УДК 004.7
DOI 10.18101/978-5-9793-1497-6-110-119

РАЗРАБОТКА СИСТЕМЫ НАВИГАЦИИ САЙТА

© Эрдынеев Жаргал Буладович
студент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: ezha134@gmail.com

© Тонхоноева Антонида Антоновна
кандидат педагогических наук, доцент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: ant_ton@mail.ru

Сегодня невозможно представить развитие нашего общества без использования тех возможностей, которые предоставляет Интернет. Руководители компаний и представители бизнеса рекламируют свою продукцию не только традиционным способом, но и привлекают своих потенциальных клиентов через глобальную сеть. Использование web-сайтов в бизнесе позволяет быстро информировать о предлагаемых услугах и товарах. При разработке web-портала, как правило, предполагается, что ресурс будет состоять из нескольких web-страниц, при создании одностраничного сайта - из нескольких разделов. Из-за большого объема информации пользователю достаточно неудобно ориентироваться в ресурсе без специальных средств переходов по порталу, и, как результат, потенциальный клиент может покинуть web-сайт, не достигнув цели. Данную проблему можно решить, если разработать хорошо продуманную систему навигации сайта. Для создания навигационного меню использовались язык разметки HTML, каскадные таблицы стилей CSS, язык сценариев JavaScript.
Ключевые слова: Интернет, сайт, HTML, CSS, навигация, меню, контейнер.

Интернет прочно вошел в нашу жизнь, уже сложно представить развитие общества без использования тех возможностей, которые предлагает глобальная сеть. Использование web-сайтов в бизнесе позволяет быстро информировать о предлагаемых услугах, продвигать товары, привлечь большее число потребителей услуг. При разработке web-портала, как правило, предполагается, что ресурс будет состоять из нескольких web-страниц, при создании одностраничного сайта - из нескольких разделов. Из-за большого объема информации пользователю достаточно неудобно ориентироваться в ресурсе без специальных средств переходов по порталу, и, как результат, потенциальный клиент может покинуть web-сайт, не достигнув цели. Данную проблему можно решить, добавив хорошо продуманную систему навигации сайта. В качестве примера рассмотрим разработку сайта фотографа, содержащего разделы: «Главная», «Услуги», «Портфолио», «Обо мне», «Контакты».

Для создания навигационного меню используются три файла:

- файл, содержащий меню (index.html);
- файл, отвечающий за визуальный стиль меню (style.css);
- расширение для настройки отображения меню (app.js).

В файле index.html для создания навигационного меню нужно задать тег <section>, в котором для атрибута id задать любое значение, например, header. Применение данного тега поможет создать отдельный раздел документа, что поможет в настройке будущих элементов, относящихся к данному разделу.

Далее создаем контейнер, в котором будут определены элементы, изменяемые в дальнейшем в файле style.css. Создание контейнера происходит с помощью тега , также стоит добавить атрибут class со значением header container, контейнер будет отвечать за общее стилевое оформление.

```
<section id=”header”>
    <div class=”header container”>
      …
    </div>
</section>
```

Внутри него создаем ещё один контейнер, но с другим классом nav-bar. Также внутри него создаем ещё один контейнер с классом nav-list, в котором будет создан список разделов с помощью тега маркированного списка <ul>, а также кнопка вывод меню (hamburger и bar) для пользователей с малым экранным разрешением.

Для создания пунктов меню необходимо добавить элементы внутрь тега <ul>, которые будут представлять из себя ссылки разделов. Так как меню будет состоять из 5 разделов, необходимо разместить 5 тегов элемента списка <li>. После внутри тега <li> нужно написать наименования разделов.

С помощью тега <a> создается ссылка, по которой будет происходить переход пользователя в выбранный раздел. Данный тег нужно поместить в каждый элемент списка. Для того чтобы сделать ссылки рабочими, необходимо добавить к нашим тегам ссылки атрибут href. Данный атрибут будет содержать идентификатор каждого раздела. Наименования идентификатора выбираем исходя из наименований разделов.

```
<section id="header">
        <div class="header container">
                <div class="nav-list">
                        <div class="hamburger"><div class="bar"></div></div>
                        <ul>
                                <li><a href="#main-page">Главная</a></li>
                                <li><a href="#services">Услуги</a></li>
                                <li><a href="#projects">Портфолио</a></li>
                                <li><a href="#about">Обо мне</a></li>
                                <li><a href="#contact">Контакты</a></li>
                        </ul>
                        </div>
                </div>
        </div>
</section>
```

Файл style.css отвечает за стилизацию меню. Для начала настроим расположение навигационного меню используя свойства position, z-index, left, top, width и height.

```
#header {
```

```
        position: fixed;
        z-index: 111;
        left: 0;
        top: 0;
        width: 100vw;
        height: auto;
}
```

1. **position** – позволяет установить или изменить положение элемента: fixed – данное значение указывает, что элемент абсолютно позиционирован, но привязывается к указанному месту на экране и не позволяет элементу менять свое положение, если пользователь пролистывает web-страницу;
2. **z-index** – позволит расположить элемент выше или ниже любых других элементов web-сайта. Если данное свойство не будет указано, то другие элементы будут накладываться друг на друга в том порядке, в котором они описаны в коде html, поэтому зададим большое значение, например, 111, чтобы элемент отображался поверх других;
3. **left** – указывает направление смещения позиционированного элемента от левого края, со значением 0 он не будет смещаться;
4. top – указывает направление смещения позиционированного элемента от верхнего края, со значением 0 он не будет смещаться;
5. **width** – отвечает за ширину элемента. Задав значение 100 с единицей измерения vw – единица эквивалентная 1% от текущей ширины области отображения браузера, элемент станет отображаться на всю ширину в зависимости от разрешения экрана пользователя;
6. **height** – определяет высоту элемента: auto – данное значение задает высоту исходя из содержимого элемента.

В дальнейшем настраиваем само навигационное меню.

```
#header .header {
    min-height: 8vh;
    background-color: rgba(31, 30, 30, 0.24);
    transition: .3s ease background-color;
}
```

1. **min-height** – устанавливает минимальную высоту элемента контейнера. Задав значение 8 с единицей измерения vh, элемент станет отображаться в высоту занимая лишь 8 % высоты разрешения экрана пользователя;
2. **background-color** – устанавливает цвет фона элемента контейнера. Используя цветовую модель rgba, можно настроить цвет;
3. **transition** –возможность контролировать скорость анимации элемента: **.3s** – указывает время анимации;
4. **ease** – указывает, что скорость анимации вначале медленная, а к концу промежутка времени начнет ускоряться;
5. **background-color** – свойство, к которому идет переход, в этом случае цвет меню будет плавно изменяться к заранее заданному цвету.

Далее настраиваем расположение разделов для ПК версии браузера.

```
#header .nav-bar {
    display: flex;
    align-items: center;
```

```
    justify-content: space-between;
    width: 100%;
    height: 100%;
    max-width: 1300px;
    padding: 0 10px;
}
```

1. **display** – определяет визуальное отображение элементов контейнера на странице: **flex** – данное значение задает адаптивную компоновку элементов – делает все элементы гибкими/делает контейнер гибким;
2. **align-items** – выравнивает элементы контейнера в перпендикулярном направлении: **center** – данное значение выравнивает содержимое по центру контейнера;
3. **justify-content** – отвечает за распределение пространства между элементами контейнера: **space-between** – данное значение равномерно распределяет элементы по всей строке;
4. **width –** отвечает за ширину, в котором могут находиться элементы внутри контейнера, значение **100%** дает возможность расположить элементы по всей доступной ширине самой навигационного меню(**#header**), ширина которого настраивалась в начале;
5. **height –** определяет высоту, которую может занять контейнер, значение **100%** дает возможность расположить элементы по всей доступной высоте самого навигационного меню(**#header**), высоту которого мы настраивали в начале;
6. **max-width –** задает максимальную ширину контейнера. Значение **1300** означает максимальную ширину контейнера в **px**(пикселях);
7. **padding –** устанавливает внутренние отступы/поля со всех сторон контейнера. Первое значение – **0**, определяет поля сверху и снизу, второе значение – **10px**, определяет поля слева и справа.

Теперь настроим маркированный список контейнера, но большинство свойств будут использованы для работы с экранами малого разрешения.

```
#header .nav-list ul {
    list-style: none;
    position: absolute;
    background-color: rgb(31, 30, 30);
    width: 100vw;
    height: 100vh;
    left: 100%;
    top: 0;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    z-index: 1;
    overflow-x: hidden;
    transition: .5s ease left;
}
```

1. **list-style –** отвечает за отображение вида марки в списке. Значение **none** полностью убирает отображение марки;
2. **position.** Значение **absolute** задает абсолютное позиционирование элементов маркированного списка;
3. **background-color.** Данное свойство будет использоваться для работы с экранами малого разрешения, выступая цветовым фоном выводящего меню;
4. **width.** Список будет полностью будет занимать всю ширину отображения благодаря значению **100vw;**
5. **height.** Список будет полностью занимать всю высоту отображения благодаря значению **100** с единицей измерения **vh –** единица эквивалентная 1% от высоты области отображения браузера;
6. **left.** Так как у нас абсолютное позиционирование, то положение элемента будет определяться от его левого края на **100%;**
7. **top.** Со значением **0** будет исключать возможность смещения меню от верхнего края;
8. **display: flex**. Задает элементам гибкость/делает весь контейнер гибким;
9. **flex-direction –** указывает основное направление flex-элементов, а также их положение. Значение **column** – задает направление flex-элементов сверху вниз.
10. **justify-content.** Значение **center** располагает flex-элементы друг за другом по центру контейнера, то есть слева и справа от элементов расстояние до границ контейнера будет одинаковым;
11. **align-items: center.** Выравнивает элементы по центру контейнера;
12. **z-index: 1.** Задает позицию отображения элементов;
13. **overflow –** управляет отображением элементов контейнера, если размер содержимого превышает допустимую дину или ширину. Значение **hidden** означает что содержимое контейнера будет скрыто;
14. **transition: .5s ease left.** Отвечает за плавное появление меню, появление происходит справа налево в течении 0.5 секунды.

    Также укажем область, которая будет оставаться после вывода меню.

```
#header .nav-list ul.active {
   left: 0%;
}
```

- left: 0%. Означает что вывод меню будет идти до левого края.

Настроим стиль ссылок.

```
#header .nav-list ul a {
   font-size: 2.5rem;
   font-weight: 500;
   letter-spacing: .2rem;
   text-decoration: none;
   color: white;
   text-transform: uppercase;
   padding: 20px;
   display: block;
}
```

1. **font-size –** отвечает за размер шрифта. Значение **2.5** с единицей измерения **rem** – единица типографии, равная корневому значению font-size (данное

свойство прикреплено к тегу **html**, в котором размер шрифта составляет 10px);

2. **font-weight –** отвечает за насыщенность шрифта. Чем больше значение, тем жирнее получается шрифт;
3. **letter-spacing –** определяет межбуквенное расстояние в тексте. Значение обозначит интервал, который остается относительно размера шрифта;
4. **text-decoration –** отвечает за оформление текста т.е. его подчеркивание, перечеркивание или надчеркивание. Значение **none** полностью отменяет какие-либо оформления текста;
5. **color –** управляет цветом шрифта. Так как по умолчанию цвет шрифта черный, а фон навигационного меню темный, необходимо установить светлый цвет шрифта написав значение **white**;
6. **text-transform –** управляет преобразование текста элемента в заглавные или прописные символы. Значение **uppercase** означает что все символы становятся прописными;
7. **padding.** Установит отступ со всех сторон в размере **20px**;
8. **display.** Значение **block** означает что любые элементы становятся блочными, он используется для того чтобы увеличить зону щелчка;

Для того чтобы пользователь был уверен, что он выбрал нужный ему раздел, добавим эффект изменения цвета шрифта с помощью псевдокласса **hover**.

Следующим этапом является создание значка меню, который будет отображаться в том случае, если разрешение экрана или область отображения браузера будет минимальным.

```
#header .hamburger {
    height: 60px;
    width: 60px;
    display: inline-block;
    border: 3px solid crimson;
    border-radius: 50%;
    position: relative;
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 100;
    cursor: pointer;
    transform: scale(.8);
    margin-right: 20px;
}
```

1. **height: 60px.** Задаст высоту значка;
2. **width: 60.** Задаст ширину значка;
3. **display.** Значение inline-**block** генерирует блочный элемент который будет обтекаться с другими элеменами.
4. **border –** универсальное свойство, которое одновременно задает толщину, стиль и цвет границы вокруг элемента, в данном случае толщина границы равна 3 пикселям – **3px**, стиль границ будет нарисован сплошной линией – **solid**, а цвет линии багровый – **crimson**.
5. **border-radius –** позволяет задать угол скругления вокруг рамки элемента;

6. **position.** Значение **relative** дает возможность в дальнейшем задавать смещение относительно его изначальной позиции;
7. **display: flex.** Позволит в дальнейшем создать дополнительные элементы внутри значка;
8. **align-items: center.** В дальнейшем поможет выровнять элементы по центру значка;
9. **justify-content: center.** Расположит flex-элементы друг за другом по центру контейнера;
10. **z-index: 100.** Задаст позицию отображения на области навигационного меню;
11. **cursor –** отвечает за форму курсора, когда он находиться в пределах элемента. Значение pointer отображает курсор мыши в виде указывающей руки**;**
12. **transform –** отвечает за трансформацию элемента. Значение scale(.8) позволит задать размер значка по горизонтали и вертикали**;**
13. **margin-right –** устанавливает отступ от правого края элемента. Положительное значение увеличивает расстояние между соседними элементами, когда как отрицательное сокращает**.**

Также добавим эффект пульса для значка с помощью псевдоэлемента after.

```
#header .hamburger:after {
    position: absolute;
    content: '';
    height: 100%;
    width: 100%;
    border-radius: 50%;
    border: 3px solid crimson;
    animation: hamburger_puls 1s ease infinite;
}
```

1. **position: absolute.** Элемент будет размещен относительно первого родительского элемента (значка);
2. **content –** отвечает за содержание элемента. Значение **‘_‘** задает пустое содержание;
3. **height: 100%.** Высота элемента будет полностью соответствовать высоте значка – 60px;
4. **width: 100%** . Ширина элемента будет полностью соответствовать ширине значка – 60px;
5. **border-radius: 50%.** По умолчанию элемент обладает квадратной рамкой, используя данное свойство эффект пульса будет круглым;
6. **border: 3px solid crimson.** Задаем идентичные значения свойства что у родительского элемента(значка);
7. **animation –** определяет список применимых анимаций к элементу. Значение **hamburger_puls** – является именем анимации, **1s** – время, которое будет занимать анимация в рамках одного цикла, **ease** – анимация будет плавным, **infinite** – анимация будет проигрываться бесконечно.

Для того чтобы анимация заработала необходимо использовать команду **@keyframes**, в котором будут установлены кадры при воспроизведении анимации элемента - переход от одной позиции к другой (рис.1).

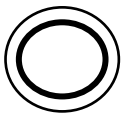

Рис. 1

```
@keyframes hamburger_puls {
```

```
    0% {
      opacity: 1;
      transform: scale(1)   }
    100% {
      opacity: 0;
      transform: scale(1.4) }
  }
```

Элемент **bar** находится внутри элемента **hamburger,** что позволяет создать любые элементы внутри значка (рис.2). В нашем случае это будут три линии, которые будут исходить из одной линии.

Рис. 2

Элемент **bar** находится внутри элемента **hamburger,** что позволяет создать любые элементы внутри значка. В нашем случае это будут три линии, которые будут исходить из одной линии.

Используя псевдоэлементы **after** и **before,** создаем плавную анимацию смещения дубликатов верх и вниз от родительского элемента.

```
#header .hamburger .bar::after,
#header .hamburger .bar::before {
  content: '';
  position: absolute;
  height: 100%;
  width: 100%;
  left: 0;
  background-color: crimson;
  transition: .3s ease;
  transition-property: top, bottom;
}
```

Исключение и создание дубликатов линии после открытия или закрытия меню.

```
#header .hamburger .bar::after {
  top: 8px;
}
#header .hamburger .bar::before {
  bottom: 8px;
}
#header .hamburger.active .bar::before {
  bottom: 0;
}
#header .hamburger.active .bar::after {
  top: 0;
}
```

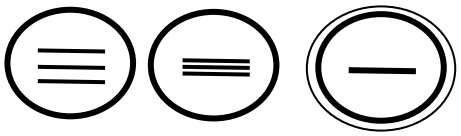

Рис. 3.

Далее необходимо создать запрос @media, который будет изменять вид навигационного меню в зависимости от того от области отображения браузера. Одна из ключевых свойств данного запроса является min-width, в котором будет указываться так называемая контрольная точка изменения, в данном случае указано 1200 пикселей, то есть если область отображения будет меньше или больше заданного значения будут происходить изменения.

```
@media only screen and (min-width:1200px){
    #header .hamburger {
    display: none;
  }
  #header .nav-list ul {
    position: initial;
    display: block;
    height: auto;
    width: fit-content;
    background-color: transparent;
  }
  #header .nav-list ul li {
    display: inline-block;
  }
  #header .nav-list ul li a{
    font-size: 1.8rem;
  }
  #header .nav-list ul a:after {
    display: none;
  }
}
```

Последним этапом создания является создание сценария, который будет отвечать за активацию меню и изменение фона навигационного меню при прокрутке страницы.

```
const hamburger = document.querySelector('.header .nav-bar .nav-
list .hamburger');
const mobile_menu = document.querySelector('.header .nav-bar .nav-list ul');
const menu_item = document.querySelectorAll('.header .nav-bar .nav-
list ul li a');
const header = document.querySelector('.header.container');
hamburger.addEventListener('click', ()=> {
  hamburger.classList.toggle('active');
  mobile_menu.classList.toggle('active');
});
document.addEventListener('scroll',()=> {
  var scroll_position = window.scrollY;
  if(scroll_position > 250) {
    header.style.backgroundColor = "#29323c";
  }
  else {
    header.style.backgroundColor = 'transparent';
```

```
    }
});
menu_item.forEach((item) => {
    item.addEventListener('click', () => {
        hamburger.classList.toggle('active');
        mobile_menu.classList.toggle('active');
    });
});
```

Создание навигационной структуры сайта предоставляет пользователям легкий способ перехода от одного раздела к другому, тем самым снижается время для поиска нужной ему информации.

Литература:

1. Клонингер К. Свежие стили Web-дизайна // К. Клонингер. – М: ДМК Пресс, 2009. – 224 с.
2. Мейер Э. CSS. Каскадные таблицы стилей // Э. Мейер – СПб: Символ-Плюс, 2008. – 576 с.

DEVELOPMENT OF A SITE NAVIGATION SYSTEM

Zhargal B. Erdyneev
Student,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: ezha134@gmail.com

Antonida A. Tonkhonoeva
Cand. Sci. (Education), A/Prof.,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: ant_ton@mail.ru

Today, it is impossible to imagine the development of our society without taking advantage of the opportunities offered by the Internet. Company executives and business representatives advertise their products not only in a traditional way, but also attract their potential customers through a global network. The use of web-sites in business can quickly inform about the offered services and goods. When developing a web portal, it is generally assumed that the resource will consist of several web pages, and when creating a one-page site, it will consist of several sections. Due to a large amount of information, it is rather uncomfortable for a user to navigate a resource without special means of navigation through the portal and as a result, a potential client may leave the web site without reaching the goal. This problem can be solved if a well-thought-out navigation system for the site would be developed. HTML markup language, cascading CSS style sheets, and JavaScript scripting language have been used to create a navigation menu.
Keywords: Internet, web application, navigation, design, HTML, CSS, PHP.