to the display -->
    <!---The text is formatted as h1, bold, large -->
    <h1>MY FIRST WEB PROGRAM</h1>
    <!---An ordinary text to be displayed is formatted as a paragraph -->
    <p>Welcome to a learning adventure!</p>
    <p>We will learn step by step
        how to make a tutor for learning
        articles and plurals of the German language.
    </p>
    <!---End of the body -->
</body>

<!---end of html text -->
```

```
<!--Almost every html tag or instruction
    begins with < >  and ends with <  />
 -->

</html>
```

All html documents have a basic structure.

1.1 THE HTML TEMPLATE

The basic structure of an html document is an immutable skeleton but, anyway, some tiny changes can be made as in the declared language.

4 Example. *The following is the template or basic skeleton that defines the common structure to all html programs. This is the HTML template into which the code can be inserted:*

```
<!doctype html>
<html lang="en">

 <head>
  <meta charset="utf-8" />
  <title>HTMLtemplate</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
  <style>
             ...
  </style>
 </head>

 <body>
  <script type="text/javascript">
             ...
  </script>
 </body>

</html>
```

We can see the appropriate location as for the styling CSS instructions as for our personal JavaScript programming.

5 Exercise. *Insert comments into the appropriate location of the basic html skeleton.* Answer

1.2 LET US RUN OUR FIRST PROGRAM!

We will see two methods to run the programs of this volume: the first for persons that do not have much time and the second for persons that want to gain independence and have time to study and practice.

Our first method is as follows:

All programs of this volume were zipped by the Author into the file `InvitationWebProgramming.zip` that accompany this PDF and that can also be downloaded directly. That file must be extracted, and each program can be run separately in one instant. This file also contains the Latex source of this PDF. This is done because you have the right to do something better.

6 Receipt. *The procedure to run an HTML program that already exists in a given directory is as follows:*

1. To run a saved program on a desktop: open your preferred browser, open a new tab, click over the open window, press at the same type the keys CTRL and O, where the O means OPEN (it is not a zero). At this prompt, a dialogue that contains your directories and files will appear. Choose your file and click over the CHOOSE or OPEN or equivalent button. The program will run immediately.

2. To run the code on Mobile: copy-paste the code to a new document and save it with the name `MyFirstProgram.html`. If you already have the program as a file on a desktop, attach the saved file to a mail to yourself, save it on mobile. To run the program, just tap on its name.

Our second method is the following:

A program is not an intention. It is a fact. This means that it is designed to be run and must run seamlessly. We will use a browser, anyone, to run our program. That is possible if

the program already exists as a file in the memory of the used computer machine, tablet or cell phone. Thus, our first task is to save an appropriate file.

All programs in HTML, JS, and CSS are encoded as plain text so, they can be typed and read with the most rudimentary text processors. All that is needed is to save programs with the correct suffix: `.html` for HTML, `.js` for JS and `.css` for CSS, but we will use only one type of file, with suffix `.html`, in which everything is included, html directrices, JS methods or procedures or functions, and CSS styling. Hence, to create an html document our procedure is: use any text processor, and save your work as a file with suffix `.html`.

7 Receipt. *To save a file with our first program, do this:*

1. *Highlight the code directly over this pdf document and copy it to the clipboard.*
2. *Paste it to a new document in any word processor.*
3. *Save it with the suffix* `.html` *as:* `MyFirstProgram.html`

A correctly written html code can be run in any browser. So, the **Developer**, the person that makes computer programs, must take care of all possible incompatibilities.

8 Exercise. *Run program* `MyFirstProgram.html`*. You shall see a welcome message and an invitation.*

The WEB has been built over the criterion that the privacy and freedom of the User is the fundament of communication. That is why you have the right to know exactly what you are running. To empower that right, browsers have the possibility to show the code. In Firefox, one just needs to right click in the window that contains the program and select `Show the source code` or equivalent. In other browsers one may need

to look for tools for Developers and choose to view the source code among the various offers.

9 Challenge. *Programming is difficult. It would better to say that it is extremely difficult. So, many Developers argue that they do not want you to copy and reuse in an instant the code that they developed during large months of hard work. Research on what they do to safeguard their work but complying approximately with the requirement for total freedom and the right to know what one is running.*

10 Exercise. *Save and run a file with the following code that represents an html document with all possible complexities: html skeleton, css styling and JS code. Pay attention to the fact that each language has its own markers for the comments. The program adds two numbers: 1+2 = 3.*

```
<!doctype html>
<html lang="en">


<head>
    <meta charset="utf-8" />

    <!-- Title of this HTML document -->
    <title>MyFirstCSS_JSprogram</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>
        /* This is a comment in css */
        /* For an HTML document, Styling is done in css */
        /* If the background is not not declared, it is white,
           otherwise, it must be given a color */
        body {
            background-color: lightblue;
        }
    </style>
</head>

<body>
    <!-- Prints an important message into the page, a substring appears in italic -->
    <h1>My first <i>html-css-js program </i> for the browser</h1>
    <!-- A line in the document is given a name for future calling.
        A message is printed to that line -->
    <p id="demo">Click the button and see the effect here.</p>


    <script>
        //This JS script adds two numbers and reports the result to the browser
        //declaration of two variables
        var a = 1;
        var b = 2;

        //This functions adds two numbers and returns the result to the main thread of the program.
        function sum(a, b) {
            return a + b;
        }

        //Function sum(a,b) is called and output written to the browser
        function report() {
            //Variables a and b are printed, the sum method is called and all is joined into one single string which is the answer.
```

```
            let answer = a.toString() + " + " + b.toString() + " = " + sum(a, b);
            //The answer is printed to the document.
            document.getElementById("demo").innerHTML = answer;
        }
    </script>

    <!-- A button must be clicked on to run the script -->
    <button type="button" onclick="report()">Try it</button>

</body>

</html>
```

How does the interaction between HTML and CSS work? Every HTML element has a list of properties that the User can control. A CSS program is a specification of some of these properties. Therefore, it is a list with assigned values. The Author has not been interested in understanding the details, so his method is to ask the web for a specific question, see which code is available, choose the one that fits best, copy, paste, modify, and modify again until it comes up with something acceptable. An expert is someone who knows the why's and the how's and knows how to play with them.

1.3 SPECIALIZED EDITORS

If one tries to understand the code of a complex program using an ordinary text processor, one rapidly becomes a hero. The reason is that it is extremely difficult to distinguish what is what. So, specialized editors have been developed that assign different colors to different lines of code depending on their roles. They are marvelous.

The easiest way to become acquainted with one of such editors is to find one for the used browser. In Mozilla Firefox, one looks for an extension with theme `html editor`. From various options, I tried HTML5 Editor. It is enough to visualize the code that can be made more ordered with the bar option *beautify code*. One can play with it to see how to open a file, how it runs, how to modify it, how to save, i.e., how to download it but with an ad hoc name given by the extension. One

must rapidly rename it with an appropriate, easy to remember name that points to its function. For our first program, the right name is `MyFirstProgram.html`. To run a program, one just clicks the run button in the bar up right. The HTML5 Editor has an advantage: it manages the HTML, CSS, JS parts separately in distinct files. Such division is frequently used and loved because different parts have diverse degree of complexity and easy sections get spontaneously frozen and put apart.

With more time and interest, one can install a free professional code editor. Most persons single out the *Visual Studio Code* by Microsoft as the best free code editor. Other renowned options are Notepad++, Eclipse, Netbeans. The Author uses all of these, even at the same time, but Studio Code is his preferred option. To run saved programs, he uses Firefox or Chrome under Ubuntu. Windows has its own options, also very good. To edit a program, one follows the same rules as for an ordinary document: one make changes, saves them, tests them in the browser, makes the corrections.... again and again until one is done.

Notepad++ is native on Windows but can be run under Ubuntu with Win. It is necessary to select HTML as the Language in the menu Languages. This must be done for each program. This will activate the syntax highlighting, and it is easier to save the program with the `.html` suffix.

1.4 BROWSERS MAY BE DIFFERENT

For very basic tasks, different browsers behave identically. But, in spite of the high uniformity, there are differences. That is why it is normal that a slightly complex application that was developed for Firefox over a desktop does not run in the cell phone which by default has Chrome. So, each browser offers its own documentation. That of Mozilla Firefox is in

https://Developer.mozilla.org/en-US/

The documentation contains explanations for every detail of their implementation of HTML, CSS, JavaScript and some other more advanced items. One must use this documentation to learn and to consult questions of the most varied nature. And to review what once was leaned. For instance, one can see a list of the various areas that HTML covers and that CSS can embellish.

1.5 CORRECTING PROGRAMS

Programming is difficult. This assertion has two meanings. First: programming is one of the most exciting enterprises because each step is a challenge, whose solution produces great joy. Second: one or two women in the world can develop a not-too-short program that runs immediately. The rest of us must suffer a lot to make our programs run because we always commit too many errors, flaws or bugs.

An identified flaw in a computer program is called a **bug**. This naming comes from history:

The ENIAC (Electronic Numerical Integrator and Computer), the first electronic computer, was made of tubes and relays. It was made by the military during the Second World War to simulate ballistic trajectories under real conditions but began operation in 1946. One morning, the computer did not function. After revising everything, a bug, an insect, was found interfering with a relay. From that day on, every trouble in a software application is called a bug.

Bugs in self-contained programs have two main sources: misconceptions and ill encoding. You are doomed to commit many errors or bugs before you can make a program run, and much more to make it output what it must. Yes: programming is difficult. So, let us try to understand where the difficulty comes

from.

11 The origin of bugs *Apart from the tight syntax of all actual programming languages, the difficulty of programming stems from three very simple facts:*

- There are too many options to modify.
- Most modifications render programs useless.
- Minor changes that work produce very diverse effects or otherwise the language would lack enough expressiveness.
- We say: all languages, human or not, are chaotic else very incipient.

Hence, it is extremely improbable to pick up one change that serves a predetermined purpose. That is why you must study a lot to succeed, think very finely, be very careful, and be armed with unbeatable patience.

The next exercise is a very important challenge because it forces you to look beyond your noise.

12 Exercise. *Our theory is directly related to our origin, the origin of mankind. According to Science, our origin is explained by the Evolutionary Theory of the Origin of Species. Our theory and experience has something to say:*

1. *Prove that the DNA is software, i.e., that it contains encoded instructions to make things.*
2. *Prove that the DNA programming language has all the chaotic properties of our programming languages.*
3. *Prove that the Evolutionary Theory of the origin of species is obviously false.*

Answer

Now, what can be done to ameliorate our suffering with the immanency of bugs?

The community has developed special tools for the purpose. For instance, Studio Code immediately detects syntactic bugs, which paints in red the line where the error is caught. An instance of a syntactic bug happens when a command of the language is misspelled, say, if instead of typing

`<title>This is my Title</title>` one types
`<title>This is my Title<title/>`

Studio Code and other editors are extensible, they allow the addition of appropriate extensions to help with nice formatting of the code to ease the reading and interpretation. Thus, one hovers over the icons to discover that corresponding to extensions, and one prints as search prompt: `html, css, js`. A lot of options are displayed and one chooses one with many stars and many functions.

The most terrible bugs are those in which the syntax is correct but the program does not run, or runs but breaks the purpose of the Developer. These bugs are generically called *run-time errors*.

There are special tools for **debugging**, the processes of clearing out bugs, but the Author prefers to detect errors by a simple rule: **a program is known by its fruits**. So, when something does not work or works but not as desired, the Author inserts special code to print to the console of the browser a comment or the value of a given variable. One also can command to print those messages to special windows that are called Text Areas. In that way, debugging can be made into a public activity in which each person can see with his or her own eyes that the program runs as intended.

The *console* is a window to which the browser displays a lot of information, including our own logs or reports. Open the console before launching the program. To open it, go to the menu of the browser, select the item *Tools* or *Other Tools* or *Additional Tools* or *Tools for Developers* and next look for the console. In Firefox, the menu consists of three horizontal minibars in the upper-right corner.

13 Exercise. *Download Visual Studio Code for your system, install it, add a 5-star extension for* `html`*, go to menu* `file`*, choose* `new text file`*, and in the emerging window paste the program* `MyFirstCSS_JSprogram` *from this PDF. Make some minor syntax errors to see how Studio Code reacts, and undo them. Open and run the program over a browser. In the editor, search in the program the function that sums, and change the + (sum) by a * (multiplication). Save the new program. Open and run the new program over a browser. You are seen an extremely dangerous error because the browser runs the program and everything seems to be OK, but it does not do what it must, to add two numbers.*

Chapter 2

AN APP FOR A REAL NEED

The Author has dreamed of learning German, but he is not that good for that. In particular, German strongly relies on declensions, something that does not exist in English. This is a source of tremendous difficulty with various pillars. One of them is that the articles depend on the genre. It seems that the genre of nouns and articles are capricious so, they need to be memorized by heart or your declensions will be wrong and the sense of what you say may slide away. So, the Author decided to make an app to help himself and to share its service with the world, i.e., the app must be executable by any User from her or his own desktop or cell phone.

While this app maybe important for some few persons, it illustrates the solution to an important family of problems: to make a self-containing app that must interact with people in a rather complex way.

Since this chapter is intended to be introductory, let us discuss now the overall task that we face to.

2.1 THE GUI (GRAPHIC USER INTERFACE)

Let us think of the control room of an aircraft or ship. It allows us to see the relevant information to take decisions and has the

means to make commands. Our purpose is to learn how to do a simulation of this situation in computing. The result is known as **GUI** (Graphic user Interface).

We will divide the whole design of a specific interface in many steps, adding more and more function until the complete function is fulfilled. Our very first task is to show how to use buttons to publish information.

2.2 HTML BUTTONS

HTML has a basic button, which has no function apart from existing just as it was created. But some tens of variants exist that build different functions upon the basic motif.

14 Example. *We show here two types of buttons. The simplest one shows a static, invariant message, and an* `onclick-button`*, which is a variant that displays an alert, a message with relevant information, if one clicks it. This is useful when one has a long message to display that one reads once and then gets hidden.*

```
<!doctype html>
<html lang="en">

<head>
 <meta charset="utf-8" />
 <title>HtmlButtons.html</title>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <style> </style>
</head>

<body>
 <h1>The button Element</h1>
 <button type="button">I am a simple button. Look at Me!</button>
 <br>
 <button type="button" onclick="alert('Hello world!')">I am an onClick button. Click Me!</button>
</body>

</html>
```

15 Exercise. *Run the program* `HtmlButtons.html`

2.3 BEAUTIFUL BUTTONS

The web has been developed by people who are not satisfied with solving tasks but need moreover to do it with style, elegance, and distinction, converting every product in a beautiful artwork. This demands a terrible effort from the Developer that must master many fine details of various complex machineries. We will scratch that magnificent world by showing some simple but significative examples.

Our first task is to illustrate an example of using CSS to beautify a button in an as simple way as possible.

16 Example. *Program* `HtmlButtons2.html` *shows a button that is embellished with the help of CSS. This program exhibits a button with name PURPOSE that when clicked presents an alert, a separate, closable mini-window with a message about the purpose of this web-app.*

17 Exercise. *Run the program* `HtmlButtons2.html`

18 Exercise. *Add a second button to the previous program to recognize the credits of all the Developer community that with its blogs allowed us to learn and progress.* Answer

2.4 GROUP OF BUTTONS

When various buttons share the same properties, CCS allows them to be grouped to ease global management but conserving its individuality, the capability to be unique in some respect.

19 Ejemplo. *The Program* `MasculineFeminine.html` *presents two clickable options. They denote the genres* `feminine` *and* `masculine`*, that conform a CSS-group, and on click they*

present a message. These buttons allow change of color on hovering, when the User posits the mouse tip over a button. CSS allows changing any property of a button on hovering.

20 Ejercicio. *Run program* `MasculineFeminine.html`

21 Ejercicio. *Add* `Neutral` *as a third option to the previous program.* Answer

2.5 MY FIRST WEB-APP

Our buttons serve to enable interactive programming, one when the User can prompt the program with the equivalent of a question or petition and this responds with an appropriate action.

22 Example. *The Author developed an app to help himself to learn articles in German. To that aim, he used the material so gently offered by* Anna Zwolinska-Simon *in her web site* https://deutsch-mit-anna.de/lektion/nomen-liste/. *The app is in program* `ArticlesByAnna.htm`.

23 Ejercicio. *Run the program* `ArticlesByAnna.htm`, *test and evaluate it.* Answer

The problem with the anterior program is that the number of presented nouns is 30, but the German language may have 80000. To add more nouns, we used the list presented by Nery, which also has plurals.

http://frequencylists.blogspot.com/2015/12/the-2000-m

We took more than 500 nouns and plurals from the list by Nery, and added a phrase which was composed by Gemini, one AI of Google. Using a web translator (TWP = Translate Web Pages), an extension to Mozilla Firefox, we also provided a translation of the phrase into English.

Trying to learn 500 articles at once maybe an impossible task. But working with 10 articles at a time is really workable. Thus, our plan is to present the material in chunks of ten nouns. To that aim, we plan to use an HTML-slider, which is a rule with a sliding thumb, whose position determines a value that can be bound to a procedure, method or function in the JS code.

24 Example. *The code for a slider is presented in program* `HTML_Slider.html`. *Run it and give a glance at the code.*

25 Example. *We developed a program that takes chunks of 10 articles,* `GermanArticlesPlurasSlider.html`.

26 Exercise. *Run the program* `GermanArticlesPluralsSlider.html` *and evaluate it.* *Answer*

27 Challenge. *A problem with the anterior program is that chunks of tens may sound a little boring for slightly advanced learners. Add the option of choosing chunks with 20 words or maybe with many options so that each one could regulate the complexity of the exercises.*

We have chosen to present articles in groups of tens. But we lost the capability to test our memory against many nouns taken at random. Our solution is to implement two modes of operation. The first is for learning, in chunks of ten, and the second is for examination, in which all words until the present are taken in the exercise. The solution offered by HTML to implement this idea of two operation modes is a set of radio buttons that are mutually exclusive, e.i, only one of them can be selected at a given time.

28 Exercise. *Look at the web to see an HTML program that uses radio-buttons to make a choice together with a reaction to clicking one of them. Hint: prompt your search machine with* html radio button onclick. *Answer*

29 Exercise. *Following the examples of Internet which were studied for the previous exercise, devise an HTML program with two mutually exclusive options:* learning else exam. *Upon clicking anyone button, an action must be given.* Answer

Our plan is that in Learning mode, nouns are given by chunks of tens so, the exercise consists of the last 10 words just before the thumb of the slider. But in Exam mode, the exercise takes all words before the thumb, starting with the first noun and ending in the present value of the thumb.

30 Example. *We implemented our ideas in the program* `GermanArticlesPlurasTutor.html`. *It includes a mode of operation that can take on two values: learning, by chunks of tens, and examen, with many words presented randomly.*

31 Exercise. *Run the program* `GermanArticlesPlurasTutor.html` *and evaluate it.* Answer

32 Challenge. *A severe problem with the anterior program is that words listed at the beginning are more reinforced than those at the end. Create a remedy for this, else show that no remedy exists.*

33 Challenge. *Design and implement a scoring system with automatic promotion. A new try is conceded to practice the defective cases.*

34 Procedure to build the list of nouns. *The database with the nouns and plurals used by the Author is contained inside the code. The Author suffered a lot to make this database, possibly because of his use of free web services and software. To challenge everybody to do something better, let us count our story:*

1. The list provided by Nery was pasted into a worksheet. Because the list is compound, various columns were necessary. Each line was numbered to ease localization if, according to the Author, a correction is needed.

2. 100 substantives were chosen and then pasted into a new page or window of a word processor. It was saved with an appropriate name, say `Nouns100To200.txt`.

3. This file was processed by the program `SeparateListForGemini.html`, which added separation marks, quotes and commas.

4. The new text was fed to Gemini, an AI of Google, that pleasantly helped us with articles and phrases.

5. The result was completed with the data of Nery and with translations made by a web translator (TWP = Translate Web Pages), an extension of Firefox. To that aim, the material to be translated was pasted into an email draft so, that it could be picked up by the web translation services that we already installed in Firefox.

6. Translations were added into a column of the used worksheet.

7. The output was separated by columns. To add separating marks, commas and quotes, the material was copied to the clipboard and pasted into a word processor in a new document. This was saved with an appropriate name, say, `NounsAll100To200.txt`.

8. The saved file was opened and processed by program

   `SeparateListForGemini.html`.

   That program added quotes and commas and printed results to a TextArea. The result is accepted by both, Gemini and the array structure of JS in Visual Studio Code.

9. The result was picked and pasted at the inputArray of program `GermanArticlesPlurasWebTutor.html`.

This procedure was repeated various times because Gemini does not accept too long tasks.

Chapter 3

MY OWN WEBSITE

We got an app in the previous chapter. It is a program that is kept in the memory of a local desktop computer and that runs in an installed web-browser. How can I make my app visible and runnable for all Users in the world?

3.1 Servers

Computers can communicate among themselves and if you give permission, a foreign computer can look into your directories, read specific files and write messages to disk. So, the web consists of a set of computers that allow one another to freely look into a directory whose name must be PUBLIC. A computer with such capabilities is called a **Server** and it usually can control other computers and devices.

Any computer can be made into a personal server. To make our app available to the whole world, one just saves it to the public directory. But to be perceived as a website, one must give it a name and register it in a worldwide control office of websites. The problem with this solution is that one must permanently keep the computer on. This is costly in energy, operation and maintenance. Actually, it is extremely difficult to have a server running without interruptions during a long time. A functional

solution is to pay for *a piece of memory + operation + maintenance* in a computer of a company that specializes in offering hosting. This is the most common solution for lay people. So, one looks around where are hosting services, one compares them in terms of diverse measures and chooses the best.

In the public directory, there are many apps. That is why a website must have a head that is defined by a program in HTML that is called `index.html` or `index.php` because it may contain instructions in PHP, a language that allows the analysis of answers of clients. To see how an `index.php` file looks, one calls a website, and opens the code by right-clicking or going to tools for Developers. It might looks very similar to our app and shares the same skeleton. According to the interest of the Developer, the index may contain links to diverse images, sounds, and programs.

Websites compete for visits. An important factor, maybe the decisive one, is the velocity of reaction of the website. So, it is recommended to avoid beauty that demands too much time to be uploaded.

This is basically the whole dealt.

3.2 CONCLUSION

With the advent of development of programs by AI, it may seem superfluous to learn to program the web. In response, our politics is simple: everyone has the right to become an expert, a person who knows the how's and the why's. And, as we have seen, to know the fundaments of web programming is not that difficult, and we count moreover with too much help from excellent and free tools developed by the Industry and by blogs written by the community.

3.3 ANSWERS TO EXERCISES

5, page 8. The skeleton can be commented as follows:

```
<!doctype html>
<html lang="en">

<!-- This is a HTML comment. It will not appear in the web page. -->
<!-- Declaration of HTML as the language of this page -->
<html>

<head>
    <meta charset="utf-8" /> <!-- Beginning of heading -->
    <title>CommentedHTMLtemplate.html</title>
    <!-- Insert here the parameters of the used encoding and services, say, the visibility for mobile and for what operating systems.  -->
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <style>
        /* Put here the CSS style directrix of your HTML document, say, font size.  */
        /*This is a comment in css */
    </style>

    <!-- All HTML instructions are enclosed by a beginning mark and an ending mark. -->
    <!-- Marks are reserved words delimited by  < >  for the beginning, and </ > for the ending. -->
    <!-- Example:     <title>My Title</title>          -->

</head><!-- End of HTML heading -->

<!-- Beginning of  the body with the programming in HTML and
scripts in other languages, say JS (JavaScript)-->

<body>


    <!-- Put here your HTML instructions -->

    <!-- Inside an HTML document, you can insert code in various programming languages. -->
    <!-- So, a declaration of the language is necessary  but for JS it can be dispensed because -->
    <!-- JS is by 2025 the official language of the web. -->

    <!-- Beginning of JS code -->
    <script type="text/javascript">
        //This is a JS comment. It will be ignored by the JS interpreter,
        //that is a sort of compiler that works line by line.
        //Put here your JS code
    </script>
    <!-- End of JS code -->

    <script>
        //You can add more JS code here, say, another section of a program.
    </script>

    <!-- End of HTML programming  -->
</body>

<!-- The next line indicates the end of HTML document -->

</html>
```

12, page 15. The relation of our theory with the Evolutionary Theory of the origin of species is as follows:

1. The DNA is software because it contains instructions to synthesize amino acid chains and these instructions are encoded according to the genetic code.

2. Let us consider the extremely great variability of dogs, cats, and horses. This means that the DNA language is highly

expressive with too many options to make changes. But in spite of all variabilities, the offspring of a dog is a dog, and that of cats is a cat, and that of horses is also a horse. This implies that most mutations, that are at random, produce abortion, their fruit is not viable. In short: the DNA language is chaotic.

3. The Evolutionary Theory says that all species are related by descent, by accumulation of divergent characters that were filtered by a process of selection: those that are best suited for their environments leave more offspring and with the passing of generations, perfection and complexity appear. So, the chimp, the gorilla and the human have a common ancestor that eventually could be a chimp or a gorilla or a bonobo. Let us suppose that this was indeed so. Since the DNA language is highly chaotic, we have a mandatory prediction: along with the arising of each species, an innumerable quantity of bugs, of imperfect traits should have been left by the Evolutionary Process. They must be observable as defective inheritable marks that should be profusely represented as in the fossil record as in the body of every baby and hence in everybody. ¿Is that true? No. This is not true. Hence, the Evolutionary Theory is obviously false.

4. To have authority over this theme and to be a source of fresh ideas, you need to suffer hundreds (or thousands?) of hours programming and correcting bugs. So, you will simply ask: *Do we appeared by Evolution? If so, where are the bugs?* With these questions, everyone will catch that you are one more person that has understood as a personal experience that the Evolutionary Theory is obviously false.

5. More details over the view of the Author in
   https://www.ejristos.com/GPfalsifiesEvo/ScientificMethod

18, page 21. Program `HtmlButtons3.html` shows two onclick-buttons, one with the purpose and the other for recognizing the credits of the community.

21, page 22. Program `MasculineFeminineNeutral.html` offers three options for the type of article: masculine, feminine and neutral.

23, page 22. The program is extremely useful to ease the learning of articles in German.

26, page 23. The idea of taken groups of 10 is good.

28, page 23. We found the program to control radio-buttons by
https://www.w3schools.com/js/tryit.asp?filename=tryjs_form
This program can be run directly in the server of that site. The code is the following (the version in our `.zip` file has some modifications to enable running the program everywhere and without complaints):

```
 <!doctype html>
<html lang="en">

 <head>
  <meta charset="utf-8" />
  <title>HtmlRadioButtons</title>
   <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="author" content="https://www.w3schools.com/js/tryit.asp?filename=tryjs\_form\_radio">



    <script>
        function check(browser) {
            document.getElementById("answer").value=browser;
        }
    </script>

</head>
```

```
<body>

    <p>What's your favorite browser?</p>

    <form>
        <input type="radio" name="browser" onclick="check(this.value)" value="Internet Explorer">Internet Explorer<br>
        <input type="radio" name="browser" onclick="check(this.value)" value="Firefox">Firefox<br>
        <input type="radio" name="browser" onclick="check(this.value)" value="Opera">Opera<br>
        <input type="radio" name="browser" onclick="check(this.value)" value="Google Chrome">Google Chrome<br>
        <input type="radio" name="browser" onclick="check(this.value)" value="Safari">Safari<br>
        <br>
        Your favorite browser is: <input type="text" id="answer" size="20">
    </form>

</body>
</html>
```

29, page 24. In program `ChooseOperationMode.html` a set of two mutually exclusive radio buttons is presented that offers two options *learning else exam*. Upon clicking, they give a value, 10 else 100, to a variable.

31, page 24. The program looks good and seems to have a positive effect on the learner.