Javascript from scratch
ESGI B3 AL - IW

Nicolas Hersant - Galactic Robots

Javascript : définition

Le Javascript est un langage de programmation de scripts orienté objet.

C'est un langage interprété : il n'y a pas de compilation. Le code source reste tel quel, pour l'exécuter, on doit le fournir à un interpréteur qui se chargera de le lire et de réaliser les actions demandées.

Chaque navigateur possède un interpréteur Javascript, qui diffère selon le navigateur.

Le Javascript a été conçu pour être utilisé conjointement avec le HTML.

Il est aussi utilisé dans de nombreux environnements extérieurs aux navigateurs web tels que Node.js, mongoDB, Apache CouchDB voire Adobe Acrobat.

Javascript : définition

Le standard qui spécifie JavaScript est **ECMAScript**.

Une sixième version majeure du standard a été finalisée et publiée le 17 juin 2015. Cette version s'intitule officiellement ECMAScript 2015 mais est encore fréquemment appelée ECMAScript 6 ou **ES6**.

JavaScript ne doit pas être confondu avec le langage de programmation **Java**.

Java et JavaScript sont deux marques déposées par **Oracle** dans de nombreux pays mais ces deux langages de programmation ont chacun une syntaxe, une sémantique et des usages différents.

https://developer.mozilla.org/fr/docs/Web/JavaScript

Javascript : principes de base

```
const a = 2
const une_variable_plus_longue = "Ma super chaîne"
```

```
let a = 2
a = "Je suis une chaîne maintenant !"
```

```
const a = "Ceci n'est pas problématique"
const b = 'Ceci n\'est pas problématique'
```

```
const name = 'John'
const phrase = `Je m'appele ${name}`
```

```
const a = 2
const b = 3.4123
const c = -509
const d = 1/3
```

```
const vrai = true
const je_suis_faux = false
```

```
const eleves = ['Jean', 'Marc', 'Marion']
const demo = [true, 10, 'Marc']

eleves[0] // Jean
eleves[2] // Marion
```

Javascript : principes de base

```
if (true) {
    let a = 3
}
console.log(a) // ERREUR, a is not defined
```

```
const object = { a: 1, b: 2, c: 3 };

for (const property in object) {
  console.log(property) // affichera alternativement : a, b, c
}

const eleves = ['Jean', 'Marc', 'Marie']
for (const i in eleves) {
    console.log(i) // affichera alternativement : 0, 1, 2
}
```

Javascript : principes de base

```
const eleve = {
   clef: 'valeur',
   nom: 'Jean',
   age: 18,
   notes: [10, 4, 18]
}
```

```
eleve.nom // Jean
eleve.notes // [10, 4, 18]
eleve.notes[1] // 4
// On peut aussi utiliser une notation
eleve['notes'] // [10, 4, 18]
```

```
const eleve = {
    notes: {
        math: 18,
        francais: 14
    }
}
// Pour récupérer la note de math
eleve.notes.math // 18
eleves.nom // undefined
```

```
undefined // quand on essaie d'accéder à une variable ou valeur inexistante
null // représente l'absence intentionnelle de toute valeur
NaN // 'not a number'
```

Javascript : principes de base

Documentation sur les fonctions : (à garder ouvert dans un onglet)

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Functions

```
<script>
    // Ecriture des fonctions
    function styleClassique () {
        console.log('classique')
    }
    let styleFlechee = () => console.log('fonction fléchée')
    styleFlechee()
</script>
```

```
// gestion des arguments
function func1(a, b, c) {
    console.log(arguments[0]);
    // Expected output: 1
    console.log(arguments[1]);
    // Expected output: 2
    console.log(arguments[2]);
    // Expected output: 3
}
func1(1, 2, 3);
```

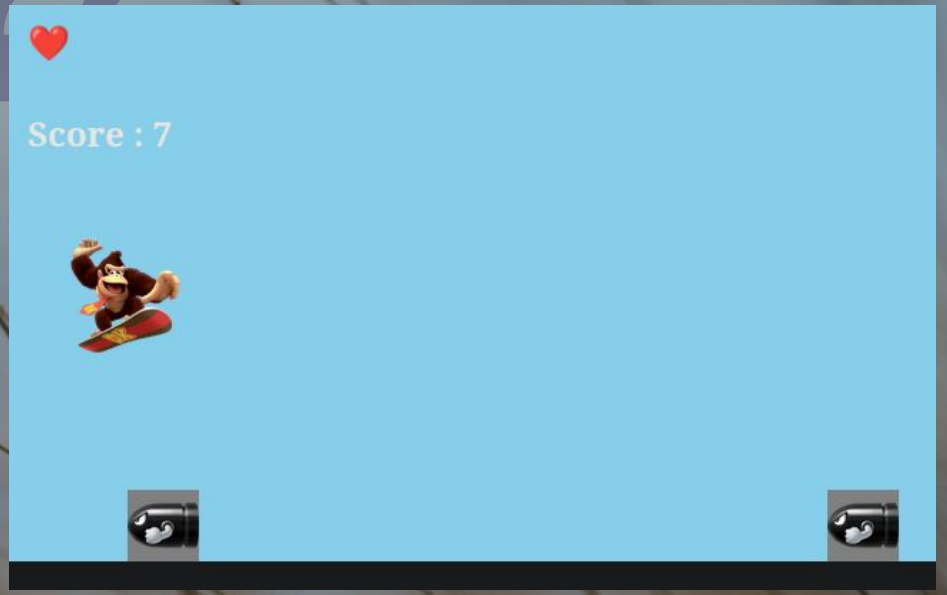
Javascript : exercices

Créer un système de recommandation qui conseille le bon film en fonction de l'âge de l'utilisateur

- Si l'utilisateur a moins de 13 ans (13 ans inclu) on lui affichera "Lilo & Stitch"
- Si l'utilisateur a plus de 13 ans et moins de 60 ans (strictement) on lui affichera "Matrix"
- Si l'utilisateur a plus de 60 ans on lui affichera "Ben-Hur"

```
const birthyear = prompt('Quel est votre année de naissance ?')
// Ecrire votre code ici, afficher le film à l'aide de console.log('votre réponse')
```

Javascript : projet Old School Plateformeur Live Coding



Javascript

🔹 Étape 1 : Initialisation du projet et premier affichage

Objectif : Créer une structure HTML/CSS/JS simple. Introduire la manipulation du DOM.

🔧 Compétences :

- Créer des fichiers HTML, CSS et JS
- Sélectionner des éléments avec `document.querySelector`
- Modifier dynamiquement le contenu et le style

Javascript

🔹 Étape 2 : Mouvement basique gauche-droite

Objectif : Gérer des événements clavier pour déplacer un élément.

🔧 Compétences :

- `addEventListener`
- `keydown` / `keyup`
- Modification dynamique des coordonnées via le style

🔹 Étape 3 : Ajout du saut

Objectif : Créer un saut avec effet gravité (simple).

🔧 Compétences :

- `setInterval` / `setTimeout`
- Fonction anonyme et temporisation
- Animation simplifiée

Javascript

🔹 Étape 4 : Génération d'obstacles

Objectif : Ajouter des objets dynamiques, comme des ennemis ou des blocs.

🔧 Compétences :

- `createElement`, `appendChild`
- Manipulation du DOM dynamique
- Animation avec `setInterval`

Javascript

🧪 TP Étape 5 : Détection de collisions (joueur vs obstacle)

📁 Contexte :

Tu as un joueur qui saute et se déplace, et des obstacles qui apparaissent régulièrement.
Tu dois maintenant détecter s'il entre en collision avec un obstacle.

🔨 Étapes à suivre :

1. Ajouter une fonction `checkCollision`

Elle prend deux éléments DOM et renvoie `true` s'ils se chevauchent.

2. Intégrer la détection dans le déplacement de l'obstacle

Dans ta fonction `createObstacle`, ajoute un test de collision avec le joueur :

✅ Critères de réussite

- Une collision affiche bien "Game Over"
- Le jeu s'arrête quand une collision a lieu
- Le code reste clair et modulaire

Javascript

🧪 TP Étape 5 (suite) : Gestion des points de vie

1. Ajouter une variable `lives`

2. Ajouter un affichage dans le HTML

3. Ajouter une fonction pour mettre à jour l'affichage :

4. Modifier la gestion de collision :

Remplace le `alert("Game Over")` par une gestion avec perte de vie :

✅ Critères de réussite

- Le joueur commence avec 3 cœurs affichés
- À chaque collision, un cœur est perdu
- Le jeu ne s'arrête que lorsque les points de vie tombent à 0
- Les obstacles disparaissent quand ils touchent le joueur

Javascript

🧪 TP Étape 6 : Ajout d'un compteur de score

Maintenant que le joueur peut survivre à plusieurs collisions, il est temps de **le récompenser s'il reste en vie** ! On va ajouter un **score** qui augmente à chaque seconde, et potentiellement aussi quand le joueur évite un obstacle.

1. Ajouter une variable de score

2. Afficher le score dans le HTML

3. Ajouter une fonction `updateHUD` modifiée :

4. Mettre à jour le score avec un timer global

✅ **Critères de réussite :**

- Le score augmente bien chaque seconde pendant que le joueur est en vie
- Le HUD affiche dynamiquement les points de vie **et** le score
- Le jeu s'arrête quand les points de vie atteignent 0, et le score se fige

Tu peux aussi donner des points à chaque obstacle évité. Par exemple, dans la condition où l'obstacle sort de l'écran sans collision :

Javascript

🧪 TP Étape 7 : Gestion de la fin de jeu et progression de niveau

Le joueur peut maintenant survivre et marquer des points. On veut :

- Le féliciter s'il atteint un score élevé
- Ajouter des **phases de niveau** qui accélèrent les obstacles
- Proposer un bouton **"Rejouer"** en fin de partie

1. Créer une zone d'affichage de fin de jeu

2. Gérer la fin de jeu dans le code

3. Ajouter une fonction `restartGame`

4. Bonus : niveaux progressifs

✅ Critères de réussite :

- Un écran "Game Over" apparaît à la fin de la partie
- Le score conditionne le message final
- Le bouton **Rejouer** remet tout à zéro
- La difficulté évolue avec le score

Javascript

🧪 TP Étape 8 : Création d'un menu de démarrage

1. Ajouter le menu dans le HTML
2. Masquer la scène de jeu tant que le jeu n'a pas commencé
3. Ajouter l'événement `start` dans le JS
4. Créer la fonction `startGame()` pour initialiser

✅ Critères de réussite :

- Le jeu **ne commence que lorsque l'utilisateur clique sur "Commencer"**
- Le menu disparaît alors et le joueur prend le contrôle
- Le reste du jeu se déroule normalement

Javascript

🧪 TP Étape 9 : Choix du personnage au lancement du jeu

1. Ajouter les personnages au menu de démarrage
2. Gérer le choix de personnage
3. Appliquer le style au `#player` lors du lancement du jeu

✅ Critères de réussite :

- L'utilisateur voit plusieurs personnages visuellement différents
- Le clic sur un personnage le sélectionne et active le bouton "Commencer"
- En jeu, le personnage a bien la couleur choisie
- Le joueur peut rejouer en relançant un nouveau choix

Javascript

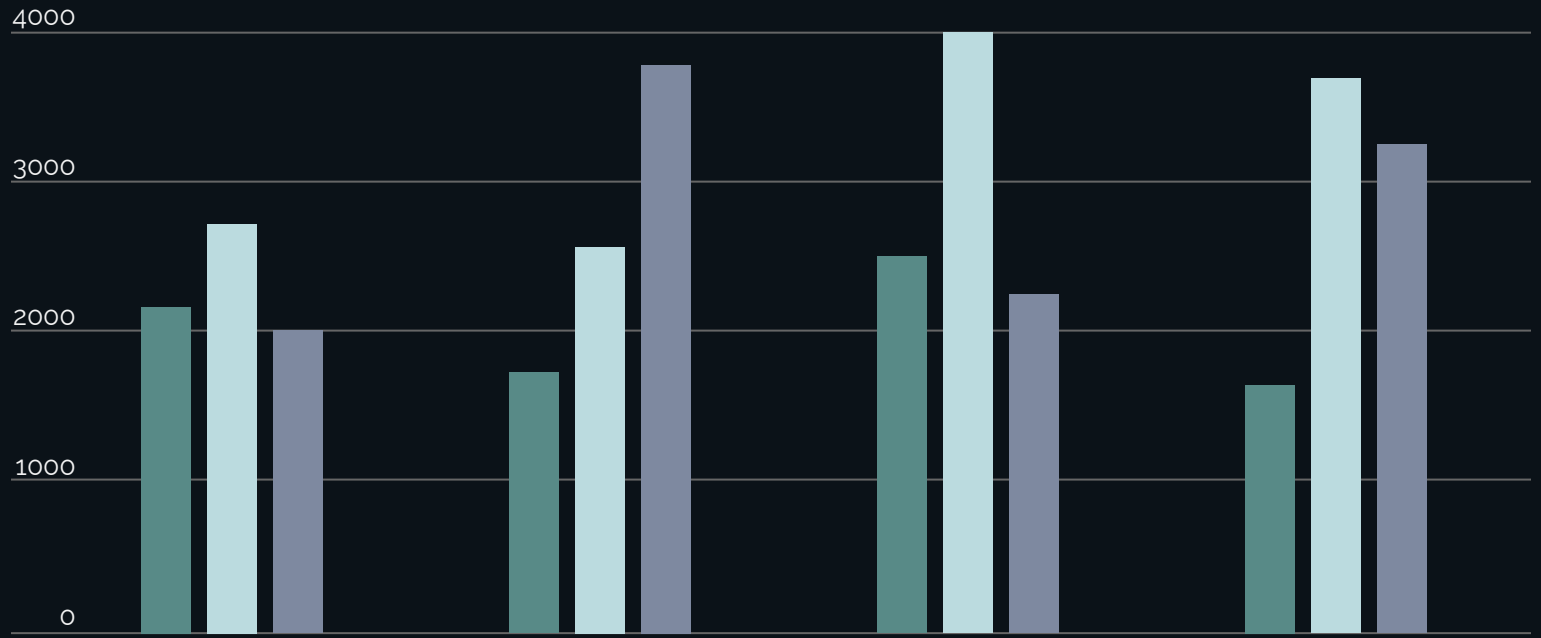
🧪 TP Étape 12 : Statistiques de parties avec objets, tableaux et fonctions natives

1. Créer une base de données locale en mémoire
2. Lier la fin de partie à l'enregistrement des données
3. Afficher les stats à l'écran

stats attendues :
nombre de parties,
score moyen avec reduce()
top1 avec map() et max()
taux victoire avec filter()
perso principale avec
reduce() et Object.entries()

✅ Critères de réussite :

- À chaque fin de partie, une entrée est ajoutée à l'historique
- Les statistiques reflètent fidèlement les données
- Les fonctions JS natives sont bien utilisées pour chaque calcul



You can insert graphs from Excel or Google Sheets

Javascript

🧪 TP Étape 11 : Modularisation du code avec ES6 Modules

🧩 Étapes à suivre :

1. Réorganiser les fichiers

Crée un dossier `js/` avec les fichiers suivants :

bash

```
/js
  ├─ main.js
  ├─ game.js
  ├─ hud.js
  ├─ player.js
  ├─ obstacle.js
```

3. Modifier le script dans le HTML principal

index.html

html

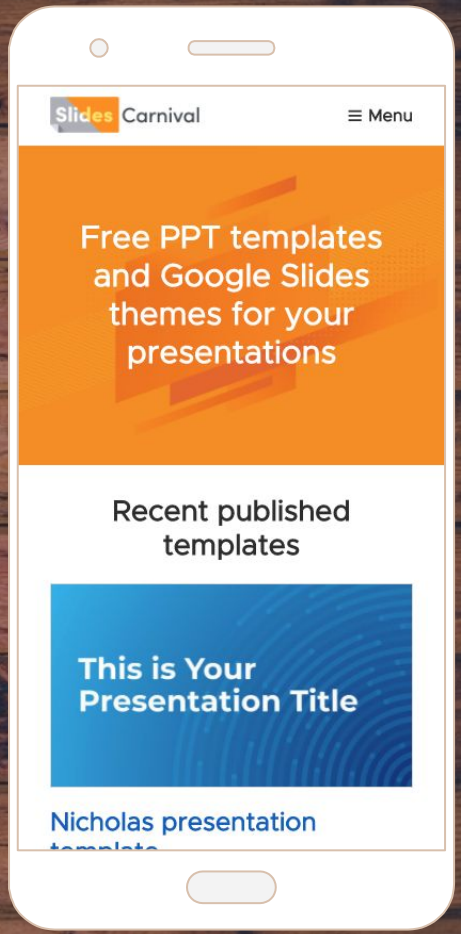
```
<script type="module" src="js/main.js"></script>
```

✅ Critères de réussite :

- Le code est **réparti proprement** dans des modules selon leur responsabilité
- Chaque module **n'expose que ce qui est utile** (`export`)
- Le jeu fonctionne exactement comme avant

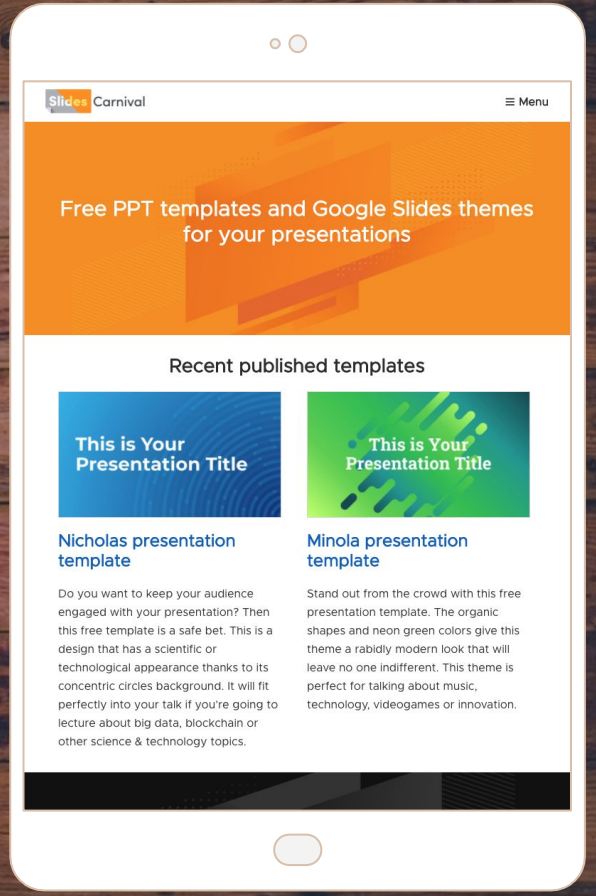
mobile project

Show and explain your web, app or software projects using these gadget templates.



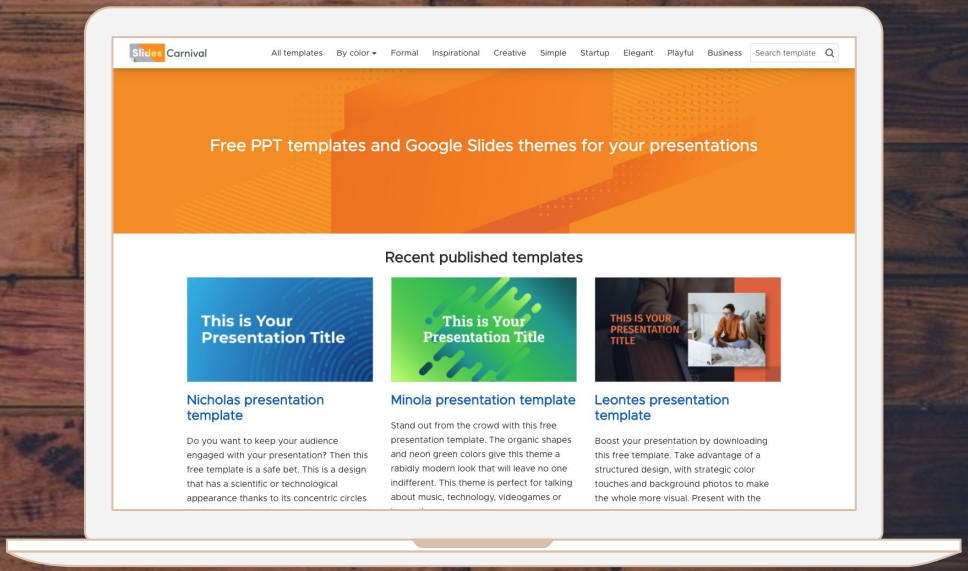
tablet project

Show and explain your web, app or software projects using these gadget templates.



desktop project

Show and explain your web, app or software projects using these gadget templates.




thanks!
ANY QUESTIONS?
You can find me at:
@username
user@mail.me

credits

Special thanks to all the people who made and released these awesome resources for free:

- ✘ Presentation template by SlidesCarnival
- ✘ Photographs by Unsplash

presentation design

This presentations uses the following typographies:

- ✘ Titles: Homemade Apple
- ✘ Body copy: Raleway

Download for free at:

https://www.fontsquirrel.com/fonts/raleway

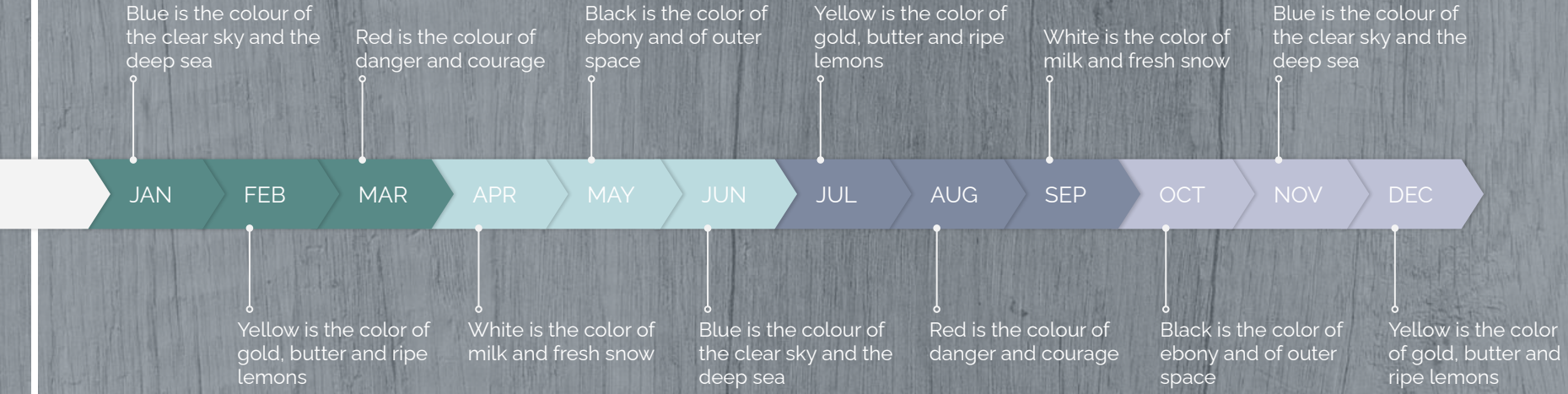
https://www.fontsquirrel.com/fonts/homemade-apple

You don't need to keep this slide in your presentation. It's only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

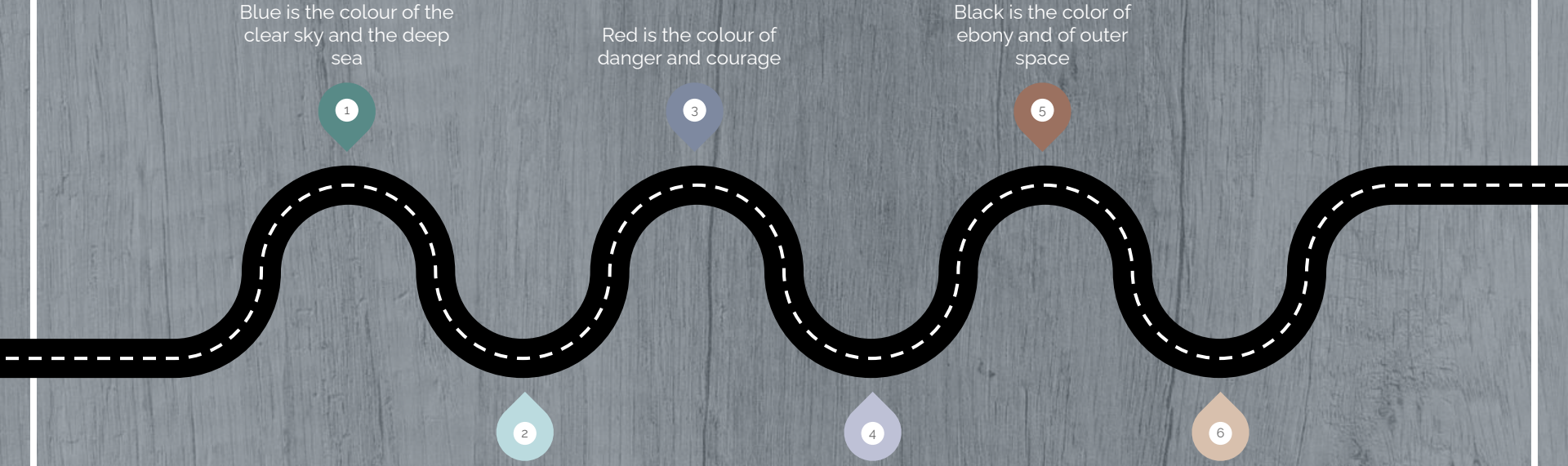
2.
EXTRA RESOURCES

For Business Plans, Marketing Plans, Project Proposals, Lessons, etc

Timeline

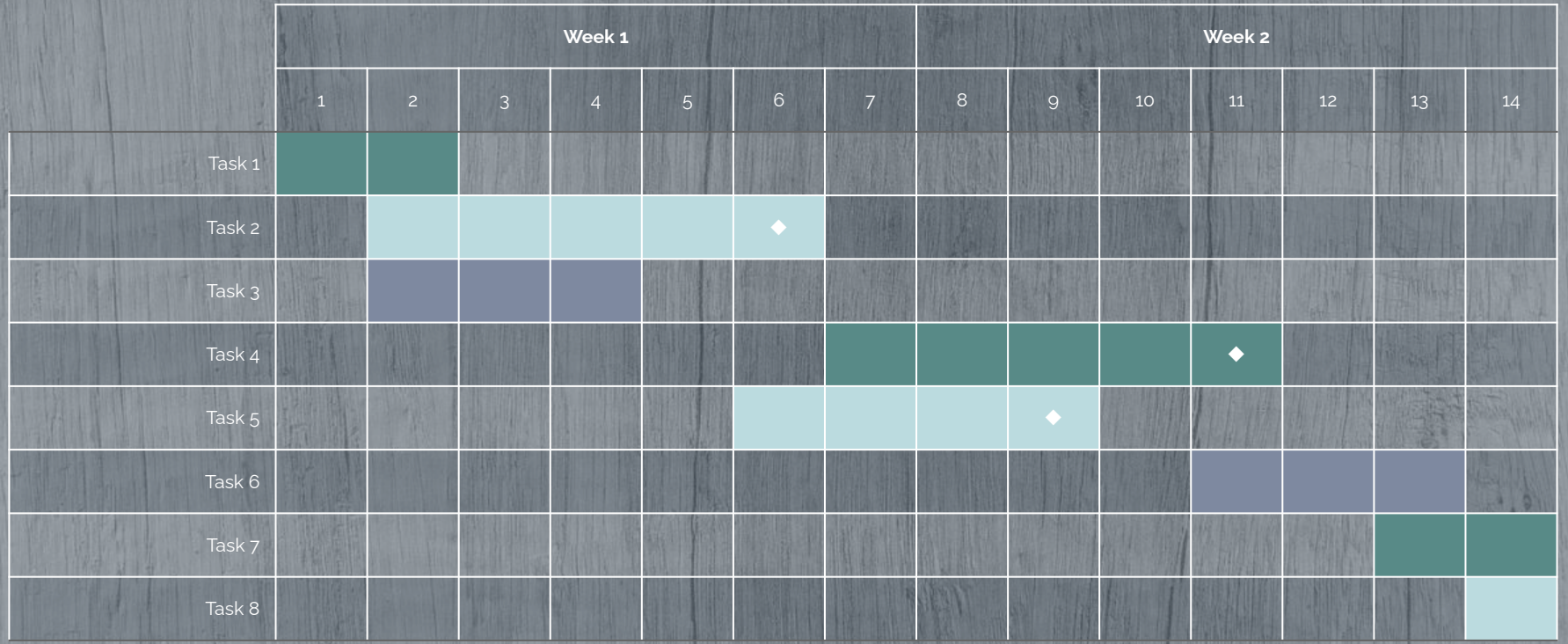

Blue is the colour of the clear sky and the deep sea
Red is the colour of danger and courage
Black is the color of ebony and of outer space
Yellow is the color of gold, butter and ripe lemons
White is the color of milk and fresh snow
Blue is the colour of the clear sky and the deep sea
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC
Yellow is the color of gold, butter and ripe lemons
White is the color of milk and fresh snow
Blue is the colour of the clear sky and the deep sea
Red is the colour of danger and courage
Black is the color of ebony and of outer space
Yellow is the color of gold, butter and ripe lemons

Roadmap



1. Blue is the colour of the clear sky and the deep sea
2. Yellow is the color of gold, butter and ripe lemons
3. Red is the colour of danger and courage
4. White is the color of milk and fresh snow
5. Black is the color of ebony and of outer space
6. Blue is the colour of the clear sky and the deep sea

Gantt chart


Week 1
Week 2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
Task 7
Task 8

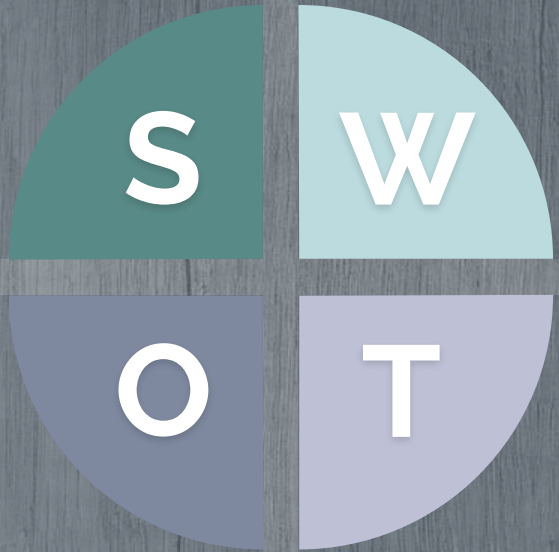
SWOT Analysis

STRENGTHS

Blue is the colour of the clear sky and the deep sea

WEAKNESSES

Yellow is the color of gold, butter and ripe lemons



Black is the color of ebony and of outer space

OPPORTUNITIES

White is the color of milk and fresh snow

THREATS

Business Model Canvas

Key Partners
Insert your content

Key Activities
Insert your content

Key Resources
Insert your content

Value Propositions
Insert your content

Customer Relationships
Insert your content

Channels
Insert your content

Customer Segments
Insert your content

Cost Structure
Insert your content

Revenue Streams
Insert your content

Funnel

Insert your content

Insert your content

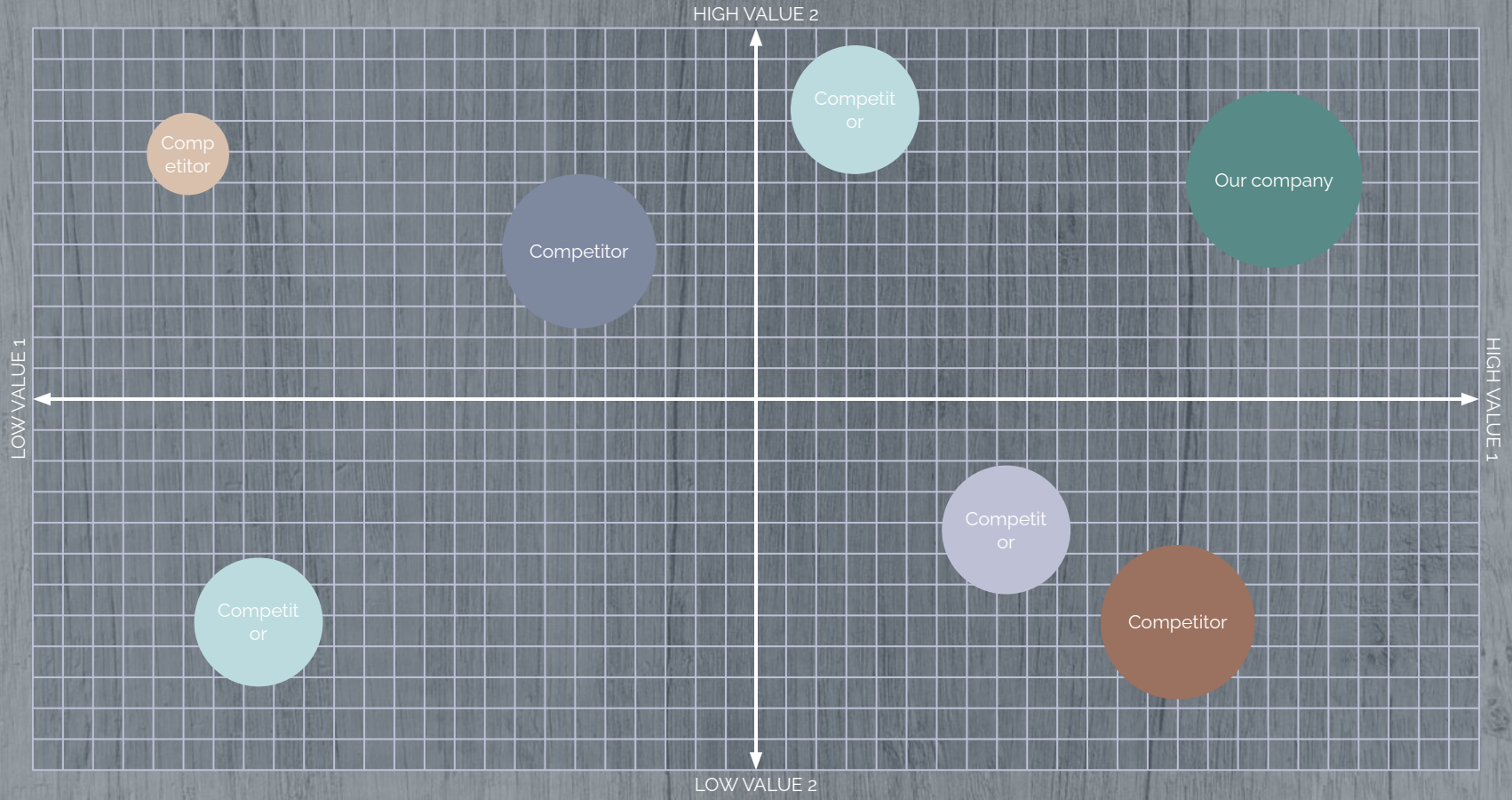
Insert your content

Insert your content

Insert your content

Insert your content

Competitor Matrix



Weekly Planner

	SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
09:00 - 09:45	Task	Task	Task	Task	Task	Task	Task
10:00 - 10:45	Task	Task	Task	Task	Task	Task	Task
11:00 - 11:45	Task	Task	Task	Task	Task	Task	Task
12:00 - 13:15	✔ Free time	✔ Free time	✔ Free time	✔ Free time	✔ Free time	✔ Free time	✔ Free time
13:30 - 14:15	Task	Task	Task	Task	Task	Task	Task
14:30 - 15:15	Task	Task	Task	Task	Task	Task	Task
15:30 - 16:15	Task	Task	Task	Task	Task	Task	Task