**JavaScript (JS)**

1. **Основные типы данных:**
   * **Примитивы: string, number, boolean, undefined, null, symbol, bigint.**
   * **Сложные типы: object, array, function.**
2. **Операторы:**
   * **Арифметические: +, -, \*, /, %, ++, --.**
   * **Сравнения: ==, ===, !=, !==, >, <, >=, <=.**
   * **Логические: &&, ||, !.**
   * **Присваивания: =, +=, -=, \*=, /=, &&=, ||=, ??=.**
   * **Тернарный оператор: condition ? expr1 : expr2.**
3. **Функции:**
   * **Объявление: function funcName() { }**
   * **Стрелочные функции: const add = (a, b) => a + b;**
   * **Параметры по умолчанию: function greet(name = 'Guest') { ... }**
   * **Rest параметры: function sum(...args) { ... }**
   * **Arrow functions: Не имеют собственного this.**
4. **Массивы и объекты:**
   * **Методы массивов: .map(), .filter(), .reduce(), .forEach(), .push(), .pop(), .shift(), .unshift().**
   * **Объекты: { key: value }, доступ через точку или скобки: obj.key или obj['key'].**
5. **Асинхронность:**
   * **Callback функции: setTimeout(), setInterval().**
   * **Promise: .then(), .catch(), .finally().**
   * **Async/Await: async function foo() { const res = await fetch(url); }.**
6. **Модули:**
   * **Импорт/Экспорт: import { func } from './module';, export default MyClass;.**
7. **DOM:**
   * **Поиск элементов: document.getElementById(), document.querySelector().**
   * **Манипуляции с элементами: element.innerHTML, element.textContent, element.style.**
8. **Обработка событий:**
   * **element.addEventListener('event', callback).**
   * **Основные события: click, mouseenter, keydown, submit, change.**
9. **Замыкания (Closures):**
   * **Функции, которые могут захватывать переменные из внешнего контекста.**
10. **this:**
    * **В стрелочных функциях this не привязан к объекту, а ссылается на this из внешней области.**

**TypeScript (TS)**

1. **Типы данных:**
   * **Примитивы: string, number, boolean, null, undefined, symbol, bigint.**
   * **Типы объектов: object, array, tuple, enum, any, unknown.**
   * **Типы функций: Типы параметров и возвращаемых значений.**
   * **Параметры по умолчанию: function greet(name: string = 'Guest').**
2. **Типизация объектов и интерфейсов:**
   * **Интерфейсы: interface Person { name: string; age: number; }**
   * **Типы объектов: type Point = { x: number, y: number }.**
   * **Чтение свойств с использованием ключей: let key: keyof Person.**
3. **Типы функций:**
   * **const add = (a: number, b: number): number => a + b;**
   * **Функции с переменным числом параметров: function sum(...args: number[]): number { return args.reduce((a, b) => a + b, 0); }**
4. **Generics:**
   * **Позволяют создавать функции и классы с обобщенными типами.**
   * **Пример: function identity<T>(arg: T): T { return arg; }.**
5. **Типы данных any, unknown, never:**
   * **any: Отключает проверку типов.**
   * **unknown: Неизвестный тип, требует проверки перед использованием.**
   * **never: Возвращает тип, который никогда не должен завершаться (например, функции с бесконечным циклом или выбрасывающие исключение).**
6. **Типы классов:**
   * **Классы и наследование: class Animal { constructor(public name: string) {} }**
   * **Модификаторы доступа: public, private, protected.**
7. **Модули и пространства имен:**
   * **Экспорт/Импорт: export, import.**
   * **Пространства имен: namespace MyNamespace { export class MyClass {} }.**
8. **Типы данных и кастинг:**
   * **Приведение типов: const num = <number>someVar; или const num = someVar as number;.**
9. **Декораторы:**
   * **Классовые декораторы: @component, @decorator.**

**Angular**

1. **Основные концепции:**
   * **Модули (Modules): Основной строительный блок Angular-приложений. Модуль содержит компоненты, сервисы и другие элементы.**
   * **Компоненты (Components): Классы с декоратором @Component, содержащие логику и разметку для представления.**
   * **Сервисы (Services): Классы для логики приложения, такие как взаимодействие с API или работа с данными.**
   * **Директивы (Directives): Изменяют поведение DOM-элементов. (например, ngIf, ngFor).**
   * **Пайпы (Pipes): Преобразуют данные в представлении (например, date, currency).**
2. **Декораторы:**
   * **@Component: Для объявления компонента.**
   * **@NgModule: Для объявления модуля.**
   * **@Injectable: Для пометки класса как сервиса, который можно внедрить через Dependency Injection (DI).**
3. **Routing:**
   * **Router: Управление навигацией между страницами.**
   * **Маршруты: Настройка с помощью RouterModule.forRoot() и RouterModule.forChild().**
   * **Параметры маршрутов: ActivatedRoute для получения параметров маршрута.**
4. **Data Binding:**
   * **Interpolation: {{ value }}**
   * **Property Binding: <img [src]="imageUrl">**
   * **Event Binding: <button (click)="onClick()">**
   * **Two-Way Binding: [(ngModel)]="value"**
5. **Dependency Injection:**
   * **Внедрение зависимостей через конструктор. Пример: constructor(private userService: UserService) {}.**
   * **Injector используется для внедрения сервисов.**
6. **Forms:**
   * **Template-driven Forms: Простой подход, используется с директивами ngModel.**
   * **Reactive Forms: Использует FormControl, FormGroup для создания динамических форм.**
7. **HTTP:**
   * **HttpClient для работы с HTTP-запросами: this.http.get('/api/data').**
   * **Обработка ошибок через catchError.**
8. **RxJS:**
   * **Операторы, такие как map, filter, switchMap, catchError, mergeMap для работы с асинхронными потоками данных.**
   * **Подписка: observable.subscribe().**
   * **Операторы для работы с Observable.**
9. **Lifecycle Hooks:**
   * **ngOnInit: Вызывается при инициализации компонента.**
   * **ngOnChanges: Вызывается при изменении входных данных.**
   * **ngOnDestroy: Для очистки ресурсов, когда компонент уничтожается.**
10. **Angular CLI:**
    * **Создание компонентов: ng generate component component-name**
    * **Запуск приложения: ng serve**
    * **Создание сервисов: ng generate service service-name**

Here's a basic JavaScript browser script cheatsheet to help you get started with web development. It covers essential concepts and common tasks.

**1. Selecting HTML Elements**

* **Get element by ID:**
* const element = document.getElementById('elementId');
* **Get elements by class name:**
* const elements = document.getElementsByClassName('className');
* **Get elements by tag name:**
* const elements = document.getElementsByTagName('div');
* **Query selector (CSS selector):**
* const element = document.querySelector('.my-class');
* **Query selector for all matching elements:**
* const elements = document.querySelectorAll('.my-class');

**2. Manipulating HTML Content**

* **Change innerHTML:**
* element.innerHTML = 'New content';
* **Change text content:**
* element.textContent = 'New text';
* **Change attribute:**
* element.setAttribute('src', 'new-image.jpg');
* **Get attribute:**
* const src = element.getAttribute('src');
* **Add/remove class:**
* element.classList.add('new-class');
* element.classList.remove('old-class');

**3. Event Handling**

* **Add event listener:**
* element.addEventListener('click', function() {
* alert('Element clicked!');
* });
* **Remove event listener:**
* element.removeEventListener('click', function);
* **Event object (for more info):**
* element.addEventListener('click', function(event) {
* console.log(event.target);
* });

**4. DOM Manipulation (Creating Elements)**

* **Create new element:**
* const newElement = document.createElement('div');
* **Append new element to the DOM:**
* document.body.appendChild(newElement);
* **Insert element before another:**
* const parent = document.getElementById('parent');
* const referenceNode = document.getElementById('reference');
* parent.insertBefore(newElement, referenceNode);

**5. Forms and Input**

* **Get form values:**
* const username = document.getElementById('username').value;
* **Set form values:**
* document.getElementById('username').value = 'new value';
* **Submit a form:**
* document.getElementById('form').submit();

**6. Timer Functions**

* **Set a timeout (delayed execution):**
* setTimeout(function() {
* alert('Hello after 2 seconds');
* }, 2000);
* **Set an interval (repeated execution):**
* const intervalId = setInterval(function() {
* console.log('This runs every 1 second');
* }, 1000);
* **Clear interval:**
* clearInterval(intervalId);
* **Clear timeout:**
* clearTimeout(timeoutId);

**7. Local Storage**

* **Set item in local storage:**
* localStorage.setItem('key', 'value');
* **Get item from local storage:**
* const value = localStorage.getItem('key');
* **Remove item from local storage:**
* localStorage.removeItem('key');
* **Clear all local storage:**
* localStorage.clear();

**8. Window and Document Properties**

* **Get window size:**
* const width = window.innerWidth;
* const height = window.innerHeight;
* **Scroll to top of the page:**
* window.scrollTo(0, 0);
* **Reload page:**
* window.location.reload();
* **Change URL:**
* window.location.href = 'https://www.example.com';

**9. Alerts and Prompts**

* **Alert box:**
* alert('This is an alert!');
* **Confirmation box (yes/no):**
* const result = confirm('Do you want to proceed?');
* **Prompt box (user input):**
* const name = prompt('Enter your name');

**10. Handling Errors**

* **Try/Catch:**
* try {
* throw new Error('Something went wrong');
* } catch (error) {
* console.error('Error:', error.message);
* }

**11. JavaScript ES6+ Features**

* **Arrow functions:**
* const add = (a, b) => a + b;
* **Template literals (multi-line strings):**
* const name = 'John';
* const greeting = `Hello, ${name}!`;
* **Destructuring (arrays/objects):**
* const [a, b] = [1, 2];
* const {name, age} = {name: 'John', age: 30};
* **Promises (async handling):**
* fetch('https://api.example.com')
* .then(response => response.json())
* .then(data => console.log(data))
* .catch(error => console.error(error));
* **Async/await:**
* async function fetchData() {
* try {
* const response = await fetch('https://api.example.com');
* const data = await response.json();
* console.log(data);
* } catch (error) {
* console.error(error);
* }
* }