Selecció d'arquitectures i eines de programació.

Models d'execució de codi.

Si els llenguatges de programació es classifiquen quant a la manera d'executar-se, diferenciar dos grans tipus:

- **Compilats**: aquells que necessiten d'un compilador com a pas previ a l'execució, no fent immediat l'accés al codi font. Tot i això, els llenguatges compilats acostumen a ésser més ràpids que els interpretats en temps d'execució.

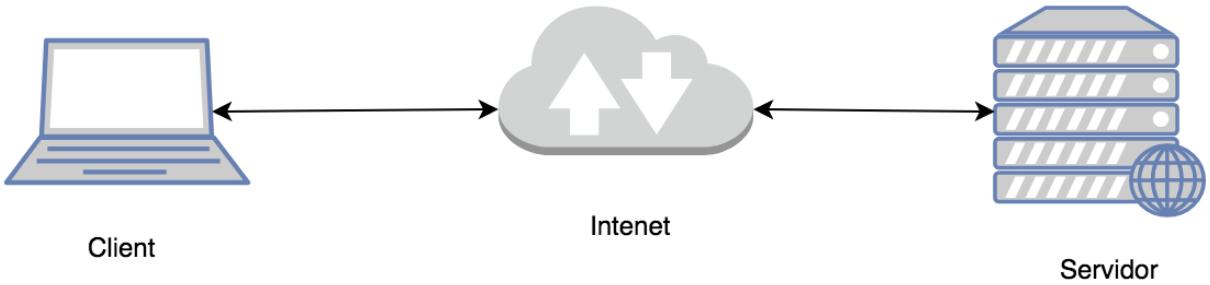
  Exemples: C, Java (a Bytecode), Pascal...

- **Interpretats**: aquells que s'executen mitjançant un intèrpret, que processa les comandes que inclou el programa una a una, fent que siguin menys eficient en

temps d'execució que els compilats. Tot i això, tenen l'avantatge que el codi font és públic, ja que no cal compilar-lo per a la seva execució.

Exemples: Javascript, Python, PHP, Ruby...

Tanmateix, donada una arquitectura client-servidor, els llenguatges de programació també es poden classificar segons on s'executen:



- **Client**: Principalment gestionen l'aparença de la pàgina web i el seu contingut, però poden interactuar amb el servidor i dades locals, tals com les galetes.

  Exemples: HTML, JavaScript, CSS, Ajax, jQuery...

- **Servidor**: Acostumen a executar tasques relacionades amb el processament de peticions de clients, enviament de pàgines web, estructuració d'aplicacions web, integració amb servidors i bases de dades, gestió de les bases de dades, codificació de dades a HTML…
  Exemples: Java, Python, PHP, Ruby…

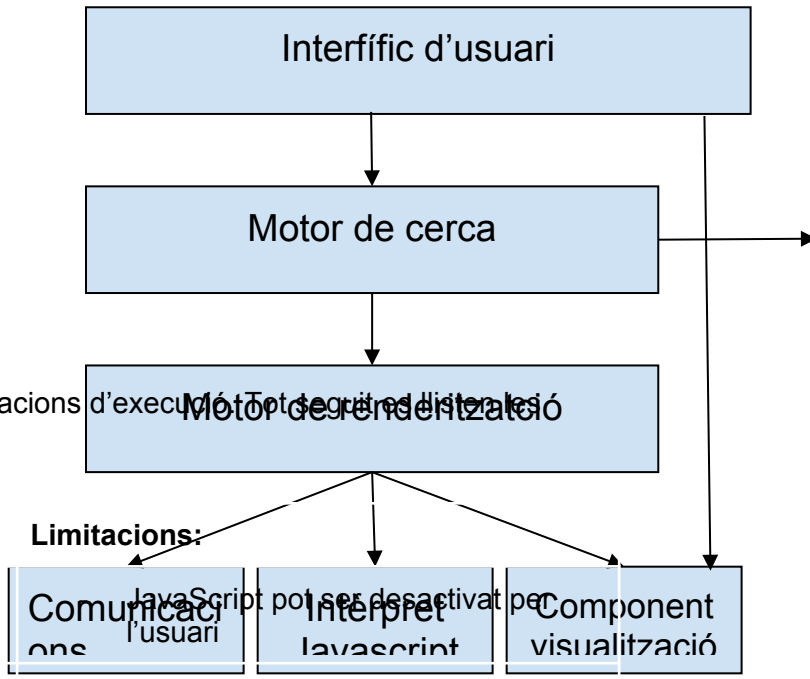
Codi font

Mecanismes d'execució de codi en un navegador web.

Tot i que cada navegador disposa de la seva pròpia arquitectura, la gran majoria disposen de blocs comuns anomenada arquitectura de referència. Aquesta arquitectura està formada pels següents blocs:

- **Interfície d'usuari:** cadascuna de les parts visibles del navegador, menys la que mostra la pàgina web.
- **Motor de cerca:** carrega una adreça determinada i suporta els mecanismes bàsics com anar una pàgina endarrere, endavant o tornar a carregar la mateixa pàgina. També gestiona les alertes JavaScript que es mostren a l'usuari i el procés de càrrega d'una pàgina nova.
- **Motor de renderització:** s'encarrega de generar la pàgina web que s'ha demanat al servidor a partir del codi que li ha arribat a través d'internet, establint les dimensions exactes a mostrar de cada mòdul.

- **Comunicacions:** gestiona tot allò que té a veure amb peticions de xarxa, com els protocols de transferència de fitxers i documents utilitzats a internet. A més, identifica la codificació de les dades obtingudes per saber si és de tipus adio, video, text...
- **Intèrpret JavaScript:** és motor encarregat d'interpretar el codi JavaScript al navegador, seguin l'estàndard ECMAScript.
- **Component de visualització:** ofereix funcionalitats relacionades amb la visualització dels continguts HTML d'una pàgina web, tals com primitives de dibuix i posicionament a una finestra, o fonts tipogràfiques, entre d'altres.
- **Persistència de dades:** funciona com magatzem de dades al navegador per emmagatzemar galtes, certificats,historial i sessions d'usuari entre d'altres.



Capacitats i limitacions d'execució

JavaScript, presenta certes avantatges i limitacions d'execució. Totes seguit es llisten les principals:

Avantatges:

- Requereix de menys interacció amb el servidor
- Reacciona amb la interacció amb l'usuari
- Modifica estils i continguts

Limitacions:

- JavaScript pot ser desactivat per l'usuari
- No interactua directament amb els llenguatges del servidor
- No pot accedir a: les preferències del navegador, finestra...

- Pre-processa dades al client
- Multiplataforma, pot ser executat per gairebé qualsevol combinació de sistema operatiu i navegador web
- No pots accedir al sistema de fitxers del client
- Cada navegador web disposa d'un intèrpret diferent, que pot produir resultats lleugerament diferents segons el navegador.

Llenguatges de programació en entorn client

Els llenguatges de programació en entorn client son aquells que s'executen al navegador web, és a dir, al costat client donada una arquitectura client-servidor. El llenguatge client per excel·lència correspon a HTML, ja que la gran majoria de pàgines web estan codificades en aquest llenguatge per a descriure en forma de text l'estructura i el contingut. Amb l'objectiu de millorar la interacció amb l'usuari, es poden incloure tots els llenguatges d'script -és a dir, de guions-, com JavaScript o VBScript.

Toti això, hi ha una gran varietat de llenguatges en l'entorn client. Els principals estan llistats i breument descrits a continuació:

- HTML i derivats:
  - **HTML**: de l'anglès *Hyper Text Markup Language* (llenguatge de marques d'hipertext). És el llenguatge de marques més utilitzat a Internet.
  - **Dinàmic HTML (DHTML)**: integració d'HTML amb llenguatges d'*scripting,* fulls d'estil personalitzats (CSS) i la identificació de continguts d'una pàgina web en format d'arbre (DOM).
  - **XML**: llenguatge utilitzat per a descriure dades per a la seva transferència eficient sense mostrar-les, com al cas d'HTML.
  - **EML**: implementació de XML per a eleccions.
  - **XHTML**: adaptació d'HTML al llenguatge XML.
- Llenguatges de programació de guions *(script)*:
  - **JavaScript**: el llenguatge de programació *d'scripting* més utilitzat.
  - **VBScript**:llenguatge de programació de guions basat en Visual Basic, de Microsoft.
- Altres:
  - **CSS**: de l'anglès *Cascade Style Sheets* (fulls d'estil en cascada). Serveixen per separar el format que es vol donar a la pàgina web de l'estructura de la pàgina web i les instruccions.
  - **Applets de Java**: petits components (objectes independents) integrats en una pàgina web i programats en Java. Actualment en desús.
  - **Adobe Flash**:tecnologia d'animació d'Adobe que utilitza ActionScript com a llenguatge principal.

Programació de guions

El primer llenguatge de programació de guions fou LiveScript, que es va desenvolupar per Netscape com a alternativa més lleugera a Java per a suportar comportaments dinàmics tan

al client com al servidor. Més tard, l'any 1995, Netscape i Sun va llançar Navigator 2 i també van aprofitar per a canviar el nom de LiveScript a JavaScript, aprofitant la popularitat de Java.

L'èxit de JavaScript va provocar que Microsoft llancés la seva pròpia versió de JavaScript - tot i que ja disposaven de VBScript-, denominada JScript. A més, JScript estava integrat amb els seus propis navegadors web a partir d'Internet Explorer 3.

Tot i que les diferències entre JavaScript i JScript no eren gaires, obligava als desenvolupadors a programar dues vegades la mateixa funcionalitat, una per a cada llenguatge. A més, a les pàgines web s'havien d'introduir sentències condicionals per a identificar quin navegador estava executant la pàgina i poder executar un codi o l'altre.

Com a conseqüència d'aquestes limitacions, l'ECMA *(European Computer Manufacturers Association)* va iniciar un esforç d'estandardització que va derivar en la publicació de l'estàndard ECMAScript. A dia d'avui, tan JavaScript com JScript són conformes a l'esmentat estàndard, tot i que JavaScript s'ha imposat i s'utilitza com a denominador comú per a referir-se al propi llenguatge i a l'estàndard.

Integració del codi amb les etiquetes HTML

Hi han dues maneres per a integrar codi Javascript a un fitxer html:

1) Inserint el codi JavaScript en el mateix document HTML

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple integració 1</title>
	</head>
	<body>
		<!-- obro tag per inserir codi JavaScript -->
		<script>
			//Aquí va el codi JavaScript
		</script>
		<!-- tanco tag per inserir codi JavaScript -->
	</body>
</html>
```

2) Referenciant el document JavaScript al document HTML

```
<!DOCTYPE html>
<html>
	<head>
		<!--  obro tag script i amb
		source indico la localització del
		fitxer de JavaScript -->
		<script src="./js/codi.js"></script>
```

```
            <title>Exemple integració 2</title>
      </head>
      <body>
      </body>
</html>
```

Eines de programació sobre clients web. Tecnologies i llenguatges associats

Per a programar en JavaScript únicament és necessari un editor de text com *gedit,* que ja incorpora edició i verificació de sintaxis per colors. Tot i això, també existeix la possibilitat de programar amb IDEs (de l'anglès *Integrated Development Environment)*, que ja incorporen les eines requerides tant per escriure part de les etiquetes automàticament com per testejar el codi.

Aplicació i verificació de la sintaxi del llenguatge

Selecció de llenguatge de programació de clients web

De tots els llenguatges de programació web en l'entorn client, el més utilitzat per aportar dinamisme a les pàgines web és JavaScript. Javascript s'acostuma a utilitzar amb el llenguatge de marques HTML i el codi CSS, per tal de proporcionar a la plana web l'estructura i l'estil desitjat.

Comentaris

Es poden inserir els comentaris al codi JavaScript amb els símbols:

- "//"per a comentaris d'una línia
- "/*" i "*/" per obrir i tancar el comentari, respectivament

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple comentaris</title>
		<!-- això és un comentari en HTML -->
	</head>
	<body>
		<script>
			//això és un comentari d'una línia en JS
			/*
			això és un comentari
			multilínia en JS
			*/
		</script>
	</body>
</html>
```

Sentències

En un llenguatge de programació, com JavaScript, cadascuna de les instruccions que formen el programa que executarà l'ordinador s'anomenen sentències.

Com es veurà a continuació, les sentències en JavaScript estan formades per comentaris, variables, operadors, condicionals, bucles i paraules clau.

Com a bona pràctica, encara que no és necessari, separem les sentències amb el punt i coma ";". També és recomanable utilitzar un únic punt i coma ";" per línia, tot i que Javascript permet declarar-ne varis en una mateixa línia.

Variables

A JavaScript, s'ha de tenir en compte certes normes i consideracions sobre les variables:

- Una variable es declara amb la paraula reservada «var».

  Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple declaració variables</title>
	</head>
	<body>
		<script>
			//declaració de la variable
			var a;
			//també es pot declarar més d'una variable
			//a la mateixa línía
			var b, c, d, e, f, g;
			//fins i tot com s'ha vist anteriorment
			//amb vàries sentències en una línia
			var h; var i; var j;
		</script>
	</body>
</html>
```

- Els identificadors de les variables:
  - Han de ser únics, és a dir, no es poden repetir.
  - Han de començar amb:
    - una lletra,
    - el símbol de subratllat "_",
    - o el de dòlar "$".
- Només poden contenir:
  - lletres,
  - números, i
  - els símbols "_" i "$".
- Tampoc es poden utilitzar paraules reservades per a declarar identificadors de variables ("while", "for", "new"...). La llista sencera es pot consultar, per exemple, a l'associació que defineix l'estàndard. A continuació està enllaçada l'estàndard d'aquest any: https://www.ecma-international.org/ecma-262/.
- Finalment, cal tenir en compte que a JavaScript:
  - Es distingeix entre majúscules i minúscules. No és el mateix la variable "a" que "A".
  - Tot i que no és obligatori declarar les variables, és convenient.
  - Les variables són dinàmiques (per exemple, poden canviar de tipus 'number' a 'string', no com a altres llenguatges).

Assignacions

Les assignacions de valors a variables es fan amb l'operador "=" i tenen lloc de dreta a esquerra.

Exemple:

NOTA:
- El mètode "window.prompt()" mostra una finestra a l'usuari on se li demana una entrada.
- El mètode "console.log()" mostra per consola.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple assignacions</title>
		</head>
	<body>
		<script>
			//declaració de la variable
			var a;
			//assignació del valor 17
			a=17;
			//declaració i assignació de la variable
			var b=17;
			//també es pot assignar un valor des de la pantalla
			var c=prompt("Introdueix un valor");
			console.log(c);
		</script>
	</body>
</html>
```

Tipus de dades

JavaScript distingeix entre els següents tipus de dades, entre d'altres:

- Undefined
- Number
- Strings
- Boolean

Undefined

El tipus de variable «undefined» correspon a una variable que ha estat declarada, però no inicialitzada. És a dir, no emmagatzema cap tipus de dada.

Exemple:

NOTA:
- L'operador "typeof" retorna un string indicant el tipus de dada de la variable.

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple undefined</title>
		</head>
	<body>
		<script>
			// declarem la variable «a» sense inicialitzar-la
			var a;
			// imprimim per consola el tipus amb la funció typeof
			console.log(typeof a); // tipus undefined
		</script>
	</body>
</html>
```

Number

Una variable és de tipus "number" quan emmagatzema un número. Es poden assignar diferents tipus de valors (enter, decimal) i bases (base octal, decimal i hexadecimal) a les variables. Així doncs, les variables de tipus número es poden assignar de diferents maneres:

- Amb un número enter var = 17;
- Amb un número decimal var=0.17;
- Amb notació exponencial var=17**e**3;
- En base octal var=**0**17; //el 1r dígit ha de ser un "0"
- En base hexadecimal var=**0x**17; //el valor ha de començar per "0x"

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple números</title>
		</head>
	<body>
		<script>
			//declarem la variable a
			var a;
			//mostrem el valor de la variable per consola
```

```
                console.log(a); // undefined
                //mostrem el tipus de la variable per consola
                console.log(typeof a); // undefined
                //assignem el número 17
                a = 17;
                console.log(a); // 17
                console.log(typeof a); // number
                a=0.17;
                console.log(a); // 0.17
                console.log(typeof a); // number
                a=17e3;
                console.log(a); //17000 en base 10
                console.log(typeof a); // number
                a=017;
                console.log(a); //15 en base 10
                console.log(typeof a); // number
                a=0x17;
                console.log(a); //23 en base 10
                console.log(typeof a); // number
        </script>
    </body>
</html>
```

NOTA: Per a esborrar el valor d'una variable, se li assigna el valor "undefined". Automàticament, tant el valor com el tipus de variable passa a ésser "undefined".

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemple esborrar variable</title>
        </head>
    <body>
        <script>
            //inicialitzem a valor 15
            var a=15;
            //imprimim el valor per consola
            console.log (a); //15
            //imprimim tipus per consola
            console.log(typeof a); //number
            //esborrem el valor de la variable a
            a=undefined;
            //imprimim el valor per consola
            console.log (a); //undefined
            //imprimim el tipus de variable per consola
```

```
                console.log (a); //undefined
            </script>
        </body>
</html>
```

Strings

Una variable és de tipus "string" quan emmagatzema una cadena de caràcters. Aquest tipus de variable es pot inicialitzar de dues maneres:

- Amb cometes simples, tant a l'inici amb al final de la cadena: " ' "
- Amb cometes dobles, tant a l'inici com al final de la cadena: " '' "

Exemple:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Exemple cadenes</title>
        </head>
        <body>
                <script>
                        var cad1 = "Hola";
                        var cad2 = 'Hola';
                        var cad3 = "M'agrada la xocolata";
                        // el simbol \ davant de ' indica que ' no té
                        //relació amb el final o inici de cadena
                        var cad4 = 'M\'agrada la xocolata';
                        console.log(cad1);
                        console.log(cad2);
                        console.log(cad3);
                        console.log(cad4);
                </script>
        </body>
</html>
```

Boolean

Una variable és de tipus «boolean» quan emmagatzema el valor lògic "true" o "false".

Exemple:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Exemple variables dinàmiques</title>
        </head>
        <body>
                <script>
                        //declaració i assignació com a true
```

```
                var a=true;
                //assignació com a false
                b=false;
            </script>
        </body>
</html>
```

NOTA: Com s'ha vist anteriorment, les variables són dinàmiques. És a dir, una variable no està lligada únicament a un únic tipus de dada, sinó que una variable pot anar canviant de tipus de dada.

Exemple:

```
<!DOCTYPE html>
<html>
        <head>
                <title>Exemple variables dinàmiques</title>
        </head>
        <body>
                <script>
                        //declaració de la variable
                        var a;
                        //comprovem el seu tipus
                        console.log(typeof a); // undefined
                        a=17;
                        console.log(typeof a); // number
                        a="Hola";
                        console.log(typeof a); // string
                        a=false;
                        console.log(typeof a); // boolean
                </script>
        </body>
</html>
```

Operadors

Hi han dos tipus d'operadors de variables:

- Aritmètics
- Lògics

Operadors aritmètics

Els operadors aritmètics s'utilitzen amb variables de tipus Number o String. Es poden diferenciar les següents operacions:

- Suma a=b+c;

- Resta a=b-c;
- Multiplicació a=b*c;
- Exponent a=b**c; //a=b^c
- Divisió a=b/c;
- Mòdul a=b%c
- Increment a++;
- Decrement a--;

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple Operadors Aritmètics</title>
	</head>
	<body>
		<script>
			//Operadors aritmètics
			var a, b, c;
			console.log("Declaració de variables");
			console.log("a = "+a+"");
			console.log("b = "+b+"");
			console.log("c = "+c+"");
			//assignació de valors
			console.log("Assignació de valors");
			a=0, b=10, c=20;
			console.log("a = "+a+"");
			console.log("b = "+b+"");
			console.log("c = "+c+"");
			//suma
			console.log("Operador suma '+'");
			a=b+c;
			console.log("a = b + c = "+a+"");
			//resta
			console.log("Operador resta '-'");
			a=b-c;
			console.log("a = b - c = "+a+"");
			//multiplicació
			console.log("Operador multiplicació '*'");
			a=b*c;
			console.log("a = b * c = "+a+"");
			//exponent
			console.log("Operador multiplicació '*'");
			a=b**c;
			console.log("a = b * c = "+a+"");
			//divisió
```

```
                    console.log("Operador divisió '/'");
                    a=b/c;
                    console.log("a = b / c = "+a+"");
                    //mòdul
                    console.log("Operador mòdul '%'");
                    a=b%c;
                    console.log("a = b % c = "+a+"");
                    //increment
                    console.log("Operador increment '++'");
                    b++;
                    console.log("El resultat de 'b++;' és "+b+"");
                    //decrement
                    console.log("Operador decrement '--'");
                    b--;
                    console.log("El resultat de 'b--;' és "+b+"");
            </script>
        </body>
<html>
```

Alternativament, alguns operadors aritmètics poden incloure l'operador d'assignació:

- Suma a+=b; // a=a+b;
- Resta a-=b; // a=a-b;
- Multiplicació a*=b; // a=a*b;
- Divisió a/=b; // a=a/b;
- Mòdul a%=b; // a=a%b;

Operadors lògics

Els operadors lògics retornen una variable de tipus boolean en funció de la comparació.

- Igual valor ==
- Igual valor i tipus ===
- Diferent valor !=
- Diferent valor o diferent tipus !==
- Més gran >
- Més petit <
- Més gran o igual >=
- Més petit o igual <=
- Conjunció lògica &&
- Disjunció lògica ||
- Negació lògica !
- Operador ternari ?[1]

[1] L'operador ternari es tracta en detall a la secció de condicionals

Condicionals

Les instruccions de condicions més habituals són:

- If
- Else
- Else if
- Switch

If

S'utilitza "if" quan es vol executar un bloc de codi si es compleix una condició en concret.

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Exemple if</title>
	</head>
	<body>
		<script>
			a=prompt("Entra un número");
			if (a%3==0){
				console.log(a+ " és divisible per 3");
			}
			if(a%3!=0){
				console.log(a+ " no és divisible per 3");
			}
		</script>
	</body>
</html>
```

Else

S'utilitza "else" quan es vol executar un bloc de codi si la condició "if" és falsa.

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Exemple if-else</title>
	</head>
	<body>
		<script>
			a=prompt("Entra un número");
			if (a%3==0){
```

```
                        console.log(a+ " és divisible per 3");
                }
                //Ja no cal insertar la validació
                //De l'anterior exemple
                else{
                        console.log(a+ " no és divisible per 3");
                }
        </script>
    </body>
</html>
```

NOTA:

- L'operador ternari (?) s'acostuma a utilitzar com a una drecera per al condicional "if". Tant l'anterior exemple com el posterior fan exactament el mateix.

Exemple:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Exemple if</title>
    </head>
    <body>
        <script>
            a=prompt("Entra un número");
            a%3==0 ? console.log(a+ " és divisible per 3") :
            console.log(a+ " no és divisible per 3");
        </script>
    </body>
</html>
```

Else-if

S'utilitza "else if" per especificar noves condicions després d'avaluar "if" i abans d'avaluar "else".

Exemple:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Exemple else if</title>
```

```
	</head>
	<body>
		<script>
			var num1 = prompt("Entra un número");
			var num2 = prompt("Entra un altre número");

			//comprovem si num1 és més gran que num2
			if(num1 > num2){
				console.log(+num1+" és més gran que "+num2+"");
			//num1 pot ser més petit o igual que num2
			//comprovem si és més petit
			} else if (num1 < num2){
				console.log(+num1+" és més petit que "+num2+"");
			//si num2 no és més petit
			//ni més gran que num2, són iguals
			} else {
				console.log("Els dos números són iguals");
			}
		</script>
	</body>
</html>
```

Switch

S'utilitza "switch" quan s'ha de comparar una expressió vàries vegades i només una és la correcta.

NOTA:

- Un string es passa a number amb la funció "parseInt()". Aquesta funció pot resultar útil ja que el mètode "window.prompt()" retorna tipus string, encara que s'hagi entrat un número, i pot ser necessari convertir el tipus a number per a realitzar certes operacions.

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Exemple switch</title>
	</head>
	<body>
		<script>
			var num1 = window.prompt("Entra el primer operand");
			var operacio = window.prompt("Entra operació (+,-,*,/)");
			var num2 = window.prompt("Entra el segon operand");
```

```
                    var res;
                    //com window.prompt() retorna strings
                    //cal passar-los a number per poder operar
                    num1=parseInt(num1);
                    num2=parseInt(num2);
                    switch(operacio){
                          case "+":
                                res=num1+num2;
                                console.log(""+num1+" + "+num2+" = "+res+"");
                                break;
                          case "-":
                                res=num1-num2;
                                console.log(""+num1+" - "+num2+" = "+res+"");
                                break;
                          case "*":
                                res=num1*num2;
                                console.log(""+num1+" * "+num2+" = "+res+"");
                                break;
                          case "/":
                                res=num1/num2;
                                console.log(""+num1+" / "+num2+" = "+res+"");
                                break;
                          //si l'usuari no ha entrat un dels valors anteriors
                          //s'executarà aquest default
                          default:
                                console.log("Has entrat un altre valor.");
                    }
              </script>
        </body>
</html>
```

Bucles

S'utilitzen per executar codi un número determinat o indeterminat de vegades.

- For
- While
- Do while

For

S'utilitza "for" quan es vol executar un bloc de codi un número determinat de vegades.

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Exemple for</title>
	</head>
	<body>
		<script>
			//Mostrem els 10 primers números enters
			for(i=1; i<=10; i++){
				console.log(i);
			}
		</script>
	</body>
</html>
```

While

S'utilitza "while" quan es vol executar un bloc de codi sempre que una condició sigui certa.

Exemple:

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Exemple while</title>
	</head>
	<body>
		<script>
			var factorial=1, num;
			var num=window.prompt("Entra un número");
			//es passa l'string a number per poder operar
```

```
                var num=parseInt(num);
                //calculem el factorial
                while(num>1){
                        factorial=factorial*num;
                        num--;
                }
                //mostra valor per pantalla
                window.alert("El factorial és "+factorial+"");
        </script>
    </body>
</html>
```

Do while

S'utilitza "do while" quan es vol executar un bloc de codi almenys una vegada, o més, sempre que la condició sigui certa.

Exemple:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Números primers</title>
    </head>
    <body>
        <script>
            var dividend, divisor, quocient, residu, esPrimer=true;
            //inicialitzem el valor del dividend, és a dir
            //el número que volem comprovar si es primer
            dividend = window.prompt("Entra un número:");

            //els números 1 i 2 reben un tractament especial
            //al ser l'inici de la sèrie
            if(dividend!=1 && dividend !=2){
                //comencem a dividir entre dos, tots els números
                //són divisibles entre ell mateix i l'1
                divisor=2;
                //comencem a provar amb diferents divisors
                //fins que quocient > divisor
                do{
                    quocient=dividend/divisor;
                    residu=dividend%divisor;
                    //mostrem per pantalla variables
                    console.log("dividend: "+dividend+"");
                    console.log("divisor: "+divisor+"");
                    console.log("quocient: "+quocient+"");
                    console.log("residu: "+residu+"");
                    console.log("---------");
```

```
                                //si el residu es 0, vol dir que el número
                                //és divisible per altres números
                                if (residu==0) esPrimer=false;
                                //passem següent divisor
                                divisor++;
                        //seguim fins que el quocient es menor que el
                        //divisor, llavors seria redundant
                        }while(quocient>divisor);
                }
                //imprimim per pantalla el resultat
                if(esPrimer){
                        console.log(""+dividend+" és primer");
                } else {
                        console.log(""+dividend+" no és primer");
                }
        </script>
    </body>
</html>
```

Eines per programar, provar i depurar el codi

Per a programar i provar codi en JavaScript únicament és necessari un editor de text com *gedit* i qualsevol navegador web, respectivament. A més, els navegadors web també inclouen eines per a depurar el codi que són accessibles a través del teclat amb F12.

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes Nadius. Date

- Propietats:
  - No té propietats
- Mètodes:
  - Get: de lectura
    - getDate(): retorna el número corresponent al dia del mes entre 1 i 31
    - getDay(): retorna el número corresponent al dia de la setmana entre
    - getFullYear(): retorna un número de 4 xifres corresponent a l'any
    - getHours(): retorna l'hora entre 0 i 23
    - getMilliseconds(): retorna els milisegons entre 0 i 999
    - getMinutes(): retorna els minuts entre 0 i 59
    - getMonth(): retorna el número de mes entre 0 -gener- i 11 -desembre-
    - getSeconds(): retorna els segons entre 0 i 59
    - getTime(): retorna el número de milisegons que han passat des de l'1 de gener de 1970
    - getTimezoneOffset(): retorna la diferència en minuts entre l'hora UTC i l'hora local
    - getUTCDate(): retorna el número del dia del més respecte el temps universal coordinat
    - getUTCDay(): retorna el número del dia de la setmana respecte el temps universal coordinat
    - getUTCFullYear(): retorna el número de l'any respecte el temps universal coordinat
    - getUTCHours(): retorna l'hora respecte el temps universal coordinat
    - getUTCMilliseconds(): retorna els milisegons respecte el temps universal coordinat
    - getUTCMinutes(): retorna els minuts respecte el temps universal coordinat
    - getUTCMonth(): retorna el número del mes respecte el temps universal coordinat
    - getUTCSeconds(): retorna els segons respecte el temps universal coordinat
    - getDate(): retorna el número corresponent al dia del mes entre 1 i 31
    - getDay(): retorna el número corresponent al dia de la setmana entre el 0 -diumenge- i el 6 -dissabte-
    - getFullYear(): retorna un número de 4 xifres corresponent a l'any
    - getHours(): retorna l'hora entre 0 i 23
    - getMilliseconds(): retorna els milisegons entre 0 i 999
    - getMinutes(): retorna els minuts entre 0 i 59
    - getMonth(): retorna el número de mes entre 0 -gener- i 11 -desembre-

        - getSeconds(): retorna els segons entre 0 i 59
        - getTime(): retorna el número de milisegons que han passat des de l'1 de gener de 1970
        - getTimezoneOffset(): retorna la diferència en minuts entre l'hora UTC i l'hora local
        - getUTCDate(): retorna el número del dia del més respecte el temps universal coordinat
        - getUTCDay(): retorna el número del dia de la setmana respecte el temps universal coordinat
        - getUTCFullYear(): retorna el número de l'any respecte el temps universal coordinat
        - getUTCHours(): retorna l'hora respecte el temps universal coordinat
        - getUTCMilliseconds(): retorna els milisegons respecte el temps universal coordinat
        - getUTCMinutes(): retorna els minuts respecte el temps universal coordinat
        - getUTCMonth(): retorna el número del mes respecte el temps universal coordinat
        - getUTCSeconds(): retorna els segons respecte el temps universal coordinat
    - Set: mètodes d'escriptura
        - setDate(): estableix el valor del dia del mes
        - setFullYear(): estableix el valor de l'any
        - setHours(): estableix el valor de l'hora
        - setMilliseconds(): estableix el valor dels milisegons
        - setMinutes(): estableix el valor dels minuts
        - setMonth():estableix el valor del mes entre 0 -gener- i 11 -desembre-
        - setSeconds(): estableix el valor dels segons
        - setTime(): estableix una data agregant o restant un número específic de milisegons a la mitjanit de l'1 de gener de 1970
        - setUTCDate(): estableix el valor del dia del mes respecte al temps universal coordinat
        - setUTCFullYear(): estableix el valor de l'any respecte al temps universal coordinat
        - setUTCHours(): estableix el valor de l'hora respecte al temps universal coordinat
        - setUTCMilliseconds(): estableix el valor dels milisegons respecte al temps universal coordinat
        - setUTCMinutes(): estableix el valor dels minuts respecte al temps universal coordinat
        - setUTCMonth(): estableix el valor del mes entre 0 -gener- i 11 -desembre- respecte al temps universal coordinat
        - setUTCSeconds(): estableix el valor dels segons respecte al temps universal coordinat
    - To: mètodes de conversió

- toDateString(): converteix la data de l'objecte en una cadena de caràcters
- toLocaleDateString(): converteix la data de l'objecte en una cadena de caràcters segons els acords locals
- toLocaleTimeString(): retorna la part del temps de l'objecte "Date" en una cadena de caràcters segons els acords locals
- toLocaleString(): converteix un objecte "Date" en una cadena de text, segons les convencions locals
- toTimeString(): converteix la part de temps d'un objecte "Date" en una cadena
- toUTCString(): converteix un objecte "Date" en una cadena segons el temps universal coordinat

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes Nadius. Math

- Propietats/Constants:
  - E: retorna el número d'Euler (aproximadament 2.718)
  - LN2 retorna el logaritme neperià de 2 (aproximadament 0.693)
  - LN10: retorna el logartime neperià de 10 (aproximadament 2.302)
  - LOG2E: retorna el logaritme en base 2 d'E (aproximadament 1.442)
  - LOG10E: retorna el logartime en base 10 d'E (aproximadament 0.434)
  - PI: retorna el número PI (aproximadament 3.14)
  - SQRT1_2: retorna l'arrel quadrada d'1/2 (aproximadament 0.707)
  - SQRT2: retorna l'arrel quadrada de 2 (aproximadament 1.414)
- Mètodes:
  - abs(<number>): retorna el valor absolut
  - acos(<number>): retorna l'arccosinus, en radians. <number> ha de tenir un valor entre [0,1]
  - acsin(<number>): retorna l'arcsinus, en radians. <number> ha de tenir un valor entre [0,1]
  - atan(<number>): retorna l'arctangent, en radians.
  - ceil(<number>): retorna el número enter superior
  - cos(<number, en radians>): retorna el cosinus. <number> ha d'expressar-se en radians.
  - exp(<number>): retorna e^x, passan-li l'exponent
  - floor(<number>): retorna el número enter inferior
  - log(<number>): retorna el logaritme en base "e"
  - max(<number1>[,<number2>, <number3>, ...]): retorna el número màxim dels números entrats com a arguments
  - min(<number1>[,<number2>, <number3>, ...]): retorna el número mínim dels números entrats com a arguments
  - pow(<base>,<exponent>): retorna el resultat d'un número elevat a una potència pasada com a argument base^exponent
  - random(): retorna un número aleatori entre 0 i 1
  - round(<number>): arrodoneix un número al número enter més pròxim

- sin(<number, en radians>): retorna el sinus. <number> ha d'expressar-se en radians
- sqrt(<number>): retorna l'arrel quadrada
- tan(<number>): retorna la tangent. <number> ha d'expressar-se en radians

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes Nadius. Number

- Propietats/Constants:
  - MAX_VALUE: retorna el número més gran disponible a JavaScript
  - MIN_VALUE: retorna el número més petit disponible a JavaScript
  - NaN: Representa el valor Not a Number
  - NEGATIVE_INFINITY: Representa l'infinit negatiu
  - POSITIVE_INFINITY: Representa l'infinit positiu
- Mètodes:
  - toExponential(): converteix un número en notació exponencial
  - toFixed(): formateja el número amb la quantitat de dígits decimals que es passin com a paràmetre
  - toPrecision(): fomateja el número amb la longitud que es passi com a paràmetre

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes Nadius. String

- Propietats
  - length: longitud de la cadena de text
- Mètodes:
  - Cerca:
    - charAt(<position>): retorna el caràcter de la posició "position"
    - charCodeAt(<position>): retorna el codi en UTF-16 del caràcter de la posició <position>
    - indexOf(<string>, [<fromIndex>]): retorna la primera posició de la variable <string> a la cadena. Comença a buscar des de la posició <fromIndex>. Si no s'especifica <fromIndex> comença a buscar des de la posició 0. Retorna -1 si no es troba el valor
    - lastIndexOf(<string>, [<fromIndex>]): retorna l'última posició de la variable <string>. Comença a buscar des de la posició <fromIndex>. Si no s'especifica <fromIndex> comença a buscar des de l'última posició de l'string cap enrere. Retorna -1 si no troba el valor.
    - search(<string>|<regEx>): busca <string> o <regEx> -una expressió regular- en una cadena i retorna la posició de la coincidència. Si no la troba, retorna -1.
    - includes(<string>): retorna true si la cadena conté la variable <string>
    - startsWith(<string>): retorna true si la cadena comença amb la variable <string>

        - endsWith(<string>): retorna true si la cadena acaba amb la variable <string>
        - replace(<string1|regEx>, <string2>): busca coincidències de <string1|regEx> en una cadena i si existeixen, les reemplaça per <string2>. Si com a primer paràmetre passem un string, només reemplaça la primera coincidència. Si passem una expressió regular, reemplaça totes les coincidències.
    - Comparació:
        - //localeCompare(str2): compara cadascuna de les lletres que formen les dues cadenes i, retorna -1 si la primera cadena és més gran, 0 si son iguals, i 1 si la segona cadena és més gran
    - Concatenació:
        - concat(<string1>, [<string2>, <string3>,...]): concatena dues o més cadenes a la cadena que crida el mètode i retorna la nova amb la concatenació
    - Extracció:
        - slice(<position1> [, <position2>]): extreu una part de la cadena en base als paràmetres de la cadena indicats com a inici -<position1>- i final -<position2>-. Cal tenir present que <position1> i <position2> poden ser negatius, però <position2> sempre ha de ser més gran que <position1>. Si únicament s'utilitza un valor, retorna la cadena des d'aquella posició al final.
        - substring(<position1> [, <position2>]): fa el mateix que l'slice, però no cal tenir en compte que <position2> ha de ser més gran que <position1>, ho corregeix automàticament.
        - split(<separador>, [<numVegades>]): divideix una cadena en base a un separador que s'ha passat com a paràmetre i emmagatzema les diferents parts a un array. <numVegades> és opcional i indica el número de vegades que es vol fer l'operació.
        - trim(): extreu els espais en blanc tant de l'inici com del final de la cadena.
    - Altres:
        - toLowerCase(): converteix una cadena a minúscules
        - toUpperCase(): converteix una cadena a majúscules

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. Navigator

- Propietats:
  - onLine: retorna true si el navegador té connexió a Internet
  - language: retorna un string indicant l'idioma preferent de l'usuari, generalment l'idioma de la interfície d'usuari del navegador. Retorna <null> quan és indefinit
  - cookieEnabled: retorna <true> si les galetes estan activades al navegador
  - platform: retorna la plataforma sobre la qual està corrent el navegador
  - appName: regorna el tipus del navegador, però ara és un valor comú a la majoria de navegadors.
  - appVersion: retorna la versió del navegador.
  - userAgent: retorna informació sobre el tipus de navegador, tot i que pot ser modificada per l'usuari (veure general.useragent override in about:config). appName i appversion teòricament retornen el tipus de navegador, però a la pràctica no podem basar-nos en elles per trobar aquesta informació, ja que és comuna a la majoria de navegadors.
- Mètodes:
  - javaEnabled(): retorna <true> si Java està activat al navegador

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. Screen

- Propietats:
  - width: amplada total de la pantalla en píxels
  - height: altura total de la pantalla en píxels
  - availWidth: amplada de la pantalla disponible per a finestres
  - availHeight: altura total de la pantalla disponible per a finestres
- Mètodes:
  - L'objecte screen no disposa de mètodes

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. History

- Propietats:
  - length: número d'URLs a l'historial de la finestra
- Mètodes:
  - back(): carrega la URL anterior a l'historial
  - forward(): carrega la URL següent a l'historial
  - go(<número>|<url>): si passem una <URL>, va a la URL especificada. Si li passem un número positiu, va endavant <número> de pàgines; i si li passem un número negatiu, va endarrera <número> de pàgines

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. Location

- Propietats:
  - href: URL de la finestra
  - hostname: nom del host de la pàgina
  - pathname: nom del pathname de la pàgina
  - protocol: protocol de la pàgina
  - hash: hash o ancla de la pàgina. Permet indicar posicions de la pàgina.
  - host: nom del hostname i el port
  - origin: nom del protocol, hostname i el port
  - search: querystring de la pàgina (el que va darrere de index.html?)
- Mètodes:
  - assign(<url>): assigna un nou document a la pàgina
  - reload(): torna a carregar la pàgina
  - replace(<url>): assigna un nou document a la pàgina, però esborrant l'historial de la finestra

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. Window

- Propietats:
  - Nom:
    - name: Name de la finestra. És diferent al Name de la variable i al Name de la pestanya de la finestra.
  - Mides o distàncies:
    - innerHeight: altura de la part visible de la finestra, incloent l'scroll vertical
    - innerWidth: amplada de la part visible de la finestra, incloent l'scroll hortizontal
    - outerHeight: altura exterior de la finestra
    - outerWidth: amplada exterior de la finestra
    - scrollX: retorna el número de píxels recorreguts respecte l'eix horitzonal
    - scrollY: retorna el número de píxels recorreguts respecte l'eix vertical
  - Estats:
    - closed: indica si la finestra està tancada. Retorna <true> si la finestra està tancada, <false> si està oberta.
  - Referències, útils per treballar amb 'iframes'.
    - self: retorna una *referència* a l'iframe/finestra actual
    - parent: retorna una *referencia* a l'iframe/finestra que el conté
    - top: retorna una *referencia* a la l'iframe/finestra arrel
  - Iframes:
    - length: retorna el número de frames/iframes de la finestra
    - frames: retorna un llistat d'objectes de tipus frame/iframe
  - Altres objectes del navegador:

        - document: retorna una *referència* a l'objecte Document de la finestra
        - navigator: retorna una *referència* a l'objecte Navigator de la finestra
        - screen: retorna una *referència* a l'objecte Screen de la finestra
        - history: retorna una *referència* a l'objecte History de la finestra
        - location: retorna una *referència* a l'objecte Location de la finestra
- Mètodes:
    - Bàsics:
        - open(<url|rutaFitxer>,<name>,<propietats>): obre una nova finestra
            - <url|rutaFitxer>:
                - URL: ha de començar per http…
                - rutaFitxer: si no comença per http…
            - <name>:
                - _blank: la finestra es carrega a una nova finestra
                - _self: la finestra reemplaça la finestra on ens trobem
                - _parent: la finestra es carrega al frame 'parent' (útil per treballar amb un nivell de frames)
                - _top: la finestra substitueix tots els frames que estan carregats a la pàgina (útil per treballar amb més d'un nivell de frames)
                - name: serveix per posar un nom a la pàgina, però cal tenir en compte que el nom és diferent al de la variable Window i al window.title
            - <propietats>: si s'utilitza aquest paràmetre, aquelles propietats que no estan llistades son desactivades, excepte "titlebar" i "close" que per defecte estan activades. Més detall a: https://www.w3schools.com/jsref/met_win_open.asp */
        - close(): serveix per tancar la finestra
        - print() - imprimeix la pàgina actual
        - stop() - atura la carrega de la finestra. pot ser útil si la càrrega de la finestra triga molt
        - focus() - passa la finestra a davant de totes les altres
        - blur() - treu la finestra de la primera posició de la pantalla
    - Diàleg:
        - alert(<string>): mostra el missatge <string> a una finestra
        - confirm(<string>): mostra el missatge <string> a una finestra. Retorna <true> si l'usuari prem OK, o <fals> si l'usuari cancel·la o tanca la finestra
        - prompt(<string>): mostra el missatge <string> a una finestra. A més, l'usuari pot escriure un missatge
    - Desplaçament:
        - moveTo(<xPixels>, <yPixels>): mou la finestra a la posició <xPixels> i <yPixels>, prenent com a referència la cantonada superior esquerra de la pantalla
        - moveBy(<xPixels>, <yPixels>): mou la finestra <xPixels> i <yPixels>, prenent com a referència la posició actual de la finestra. <xPixels> i <yPixels> poden tenir valors positius i negatius

        - scrollBy(<xPixels>, <yPixels>): desplaça el contingut de la finestra una quantitat específica de pixels. S'ha d'executar a la mateixa finestra
        - //scrollTo(<xPixel>, <yPixel>): desplaça el contingut de la finestra fins a <xPixel>, <yPixel>. S'ha d'executar a la mateixa finestra
    - Mida/Desplaçament:
        - resizeTo(<xPixels>, <yPixels>): redimensiona la mida externa de la finestra a <xPixels> d'amplada i <yPixels> d'altura. Utilitzar-ho amb finestres de tipus pop-up, a finestres pares, acostuma a estar desactivat als navegadors.
        - resizeBy(<xPixels>, <yPixels>): redimensiona la mida externa de la finestra <xPixels> d'amplada i <yPixels> d'altura. Utilitzar-ho amb finestres de tipus pop-up, a finestres pares, acostuma a estar desactivat als navegadors.
    - Esdeveniments relacionats amb temps:
        - setTimeout(<function>, <timeout>): executa la funció "function" passats "timeout" milisegons
        - setInterval(<function>, <timeout>): executa la funció "function" cada "timeout" milisegons. El nom de la funció, sense "()"
        - clearTimeout(<variable>): atura l'execució de setTimeout quan se li passa la referència a través d'una variable
          clearInterval(<variable>): atura l'execució de setInterval quan se li passa la referència a través d'una variable

Identificació i aplicació dels objectes predefinits del llenguatge. Objectes associats al Navegador. Document

- Propietats:
    - cookie: serveix per:
        - crear cookies
        - retornar les cookies separades per comes
    - domain: retorna o desa el valor del domini del document
    - embeds: retorna un llistat d'objectes encastats
    - forms: retorna el llistat dels elements de tipus formulari del document
    - images: retorna una col·lecció de les imatges del codi HTML
    - lastModified: retorna la data i hora en que el document va ser modificat per última vegada
    - links: per a cada element <a> o <area> del codi HTML, retorna l'atribut HREF
    - plugins: retorna una col·lecció dels tags <embed> al document
    - referrer: retorna la url de la pàgina que ha linkat a la pàgina actual
    - title: conté el títol del document. Es pot llegir, o escriure.
    - URL: propietat de lectura que retorna la URL de la finestra
- Mètodes:
    - document.open(): obre el document. Quan un document que està tancat s'obre, es perden totes les dades del document.

- document.write(): escriu al document de la finestra. S'acostuma a utilitza per a testejar. Quan cridem document.write() a un document ja tancat, l'obre automàticament.
- document.writeln(): com el document.write(), però insereix una línia després de l'execució amb el caràcter '\n'.
- document.close(): tanca el document de la finestra

Identificació i aplicació dels objectes predefinits del llenguatge. Marcs: Frames i iFrames

Els marcs s'utilitzen per dividir la finestra d'una aplicació web en 2 o més parts independents. JavaScript possibilita que s'interactuï entre aquests sectors independents, que s'anomenen *frames*, o en català, marcs. Tot i això, no s'acostumen a utilitzar per temes d'usabilitat.

- Frames: tag HTML obsolet des d'HTML 5.
- iFrames: des d'HTML 5 s'utilitza l'etiqueta <iframe>. Gràcies a Javascript podem interactuar amb iframes carregant noves pàgines o variant el seu contingut.
  Per a realitzar comunicacions entre marcs, és necessari utilitzar la propietat **contentWindow**, que retorna la referència a l'objecte 'window' d'un element de tipus iframe.
  - Propietats:
    - width, height: amplada i altura de l'iframe. També els podem declarar amb el CSS.
    - name: el nom del <iframe>.
    - src: URL del document que incrustarem a l'<iframe>.
    - srcdoc: el contingut HTML que incrustarem a l'<iframe>

Identificació i aplicació dels objectes predefinits del llenguatge. Galetes

Quan un servidor envia una pàgina a un usuari, talla la connexió i s'oblida de qui l'ha demanada. Si el mateix usuari li torna a demanar una altra pàgina, el servidor no té manera d'identificar qui li va demanar. Per a resoldre aquest problema, es van inventar les 'cookies'.

Les cookies són dades associades al navegador per a emmagatzemar informació relativa a l'usuari. Existeixen per a que el servidor tingui present certes característiques sobre qui li solicita la pàgina. A més, les cookies són independents de la instància de navegador utilitzada. És a dir, les cookies emmagatzemades a un navegador no es comparteixen amb els altres navegadors. Les cookies de Firefox no tenen perquè ser les mateixes que les d'Opera o Chrome, perquè són 'bases de dades/cookies' diferents.

Les cookies poden tenir data de caducitat. Si no l'especifiquem, les cookies desapareixen quan es tanca el navegador. També podem determinar on es guarden les cookies, que acostuma a ser al navegador. Tot i això, també es poden desar a llocs diferents.

- Creació:
  - "<nomCookie>=<valorCookie>"
- Propietats:
  - expires=<data de caducitat-formatUTC->, si s'especifica, determina la vigència de la cookie encara que es tanqui el navegador.
  - path=<cookiePath>, defineix on té validesa la cookie.

Identificació i aplicació dels objectes predefinits del llenguatge. Local Storage

Les galetes s'inclouen en cada *request* HTTP, encara que no les necessitem, i no estan encriptades (a no ser que utilitzem SSL en el servidor). Això representa una disminució de la velocitat de transmissió. A més, les galetes estan limitades a uns 4 KB d'informació.

Seria bo que en els clients web disposéssim d'un espai d'emmagatzematge que fos persistent, i que no es transmetés al servidor.

L'emmagatzematge DOM (*DOM Storage*) és el nom donat a un conjunt de característiques relacionades amb l'emmagatzematge introduïdes a HTML5 i ara detallades en l'especificació *W3C Web Storage*.Dins del *DOM Storage* ens interessa principalment l'objecte *localStorage* per accedir a l'emmagatzematge persistent, i l'objecte *sessionStorage* per guardar un espai d'emmagatzematge disponible durant la sessió de navegació:

- *localStorage*: guarda informació que quedarà emmagatzemada un temps indefinit, sense importar que el navegador es tanqui.

- *sessionStorage*: emmagatzema les dades d'una sessió, que s'eliminen quan es tanca.

Tant en el cas de *localStorage* com *sessionStorage* utilitzem els mètodes *setItem()* i *getItem()* per escriure i llegir els parells nom-valor per emmagatzemar la informació.