UT4. Introducció a Javascript

UT4. Introducció a Javascript

4.1 Introducció
4.2 Estructura
4.3 Variables i valors
4.4 Tipus de dades i operadors

4.1 Introducció

En l'inici de l'era d'Internet, les pàgines web eren bàsicament estàtiques, mostraven un contingut fix sense possibilitat d'interacció amb l'usuari.

Però quan va aparèixer "La Web 2.0" es va incrementar l'interacció i usabilitat, millorant l'experiència de l'usuari en la navegació.

És aquí on el llenguatge JavaScript té un rol important. La programació en el costat del client codificada dins les pàgines web, és la responsable de fer pàgines web atractives tal com les coneixem avui dia.

4.1 Introducció

Un poc d'història

És important dir que el llenguatge ha evolucionat per dos costats:

- Per el propi llenguatge: Ha anat incorporant operadors, estructures de control, regles de sintaxi, etc.
- Per els navegadors: Han anant incorporant noves instruccions (per manipular nous elements).

Els autors de JavaScript, van proposar el 1997 a l'European Computer Manufacturers Association (ECMA), que el seu llenguatge es definís com a estàndard. Així és com va néixer l'ECMAScript, que és l'estàndard de definició del llenguatge Javascript. Al llarg del temps, s'han anant publicant distintes versions d'ECMAScript. En els inicis d'Internet hi havia bastants problemes de compatibilitat entre navegadors (Internet Explorer, Netscape Navigator, Opera) a diferents nivells, i això va fer prendre consciència de la importància en l'adopció d'estàndards.

A partir del 2012, tots el navegadors actuals suporten completament ECMAScript 5.1. Des de Juny 17, 2015, ECMA International, va publicar la versió número 6, que oficialment s'anomena ECMAScript 2015, i que inicialment se va dir ECMAScript 6 o ES6.

4.1 Introducció

Un poc d'història

Per tal d'avançar en la compatibilitat entre navegadors web, no només és important que implementin el motor de JavaScript segons els estàndards ECMAScript, sinó que els diferents elements que hi ha en una web (botons, caixes de text, enllaços...) es comportin de la mateixa manera i responguin als mateixos events (esdeveniments).

És per això que el W3C va definir el **Document Object Model (DOM)**, o Model d'Objectes del Document, que defineix un estàndard d'objectes per representar documents HTML i/o XML.

4.1 Introducció

Què és JavaScript

JavaScript és un llenguatge de guions (script, en anglès). És conegut sobretot pel seu ús en pàgines web, però també es pot utilitzar per realitzar tasques de programació i administració que no tenen res a veure amb la web.

Si ens centram amb JavaScript com a llenguatge que s'utilitza amb pàgines web, podem dir que millora una pàgina HTML, afegint interacció de l'usuari, animació, ajudes a la navegació, etc.

El nom de JavaScript no deriva del llenguatge de programació Java, tot i que tots dos comparteixen una sintaxi similar. Semànticament, JavaScript és més pròxim als llenguatges Self i ActionScript (basat també en l'ECMAScript). El nom JavaScript és una marca registrada per Oracle Corporation.

- Llenguatges **interpretats** o de guions: Els llenguatges de programació es divideixen, quant a la manera d'executar-se, entre els interpretats i els compilats. Alguns llenguatges interpretats s'anomenen també llenguatges de guions (scripts en anglès). Els llenguatges interpretats s'executen mitjançant un intèrpret, que processa les ordres que inclou el programa una a una. Els llenguatges compilats necessiten del compilador com a pas previ a l'execució. **JavaScript és un llenguatge interpretat**.

Javascript és un llenguatge single-thread, és a dir, té només un fil d'execució, per la qual cosa les tasques es van executant seqüencialment.

4.1 Introducció

Avantatges de JavaScript en les pàgines web

- Permet una gran quantitat d'efectes dins les pàgines. Entre d'altres: finestres emergents (Pop-up), desplaçaments (scrolls), transicions d'imatges, etc.

- Afegeix interactivitat amb l'usuari.

- Proporciona integració amb altres connectors o extensions(plugins): no proporciona només accés als objectes HTML, també proporciona accés a objectes específics del navegador com Adobe Acrobat, Media Player, etc.

- Permet validació dels formularis en el costat del client. Una validació inicial dels formularis és possible per eliminar simples errors, com per exemple: com assegurar-se de què el format de data, DNI, correu electrònic o telèfon són correctes. Com a resultat, l'usuari té una resposta més ràpida que si el control es fes en el costat del servidor.

- Permet accedir a informació del sistema.

4.1 Introducció

Desavantatges de JavaScript en les pàgines web

- **La seguretat** és el principal problema a JavaScript. Els fragments de codi JavaScript que s'afegeixen a les pàgines web es descarreguen en els navegadors i s'executen en el costat del client, permetent així la possibilitat que un codi maliciós es pugui executar en la màquina client i així explotar alguna vulnerabilitat de seguretat coneguda en les aplicacions, navegadors o fins i tot del sistema operatiu. Per prevenir això, existeixen estàndards de seguretat que restringeixen l'execució de codi, en els navegadors. Aquestes restriccions són per exemple, la de deshabilitar l'accés a l'escriptura o lectura de fitxers.

- Tendeix a introduir una **quantitat enorme de fragments de codi** en els nostres llocs web. Això es resol fàcilment emmagatzemant el codi JavaScript en fitxers externs amb extensió JS. Així la pàgina web queda molt més neta i llegible. Actualment quasi sempre es separa totalment la part del contingut HTML, la part de funcionalitat JavaScript, i la part de disseny i maquetació (CSS), així tindrem tres perfils d'usuaris diferents (el que genera continguts HTML, el programador web, i el dissenyador), que podran treballar en el mateix projecte però en arxius diferents.

4.1 Introducció

Desavantatges de JavaScript en les pàgines web

- **La seguretat** és el principal problema a JavaScript. Els fragments de codi JavaScript que s'afegeixen a les pàgines web es descarreguen en els navegadors i s'executen en el costat del client, permetent així la possibilitat que un codi maliciós es pugui executar en la màquina client i així explotar alguna vulnerabilitat de seguretat coneguda en les aplicacions, navegadors o fins i tot del sistema operatiu. Per prevenir això, existeixen estàndards de seguretat que restringeixen l'execució de codi, en els navegadors. Aquestes restriccions són per exemple, la de deshabilitar l'accés a l'escriptura o lectura de fitxers.

- Tendeix a introduir una **quantitat enorme de fragments de codi** en els nostres llocs web. Això es resol fàcilment emmagatzemant el codi JavaScript en fitxers externs amb extensió JS. Així la pàgina web queda molt més neta i llegible. Actualment quasi sempre es separa totalment la part del contingut HTML, la part de funcionalitat JavaScript, i la part de disseny i maquetació (CSS), així tindrem tres perfils d'usuaris diferents (el que genera continguts HTML, el programador web, i el dissenyador), que podran treballar en el mateix projecte però en arxius diferents.

4.2 Estructura

- És important saber que JavaScript **distingeix entre majúscules i minúscules**, és a dir, és **case-sensitive**.
- Existeixen dues maneres de separar instruccions. La primera és a través del caràcter **punt i coma** (;) i la segona és a través d'un **salt de línia**. Per tant el punt i coma al final de la instrucció no és obligatori si la següent comença a una nova línia. Si estan a la mateixa línia serà obligatori posar-ho. De totes maneres, es recomanable fer ús d'ell sempre.
- No té en compte els espais en blanc ni les noves línies.
- Utilitza el conjunt de caràcters Unicode.

4.2 Estructura

Comentaris a JavaScript

Quan el comentari ocupa una sola línia s'escriurà a partir de dues barres (//), i quan el comentari s'escriu en més d'una línia anirà entre / * * /.

```
<script>
   // Aquest és un comentari d'una única línia
   alert ("Exemple de comentaris en línia");//Aquí hi pot haver-hi un altre comentari en una línia
</script>
```

Nota: Com veurem més endavant la funció alert () és una declaració simple, que mostra un quadre de diàleg que conté un missatge. El missatge es col·loca entre cometes dins del parèntesi de la funció

```
<script>
      alert ("Exemple de comentari multi-línia");
      /* Una línia de comentari
      Una altra línia de comentari */
</script>
```

4.2 Estructura

On col·locam el codi JavaScript

El codi Javascript es defineix a través de l'element <script>. Aquesta té un atribut de tipus, que s'utilitza per indicar el tipus de llenguatge que utilitzarem (en aquest cas serà Javascript). Amb HTML 4 i XHTML 1.x, el tipus d'atribut és obligatori, en canvi, amb HTML5, no ho és.

- **Javascript dins la pàgina**: Para situar el codi JavaScript directament a la pàgina web, es col·loca dins de l'element <script>, al <head> o al <body>.
- **Javascript extern**: Es convenient en molts de casos, escriure el codi JavaScript en un arxiu extern amb l'extensió "js". Aquest arxiu es crida des de la pàgina web amb la instrucció <script src="./arxiujavascript.js"></script>, que pot anar al <head> o al <body>.

4.2 Estructura

Javascript dins la pàgina

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Dins de la pàgina</title>
       <script>
           alert ('Codi dins el head'); //La instrucció alert() mostra un missatge per pantalla.
       </script>
    </head>
    <body>
       <script>
           alert ('Codi dins de la pàgina'); //La instrucció alert() mostra un missatge per pantalla.
       </script>
   </body>
</html>
```

4.2 Estructura

Javascript extern

pagina.html

```
<!DOCTYPE html>
<html>
   <head>
        <meta charset="utf-8">
       <title>JavaScript extern</title>
       <script src="exemple1.js"></script>
   </head>
   <body>
       <script src="exemple2.js"></script>
   </body>
</html>
```

exemple1.js

```
alert ('Codi dins el head');
```

exemple2.js

```
alert ('Codi dins de la pàgina');
```

4.2 Estructura

Visualització o sortida de les dades en JavaScript

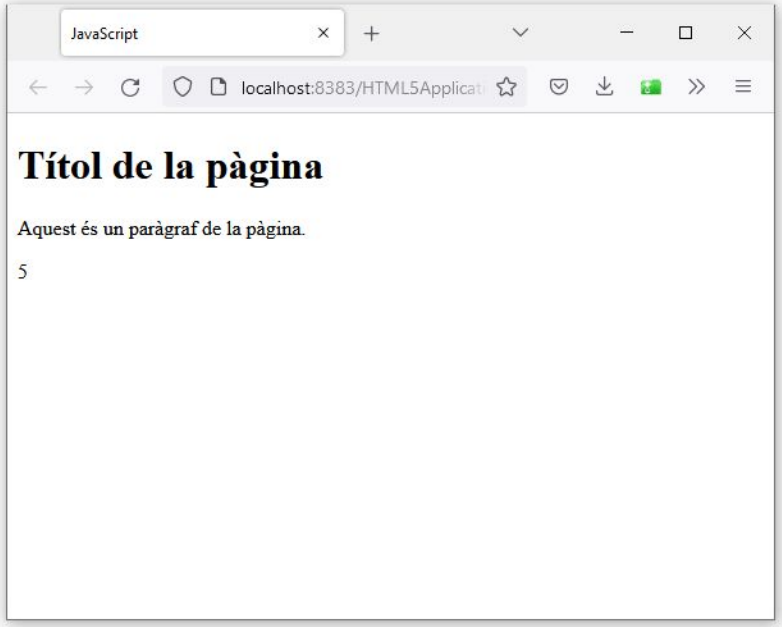
Es poden mostrar les dades de diferents formes:

- Escrivint a la sortida d'HTML utilitzant el mètode d'accés al DOM (que veurem més endavant): "document.write()". El DOM ens permet accedir al codi HTML i modificar-lo.
- Creant un quadre o finestra emergent utilitzant: window.alert(), o simplement alert(). Aquest s'utilitza per interaccionar amb el navegador. És el mateix utilitzar "window.alert()" o directament "alert(), ja que window fa referència a l'objecte global.
- Escrivint directament a la consola del navegador utilitzant: console.log().

4.2 Estructura

Ús de document.write()

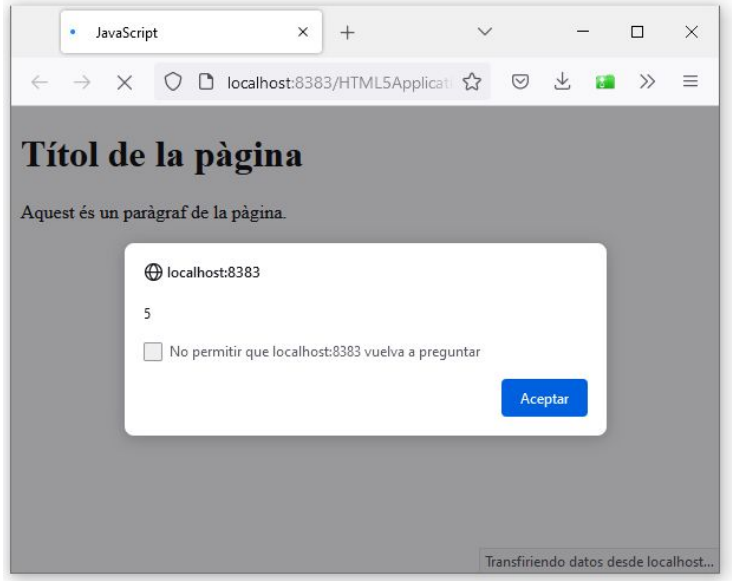
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JavaScript</title>
    </head>
    <body>
        <h1>Títol de la pàgina</h1>
        <p>Aquest és un paràgraf de la pàgina.</p>
        <script>
            document.write(5);
        </script>
    </body>
</html>
```



4.2 Estructura

Ús de window.alert()

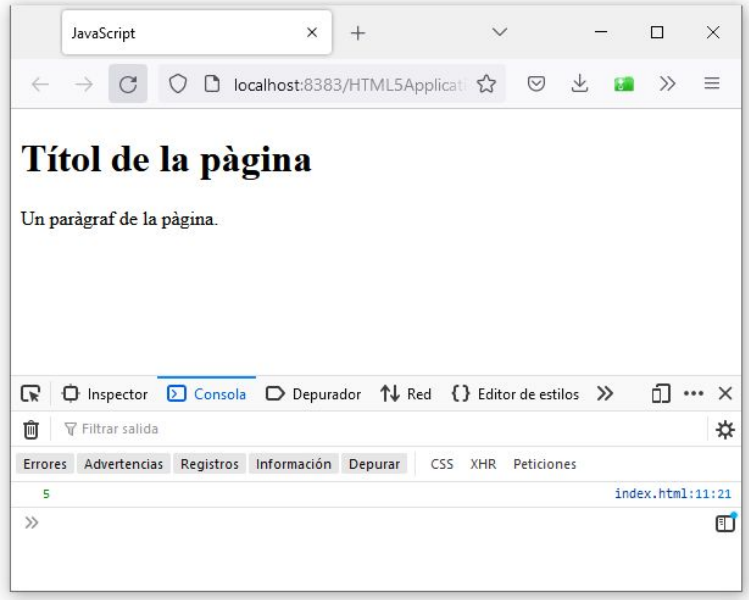
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JavaScript</title>
    </head>
    <body>
        <h1>Títol de la pàgina</h1>
        <p>Aquest és un paràgraf de la pàgina.</p>
        <script>
            window.alert(5); //o també alert().
        </script>
    </body>
</html>
```



4.2 Estructura

Ús de console.log()

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JavaScript</title>
    </head>
    <body>
        <h1>Títol de la pàgina</h1>
        <p>Un paràgraf de la pàgina.</p>
        <script>
            console.log(5);
        </script>
    </body>
</html>
```



4.2 Estructura

Confirmació de la sortida de dades

- **Ús de window.confirm()**

  Aquest mètode ens mostra una finestra de diàleg amb un missatge i dos botons: Acceptar i Cancelar, de manera que permet que l'usuari accepti continuar o cancelar l'acció.

Sintaxi:

```
resultat = window.confirm(missatge);
```

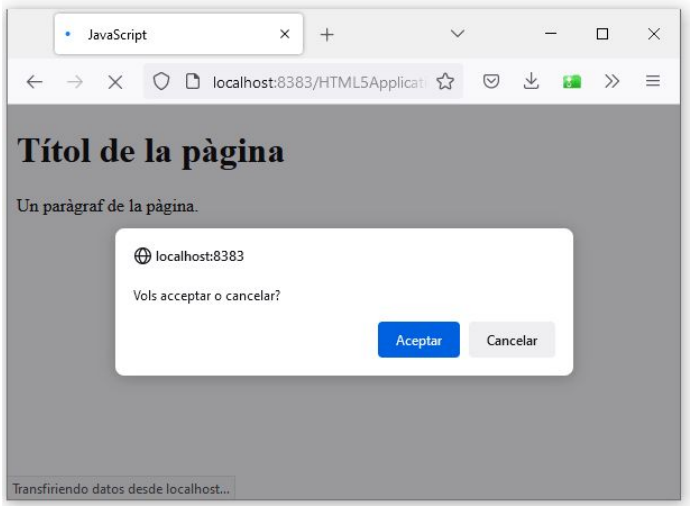
On:

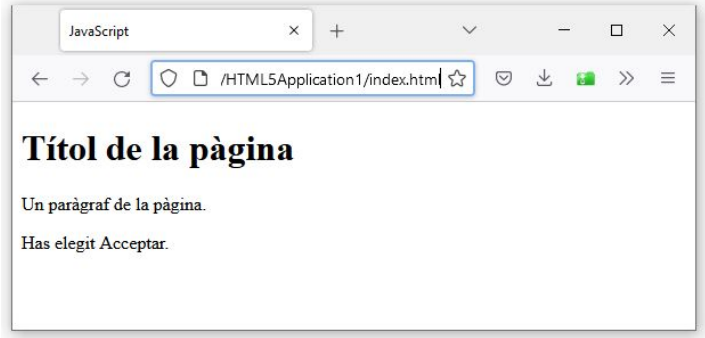
missatge és la cadena que es mostra opcionalment en el quadre de diàleg.
resultat: És la variable on es guardarà el valor resultant de triar Acceptar o Cancelar (true o false).

4.2 Estructura

Ús de window.confirm()

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JavaScript</title>
    </head>
    <body>
        <h1>Títol de la pàgina</h1>
        <p>Un paràgraf de la pàgina.</p>
        <script>
            resultat = window.confirm("Vols acceptar o cancelar?");
            if(resultat) {
                document.write("Has elegit Acceptar.");
            }
            else {
                document.write("Has elegit Cancelar.");
            }
        </script>
    </body>
</html>
```





4.2 Estructura

Recollir dades amb prompt

Ens mostra un quadre de diàleg que ens permet mostrar el missatge amb un finestra emergent, i a la vegada recollir un valor (de tipus string).

Sintaxi:

```
var info= prompt("text","valor/text per defecte");
```

On:

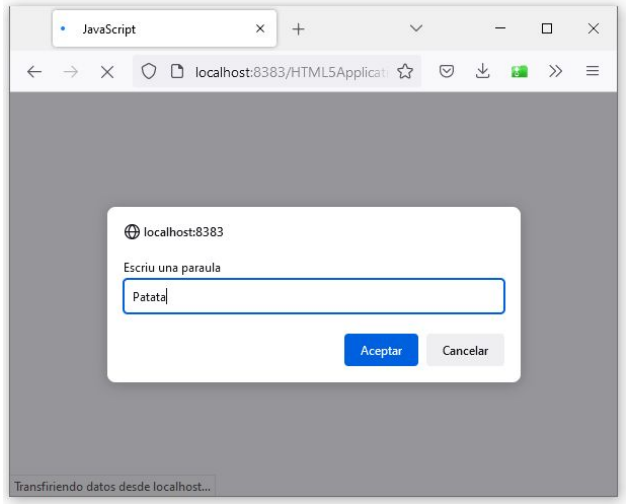
Text: És el missatge que es mostrarà per pantalla demanant una informació o un valor.

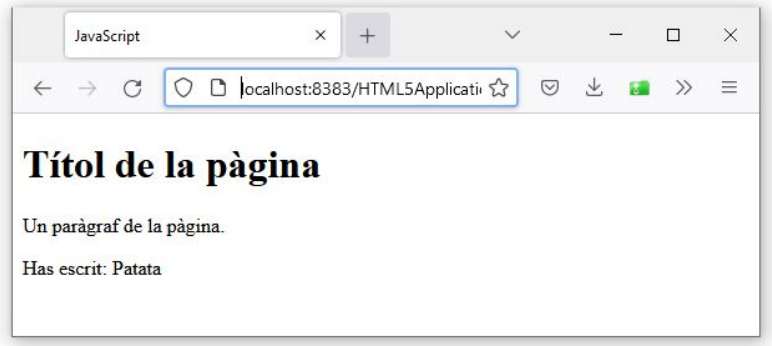
valor/text per defecte (Opcional): Per escriure un valor o una informació per defecte per si l'usuari no n'escriu.

4.2 Estructura

Recollir dades amb prompt

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>JavaScript</title>
    </head>
    <body>
        <h1>Títol de la pàgina</h1>
        <p>Un paràgraf de la pàgina.</p>
        <script>
            var info = prompt("Escriu una paraula");
            document.write("Has escrit: " + info);
        </script>
    </body>
</html>
```





4.3 Variables i valors

Identificadors

Els identificadors són noms i han de ser únics. S'utilitzen identificadors per anomenar variables (paraules clau, funcions i etiquetes). Les normes per a noms legals o permesos, són el mateix en la majoria de llenguatges de programació:

- El primer caràcter ha de ser una lletra, un guió baix (_) o un signe de dòlar ($).
- Els caràcters posteriors poden ser lletres, dígits, guions baixos o signes de dòlar.
- No es permeten números com a primer caràcter.

4.3 Variables i valors

Variables

La creació d'una variable a Javascript s'anomena "declarar" una variable. Declarar una variable vol dir, que es reserva espai d'emmagatzemament en memòria.

Les variables han de ser identificadors únics i han de complir les següents normes:

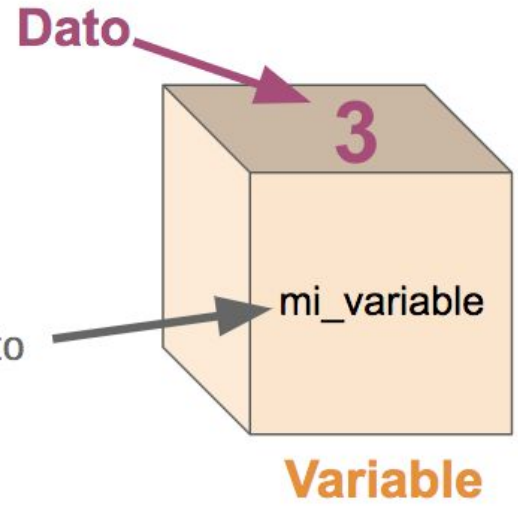
- Només poden contenir lletres, dígits, signe de subratllat (_) i el signe de dòlar ($).
- Han de començar amb una lletra, o bé amb el signe de subratllat (_) o el signe de dòlar ($).
- Recordar que es distingeix entre majúscules i minúscules (la variable x és diferent de la variable X).
- No es poden utilitzar les paraules reservades de JavaScript (while, for, next…).

4.3 Variables i valors

Variables

Variables JS

Cada variable tiene un nombre, de modo que podamos acceder a ese dato siempre que necesitemos.



4.3 Variables i valors

Cóm declarar variables: Abans d'ES6, la única forma de crear variables en JavaScript era amb la paraula "var". Però a partir d'aquest estàndard hi ha tres tipus de declaracions en JavaScript:

- **var:** Declara variables en àmbit global. Les variables d'àmbit global es poden emprar en qualsevol lloc de l'script.
- **let:** Declara variables locals en àmbit local. Les variables d'àmbit local només tenen validesa dins de l'àmbit d'una funció.
- **const:** Declara constants de només lectura.

4.3 Variables i valors

Àmbit de les variables: L'àmbit de les variables és el lloc on estan disponibles.

Les normes que s'apliquen per definir el tipus de variable (àmbit global o àmbit local), es resumeixen en:

- Qualsevol variable declarada o no declarada a l'arrel d'un script és sempre d'àmbit **global**.
- Qualsevol variable declarada dins d'una funció és d'àmbit **local**.
- Una variable no declarada dins d'una funció adquireix àmbit **global**.

Les variables de JavaScript poden contenir números i també valors de text:

- Els valors de text s'anomenen cadenes de text.
- Les cadenes s'escriuen dins de cometes dobles o simples.
- Els números s'escriuen sense cometes

4.3 Variables i valors

- **Variable amb "var":** JavaScript és lo que es coneix com a un llenguatge dinàmic, això vol dir que les variables es creen mentre el programa s'executa. No és necessari assignar-lis valors en el moment de la declaració, encara que es pot fer.

Exemples:

```
var gran = true; // booleà
var edat = 21; // Sencer
var preu = 169.68; // Real
var nom = 'Joan'; // String
```

4.3 Variables i valors

- **Variables amb "let" :** Les variables de tipus "let", tenen àmbit de bloc, això vol dir que només poden ser utilitzades a partir de la seva definició en el bloc de codi. Moltes vegades és millor "let", ja que pot evitar situacions en les que el codi pot ser confús. Un bloc és un fragment de codi delimitat per {}**.**

- **Variables amb "const":** Aquest tipus de variable si requereixen que tenguin un valor al ser declarades, i a més aquests valors no poden ser re-assignats.

4.3 Variables i valors

Valors
La sintaxi de JavaScript defineix dos tipus de valors: valors fixos i valors variables:

- Els valors fixos s'anomenen **literals**. S'utilitzen per a representar valors en Javascript. Com el seu nom indica són literalment proporcionats pel programador en el codi.
- Els valors variables són les pròpies variables.

Expressions

- Una expressió és una combinació de valors, variables i operadors, que calcula amb un valor.
- El càlcul s'anomena avaluació.
- Els valors poden ser de diversos tipus, com números i cadenes.

4.3 Variables i valors

Paraules reservades de Javascript

- break
- case, catch, continue
- default, delete, do
- else
- finally, for, function
- if, in, instanceof
- new
- return
- switch
- this, throw, try, typeof
- var, void, let
- while, with

4.4 Tipus de dades i operadors

Tipus de dades

Els tipus de dades en JavaScript són una mica diferents en comparació amb altres llenguatges de programació com a C o JAVA. JavaScript és un **llenguatge feblement tipat**, això vol dir que no és necessari definir el tipus de dada. Però això no vol dir que no tengui un tipus de dada, ja que aquest es defineix temps d'execució per JavaScript.

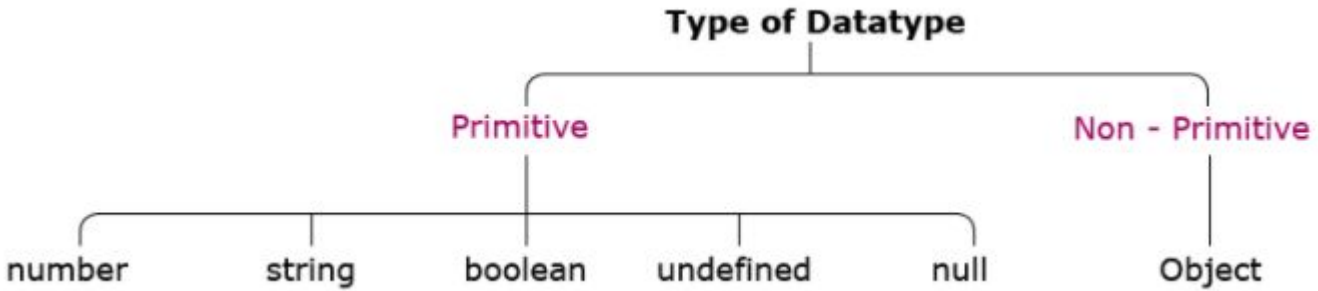
Aquest comportament i especificació dels tipus de dades aplica per a qualsevol lloc on s'executi JavaScript.

Es diu que JavaScript té un sistema de tipus de dades dinàmic, això és perquè les variables que es defineixen poden rebre qualsevol altre tipus de dada en qualsevol moment, i anar canviant de tipus segons el valor que guarden. El que realment ens interessa és el valor i que podem fer amb aqueix valor, no tant el tipus.

4.4 Tipus de dades i operadors

Tipus de dades

Com podem veure a la taula, tenim dos grans tipus de dades: el tipus primitius i els tipus objecte:



4.4 Tipus de dades i operadors

Tipus primitius

- Aquest tipus de dades són immutables
- Els seus valors contenen només una dada.
- Després d'assignar a una variable un valor primitiu, si es vol canviar aquest valor, és necessari reassignar-n'hi un de nou.

4.4 Tipus de dades i operadors

- **String:** És una cadena de caràcters. Es defineix entre cometes simples o dobles.
  - `var text = 'Això és un String'`

- **Boolean:** Representa una entitat lògica i pot tenir dos valors: true i false. Molt útil per a prendre decisions en els programes.
  - `const Estudiant = false;`

- **Number:** Aquest és el tipus de dada numèrica junt al recent tipus definit en 2020 anomenat BigInt. La forma de definir aquest tipus de dada és molt senzill, basta escriure el número (independentment de si és sencer, flotant, decimal, etc.).
  - `const edat = 33;`
  - `const descompte = 0.30;`

  Existeix un número especial anomenat NaN que és l'error d'una operació amb números.

4.4 Tipus de dades i operadors

- **BigInt:** S'utilitza per a representar valors molt grans (per a càlculs matemàtics per exemple). Per utilitzar aquest tipus de dada s'afegeix la lletra "n" al final.
  - `const sencermoltgran = 9999999999999999999999999999999999999999999n;`

- **null:** S'utilitza per indicar que alguna cosa existeix però no té valor, el seu valor serà null (quan una variable té el valor null, direm que és del tipus null).

- **undefined:** Representa alguna cosa que no està definida. És el valor automàtic de les variables, paràmetres i propietats d'objectes que encara no tenen cap valor definit.

- **Symbol:** S'utilitza principalment per a garantir que el seu valor sigui únic i immutable. Es pot utilitzar com a propietats d'objectes per reduir accessos a aquestes propietats.

4.4 Tipus de dades i operadors

Tipus objecte

- **Objectes**:
    - S'utilitzen per emmagatzemar col·leccions de dades. Per exemple, a la vida real, si tenim un plat, aquest seria l'objecte, i les seves característiques (color, forma, material, etc.), serien les propietats.
    - A més dels propis objectes (Object), els arrays, les funcions (Function), les dates (Date) i les expressions regulars (RegExp), són objectes.
- **Propietats**:
    - Les propietats s'especifiquen mitjançant una col·lecció de parells "clau:valor" separats per comes:
        - La clau és sempre una cadena i ha de ser única.
        - El valor pot ser una primitiva, un objecte i també una funció. El valor no potser "undefined".
    - Cada propietat és com una variable.
- **Mètodes**: Són instruccions que poden canviar els valors que hem assignat a les propietats. Són les funcions.

4.4 Tipus de dades i operadors

- **Crear un objecte:** La forma més fàcil per crear un objecte és utilitzant la definició dels anomenat "objectes literals", on simplement s'obrin i es tanquen claus.

Crear objecte literal:

```
var = cotxe {
        portes: “2”,
        rodes: “4”,
        color: “blau”
    };
```

Tenim un objecte "cotxe" que té tres propietats "portes", "rodes" i "color"

Objecte buit:

```
let cotxe = {};
```

4.4 Tipus de dades i operadors

- **Crear un objecte:** Hi ha una altra manera de crear objectes:

```
var cotxe = new Object();
cotxe.portes = “2”;
cotxe.rodes = “4”;
cotxe.color = “blau”;
```

Els objectes són dinàmics, això vol dir que les seves propietats no tenen per què ser definides en el moment en què es crea l'objecte, podem afegir noves propietats a l'objecte en temps d'execució, (indicant el nom la propietat i assignant-li un valor o funció).

4.4 Tipus de dades i operadors

- **Accedir a una propietat:** Es pot accedir a les propietats de l'objecte amb un punt.

A l'exemple anterior si escrivim:

```
cotxe.portes
```

Això és: Accedim a la propietat portes de l'objecte cotxe.

- **Esborrar una propietat:** Per esborrar una propietat utilitzarem "delete".

Per exemple:

```
delete cotxe.portes
```

4.4 Tipus de dades i operadors

- **Accedir a un mètode:** Per accedir ho farem també amb un punt: "nom_del_mètode.function()".

  Exemple:

```
var = cotxe {
        portes: “2”,
        rodes: “4”,
        color: “blau”,
        frenada: function frenar(){
            ...
        }
      };

cotxe.frenada; //Executarem el mètode frenar.
```

4.4 Tipus de dades i operadors

Date

- "Date" és un objecte.
- Ens permet treballar tant amb dates com amb hores. Es pot utilitzar per emmagatzemar hores de creació o modificació, mesurar el temps, etc.
- Zones horàries: En JavaScript, treballarem o bé amb l'hora local (la que tendrà el nostre dispositiu), o bé amb l'hora universal coordinada (UTC). De forma predeterminada, quasi sempre, en JavaScript es retorna la data/hora local. Per obtenir UTC, s'ha d'especificar. Si s'escriu sense paràmetres retorna la data i l'hora actual del sistema.

  Podem fer:

  - **new Date():** Crea un objecte amb hora i data actual. Mostrarà una data com una cadena de text.
  - **new Date(milisegons):** Crea un objecte de data de temps zero (des de les 00:00:00 UTC del 1 de gener de 1970 més els milisegons especificats.
  - **new Date(cadenaData):** Crea un objecte de data a partir d'un string de data.
  - **new Date(any, mes, dia, hores, minuts, segons, milisegons):** Crea un objecte de data amb una data i hora específiques.

4.4 Tipus de dades i operadors

Arrays

- Són un conjunt de dades ordenades per posicions i totes associades a una sola variable.
- Els arrays són objectes.
- Les dades poden ser de qualsevol tipus de dada, és a dir, és pot crear un array que tengui una cadena a la primera posició, un número a la segona, etc.
- Per crear un array s'utilitza "new", ja que sabem que és un objecte.
- Hi ha dues maneres de crear un array:
    - Crear un array sense dades i sense dimensió: var arrayexemple = new Array();
    - Crear un array amb dimensió: var arrayexemple = new Array(n);

4.4 Tipus de dades i operadors

Arrays

Exemples:

- Array anomenat "adreces", però inicialment buit i sense dimensió, declarat amb ():

```
var adreces = new Array();
```

- Array anomenat "adreces", inicialment buit però amb dimensió:

```
var adreces = new Array(10);
```

- Array anomenat "noms", però inicialment buit i sense dimensió, declarat amb []:

```
var noms = [];
```

- Array anomenat "estiu", amb 3 posicions, i ja assignam les dades:

```
var estiu = new Array("juny", "juliol", "agost");
```

- Array anomenat "colors" amb 3 posicions, amb dades, però declarat amb []:

```
var colors = ["vermell", "blau", "blanc"] ;
```

4.4 Tipus de dades i operadors

Arrays

Exemple: Cream un array anomenat "diaSetmana", inicialment amb 7 posicions buides. I després assignam valors a cada posició.

```
<script>
   var diaSetmana = new Array(7); // Cream un array amb 7 posicions
   diaSetmana[0] = "Dilluns";  //donam valors a cada una de les posicions de l'array.
   diaSetmana[1] = "Dimarts";
   diaSetmana[2] = "Dimecres";
   diaSetmana[3] = "Dijous";
   diaSetmana[4] = "Divendres";
   diaSetmana[5] = "Dissabte";
   diaSetmana[6] = "Diumenge";
</script>
```

4.4 Tipus de dades i operadors

Expressions regulars

Són patrons utilitzats per trobar una determinada combinació de caràcters dins d'una cadena de text.

Una expressió regular pot crear-se de qualsevol de les següents maneres:

- Utilitzant una representació literal de l'expressió regular, consistent en un patró entre barres: var er = /ab+c/;

- Cridant a la funció constructora de l'objecte RegExp: var exp = new RegExp('ab+c');

4.4 Tipus de dades i operadors

Expressions regulars

Exemples:

```
var patro = /info/;
var patro = new RegExp("info");
```

Aquest patró o expressió, és molt senzill, a un comparació amb una cadena, retornaria "true" en el cas de que la cadena amb la que se compara sigui igual a "info".

4.4 Tipus de dades i operadors

Operadors

Quan es creen programes en qualsevol llenguatge, s'utilitzen els operadors, que serveixen per fer càlculs i operacions. El resultat d'aquests càlculs i operacions es lo que fa que el programa faci alguna cosa, i ens ofereix resultats.

Operadors aritmètics:

+ Suma de dos valors.
- Resta de dos valors. També pot utilitzar-se per canviar el signe d'un número.
* Multiplicació de dos valors.
/ Divisió de dos valors.
% El mòdul de la divisió de dos números (3%2 ens retornarà 1)
++ Increment en una unitat, s'utilitza amb un sol operand.
-- Decrement en una unitat, utilitzat amb un sol operand.

Operadors d'assignació

= Assignació. Assigna la part de la dreta a la part de l'esquerra.
+= Assignació amb suma. Realitza la suma de la part de la dreta amb la de l'esquerra i guarda el resultat a la part esquerra.
-= Assignació amb resta.
*= Assignació de la multiplicació.
/= Assignació de la divisió.
%= S'obté el mòdul i s'assigna.

Operadors de cadenes

+ Concatena dues cadenes.

4.4 Tipus de dades i operadors

Operadors

Operadors aritmètics de JavaScript	
Operador	Descripció
+	Suma
-	Resta
*	Multiplicació
**	Exponent
/	Divisió
%	Mòdul (reste de la divisió)
++	Increment
--	Decrement

4.4 Tipus de dades i operadors

Operadors

Operadors de comparació de JavaScript	
Operador	Descripció
==	Igual a
===	Igual valor i Igual tipus
!=	No igual
!==	No igual valor o no igual tipus
>	Més gran
<	Més petit
>=	Més gran o igual
<=	Més petit o igual
?	Operador ternari (per sentències if)

4.4 Tipus de dades i operadors

Operadors

Operadors de tipus	
Operador	Descripció
typeof	Retorna el valor de la variable
instanceof	Retorna true si si un objecte és una instància d'un tipus d'objecte

Operadors d'assignació de JavaScript		
Operador	Exemple	Notació convencional
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Ajustar fila de tabla

4.4 Tipus de dades i operadors

Operadors

Operadors de bits de JavaScript				
Operador	Descripció	Ex	Resultat	Decimal
&	AND	0101 & 0001	0001	1
|	OR	0101 | 0001	0101	5
~	NOT	~0101	1010	10
^	XOR	0101^0001	0100	4
<<	Desplaça a l'esquerra un número especificat de bits.	0101<<1	1010	10
>>	Desplaça a la dreta un número especificat de bits.	0101>>1	0010	2
>>>	Desplaça a la dreta, descartant i substituint per ceros.	5>>>1	0010	2

Operadors lògics	
Operador	Descripció
&&	And
||	Or
!	Not

4.4 Tipus de dades i operadors

L'operador typeof:

Podem utilitzar l'operador de tipus per trobar el tipus d'una variable. Retorna el tipus d'una variable o d'una expressió:

Exemples:

```
var numero = 2;
alert (typeof numero ) / / Mostra "number"
-----------------------
var text = "eltext";
alert (typeof eltext) / / Mostra: "string"
-----------------------
var varbooleana = false;
alert (typeof varbooleana) / / Mostra: "boolean"
```

4.4 Tipus de dades i operadors

L'operador typeof:

També permet provar l'existència d'una variable:

Exemple:

```
alert (typeof variable) // Mostrarà "undefined"
```

Si la instrucció typeof retorna "undefined", és que la variable és inexistent o està declarada però no té cap valor.

4.4 Tipus de dades i operadors

L'operador typeof:

És molt aconsellable demanar el tipus d'una variable, en alguns casos:

Exemple:

```
let variable;
if (typeof variable == 'undefined') {
   alert ('La variable és undefined, no té valor definit);
}
```