TEXT CLASS REVIEW

TEMAS A TRATAR EN LA CUE

- Elementos de la experiencia de usuario: planos primarios.
- Diseñadores de experiencia de usuarios (UX).
- Representaciones visuales.
- Diseñadores de interfaces de usuario (UI).
- Maquetador.
- Guías de estilo y representaciones visuales.
- Desarrollador front-end.
- Metodología de arquitectura de CSS.

ELEMENTOS DE LA EXPERIENCIA DE USUARIO: PLANOS PRIMARIOS

Antes de empezar a crear un sitio web y profundizar en la maquetación de éste, es importante entender el flujo de trabajo a utilizar, con el cual se crearán las interfaces de usuario (representaciones visuales) que posteriormente serán convertidas en código.

Para poder crear una buena experiencia para el usuario, es importante seguir paso a paso un proceso que va desde lo más abstracto a lo más concreto. Garret, en su libro “Los Elementos de la Experiencia del Usuario”, divide este proceso en 5 planos primarios, los cuales transforman una hipótesis basada en supuestos a interfaces de usuarios tangibles.

Vamos a revisar estos 5 planos, desde el más abstracto al más concreto:

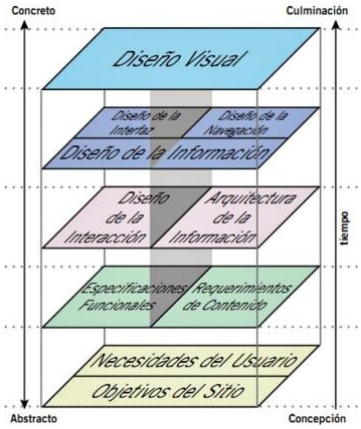

Imagen 1. Planos primarios

- **Plano de estrategia:** éste incorpora tanto lo que las personas que desarrollan el sitio web quieren obtener, como también lo que los usuarios quieren obtener del sitio.
- **Plano de alcance:** está compuesto por las funcionalidades y contenidos del sitio web. Responde a la pregunta de si un contenido o función debe ser incluida en el sitio.
- **Plano de estructura:** éste define cómo los usuarios navegarán en un sitio web, es decir, concreta la arquitectura de la información del sitio y el tipo de interacción que tendrá el usuario con éste.
- **Plano de esqueleto:** éste corresponde al posicionamiento de los componentes del sitio web, tales como: botones, pestañas, fotos y bloques de texto. Se diseña con el fin de producir el mayor efecto y eficiencia.
- **Plano de superficie:** en ésta, el usuario ve una serie de páginas web, compuestas de imágenes y texto. Algunas de las imágenes tendrán un efecto cuando las cliqueas, y otras serán solo ilustraciones.

DISEÑADORES DE EXPERIENCIA DE USUARIOS (UX)

Son los encargados de investigar y analizar la información obtenida de los planos abstractos (estrategia, alcance, estructura y esqueleto). Éstos tienen la tarea de investigar la conducta del usuario, analizar los resultados de dichas investigaciones y monitorear el desarrollo del proyecto. Por lo general, entregan a su equipo toda la información obtenida utilizando una representación visual llamada Wireframe.

REPRESENTACIONES VISUALES

Éstas corresponden a la aplicación de una idea o concepto en un diseño visual, es decir, creaciones de diseño que describen el proceso de ideación de una interfaz de usuario. Existen representaciones con distinta fidelidad:

- *Fidelidad baja:* son bosquejos de la estructura y/o contenido de una página web.
    - *Sketch:* representación en papel de una página web o aplicación, y es la manera más rápida de visualizarlas.
    - *Wireframe:* entendido como el esqueleto de una página web o aplicación. No tiene diseño y se crea generalmente con programas especializados, como: Balsamiq, Axure R, entre otros.
- *Fidelidad media:* son representaciones digitales de todo el contenido visual de una página web, tanto la estructura como el contenido de las representaciones de baja fidelidad.
    - *Mockup:* éste utiliza diseño interfaz. Combina las características de los wireframes con el diseño visual, como: colores, fuentes, imágenes, entre otros. Se pueden crear en Photoshop, Illustrator, Sketch, Adobe XD, entre otros.
- *Alta fidelidad:* es el tipo de representación más fiel de una página web sin utilizar código. En ésta se simulan las interacciones que pueda tener el usuario con la interfaz.
    - *Prototipo:* corresponde a un mockup enriquecido con piezas de la experiencia de usuario, como son las interacciones y animaciones.

DISEÑADORES DE INTERFACES DE USUARIO (UI)

Su trabajo consiste en diseñar las interfaces con las que interactuará el usuario. Dichas interfaces, son la suma de: la arquitectura de la información, los elementos visuales y los patrones de interacción creados por el diseñador UX. Generalmente, los diseñadores UI proporcionan a su equipo representaciones visuales de mediana y alta fidelidad, además de la guía de estilos, que dan una idea de los elementos que componen la interfaz de usuario.

MAQUETADOR

También llamados desarrolladores UI o diseñadores front-end, son los encargados de realizar la maquetación.

Ésta corresponde al proceso de traducir las representaciones visuales a código HTML, CSS y JavaScript de manera fiel, es decir, que debe contener todas las características visuales y funcionales determinadas por los diseñadores UX/UI.

GUIA DE ESTILOS Y REPRESENTACIONES VISUALES

Para maquetar, es necesario lo siguiente:

1. Identificar el lenguaje de una representación visual, es decir, es importante conocer estos elementos y así poder tener claridad en cuanto a la identidad de la marca y el lenguaje de diseño. Dichos elementos se encuentran dentro de una guía de estilos.
2. Una guía de estilo, la cual contiene las líneas, reglas, colores y uso de los elementos visuales que están incluidos en la interfaz de usuario. Son de carácter optativo.

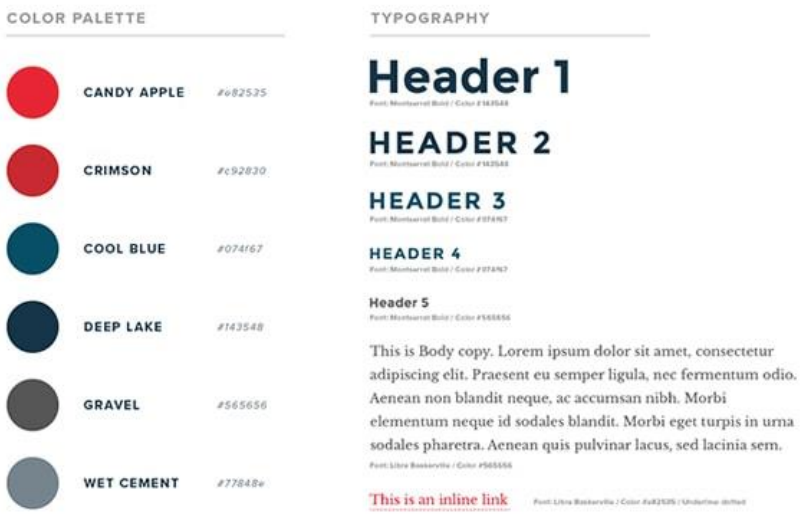

Imagen 2. Guía de estilos

3. Traducir el lenguaje visual a una maqueta. Luego de identificar los componentes de una representación, es necesario traducirlos a HTML y CSS.
4. Documentar el proyecto maquetado. Es importante hacer registro de lo codificado, para así poder guiar al desarrollador que implementará el backend. Dicho registro, debe contener una descripción de los elementos incluidos en el proyecto, y una guía con recomendaciones sobre la estructura y estilos que han sido aplicados.

DESARROLLADOR FRONT-END

Son los encargados de construir, en base a códigos, las interfaces web que utilizamos a partir de ideas o diseños. Algunas de las tecnologías que maneja un front-end, son las que se muestran a continuación:

Imagen 3. Tecnologías Front-End

El front-end es la capa frontal y lo que se ve en un sitio web o en una aplicación. Éste se utiliza para unir la interfaz gráfica de usuario (graphic user interface GUI, en inglés) y la ejecución de las acciones.

Sus funciones principales son:

- Traducir diseños a lenguaje de programación.
- Desarrollar la parte visual de la web.

- Diseñar la estructura de la web.
- Facilitar la navegación del usuario.
- Ocuparse de los componentes externos del sitio o aplicación web.

METODOLOGÍAS DE ARQUITECTURA DE CSS

Son aquellas que sirven como guía de estilo y ayudan a organizar y estructurar el CSS, haciendo que sea fácil escalarlos y mantenerlos por otros desarrolladores.

OOCSS

Estos separan la estructura y el contenido de la interfaz en objetos CSS, que pueden ser modificados independientemente.

- Separar la estructura del diseño, en la siguiente imagen podemos ver un código CSS sin aplicar los principios OOCSS:

```css
#button {
    width: 200px;
    height: 50px;
    padding: 10px;
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #aaa);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
  }
  #widget {
    width: 500px;
    min-height: 200px;
    overflow: auto;
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #222);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
  }
```

Y ahora podemos observar un código CSS aplicando los principios de OOCSS:

```css
.button {
  width: 200px;
  height: 50px;
  padding: 10px;
}
.widget {
  width: 500px;
  min-height: 200px;
  overflow: auto;
}
.skin {
  border: solid 1px #ccc;
  background: linear-gradient(#ccc, #aaa);
  box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

Ahora bien, todos los elementos usan clases. Los estilos comunes se combinan en una «piel» re-utilizable y nada se repite innecesariamente. Sólo tenemos que aplicar la clase de .skin a todos los elementos, y el resultado será el mismo que en el primer código, excepto con menos código y una posibilidad que permite su posterior re-utilización.

- Separar el contenedor del contenido: para ilustrar el segundo principio, veamos un ejemplo. Es muy común que tengamos un código CSS similar a éste:

```css
header h1 {
    font-family: 'Roboto', Helvetica, sans-serif;
    font-size: 2em;
    color: #F44;
  }
  /*mas código css...*/
  footer h1 {
    font-family: 'Roboto', Helvetica, sans-serif;
    font-size: 1.5em;
    color: #F44;
    opacity: 0.5;
    filter: alpha(opacity = 50);
  }
```

En este caso, estamos repitiendo código otra vez, y los estilos que se declaran usando el selector descendiente no son reutilizables, ya que dependen de un contenedor en particular (en el ejemplo: header y footer).

Una buena aproximación siguiendo la metodología OOCSS sería:

```
h1{
  font-family: 'Roboto', Helvetica, sans-serif;
  color: #F44;
}
/* ... */
h1, .h1-size { font-size: 2em;   }
h2, .h2-size { font-size: 1.8em; }
h3, .h3-size { font-size: 1.5em; }
/* ... */
.transparente {
  opacity: 0.5;
  filter: alpha(opacity = 50);
}
```

- Si seguimos los principios de OOCSS, nos aseguraremos de que nuestros estilos no dependen de ningún elemento contenedor. Esto significa que se podrán volver a utilizar en cualquier parte del documento, independientemente de su contexto estructural.

SMACSS

Funciona mediante la organización de las reglas CSS en 5 categorías: Base, Maquetación, Módulo, Estado y tema.

- Base: son las reglas básicas para elementos, atributos, pseudo-clases, entre otros. Un ejemplo sería: *Normalize.css*.

```
h1 {
    font-size: 32px;
  }
  div {
    margin: 0 auto;
  }
  a {
    color: blue;
  }
```

- Maquetación: son las reglas de estilo que están relacionadas con el diseño estructural de las páginas. Por ejemplo: contenedores, grillas, entre otros. Éstas van con el prefijo layout- o l-.

```
.layout-sidebar {
    width: 320px;
  }
  .l-comments {
    width: 640px;
  }
```

- Módulos: son componentes reutilizables y modulares.

```
.call-to-action-button {
  text-transform: uppercase;
  color: #FFF200;
}
.search-form {
  display: inline-block;
  background-color: E1E1E1;
}
```

- Estados: son las reglas de estilo que especifican el estado actual de un elemento en la interfaz.

```
.is-hidden {
  display: none;
}
.is-highlighted {
  color: #FF0000;
  background-color: #F4F0BB;
  border: 1px solid #CBBD15;
}
```

- Temas: en OOCSS se le llama “skin” o piel. En SMACSS es opcional, los estilos visuales pueden estar integrados a los módulos y estados, o separados para sitios en donde el usuario pueda elegir un tema, para sitios multi-lenguaje, entre otros.

SMACSS ofrece una nomenclatura más simple que BEM. No hay nombres para los estilos base, ya que se usan los propios selectores (h1, p, a, entre otros). A los módulos se les da un nombre de clase único, y los sub-componentes y las variaciones son prefijados con el nombre del módulo padre.

BEM

Esta será la metodología que utilizaremos, debido a: su propuesta para escribir clases sin repetición de estilos, lo que evita conflictos asociados a la cascada CSS; su popularidad y aceptación por parte de la comunidad de desarrolladores; y por la facilidad para aprenderla. BEM significa bloque, elemento y modificador.

Este enfoque garantiza que todos los que participan en el desarrollo de un sitio web trabajen con un único código base, y hablen el mismo idioma. El uso de la convención de nomenclatura adecuada de BEM le preparará mejor para los cambios de diseño realizados en su sitio web

Para saber más sobre esta metodología, debes revisar los videos y ejercicios de este CUE (1).

También existen algunas otras metodologías, como:

- *Atomic Design*: divide los elementos de una interfaz en átomos, los cuales se pueden unir formando moléculas y organismos.
- *SUITCSS*: presenta las mejores prácticas a usar en la creación de clases y estructura CSS.