



blockchain skills for Europe

## D5.2.1: Curriculum Structure

06/2022



Co-funded by the  
Erasmus+ Programme  
of the European Union



## PROJECT DETAILS

Project acronym: CHAISE  
Project name: A Blueprint for Sectoral Cooperation on Blockchain Skill Development  
Project code: 621646-EPP-1-2020-1-FR-EPPKA2-SSA-B

## Document Information

Document ID name: CHAISE\_WP5\_D5.2.1\_CHAISECurriculumStructure\_2023-05-30  
Document title: D5.2.1\_CHAISECurriculumStructure  
Type: Report  
Date of Delivery: 30/05/2023  
WP Leader: UT  
Task Leader: UPC  
Implementation Partner: UPC  
Dissemination level: Public

## DOCUMENT HISTORY

Versions	Date	Changes	Type of change	Delivered by
1.0.0	25/05/2022	Initial document	-	UPC
1.0.1	06/06/2022	Lecture Content	-	UPC
1.0.2	10/06/2022	EQF/QA Scheme	-	ACQUIN
1.0.3	20/06/2022	Feedback	-	UPC
1.0.4	04/08/2022	Last feedback	-	UPC
1.0.5	20/11/2022	Annex	-	UPC

## DISCLAIMER

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

This document is proprietary of the CHAISE Consortium. Project material developed in the context of Project Management & Implementation activities is not allowed to be copied or distributed in any form or by any means, without the prior written agreement of the CHAISE consortium.

CHAISE Consortium			
Partner Number	Participant organisation name	Short name	Country
1	Université Claude Bernard Lyon 1	UCBL	FR
2	International Association of Trusted Blockchain Applications	INATBA	BE
3	Fujitsu Technology Solutions NV	FUJITSU	BE
4	Ministry of Education and Religious Affairs	YPEPTH	GR
5	ECQA GmbH	ECQA	AT
6	DIGITALEUROPE AISBL	DIGITALEUROPE	BE
7	IOTA Stiftung	IOTA	DE
8	Universitat Politècnica de Catalunya	UPC	ES
9	Duale Hochschule Baden-Württemberg	DHBW	DE
10	Associazione CIMEA	CIMEA	IT
11	INTRASOFT International S.A.	INTRASOFT	LU
12	Institute of the Republic of Slovenia for Vocational Education and Training	CPI	SI
13	European DIGITAL SME Alliance	DIGITAL SME	BE
14	University of Tartu	UT	EE
15	Univerza V Ljubljani	UL	SI
16	BerChain e.V.	BERCHAIN	DE
17	Italia4Blockchain	ITALIA4BLOCKCHAIN	IT
18	Autoritatea Națională pentru Calificări	ANC	RO
19	Akkreditierungs ,Certifizierungs- und Qualitätssicherungs- Institut e.V.	ACQUIN	DE
20	EXELIA	EXELIA	GR
21	INDUSTRIA Technology Ltd	INDUSTRIA	BG
22	Crypto4all	C4A	FR
23	Economic and Social Research Institute	ESRI	IE

## Abbreviations

AF	Application Form
CEDEFOP	European Centre for the Development of Vocational Training
CV	Curriculum
D	Deliverable
DG	Directorate General
EACEA	Education, Audiovisual and Culture Executive Agency
EQF	European Qualification Framework
EC	European Commission
ECVET	European Credit system for Vocational Education and Training
EU	European Union
D	Deliverable
ICT	Information and Communications Technology
KPI	Key Performance Indicator
M	Month
MOOC	Massive Open Online Course
OER	Open Educational Resources
PM	Project Management
PMT	Project Management Team
PT	Points
QA	Quality Assurance
SC	Steering Committee
SME	Small and Medium-sized Enterprise
SSA	Sector Skill Alliance
T	Task
TL	Task Leader
VET	Vocational Education and Training
WP	Work Package
WPL	Work Package Leader



## TABLE OF CONTENTS

<b>Abbreviations</b>	<b>4</b>
<b>1. Introduction</b>	<b>8</b>
<b>2. Requirements</b>	<b>9</b>
<b>3. CV Structure</b>	<b>9</b>
3.1 Learning Outcomes	10
3.1.1 Definition of the learning outcomes per module divided by Knowledge, Skills, Responsibility and Autonomy.	11
3.1.2 Definition of the curriculum modules and the different learning paths for Developers, Architects or Managers.	11
3.2 EQF Level	11
3.3 CV Modules	12
3.3.1 Modules list	13
3.3.2 Modules Structure	13
3.3.3 Modules Content	15
3.3.4 Lecture Example	17
3.4 CV Training and Teaching Methodology	17
3.4.1 Methodology	17
3.4.2. EQF5 Level Justification and challenges	18
3.5 CV Performance	20
3.6 CV Hours Structure	22
<b>4. QA Scheme</b>	<b>25</b>
4.1 Introduction	25
4.2 Involved education members	25
<b>References</b>	<b>38</b>
<b>ANNEX</b>	<b>40</b>

## LIST OF TABLES

TABLE 1 - MODULES CONTENT

TABLE 2 - HOURS STRUCTURE

TABLE 3 - INTERNATIONAL CERTIFICATION PROCEDURE

## LIST OF FIGURES

FIGURE 1 - REQUIREMENTS OF THE TASK T5.2

FIGURE 2 - CV LEARNING PATHS

FIGURE 3 - EQF5

FIGURE 4 - EQF6

FIGURE 5 - ADAPTION TO EQF6

FIGURE 6 - BLOOM TAXONOMY

## 1. Introduction

CHAISE is a Sector Skills Alliance financed by the Erasmus+ programme with the mission of developing a strategic approach on blockchain skills development for Europe as well as delivering future-proof training solutions, in order to tackle blockchain skill shortages and to respond to the current and future skill needs of the European Blockchain workforce.

The project is divided in various work packages and this document is focused on the WP5 Joint Curriculum Design and Delivery and specifically on the second task: T5.2 Design of CHAISE Curriculum Structure.

The main target of this task is to define the CHAISE CV Structure from the modules, hours and lectures perspective. Below, the objectives are described:

- OBJECTIVE 1: Definition of a sector-specific VET curriculum structure. 5 Semester duration CV structure with 1,200 teaching hours and 900 hours work-based learning (equivalent to 150 ECVET).
- OBJECTIVE 2: Create learning units with the following specifications: duration, weighting of outcomes, learning methods and assessment criteria according to ECVET principles.
- OBJECTIVE 3: Ensure that the curriculum corresponds to 5th or 6th EQF level.
- OBJECTIVE 4: Define QA scheme and procedures.

In the following sections, the document defines the main requirements, methodology and structure of the global CV followed by an explanation focused on the CV modules itself. Finally, the QA scheme of the CV is detailed.

## 2. Requirements

From the document “*CHAISE\_Detailed description of the Project*”, the main requirements to be taken into account for the design and creation of the curriculum structure have been extracted:

This T5.2 task includes the definition of a sector-specific VET curriculum structure:

- Create learning units with the following specifications: duration, weighting of outcomes, learning methods and assessment criteria according to ECVET principles.
- The curriculum will correspond to 5th or 6th EQF level, and will be divided into 3 main components:
  - Technical modules.
  - Non-technical modules including soft skills.
  - Work-based learning.
- 5 Semester duration: 1,200 teaching hours and 900 hours work-based learning.

FIGURE 1 - REQUIREMENTS OF THE TASK T5.2



---

### 3. CV Structure

Once the main requirements have been identified, we proceed to define the CV and its respective modules based on the following steps:

- Review and apply the results of the previous task (T5.1) in which the learning outcomes have been defined for the design of the CV structure.
- Decide the level of definition and teaching of the contents of the module based on the European Qualification Framework (EQF) and the requisites of the project.
- Definition of the modules and associated lectures that form the structure of the curriculum.
- Analyse and define the best methodology for teaching and evaluating the curriculum.
- Finally, design of the complete CHAISE curriculum structure with the corresponding hours per module.

### 3.1 Learning Outcomes

The task T5.1 was focused on defining what learners should know, understand and be able to do upon successful completion of the “CHAISE” Blockchain course. The information was defined in the document D5.1.1 - Blockchain learning outcomes report.

In order to work on the correct structure of the curriculum, the following results of the previous task have been taken into account:

#### 3.1.1 Definition of the learning outcomes per module divided by Knowledge, Skills, Responsibility and Autonomy.

The content of the different modules have been designed following the information obtained in that previous task. Also, the distribution of the lectures and hours per lecture have been done to benefit the achievement of the competencies described in the learning outcomes.

The modules have been divided by the CHAISE partners that collaborate in the task T5.3 to create the content of each of them, as listed in the section 3.4 CV Modules.

#### 3.1.2 Definition of the curriculum modules and the different learning paths for Developers, Architects or Managers.

<i>Transversal Skills (M, A, D)</i>			
<b>1. Regulation and Legal Aspects</b> <b>2. Governance of Blockchain Systems</b>			
<i>Technical Basics (D, A, M)</i>		<i>Business Basics (M, A, D)</i>	
<b>3. Fundamentals of Blockchain and Distributed Ledger Technologies</b>		<b>4. Blockchain Business Management and Planning</b>	
<i>Technical Blockchain Specialisation (D, A)</i>		<i>Business Blockchain Specialisation (M)</i>	
<b>5. Blockchain Security and Digital Identity</b> <b>6. Blockchain System Architecture &amp; Consensus Protocols</b>		<b>7. Blockchain Platforms</b> <b>8. Marketing and Customer Support</b>	
<i>BC Conception &amp; Use Case Development (A)</i>	<i>BC Engineering &amp; Development (D)</i>	<i>Strategic Business Management (A, M)</i>	<i>Operational Business Management (D, M)</i>
<b>9. Applied Cryptography</b>	<b>10. Smart Contracts and Digital Currency Programming</b>	<b>11. Developing use cases: From ideas to services</b>	<b>12. Game Theory in Blockchain</b>

FIGURE 2 - CV LEARNING PATHS

## 3.2 EQF Level

The project description states that the EQF level for this CHAISE CV should be between 5 and 6. The following captures are the definition of the levels according to the European Union:

### EQF - Level 5

+ Level 5 - learning outcomes		
Knowledge	Skills	Responsibility and autonomy
Comprehensive, specialised, factual and theoretical knowledge within a field of work or study and an awareness of the boundaries of that knowledge	A comprehensive range of cognitive and practical skills required to develop creative solutions to abstract problems	Exercise management and supervision in contexts of work or study activities where there is unpredictable change; review and develop performance of self and others

FIGURE 3 - EQF5

### EQF- Level 6

+ Level 6 - learning outcomes		
Knowledge	Skills	Responsibility and autonomy
Advanced knowledge of a field of work or study, involving a critical understanding of theories and principles	Advanced skills, demonstrating mastery and innovation, required to solve complex and unpredictable problems in a specialised field of work or study	Manage complex technical or professional activities or projects, taking responsibility for decision-making in unpredictable work or study contexts; take responsibility for managing professional development of individuals and groups

FIGURE 4 - EQF6

Based on this information, an analysis and comparison of the levels has been carried out with the different partners of this task to determine the level that best agrees with the objectives of the project and with the adaptation of the CV to the methodology of teaching of the different countries in which the CV will be applied.

The result of that analysis: **The CHAISE CV will be designed to accomplish the EQF5 and some methodological and concept advice will be given in every module to easily adapt the CV to a EQF6.**

In section 3.3.2 where the module structure is described, there is an explanation about the advice that will be given in every lecture to help the teacher to adapt the level of the CV from EQF5 to EQF6 if desired.

### 3.3 CV Modules

In this section, the modules obtained from the tasks T5.1 will be listed, minimally described, and structured according to the requirements of the project. Also, the section contains the assignment of each module to a partner of task T5.3 and a list of the readings already defined by each of them.

#### 3.3.1 Modules list

1. Introduction to Blockchain Technology.
2. Regulation, Legal aspects, and Governance of Blockchain Systems.
3. Fundamentals of Blockchain and Distributed Ledger Technology.
4. Blockchain Business Management and Planning.
5. Blockchain Security and Digital Identity.
6. Blockchain System Architecture and Consensus Protocols.
7. Blockchain Platforms.
8. Marketing and Customer Support.
9. Applied Cryptography.
10. Smart Contracts.
11. Developing Use Cases: From Ideas To Service.
12. Game Theory In Blockchains.

#### 3.3.2 Modules Structure

Every module must include the following:

- 4 LECTURES:
  - 80-120 slides (20-30 slides per lecture).
  - 20-30 pages of slide notes (5-6 pages of slide notes per lecture).

A lecture is a formal talk about a specific subject given to a group of students that will be summarised in 20-30 slides. Each lecture will have a topic related to the blockchain and will be self-contained, the set of 4 lectures will form a CV module. The modules will form part of an itinerary but can also be used independently according to the needs of the educational centre that uses it.

The slides notes per lecture is a document that has to include the description of all the topics and contents described in the slides together with some teaching tips and give some advice to adapt the lecture from EQF5 to EQF6. This advice will be given by some new topics proposals to include in the lectures, see the following example:



Include the explanation of the compilation process with its OPcodes: <https://www.ethervm.io/>.

FIGURE 5 -ADAPTION TO EQF6

Also, it can be useful to use the slides notes to transcript the video lecture, which will be introduced below, with the extra information needed to understand the topic from the point of view of the student.

- 4 VIDEOS:
  - 1 video per lecture.

A video will contain a taught lecture with the whole information needed by the student to understand and apply the lecture contents.

- 4 PRACTICAL EXERCISE:
  - 1 practical exercise per lecture.

A practical exercise is a guided practice about a specific lecture topic.

- 4 CASE STUDIES:
  - 1 case study per lecture.

A case study is a real scenario where a lecture and practical exercise can be applied and understood better.

- 20 QUESTIONS/ANSWERS:
  - 5 questions/answers per lecture.

A question/answer exercise is a self appraisal activity where the student will be able to check if the content explained in the lecture is correctly studied.



- **MULTIPLE CHOICE QUESTIONS:**
  - 10 multiple choice questions per lecture.

A multiple choice question is a performance module procedure to evaluate the students knowledge and understanding of the topics.

### 3.3.3 Modules Content

In this section, there are the lectures of each module described and the associated partner that will develop them:

<b>1. INTRODUCTION TO BLOCKCHAIN TECHNOLOGY</b>		
Lecture 1.1:	UT	Introduction to Blockchain Technology
Lecture 1.2:	UT	Blockchain History and Future

<b>2. REGULATION, LEGAL ASPECTS, AND GOVERNANCE OF BLOCKCHAIN SYSTEMS</b>		
Lecture 2.1:	UCBL	Blockchain basics to set the regulation and governance context and requirements
Lecture 2.2:	UCBL	Governance and regulation background
Lecture 2.3:	UCBL	Blockchain ecosystem
Lecture 2.4:	UCBL	Regulation strategy
Lecture 2.5:	UCBL	Blockchain governance
Lecture 2.6:	UCBL	Blockchain as a regulation mean for GDPR

<b>3. FUNDAMENTALS OF BLOCKCHAIN AND DISTRIBUTED LEDGER TECHNOLOGY</b>		
Lecture 3.1:	UL	Information and communications systems for decentralised solutions - Part 1
Lecture 3.2:	UL	Information and communications systems for decentralised solutions - Part 2
Lecture 3.3:	UT	Blockchain components and characteristics
Lecture 3.4:	UT	Distributed information systems and their information security management principles

#### 4. BLOCKCHAIN BUSINESS MANAGEMENT AND PLANNING

Lecture 4.1:	DHBW	The Blockchain Sector - An industry overview of Blockchain use cases and applications and scenarios (good practices)
Lecture 4.2:	DHBW	Applied Digital Ethics & Technology Assessment for Blockchain
Lecture 4.3:	DHBW	Fundamentals of business management methods (applied to Blockchain use cases) - Part 1
Lecture 4.4:	DHBW	Fundamentals of business management methods (applied to Blockchain use cases) - Part 2

#### 5. BLOCKCHAIN SECURITY AND DIGITAL IDENTITY

Lecture 5.1:	UL	Blockchain Honey pots
Lecture 5.2:	UL	Smart contract security
Lecture 5.3:	UT	Security risks analysis of blockchain-based applications
Lecture 5.4:	UT	Identity management and access control models of blockchain-based applications

#### 6. BLOCKCHAIN SYSTEM ARCHITECTURE AND CONSENSUS PROTOCOLS

Lecture 6.1:	IOTA	Basics in blockchain system architecture - Part 1
Lecture 6.2:	IOTA	Basics in blockchain system architecture - Part 2
Lecture 6.3:	IOTA	Different consensus protocols
Lecture 6.4:	IOTA	DLT examples

#### 7. BLOCKCHAIN PLATFORMS

Lecture 7.1	UL	Overview of platform characteristics
Lecture 7.2	UL	Performance and Scaling
Lecture 7.3	UL	Ethereum platform and ecosystem
Lecture 7.4	IOTA	Comparison of selected platforms: IOTA, Hyperledger, others.

#### 8. MARKETING AND CUSTOMER SUPPORT

Lecture 8.1	DHBW	Use of Blockchain in Marketing
Lecture 8.2	DHBW	Marketing for Blockchain (applied to Blockchain use cases)
Lecture 8.3	DHBW	Marketing and Customer Support - Part 1
Lecture 8.4	DHBW	Marketing and Customer Support - Part 2

## 9. APPLIED CRYPTOGRAPHY

Lecture 9.1	IOTA	Cryptographic paradigms
Lecture 9.2	UPC	Hash concept
Lecture 9.3	UPC	Hashes in blockchain
Lecture 9.4	IOTA	Zero knowledge and blockchain

## 10. SMART CONTRACTS

Lecture 10.1	UPC	Building simple smart contracts
Lecture 10.2	UCBL	Interacting with the blockchain through smart contracts
Lecture 10.3	UCBL	Building more advanced smart contracts
Lecture 10.4	UPC	Tokenizing assets with blockchain

## 11. DEVELOPING USE CASES: FROM IDEAS TO SERVICE

Lecture 11.1	UT	Business Model for Blockchain Use Case
Lecture 11.2	UT	Blockchain Use Case Redesign
Lecture 11.3	UT	Blockchain Use Case MVP
Lecture 11.4	UT	Blockchain Use Case Roadmap

## 12. GAME THEORY IN BLOCKCHAINS

Lecture 12.1	UPC	Basic remote purchase
Lecture 12.2	UPC	Extended remote purchase
Lecture 12.3	UPC	Game theory approach for fees
Lecture 12.4	UPC	Game theory behind Proof of Stake (PoS)

TABLE 1- MODULES CONTENT

### 3.3.4 Lecture Example

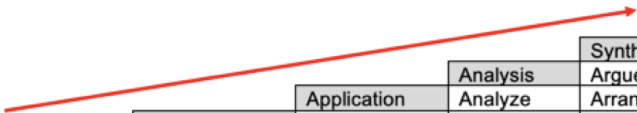
We have prepared a Lecture Example to facilitate the task T5.3.  
The lecture example is shown in the Annex of this document.

## 3.4 CV Training and Teaching Methodology

### 3.4.1 Methodology

The section contains a description of the usage of the Bloom Taxonomy, how the lectures have been developed and finally how the modules can be used together or individually.

The Bloom Taxonomy allows cognitive processes to be hierarchized at different levels and serves to facilitate evaluation tasks with verbs that can be associated with each level and can be used to specify learning objectives:



				Synthesis	Evaluation
		Application	Analysis	Argue	Appraise
	Comprehension	Apply	Analyze	Arrange	Assess
Knowledge	Compare	Choose	Compare	Collect	Attack
Label	Describe	Complete	Contrast	Combine	Choose
List	Discuss	Construct	Debate	Compile	Compare
Name	Explain	Demonstrate	Determine	Design	Conclude
Outline	Express	Develop	Distinguish	Devise	Contract
Present	Identify	Discover	Examine	Establish	Convince
Recall	Recognize	Employ	Experiment	Integrate	Decide
Recollect	Report	Illustrate	Identify	Make	Defend
Relate	Rewrite	Interpret	Inspect	Manage	Evaluate
State	Review	Operate	Order	Organize	Grade
Tell	Solve	Practice	Separate	Summarize	Interpret

FIGURE 6 - BLOOM TAXONOMY

This taxonomy applied in the CHAISE CV helped to set the European Qualification Level in the 5th level as fixed as a project requirement, so following this method the CV have been divided into several modules which they main objectives are to give the students a comprehensive, specialised and theoretical knowledge within the field of blockchain technologies to obtain a range of cognitive and practical skills and required to develop creative blockchain solutions.

The modules form the CHAISE CV and can be taught as a complete course, by paths of learning, as presented in section 3.1.2 or individually to support other courses.

### 3.4.2. EQF5 Level Justification and challenges

The coverage of National Qualification Frameworks (NQFs) with European Qualification Framework (EQF) and the learning outcomes-based approach has been expanded in recent years. According to CEDEFOP (2020), the post-pandemic landscape in Europe will be confronted with a number of challenges that will affect VET. They are related to:

- The adoption of emerging technologies and new forms of work organisation.
- Political challenges and sustainable development.
- Structural trends and economic downturn (job losses and decline in European economy).

The most important element of compatibility and comparability of qualifications in the EU is the *learning outcomes approach*. Learning outcomes define and describe qualifications in terms of what people are expected to know and are able to understand after completion of a VET programme (CEDEFOP, 2019). In the CHAISE curriculum that addresses EQF level 5, the defined learning outcomes will help explore the similarities and differences in the content of other programmes at national level.

In terms of EQF two challenges can be identified:

1. The development of national frameworks has been driven by the education sector, whereas businesses have been reluctant to embrace them.
2. The participating countries in the EQF have not agreed on a common procedure for exchanging information.
3. ECVET compatibility with your national qualification framework:

CEDEFOP has created a common guide on the necessary conditions for ECVET implementation. It can be found here: [https://www.cedefop.europa.eu/files/4113\\_en.pdf](https://www.cedefop.europa.eu/files/4113_en.pdf)

## PROPOSALS TO FACILITATE THE PROCESS OF ADAPTING THE CV

The process of adapting a curriculum in a national context is related to the extent to which national systems are influenced by international standards and market trends. The national VET systems cannot exist in isolation from educational and technological developments at international level. The challenge that arises is how to balance the international requirements



with national needs. The facilitation of alignment between national and European level is through the continuous adaptation of national VET standards, programmes and certificates to take into consideration external requirements and needs (CEDEFOP, 2020).

So far, the referencing between NQF and EQF has been achieved by 36 countries (CEDEFOP, 2019). In terms of CHAISE it includes all participating countries except for Spain. Apart from the *learning outcomes approach*, the comparison among qualifications across Europe can be done through *existing reference EU tools, such as ESCO*. Another way is the *involvement of stakeholders from NQF* in the skills development phase. Another suggestion towards alignment is to take *into consideration the new Skills agenda*, launched on 1 July 2020 by the European Commission. The Skills agenda is referring to skills intelligence and how European education providers can contribute to green and digital transition.

Lastly, another way to ensure comparability between EQF and NQF is the *use of level descriptors* that indicate the location of a particular qualification. Level descriptors can be seen as the most generic and abstract articulation of learning outcomes. This enables learners and education providers to position a qualification in relation to other qualifications (Bjørnåvold & Rusu, 2018). The descriptors should (Bjørnåvold & Rusu, 2018, p. 15) :

- “a) be sufficiently general to accommodate different parts of education and training systems;*
- (b) be sufficiently detailed and multifaceted to capture the institutional complexities, priorities and stakeholder interests of the national qualification system;*
- (c) capture domains and subdomains of learning (horizontal dimension);*
- (d) be able to reflect and capture how knowledge, skills and competences increase in breadth, depth and complexity when moving from lower to higher levels (vertical dimension);*
- (e) act as a reference point for international comparison.”*

A list of level descriptors in national qualifications frameworks (NQFs) in EU member states can be found here [https://www.cedefop.europa.eu/files/5566\\_en.pdf](https://www.cedefop.europa.eu/files/5566_en.pdf) (starting from page 58).

## Countries

In Germany, advanced vocational qualifications at tertiary level are nationally recognised vocational qualifications at EQF levels 5 to 7 (CEDEFOP, 2020).

More specifically:

- (a) professional specialist (Geprüfte Berufsspezialist) (EQF level 5, ISCED level 554);
- (b) bachelor professional: master craftsperson, specialist (EQF level 6, ISCED level 554, 665);
- (c) master professional: management and expert (EQF level 7)

The adaptation between EQF5 and EQF6 in Germany has taken place through a legislation in January 2020 that reinforces parity of esteem between academic studies and higher VET by legally assigning to them the same NQF levels (CEDEFOP, 2019). The title Meister is now legally equivalent to professional bachelor.

### 3.5 CV Performance

The performance indexes will be completed with continuous evaluation per module. Continuous evaluation is a process that must be planned, related to the objectives, competencies and established criteria. These aspects must be clearly described so that the student knows how he will be evaluated, having the control of their learning and their qualification, in the same way, in this process it is necessary to make use of various valid and reliable evaluation instruments that manage to measure in a more precise way what they are trying to know.

From the perspective of a continuous evaluation, the main points to be performed by the CV teachers will be:

- 5 Questions/answers per module.
- Multiple-choice questions.
- Case studies.

The teacher will be autonomous to define each the different weights of evaluation per the above points and also include to the evaluation various aspects as: autonomy of the student, proactivity in the learning, teamwork capacities and others.

However, below there is a proposal of the evaluation per module:

30% - 5 Questions/answers per module.

30% - Multiple-choice questions.

40% - Autonomy, proactivity, teamwork in the case studies.

Finally, the performance of the CV structure will be the average mark from the whole modules.

Apart from the students' evaluation, we think that it is important to include a method to analyse the performance of the CV from the students' experience point of view.

Our proposal is to include an anonymous survey at the end of every module to evaluate the students' level of satisfaction about the content of the lecture (slides, video, practice, use case and questions).

The main objective of carrying out this CV evaluation process is to improve its content and adapt it to the education model closest to each country of implementation, as explained in section 4 of this document.

### 3.6 CV Hours Structure

In the following table the different modules are listed with their respective hours of teaching and practice. The hours that appear in the table correspond to the number of hours that the student has to dedicate to properly understand and assume the concepts of each module.

	Module	ECVET	Teaching h	Practice h	Lecture No.
1	Introduction to blockchain technology	5	50	20	2
2	Regulation, legal aspects and governance of blockchain systems	15	150	60	6
3	Fundamentals of blockchain and distributed ledger technology	10	100	40	4
4	Blockchain business management and planning	10	100	40	4
5	Blockchain security and digital identity	10	100	40	4
6	Blockchain system architecture and consensus protocols	10	100	40	4
7	Blockchain platforms	10	100	40	4
8	Marketing and customer support	10	100	40	4
9	Applied cryptography	10	100	40	4
10	Smart contracts	10	100	40	4
11	Developing use cases: from ideas to service	10	100	40	4
12	Game theory in blockchains	10	100	40	4

	TOTAL	120	1200	480	48
--	-------	-----	------	-----	----

TABLE 2 - HOURS STRUCTURE

Below you can find a distribution of the hours per lecture, divided between Teaching and PRactice and the different activities that the student can do and the hours that must dedicate to properly follow the course and understand the lectures modules:

## TEACHING

- **Lecture Slides: 10 hours**  
Participate in the class, revise the slides as needed to understand the topic, identify and solve doubts with the teacher and take notes or summarise the topic.
- **Video lecture: 10 hours**  
  
View the video, take notes of the topic, revise the video various times and identify doubts that can later be solved with the teacher.
- **Questions: 3 hours**  
  
Read carefully the questions, prepare a draft of the answers, check the slides and the video to ensure that the solution draft is correct and write and present the final version.
- **Multiple choice test: 2 hours**  
  
Study the lecture and evaluate how much he/she has understood the topic of the lecture. Dedicate some time to check the answers, mainly the wrong ones.

## PRACTICE

- **Case study: 10h**

Execute and develop the case study proposed by the teacher as many times as needed to understand it. Share the case study conclusions with other students and take notes on the main points.

- Practice: 10h

Execute the practice proposed by the teacher, minimally twice, the first time with the class notes and another one alone.

The proposal of the specific hours per module can be summarised in three groups:

### **MODULE 1**

- Lectures – 8 h
- Practical exercises – 8 h
- Case studies – 8 h
- 10 questions/answers – 4 h
- 20 multiple choice questions – 4 h
- Self-study – 18 h
- Review of lecture material using slides and videos
- Self-preparation for final test

### **MODULE 2**

- Lectures – 24
- Practical exercises – 24 h
- Case studies – 24 h
- 60 questions/answers – 8 h
- 60 multiple choice questions – 8 h
- Self-study – 62 h
- Review of lecture material using slides and videos
- Self-preparation for final test

### **MODULE 3 - 12**

- Lectures – 16
- Practical exercises – 16 h
- Case studies – 16 h
- 20 questions/answers – 4 h



- 40 multiple choice questions – 4 h
- Self-study – 44 h
- Review of lecture material using slides and videos
- Self-preparation for final test

## 4. QA Scheme

### 4.1 Introduction

The QA scheme aims at securing the continuous improvement and assessment of the Joint Curriculum and assure its alignment and responsiveness to evolving labour market developments and skills needs. The purpose of this chapter is to facilitate curricula adaptation responding to changing and emerging labour market needs and educational policy shifts as a process of wider consultation with all relevant stakeholders.

Quality enhancement is the sum of many methods of institutional development, ranging from competitive hiring procedures, creating appropriate funding opportunities, to facilitating communication between disciplines and supporting innovative initiatives through institutional incentives.

The Bologna reforms may serve as a good case in point: while quality assurance is an important part of the Bologna reforms, the latter's relevance to quality goes far beyond the confines of quality assurance alone. Seen from their bright side, the Bologna reforms could improve quality in multiple ways: through the opportunities they offer to reflect and review curricula, to reform teaching methods (student-centred learning, continuous assessment, flexible learning paths) and even through strengthening horizontal communication and institutional transparency (EUA, 2008). Systematic quality assurance of training programmes is of high importance in order to assure relevance of qualifications within the labour market.

### 4.2 Involved education members

The involvement of the senior management and active participation of all members of the education institute is a precondition to ensure the sustainability of the training

programmes, as well as their relevance with current and future market trends. Quality assurance of the training programme is enhanced when the involvement of different stakeholders results into a development plan that defines areas and problems that need to be changed.

As internal and external actors we can consider:

- Professors and trainers.
- Senior management staff.
- Quality assurance managers.
- Administrative and IT staff.
- Communication officers.
- International affairs department.
- Policy officers.
- External network of education institutes.
- Students associations.
- Chambers, unions representing market needs in a specific field.
- Ministries of education.
- European umbrella associations in HEI & VET.

According to Cedefop (2009), transparency of the processes and results is not automatically assured for external clients and customers, which is why self-assessment of training programmes needs to be supplemented by an active publication and communications strategy. It has become standard in several European countries for educational providers to publish the results of their self-assessments on their organisation's website. However, an official obligation to make the results of the self-assessment available to customers only exists in relatively few cases.

### **4.3 EU tools for quality assurance**

As a backbone to ensuring long-term quality of the curriculum, four EU tools are taken into consideration. ECVET, the European Credit System in Vocational Education and Training, EQARF, the European Quality Assurance Reference Framework, the European Qualifications Framework, EQF and EQAVET, the

European Quality Assurance Reference Framework for Vocational Education and Training.

According to official glossary (CEDEFOP, 2011, p. 59) **ECVET** is the: *“Technical framework for transfer, recognition and, where appropriate, accumulation of individuals’ learning outcomes to achieve a qualification. ECVET tools and methodology comprise the description of qualifications in units of learning outcomes with associated points, a transfer and accumulation process and complementary documents such as learning agreements, transcripts of records and ECVET users’ guides”*.

**EQARF** was adopted by the European Parliament and the Council in June 2009 as a Recommendation on the establishment of a European Quality Assurance Reference Framework for Vocational Education and Training. More specifically: *“it describes the cycle of the quality of VET in four phases (planning, implementation, evaluation and re-examination/revision), and proposes, for each of them, a selection of criteria, descriptors and indicators to improve the management of quality both at VETsystem and VET-providers levels”*. (European Commission, 2012, p. 3).

EQARF can be applied at both the system and VET provider levels and can therefore be used to assess the effectiveness of VET. It gives a particular emphasis to the improvement and evaluation of the ‘outputs’ and ‘outcomes’ of VET in terms of increasing employability, improving the match between demand and supply, and promoting better access to lifelong training, in particular for disadvantaged people.

**EQF** is defined as a: *“Reference tool for describing and comparing qualification levels in qualifications systems developed at national, international or sectoral levels”*. (CEDEFOP, 2011, p. 65).

**EQAVET** is defined as the: *“Reference framework to help EU Member States and participating countries develop, improve, guide and assess the quality of their own vocational education and training systems. The methodology proposed by the framework is based on:*

- *a cycle consisting of four phases (planning, implementation, assessment and review) described for VET providers/systems.*
- *quality criteria and indicative descriptors for each phase of the cycle.*

- *common indicators for assessing targets, methods, procedures and training results – some indicators are to be based on statistical data, others are of a qualitative nature*". (CEDEFOP, 2011, p. 67-68).

All above frameworks are part of a shift toward assessing learning outcomes (i.e. what the learner knows and can do at the end of the learning process) rather than inputs.

#### 4.4 The QACEP Quality Framework

The QACEP Quality Framework is a reference guide for Higher Education Institutions designed to assist them in the management of the quality of their continuing education programmes, by fostering the development of a continuous improvement. The goal is to assist in the development of more efficient ways and means for delivering better outputs with the available inputs.

The target groups of the QACEP Quality Framework are continuing education programme managers, coordinators, teachers/academics, governing bodies and managers of higher education institutions.

The QACEP Quality Framework was developed as part of a co-funded project and can be also relevant in continuing education programmes.

It is built on the Plan-Do-Check-Act (PDCA) cycle idea: a problem solving process to facilitate continuous improvement in organizations. The PDCA concept emphasises that improvement must start with careful planning, lead to effective action, go through monitoring and improvement and re-visit the planning stage again resulting in an improved activity.

Therefore, the Framework is organised into four parts, corresponding to the following phases:

- Planning and design.
- Implementation and delivery.
- Programme monitoring.
- Programme improvement.

For each phase the Framework identifies key elements and features. The key elements presented in the planning and design phase must also be considered for all other phases. The QACEP Comparative Analysis Report can be found here: [www.qacep.eu](http://www.qacep.eu).

## 4.5 Curriculum quality assurance & periodic review

In a curriculum quality assurance framework, a review mechanism is an embedded element, which key objectives are:

- To evaluate program curriculum, effectiveness and sustainability.
- To provide an opportunity for planning for the future.

Periodic review and curriculum renewal, mainly seek to address the following questions:

- Is the demand (both learner and employment) sustainable?
- Is the level of satisfaction in meeting learner and workplace needs acceptable?
- Is the programme effectively responding to external needs and challenges?
- Are resources (learning, human and physical) necessary for the program available?
- Is the programme congruent with the strategic direction of the education and training provider?
- Are learners learning what they are intend to learn?

The quality assurance of training programmes is crucial in considering the specific objectives of these programmes, the specific target groups, the variety of stakeholders involved, and their relationship with the labour market and society. From the beginning of identifying the strategic objectives and management, the education institute should take into consideration the questions:

- To what extent is there a strategy on how to interact and communicate with stakeholders on the local, national and international level?
- Which stakeholders are structurally involved in the quality management?

- To what extent are the needs of stakeholders (labour market, professional bodies, etc.) assessed?
- To what extent are the procedure(s) for admission clearly communicated to the stakeholders?
- To what extent are the defined learning outcomes in line with the needs of target groups and stakeholders (including labour market)?
- To what extent are the content of the programme and the expected learning outcomes well described to the stakeholders?
- To what extent are the different stakeholders involved in programme monitoring?
- To what extent are different stakeholders consulted on the results of monitoring (teachers, learners, companies etc.)?

In any quality assurance system one of the key features is the improvement that is developed and implemented following the monitoring of the programme including the ascertaining of views of all stakeholders on the delivery and outcomes of the programme. Key to delivering an improvement and enhancement of programmes is accurate data and evidence on which to base any changes to be made. Thus, it is essential that the monitoring phase on the delivery is conducted in a timely and thorough fashion.

Self-reflection together with evaluation of the strengths and weaknesses of all aspects of the programme, conducted in an evidence-based manner, will ensure the alignment of the needs of all stakeholders (learners, labour market, teachers, institutions, etc.) and result in delivery of an enhanced programme.

Crucial aspects here include:

- To what extent are the quality improvement system and criteria transparent to all stakeholders?
- To what extent are quality improvement actions discussed with and communicated to internal and external stakeholders?
- Which stakeholders (incl. participants) are being questioned in the evaluation?

The ultimate aim of any quality assurance exercise must be the improvement and enhancement of a programme/activity and it is important that all stakeholders – students, staff and external stakeholders – are facilitated in their engagement with the process. Funding agencies, including government and industry, where applicable, also play an important role in the quality assurance and have an interest in the outcomes and developments.

## 4.6 Online learning environment: aspects of Human Computer Interaction

Information and Communication Technologies (ICTs) have played a key role in supporting the continuation of teaching and learning at university-level education (HEIs)/VET during the COVID-19 pandemic. Based on a recent study (OECD, 2020), the unprecedented pandemic has activated measures at global scale that disrupted the normal functioning of universities. The responsible departments at HEIs/VETs had to create effective synergies with the IT departments and teaching staff to put in place alternative methods for students.

The challenge was dependent on previous knowledge of the education institute in this field, as well as relevant support from government, enabling a smoother or harder transition to the fully digitalised learning environment. According to Scheuermann & Pedró (2009), there is an **absence of broader set of internationally comparable indicators**. These indicators could monitor progress in ICT uptake and unveil important information about use. Since frequency and purpose cannot be easily measured, comparable data and practices remain at national level, making it difficult to put in place tools for benchmarking policies at EU-level.

In many European universities and VET providers, digital learning environment served only to complement physical classrooms. Peer-to-peer interaction had to be replaced by virtual rooms. Many of ICT systems are not designed to cover learners with special needs or distraction disorders, a parameter that needs to be further researched and taken into consideration for the future. It is also impairing the efforts of more shy or less socialised students to interact with other peers or teachers, a behaviour that could be identifiable at physical level. Assignments and exams have also been affected.

In comparison to sit-down exams, ICT systems should be flexible enough to ensure more time for students to complete them and more time for teachers to grade them.

Despite the challenges, the adoption of ICT methods provides students with the necessary skills to integrate into society and professional life, where technology-related competencies are an integral part of 21st century education.

## 4.7 Quality criteria in online learning environment

Further to the use of EU frameworks, standards and EQAVET indicators, ACQUIN proposes complementary tools for evaluating an online learning environment. This proposal is based on current ICT discourse and can be conducted by the assessors/external reviewers when evaluating the platform.

According to **Jakob Nielsen's Heuristics** (<https://www.nngroup.com/articles/ten-usability-heuristics/>) assessors/external reviewers will be asked to provide their input based on the following 5 Heuristics:

### **Heuristic 1: Visibility of system status**

The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time.

### **Heuristic 2: Match between system and the real world**

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

### **Heuristic 3: User control and freedom**

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.



### **Heuristic 7: Flexibility and efficiency of use**

Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

### **Heuristic 8. Aesthetic and minimalist design**

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

The below scale is used in the heuristic evaluation:

#### **Severity scale**

0 = I don't agree that this is a usability problem at all

1 = Cosmetic problem only: need not be fixed unless extra time is available

2 = Minor usability problem: fixing this should be given low priority

3 = Major usability problem: important to fix, so should be given high priority

4 = Usability catastrophe: imperative to fix this as soon as possible

The heuristic evaluation can be accompanied by usability research conducted by the HEIs/VET providers in order to measure the feedback from learners and instructors that are the end-users of the platform. This way the platform will meet high standards of quality in terms of design, use and operability.

## **4.8 External assessment of training programme by QA agencies**

The goal of an international certification procedure of a training programme is to assess the programme's existing quality and recommend improvements. Accountability and enhancement are at the core of the procedure. Peer-review

experts evaluate and assess the programme. To guarantee impartiality, the experts scrutinize the programme against a set of criteria.

The international certification procedures in Europe should comply with the "Standards and Guidelines for Quality Assurance in the European Higher Education area" (ESG). The ESG standards ([https://www.enqa.eu/wp-content/uploads/2015/11/ESG\\_2015.pdf](https://www.enqa.eu/wp-content/uploads/2015/11/ESG_2015.pdf)) define both assessment criteria and criteria for the accreditation process. They were adopted at the European Higher Education Area Ministerial Conference in 2015.

The basis of the procedure is the self-assessment report of the HEIs/VET providers for the evaluation of the programme by peer experts. HEI/VET provider submits the self-assessment report indicating the ways in which the programme complies with the ESG standards and other EU tools (i.e. EQAVET indicators).

Within the ESG framework, the procedure may check the compliance of the programme with national legislation, as well as national and international scientific standards, such as ECVET and European Credit Transfer and Accumulation System (ECTS).

According to ESG, the following ten criteria of internal quality assurance are evaluated:

- ESG 1.1 Policy for quality assurance.
- ESG 1.2 Design and approval of programmes.
- ESG 1.3 Student-centred learning, teaching and assessment.
- ESG 1.4: Student admission, progression, recognition and accreditation.
- ESG 1.5: Teaching staff.
- ESG 1.6: Learning resources and student.
- ESG 1.7: Information management.
- ESG 1.8: Public information.
- ESG 1.9: On-going monitoring and periodic review of programmes.
- ESG 1.10: Cyclical external quality assurance.

An international certification procedure of a curriculum held by QA agencies operating in higher education and higher VET can have the below positive effects:

- Enhance the international readability of curricular structures and the underlying quality assurance systems than can in turn increase cooperation and competition, mobility and institutional good practice, with quality enhancement occurring as a natural consequence of wider and deeper comparisons.

- Increased mutual trust in each others 'quality assurance systems would result in increased trust in the quality of education provision in those systems, thereby resulting in cross-border movement.
- Enhanced quality in teaching based on learning outcomes and student-centered teaching, also in cases where learners choose a different learning path such as the training offered through an Erasmus+ project.

A typical international certification procedure in ACQUIN is described in the below table:

Phases	ACQUIN	Peer-review experts	Higher Education Institution (HEI)
Contract between HEI and ACQUIN			
Nomination of the peer-review experts			HEI sends preliminary information about the programmes (profile information)
	ACQUIN appoints experts		
			HEI accepts the peer-review experts
Self-assessment report			HEI prepares and submits self-assessment report
	ACQUIN checks validity and completeness of the self-assessment report		
Organising the site visit	ACQUIN programme manager as a contact person accompanies and supports the HEI in organising the site visit		HEI organises the site visit in coordination with the programme manager
	ACQUIN programme manager provides experts with the essential information and prepares them for their task		
Site visit and reporting	ACQUIN Programme manager coordinates and accompanies the on-site visit	Experts discuss with HEI representatives	HEI management, teaching staff, and students provide comprehensive insight in programme(s)
		Experts compile an assessment report	
Certification decision			HEI gives a statement on the report – if necessary
	Accreditation Commission of ACQUIN decides about the certification		
	ACQUIN publishes the assessment report including the formal certification decision		HEI is informed about the decision and receives certificates and documents

TABLE 3 - INTERNATIONAL CERTIFICATION PROCEDURE: SOURCE ACQUIN GUIDELINES

Since VET has not developed specific standards for evaluation, ESG flexible character can serve a good basis for the external review of the programme. These standards can be used along with EQAVET indicators. Implementing EQAVET is a challenge at both national and institutional level. The Quality Assurance methodology proposed within EQAVET and which needs to be implemented within VET provision at national level is based on three main aspects:

36

- a quality cycle consisting of four phases (planning, implementation, evaluation and review) described for VET providers/ systems.
- quality criteria and indicative descriptors which feed into each phase of the cycle to direct providers on how to implement the quality cycle.
- and common indicators (quantitative and qualitative) for assessing targets, methods, procedures and training results at both system and provider level.

Once changes are made, data will provide information on their impact and help VET providers to sustain the quality of their own efforts.

Here are some tips which might be useful when initiating the monitoring of quality assurance approach:

1. Build self-monitoring into the implementation of quality assurance management processes from the beginning.
2. Negotiate between stakeholders to decide 'what' to self-monitor and keep them informed of the state of the process and the results of the exercise.
3. Ensure that EQAVET indicators are clearly understood and commonly interpreted by all stakeholders.
4. Identify a data collection system/procedure for [inputs](#), [outputs](#) and [outcomes](#) as encapsulated in the EQAVET indicators.
5. Record information in sufficient detail to provide for improvement actions , future evaluations and to illustrate [accountability](#).
6. Check that results are directly linked to the stated objectives and to other factors which may have a key influence on the process.
7. The process will only be complete when [evidence](#) has been put to use, e.g. in planning for improvement and in reporting on quality.

## 4.9 Concluding remarks

Based on the above, external reviews in education institutes (either through accreditation procedures or institutional reviews) presuppose a sufficient degree of institutional autonomy in order recommendations and intended action plans to be able to be realized.

Among the success factors of constructive development in quality, time and willingness of academics, institutional leadership and quality assurance departments are worth mentioning. The below Annexes are integral part of the current guidelines and provide further input, checklists and templates.

Another success factor consists in the frequency of the quality assurance cycle. Too frequent reviews may result in evaluation fatigue and demotivation to engage in meaningful dialogue. The choice of external assessors/experts is also vital in order to cover different disciplinary areas to allow for enlarged horizons.

## References

EUA. (2008). Implementing and using quality assurance: Strategy and Practice. A selection of papers from the 2nd European Quality Assurance Forum. Available at: <https://eua.eu/component/publications/publications/113-egaf-paper/466-implementing-and-using-quality-assurance-strategy-and-practice.html>

European Commission Directorate-General for Education, Youth, Sport and Culture. (2012). EQARF, European quality assurance reference framework for vocational education and training. Publications Office. DOI: doi/10.2766/49917

CEDEFOP. (2020). Vocational Education and Training in Europe, 1995–2035: Scenarios for European Vocational Education and Training in the 21st Century.

CEDEFOP. (2015). Ensuring the quality of certification in vocational education and training. Luxembourg: Publications Office. Cedefop research paper; No 51. <http://dx.doi.org/10.2801/25991>.

CEDEFOP. (2013). Renewing VET provision Understanding feedback mechanisms between initial VET and the labour market. Luxembourg: Publications Office. 2013 – VI, 166 p [https://www.cedefop.europa.eu/files/5537\\_en.pdf](https://www.cedefop.europa.eu/files/5537_en.pdf).

CEDEFOP. (2011). Glossary: quality in education and training. Luxembourg: Publications Office. [http://www.cedefop.europa.eu/files/4106\\_en.pdf](http://www.cedefop.europa.eu/files/4106_en.pdf)

CEDEFOP. (2009). Accreditation and quality assurance in vocational education and training. Luxembourg: Publications Office of the European Union. [https://www.cedefop.europa.eu/files/3061\\_en.pdf](https://www.cedefop.europa.eu/files/3061_en.pdf)

European Commission Directorate-General for Education and Culture. (2005). Fundamentals of a “common quality assurance framework” (CQAF) for VET in Europe, European Commission, Directorate General for Education and Culture, Brussels. (Accessed 10 June 2022). Available at: [http://www.bmukk.gv.at/medienpool/18122/fundamentals\\_of\\_a\\_cqaf\\_for\\_v.pdf](http://www.bmukk.gv.at/medienpool/18122/fundamentals_of_a_cqaf_for_v.pdf).

Scheuermann, F., Pedró, F. (2009). Assessing the effects of ICT in education Indicators, criteria and benchmarks for international comparisons.

OECD. (2020). Supporting the continuation of teaching and learning during the COVID-19 Pandemic. Available at:

39

---

<https://www.oecd.org/education/Supporting-the-continuation-of-teaching-and-learning-during-the-COVID-19-pandemic.pdf>

Grm, S. P., Bjørnåvold, J., & Rusu, A. (2018). Analysis and Overview of NQF Level Descriptors in European Countries. Cedefop Research Paper. No 66. Cedefop-European Centre for the Development of Vocational Training.

Cedefop. (2020). Vocational Education and Training in Europe, 1995–2035: Scenarios for European Vocational Education and Training in the 21st Century.

European Centre for the Development of Vocational Training (Cedefop) European Training Foundation (ETF). (2020). The importance of being vocational: challenges and opportunities for VET in the next decade: Cedefop and ETF discussion paper.

European Centre for the Development of Vocational Training (Cedefop). (2019). Briefing note - NQF developments 2019. Available at: <https://www.cedefop.europa.eu/en/publications/9150#group-details> (Accessed June 2022).



## ANNEX

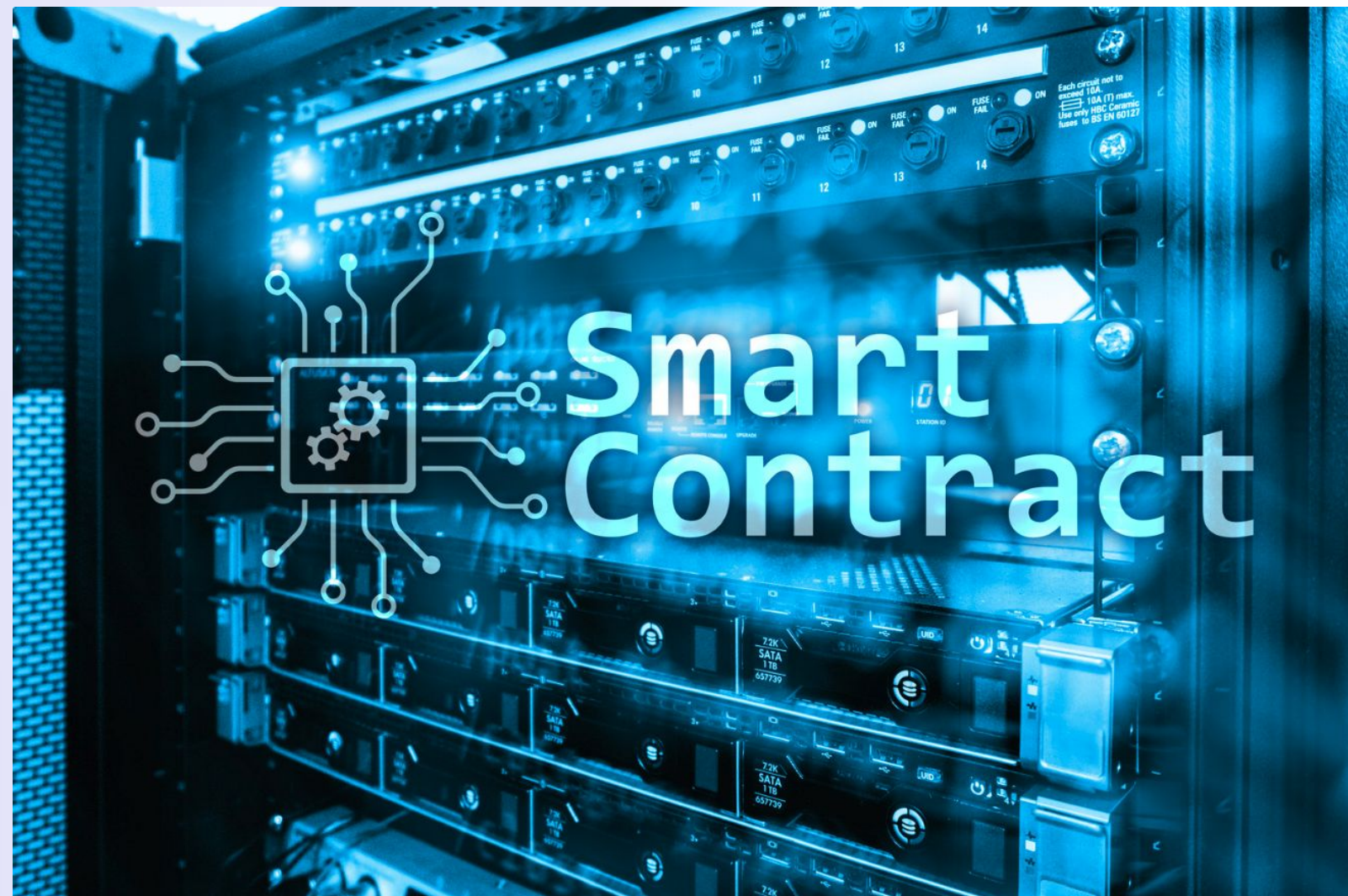
# Module 10: Smart Contracts

## Lecture 1: Simple Smart Contracts



Co-funded by the  
Erasmus+ Programme  
of the European Union

# Table of Contents



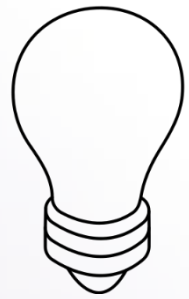
- ▶ Smart Contract Concept
- ▶ Writing Smart Contracts

## 10. SMART CONTRACTS

Employ programming language(s) to develop smart contracts and digital currency.

Knowledge	Skills	Responsibility and Autonomy
<b>Knows / Aware of:</b> <ul style="list-style-type: none"> <li>Frontend and Backend development.</li> <li>User experience (UX) design principles.</li> <li>Smart contract design and implementation.</li> </ul>	<b>Able to:</b> <ul style="list-style-type: none"> <li><b>LO10.1:</b> Apply good practices for developing smart contracts and describe the advantage of blockchain technology.</li> </ul>	<b>Capable to:</b> <ul style="list-style-type: none"> <li>Relate the frontend and backend components of the blockchain-based application.</li> <li>Integrate a creative environment to support observation, ideation, reflection, building and rebuilding of the blockchain-based application prototypes.</li> </ul>

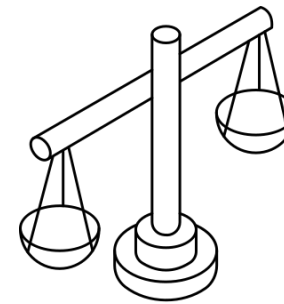
# Learning Objectives



## Smart Contract Concept

---

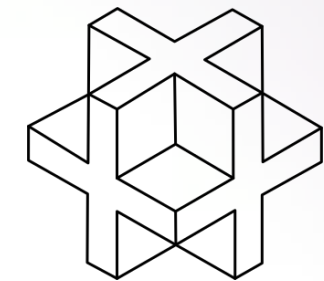
Blockchain Main Features  
Anatomy of a Smart Contract



## Simple Smart Contract

---

Frontend and Backend Smart  
Contract



## Writing Smart Contracts

---

Deploy a Smart Contract  
Functions and examples



# Smart Contract Concept



# Blockchain Main Features

- Blockchains are "super systems" to manage state:
  - Most replicated systems in the world (highest possible availability).
  - Tamper-proof (immutable).
  - Public blockchains are the most transparent systems in the world.
- As a result several things are unfeasible in blockchain:
  - Denial of Service (DoS) attacks.
  - Censorship.

# Use State for What?

- Blockchains are not "just" distributed data repositories, you can also define logic around state:
  - Which transactions are accepted or which are not.
  - How a transaction modifies the state.
- Use state for what?
  - State machine of a cryptocurrency.
  - Other applications: use blockchain as a platform for building applications, that is, program our state machine like a computer.
  - This was the vision of Ethereum back in 2013.



# A Blockchain as a “Global Computer”

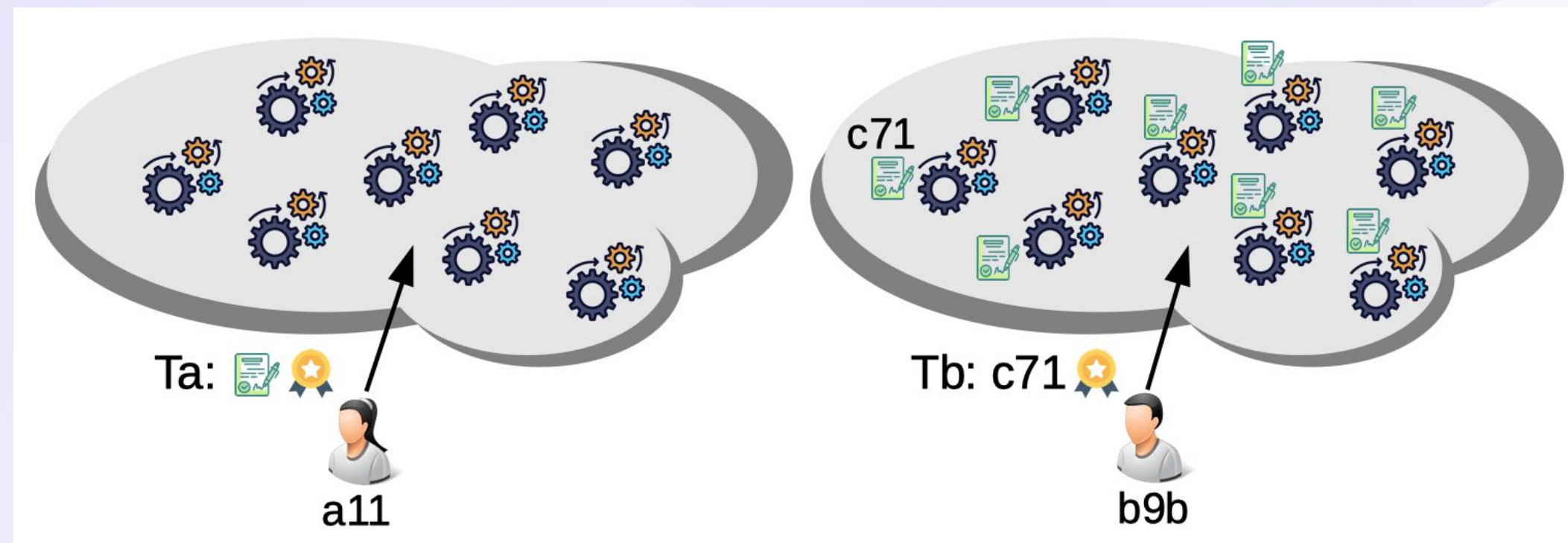
## Blockchain Global Computer

*A deterministic computer, very highly replicated that operates on inputs (transactions) under consensus.*

- A transaction can “install” (deploy) code in the global computer.
- This code is the famous smart contract (AKA chaincode).
- We can execute functions of deployed smart contracts with other transactions.

# Smart Contracts I

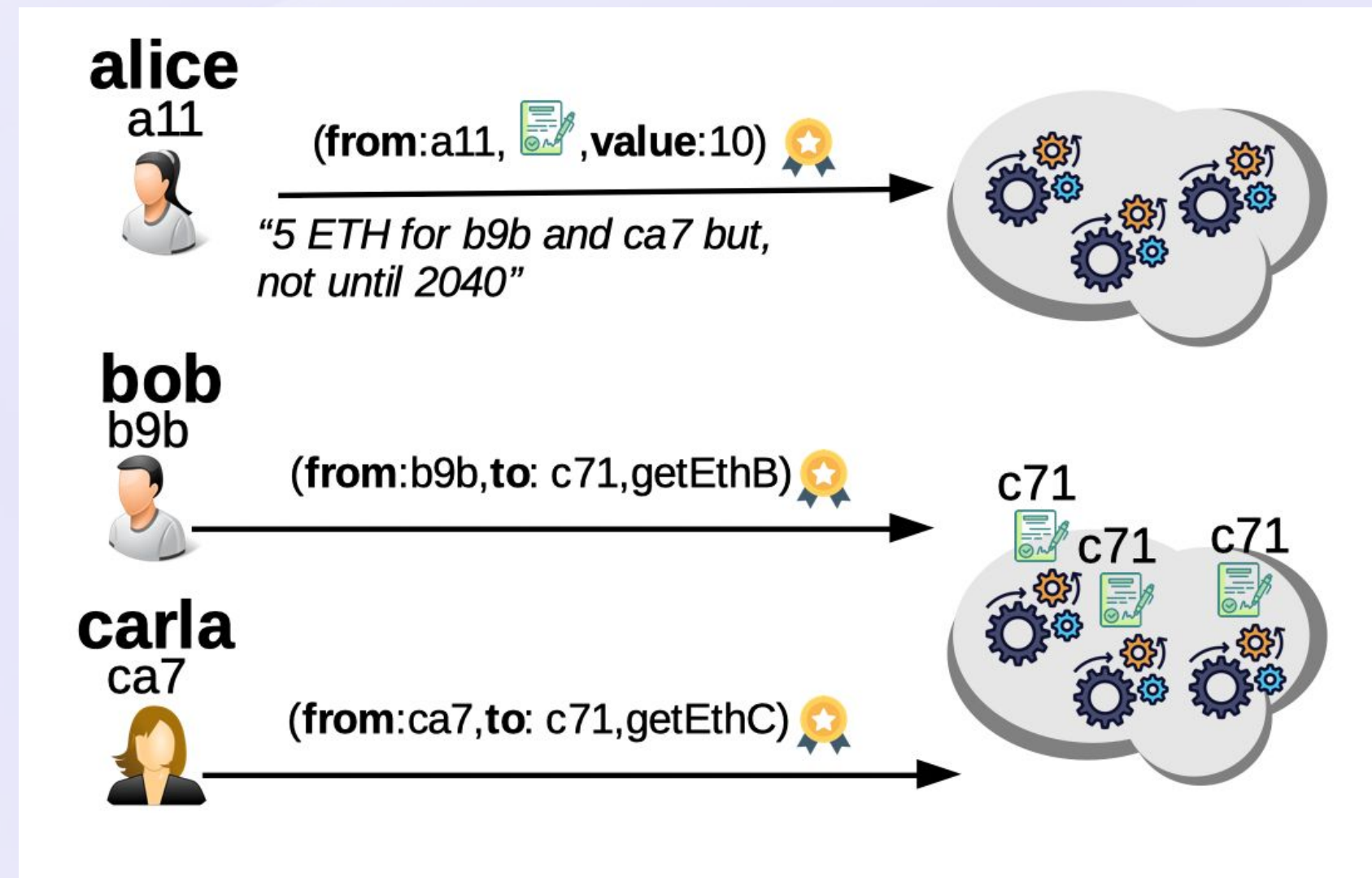
Users can do more sophisticated things with smart contracts.



Note. Contract might be a "wrong" name because by default smart contracts do not have any legal validity (they are just code).

# Smart Contracts II

Example:



Another example: define how our taxes should be invested.



# Anatomy of a Smart Contract

Code (rules) is immutable and provides functions to mutate the contract's state.

## Smart Contract

**address:** c71  
**balance:** 0.5

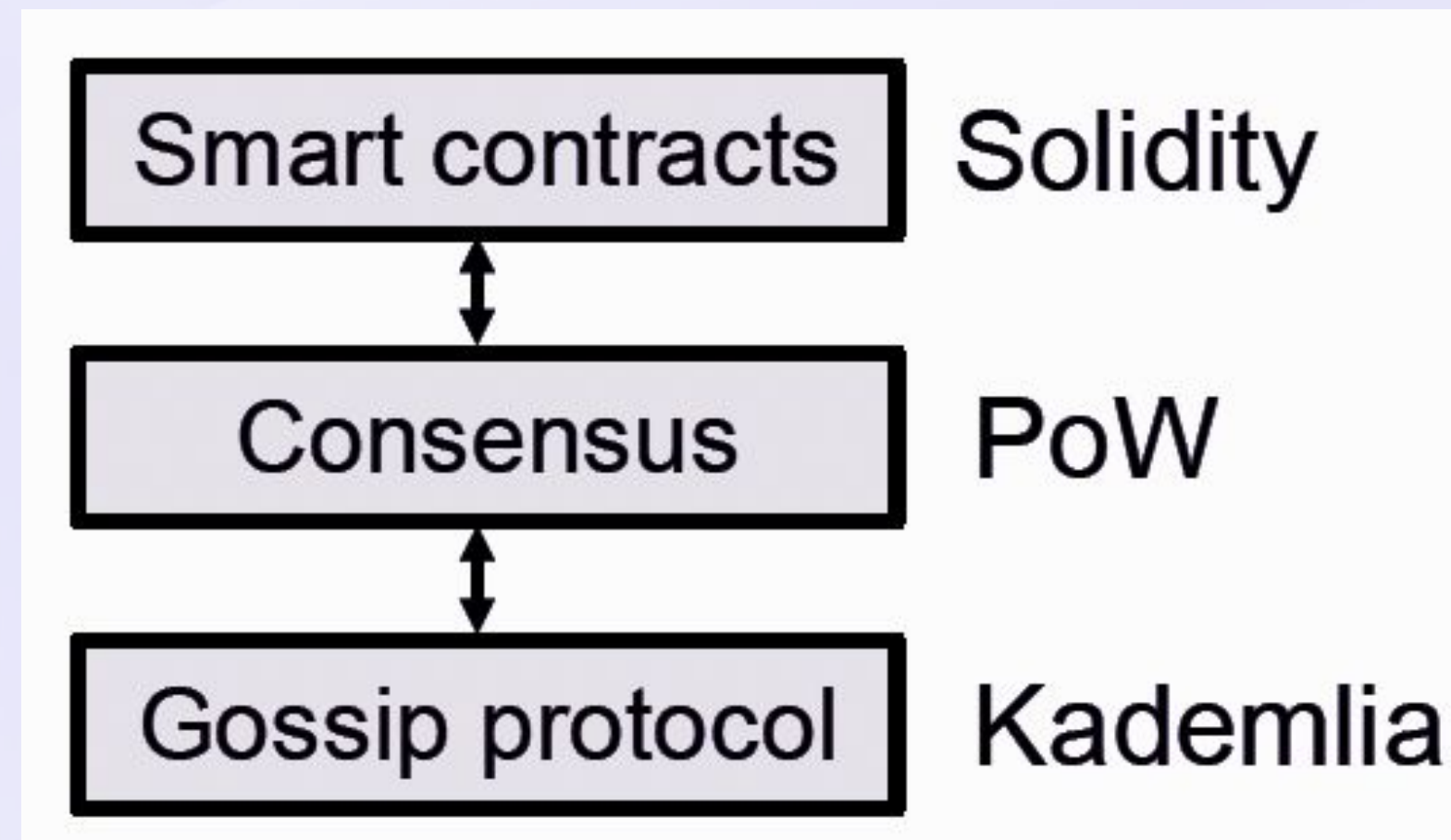
**storage:** 

**code:**  f1()  
f2()  
...

# Example of our Smart Contract Code

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract AliceWill {
5      // Contract storage
6      bool public bobOK = false;
7      bool public carlaOK = false;
8
9      // Contract functions
10     constructor() payable {}
11
12     function getEthB() external {
13         require(block.timestamp > 2208985200);
14         require(msg.sender == 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4);
15         if(carlaOK == true) {
16             payable(0x5B38Da6a701c568545dCfcB03FcB875f56beddC4).transfer(5 ether);
17             payable(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c).transfer(5 ether);
18         } else { bobOK = true; }
19     }
20
21     function getEthC() external {
22         require(block.timestamp > 2208985200);
23         require(msg.sender == 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c);
24         if(bobOK == true) {
25             payable(0x5B38Da6a701c568545dCfcB03FcB875f56beddC4).transfer(5 ether);
26             payable(0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c).transfer(5 ether);
27         } else { carlaOK = true; }
28     }
29 }
```

# Blockchain Layers

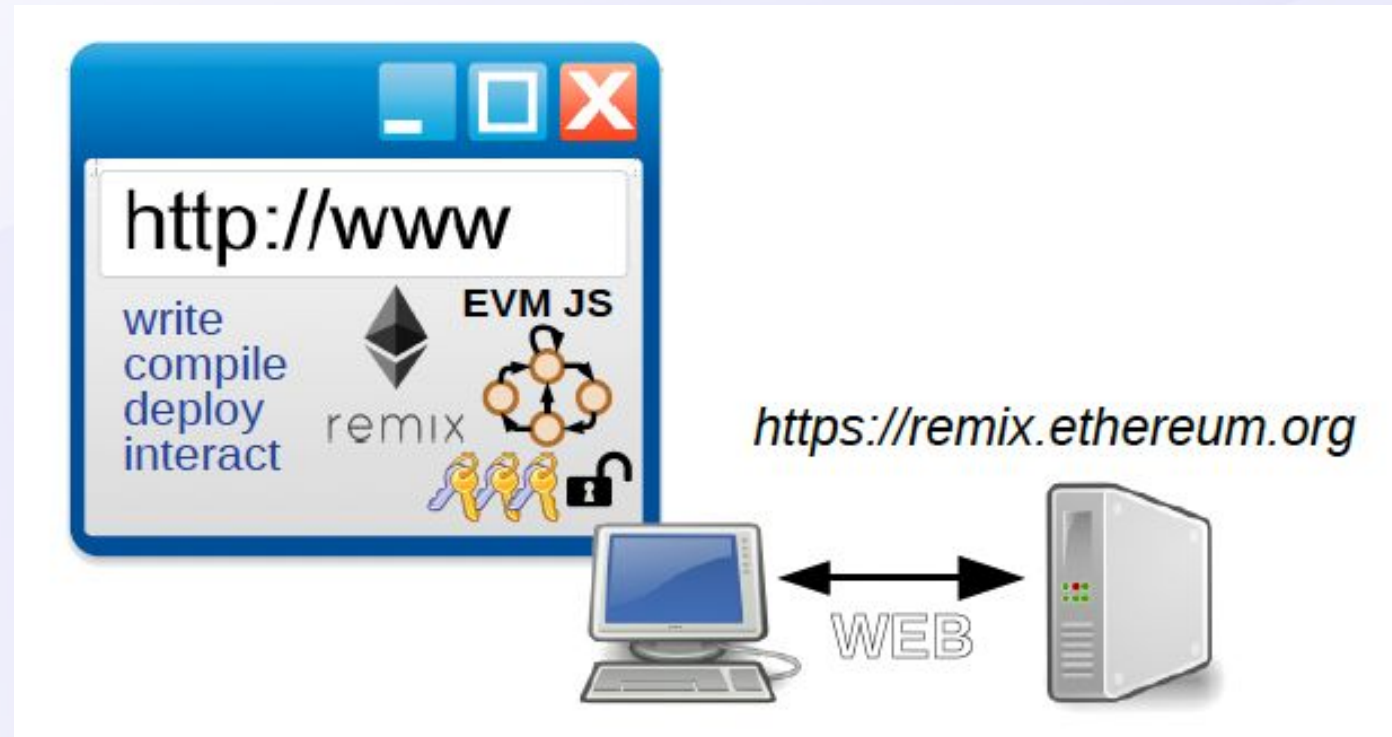




# Writing Smart Contracts



# Basic Ethereum Remix



- Remix can execute a local Ethereum virtual Machine (EVM) in the browser.
- This is the easiest way of having the experience of creating a smart contract.

1. Write, compile, deploy and debug your smart contracts.
2. Interact with smart contracts: transactions and queries.
3. You get dummy accounts with Ether to create transactions.
4. Estimate the costs of each transaction... and much more...

<https://remix.ethereum.org>



# Simple Smart Contract I

## Smart Contract: SimpleStorage

address: 9a7

storage:



storedValue  
7

code:



setStoredValue()  
getStoredValue()

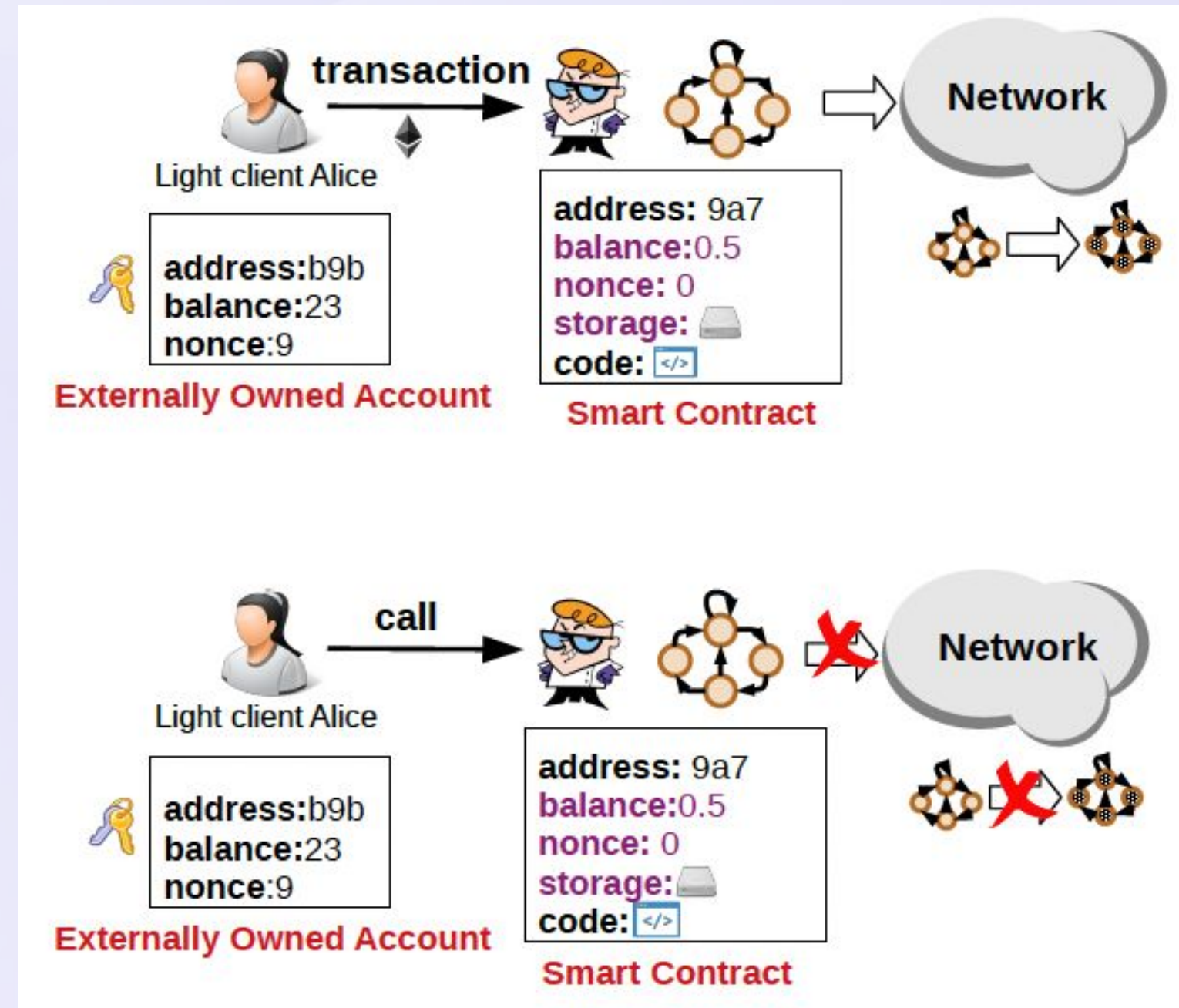
- The storage of the smart contract will store a positive integer.
- We have two functions to get and set the stored value.

# Simple Smart Contract II

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract SimpleStorage {
5      uint256 storedValue; // we are storing 256 bits on blockchain
6
7      function getStoredValue() public returns (uint256) {
8          return storedValue;
9      }
10
11     function setStoredValue(uint256 newValue) public {
12         storedValue = newValue;
13     }
14 }
```

- Pragma's are instructions to compilers about how to treat source code.
- From the same source code we can create multiple instances (each will be deployed in a different address).
- If you do any modification you have to re-deploy a new contract.

# Transactions and Calls (Queries)



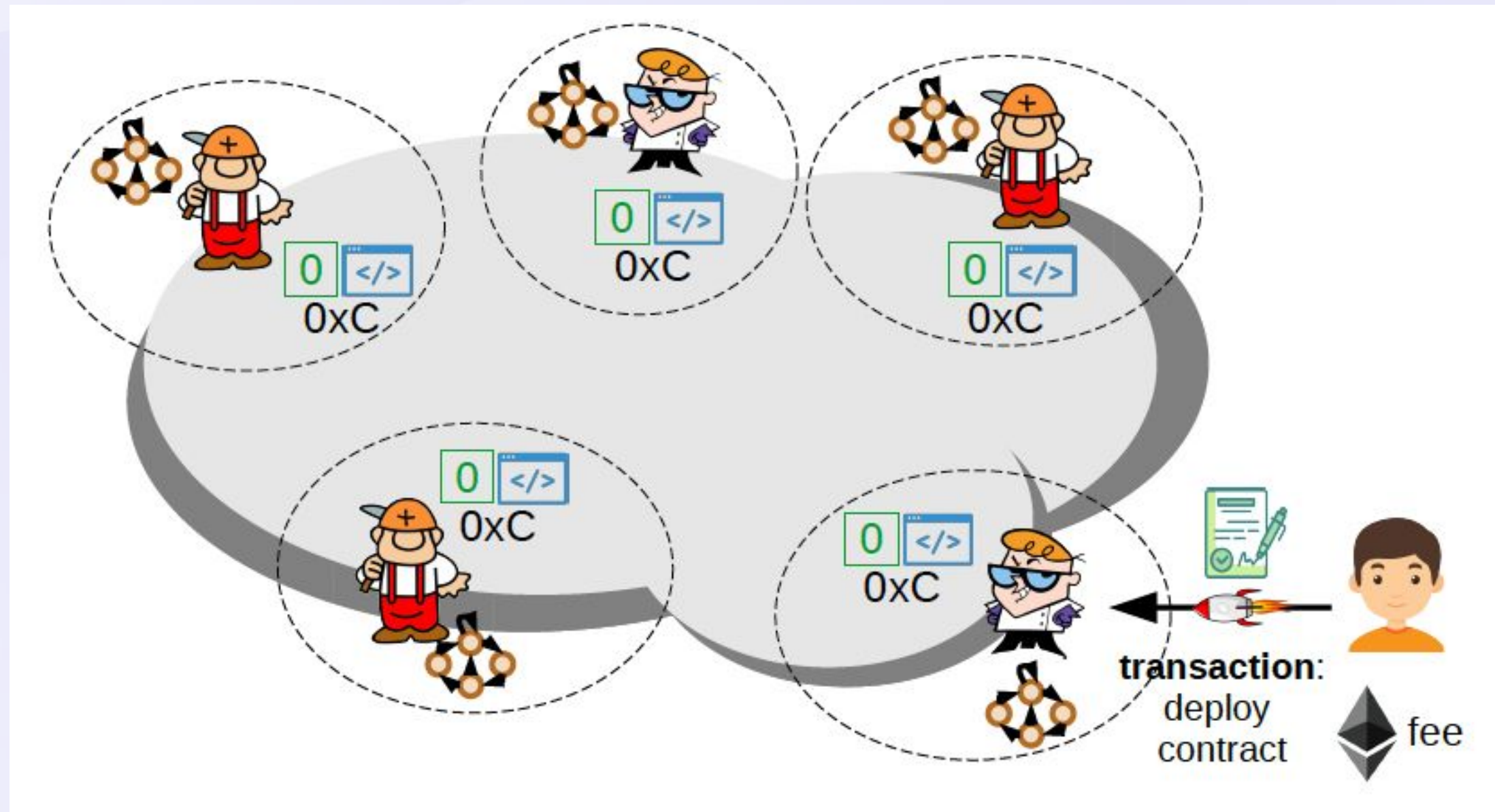


# View Functions

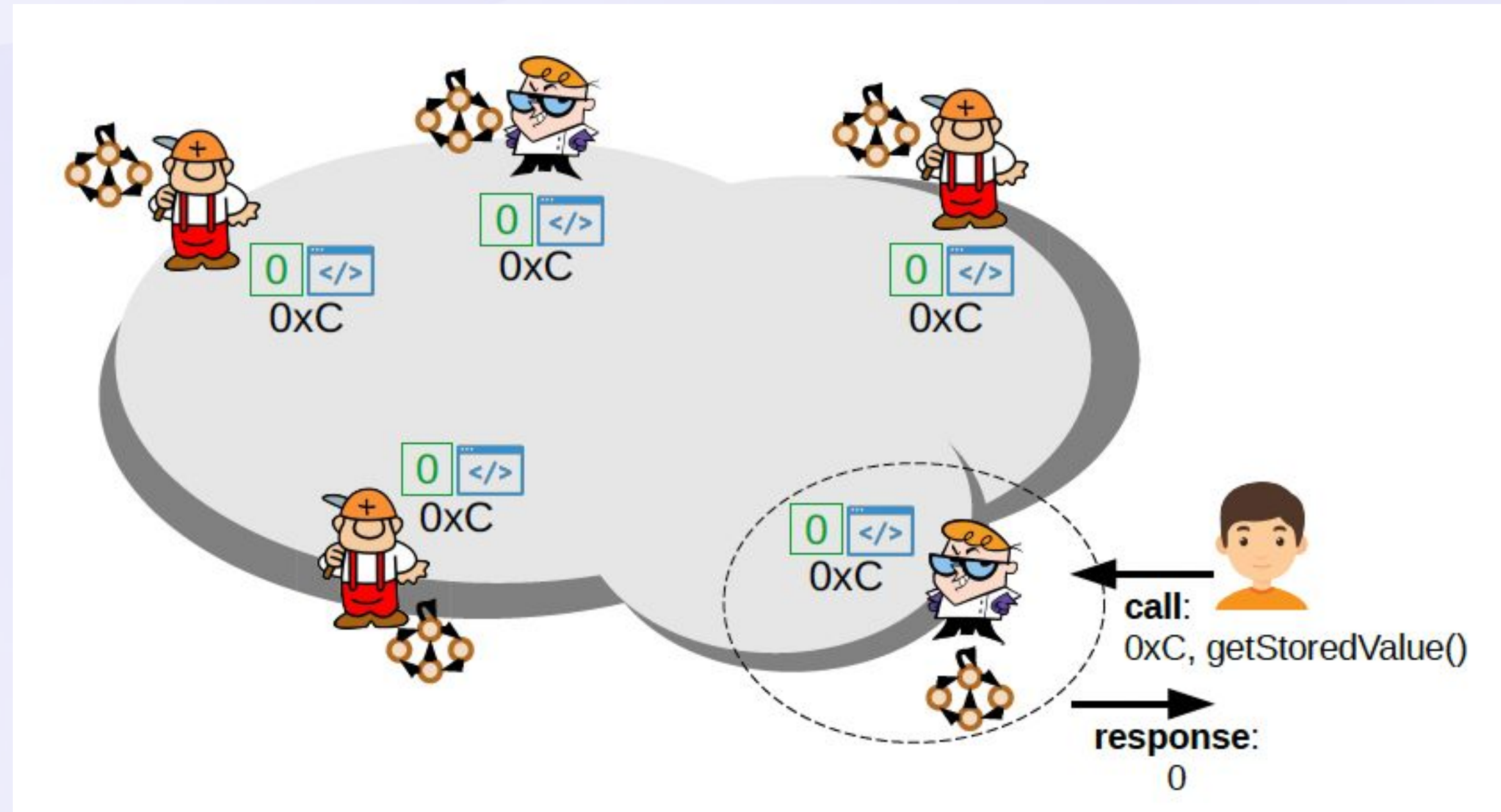
```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.0;
3
4  contract SimpleStorage {
5
6      uint256 storedValue;
7
8      function getStoredValue() public view returns (uint256) {
9          return storedValue;
10     }
11
12     function setStoredValue(uint256 _newStoredValue) public {
13         storedValue = _newStoredValue;
14     }
15 }
```

- View functions do not require a transaction only a call.
- View functions can return immediately (transactions need to be ordered by consensus).

# Deploy a Smart Contract

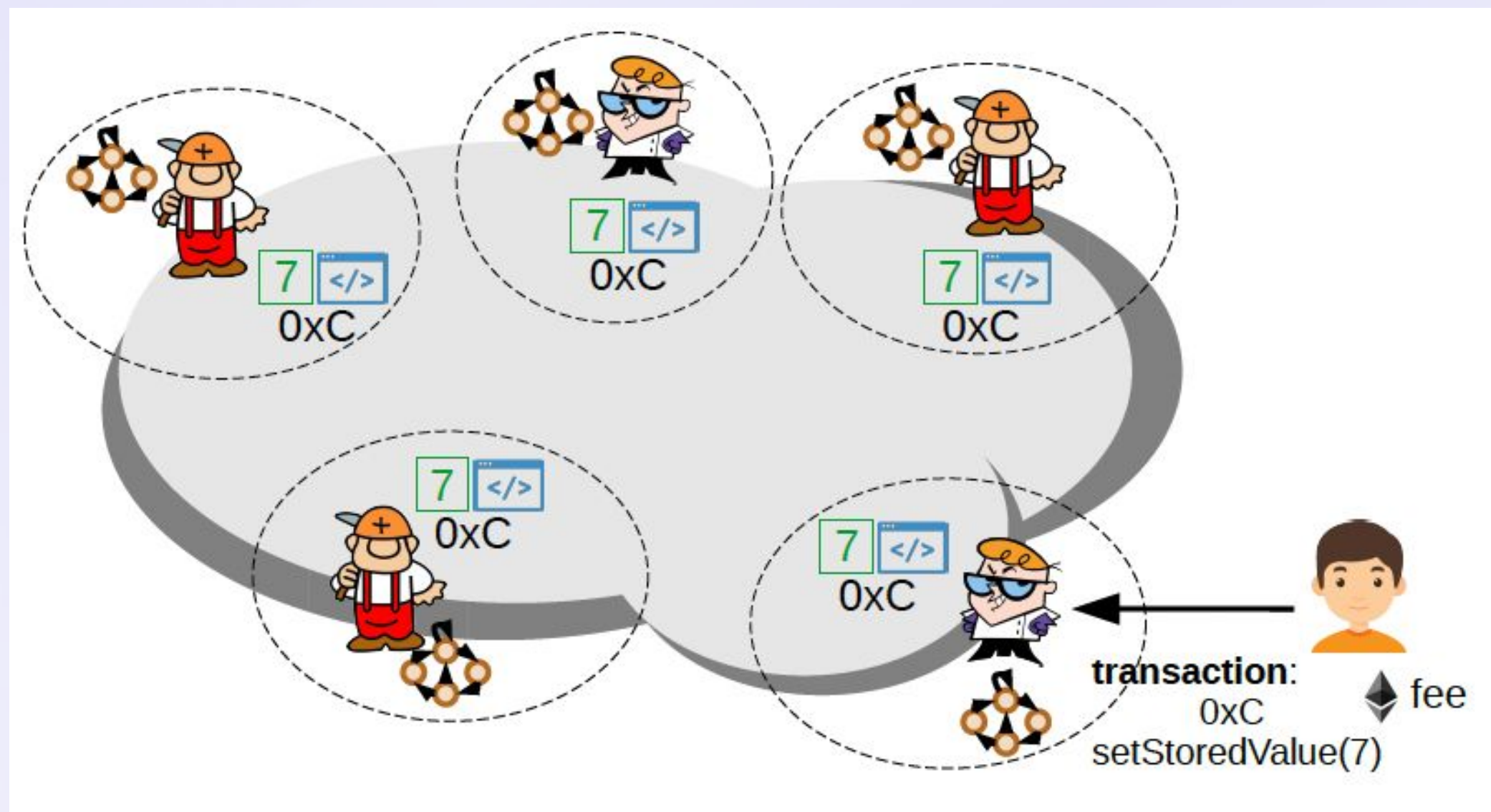


# Call an Existing Smart Contract





# Transact with an Existing Smart Contract



# Public State Variables

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract SimpleStorage {
5
6     uint256 public storedValue;
7
8     function setStoredValue(uint256 newValue) public {
9         storedValue = newValue;
10    }
11 }
```

- We can get rid of the getter function making the state variable public.
- The language (solidity) creates a getter function automatically when compiled.
- If deployed in the public network, anyone can send a transaction calling setValue() and change the state of this contract.



# Summary

- Blockchain Main Features
- Smart Contracts Concept
- Anatomy of a Smart Contract
- Basic Ethereum Remix
- Simple Smart Contract
- Deploy a Smart Contract







Co-funded by the  
Erasmus+ Programme  
of the European Union





# Module 10: SMART CONTRACTS

## Lecture 1: Simple Smart Contracts



Co-funded by the  
Erasmus+ Programme  
of the European Union



## TABLE OF CONTENTS

<b>1 INTRODUCTORY PARAGRAPH</b>	<b>3</b>
<b>2 LECTURE NOTES</b>	<b>4</b>
<b>3 PRACTICAL EXERCISES</b>	<b>5</b>
SIMPLE STORAGE CONTRACT	5
3.1 INTRODUCTION	5
3.2 REMIX SETUP	5
3.3 PROGRAMMING THE SMART CONTRACT	6
3.3.1 SIMPLE STORAGE	6
LICENSE	6
STORAGE VARIABLE	7
3.3.2 ADVANCED STORAGE	12
CONTRACT CODE	12
MAPPING AND ADDRESS DATA TYPES	13
FINAL CONTRACT VIEW	14
<b>4 CASE STUDY</b>	<b>15</b>
ESCROW SMART CONTRACT	15
4.1 DESCRIPTION	15
4.1.1 THE DEPOSIT	15
4.1.2 THE WITHDRAW	16
4.2 IMPLEMENTATION	16
<b>5 QUESTIONS AND ANSWERS</b>	<b>17</b>
DESCRIPTION	17
QUESTION AND ANSWER No.1	17
QUESTION AND ANSWER No.2	17
<b>6 MULTIPLE-CHOICE QUESTIONS</b>	<b>20</b>



## **1 INTRODUCTORY PARAGRAPH**

This document presents lecture notes, practical exercises, case studies, questions and answers, and multiple-choice questions related to Module 10, Lecture 1.



## 2 LECTURE NOTES

## 3 PRACTICAL EXERCISES

### Simple Storage Contract

#### 3.1 Introduction

In this practice, we are going to work on the concepts about smart contracts learnt in the theoretical part. The main purpose is to implement our first Ethereum smart contract. We remark that we will deploy it in a local EVM, not in a public network (this will be done in further lessons). With local EVM we mean an instance of Ethereum blockchain that we use for test purposes and runs only with one node in our computer, and in the case of this practice, in our browser.

To implement our first smart contracts, we will use:

- Remix: a free and open source integrated development environment (IDE) that we can use to write, compile, and debug Solidity code in our browser.
- Solidity: an object-oriented, high-level language for implementing smart contracts designed to target the Ethereum Virtual Machine (EVM).

The objectives of this practice are:

To implement our first smart contracts.

Learn how to use Remix.

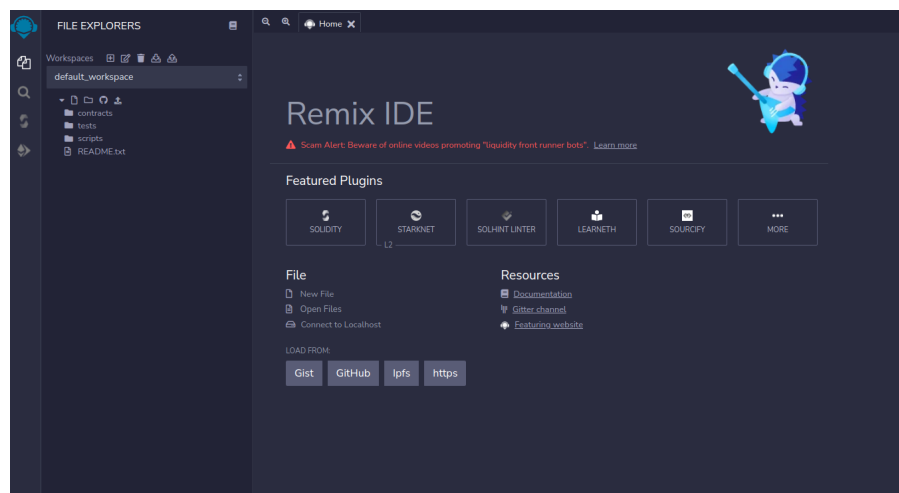
Practice and increase our knowledge of Solidity's syntax.

#### 3.2 Remix Setup

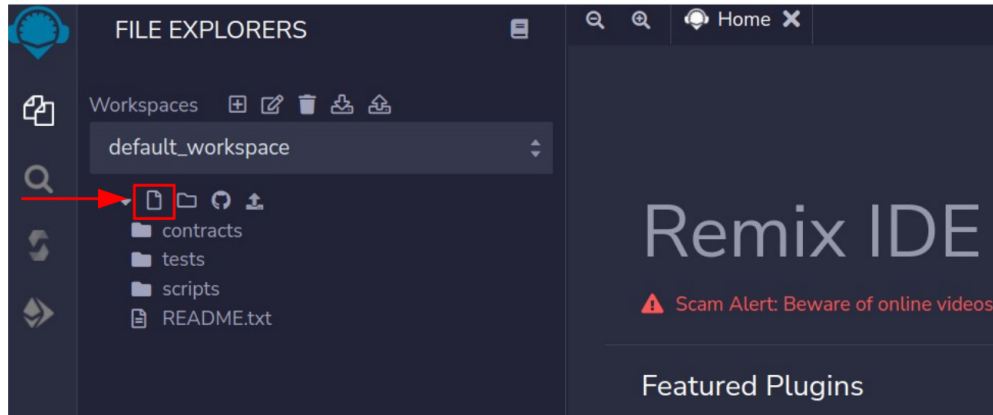
First of all, we are going to open the Remix IDE to start programming our first smart contract.

To open the remix web application, we have to open the following link from a web browser: **<https://remix.ethereum.org>**

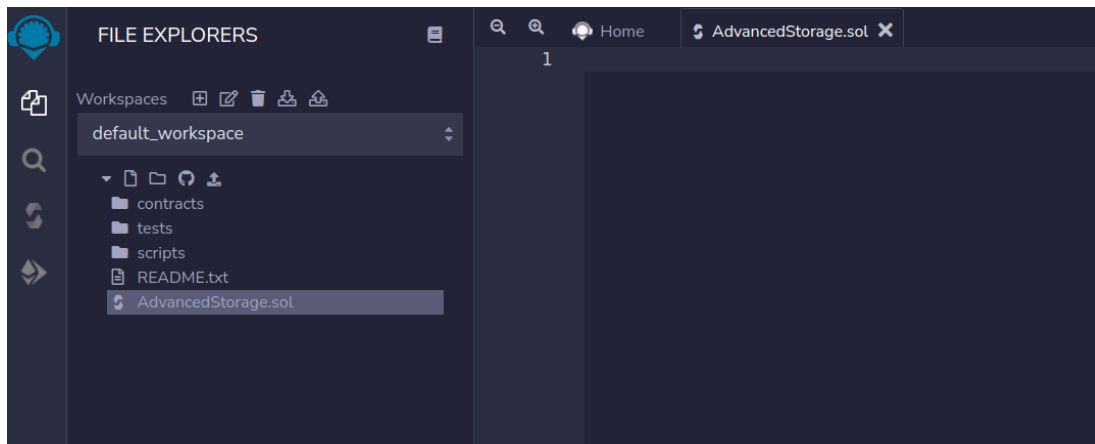
After opening the link in the browser, the following screen will appear:



After opening remix, we will create a new smart contract by clicking on the indicated button:



After clicking the button, a new document will appear, and we have to write the name for the file. In our case, we will write **AdvancedStorage.sol**:



Now, we have everything ready in remix to start programming our smart contract in this file.

### 3.3 Programming the smart contract

First of all, we are going to code the SimpleStorage contract shown in the theoretical part and then, expand it to a more advanced one.

#### 3.3.1 Simple Storage

In this smart contract, we want to allow anybody to set (change) the value of a storage variable, which means, to modify the blockchain state. Also, everyone will be able to execute a query to view the current value of the variable.

#### License

The first thing we need to do is to add the license for the smart contract. This part is not



required, but it is highly recommended and if you don't add it, Remix will throw you a warning.

The license is always specified on the first line of a smart contract:

```
1 // SPDX-License-Identifier: MIT
```

In this case, we are specifying the MIT License, which is a permissive free software license originating at the Massachusetts Institute of Technology (MIT) in the late 1980s.

## Compiler version

Then we have to add the compiler version of that we want to use:

```
1 pragma solidity ^0.8.0;
```

The compiler transforms the Solidity code into low level instructions called OPCODEs that can be interpreted and executed by the Ethereum Virtual Machine (EVM). In this case, we are specifying that we can use any compiler with a version greater or equal than 0.8.0.

## Contract scope

Now, we can define our smart contract by adding the following:

```
1 contract SimpleStorage {  
2  
3 }
```

The contract's logic will be defined inside contract scope (between the {}).

## Storage variable

Storage means that the value of this variable is permanently stored on-chain (forms part of the blockchain's state). Variables declared inside the contract scope but outside of any function are storage by default. uint256 type refers to a 256-bit unsigned integer numeric type. Also, In Solidity there are more types (booleans, strings, bytes, etc).

To declare a uint256 storage variable called storedValue add the following code:

```
1 uint256 public storedValue;
```

With the use of the public keyword, during the compilation stage, the Solidity compiler will automatically define a public view function to query the variable value (Getter). With this getter, everyone can query its value.

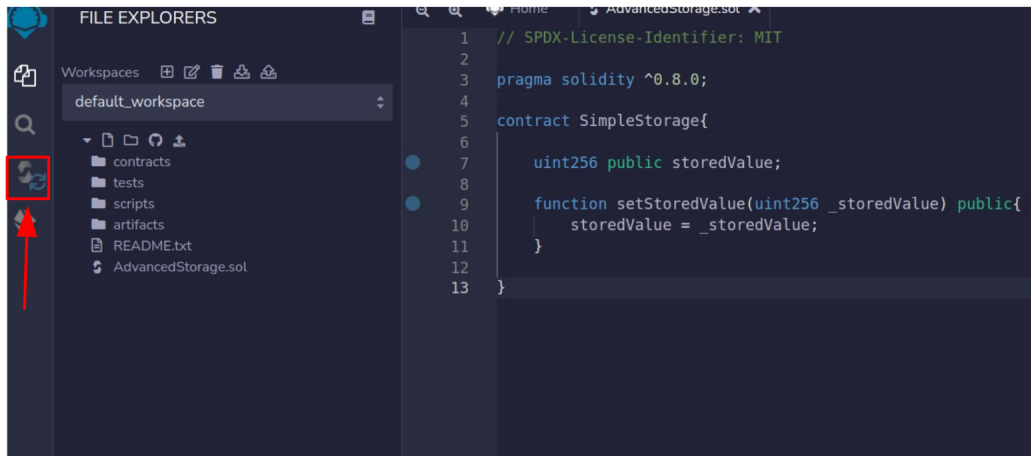
## Function definition

This function replaces the value of storedValue with the value received as argument. Notice that the function is declared as public, which means that it can be called

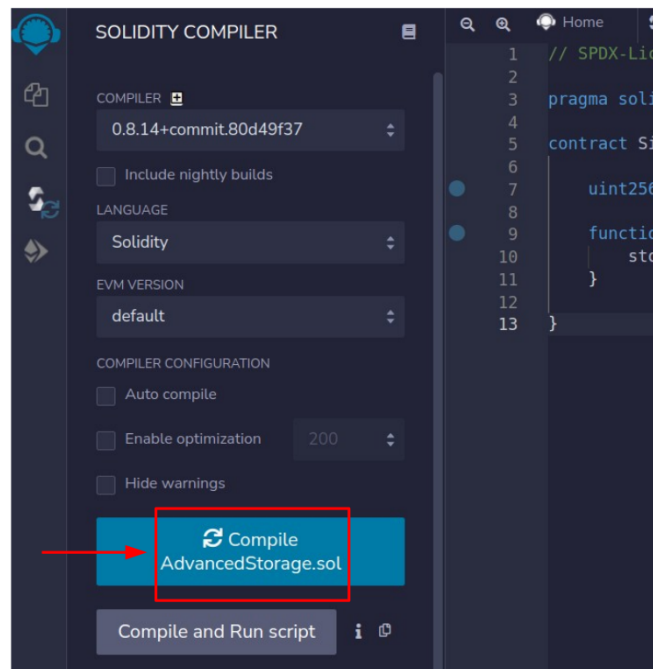
externally (from an external account) and internally (from another function of the smart contract). This kind of parameter is called visibility modifier.

## Compiling the smart contract

Now, in order to be able to deploy our smart contract, we must compile it first. To go to compilation screen we have to click in the icon indicated below:

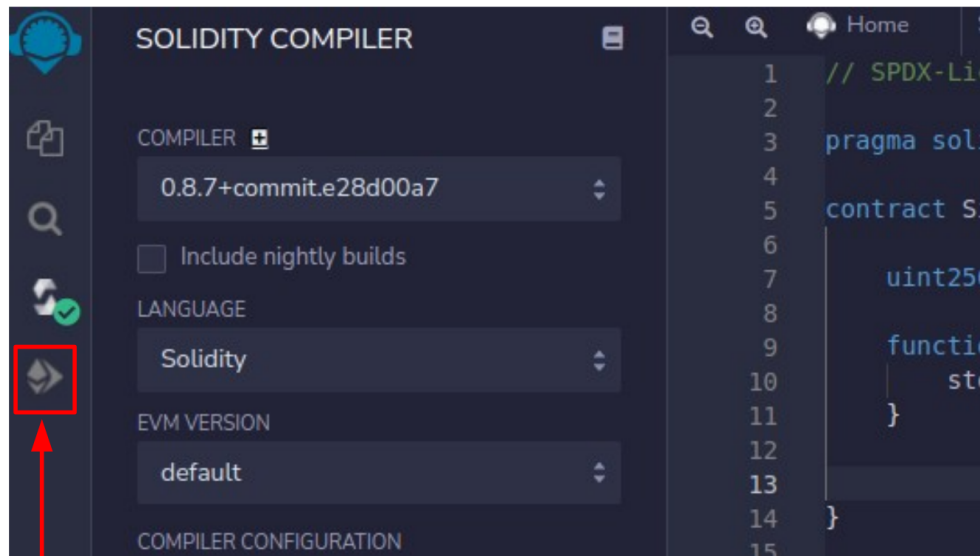


And then, we should click Compile. After that, if the compilation has been successful, we will already be able to deploy the contract. Otherwise the compiler will throw errors.



## Deploying the smart contract

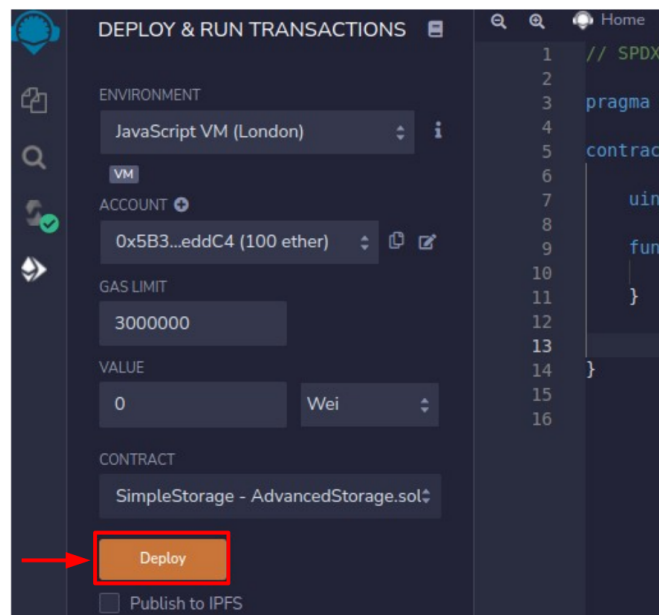
Finally, we are going to deploy the smart contract in a local network running inside Remix. By clicking in the button indicated below we are going to move to "Deploy & Run transactions screen" tab:



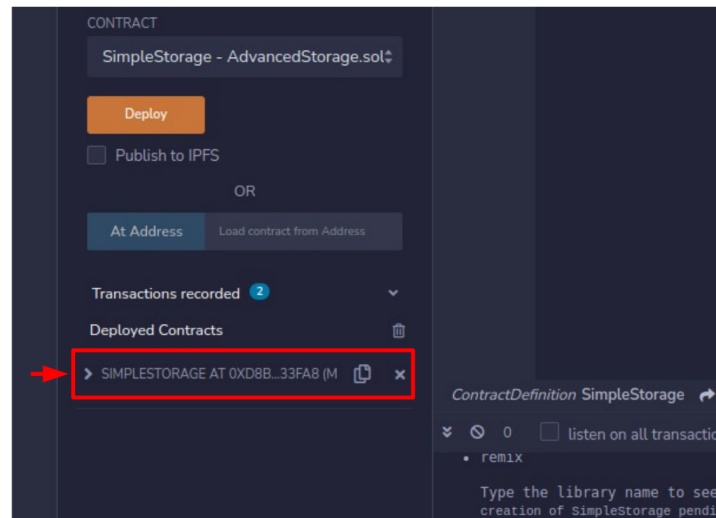
In this menu, we have many configuration parameters for the deployment and contract debugging.

With the "ENVIRONMENT" parameter set to London the contract is going to be deployed in local EVM.

Now we have to select the already compiled contract in the "CONTRACT" dropdown menu, and click on Deploy.



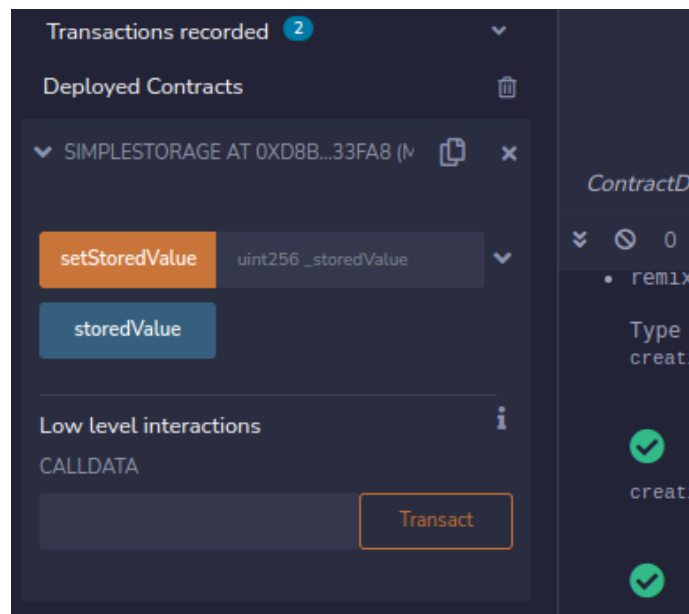
After this, the smart contract should be already deployed and it will appear a drop-down interaction menu. Note that the deployed contract has been assigned to a specific ethereum address.



### Interacting with the smart contract

Now with the smart contract already deployed, we can interact with it by using the user interface generated by remix.

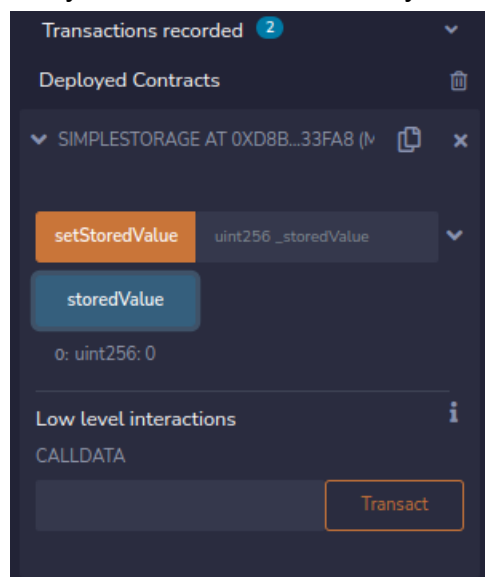
This user interface should look like:



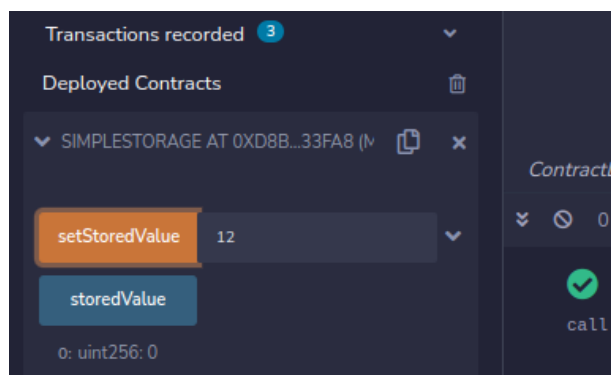
In the GUI we have the following buttons:

- The **setStoredValue** button is used to call the *setStoredValue* function. The orange colour of this button means that we have to send a call transaction to execute it and pay the corresponding fees. This is because the execution of this function, in a real situation, might change the blockchain state in all nodes of the network.
- The **storedValue** button is used to call *storedValue* getter function, and will return its value. The blue colour of this button means that it is a view function and we do not have to pay fees to call it because this function doesn't change the state and only supposes a query to a single node.

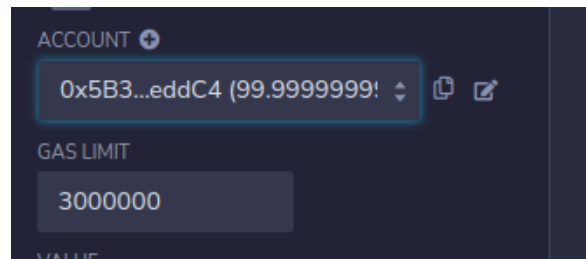
Solidity variables are set to 0 by default, let's check it by clicking on the blue button.



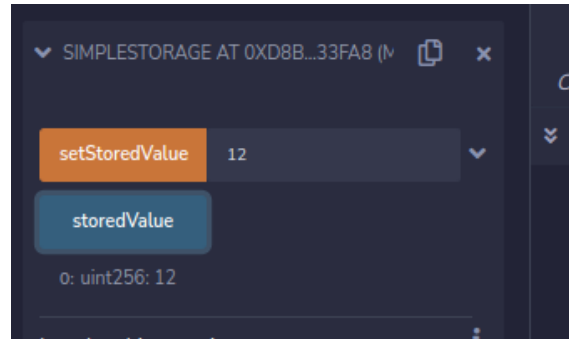
Now, we are going to change the value of storedValue to 12. To do that we have to call *setStoredValue* and pass the value as function parameter (fill the parameter box and click the orange button).



Remember that in order to call this function, an externally owned account has to pay the fees, because a transaction is needed. If you look at the accounts, you will notice that the first account now does not have 100 ethers but a little less because it paid for the fees of the transaction.



If we query again storedValue we will see that it is 12.



### 3.3.2 Advanced Storage

In this section, we are going to create a new simple storage contract but with extra logic.

- The new contract should have the following characteristics:
- The initial value of storedValue has to be 5.
- Should have a unique owner address allowed to call setStoredValue and addAccount (detailed later) functions.
- Should have an addresses list named allowedAccounts which only has permission to call setStoredValue.

We already saw how to create, compile and deploy smart contracts so now, let's focus on the new logic.

#### Contract code

The new contract will include three storage variables, a constructor function, a function to set the value of storedValue and a function to add accounts in allowedAccounts list.

### Mapping and address data types

We will use the owner variable to store the contract owner address. Solidity has a 20 bytes length special data type to store addresses.

We will use the allowedAccounts mapping to store the list of accounts allowed by the owner.

Note that mapping type is useful because it allows efficient search over a key-value list, being more gas efficient than an array. In our case, the keys types are addresses and the values types are bool.

```
1 uint256 public storedValue;  
2 address public owner;  
3 mapping(address => bool) allowedAccounts;
```

### The constructor function

A constructor in Solidity is a special function that is executed only once, when the contract is deployed. The constructor is typically used to initialize state variables.

In our contract we will use the following constructor:

```
1 constructor() {  
2     storedValue = 5;  
3     owner = msg.sender;  
4 }
```

Notice that we are setting the initial value of storedValue to 5, and initial owner value to the account address that sends the deployment transaction. msg.sender is a Solidity global variable that contains the transaction sender address.

### Require syntax

#### *setStoredValue*

In order to implement ownership functionality we are going to use Solidity's require() function, that is a special function which validates a boolean condition. If the condition evaluates to false, the calling transaction will revert.

```
1 function setStoredValue(uint256 _storedValue) public{  
2     require(msg.sender == owner || allowedAccounts[msg.sender] == true);  
3     storedValue = _storedValue;  
4 }
```

In our case, if the account that is calling the function is not the owner or an allowed account, the transaction will revert without changing the storedValue variable.

#### *addAccount*

To add an account to the allowedAccounts list, we have to define the following function:





```
1 function addAccount(address _account) public {  
2     require(msg.sender == owner);  
3     allowedAccounts[_account] = true;  
4 }
```

This function can only be called for the owner, if the account that is calling this function is not the owner, again the transaction will revert without changing allowedAccounts list.

## Final contract view

As the last step, let's put all our code together:

```
1 // SPDX-License-Identifier: MIT  
2  
3 pragma solidity ^0.8.0;  
4  
5 contract AdvancedStorage{  
6  
7     uint256 public storedValue;  
8     address public owner;  
9     mapping(address => bool) allowedAccounts;  
10  
11     constructor() {  
12         storedValue = 5;  
13         owner = msg.sender;  
14     }  
15  
16     function setStoredValue(uint256 _storedValue) public{  
17         require(msg.sender == owner || allowedAccounts[msg.sender] == true);  
18         storedValue = _storedValue;  
19     }  
20  
21     function addAccount(address _account) public {  
22         require(msg.sender == owner);  
23         allowedAccounts[_account] = true;  
24     }  
25  
26 }
```

Now, compile, deploy and test the contract characteristics using already described Remix functionalities.

Extra challenges:

- Read about modifiers and use them to structure your code better <https://solidity-by-example.org/function-modifier/>.
- Add, test and verify a new function to allow only the owner to remove addresses from allowedAccounts mapping.



## 4 CASE STUDY

### Escrow Smart Contract

One of the main uses of smart contracts is in those applications where we need to manage value between parties and it is not possible for them to trust each other (e.g. buying through the Internet). The immutability, high-level programmability, and built in value management mechanics (programmable money) of some blockchains, such EVM based ones, make them ideal for this type of applications.

In the following use case, we are going to analyse how to implement one of these applications with a smart contract.

#### 4.1 Description

We have two people, Alice and Bob. Alice wants to give 1 Ether to Bob and she doesn't want to allow Bob to spend that value until a month has passed, but she has no guarantee that Bob will fulfill the deal.

The traditional way of solving this is to introduce an intermediary(third source of trust) that receives the money from Alice and pays it to Bob after a certain condition has been met, in our case, that a month has passed.

This type of service is called an escrow. An escrow is a legal concept that describes a financial instrument whereby an asset or money in custody is held by a third party on behalf of two other parties who are in the process of completing a transaction.

However, this solution has two problems: on the one hand, Alice and Bob probably might have to pay a high fee for the escrow service and, on the other hand, and most importantly, they have to trust the third party providing the service.

With a smart contract on a blockchain, there is no need for a trusted intermediary because the rules are written into the smart contracts (money programmability) and no one can manipulate these rules (immutability).

##### 4.1.1 The deposit

First, Alice, who wants to give 1 Ether to Bob, has to create and deploy an instance of an escrow smart contract where she puts the correct logic so that Bob can withdraw the ether only after a month. Let's explain how Alice can create this smart contract:

1. Alice writes the smart contract code, with the proper storage and functions.
2. Alice compiles and deploys the smart contract on the blockchain.
3. Alice deposits 1 Ether in the smart contract.



#### 4.1.2 The withdraw

Finally, when a month has passed since Alice's deposit, Bob can call a function to withdraw his Ether. Notice that if Bob tries to call the withdraw before 1 month, the transaction has to be reverted.

#### 4.2 Implementation

You must implement a smart contract for this use case. We suggest you use remix IDE to write, deploy and test your smart contract. The smart contract can be coded in different ways, think about which would be the best to avoid malicious behaviours.

Below is shown a few tips to help you with implementation building process:

- Alice has one account and Bob another one to interact with the smart contract.
- Alice is the owner of the smart contract.
- In the constructor, you have to record in the smart contract storage the addresses of Alice and Bob.
- The smart contract has a function called `deposit()` which is payable. This function can only be successfully executed if called by Alice sending 1 Ether.
- The smart contract has a function called `withdraw()`. This function can only be successfully executed if called by Bob after 1 month has passed.
- For managing time, use `block.timestamp`, whose format is Unix time in seconds.

## 5 QUESTIONS AND ANSWERS

### Description

Below are five questions and answers that require some reflection.

### Question and Answer No.1

**Q:** Create a smart contract with a `addTwoIntegers` function to add two integers and store the result in a storage variable. The storage variable must be able to be queried by anyone through an external transaction. The solution code must have only one function declared.

**A:** Is not needed to declare explicitly a function to query storage variable, the use of "public" visibility modifier along with a storage declaration forces Solidity to create a getter during compilation.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Contract{

    int256 public result;

    function addTwoIntegers(int256 _a, int256 _b) public {
        result = _a + _b;
    }
}
```

### Question and Answer No.2

**Q:** Explain the purpose of the payable keyword in a function declaration, and the `address(this).balance` expression.

**A:** In Solidity, the payable keyword in a function declaration means that this function can receive Ether when it's executed, i.e. when the function is called the value of the calling transaction can be different from 0.

Balance is a property of the address type and when we call it, returns the ether balance that is holding that specific address. The "this" expression returns the current contract's instance type object; we can explicitly convert to Address type using "address(this)" expression.

So the expression `address(this).balance` returns the Ether balance of the current contract instance.

### Question and Answer No.3

**Q:** Modify the contract coded at Question 1 so that the storage can only be updated if 1 ether is sent along with the transaction. Constrain a max value of 5 eth in the contract balance.

**A:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Contract{

    int256 public result;

    function addTwoIntegers(int256 _a, int256 _b) public payable {
        require((msg.value == 1 ether) && (address(this).balance <= 5 ether));
        result = _a + _b;
    }
}
```

### Question and Answer No.4

**Q:** Implement ownership functionality to the contract. The owner must be automatically set to the deployer address during the deployment. Use a modifier to structure better your code and restrict the addTwoIntegers function to only accept calls from the owner.

**A:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Contract{

    int256 public result;
    address public owner;

    modifier onlyOwner {
        require(msg.sender == owner, "You are not the owner");
        _;
    }

    constructor(){
        owner = msg.sender;
    }

    function addTwoIntegers(int256 _a, int256 _b) public payable onlyOwner {
        require((msg.value == 1 ether) && (address(this).balance <= 5 ether));
        result = _a + _b;
    }
}
```

### Question and Answer No.5

**Q:** Modify the addTwoIntegers function in order to allow everyone to call it, and if in the call the max ether value in the contract is reached, all the funds have to be sent to the owner address. Use a constant MAX\_ETH to hold max ether value and avoid "Magic Numbers".

**A:**

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Contract{

    int256 public result;
    address public owner;
    uint256 constant MAX_ETH = 5 ether;

    modifier onlyOwner {
        require(msg.sender == owner, "You are not the owner");
        _;
    }

    constructor(){
        owner = msg.sender;
    }

    function addTwoIntegers(int256 _a, int256 _b) public payable {
        require((msg.value == 1 ether) && (address(this).balance <= MAX_ETH));
        if(address(this).balance == MAX_ETH){payable(owner).transfer(MAX_ETH);}
        result = _a + _b;
    }
}
```





## 6 MULTIPLE-CHOICE QUESTIONS

Below are ten multiple-choice questions. One or multiple choices can be correct. Correct alternatives are marked in bold.

### Multiple Choice question No. 1

**Q:** Regarding the deployed code of an instance of a smart contract:

- Only the miners can change the deployed code.
- Deployed code is immutable.
- Deployed code is mutable.
- Only the users can change the deployed code.

### Multiple Choice question No. 2

**Q:** Regarding the storage of a deployed instance of a smart contract:

- The storage of a deployed smart contract is immutable.
- The storage of a deployed smart contract can be muted by functions.
- We cannot control which accounts will be able to mute the storage.
- Can only be muted by the constructor function i.e. during deployment.

### Multiple Choice question No. 3

**Q:** In the context of smart contract code, the expression `address(this)`:

- Changes the address of current to contract's instance to "this" variable.
- Returns current contract's instance type object.
- Returns the address of the current contract's instance.
- Returns the ether balance of "this" address.

### Multiple Choice question No. 4

**Q:** Regarding Ethereum transactions:

- A transaction cannot be used to execute a function of a smart contract.
- The execution of a transaction is free for externally owned accounts (EOAs).
- In order to execute any function of a smart contract the sending of a transaction is needed.
- All answers are incorrect.

### Multiple Choice question No. 5

**Q:** Regarding queries to smart contracts:

- The execution of query functions has a cost because it changes the blockchain state.
- A getter is not a query function.



- An externally owned account (EOA) has to pay a fee for executing query functions.
- All the statements are incorrect.

### Multiple Choice question No. 6

**Q:** Regarding the steps to have a smart contract on a blockchain:

- First, Solidity code is written, then the Solidity code is deployed and finally the Solidity code is compiled to obtain EVM (Ethereum Virtual Machine) bytecode.
- First, Solidity code is written, then Solidity code is compiled to obtain EVM (Ethereum Virtual Machine) bytecode and finally, the bytecode is deployed.
- First, Solidity code is written, and finally, this code is deployed.
- All answers are incorrect.

### Multiple Choice question No. 7

**Q:** Regarding Solidity programming language:

- Solidity is designed to target the Bitcoin stack machine.
- Solidity is a very low-level language for implementing smart contracts in which you write programs using byte code.
- Solidity is a high-level, strictly typed language designed to be compiled to bytecode that targets the EVM (Ethereum Virtual Machine).
- All answers are incorrect.

### Multiple Choice question No. 8

**Q:** Which of the following is not a solidity data type?

- uint256.
- address.
- bool.
- transaction.

### Multiple Choice question No. 9

**Q:** In Solidity, when a function is defined adding the keyword public:

- The function can be called externally and internally (from another function).
- The function cannot be called externally.
- The function can be called externally but not internally (from another function).
- The function can be called internally (from another function) but not externally.

### Multiple Choice question No. 10



**Q:** Regarding the structure of a smart contract, which of the following statements is correct?

- We first specify the license, then the compiler version and, finally, the contract code.
- We first specify the compiler version, then the license and, finally, the contract code.
- We first specify the compiler version and then the contract code.
- The other statements are incorrect.



Questions? Contact us.

