**Programe folosite în crearea site-urilor**

**1.HTML - (HyperText Markup Language)**

HTML este un standard folosit pentru construirea paginilor web și are abilitatea de a aduce împreună text, imagini, sunete și video. Documentele HTML sunt fișiere cu text simplu, de aceea ele pot fi editate folosind editoare standard ca NotePad și WordPad. Acestea pot fi salvate cu extensia .html, .htm sau .shtml și afișate folosind browsere ca Mozilla Firefox, Chrome, Safari, Opera, Internet Explorer și altele.

HyperText Markup Language este instrumentul de formatare standard pentru World Wide Web, deoarece are proprietatea de multi-platformă, ceea ce înseamnă că este compatibil și poate rula pe diferite tipuri de computere și dispozitive mobile cu diferite sisteme de operare.

**1.1.Scurtă istorie**

La sfârşitul anului 1991, cercetătorul britanic Tim Berners-Lee publică prima scriere, numită „HTML Tags”, publicaţie ce descrie 18 elemente care conțin designul inițial, relativ simplu de HTML. Cu excepția etichetei hyperlink, acestea au fost puternic influențate de SGML (Standard Generalized Markup Language) , un standard pentru definirea limbajelor de marcare generalizate pentru documente, folosit la CERN, acolo unde Tim Berners-Lee lucra.. Unsprezece dintre aceste elemente există încă în HTML.

Ca urmare a apariţiei HTML, Berners-Lee a dezvoltat un protocol de acces, prin hyperlink-uri, numit HTTP (HyperText Transfer Protocol).

Fiind într-un continuu proces de dezvoltare, cu modificări aduse constant, HTML a evoluat în mai multe versiuni, astfel:

* HTML 2.0 (Noiembrie 1995)
* HTML 3.2 (Ianuarie 1997)
* HTML 4.1 (Decembrie 1999)
* HTML5 (în curs de dezvoltare din octombrie 2011 şi până în prezent)

O altă descoperire importantă o reprezintă apariţia CSS (Cascading Style Sheets), alături de HTML 4.0, deoarece a permis separarea opţiunilor de formatare şi stilizare a paginilor web de scopul principal al HTML, de definire a structurii paginii web. Astfel, odată cu introducerea CSS, s-a putut realiza separarea noţiunilor de formatare de documentul HTML într-un alt fişier, ce poate fi accesat şi inclus în document, folosind o etichetă de legătură (link tag).

**1.2.Tag-uri HTML**

Tag-urile HTML sunt cuvinte și simboluri text incluse în paranteze unghiulare, <tag>, descriind de obicei începutul și finalul unei informații sau comenzi. Paginile web sunt vizualizate pe Internet cu ajutorul browserelor web, care citesc codurile HTML și le afișează astfel încât să poată fi citite de vizitatori. Practic, browserele web afișează informația conținută între tag-uri într-un anumit mod specificat de tag-urile corespunzătoare.

**1.2.1.Tag-uri principale**

|  |  |
| --- | --- |
| <!DOCTYPE> | Definește tipul de document |
| <html> | Definește un document HTML |
| <head> | Cuprinde informații despre document |
| <title> | Setează titlul |
| <body> | Definește cuprinsul documentului |
| <h1> to <h6> | Setează rubricile documentului |
| <p> | Setează apariția unui paragraf |
| <br> | Inserează o pauză de linie |
| <hr> | Definește o schimbare tematică a conținutului |
| <!--...--> | Definește un comentariu |

**Tabelul 1.1**

**1.2.2.Tag-uri de formatare**

Procedura corectă pentru formatarea textului și stilizarea lui este utlizarea CSS, însă în situații rare, se pot folosi și tag-uri HTML.

|  |  |
| --- | --- |
| <abbr> | Definește o abreviere sau un acronim |
| <address> | Definește informațiile de contact ale autorului |
| <b> | Setează îngroșarea textului |
| <blockquote> | Marchează o secțiune care este un citat din altă sursă |
| <mark> | Marchează textul prin evidențierea lui |
| <template> | Definește un șablon |
| <time> | Setează data/ceasul |
| <u> | Marchează textul într-un stil diferit față de cel normal |

**Tabelul 1.2**

Tag-uri ca <big>, <center>, <font>, <strike>, <tt> nu mai sunt compatibile în HTML5, recomandându-se folosirea CSS pentru suplinirea lor.

**1.2.3.Tag-uri pentru formulare și introducere**

|  |  |
| --- | --- |
| <form> | Definește un formular pentru introducere date |
| <input> | Marchează posibilitatea introducerii datelor |
| <textarea> | Definește mai multe linii pentru introducerea datelor |
| <button> | Definește un buton |
| <select> | Definește o listă |
| <optgroup> | Definește un grup de acțiuni dintr-o listă |
| <option> | Definește o opțiune dintr-o listă |
| <label> | Definește o etichetă pentru elementul introdus |
| <fieldset> | Grupează elemente asemănătoare într-un formular |
| <datalist> | Specifică o listă de opțiuni predefinite |

**Tabelul 1.3**

**1.2.4.Tag-uri pentru imagini**

|  |  |
| --- | --- |
| <img> | Definește o imagine |
| <map> | Definește o imagine-hartă |
| <area> | Definește o zonă într-o imagine-hartă |
| <canvas> | Este folosit pentru grafică, prin scripturi |
| <picture> | Definește un container pentru mai multe imagini |
| <svg> | Definește un container pentru grafice cu vectori scalabili |

**Tabelul 1.4**

**1.2.5.Tag-uri Audio/Video**

|  |  |
| --- | --- |
| <audio> | Definește conținut audio |
| <source> | Definește conținut media (<video>,<audio>,<picture>) |
| <track> | Definește text vizibil atunci când rulează conținut media |
| <video> | Definește conținut video |

**Tabelul 1.5**

**1.2.6.Tag-uri de legătură**

|  |  |
| --- | --- |
| <a> | Definește un hyperlink |
| <link> | Definește relația cu o resursă externă |
| <nav> | Definește legăturile în bara de navigare |

**Tabelul 1.6**

**1.2.7.Tag-uri pentru liste**

|  |  |
| --- | --- |
| <ul> | Definește o listă neordonată |
| <ol> | Definește o listă ordonată |
| <li> | Definește un element al listei |
| <dl> | Definește o listă de descrieri |
| <dt> | Definește un termen într-o listă de descrieri |
| <dd> | Definește o descriere a unui termen într-o listă |

**Tabelul 1.7**

**1.2.8.Tag-uri semantice și de stil**

|  |  |
| --- | --- |
| <style> | Definește informații stilistice |
| <div> | Definește o secțiune |
| <span> | Definește o secțiune |
| <header> | Definește antetul unui document/secțiune |
| <footer> | Definește subsolul unui document/secțiune |
| <main> | Definește conținutul principal dintr-un document |
| <section> | Definește o secțiune |
| <article> | Definește un articol |
| <details> | Definește detalii care se pot vizualiza/ascunde |
| <dialog> | Marchează o fereastră de dialog |
| <data> | Asignează conținutul unei traduceri automate |

**Tabelul 1.8**

**1.2.9.Tag-uri meta**

|  |  |
| --- | --- |
| <head> | Definește informații despre document |
| <meta> | Definește metadate despre document |
| <base> | Specifică adresa URL de bază |

**Tabelul 1.9**

**1.2.10.Tag-uri pentru programare**

|  |  |
| --- | --- |
| <script> | Definește un script |
| <embed> | Definește un container pentru o aplicație externă |
| <object> | Definește un obiect |
| <param> | Definește un parametru pentru un obiect |

**Tabelul 1.10**

**1.3.Atribute HTML**

Elementele HTML pot avea atribute în deschiderea etichetei. Atributul oferă informații suplimentare despre acel element. Atributele HTML sunt adăugate în eticheta de deschidere și pot avea o anumită valoare. Cele mai multe dintre atributele unui element sunt perechi nume-valoare, separate prin „=” și scrise în eticheta de start a unui element, după numele elementului. Valoarea poate fi între ghilimele simple sau duble, cu toate că valorile compuse din anumite caractere pot fi lăsate necotate în HTML. Lăsând valorile atributelor necotate este considerată o acțiune cu rezultate nesigure.

**1.3.1.Atribute comune**

Există mai multe atribute comune care pot apărea în mai multe elemente:

* Atributul „id” oferă un identificator unic pentru un element. Acest lucru este folosit pentru a identifica elementul astfel încât foile de stil pot modifica proprietățile sale de prezentare, iar script-urile pot modifica, anima sau șterge conținutul sau prezentarea acestuia. Anexat la URL-ul paginii, acesta oferă un identificator unic global pentru element, de obicei o sub-secțiune a paginii.
* Atributul „class” oferă o modalitate de clasificare a elementelor similare. Acest lucru poate fi utilizat în scopuri semantice sau de prezentare. De exemplu, un document HTML ar putea folosi semantic desemnarea class=„notație” pentru a identifica faptul că toate elementele cu această valoare de clasă sunt subordonate textului principal al documentului. În prezentare, astfel de elemente ar putea fi adunate împreună și prezentate ca note de subsol pe o pagină, în loc de a apărea în locul în care apar în sursa HTML. Atributele de clasă sunt utilizate semantic în microformate și pot avea valori multiple specificate.
* Atributul „style” este folosit pentru a atribui proprietăți de pezentare unui anumit element. Se consideră mai bună practica de a folosi un atribut „id” sau „class” al unui element pentru a selecta elementul din interiorul unei foi de stil.
* Atributul „title” este folosit pentru a atașa explicație subtextuală unui element. În cele mai multe browsere acest atribut este afișat ca o notă informativă.
* Atributul „lang” identifică limbajul natural al conținuturilor elementului, care pot fi diferite de cele din restul documentului.

**1.3.2.Atribute boolean**

O serie de atribute sunt atribute booleane. Prezența unui atribut boolean pe un element reprezintă valoarea adevărată, iar absența atributului reprezintă valoarea falsă.

Dacă atributul este prezent, valoarea acestuia trebuie să fie șirul gol sau o valoare care este o potrivire nesemnificativă ASCII pentru numele canonic al atributului, fără niciun spațiu alb sau principal. Valorile „adevărat” și „fals” nu sunt permise atributelor booleane. Pentru a reprezenta o valoare falsă, atributul trebuie omis în totalitate.

|  |  |
| --- | --- |
| checked | Indică selecția într-o căsuță de verificare |
| selected | Indică opțiunea selectată dintr-o listă de opțiuni |
| disabled | Dezactivează butoane, opțiuni, selecții, etc. |
| readonly | Împiedică utilizatorul să scrie text |

**Tabelul 1.3.1**

**1.3.3.Atribute prezentaționale**

Atributele prezentaționale au ca utilitate stilizarea elementelor HTML.

|  |  |
| --- | --- |
| style | Definește stilurile CSS |
| size | Definește lungimea unui element |
| color | Definește culoarea unui element |
| border | Setează liniile de frontieră într-un tabel |
| cellspacing | Schimbă spațiul între celulele tabelului |
| align | Aliniază stânga, dreapta și centru |

**Tabelul 1.3.2**

Dintre toate atributele prezentaționale doar „style” și „size” se mai folosesc în HTML5, toate celelalte fiind înlocuite de atribute CSS.

**1.4.HTML5**

HTML5 este un limbaj utilizat pentru structurarea și prezentarea conținutului pentru World Wide Web, reprezentând o tehnologie nucleu pentru Internet.

Propus inițial pentru software-ul Opera, HTML5 este a cincea revizuire a standardului HTML și din octombrie 2011 este în curs de dezvoltare. Obiectivele sale principale au fost acelea de a îmbunătăți limbajul cu un suport pentru cele mai recente apariții multimedia, menținându-l în același timp, ușor de citit de oameni și bine înțeles de computere și device-uri.

HTML5 este destinat să subsumeze nu numai HTML4, dar și XHTML1 și DOM2HTML.  
Urmărind predecesorii săi imediați HTML 4.01 și XHTML 1.1, HTML5 este un răspuns la observația că HTML și XHTML, utilizate în comun pe World Wide Web, sunt un amestec de caracteristici introduse de specificații diferite. Împreună cu acestea mai sunt și caracteristicile diferite aduse de software, de browsere, și multe erori de sintaxă în documentele web existente. Astfel, HTML5 devine o încercare de a defini un singur limbaj de marcare, care poate fi scris în oricare dintre sintaxele HTML sau/și XHTML. Acesta include modele detaliate de prelucrare pentru a încuraja mai multe implementări interoperabile, extinde, îmbunătățește, raționalizează disponibilitățile pentru documentele web și introduce marcaje și interfețe de programare a aplicațiilor (API) pentru aplicații web complexe.

Din aceste motive, HTML5 este un bun candidat pentru aplicațiile de platforme mobile. Multe caracteristici ale HTML5 au fost create din considerarea că va trebui să devină capabil să ruleze pe dispozitive cum ar fi smartphone-urile sau tabletele.

În special, HTML5 aduce multe noi caracteristici sintactice. Acestea cuprind elemente ca <video>, <audio> și <canvas>,  precum și integrarea de conținut cu grafică vectorială scalabilă (SVG), care a înlocuit utilizarea tag-ului generic <object>. Aceste noutăți sunt proiectate pentru a facilita includerea și manipularea în web a conținuturilor multimedia și grafice fără a fi nevoie să se recurgă la proprietățile de plugin și API. Alte noi elemente ca <section>, <article>, <header> și <nav> sunt proiectate să îmbunătățească conținutul semantic al documentelor. Noi atribute au fost introduse în același scop dar, în același timp, unele elemente și atribute au fost îndepărtate. Elemente ca <a>, <cite> și <menu> au fost schimbate, redefinite și standardizate. API-urile și DOM-urile (Document Object Model) nu mai sunt lăsate pe planul secund și devin părți fundamentale în specificațiile HTML5.

 HTML5, de asemenea, definește in câteva detalii, prelucrările necesare pentru documentele invalide, astfel încât sintaxa erorilor va fi tratată uniform de toate browserele cunoscute.

**1.4.1.Caracteristicile HTML5**

**1.4.1.1.Marcarea**

HTML5 introduce elemente și atribute care reflectă utilizarea tipică pe site-urile web moderne. Unele dintre ele sunt înlocuiri semantice pentru utilizări comune ale elementelor de bloc generic (<div>) și inline (<span>), de exemplu <nav> (bloc de navigare site), <footer> (de obicei referindu-se la partea de jos a paginii web sau la ultimele linii de cod HTML), sau <audio> și <video> în loc de <object>. Unele elemente depreciate din HTML 4.01 au fost retrase, inclusiv elemente pur formale, cum ar fi <font> și <center>, ale căror efecte au fost de mult timp înlocuite cu mai capabilul CSS. Există, de asemenea, un accent reînnoit pe importanța de scripting DOM (de exemplu, JavaScript) în comportamentul Web.

Sintaxa HTML5 nu se mai bazează pe SGML, în ciuda similitudinii marcării sale.

Aceasta este dotată cu o noua linie introductivă care arată ca o declarație de tip document SGML, <!DOCTYPE html>, care declanșează modul de redare care respectă standardele.

**1.4.1.2.Noi API-uri**

Față de specificarea marcării, HTML5 specifică și scripting pentru interfețe de programare a aplicațiilor (API-uri), care pot fi utilizate cu JavaScript.

Se introduc, astfel, noi API- uri, cum ar fi:

* Elementul canvas pentru modul direct de proiectare 2D
* Redare media cronometrată
* Aplicații web offline
* Editarea de documente
* Drag-and-drop
* Mesagerie cross-document
* Managementul istoricului browser-ului
* Înregistrare manipulant de protocol și tip MIME
* Microdate
* Stocare web, un cadru de stocare pereche de chei-valoare care oferă un comportament similar cu cookie-uri, dar cu o capacitate de stocare mai mare

Nu toate tehnologiile de mai sus sunt incluse în specificația W3C HTML5, deși ele sunt în specificația HTML WHATWG. Unele tehnologii conexe, care nu fac parte nici din HTML W3C, nici din specificația WHATWG HTML, sunt după cum urmează:

* *Geolocation*, folosit pentru localizare
* *Web SQL Database*, o bază de date SQL localp (nu mai este menținută)
* *Indexed Database API*, o stocare de chei-valori indexate ierarhic (fosta WebSimpleDB)
* *HTML5 API File*, se ocupă de încărcarea fișierelor și manipularea lor
* *Directories and System*, un API destinat să satisfacă cazuri de utilizare stocare

pe parte de client, care nu sunt deservite de bazele de date

* *File Writer*, un API pentru scrierea de fișiere din aplicații web
* *Web Audio API*, un API JavaScript de nivel înalt pentru prelucrarea și sintetizarea sunetelor în aplicațiile web
* *ClassList API*, folosit pentru a adăuga, elimina și comuta clasele unui element

HTML5 nu poate oferi animație în pagini web. Este necesară o funcționalitate suplimentară JavaScript sau CSS3 pentru animarea elementelor HTML.

**1.5.Editoare HTML**

Codul HTML al unei pagini web poate fi scris cu oricare editor de text, însă editoarele HTML specializate pot oferi conveniență și funcționalitate suplimentară. De exemplu, multe editoare HTML nu lucrează numai cu HTML, ci și cu tehnologii corespunzătoare precum CSS, XML și JavaScript sau ECMAScript. În unele cazuri, pot administra și comunicații cu servere la distanță via FTP și WebDAV, dar și sisteme de management al versiunii precum CVS sau Subversion.

**1.5.1.Tipuri de editoare HTML**

Există diferite tipuri de editoare HTML: text, obiect și WYSIWYG (What You See Is What You Get).

**1.5.1.1.Editoarele de text**

Editoarele de text (sursă) folosite pentru HTML oferă de obicei, posibilitatea evidențierii sintaxei. Template-uri, bare de utilități și accesul rapid al tastaturii pot introduce ușor elemente și structuri HTML. Wizard, bulele informative și autocompletarea pot ajuta în sarcinile obișnuite.

Editoarele de text HTML includ de obicei, funcțiuni integrate sau integrarea cu instrumente externe pentru astfel de sarcini precum sursa și controlul versiunii, verificare linkuri, verificare cod și validare, curățire cod și formatare, verificare gramatică, încărcare prin FTP sau WebDAV și structurare ca un proiect.

Editoarele de text necesită înțelegerea de către utilizator a HTML și a oricărei alte tehnologii web pe care designerul vrea să o folosească, precum CSS, JavaScript și limbaje scripting pentru server.

**1.5.1.2.Editoare de obiecte**

Unele editoare permit editarea alternativă a textului sursă al obiectului într-un mod vizual mai organizat decât simpla evidențiere prin colorare, dar fără WYSIWYG. Unele editoare WYSIWYG includ opțiunea de folosire a ferestrei de paletă care permite editarea parametrilor în câmpuri pentru fiecare parametru, sau ferestre de text pentru a edita întregul grup de text sursă pentru obiectul selectat. Ele pot include widget-uri pentru a prezenta și selecta opțiuni când sunt editați parametrii.

**1.5.1.3.Editoare WYSIWYG**

Editoarele HTML WYSIWYG oferă o interfață de editare care arată cum va fi afișată pagina într-un browser web. Unele editoare, precum cele sub forma extensiilor de browser, permit editarea într-un browser web. Întrucât folosirea unui editor WYSIWYG nu necesită cunoștințe HTML, sunt ușor de folosit de un utilizator, mediu ca nivel, de calculator.

Vizualizarea WYSIWYG este realizată prin inserarea unui dispozitiv asemănător cu cel folosit într-un browser web. Acest dispozitiv a fost îmbunătățit considerabil de dezvoltatorii editoarelor pentru a permite tipărirea, copierea, ștergerea și mutarea conținutului. Scopul este ca oricând pe perioada editării, rezultatul sa fie văzut așa cum se va vedea el într-un browser web.

Editoarele WYSIWYG fac web designul mai rapid și mai ușor, dar mulți profesioniști încă folosesc editoarele text, în ciuda faptului că cele mai multe editoare WYSIWYG au și un mod de editare manual pentru codul HTML. Web-ul nu a fost creat inițial ca un mediu vizual, iar încercările de a avea mai mult control, precum prin CSS, au fost foarte puțin sprijinite de majoritatea browserelor web. Din această cauză, codul generat automat de editoarele WYSIWYG dăunau adesea mărimii fișierelor și compatibilității cu browserele.

**2.CSS - (Cascading Style Sheets)**

Cascading Style Sheets este un standard pentru formatarea și afișarea elementelor unui document HTML, permițând definirea construcțiilor de stil ca fonturi, culori și poziționare.

Odată cu introducerea HTML 3.2 au fost introduse și atributele de personalizare a tag-urilor precum "font", "color" etc. A fost atunci cand limbajul de programare HTML a devenit greoi. Fiecare pagină a website-ului trebuia luată separat și modificate proprietățile elementelor principale. Aceasta problemă a fost rezolvată în versiunea 4.0 a HTML-ului. Toate atributele de personalizare au fost scoase și salvate într-un fișier extern cu extensia ".css".

În felul acesta, modificând un singur fișier, putem schimba forma în care sunt afișate toate paginile unui website.

**2.1.Inserarea foilor de stil**

O foaie de stil CSS poate fi inserată prin mai multe modalități, astfel:

a)**Ca fișier extern**

O foaie de stil externă este ideală atunci când stilul respectiv se aplică mai multor pagini. Cu o foaie CSS externă, puteți modifica aspectul întregului site modificând un singur fișier.

Fiecare pagină trebuie să se lege la foaia de stil folosind tagul <link> în interiorul secțiunii head:

<html>

<head>

<link rel=”stylesheet” type=”text/css” href=”style.css”/>

</head>

<body>

.......

</body>

</html>

unde:

- atributul **rel** specifică relația dintre documentul apelant și documentul legat. Are numeroase valori posibile, dar în cazul de față valoarea sa trebuie să fie **stylesheet**;

- atributul **type** specifică tipul documentului legat, adică definește ce fel de resursă este (extensia fișierului nu este suficientă pentru a determina această) - în cazul nostru trebuie să aibă valoarea **text/css;**

- atributul **href** cum am văzut și la elementul link, specifică adresa (URL) de unde se introduce fișierul foaie de stil.

O foaie externă de stil poate fi scrisă în **Notepad++,** nu trebuie să conțină taguri html și trebuie salvată cu extensia .css. Un exemplu de foaie de stil externă, memorată în fișierul style.css, este prezentată mai jos:

hr {color:sienna;}

p {margin-left:20px;}

body {background-image:url("roses.gif");}

b) **Ca foaie internă de stil**

O foaie internă de stil trebuie folosită când un singur document are un anumit stil. Foaia internă de stil se definește în secțiunea **head**a paginii HTML folosind tagul **<style>**, ca în exemplul următor:

<head>

<style type="text/css">

hr {color:#424242;}

p {margin-left:20px;}

body {background-image:url("roses.gif");}

</style>

</head>

c) **Ca stiluri inline**

Un stil inline pierde multe din avantajele foilor de stiluri amestecând conținutul cu prezentarea. Se recomandă folosirea acestei metode cât mai rar. Pentru a utiliza un stil inline, se folosește atributul style în tagul respectiv. Atributul style poate conține orice proprietăți CSS. Exemplul următor ilustrează cum se poate schimba culoarea și marginea stângă a unui paragraf:

<p style="color:#424242; margin-left:20px">Acesta este un paragraf.</p>

d) **Ca foi multiple de stiluri**

Dacă anumite proprietăți ale unui selector au fost setate în foi de stiluri diferite, valorile vor fi moștenite din cea mai specifică foaie de stil. Spre exemplu,într-o foaie de stil externă, selectorul h3 are setate următoarele trei proprietăți:

h3 {

color:red;

text-align:left;

font-size:8pt;

}

Și foaia de stil intern setează pentru **h3** următoarele proprietăți:

h3 {

text-align:right;

font-size:20pt;

}

Dacă pagina care conține această foaie internă de stil se leagă și la foaia de stil externă, atunci proprietățile pentru h3 vor fi:

color:red;

text-align:right;

font-size:20pt;

Culoarea este moștenită din foaia externă, iar alinierea și dimensiunea fontului sunt înlocuite cu valorile din foaia internă de stil.

**2.2.Ordinea de aplicare a foilor de stil**

Termenul „Cascading” se referă la maniera în care fiecare regulă de stil se aplică elementelor într-un document HTML. Într-o foaie de stiluri, acestea formează o ierarhie în care stilurile mai specifice înlocuiesc stilurile mai generale. Este responsabilitatea CSS să determine prioritatea regulilor de stil conform acestei ierarhii, care stabilește, astfel un efect de cascadă.

La modul general, putem spune că stilurile se vor mixa într-o foaie de stil virtuală după următoarele reguli:

* setările implicite ale browserului
* foi de stiluri externe (din fișiere CSS)
* foi de stiluri interne (din secțiunea head)
* stiluri inline (din tagul elementului HTML)

**2.3.Selectori**

Odată cu adăugarea elementelor într-o pagină web, acestea pot fi stilizate folosind CSS. Un selector definește exact cărui element sau căror elemente din documentul HTML li se vor aplica aceste stiluri ca mărime, culoare sau poziționare.

Cei mai comuni selectori folosiți sunt:

* **selectorii de tip element** – vizează elemente în funcție de numele acestora;

De exemplu, având codul HTML:

<body>  
 <h1> Headline 1 </h1>  
 <h1> Headline 2 </h1>

<p> Paragraph 1 </p>  
 <p> Paragraph 2 </p>  
 </body>

Dorim să selectăm pentru stilizare doar elementele <p> din documentul HTML. Astfel, va fi nevoie doar să specificăm numele elementului și să definim regulile de formatare.

Sintaxa CSS: **p** {  
 color: red;  
 font-size: 20px;  
 }

* **selectorii de tip id** - vizează elementul care este definit printr-un id specificat;

Exemplu HTML: <h1 **id**="first-headline"> Headline 1 </h1>

Sintaxa CSS: **#first-headline** {  
 color: purple;  
 font-size: 30px;  
 }

* **selectorii de tip clasă** – permit selectarea unui element pe baza atributului de clasă, indiferent de numele etichetei lui;

Exemplu: a) HTML:

<h1 **class**="my-first-class"> Headline 1 </h1>

b) Sintaxa CSS:

**.my-first-class**{  
 color:green;  
 }

* **selectorii de tip atribute** – vizează elemente cu un atribut și o valoare anume; acest tip de selectori poate fi util în cazul câmpurilor de intrare;

De exemplu, având codul HTML: <body>  
 <input type="text" />  
 <input type="number" />  
 </body>

Pentru a specifica reguli doar pentru intrările de numere, de exemplu, trebuie să definim un selector de atribute (după numele etichetei din paranteze, scriind numele atributului și valoarea acestuia) ca mai jos:

**input[type=number]** {  
 background: gray;  
 }

Astfel regula se va aplica doar pentru câmpurile de introducere a numărului.

* **selectorul universal** – vizează toate elementele din pagină, aplicându-le simultan aceleași reguli.

Sintaxa: **\*** {  
 font-size: 20px;  
 }

**3.JavaScript**

JavaScript (JS) este un limbaj de programare orientat obiect, folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea site-urilor web, dar este folosit și pentru accesul la obiecte încapsulate (embedded objects) în alte aplicații.

JavaScript este un limbaj de scriptare (scripting language), reprzentând unul dintre cele trei limbi de bază utilizate pentru a dezvolta site-uri web. În timp ce HTML și CSS oferă structura și stilul unui site web, JavaScript permite adăugarea unor noi funcționalități și comportamente pe site-ul web, permițând vizitatorilor site-ului web să interacționeze cu conținutul în mai multe moduri. Astfel, folosit pentru a controla conținutul dinamic al site-ului web, JavaScript introduce caracteristici precum:

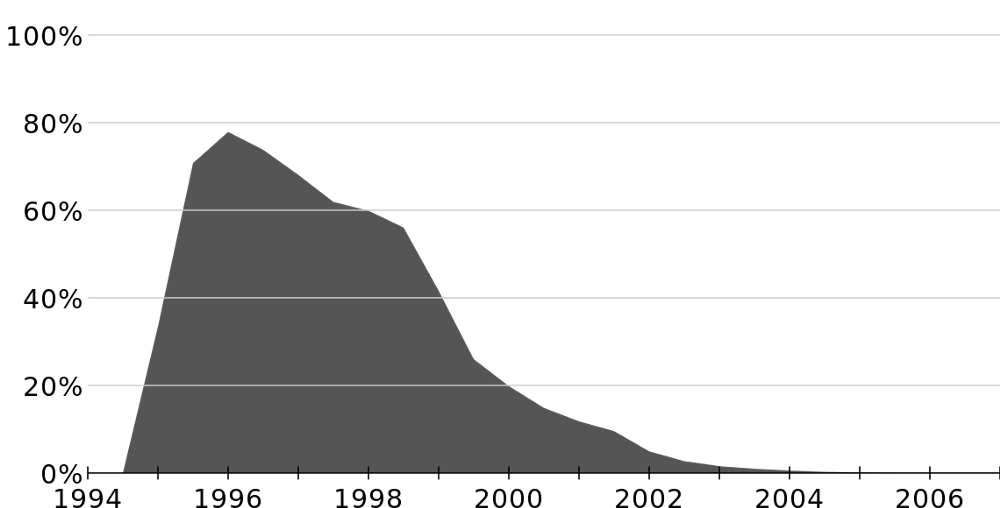
* grafică animată
* diapozitive foto
* sugestii de completare automată a textului
* forme interactive

**3.1.Scurtă istorie**

Începutul și mijlocul anilor 1990 a fost o perioadă importantă pentru Internet. Jucătorii cheie precum Netscape și Microsoft se aflau în mijlocul „războiului browserelor”, astfel avem pe un flanc Navigator-ul Netscape (cu JavaScript), iar pe celălalt flanc, Internet Explorer-ul Microsoft (cu Jscript).

În septembrie 1995, un programator Netscape, pe nume Brandan Eich, a dezvoltat un nou limbaj de scriptare în doar 10 zile. Inițial a fost numit Mocha, dar a devenit rapid cunoscut sub numele de LiveScript și, mai târziu, JavaScript.

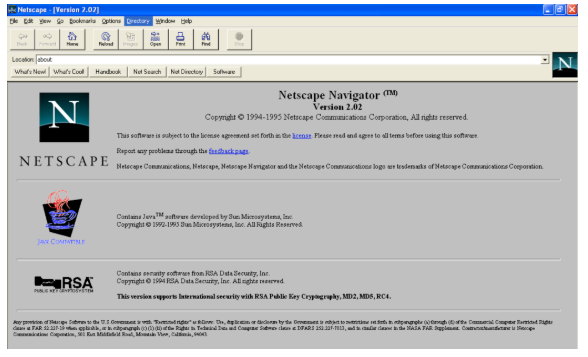
Netscape, pentru o perioadă, a făcut cel mai bun browser din lume și s-a bucurat de dominarea pieței.



**Figura 1. Cota de piață Netscape Navigator**

La sfârșitul anului 1995, proiectul Internet Explorer a fost demarat într-o încercare completă de combatere a controlului asupra platformei emergente din Netscape.

Până când lumea browserului s-a schimbat și Internet Explorer a devenit browserul dominant, JavaScript a evoluat în standardul aprobat pentru scrierea procesării interactive rulate într-un browser web.



**Figura 2. Netscape Navigator 2**

**3.1.1.JavaScript vs. Java**

Există deseori o oarecare confuzie în legătură cu cele două noțiuni, însă JavaScript și Java nu au aproape nimic în comun. Numele JavaScript a provenit din suportul Netscape al applet-urilor Java din browserul său. Mulți spun că a fost, de asemenea, o tactică de marketing pentru a devia o oarecare atenție de la Java, care a fost cel mai zgomotos limbaj din acea vreme. Pentru a rula programe Java, codul trebuie mai întâi compilat într-o formă executabilă. Pe de altă parte, JavaScript a fost creat pentru a fi interpretat în timpul rulării, ceea ce îl face mult mai dinamic.

JavaScript nu a reușit să înceapă cel mai bine. Nu s-a comportat la fel de bine, iar cei care se dezvoltă în Java au considerat JavaScript mai mult ca un „lipici interfață-utilizator” pentru a fi folosit mai ales de designeri și alți non-ingineri. Însă realitatea este că a avea un limbaj „lipici” a permis Internetului să înflorească cu adevărat. Programatorii ar putea utiliza evenimente și pot compune componente interactive. Și datorită acestui fapt, JavaScript s-a răspândit ca un incendiu și a devenit foarte repede „lingua franca” a web-ului.

**3.1.2.Ajax: O nouă abordare a aplicațiilor web**

Anul 2005 s-a dovedit a fi unul mare pentru JavaScript. O lucrare lansată de Jesse James Garrett a prezentat Ajax, o suită revoluționară de tehnologii care includea JavaScript. Ajax a îmbunătățit mult experiența utilizatorului, permițând paginilor web să se simtă mai mult ca niște aplicații desktop native. Aceasta a împins cu adevărat JavaScript în centrul atenției ca limbaj de programare profesional.

Această lucrare este considerată a fi una dintre elementele de bază fondatoare ale comunității JavaScript. La vremea respectivă, JavaScript a avut multe provocări, inclusiv natura sa verbală atunci când face lucruri simple și probleme de incompatibilitate între browsere. Comunitatea a răspuns cu cadre și biblioteci JavaScript mari și populare, cum ar fi Dojo și Mootools și jQuery.

Ajax reunește mai multe tehnologii, fiecare înflorind la propriu:

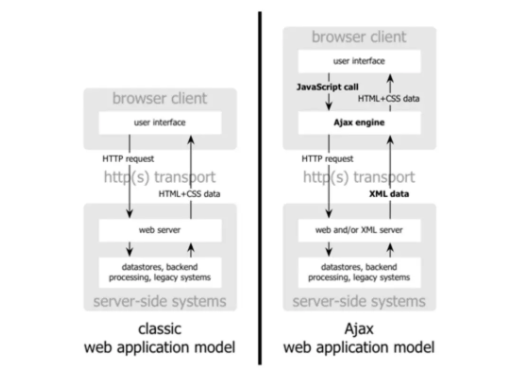
• prezentare bazată pe standarde folosind XHTML și CSS;

• afișare dinamică și interacțiune cu ajutorul Document Object Model;

• schimbul și manipularea datelor folosind XML și XSLT;

• recuperare asincronă de date utilizând XMLHttpRequest;

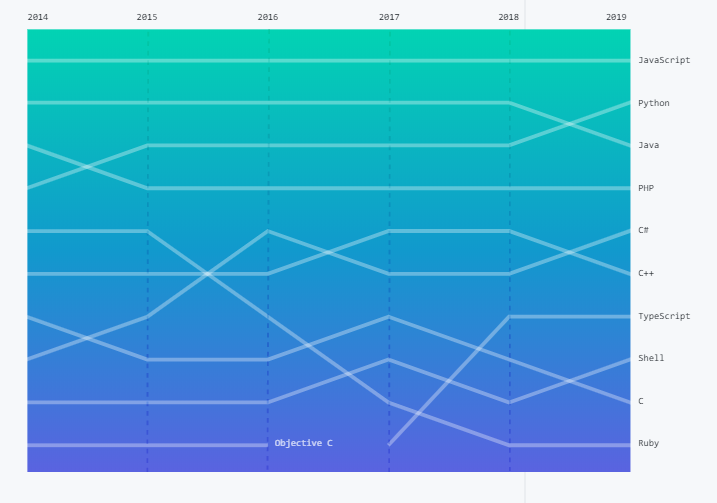
• JavaScript, având rolul de a leaga totul împreună.



**Figura 3.   
Modelul tradițional pentru aplicații web (stânga) comparativ cu modelul Ajax (dreapta)**

**3.1.3. Ascensiunea JavaScript**

De la începutul său ușor stângaci, JavaScript a ajuns să fie cel mai popular limbaj de programare din lume. Conform unui raport al octoverse.github.com, JavaScript este cel mai folosit limbaj de programare, din 2014 până în 2019. O serie de cadre și biblioteci JavaScript, cum ar fi Ember, Angular, React și Vue, au fost dezvoltate pentru a permite scrierea aplicațiilor web puternice și complicate folosind echipe mici în perioade scurte de timp. Alături de software-ul client și server, acum este posibil scrierea aplicațiilor mobile native folosind JavaScript. Nu este surprinzător, acest lucru devine din ce în ce mai popular datorită capacității de a partaja cod între lumile de telefonie mobilă și web.



**Figura 4. Topul limbajelor de programare în timp**

Ascensiunea JavaScript către cel mai popular limbaj de programare din lume este sinonimă cu creșterea Internet în sine.

Creat din necesitate, este utilizat pentru a construi 95,2% (1,52 miliarde) de site-uri web astăzi, inclusiv unele dintre cele mai mari din lume, cum ar fi Facebook și YouTube. Fără aceasta, nu am avea aplicații web populare și utile, cum ar fi Google Maps și eBay.

**3.2.Biblioteci și cadre JavaScript**

Cazurile de utilizare în care JavaScript a demonstrat că înlocuiește platformele tradiționale sunt în continuă creștere. Pe măsură ce domeniul de utilizare crește, devine extrem de important să se utilizeze o metodă de distribuție a codului, care nu numai că aduce simplitate în dezvoltarea ulterioară, ci permite și reutilizarea codului de către mase. Bibliotecile JavaScript rezolvă foarte bine această problemă și furnizează cod JavaScript pre-scris care face ușor de făcut sarcinile comune sau complexe. Bibliotecile JavaScript vizează adesea sarcini specifice, cum ar fi manipularea DOM, configurarea cadrului și gestionarea AJAX.

Diferența cheie între bibliotecile JavaScript și cadre este că bibliotecile constau din funcții pe care o aplicație le poate apela pentru a îndeplini o sarcină, în timp ce un cadru definește modul în care un dezvoltator proiectează o aplicație. Cu alte cuvinte, cadrul apelează la codul aplicației, mai degrabă decât invers. Natura acestor instrumente s-a schimbat în ultimii ani. În 2015, de exemplu, jQuery era cea mai bună bibliotecă JavaScript. Astăzi, potrivit JavaScript Scene, bibliotecile JavaScript tradiționale rămân importante, dar se confruntă cu o concurență sporită alcătuită din cadrele JavaScript, cum ar fi React, Angular, Node.js și Ember.js.

**3.2.1.Biblioteci JavaScript**

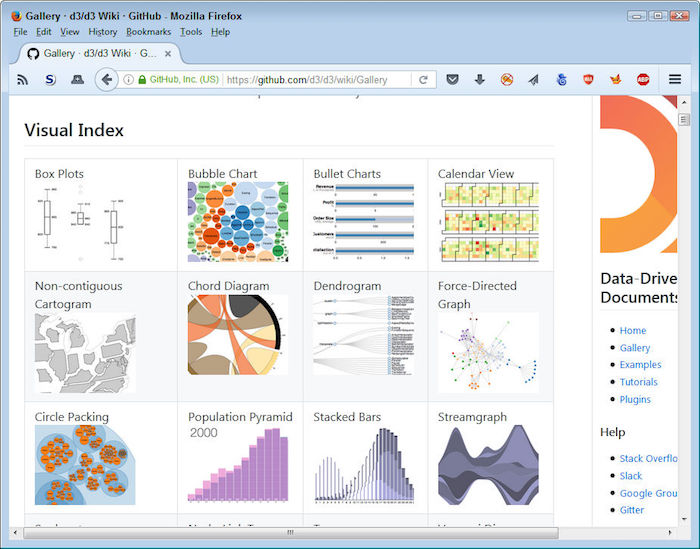
* **D3.js**

Multe site-uri web moderne sunt bazate pe date. Site-urile de știri, de exemplu, trebuie să-și reîmprospăteze conținutul în mod constant, ceea ce face imposibilă reconfigurarea la fiecare câteva secunde pentru a susține aceste modificări. Biblioteca D3.js este unică prin faptul că este axată pe date. Pe lângă flexibilitate extremă, prezentarea relativ simplificată a D3.js ajută la îmbunătățirea vitezei, eliminând o mulțime de bagaje pe care alte biblioteci le folosesc.

Principalul avantaj al acestei biblioteci este manipularea sa extrem de flexibilă a conținutului.

Pentru un site de comerț electronic însă, este posibil ca această bibliotecă să nu fie cea mai bună alegere, pentru a evita transformarea site-ului într-o prezentare de date și informații.

Imaginea de mai jos arată doar câteva dintre numeroasele prezentări de date care se găsesc în indexul vizual al D3.js.



**Figura 5. Indexul vizual al D3.js**

* **jQuery**

Biblioteca **jQuery** a fost concepută pentru a ușura și îmbunătăți procese precum traversarea arborelui DOM în HTML, managementul inter-browser al evenimentelor, animații și cereri tip AJAX. jQuery și-a câștigat statutul de conducător al paginilor web. Multe site-uri utilizează jQuery din trei motive esențiale:

- este ușor de învățat, oferind mai multe exemple decât majoritatea bibliotecilor

- este rapid

- are o bază uriașă instalată, bucurându-se de o cantitate semnificativă de sprijin comunitar și contribuitori la proiect.

jQuery se poate folosi pentru a rezolva următoarele probleme specifice programării web:

- selecții de elemente în arborele DOM, folosind propriul motor de selecții open source Sizzle, un proiect născut din jQuery

- parcurgerea și modificarea arborelui DOM (incluzând suport pentru selectori CSS3 și XPath simpli)

- înregistrarea și modificarea evenimentelor din browser

- manipularea elementelor CSS

- efecte și animații

- cereri tip AJAX

- extensii (plugin-uri)

* **jQuery UI**

jQuery UI este doar unul dintre numeroasele suplimente pentru jQuery, dar este cel mai des utilizat. Este folosit pentru a adăuga elemente grafice de bază în aplicație după ce este încorporat jQuery. Biblioteca jQuery UI vine cu o mulțime de widget-uri interesante, care fac posibilă configurarea aplicației pentru a oferi servicii utile.

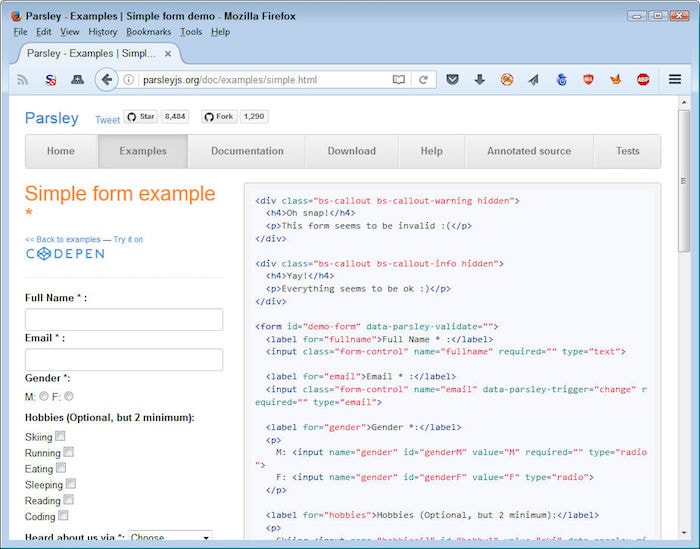
Această bibliotecă iese în evidență din mai multe motive. Cel mai important, JQuery UI funcționează cu aproape orice browser.

Problema cu grupul de biblioteci jQuery este că au devenit mari. Dimensiunea bibliotecilor JQuery le poate face să se încarce lent pe dispozitive mai mici. Bibliotecile terțe părți alternative, cum ar fi Granim.js și Multiple.js, au apărut pentru a oferi un subset de funcții jQuery, uneori cu un set focalizat de funcții suplimentare, cu timp de încărcare semnificativ mai rapid.

* **Parsley**

Validarea formularului într-o pagină web este o sarcină esențială. Deoarece datele sunt acum analizate de mașini, acuratețea acestora este mai esențială ca niciodată. Obținerea unor date „curate” necesită mult timp și nu este niciodată la fel de exactă ca obținerea unor informații corecte oferite de utilizator.

Parsley oferă mai multe tehnici de validare a formularului decât în ​​orice altă bibliotecă JavaScript. Chiar și validarea presupus simplă prezentată mai jos conține multe tipuri de validare diferite, împreună cu codul necesar pentru a efectua sarcina.



**Figura 6. Exemplu de cod pentru mai multe tipuri de validare**

* **QUnit**

Chiar dacă este utilizat un script, testarea manuală a aplicațiilor complexe, dacă poate găsi chiar toate erorile, consumă mult timp și în principiu, este obligatorie.

Funcționalitatea de testare oferită de multe biblioteci însă, lasă mult de dorit.

QUnit este una dintre multele biblioteci de testare a unităților disponibile pentru JavaScript. În programare, testarea unităților este o metodă de testare software prin care unitățile individuale de cod sursă, seturi de unul sau mai multe module de programe, împreună cu datele de control asociate, procedurile de utilizare și procedurile de operare, sunt testate pentru a determina dacă sunt potrivite pentru utilizare.

Scopul testării unităților este de a izola fiecare parte a programului și de a arăta care părți individuale sunt corecte și care nu.

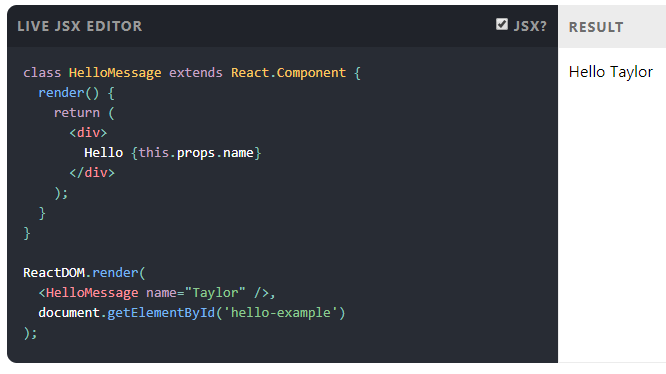
* **React**

Considerată de obicei bibliotecă, React este uneori denumit cadru. În contextul abordării Model-View-Controller (MVC) pentru crearea de aplicații la scară largă, React oferă posibilitatea de vizualizare.

React nu se bazează pe DOM-ul (Document Object Model) browserului, ci folosește în schimb un DOM virtual care afișează rapid informațiile. Unul dintre cele mai bune lucruri despre React este că este declarativ, ceea ce înseamnă că îi spui cadrului ce vrei să faci, nu cum să faci.

Prețul plătit însă pentru viteza și capacitatea de utilizare a MVC, este un anumit grad de complexitate. Chiar și o componentă mică necesită un pic de cod (așa cum demonstrează exemplele de pe site-ul React). Se câștigă însă, flexibilitate și viteză atunci când proiectele sunt cu adevărat mari.

Componentele React implementează o metodă de randare care preia date de intrare și returnează ceea ce trebuie afișat. Exemplul de mai jos utilizează o sintaxă asemănătoare XML numită JSX. Datele de intrare care sunt transmise către componentă pot fi accesate de render() prin this.props.



**Figura 7. Exemplu cod React**

**3.3.Cadre JavaScript**

* **Angular**

Angular face posibilă extinderea HTML-ului, acționând ca porțiunea de control a MVC. Controlerele sunt comportamentul din spatele componentelor DOM. Angular face posibilă crearea de noi comportamente într-o manieră naturală și simplă. Extensiile sfârșesc prin a fi adaosuri la HTML, mai degrabă decât ceva fixat în loc. Motivele esențiale pentru utilizarea acestui produs sunt viteza și performanța și setul de instrumente pentru dezvoltarea aplicațiilor de care dispune. Este complet extensibil și funcționează bine cu alte biblioteci. Fiecare caracteristică poate fi modificată sau înlocuită pentru a se potrivi cu fluxul de lucru unic de dezvoltare și nevoile funcției.

Cu toate acestea, codificarea poate deveni complexă, iar acest cadru poate fi mai potrivit pentru întreprinderi decât startup-urile mici. Este important de reținut că Angular se bazează și pe TypeScript în loc de JavaScript pur, ceea ce crește curba de învățare, dar oferă scalabilitate.

Din fericire, Angular vine cu un tutorial extins care ar trebui să faciliteze utilizarea lui de către dezvoltatorii cu experiență.

* **Ember.js**

Companii cunoscute precum Microsoft, Netflix și LinkedIn folosesc Ember.js, deoarece utilizează modelul Model-View-View-Model (MVVM) și încorporează cele mai bune practici ca parte a cadrului. Cel mai bun dintre toate este scalabilitatea. Interesant este că, în timp ce Ember.js se concentrează pe dezvoltarea web, se poate utiliza și pentru a construi aplicații mobile și desktop - a fost folosit pentru a construi Apple Music.

Spre deosebire de multe alte cadre, utilizatorul poate folosi suita completă de instrumente bazate pe Ember pentru a crea un mediu de dezvoltare compatibil. CLI Ember (interfața liniei de comandă) oferă acces la o multitudine de instrumente care se pot utiliza folosind scripturi. Ember Data, de exemplu, oferă acces la date web utilizând maparea relațională cu obiecte (ORM). Ember Inspector, între timp, este un plugin pentru browserele Firefox și Chrome, concepute pentru a facilita depanarea.

* **Node.js**

Node.js este un mediu de rulare JavaScript asincron, bazat pe evenimente, cu unele caracteristici unice, care execută cod JavaScript în afara unui browser web.

Acest produs seamănă mai mult cu jQuery decât cu jQueryUI, prin faptul că nu este folosit pentru a crea interfețe-utilizator, ci pentru a crea cod care reacționează la evenimente - un client generează evenimente, iar serverul răspunde la acele evenimente.

Cu Node.js, dezvoltatorii pot construi:

- aplicații backend

- blog-uri

- sisteme de management al clienților

- servicii în timp real cum ar fi aplicații de chat și jocuri

- API-uri REST

- aplicații de rețele sociale

- utilități și instrumente

* **Vue.js**

Scalând între o bibliotecă și un cadru complet, Vue utilizează un model de dezvoltare bazat pe componente.

Funcțiile Key Vue includ componente, șabloane, tranziții și legarea datelor în două sensuri, dar poate cea mai distinctivă caracteristică este sistemul său de „reactivitate”, prin care actualizarea unui obiect JavaScript în Vue actualizează automat și discret șabloanele Vue.

Vue a fost creat pentru a fi utilizat de oricine, cu cunoștințe de lucru HTML, CSS și JavaScript.