PROYECTO DE SOFTWARE

Conceptos básicos

Cursada 2022

¿QUÉ ABORDAREMOS EN ESTE VIDEO?

- Software libre.
- HTML-CSS: aspectos básicos.
- Introducción al DOM.

PENSEMOS EN LAS SIGUIENTES SITUACIONES ...

Necesitamos desarrollar una aplicación para un amigo que tiene una agencia de remises.
Estamos a cargo de un equipo de desarrollo y nos solicitan implementar una aplicación para hacer un seguimientos de trámites.

- ¿Con qué herramientas se les ocurre desarrollarlo?¿Esta elección podría limitarnos de alguna forma?
- ¿Qué pasa cuando nosotros no queremos hacer más el mantenimiento de este software? ¿Quién lo podría hacer?

¿PODEMOS PENSAR EN SOFTWARE LIBRE PARA ESTO?

- Por supuesto: **¡SI!**
- El **software libre** es el software donde los usuarios tienen la **libertad** para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software.
- **El acceso al código fuente es una condición necesaria.**
- La definición oficial pueden encontrarla en el **sitio del proyecto GNU**

CATEGORÍAS DE SOFTWARE LIBRE Y PRIVATIVO

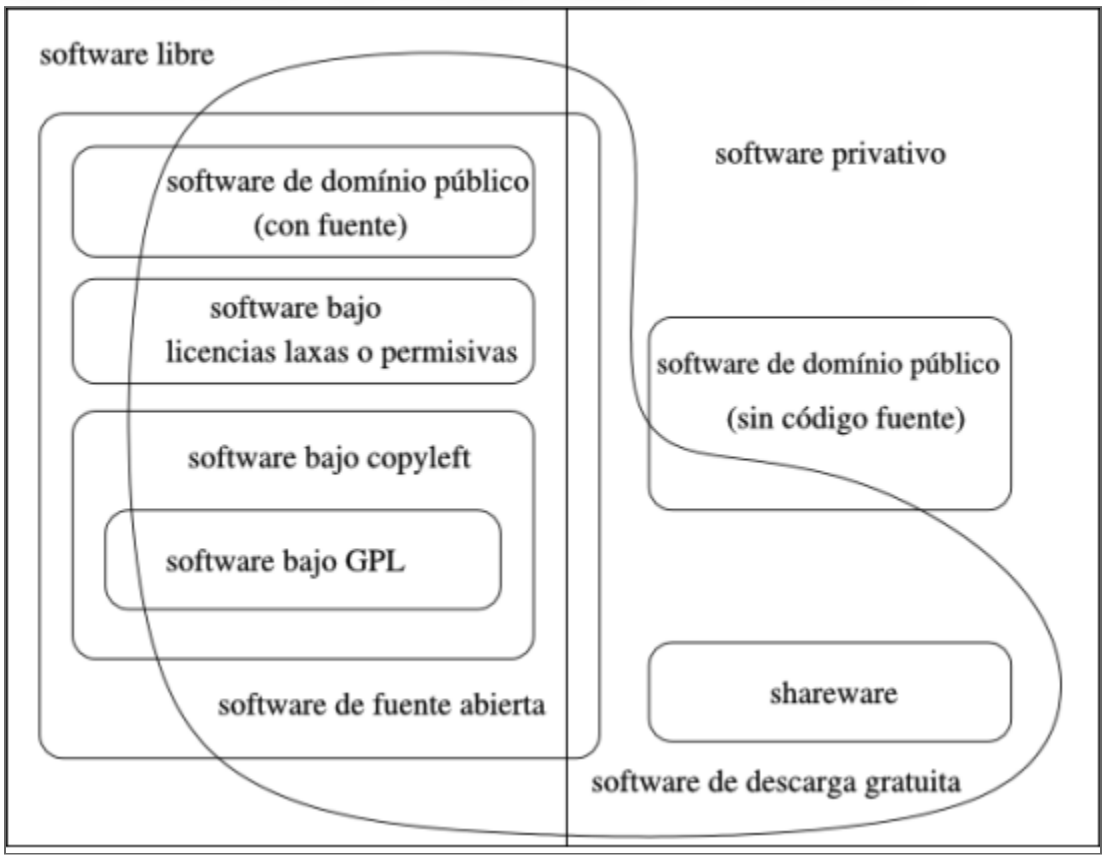


Imagen obtenida de **https://www.gnu.org/philosophy/categories.html**

LICENCIAS: ¿CUÁL ELEGIR?

Es una pregunta recurrente, y la respuesta es: depende de varios factores

- En qué contexto se desarrolló el software: ¿en nuestra casa/en la universidad/en el trabajo?
- En qué nos basamos para desarrollar el software: si se usó algo GPL no se podrá cerrar... pero si se usó algo con copyright no se podrá licenciar como GPL.
- Ejemplo de argumentos válidos:
  - **https://www.gnu.org/philosophy/university.html**
  - **http://producingoss.com/es/license-choosing.html**.

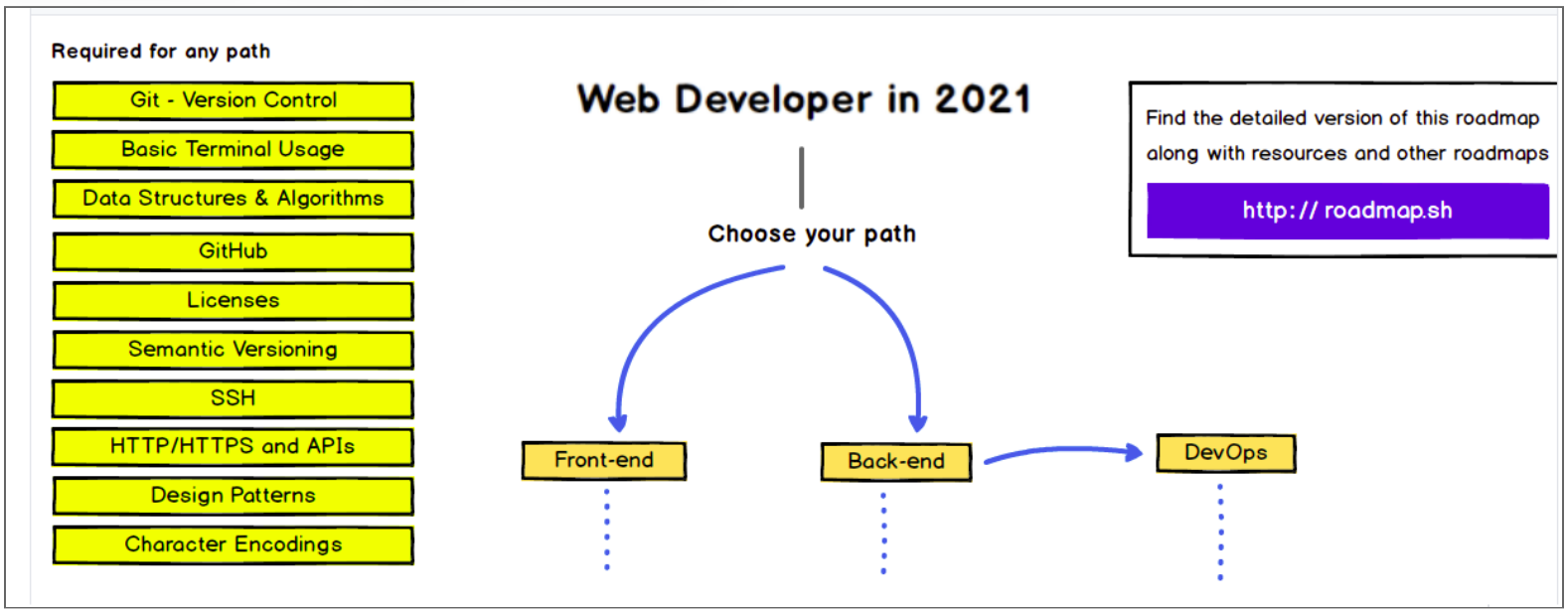
¿POR QUÉ ARRANCAR CON ESTE TEMA?

- Las herramientas y tecnologías que **usaremos** serán **sofware libre**.
- El software que **desarrollaremos** en la matería será **software libre**.

OBJETIVO GENERAL DE LA MATERIA

Desarrollar una aplicación web integrando distintas herramientas y tecnologías.
Se usará **Python** en el servidor y **Javascript** en el cliente.

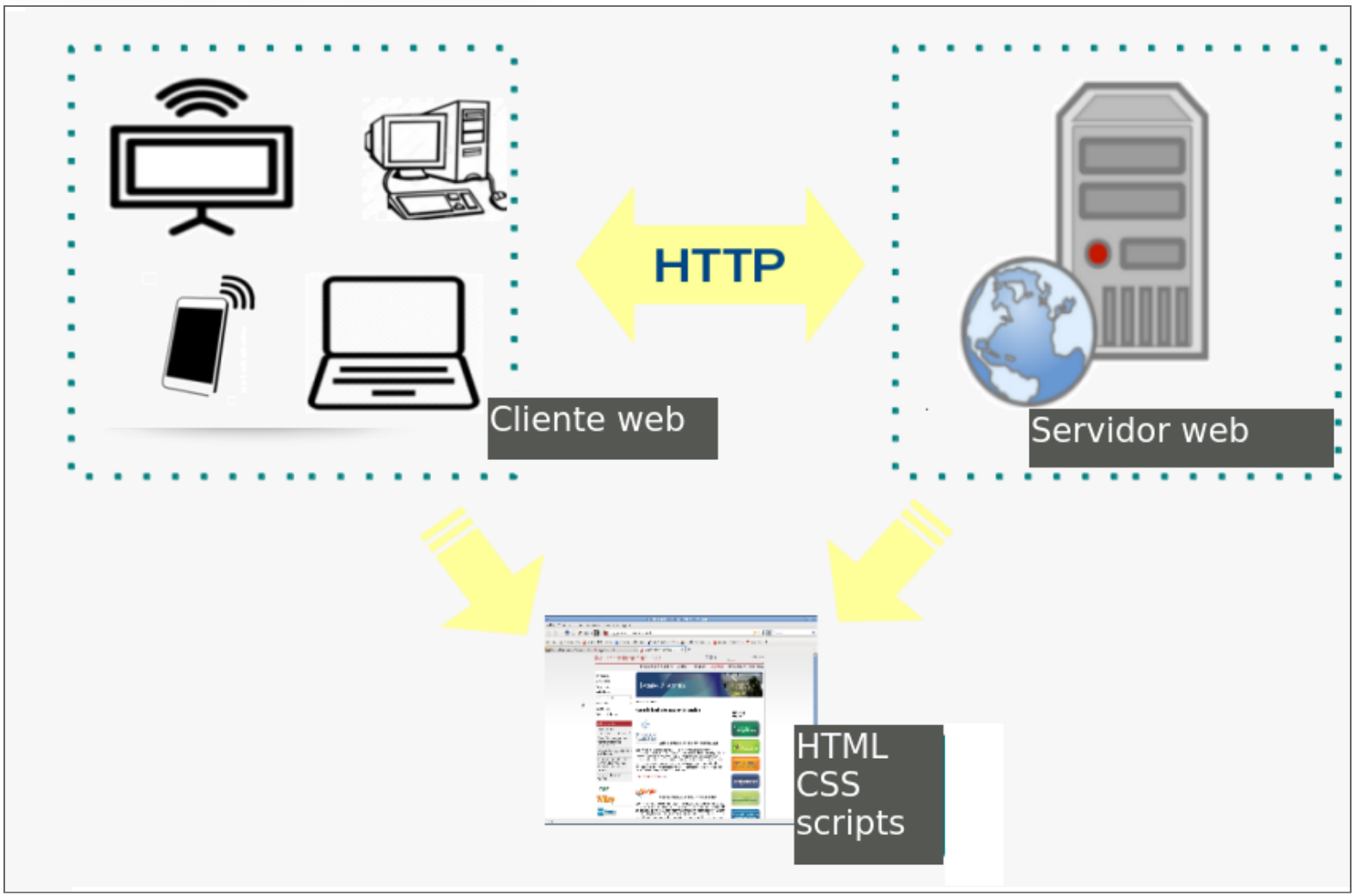
DESARROLLO WEB: ¿QUÉ INVOLUCRA?



[developer-roadmap 2021]

INICIEMOS POR LO MÁS SIMPLE...

LA ARQUITECTURA WEB BÁSICA



ESPECIFICACIONES PÚBLICAS

¿Dónde?

RFC: **Request For Comments**

El consorcio de la web

El consorcio de la web se encarga del desarrollo de estándares y guías que aseguren el crecimiento futuro de la web.

VOLVAMOS A NUESTRO PROBLEMA...

Si tenemos que decidir qué tipo de desarrollo proponer o con qué tecnologías trabajar, **¿por dónde empezamos?**, **¿qué deberíamos analizar?**

Podemos analizar algunas tendencias y estadísticas.

ALGUNAS ESTADÍSTICAS

- Sobre servidores: **https://news.netcraft.com/**
- Sobre navegadores: **https://gs.statcounter.com/browser-market-share#monthly-202007-202107-bar**

TENDENCIAS ...

- **2021**
- Ver estadísticas de nuestra región.

PARA ANALIZAR

- Estas estadísticas nos dan un panorama del contexto.
- También nos permiten analizar las tendencias y características de usos.

  Decidir qué tecnologías usar...

- Podemos retomar esto al final de la cursada ...

LA WORLD WIDE WEB

- Más de 30 años. **A Proposal, Tim Berners-Lee, CERN**
- La propuesta era (y sigue siendo) sencilla:
  - Un mecanismo para identificar todo recurso accesible en la web: **URL/URI**
  - Un protocolo simple: **HTTP**
  - Un lenguaje para definir hipertextos: **el lenguaje HTML**

SOBRE LAS URI/URL

- La especificación es la **RFC 3986**

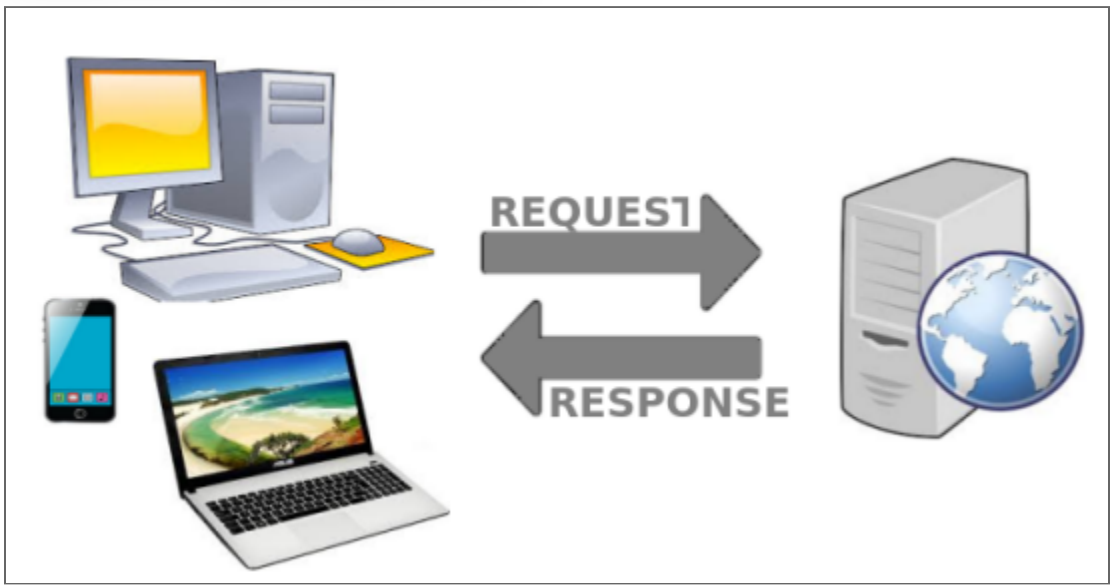
Ejemplos típicos

```
    http://www.servidor.com.ar/especificacion#parte3
    https://www.taller.com.ar/info.php?id=12&qq=11
    mailto:proyecto@info.unlp.edu.ar

file:///home/claudia/git/proyecto2014/2020/teorias/clase2/
index.html
```

- URL encoding
  - Las URLs se transmiten en ASCII.
  - Algunos caracteres deben convertirse.

```
../Clase%201/EjemplosClase1/Ejemplo%20con%20enlaces.html
```

SOBRE EL PROTOCOLO HTTP



- HTTP es un protocolo simple: una transacción HTTP consta de 4 pasos: **inicio conexión - solicitud - respuesta - cierre conexión**
- HTTP es un **protocolo sin estado**: ¿qué significa esto?
- **RFC 7540 (2015)**

ALGUNAS SOLICITUDES HTTP

- **GET**: retorna la información identificada por la URI-solicitada.
- **HEAD**: retorna sólo los encabezados, sin el cuerpo de la respuesta.
- **POST**: se utiliza para el envío de datos al servidor. Estos datos se incluyen el el cuerpo en el mensaje.

La respuesta

- El servidor retorna un código que indica el estado de la solicitud (por ejemplo: 200) y el recurso.
- Más info sobre **solicitudes HTTP**
- Un poco más sobre **HTTP2**

EL LENGUAJE HTML

```
<!DOCTYPE html>
<html lang="es">

<head>
<title>Título</title>
</head>

<body>
<h1>Proyecto de Software</h1>
<h2>Reglamento de cursada</h2>
</body>

</html>
```

- En el sitio W3C está la **especificación** oficial.
- La última versión en **Web Hypertext Application Technology Working Group**

HTML: BÁSICO

¿Qué indica la cláusula **doctype**?

```
<!DOCTYPE html>
```

Los elementos pueden tener atributos.

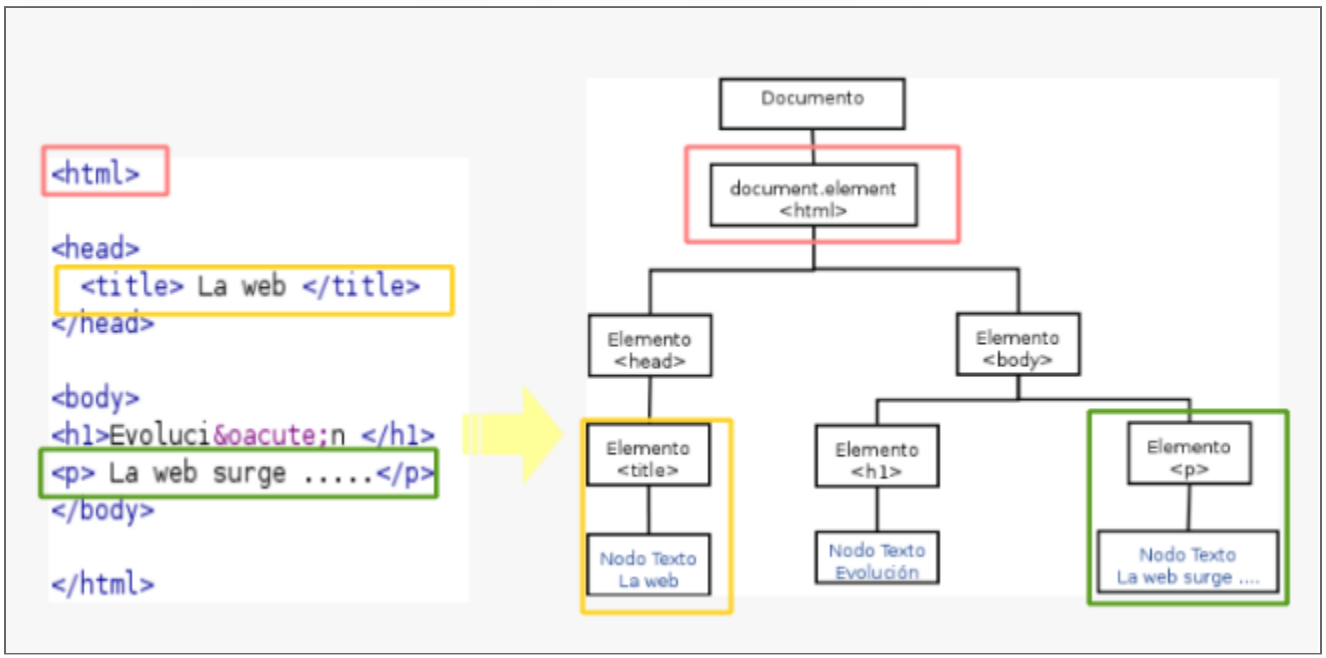
```
<html lang="es">
```

- ¿Qué elementos contiene el encabezado? ¿Y el cuerpo?

```
<!DOCTYPE html>
<html lang="es">

<head>
<title>Título</title>
</head>

<body>
<h1>Proyecto de Software</h1>
<h2>Reglamento de cursada</h2>
    Acá va el texto del reglamento
</body>

</html>
```

- Más info: **Tutorial HTML & CSS Is Hard**

DOM: EL MODELO DE OBJETOS DEL DOCUMENTO



DOM: EL MODELO DE OBJETOS DEL DOCUMENTO

- El modelo de objetos del documento es una **API**, que permite acceder a los contenidos de un documento HTML.
- Proporciona una **representación** estructurada, **orientada a objetos**, de los elementos individuales y el contenido del documento, con métodos para recuperar y fijar las propiedades de dichos objetos.
- Proporciona métodos para agregar y eliminar objetos al documento.
- También proporciona una **interfaz estándar** para trabajar con **eventos**.
- La especificación en: **https://dom.spec.whatwg.org/**

DOM: EL MODELO DE OBJETOS DEL DOCUMENTO

Entre otras cosas, define una forma de acceder a un elemento a través del atributo **id**.

```
<h1>Proyecto de Software</h1>
<h1 id="reglamento">Reglamento de cursada</h1>
```

```
document.getElementById('reglamento')
```

- Más adelante vamos a retomar esto con mayor profundidad.

SEGUIMOS UN POCO MÁS CON EL LENGUAJE

Los campos meta

```
    <meta name="keywords" content="HTML,CSS,JavaScript">
    <meta name="description" content="Guía  de la materia
Proyecto de Software-UNLP.">
```

- ¿Qué pasa con los caracteres especiales?

```
    <meta charset="UTF-8">
```

- ¿Qué les parece que indicamos con este campo?

```
    <meta name="robots" content="noindex" />
```

FORMULARIOS EN LA ARQUITECTURA WEB

Observemos este código

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Formularios web</title>
</head>

<body>
    <h1> Mi color favorito</h1>
    <form method="GET" action="proceso.php">
        Nombre: <input name="nombre" type="text"
placeholder="Ingresá tu nombre acá " />
        Color Favorito: <input name="color_favorito"
list="colores" />
        <datalist id="colores">
            <option value="Azul" />
            <option value="Rojo" />
            <option value="Verde" />
            <option value="Rosa" />
            <option value="Amarillo" />
        </datalist>
        <input type="submit" />
    </form>
</body>
```

```
</html>
```

- Más **info sobre formularios**

METHOD="GET"

El mensaje HTTP sería así:

```
    GET /proceso.php?nombre=claudia&color_favorito=Azul
HTTP/1.1
    Host: www.servidor.com
    User-Agent: Mozilla/5.0
    Accept: image/gif, image/jpeg, text/html
    Accept-language: es, en
    Accept-Charset: iso-8859-1
```

METHOD="POST"

El mensaje HTTP sería así:

```
POST /proceso.php HTTP/1.1
Host: www.servidor.com
User-Agent: Mozilla/5.0
Accept: image/gif, image/jpeg, text/html
Accept-language: es, en
Accept-Charset: iso-8859-1

nombre=claudia&color_favorito=Azul
```

GET VS. POST

GET	POST
Los datos son visibles en la URL	No se ven
Hay límite de caracteres	No tiene
Los resultados pueden guardarse como favoritos	No se guardan
Quedan en el historial del navegador	No se guardan

- Entonces, ¿cuándo uso uno u otro?
- Más info este **articulo**

RETOMEMOS EL FORMULARIO ANTERIOR

- Agregamos el atributo **id** en el primer input: ¿para qué nos serviría esto?

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Formularios web</title>
</head>

<body>
    <h1> Mi color favorito</h1>
    <form method="GET" action="proceso.php">
        Nombre: <input id ="nombre" name="nombre"
type="text" placeholder="Ingresá tu nombre acá " />
        Color Favorito: <input name="color_favorito"
list="colores" />
        <datalist id="colores">
            <option value="Azul" />
            <option value="Rojo" />
            <option value="Verde" />
            <option value="Rosa" />
            <option value="Amarillo" />
        </datalist>
        <input type="submit" />
    </form>
```

```
</body>

</html>
```

ID VS. NAME

- Usando la API DOM podemos acceder a un elemento a través del **id**.

```
document.getElementById('nombre')
```

- **name** nos permite acceder a este datos desde el programa que lo procesa (action)

¿CÓMO VISUALIZAMOS EL DOCUMENTO HTML?

HOJAS DE ESTILO: CSS (CASCADING STYLE SHEETS)

- El documento tiene una estructura y una forma de visualización
  - **Estructura**: usando las etiquetas HTML más apropiadas.
  - **Visualización**: usando hojas de estilo
- Ver sitio **CSS Zen Garden**

CSS: BÁSICO

Reglas: selectores y propiedades

```
p {
        color: red;
        font-size: 14px;
    }
ol, ul {
        color: blue;
        font-size: 18px;
    }
ol li {
        color: blue;
    }
```

- ¿A qué elementos estamos especificando estilos?
- ¿Cuáles son los selectores y cuáles las propiedades?

CSS: BÁSICO

¿Dónde se ubican estas reglas?

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Formularios web</title>
    <style>
        h1 {color:red}
    </style>
</head>

<body>
    <h1 style="color:red"> Mi color favorito</h1>
</body>

</html>
```

Los archivos .css

```
<link rel="stylesheet" href="estilo.css">
```

¿CÓMO SE APLICAN?

- **Estilos definidos en CSS externos.**
- **Estilos definidos en el documento html (tag style).**
- **Estilos en línea: atributo style en las etiquetas.**

Imagen sacada de: https://internetingishard.com

CSS: UN POCO MÁS

Clases:

```
p.aviso {
    color: white;
    background-color: red;
}
.aviso { color: white;
    background-color: red;
}
```

Pseudo clases:

```
a:visited {color: yellow }
```

Pseudo elementos:

```
p:first-letter{
    font-size: 220%;
    float: left }
```

TAREA PARA EL HOGAR...

¿De qué color se muestra el texto "Mi color favorito"?

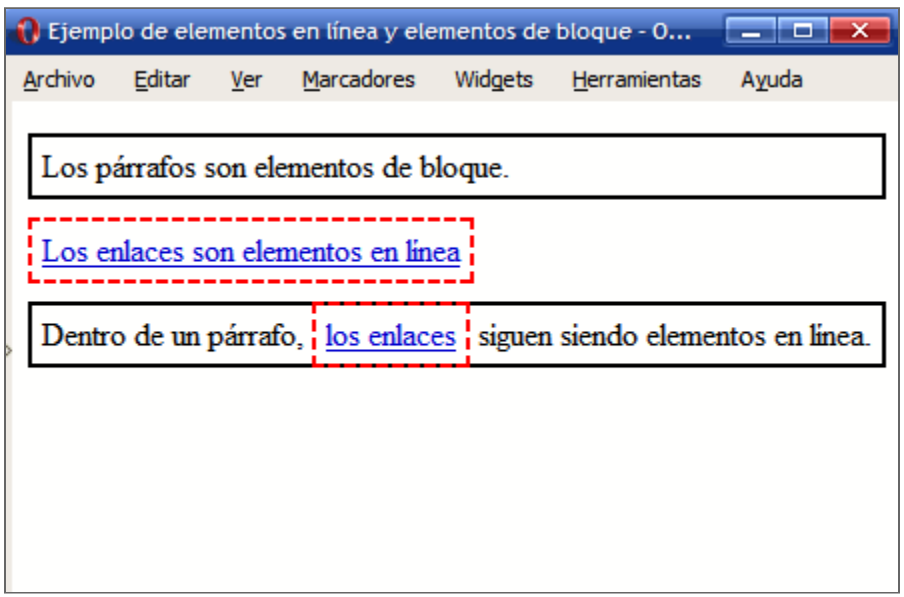
```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Jugando con estilos</title>
    <style>
        h1 {color:orange}
           h1.verde {color: green}
            .azul {color: blue}
        #amarillo {color: yellow}
    </style>
</head>

<body>
    <h1 class="verde" id="amarillo" style="color:red">
Mi color favorito</h1>
</body>

</html>
```

SELECTORES

a	Un elemento
a, b, c	Varios elementos
a b	Elementos descendientes
a > b	Elemento hijo directo
a + b	Elementos adyascentes
a:last-child	Último elemento a
...	...

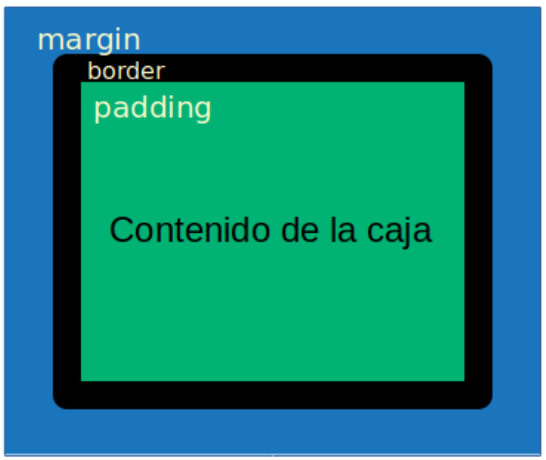
CSS: MODELO DE CAJAS

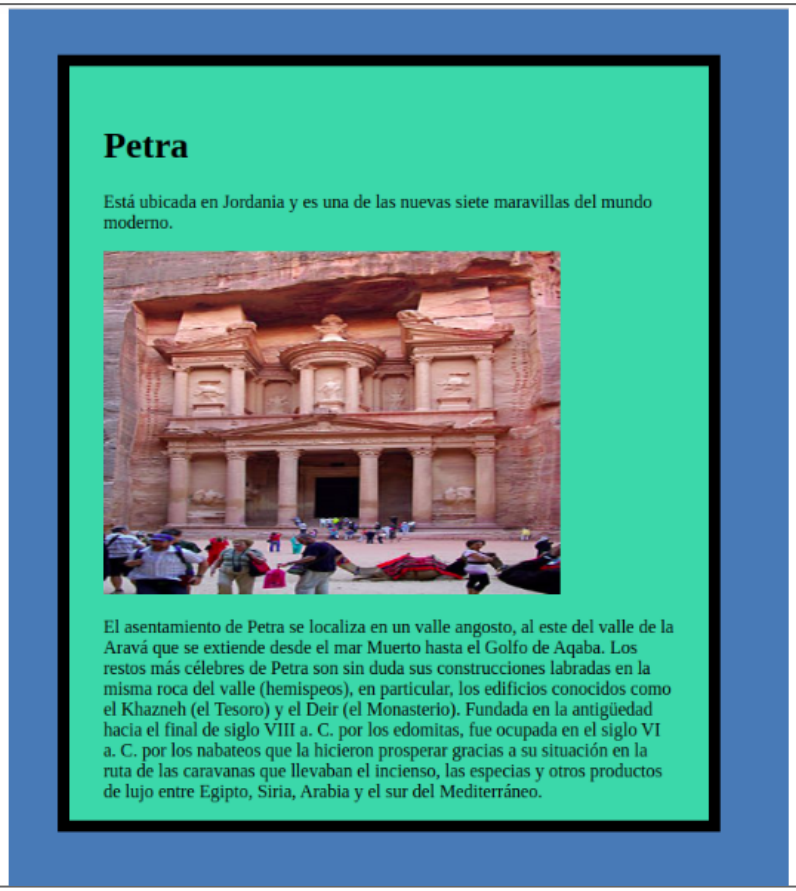
- Todos los elementos de un doc HTML se representan mediante cajas rectangulares.
- Cajas creadas por los elementos de línea y los elementos de bloque



- Más info: **cap. 5 del tutorial Interneting Is Hard**

CSS: MODELO DE CAJAS (CONT.)





Ver **ejemplo**

DISEÑO RESPONSIVO

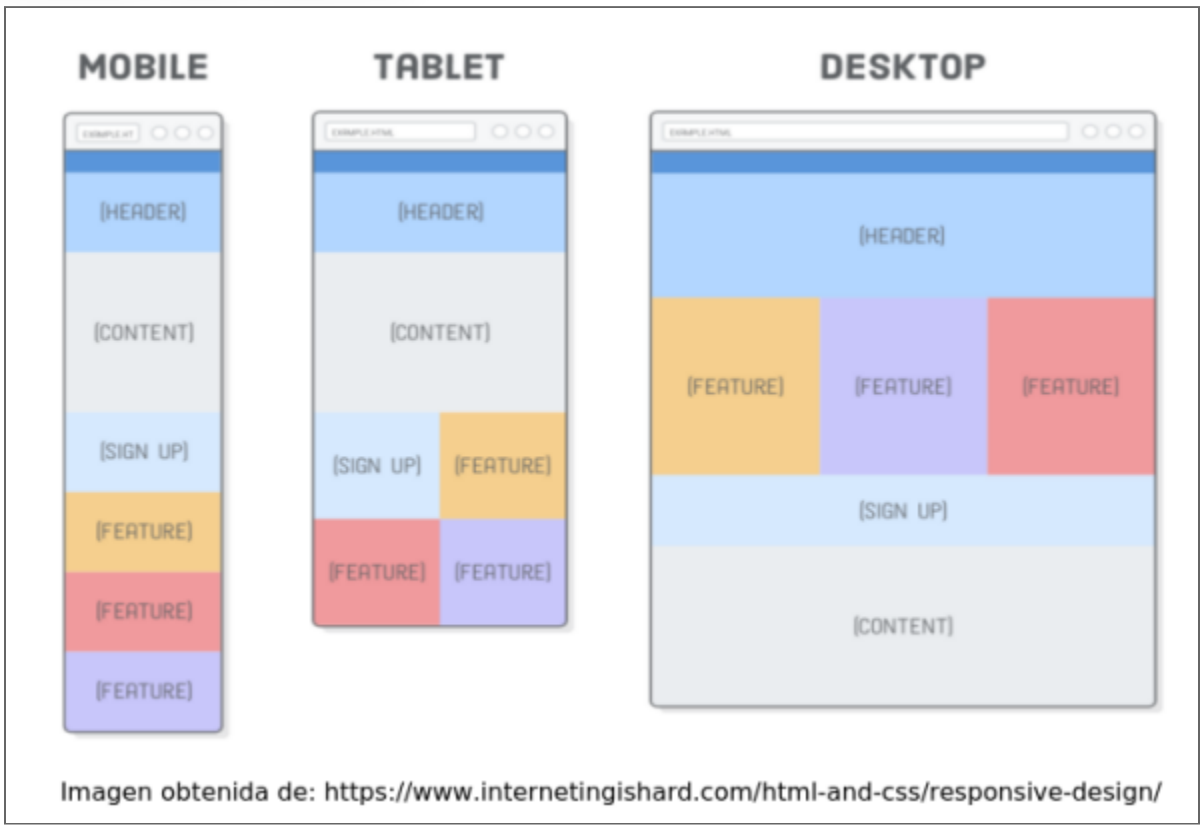


Imagen obtenida de: https://www.internetingishard.com/html-and-css/responsive-design/

- **Algunos datos**
- ¿Cómo lo hacemos?

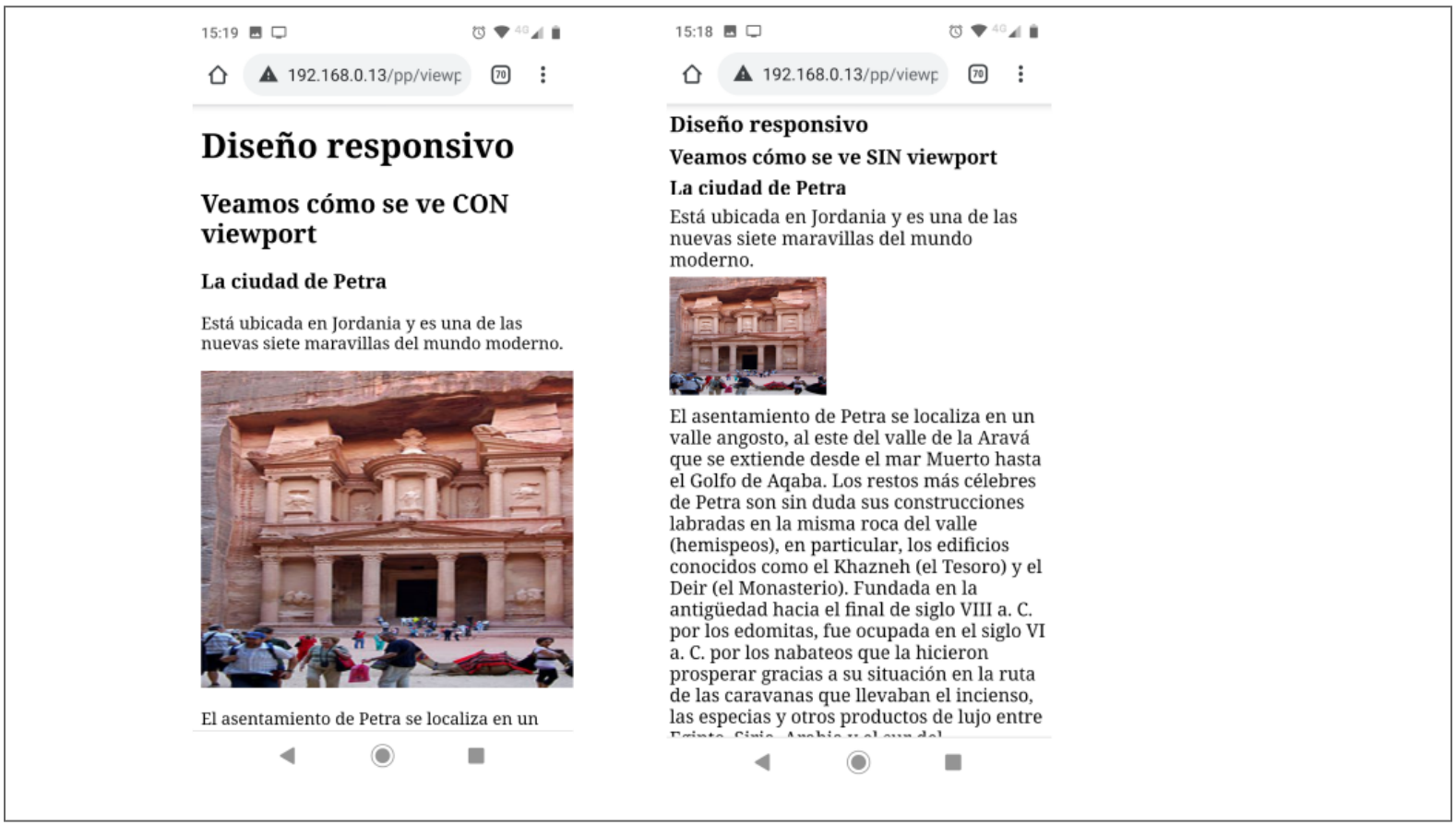
DISEÑO RESPONSIVO

- ¿Qué el el viewport?

```
        <meta name="viewport" content="width=device-width,
initial-scale=1.0">
```

VIEWPORT

Ver ejemplo **viewport.html**



MEDIA QUERIES

Permiten definir estilos específicos dependiendo del medio donde se mostrará la página.

```
    <!DOCTYPE html>
    <html lang="es">

    <head>
        <title>Diseños responsivos</title>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-
width, initial-scale=1.0">
        <style>
            @media (min-width: 600px) {
                body {
                  background-color: lightgreen;
                }
                }
            @media (max-width: 599px) {
                body {
                  background-color: blue;
                }
            }
        </style>
    </head>
```

```
<body>
    <h1> Diseño responsivo</h1>
</body>

</html>
```

MEDIA QUERIES

Es posible utilizar operadores para hacer más específica la regla:

```
    @media (min-width: 700px) and (orientation: landscape)
{ ... }
    @media tv and (min-width: 700px) and (orientation:
landscape) { ... }
```

Se pueden hacer cosas interesantes como: aplicar una hoja de estilo a todos los dispositivos monocromáticos:

```
    @media all and (monochrome) { ... }
```

Media types: **braille**, embossed, handheld, print, projection, screen, **speech**, tty, tv

```
    @media all and (monochrome) { ... }
    @media speech {
          body { voice-family: male }
    }
```

- Más info en **https://developers.google.com**
- También en **cap. 10 del tutorial Interneting Is Hard**

FLEX Y GRID

- Veamos estos ejemplos: **flexbox.html** y **grid.html**
  - Ejemplos sacados del curso **CS50's Web Programming with Python and JavaScript 2020**
  - Se pueden **dercargar todos los ejemplos** de la clase del curso anterior.
- Más **sobre flexbox** y **sobre grid**

BOOTSTRAP, MATERIALIZE, ...

- Bootstrap
- Materialize

¿Qué son? ¿Usaron?

PARA LA PRÓXIMA CLASE

Vamos a hacer una encuesta de repaso con estos temas... por los primeros puntos de teoría.

ALGUNAS REFERENCIAS

- **All the tags**
- Curso **HTML and CSS - Lecture 0 - CS50's Web Programming with Python and JavaScript 2020**
- **Tutorial HTML & CSS Is Hard**