CIT 261 Final Project

A. Student – Winter 2019

Description

The Plan of Salvation is an application that explains the plan of salvation in an interactive way, it is perfect to teach youth and children about this topic. The user needs to hover the images in order to read the explanation of every step of the plan. When the user does it, the image will flip and move down the page and it will display the explanation of the step, it will also change a shadow in the border of the images. Finally, the user can send a form with his information if interested and the form will disappear, and the user will get a message with his name in it.

All the text of the page comes from an external website where it is saved as a JSON file.

CSS Features

All the styles of the page and animations where created with CSS, the colors, the shape of the boxes, the shape of the borders, angles and shadows, as well as the size and font of the text

Code Topics

- ### AJAX

  The text comes from an external page stored as a JSON file from an external site. Once the data is loaded, the arrow function will save the data in the data variable and then create some h2 tag for the title, p tag to add the paragraph, it will do it for every image.

```
    const thePlan = new XMLHttpRequest();
        thePlan.open('GET',
'https://rodriguezoswaldo.github.io/cit261/salvationplan/salvation.json
', true);
        thePlan.onload = () => {
            var theData = JSON.parse(thePlan.responseText);
            console.log(theData);
            console.log(theData[0].title);
            document.getElementById('pre Life').innerHTML = "<h2>" +
theData[0].title + "</h2>" + "<p>" + theData[0].explain + "</p>";
            document.getElementById('creation').innerHTML = "<h2>" +
theData[1].title + "</h2>" + "<p>" + theData[1].explain + "</p>";
            document.getElementById('earthLife').innerHTML = "<h2>" +
theData[2].title + "</h2>" + "<p>" + theData[2].explain + "</p>";
            document.getElementById('atonement').innerHTML = "<h2>" +
theData[3].title + "</h2>" + "<p>" + theData[3].explain + "</p>";
            document.getElementById('phyDeath').innerHTML = "<h2>" +
theData[4].title + "</h2>" + "<p>" + theData[4].explain + "</p>";
            document.getElementById('resurrection').innerHTML = "<h2>"
+ theData[5].title + "</h2>" + "<p>" + theData[5].explain + "</p>";
            document.getElementById('aftLife').innerHTML = "<h2>" +
theData[6].title + "</h2>" + "<p>" + theData[6].explain + "</p>";
        }
        thePlan.send();
```

- CSS3 Animations & Transitions

  Once the page is load the *header* of the page will appear from the inner part of the page, will flip, and finally make bigger. It only happens once. It uses two transform properties (*scale* and *rotate*) and the *opacity* property.

```
@keyframes theTitle{
            0%{
               transform: rotateX(360deg) scale(0.5);
               opacity: 0;
            }
            50%{
                transform: rotateX(00deg) scale(1);
                opacity: 0.5;
            }
            100%{
                transform: rotateX(0deg) scale(1.5);
                opacity: 1;
            }
        }

#title {
            animation-name: theTitle;
            animation-duration: 3s;
            animation-delay: 1s;
            animation-fill-mode: forwards;
            transition-property: transform, opacity;
            transition-duration: 1s;
            opacity: 0;
        }
```

  When the user hover the images it will show the back of “the cards” and the shadow of the boxes will change the color several times while they are reading the content. It happens with every box.

```
@keyframes theBack{
            0%{
                box-shadow: rgb(78, 107, 136) 0px 4px 4px;
            }
            50%{
                box-shadow: rgb(124, 104, 77) 0px 15px 15px;
            }
            100%{
                box-shadow: rgb(92, 145, 85) 0px 25px 25px;
            }
        }

.back {
            position: absolute;
            width: 100%;
            height: 100%;
            backface-visibility: hidden;
            background: rgb(67, 162, 165);
            color: rgb(45, 90, 167);
            transform: rotateY(180deg);
            border-radius: 30px;
            box-shadow: #159 0px 25px 60px;
            border: 2px solid #111111;
```

```
            animation-name: theBack;
            animation-duration: 3s;
            animation-iteration-count: infinite;
            animation-fill-mode: forwards;
            transition-property: box-shadow;
        }
```

The button that submit the form have an active animation all the time, it changes the border radius of the button continually.
However, once the user hover the button, it will change the background and yet changing the border-radius.

```
@keyframes btn{
            0%{
                border-radius: 12px 12px 12px 12px;
            }
            25%{
                border-radius: 0px 0px 0px 0px;
            }
            50%{
                border-radius: 12px 12px 12px 12px;
            }
            75%{
                border-radius: 0px 0px 0px 0px;
            }
            100%{
                border-radius: 12px 12px 12px 12px;
            }
        }
        @keyframes btnHover{
            0%{
                background-color: #159;
                border-radius: 12px 12px 12px 12px;
            }
            25%{
                border-radius: 0px 0px 0px 0px;

            }
            50%{
                background-color:rgb(117, 91, 36);
                border-radius: 12px 12px 12px 12px;
            }
            75%{
                border-radius: 0px 0px 0px 0px;
            }
            100%{
                background-color: rgb(45, 117, 36);
                border-radius: 12px 12px 12px 12px;
            }
        }
#btn{
            font-size: 15px;
            padding:15px 20px ;
            border: #159 2px solid;
            color:white;
            background: rgb(15, 116, 116);;
            animation-name:btn;
```

```
            animation-duration: 5s;
            animation-iteration-count: infinite;
            animation-fill-mode: forwards;
            transition-property: border-radius;
        }

        #btn:hover{
            background: #999;
            color:black;
            border: #199 2px solid;
            animation-name:btnHover;
            animation-duration: 5s;
            animation-iteration-count: infinite;
            animation-fill-mode: forwards;
            transition-property: border-radius, background-color;
        }
```

The header " *Do you want to learn more"* has an animation as well. When the user hover the title, actives the animation.

```
@keyframes header{
            0%{
                transform: skewY(20deg);
            }
            25%{
                transform: skewY(-20deg);
            }
            50%{
                transform: skewY(20deg);
            }
            75%{
                transform: skewY(-20deg);
            }
            100%{
                transform: skewY(20deg);
            }
        }
.header:hover{
            animation-name: header;
            animation-duration: 10s;
            animation-iteration-count: infinite;
            animation-fill-mode: forwards;
            transition: transform 6s;
        }
```

- CSS3 Transforms

I used the CSS transform property all over the page starting with the title,

```
  #title {
            animation-name: theTitle;
            animation-duration: 3s;
            animation-delay: 1s;
            animation-fill-mode: forwards;
            transition-property: transform, opacity;
            transition-duration: 2s;
}
```

```
@keyframes theTitle{
            0%{
               transform: rotateX(360deg) scale(0.5);
               opacity: 0;
            }
            50%{
                transform: rotateX(00deg) scale(1);
                opacity: 0.5;
            }
            100%{
                transform: rotateX(0deg) scale(1.5);
                opacity: 1;
            }
        }
```

And the boxes with the images.

```
  . .thecard {
            position: absolute;
            width: 14%;
            height: 100%;
            transform-style: preserve-3d; /* used in order to show the
back of the card */
            transition-property: all;
            transition-duration:  1s;
            transition-timing-function: ease-out;
            cursor: pointer;
        }
These properties will flip the card, change the position, and change
the size respectively.

.thecard:hover {
            transform: rotateY(180deg) translateY( 400px) scale(1.5);
        }
```

This one will flip the back part of the card so it will show the content

```
.back {
            position: absolute;
            width: 100%;
            height: 100%;
            backface-visibility: hidden;
            background: rgb(67, 162, 165);
            color: rgb(45, 90, 167);
            transform: rotateY(180deg);
}
```

The animation of the header was created using *skewY* of the transform property

```
@keyframes header{
            0%{
                transform: skewY(20deg);
            }
            25%{
                transform: skewY(-20deg);
            }
            50%{
                transform: skewY(20deg);
```

```
            }
            75%{
                transform: skewY(-20deg);
            }
            100%{
                transform: skewY(20deg);
            }
        }
```

- CSS3 Transitions

I used transition along with all the animations in order to make them look better. For example, in the title I used these.

```
.   #title {
            animation-name: theTitle;
            animation-duration: 3s;
            animation-delay: 1s;
            animation-fill-mode: forwards;
            transition-property: transform, opacity;
            transition-duration: 2s;
```

In the case of the cards I used several transition properties.

```
.thecard {
            position: absolute;
            width: 14%;
            height: 100%;
            transform-style: preserve-3d; /* used in order to show the
back of the card */
            transition-property: all;
            transition-duration:  1s;
            transition-timing-function: ease-out;
            cursor: pointer;
        }
```

I also used them in the button for the for, this time I added the 4 properties into one.

```
#btn{
            font-size: 15px;
            padding:15px 20px ;
            border: #159 2px solid;
            color:white;
            background: rgb(15, 116, 116);;
            animation-name:btn;
            animation-duration: 5s;
            animation-iteration-count: infinite;
            animation-fill-mode: forwards;
            transition: border-radius 3s linear ease-in;
            cursor:pointer;
        }
```

As well as disappearing the form.

```
btn.addEventListener('click', ()=>{
            form.style.transition = 'opacity 3s linear 1s' ;
            form.style.opacity = '0';
        });
```

- ## CSS3 Using JavaScript

  I added CSS using JavaScript to the form, disappearing it when the use clicks the submit button.

```
let form = document.getElementById('form');
            btn.addEventListener('click', ()=>{
            form.style.transition = 'opacity 3s linear 1s' ;
            form.style.opacity = '0';
        });
```

  I also added the style to the message given to the user after they submit the form

```
let userMessage = document.createElement('p');
            //class name in order to style the element
    userMessage.className ='message';
```

- ## DOM Document Object Model

  I manipulated the DOM while sending a message to the user once he has submitted the form. This response is inside a *new element created* when the button is clicked, this get two elements from the form the phone number and the User Name, and concatenate the message and the elements. And then print it in the page.

```
      //Creating the new element p
            let userMessage = document.createElement('p');
            //class name in order to style the element
            userMessage.className ='message';
            //Message for the user
            message = 'Thank you '+ parseName[0].name + '! We will get
in touch with you in short to your number ' + parseName[0].phone;
          //Adding the message to the p element
            userMessage.append(message);
            //Selecting the space where the message will be posted for
the user to see!
            let theContainer = document.getElementById('practice');
            //Adding the text to the space in the page
            theContainer.append(userMessage);
```

- ## HTML5 Tags for Media

  This application doesn't use any of these tags.

- ## JavaScript

  My application includes almost all the core functionalities that JS offers.

  - Functions, arrays, loops, switch statements, object notation, inline functions, and string operations

```
let users = [];
        const addUsers = (ev) => {
            ev.preventDefault(); //to stop the form submitting
            let user = {
```

```
                name: document.getElementById('name').value,
                email: document.getElementById('email').value,
                phone: document.getElementById('phone').value
            }
            users.push(user);//pusing into the Array Movies the input
store in the variable movie.
            document.forms[0].reset();// clear the form for the next
entries
            //saving to localStorage as a String
            localStorage.setItem('UserInfo', JSON.stringify(users));

            ////////////////////////////////////////////////////
            //Printing  a message for the user after submiting the form
            //Getting Data Typed in by the user
            var name = localStorage.getItem('UserInfo');
            //Converting the data into an object
            var parseName = JSON.parse(name);
            //Creating the new element p
            let userMessage = document.createElement('p');
            //class name in order to style the element
            userMessage.className ='message';
            //Message for the user
            message = 'Thank you '+ parseName[0].name + '! We will get
in touch with you in short to your number ' + parseName[0].phone;
           //Adding the message to the p element
            userMessage.append(message);
            //Selecting the space where the message will be posted for
the user to see!
            let theContainer = document.getElementById('practice');
            //Adding the text to the space in the page
            theContainer.append(userMessage);
        }
        document.addEventListener('DOMContentLoaded', () => {
            document.getElementById('btn').addEventListener('click',
addUsers);
        });
```

- JavaScript Events

I used only two JS events in my page, I used the DOMContentLoaded which will run as soon as all the content in the page is loaded. I also used the click event which runs once the user clicks the button and this one will add the data typed in by the user inside an object, and inside the LocalStorage. The other event runs when the user click on the same btn, but this one will hide the form from the user.

```
document.addEventListener('DOMContentLoaded', () => {
            document.getElementById('btn').addEventListener('click',
addUsers);
        });
let form = document.getElementById('form');
            btn.addEventListener('click', ()=>{
            form.style.transition = 'opacity 3s linear 1s' ;
            form.style.opacity = '0';
        });
```

- JavaScript Objects

I added one JavaScript Object to my page, this object will be filled by the user's input.

```
let users = [];
        const addUsers = (ev) => {
            ev.preventDefault(); //to stop the form submitting
            let user = {
                name: document.getElementById('name').value
                email: document.getElementById('email').value,
                phone: document.getElementById('phone').value
            }
```

- JSON

The data used in the page came from a JSON file stored in an external page, but it was also created by me, and it can be found here. I also used two JSON functions JSON.stringify() to convert the object into a string (text) in order to save it in LocalStorage, and JSON.parse() to convert the string back into an object.

```
            //saving to localStorage as a String
            localStorage.setItem('UserInfo', JSON.stringify(users));

            ////////////////////////////////////////////////////
            //Printing a message for the user after submiting the form
            //Getting Data Typed in by the user
            var name = localStorage.getItem('UserInfo');
            //Converting the data into an object
            var parseName = JSON.parse(name);
```

- Local Storage

Having the data inside the object I saved this object into LocalStorage as a String, after that, I got part of the data from LocalStorage, converted back into an object and used some of the properties of this object in order to give a message to the final user.

```
let users = [];
        const addUsers = (ev) => {
            ev.preventDefault(); //to stop the form submitting
            let user = {
                name: document.getElementById('name').value,
                email: document.getElementById('email').value,
                phone: document.getElementById('phone').value
            }
            users.push(user);//pusing into the Array Movies the input
store in the variable movie.
            document.forms[0].reset();// clear the form for the next
entries
            //saving to localStorage as a String
            localStorage.setItem('UserInfo', JSON.stringify(users));

            ////////////////////////////////////////////////////
            //Printing a message for the user after submiting the form
            //Getting Data Typed in by the user
            var name = localStorage.getItem('UserInfo');
```

Design Process

I gave a lot of thinking before deciding what to do. I followed some tutorials and wrote some pages using JS and CSS until I talked to my wife, and the idea came to create a Plan of Salvation interactive which would be helpful for even sharing the gospel. Then I started thinking how I wanted the user to use it, and the idea of creating 7 flipping boxes explaining each step of the plan of salvation was born.

I then started by creating the 7 boxes, style them, gave them shape, colors and the animation desired. I then modified their background and used images as background instead of simple images inside the boxes. Once I had the animation that I wanted in the way that I wanted, I started working in the text that the user would see, there I created the JSON file, and created the AJAX request and started populating the cards with the data.

After that I created the form and the object which was going to store the data type by the user, later I created the response for the user. And finally I review every part of the code looking for ways to improve the readability of the code, and fixing some bug. The last thing I did was to disappear the form when the user clicks on it.

Changes from Original Concept

The original concept didn't change at all, the final product what I have thought originally. Along the way I had considered to make some design changes, such as using and animated background which would hurt the application as a whole, and would have probably be a little bit distracting from the original purpose of the application. And for this last one I decided not to add it. So, it has all what I thought in the better and simple way for the user to interact with it, and to have interest in learning more about it.

Problems and Resolutions

Most of the problems that I had occurred because I mistyped some of the properties, for example I was writing animation-property, instead of animation property. And It took me a while to find that error, which I think happened because I learned the two properties together. After a careful review of the code I found the errors and fixed them. I was also having some problems with positioning the boxes inline, until I used 'display: flex;' and fixed the problem. Also I had problems with the position of the boxes because they were going outside to the left part of the container, and finally I was able fixed the problem setting the position of the boxes with the following CSS property.

```
.card1 {
    right: 1200px;
}

.card2 {
    right: 1000px;
}

.card3 {
    right: 800px;
```

```
}

.card4 {
    right: 600px;
}

.card5 {
    right: 400px;
}

.card6 {
    right: 200px;
}

.card7 {
    right: 0px;
}
```