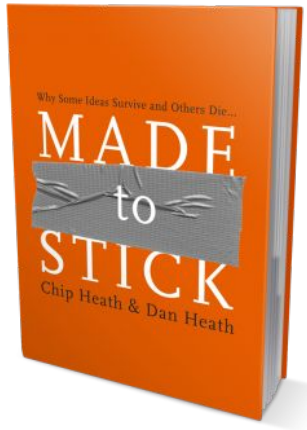
Welcome to Foundations of Web Development!

AIM Code School
Vanessa Kasun

First...What is a Developer?

1. A web developer is a programmer who specializes in, or is specifically engaged in, the development of World Wide Web applications using a client–server model.
2. Applications Developers Use
    a. HTML
    b. CSS
    c. JavaScript
        i. PHP
        ii. .NET
        iii. Python
        iv. Java
        v. Angular
        vi. Vue
        vii. React



Different Developers

Front End Developers

Front-end Developers are responsible for implementing visual elements that users see and interact within a web application. In general, they are supported by back-end web developers, who are responsible for server-side application logic and integration of the work front-end developers do.

Back End Developers

Back-end developers are usually responsible for writing the web services and APIs used by front-end developers and mobile application developers.A back-end web developer is responsible for server-side web application logic as well as the integration of the front-end part.

FullStack Developers

Full Stack Developers are computer programmers who are proficient in both front and back end coding. Their primary responsibilities include designing user interactions on websites, developing servers and databases for website functionality and coding for mobile platforms.

Front-End Web Development Rules!

Tip

Plan your projects!

INVEST YOUR SKILLS

Be curious

Practice, Practice, Practice!

Front-End Languages

- HTML/CSS
- JavaScript
- jQuery/ Bootstrap
- CSS Preprocessors
  - SCSS
  - SaSS
- RESTful Services/APIs
- Vue
- React
- Angular
- PHP
- MySQL

Front-End Tools

- PhotoShop / Adobe Illustrator
- IDE
  - Visual Studio Code
  - IntelliJ
  - Atom
- Boilerplates
- CMS (content management systems)
- Git/Version Control
- Web Hosts
- Google Fonts

This is an example of what VERY BASIC front-end web development CODE looks like… HTML….CSS….JS

HTML

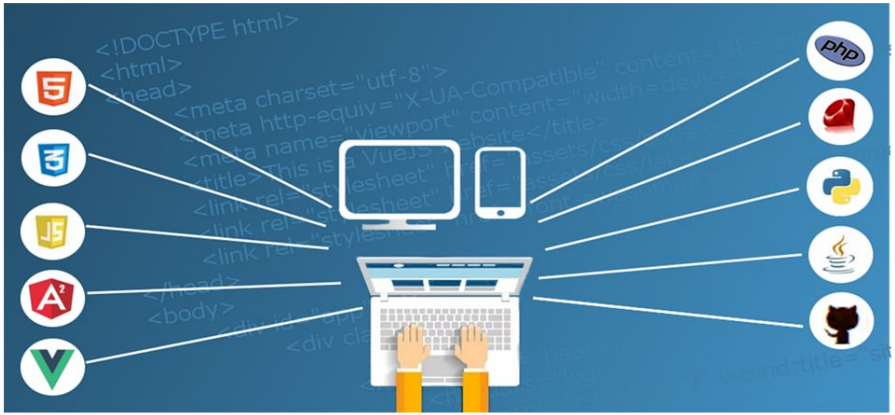
```
<!-- /end gameboard -->
<!-- guide     -->
<section class="guide clearfix ">
 <h2> What is this? </h2>
 <p> Although coded entirely from scratch, this game is a (lackluster) copy of Gabriele Cirulli's 2048, http://2048game.com/.
 </p>
</section>
<section class="guide clearfix">
 <h2> How do I play? </h2>
 <p> Tiles with matching number values can be merged into a single tile, which receives the values' sum.
 </p>
 <p> To move the board, use the directional arrows - or swipe.</p>
 <div class="guide_arrow-wrap">
  <span class="guide_arrow"> &uHar; </span>
  <span class="guide_arrow"> &lHar; </span>
  <span class="guide_arrow"> &rHar; </span>
  <span class="guide_arrow"> &dHar; </span>
 </div>
 <p> To win, get a 2048 tile.
 </p>
</section>
<!-- /end guide  -->
</main>
<!-- /end main  -->

<!-- templates -->
<script type="text/html" id="template_grid_cell">
 <div class="grid_cell"></div>
</script>

<script type="text/html" id="template_tile">
 <div class="tile">
  <span class="tile_number"> </span>
 </div>
</script>
<!-- /end templates -->
```

CSS (SCSS)

```
/*
  Variables:
*/

$grid-max-width: 500px;
$grid-padding: 8px;
$grid-border-radius: 5px;
// white
$color-background: #f2efea;
$color-accent1: #f9d49a;
$color-accent2: #d4a8cf;
$color-list: #00d0a4, #dd7373, #7d53de, #6622cc, #00bfb2, #c06ff2, #340068,
 #3e92cc, #d8315b, #1c0b19, #1c0b19;
/**/

*,
*:before,
*:after {
 box-sizing: border-box;
}

button,
a {
 &:hover {
  cursor: pointer;
 }
}

.clearfix::after {
 content: "";
 display: block;
 clear: both;
}

html {
 min-height: 100%;
 width: 100%;
 font-size: 16px;
```

JS (Babel)

```
// if UP: check next position
if (direction === "up") {
 getNext = this.y > 0 ? this.game.board[this.x][this.y - 1] : false;
 nextPositionArray.push(this.x, this.y - 1);
} else if (direction === "right") {
 // if RIGHT: check next position
 getNext = this.x < 3 ? this.game.board[this.x + 1][this.y] : false;
 nextPositionArray.push(this.x + 1, this.y);
} else if (direction === "down") {
 // if DOWN: check next position
 getNext = this.y < 3 ? this.game.board[this.x][this.y + 1] : false;
 nextPositionArray.push(this.x, this.y + 1);
} else if (direction === "left") {
 // if LEFT: check next position
 getNext = this.x > 0 ? this.game.board[this.x - 1][this.y] : false;
 nextPositionArray.push(this.x - 1, this.y);
}
// Check if next position contains match or is empty
isNextMatch =
 getNext &&
 getNext.tilesArray.length === 1 &&
 getNext.tilesArray[0].valueProp === this.valueProp;
isNextEmpty = getNext && getNext.tilesArray.length === 0;
//

// "check only" mode; only to check if tile can move
if (checkFlag) {
 return isNextEmpty || isNextMatch ? true : false;
} else if (isNextEmpty || isNextMatch) {
 // not "check only" mode; will actually run move logic
 this.setPosition(nextPositionArray[0], nextPositionArray[1]);
 this.removeOldPosition(getX, getY);
 // do NOT continue to move if a tile has matched - and therefore MERGED into adjoining tile
```



Tip

Everything is connected. To become a good programmer you must understand all working parts of web development.

Other Information you need to know as a Front- End web dev

- Internet/Web Browsers
- API's
- DNS (domain name system)
- HTTP/Networks
- Web Hosting
- Command Line (CLI)
- Node.js
- JSON(JavaScript Object Notation)
- App Frameworks
- Back- End Languages and API

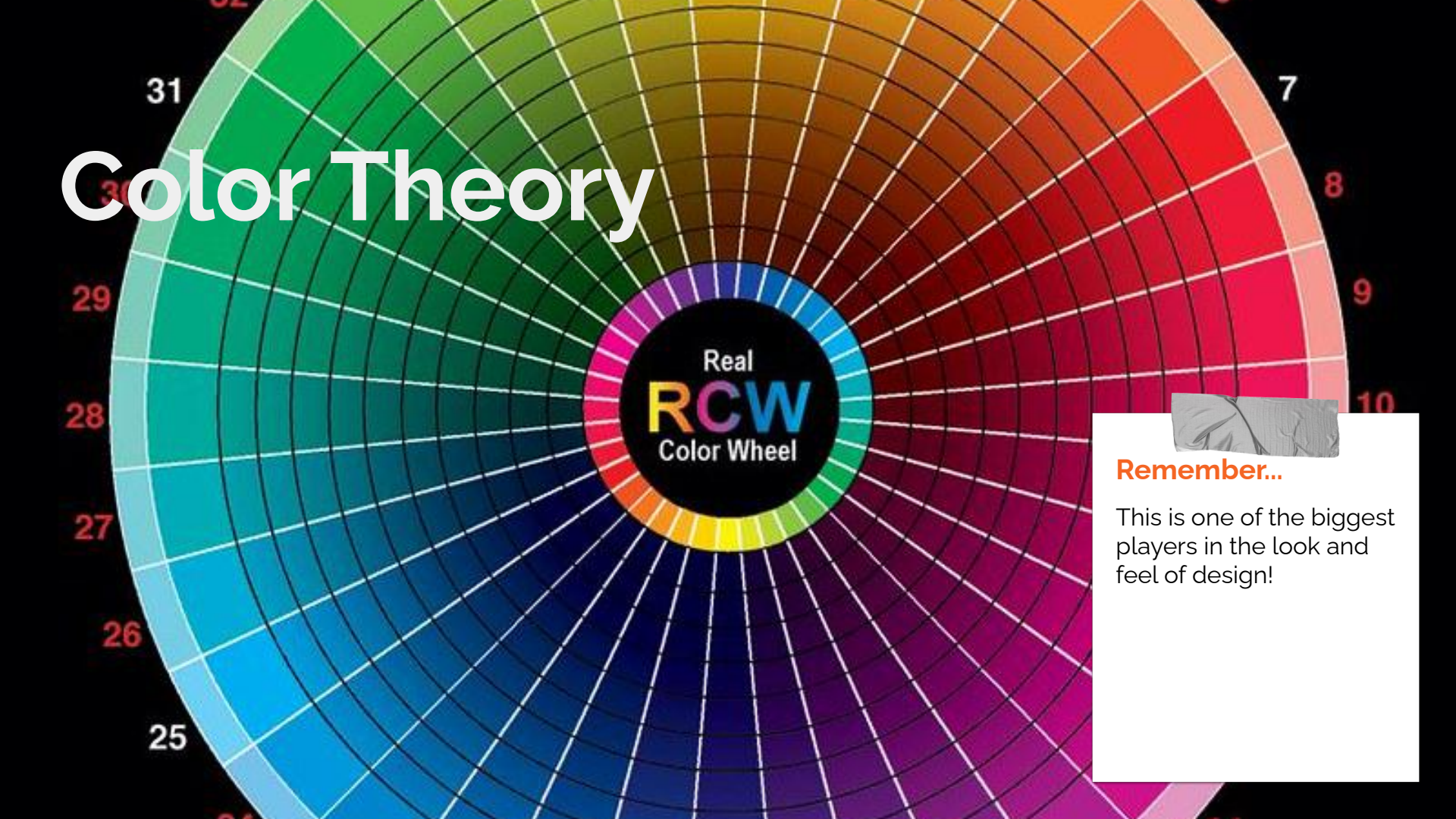
Let's Get Started on Foundation of Web Development 101

Tip

Bare with me! This information may SEEM elementary but it is an EXTREMELY IMPORTANT concept to web development and design.

Thinking Design!

Color Theory



Remember...

This is one of the biggest players in the look and feel of design!

The Color Wheel and Color Theory

Color theory emerged when impressionist started experimenting with capturing light. In short, it boils down to how to human eye translates light with waves into color. Colors that match have similar complementary waves.

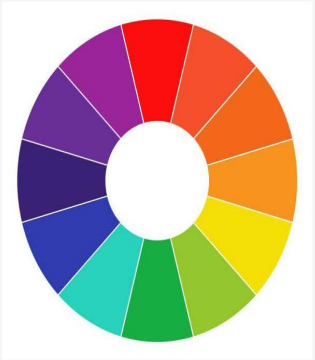
Primary Colors:

Red, Yellow, Blue

They are the basic colors of any color wheel and the color of spectrum as a whole.

To create and color, you'll have to use a combination of the primary colors.

The Color Wheel



Main Colors:

Red	Green
Orange	Blue
Yellow	Violet

In-Between Colors:

Red-Orange	Blue-Green
Yellow-Orange	Blue-Violet
Yellow-Green	Red-Violet

Secondary and Tertiary Colors

Secondary Colors:

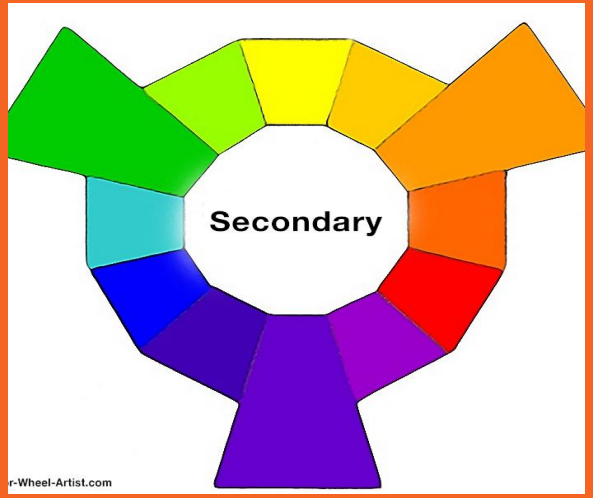
Orange, Green, Purple

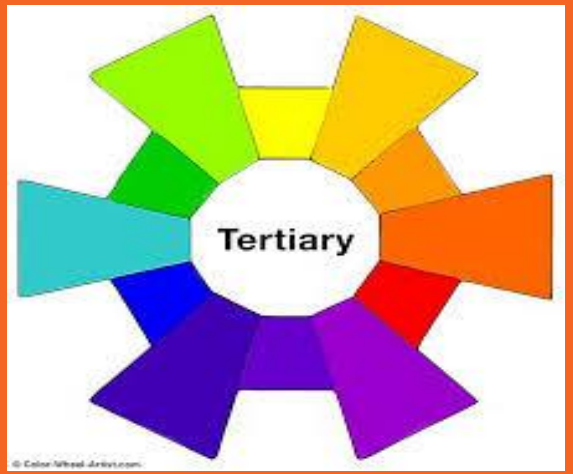
1. Created with 2 primary colors
2. Lies opposite of primary colors
3. Opposite pairs are referred to as being complementary

Tertiary Colors:

1 primary + 1 secondary color

There are more tertiary colors than primary or secondary, owing to more combinations which can be made.





Warm & Cool Colors

Warm Colors:

Reds-Oranges-Yellows

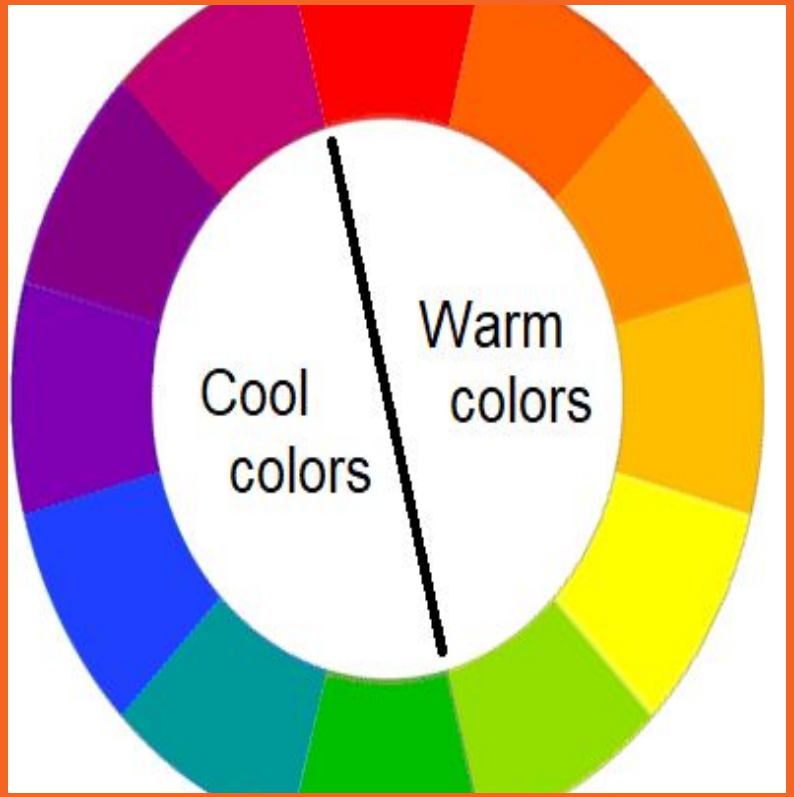
Evokes a feeling of warmth

Cool Colors:

Greens-Blues-Purples

Evokes a calm feeling. They are more subtle than warmer colors.

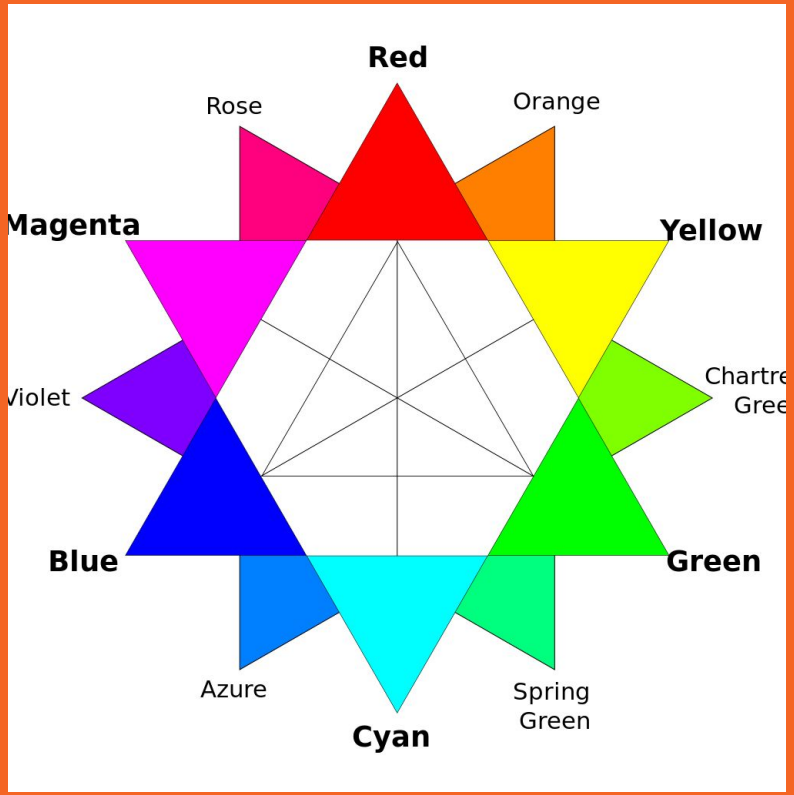
We'll get more into this is color schemes.

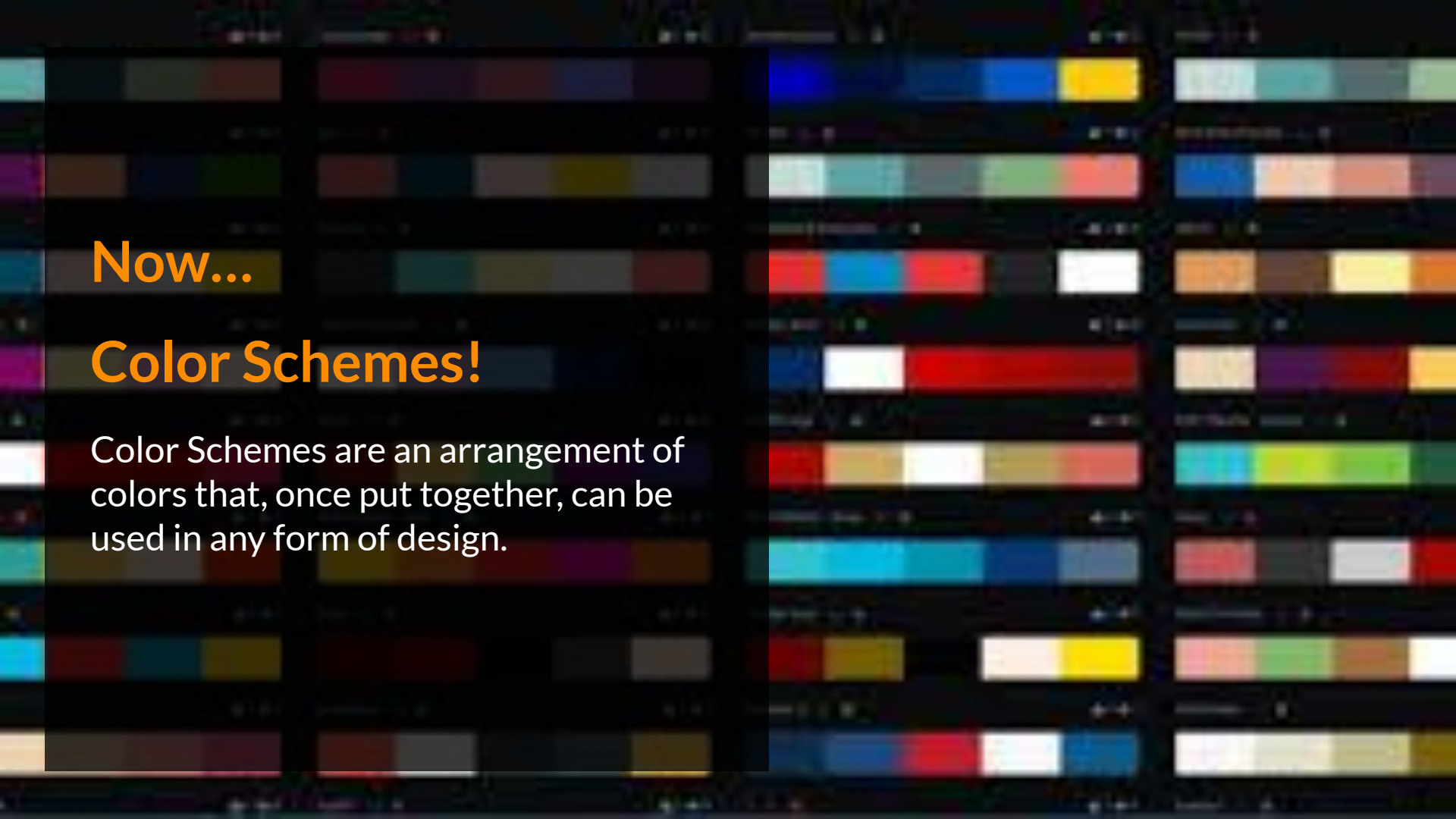


RGB: The Additive Color Model

- Red, green, and blue light are added together in various combination to reproduce a wide spectrum of colors.
- LCD, plasma and LED displays all utilize RGB model.

We'll discuss this in more detail later in the course.

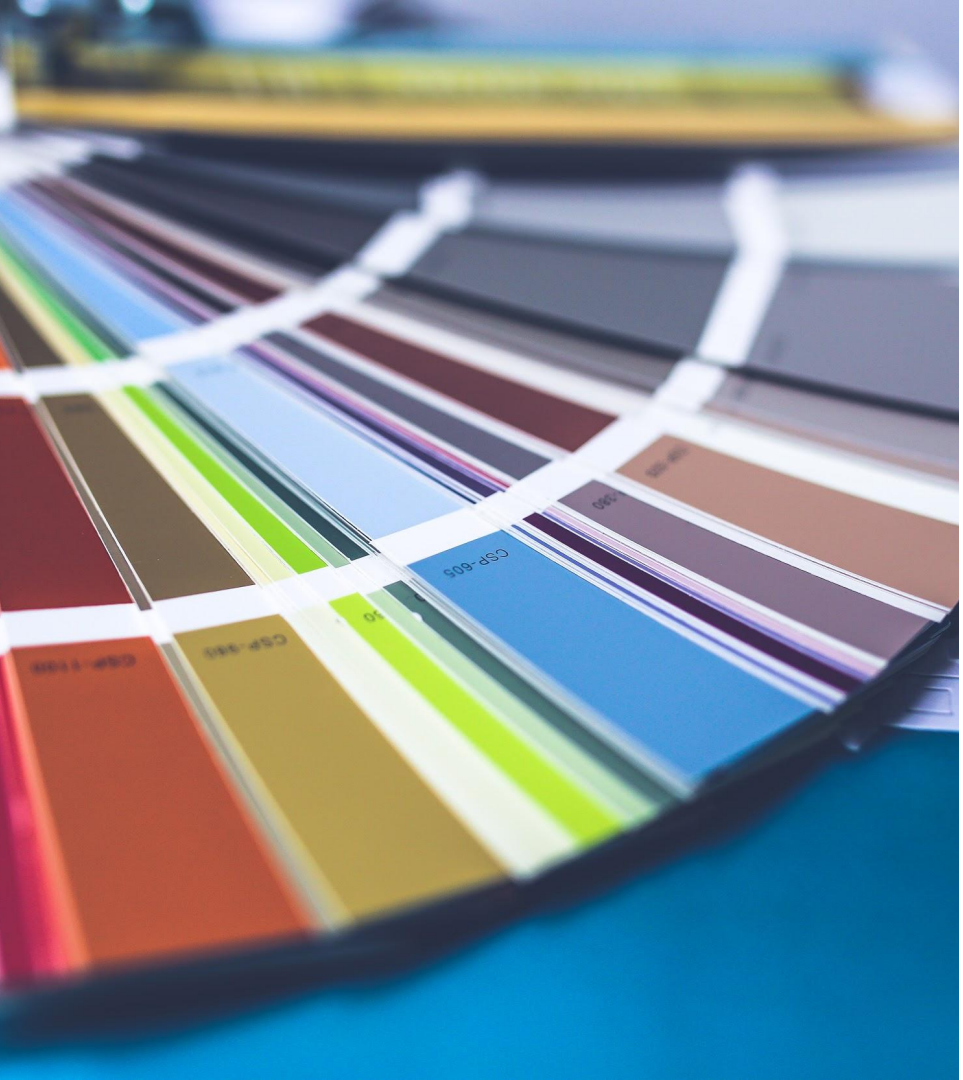




Now...

Color Schemes!

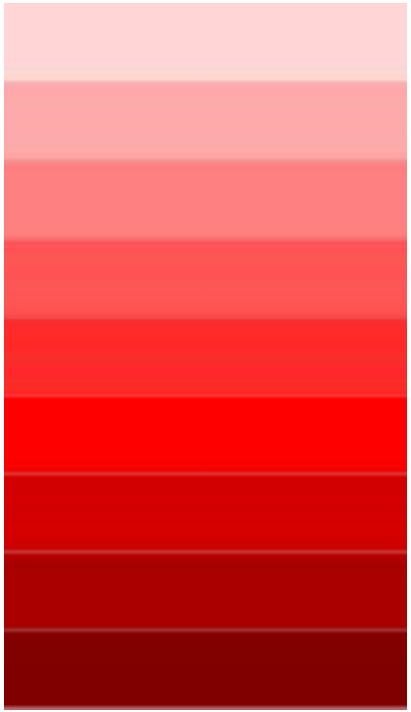
Color Schemes are an arrangement of colors that, once put together, can be used in any form of design.



Main Color Schemes

- Monochromatic
- Analogous
- Complementary
- Split Complementary
- Tradic
- Tetradic

We are going to cover the first three only… please visit this link to study the other color schemes.



Monochromatic

These schemes use one hue (such as red or blue) and then uses various tints or shades of that chosen hue.

These are the easiest color schemes to create as it only involves that one hue - so you know that the few base colors you choose with work in harmony with one another.

Analogous

These schemes are created by choosing hues that are next to each other on the color wheel.

Example: green, yellow-green, yellow ect..

The easiest way to use an analogous color scheme is to choose one of the colors as the main, dominant color and then use the other two more sparsely, as accent colors to the first.

Complementary

These schemes are created by choosing two colors opposite of each other on the color wheel.

Example: orange & purple, green & red

These colors offer high contrast and have a high impact. For this reason this color scheme can be somewhat hard to pull off. They work better when you have elements of a design you want to stand out, rather than using them in fuller doses across the whole design.

What's Next?... UX!

(User Experience)

User experience is a person's emotions and attitudes about using a particular product, system or service. Including practical, experiential, affective, meaningful and valuable aspects of human interaction.

Understand more about UX

Tip

Your design and the entire integration of your website needs to create a meaningful and relevant experience for the user.

UX for the web

- ➔ User Centered Design (UCD)
- ➔ Aesthetically may not always work with function/practically
- ➔ More complicated means less enjoyment
- ➔ The longer the user remains on a website/app the greater the chance of purchase decision or conversion. That is determined within the first 3 seconds.

Prime purpose is to provide a system to fulfill users needs. Mainly this is to interact with your product or service. (UI is not about eye candy!)

User Interface (UI)

UI : Practical to-dos

- Understand user behavior
- Influence the end-user
- Identify user needs (mobile?)
- Understand user wants

Best Practices

- Be consistent with design
- Easy to navigate layout/design
- Communicate clearly.(messages and labels)

Include key principles, questions, and elements to keep in mind.

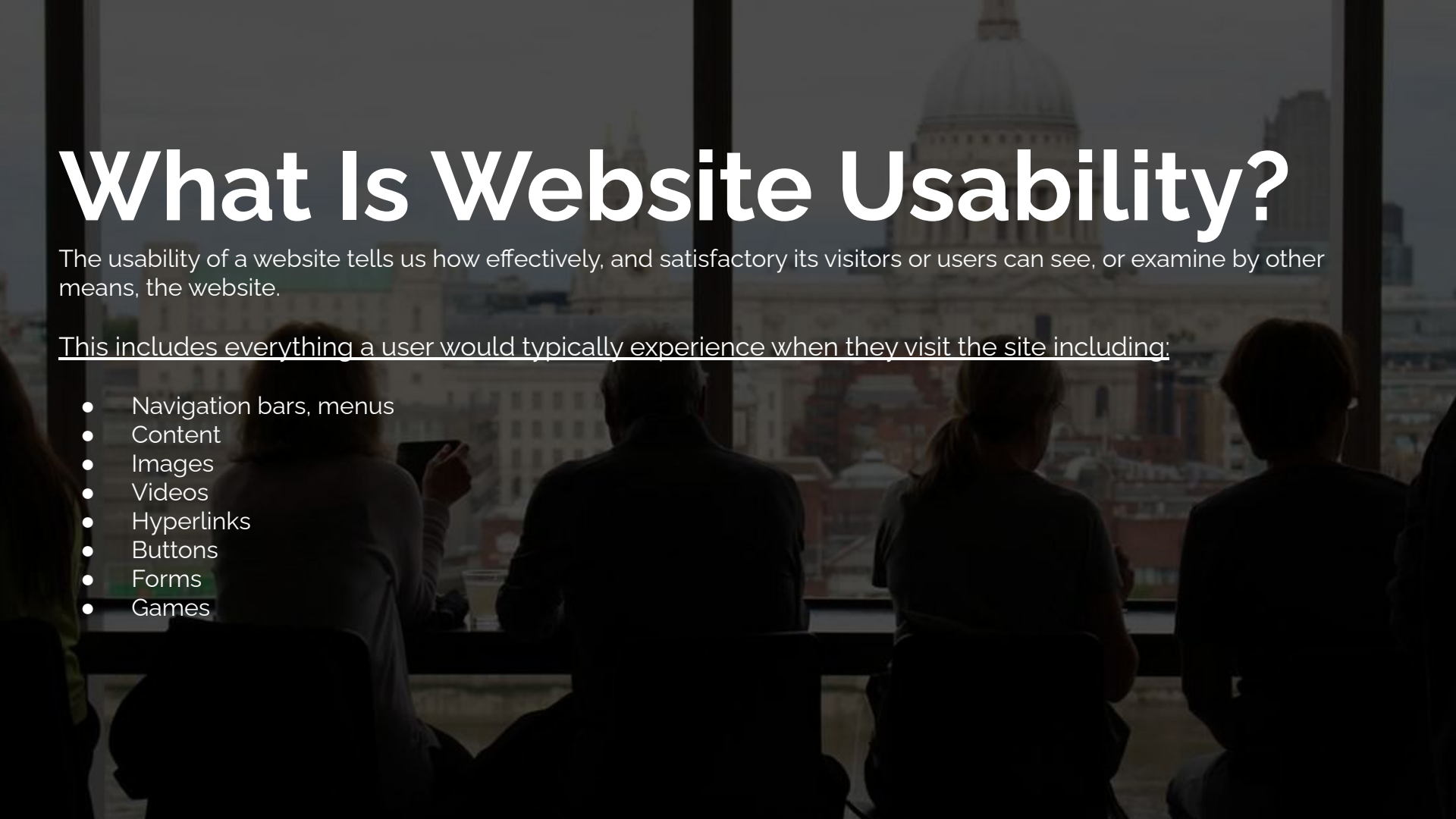
User Centered Design (UCD)

UCD: Standards 6 Key Principles

- Design is based upon an explicit understanding of users, tasks, and environments.
- User test runs throughout design and development
- Design is driven and refined by user centered evaluations.
- The processes is iterative.
- Design addresses the whole user experience.
- Design includes multidisciplinary skills and perspectives.

UCD Questions

- ➔ Who is the targeted audience? (the users of the site)
- ➔ What is the main tasks and goals of the users
- ➔ What functions do the users need from the site?
- ➔ What information might the user need, and in what form do they need it?
- ➔ How does the targeted audience think the site should work?
- ➔ What are the extreme environments?
- ➔ Does the GUI utilize different input modes such as touching, speaking, gestures, or orientation?



What Is Website Usability?

The usability of a website tells us how effectively, and satisfactory its visitors or users can see, or examine by other means, the website.

This includes everything a user would typically experience when they visit the site including:

- Navigation bars, menus
- Content
- Images
- Videos
- Hyperlinks
- Buttons
- Forms
- Games

Usability Checklist

- **Page Load Time**
  Optimizing your pages is IMPORTANT!
- **Clear Purpose**
  The purpose of the page and critical actions are clear within 5 seconds
- **Logo/ Home links**
  Logo is easy to find and links to the home page
- **Easy "About" Info**
  Easy to find about, contact, and home information and links.
- **Consistency**
  Be consistent throughout your design. Layout, navigation, search, logo etc.

Thinking Algorithmically

Why are algorithms important?

Algorithms are at the very core of successful and efficient development. You'll use them as you learn to code, you'll be asked about them in technical interviews, and they'll likely be part of your day-to-day development work.

Learning common algorithms individually is helpful, but what's even better is getting used to algorithmic thinking. If you can train your brain to understand and follow algorithmic logic, writing your own algorithms will become much more intuitive

What does it Mean to Thinking Algorithmically?

Thinking algorithmically is a mindshift from how we, as people, usually think. It is more of a systematic way of thinking through problems and solutions in a way that's similar to how a computer would run.

But that's surprisingly difficult. We all have unconsciously built up shortcuts, assumptions, and rules of thumb that we use to help us solve everyday problems without thinking about them.

For instance, take the simple task of sorting 10 numbers. As it stands, we can take a look at them, tell pretty quickly what the order should be, and arrange the numbers correctly.

However, we're not used to breaking our thought process down into its individual steps and translating that to what computers can do. For instance, computers can't jump to general spots in a dictionary to find a word based on its spelling. A computer has to have very specific instructions on where to start (plus they can't do truly random numbers).

So how do you think algorithmically?

At its essence, algorithmic thinking is thinking about how to solve a problem in a systematic way. It's about:

1. Defining the problem clearly
2. Breaking the problem down into small, simple parts
3. Define the solution for each part of the problem
4. Implementing the solution
5. Making it efficient (eventually)

But wait... what is an algorithm?

The truth about algorithms

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation.
... DON'T PANIC!!!!....

Before started learning about algorithms, I was intimidated by them. I felt that they were complicated, hard to learn, and heavily mathematical. Algorithms felt complex and beyond my ability to understand.
I quickly realized that was a common misconception amongst many new developers. Long story short, don't let big or unfamiliar terms scare you away or discourage you from feeling capable of learning.

Just like anything else you've learned in life, it takes practice and practice and more practice.
The simple truth is that algorithms are just ways to do things. They're processes to solve a type of problem.
Some examples of algorithms are:

- Finding a word in a dictionary
- Sorting a list of numbers
- Finding prime numbers
- Baking a cake
- Doing laundry
- Making a peanut butter and jelly sandwich

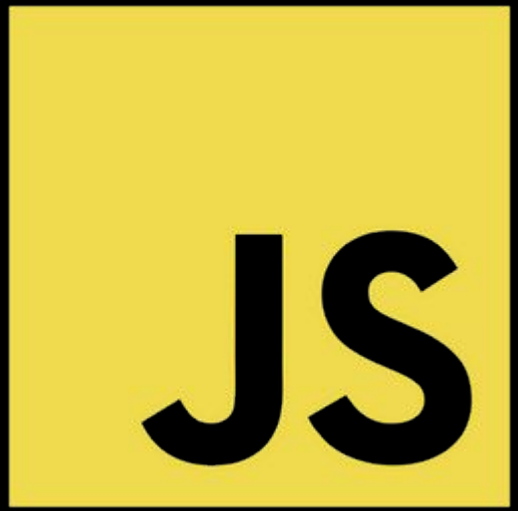
Instructions Exercise

1. Pick a topic from the previous slide.
2. Write down/ type out, the step by step instructions of how to complete the task.

Pretend that you are writing the instructions down for someone who has NEVER EVER heard of the task or any of the "things" needed to complete the task. (like a baby). Give lots of detail and really break down each thought process and jot it down as you go!

Getting Familiar with HTML & CSS

Introduction to JavaScript



What is JavaScript?

JavaScript, often abbreviated as **JS**, is a programming language that conforms to the **ECMAScript** specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

JavaScript can do ANYTHING!!!

An even better way of understanding what **JavaScript** does is to think about certain web features you use every day and likely take for granted—like when your Facebook timeline automatically updates on your screen or Google suggests search terms based on a few letters you've started typing. In both cases, that's JavaScript in action.

Foundational Trio

When most people learn to code, they start with good old HTML and CSS. From there, they move on to JavaScript. Which makes sense! The three elements together form the backbone of web development.(**the foundational trio)**

For those not familiar:

- **HTML** is the structure of your page—the headers, the body text, any images you want to include (bones)
- **CSS** controls how that page looks (it's what you'll use to customize fonts, background colors, etc.) (skin)
- **JavaScript** is the magic third element. Once you've created your structure (HTML) and your aesthetic vibe (CSS), JavaScript makes your site or project dynamic. (characteristics)

JavaScript in Moodle.

That Was A Lot To Cover!!!

Let's move on and talk about basic photo editing!

Photopea