



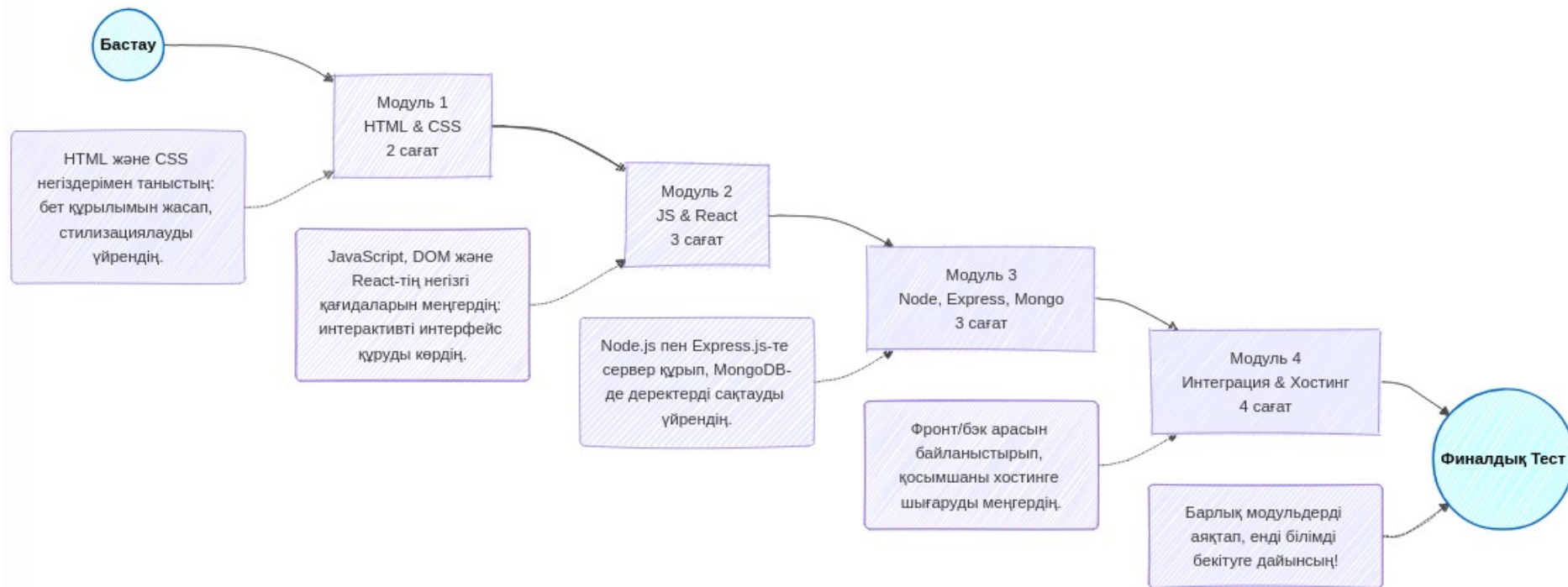
12 сағатта сайт жасауды үйренеміз

ҚМЖИ кафедрасының
оқытушысы Жумағалиев С.Қ.

Курстың жалпы құрылымы

- **1-модуль:** Веб-әзірлеуге кіріспе және MERN стекі (2 сағат)
 - MERN стекімен танысу және веб-әзірлеуге кіріспе
 - HTML және CSS негіздері
- **2-модуль:** Клиенттік бөлік (Frontend) — React (3 сағат)
 - JavaScript негіздері және DOM манипуляциясы
 - React арқылы клиенттік бөлікті әзірлеу
- **3-модуль:** Серверлік бөлік (Backend) — Node.js және Express.js (3 сағат)
 - Node.js және Express.js арқылы серверлік бөлікті құру
 - MongoDB мәліметтер базасымен жұмыс істеу
- **4-модуль:** MERN стегін интеграциялау және веб-қосымшаны орналастыру (4 сағат)
 - Клиенттік және серверлік бөліктерді біріктіру
 - Веб-қосымшаны серверге орналастыру және жобаны аяқтау

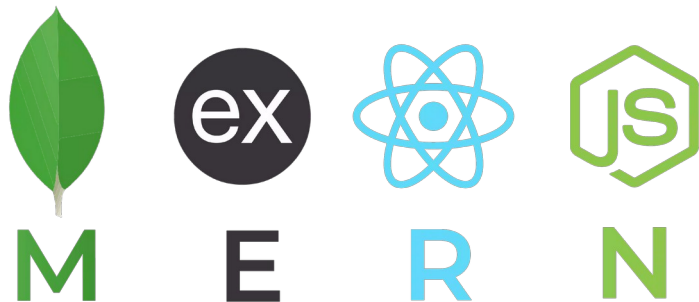
Курс өту жолы (roadmap)



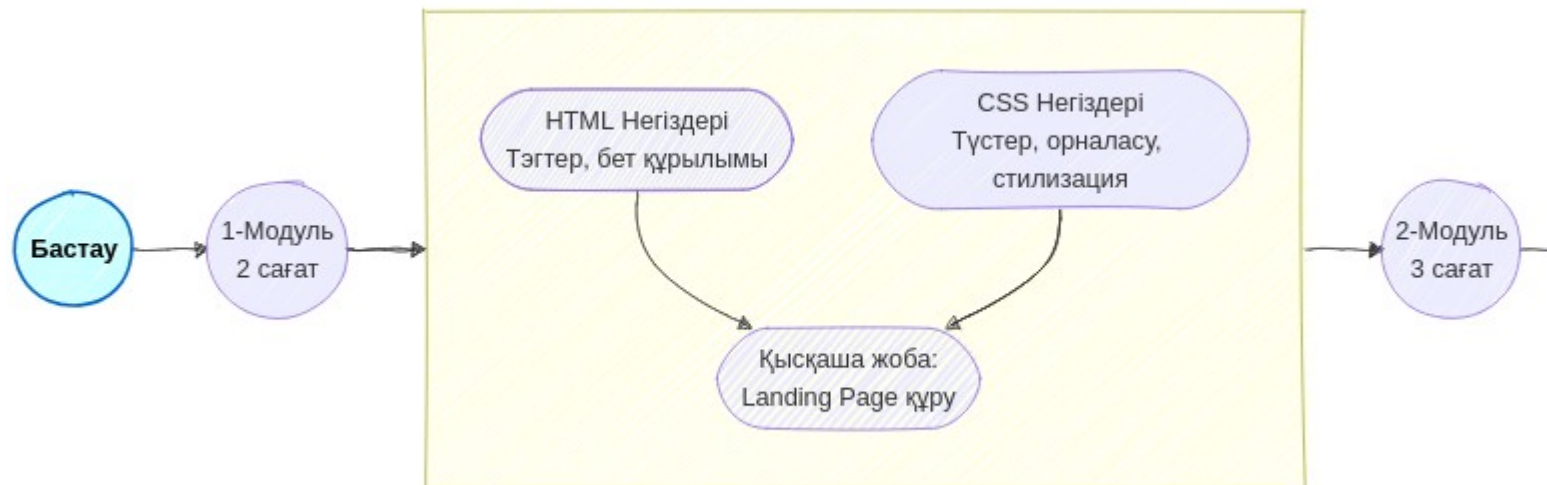
1-модуль

Тақырыптары

- MERN стегімен танысу және веб-әзірлеуге кіріспе
- HTML және CSS негіздері

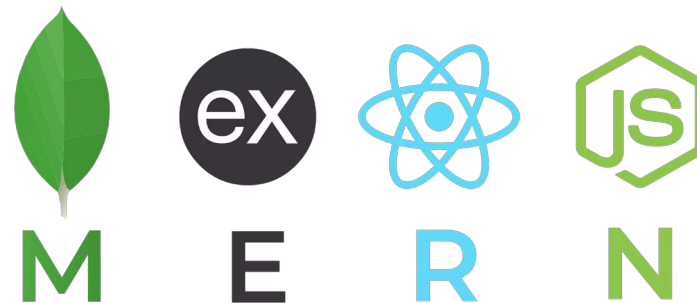


1-модуль өту жолы (roadmap)



MERN стегімен танысу және веб-әзірлеуге кіріспе

MERN стегімен танысу – қазіргі заманғы веб-әзірлеудің негізгі құралдарын меңгеру үшін маңызды қадам. MERN стегі төрт негізгі технологияны біріктіреді: MongoDB (мәліметтер базасы), Express.js (серверлік жақтауы), React.js (пайдаланушы интерфейсін құру) және Node.js (серверлік бағдарламалау). Бұл технологиялар бірлесіп жұмыс істеп, әзірлеушіге толыққанды веб-қосымшаларды тиімді әрі жылдам жасауға мүмкіндік береді. MERN стегін үйрену арқылы сіз динамикалық веб-қосымшалар мен күрделі серверлік құрылымдарды құру дағдыларын игеріп, нарықта жоғары сұранысқа ие маман бола аласыз.



MERN стегі деген не?

- Аббревиатура: MongoDB, Express.js, React, Node.js
- Толық мағынасы: JS негізінде фронтенд, бэкенд және дерекқорды бір жүйеде біріктіру
- MERN стек компоненттері:
 - **MongoDB:** NoSQL мәліметтер базасы
 - **Express.js:** Node.js ортада жұмыс істейтін серверлік фреймворк
 - **React:** Клиенттік жағын құруға арналған JavaScript кітапханасы
 - **Node.js:** JavaScript орындау ортасы (серверлік платформа)

Веб-әзірлеуге кіріспе

- Клиент–сервер архитектурасы

- Клиент–сервер архитектурасы – веб-қосымшалардың жұмыс істеу негізі. Бұл модельде клиент (пайдаланушының құрылғысы) серверге сұраныстар жіберіп, жауап алады. Сервер ақпаратты өңдейді, сақтайды және клиентке керекті деректерді қайтарады. Осы архитектура қолданушылар мен серверлер арасында тиімді байланыс орнатуға мүмкіндік береді.

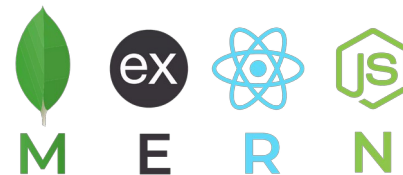
- HTTP сұраныс/жауабы

- HTTP протоколы арқылы клиент пен сервер арасында мәлімет алмасу жүзеге асырылады. Клиент серверге GET, POST, PUT, DELETE сияқты сұраныстар жібереді. Сервер сұранысты өңдеп, жауап ретінде деректер немесе хабарламалар жібереді (мысалы, 200 OK немесе 404 Not Found). Бұл механизм веб-қосымшалардың дұрыс жұмыс істеуі үшін негізгі болып табылады.

- Веб-қосымшаның қарапайым құрылымы

- Қарапайым веб-қосымша үш негізгі қабаттан тұрады: клиенттік, серверлік және деректер базасы. Клиенттік қабат пайдаланушы интерфейсін көрсетіп, серверге сұраныстар жібереді. Серверлік қабат бизнес-логиканы орындайды және деректер базасымен жұмыс істейді. Деректер базасы барлық қажетті ақпаратты сақтап, сервер сұрауы бойынша мәліметтерді қайтарады.

Салыстыру мысалы



End User with
Browser



Request
→
←
Response

API



Server Back-end
System



Customer

Make the
Order
→
←
Delivery of
order



Waiter

Take the
Order
→
←
Bringing
from Kitchen



Chef

HTML-дің негізгі құрылымы



The screenshot shows a code editor with a tab labeled 'index.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   Сәлем әлем!
10 </body>
11 </html>
12
```

- <!DOCTYPE html>
- <html>, <head>, <body> тегтері
- Мета-тегтер, <title>, <link> сияқты элементтер

HTML тегтерінің мысалдары

```
<h1>Сәлем әлем!</h1>
<p>мынау қарапайым сөйлем</p>
<a href="/index.html">Негізгі бет</a>
<img src="" alt="Мына жерде бейне тұру керек">
<ul>
  <li>бірінші</li>
  <li>екінші</li>
  <li>үшінші</li>
</ul>
<ol>
  <li>элемент</li>
  <li>элемент</li>
  <li>элемент</li>
</ol>
```

Сәлем әлем!

мынау қарапайым сөйлем

[Негізгі бет](#) Мына жерде бейне тұру керек

- бірінші
- екінші
- үшінші

1. элемент
2. элемент
3. элемент

CSS негіздері

- Стиль қосу жолдары: Inline, Internal, External
- Синтаксис: селектор { қасиет: мән; }
- Жиі қолданылатын қасиеттер:
 - color, background-color, font-size, margin, padding және т.б.

```
css style.css > ...  
1 .main-page-heading {  
2     color:  chartreuse;  
3     font-weight: 900;  
4     font-size: large;  
5 }  
6
```

```
<h1 class="main-page-heading">  
    Сәлем әлем!  
</h1>
```

Сәлем әлем!

мынау қарапайым сөйлем

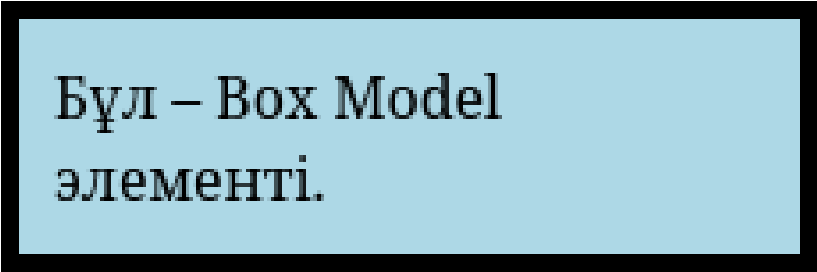
[Негізгі бет](#) Мына жерде бейне тұру керек

CSS позиционандырылу және layout

- Box model ұғымы (content, padding, border, margin)
 - Box model – HTML элементінің кеңістікте орналасу құрылымын анықтайтын негізгі модель. Ол төрт негізгі бөліктен тұрады: content (мазмұн), padding (мазмұн мен шекара арасындағы бос орын), border (элементтің шекарасы) және margin (элементтің сыртқы шекарасынан бос орын). Box model көмегімен әрбір элементтің нақты өлшемдерін есептеп, басқа элементтермен дұрыс жұмыс істеуге болады.
- Flexbox немесе Grid жүйелері
 - Flexbox – элементтерді икемді жол немесе баған бойымен орналастыруға мүмкіндік беретін жүйе. Ол динамикалық орналасуды жеңілдетіп, экран өлшеміне бейімделуді қамтамасыз етеді. Grid – екі өлшемді тор жүйесі, ол элементтерді қатар мен баған бойымен ұйымдастыруға мүмкіндік береді. Екі жүйе де күрделі layout құру үшін өте қолайлы және қолданушыларға заманауи интерфейстер жасауға көмектеседі.
- Элементтерді парақта дұрыс орналастыру
 - Элементтерді дұрыс орналастыру үшін CSS қасиеттерін (position, display, align-items, justify-content) тиімді пайдалану қажет. Flexbox немесе Grid жүйелерін қолданып, элементтердің орналасуын симметриялы және пайдаланушыға ыңғайлы етуге болады. Сонымен қатар, медиасұраныстар арқылы орналасуды әртүрлі экран өлшемдеріне бейімдеу маңызды. Бұл тәсіл веб-қосымшаны заманауи және жауап беретін етіп жасауға көмектеседі.

Box model ұғымы (content, padding, border, margin)

```
<div  
  style="width: 200px;  
  padding: 10px;  
  border: 5px solid ■ black;  
  margin: 20px;  
  background-color: □ lightblue;">  
  Бұл □ Box Model элементі.  
</div>
```



Бұл – Box Model
элементі.

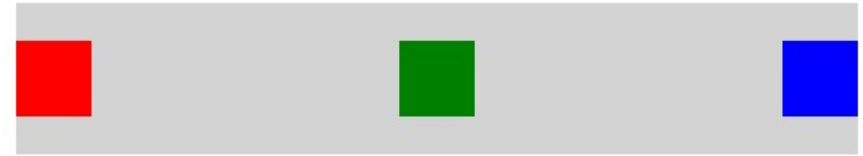
- Түсініктеме:
 - width: 200px – контенттің ені.
 - padding: 10px – мазмұн мен шекара арасындағы бос орын.
 - border: 5px solid black – шекара ені мен стилі.
 - margin: 20px – элементтің сыртқы бос орны.

Flexbox немесе Grid жүйелері

```
<div
  style="display: flex;
    justify-content: space-between;
    align-items: center;
    height: 100px;
    background-color: lightgray;">
  <div
    style="width: 50px;
      height: 50px;
      background-color: red;"></div>
  <div
    style="width: 50px;
      height: 50px;
      background-color: green;"></div>
  <div
    style="width: 50px;
      height: 50px;
      background-color: blue;"></div>
</div>
```

Сәлем әлем!

Бұл – Box Model
элементі.

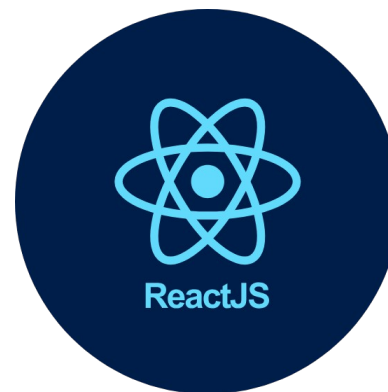
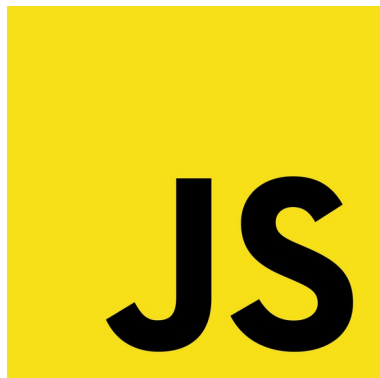


- Түсініктеме:
 - display: flex – Flexbox қолдану.
 - justify-content: space-between – элементтер арасында бос орын қалдыру.
 - align-items: center – элементтерді тік ось бойынша орталау.

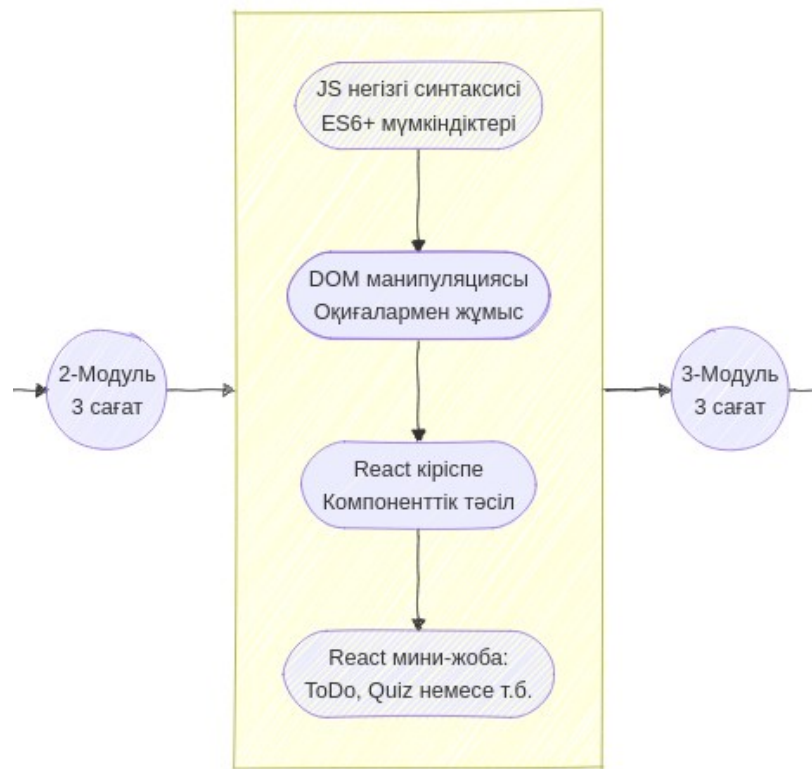
2-модуль

Тақырыптары

- JavaScript негіздері және DOM манипуляциясы
- React арқылы клиенттік бөлікті әзірлеу



2-модуль өту жолы (roadmap)



JavaScript негіздері

- Айнымалылар: var, let, const
 - JavaScript-те айнымалыларды var, let және const көмегімен анықтайды. var – глобалдық немесе функция деңгейінде қолданылады, бірақ блок скоупты қолдамайды. let – блок деңгейінде қолданылады және қайта анықталуы мүмкін. const – тұрақты мәндерді сақтау үшін пайдаланылады және тек анықталған кезде ғана мән беріледі. Осы айнымалыларды дұрыс қолдану кодтың тиімділігі мен қауіпсіздігін арттырады.
- Мәліметтер типтері: Number, String, Boolean, Object, т.б.
 - JavaScript мәліметтер типтері динамикалық және оларды өзгерту оңай. Number – сандарды сақтайды, String – мәтіндер үшін, Boolean – ақиқат немесе жалған мәндерге арналған. Object – күрделі деректер құрылымдарын сақтауға мүмкіндік береді, ал Array – деректердің тізімін сақтау үшін қолданылады. Бұл типтер әртүрлі есептерді шешуге мүмкіндік береді.
- Операторлар: +, -, *, /, %, ++, --, &&, ||
 - Операторлар арифметикалық, логикалық және салыстыру операцияларын орындау үшін қолданылады. Арифметикалық операторлар (+, -, *, /, %) сандармен жұмыс істеу үшін, ал инкремент (++) пен декремент (--) санды арттыру немесе азайту үшін қолданылады. Логикалық операторлар (&&, ||) шарттарды тексеріп, сәйкесінше әрекет етуге көмектеседі.
- Блок скоуп, функция анықтау (function, arrow function)
 - Блок скоуп айнымалыларды тек сол блок ішінде пайдалануға мүмкіндік береді. Қарапайым функцияларды function кілт сөзімен анықтайды, ал қысқа синтаксис үшін arrow function пайдаланылады. Бұл функциялар кодты қайта қолдануды және құрылымдауды жеңілдетеді, бағдарламаның оқылуын арттырады.

Айнымалылар: var, let, const

JS index.js > ...

```
1 // var - глобалды немесе функция деңгейінде
2 var globalVariable = "Бұл var айнымалысы";
3
4 // let - блок деңгейінде
5 let blockScopedVariable = "Бұл let айнымалысы";
6
7 // const - тұрақты мән
8 const constantVariable = "Бұл const айнымалысы";
9
10 console.log(globalVariable);
11 console.log(blockScopedVariable);
12 console.log(constantVariable);
13
```



Inspector



Console



Filter Output

Бұл var айнымалысы

Бұл let айнымалысы

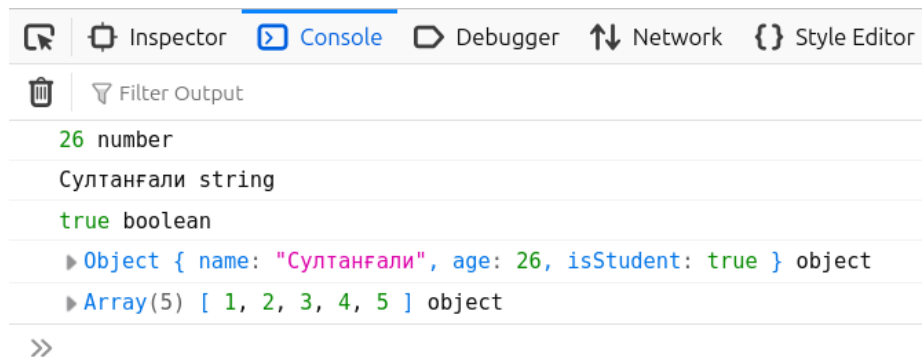
Бұл const айнымалысы



Мәліметтер типтері: Number, String, Boolean, Object, т.б.




JS index.js > ...



```
1 // Number
2 let age = 26;
3 // String
4 let name = "Султанғали";
5 // Boolean
6 let isStudent = true;
7 // Object
8 let user = {
9   name: "Султанғали",
10  age: 26,
11  isStudent: true
12 };
13 // Array
14 let numbers = [1, 2, 3, 4, 5];
15 console.log(age, typeof age);
16 console.log(name, typeof name);
17 console.log(isStudent, typeof isStudent);
18 console.log(user, typeof user);
19 console.log(numbers, typeof numbers);
20
```



*Операторлар: +, -, , /, %, ++, --, &&, ||

```
JS index.js > ...
1 // Арифметикалық операторлар
2 let a = 10;
3 let b = 3;
4 console.log("Қосу:", a + b); // 13
5 console.log("Азайту:", a - b); // 7
6 console.log("Көбейту:", a * b); // 30
7 console.log("Бөлу:", a / b); // 3.333
8 console.log("Қалдық:", a % b); // 1
9 // Инкремент және декремент
10 a++;
11 console.log("Инкремент:", a); // 11
12 b--;
13 console.log("Декремент:", b); // 2
14 // Логикалық операторлар
15 let isTrue = true;
16 let isFalse = false;
17 console.log("AND:", isTrue && isFalse); // false
18 console.log("OR:", isTrue || isFalse); // true
19
```

 Inspector  Console  Debugger

  Filter Output

Қосу: 13
Азайту: 7
Көбейту: 30
Бөлу: 3.3333333333333335
Қалдық: 1
Инкремент: 11
Декремент: 2
AND: false
OR: true

>>

Блок скоуп, функция анықтау (function, arrow function)

```
JS index.js > ...
1  // Блок скоуп
2  if (true) {
3      let scopedVariable = "Бұл блок скоуп ішінде";
4      console.log(scopedVariable);
5  }
6  // console.log(scopedVariable);
7  // Қате: scopedVariable анықталмаған
8  // Қарапайым функция
9  function greet(name) {
10     return `Сәлем, ${name}!`;
11 }
12 console.log(greet("Султанғали"));
13 // Arrow функция
14 const add = (x, y) => x + y;
15 console.log("Қосынды:", add(5, 7));
16
```



Inspector



Console



Filter Output

Бұл блок скоуп ішінде

Сәлем, Султанғали!

Қосынды: 12



Басқару құрылымдары

- Шартты операторлар: if...else, switch
 - if...else операторы белгілі бір шартты тексеріп, сәйкес әрекеттерді орындауға мүмкіндік береді. Егер шарт орындалса, бір код бөлігі орындалады, ал егер орындалмаса, басқа бөлігі орындалады. switch операторы бірнеше шарттарды тексеруге ыңғайлы және әртүрлі мәндер үшін сәйкес әрекеттерді орындауға мүмкіндік береді. Бұл операторлар қолданушылардың енгізген деректерін тексеру немесе бағдарлама логикасын басқару үшін өте тиімді.
- Циклдар: for, while, do...while
 - for циклі белгілі бір қадаммен қайталанатын әрекеттер үшін қолданылады. while циклі, шарт орындалғанша қайталанып, әрекеттерді орындайды. do...while циклі шартты тексермес бұрын кемінде бір рет орындалады. Бұл циклдар массивтермен жұмыс істеу, деректерді өңдеу және есептер шығару кезінде кеңінен қолданылады.
- Мысалдар: массивті қайталап шығу, есептер шығару
 - Массив элементтерін қайталап шығу үшін for немесе forEach пайдаланылады. Мысалы, массивтегі сандардың қосындысын табу үшін for циклі қолданылады. while циклі есептер шығаруда, мысалы, санның факториалын табу үшін жиі қолданылады. Бұл конструкциялар бағдарламалауда күрделі есептерді оңай шешуге мүмкіндік береді.

Шартты операторлар: if...else, switch

JS index.js > ...

```
1  let number = 10;
2
3  if (number > 0) {
4    console.log("Сан оң мәнді.");
5  } else if (number < 0) {
6    console.log("Сан теріс мәнді.");
7  } else {
8    console.log("Сан нөлге тең.");
9  }
10
```



Inspector



Console



Filter Output

Сан оң мәнді.



Шартты операторлар: switch

JS index.js > ...

```
1  let day = 3;
2  switch (day) {
3    case 1:
4      console.log("Дүйсенбі");
5      break;
6    case 2:
7      console.log("Сейсенбі");
8      break;
9    case 3:
10     console.log("Сәрсенбі");
11     break;
12    case 4:
13     console.log("Бейсенбі");
14     break;
15    case 5:
16     console.log("Жұма");
17     break;
18    default:
19     console.log("Демалыс күні");
20  }
```



Inspector



Console



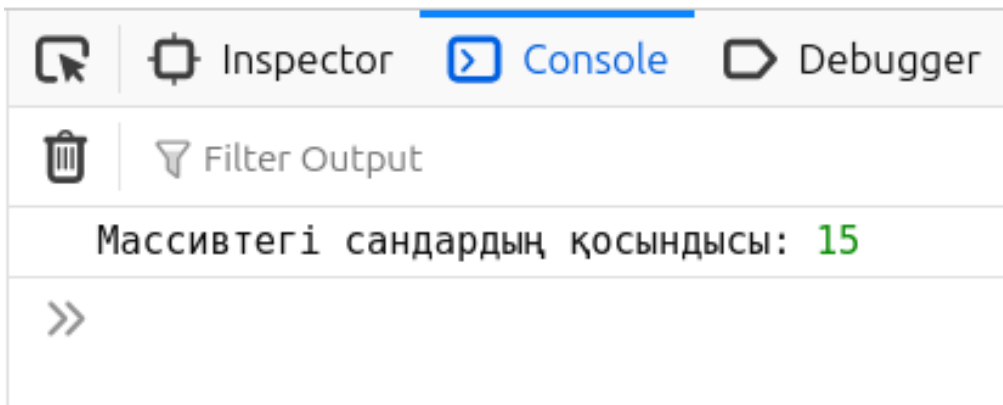
Filter Output

Сәрсенбі



Цикл: for

```
JS index.js > ...
1  let numbers = [1, 2, 3, 4, 5];
2  let sum = 0;
3
4  for (let i = 0; i < numbers.length; i++) {
5    sum += numbers[i];
6  }
7
8  console.log("Массивтегі сандардың қосындысы:", sum);
9
```



Цикл: while

JS index.js > ...

```
1  let counter = 5;  
2  
3  while (counter > 0) {  
4    console.log("Counter:", counter);  
5    counter--;  
6  }  
7
```



Inspector



Console



Filter Output

Counter: 5

Counter: 4

Counter: 3

Counter: 2

Counter: 1



Цикл: do...while

```
JS index.js > ...  
1  let count = 0;  
2  
3  do {  
4    console.log("Count:", count);  
5    count++;  
6  } while (count < 5);  
7
```



Inspector



Console



Filter Output

Count: 0

Count: 1

Count: 2

Count: 3

Count: 4



Массивті қайталап шығу: forEach

JS index.js > ...

```
1  let fruits = ["Алма", "Банан", "Шие"];
2
3  fruits.forEach((fruit, index) => {
4    |    console.log(`${index + 1}: ${fruit}`);
5  });
6
```



Inspector



Console



Filter Output

1: Алма

2: Банан

3: Шие



DOM манипуляциясы

- `document.getElementById()`, `document.querySelector()`
 - DOM (Document Object Model) элементтерімен жұмыс істеу үшін `document.getElementById()` және `document.querySelector()` әдістері кеңінен қолданылады. `getElementById()` – элементті оның `id` атрибуты арқылы табуға мүмкіндік береді. Ал `querySelector()` CSS селекторлары арқылы кез келген элементті таңдап, оны икемді түрде басқаруға мүмкіндік береді. Бұл әдістер веб-қосымшаның элементтерін жылдам тауып, олардың қасиеттерін өзгертуге мүмкіндік береді.
- Элементтің қасиеттерін өзгерту (`element.innerHTML`, `element.style`)
 - Элементтің мазмұнын өзгерту үшін `element.innerHTML` қолданылады. Бұл әдіс элементтің ішіндегі мәтінді немесе HTML құрылымын өзгертуге мүмкіндік береді. Ал `element.style` көмегімен элементтің CSS қасиеттерін динамикалық түрде өзгертуге болады, мысалы, түсін, өлшемін немесе орнын. Бұл қасиеттер интерактивті интерфейстер құру кезінде өте пайдалы.
- Оқиғалар (Events): `onclick`, `onchange`, `onmouseover`, т.б.
 - JavaScript оқиғаларға (events) жауап беруге мүмкіндік береді. `onclick` – пайдаланушы элементті басқанда орындалатын оқиға. `onchange` – форма өрістеріндегі мән өзгергенде іске қосылады. `onmouseover` – пайдаланушы элементтің үстіне меңзерді жылжытқанда орындалады. Оқиғалар веб-қосымшаның интерактивтілігін арттырып, пайдаланушы әрекеттеріне сәйкес жауап қайтаруға көмектеседі.

document.getElementById(), document.querySelector()

```
index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <h1 id="title">Бұл тақырып</h1>
5      <p class="paragraph">Бұл параграф</p>
6      <button onclick="changeContent()">Мазмұнды өзгерту</button>
7      <script>
8          function changeContent() {
9              // getElementById арқылы элементті табу
10             let title = document.getElementById("title");
11             title.innerHTML = "Тақырып өзгерді!";
12             // querySelector арқылы элементті табу
13             let paragraph = document.querySelector(".paragraph");
14             paragraph.innerHTML = "Параграфтың мазмұны жаңартылды.";
15         }
16     </script>
17 </body>
18 </html>
19
```

Бұл тақырып

Бұл параграф

Мазмұнды өзгерту

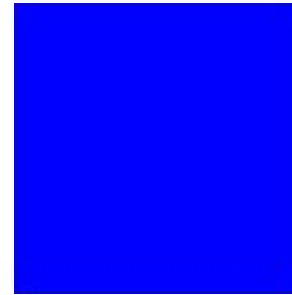
Тақырып өзгерді!

Параграфтың мазмұны жаңартылды.

Мазмұнды өзгерту

Элементтің қасиеттерін өзгерту (element.innerHTML, element.style)

```
index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <body>
4    <div id="box"
5      style="width: 100px;
6        height: 100px;
7        background-color: blue;"></div>
8    <button onclick="changeStyle()">Стильді өзгерту</button>
9    <script>
10     function changeStyle() {
11       let box = document.getElementById("box");
12       // Элементтің стилін өзгерту
13       box.style.backgroundColor = "red";
14       box.style.width = "150px";
15       box.style.height = "150px";
16       // Мазмұнын өзгерту
17       box.innerHTML = "Өзгерді!";
18       box.style.color = "white";
19       box.style.textAlign = "center";
20       box.style.lineHeight = "150px";
21     }
22   </script>
23 </body>
24 </html>
```



Стильді өзгерту



Стильді өзгерту

Оқиғалар (Events): onclick, onchange, onmouseover

```
index.html > html > body > script > showValue
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <input type="text" id="inputField"
5          placeholder="Атыңызды жазыңыз"
6          onchange="showValue()" />
7      <button id="clickButton" onclick="buttonClicked()">Басыңыз</button>
8      <div id="hoverBox" onmouseover="hoverEffect()"
9          style="width: 100px; height: 100px; background-color: yellow;"></div>
10     <p id="output"></p>
11     <script>
12         // onchange оқиғасы
13         function showValue() {
14             let inputField = document.getElementById("inputField");
15             document.getElementById("output").innerHTML = "Жазғаныңыз: "
16             + inputField.value;
17         }
18         // onclick оқиғасы
19         function buttonClicked() {
20             alert("Түйме басылды!");
21         }
22         // onmouseover оқиғасы
23         function hoverEffect() {
24             let hoverBox = document.getElementById("hoverBox");
25             hoverBox.style.backgroundColor = "green";
26         }
27     </script>
```



Жазғаныңыз: Султангали

React философиясы

- Компоненттік тәсіл: әр экран бөлігі жеке компонент ретінде жазылады
 - React компоненттік тәсілге негізделген, яғни әрбір интерфейс бөлігі жеке компонент ретінде жазылады. Әр компонент өзінің функционалдығы мен көрінісін анықтайды, бұл кодты қайта қолдануға және жобаның құрылымын оңай басқаруға мүмкіндік береді. Мысалы, батырма, форма немесе бүкіл бет компонент ретінде қарастырылады және оларды біріктіру арқылы толыққанды қосымша құрылады.
- props және state арқылы мәлімет алмасу
 - React-та мәлімет алмасу үшін props және state қолданылады. props – компонентке сырттан берілетін деректер, олар өзгермейді және тек оқуға арналған. state – компоненттің ішкі күйі, ол өзгертіліп, интерфейсін жаңарту үшін пайдаланылады. Бұл тәсіл компоненттер арасында деректерді тиімді тасымалдауға және қосымшаның интерактивтілігін арттыруға мүмкіндік береді.
- Virtual DOM және React-тің негізгі артықшылықтары
 - React Virtual DOM технологиясын қолданады, ол нақты DOM-ды өзгертпес бұрын оның жеңіл көшірмесінде барлық өзгерістерді есептеп, тек қажетті бөліктерді жаңартуға мүмкіндік береді. Бұл өнімділікті едәуір арттырып, қосымшаның жұмысын жылдамдатады. Сонымен қатар, React-тің компоненттік тәсілі мен бір жақты деректер ағыны кодты оңай тестілеуге және техникалық қызмет көрсетуге ықпал етеді.

React орнату

- `create-react-app` (немесе `Vite`) арқылы жобаны бастау
- `npm start` / `yarn start`: жергілікті серверде қолданбаны көру
- Жобаның құрылымдық папкалары: `src/`, `public/`, `package.json`

create-react-app (немесе Vite) арқылы жобаны бастау

```
ayangali@lenovo-legion-Y530-15ICH: ~/Desktop
File Edit View Search Terminal Help
ayangali@lenovo-legion-Y530-15ICH:~/Desktop$ npm create vite@latest
Need to install the following packages:
create-vite@6.1.1
Ok to proceed? (y) y

> npx
> create-vite

✓ Project name: ... SultangaliReactApp
✓ Package name: ... sultangalireactapp
✓ Select a framework: > React
✓ Select a variant: > JavaScript

Scaffolding project in /home/ayangali/Desktop/SultangaliReactApp...

Done. Now run:

  cd SultangaliReactApp
  npm install
  npm run dev

ayangali@lenovo-legion-Y530-15ICH:~/Desktop$
ayangali@lenovo-legion-Y530-15ICH:~/Desktop$
```

npm start / yarn start: жергілікті серверде қолданбаны көру

```
ayangali@lenovo-legion-Y530-15ICH: ~/Desktop/SultangaliReactApp
File Edit View Search Terminal Help
ayangali@lenovo-legion-Y530-15ICH:~/Desktop$
ayangali@lenovo-legion-Y530-15ICH:~/Desktop$ cd SultangaliReactApp
ayangali@lenovo-legion-Y530-15ICH:~/Desktop/SultangaliReactApp$ npm install

added 254 packages, and audited 255 packages in 3m

105 packages are looking for funding
  run `npm fund` for details

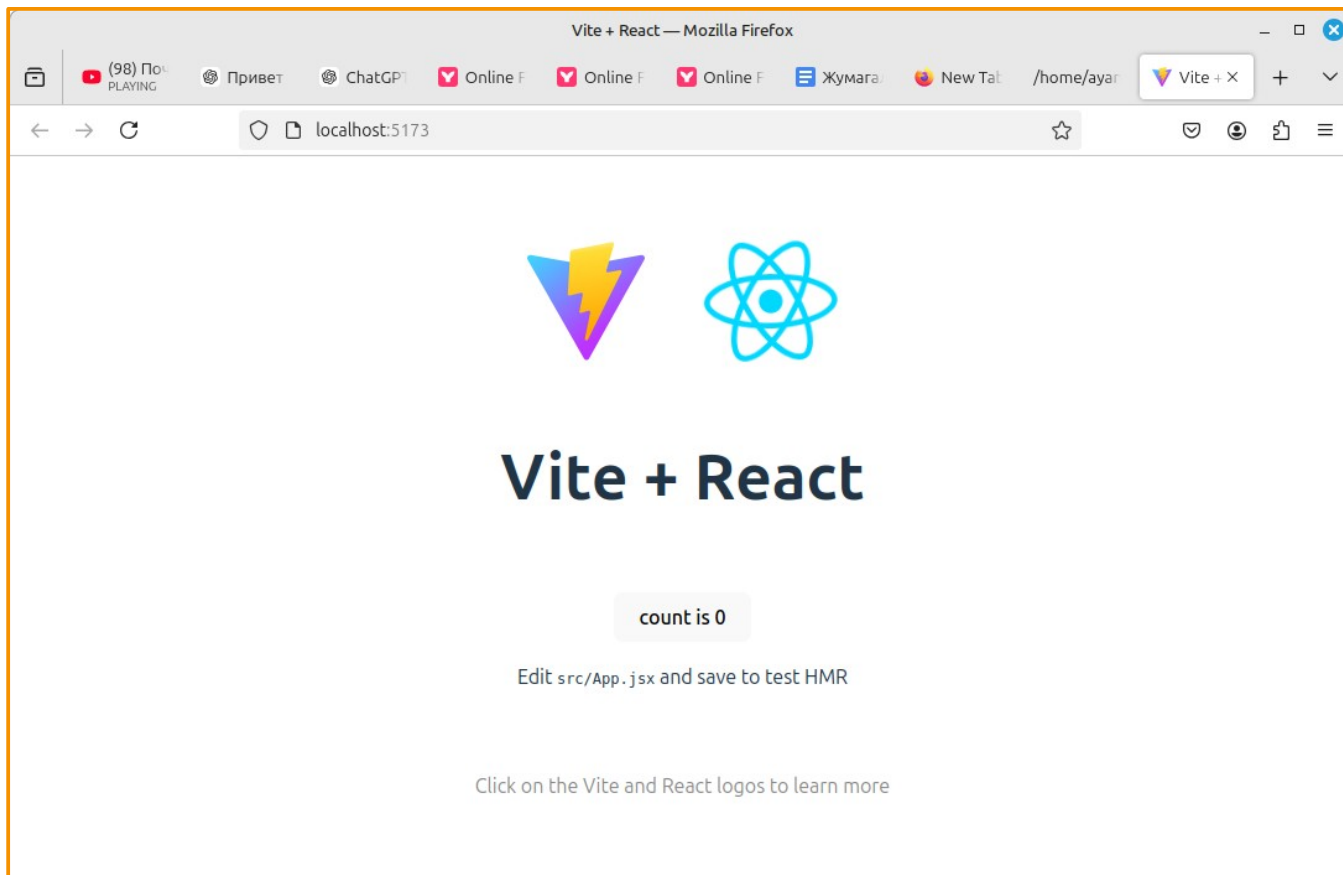
found 0 vulnerabilities
ayangali@lenovo-legion-Y530-15ICH:~/Desktop/SultangaliReactApp$ npm run dev

> sultangali-reactapp@0.0.0 dev
> vite

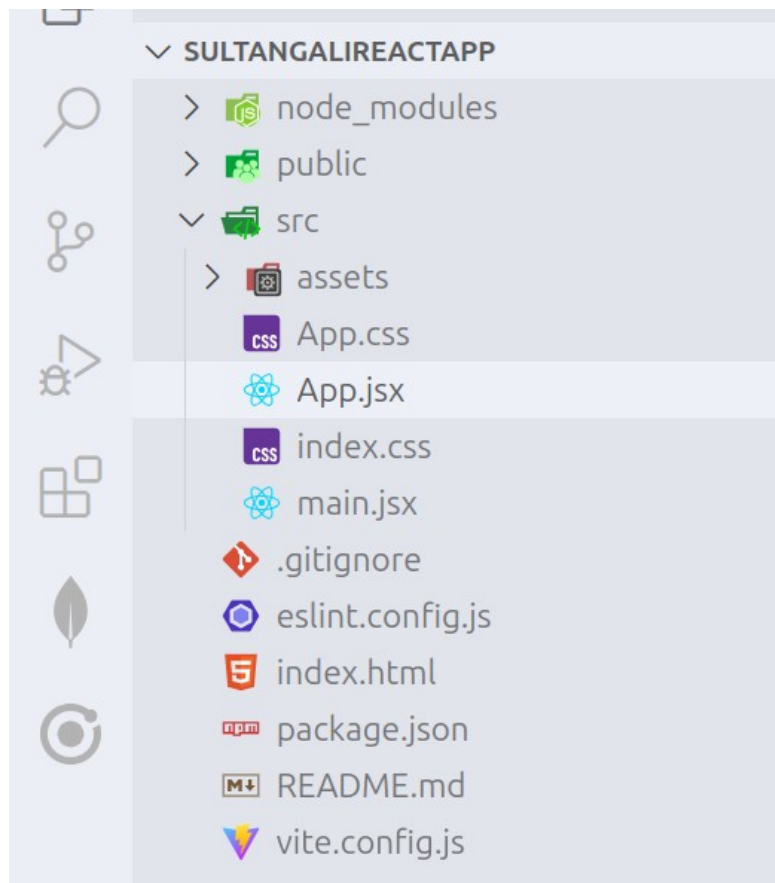
VITE v6.0.5 ready in 165 ms

→ Local:   http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

npm start / yarn start: жергілікті серверде қолданбаны көру



Жобаның құрылымдық папкалары: src/, public/, package.json



JSX түсінігі

- HTML мен JavaScript-ті бірге қолдану
- Компонент қайтарған кезде <div>, <p> сияқты тегтерді пайдалану
- Қарапайым мысал:

```
function Hello() {  
  |   return <h1>Сәлем!</h1>  
}
```


props және state ұғымдары

- props: компонентке сырттан берілетін деректер
- state: компоненттің ішкі күйі
- useState hook қолдану мысалы:

```
function App() {  
  const [count, setCount] = useState(0)
```

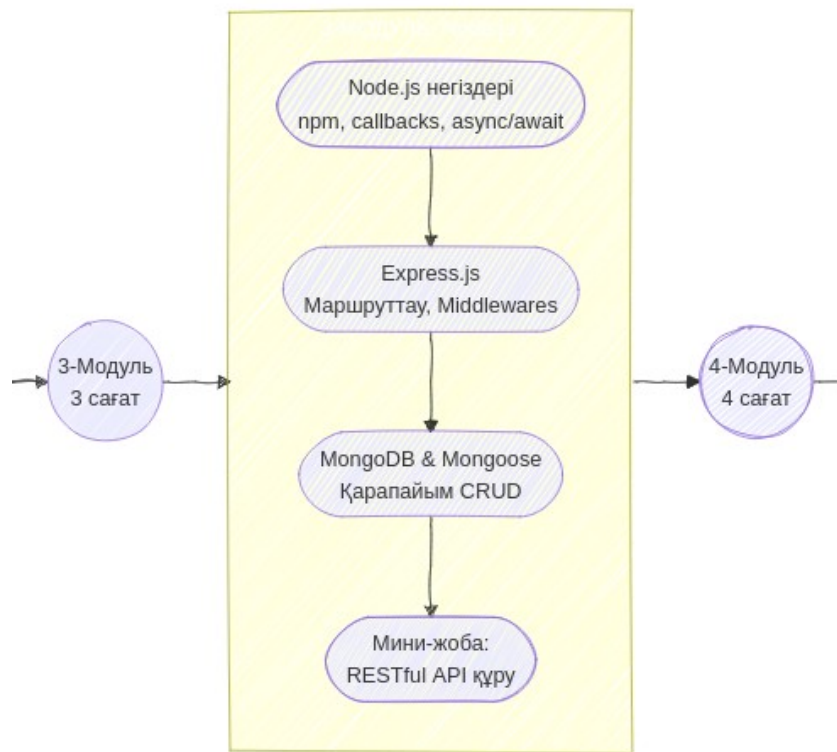
3-модуль

Тақырыптары

- Node.js және Express.js арқылы серверлік бөлікті құру
- MongoDB мәліметтер базасымен жұмыс істеу



3-модуль өту жолы (roadmap)



Node.js негіздері

- Node.js-тың шығу тарихы, артықшылықтары
 - Node.js 2009 жылы Райан Дальдың бастамасымен JavaScript-ті серверлік ортада қолдану үшін жасалды. Ол Chrome V8 қозғалтқышына негізделген және оқиғаларға негізделген, асинхронды енгізу-шығару үлгісін қолданады. Node.js-тің басты артықшылығы – жоғары өнімділік және масштабтылық, өйткені ол блокталмайтын операциялар мен бір ағынды архитектураны қолданады. Сонымен қатар, ол бір тілде (JavaScript) жазуға мүмкіндік береді, бұл фронтенд пен бэкендті біріктіруге жағдай жасайды.
- npm арқылы пакеттер орнату
 - Node.js-пен бірге npm (Node Package Manager) орнатылады, ол әлемдегі ең үлкен JavaScript пакеттер репозиторийі болып табылады. npm install командасы арқылы кез келген пакетті оңай орнатуға болады. Мысалы, npm install express командасы Express.js фреймворкін жобада қолдануға мүмкіндік береді. Бұл құрал әзірлеушілерге үшінші тарап кітапханаларын пайдалану және өз пакеттерін басқару мүмкіндігін береді.
- require/import синтаксисі
 - Node.js-та модульдерді қосу үшін екі түрлі синтаксис қолданылады: require және import. require – CommonJS синтаксисіне жатады және Node.js-тың алғашқы нұсқаларында пайдаланылды, мысалы:

```
server > JS index.js > ...
1  import express from 'express'
2
3  const express = require('express')
4
```
 - import – ES6 стандартын қолдайтын заманауи синтаксис. Ол ECMAScript модульдері үшін қолданылады, бірақ оны қолдану үшін конфигурация қажет болуы мүмкін:

Express.js фреймворкі

- Орнату: `npm install express`
 - Express – бұл Node.js негізінде серверлік қосымшаларды құруға арналған жеңіл және икемді фреймворк. Оны пайдалану үшін алдымен орнату қажет: `npm install express`. Бұл команда Express-ті жобаға қосып, оны `node_modules` папкасында сақтайды, әрі пайдалануға мүмкіндік береді. Express веб-қосымшаларды, API-ларды және серверлерді тез әрі оңай құруға арналған.
- Негізгі ұғымдар: `app`, `router`, `middleware`
 - Express-та `app` – қосымшаның негізгі объектісі, ол серверді басқарып, маршруттарды, `middleware`-ді және басқа функционалдарды анықтайды. `Router` – маршруттарды басқаруға арналған модуль, ол маршруттарды топтастыруға және ұйымдастыруға мүмкіндік береді. `Middleware` – сұраныс пен жауап арасында орындалатын функциялар. Олар деректерді өңдеу, аутентификация, немесе логгерлер сияқты қосымша функционалдарды орындау үшін пайдаланылады.
- Маршруттар: `app.get('/', ...)`, `app.post('/', ...)`
 - Маршруттар – клиенттің сұраныстарына жауап беру үшін анықталатын жолдар. `app.get('/')` – клиент серверге GET сұранысын жіберген кезде орындалады. Мысалы, басты бетке сұраныс жасағанда, сервер жауап қайтарады. `app.post('/')` – серверге POST сұранысын жіберу үшін қолданылады, ол әдетте формаларды жіберу немесе деректерді сақтау үшін пайдаланылады.

RESTful API концепциясы

server > route > JS user.routes.js > [x] default

```
1  import express from 'express'
2  import * as controller from '../controller/index.js'
3  import * as validation from '../service/validation.js'
4  import validationHandler from '../service/validationHandler.js'
5  import checkAuth from '../middleware/checkAuth.js'
6
7  const userRouter = express.Router()
8
9  userRouter.post('/auth/registration', controller.user.registration)
10 userRouter.post('/auth/login', validation.login, validationHandler,
11   controller.user.login)
12 userRouter.patch('/me/update', checkAuth, validation.updateProfile,
13   validationHandler, controller.user.update )
14 userRouter.get('/me', checkAuth, controller.user.me)
15 userRouter.get('/all-users', checkAuth, controller.user.all)
16
17 export default userRouter
```

MongoDB-ге кіріспе

- NoSQL базасының ерекшелігі
 - NoSQL мәліметтер базасы құрылымданбаған немесе жартылай құрылымданған деректерді сақтау үшін қолданылады. Ол реляциялық мәліметтер базасына тән кестелер мен жолдарды пайдаланудың орнына, деректерді икемді форматта (JSON, BSON) сақтайды. NoSQL базасының басты ерекшелігі – масштабталуы жоғары және деректердің өзгермелі құрылымын қолдауы. Бұл оны үлкен деректер мен жылдам жұмыс істеуді қажет ететін қосымшалар үшін тамаша таңдау етеді.
- Collection, document, field
 - MongoDB сияқты NoSQL базаларында деректер collection, document, және field түрінде сақталады. Collection – дәстүрлі реляциялық базадағы кестеге ұқсас, бірақ ол икемді құрылымға ие құжаттар жиынтығы. Document – деректерді JSON немесе BSON форматында сақтайтын бірлік, яғни жазба. Field – құжат ішіндегі жеке қасиет немесе мән.
- Локалдық немесе бұлтты MongoDB (Atlas) пайдалану
 - MongoDB-ны локалдық серверде немесе бұлтта пайдалануға болады. Локалдық MongoDB деректерді серверге тәуелсіз басқаруды қамтамасыз етеді және дамытушылар үшін ыңғайлы. Ал MongoDB Atlas – бұлтта орналасқан басқарылатын қызмет, ол мәліметтер базасының қауіпсіздігін, резервтік көшірмесін және масштабталуын оңай қамтамасыз етеді. Бұлттық шешімдер қосымшаны жылдам орналастыру мен қолжетімділікті арттыру үшін өте тиімді.

MongoDB мәліметтер базасымен жұмыс істеу

The screenshot displays the MongoDB Compass application window titled "MongoDB Compass - usersDB/conferenceDB". The interface includes a top menu bar with "Connections", "Edit", "View", and "Help". On the left, the "Compass" sidebar shows "My Queries" and a "CONNECTIONS (1)" section with a search bar and a tree view of databases: "usersDB" (containing "admin") and "conferenceDB" (containing "articles", "todos", "users", "config", "local", and "todoDB"). The main area shows the "conferenceDB" database selected, with a "Sort by" dropdown set to "Collection Name". It lists three collections: "articles", "todos", and "users", each with a table of statistics.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
articles	20.48 kB	26	200.00 B	2	73.73 kB
todos	4.10 kB	0	0 B	1	4.10 kB
users	20.48 kB	8	487.00 B	2	73.73 kB

MongoDB мәліметтер базасымен жұмыс істеу

```
export const login = async (req, res) => {
  try {
    const { login, password } = req.body;

    let user = "";

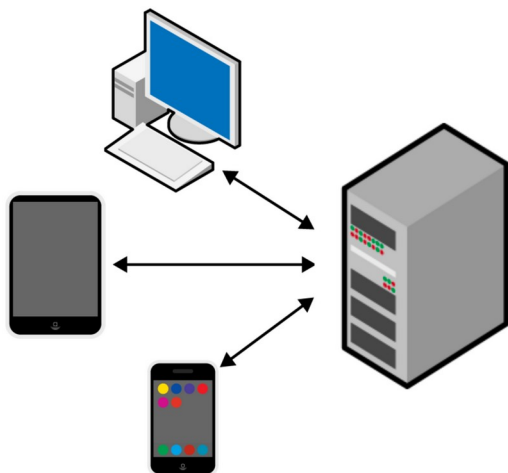
    const validateEmail = (email) => {
      return email.match(
        /^[^<>()[]\.\.,;:\s@"']+(\.[^<>()[]\.\.,;:\s@"']+)*)|(\".+\")@((\[[0-9-
      );
    };

    if (validateEmail(login)) {
      user = await User.findOne({
        email: login,
      });
      if (!user) {
        return res.status(404).json({
          message: `${login} желіде жоқ`,
        });
      }
    }
  }
}
```

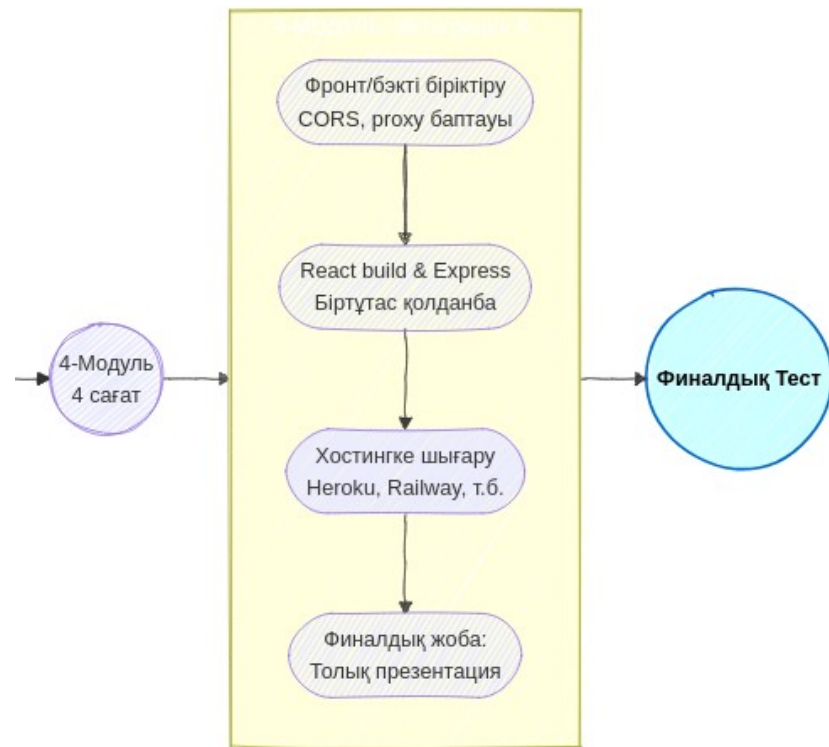
4-модуль

Тақырыптары

- Клиенттік және серверлік бөліктерді біріктіру
- Веб-қосымшаны серверге орналастыру және жобаны аяқтау



4-модуль өту жолы (roadmap)



MERN жобасын біріктірудің негізгі қадамдары

- React фронтендінің дайын build-нұсқасын алу
- Express серверінің public немесе build қалтасынан статикалық файлдарды тарату
- CORS немесе проху орнату

Жобаны build жасау (React)

- `npm run build`
- Build қалтасында HTML, CSS, JS-тің минификацияланған нұсқалары
- Express-ке кіріктіру (мысалы, `app.use(express.static('build'))`)

Node.js/Express-пен біріктіру архитектурасы

- Бір репозиторийде немесе екі бөлек репода сақтау
- Порттарды бөлу (React – 3000, Server – 3001) немесе бір портта біріктіру
- Қолданба логикасын бөлу: клиент жағы vs сервер жағы

Веб-қосымшаны хостингке шығару

- Google Cloud Console арқылы серверге жобаны көтеру

Quick setup — if you've done this kind of thing before

HTTPS

SSH

git@github.com:sultangali/123.git



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

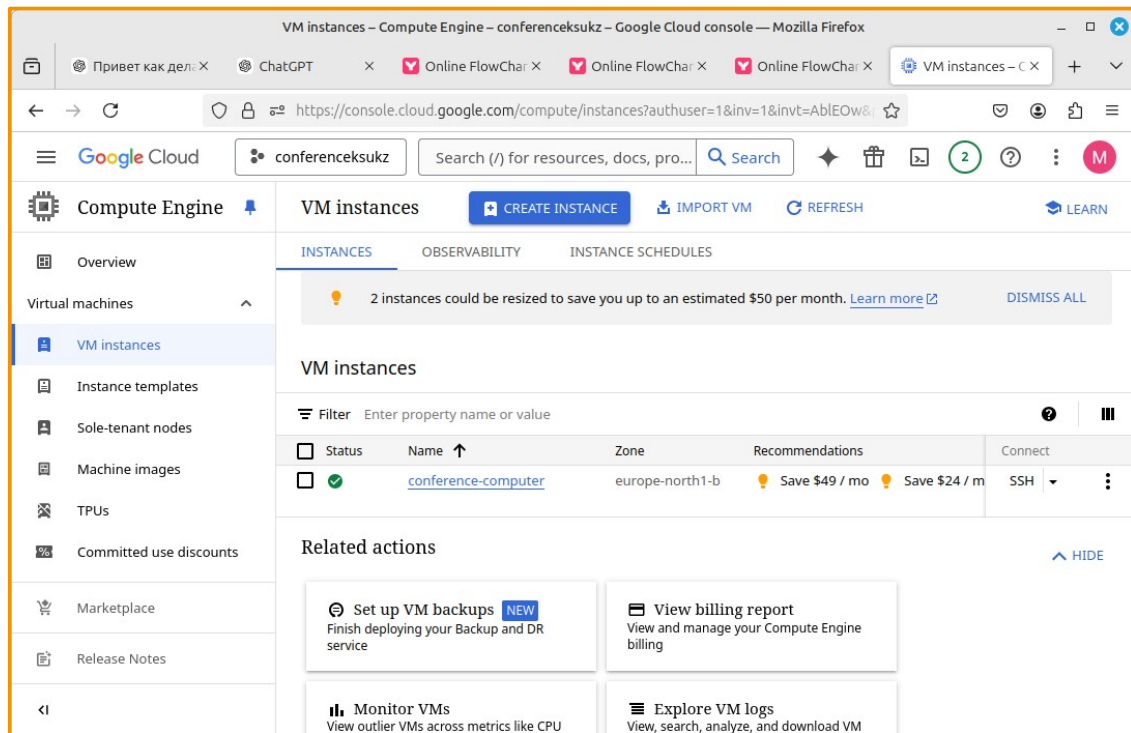
...or create a new repository on the command line

```
echo "# 123" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:sultangali/123.git
git push -u origin main
```



Веб-қосымшаны хостингке шығару

- Google Cloud Console арқылы серверге жобаны көтеру



КУРС СОҢЫНДАҒЫ ТЕСТ

Тестке дайындалу

12 сағаттық бағдарламаның мазмұнын қайталау

MERN стегінің негізгі ұғымдары:

MongoDB, Express, React, Node

HTML, CSS, JS, RESTful API, т.б.

Қосымша ресурстар

- [React ресми құжаттамасы](#)
- [Node.js](#)
- [Express.js](#)
- [MongoDB](#)
- [FreeCodeCamp](#)
- [MDN Web Docs](#)

Назарларыңызға рақмет!

- Курсты жүргізген: Жумагалиев С.Қ.
- Сұрақтарыңыз болса төмендегі сілтемелер арқылы хабарласа аласыздар:
 - +7 776 511 1441
 - sultangali.zhumagaliyev@gmail.com
 - <https://github.com/sultangali>